KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

PROGRAMLAMA ÖĞRENME ORTAMLARI İÇİN GELİŞTİRİLEN PROGRAMLAMA TEKNİKLERİNİN ANALİZİ

YÜKSEK LİSANS TEZİ

SEFA ARAS

HAZİRAN 2018
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

PROGRAMLAMA ÖĞRENME ORTAMLARI İÇİN GELİŞTİRİLEN PROGRAMLAMA TEKNİKLERİNİN ANALİZİ

Sefa ARAS

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"BİLGİSAYAR YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 16 / 05 /2018
Tezin Savunma Tarihi : 07 / 06 /2018

Tez Danışmanı : Dr. Öğr. Üyesi Eyüp GEDİKLİ

Trabzon 2018

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Sefa ARAS Tarafından Hazırlanan

PROGRAMLAMA ÖĞRENME ORTAMLARI İÇİN GELİŞTİRİLEN PROGRAMLAMA TEKNİKLERİNİN ANALİZİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 22 / 05 / 2018 gün ve 1754 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Cemal KÖSE …...……...……

Üye : Prof. Dr. Abdulsamet HAŞILOĞLU …...……...……

Üye : Dr. Öğr. Üyesi Eyüp GEDİKLİ ……...……...……

Prof. Dr. Sadettin KORKMAZ

Enstitü Müdürü

ÖNSÖZ

Bu tez çalışmasında basit sözdizimi olan, görev tabanlı, özelleştirilmiş geri bildirimleriyle kullanıcıyı yönlendiren yeni bir programlama öğrenme ortamının tasarımı ve geliştirilmesi yapılmıştır. Geliştirilen ortamda derleyici tasarım tekniklerinin kullanılmasıyla, kullanıcıların yazdığı kodların derinlemesine analizinin yapılması sağlanmaktadır. Analizler sonucunda yapılan doğru yönlendirmelerle başarılı bir programlama öğrenme süreci hedeflenmektedir. Bu ortam bilgisayar bilimleri ve mühendislik teknikleri kullanılarak açık kaynaklı olarak geliştirilmiş olup bu tür ortam geliştirmek isteyen araştırmacılara çatı olma niteliğindedir

Bu yüksek lisans tezimde; bilgi ve tecrübesini benimle paylaşan, değerli zamanını bana ayırıp çalışmamı bilimsel temeller ışığında şekillendiren ve çalışmanın tamamlanmasında büyük katkısı olan danışman hocam Dr. Öğr. Üyesi Eyüp GEDİKLİ'ye ve konu hakkındaki bilgi ve birikimiyle tezin oluşma ve hazırlanma sürecinde büyük yardımı dokunan sayın hocam Dr. Öğr. Üyesi Özcan ÖZYURT'a teşekkürü borç bilirim. Her an yanımda olan, sevgi, ilgi ve bilgisini benden hiçbir zaman eksik etmeyen ve bana her fırsatta yardımcı olan eşim Elif ARAS'a ve manevi desteğini her zaman yanımda hissettiğim tüm aileme sonsuz teşekkür ederim.

Sefa ARAS

Trabzon 2018

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Programlama Öğrenme Ortamları İçin Geliştirilen Programlama Tekniklerinin Analizi” başlıklı bu çalışmayı baştan sona kadar danışmanım Dr. Öğr. Üyesi Eyüp GEDİKLİ’nin sorumluluğunda tamamladığımı, verileri kendim topladığımı, deneyleri ilgili laboratuvarlarda yaptığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 07/06/2018

Sefa ARAS

İÇİNDEKİLER

Sayfa No

Yüksek Lisans

ÖZET

PROGRAMLAMA ÖĞRENME ORTAMLARI İÇİN GELİŞTİRİLEN PROGRAMLAMA TEKNİKLERİNİN ANALİZİ

Sefa ARAS

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Eyüp GEDİKLİ
2018, 61 Sayfa

Programlama öğrenimine erken yaşlarda başlanmasına yönelik yapılan çalışmaların son yıllarda arttığı gözlenmektedir. Bireylerin üst düzey bilişsel becerilerini geliştiren programlama, günümüzde herkes için edinilmesi gereken bir yetkinlik olarak görülmektedir. Mevcut öğrenme ortamlarında genellikle görsel bileşenler kullanılarak programlama yapılmaktadır. Ancak yapılan çalışmalarda görsel bileşenlerle programlama yapmanın gerçek programlama dillerine geçişe katkısı az olduğu sonucuna varılmaktadır. Bu sebeple bu çalışmada herkesin programlama öğrenebileceği yeni bir ortam geliştirilmiştir. Programlama öğrenimini kolaylaştırmak ve gerçek programlama dillerine geçişe katkı sağlamak için basit sözdizimi olan gerçek programlama dillerine benzeyen yeni bir programlama dili tanımlanmıştır. Bu dil sözcüksel, sözdizimsel ve semantik analiz aşamaları ile anlam ve dilbilgisi açısından kontrol edilmektedir. Belirtilen derleyici tasarım teknikleri sonlu durum makineleri kullanılarak uygulanmaktadır. Oluşturulan dilin çözümlenmesi ve anlamlandırılması düzenli ifadeler ile gerçekleştirilmektedir. Kullanıcıya serbest bir çalışma alanı yerine, programlama kavramlarını kapsayan görevlerin bulunduğu bir ortam sunulmaktadır. Görev tabanlı olan bu ortam akıllı geri bildirimleri ile kullanıcıyı yönlendirerek başarılı bir öğrenme süreci hedeflemektedir. Programlama öğrenme ortamı bilgisayar bilimleri ve mühendislik teknikleri kullanılarak açık kaynak kodlu olarak geliştirilmiş olup bu tür ortam geliştirmek isteyen araştırmacılara çatı olma niteliği taşımaktadır.

Anahtar Kelimeler: Programlama öğrenme ortamı, İnsan bilgisayar etkileşimi, Derleyici tasarımı, Düzenli ifadeler, Sonlu durum makineleri

Master Thesis

SUMMARY

ANALYSIS OF DEVELOPED PROGRAMMING TECHQNIQUES FOR PROGRAMMING LEARNING ENVIRONMENTS

Sefa ARAS

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Asst. Prof. Dr. Eyüp GEDİKLİ
2018, 61 Pages

It has been observed that the studies for starting programming learning at an early age have increased in recent years. Programming that enhances the high level cognitive skills of individuals is seen as a competency that must be acquired for everyone nowadays. In current programming learning environments, programming is usually done using visual components. However, it is emphasized that there is little contribution to the transition to actual programming languages by programming with visual components in the studies carried out. For this reason, a new programming learning environment has been developed in this study in which everyone can learn programming. To facilitate programming learning and contribute to the transition of actual programming languages, a new programming language is described which is simple syntax and similar to real programming languages. This language is controlled in terms of meaning and grammar with lexical, syntax and semantic analysis steps. The specified compiler design techniques are implemented using finite state machines. Analysis of the created programming language is performed by regular expressions. Instead of providing a free workspace for the user, an environment with quests covering programming concepts is presented. This quest based environment aims at the successful learning process by guiding to the user through feedbacks. The programming learning environment has been developed as an open source software by using computer science and engineering techniques and it is a framework for researchers seeking to develop a similar environment.

Key Words: Programming learning environment, Human computer interaction, Compiler design, Regular Expressions, Finite state machine

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

AST	Abstract Syntax Tree
BNF	Backus Normal Form
CGI	Computer Generated Imagery
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
IR	Instruction Selection
IS	Instruction Selection
ISC	Instruction Scheduling
JVM	Java Virtual Machine
MIT	Massachusetts Institute of Technology
OS	Operating System
PHP	Hypertext Preprocessor
RA	Register Allocation
UML	Unified Modeling Language

1. GENEL BİLGİLER

1.1. Giriş

Günümüz teknolojisinin önemli bir parçası olan yazılım teknolojisi her geçen gün gelişmekte ve çeşitlenmektedir. Yazılım teknolojisindeki bu çeşitlenmeye rağmen temelinde olan programlama kavramı büyük ölçüde aynı kalmaktadır. Bilgisayar programı en genel biçimde; bilgisayara komutlar yardımıyla işlem yaptırmak olarak tanımlanmaktadır [1]. Aynı bilgisayar programı farklı programlama dilleri ile geliştirilebilmektedir. Bu sebeple bir programlama dilini öğrenmek başka bir programlama dilini öğrenmeyi kolaylaştırmaktadır. Bu bağlamda programlama öğrenimini gerçek programlama dilleri yerine daha basit sözdizimi olan programlama öğrenimi için geliştirilen programlama dilleri ile yapılabilmektedir.

Teknoloji odaklı gelişimin sürdürülebilmesi için yazılım alanında üretken ve yaratıcı bireylerin yetiştirilmesi gerekmektedir. Bu gerekliliğin temelinde büyük ölçüde programlama kavramı bulunmaktadır. Programlama yapmak bilgisayar programı oluşturmanın yanı sıra problem çözmek, analitik düşünmek, sebep-sonuç ilişkisi kurmak anlamına da gelmektedir. Bu sebeple günümüzde programlama becerisi herkes için edinilmesi gereken bir yetkinlik olarak görülmektedir [2, 3]. Bu amaç doğrultusunda programlama öğreniminin geliştirilmesi ve yaygınlaştırılması ülkelerin hedefleri arasına girmiştir [4].

1.2. Çalışmanın Amacı ve Önemi

Programlama öğrenimi; bilgisayar bilimleri kavramlarını öğretmenin yanı sıra bilişsel becerileri de geliştirmektedir. Bu gelişimin sağlanması, programlama öğrenme başarısına bağlı kalmaktadır. Bireyler gerçek programlama dilleriyle programlama öğrenirken sözdizimi zorlukları ile karşılaşmaktadırlar. Bu zorlukları gidermek programlama yapmaktan daha uzun zaman almaktadır [5]. Bu tür problemlere çözüm olması açısından programlama öğrenme ortamları ortaya çıkmıştır.

Programlama öğrenme ortamlarında genellikle görsel bileşenler kullanılarak programlama yapılmaktadır [3]. Ancak görsel bileşenlerle programlama yapmak gerçek programlama dillerine geçişte bireylerin zorlanmasına sebep olmaktadır [5]. Metin tabanlı öğrenme ortamlarında ise basit komutlar kullanılarak programlama yapılmaktadır. Bu komutların yapısı gerçek programlama dillerine benzememektedir. Bu durum gerçek programlama dillerine benzeyen bir öğrenme ortamına olan ihtiyacı işaret etmekte ve bu çalışmanın gerekliliğini ortaya koymaktadır.

Bu çalışmada basit sözdizimi olan görev tabanlı, özelleştirilmiş geri bildirimleri ile kullanıcıları yönlendiren, oyunlaştırma teknikleriyle motivasyon sağlayan yeni bir programlama öğrenme ortamı geliştirilmiştir. Geliştirilen ortam kapsamında gerçek programlama dillerine benzer bir sözdizimi olan yeni bir programlama dili tanımlanmıştır. Programlama öğrenme ortamı bilgisayar bilimleri ve mühendislik teknikleri kullanılarak açık kaynaklı [6] olarak geliştirilmiş olup bu tür ortam geliştirmek isteyen araştırmacılara çatı olma niteliğindedir.

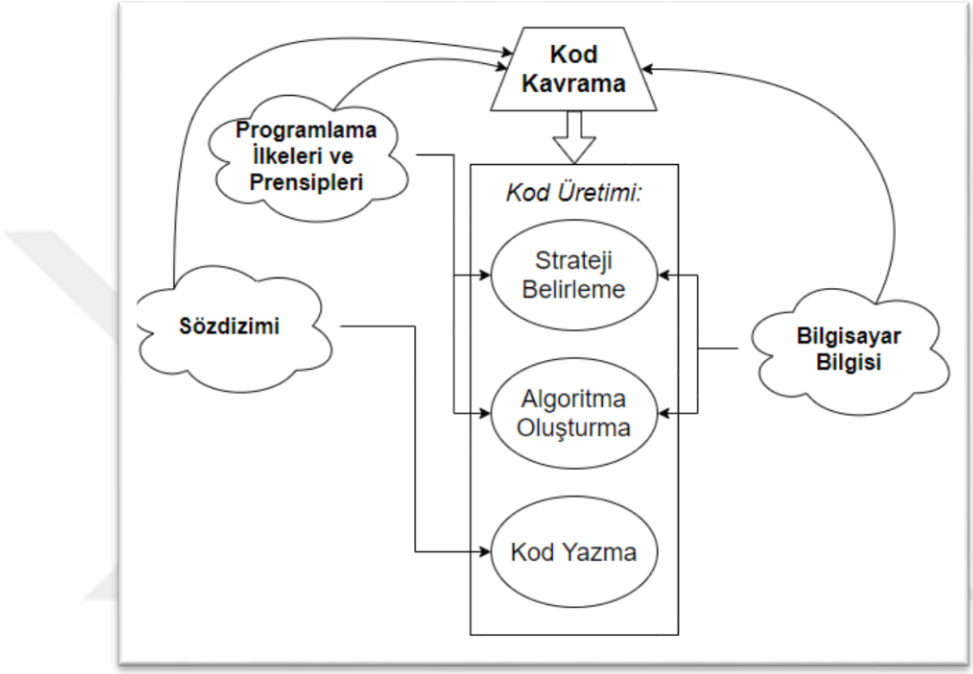
Geliştirilen programlama öğrenme ortamında, oluşturulan dili derlemekten ziyade analiz etme amacı bulunmaktadır. Kullanıcının yazdığı kod analiz edilmekte ve bu analizler sonucunda yapılan doğru yönlendirmelerle başarılı bir öğrenme süreci hedeflenmektedir. Kodun incelenmesi ve ayrıştırılması, sözcüksel ve sözdizimsel analiz aşamaları ile yapılmaktadır. Sonlu özdevinir makineler kullanılarak kodun doğruluğu ve görevin başarı durumu kontrol edilmektedir. Düzenli ifadeler kullanıcının yazdığı kodu ayrıştırarak, kod üzerinde gerekli analizlerin yapılmasını sağlamaktadır. Belirtilen yöntemler sayesinde oluşturulan programlama dili çözümlenerek, kullanıcıya programlama öğrenimine faydalı geri bildirimler verilmektedir.

1.3. Programlama Öğrenimi

Programlamaya yeni başlayan bireylerin ihtiyaç duyduğu farklı türlerdeki bilgi (bilgisayarlar, sözdizimi, programlama ilkeleri ve kavramları gibi) ile kod kavraması ve üretimi arasında bir bağımlılık ilişkisi bulunmaktadır (Şekil 1.1). Programlama yapabilmek için izlenmesi gereken adımlar şu şekildedir [7]:

1. Verilen bir problemin durumu veya gerekliliği incelenmeli, en uygun programlama stratejisinin kullanılması gerekmektedir.

2. Problemi çözmeye yönelik bir algoritma üretilmelidir. Kodun bilgisayarda çalıştırılma bilgisi, programcıların algoritma talimatlarının sırasını etkili ve verimli bir şekilde kullanmasını sağlamaktadır.
3. Algoritmanın kullanılacak programlama dilinde yazılması, programlama dilinin sözdizimi hakkında bilgiye sahip olmak gerekmektedir.



Şekil 1.1. Programcının sahip olması gereken bilgi ve beceri [7].

Programlama stratejisi, belirli bir problemi çözmek için programlama bilgisinin uygulanma biçimini ifade etmektedir. Programlama problemine bir çözüm oluştururken programlama bilgisinin doğru bir şekilde uygulanması için programlama stratejilerinin gelişimi önem kazanmaktadır. Programlama sürecinde önemli olan bilgi ve beceriler Şekil 1.1'de gösterilmektedir [7]. Bir programlama öğrenme ortamı bu bilgi ve becerileri dikkate alarak geliştirilmesi gerekmektedir.

Programlama öğrenimi; bireylerin problem çözme [8, 9], matematik [10], mantık [11], programlama becerisi [12], sayısal düşünme [13], algoritmik düşünme [14] ve farklı düşünme [15] gibi üst düzey bilişsel becerilerini geliştirmektedir. Bu becerilerin yanı sıra özel bir durum çalışmasında işitme engelli bir bireyin programlama öğrenerek sosyal yönlerinin geliştiği gözlemlenmiştir [16].

1.3.1. Erken Yaşlarda Programlama Öğrenimi ve Önemi

Erken yaşlarda temel programlama kavramlarının öğretilmesi eğitim ve bilim camiasının ilgisini çekmektedir [17]. Uluslararası araştırmalar erken yaşlarda programlama öğreniminin bilişsel işlevlerin gelişimi üzerinde önemli bir etkiye sahip olduğunu vurgulamaktadır [18, 19].

Programlama öğrenimi bireyler için zor ve zaman alıcı bir süreç olarak kabul edilmektedir [20, 21]. Bireyler programlama öğrenirken bilgisayar bilimleri kavramlarını anlamakta güçlük çekmektedirler. Lisans düzeyindeki öğrenciler bile programlamayı kafa karıştırıcı bulmaktadırlar. Bu sebeplerden dolayı programlama öğrenimine erken yaşlarda başlanması önem kazanmaktadır [11].

Son yıllarda programlama öğrenimine erken yaşlarda başlanmasına yönelik çeşitli adımlar atılmaktadır. Teknolojinin erken yaşlarda öğrenilmesi için yeni öğrenme standartları ve uygulamaları oluşturulmaktadır [22]. Bireylerin programlamaya olan ilgisinin artması ve olumlu tavır sergilemesi için yeni yöntemler geliştirilmektedir [17, 23]. Gelişmiş ülkeler bilgisayar bilimlerinin erken yaşlarda öğrenilmesi için müfredatlarını güncellemişlerdir [10, 13]. Yapılan çalışmalar programlama öğreniminin erken yaşlarda yapılabileceğini [17, 24, 25] ve bu eğitimin olumlu sonuçları olduğunu [11, 19, 25] ortaya koymaktadır.

1.3.2. Programlama Öğrenme Ortamları

Java veya C# gibi gerçek programlama dilleri, bilgisayarın hesaplama tarzına çok benzeyen bir gösterimi bulunmaktadır. Buna karşılık görsel programlama dilleri, insan diline daha yakın bir temsili kullanmaktadır. Görsel programlama dilleri ve öğrenme ortamları genellikle sözdizimini daha az ve basit tutarak sözdizimsel karmaşıklığı azaltmaktadır. Bu özellik sayesinde bireyler, programlama yazım karmaşıklığından kurtulup programlama mantığını anlamaya çalışmaktadırlar [24]. Bu sebeple programlama öğrenimini kolaylaştırmak için programlama öğrenme ortamları geliştirilmektedir.

1.3.2.1. Scratch

Scratch interaktif hikâyelerin, animasyonların ve oyunların oluşturulabileceği ücretsiz, açık kaynak kodlu ve multimedya açısından zengin bir programlama öğrenme ortamıdır [26]. Scratch kullanıcıların bilgisayar bilimleri kavramlarını daha iyi anlaması için animasyonlar ve sunumlar izlemesine imkân vermektedir. Görsel bir programlama dili olan Scratch ortamında sürükle bırak yöntemiyle kod blokları çerçeveye aktarılmakta ve kod bloklarının birleştirilmesiyle de program oluşturulmaktadır [11]. MIT Medya Laboratuvarı tarafından 2013 yılında geliştirilmiş ve 40 farklı dil desteği bulunmaktadır. Özellikle 8 ve 16 yaş grubu bireyler için tasarlanmış olsa da her yaş grubuna hitap etmektedir [27].

Scratch; animasyon ekranı, görsel bileşen parçaları ve çalışma alanı olmak üzere üç ana bölümden oluşmaktadır. Görsel bileşenlerin sürükle bırak yöntemiyle çalışma alanına aktarılmasıyla yapılan programın sonucu animasyon ekranında görüntülenmektedir. Scratch ortamında program; değişken, ifade, koşul ve döngü gibi program komutlarını ifade eden görsel bileşenlerden oluşmaktadır. Bu etkileşim modeli öğrencilerin motivasyonunu olumsuz etkileyen sözdizimi hatalarını önlemektedir [5, 28]. Scratch en çok tercih edilen programlama öğrenme ortamı olup üst düzey bilişsel becerileri geliştirdiği yapılan çalışmalarda vurgulanmaktadır [29, 30]. Bu ortamın önemli bir tasarım hedefi de; kullanıcının bu ortam üzerinde kendi başına programlama yapmayı öğrenebilmesidir [28].

1.3.2.2. ScratchJr

ScratchJr MIT Medya Laboratuvarı tarafından geliştirilen programlama öğrenme ortamıdır. ScratchJr popüler bir ortam olan Scratch programlama dilinden esinlenmektedir. Erken yaşlardaki bireylere yönelik geliştirilmiş olup buna uygun tasarım öğelerinin kullanılmasına dikkat edilmektedir. ScratchJr görsel programlama dili kullanılarak etkileşimli hikâyeler ve oyunlar oluşturmaya uygun bir altyapı sunmaktadır [31]. Ortam web ve mobil platformlarına uygun olarak geliştirilmiştir.

ScratchJr ortamında kullanılan programlama dili, programlama bloklarından oluşmaktadır. Blok şekillerinin tasarımı gereği sözdizimi hatası yapmak mümkün olmamaktadır. Ortamda bulunan program blokları sözdizimsel sıralamasına göre yapboz parçaları gibi birleştirilmektedir. Geleneksel programlama dillerinde olduğu gibi yukarıdan

aşağıya doğru bir yazım formatına karşın ScratchJr soldan sağa doğru bir yazım formatı sunmaktadır [25].

ScratchJr 28 farklı bloktan oluşmaktadır. Kullanıcılar blokları uygun şekilde birleştirerek ekranda görünen karakterlere komut göndermektedirler. Kolay kullanıma sahip arayüzü genç kullanıcılar için rahatlık sağlamaktadır. Metin, renk ve sesler eklenerek yapılan projeler özelleştirilebilmektedir. Bu yönüyle ScratchJr kullanıcılara eğlenceli bir ortamda problem çözme becerilerini geliştirme imkânı sunmaktadır [18].

1.3.2.3. Code.org

Code.org, kod bloklarının sürükle bırak yöntemiyle çerçeveye taşınıp birleştirilmesi sonucu program oluşturulmasını sağlayan programlama öğrenme ortamıdır [32]. Ortam bir oyun olarak yapılandırılmış olup çevrimiçi eğitici videoları da barındırmaktadır. Bu eğitim materyalleri yardımıyla kullanıcılar programlama kavramlarını öğrenebilmektedirler [4]. Code.org 2013 yılından itibaren 45 farklı dil desteği ile kullanıcıların hizmetine sunulmuştur. Her yaş grubu için ayrı etkinlikler hazırlayarak programlama öğrenme sürecini kolaylaştırmaktadır. Ayrıca bloklar ile oluşturulan programın, JavaScript programlama dilindeki karşılığı kullanıcıya gösterilmektedir. Bu işleviyle birlikte gerçek programlama dillerine geçişi kolaylaştırma hedefi bulunmaktadır [27].

Code.org ile programlama öğrenen bireyler, programlamaya karşı olumlu tutum sergilemektedirler. Ancak bu öğrenme ortamı, bireylerin problem çözmeye yönelik yansıtıcı düşünme becerisine olumlu bir etki yapmamaktadır [4].

1.3.2.4. Cherp

Cherp, kullanıcıların programlamaya ilgi çekici bir giriş yapmasını sağlamak için geliştirilen görsel programlama dilidir. Kullanıcılar görsel bileşenler ile geliştirdiği programın çıktısını robotlar üzerinde görebilmektedir [33]. Cherp programlama öğrenme ortamında görsel bileşenler, robotların şekil ve hareketlerinden esinlenilerek oluşturulmuştur. Bu ortamda programlama yaparken sözdizimsel hata yapılması mümkün olmamaktadır. Sözdizimsel hatayı ortadan kaldırmak için görsel bileşenlerin çerçeveye

yerleştirilmesi sırasında fiziksel engeller ortaya çıkmaktadır. Ancak doğru bir sözdizimi olduğu zaman görsel bileşenler birleşebilmektedir [10].

1.3.2.5. Vimap

Vimap; özel amaçlar için kolayca değiştirilebilen ve genişletilebilen, çok katmanlı, esnek bir programlama dili ve modelleme ortamıdır. Kullanıcılar ortam üzerinde görsel simülasyonlar oluşturabilmekte veya mevcut simülasyonu düzenleyebilmektedir. Kullanıcılar program geliştirme sürecinde değişimleri gerçek zamanlı olarak görebilmektedirler. Vimap, sistem çalışırken ilgili parametreleri güncelleyerek dinamik bir ortam sağlamaktadır. Kullanıcılar mevcut görsel bileşenleri kullanarak program geliştirmenin yanı sıra özelleştirilmiş bir görsel bileşen oluşturup, bunu program geliştirme aşamasında kullanabilmektedirler [21].

1.3.2.6. Ladybug

Utah Üniversitesinde geliştirilen Ladybug, Logo tabanlı programlama öğrenme ortamıdır. Kullanıcılar kısıtlı bir kelime haznesine sahip olan basit bir yazılım aracıyla uğur böceğine komutlar göndermektedir. Programlama sadece; geriye git, ileri git, sağa dön ve sola dön komutlarıyla yapılmaktadır. Komutlar kullanım kolaylığı açısından basit düğmeler ile uygulanabilmektedir. Program, belirtilen komutları içeren bir dizi olarak birikmektedir. Kullanıcı istediği zaman bu dizide, komut ekleme veya çıkarma işlemi yaparak programını güncelleyebilmektedir. Uğur böceği, gönderilen komutları uygularken, hareket ettiği yolda çizgiler oluşturmaktadır [9].

Ladybug ortamında kullanıcıdan verilen görevi başarılı ile bitirmesini beklenmektedir. Verilen görevler girilen labirentlerden çıkılması üzerine kurgulanmaktadır. Bu aktivite programlama öğrenimi sırasında problem çözme becerisinin gelişimini de desteklemektedir [9].

1.3.2.7. RoboMind

RoboMind, önceden herhangi bir programlama bilgisine sahip olmayan kullanıcılar tarafından otomasyona ve programlamaya giriş amacıyla kullanılan programlama öğrenme ortamıdır. Kullanıcılar RoboMind ile harita üzerinde, bir robot aracılığıyla dünyadaki nesnelerle iletişime girmektedir [34]. Bu ortam Java programlama dilinin sözdizimine benzer metin tabanlı programlama dili sağlamaktadır. Standart bir komut seti robotun hareketini kontrol etmektedir. Döngüler, iç içe geçmiş ifadeler ve kullanıcı tanımlı prosedürler gibi temel programlama kavramları kullanılmaktadır. RoboMind kullanıcıya serbest programlama yapabileceği bir metin düzenleyicisi sağlamaktadır [7].

1.3.2.8. Kibo

Kibo, Tuft Üniversitende Dev Tech araştırma grubu tarafından geliştirilmiş ve Kinder Lab tarafından ticarileştirilmiş programlama öğrenme ortamıdır. Temel mühendislik ve programlama kavramlarını öğretmek için tasarlanmıştır. Bu ortamda ahşap bloklar, ekran zamanı olmadan bir araya getirilerek programlama yapılmaktadır [35]. Kibo kiti; tekerlekler, motorlar, ışıklandırma ve çeşitli sensörler de dâhil olmak üzere kolay bağlantılı robotik malzemeler içermektedir. Programlama yapılırken kullanılan ahşap bloklarının her birine ait barkodlar bulunmaktadır. Kibo robotunun önüne yerleştirilmiş bir tarayıcı bu barkodları tarayarak robotlara programı göndermektedir. Kibo her programlama dilinde olduğu gibi bir sözdizimine sahiptir [17].

Kullanıcılar Kibo kullanarak giderek artan karmaşık programları ve dilin sözdizimi kurallarını yönetmeyi öğrenerek çalışma belleği becerilerini güçlendirmektedir. Robotlar ile çalışmak, kullanıcıların işbirliği içinde takım çalışması yapma olgusunu geliştirmektedir [17]. Robotik kitlerin mühendislik ve teknoloji kavramlarını öğretme başarısı diğer ortamlara göre daha yüksek olduğu söylenmektedir [36].

1.3.2.9. Greenfoot

Greenfoot erken yaşlardaki bireyler programlama öğretmeyi amaçlayan bütünleşmiş bir ortamdır. Bu ortamda gerçek programlama dili olan Java ile programlama yapılmaktadır.

Greenfoot dâhili olarak Java sanal makinesini (JVM) kullanmaktadır [37]. Java dilini kullanmasından dolayı nesne yönelimli programlamayı desteklemektedir. Greenfoot ortamının en önemli amacı nesneleri görselleştirerek nesne yönelimli programlama kavramının iyice anlaşılmasını sağlamaktır [38].

1.4. Oyunlaştırma

2008 yılında ortaya çıkan oyunlaştırma kavramı 2010 yılından itibaren yaygın olarak kabul görmeye başlamıştır. Oyunlaştırma; oyun tasarım öğelerinin oyun dışı ortamlarda kullanılması olarak tanımlanmaktadır [39]. Oyunlaştırma kavramının insan bilgisayar etkileşimini destekleme, eylemlerin kalitesini ve verimliliğini artırma, motivasyon sağlama gibi özellikleri ön plana çıkmaktadır [40].

Oyunlaştırma, ödüller ile özdeşleşen bir kavram olarak bilinmektedir. Çoğu oyunlaştırma sistemi ilgiyi artırmak için puan, seviye, skor, başarı veya rozet kazandırmaya odaklanmaktadır. Ödül sistemi ödüller gelmeye devam ettiği sürece çalışmaktadır. Ödül durduğunda veya sabit kaldığında ilgi çekici özellikler kaybolmaya başlamaktadır. Ayrıca ödül karşılığında yapılan bir görevin, ödül olmadan yapılma ihtimali oldukça düşük olmaktadır. İlgi çekiciliğin kaybolmasını önleyen en etkili ödül sistemi; farklı seviyelerde artan oranda ödüller verilen sistemler olduğu söylenmektedir [41].

Oyunlar sadece eğlence sektöründe değil; savunma, eğitim, bilimsel keşif, sağlık hizmetleri, acil durum yönetimi, şehir planlaması, mühendislik, din ve politika gibi endüstriler tarafından da kullanılmaktadır. Bu tür uygulamalara ciddi oyun denilmekte ve eğitmek, araştırmak veya tanıtmak amacıyla kullanılmaktadır. Ciddi oyunlar (eğitsel oyunlar) fazla miktarda kaynak, oyun tasarım bilgisi ve grafik kullanırken e-öğrenme ortamları tasarımı fazlaca önemsememektedir. Bu sebeple eğitsel oyunlar ilgi çekmede daha başarılı olduğu görülmektedir [42].

Geleneksel öğrenme yöntemleri birçok öğrenci tarafından etkisiz ve sıkıcı olarak algılanmaktadır. Araştırmacılar sürekli olarak yeni öğretim yaklaşımları aramasına rağmen, öğrenci motivasyonu ve katılımı konusunda yetersiz kalındığı büyük ölçüde kabul görmektedir. Eğitsel oyunların öğrenme araçları olarak kullanılması umut veren bir yaklaşım olarak kabul edilmektedir. Eğitsel oyunlar bilgiyi öğretmenin yanı sıra problem çözme, işbirliği ve iletişim gibi önemli becerileri de desteklemektedir [43]. Oyunlaştırmanın

öğrenme sürecine büyük ölçüde olumlu etki ettiği yapılan çalışmalarda ifade edilmektedir [44].

1.5. Dil Çeviricileri

Programlama dilleri, makinelere hesaplamaları açıklayan gösterimler olarak tanımlanmaktadır. Çalışan tüm yazılımlar programlama dillerinde yazıldığından dolayı bütün bilgisayarlar programlama dillerine bağlı olmaktadır. Ancak bir programın çalıştırılmadan önce bir bilgisayarın yürütebileceği forma dönüştürülmesi gerekmektedir. Bu çeviriyi yapan yazılım sistemlerine dil çeviricileri (dil işlemcileri) denilmektedir [45].

Şekil 1.2. Dil çeviricilerinin genel yapısı

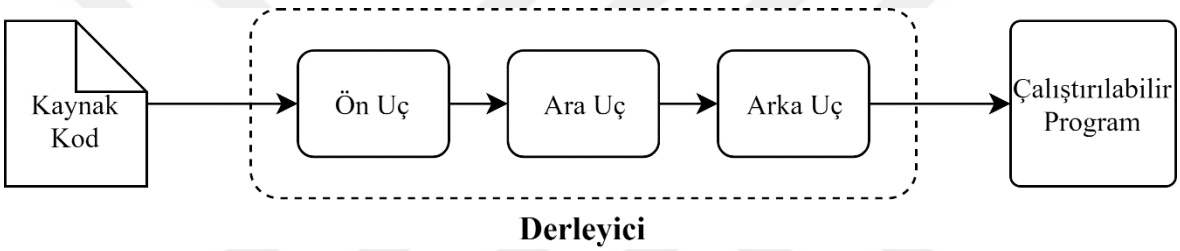
En genel haliyle çeviriciler, belirli bir dildeki program metnini (kaynak dil) girdi olarak kabul eden ve makinenin anlayacağı farklı bir dilde aynı anlamlı program metnini (hedef dil) çıktı olarak üreten programlar olarak tanımlanmaktadır (Şekil 1.2). Bu süreç çeviri olarak adlandırılmaktadır [46]. Dil çeviricileri; yorumlayıcılar ve derleyiciler olmak üzere ikiye ayrılmaktadır.

1.5.1. Yorumlayıcılar

Yorumlayıcılar, kaynak programlama dilinde yazılan programları bloklar halinde veya satır satır hedef programa çevirmektedir. Yorumlama işleminde ilk anda programın bütününe bakılmadığı için, hedef kod çalıştırılmadan önce hata yakalanamamaktadır. Yorumlayıcı, programı satır sırasına göre çevirmektedir. Bu sebeple yorumlayıcı, hatalı olan kod parçası geldiği anda hata vermekte ve programdaki diğer hatalar ile ilgilenmeden programı sonlandırmaktadır. Yorumlayıcılar; kaynak programlama dilini çalışma zamanında hedef programlama diline dönüştürmektedir. Bu durum yorumlayıcıların yavaş çalışmasına sebep olmaktadır [47].

1.5.2. Derleyiciler

Derleyiciler bir dilde yazılmış programı başka bir dile çeviren araçlar olarak bilinmektedir. Metni bir dilden başka bir dile çevirmek için hem giriş hem de çıkış dilinin biçiminin, sözdiziminin ve içeriğinin bilinmesi gerekmektedir. Derleyiciler, içeriği kaynak dilden hedef dile eşlemek için bir şemaya ihtiyaç duymaktadır. Derleyicilerde kaynak dil için ön uç, hedef dil için arka uç bulunmaktadır. Ön ve arka ucu birbirine bağlamak için ise dilden bağımsız bir ara uç bulunmaktadır (Şekil 1.3). Ara uç bölümünde; yazılan kaynak kod, hedef koda dönüştürülmeden önce mümkün olduğunca uygun programa dönüştürülmektedir [48].



Şekil 1.3. Derleyicilerin genel yapısı

Bir bilgisayar programı kısa komutlar içeren makine diliyle oluşturulmaktadır. Makine dilinde programlama yapmak uzun süren hata ayıklama sürecine sebep olmaktadır. Bundan dolayı insanların daha kolay program geliştirebileceği yüksek seviyeli programlama dilleri ortaya çıkmıştır. Yüksek seviyeli programlama dilleri programlama sürecini kolaylaştırmaktadır. Ancak bu dilleri bilgisayarların anlayabilmesi için tercümanlara (çeviriciler) ihtiyaç duyulmaktadır. Bir derleyici geliştirilen programları bilgisayarın anlayabileceği düşük seviyeli programlama dillerine dönüştürmektedir. Bu süreçte derleyici, programdaki hataları tespit etmeye ve raporlamaya da çalışmaktadır [49].

Programlama için yüksek seviyeli bir dil kullanmak hızlı program geliştirme üzerinde büyük bir etkiye sahip olmaktadır. Bunun başlıca sebepleri şunlardır:

- Makine dillerine kıyasla programlama dilleri tarafından kullanılan notasyon insanların problem hakkında düşündükleri çözüme daha yakın bir temsili kullanmaktadır.
- Derleyiciler bazı bariz programlama hatalarını tespit edebilmektedir.

- Yüksek seviyeli programlama dilinde yazılan programlar makine dilinde yazılmış eşdeğer programlardan daha kısa olma eğilimi göstermektedir.

Yüksek seviyeli programlama dili kullanmanın bir diğer avantajı ise aynı programın birçok makine diline derlenebilmesi ve böylece birçok farklı makinede çalıştırılabilmesidir. Öte yandan yüksek seviyeli bir dilde yazılmış ve otomatik olarak makine diline çevrilmiş programlar, makine dilinde kodlanmış programlardan daha yavaş çalışmaktadır. Bu nedenle bazı kritik programlar hala makine dilinde yazılmaktadır. Bununla birlikte iyi bir derleyici, iyi yapılandırılmış programları çevirirken, makine kodunun hızına yakın olabilmektedir [49].

1.5.2.1. Ön Uç

Derleyicinin bir programı makine koduna çevirebilmesi için kaynak programlama dilinin hem sözdizimini hem de anlamını bilmesi gerekmektedir. Bu bilgiler derleyicinin ön uç bölümünde bulunmaktadır. Ön uç, kaynak programlama dilinin çözümlenmesini ve ara temsil (IR) olarak adlandırılan ara kodun ara uca aktarılmasını sağlamaktadır. Ön uç, dönüştürme işlemini başarı ile gerçekleştiremezse ilgili sözdizimi hatasını kullanıcıya rapor etmektedir.

Matematiksel olarak kaynak dil, belirli sonlu kurallar kümesi tarafından tanımlanan bir dilbilgisi olarak adlandırılan sonsuz dizeyi ifade etmektedir. Ön uçta tarayıcı ve ayrıştırıcı olarak tanımlanan yapılar; giriş kodunun, dilbilgisi tarafından tanımlanan geçerli kural kümesinin bir üyesi olup olmadığını belirlemektedir [48].

1.5.2.1.1. Ara Temsil

Derleyicinin ön ucunda yapılan son işlem ara temsili (ara kodu) oluşturulmasıdır. Ara temsilin kaynak dile, hedef dile ve derleyicinin uyguladığı özel dönüşümlere bağlı olarak farklı gösterim şekilleri bulunmaktadır. Ara temsiller grafikler ve semboller ile gösterilebildiği gibi makine koduna benzer şekilde de gösterilebilmektedir. Her derleyici kaynak ve hedef dili göz önünde bulundurarak ara temsilin gösterimine dair bir strateji belirlemektedir. Özel seçimler, derleyicinin kodu dönüştürme ve iyileştirme yeteneğini etkilemektedir [48].

1.5.2.2. Ara Uç

Ön uç, giriş programında karşılaştığı ifadeleri sırayla ele almaktadır. Bu nedenle ilk ara temsil derleyicinin oluşturabileceği herhangi bir çerçevede çalışacak genel uygulama stratejilerini içermektedir. Ancak çalışma zamanında kod daha kısıtlı ve öngörülebilir bir bağlamda yürütülmektedir. Ara uç, kodun ara temsil biçimini incelemekte ve bu kodu daha verimli bir şekilde hesaplamaktadır [48]. Gereksiz kodun silinmesi, kullanılmayan bellek alanlarının boşaltılması ve ortak ifadelerin sadeleştirilmesi gibi iyileştirmelerle programın çalışma zamanını olumlu yönde etkilemektedir [47].

```
a = 1
b = 3
c = 5
for i = 1 : n
        read d
        a = a * 2 * b * c * d
end
```

(a) Yazılan Kod

```
a = 1
b = 3
c = 5
t = 2 * b * c
for i = 1 : n
        read d
        a = a * t * d
end
```

(b) Optimim Kod

Şekil 1.4. Ara ucun kodu optimize etmesi

Bu aşama optimize edici olarak da adlandırılmaktadır. Örnekte (Şekil 1.4) verilen giriş kodunun (a) optimum olmadığı görülmektedir. Döngü içerisinde bulunan çarpma işleminde, döngünün ürettiği sonuca bağlı olmayan hesaplamaların döngü dışında hesaplanması gerekmektedir. Bu hesaplama işleminin döngü içerisinde yapılması **bir** kez yapılacak hesaplama işleminin **n** kez yapılmasına sebep olmaktadır. Bu hesaplama işleminin döngü dışına taşınması kodu optimum (b) hale getirmektedir.

1.5.2.3. Arka Uç

Derleyicinin arka ucu, programın ara temsil formunu makine koduna dönüştürmektedir. Bu aşamaya kod üretim aşaması da denilmektedir. Her ara temsil işlemi için makine operasyonları oluşturulmakta ve operasyonların verimli bir şekilde yürütülmesi

için emirler belirlenmektedir. Değerlerin yazmaçlarda (register) veya hafızalarda (memory) saklanmasına karar verilmektedir [48]. Bu işlemler hedef makine mimarisine bağlı olarak gerçekleştirilmektedir [47]. Arka uç hedef programlama dilinin hem sözdizimini hem de anlamını bilmektedir.

1.5.2.3.1. Komut Seçimi (IS)

Kod oluşturma işleminin ilk aşamasında ara temsil formundaki koda karşılık makine kodu seçilmektedir. Komut seçimi her bir ara temsil işlemini, bir veya daha fazla sayıda hedef makine işlemine eşleştirmektedir [48]. Komut seçici, hedef makinenin komutlarına bağlı seçimler yapmaktadır. Hedef makinede bulunan özel komutlardan faydalanabilmektedir. Özel komutların kullanılması performans veya hafıza kullanımı bakımından fayda sağlamaktadır.

1.5.2.3.2. Kayıt Tahsisi (RA)

Komut seçimi sırasında derleyici, hedef makinenin sınırlı sayıda yazmaçlara sahip olduğu gerçeğini kasıtlı olarak görmezden gelmektedir. Sanal yazmaçları kullanarak yeterli sayıda yazmaçlara sahip olduğunu varsaymaktadır. Kayıt tahsisi aşamasında bu sanal yazmaçlar hedef makinedeki gerçek yazmaçlar ile eşlenmektedir. Bu noktada hangi değerlerin hangi yazmaçlarda saklanması gerektiğine karar verilmektedir. Kayıt tahsisi aşamasında kod yeniden yazılmakta ve programın yeterliliğini sağlayan en az sayıda yazmaç kullanılmaktadır [48].

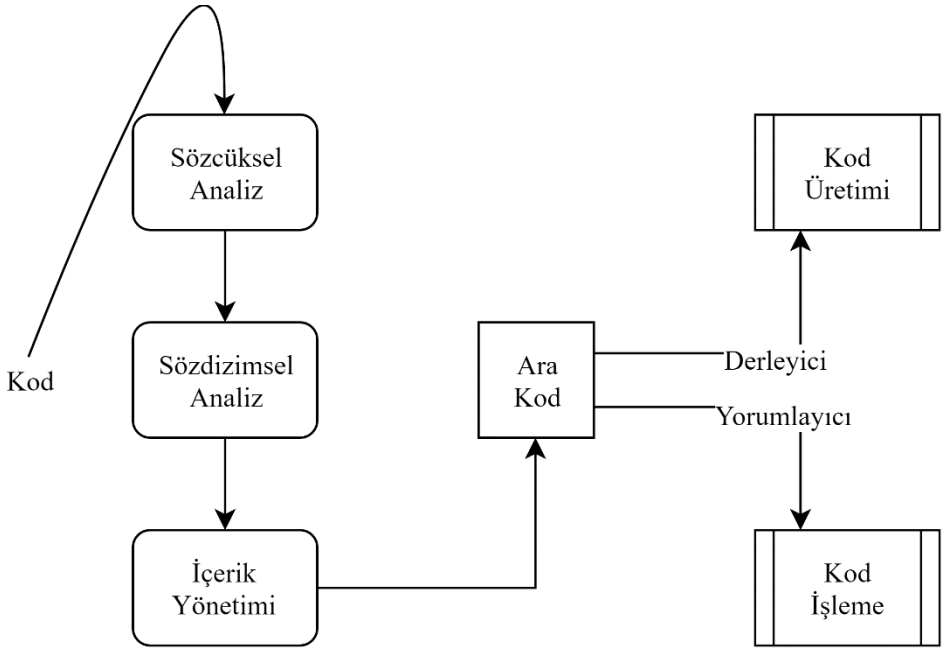
1.5.2.3.3. Talimat Planlama (ISc)

Hızlı bir şekilde yürütülecek kod üretmek için talimat planlayıcısı hedef makinenin performans kısıtlarını dikkate alarak işlemleri yeniden sıralamaktadır. Farklı işlemlerin çalışma zamanı (execution time) farklı sürelerde gerçekleşmektedir. Bellek erişim işlemleri uzun süren işlemler olurken aritmetik işlemler, özellikle bölme, daha kısa sürede gerçekleşmektedir. Bu süreler iyi planlanmaz ise kodun performansı önemli ölçüde azalmaktadır [48].

Birçok işlemcide bir işlem yürütülürken yeni bir işlem başlatabilecek donanım desteği bulunmaktadır. Bir işlemin çıkış değerini başka bir işlem giriş değeri olarak almadığı sürece program normal olarak yürütülmektedir. Ancak bir işlem uzun çalışma süresi olan başka bir işlemin sonucunu erken okumayı denerse, işlemci çalışma süresi uzun olan işlem tamamlanana kadar değere ihtiyacı olan işlemi geciktirmektedir. Bu işlem ihtiyaç duyduğu giriş değeri hazır olana kadar çalışmaya başlayamamakta ve tüm programın performansını olumsuz yönde etkilemektedir. Bundan dolayı işlemlerin iyi bir şekilde planlanması programın çalışma zamanını ciddi şekilde kısaltmaktadır.

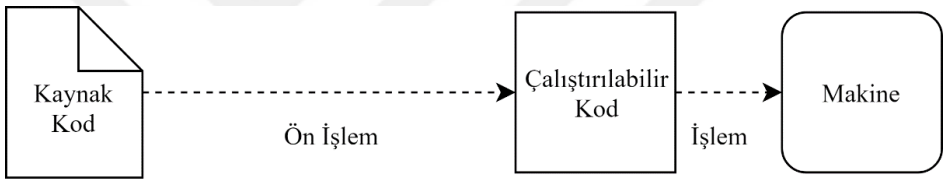
1.5.3. Yorumlayıcılar ve Derleyiciler Arasındaki Farklar

Derleyiciler ve yorumlayıcılar yaptığı işlemler bakımından büyük ölçüde aynı kalmaktadır. Derleyiciler sözdizimi ağacından (AST) çalıştırılabilir kod üretirken, yorumlayıcılar sözdizimi ağacı ifadelerini doğrudan işlemektedir (Şekil 1.5). Yorumlayıcıların sözdizimi ağacının aynı parçasını birçok kez işlemesi derleyicilerden daha yavaş çalışmasına sebep olmaktadır [45]. Ancak yorumlayıcı geliştirmek derleyici geliştirmeye göre daha kolay olduğundan, yorumlayıcılar performansın önemli olmadığı sistemlerde tercih edilmektedir [49].

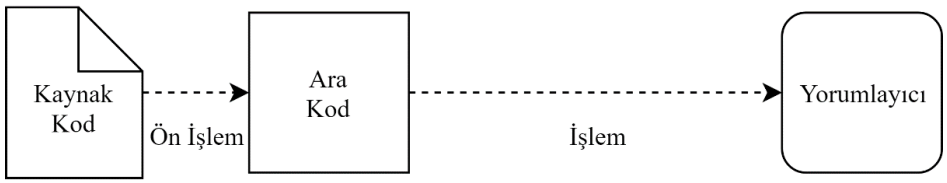


Şekil 1.5. Yorumlayıcı ve derleyici arasındaki işlem farkları

Derleyiciler, işlem adımlarını kaynak programlama dilinde yazılan kodun tamamına uygulamaktadır. Yazılan programda sözdizimsel bir hata mevcutsa kod üretim aşamasına geçmeden ilgili hatayı rapor etmektedir. Programın tamamı tek seferde analiz edildiği için ön işlem aşaması uzun, programın çalışma zamanı kısa olmaktadır. Derleyici kullanan programlama dillerine örnek olarak Pascal, C, Visual Basic gibi programlama dilleri verilebilmektedir. Yorumlayıcılar, işlem adımlarını koda satır satır uygulamaktadır. Programda hata mevcutsa, yorumlayıcı hataya ulaşana kadar programı çalıştırmakta ve hata ile karşılaştığında programı sonlandırmaktadır. Program satır satır analiz edildiği için ön işlem aşaması kısa, programın çalışma zamanı uzun olmaktadır. Yorumlayıcı kullanan programlama dillerine örnek olarak HTML, PHP, JavaScript gibi programlama dilleri verilebilmektedir. Bu dillerin yanı sıra hem derleyici hem de yorumlayıcı kullanan Java gibi programlama dilleri de bulunmaktadır.



(a) Derleyici



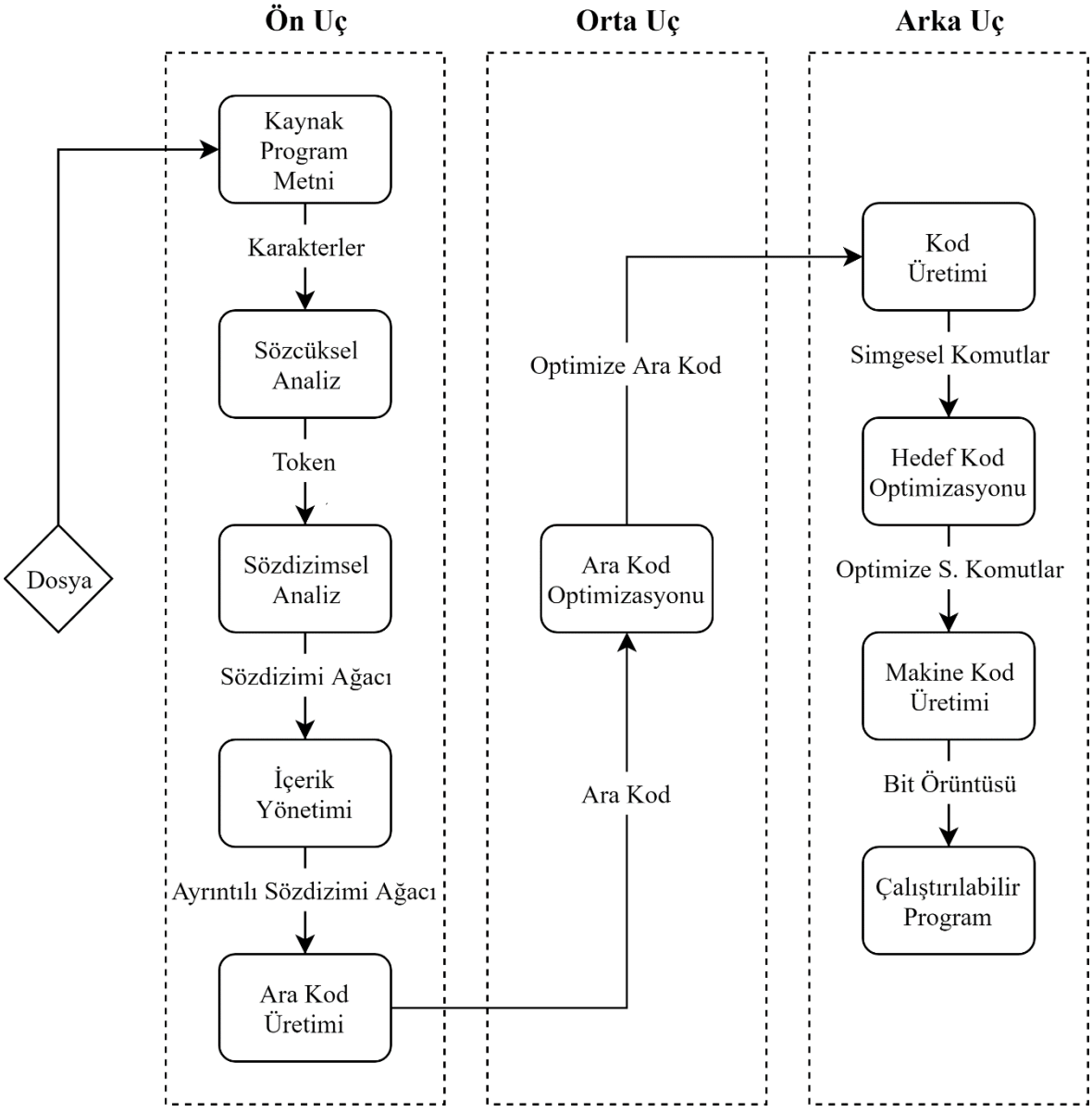
(b) Yorumlayıcı

Şekil 1.6. Yorumlayıcı ve derleyici arasındaki işlem süre farkı

Derleyiciler kaynak programlama dilinden hedef makine üzerinde çalışabilecek kod üretirken (Şekil 1.6.a), yorumlayıcılar kaynak programlama dilinden ürettiği ara kodu hemen işlemektedir (Şekil 1.6.b). Derleyicilerin kod üretim aşaması uzun sürerken, yorumlayıcılar satır satır çevirme işlemi yaptığından dolayı kod üretim aşaması daha kısa sürmektedir. Ancak çalışma zamanı açısından değerlendirme yapıldığında derleyicilerin çıktısı yorumlayıcıların çıktısına göre daha iyi performans göstermektedir.

1.6. Derleyici Aşamaları

Derleyici en genel haliyle, kaynak programı semantik olarak eşdeğer bir hedef programa eşleyen yazılım olarak tanımlanmaktadır. Bu eşleşme işlemi; analiz, optimize ve sentez olarak adlandırılan üç işlem adımından oluşmaktadır. Analiz işlemi ön uçta, optimize işlemi ara uçta, sentez işlemi ise arka uçta gerçekleşmektedir (Şekil 1.3).



Şekil 1.7. Derleyici işleyiş diyagramı

Analiz bölümünde; kaynak programlama dili, dilbilgisi uygunluğu açısından analiz edilmektedir. Program, sözdizimi ve semantik açısından değerlendirilmekte ve hata varsa bilgilendirici mesaj üretilmektedir. Yapılan analiz ile ara uca gönderilecek ara kod oluşturulmaktadır. Optimize bölümünde; ara kod, en uygun biçimde yeniden yazılmakta ve iyileştirilmiş ara kod sentez işlemi için arka uca gönderilmektedir. Sentez bölümünde; iyileştirilmiş ara kod, çalıştırılabilir programa dönüştürülmektedir. Kod üretim aşaması kaynak makineye göre gerçekleştirilmektedir. Bütün işlem adımlarının uygulanmasından sonra, program kaynak makinede çalıştırılabilmektedir [45].

Derleyici işlem adımları ve her adımın ürettiği çıktılar Şekil 1.7'de gösterilmektedir. Kaynak program metni modülü kaynak program dosyasını bulmakta, etkin bir şekilde okumakta ve karakter akışına dönüştürmektedir. Sözcüksel analiz (lexical analysis) modülü karakter akışını tokenlere ayırmaktadır. Tokenler, kodları açıklayıcı ifadelerden oluşmaktadır. Sözdizimsel analiz (syntax analysis) modülü tokenleri sözdizimi ağacına dönüştürmektedir. İçerik yönetim modülü programdaki içerik bilgilerini toplamakta ve sözdizimi ağacı düğümlerine not etmektedir. Ara kod üretim modülü sözdizimi ağacındaki kaynak dile özgü yapıları daha genel yapılara çevirmektedir. Ara kod genellikle ifade ve kontrol akış komutlarından oluşmaktadır. Ara kod optimizasyon modülü kod üretim aşamasının etkinliğini artırmak amacıyla ara kod üzerinde iyileştirmeler yapmaktadır. Kod üretim modülü; ara kodu, simgesel bir hedef makine komut listesine dönüştürmektedir. Komut seçimi, kayıt tahsisi ve talimat planlama işlemleri bu aşamada gerçekleştirilmektedir.

Hedef kod optimizasyon modülü simgesel makine komut listesini, daha hızlı veya daha kısa komutlar ile değiştirerek iyileştirmeye çalışmaktadır. Makine kod üretim modülü simgesel makine komutlarını ilgili bit modellerine dönüştürmektedir. Bit örüntüsü kullanılarak işletim sisteminin gereklerine göre çalıştırılabilir program oluşturulmaktadır [46].

1.6.1. Sözcüksel Analiz

Derleyicinin ilk aşamasında program metnine ait karakter dizisinin sözcüksel analizi yapılmaktadır [45]. İlk olarak program metninde bulunan boşluk, sekme, yeni satır gibi beyaz karakterler (white-space) filtrelenmektedir [49]. Sonrasında program metninde bulunan karakterler token olarak adlandırılan simgelere eşleştirilmektedir. Karakterler

sözcükler halinde toplanmakta ve her kelimenin kaynak dilbilgisinde geçerli olup olmadığı kontrol edilmektedir [48].

Sözcüksel analiz aşamasının temel amacı bir sonraki aşama olan sözdizimsel analiz aşamasında işleri kolaylaştırmaktır. Bu aşamada karakterlerin gereksiz kısımlarının elenmesi, kodların simgeleştirilmesi ve sözdizimsel analizin daha hızlı yapılması sağlanmaktadır. Basit sistemler sözcüksel ve sözdizimsel analiz aşamalarını birleştirse de ayrık olmasının verimlilik ve modülerlik gibi avantajları bulunmaktadır [49].

1.6.2. Sözdizimsel Analiz

Derleyicinin ikinci aşamasında tokenlerin sözdizimsel analizi yapılmaktadır. Bu işlem adımına ayrıştırma (parsing) aşaması da denilmektedir. Sözcüksel analizden elde edilen tokenler, sözdizimi ağacı olarak adlandırılan ağaç yapısına dönüştürülmektedir [45]. Sözdizimi ağacının düğümlerini tokenler oluşturmaktadır. Ağaç yapısı oluşturulurken kaynak programlama dilinin dilbilgisi kurallarına göre sözdizimi denetimi de yapılmaktadır. Denetim yapılırken bulunan hata, sözdizimi hatası olarak bilgi mesajı şeklinde rapor edilmektedir [49].

1.6.3. İçerik Yönetimi

Derleyicinin üçüncü aşamasında sözdizimi ağacının anlamsal (semantik) analizi yapılmaktadır. Analiz yapmak için kaynak programlama dilinin sembol tablosu kullanılmaktadır. Ayrıca tip bilgisi toplanmakta ve ara kod oluşturma sırasında kullanılmak üzere bu bilgi, sözdizimi ağacında veya sembol tablosunda saklanmaktadır. İçerik yönetimi aşamasında tip denetimi yapılmaktadır. Derleyici uygun olmayan kullanımları, ilgili hata mesajıyla bildirmektedir [45].

Derleyicinin giriş programında kodlanmış ayrıntılı hesaplama işlemleri hakkında geniş bir bilgi edinmesi gerekmektedir. Hangi değerlerin nasıl temsil edildiğini, nerede bulunduklarını ve değişkenler arasında nasıl geçiş yaptığını bilmektedir. Programın harici dosya ve cihazlar ile nasıl etkileşimde bulunduğunu analiz etmektedir. Tüm bu bilgileri bağlamsal bilgiyi kullanarak kaynak koddan üretebilmektedir. Böylece derleyici; tarayıcı ve

ayrıştırıcı için tipik olanın dışında daha derin bir analiz yapmaktadır. Bu aşama anlamsal analiz olarak da adlandırılmaktadır [48].

1.6.4. Ara Kod Üretimi ve Optimizasyonu

Bir kaynak programın hedef koda çevrilme sürecinde bir derleyici çeşitli formlara sahip olabilen bir veya daha fazla ara kod oluşturabilmektedir. Sözdizimsel analiz aşamasında oluşturulan sözdizimi ağacı da ara kod (ara temsil) biçimi olarak kullanılmaktadır. Derleyiciler kaynak programın sözdizimsel ve anlamsal analizinden sonra soyut bir makine için program olarak nitelendirilen ara kodlar oluşturmaktadır. Bu ara kodun önemli özellikleri şunlardır [45]:

- Ara kodun üretilmesi kolay olmalıdır.
- Ara kodun hedef makineye dönüştürülmesi kolay olmalıdır.

Derleyicilerin yüksek seviyeli programlama dilini doğrudan hedef makine koduna dönüştürmesi gerekmemektedir. Çoğu derleyici yüksek seviyeli programlama dili ile düşük düzeyli makine dili arasında orta düzey bir dil kullanmaktadır. Bu orta düzeyli dile ara kod veya ara temsil denilmektedir [49].

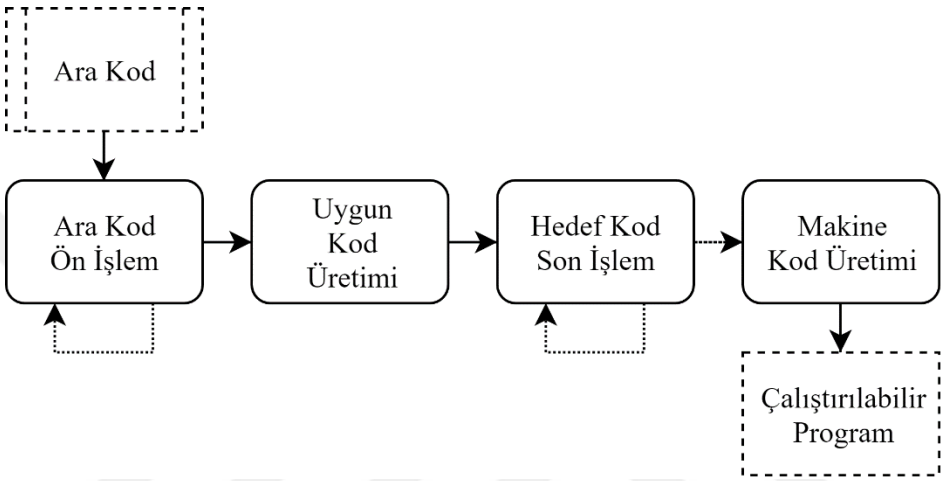
Ara kod optimizasyon aşamasında, daha iyi bir hedef kodun ortaya çıkması için ara kod iyileştirilmeye çalışılmaktadır. Genellikle hedef hız performansı olsa da kısa kod veya daha az güç tüketimi gibi hedefler de olabilmektedir. Basit bir ara kod oluşturma algoritması ve ardından bu kodun optimizasyonu, iyi bir hedef kodun oluşturulması için en uygun yol olarak düşünülmektedir. Farklı derleyiciler için ara kod optimizasyonu süreleri arasında büyük farklar oluşabilmektedir. Derleyicileri optimize etmek için bu aşamada önemli miktarda zaman harcanmaktadır. Optimizasyon aşamasında en temel amaç derleme işlemini yavaşlatmadan hedef programın çalışma süresini önemli ölçüde azaltmaktır. [45].

1.6.5. Hedef Kod Üretimi ve Optimizasyonu

Derleme sürecinde optimize edilmiş ara kod kullanılarak kod üretim aşaması gerçekleştirilmektedir. Kod üretim aşaması ara kodun çalıştırılabilir programa dönüştürülmesiyle tamamlanmaktadır (Şekil 1.8). Kod üretimi, sözdizimi ağacının anlamsal

bütünlüğü bozulmadan hedef kodlara uygun sözdizimi ile eşlenmesi olarak tanımlanmaktadır. Bu eşleşme işlemine ağaç yeniden yazma denilmektedir [46].

Üretilen kodun; tam olarak aynı anlamda, yüksek hızlı, küçük boyutlu ve düşük enerji tüketimli olması istenilmektedir. Uygun derleme tekniklerinin kullanılmasıyla doğruluk ve yüksek hız elde edilmektedir. Az hafıza kullanımı ve düşük enerji tüketimi bir dereceye kadar optimizasyon teknikleriyle sağlanabilmektedir.



Şekil 1.8. Kod üretimi işleyiş diyagramı

1.6.6. Makine Kod Üretimi ve Çalıştırılabilir Program

Derleyicinin son aşamasında; kaynak program, makine koduna eşlenmekte ve çalıştırılabilir program olarak çıkartılmaktadır. Program tarafından kullanılan değişkenlerin her biri için yazmaç ve hafıza alanları belirlenmektedir. Ara talimatlar, aynı görevi yerine getiren makine talimat dizisine çevrilmektedir. Makine kod üretiminin önemli bir parçası değişkenlerin tutulacağı alanın uygun şekilde atanması olarak kabul görmektedir [45].

Şekil 1.9'da R1 ve R2 yazmaçları kullanılarak ara kod (a) makine koduna (b) dönüştürülmektedir. Makine kodunda a değişkeninin değeri hesaplanırken R2 yazmacına b değeri kayıt edilmekte ve 10 ile çarpılmaktadır. Yine c değişkeninin değeri hesaplanırken R1 yazmacına d değeri kayıt edilmekte ve önceden hesaplanan a değişkeninin değeri (R2) ile toplanmaktadır. Makine kodunda bulunan "#" karakteri ara koddaki sabit değeri temsil etmektedir. Sonuç olarak R1 yazmacında elde edilen değer c değişkenine atanmakta ve hesaplama işlemi tamamlanmış olmaktadır.

a = b * 10
c = d + a
 	
(a) Ara Kod
	LD R2, b
	MUL R2, R2, #10
	LD R1, d
	ADD R1, R1, R2
	ST c, R1
	
	(b) Makine Kodu

Şekil 1.9. Ara kodun makine koduna çevrilmesi

1.7. Düzenli İfadeler

Düzenli ifadeler; belirli bir karakter kümesinin, bir metin içerisinden bulunmasını sağlamaktadır. Karakter kümesi bir kalıp olarak tanımlanarak metin içerisinde aranmaktadır [50]. Kalıp oluşturulurken düzenli ifadelerin Tablo 1.1'de verilen karakterleri kullanılmaktadır.

Tablo 1.1. Düzenli ifade karakterleri ve anlamları [51, 52].

Karakter	**Anlam**	**Örnek**	**Eşleşen Metin**
.	Herhangi bir karakter	a..a	aaaa, abca, adea
*	Sıfır veya sınırsız sayıda karakter	ab*a	aa, aba, abba, abbba
+	En az bir kez karakter	ab+a	aba, abba, abbba
?	Sıfır veya bir kez karakter	ab?a	aa, aba
()	İçindeki bütün karakterler	a(bc)?a	aa, abca
[]	İçindeki herhangi bir karakter	a[bc]a	aba, aca
{}	Karakter(ler)in tekrar sayısı	ab{3}a	abbba
^	Metin başındaki karakter(ler)	^a	a, ab, ac, abc
[^]	Bulunmayacak karakter(ler)	[^abc]	d, e, dd, de, def
$	Metin sonundaki karakter(ler)	a$	ba, ca, bca
|	Veya	(a|b)	a, b

Düzenli ifadeler birçok programlama dilinde aynı sözdiziminde kullanılabilmektedir. Düzenli ifadelerde karakterler kullanıldığı gibi karakter setlerini ifade eden komutlar (Tablo 1.2) da kullanılmaktadır.

Tablo 1.2. Düzenli ifade komutları ve anlamları [53, 54].

Komut	**Komut Değeri**	**Anlam**
\d	[0-9]	Herhangi bir rakam
\D	[^0-9]	Rakam haricindeki karakterler
\w	[a-zA-Z0-9_]	Herhangi harf, rakam veya altçizgi (alfa numerik karakterler)
\W	[^a-zA-Z0-9_]	Alfa numerik haricindeki karakterler
\s	[\t\n\r\f]	Herhangi bir boşluk karakteri
\S	[^\t\n\r\f]	Boşluk haricindeki karakterler

Düzenli ifadelerin karakter ve komutlarının yanında düzenleyici ifadeleri de kullanılmaktadır. Bu düzenleyiciler duyarlılığı belirtmektedir:

- x : Boşlukların dikkate alınmayacağını ifade etmektedir.
- i : Büyük küçük harf duyarlılığını kaldırmaktadır.
- s : Tüm satırları tek bir satır gibi işleme almaktadır.
- m : Çoklu eşleme yapmaya imkan sağlamaktadır.
- g : Bütün eşleşen ifadeler üzerinde işlem yapmayı sağlamaktadır [55, 56].

1.8. Sonlu Özdevinirler

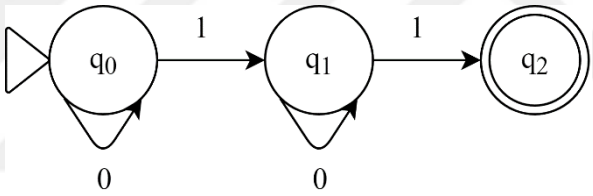
Sonlu özdevinirler, bellek ve hesaplama işlemleri için kullanılan sonlu sayıda duruma sahip olan soyut makineler olarak tanımlanmaktadır. Durumların sonlu sayıda olması, soyut makinelerin fiziksel makinelere dönüştürülmesi fikrini ortaya çıkarmaktadır [57]. Sonlu özdevinir makineler birçok önemli yazılım ve donanım türü için kullanışlı bir model sunmaktadır. Sonlu özdevinirler;

- Sayısal devrelerin davranışlarını kontrol etmek,
- Tipik bir derleyicinin sözcüksel analizi aşamasını, yani girdi metnini tokenlere ayrıştırmak,

- Büyük metinlerde kelime, deyim veya desen aramak,
- Sonlu sayıda duruma sahip her türlü sistemde doğrulama işlemi yapmak için kullanılmaktadır [58].

Sonlu özdevinir makineler belirli ve belirsiz olmak üzere ikiye ayrılmaktadır. Belirsiz sonlu özdevinirler durumlar arası geçişin karışık olduğu, her durum için bir sonraki değerde hangi duruma gideceği belirli olmayan makinelerdir. Belirli sonlu özdevinirler ise bir durumdan başka duruma sadece bir değer ile giden, lamda veya epsilon girdilerini kabul etmeyen makinelerdir [59]. Belirli sonlu özdevinir ($Q, \sum, \delta, q_0, F$) ile tanımlanmaktadır:

1. Q : Sonlu sayıda elemana sahip durumlar kümesi.
2. $\sum$: Sonlu sayıda elemana sahip semboller kümesi.
3. δ : Geçiş fonksiyonu.
4. q_0 : Girdinin işlenmeye başlanacağı başlangıç durumu.
5. F : Q kümesinde tanımlı sonlu durum(lar) kümesi.



Şekil 1.10. Sonlu özdevinir makine durum diyagramı

Sonlu özdevinir, makine durum diyagramı (Şekil 1.10) adı verilen çizge ile gösterilmektedir. Bu çizgede; düğümler durumları, ok işaretleri geçişleri, karakterler girdileri, girdisiz geçiş yapılan düğüm başlangıç durumunu ve iç içe iki çember şeklinde çizilen düğüm son durumu ifade etmektedir.

2. YAPILAN ÇALIŞMALAR, BULGULAR VE TARTIŞMA

Bu bölümde yapılan çalışmanın teknik ve yöntemi, sistem mimarisi, içerik ve arayüzü, geri bildirim sistemi detaylı bir şekilde anlatılmaktadır. Yapılan çalışma anlatılırken elde edilen bulgular ve tartışma aynı bölümde yapılmaktadır.

2.1. Kullanılan Teknolojiler

Geliştirilen programlama öğrenme ortamı bir kurulum gerektirmemesi ve her cihazdan ulaşılabilir olması sebebiyle web uygulaması olarak geliştirilmiştir. Bir web uygulamasının geliştirilmesi ve yayınlanması için bir sunucuya ve sunucu tarafında çalışan bir programlama diline ihtiyaç duyulmaktadır. Yaygın olarak kullanılması ve birçok platformu desteklemesi sebebiyle sunucu olarak Apache tercih edilmiştir. Ortam, animasyonların eklentiye ihtiyaç duymadan çalışabilmesi için JavaScript ve PHP programlama dilinde geliştirilmiştir.

2.1.1. Apache

Apache Software Foundation tarafından geliştirilen ve sürdürülen Apache; en çok kullanılan açık kaynak kodlu bir web sunucu yazılımıdır. Dünyadaki tüm web sunucularının %67'sinde çalışmaktadır. Hızlı, güvenilir ve güvenli olarak kabul edilmektedir. Uzantıları ve modülleri kullanarak birçok farklı ortamın ihtiyaçlarını karşılamak için son derece özelleştirilebilir bir program olmaktadır. [60].

2.1.2. PHP

PHP özellikle web geliştirme için uygun olan ve HTML içine gömülebilen, yaygın olarak kullanılan açık kaynak kodlu genel amaçlı bir betik programlama dilidir. PHP dilini kullanıcı tarafında çalışan dillerden ayıran, sunucu tarafında çalıştırılıyor olmasıdır. Temel olarak sunucu-taraflı programlamaya odaklandığından CGI uygulamalarının yaptığı her şeyi, örneğin formdan veri toplama, dinamik sayfa içeriği oluşturma, ya da çerez alıp gönderme gibi işlemleri yapabilmektedir [61].

PHP bütün büyük işletim sistemlerinde; Linux, birçok Unix türevi (HP-UX, Solaris, OpenBSD vb.), Microsoft Windows, Mac OS X, Risc OS dahil olmak üzere çok çeşitli platformlarda çalışmaktadır. Benzer bir biçimde bugün yaygın biçimde kullanılan HTTP protokolünü kullanan sunucuların büyük bir kısmını desteklemektedir. Bunlara Apache, IIS ve daha birçok sunucu örnek olarak gösterilebilmektedir. Bu sunuculara FastCGI kullanan lighttpd ve nginx gibi sunucular da dâhil olmaktadır. PHP modül olarak kullanılabildiği gibi bir CGI işleyicisi olarak da kullanılabilmektedir [62].

2.1.3. JavaScript

JavaScript, birinci sınıf işlevlere sahip, hafif, yorumlanmış, nesne yönelimli bir programlama dilidir. Web sayfaları için betik bir dil olarak bilinir ancak çoğu tarayıcı dışı ortamda da kullanılmaktadır. Dinamik, prototip tabanlı, çoklu paradigma içeren, nesne yönelimli ve işlevsel programlama stillerini desteklemektedir. JavaScript, web sayfası davranışını kontrol etmek için yaygın olarak kullanılan, öğrenmesi kolay, güçlü ve istemci tarafında çalışan bir betik dildir. Popüler yanlış anlamaların tersine, JavaScript "Yorumlanan Java" değildir [63].

JavaScript sözdizimi, dili öğrenmek için gereken yeni kavramların sayısını azaltmak için hem Java hem de C ++ ile kasten benzer olarak geliştirilmiştir. JavaScript, hem prosedürel hem de nesne yönelimli bir dil olarak işlev görebilmektedir. Nesneler, C ++ ve Java gibi derlenmiş dillerdeki ortak sözdizimsel sınıf tanımlarının tersine, çalışma zamanında başka nesnelere boşaltma yöntemiyle ve özellikleri ekleyerek JavaScript dilinde programatik olarak oluşturulmaktadır. Bir nesne oluşturulduktan sonra benzer nesneler oluşturmak için bir plan (veya prototip) olarak kullanılabilmektedir [63].

2.2. Yöntem

Bu çalışmada uzman sistem sayılabilecek bir programlama öğrenme ortamı geliştirilmiştir. Ortamda, basit sözdizimi olan yeni bir programlama dili tanımlanmıştır. Bu dil derleyici teknikleri ile analiz edilmektedir. Düzenli ifadeler ve sonlu özdevinir makineler kullanılarak doğrulama işlemi yapılmaktadır. Yapılan analiz ve doğrulama işlemleri sonucunda gerekli olan yerlerde geri bildirimler oluşturulup kullanıcıya sunulmaktadır.

Geliştirilen ortamda kullanıcıya görevler verilerek programlama kavramlarının öğretilmesi amaçlanmaktadır.

2.2.1. Oluşturulan Programlama Dili ve Özellikleri

<harf> ::= A | B | C | Ç | D | E | F | G | Ğ | H | I | İ | J | K | L | M | N | O | Ö | P | Q | R | S | Ş | T | U | Ü | V | W | X | Y | Z | a | b | c | ç | d | e | f | g | ğ | h | ı | i | j | k | l | m | n | o | ö | p | q | r | s | ş | t | u | ü | v | w | x | y | z

<rakam> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<mantıksal ifade> ::= doğru | yanlış

<karakter> ::= <harf> | <rakam>

<sayı> ::= <rakam> | <sayı><rakam>

<değişken> ::= <harf> | <değişken><harf> | <değişken><sayı>

<metin> ::= “ <karakter> [<karakter>] ”

<ifade> ::= <metin> | <sayı> | <değişken>

<operatör> ::= <aritmetik operatör> | <mantıksal operatör> | <deyim>

<kod> ::= <boş> | <bir satır kod> | <kod><bir satır kod>

<aritmetik operatör> ::= + | - | * | / | =

<mantıksal operatör> ::= < | ≤ | > | ≥ | == | != | && | ||

<mantıksal ifade> ::= <ifade><mantıksal operatör><ifade>

<deyim> ::= eğer | değilse eğer | değilse | seçenek | tekraret | olduğu sürece

<ayırıcı> ::= , | : | ; | durum | bitir | döndür

<parantez> ::= (|) | { | } | [|]

<eğer> ::= eğer(<mantıksal ifade>) { <kod> }

<değilse eğer> ::= değilse eğer(<mantıksal ifade>) { <kod> }

<değilse> ::= değilse { <kod> }

<seçenek> ::= seçenek(<değişken>) { durum <ifade> : <kod> bitir }

<tekraret> ::= tekraret(<sayı>) { <kod> }

<olduğu sürece> ::= olduğu sürece (<mantıksal ifade>) { <kod> }

Şekil 2.1. Oluşturulan programlama dilinin BNF gösterimi

BNF (Backus Naur Form), programlama dillerinin sözdizimlerini tanımlamak için yaygın olarak kullanılan özyinelemeli yapıda gösterimdir [64, 65]. Oluşturulan programlama diline ait BNF gösterimi Şekil 2.1'de verilmektedir.

Programlama öğrenmek zor ve zaman alıcı bir süreç olarak kabul edilmektedir. Bu zorluğun başında, programlama dillerinin karmaşık yapıda olan sözdizimleri gelmektedir [20, 21]. Bireyler programlama öğrenme sürecini gerçek programlama dilleriyle yaptıkları zaman bu zorlukla yüzleşmek zorunda kalmaktadır. Ancak basit sözdizimi olan bir programlama diliyle hem programlama öğrenilirken hem de algoritma kavramı anlaşılmaktadır. Algoritma kavramının anlaşılması gerçek programlama dilleriyle program geliştirmeyi kolaylaştırmaktadır. Bu sebeple geliştirilen programlama öğrenme ortamında basit sözdizimi olan bir programlama dili oluşturulmuştur.

2.2.2. Programlama Dili ve Token Simgeleri

Tokenler programlama sembol ve karakterlerini belirten simgelerdir. Derleyiciler program metnini sözcüksel analiz aşamasında tokenlere dönüştürmektedir. Bu işlem, anlam ve dilbilgisinin farklı aşamalarda kontrol edilmesi için yapılmaktadır. Oluşturulan programlama dili ve token karşılığı Tablo 2.1'de gösterilmektedir.

Tablo 2.1. Token tablosu

Kaynak Kod	Token	Kaynak Kod	Token
+	ADD	-	SUB
*	MUL	/	DIV
=	EQ	<	LW
≤	LE	>	GR
≥	GE	==	EE
!=	NE	||	OR
&&	AND	,	CM
:	TD	;	END
(	LBT	)	RBT
{	LCB	}	RCB

Tablo 2.1. Token tablosu devamı

Kaynak Kod	Token	Kaynak Kod	Token
[	LBB	]	RBB
“	DQ	değişken	ID
metin	STR	sayı	INT
ifade	EXP	aritmetik operatör	AO
mantıksal operatör	LO	mantıksal ifade	LXP
durum	ST	bitir	BR
döndür	RT	kod	CD
eğer	IF	değilse eğer	ELIF
değilse	ELSE	seçenek	SW
tekraret	FOR	olduğu sürece	WHL
boş	EMP	uygunsuz karakter	UE

Tokenler derleyicinin sözcüksel analiz aşamasında oluşturulmakta ve sözdizimsel analiz aşamasına girdi olarak gönderilmektedir. Her programlama dili sembolü ve karakter kümesi için farklı token adlandırması olması gerekmektedir.

2.3. Programlama Dilinin Analizi

Geliştirilen programlama öğrenme ortamında dili derlemekten ziyada analiz edip kullanıcıyı yönlendirme amacı bulunmaktadır. Kullanıcının yazdığı kod sözcüksel ve sözdizimsel analiz aşamaları ile dilbilgisi kurallarınca kontrol edilmektedir. Yazılan kod dilbilgisi açısından doğrulandıktan sonra verilen görevi yerine getirmesi bakımından incelenmektedir. Yapılan analizler sonucu elde edilen geri bildirimler kullanıcıya verilmektedir. Geri bildirimler “hatalı” veya “başarısız” gibi bilgi içermeyen mesajlar yerine özelleştirilmiş, kullanıcıyı yönlendiren bilgi mesajlardan oluşmaktadır.

2.3.1. Sözcüksel Analiz

Sözcüksel analiz aşamasında program metni tokenlere eşlenerek sözdizimsel analiz aşamasına gönderilmektedir. Bu aşamada ilk olarak program metnindeki beyaz boşluk karakterleri silinmekte, sonrasında program metni tokenlere dönüştürülmektedir. Sözcüksel analiz aşaması algoritması aşağıdaki gibidir:

1. Program metni kaynak dosyadan okunur.
2. Program metnindeki beyaz boşluk karakterleri silinir.
3. Program metnindeki karakterler token dizisine dönüştürülür.

Algoritmada beyaz boşluk karakterlerinin silinmesinde düzenli ifadeler kullanılmıştır. Token eşleme işlemi kaynak kodun karakter akışına göre gerçekleştirilmektedir. Bir karakter bir token olabileceği gibi birden fazla karakter de bir token olabilmektedir (Tablo 2.1).

```
programMetni = oku(kaynakDosya)
programMetni = programMetni.değiştir(/\s/g, “”)
tokenDizisi = tara(programMetni)
```

Şekil 2.2. Sözcüksel analiz aşaması sözde kodu

Sözcüksel analiz aşaması sözde kodunda (Şekil 2.2) değiştir() fonksiyonu düzenli ifadeleri kullanarak program metnindeki beyaz boşluk karakterlerini silmektedir.

```
fonksiyon tara(programMetni) {
    kaynakKodParçası = parçaBul(programMetni)
    programMetni = programMetni.değiştir(/kaynakKodParçası/, “”)
    return parçaTokenAdı + “ ” + tara(programMetni)
}
```

Şekil 2.3. Sözcüksel analiz aşaması tara fonksiyonu sözde kodu

Tara() fonksiyonu özyinelemeli olarak program metnini token dizisine dönüştürmektedir. Kaynak kod parçası oluşturulan programlama dilinde kullanılabilecek her türlü sembol ve karakter kümesini ifade etmektedir. Bu kod parçasını program metninde parçaBul() fonksiyonu bulmaktadır. Önce bulunan kod parçası program metninden çıkartılmakta, sonra bu parçayı ifade eden token, diziye eklenmektedir.

Sözcüksel analiz aşaması örnek bir hesaplama işlemi üzerinden şu şekilde gerçekleştirilmektedir:

$$a = 10 * (5 + 20) \quad (1)$$

Hesaplama işleminin (1) oluşturulan programlama dilindeki karşılığı aşağıdaki verilmektedir:

- sayı = 10 * (5 + 20);

Yukarıda verilen tek satır kodun token dizisi Şekil 2.4'de gösterilmektedir.

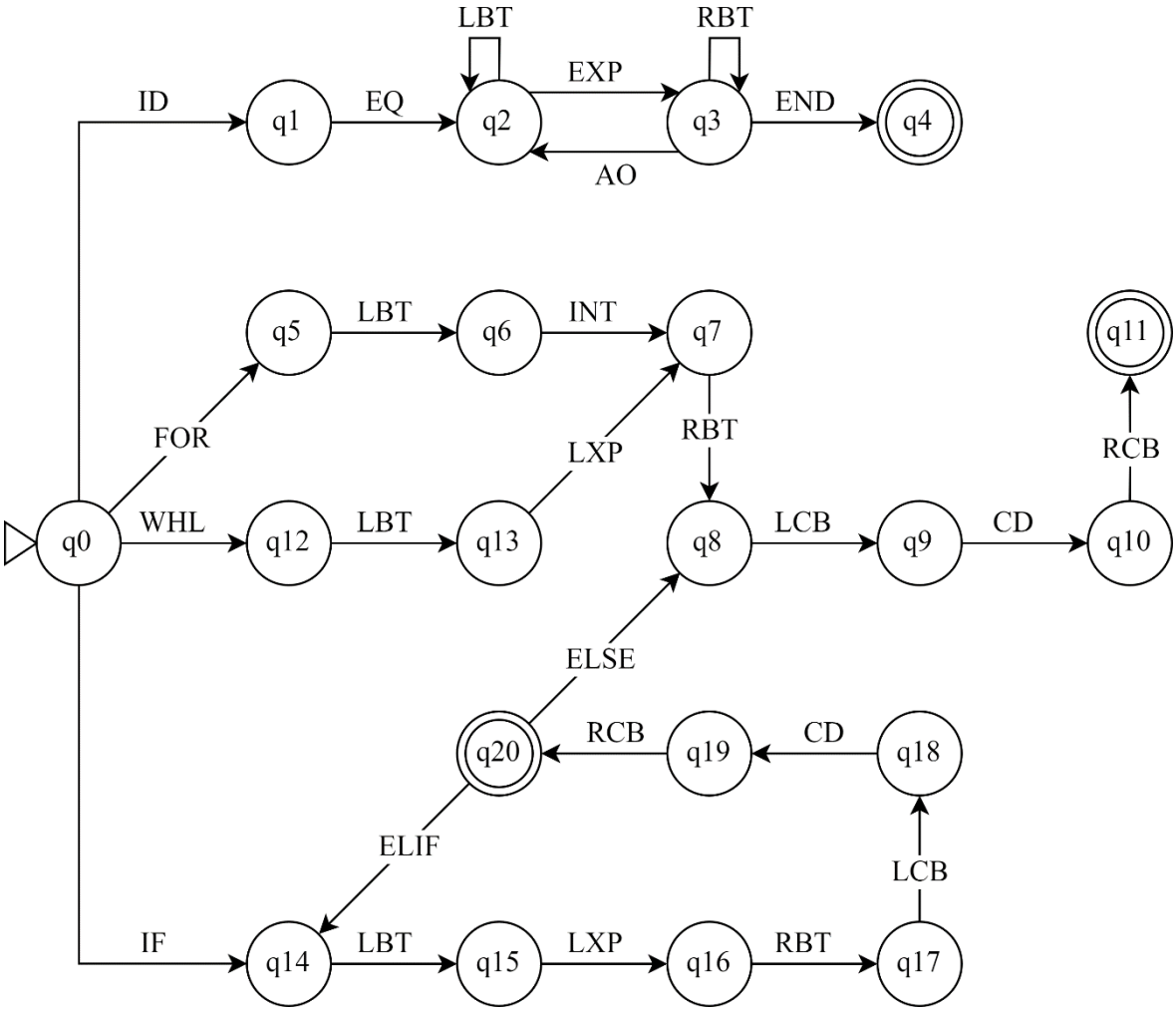
ID	**EQ**	**INT**	**MUL**	**LBT**	**INT**	**ADD**	**INT**	**RBT**	**END**
sayı	=	10	*	(	5	+	20	)	;

Şekil 2.4. Denklem 1'e ait token dizisi

Sözcüksel analiz aşamasının temel amacı sözdizimsel analiz aşamasında işleri kolaylaştırmaktır. Bu iki aşama basit derleyicilerde birleştirilse de ayrı olarak gerçekleştirilmesinin verimlilik ve modülerlik gibi avantajları bulunmaktadır [66].

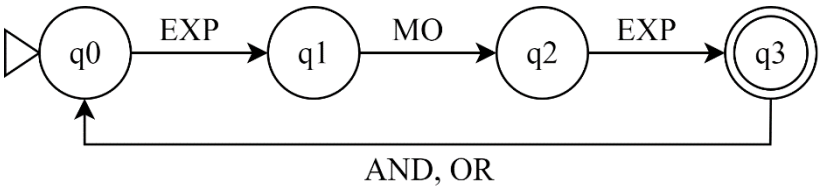
2.3.2. Sözdizimsel Analiz

Sözdizimsel analiz aşamasında token dizisi dilbilgisi açısından denetlenmektedir. Kodda sözdizimi hatası varsa ilgili hata mesajıyla kullanıcıya bildirilmektedir.



Şekil 2.5. Sözdizimsel analizi gerçekleştiren özdevinirin durum diyagramı

Sözdizimsel analiz aşaması, özdevinir makine (Şekil 2.5) ile gerçekleştirilmektedir. Sözcüksel analiz aşamasından gelen token dizisi, programlama dili sözdizimi kurallarına göre kontrol edilmektedir. Tespit edilen sözdizimi hataları yönlendirici geri bildirimlerle kullanıcıya gösterilmektedir.



Şekil 2.6. Mantıksal gerçekleştiren özdevinirin durum diyagramı

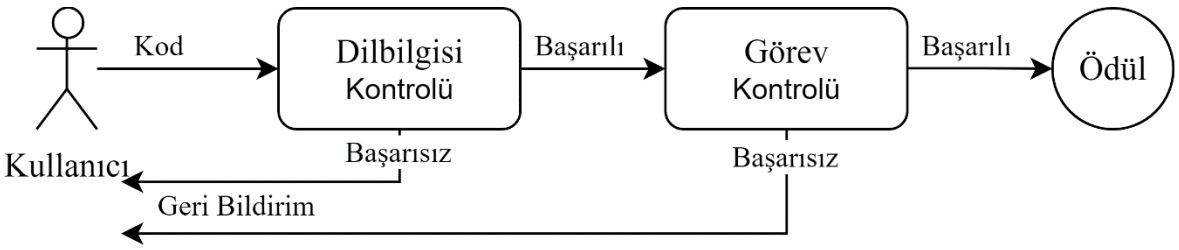
Mantıksal ifade (LXP) doğru (true) ya da yanlış (false) sonucunu oluşturan mantıksal hesaplama işlemini belirtmektedir. Bu mantıksal işlemi doğrulayan özdevinirin durum diyagramı Şekil 2.6'da gösterilmektedir. Şekil 2.5 ve 2.6'da gösterilen genel token ifadelerinin karşılıkları şu şekildedir:

- EXP → ID, STR, INT
- AO → ADD, SUB, MUL, DIV
- MO → LW, LE, GR, GE, EE, NE

Sözdizimsel analiz aşaması program metninin dilbilgisi denetimini yapmaktadır. Eğer bir program özdevinir makine tarafından doğrulanmaz ise o program metninde sözdizimi hatası bulunmaktadır. Sözdizimi hatası tespit edildiğinde geri bildirim sistemiyle kullanıcıya bildirilmektedir. Mesaj içeriği sözdizimi hatasını ve bu hatanın çözümüne yönelik ipucu barındırmaktadır. Bu şekilde kullanıcının programlama öğrenimine katkı sağlanması hedeflenmektedir.

2.3.3. Görev Sistemi

Geliştirilen ortamda programlama kavramları oluşturulan görevlerle öğretilmektedir. Görev ve animasyonlar sayesinde kullanıcı, oyun oynar gibi programlama öğrenme imkânı bulmaktadır. Kullanıcı önceden oluşturulan senaryolar ile verilen görevi yerine getirmeye çalışmaktadır. Programlama kavramlarını sırasıyla takip eden görevler sürekli zorlaşarak ilerlemektedir.



Şekil 2.7. Görev sistemi genel yapısı

Kullanıcı verilen bir görevi yerine getirmek için öncelikle doğru sözdiziminde kod yazması gerekmektedir. İlk olarak kullanıcının yazdığı kod sözdizimi özdevinir makinesi tarafından denetlenmektedir. Bir hata bulunursa kullanıcıya geri bildirim olarak

gösterilmektedir. Sözdizimi başarılı bir şekilde doğrulandığında anlamsal olarak görev başarısı denetlenmektedir. Kullanıcının yazdığı kod iki aşamada da doğrulanırsa görev başarılı kabul edilmekte ve kullanıcı ödüllendirilmektedir. Görev sisteminin bahsedilen yapısı Şekil 2.7'de gösterilmektedir. Görev sistemi örnek bir senaryo ile şu şekilde gerçekleştirilmektedir:

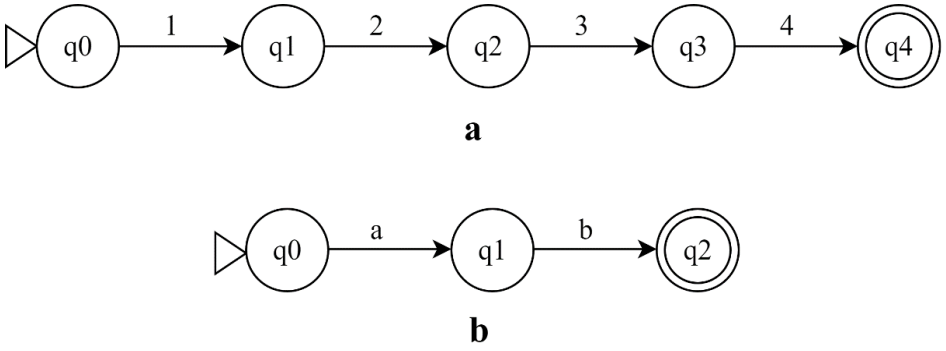
- Senaryo 1: Bahçede bulunan üç kediye farklı isimler verip bunları bir dizide saklayabilir misin?

Verilen görevi yerine getirmek için bir kaç farklı yol izlenebilmektedir. Bu sebeple bütün doğru sonuçları kabul eden bir yapı oluşturulması gerekmektedir. Hatta kullanıcı optimum olmasa bile daha uzun bir çözüm geliştirebilmektedir. Kullanıcının bu görev çözümüne yönelik oluşturduğu kod anlam ve dilbilgisi açısından kontrol edilmektedir. Kullanıcı görevi başarılı bir şekilde gerçekleştirmesiyle ödül alırken, başarısızlık durumunda yaptığı hatalar geri bildirim olarak kullanıcıya sunulmaktadır. Senaryo 1 ile verilen görevin en uygun çözümü Şekil 2.8'de gösterilmektedir.

```
kediSayısı = 3;
[] kediAdları = kediSayısı;
sayaç = 1;
tekraret(kediSayısı) {
        kediAdları[sayaç] = "Kedi " + sayaç;
        sayaç = sayaç + 1;
}
```

Şekil 2.8. Senaryo 1'de verilen görevin en uygun çözümü

Kullanıcı serbest bir kod yazma alanına sahip olduğundan bu görevi farklı değişken isimleriyle yerine getirebilmektedir. Oluşturduğu kodun yazımı farklı olsa bile aynı sonucu elde edebilmektedir. Bu yüzden her görevi karşılayan bir yapıyla kullanıcı kodu değerlendirilmektedir. Şekil 2.8'de çözümü verilen görevin genel çözüm yapısını doğrulayan sonlu durum makinesi Şekil 2.9'da gösterilmektedir.



Şekil 2.9. Senaryo 1'de verilen görevi doğrulayan sonlu durum makinesi

Sonlu durum makinesinde genel çözüm yapısı (Şekil 2.9.a) ve döngü bloğu (Şekil 2.9.b) ayrı ayrı doğrulanmaktadır. Durum geçişlerini ifade eden değerler aşağıda verilmektedir:

1. ID EQ **3** END
2. LBB RBB ID[1] EQ **3** END
3. ID[2] EQ **1** END
4. FOR LBT **3** RBT LCB CD RCB
5. CD:
    a. ID[1] LBB INT RBB EQ STR END
    b. ID[2] EQ ID[2] ADD **1** END

Aynı numaralı tokenlerin (ID) aynı isimli değişken olması gerekmektedir. Ayrıca atama işleminde metinin her defasında farklı değeri alması gerektiği de kontrol edilmektedir.

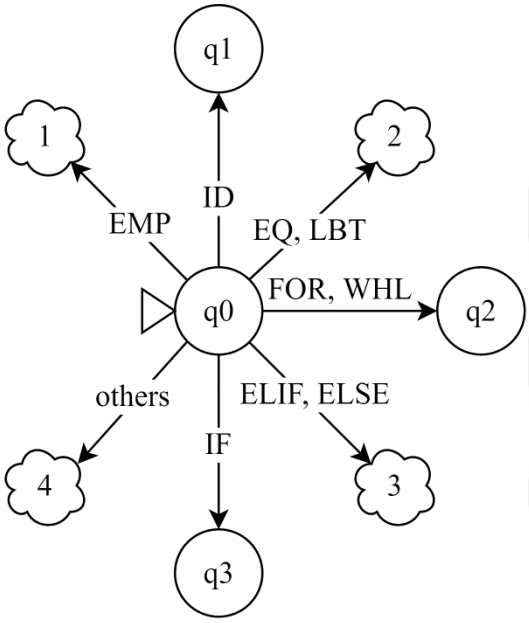
2.3.4. Geri Bildirim Sistemi

Gerçek programlama dillerinin gösterimi bilgisayarın çalışma mantığına benzemektedir. Programlamaya yeni başlayan bireyler bilgisayarca düşünme becerisine sahip olmadığından sözdizimini karmaşık bulmaktadırlar [24]. Program geliştirirken bu zorluklarla yüzleşen bireyler çözüm üretme noktasında derleyicilerin sağladığı kısıtlı bilgiye sahip olmaktadırlar. Derleyiciler programlama geliştirme sürecinde kullanıcıya yalnızca yaptığı sözdizimi hatalarını göstermektedir. Ancak yeni programlama öğrenen bireyler için bu bilgi yeterli düzeyde olmamaktadır.

Programlamaya yeni başlayan bireyler başarılı ve etkin bir öğrenme süreci için yazdıkları kod hakkında detaylı bilgi mesajlarına ihtiyaç duymaktadırlar. Geliştirilen

programlama öğrenme ortamında hem sözdizimi hem de görev başarısı açısından kullanıcıya yeterli düzeyde geri bildirimler sağlanmaktadır. Bu geri bildirimler yazılan kodun, derleyici tasarım teknikleriyle analizi sonucunda oluşturulmaktadır.

Sözdizimine yönelik yapılan geri bildirim sistemi iki özdevinir makineden oluşmaktadır. Bu özdevinirler; operatör işlemleri ve deyimler (koşul ve döngü) için ayrık bir şekilde geliştirilmiştir. Koşul ve döngü için önerilen özdevinir makine operatör özdevinir makinesini kullanarak blok içerisini doğrulamaktadır.



Şekil 2.10. Geri bildirim sisteminin genel yapısı

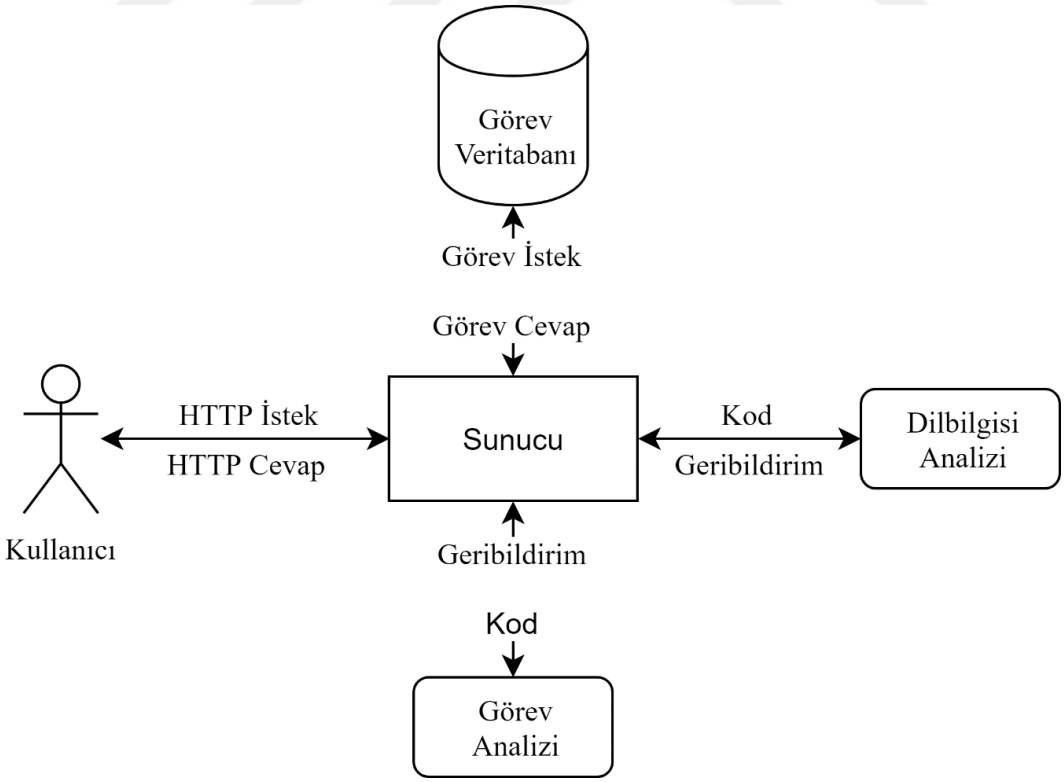
Şekil 2.10'da geri bildirim sistemi genel yapısı verilmektedir. Geri bildirim sisteminde ilk olarak sözdizimi sonrasında görev başarı durumu kontrol edilmektedir. Sözdizimi kontrolü Şekil 2.5'de verilen özdevinir makine ile gerçekleştirilmektedir. Durum geçişleriyle son duruma gidilmez ise ilgili geri bildirim mesajı oluşturulmaktadır. Özdevinir makineler sayesinde kullanıcının tam olarak nasıl bir hata yaptığı keşfedilmektedir. Sözdizimi kontrolünden sonra görevin başarı durumu her göreve özgü oluşturulan token çözüm yapılarıyla kontrol edilmektedir. İki kontrol aşamasında da hatalar ya da eksiklikler geri bildirimler ile kullanıcıya gösterilmektedir. Şekil 2.10'da gösterilen geri bildirimlerin değerleri aşağıda verilmektedir:

1. Görevi yapabilmek için kod yazmalısın.
2. Değişken ya da komut adını unutmuş olmalısın.
3. Değilse komutlarını kullanabilmek için önce eğer komutunu kullanmalısın.
4. Kod satırına yanlış bir şekilde başlamış olmalısın.

Kullanıcı kod satırına hatalı bir şekilde başladığı zaman Şekil 2.10'da gösterildiği gibi ilgili geri bildirimler kullanıcıya verilmektedir. Doğru bir şekilde başladığında ise Şekil 2.5'de gösterilen özdevinir makine kullanılarak doğrulama işlemi devam etmektedir. Tespit edilen hatalar Şekil 2.10'da gösterilen geri bildirimler gibi oluşturularak kullanıcıya sunulmaktadır.

2.4. Geliştirilen Programlama Öğrenme Ortamı

Programlama öğrenme ortamı istemci-sunucu modelinde bir web uygulaması olarak geliştirilmiştir. Ortama kullanıcı HTTP protokolü ile erişmektedir. Ortama giren kullanıcı ilk olarak görevi görüntülemektedir. Bu görevi yerine getirebilmek için kod yazıp bu kodu



Şekil 2.11. Geliştirilen uygulamanın sistem mimarisi

ortama göndermektedir. Kullanıcı kodu dilbilgisi ve görev analizi aşamalarıyla incelenmektedir. Kullanıcı başarısız olduğunda yazdığı koda ilişkin ortamdan geri bildirimler almaktadır. Görevi başarılı bir şekilde yerine getiren kullanıcı bir sonraki göreve geçmektedir. Görevler basitten zora doğru ilerleyen bir sıralamayı takip etmektedir. Kullanıcının bütün görevleri bitirdiğinde programlama kavramlarını öğrenmesi beklenmektedir. Geliştirilen programlama öğrenme ortamının sistem mimarisi Şekil 2.11'de gösterilmektedir.

2.4.1. Ortamın Geliştirilmesi

Programlama öğrenme ortamı bir web uygulaması olarak HTML, CSS, PHP ve JavaScript programlama dilleri kullanılarak geliştirilmiştir. HTML içeriği etiketleyerek tarayıcıda görünmesini, CSS ise görüntülenen içeriğin düzenlenmesini sağlamaktadır. PHP dili veritabanı ve sayfa görüntüleme işlemleri için kullanılmaktadır. Uygulamanın büyük bölümü JavaScript programlama diliyle oluşturulmuştur. Animasyonların oynatılması, yazılan kodun sisteme gönderilmesi, yazılan kodun sözdizimi ve görev analizleri gibi işlemler betik bir dil olan JavaScript ile gerçekleştirilmiştir.

Yazılım geliştirme sürecinde çağlayan süreç modeli esas alınmıştır. Bu süreçte gereksinimlerin tanımlanması, sistem ve yazılım tasarımı, kodlama ve test adımları baştan sona en az bir kez uygulanacak şekilde geliştirme yapılmıştır. Bu model genellikle iyi tanımlı ve üretimi çok zaman gerektirmeyen yazılım projeleri için kullanıldığından dolayı tercih edilmiştir.

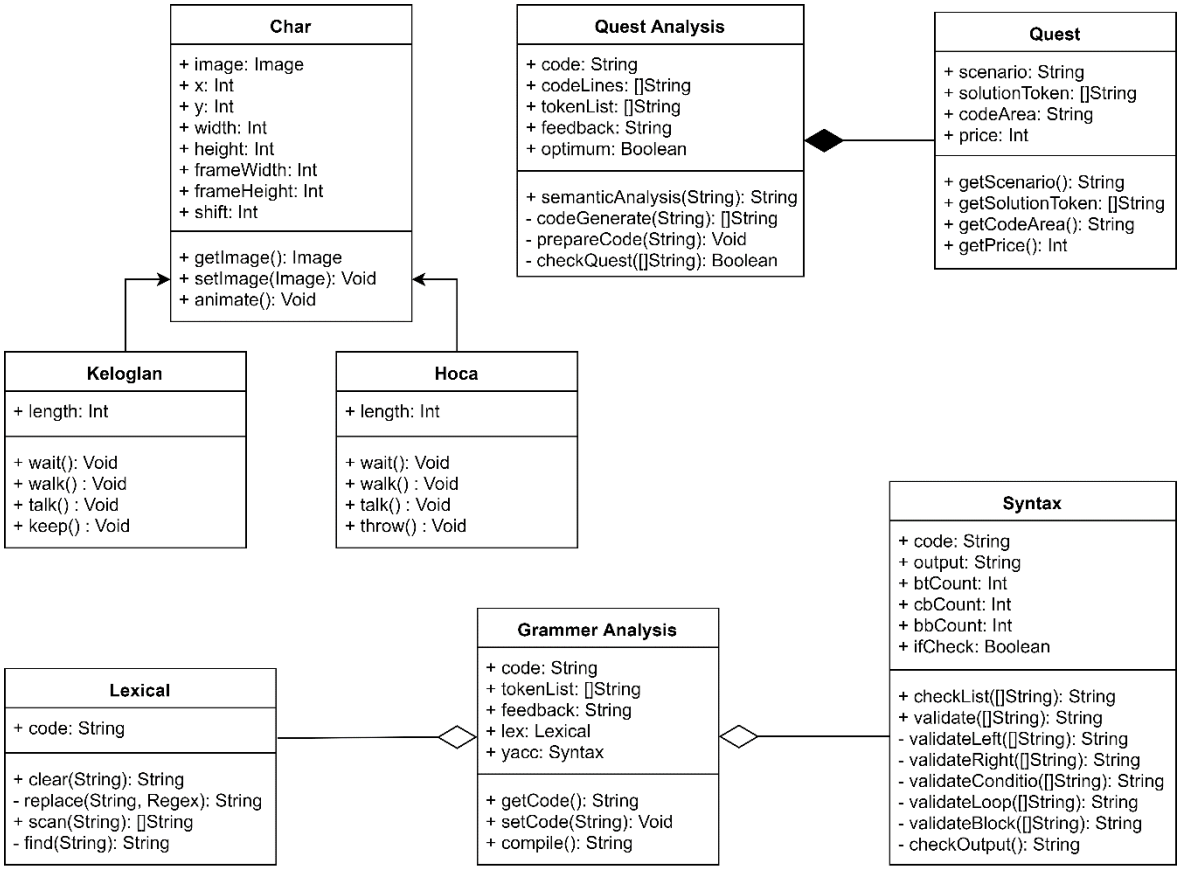
Geliştirilme sürecinde yazılımın tasarımı 4+1 bakışı ile gerçekleştirilmiştir. Bu bakış ile planlama ve tasarım doğru bir şekilde ortaya koyulmakta ve yazılım geliştirme süreci kolaylaşmaktadır. 4+1 bakışında temel bakış açıları şunlardır:

- Tasarım Bakışı (Desing View)
- İşlem Bakışı (Process View)
- Bileşen Bakışı (Component View)
- Dağılım Bakışı (Deployment View)
- Kullanım Durumları Bakışı (Use Case View)

Ortamın yazılım geliştirme sürecinde tasarım yapılırken UML (Unified Modeling Language) kullanılmıştır. UML diyagram çizmek için kullanılan ilişkisel modelleme dilidir.

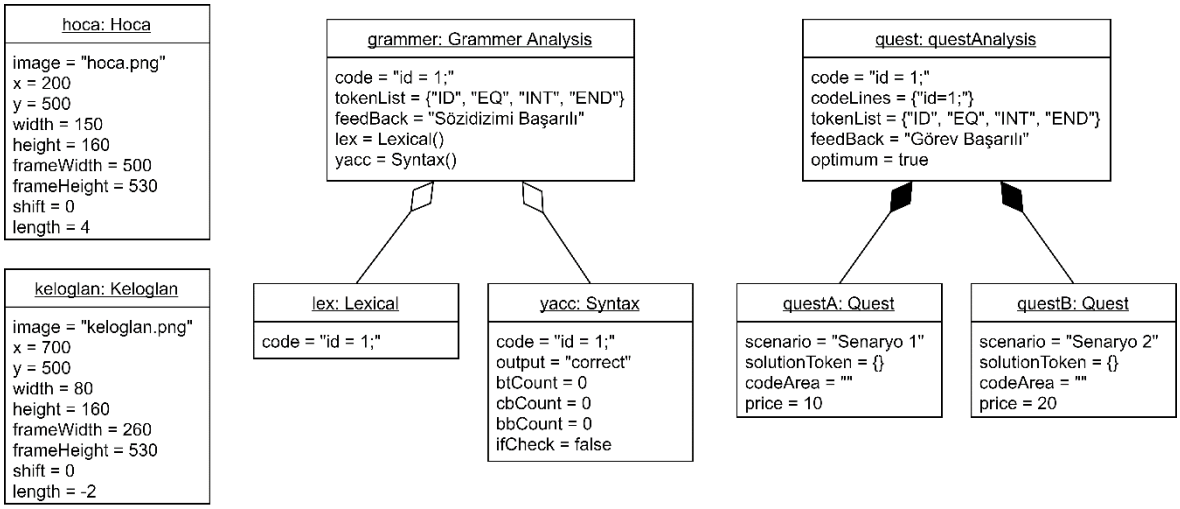
2.4.1.1. Tasarım Bakışı

Geliştirilen ortamın sınıfı diyagramı Şekil 2.12'de gösterilmektedir. Keloglan ve Hoca sınıfları Char sınıfından türetilmektedir. Bahsedilen üç sınıf animasyon ekranında hareket edebilen nesneleri temsil etmektedir. Dilbilgisi denetimi Lexical ve Syntax sınıflarının birleşimiyle oluşan Grammer Analysis sınıfı tarafından yapılmaktadır. Quest sınıfı göreve ait bilgileri barındırırken Quest Analysis sınıfı ise görev kontrolünü gerçekleştirmektedir.



Şekil 2.12. Geliştirilen ortamın sınıf diyagramı

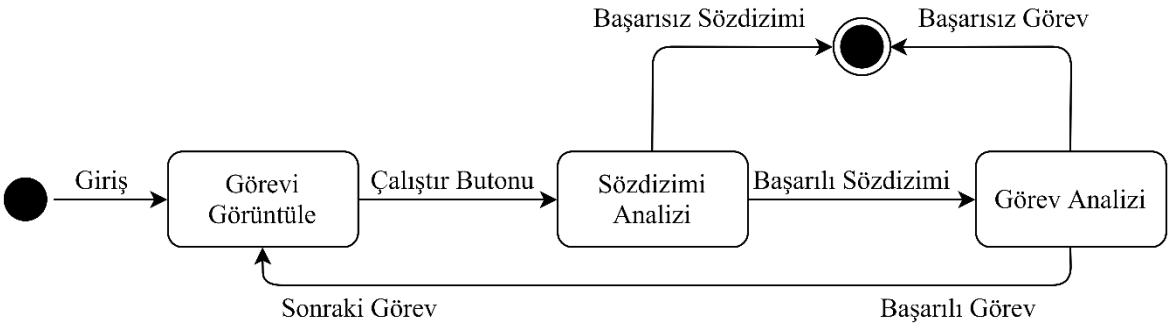
Geliştirilen ortamın nesne diyagramı Şekil 2.13'de gösterilmektedir. Nesne diyagramında sınıfların birbiriyle olan ilişkileri sınıf örnekleriyle birlikte verilmektedir. Keloglan ve Hoca örneği animasyon yapabilmek için gerekli değişkenlere ve değerlere sahip olmaktadır. Grammer örneği lex ve yacc örneklerini kullanarak sözdizimi kontrolünü gerçekleştirmektedir. Quest örneği görevlerin başarı durumunu kontrol etmekte, questA ve questB örnekleri ise görevlerle ilgili bilgileri tutmaktadır.



Şekil 2.13. Geliştirilen ortamın nesne diyagramı

2.4.1.2. İşlem Bakışı

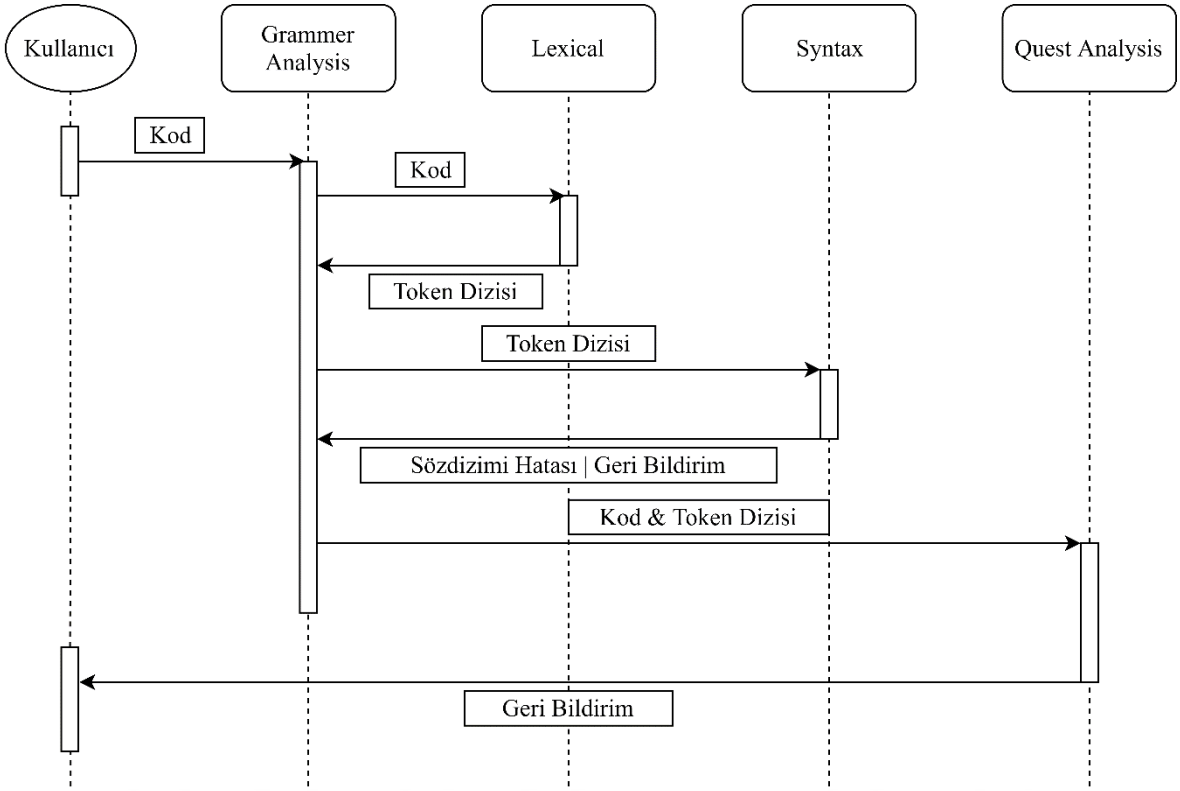
İşlem bakışı sistemin dinamik davranışlarını işlem bakımından incelemektedir. Bu bakışta durum (state), ardışık (sequence), aktivite (activity) ve işbirliği (collaboration) diyagramları bulunmaktadır.



Şekil 2.14. Geliştirilen ortamın durum diyagramı

Geliştirilen ortamın durum diyagramı Şekil 2.14'de verilmektedir. Durum diyagramı ile sistemin davranışı gösterilmektedir. Durum diyagramları nesnelerin durumlarını, geçişlerini ve olayları içermektedir. Kullanıcı sisteme girdiğinde animasyonlarla birlikte görevi görüntülemektedir. Görevin çözümüne yönelik yazdığı kodu çalıştır butonu ile

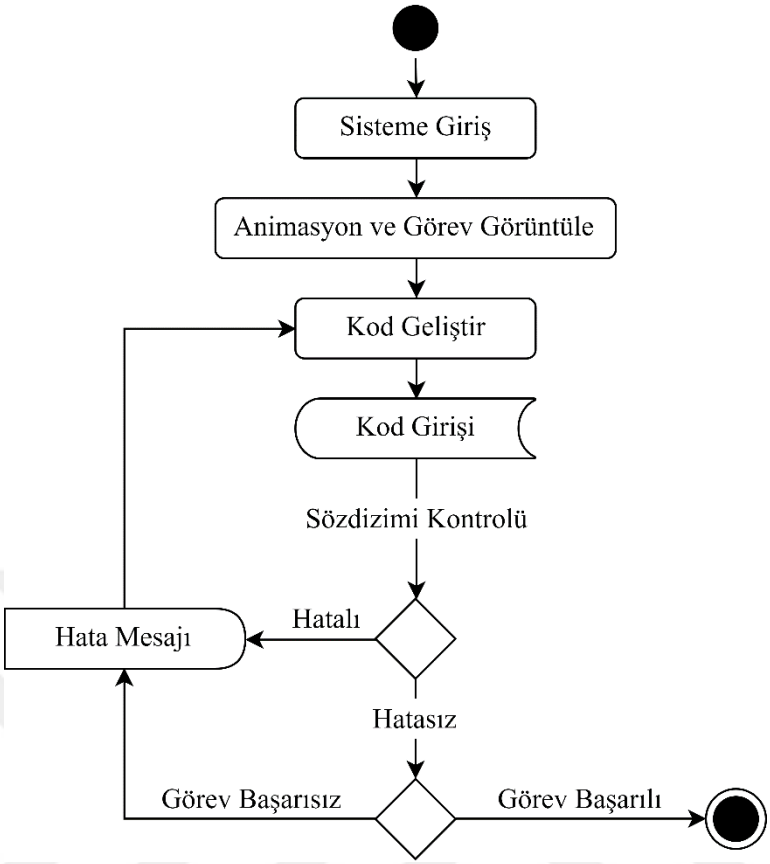
sisteme göndermektedir. Gönderilen kod yapılan analizler sonucunda kullanıcıyı yönlendirmektedir.



Şekil 2.15. Geliştirilen ortamın ardışık diyagramı

Geliştirilen ortamın ardışık diyagramı Şekil 2.15'de gösterilmektedir. Ardışık diyagram ile sistemdeki nesneler arasındaki mesajlaşma olayları zaman akışına göre verilmektedir. Dikey boyut olayların oluşma sırasını zamanına göre, yatay boyut ise mesajların gönderildiği nesne örneklerini göstermektedir. Ardışık diyagram akışı soldan sağa doğru olmaktadır.

Geliştirilen ortamın aktivite diyagramı Şekil 2.16'da gösterilmektedir. Aktivite diyagramı ile sistemin dinamik görünümü modellenmektedir. Aktivite diyagramında işlemler arası akış; ardışık, dallanma ya da eşzamanlı olabilmektedir. Sisteme giren kullanıcı animasyonlar aracılığıyla görevleri görüntülemektedir. Kullanıcı tarafından görevin çözümüne yönelik geliştirilen kod sisteme gönderilmektedir. Sistem, gönderilen kodu kontrol etmekte ve kullanıcıya mesajlar oluşturmaktadır. Görevi başarılı bir şekilde tamamlayan kullanıcı bir sonraki göreve geçerek ilerlemektedir.

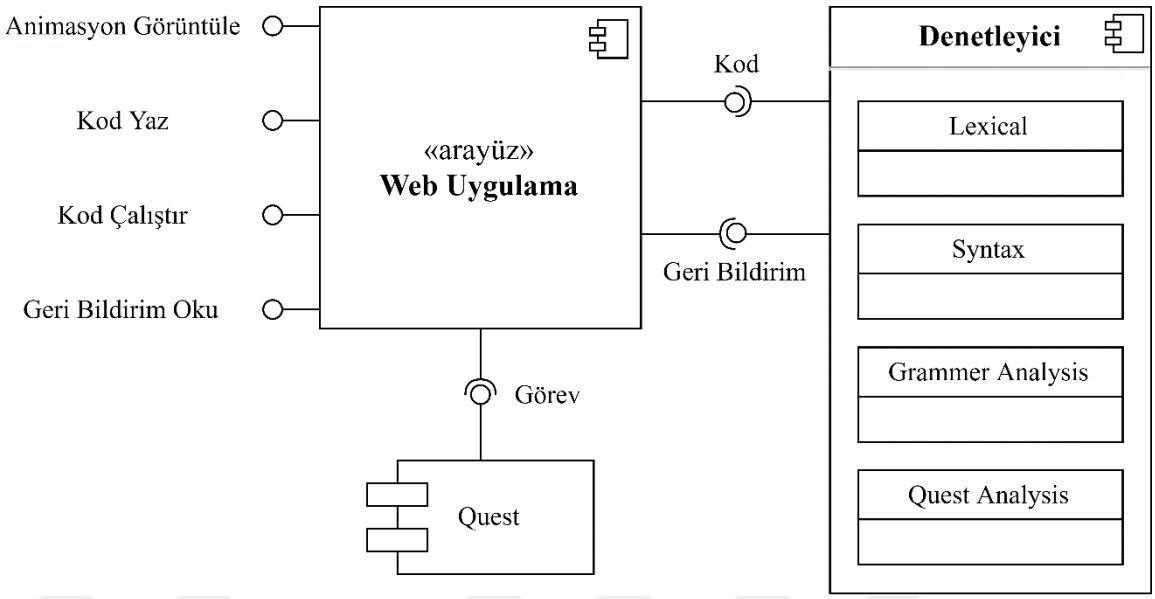


Şekil 2.16. Geliştirilen ortamın aktivite diyagramı

2.4.1.3. Bileşen Bakışı

Bileşen bakışı sistemin yazılım bileşenlerini ve bu bileşenler arasındaki bağlantının nasıl olduğunu göstermektedir. Bileşen diyagramı sistemin uygulanma perspektifini ve sistemdeki tasarım öğelerinin gruplandırılmasını yansıtmaktadır. Bu diyagramlar oluşturulurken bileşen, paket, sınıf gibi öğeler kullanılmaktadır.

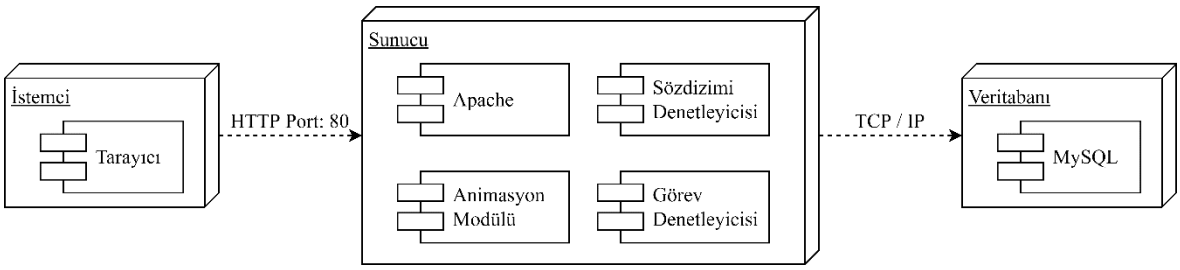
Geliştirilen ortamın bileşen diyagramı Şekil 2.17'de gösterilmektedir. Ortamda kullanılan Lexical, Syntax, Grammer Analysis ve Quest Analysis sınıfları denetleyici bileşeni adı altında gruplanmaktadır. Denetleyici bileşeni kullanıcının yazdığı koda ihtiyaç duymakta ve bu kodu işledikten sonra geri bildirim oluşturmaktadır. Denetleyicinin oluşturduğu geri bildirimler ve Quest sınıfından oluşan görev örnekleri kullanıcıya web uygulama aracılığıyla gösterilmektedir. Kullanıcı web uygulama sayesinde animasyon görüntüleme, kod yazma, kod çalıştırma ve geri bildirim okuma işlemlerini gerçekleştirebilmektedir.



Şekil 2.17. Geliştirilen ortamın bileşen diyagramı

2.4.1.4. Dağılım Bakışı

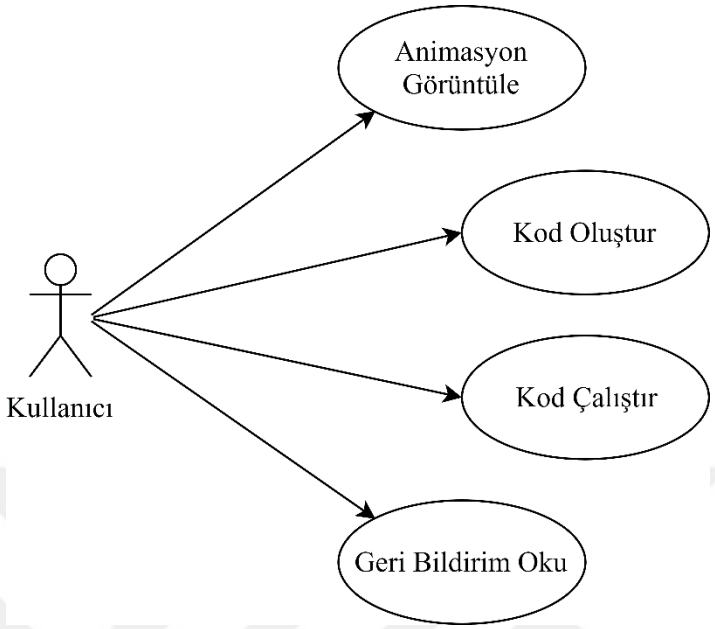
Dağılım bakışı sistemin donanım açısından dağılımını incelemektedir. Bu bakış açısındaki temel amaç bileşenlerin dağılımını göstermek ve performans engellerini teşhis etmektir.



Şekil 2.18. Geliştirilen ortamın dağılım diyagramı

Geliştirilen ortamın dağılım diyagramı Şekil 2.18'de gösterilmektedir. İstemci-sunucu modelinde geliştirilen ortamın bileşenleri arasındaki haberleşme şekilde görülmektedir.

2.4.1.5. Kullanım Durumları Bakışı

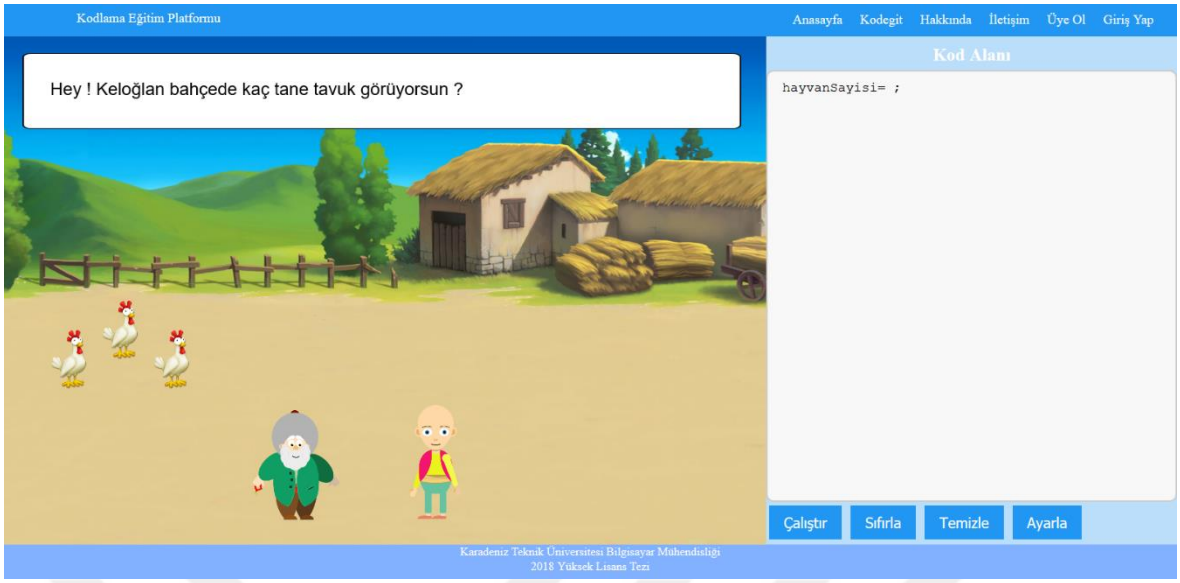


Şekil 2.19. Geliştirilen ortamın kullanım durumları diyagramı

Geliştirilen ortamın kullanım durumları diyagramı Şekil 2.19'da gösterilmektedir. Kullanım durumları veya senaryoları; sistemin işlevselliğini açıklamak amacıyla kullanılan çizimlerdir. Genellikle yazılım projelerinde ilk olarak kullanım durumları diyagramı oluşturulmaktadır.

2.4.2. Ortamın Arayüzü

Geliştirilen öğrenme ortamın arayüzü modeli Şekil 2.20'de gösterilmektedir. Ortam, animasyon ve editör olmak üzere iki ana bölümden oluşmaktadır. Kullanıcı görevleri ve yazdığı kodun sonuçlarını animasyon ekranında interaktif olarak görebilmektedir. İnsan bilgisayar etkileşimini destekleyen bu yapının kullanıcıyı motive etmesi beklenmektedir. Editör bölümünde kullanıcı kod yazıp bunu ortama göndermektedir. Editör butonları ile kod parçaları pratik bir şekilde uygulanmaktadır.



Şekil 2.20. Geliştirilen programlama öğrenme ortamının arayüzü

Şekil 2.20'de görüldüğü gibi Nasreddin Hoca ve Keloğlan karakterleri kullanılarak senaryolar gerçekleştirilmektedir. Nasreddin Hoca, Keloğlan'a hazırlanan senaryolardaki görevleri vermektedir. Kullanıcı yazdığı kod ile Keloğlana bu görevleri yaptırmaya çalışmaktadır. Kullanıcı başarısız olduğunda Nasreddin Hoca bilge kişiliğiyle kullanıcıya geri bildirimleri söylemektedir. Kullanıcının yazdığı kod Keloğlan tarafından uygulanmaktadır.

2.4.3. Ortamın İçeriği

Geliştirilen ortamın temel programlama kavramlarını öğretme hedefi bulunmaktadır. Ortamın kapsadığı programlama kavramları şunlardır:

- Değişken kavramı (8 Senaryo)
- Veri tipleri (8 Senaryo)
- Aritmetik işlemler (4 Senaryo)
- Mantıksal işlemler (4 Senaryo)
- Koşul deyimleri (13 Senaryo)
- Döngü deyimleri (9 Senaryo)
- Diziler (10 Senaryo)

Her bir konu başlığı ve konuya ait senaryo sayısı yukarıda verilmektedir. Her bir senaryo alanında uzman kişiler ile birlikte geliştirilmiştir. Senaryolar oluşturulurken basit görevlerden zor görevlere doğru ilerleyen bir sıralamayı takip etmektedir. Her konu başlığına ait bir adet örnek senaryo sırasıyla aşağıda verilmektedir.

2.4.3.1. Değişken Kavramı

Senaryo: Bahçede üçer tane tavuk, kedi ve köpek bulunmaktadır. Nasreddin Hoca; bahçedeki hayvanlardan hangisinden kaç tanesini yanlarından görmek istediğini sormaktadır.

Amaç: Bu senaryo ile değişken kavramının öğretilmesi planlanmaktadır. Değişkenin bilgi saklayan bir kavram olduğu vurgulanmaktadır.

Kod Alanı: Herhangi yardımcı kod bulunmamaktadır.

Cevap:

i. değişken1 = “isim”;
ii. değişken2 = sayı;

Token Yapısı:

i. ID EQ STR END
ii. ID EQ INT END

Hata Durumları ve Geri Bildirimler:

1. Sözdizimi hatası:
   Geri Bildirim: İlgili sözdizimi hatasını belirten bilgi mesajı.
2. Değişken1’in yazılmaması:
   Geri Bildirim: Hayvan ismini ifade edecek bir değişken kullanmalısın.
3. Hayvan isminin yanlış yazılması:
   Geri Bildirim: Hayvan ismini yanlış yazmış olmalısın.
4. Bahçede bulunmayan bir hayvan isminin yazılması:
   Geri Bildirim: Bahçede bulunan hayvanlardan birisini yazmalısın.
5. Değişken2’nin yazılmaması:
   Geri Bildirim: Hayvan sayısını ifade edecek bir değişken kullanmalısın.
6. Hayvan sayısının yanlış yazılması:
   Geri Bildirim: Hayvanları tekrar saymalısın.
7. Bahçede bulunan hayvan sayısından daha büyük bir sayı yazılması:

Geri Bildirim: Bahçede kaç tane hayvan olduğuna dikkat etmelisin.

8. Hayvan sayısının harflerle yazılması:

   Geri Bildirim: Hayvan sayısını ifade ederken rakam kullanmalısın.

2.4.3.2. Veri Tipleri

Senaryo: Bahçede üç tane tavuk bulunmaktadır. Nasreddin Hoca; bahçedeki hayvanın ismini ve sayısını sormaktadır.

Amaç: Bu senaryo ile veri tiplerinin öğretilmesi planlanmaktadır. Değişken sayı veya metin tipinde olabileceği vurgulanmaktadır.

Kod Alanı:

i. hayvanAdı = “ ”;
ii. hayvanSayısı = ;

Cevap:

i. hayvanAdı = “tavuk”;
ii. hayvanSayısı = 3;

Token Yapısı:

i. ID EQ DQ tavuk DQ END
ii. ID EQ 3 END

Hata Durumları ve Geri Bildirimler:

1. Sözdizimi hatası:

   Geri Bildirim: İlgili sözdizimi hatasını belirten bilgi mesajı.

2. Hayvan ismini yanlış yazması:

   Geri Bildirim: Hayvan ismini yanlış yazmış olmalısın.

3. Tavuktan farklı bir hayvan ismi yazması:

   Geri Bildirim: Bahçede bulunan hayvanın ismini yazmalısın.

4. Hayvan sayısını yanlış yazması:

   Geri Bildirim: Hayvanları tekrar saymalısın.

5. Hayvan sayısını harflerle yazması:

   Geri Bildirim: Hayvan sayısını ifade ederken rakam kullanmalısın.

2.4.3.3. Aritmetik İşlemler

Senaryo: Bahçede 3 tane tavuk ve 2 tane kedi bulunmaktadır. Nasreddin Hoca, toplam hayvan sayısını mevcut değişkenler ile toplamasını ve bir değişkende bilgi olarak saklamasını söylemektedir.

Amaç: Bu senaryo ile aritmetik işleminin öğretilmesi hedeflenmektedir. İki sayıyı değişkenler ile toplaması vurgulanmaktadır.

Kod Alanı:

i. tavukSayısı = 3;
ii. kediSayısı = 2;
iii. hayvansayısı =

Cevap:

i. tavukSayısı = 3;
ii. kediSayısı = 2;
iii. hayvanSayısı = tavukSayısı + kediSayısı;

Token Yapısı:

i. ID^1 EQ 3 END
ii. ID^2 EQ 2 END
iii. ID EQ ID^1 ADD ID^2 END

Hata Durumları ve Geri Bildirimler:

1. Sözdizimi hatası:
   Geri Bildirim: İlgili sözdizimi hatasını belirten bilgi mesajı.
2. Değişkenlerin isimlerinin yanlış yazılması:
   Geri Bildirim: Hayvanların sayılarını ifade eden değişkenlerin ismini yanlış yazmış olmalısın.
3. Toplama işleminde değişken yerine sayı kullanılması:
   Geri Bildirim: Hayvan sayısını toplarken mevcut değişkenleri kullanmalısın.
4. Toplama operatörü yerine farklı bir operatör kullanılması:
   Geri Bildirim: Toplam hayvan sayısını elde etmek için toplama işlemi yapmalısın.

2.4.3.4. Mantıksal İşlemler

Senaryo: Bahçede 7 tane tavuk bulunmaktadır. Nasreddin Hoca tavuk sayısıyla ilgili bazı sorular sormaktadır. Sorulan soruların kod karşılığı kod alanında oluşturulmaktadır. Bu sorulara karşılık doğru veya yanlış cevabını beklenmektedir.

Amaç: Bu senaryo ile mantıksal işleminin öğretilmesi hedeflenmektedir. Sorular sorulduğunda bu işlemin kod karşılığı kod alanında oluşturularak mantıksal operatörlerin öğretilmesi planlanmaktadır.

Sorular:

i. Bahçedeki tavuk sayısı 7'ye eşit midir?
ii. Bahçedeki tavuk sayısı 5'e eşit midir?
iii. Bahçedeki tavuk sayısı 7'den farklı mıdır?
iv. Bahçedeki tavuk sayısı 5'den farklı mıdır?

Kod Alanı ve Cevaplar:

i. tavukSayısı == 7 Doğru
ii. tavukSayısı == 5 Yanlış
iii. tavukSayısı != 7 Yanlış
iv. tavukSayısı != 5 Doğru

Hata Durumları ve Geri Bildirimler:

1. Doğru ve yanlış kararlarını hatalı belirlemesi:
   Geri Bildirim: Hayvanları tekrar saymalısın.

2.4.3.5. Koşul Deyimleri

Senaryo: Kümeste 4 tane yumurta bulunmaktadır. Nasreddin Hoca; kümeste 3'den fazla yumurta varsa yumurtaları toplamasını söylemektedir.

Amaç: Bu senaryo ile eğer koşulunun öğretilmesi hedeflenmektedir. Eğer koşulu doğru olduğunda blok içerisinin çalışacağı vurgulanmaktadır.

Kod Alanı:

i. yumurtaSayısı = 4;

Cevap:

i. yumurtaSayısı = 4;
ii. eğer(yumurtaSayısı > 3) {

iii. topla(“yumurta”);

iv. }

Token Yapısı:

i. ID[1] EQ 4 END

ii. IF LBT ID[1] GR 3 RBT LCB

iii. topla LBT DQ yumurta DQ RBT END

iv. RCB

Hata Durumları ve Geri Bildirimler:

1. Sözdizimi hatası:
   Geri Bildirim: İlgili sözdizimi hatasını belirten bilgi mesajı.
2. Eğer koşulunun yazılmaması:
   Geri Bildirim: Şarta bağlı bir eylem varsa eğer koşulunu kullanmalısın.
3. Eğer koşulunun parantezlerinin yazılmaması:
   Geri Bildirim: Eğer koşulundaki şartı parantez içinde yazmalısın.
4. Eğer koşulunun süslü parantezlerinin yazılmaması:
   Geri Bildirim: Şarta bağlı eylemi süslü parantez içinde yazmalısın.
5. Eğer koşulunun şartında yumurtaSayısı değişkeninden farklı bir ifade kullanılması:
   Geri Bildirim: Koşulu yumurta sayısına göre oluşturmalısın.
6. Eğer koşulunun şartında operatörün yanlış yazılması:
   Geri Bildirim: Karşılaştırma operatörünü doğru kullanmalısın.
7. Eğer koşulunun şartında 3 sayısının yerine farklı bir ifade kullanılması:
   Geri Bildirim: Kümesteki yumurtalar 3’den fazla ise yumurtaları toplamalısın.
8. Topla fonksiyonunun yanlış yazılması:
   Geri Bildirim: Topla komutunu, toplamak istediğin nesneyi parantez içine yazarak kullanmalısın.

2.4.3.6. Döngü Deyimleri

Senaryo: Bahçede 10 tane buğday çuvalı bulunmaktadır. Nasreddin Hoca; buğday çuvallarını ambara taşımasını söylemektedir.

Amaç: Bu senaryo ile döngü deyiminin öğretilmesi hedeflenmektedir. Tekrarla deyimi belirtilen sayı kadar blok içerisini çalıştıracağı vurgulanmaktadır.

Kod Alanı: Herhangi yardımcı kod bulunmamaktadır.

Cevap:

i. tekraret(10) {
ii. taşı(“buğdayÇuvalı”);
iii. }

Token Yapısı:

i. FOR LBT 10 RBT LCB
ii. taşı LBT DQ buğdayÇuvalı DQ RBT END
iii. RCB

Hata Durumları ve Geri Bildirimler:

1. Sözdizimi hatası:
   Geri Bildirim: İlgili sözdizimi hatasını belirten bilgi mesajı.
2. Tekraret deyiminin yazılmaması:
   Geri Bildirim: Aynı işi birçok kez yapmak için tekraret deyimini kullanmalısın.
3. Tekraret deyiminin parantezlerinin yazılmaması:
   Geri Bildirim: Tekraret deyiminin çalışma sayısını parantez içinde yazmalısın.
4. Tekraret deyiminin süslü parantezlerinin yazılmaması:
   Geri Bildirim: Tekrar edecek eylemi süslü parantez içinde yazmalısın.
5. Tekraret deyiminde parantez içinde 10’dan farklı bir ifade kullanılması:
   Geri Bildirim: 10 tane buğday çuvalını taşımalısın.
6. Taşı fonksiyonunun yanlış yazılması:
   Geri Bildirim: Taşı komutunu, taşımak istediğin nesneyi parantez içine yazarak kullanmalısın.

2.4.3.7. Diziler

Senaryo: Bahçede birer tane kedi, köpek ve tavuk bulunmaktadır. Nasreddin Hoca; bu 3 farklı hayvan isimlerinden bir dizi oluşturmasını söylemektedir.

Amaç: Bu senaryo ile dizilerin öğretilmesi hedeflenmektedir. Dizilerin sınırlı bir eleman sayısı ve her bir elemanın farklı değer alabileceği vurgulanmaktadır.

Kod Alanı: Herhangi bir yardımcı kod bulunmamaktadır.

Cevap:

i. [] hayvan = 3;
ii. hayvan[1] = “kedi”;

iii. hayvan[2] = “köpek”;

iv. hayvan[3] = “tavuk”;

Token Yapısı:

i. LBB RBB ID[1] EQ 3 END

ii. ID[1] LBB 1 RBB EQ DQ kedi DQ END

iii. ID[1] LBB 2 RBB EQ DQ köpek DQ END

iv. ID[1] LBB 3 RBB EQ DQ tavuk DQ END

Hata Durumları ve Geri Bildirimler:

1. Sözdizimi hatası:
   Geri Bildirim: İlgili sözdizimi hatasını belirten bilgi mesajı.
2. Dizi tanımlamasında 3 sayısından farklı bir ifade kullanılması:
   Geri Bildirim: Bahçede bulunan hayvan sayısına dikkat etmelisin.
3. Dizi atamasında kedi, köpek ve tavuk kullanılmaması:
   Geri Bildirim: Bahçede bulunan hayvanlara dikkat etmelisin.
4. Dizi atamasında 1, 2 ve 3’den farklı bir sayı kullanılması:
   Geri Bildirim: Dizinin boyutundan büyük bir elemanı olmayacağını unutmuş olmalısın.

Geliştirilen ortamda bilgi içerikli materyal sunmak yerine görev temelli yaklaşım kullanılmaktadır. Her konuyu kapsayan fazla sayıda görev önceden oluşturulup veri tabanında saklanmaktadır. Kullanıcıya bilgi sunmak yerine verilen görevlerden bilgiye ulaşması beklenmektedir. Bu oyunlaştırma tekniği sayesinde kullanıcının sıkılmadan eğlenceli bir ortamda programlama öğrenmesi hedeflenmektedir.

3. SONUÇLAR

- Mevcut programlama öğrenme ortamları genellikle görsel bileşenlerle programlama yapılan platformlardır. Bu tür ortamlarda görsel bileşenlerin fiziksel yapısından dolayı sözdizimi hatası yapmak mümkün olmamaktadır. Bu ortamlarda kullanıcılar programlama mantığını anlamaktan ziyade görsel bileşenleri birleştirebilmeyi hedeflemektedir. Ayrıca görsel bileşenlerle programlama öğrenen bireyler gerçek programlama dillerine geçtiklerinde zorluk yaşamaktadırlar. Bu tür problemleri engellemek için geliştirilen öğrenme ortamında basit sözdizimi olan gerçek programlama dillerine benzeyen yeni bir dil oluşturulmuştur. Bu programlama diliyle kullanıcıların programlama kavramlarını öğrenmesi ve gerçek programlama dillerine kolay adapte olması amaçlanmıştır.
- Mevcut ortamlarda yazılan kodu derlemek için genellikle derleyici geliştiren araçlardan derleyici üretilmektedir. Hazır araçlarla bu tür ortamların geliştirilmesi gerekli analizlerin yapılmasının önüne geçmektedir. Bahsedilen ortamlarda programlama dili üzerinde yeterli analiz yapılmadan, dil derlenmekte ve elde edilen çıktı çalıştırılmaktadır. Bu sebeple sözdizimi hatası olan veya istenilen şartları sağlamayan koda karşılık "hatalı" veya "başarısız" gibi bilgi içermeyen mesajlar oluşturulmaktadır. Bu mesajlarla kullanıcı, eksik yönlerini ve hatalarını anlamadan çözüm üretmeye çalışmaktadır. Öğrenme sürecini yeterli düzeyde desteklemeyen bu tür ortamlara karşılık geliştirilen ortamda yazılan kodun derinlemesine analizi yapılmakta ve öğrenme sürecine faydalı geri bildirimler oluşturulmaktadır. Geliştirilen ortam bilgisayar bilimleri ve mühendislik teknikleri kullanılarak açık kaynaklı olarak geliştirilmiş olup bu tür ortam geliştirmek isteyen araştırmacılara çatı olma niteliğindedir.
- Mevcut ortamlarda bireyler İngilizce dilini yeterli düzeyde bilmemesinden kaynaklı zorluklar yaşamaktadır. Hem programlama hem de yabancı kelimeleri aynı anda öğrenmeye çalışmaktadır. Programlama komutlarının Türkçe olması erken yaşlarda programlama öğreniminin daha kolay yapılmasını sağlamaktadır.
- Derleyici tasarım tekniklerinin kullanılması, kullanıcının yazdığı kodun derinlemesine analizinin yapılmasını sağlamaktadır. Bu analizler sözcüksel ve

sözdizimsel analiz aşamalarının düzenli ifadeler ve sonlu durum makinelerinin birlikte kullanılmasıyla yapılmaktadır. Kodun derinlemesine analizi sayesinde kullanıcının öğrenemediği programlama kavramları tam olarak tespit edilebilmektedir. Bu tespitlerle birlikte kullanıcıya yönlendirici geri bildirimler sağlanarak başarılı bir öğrenme süreci hedeflenmektedir.

- Ortamda kültürel simgelerimiz olan Nasreddin Hoca ve Keloğlan karakterleri kullanılarak animasyonlar gösterilmektedir. Kültürel simgelerin kullanılmasıyla ortamın ilgi çekici ve motivasyon artırıcı olması beklenmektedir. Ayrıca Nasreddin Hoca karakterinin bilge kişiliğiyle geri bildirim vermesiyle kullanıcının başarılı bir şekilde yönlendirilmesi hedeflenmektedir.

4. ÖNERİLER

- Görsel bileşenlerle programlama öğrenilen ortamlarda görsel bileşenlerin fiziksel yapısından dolayı sözdizimi hatası yapmak mümkün olmamaktadır. Öğrenme sürecinin hata yapılamayan bir ortamda gerçekleştirilmesinin dezavantajları bulunmaktadır. Bireyler öğrenme sürecinde hata yapmakta ve bu hataları çözerek öğrenme sürecini desteklemektedir. Bu sebeple programlama öğrenme ortamlarında sözdizimi hatası yapılabilmesi ve yapılan bu sözdizimi hatalarının düzeltilmesine yönelik kullanıcıya yeterli düzeyde bilgi sunulması gerekmektedir. Bahsedilen modelde olan programlama öğrenme ortamlarının daha başarılı olacağı düşünülmektedir.
- Geliştirilen ortam değişenler, veri tipleri, aritmetik ve mantıksal işlemler ile koşul ve döngü deyimleri konularını kapsamaktadır. Temel programlama kavramlarını kapsayan bu ortama fonksiyon kavramı da eklenerek ortamın geliştirilmesine devam edilebilir. Ayrıca sınıf yapısı ile birlikte nesne yönelimli programlama konusu da eklenerek daha kapsayıcı bir programlama öğrenme ortamına dönüştürülebilir. Oluşturulan senaryo sayısı artırılarak programlama kavramlarının öğrenimi ve konunun kavranması desteklenebilir.
- Geliştirilen ortam kullanılarak öğrencilerin programlama öğrenme başarısı üzerinde deneysel çalışma yapılabilir. Geliştirilen ve mevcut öğrenme ortamları, programlama öğrenme başarısı açısından başarı testleri ile kıyaslanabilir. Ayrıca programlama öğreniminin üst düzey bilişsel becerileri geliştirmesi noktasında deney ve kontrol grupları oluşturularak çalışmalar artırılabilir. Bahsedilen ve benzeri yapılacak araştırmalar sonucunda programlama öğreniminin nasıl daha başarılı bir şekilde yapılabileceği keşfedilebilir.
- Uygulama yazılımı, nesne yönelimli programlama ilkeleri ve Solid prensiplerine uygun olarak geliştirilmiştir. Geliştirilen uygulama ile bireylerin bir öğreticiye ihtiyaç duymadan kendi başlarına öğrenebilmeleri amaçlanmıştır. Bu doğrultuda senaryolar oluşturularak programlama kavramlarının öğretilmesi hedeflenmiştir. Bu yönüyle geliştirilen çerçeve sadece programlama öğrenimi için değil sayısal veya

sözel bütün dersler için bir öğrenme ortamı olarak düşünülebilir ve bu amaç doğrultusunda geliştirilebilir.

5. KAYNAKLAR

1. Taheri, S. M., Sasaki, M. ve Ngetha, H. T., Evaluating the Effectiveness of Problem Solving Techniques and Tools in Programming, Science and Information Conference (SAI), Temmuz 2015, London, Bildiriler Kitabı, 928-932.

2. Kalelioğlu, F. ve Gülbahar, Y., The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective, Informatics in Education, 13, 1 (2014).

3. Numanoğlu, M. ve Keser, H., Programlama Öğretiminde Robot Kullanımı-Mbot Örneği, Bartin Üniversitesi Egitim Fakültesi Dergisi, 6, 2 (2017) 497.

4. Kalelioğlu, F., A New Way of Teaching Programming Skills to K-12 Students: Code.org, Computers in Human Behavior, 52 (2015) 200-210.

5. Armoni, M., Meerbaum-Salant, O. ve Ben-Ari, M., From Scratch to "Real" Programming, ACM Transactions on Computing Education (TOCE), 14, 4 (2015) 25.

6. https://github.com/sefaaras/kodegit Kodlama Eğitim Platformu. 1 Mayıs 2018.

7. Koorsse, M., Cilliers, C. ve Calitz, A., Programming Assistance Tools to Support the Learning of IT Programming in South African Secondary Schools, Computers & Education, 82 (2015) 162-178.

8. Akcaoglu, M. ve Koehler, M. J., Cognitive Outcomes from the Game-Design and Learning (GDL) After-School Program, Computers & Education, 75 (2014) 72-81.

9. Fessakis, G., Gouli, E. ve Mavroudi, E., Problem Solving by 5–6 Years Old Kindergarten Children in a Computer Programming Environment: A Case Study, Computers & Education, 63 (2013) 87-97.

10. Kazakoff, E. R., Sullivan, A. ve Bers, M. U., The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood, Early Childhood Education Journal, 41, 4 (2013) 245-255.

11. Sáez-López, J. M., Román-González, M. ve Vázquez-Cano, E., Visual Programming Languages Integrated Across the Curriculum in Elementary School: A Two Year Case Study Using "Scratch" in Five Schools, Computers & Education, 97 (2016) 129-141.

12. Garcia, P. G. F. ve De la Rosa, F., RoBlock–Web App for Programming Learning, International Journal of Emerging Technologies in Learning (iJET), 11, 12 (2016) 45-53.

13. Lee, I., Martin, F. ve Apone, K., Integrating Computational Thinking Across the K-8 Curriculum, Acm Inroads, 5, 4 (2014) 64-71.

14. Rogozhkina, I. ve Kushnirenko, A., Piktomir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment, Procedia-Social and Behavioral Sciences, 28 (2011) 601-605.

15. Lin, J. M. C. ve Liu, S. F., An Investigation into Parent-Child Collaboration in Learning Computer Programming, Journal of Educational Technology & Society, 15, 1 (2012) 162.

16. Miller, P., Learning with a Missing Sense: What Can We Learn from the Interaction of a Deaf Child with a Turtle?, American Annals of the Deaf, 154, 1 (2009) 71-82.

17. Elkin, M., Sullivan, A. ve Bers, M. U., Programming with the KIBO Robotics Kit in Preschool Classrooms, Computers in the Schools, 33, 3 (2016) 169-186.

18. Papadakis, S., Kalogiannakis, M. ve Zaranis, N., Developing Fundamental Programming Concepts and Computational Thinking with Scratchjr in Preschool Education: A Case Study, International Journal of Mobile Learning and Organisation, 10, 3 (2016) 187-202.

19. Aras, S., Özyurt, Ö. ve Özyurt, H., Erken Yaşlarda Kodlama Eğitimine Yönelik Araştırmalardaki Eğilimin İncelenmesi: 2009-2016 Döneminde Yayımlanan Makalelerin İçerik Analizi, 11th International Computer and Instructional Technologies Symposium (ICITS), Mayıs 2017, Malatya, Bildiriler Kitabı, 399-400.

20. Lau, W. W. ve Yuen, A. H., Exploring the Effects of Gender and Learning Styles on Computer Programming Performance: Implications for Programming Pedagogy, British Journal of Educational Technology, 40, 4 (2009) 696-712.

21. Sengupta, P., Farris, A. V. ve Wright, M., From Agents to Continuous Change via Aesthetics: Learning Mechanics with Visual Agent-Based Computational Modeling, Technology, Knowledge and Learning, 17, 1-2 (2012) 23-42.

22. Bers, M. U., Flannery, L., Kazakoff, E. R. ve Sullivan, A., Computational Thinking and Tinkering: Exploration of an Early Childhood Robotics Curriculum, Computers & Education, 72 (2014) 145-157.

23. Denner, J., Werner, L. ve Ortiz, E., Computer Games Created by Middle School Girls: Can They be Used to Measure Understanding of Computer Science Concepts?, Computers & Education, 58, 1 (2012) 240-249.

24. Lye, S. Y. ve Koh, J. H. L., Review on Teaching and Learning of Computational Thinking Through Programming: What is Next for K-12?, Computers in Human Behavior, 41 (2014) 51-61.

25. Portelance, D. J., Strawhacker, A. L. ve Bers, M. U., Constructing the Scratchjr Programming Language in the Early Childhood Classroom, International Journal of Technology and Design Education, 26, 4 (2016) 489-504.

26. https://scratch.mit.edu/ Scratch. 1 Mayıs 2018.

27. Özyurt, Ö., Özyurt, H. ve Aras S., Çocukların Kodlama Öğrenebilecekleri Ortamların İncelenmesi, 10th International Computer and Instructional Technologies Symposium (ICITS), Mayıs 2016, Rize, Bildiriler Kitabı, 752-759.

28. Maloney, J., Resnick, M., Rusk, N., Silverman, B. ve Eastmond, E., The Scratch Programming Language and Environment, ACM Transactions on Computing Education (TOCE), 10, 4 (2010) 16.

29. Çatlak, Ş., Tekdal, M. ve Baz, F. Ç., Scratch Yazılımı ile Programlama Öğretiminin Durumu: Bir Doküman İnceleme Çalışması, Journal of Instructional Technologies & Teacher Education, 4, 3 (2015).

30. Burke, Q., The Markings of a New Pencil: Introducing Programming as Writing in the Middle School Classroom, Journal of Media Literacy Education, 4, 2 (2012) 121-135.

31. https://www.scratchjr.org/ ScratchJr. 1 Mayıs 2018.

32. https://code.org/ Code.org. 1 Mayıs 2018.

33. http://tkroboticsnetwork.ning.com/page/kiwi-software-cherp Cherp. 1 Mayıs 2018.

34. https://www.robomindacademy.com/go/robomind/home Robomind. 1 Mayıs 2018.

35. http://kinderlabrobotics.com/ Kibo. 1 Mayıs 2018.

36. Sullivan, A. ve Bers, M. U., Robotics In the Early Childhood Classroom: Learning Outcomes from an 8-Week Robotics Curriculum in Pre-Kindergarten Through Second Grade, International Journal of Technology and Design Education, 26, 1 (2016) 3-20.

37. https://www.greenfoot.org/door Greenfoot. 1 Mayıs 2018.

38. Kölling, M., The Greenfoot Programming Environment, ACM Transactions on Computing Education (TOCE), 10, 4 (2010) 14.

39. Deterding, S., Sicart, M., Nacke, L., O'Hara, K. ve Dixon, D., Gamification. Using Game-Design Elements in Non-Gaming Contexts, Human Factors in Computing Systems, Mayıs 2011, Texas, Genişletilmiş Özet Bildiriler Kitabı, 2425-2428.

40. Deterding, S., Dixon, D., Khaled, R. ve Nacke, L., From Game Design Elements to Gamefulness: Defining Gamification, 15th International Academic MindTrek Conference: Envisioning Future Media Environments, Eylül 2011, Tampere, Bildiriler Kitabı, 9-15.

41. Nicholson, S., Gamification in Education and Business, Wood, L. C. ve Reiners, T., First Edition, 1-20, Springer, New York, 2015.

42. Muntean, C. I., Raising Engagement in E-Learning Through Gamification, 6th International Conference on Virtual Learning (ICVL), Ekim 2011, Cluj-Napoca, Bildiriler Kitabı.

43. Dicheva, D., Dichev, C., Agre, G. ve Angelova, G., Gamification in Education: A Systematic Mapping Study, Journal of Educational Technology & Society, 18, 3 (2015) 75.

44. Hamari, J., Koivisto, J. ve Sarsa, H., Does Gamification Work? - A Literature Review of Empirical Studies on Gamification, 47th Hawaii International Conference on System Science (HICSS), Ocak 2014, Hawaii, Bildiriler Kitabı, 3025-3034.

45. Aho, A. V., Lam, M. S., Sethi, R. ve Ullman, J. D., Compilers, Principles, Techniques and Tools, Second Edition, Pearson Education, Boston, 2007.

46. Grune, D., Van Reeuwijk, K., Bal, H. E., Jacobs, C. J. ve Langendoen, K., Modern Compiler Design, Second Edition, Springer Science & Business Media, New York, 2012.

47. Efendioğlu S., Polinomlar İçin Bir Simgesel Hesaplama Çatısının Tasarımı ve Gerçeklenmesi, Yüksek Lisans Tezi, K.T.Ü., Fen Bilimleri Enstitüsü, Trabzon, 2017.

48. Cooper, K. ve Torczon, L., Engineering a Compiler, Second Edition, Elsevier, Boston, 2011.

49. Mogensen, T. Æ., Basics of Compiler Design, First Edition, Diku, Copenhagen, 2009.

50. Sidhu, R. ve Prasanna, V. K., Fast Regular Expression Matching Using Fpgas, 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Mart 2001, Bildiriler Kitabı, 227-238.

51. Yu, F., Chen, Z., Diao, Y., Lakshman, T. V. ve Katz, R. H., Fast And Memory-Efficient Regular Expression Matching For Deep Packet Inspection, Architecture for Networking and Communications Systems (ANCS), Aralık 2006, Bildiriler Kitabı, 93-102.

52. http://bidb.itu.edu.tr/eskiler/seyirdefteri/blog/2013/09/08/düzenli-i-fadeler-(regular-expressions) Düzenli İfadeler. 1 Mayıs 2018.

53. Owens, S., Reppy, J. ve Turon, A., Regular-Expression Derivatives Re-Examined, Journal of Functional Programming, 19, 2 (2009) 173-190.

54. https://help.libreoffice.org/Common/List_of_Regular_Expressions/tr Regular Expression. 1 Mayıs 2018.

55. Garofalakis, M. N., Rastogi, R. ve Shim, K., SPIRIT: Sequential Pattern Mining with Regular Expression Constraints, 25th International Conference on Very Large Data (VLDB), Ekim 1999, Edinburgh, Bildiriler Kitabı, 7-10.

56. http://myweb.sabanciuniv.edu/alper/regular-expression-duzenli-ifadeler/ Düzenli İfadeler. 1 Mayıs 2018.

57. Rabin, M. O. ve Scott, D., Finite Automata and Their Decision Problems, IBM Journal Of Research And Development, 3, 2 (1959) 114-125.

58. Hopcroft, J. E., Motwani, R. ve Ullman, J. D., Introduction to Automata Theory, Languages, and Computation, Third Edition, Pearson Education, Boston, 2008.

59. Moore, C., Automata, Languages, and Grammars, Lecture Notes, 24.01.2015.

60. https://httpd.apache.org/ABOUT_APACHE.html Apache. 1 Mayıs 2018.

61. http://php.net/manual/en/intro-whatis.php What is Php. 1 Mayıs 2018.

62. http://php.net/manual/en/intro-whatcando.php What can do Php. 1 Mayıs 2018.

63. https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript Javascript. 1 Mayıs 2018.

64. Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Naur, P., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., Van Wijngaarden, A. ve Woodger M., Report on the Algorithmic Language ALGOL 60, Numerische Mathematik, 2, 1 (1960) 106-136.

65. Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J. ve Perlis, A. J., Revised Report on the Algorithmic Language Algol 60, P. Naur, First Edition, Springer, New York, 1969.

66. De Graaf, D., Practical Use of Automata and Formal Languages in the Compiler Field, Yüksek Lisans Tezi, University of Amsterdam, Informatica, Amsterdam, 2017.

ÖZGEÇMİŞ

Sefa ARAS, 1991 Trabzon doğumludur. İlköğretim eğitimini Araklı Cumhuriyet İlköğretim Okulu'nda ve lise eğitimini Kanuni Anadolu Lisesi'nde tamamlamıştır. 2009 güz döneminde Gazi Üniversitesi Bilgisayar Mühendisliği Bölümü'nde başladığı lisans eğitiminden 2014 yılı bahar döneminde mezun olmuştur. 2014 yılının Kasım ayında başladığı vatani görev olan askerlik hizmetinden 2015 yılının Nisan ayında terhis olmuştur. 2015 yılının güz döneminde Karadeniz Teknik Üniversitesi; Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine ve Yazılım Mühendisliği Bölümü'nde araştırma görevlisi olarak görev yapmaya başlamış olup bu eğitim ve görevini sürdürmektedir.