28857

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SİMÜLASYON AMAÇLI UZMAN SİSTEMLER VE KUYRUK MODELLERİNİN SİMÜLASYONU İÇİN BİR UZMAN SİSTEM TASARIMI

YÜKSEK LİSANS TEZİ

END.MÜH. TUFAN DEMİREL

İSTANBUL 1993

TEŞEKKÜR

Bu çalışmam esnasında, beni yönlendiren ve yardımlarını esirgemeyen Hocam Sayın Prof. Yaşar Baki CENGİZ Bey'e içtenlikle teşekkür ederim.

Ayrıca çalışmalarım sırasındaki çevirilerimde, yardım ve destek gördüğüm arkadaşım Endüstri Yüksek Mühendisi Halil İbrahim ERDEM'e de içtenlikle teşekkür ederim.

Tufan DEMİREL
Ocak, 1993

İÇİNDEKİLER

Sayfa No.

Sayfa No.

Sayfa No.

ÖZET

Simülasyon ve uzman sistemler arasında kuvvetli bir metodolojik benzerlik vardır. Her ikisinde de yapılan çalışma, karar vermeye yardım eden bir bilgisayar tabanlı model oluşturmaktır.

Simülasyon bir çok problemin çözümünde ve analizinde son derece başarılı bir şekilde kullanılmaktadır. Klasik simülasyon yaklaşımında, simülasyonu yapacak kişinin, modellenecek sistem hakkında bilgi sahibi olması gerekmektedir. Simülasyonu yapan kişi bu bilgiyi kullanarak problemin spesifikasyonunu oluşturur. Simülasyon için bir sonraki adım ise, genel amaçlı bir programlama dili veya özel amaçlı bir simülasyon dilinin kullanılarak, modelin bir bilgisayar programının oluşturulmasıdır. Daha sonra geliştirilen bu simülasyon modeli (programı) geçerli hale getirilir ve denemeler yapılarak sonuçlar elde edilir. Simülasyoncu daha sonra bu sonuçları analiz ederek yorumlar. İşte bir simülasyon çevriminin bütün bu kısımları, mevcut yapay zekâ ve uzman sistem teknikleri kullanılarak otomatik hale getirilebilir. Böylece simülasyonu yapılacak alan hakkında bilgi sahibi olan fakat bir simülasyon dilini bilmeyen bir kişinin bile, bir simülasyon çalışması yapması sağlanır.

Bu tezde geliştirilen simülasyon amaçlı uzman sistem daha çok, bir simülasyon çevriminin, problemin spesifikasyonunu elde etme, modelini kurma, modeli işletme veya çalıştırma ve sonuçları ortaya çıkarma safhaları üzerinde yoğunlaştırılmıştır.

Tez, altı ana bölümden oluşmaktadır. Birinci bölüm giriş bölümü olup bu bölümde, yapay zekâ, uzman sistem ve simülasyon teknikleri tanıtılmış ve bunların birbirleriyle nasıl bir ilişki içinde olduğu belirtilerek konuya bir giriş yapılmıştır.

Tezin ikinci bölümünde uzman sistemlerin genel yapıları ve geliştirilmeleri anlatılmıştır. Önce yapay zekânın ortaya çıkışı üzerinde durulmuş ve daha sonra uzman sistemlere geçilmiştir. Bu

bölümde uzman sistem kavramı ayrıntılı olarak incelenerek bir uzman sistemin bileşemleri ortaya konulmuştur. Daha sonra uzman sistemlerin dış ortam arabirimleri üzerinde durulmuş ve bir uzman sistemin çıkarım şebekeleri anlatılmıştır. Bu bilgiler verildikten sonra tipik bir uzman sistemde bilgilerin nasıl organize edildiği hususu da ayrıntılı bir şekilde açıklanmıştır. Daha sonra uzman sistem oluşturma araçları üzerinde durulmuş, ve son olarak da bir uzman sistemin geliştirilmesi ve geliştirme adımları içindeki faaliyetler incelenmiştir.

Üçüncü bölümde, simülasyon ve uzman sistemlerin birbirleriyle olan ilişkileri ayrıntılı bir şekilde incelenmiştir. Bu bölümde ilk olarak genel sistem simülasyonu bilgileri verildikten sonra bilgisayarla klasik simülasyon yaklaşımı açıklanmış, ve daha sonra uzman simülasyon sistemlerinin geliştirilmesi üzerinde durularak, otomatik olarak problemin spesifikasyonunu elde etme ve simülasyon programını veya simülasyon modelini oluşturma yapıları açıklanmış ve en sonunda da bir simülasyon üretecinin genel yapısı ortaya çıkarılmıştır. Bu bölümde daha sonra uzman sistemler ile simülasyonun birleştirilmesine ait bir sınıflandırma yapılmış, ve bir simülasyon modeli ile yapay zekâ arasındaki farklar, uzman sistemlerin simülasyon metodolojisi üzerindeki etkileri ve son olarak programlama dillerindeki gelişmelerin simülasyon ve uzman sistemlere olan etkileri anlatılmıştır.

Tezin dördüncü bölümünde, kuyruk modellerinin simülasyonu için simülasyon amaçlı bir uzman sistemin geliştirilmesi açıklanmıştır. Bu bölümde önce kuyruk sistemlerinin genel yapıları ve kuyruk sistemlerinin simülasyonu incelenmiş, daha sonra da uzman sistemin program üreteceği dil olarak seçilen SIMAN simülasyon dili ile modelleme yapısı ele alınarak, kuyruk modellerinin simülasyonuna yönelik bir uzman sistemin geliştirilmesi çalışmalarına geçilmiştir. Burada öncelikle bir diyalog arabiriminin gerekliliği vurgulanmış ve bu nedenle uzman sistem için bir diyalog arabirimi geliştirilmiştir. Daha sonra, otomatik olarak üretilecek olan simülasyon modellerinin kabul ettiği varsayımlar açıklanarak kuyruk modellerinin simülasyon programlarını otomatik olarak oluşturacak, simülasyonu işletecek ve sonuçları elde ederek kullanıcıya sunacak şekilde geliştirilen bilgisayar programının (uzman sistemin)

mantıksal yapısı anlatılmıştır. Geliştirilen bilgisayar programı Turbo PASCAL 6.0 ile yazılmıştır ve bu program SIMAN 3.5 simülasyon dilinde kuyruk modellerinin simülasyon programlarını oluşturmaktadır. Geliştirilen program, problemin spesifikasyonunu kullanıcıdan bir diyalog arabirimi ile elde ederek simülasyon modelini (programını) kurmakta, kurulan modeli işletmekte ve işletim sonucunda elde edilen istatistiksel çıktıları kullanıcıya sunmaktadır. Bu bölümde son olarak geliştirilen programın veya uzman sistemin, teorik veya yapay bir örnek üzerinde çalıştırılması gösterilmektedir.

Tezin beşinci bölümünde bir poliklinik uygulaması açıklanmaktadır. Bu bölümde ilk olarak, geliştirilen simülasyon amaçlı uzman sistemin hangi tür kuyruk modellerinde uygulanabileceği açıklanmış bir poliklinik modelinin simülasyonu için, geliştirilen program kullanılarak sonuçların elde edilmesi gösterilmiştir.

Tezin altıncı ve son bölümü ise genel değerlendirmeler ve bu tez çalışması sonucunda ortaya çıkarılan sonuçları kapsamaktadır. Bu bölümde simülasyon ve uzman sistemlerin birleştirilmesi sonucunda elde edilen avantajlar üzerinde durulduktan sonra, simülasyon, yapay zekâ ve uzman sistemlerin birbirlerini nasıl etkiledikleri ortaya konmakta ve bazı yorumlar yapılmaktadır.

SUMMARY

There is a methodological similarity between simulation and expert systems techniques. The research done on both techniques develops a computer based model which helps making decisions.

Simulation is positively used in solving and analyzing a number of problems. In traditional simulation approach, the simulationist has to know the system to be modelled. She/he using this knowledge develops a problem specification. The next step in simulation is to develop a computer program of the model using a general purpose programming language or a special purpose programming language. The simulation model (program) developed is then validated and results are obtained by experiments. Simulationist then analyzes and interprets these results. All these portions of a simulation life cycle can be automated utilizing available techniques of artificial intelligence and expert systems. Hence, even a person who knows the knowledge of the problem domain but does not know any simulation language is allowed to perform a study of simulation.

Simulation purpose expert system developed in this thesis focuses on the stages of the simulation life cycle such as obtaining problem specification, model construction, model executing and gathering the results.

The thesis consists of six fundamental sections. The first section is an introduction. In this section, artificial intelligence, expert system and simulation techniques are introduced and it is mentioned what kind of a relationship they have among themselves.

In the second section, general structures of expert systems and their developings are examined. It is first focused on the arising of artificial intelligence and then expert systems are examined. In this section the expert system concept is surveyed in detail and the portions of a expert system are explained. The external environment interfaces of expert systems are then emphasized and inference networks of an expert system are mentioned. After these are given, it is investigated in detail how the knowledges are organized in a typical expert system. Then, expert system building devices are examined and finally developing an expert system and activities in developing steps are mentioned.

In the third section, the relationship between simulation and expert system is described. In this section, general system simulation is first given and then traditional simulation approach via a computer is explained. Furthermore, it is focused on developing expert simulation systems. Automatic problem specification gathering and structures of developing a simulation model or simulation program are explained and finally general structure of a simulation generator is presented. In this section, there is also a classification of linking simulation and expert systems and differences between a simulation model and artificial intelligence, affects of expert systems on simulation methodology and eventually affects of improvements in programming languages on simulation and expert systems are given.

In the fourth chapter, developing a simulation purpose expert system for the simulation of queuing models is explained. In this section, general structures of queuing systems and simulation of queuing systems are examined, hence considering modelling frame with SIMAN simulation language which expert system will generate the program, studies of developing an expert system directed to simulation of queuing systems are considered. It is emphasized the necessity

of a dialog interface and therefore a dialog interface is developed for expert system. Subsequently, defining the assumptions approved by simulation models to be generated automatically, it is expressed the logical structure of computer program (expert system) which generates simulation programs of queuing systems automatically, executes the simulation and derive the results and presents to the user . Proposed computer program is written in Turbo PASCAL 6.0 using an IBM PC/AT. This program also generates the simulation programs of queuing models in SIMAN 3.5 simulation language. The proposed program wants the user to input problem specification via a dialog interface, builds simulation model(program),executes the model constructed and then introduce the user statistical outputs obtained at the end of execution. Finally, in this section proposed program or expert system is executed for an artificial example.

In the fifth section, a polyclinic implementation is stated. In this section it is first declared what kind of queuing models can be appropriate by proposed simulation purpose expert system. For a polyclinic model simulation it is shown how results are obtained using proposed program.

The sixth or closing section covers up general evaluations and results obtained. In this section, after it is focused on the advantages obtained by linking simulation and expert systems, it is also shown how simulation, artificial intelligence and expert systems effect each others and then some interpretions are given.

1. GİRİŞ

Herhangi bir sorunun bilgisayar yardımıyla incelenmesinde temel yaklaşım, o sorunu bilgisayarın anlayacağı bir şekilde kodlamak, başka bir deyişle, bilinen bir dilde bir bilgisayar programı yazmaktır. Böylece bilgisayardan ancak programlama bilgisine sahip olanların yararlanabilmesi bir sakınca olarak ortaya çıkmaktadır. Bu sakıncaya bir anlamda çare bulmak amacıyla, bilgisayarların ilk kullanılmaya başlandığı zamanlardan itibaren hazır program paketleri üzerinde çalışılmış ve kullanıcının, yalnızca veri girişi ile problemini çözebilmesini sağlanıştır. Bir çok matematiksel işlemler, veri tabanı uygulamaları ve kelime işlem paketleri bu tür hazır paket programlara örnek sayılabilir.

Hazır program paketleri ile yapay zekâ uygulamalarının da başladığı söylenebilir. Bigisayarın, hız ve kapasitesindeki gelişmelere paralel olarak, kullanımındaki ekonomiklik sonucu her alanda yaygınlaşması, yapay zekâ paketlerinin de gelişmesine hız kazandırmıştır. Hazır paket programlar yalnızca problem özelliklerine, problem algılama, irdeleme, çözme, sonuçlara yorum getirme ve ileride benzer problemlerde kullanmak üzere elde edilen sonuçlardan öğrenme özellikleri ilave edildiğinde yapay zekânın bugün için tanımlanan şekli ortaya çıkmaktadır.

Bilgisayarla simülasyon ele alındığında, klasik olarak bir simülasyoncu, genel amaçlı bir programlama dilini (BASIC, FORTRAN, PASCAL, vb.) veya özel amaçlı bir simülasyon dilini (SIMAN, GPSS, SLAM, vb.) kullanarak bir model oluşturur. Daha sonra model geçerli hale getirilir, denemeler yapılarak sonuçlar bulunur ve analiz edilip yorumlanır. Bir simülasyon çevriminin çeşitli kısımları mevcut yapay zekâ ve uzman sistem metot ve teknikleri kullanılarak, gerçekleştirilebilir. Tamamen birleşik yapay zekâ tabanlı simülasyon sistemleri ve simülasyon ortamları, bir kullanıcıya tüm simülasyon çevrimi boyunca otomatik destek sağlar.

Burada açıklanan çalışma, simülasyon model çevriminin daha çok model kurma, bu modeli işletme ve sonuçları elde etme üzerinde yoğunlaştırılmıştır. Bir çok simülasyon araştırmacılarına göre, yapay zekâ ve uzman sistem tekniklerinin uygulanması, model kurma esnasında yapılmaktadır. Yapay zekâ teknikleri, yeni simülasyon modellerinin otomatik olarak oluşturulduğu, model veya model parçalarına ait bilgi tabanının oluşturulması ve muhafaza edilmesinde kullanılmaktadır. Alt modellerin ve model parçalarının veri tabanı, uygun mevcut bir model için veya mevcut modellerden yeni modeller oluşturmak için kullanıcının bir veri tabanı araştırmasına olanak veren, model kuran uzman sistemler için bir temel teşkil eder.

Model kurmaya alternatif bir yaklaşım, yapay zekâ tabanlı otomatik yazılım üretme yöntemi ile ilgili yazılım mühendisliği araştırmalarıyla paralel gitmektedir. Bu yaklaşım, model geliştirmek için otomatik programlama tekniklerini kullanmaktadır. Bu tezde gerçekleştirilen çalışma da, bu alternatifin bir uygulamasıdır. Otomatik programlama, SIMAN simülasyon diliyle kuyruk sistemlerinin simülasyon modellerinin (SIMAN dilinde oluşturulan programlar) kurulmasında kullanılmaktadır. Kullanıcı, bir diyalog arabirimi ile modellenecek sistem hakkında uygun ve gerekli bilgileri sağlar. Böylece probleme ait spesifikasyonlar elde edilir. Geliştirilen sistem, kural tabanı, kuyruk sistemleri ve hedef dil (uzman sistemin, simülasyon programlarını oluşturması amaçlanan simülasyon dili) SIMAN bilgisiyle, genel simülasyon bilgilerini birleştirir. Daha sonra ise uzman sistem, amaca yönelik araştırma ve programa dönüşüm kombinasyonu boyunca bu bilgiyi, model spesifikasyonundaki diyaloğa yardımcı olmak ve simülasyon modelini otomatik olarak kurmak için kullanmaktadır.

Bir simülasyon çalışması ile uzman sistem çalışmasındaki adımlar hemen hemen aynıdır. Yani her ikisinde de amaç, karar vermeye yardım eden bir bilgisayar tabanlı model oluşturmaktır. Çoğunlukla, her ikisi de göz önüne alınan sistemdeki belirsizliği modellemektedir. Modeli oluşturmak için kullanılan yazılım, direkt olarak karar verici tarafından kullanılmalı ve veri tabanı aracılığı ile diğer yazılım sistemleri ile ilişkili olmalıdır.

Simülasyon ve uzman sistem metotlarının benzerliklerinin nedeni, her ikisinin de bilgi sunuşunu yönetmek amacıyla, bir çıkarım mekanizması (inference engine) ile bir sistemi modüler olarak göstermeleridir. Bu benzerlik hem simülasyon araştırmacıları hem de yapay zekâ araştırmacıları tarafından belirtilmiştir. Kullanıcılar tarafından programlanmış kontrol yapısına karşın bir çıkarım mekanizması, çok sayıdaki modül ile çalışır ve modüller (en azından teoride), belirli bir düzende sunulabilir ve modüller birbirinden tamamen bağımsızdırlar (ancak, pratikte bunun elde edilmesi oldukça zordur). Örneğin, olay tabanlı bir simülasyonun çıkarım mekanizması, olay sayısından ve simülasyon içindeki yazım sırasından bağımsızdır. Uzman sistemlerin bir çoğunda olduğu gibi, amaca yönelik çıkarımı kullanan simülasyon sistemlerini oluşturma çalışmaları da mevcuttur. Bir modelin detayı ve gücü, sunuşa modüller ekleyerek veya mevcut modülleri ayırarak arttırılabilir.

Uzman sistemlerin çalışmalarındaki en yaygın bilgi sunuşları, şebekeleri, yapıları, kuralları ve bunların sonuçlarını içermektedir. Çıkarım mekanizmaları da, ileriye ve geriye bağlantıları içermektedir. Herbir bağlantı, Bayes teoremi, bulanık, mantıksal (lojik) veya diğer çok sayıdaki farklı metotlara bağlı olarak güncelleşen bazı belirsizlikler yapılarıyla ilgilidir. Simülasyondaki sunuşlar, olayları, faaliyetleri ve işlem ilişkilerini içermektedir.

Benzerlikler, buradada bitmemektedir. Bir çok araştırmacı, şartlı olay ve üretim kuralları (genelde durum/hareket kuralı olarak adlandırılır) arasındaki benzerlikleri belirtmişlerdir. Buradaki üretim kuralı, yapay zekâdaki, alan bilgisi (problem sahası ile ilgili bilgi) sunuşu olarak ortaya çıkmaktadır. Simülasyonun sunulmasında şartlı olayların kullanılması yeni bir metottur. Bir faaliyetin başlangıcı veya bitişine göre şartlar listelenir. Şart karşılanırasa bunlar gözden geçirilir (taranır) ve işlem bitirilir. Her taramadan sonra zaman ilerler. Yani bir sonraki olay mekanizması yoktur. Bu, faaliyet taraması olarak belirtilen bir dünya görüşüdür. Örnek olarak tek servisçili bir kuyruk sistemi arka sayfada gösterilen şartlı olaylar şeklinde ifade edilebilir.

IF [EĞER] kuyrukta bir gezer birim var
ve
servisçi boş
THEN [İSE] servisçi meşgul olur,
gezer birimi kuyruktan çıkar,
son servise kadar zamanı işaretle.

IF [EĞER] son servis zamanı
THEN [İSE] servisçiyi boş duruma getir,
gezer birimi ortadan kaldır.

IF [EĞER] gezer birim gelir
THEN [İSE] gezer birimi kuyruğa ekle,
bir sonraki gezer birimin gelişini işaretle.

Bu örnek aynı zamanda yapay zekâdaki, temel yaklaşım mantığını da basit olarak göstermektedir.

2. UZMAN SİSTEMLERİN GENEL YAPILARI VE GELİŞTİRİLMELERİ

Yapay zekâ, belirli sınırlı alanlar için insan düşüncesinin karakteristiklerinden bazılarını (öğrenme kabiliyeti, sonuçlandırma, problemleri çözme ve konuşma dilini anlama) örnekleyen bilgisayar sistemlerinin tasarlanması ile ilgilidir. Etkileşimli insan-makina arabirimlerinin ve ilgili veri tabanlarının ortaya çıkması ile, bilgisayarlar hamveri işleme makinalarından, gerçek karar destek sistemlerine dönüşmüştür. Etkileşim, bilgisayar ile daha doğal bir ilişkiyi desteklemektedir; ilişkili veri tabanı sistemleri ise depolanan farklı tipteki veriler arasındaki ilişkilerin bulunması ve kullanılmasını sağlar. Bu kabiliyetler, direkt olarak analize ve karar verme prosesine yardım eden bilgisayar destekli sistemlerin gelişmesini mümkün kılmıştır. Yapay zekâ araştırmasının sonuçlarının uygulaması, yönetim karar desteğinin elde edilmesinde daha sonraki gelişmelere olanak vermektedir. Bu yeni sistemler, problem durumları için çözüm alternatiflerini oluşturmakta ve analiz etmektedir.

İnsanlar bilgiyi nasıl elde etmekte, nasıl organize etmekte ve nasıl kullanmaktadır? 1950'lerdeki başlangıcından bu yana yapay zekâ bu proseslerin anlaşılmasıyla ilgilenmiştir. Araştırma ve uygulamalar iki genel sınıfa ayrılmaktadır:

(1) İnsanların yaptıkları şekilde, doğal insan kabiliyetlerini (görme, konuşma vb.) yansıtan veya taklit eden araştırma ve uygulamalar,

(2) Gerçek uzman tarafından kullanılan veya kullanılmayan, öğrenilmiş kabiliyetlerin veya tecrübelerin sonuçlarını taklit eden araştırma ve uygulamalar.

İkinci uygulama sınıfı, genelde " Uzman Sistemler " olarak adlandırılan uygulamaları kapsamaktadır. Bu uygulamalar, özel olarak eğitilmiş veya yetenekli insanlar tarafından normal olarak yapılan işlerin otomasyonuyla ilgilidir. Uzman sistem uygulamaları ilk yapay zekâ

araştırmalarından farklı olabilir. Çünkü, bu şekildeki uygulamaların temel amacı, verilen bir sonuca ulaşmak için uzman bir insan tarafından kullanılan temel mekanizmayı anlamak değil, uzman bir insanın çıkardığı sonuçları tutarlı bir şekilde benzetmektir. " Uzman " veya " Bilgi Tabanlı Sistemler ", belirli bir alanda çok sayıdaki uzmanların tecrübelerini bir kurallar serisi şeklinde derlemek üzere tasarlanırlar. Buradaki kurallar, belirli bir problemle ilgili bir hareket tarzı olarak kullanıcı için sonuçlar ve kabuller çıkarmak (veya otomatik olarak elde etmek) üzere kullanılırlar.

Uzman sistemler, yapay zekâ olarak adlandırılan alanın bir alt bölümüdür. Yapay zekâ adı ilk defa 1956 yılında Jonh McCarthy tarafından ABD'de " Machine Intelligence " kongresinde ortaya konmuş bir kavramdır. Tam doğru bir kavram olmamasına rağmen bütün dünyada yerleşmiştir.

Feigenbaum ve McCorduck /1/ tarafından yapılan bir tanıma göre yapay zekâ " insanların birbirlerinde zekice olarak kabul ettikleri davranışlara sahip bilgisayarların yapılmasıyla ilgili bir bilgisayar bilimleri alt alanıdır ". Bir başka tanıma göre yapay zekâ, bilgisayarları çekici kılmak, insan zekâsına sahip bilgisayarlar geliştirmek, insanın zeki davranışlarını taklit eden makinalar (robotik) yapmaktır. Yapay zekâ ile uğraşan bilim adamları daima bilgisayar programları ile uğraşırlar ve programlar geliştirirler. Bu programlar bir anlamda bir insan gibi düşünürler, yani bir tür problem çözerler. Bu programlar, belirli bir ölçüde bir insanın muhakeme yeteneğine sahiptirler ve problemi çözerken, bu yeteneğini kullanırlar.

1960'larda yapay zekâ bilim adamları yani yapay zekâ ile uğraşan bilim adamları, çok geniş problem gruplarını çözmek için genel metotlar bularak, karmaşık düşünme işlemini, simüle etmeye çalışmışlar ve bu metotları genel amaçlı programlarda kullanmışlardır. Eğer tamamen bu tür genel amaçlı programları yapmak çok zor olsaydı, bu kimseler bunun yerine daha özelleştirilmiş programlarda kullanılmak üzere genel metotları ve teknikleri geliştirme konularına kendilerini konsantre ederlerdi. Bu nedenle 1970'lerde problemlerin formüle edilişinin tanımlanması vs. gibi teknikler bilgisayar zamanını çok almayacak ya da

bilgisayar belleğinde çok yer kaplamayacak çözümlerin başarılı olarak nasıl bulunacağı konusunu çözmek ve aramak o zaman için erken sayılılabilir. Bu stratejilerde bazı başarılara ulaşılmış, ancak elde edilen sonuçlar uygulamaya geçememiştir. Ancak 1970'li yılların sonlarına doğru yapay zekâ bilim adamları bazı önemli sorunlar olduğunu farketmişlerdir. Bunlar da, bir programı güncel yapabilmek için, problem alanı kapsamında, bir çok yüksek düzeyli bilgi yüklemesinin gerektiğidir. Bunun gerçekleştirilmesi ile birlikte özel amaçlı, çok dar bir problem alanında uzman olan bilgisayar programlarının ve bilgisayar sistemlerinin geliştirilmesi sağlanmış oldu. Dar bir problem alanında, yüsek düzeyde bilgi ile desteklenerek oluşturulmuş zeki bilgisayar programlarıyla da uzman sistemler ortaya çıktı. Sınırlı bir problem alanında yüksek seviyede bir performans elde etmek için uzman bilgisini kullanan bilgisayar programlarına " Uzman Sistemler " denilir.

Uzman sistemler, sadece akademik bir nosyon değildir ve nitekim bunların gerçek dünyada da bir çok alanda uygulamaları gerçekleştirilmiştir. Bugüne kadar geliştirilen uzman sistemlerin uygulama alanları olarak aşağıdakiler sayılabilir:

- Ticaret
- Vergilerin hazırlanması ve planlanması
- Borç verme ve kredi limitlerinin belirlenmesi
- Çeşitli hastalıkların teşhisi ve tedavisi
- Bilinmeyen bileşiklerin kimyasal özelliklerinin belirlenmesi
- Otomatik faktörlerin çizelgelenmesi ve kontrolu
- Karmaşık makinaların bakımı
- Hava alanı girişlerine uçakların atanması ve uçuşların çizelgelenmesi
- Gemilere ve uçaklara kargo yükleme
- Bilgisayarların tasarlanması ve konfigürasyonu
- Baskılı devrelerin yerleştirilmesi ve tasarlanması
- Tesis düzenleme ve yerleştirme
- Kimyasal prosesleirn kontrolu ve çizelgelenmesi
- Madenlerin içindeki artıkların analizi ve izlenmesi
- Hızlı yemek (fast food) servislerinin geliştirilmesi

Günümüze kadar geliştirilmiş olan bir çok uzman sistemden, bazıları uygulama alanlarıyla birlikte Tablo 2.1'de gösterilmektedir /2/.

Tablo 2.1. Geliştirilmiş olan bazı uzman sistemler

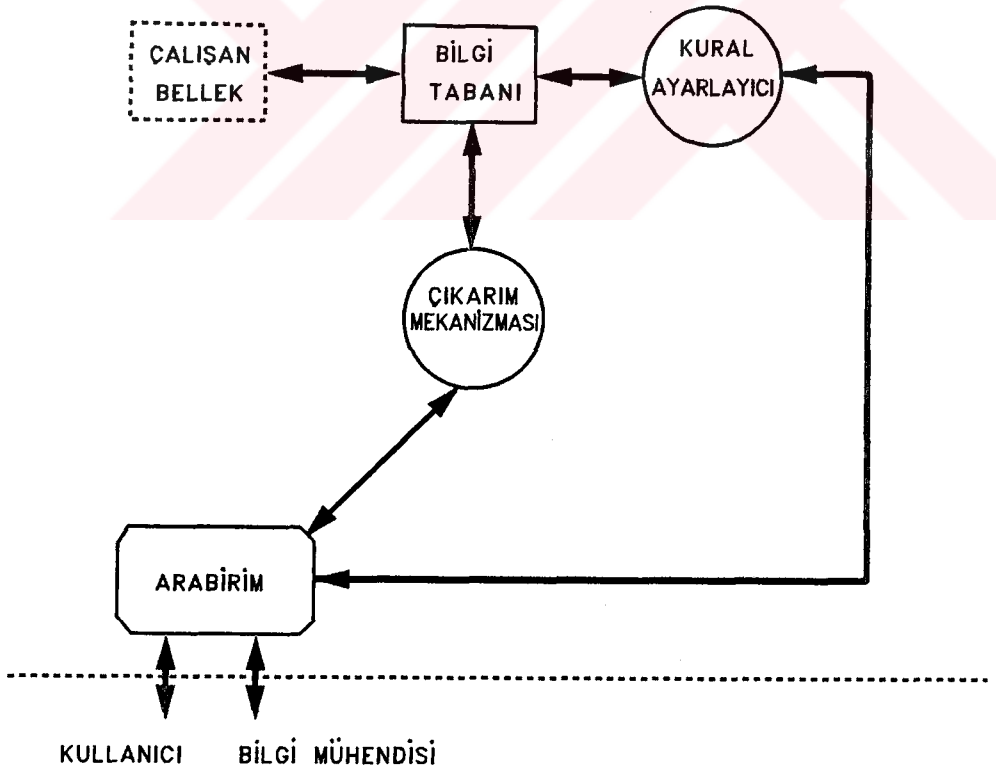
Uzman Sistem	Uygulama Alanı
ACE	Telefon kablo problemlerinin belirlenmesi
DELTA	Lokomotif tamir problemlerinin tanımlanması
DENDRAL	Kütle-spektroskopik analizin yapılması
ExperTAX	Vergilerin hazırlanması ve planlanması
INTERNEST	Dahili hastalıkların teşhisi
MYCIN	Bakteriyel hastalıkların teşhisi
Mentor	Havalandırma sistemleri problemlerinin belirlenmesi
PlanPower	Personel finansal planlamanın yapılması
PUFF	Akciğer hastalıklarının teşhisi
XCON	Bilgisayar sistem konfigürasyonlarının geliştirilmesi

2.1 UZMAN SİSTEM KAVRAMI

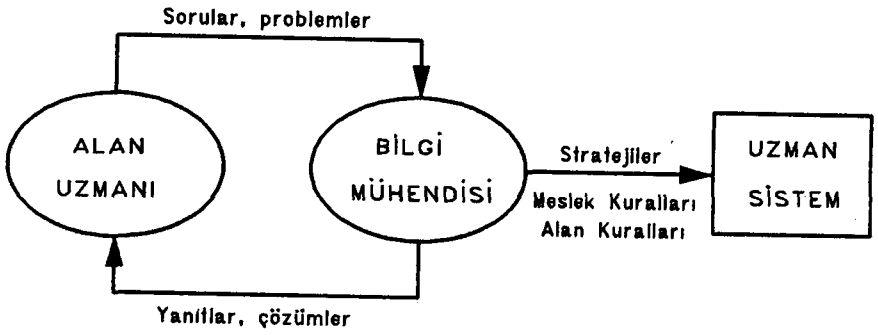
Burada, genel bir uzman sistemin kavramları ve bileşenleri göz önüne alınmaktadır. Bir uzman sistemin olası bir görünümü Şekil 2.1'de görülmektedir /3/. Şekilde kesikli hat üstüne yerleştirilmiş olan bileşenler, bir bilgisayar kapsamındadır yani, bilgisayar yapısı içerisindedir. Bu hattın altında ise, insan kullanıcıların iki tipi belirtilmekte ve uzman sistem bunlara göre tasarlanmaktadır. Bunlardan birincisi, bir bilgi mühendisine yönelik olarak kişisel tasarlanmış olanıdır. Bir bilgi mühendisi (knowledge engineer), uzman sistemin bilgi tabanı içerisine gerekli bilgiyi yerleştirmekten sorumlu olan kişidir. Bir uzman sistemin bu parçası Şekil 2.1'de görülmektedir. Bir başka deyişle, bilgi mühendisliği uzmandan bilgisayara bilgi aktarma işidir ve bazı problem alanlarında uzman kişilerle, bilgi mühendisi arasındaki özel bir etkileşim biçimini içerir. Bu etkileşim Şekil 2.2'de gösterilmektedir /4/. Bilgi mühendisi, bilgi tabanına bilgiyi yerleştirme işlemini bir arabirim ve bir kural ayarlayıcı sayesinde gerçekleştirir.

Bir bilgi mühendisi ayrıca, bir insan uzmanla bir uzman sistem arasında bir arabirimdir. Bir bilgi uzmanı herhangi bir şekilde, bir insan uzmanın uzmanlığını elde etmeli ve daha sonra, kesinleşen bu uzman raporu, belirli bir formatla, bir uzman sistem tarafından kullanılacak olan bilgi tabanının içerisine yüklemelidir. İdeal bir uzman sistem içinde, bir bilgi mühendisine gereksinim olmamalıdır. Bir alan uzmanı, uzman sistem ile direkt olarak birbirini etkilemektedir ve Şekil 2.1'de görülen bilgi mühendisinin yerini alabilmektedir.

İkinci tip te, Şekil 2.1'de görüldüğü gibi, bir uzman sistem, kullanıcı ile kolayca etkileşim sağlayacak biçimde tasarlanmıştır. Burada bir uzman sistem, karar vermeye yardımcı olamak üzere görevlendirilmiş bir kimse gibi kullanılabilmektedir. Başarılı bir bilgi mühendisi hiçbir zaman unutmamalıdır ki, bir uzman sistem, sonunda bir bilgi mühendisi veya alan uzmanı için değil, bir kullanıcının kazancı için tasarlanmaktadır.



Şekil 2.1. Bir Uzman Sistemin Genel Görünüşü



Şekil 2.2. Bir Bilgi Mühendisinin Bilgileri Uzmandan Bir Uzman Sisteme Aktarışı

Çıkarım mekanizması (inference engine), karşılaştırma safhası esnasında kullanılmaktadır. Bir çıkarım mekanizması karşılaştırma esnasında iki temel işlemi yerine getirir. İlk olarak, bir bilgi tabanının durumuyla, bilgisayarın çalışan belleğini karşılaştırır; böylece, her bir zaman periyodunda hesaplanmakta olan olaylar bilinir ve kullanıma uygun yeni durumlar sisteme ilave edilir. Bir çıkarım mekanizmasının yerine getirdiği İkinci işlem ise, hesaplama esnasında, çıkarımların yapılma sırasının kontrolunu sağlamaktadır. Çıkarım mekanizası için alternatif bir yöntem de bir bilgi işlemcisi olabilir. Bir uzman sistemin bilgi işleme elemanı olan çıkarım mekanizması, geliştirilen kurallar ile durumları birleştirmeyi veya yeni sonuçlar elde etmeyi sağlar.

Bir uzman sistem, bir bilgi tabanı (knowledge base) ile sürekli etkileşim halinde çalışır. Bunun için de bir bilgi tabanı, bütün uzman sistemlerin can damarıdır. Bir bilgi tabanı tipik olarak, durumlar ve kurallar olmak üzere, iki tür bilgi içermektedir. Bir bilgi tabanı içindeki durumlar, spesifik bir alandaki değişik olayları göstermektedir ki bunlar, bir uzman sistem çalışmasından önce bilinmelidir. Bir bilgi tabanı içindeki kurallar, basit olarak, önceden belirlenmiş olan

sezgilerin bir çeşididir. Eğer, bir bilgi tabanı, başından sonuna kadar bir insan uzmanla biribirini etkileyerek oluşturuluyorsa, kurallar, karar verme durumundaki bir uzman tarafından kullanılmakta olan olayların veya bir bilgi mühendisinin sezgilerinin bir gösterimi olarak ortaya çıkmasıdır.

Bir uzman sistemin çalışan belleği, eldeki spesifik bir probleme göre sürekli değişim halindedir. Çalışan belleğin içindekiler, durumları içermektedir. Bununla beraber bir bilgi tabanı içindeki durumlar farklıdır. Bu durumlar, görüşmeler esnasında göz önüne alınmış olan spesifik bir problem için belirlenmiştir. Daha spesifik olarak, çıkarım işleminin sonuçları yeni durumlardır ve bu durumlar, çalışan bir bellek içinde depolanmaktadır.

Buradaki son modül ise bir kural ayarlayıcısıdır (rule adjuster). Mevcut uzman sistemlerin büyük çoğunluğunda, bir kural ayarlayıcı, sadece bir kural editörü (yazıcı) olarak hizmet vermektedir. Bir kural editörü, bir uzman sistemin geliştirilme safhasında, bilgi tabanı içerisine bir bilgi mühendisi tarafından, spesifikleştirilmiş bir kuralın girilmesini sağlamaktadır. Daha ileri düzeydedeki uzman sistemlerde, bir kural ayarlayıcısı, kendi yapısı içerisine öğrenmenin ilave edilmesini sağlamak için kullanılabilecektir. Bir kural ayarlayıcısı esas itibariyle, mevcut bilgi tabanını sağlamlaştırabilmektedir.

2.1.1. Uzman Sistemlerde Dış Ortam Arabirimi

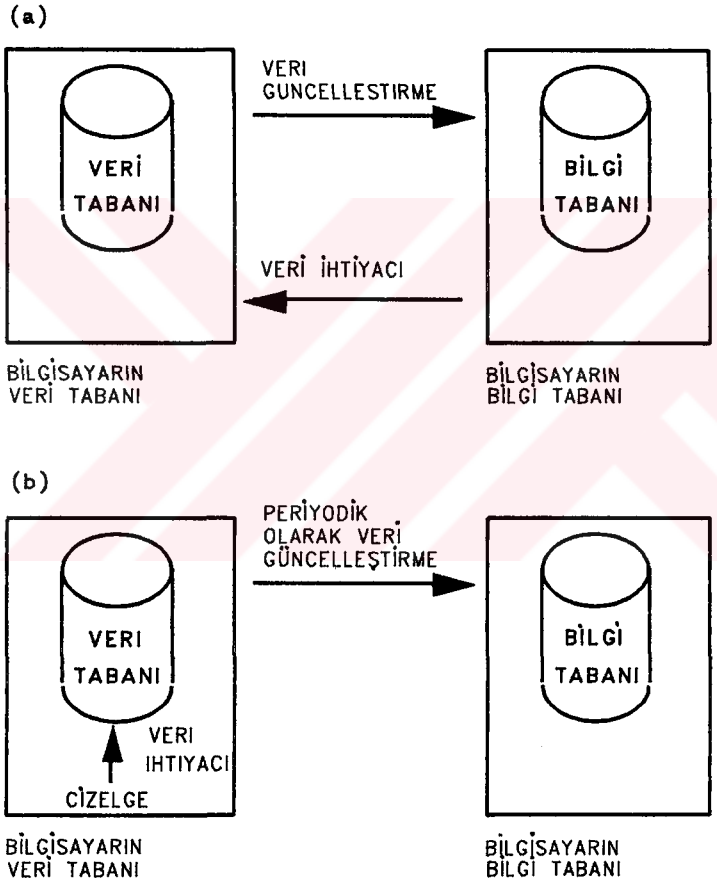
Çoğu zaman bir uzman sistem, dış ortamla birleşik olmalıdır. Bazı durumlarda bir uzman sistem, algılayıcılardan ve diğer aygıtlardan veya dış veri tabanlarından, dış verilere ihtiyaç duyacaktır. Diğer durumlarda ise bir uzman sistem, dış aygıtların faaliyetlerini kontrol etme ihtiyacındadır. Bunun nedeni de uzman sistemin, ortam hakkında daha fazla bilgi elde etme veya ortamdaki değişimleri öğrenme gereksinimidir. Böylece bir uzman sistem, kendisinin önerdiği bir sonucu kontrol edebilmektedir.

Bir uzman sistemde, veri tabanı ile bilgi tabanının entegrasyonu için iki tercih mevcuttur. Eğer bir veri tabanı mevcutsa ve yeniden tasarlanmayacaksa, daha sonraki bir konfigürasyon, genellikle veri tabanından bilgi okuyan bir uzman sistemden oluşacaktır. Uzman sistem daha sonra, veri tabanından okuduğu bilgiyi kendi bilgi tabanı formatı içerisine dönüştürmekte ve ancak bundan sonra bu bilgi üstünde muhakeme yapabilmektedir. Bu durumdaki bir yapı Şekil 2.3'de görülmektedir /5/. Bununla beraber, bir veri tabanının bir bilgi tabanı ile birleşik olarak tasarlandığı bir durumda tasarımcı, veri tabanı uzman sistemin kendi içerisinde olan bir tasarımı da göz önünde tutmalıdır.

Şekil 2.3. Bir Veri Tabanından Bir Bilgi Tabanına Veri Aktarımı

Veri tabanı ile bilgi tabanı arasındaki bir ana entegrasyonda göz önünde tutulan ilişkilerin birbirlerini nasıl etkilediğinin belirlenmesi gerekmektedir. Şekil 2.4a'da görüldüğü gibi bir bilgi tabanı, bir veri tabanından verilere gereksinim duymakta veya şekil 2.4b'de görüldüğü

gibi bir veri tabanı periyodik olarak bir bilgi tabanına veri göndermektedir /5/. Eğer bir bilgi tabanı, verilere ihtiyaç duyuyorsa, bilgi tabanının verilere ne zaman ihtiyaç duyacağının belirlenmesi ve eğer veri tabanı bir bilgi tabanına veri gönderecekse, bu işlemin formatının belirlenmesi gerekmektedir.

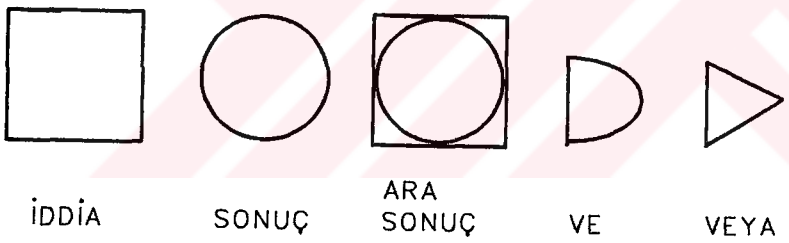


Şekil 2.4. Bir Veri Tabanına Bir Bilgi Tabanının Birleştirilmesi İçin Alternatifler

2.1.2. Uzman Sistemlerde Çıkarım Şebekeleri

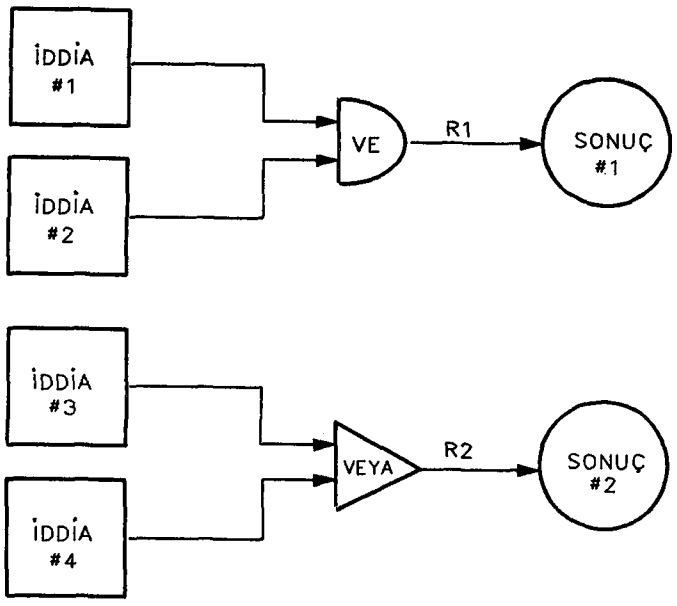
Burada anlatılan grafik yaklaşım, kural tabanı dökümantasyonunda faydalı olabilmektedir. Bu grafik yaklaşım, çıkarım şebekelerinin kullanımına dayanan bir yöntemdir.

Bu çıkarım ağları (şebekeleri), gruplanmakta ve çeşitli semboller ile gösterilmektedir. İlk olarak bir kare, bir iddayı göstermektedir ve bu, belirli bir değer ile bir önermenin atfedilmesini ortaya koymaktadır. Bir daire ise, sonucu göstermek maksadıyla kullanılmaktadır. Etrafı bir kareyle çevrilmiş daire sembolü de bir ara sonucu göstermektedir. Sonuç olarak, iddalar ve ara sonuçlar, mantıksal bağlaçlar ile birleştirilebilmektedir ve bu mantıksal bağlaçlar da AND (VE) veya OR (VEYA) operatörleridir. Bu notasyonlar bir uzman sistemin çıkarım prosedürleri tasarlanılırken kullanılır. Bu çıkarım şekelerinde kullanılan semboller Şekil 2.5'de gösterilmektedir /3/.



Şekil 2.5. Çıkarım şebekeleri sembolleri

Şekil 2.6'da /3/ göz önüne alınan çıkarım şebekesi yukarıda açıklanan notasyonların kullanımını son derece basit olarak göstermektedir. Burada dört iddia ve iki sonuç söz konusudur.



Şekil 2.6. Basit Bir Çıkarım Şebekesi Örneği

Şekil 2.6. incelendiğinde, iki kuralı gösterdiği kabul edilebilir. Bu kurallar şu şekilde belirtebilir :

Kural 1 (K1) **IF [EĞER]** eğer iddia#1 doğru
VE iddia#2 doğru
THEN [İSE] Sonuç#1 elde edilir

Kural 2 (K2) **IF [EĞER]** eğer iddia#3 doğru
VEYA iddia#4 doğru
THEN [İSE] Sonuç#1 elde edilir

Şekil 2.7. ise bir dereceye kadar daha genel bir çıkarım şebekesini göstermektedir /3/. Bu şekilde, A'dan H'ye kadar harfler tam şartları belirtmektedir. Örneğin, A kural 2'de eksiksiz bir önerme durumunu göstermekte kullanılmıştır. Burada H ve G ara sonuçları göstermektedir.

Şekil 2.7'de gösterilen şebekeye karşılık üretim kuralları seti basit olarak aşağıdaki gibi gösterilebilir :

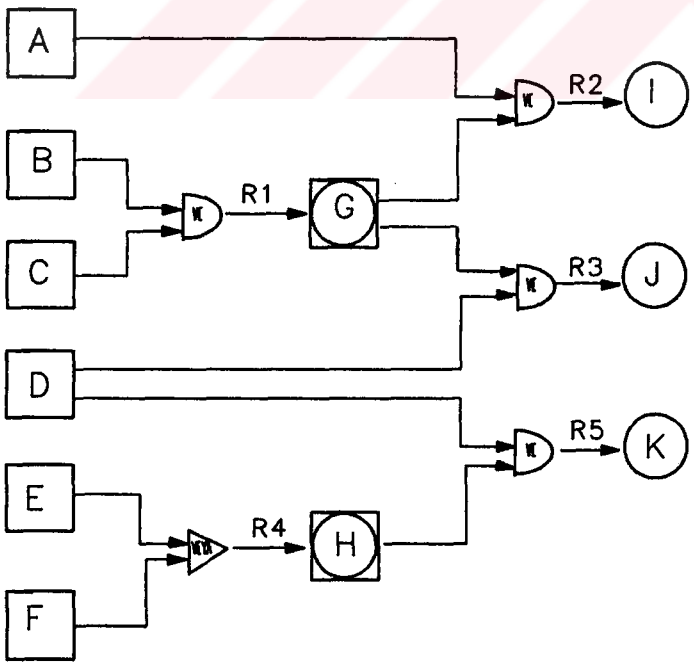
Kural 1 : IF A ve B
THEN G

Kural 2 : IF A ve G
THEN I

Kural 3 : IF D ve G
THEN J

Kural 4 : IF E veya F
THEN H

Kural 5 : IF D ve H
THEN K



Şekil 2.7. Daha Genel Bir Çıkarım Şebekesi

Çıkarım prosesleri, çıkarım şebekeleri yani ağları yardımıyla çok daha rahat ve kolay bir şekilde yapılabilir.

2.2. BİR UZMAN SİSTEMDE BİLGİLERİN ORGANİZE EDİLMESİ

Uzman sistemler, bazı özel problem alanlarında bir uzman insan ile karşılaştırma yaparak, bir performans düzeyine erişmeyi sağlayacak bilgisayar tabanlı problem çözme sistemleridir. Uzman sistemler, oluşturuldukları yapı ve gelişme prosesleri bakımından klasik programlardan farklılık göstermektedir. Klasik programlarda bilgisayara ait karar komutları üzerinde durulmuştur. Uzman sistemlerin geliştirilmesindeki temel problemlerden bir tanesi de uzmanların sahip olduğu ve kullandığı bilginin nasıl temsil edilip kullanılacağıdır. Tipik bir uzman sistemde bilgiler, aşağıda gösterildiği gibi üç kategoride organize edilebilir :

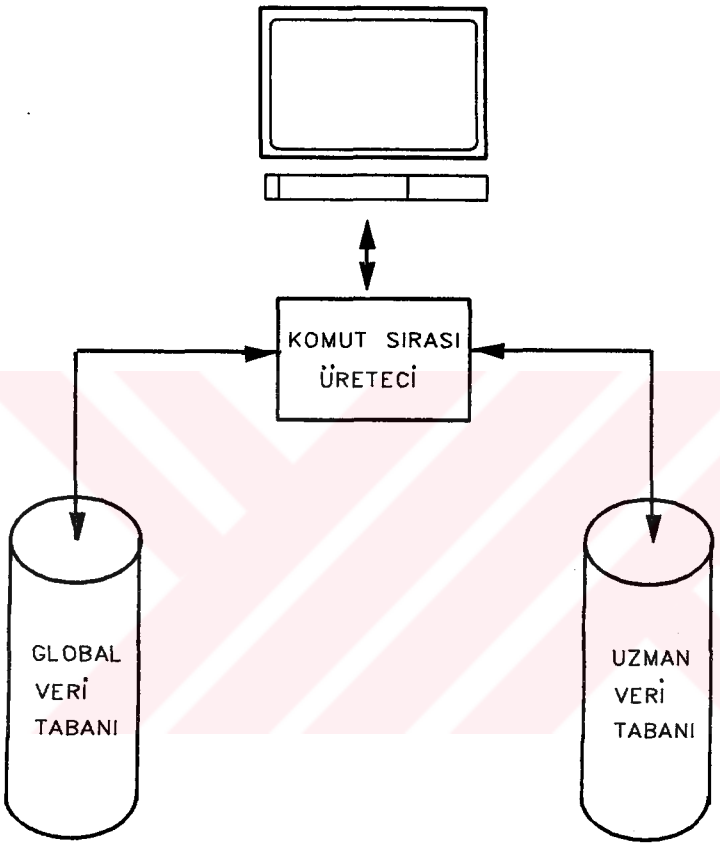
(1) Belirli bir problemi tanımlayarak, girdi verilerini (bazen açıklama bilgisi olarak tanımlanan) ve mevcut çözüm statülerini veya durumlarını izleyen global bir veri tabanı.

(2) Genel problem kontrolu ile ilgili olayları ve sezgisellikleri tanımlayan bir bilgi tabanı. Bu kontrol bilgisi, kurallar formu şeklinde olduğundan, bazen prosedüre dayalı veya ilgili bilgi olarak tanımlanır.

(3) Problem çözüm yaklaşımını tanımlayan veya problemi çözmek için verilerin ve biliglerin nasıl kullanılacağını gösteren bir kontrol veya sonuç yapısı.

Bu nedenle uzman sistemler tipik olarak, veri ve dil yapılarının oluşturulmasını sağlamaktadır. Bu yapılar, açıklama bilgisinin (problem verisi) kontrol edilebilir spesifik problem çözme bilgisinin (ilişkiler) ve Şekil 2.8'de /6/ gösterilen kullanışlı arabirimlerle bu bilginin işletilmesi için kontrol yapılarının (sonuç veya prosedüre yönelik) saklanması ve temsil edilmesi için kullanılmaktadır. Bu yüzden

uzman sistemlerin bir çoğu bilgiyi, üç düzeyde organize etmektedir. Bunlar veri tabanı, bilgi tabanı ve kontrol yapısıdır. Bir uzman sistemin oluşturulmasındaki temel konulardan birisi de, bu verinin ifade edilmesi ve saklanması problemidir. Aşağıdaki incelemelerde, tanımlanan metodolojilerin bir çoğunun tüm bu üç alanda uygulanabileceği ve uygulandığı unutulmamalıdır.



Şekil 2.8. Uzman Sistemlerde Kullanışlı Arabirimler

2.2.1. Veri Düzeyi (Global Veri Tabanı)

Global veri tabanındaki girdi, çözülmekte olan belirli bir problemle ve mevcut olay durumlarıyla ilgili veri veya olaylar formundaki açıklayıcı bir bilgidir. Açıklayıcı bilgiyi sunmak için bir çok alternatif yol vardır. Burada uygun sunuş formunun seçilmesi, uygulamaya bağlı olmaktadır. Temel ilgi odakları, etkin saklama, yeniden

kullanma ve modifikasyonun kolaylığı konuları üzerinedir. Halen kullanılan metotlar aşağıdaki şekilde açıklanmıştır.

1. Önerme Matematiği (Lojik) : Önerme matematiği, çok çeşitli deyimlerin açıklandığı geçerli bir dildir. Burada, basit açıklamalar, " uygun formlu formüller " olarak adlandırılmıştır.
Örneğin " M1 makinası boştur ",

MCH-IDLE (M1) **[MAKİNA-BOŞ (M1)]**

veya

STATUS (M1, IDLE) **[DURUM (M1, BOŞ)]**

olarak ifade edilebilir. Önerme matematiği, matematikten ziyade, geçerli bir mantıkla ilgilidir. Bu matematikle, problemler sembolik olarak ifade edilebildiğinden, yapay zekâda oldukça kullanışlıdır. Önerme matematiğinin gerçek gücü şöyle açıklanabilir. Normal arzulanan lisandaki (türkçe,ingilizce,vb.) cümleler, bir bilgisayar ile işlenerek ve diğer bilgilerle karşılaştırılarak geçerli bir forma dönüştürülür.

2. Yapılar : Yapılar, belirli bir eleman veya olay hakkındaki tüm bilgilerin, beraberce saklandığı veri yapılarıdır. Bir yapı, bir nesne veya bir kavram hakkındaki olaylar ve verilerin toplamıdır. Bu yapılar, bazen verilen bir elemana bağlı olan ispatlar grubu veya kümesidir (birim de denilebilir). Bir çok yapı tabanlı bilgi gösterim planları, farklı eleman tipleri için, farklı yapı tiplerine sahip olma, düşüncesini kapsamaktadır. Bu yapıdaki alanlar veya kanallar, yapı tipleriyle ilgili bilgileri içermektedir. Örneğin üretilecek bir parçanın yapısı, parçanın numarası, kullanılacak malzeme, parça şekline göre sınıflandırma, tezgah rotası, değişik tezgahlardaki işlem zamanları gibi kanallardan oluşmuş bir veri yapısı olabilir.

3. Semantik (Anlamsal) Şebekeler : Açıklayıcı bilgiler grafiksel olarak ifade edilebilir. Anlamsal ağlar (şebekeler) yapılara benzemektedirler, ancak bilgi, tanımlanan eleman etrafında toplanmıştır ve buradaki elemanlar bir grafikteki düğümler ile ifade edilirler. Aynı

bilgi, önerme matematiği şeklinde ifade edilebilse de, bazı tip ilişkiler en iyi grafiksel olarak açıklanabilir. Örneğin, bilgisayar programlarının kod satırları olarak yazıldıklarında anlaşılmaları yüksek seviyeli bir dil de bile çok zor veya olanaksızdır, ancak bir akış diyagramı şeklinde ifade edildiklerinde ise kolayca anlaşılabilirler. Bunun gibi aynı ilişkileri açıklamak için, önerme matematiği kullanılsa bile, uzun bir mantıksal deyim dizileri birbirlerini takip ettikleri için farklı elemanlar arasındaki ilişkiler ve etkileşimler pek açık değildir.

2.2.2. Alan Bilgisi Düzeyi

Bu bilgi düzeyi, belirli tipteki bir problemin, sorun veya alan bilgisi (yani, imalat sistemi, dağıtım sistemi, bilgisayar şebekeleri, vb.) özelliğini tanımlamaktır. Böylece sistem çözüme hazırlanır. Bu bilgi, genelde prosedüre yöneliktir ve bir problemin verilerinin, problemin çözümü için nasıl kullanılacağını açıklar. Bir uzman sistemin gücü, problem alanındaki spesifik bilgiye dayanmaktadır. Bir çok uzman sistemler, en iyi uzmanların sahip olduğu bilginin araştırılması ve daha sonra da bu bilginin kural ağı (şebekesi) oluşturulmak üzere birleştirilmesi ile elde edilen yüzlerce kuralı içermektedir. Veri düzeyi bölümünde belirlenen tekniklerin tümü alan bilgisinin ifade edilmesinde kullanılabilmektedir. Ancak bu bilgilerin ifade edilmesinde bir çok yöntem vardır. Bunlar :

1. Klasik Bilgisayar Programı : Çözüm prosedürü iyi anlaşılırsa ve bir algoritma olarak programlanırsa, çözüme ulaşmak için verileri işletmek üzere klasik bir bilgisayar programı yazılır. Bilindiği gibi bir çok simülasyon modelleri, bu şekildeki (model mantığı) problem veya alan spesifik bilgisini, temsil etmektedir.

2. Model Tabanlı Programlar : Burada, alana bağımlı bilgi matematiksel semboller ile veya model tabanlı programlar şeklinde yazılırlar. Bu programlar, belirli şartlar verilerde bulunduğunda, kontrol yapısı tarafından aktif hale getirilir. Bunların bazıları, durum

araklı araştırma programları şeklindedir. Bu programlar, mevcut durumları diğer durumlara dönüştürmek üzere matematiksel işlemler uygulayarak, bazı son veya amaç durumlarının, ilk durumdaki şeklini bulmayı hedeflemektedirler.

3. Üretim Kuralları : Belirli ilgi alanlarındaki model tabanlı programların bir tipi de üretim kurallarıdır. Bunlar,

IF < KOŞUL > THEN < İLK HAREKET >

formundaki programlardır. Şart genellikle mevcut durum hakkındaki özellikleri test eden önermelerin bir bileşimidir ve ilk hareket mevcut durumu değiştirmektedir.

4. Mantıksal Gösterim : Prosedüre dayalı bilgi ilk sıradaki, önerme mantığı şeklinde ifade edilebilir. Örnegin, **A : B_1 , B_2** şeklindeki bir önerme şöyle ifade edilebilir. B_1 ve B_2 doğruysa, **A** doğrudur veya B_1 ve B_2 nin işletilmesiyle **A**'yı sağlayan bir durumun üretilmesi için, bir prosedür söz konusudur. Bazen bu bilgi **AND / OR (VE / VEYA)** graf formunda saklanabilir. Daha sonra, bir ağaç araştırması yaparak çözüm veren önermeler kümesi bulunduğunda bir çözüm elde edilir. Önerme mantığının en önemli özelliklerinden birisi de geniş bir alan üzerindeki kuralların niteliklerine olanak tanımasıdır.

2.2.3. Kontrol Düzeyi

Kontrol düzeyinde, bilgisayar programı, cevaplandırılacak sorunun ne olacağı hakkında ve daha sonra mevcut problemi çözmek için veri tabanı ve bilgi tabanının ne şekilde kullanılmasına ait kontrol stratejisi hakkında karar verir. Kontrol stratejileri sabit veya deneysel olarak sınıflandırılabilir. Sabit bir kontrol stratejisinde, bir uygulama kuralı seçilir ve daha sonraki kabuller için herhangi bir hazırlık yapmadan uygulanır. Deneysel kontrol stratejilerinde ise, bir kural seçilir, bu kural uygulanır. Ancak tatmin edici bir çözüm

bulunmadığı taktirde (çözüm yoksa), farklı bir kuralın uygulanması için, bu hesaplama noktasına daha sonra dönmek üzere bir çalışma yapılır.

Bir kontrol komut üreteci, problemin çözülmesinde göz önüne alınan adımları organize ve kontrol eder. Bu üreteç, gerekli bilgi ve yazılım modüllerini alarak uygun sıra altında yazılım modüllerini işletir, çıktıyı analiz eder ve uygun bir formda cevaplar verir. Komut üreteci bazen, kullanıcıya nedenleri satır açıklamaları şeklinde ve sonuçların elde edilmesinde kullanılan verileri geri sunar (besler). Buradaki adımlar veya kurallar, sistem durumları **IF - THEN (EĞER - İSE)** kurallarının veya probabilistik dalların birbirlerine bağlanması gibi modellerin kullanılması ile gerçekleştirilir. Kontrol stratejisi, çözülecek problemin bir stratejisidir ve bu strateji için bir çok yaklaşım kullanılmaktadır. Bu yaklaşımlar şu şekilde açıklanabilirler :

1. **İleriye Doğru Bağlantı** : Eğer çözüm ilk veri ve şartlar (mevcut global veri tabanı) kümesinden başlatılırsa ve bazı amaç veya sonuçlara doğru hareket ederse, bu çözüm, model, veri, olay veya öncelikli olarak adlandırılır. Eğer model zamana bağlı değil ise, dal ve sınır, tepeye tırmanma veya ağaç araştırma teknikleri gibi durum aralıklı araştırma metotları bu durumda kullanılabilir.

Bu, teknik bir kurala ait koşul cümlesiyle başlayan ve eylem kurallarını harekete geçiren ve ileri doğru kurallar zinciriyle çalışan işlerde kullanılır. İleriye doğru zincirleme sırasında çıkarım mekanizması **IF (EĞER)** koşullarının bilgi tabanındaki diğer kurallarla uyumunu araştırır. Bu işlemlerle yeni değerler oluşturuldıktan sonra, eylem kuralları uygulanır.

İleriye doğru bağlantı tekniği, başlangıç bilgilerinden, ileriye doğru araştırma yaparak ilerler. Kullanıcılar, bu teknikte istedikleri düzende istedikleri kadar veri sağlarlar. Ancak, kurallar yeterli veri olduğu zaman çalışmaya başlar. Şöyle de açıklamak olanağı vardır. Yeni gelen verilerle kuralların bütün koşulları ya da diğer kuralların sonuçları eşlendiği zaman, sistem işlemeye başlar. Bu durum da, gerçekte

bir problemin sonucunu içeren bir kuralın bulunması ilkesi kullanılmaktadır.

2. Geriye Doğru Bağlantı : Arzulanan sonuç ve amaç durumu önceden biliniyorsa ancak sonucun yöntemi bilinmiyorsa, bu durumda geriye doğru çalışma arzulanır ve model, amaca veya beklentiye yönelik olarak adlandırılır. Hesaplama yapısı nümerikse bu durumda, dinamik programlama metodu kullanılabilir.

Geriye doğru bağlantı, sistemin neyi ispatlamak istiyorsa, onunla başladığı bir çıkarım metodudur. Geriye doğru bağlantı, bir kuralı eşlendiği cümleleriyle bulmak için kuralların geriye doğru arandığı işlerde kullanılır. Kurallar ortaya konan **IF (EĞER)** koşullarını deneyerek ihtiyaç duyulan yeni değerleri bulmak için bir kuraldan diğerine doğru ilerler.

Geriye doğru bağlantı tekniğinde bir problemin özel bir kısmı üzerinde çalışılır. Bu bir probleme ya da alt probleme özgü bir çözümdür. Kuralların sonuçları, yalnız göz önüne alınan problemlere özgü ya da kısmi çözüm olduğunda çalışır. Bir kural kullanılırsa, onun koşulları üzerinde odaklanılan sisteme özgü yeni bir çözüm olur.

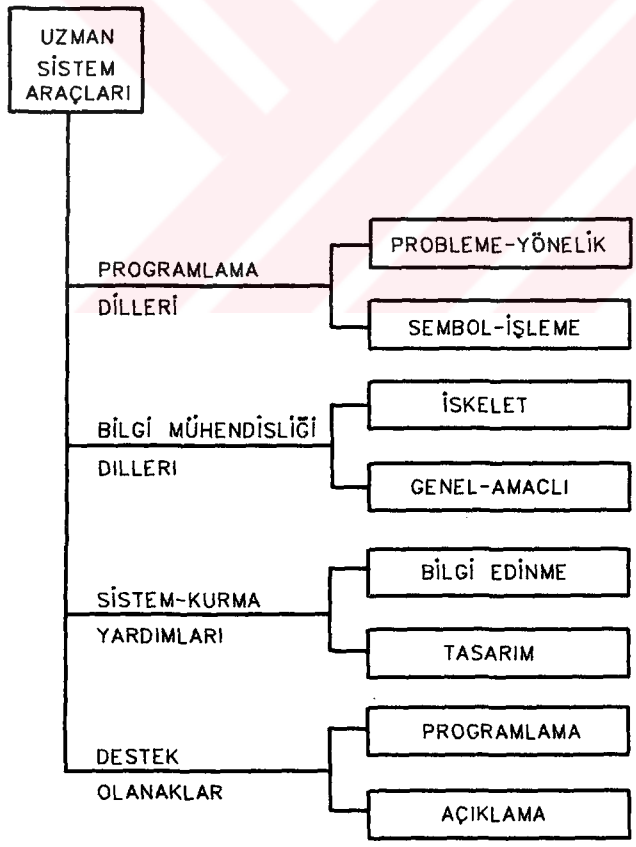
3. Problemin İndirgenmesi : Bu teknikte çözümlenecek problem, birbirinden ayrı olarak çözülecek şekilde alt problemlere bölünür veya ayrılır. Alt problemlerin çözümlerinin birleştirilmesi bazı durumda genel problemin çözümünü sağlar. Daha küçük bir araştırma alanıyla ilgili problemleri açıklayan bu yaklaşım bir amacın elde edilmesi için yapılacak çok sayıdaki ilişkisiz işlerdeki problemlere uygulanabilir. **DENDRAL**, problem indirgeme tekniğini kullanan ileriye doğru bağlantı modeline örnektir. **MYCIN**, geriye doğru bağlantı ve problem indirgeme tekniklerini kullanmaktadır.

4. Kısıt Türetme : Bu problem çözüm tekniğinde mümkün çözüm kümesi, çözümün küçük elemanları, birbirine benzemesi gerektiğini gösteren " lokal (veya bölgesel) kısıtlar " oluşturan kurallar veya

matematiksel işlemlerle, daha kısıtlanır. Kısıtlar tatmin edilmesi gerekli ilişkiler (veya alt amaçlar) olarak gözlenilebilirler.

2.3. UZMAN SİSTEM OLUŞTURMA ARAÇLARI

Uzman sistem araçları bir uzman sistem oluşturma işini basitleştiren programlama sistemleridir. Bir uzman sistem kurma aracı, uzman sistem oluşturmak için kullanılan programlama dili ve destek paketidir. Destek paketi; uzman sistemle kullanıcının etkileşimine yardımcı olan bazı kolaylıklardır. Bunlar gelişmiş grafik araçlar, kolay kullanılabilen programlar, karmaşık kusur giderici olanaklar olarak sıralabilir. Bu arada, şunu belirtmek gerekir ki, uzman sistem kurgu aracı bir uzman sistem değildir. Uzman sistem oluşturmak için kullanılan araçların genel bir görünüşü Şekil 2.9'da verilmektedir /7/.



Şekil 2.9. Uzman Sistem Oluşturma Araçları

Bir programlama dili, bir bilgisayarın çalışmasını yönlendirmek ve kontrol etmek için geliştirilmiş yapay bir dildir. Programlama dilerinin bir sınıfı, bilgi mühendisliği dilleri olarak bilinir. Bilgi mühendisliği dilleri uzman sistemleri oluşturmak ve kusurlarını gidermek için tasarlanırlar. Bilgi mühendisliği dilleri uzman sistemleri oluşturmak için özel olanaklar sağlar, fakat genellikle bilginin nasıl simgeleneceğine değindiklerinden sıradan programlama dillerine oranla daha az esnektirler.

2.3.1. Programlama Dilleri

Uzman sistem uygulamalarında kullanılan programlama dilleri genellikle ya PASCAL, BASIC veya FORTRAN gibi probleme yönelik diller, ya da LISP ve PROLOG gibi sembol işleme dilleridir.

Probleme yönelik bir dil, özel bir sınf için tasarlanmış bir bilgisayar dilidir. Örneğin FORTRAN, cebirsel hesapları etkin biçimde yapabilmek için, COBOL iş kayıtlarını tutmak için tasarlanmıştır. Fortran en çok bilimsel, matematiksel ve istatistiksel problem alanlarına uygulanabilir.

Sembol işleme dili, yapay zekâ uygulamalarında karmaşık kavramları simgelemek ve işlemek için tasarlanmış bir bilgisayar dilidir. Örneğin LISP sembolleri liste yapıları şeklinde işleyecek mekanizmalara sahiptir. Liste yapıları, herbir parçanın ya bir sembol, ya da bir başka liste olabileceği konumda parantezlerle kapatılan bir parçalar kolleksiyonu, karmaşık kavramları simgelemek için kullanılan inşa bloklarıdır. LISP uzman sistem uygulamalarında en çok kullanılan, en yaygın dildir, onu PROLOG izler. Prolog'da diğer programlama dillerinde yer alan fonksiyonlar ve alt programlara karşıt, öncekiler bulunmaktadır.

2.3.2. Bilgi Mühendisliği Dilleri

Bir bilgi mühendisliği dili, uzman sistem geliştirmek için kullanılan ve geniş bir destek çevre içine toplanmış bir uzman sistem oluşturma dillerinden oluşan ileri düzeyde bir araçtır. Bu diller iskelet sistemler ve genel amaçlı sistemler olarak sınıflandırılabilir. Diğer yandan, sistemi kuranın hazır yapım çıkarım mekanizmalarıyla tanımlanan bir kontrol şemasını kullanması gerekmesi nedeniyle, bilgi nühendisliği dillerinin getirdiği esneklik fazla değildir.

İskelet sistemler : Bütün alan bilgisini mevcut bir uzman sistemden çıkararak türetilen ve uzman sistemleri oluşturmak için tasarlanan bilgisayar dilleridir. İskelet sistemler, sistem gelişimini hızlandıran ve kolaylaştıran yapı ve yapım kolaylıkları sağlar, fakat genelleme ve esneklikten yoksundurlar, yalnız sınırlı bir problem alanına uygulanırlar ve uzman sistemi oluşturanın tasarım seçeneklerini büyük ölçüde azaltırlar.

Genel amaçlı sistemler : Kendisini farklı problem alanları ve tiplerine uygulanabilir kılan özellikler taşıyan ve uzman sistemler oluşturmak için tasarlanan bilgisayar dilleridir. Veriye ulaşma ve araştırmada, bir iskelet sistemden daha fazla kontrol sağlarlar.

2.3.3. Sistem Kurma Araçları

Sistem kurma araçları, alan uzmanının bilgisini simgeleyen ve bu bilgiyi sağlayan araçlara sahip programlarla, yapım sırasında uzman sistem tasarımına yardım eden programlardan oluşur. Bu programlar çok zor görevleri yerine getirirler ve birçoğu kullanışlı ve pratik destek geliştirmek için henüz başlangıç niteliğindeki araştırma araçlarıdır. Başlıca iki kategoriye ayrılırlar.

- Tasarım araçları (AGE vb.)
- Bilgi edinme araçları (TEIRESIAS, ROGET vb.)

AGE : Kullanıcıya, inşa blokları gibi, bir uzman sistemin bölümlerini oluşturmak için birleştirilebilen bir bileşenler dizisi sağlar. Her bir bileşen INTERLISP fonksiyonlarının bir kümesidir ve ileriye doğru bağlantı, geriye doğru bağlantı ve bir karatahta mimarisi gibi bir uzman sistemın kafesini destekler. Burada söz edilen karatahta mimarisi (blackboard architecture), karatahta olarak tanımlanan merkezi bir veri tabanı içinde haberleşen bağımsız kural gruplarının kullanımında bilgi tabanının kontrolu ve simgelenmesinin bir yoludur.

TEIRESIAS : Bir alan uzmanından, bir bilgi tabanına, bilgi transferi yapılmasına yardım eder. Kullanıcıya İngilizce'nin sınırlı sözcük grubu içinde etkileşimli olarak kural koyma olanağı tanıyarak, sistem, problem alanı hakkındaki yeni kuralları sağlar. Sistem, kuralları analiz eder, oluşumları ve bütünlüklerini dikkate alarak öneriler yapar ve onların kusurlarını gidermek için kullanıcıya yardım eder.

2.3.4. Destek Olanakları

Destek olanakları düzeltme destekleri ve bilgi tabanı editörleri (yazıcılar) gibi programlamaya yardımcı araçlarla, bitirilmiş bir sistemin kapasitesini arttıran iç girdi-çıktı ve açıklama mekanizmaları gibi araçlardan oluşur. Bunlar genellikle bilgi mühendisliği dilinin bir parçası olarak, özellikle o dille çalışmak üzere tasarlanırlar. Destek çevre aracı, aracın daha etkin ve kolaylıkla kullanımını sağlayacak araçlarla gelen ek yazılım paketleridir.

Kullanıcı programları : Bilgi tabanlı sistemlerin en önemli özelliği kullanıcı ile sistem arasında iki yönlü bir iletişim sağlamasıdır. Kullanıcı programlarının çoğu, doğal dil işleme tekniklerini kullanır.

Düzeltme destekleri : Kural tabanlı uzman sistemleri izleme mekanizmaları, uzman sistemin kabuğu olarak tanımlanmaktadır. Bu kabuk içinde kural tabanı için editör, editör içinde çıkarım mekanizması,

çelişki çözümleyici, yarı otomatik yenileme olanakları, tanımlayıcı bilgiler ve çerçeve bilgileri yer alabilir. Kullanılan bütün kuralların adları ya da numaraları listelenerek ve çağrılan bütün alt programların adları gösterilerek kullanım sistemi işletiminin bir görüntüsüne ulaşılabilir.

Bir çok kural tabanlı uzman sistem, uzman sistem kabuğu kullanılarak geliştirilmiştir. Bu kabuk, kuralları kolayca yazma olanağı sağlar. Genellikle bu kabuk, İngilizce cümleler ve aynı zamanda problem çözme stratejileri sağlar. Bunlar hangi kuralın ne zaman kullanılıcağına karar veren hazır algoritmalardır. Bu strateji kabuğun çıkarım mekanizması olarak bilinmektedir. Bu kabuk, herhangi bir bilgiye dayanmayan kural tabanlı bir uzman sistem olarak ya da uzman sistem çevresinde geliştirilebileceği, bir çevçeve olarak kabul edilir.

Hata paketi mekanizması, kullanıcıya ilerde programın nerede duracağını söyleme fırsatı verir. Böylece kullanıcı bazı hataların oluşmasından önce programın işleyişini durdurabilir ve veri tabanındaki mevcut değerleri gözden geçirebilir.

Otomatik deneme, çözümlerdeki olumsuzlukları ya da hataları açığa çıkaracak, bir dizi belirli problem üzerinde, kullanıcının bir programı otomatik olarak denemesine izin verir.

Girdi-çıktı olanakları : İşleme süresince, bilgi edinme, araçtaki mekanizmaların kendi kendine ve kullanıcının çalışan uzman sistem ile iletişim kurmasına izin verir.

Menüler, sistem tarafından sorulan bilgi girilmek istendiğinde kullanıcının içinden seçerek bilgi girmesi düşüncesiyle sağlanır. Uzman sistem çalıştığı sürece, kullanıcının istediği bilgiyi girmesine olanak verirler.

İşletim sistemine ulaşabilirlik, uzman sistemin çalışırken başka işleri kontrol etmesine ve gözlenmesine olanak verir. Örneğin gerekli

bilgiyi elde etmek için, başka bir dilde yazılmış bir simülasyon programını çalıştırabilir.

Açıklama olanakları : Geriye dönük olarak neden arama, sistemin özel bir duruma nasıl ulaştığını açıklar. Varsayımlara dayanan nedenler bulma, özel bir olay ya da kuralın farklı olması durumunda daha başka hangi sonuçlara ulaşılabileceğini açıklar. Karşıt olaylara dayanarak neden bulma, beklenen bir sonuca neden ulaşılamadığını açıklar.

Bilgi tabanı yazıcıları : Otomatik sayıcılar, bir olaya ilişkin kayıtları ve kullanıcı tarafından yapılan değişiklikleri gözler. Eğer kullanıcı bir kural ekler ya da değiştirirse, yazıcı otomatik olarak, daha sonra başvurulacak bir referans oluşturması için değişim gününü, kuralın ve kullanıcının adını saklar.

Sözdizimi kontrolu, kullanıcının doğru yazım kuralları ve doğru formatla kurallar girmesine yardım etnek için uzman sistem dilinin gramer yapısı konusundaki bilginin editör tarafından kullanılması durumunda önem kazanır. Kullanıcı gramerde olmayan bir kural ya da komut girildiğinde, yazıcı (editör) hatayı yakalar ve yanlış konusunda kullanıcıyı uyarır.

Oluşum kontrolu, girilen veri ve kuralların, sistemde varolan bilgiyle çelişip, çelişmediğini görmek için, sistem tarafından veri ve kuralların anlam yapısını ve anlamsal ağlarının kontrol edilemesinde kullanılır. Bir çelişki söz konusu olursa, editör o çelişkiye neyin neden olduğunu açıklayarak ve onu ortadan kaldırmanın yollarını göstererek, kullanıcının çelişkiyi çözmesine yardım eder.

Bilgi çıkarımı; yazan kullanıcının sisteme yeni bilgi girmesine yardım ettiği zaman kullanılır. Hatta yeni başlayan kullanıcının bile kural ekleyip, değiştirmesine yardım etmek amacıyla ileri düzeyde, detaylı bir açıklama olanağına sahip, oluşum ve sözdizimi kontrolünü birleştirir.

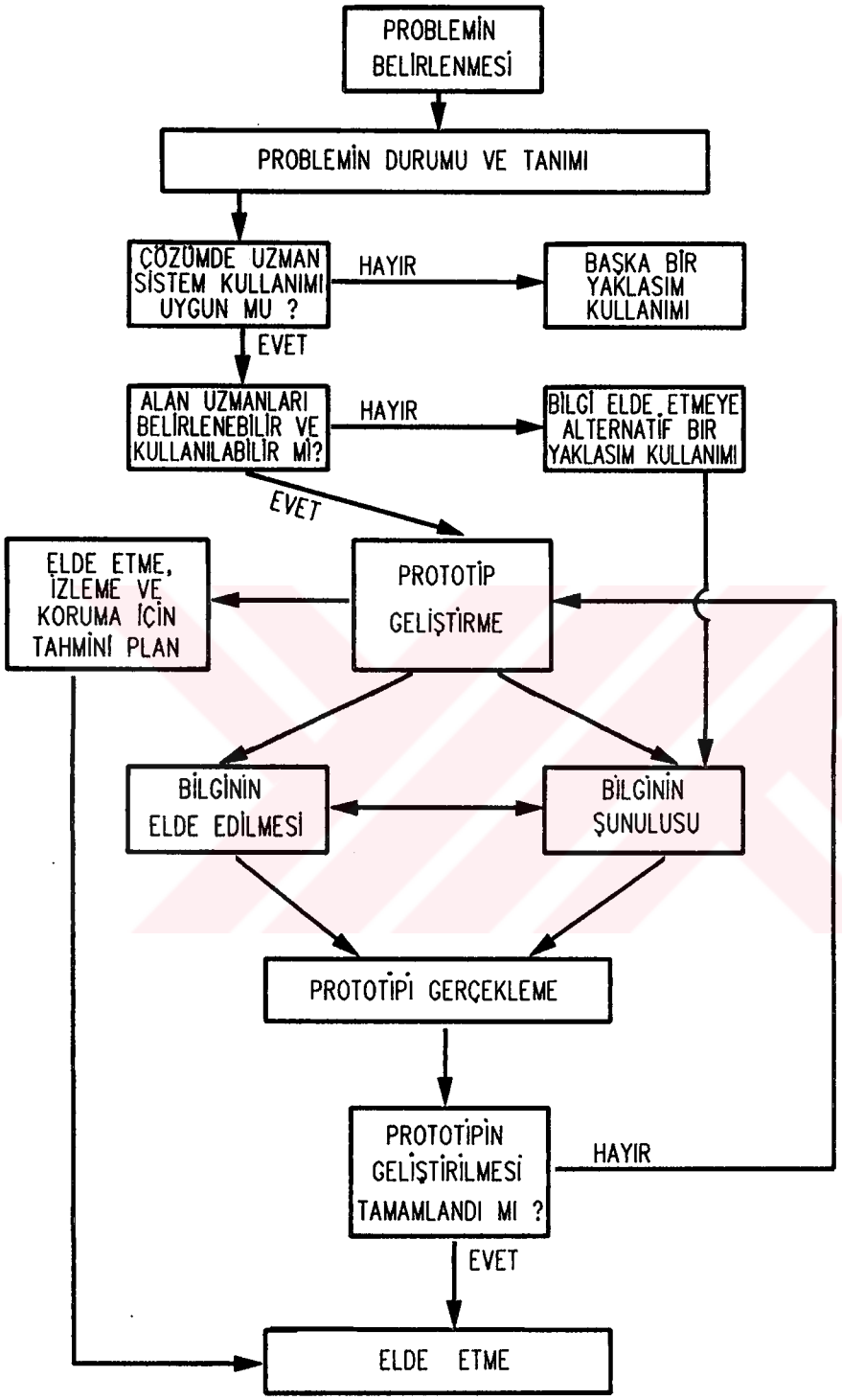
2.4. UZMAN SİSTEMLERİN GELİŞTİRİLMESİ

Bir uzman sistemin geliştirilmesi, diğer tip bilgisayar uygulamalarından farklıdır. ilk fark olarak, özel bir donanım ve yazılımın kullanımının mümkün olması gösterilebilir. Diğer bir fark uzmanın önemidir. Aktif katılım ve istek duymayan bir uzman, geliştirme çabalarında hatalara neden olabilir. Bir diğer fark ise, prototip geliştirmenin önemidir. Başlangıçtaki sisteme, genellikle küçük bir geliştirme uygulanır ve bir önceki versiyon tasfiye edilir ve genişletilir.

Şekil 2.10'da, bir uzman sistemin geliştirilmesi ve elde edilmesinin aşamalarına ait elemanların ve onların birbirleriyle olan ilişkilerinin, akış diyagramı halindeki bir formu gösterilmektedir /3/. İlk olarak, eldeki problem tanımlanmalıdır. Örneğin, problem nedir? O nerede mevcuttur? İlk olarak bu başarılmış olmalıdır. Daha sonraki adım, problemin durumunun detaylandırılarak belgelenmesidir. Bu, hedefleri, amaçları, kısıtları ve problemle ilgili değişkenleri içermektedir. Eğer mümkün ise, problemin durumu matematiksel model formunun içersine yerleştirilmiş olmalıdır. Diğer taraftan, problem elemanları şematik olarak kurulmalıdır.

Bir sonraki çaba, eldeki tamamlanmış bir problem ifadesiyle, uygun bir çözüm metodolojisinin karşılaştırlması problemidir.

Eğer uzman sistem kullanımının gerekli olduğu saptanır ve eğer bir insan uzman da mevcutsa bir sonraki hesaplamaya geçilebilir ve böylece, bilgi elde etme safhası kullanılabilir. Eğer durum böyle değilse, sık sık uygulanan diğer bir yol bir bilgi tabanı elde etmek veya oluşturmaktır. Ancak bundan sonra, bir prototip geliştirme safhasına geçilebilir. Bu safha içinde esas iki ilişki mevcuttur ve gerçekten eş zamanlı bu çabalar, bilgi elde etme ve bilgi sunuşudur. Bundan başka bu safhada asıl uzman sistemi gerçekleştirme ve elde etmek için kararlı bir plan geliştirmek başlar.



Şekil 2.10. Bir Uzman Sistem Geliştirme işleminin Akış Diyagramı

Şekil 2.10'da bilgi elde etme ve bilgi sunuşu işleri paralel olarak gösterilmektedir.

Sonuçta, bir prototip gelişimi tamamlandığı ve tatmin edildiği zaman, daha önceki çabalarla türetilmiş plan kullanılarak, elde etme safhasına geçilebilir. Elde etme, uzman sistemlerin geliştirlmesinin son safhası olarak görülürken, bu safha esnasında bazı revizyonlar gerekebilmektedir.

Bir akış diyagramı olarak verilen uzman sistemlerin geliştirilmesi yapısındaki adımları temel olarak 6 adım olarak sıralanabilinir /2/. Bu 6 adım ve her adımdaki faaliyetler şu şekilde sıralanabilir.

(1) Uygun bir problemin belirlenmesi

(2) Bir prototipin geliştirilmesi

(3) Bütün bir sistemin geliştirilmesi

(4) Sistemin değerlendirilmesi

(5) Sistemin bütünleştirilmesi

(6) Sistemin korunması (muhafaza edilmesi)

Bu adımlar bu şekilde belirlendikten sonra, her adımda yapılması gereken faaliyetler de şöyle sıralabilir.

1. Uygun bir problemin belirlenmesi : Bir uzman sistemin geliştirilmesi içindeki en önemli adım, bir problemin seçilmesidir. Uygun olmayan uygulama sahaları için bir uzman sistem geliştirmek, ya mümkün değildir veya olağan dışıdır. Bu adımdaki aktiviteler şunları içermektedir :

- Bir problem alanı ve spesifik bir işin belirlenmesi,
- Geliştirme işlemine katkıda bulunacak uzman veya uzmanların belirlenmesi,
- Probleme bir deneme yaklaşımının belirlenmesi,

- Çabaların maliyet ve kazandıracaklarının analiz edilmesi,
- Spesifik bir geliştirme planının hazırlanması.

2. Bir prototip sistemin geliştirilmesi : Bilgi mühendisi, bir uzman ile bir uygulama geliştirmeye çalışır. Tipik olarak bir uzman sistem, bir çok versiyonu geliştirildikten sonra eksiksiz olarak tamamlanır. Bir prototip geliştirmek için yaygın olarak aşağıdaki aktiviteler yerine getirilir :

- Uygulama alanı hakkında bilgilenme,
- Performans kriterlerinin ayrı ayrı belirlenmesi,
- Uzman sistem kurma araçlarının seçilmesi,
- Bir başlangıç versiyonunun geliştirlmesi,
- Bir olayın etüt edilmesiyle bu versiyonun test edilmesi,
- Uzman sistemin yeniden gözden geçirilmesi ve yeniden test edilmesi,
- Tamamlanmış bir uzman sistem için detaylandırılmış bir tasarımın geliştirilmesi.

3. Bütün bir sistemin geliştirilmesi : Bir bilgi mühendisi ve uzman, bir prototip sistem geliştirdikten daha sonra, her yönüyle tatmin edilmiş, tamamlanmış bir sistemi geliştirmeye hazırdır. Burada gerekli olan aktiviteler şunlardır :

- Bütün bir sistemin temel yapısının oluşturulması,
- Bilgi tabanının genişletilmesi,
- Bir kullanıcı arabiriminin uyarlanması,
- Sistem performansının izlenmesi.

4. Sistemin değerlendirilmesi : Bir uzman sistemin kullanıma konmadan önce, dikkatli bir şekilde değerlendirilmesi gerekmektedir. Bu adımdaki faaliyetler şunları içermektedir :

- Uzman sistemin performans kriterlerine bağlı olarak karar verme ve yeteneklerinin test edilemesi,

- Arabirimlerinin değerlendirilmesi.

5. Sistemin bütünleştirilmesi: Bir uzman sistemin, bir çalışma ortamına entegrasyonu gerekmektedir. Bunun için de şu faaliyetlerin yerine getrilmesi gerekmektedir :

- Teknoloji transferi için düzenlenmesi,
- Sistemin diğer veri tabanları, aletler veya diğer donanımlarla iletişim kurabilmesi,
- Sistemin kabiliyetlerinin kolay ve hızlı bir şekilde arttırılması.

6. Sistemin muhafazası : Uzman sistemlerin çoğunun zaman boyunca değiştirilmesi gerekmektedir. Bunun nedeni ise, şartların değişmesidir. Durumların, diğer farklı veya yeni durumları ve sezgisel değişiklikler ile şartları dikkate alması gerekmektedir. Uzman sistemlerin kullanımına destek için önemli sonuç sistemin muhafazasının sağlanmasıdır.

3. SİMÜLASYON AMAÇLI UZMAN SİSTEMLER

Günümüzde simülasyon, çok çeşitli alanlarda başarılı bir şekilde uygulanmaktadır. Simülasyon ile son derece karmaşık yapıdaki problemler bile rahatlıkla çözülebilmektedir.

Bir sistemin simülasyonu, bu sistemi temsil edebilecek bir model oluşturma işlemidir. Bu model temsil ettiği sistem üzeride yapılması çok pahalı olan veya yapılması mümkün gözükmeyen işlemlerin yapılmasına olanak verir.

Klasik bir simülasyon çalışmasında simülasyoncu,önce problemin spesifikasyonu oluşturur ve daha sonra da oluşturulan bu spesifikasyona göre, genel amaçlı bir programlama dili veya özel amaçlı bir simülasyon dilini kullanarak, modelin bir bilgisayar programını yapar. Bu yapıldıktan sonra, programın doğruluğu kontrol edilerek program çalıştırılır yani simülasyon işletilir ve sonuçlar elde edilir.

Simülasyon ile uzman sistemlerin bir birleşimi olarak ortaya çıkan simülasyon amaçlı uzman sistemler, bir kullanıcının sadece veri girişi yaparak bir simülasyon çalışmasını gerçekleştirmesine olanak verirler.

Bu bölümde ilk olarak genel bir sistem simülasyonu verildikten sonra uzman simülasyon sistemlerinin yapıları ve geliştirilmeleri üzerinde durulmuştur.

3.1. GENEL SİSTEM SİMÜLASYONU

Simülasyon, gerçek bir prosesin veya sistemin zamana bağlı olarak modelini tanımlayan matematiksel bir modeldir. Simülasyon, ister elle isterse bilgisayar ile yapılsın, bir sistemin yapay kayıtlarının oluşturulması ve gerçek sistemin işletim karakteristikleriyle ilgili

sonuçların elde edilmesinde bu yapay kayıdın incelenmesini kapsamaktadır /8/.

Zamana göre bir sistemin davranışı bir simülasyon modeli geliştirilerek incelenir. Bu model genelde, sistemin işletimiyle ilgili bir kabuller kümesi şeklindedir. Bu kabuller, sistemin ilgili elemanları veya birimleri arasındaki matematiksel, lojik ve sembolik ilişkilere göre ifade edilir. Bir model geliştirildikten ve kabul edildikten sonra, gerçek olaylarla ilgili çok çeşitli soruları araştırmak üzere kullanılabilir. Sistem performansı üzerindeki etkileri tahmin etmek için, öncelikle sistemdeki potansiyel değişiklikler simüle edilmelidir. Simülasyon, böyle sistemler geliştirilmeden önce tasarım safhasındaki sistemleri incelemek için de kullanılabilir. Bu nedenle simülasyon modellemesi, mevcut sistemdeki değişikliklerin etkisini tahmin edici bir analiz metodu ve çeşitli durumlarda yeni sistemlerin performansını tahmin edici bir tasarım metodu olarak kullanılabilir.

Bazı durumlarda, matematiksel metodlarla bile "çözülebilecek" kadar basit bir model geliştirilebilir. Bu şekildeki çözümler, diferansiyel denklemler, ihtimal teorisi veya diğer matematiksel teknikleri kullanarak bulunabilir. Çözüm genelde, sistem performansının ölçüleri olarak adlandırılan bir veya daha fazla nümerik parametrelerden oluşabilir. Ancak, çok sayıdaki gerçek sistemler oldukça karmaşıktır ve bu nedenle bu sistemlerin modellerini matematiksel olarak çözmek hemen hemen imkansızdır. Bu durumlarda, zamana göre sistemin davranışını simüle etmek için nümerik, bilgisayar tabanlı simülasyon kullanılabilir. Simülasyon sonucu, sanki gerçek sistem gözleniyormuş gibi veriler elde edilir. Simülasyon sonucu elde edilen bu veriler, sistemin performans ölçülerini tahmin etmek için kullanılır.

3.1.1. Simülasyonun Kullanım Amaçları

Özel amaçlı simülasyon dilleri, düşük operasyon maliyetleri için yüksek hesaplama kabiliyetleri ve simülasyon metodolojisindeki gelişmeler, simülasyonu yöneylem araştırmasında ve sistem analizinde en

çok kullanılan ve kabul edilen bir metod yapmıştır. Simülasyonun hangi şartlar altında kullanılması gerektiği birçok yazar tarafından incelenmiştir. Bunları genel olarak sınıflandırırsak, simülasyon aşağıdaki amaçlar için kullanılabilir:

1. Simülasyon, karmaşık bir sistemin iç yapısını veya karmaşık bir sistemdeki alt sistemi incelemek için kullanılabilir.

2. Bilgi, organizasyonel ve çevresel değişiklikler simüle edilebilir ve modelin davranışı üzerinde bu değişikliklerin etkileri incelenebilir.

3. Bir simülasyon modelinin tasarımından elde edilen bilgiler, incelenen sistemin geliştirilmesine büyük ölçüde katkıda bulunmaktadır.

4. Simülasyon girdilerini değiştirerek ve sonuçları inceleyerek, hangi değişkenlerin daha önemli olduğu ve değişkenlerin birbirlerini nasıl etkiledikleri hakkında bilgi edilinir.

5. Simülasyon, analitik çözüm metodolojisini destekleyen bir bilgi verici araç olarak kullanılabilir.

6. Simülasyon, uygulamadan önce yeni tasarımlar ve politikalar deneyerek ne olacağını görmek için kullanılabilir.

7. Simülasyon, analitik sonuçları test etmek için kullanılabilir.

3.1.2. Simülasyonun Avantajları ve Dezavantajları

Simülasyon, birçok durumda uygun bir metod olmasına rağmen, sistem analisti, belirli bir durumlardaki metodolojiyi izlemeden önce

simülasyonun avantajlarını ve dezavantajlarını gözönüne almalıdır. Simülasyonun temel avantajlarını şu şekilde sıralayabiliriz:

1. Bir model kurulduktan sonra, bu model, teklif edilen tasarımları veya politikaları analiz etmek için tekrar tekrar kullanılabilir.

2. Girdiler belirsiz olsa da, simülasyon metodları teklif edilen sistemi analiz etmek için kullanılabilir.

3. Gerçek sistemden elde edilen verilere göre simülasyon verilerini elde etme maliyeti oldukça düşüktür.

4. Uygulama açısından, simülasyon metodları analitik metodlara nazaran daha basittir. Bu nedenle, simülasyon metodları kullanan potansiyel kullanıcıların sayısı analitik metodları kullananlardan daha fazladır.

5. Analitik modeller, matematiksel olarak kontrolu sağlamak için, çok basit kabuller kullanırlar; oysa simülasyon modellerinde böyle bir sınırlama yoktur.

6. Bazı durumlarda simülasyon, bir problemin çözümünü veren tek araçtır.

Simülasyonu kullanmadan önce, gözönüne alınması gereken dezavantajlar ise şunlardır:

1. Dijital bilgisayarlara ait simülasyon modellerinin kurulması ve işletmeye alınması aşırı zaman gerektirdiğinden oldukça masraflı olabilir.

2. Genelde, simülasyon modelinin çok fazla sayıda çalıştırılması gerekir ve bu yüzden yüksek bilgisayar maliyetleri ortaya çıkar.

3. Simülasyon, analitik teknikler yeterli olduğu anlarda, nadiren kullanılır. Bu durum, kullanıcıların simülasyon metodları iyi bilmemeleri ve matematiksel temelleri unutmaları sonucu ortaya çıkar.

Schmidt ve Taylor /9/ tarafından belirtilen yukarıdaki ilk iki dezavantaj, özel amaçlı simülasyon dillerini (SIMAN, SLAM, SIMCRIPT, GPSS vb.) ve daha büyük bilgisayarları kullanarak ortadan kaldırılabilir.

3.1.3. Simülasyonun Uygulama Alanları

Simülasyon, çok çeşitli alanlarda uygulama alanına sahiptir. Hillier ve Lieberman /10/, bu tekniğin geniş uygulama alanlarını belirtmek için aşağıdaki örnekleri vermişlerdir:

1. İşletme politikaları ve uygulamalarındaki (bakım kapasitesi, tesislerin, yedek uçakların vb.) değişiklikleri test etmek için bir havayolu şirketi tarafından büyük bir havaalanındaki operasyonların simülasyonu.

2. En iyi trafik akışını belirlemek için, trafik ışıklarının simülasyonu.

3. Optimal tamir personeli sayısını belirlemek için, bakım operasyonu simülasyonu.

4. Bir radyasyon kalkanına yansıyan radyasyonun yoğunluğunu belirlemek için,bu kalkandaki yüksüz parçacıkların akış simülasyonu.

5. Uygulama, kapasite ve tesislerin şekillerindeki değişiklikleri değerlendirmek için, çelik üretim operasyonunun simülasyonu.

6. Ekonomik politika kararlarının etkilerini tahmin etmek için

ekonomi simülasyonu.

7. Savunma ve saldırı silah sistemlerini değerlendirmek için büyük çaplı askeri savaşların simülasyonu.

8. Büyük çaplı dağıtım ve envanter kontrol sistemlerinin tasarımını geliştirmek için bu sistemlerin simülasyonu.

9. Firmanın politikaları ve operasyonlarındaki değişiklikleri değerlendirmek için tüm firmanın genel operasyonlarının simülasyonu.

10. En ekonomik düzeyde, tatmin edici servis sağlamak için, gerekli parça kapasitesini belirlemek için bir telefon iletişim sisteminin simülasyonu.

11. En ideal baraj, elektrik santralı ve sulama işlerinin şeklini belirlemek için, ırmak havza operasyonlarının simülasyonu.

3.1.4. Sistem ve Sistemin Çevresi

Bir sistemi modellemek için, sistem kavramını ve sistem sınırını anlamak gerekir. Sistem, bir amaca ulaşmak için, aralarında düzenli ilişki olan veya birbirlerini etkileyen elemanlar grubu olarak tanımlanabilir. Sisteme bir örnek olarak, otomobil üretimi yapan bir üretim sistemini gösterebiliriz. Makinalar, parçalar ve işçiler bir montaj boyunca beraber çalışarak, yüksek kaliteli araçlar üretirler.

Sistem, bazen kendi dışında oluşan değişikliklerden de etkilenebilir. Bu şekildeki değişiklikler, sistemin çevresinde oluşur. Sistemlerin modellenmesinde, sistemin sınırı ve çevresi hakkında karar vermek gerekir. Bu karar, çalışmanın amacına bağlı olabilir.

3.1.5. Bir Sistemin Elemanları

Bir sistemi anlamak ve analiz etmek için, çok sayıda terim tanımlanmıştır. Bu terimleri, eleman, özellik, faaliyet, durum, olay, sistem içi, sistem dışı olarak sınıflandırabiliriz. Eleman, sistemdeki ilgili unsurdur. Özellik, bir elemanın niteliğidir. Faaliyet, belirli uzunluktaki bir zaman peryodunu temsil eder. Bir banka örneği gözönüne alındığında, müşteriler elemanlardan biri; hesap bakiyeleri bir özellik ve para yatırma bir faaliyet olabilir.

Bir çalışmadaki sistemi oluşturan elemanların kümesi, başka bir çalışmadaki tüm sistemin bir alt kümesi olabilir. Örnek olarak, para çekme ve yatırma için gerekli veznedar sayısının belirlenmesinde yukarıdaki banka incelendiğinde, sistem, veznedarlar ve hatta bekleyen müşterilerin bulunduğu kısım olarak tanımlanabilir. Eğer çalışma özel veznedar (seyahat çeki veren veznedar, ticari çek veren veznedar gibi) sayısını belirlemek üzere genişletilirse, sistemin tanımı genişletilebilir.

Sistem durumu, çalışmanın amacına bağlı olarak, herhangi bir anda sistemi tanımlamak için gerekli değişkenler kümesi şeklinde tanımlanabilir. Banka örneğindeki durum değişkenleri, meşgul veznedarların sayısı, hatta bekleyen veya servis gören müşterilerin sayısı ve bir sonraki müşterinin geliş zamanıdır. Olay ise, sistemin durumunu değiştirebilen ani oluşumdur. Sistem içi terimi, bir sistem içinde ortaya çıkan faaliyetleri ve olayları tanımlamak için; sistem dışı terimi ise, sistemi etkileyen çevredeki olayları ve faaliyetleri tanımlamak için kullanılmaktadır. Banka örneğinde, müşterinin gelişi bir sistem içi olay ve müşteri servisinin tamamlanması bir sistem dışı olaydır.

Tablo 3.1, birçok sistem için elemanlar, özellikler, faaliyetler, olaylar ve durum değişkenleri örneklerini göstermektedir /8/.

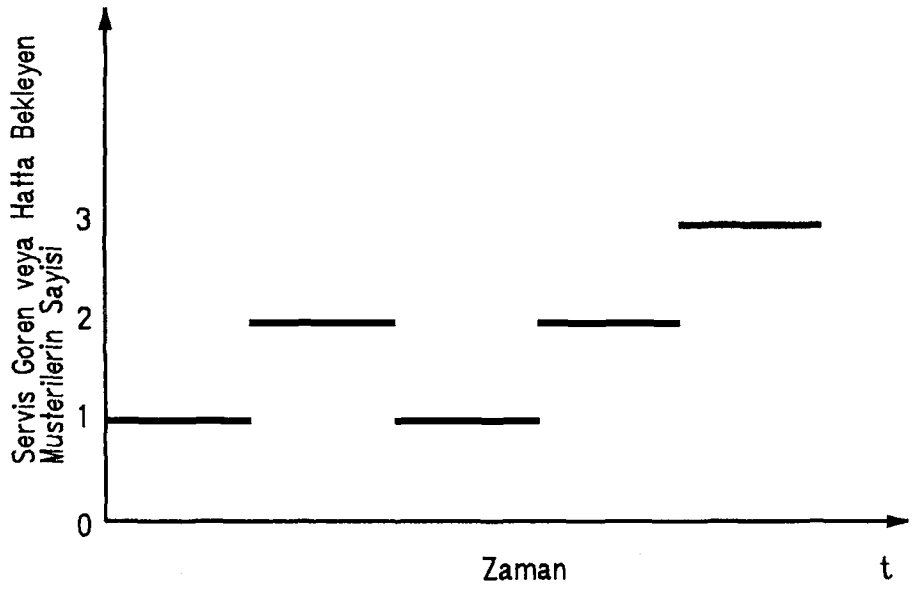
Tablo 3.1. Sistem ve Sistem Bileşenleri Örnekleri

Sistem	Elemanlar	Özellikler	Faaliyetler	Olaylar	Durum Değişkenleri
Banka	Müşteriler	Hesap Bakiyesi Kontrolu	Para Yatırma	Geliş; Ayrılış	Meşgul Veznedar Sayısı; Bekleyen Müşteri Sayısı
Tren Yolu	Trenler	Kalkış yeri; Varış Yeri	Seyahat	İstasyona Geliş	Her İstasyonda Bekleyen Tren Sayısı;Transit Tren Sayısı
Üretim	Makinalar	Hız;Kapasite; Bozulma hızı	Kaynak; Birleştirme	Bozulma	Makinaların Durumu (Meşgul,boş veya bozuk)
İletişim	Mesajlar	Uzunluk; Hedef	Arama	Hedefe Varış	Aranacak numara
Envanter	Ambar	Kapasite	Malzeme çekme	Talep	Envanter Düzeyi

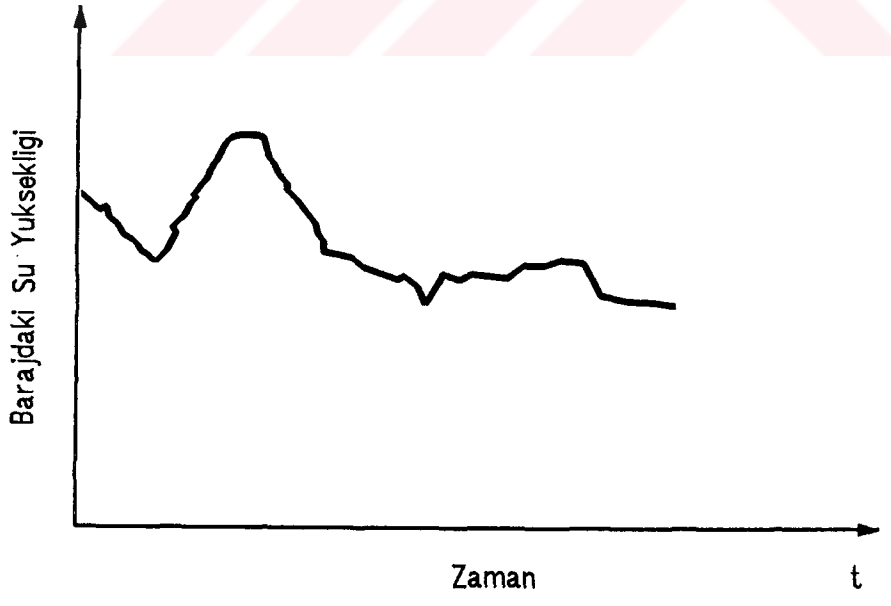
3.1.6. Kesikli ve Sürekli Sistemler

Sistemler, kesikli veya sürekli olarak sınıflandırılabilir. Pratikte çok az sayıda sistem tamamen kesikli veya süreklidir. Çünkü, tek bir değişim tipi sistemde etkili olduğundan, bir sistemi kesikli veya sürekli olarak sınıflandırmak mümkündür. **Kesikli sistem**, durum değişkeninin (değişkenlerinin) zamana göre sadece kesikli nokta kümesinde değiştiği bir sistemdir. Banka, kesikli sisteme bir örnek olarak gösterilebilir. Çünkü, durum değişkeni (bankadaki müşteri sayısı), bir müşteri geldiğinde veya müşteriye verilen servis tamamlandığında değişmektedir. Şekil 3.1, müşteri sayısının zamana göre sadece kesikli noktalarda nasıl değiştiğini göstermektedir /8/.

Sürekli sistem ise, durum değişkeninin (değişkenlerinin) zamana göre sürekli olarak değiştiği bir sistemdir. Sürekli sisteme örnek olarak, bir barajdaki su yüksekliği gösterilebilir. Yoğun yağmur yağışından sonra sular, barajın arkasındaki göle akmaktadır. Su baskınını kontrol etmek ve elektrik üretmek için sular barajdan çekilmelidir. Burada, buharlaşma da su düzeyini azaltmaktadır. Şekil 3.2, durum değişkeninin (barajdaki su yüksekliği) bu sürekli sistemde nasıl değiştiğini göstermektedir /8/.



Şekil 3.1. Kesikli Sistem Durum Değişkeni



Şekil 3.2. Sürekli Sistem Durum Değişkeni

3.1.7. Bir Sistemin Modeli

Sistemin elemanları arasındaki ilişkileri anlamak veya yeni bir politika altında sistemin nasıl davranacağını tahmin etmek için bir sistemi incelemek gereklidir. Bir sistemi incelemek için, sistemin kendisi ile deney yapmak mümkündür. Ancak, bu durum her zaman geçerli değildir. Yeni bir sistem oluşmadan önce, sistem teorik formda veya tasarım safhası şeklindedir. Mevcut sistem ile deney yapmak pratik bir yöntem değildir. Örnek olarak, enflasyon altında çalışanların etkilerini belirlemek için işsiz kişilerin oranını iki katına çıkarmak mümkün değildir. Banka örneğinde, bekleme hattı uzunluğunun etkisini incelemek için veznedar sayısını azaltmak, müşterileri memnun etmez ve bu nedenle müşteriler, hesaplarını rakip bankaya yatırırlar. Sonuç olarak, sistemlerin incelenmesi bir sistemin modeli ile yapılmaktadır.

Model, sistemi incelemek üzere sistemin örneği olarak tanımlanır. Birçok çalışma için, bir sistemin tüm detaylarını gözönüne almak gereksizdir; bu nedenle model, sadece sistemin bir yardımcısı değil, aynı zamanda sistemin basit bir şeklidir. Diğer yandan model, gerçek sistemden alınan sonuçlara izin verecek şekilde detaylı olmalıdır. Değişik araştırmalar için, aynı sistemin farklı modelleri de kurulabilir.

3.1.8. Model Tipleri

Modeller, matematiksel veya fiziksel modeller olarak sınıflandırılabilir. Matematiksel model, sistemi temsil etmek için sembolik notasyon ve matematiksel denklemleri kullanır. Simülasyon modeli, belirli tipte bir matematiksel sistem modelidir.

Simülasyon modelleri, statik veya dinamik, deterministik veya stokastik ve kesikli veya sürekli olarak sınıflandırılabilir. Monte Carlo simülasyonu olarak bilinen **statik simülasyon** modeli, zamanın belirli bir anındaki sistemi temsil etmektedir. **Dinamik simülasyon** modelleri ise, zamana göre değişen sistemleri temsil etmektedir. Saat

8.00 ile 17.00 arasında çalışan bir bankanın simülasyonu, dinamik simülasyona bir örnek olarak verilebilir.

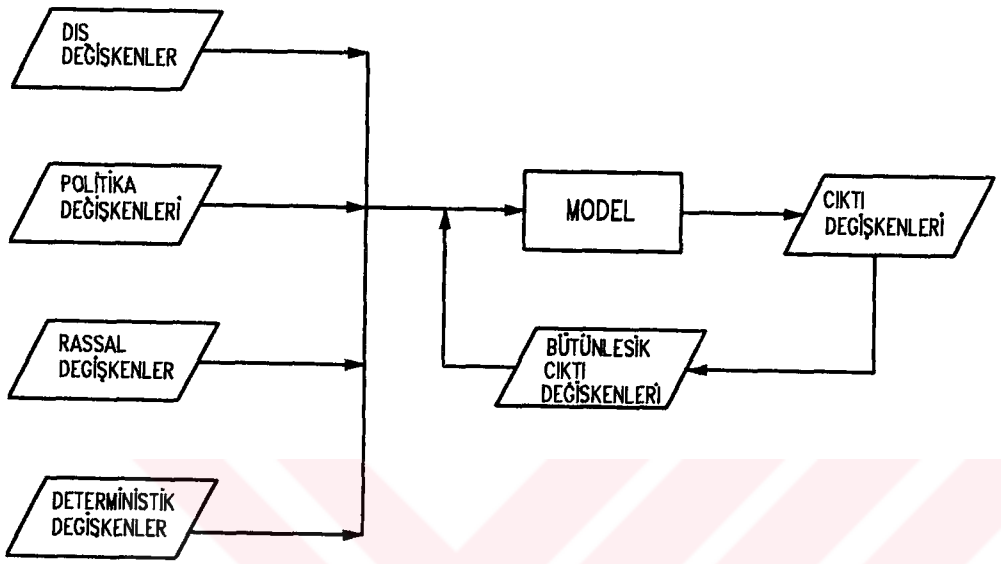
Rassal değişken içermeyen modeller, deterministik modeller olarak sınıflandırılır. Deterministik modeller, tek bir çıktı kümesi veren girdi kümesine sahiptir. Deterministik modellere örnek olarak, tüm hastaların randevu saatlerine göre geldikleri dişçi muayenehanesini gösterebiliriz. Stokastik simülasyon modeli ise, girdi olarak bir veya daha fazla rassal değişkeni gözönüne almaktadır. Rassal girdiler, rassal çıktılar oluştururlar. Çıktılar rassal olduğu için, bu çıktılar modelin gerçek karakteristiklerinin tahminleri olarak gözönüne alınabilir. Bir bankanın simülasyonu, genellikle rassal gelişler arası zamanları ve rassal servis zamanlarını kapsamaktadır. Bu nedenle, stokastik bir simülasyonda, çıktı ölçüleri (bekleyen ortalama müşteri sayısı, bir müşterinin ortalama bekleme zamanı) sistemin gerçek karakteristiklerinin istatistiksel tahminleri olarak ele alınırlar.

Kesikli ve sürekli modeller, analog bir şekilde tanımlanmıştır. Ancak, kesikli bir simülasyon modeli kesikli bir sistemi modellemek için, sürekli bir simülasyon modeli de sürekli bir sistemi modellemek için her zaman kullanılmaz. Ayrıca simülasyon modelleri, kesikli ve sürekli şekilde karma modeller olabilirler. Kesikli veya sürekli (veya hem kesikli hem sürekli) simülasyon modelini kullanma seçimi, sistem karakteristiklerinin ve çalışma amacının bir fonksiyonudur. Bu nedenle, her mesajın karakteristiği ve hareketinin çok önemli olduğu bir iletişim kanalı kesikli olarak modellenebilir. Aynı şekilde, kanaldaki mesajların akışı önemli olduğunda, sürekli simülasyon kullanarak sistem modelleme daha uygun olmaktadır. Burada ele alınan modeller, kesikli, dinamik ve stokastik yapıdadır.

3.1.9. Simülasyon Modelindeki Değişkenler

Naylor ve Gatts'e /11/ göre, simülasyonda kullanılan değişkenler, çıktı, bütünleşik çıktı, dış, politika veya rassal değişkenler olarak sınıflandırılabilir. Birçok simülasyon modelleri probabilistik

olmadığından, burada deterministik değişkenleri de gözönüne almak gerekmektedir. Şekil 3.3, Simülasyon modellerindeki bu değişkenlerin rolünü göstermektedir /2/.



Şekil 3.3. Simülasyon Modellerinde Kullanılan Değişkenler

Buradaki değişkenleri sırayla inceleyelim:

1. Çıktı Değişkenleri: Çıktı değişkenleri,analiste ve/veya yönetime gerekli ara sonuçları veya son bilgileri sağlamaktadır. Bir üretim modelinde, satılan mamullerin maliyeti ve envanterdeki yarı mamulle ilgili çıktı değişkenleri olabilir.

2. Bütünleşik Çıktı Değişkenleri: Daha önce belirtildiği gibi, simülasyon modelleri dinamiktir ve zamana göre sistemin davranışını tanımlamaktadır. Sonuçta, zaman sonunda sistemin durumu, bir sonraki zaman peryodu boyunca sistemin analizinde gerekli bir girdi olmaktadır. Bütünleşik çıktı değişkenleri, zaman peryodundan zaman peryoduna sistemin durumu hakkında bilgi vermektedir.

Bütünleşik çıktı değişkenlerine (veya seri ilişkili değişkenler) ait örnekler, üretim, finans ve pazarlama modellerinde ortaya çıkmaktadır. Peryot sonu bitmiş mamul envanteri, gelecek peryodun gelecek mamul envanterinin belirlenmesinde gerekli bir girdidir.

3. Dış Değişkenler: Sistemin davranışını etkileyen bu değişkenler, dış değişkenler olarak belirtilmektedir. Dış değişkenlerin yapısına örnek olarak şu önermeyi kullanabiliriz: " X, Y'den etkilenir, ancak Y, X'den etkilenmez". Dış değişkenler çoğunlukla, sistem dışı, çevresel veya kontrol edilemeyen değişkenler olarak adlandırılırlar. Bir üretim modelinde, grevler ve hammadde eksikliği, dış değişkenler olarak gözönüne alınabilir.

4. Politika Değişkenleri: Birçok sistem, yönetimin incelediği kontrol faktörlerini içermektedir. Örnek olarak, bir organizasyonda yönetim, hangi şartlar altında hangi makinanın alınacağına karar verebilir. Bu şekildeki faktörler, simülasyon modellerindeki politika değişkenleri olarak gözönüne alınırlar. Bu faktörler, yönetimin kararlarına ve kontroluna göre değiştiği için, politika değişkenleri bazen karar veya kontrol edilebilir değişkenler olarak da adlandırılırlar. Birçok simülasyon modellerinde politika değişkenlerinin hangi sistem politikasında daha etkin olduğu araştırılır.

5. Rassal Değişkenler: Bazı sistemlerin davranışını belirlemek için, rassal veya probabilistik yapısını anlamak gerekir. Rassal değişkenler, bu role hizmet etmektedir.

6. Deterministik Değişkenler: Birçok sistem, probabilistik olmasına rağmen, analist, deterministik bir modelin kullanıcının amaçları için yeterli bilgileri sağlayacağını bilir. Bu durumda, sadece tek değerli tahminler gerektiren deterministik değişkenler kullanılır. Sistemdeki varyasyon miktarının çok az olduğu probabilistik modellerde, bazen deterministik değişkenler de kullanılmaktadır.

3.1.10. Kesikli Olay Sistem Simülasyonu

Burada yapılan çalışmalar, kesikli olay sistem simülasyonuna dayanmaktadır. Kesikli olay sistem simülasyonu, durum değişkenlerinin, zamana göre sadece kesikli noktalarda değiştiği sistemlerin modellenmesidir. Simülasyon modelleri, analitik metodlar yerine nümerik metodlarla analiz edilirler. **Analitik** metodlar, modeli "çözmek" için indirgenmiş matematiksel analizi kullanmaktadır. Örnek olarak, bazı envanter modellerdeki minimum maliyet politikasını belirlemede diferansiyel denklemler kullanılmaktadır. **Nümerik** metodlar ise, matematiksel modelleri "çözmek" için hesaplama prosedürleri kullanırlar. Nümerik metodların kullanıldığı simülasyon modellerinde, modellerin çözümü yerine işletimi sağlanır. Yani, model kabullerine bağlı olarak sistemin yapay bir kayıdı oluşturulur ve gerçek sistem performans ölçülerini tahmin ve analiz etmek için gözlemler elde edilir. Gerçek simülasyon modelleri oldukça büyük ve saklanan - işletilen bilgilerin sayısı çok fazla olduğundan, modellerin işletimi bilgisayarlar tarafından yapılmaktadır. Ancak, küçük modellerin elle simülasyonu daha kolay anlaşılmaktadır.

3.1.11. Simülasyon Çalışmasındaki Adımlar

Şekil 3.4, tam ve hassas bir simülasyon çalışmasının yapılmasında modelciye yardım eden adımları göstermektedir /8/. Şekil 3.4'de her sembolün yanında bulunan sayılar, aşağıdaki konuları ifade etmektedir. Buna göre, simülasyon çalışmasındaki adımları şöyle sıralayabiliriz:

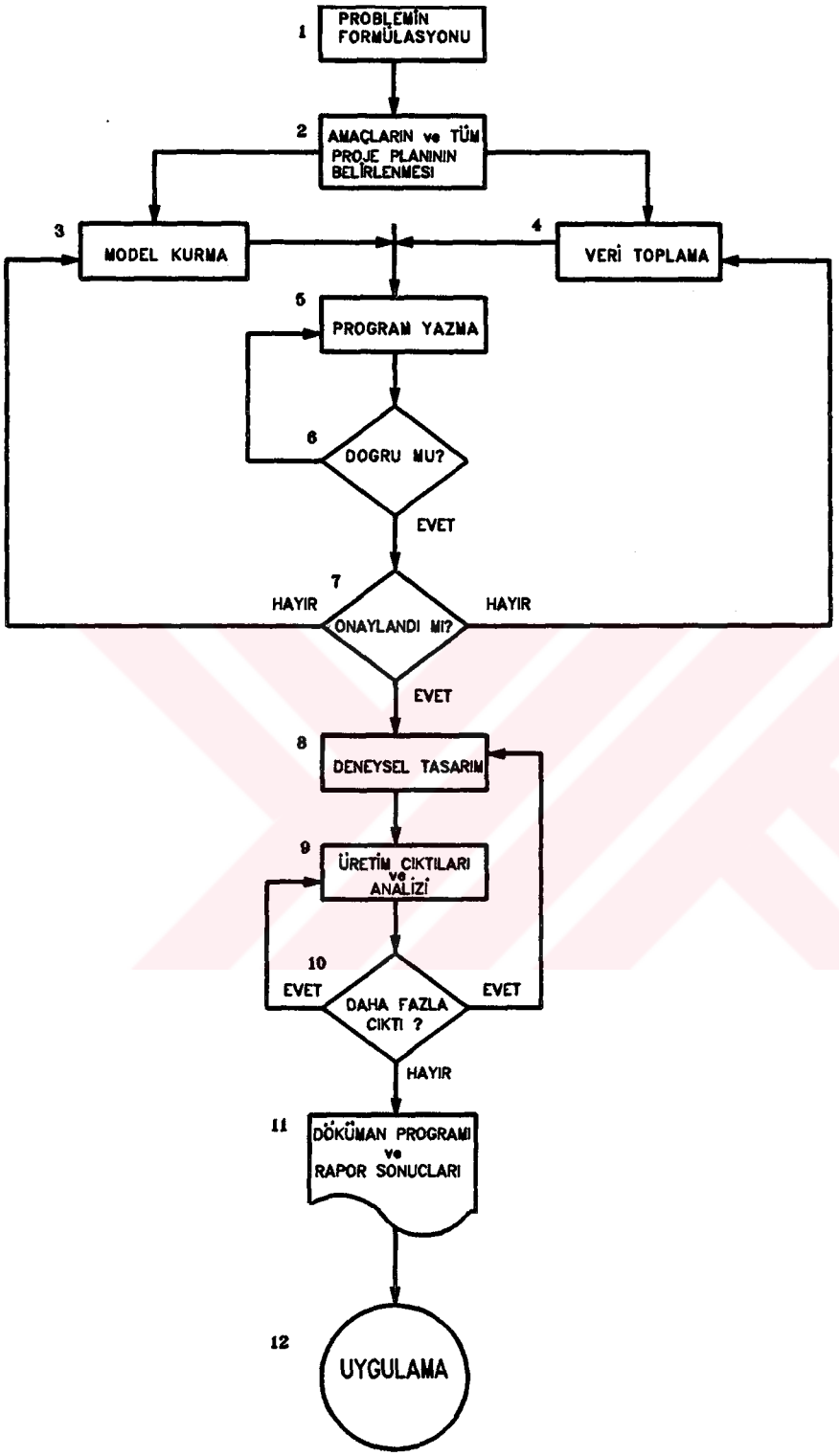
1. **Problemin Formülasyonu:** Her çalışma, problemin tanımlanmasıyla başlar. Eğer problem, yöneticiler veya diğer kişiler tarafından ortaya çıkarılmışsa, analist, tanımlanan problemi açıkça anlamalıdır. Problemin tanımı analist tarafından yapılırsa, yöneticiler veya problem sahipleri formülasyonu anlamalı ve aynı noktada karar kılmalıdırlar. Şekil 3.4'de gösterilmemesine rağmen, çalışma ilerlerken problemin yeniden formüle edilme durumu ortaya çıkmaktadır.

2. Amaçların ve Tüm Proje Planının Belirlenmesi: Amaçlar, simülasyon tarafından cevaplandırılacak soruları belirtmektedir. Bu noktada simülasyonun, belirtilen amaçlar ve formüle edilen problem için uygun bir metodoloji olup olmadığına karar verilmelidir. Simülasyonun uygun metod olduğuna karar verdikten sonra tüm proje planı, gözönüne alınacak alternatif sistemlerin tanımını ve bu alternatiflerin etkinliklerini değerlendirecek bir metodu kapsamalıdır. Bu plan ayrıca, çalışan kişilerin sayısı, çalışmanın maliyeti ve her kademenin sonundaki tahmini sonuçlarla, çalışmanın her safhasını gerçekleştirmek için gerekli gün sayısıyla ilgili çalışma planlarını da kapsamaktadır.

3. Model Kurma: Bir sistemin modelini kurma, bilimin sanat yönünü oluşturmaktadır. Her örnekte, başarılı ve uygun modelleri kurmak için gerekli kuralların bulunması zor olmasına rağmen, izlenecek bazı genel kurallar bulunmaktadır. Modelleme sanatı, problemin temel özelliklerini belirlemek, sistemi karakterize eden temel kabulleri seçmek-değiştirmek ve daha sonra kullanışlı uygun verileri elde edene kadar modeli genişletmek için bir özellik ile tamamlanır. Böylece, basit bir modelle işe başlamak ve modeli daha sonra geliştirmek en uygun adımdır. Ancak, modelin gelişmesi, modelin temel amaçlarına aykırı olmamalıdır. Bu prensibin bozulması, sadece model kurma ve bilgisayar masraflarını arttıracaktır. Model ve gerçek sistem arasında birebir karşılaştırma yapmak gereksizdir. Sadece, gerçek sistemin niteliği önemlidir.

Model kurmada, model kullanıcısını da gözönüne almak gerekmektedir. Model kullanıcısını gözönüne alarak , hem oluşan modelin kalitesi hem de modelin uygulanmasında, modeli kullanan kişinin güveni artar.

4. Verilerin Toplanması: Modelin kurulması ve gerekli verilerin toplanması arasında sabit bir ilişki bulunmaktadır. Modelin zorluk derecesi değişirken, gerekli veri elemanları da değişmektedir. Bir simülasyonun gerçekleştirilmesinde veri toplama, gerekli toplam zamanın büyük bir kısmını oluşturduğundan, veri toplamaya genellikle model kurmanın ilk safhalarında mümkün olduğu kadar erken başlamak gerekmektedir.



Şekil 3.4. Bir Simülasyon Çalışmasındaki Temel Adımlar

Çalışmanın amaçları, toplanacak veri türü üzerinde önemle durmaktadır. Banka örneğinde amaç, değişik veznedar sayısına göre bekleme hattının uzunluğunu öğrenmek ise, gerekli veri türleri, gelişler arası zamanların (günün farklı zamanlarında) dağılımı, veznedarlara ait servis-zaman dağılımı ve değişen şartlar altında bekleme hatlarının uzunluklarına bağlı önemli dağılımlar olabilir. Bu son veriler, simülasyon modelinin geçerliliğini kabul etmek için kullanılmaktadır.

5. Program Yazma: Gerçek sistemler, çok sayıda bilgi depolaması ve hesabı gerektiren modeller şeklinde ortaya çıktığından model, dijital bir bilgisayarda programlanmalıdır. Modelleyici modeli, FORTRAN, PASCAL gibi genel amaçlı bir dilde mi, yoksa GPSS, SIMSCRIPT veya SLAM gibi özel amaçlı bir simülasyon dilde mi programlayacağına karar vermelidir. Genel amaçlı diller, daha uzun bir yazılım süresi gerektirirler. Ancak, özel amaçlı dillere göre daha hızlı çalışırlar. Bununla beraber, özel amaçlı dillerde yazılım (ve hata bulma) çok çabuk olduğundan, model kurucuların çoğunluğu bu dilleri kullanmaktadır.

6. Doğru mu?: Doğruluk, simülasyon modeli için hazırlanan bilgisayar programıyla ilgilidir. Bilgisayar programı doğru çalışıyor mu? Çok karmaşık modellerde, iyi bir hata ayırıcı olmadan tüm modeli başarılı bir şekilde programlamak oldukça zordur. Girdi parametreleri ve modelin mantıksal yapısı, programda doğru bir şekilde yazılmış ise, program doğru olarak kabul edilir.

7. Uygun mu?: Uygunluk, modelin gerçek sistemi doğru bir şekilde temsil ettiğini belirlemektir. Uygunluk, modelin ayarlanması, modelin ve gerçek sistemin davranışlarının karşılaştırılması ve bu ikisi arasındaki farklılıkları kullanarak elde edilir. Bu proses, model doğru olarak kabul edilene kadar tekrar edilir. Yukarıda bahsedilen banka örneğinde, mevcut durumlarda bekleme hatlarının uzunluğuyla ilgili veriler toplanmıştır. Simülasyon modeli, bu sistem ölçüsüne cevap verir mi? Bu, uygunluk sorularından birisidir.

8. Deneysel Tasarım: Burada, simüle edilecek alternatifler belirlenmelidir. Çoğunlukla, hangi alternatifin simüle edileceği kararı,

tamamlanmış ve analiz edilmiş çıktıların bir fonksiyonu olabilir. Simüle edilecek her sistem tasarımı için, ilk peryodun uzunluğu, simülasyon çıktılarının uzunluğu ve her çıktıda verilecek cevapların sayılarıyla ilgili kararlar verilmelidir.

9. Üretim Çıktıları ve Analizi: Üretim çıktıları ve onların analizi, simüle edilen sistem tasarımlarının performans ölçülerini tahmin etmek için kullanılmaktadır.

10. Daha Fazla Çıktı?: Tamamlanan çıktı analizine bağlı olarak analist, ilave çıktılara ihtiyaç olup olmadığına ve bu ilave denemelerin hangi tasarımda yapılacağına karar vermelidir.

11. Döküman Programı ve Rapor Sonuçları: Çok sayıdaki nedenler için bir dökümantasyon gereklidir. Eğer program, aynı veya farklı analist tarafından yeniden kullanılacaksa, programın nasıl işleyeceğini anlamak gerekebilir. Burada, programın doğru olduğu kabul edilir ve böylece modeli kullanıcılar ve amaç belirleyiciler analiste bağlı olarak karar verebilirler. Aynı zamanda, program aynı veya farklı analist tarafından değiştirilecekse, bu durum uygun bir dökümantasyonla yapılabilir. Hatalı programla yapılan bir tecrübe, bu önemli adımın gerekliliğini analiste inandırmak için yeterlidir.

Bir modelin dökümantasyonunun hazırlanmasının diğer bir nedeni de, model kullanıcılarının performans girdi ve çıktı parametreleri arasındaki ilişkileri belirlemek ve bazı performans çıktı ölçülerini "optimize" eden girdi parametrelerini belirlemek için model parametrelerini değiştirmek istemeleridir.

Tüm analizin sonuçları, açık ve kısa olarak rapor edilmelidir. Bu, model kullanıcılarına (burada, karar vericiler) son formülasyonu, belirtilen alternatif sistemleri, alternatiflerin karşılaştırıldığı kriterleri, denemelerin sonuçları ve problemin uygun çözümünü gözden geçirme imkanı vermektedir. Ayrıca, kararlar daha yüksek bir düzeyde onaylanırsa, rapor, model kullanıcı/karar verici için bir onay aracı

sağlar ve modelin güvenilirliğine ve model kurma prosesine katkıda bulunur.

12. Uygulama: Uygulama sahasının başarısı, önceki 11 adımın ne kadar iyi bir şekilde uygulandığına bağlıdır. Ayrıca bu başarı, analistin tüm simülasyon prosesi boyunca son model kullanıcısı olarak ne kadar kullanıldığına bağlıdır. Eğer, model kullanıcı model kurma prosesi boyunca çalışmışsa ve bu kullanıcı modelin yapısı ve çıktıları anlarsa, başarılı bir uygulama ihtimali de artar. Aksine, model ve modelin önemli kabulleri doğru olarak açıklanmamışsa, simülasyon modelinin uygunluğuna rağmen, uygulama başarısız olacaktır.

Şekil 3.4'de gösterilen model kurma prosesi, dört safhaya ayrılabilir. 1.(Problemin Formülasyonu) ve 2.(Amacın ve Tüm Tasarımın Belirlenmesi) adımlardan oluşan ilk safha, buluş veya adaptasyon peryodudur. Problemin ilk tanımı, genelikle "belirsizdir"; başlangıç amaçları yeniden belirlenmelidir ve orjinal proje planı iyi bir şekilde oluşturulmalıdır. Bu ayarlamalar ve açıklamalar bu safhada veya diğer safha sırasında (yani analist prosese yeniden başlayabilir) yapılmalıdır.

İkinci safha, model kurma veri toplamayla ilgili olup, 3.(Model Kurma), 4.(Veri Toplama), 5.(Program Yazma), 6.(Doğrulama) ve 7.(Onaylama) adımlarını içermektedir. Adımlar arasında, sürekli bir ilişki gereklidir. Bu safha sırasında, model kullanıcısının olay dışında bırakılması, uygulama noktasında kötü sonuçlar doğurabilir.

Üçüncü safha, modelin işletilmesiyle ilgilidir. Bu safha, 8.(Deneysel Tasarım), 9.(Üretim Çıktıları ve Analizi) ve 10.(İlave Çıktıları) kapsamaktadır. Bu safha aynı zamanda, simülasyon modeli ile deney yapılması için tamamen uygun bir plana sahip olmalıdır. Kesikli olay stokastik simülasyonu aslında, istatistiksel bir deneydir. Çıktı değişkenleri, rassal hatalar içeren tahminlerdir ve bu yüzden doğru bir istatistiksel analiz gerektirir. Böyle bir felsefe, tek çıktı alan ve bu tek veriye göre yorum yapan analist ile ters düşmektedir.

Dördüncü ve son safha (Uygulama), 11.(Döküman Programı ve Rapor Sonuçları) ve 12.(Uygulama) adımlarını kapsamaktadır. Başarılı bir uygulama, model kullanıcısının sürekli bir katılımı ve prosesteki her adımın başarıyla tamamlanmasına bağlıdır. Tüm proses içinde belki de en önemli nokta 7.adımdır (Uygunluk). Çünkü, uygunsuz bir model, sonucu tehlikeli ve yüksek maliyetli olan hatalı sonuçlara yol açabilir.

3.1.12. Bir Simülasyon Modelinin İşletim Süresinin Belirlenmesi

Bir simülasyon modelinin çalıştırılmasından önce, analiz ve simüle edilecek zaman periyodunu planlanmalıdır. Bu karar önemli bir şekilde, analistin ilgilenmekte olduğu geçiş, sabit durum veya her iki koşulunda gerçekleşip gerçekleşmediğine bağlıdır. Olasılık modellerinin bu durumunda, örnekleme hatasının kabul edilebilir seviyesi, dikkatle kontrol edilmelidir.

Bir simülasyon modelinin işletim süresinin belirlenmesi için aşağıda belirtilen, iki hususun göz önüne alınması gerekmektedir :

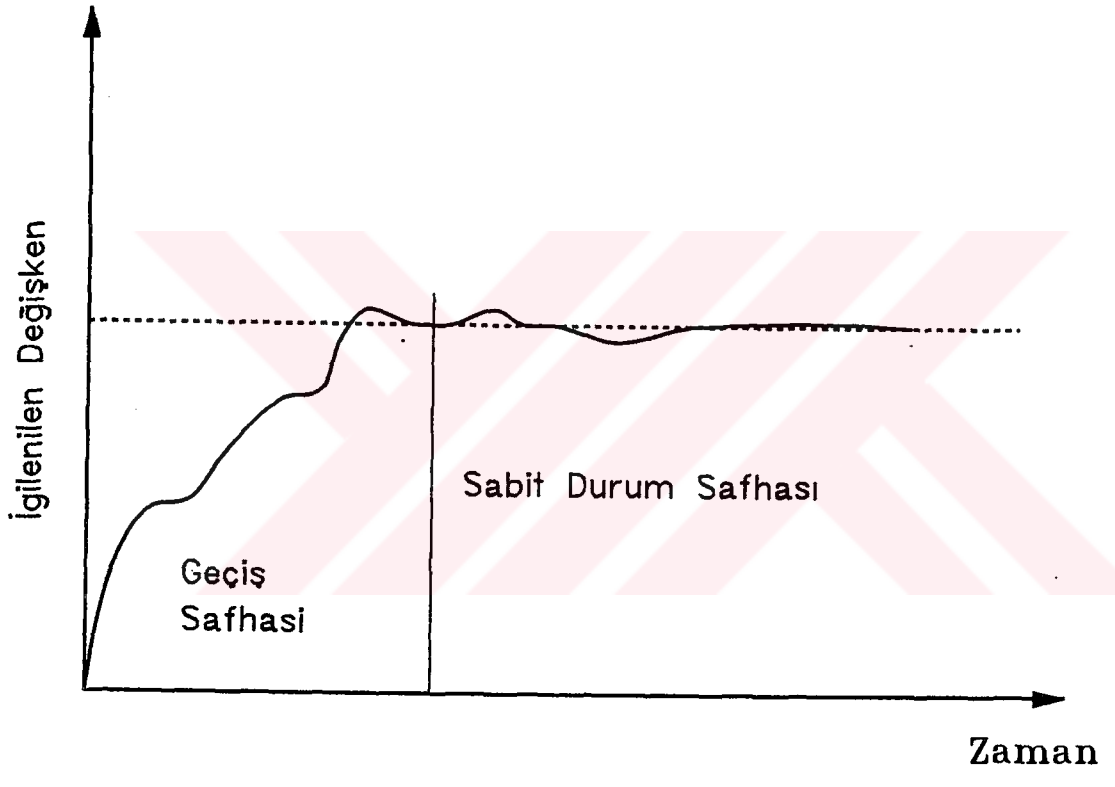
(1) Geçiş ve Sabit Durum Koşulları

(2) İstenilen İstatistiksel Doğruluktaki Gözlem Sayısının Belirlenmesi

1. Geçiş ve Sabit Durum Koşulları : Simülasyon modelleri, bazı başlangıç şartları ile çalıştırılmaktadır. Örneğin bir kuyruk simülasyonunda, tam ve açık olarak belirli bir kaç eleman gereklidir, diğerleri ise simülasyon başladığı zaman kuyruk içindedir. Bu ilk şartlar, sık sık başlangıç koşullarına başvururlar. Başlangıç koşullarının seçimi, ilk olarak sistemin davranışının incelenmesine bağlıdır.

Analist, geçiş, sabit-durum veya her iki koşulla ilgilenmektedir. Geçiş koşulları, sistem çalıştığı zaman ilk olarak ortaya çıkan,sistemin

geçici davranışıdır. Örneğin, bir banka modelini ele aldığımızda, sabahleyin bankanın açılmasında sonra, ilk müşteri bankaya girdiğinde, veznedarların ve banka memurlarının tümünü servis vermek için hazır ve boş durumda bulur. Bu koşul bir geçiş şartıdır ve bu, kuyruklar oluşmaya başlayana kadar devam eder. Sonuç olarak bir banka sistemi, tipik işlem koşullarının ortaya çıkmasıyla sabit duruma ulaşır. Geçiş ve sabit-durum koşulları şekil 3.5'de görülmektedir /2/.



Şekil 3.5. Geçiş ve Sabit-Durum Koşulları

Geçiş koşulları ile ilgilenildiği zaman, başlangıç koşulları bizim için, başlangıçta seçilmiş olan gerçek hayattaki sistemin bir yansımasıdır. Örneğin bir üretim sistemini ele aldığımızda, istenilen üretim hızına ulaşılması için, başlangıçtan sonra uzun bir ilgilenme gerektirmektedir. Bu sisteme bağlı olarak, tamamen boş bir sistemi veya belki de, proses içindeki bazı işlerden birini içermektedir. Simülasyon

sabit durum koşullarına ulaştıktan sonra sistemin çalışması, istenilen bilginin gözlemlerden elde edilmesi için durdurulabilir.

Bazı durumlarda ise sadece sabit-durum koşulları ile ilgilenilmektedir. Örnegin, bir hızlı yemek (fast-food) sisteminde, en kalabalık öğle saatine göre geliştirilen servis için isteklere diğer zamanlarda az bir ilgi olmaktadır.

Sabit durum davranışı hakkında bilgi toplamak için iki yaklaşım kullanılmaktadır. Birinci yaklaşım, sabit-durum içinde içinde sistemin nasıl göründüğünü yansıtan, sistem için başlangıç şartlarının seçilmesidir. Bu yaklaşım ile simülasyon hiçbir geçiş durumuna sahip değildir. Bununla beraber bazı durumlarda uygun başlangıç şartları, oluşum içinde bilinmemektedir. Bu, yeni bir sistem tasarlanılmaya başlandığı zaman tipik olarak ortaya çıkan bir durumdur ve bir sistemin davranışını, başlangıçtan sabit duruma erişene kadar simüle etmek için kullanılan bir yaklaşımdır, fakat istatistiklerin toplanılması geçiş koşullarından çıkana kadar ertelenmektedir.

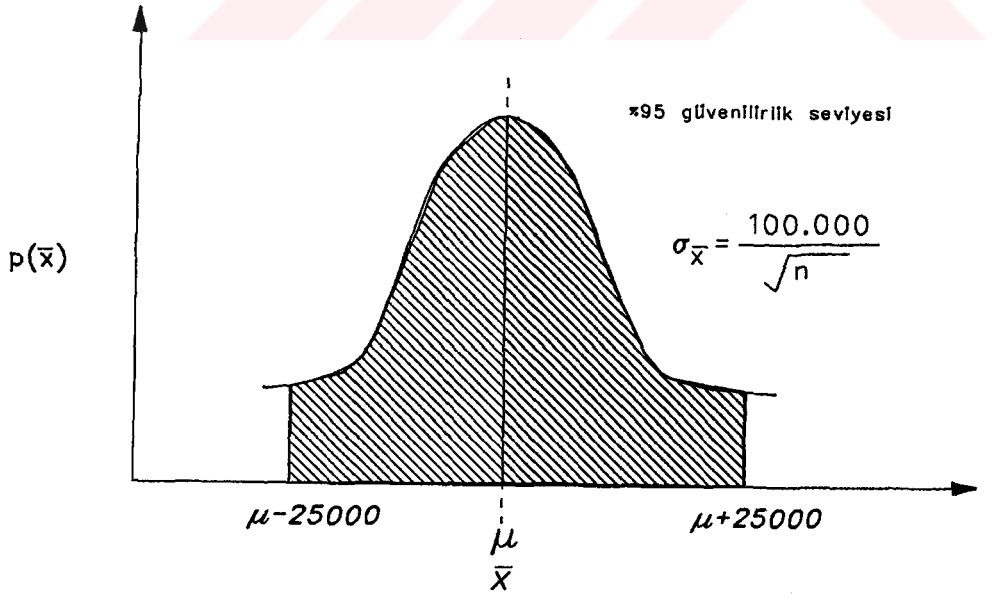
Geçiş ve sabit-durum koşullarının her ikisiyle de ilgilenildiği zaman, simülasyon yeterince uzun bir zaman periyodu boyunca çalıştırılır ve böylece her ikisinin de gözlenmesi mümkün olmaktadır. Başlangıç koşulları sistemin çalıştırılmasında kullanılan gerçek hayattaki sistemin bir yansımasıdır. Zaman içinde birkaç noktada istatistiksel çıktı ve sonuçlar elde edilir, böylece analist, geçiş ve sabit durumdaki sistem davranışlarının her ikisi hakkında da bilgi sahibi olur.

2. İstenilen İstatistiksel Doğruluktaki Gözlem Sayısının Belirlenmesi : Analist,olasılık simülasyon modelleri ile, istenilen istatistiksel doğruluktaki gözlemleme hakkında düşünmelidir. Simülasyonun işletim süresi yeterince uzun olmalıdır ve böylece de, örnekleme hatasının olabilirliğinin istenilen seviyesi kontrol edilebilmektedir. Burada bu probleme birkaç yaklaşım anlatılmaktadır.

Bir olanak, işletim süresinin yeterince uzun yapılması ve böylece de herhangi bir mümkün örnekleme hatasının kolayca kontrol edilebilmesidir. Bu yaklaşım, bir çok basit simülasyon modelleri için bazen doğruluğu kabul edilebilir. Bununla beraber, çoğu simülasyon modelleri için bu yaklaşım, gereksiz yere normalden fazla makina maliyetlerine neden olur.

Tercih edilen bir alternatif, kabul edilebilir bir hata miktarı ve istenilen seviyede istatistiksel hassasiyet belirlemek ve daha sonra işletim boyutunu belirlemektir. Simüle edilmiş gözlemler istatistiksel olarak bağımsız olduğu zaman veya yaklaşık olarak böyle olduğunda bu yaklaşım normal olarak mümkündür, çünkü, simülasyon çıktıları, rassal olarak örneklenmiş gözlemlere karşılık gelmektedir.

Örneğin bir yönetim, ±25.000 TL ile %95 güvenilirlik derecesinde olmak üzere gelecek yıllar için ortalama yıllık kârlarını simüle etmek istemektedir. Standart sapma 100.000 TL'dir. Bu durum, Şekil 3.6'da hazırlanmış olarak gösterilmektedir /2/. Gerekli hesaplamalar Tablo 3.2'de verilmektedir /2/. İşlemler sonucu, işletim boyutu olarak 61 iterasyon belirlenmiştir.



Şekil 3.6. İterasyon Sayısının Hesaplanması

Tablo 3.2. İterasyon Sayısının Hesaplanması

$$Z = \frac{X - \mu}{\sigma / \sqrt{n}}$$

$$Z = 1.96 = \frac{(\mu + 25.000) - \mu}{100.000 / \sqrt{n}}$$

$$Z = 1.96 = \frac{25.000}{100.000 / \sqrt{n}}$$

% 95 güvenirlik seviyesi için

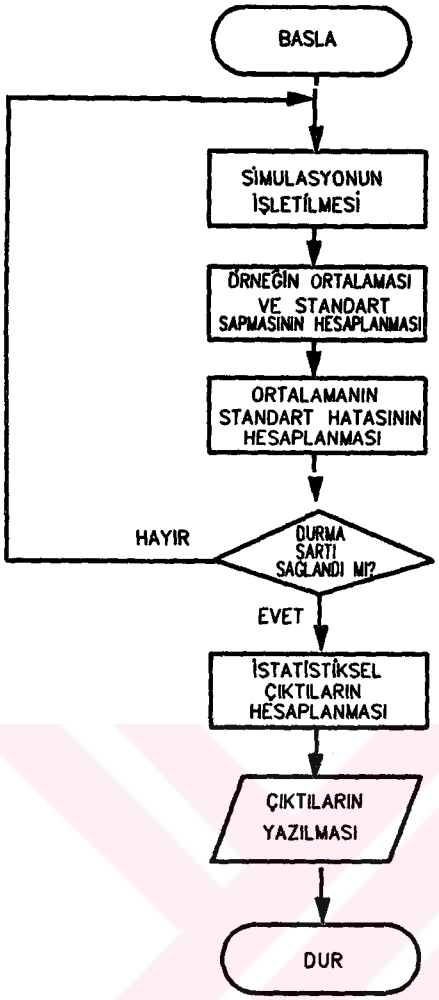
$$\sqrt{n} = \frac{1.96(100.000)}{25.000}$$

$$n \approx 61$$

Bir diğer alternatif, kompüterize edilmiş model içinde otomatik durma şartı kurmak, modelin durdurulmasını gerektirmez. Bu mekanizma ile programlanan koşul meydana geldiği zaman simülasyon otomatik olarak durur (bu durumda simülasyon, istenilen istatistiksel hassasiyetteki gözlemlere sahip olduğu zaman durmaktadır). Daha sonra her iterasyonda ilgilenilen çıktı değişkeni ile ilgili standart sapma yeniden hesaplanabilir.

Daha sonra, koşum (işletim) içindeki iterasyonların sayısına bağlı olarak ortalamanın standart hatası hesaplanabilir. Eğer satandart hata, istenilen hassasiyeti verecek miktarda azaltılabilirse, belirlenen çıktının sağlanması ileönceden programlanmış mantık ile simülasyon durdurulur.

Bu tip bir otomatik durma şartı için ana mantık Şekil 3.7'de akış diyagramı halinde gösterilmektedir /2/.



Şekil 3.7. Otomatik Durma Şartının Mantıksal Yapısı

3.2. BİLGİSAYARLA SİMÜLASYONDA KLASİK YAKLAŞIM

Simülasyon modellemesindeki klasik bir yaklaşım Şekil 3.8' de gösterilmektedir. Klasik yaklaşıma göre, simülasyon yapan kişinin, modellenecek sisteme veya probleme ait genel bir bilgisi olması gerekmektedir. Bu bilgiye göre simülasyonu yapan kişi, daha sonra sistemin modelini tanımlamalıdır. Sonuç ise, detaylı bir problem spesifikasyonudur.

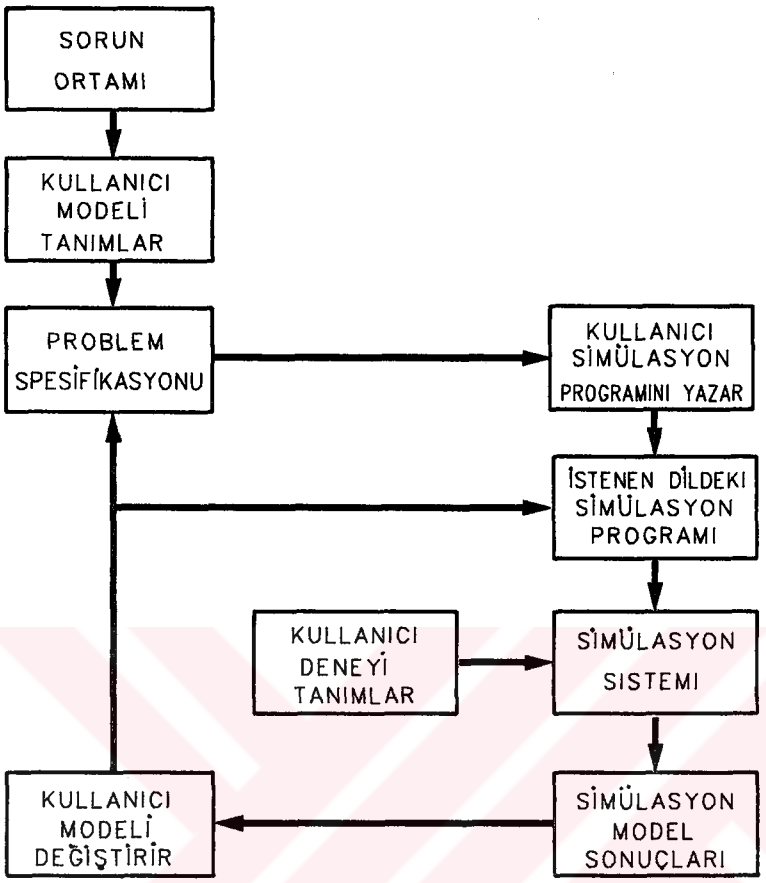
Simülasyon için bir sonraki adım, problem spesifikasyonuna göre, genel amaçlı bir programlama dili veya özel amaçlı bir simülasyon dili

kullanılarak, modelin bir bilgisayar programının oluşturulmasıdır. Daha sonra, geliştirilen bu simülasyon programı, tüm yazılım hatalarının düzeltilmesi için incelenir. Hatalar düzeltildikten sonra, model doğru olarak kabul edilir ve onaylanır. Modelin doğruluğu, simülasyon modelinin bir simülasyon dili tarafından doğru olarak yazılması ile oluşur. Başka bir deyişle modelin doğruluğu, programın düzgün bir şekilde çalıştığını gösteren kontrol prosesidir. Modelin onaylanması ise, modelin davranışı ile gerçek sistemin davranışının karşılaştırılmasıyla ve gerekirse modelin, gerçek sistemi temsil etmek üzere kabiliyetlerinin arttırması için değiştirilmesi ve bunun sonucu olarak, modelin gerçek sistemi yansıtmasıyla olur.

Deneyin yapılamasında simülasyoncu, başlangıç şartlarını, duruş şartlarını, sabit durum (denge) ve örnek boyutunu belirlemelidir. Simülasyon genelde, sistem boş ve serbest olduğu kabul edilerek başlatılır. Bu yüzden, sistem dengede olmadan önce, bir zaman periyodu boyunca çalıştırılmalıdır.

Buradaki problem, denge halindeki bir sistemi göz önüne alarak yapılan, modelcinin ihmal etmek istediği hatanın üstündeki noktayı bulmaktır. Conway Tekniği /12/, genelde dengeyi belirlemek için kullanılır. Bu teknik, periyodik olarak ölçümleri elde ederek simülasyonu oluşturur. Her tekrardan sonra, elde edilen istatistiklerin herbiri, sistemin davranışını göstermek için bir zaman fonksiyonu olarak belirtilir. Conway Tekniği, bir ölçüm ihmal edilen kümenin minimum veya maksimum değeri olana kadar tüm ölçümleri ihmal eder. Bu ihmal edilen ölçüm kümesi, toplanan verilerden çıkartılan standart ölçüm kümesi olarak kullanılır.

Bir simülasyon çalışmasını durdurmak için gerekli şartlar, başlangıç şartlarına benzerdir. Örneğin, üretim simülasyonlarının çoğunda, istenilen sayıda parça elde edildiğinde durdurulur. Bu yaklaşımı kullanarak, simülasyon modeline giren gezer birimlerin sayısı sınırlandırılmaz. Bu yüzden çalışmanın sonunda gezer birimler halâ sistemde kalıp, kullanılan kaynaklar olurlar.



Şekil 3.8. Bilgisayarla simülasyonda klasik bir yaklaşım

3.3. UZMAN SİMÜLASYON SİSTEMLERİNİN TASARLANMASI

Simülasyon modeli kurmaya klasik bir yaklaşım Şekil 3.8'de /13/ görüldüğü gibi, sistem hakkında detaylı bilgileri elde eden, simüle edilecek olan sistemi geliştiren ve elle bir program üreten bir simülasyoncuyu içermektedir. Buradaki yazılım, genel amaçlı programlama dili veya özel amaçlı bir simülasyon diliyle, sistemin bilgisayarda icra edilebilen bir modelidir. Bilgisayarla simülasyona klasik yaklaşımda, bir simülasyoncu tarafından yerine getirilen, elde etme ve programlama işlerinin her ikisi de otomatik olabilmektedir.

Klasik simülasyon yaklaşımı ile yapay zekâ tekniklerini birleştiren mevcut araştırmalar, iki spesifik alanda toplanmışlardır. İlk alan, problem spesifikasyon prosesinin otomatikleştirilmesine yönelik olarak yapay zekânın kullanımıdır. İkinci ve daha zor olan saha ise, arzulanan simülasyon dilinde otomatik olarak programın oluşturulması için yapay zekânın kullanılmasıdır.

3.3.1. Problem Spesifikasyonunun Otomatik Olarak Elde Edilmesi

Simülasyon modeli kurmanın, model ile ilgili spesifikasyonları elde etme kısmını otomatikleştirme yaklaşımları için üç alternatif yapı, Şekil 3.9'da şematik olarak görülmektedir /13/. Otomatik problem spesifikasyonu, kullanıcıya bir yapay zekâ desteği olarak göz önüne alınabilir.

Tam bir modelin kurulması için, örneğin bir kuyruk sistemini göz önüne aldığımızda, kuyruk modelinin yapısı, kuyruk disiplinleri, gelişlerin dağılımı ve her bir servis süresinin dağılımları ve parametreleri gibi ilave detaylara gereksinim duyulmaktadır. Bu bilgiler de dış ortamdan bir kullanıcıdan otomatik olarak elde edilebilmektedir.

Burada, iki tür bilgi içeren bilgi tabanları, otomatik problem spesifikasyonunu desteklemek için kullanılmaktadır. Bu bilgi tabanındaki bigiler, problem alanına yönelik bilgi ve simülasyon modelleme bilgisidir. Sonuçta, her problem alanı kendi bilgi tabanını gerektirir. Simülasyon modelleme bilgisi de, arzulanan dilde son simülasyon programının yazılması için gerekli bilgidir.

Bu bilgi tabanları daha sonra, sistem ile kullanıcı arasındaki karşılıklı diyaloğu yöneten bir kurallar kümesini tanımlamak için kullanılır. Sistem, kullanıcının spesifik sorulara verdiği cevaba bağlı olarak, değişik kuralları kullanarak farklı bir mantık işletebilmektedir. Karşılıklı diyaloğun tamamlanmasında sistem, problem veya modelin bir iç spesifikasyonunu tanımlayacaktır.

Simülasyon modelinin veya problem spesifikasyonunun detaylı bir yapıda tanımlanmasında kullanıcıya yardım etmek üzere üç yaklaşım kullanılabilmektedir. Şekil 3.9'da görülmekte olan bu üç yaklaşım şunlardır :

(1) Doğal dil arabirimi

(2) Etkileşimli grafik arabirimi

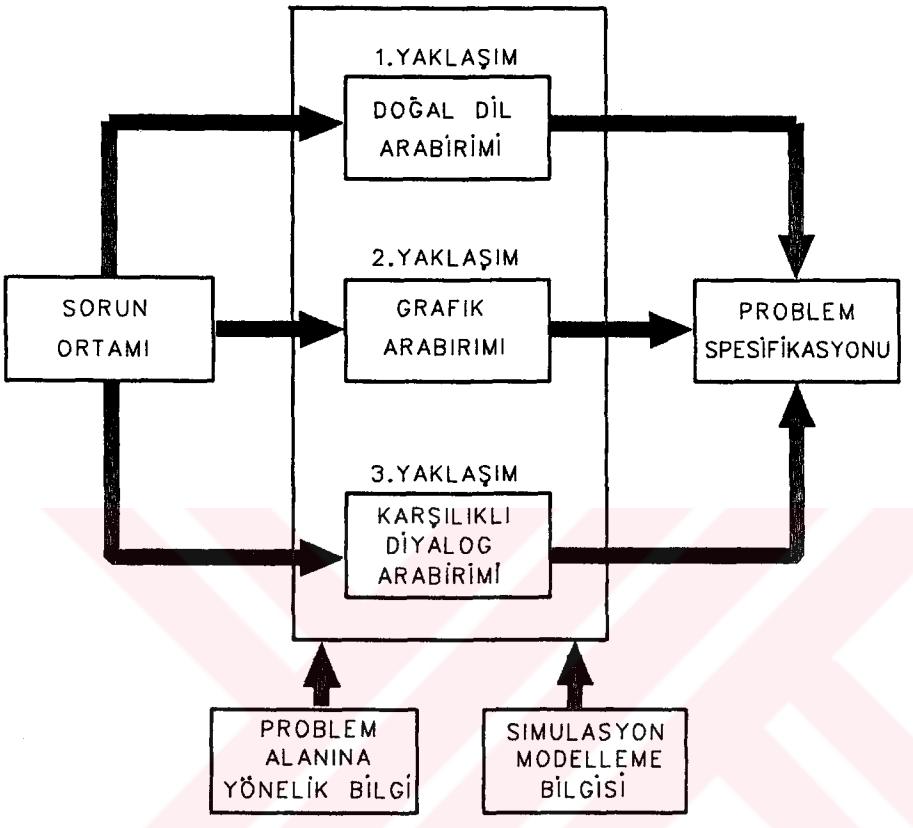
(3) Karşılıklı diyalog arabirimi

1. Doğal Dil Arabirimi : Bir doğal dil arabirimi, sorunun veya problemin bir kullanıcı tarafından, bilgisayara bir klavye yardımıyla serbest metin formatında tanımlanmasını sağlar. Girilen bu metin, bir doğal dil arabirimi tarafından önce dilbilgisi, daha sonra da içerik yönünden incelenir ve yorumlanır. Bu sistemleirn bir çoğu, eksik bilgi ve olası tutarsızlıkları belirlemek için, kullanıcıyla etkileşimli olarak çalışır.

Doğal dil etkileşimli paketlere örnek olarak, Heidorn /14/ tarafından (NLPQ Natural Language Programming for Queueing Simulations 1974) kuyruk modellerinin simülasyonu için geliştirilen doğal dil programlama sistemi verilebilir. Bu sistemde, sorun ingilizce dilinde bilgisayara girilir ve daha sonra NLPQ, GPSS dilinde simülasyon programı oluşturur.

Diğer bir örnek ise, elektronik üretim simülasyon sistemidir (EMSS, Elektronic Manufacturing Simulation System). Ford ve Schroer /15/ tarafından geliştirilen bu sistemde, sorun yine ingilizce dilinde bilgisayara girilir ve EMSS de SIMAN dilinde simülasyon programları oluşturur.

2. Etkileşimli Grafik Arabirimi : Problemin tanımlanmasında kullanıcıya yardımcı olan ikinci bir yaklaşım da etkileşimli grafik arabirimidir. Grafiksel etkileşimde, simülasyonu yapılacak sistemin



Şekil 3.9. Otomatik Problem Spesifikasyonuna Ait Üç Yaklaşım

oluşturulması için, kullanıcı tarafından klavye veya bir mouse ile seçilebilen, şekillerde oluşan bir menü mevcuttur. Sistem oluşturulduktan sonra, kullanıcı klavye yardımı ile şekillere karşılık gelen özelllikleri girer.

Grafiksel etkileşimli paketlere bir örnek olarak, Khoshnevi ile Chen /16/ tarafından LISP dilinde yazılmış olan paket gösterilebilir.

Burada kullanıcı için modelin oluşturulmasında kullanılmak üzere bir şekil kütüphanesi mevcuttur. Bu şekiller, " CREATE ", " SEIZE ", " SERVICE ", " TRANSFER " ve " GATE " bloklarını içermektedir. Kullanıcı, bu şekillerden seçim ve ekranda uygun yerleştirme ile üretim sistemini ve iş akışını tanımlar. Sistem, eksik ve fazla bilgileri kontrol ederek kullanıcıyı harekete geçirir. Modelin grafik şekli tamamlandıktan ve gerekli özellikler girildikten sonra, sistem otomatik olarak uygun SLAM simülasyon programını oluşturur.

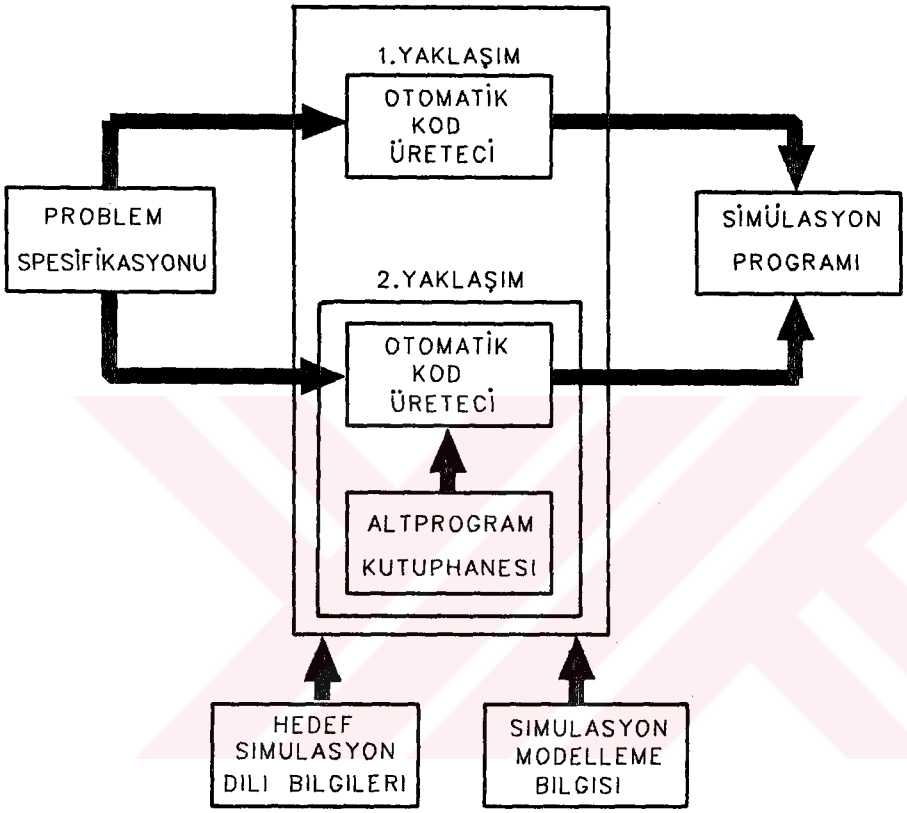
3. Etkileşimli Diyalog Arabirimi : Problem spesifikasyonun tanımlanmaında kullanıcıya yardım eden üçüncü yaklaşım da karşılıklı diyalog arabirimidir. Sistem burada, kullanıcıya sorulan bir dizi sorulardan oluşmaktadır. Soruların sırası, kullanıcının cevaplaplarına bağlı olarak değişebilmektedir. Bir problem spesifikasyonunun tanımlanması için kullanılan üç yaklaşımdan, geliştiriciler tarafından en çok kullanılanı, karşılıklı diyalog arabirimidir.

Karşılıklı diyalog arabirimini kullanan çok sayıda sistem geliştirilmiştir. Haddock ve Davis (1985), esnek üretim sistemi (FMS) için bir simülasyon üreteci geliştirmişlerdir. Sistem burada, esnek imalat sisteminin karakteristiklerinin bir simülasyon modeline çevrilmesinde kullanıcıya yardım eden menülü bir ekran kümesinden oluşmaktadır. Bu giriş karakteristikleri, esnek imalat sistemindeki, istasyon sayısı ve tiplerini, envanter transfer araçlarının, geliş hızları, hazırlıkları ve işleme zamanlarını içermektedir. Sistem IBM PC'de ve BASIC ile yazılıp, SIMAN dilinde simülasyon programını oluşturmaktadır.

Brazier ve Shannon /18/, otomatik klavuzlu araç sistemlerinin modellenmesi için otomatik bir programlama sistemi geliştirmiştir. Sistem kullanıcıya otomatik klavuzlu araçların tanımlanmasında yardımcı olmak üzere karşılıklı bir diyalog kullanılır. Problem spesifikasyonu elde edildikten sonra sistem uygun SIMAN programını yazar. Bu paket IBM PC'de Turbo PROLOG'da yazılmıştır.

3.3.2. Otomatik Olarak Simülasyon Programını Oluşturma Yaklaşımları

Simülasyon ile yapay zekânın birleştirilmesine ait otomatik problem yaklaşımının şematik bir görünümü Şekil 3.10'da verilmektedir /13/.



Şekil 3.10. Otomatik Programlama Sistemlerindeki Yaklaşımlar

Temelde, iç problem spesifikasyonlarının alınması ve daha sonra arzulanan simülasyon dilinde programın yazılması için iki farklı yaklaşım uygulanmaktadır. İlk yaklaşım, problem spesifikasyonuna ait iç yapıya göre, direkt olarak bir otomatik kod üreticiyle simülasyon programını yazmaktır.

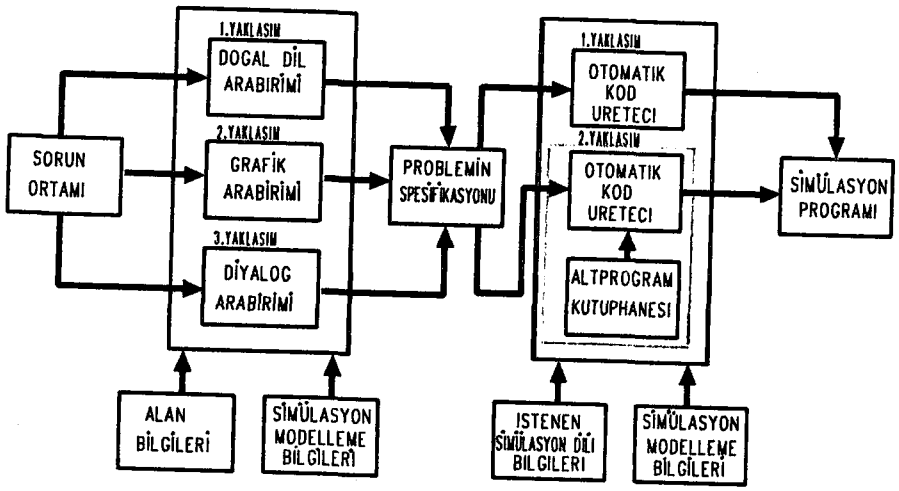
İkinci yaklaşımda ise, simülasyon programının otomatik olarak yazılmasına yardımcı olan önceden tanımlanmış altprogramlarda (makrolar) oluşan bir kütüphaneyi kullanarak, bu makroların uygun bir

kombinasyonu ile simülasyon programını oluşturur. Bu şekildeki bir yaklaşımın avantajı, literatürde daha önceden incelenen problemler yerine daha modernize problemleri çözebilmesidir. Dezavantajı ise, bir çok alt-programların belirli alana yönelik olması ve bu makroların bilgisayar belleğinde oldukça fazla yer kaplamasıdır.

Otomatik simülasyon yazılımını desteklemek için iki bilgi türü kullanılmaktadır. Bu bilgi türleri, simülasyon modelleme bilisi ve arzulanan simülasyon dili bilgisidir. Simülasyon modelleme bilgisi, Şekil 3.9'daki otomatik problem spesifikasyonu için kullanılan bilgi tabanıyla aynıdır. Arzulanan simülasyon dili bilgisi, simülasyon diline yönelik bilgidir. Bu bilgi tabanları daha sonra, iç problem spesifikasyonlarına göre uygun simülasyon programını oluşturmak için bir kural kümesinin tanımlanmasında kullanılır.

3.3.3. Simülasyon Programı Üreteçlerinin Yapısı

Simülasyon programı üreteçleri, kullanıcıdan alınan veriler ile bilinen bir simülasyon dilinde, simülasyon programını otomatik olarak oluşturan paket programlardır. Bu tür üreteçlerin genel yapısı Şekil 3.11'de verilmektedir /19/. Genelde iki aşamalı olan üreteçlerin birinci aşamasında diğer bütün yapay zekâ paketlerinde de bulunan bir sorun algılama mekanizması vardır. Yani daha önce incelediğimiz gibi, problem spesikasyonun otomatik olarak elde edilmesi modülü vardır. İkinci aşamada ise bir simülasyon programı oluşturma modülü yer alır. Sonuç olarak bu iki modülün birleştirilmesi ile ortaya çıkan yapı bir simülasyon üretecidir. Bir simülasyon üreteci, kullanıcıdan alınan veriler ile önce problemin spesifikasyonunu elde eder ve daha sonra ise bu problem spesifikasyonunu kullanarak bilinen bir simülayon dilinde otomatik olarak bir simülasyon programını oluşturulur. Sistem, simülasyon programının oluşturulacağı alana yönelik bilgileri, seçilen simülasyon dili bilgilerini ve genel simülasyon modelleme bilgilerini de İçermektedir.



Şekil 3.11. Bir Simülasyon Üretecinin Genel Yapısı

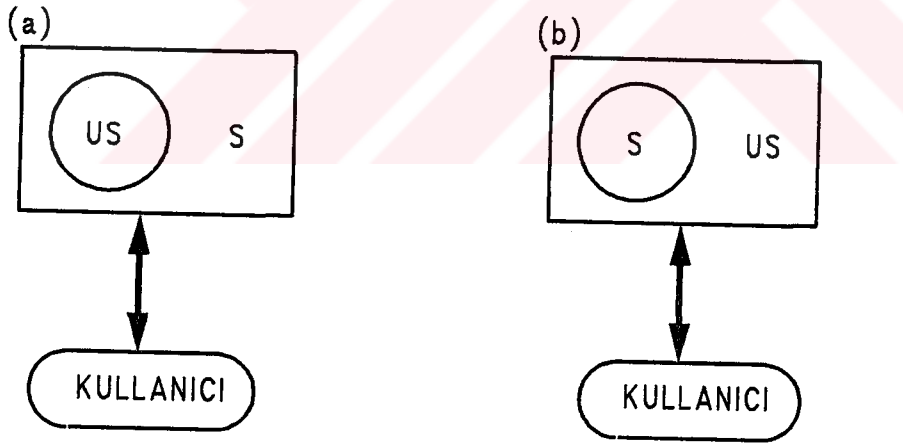
3.4. SİMÜLASYON ve UZMAN SİSTEMLERİN BİRLEŞTİRİLMESİ İÇİN BİR SINIFLANDIRMA

Simülasyon ve uzman sistemler arasında kuvvetli bir metodolojik benzerlik vardır. Bu benzerlik kullanılabilir, ancak iki alan arasındaki çapraz verim artışı sadece bu benzerliğe bağlı değildir.

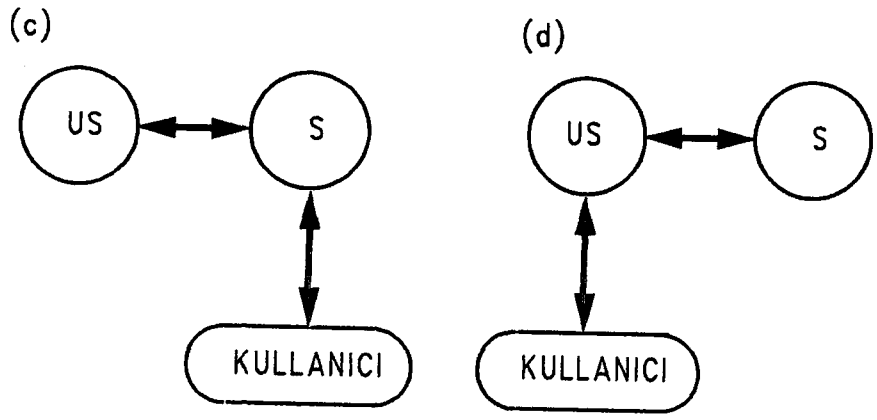
Şekil 3.12 - 3.15 arasında, simülasyon ve uzman sistemlerin birleştirilmesine ait bir sınıflandırma gösterilmektedir /20/. Bu iki elemanın birleştirilmesindeki en açık yol, Şekil 3.12a'da veya ters olarak Şekil 3.12b'de bir simülasyon modeli içindeki bir uzman sistemin, yerleştirilmesi ile olmaktadır. Burada bir çok simülasyon modelleri, veri yerine bilgiyi kullanmaktadır. Örneğin, bir kuyruk öncelik kuralı, bir bilgidir. Bu şekildeki bir kuralı program olarak yazmak yerine, bilgi tabanında saklamak daha uygundur. Bir uzman sistemin içinde simülasyon kullanılması için iki neden vardır. Birincisi uzman sistem, kullanıcı için bazı sonuçların elde edilmesinde, simülasyonu işletmektedir. Buna örnek olarak zeki bir öğretim sistemi olan, SOPHIE

gösterilebilir. İkincisi ve daha önemlisi ise, uzman sistemin bir veya birden fazla zamana bağlı değişkenlerin kullanılması, böylece bunların değerlerinin güncelleştirilmesi için simülasyona ihtiyaç duyulmasıdır.

Ayrı bir yazılım olarak tasarlanan, geliştirilen ve uygulanan, simülasyon ve uzman sistemler paralel olarak birbirlerini etkilerler. Bir simülasyon modeli, bir uzman sisteme sorular sormaktadır. Bu yapı Şekil 3.13c'de görülmektedir. Bu, simülasyonun karmaşık bir sistem için geliştirildiği ve bir uzman sistemin, bu sistem içinde karar vermenin bir parçası şeklinde ortaya çıktığı durumlarda faydalı olmaktadır. Bu durumda simülasyon, karar kurallarını simüle etmek veya program halinde yazmaktan ziyade, bu durumu elde edebilir. Simülasyonu işleten veya sonuçlarını kullanan uzman sistemler, sekil 3.13d'de görülmektedir. Bu şekildeki uzman sistemler, bilgi mühendislerinin artan bir ilgisini oluşturmaktadır. Kullanıcı üzerinden veya gerçek bir ortamdan bir uzman sistemin test edilmesinden ziyade uzman sistem, simülasyon ile test edilebilir. Burada sadece geliştirme zamanı azaltılmamakta aynı zamanda,



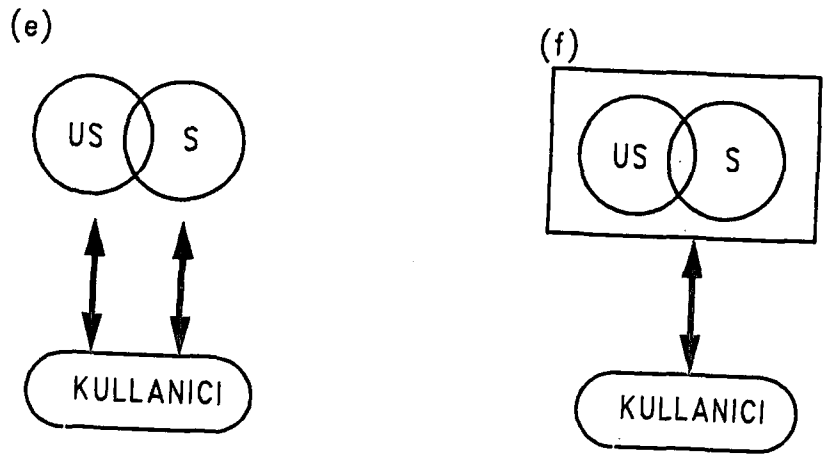
Şekil 3.12. Simülasyon (S) ve Uzman Sistemlerin (US) İçiçe Yerleştirilerek Birleştirilmesi



Şekil 3.13. Simülasyon (S) ve Uzman Sistemlerin (US) Paralel Bir Şekilde Birleştirlmesi

test daha anlaşılır olamaktadır. Ayrıca, eğer simülasyon geçerli bir model ise, bu durumda simülasyonu kontrol eden veya simülasyona cevap veren bir uzman sistemin kendisi de geçerlidir. Bu yaklaşım, gerçek zamanlı proses kontrolunda faydalı olmaktadır. Bir uzman sistemin son kontrolu yaptığı sahalardaki proses, bu prosesin sürekli bir simülasyonu ile test edilebilir. Bu, daha fazla araştırma gerektiren ilginç bir sahadır.

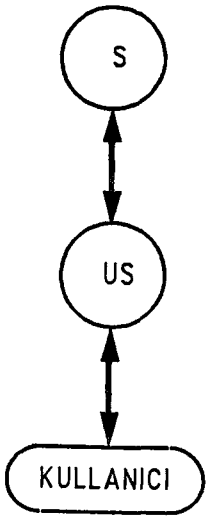
Şekil 3.12a - 3.13d'ye kadar temel yöntemlerin kullanıcıları diğerlerine direkt olarak geçiş yapamamakta olsa bile, bir çok örnekte, hem uzman sistem, hem de simülasyon bazı işleri yapmak için beraberce kullanılmaktadır. Şekil 3.14e'de etkin olarak, her ikiside (uzman sistemlerin ve simülasyonun) bazı verileri ortak olarak kullandıklarından, simülasyon ve uzman sistem bir işte ortak olarak çalışmaktadır. Bu şekildeki bir ortak yapı veya beraber olamanın uygulaması, uzman sistemin tamamlanmış bir simülasyon modeli veya modelin geliştirilmesinde, kullanıcıya yardım eden bir yöntem olduğunda ortaya çıkmaktadır.



Şekil 3.14. Simülasyon (S) ve Uzman Sistemlerin (US) Bir İşbirliği Yapacak Şekilde Birleştirilmesi

Deneye ve analize yardım eden uzman sistemler çalışma için ilginç bir alan oluşturmaktadır. Simülasyon modellerinin, tecrübesiz simülasyon kullanıcılarına yani müşterilere sunulmasında artan bir eğilim olamakta ve bu şekildeki modellerin yanlış kullanılmasını engellemek ve kullanımını desteklemek amacıyla büyük bir artış olmaktadır.

Ortak, simülasyon ve uzman sistem, şekil 3.14f'deki gibi yazılımın büyük bir kısmı tarafından kapsanmaktadır. Buradaki herbir eleman direkt olarak karar verici tarafından kullanılan büyük bir karar destek sisteminin bir parçası veya bir simülasyon ortamının bir elemanı olabilir. Hem simülasyona hem de bilgi tabanlı metotlara bağlı yeni araçlar bu kategoriye düşmektedir. Örnek olarak, Carnegie-Melon Üniversitesi'nde geliştirilen " Bilgi Tabanlı Simülasyon Sistemi " ve Rand Şirketi'nde geliştirilen " Kural Esaslı Simülasyon Sistemi " verilebilir.



Şekil 3.15. Zeki Ön Elemanlar (Intelligent Front Ends)

Bilgi tabanlı metotlara ait en önemli uygulamalardan birisi de " Zeki Ön Elemanlar (Intellicent Front Ends,IFE) " olamaktadır. Bu formdaki bir yapı şekil 3.15'da görüldüğü gibidir /20/. Bu bir uzman sistemdir ve bir simülasyon paket programı ile kullanıncı arasında olup, kullanıcı ile bir diyalog kurarak, paket program kullanmak için gerekli komutları veya programı oluşturur ve daha sonra programdaki sonuçları yorumlar ve açıklar. Her ne kadar simülasyonu işletmese ve sonuçları yorumlamasa da simülasyon program üreteçleri zeki ön elemanların fonksiyonlarından bazılarını gerçekleştirir. Burada kullanılan bilgler şunları içermektedir :

(1) Diyalog iletimi (Doğal dil arabirimi veya en azından kullanıcıya yönelik serbest formatlı giriş).

(2) Bazı kullanıcı modeli,buradaki sistem kullanıcı tecrübeli veya tecrübesiz olsun kullanıcıya bağlı ihtiyaçlarını ayarlar.

(3) Hedef dilin modeli (Burada bazı kararlar kullanıcıya danışılmadan zeki ön eleman tarafından verilir).

Bazı simülasyon üreteçleri 3 numaralı maddenin elemanlarını içermektedirler. Çok az sayıdaki üreteçler ise 1 ve 2 numaralı maddeleri kapsamaktadırlar. Bunlardan bir istisna olarak FORTRAN ile yazılan

ekolojik sistemler ile ilgili dinamik modeller üreten " ECO " verilebilir. Burada, Doukids ve Paul /21/ 1 nolu maddeyi sağlayacak şekilde faaliyet-çevrim diyagramlarına bağlı model spesifikasyonlarının çıkarılması için doğal dil tabanlı bir sistem geliştirmiştir.

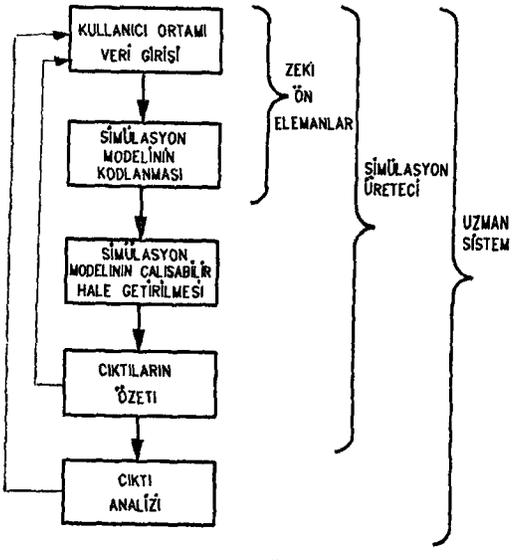
Simülasyon ve uzman sistemlerin birleştirilmesindeki kabulde, yukarıda açıklanan sınıflandırmalar kullanışlı olamaktadır. Bu şekildeki bir birleşim ciddi sonuçlar ortaya çıkartabilir. Uzman sistemler, ilgi duyulmasına karşın, tam olarak uygulanması çok az olamaktadır. Yukarıdaki analiz de, simülasyonda uzman sistemlerin uygulanmasında üç noktayı ortaya çıkartmaktadır. Bu üç nokta şu şekilde özetlenebilir:

(1) Mevcut simülasyon ve bilgi tabanlı metotların birleştirilmesiyle oluşturulan yeni simülasyon araçları.

(2) Özellikle deney ve analiz safhasında tecrübesiz simülasyoncular için fikir veren sistemler.

(3) Mevcut simülasyon paket programları için geliştirilen zeki ön elemanlar

Zeki ön elemanlar, simülasyon üreteçleri ve uzman sistemlerin birbirleriyle olan bağlantıları Şekil 3.16'da gösterilmektedir /22/.



Şekil 3.16. Uzman sistemler, Simülasyon Üreteçleri ve Zeki Ön Elemanlar

3.5. BİR SİMÜLASYON MODELİ İLE YAPAY ZEKA ARASINDAKİ FARKLAR

Simülasyon dilleri veya simülasyon modelleri yapay zekâda kullanılan çok sayıdaki fikirleri içermektedir. Örnek olarak, yapılacak işlemlerin veya gezer birimlerin (Sisteme gelen üniteler) kabiliyetlerini (Dinamik olarak değişen), karakteristiklerini tanımlayan özellikler (YAPI), sistemdeki gezer birimlerin akışını dinamik olarak değiştirmek (ÜRETİM KURALLARI veya ŞARTLI OLASILIKLAR), durum değişkenlerine bağlı olarak sistemi değiştirmek (MODEL TABANLI PROGRAMLAR) ve şebeke formundaki sistem bilgilerinin sunulması gösterilebilir. Doğal olarak, " bugün bildiğimiz doğru bir şekilde yazılmış simülasyon modeli ile yapay zekâ tabanlı uzman simülasyon sistemi arasındaki fark nedir? " şeklinde bir soru sorulabilir.

Temel fark, modelin kurulması ve işletilmesinde ortaya çıkabilir. Bugün simülasyon, modeli tanımlayan bir senaryoya (Girdiler), karar veren, deneyi işleten, sonuçları analiz eden, başka bir senaryoya karar veren, deneyi çalıştıran ve benzeri durumlardaki iteratif bir prosestir. Mevcut simülasyon sistemleri, uygun bir modele karar vermede veya problemimizin çözümünü bulmamızda, modelin incelenmesinde bize yardımcı olmamaktadır. Modelleyiciler, sistemin mantığını ve davranışını zorunlu değimler kümesine dönüştürmelidirler. Bu değimler daha sonra, kontrol değimi tarafından durdurulana kadar, yukarıdan aşağıya doğru işletilmelidirler. Yapay zekâ tabanlı bir uzman sistem, modelleyicinin sistem hakkındaki bilgileri açıkladığı (özellikle elemanların tanımı), amacı tanımladığı ve çözümün bulunması için bilgisayarın çalıştırıldığı çok farklı bir metodolojiyi izlemektedir. Sistem hakkındaki bir bilginin açıklanmasında, modelleyici, tüm mümmkün ve/veya kabul edilebilir modellerin tümünü tanımlamaktadır. Uzman simülasyon sisteminin sorumluluğu arzulanan spesifikasyonları sağlayan modeli otomatik olarak bulmak, istenilen çözümün elde edilmesinde uygun bir araştırma yöntemini işletmektedir.

İkinci fark ise, yapay zekâ tabanlı bir sistemde veri tabanı, bilgi tabanı ve kontrol yapısı birbirinden ayrıdır ve her biri diğerini etkilemeden kolayca değiştirilebilir. Bugün bir çok simülasyon

modelleri, her ne kadar bazı simülasyon dilleri iki veya üç kısma ayrılsa bile (model, deney, çıktı analizi gibi), bilgi ve kontrolun entegrasyonu sağlanmıştır.

Üçüncü fark veri tabanının yapısındadır. Klasik veri tabanı sistemleri, statik (verilerin işlenmesi esnansında) veri tabanlarında gerçekleştirilen geçerli olarak tanımlanmış veri yapılarını ve operasyonlarını gerektirmektedirler. Bu, ihtiyaçların değişmesine bağlı olarak, kolayca değişmeyen çok kararlı işlemdir. Aksine, yapay zekâ, veri tabanı sisteminin yapısı, toplanacak ve kullanılacak ilave bir veri tipini sağlamaktadır. Bu sembolik veriler, dar bir problem alanı hakkındaki olaylara ait bilgileri, yargıları, kuralları, sezgiyi ve deneyimi (tecrübeye dayalı bilgi) temsil etmektedir. Sezgisel veri, veri tabanındaki, klasik veri elemanları arasındaki ilişki kurallarını vermektedir ve burada kolayca eklemeler yapılabilir ve de değiştirilebilir. Klasik veri tabanı ile sezgisel verinin birleştirilmesi sonucu, " Bilgi Tabanı " olarak adlandırılır. Buna göre, yapay zekâ sistemleri, klasik veri tabaınında gerekli olan veri tabanındaki veri elemanlarının direkt olarak ilişkisinin gösterilesine bağlı değildir. Bu durum, bilgi tabanının değerlendirilmesi (sonuç mekanizması olarak bilinen) bilgi tabanındaki verinin yorumlanması, lojik sonuçların oluşturulması ve konu ile ilgili bilginin çıkartılmasında, bilgi tabanının değiştirilmesinde, sistemin farklı prosesleri kullanmasına izin vermektedir. Bu ilave esneklik ile yapay zekâ sistemleri, klasik veri tabanı bilgi sistemleri ile çözülemeyen bazı problemlerin çözümünü verebilmektedir.

Mevcut simülasyon modelleri ve bir uzman simülasyon sistemi arasındaki dördüncü fark, belirli karakteristikler şeklinde olabilir. Aşağıda, klasik simülasyonun karakteristiklerinden bazıları verilmektedir. Bunlar :

- Temel olarak nümerik,
- Algoritmik (Çözüm adımları kesin olarak belirli),
- Entegre bilgi ve kontrol,

- Ayrı olarak veya modelin dışında yapılan, modelleme işlem adımları (Örneğin, uygunluk için girdi verilerinin test edilmesi, örnek boyutunun belirlenmesi, işletilecek deneyin tasarlanması vb.),
- Önceden planlanmayan model hiçbir şey yapamaz (Yani kullanıcı, programın operasyonlarını açıklamalıdır).

Diğer yandan yapay zekâ tabanlı uzman simülasyon sistemi şu özelliklere sahiptir:

- Bir çok sembolik prosesler sahiptir,
- Modele yönelik araştırma kullanmaktadır (Çözüm adımları belirli değildir),
- Bilgi alanından ayrı bir komut yapısına sahiptir,
- Kullanıcı tarafından verilen kararların minimize edilmesi için mümkün uzman bilgiler modelin içinde bulunmalıdır,
- Kendi tecrübesinden öğrenebilecek ve gerektiğinde kendini değiştirebilecek bir modele sahip olması gerekir. Genelde, yapay zekâ sistemleri teorik olarak mümkün olmasına rağmen bunu henüz tam olarak başaramamaktadır.

Farklılıkla ilgili olarak beşinci metot ise, yapay zekâ tabanlı uzman simülasyon sisteminin kullanıldığı yerdir. Michie'e göre bir uzman sistem için üç farklı kullanıcı modu vardır /23/. Bunlar :

(1) Sistem bilgisini arttırmak ve azaltmak, öğretici olarak kullanıcı (kullanıcı öğretici rolünde)

(2) Problemlere cevap verme (kullanıcı müşteri rolünde)

(3) İnsanların kullanım amacıyla bilgi tabanının toplanması (kullanıcı öğrenci rolünde)

" Kullanıcı öğretici rolünde " modunda, simülasyon sistemi güçlü model geliştirme kabiliyetleri sağlamaktadır. Bu kabiliyetler, kullanıcılara yeni bir modelleme dilinin öğrenilmesi için 100 veya daha fazla saat harcamadan, gerçek veya teklif edilen sistemin sunulmasını kolayca geliştirmesine ve değiştirmesine olanak vermektedir. Modelin formu veya veriler için gerekli formatı bilmeden sistem veya gezer birimler hakkındaki tanımlayıcı bilginin kullanıcı tarafından geliştirilmesine olanak veren ayrı bir veri girişi arabirimi olmalıdır. Kullanıcı (veya sistem), kullanıcı tarafından model bilgisayar programında direkt programlama değişiklikleri yapılmadan model üzerinde, basit modifikasyonlar yapabilmelidir. Bilgi tabanı bu yapıda oluşturulur.

" Kullanıcı müşteri rolünde " modunda, simülasyon, kullanıcıya kendi ana dilinde soru sormasını sağlar ve sistemin hangi tür deneyi çalıştırması gerektirdiğine, gerekli örnek boyutuna vb. karar verir. Simülasyon sistemi, kullanıcı tarafından yönetilmeden sistemin mevcut durumuna bağlı olarak (rotalar, işlem zamanları vb.) soruları cevaplamak üzere ihtiyaç duyduğu bilgileri ortak veri tabanından alır ve bu sistem öğrenme yeteneğine sahip olup, sistem çalışırken kendini modifiye edebilmektedir. Bu durum, işletim boyunca geriye doğru izleme veya zaman sapmaları özelliğine ve ileriye veya geriye doğru bağlanma özelliğine sahip olabilir. Örnek olarak sistem optimal planının veya programın bulunmasına ya da belirli bir amaca nasıl erişileceğini bulabilmektedir. Bu optimizasyon deneylerinde tepki-yüzey metodolojisi kullanılarak limitli bir uzantı haline getirilebilir. Ancak, bugünkü modellerin bir çoğu, verilen bir senaryo (girdiler) için, olası sonuçlar vermektedir. Son olarak kullanıcı istenilen çıktı tipini veya formunu seçebilmelidir.

" Kullanıcı öğrenci rolünde " modunda, sistem uzmanlar tarafından verilen sayısal kararların (planlama) sonuçlarını alma özelliğine sahiptir ve kullanılan kuralların veya sezgisel metotların özetini çıkartır. Böylece diğer kişilere de öğretilebilir. Ayrıca model çözümlerini açıklama özelliğine sahip olamalıdır ve böylece kullanıcı

tarafından incelenen problem hakkında yeni görüş açılarına sevk etme (yöneltme) özelliğine sahip olmalıdır.

3.6. UZMAN SİSTEMLERİN SİMÜLASYON METODOLOJİSİ ÜZERİNDEKİ ETKİLERİ

Yapay zekâ ve özellikle uzman sistemler sahasından elde edilen teknikler ve teorik sonuçlar, simülasyon uygulamaları için yeni ve ilginç bir teknoloji ortaya çıkarmaktadır. Ancak bu teknoloji halen başlangıç aşamasındadır. Burada şöyle bir soru sorulabilir; " Bu teknik simülasyon metodolojisi üzerinde önemli bir etkiye sahip olacakmıdır? ". Bu soruya yaklaşımdaki bir yol, uzman sistem tekniğinin uygulanması için uygun bir sahanın olup olmadığını görmek üzere simülasyonu incelemektir. Burada hatırlanması gereken bir konu da, uzman sistemin çözümü için belirli bir problemin uygunluğunu belirlemek üzere göz önüne alınan kriterler, bu yeni tekniğin başarıları ile beraber bir çok başarısızlıklarının sonucudur. Aşağıda bu karakteristiklerin bazıları incelenmiştir /24/.

(1) " Bir veya daha fazla uzman kişlerin varlığı önemlidir ". Uzman sistem ile ilgili genel bir yanlış anlama, uzman sistem metodunun hiçbir şey olamadan bilgiyi oluşturacağı ve uzman olacağıdır. Metodoloji ile ilgili teoriler ve kavramlarda ortaya çıkan avantajlar simülasyonun sanat kısmının ispat edilen metotlar ile yer değiştirme arzusuna sevk etmektedir.

(2) " Bilgi tabanı yeterli derecede sınırlı olamalıdır ve tercihen spesifik alanda olmalıdır ". Bu karakteristikten elde edilen, üretim veya bilgisayar şebekeleri gibi belirli alanlarda sınırlı olan uzman sistem simülasyon sistemleridir. Genel amaçlı bir genel uzman sistem simülasyon sistemi çoğunlukla geçerli değildir ve yeni sistemler spesifik alanlarda olamalıdır.

(3) " Problem ile ilgili bir uzmanın performansını yeni başlayan veya stajerin performansından daha iyidir ". Uzman tarafında

problemin başarılı çözümleri bir amatör tarafından oluşturulan çözümden daha iyi ve daha hızlıdır; ve daha büyük başarı tecrübeye göre elde edilen bilgi veya mental kabiliyete bağlıdır. Bu kriter, uzman kişilerin kabiliyetlerinin amatörlere aktarılması için kullanılabilir.

(4) " Problem kararları çok değişik alternatiflerin göz önüne alınması ve belirsizliklerin de göz önüne alınmasını gerektirir ". Karar alternatifleri, uzman, bu alternatiflerden haberdar olsa bile yeterli büyüklükte sayısal olabilir. Ayrıca, bir alternatifin uygunluğunu belirleyen faktörler, belirsizlik içerebilir. Uzman (pratik tabanlı olarak çalışmıyorsa), mümkün sonuçlar ve delilleri arasındaki destek derecelerini atayabilmelidir. Simülasyon çalışmasının, bu karakteristikler altında uygun olduğunu görmek amacıyla sadece, başlangıç koşulları, otokorelasyon, örnek boyutunun belirlenmesi gibi konular göz önüne alınmalıdır.

(5) " Gezer birimlerin (sisteme hizmet görmek için gelen üniteler) sayısı, ilişkili özellikleri ve alternatif veya karar faktörlerini tanımlayan kullanışlı ilişkiler sınırlıdır ". Temelde bilgileri temsil etmek ve kullanmak için uzman sistemin işleteceği (elemanlar, olaylar, karakteristikler vb.) elemanlar sayıca sınırlıdır. Buna göre mevcut uzman (bazı özel eğitimlerle) hangi elemanın, hangi karakteristiklerinin ve bunlar arsındaki hangi ilişkinin problemle ilgili olduğunu tanımlayabilir. Bu durumun karakterize edilmesindeki diğer bir yol ise işin, uzmanın gerçek düşüncesine göre bazı sınırlı zaman periyodu gerektiren bağımsız işlere bölünüp, bölünmeyeceğini belirlemektir.

Pratik bir görüş açısından ise, simülasyon sistemleri için uzman sistem metodunun kullanılmasının pratikliği göz önüne alındığında, hesaba katılması gereken birçok özellikler vardır. Gelişmenin maliyeti ve kâr arasındaki değişim yüksektir. Ancak bu şekildeki sistemlerin geliştirilmesinin potansiyel kârı, gelişimi kaçınılmaz yapmaktadır

Burada diğer bir konu da ele alınmalıdır. Simülasyon sadece uzman sistem metoduna ihtiyaç duymamakta aynı zamanda uzman sistemlerin de simülasyona ihtiyacı vardır. Eğer yapay zekâ ve uzman sistemlerin mevcut kullanılmaları incelenirse bu kullanımlardan bazılarının zaman alanlı olduğu (yani, karar için bir zaman boyutu yoktur) görülür. Çünkü yapay zekâ uzmanları, zamanın nasıl kullanılacağını henüz bilmemektedirler. Simülasyonun gerçek gücünden birisi, zamana göre olayları ve etkilerini tahmin etme ve projelendirme özelliğidir. Uzman sistemin kullanımı, simülasyon metodu birleştirilene ve kullanılana kadar sınırlı kalmaktadır. Bu yüzden, her iki alan da birbirlerinden yararlandığından dolayı birleşme kaçınılmazdır.

3.7. PROGRAMLAMA DİLLERİNDEKİ GELİŞMELERİN SİMÜLASYON ve UZMAN SİSTEMLERE ETKİLERİ

Japon " Beşinci Jenerasyon Bilgisayar Projesi " ve Amerika'da Austin Teksas'ta Mikro Elektronik ve Bilgisayar Şirketi'nde simülasyonla ilgili geniş çalışmalar yapılmıştır. Bu çabaların her ikisi de uygulama yapmak üzere donanımda ve yazılımdaki, üstünlükleri geliştirmeye yöneliktir. Burada " Beşinci Jenerasyon " nereden geldiği ve ne anlamı verdiği kesin olarak belirli değildir. Yazılım terimlerinde, bilgisayarla iletişim ve bağlantı yolu ile ilgilidir. İlk jenerasyon makina dili, ikincici ASSEMBLER dilidir. Üçüncü jenerasyon dilleri ise, FORTRAN, BASIC, COBOL, C, PASCAL ve ADA gibi genel amaçlı yüksek düzeyli dilleri içermektedir. Bunlar prosedüre yönelik dillerdir /6/.

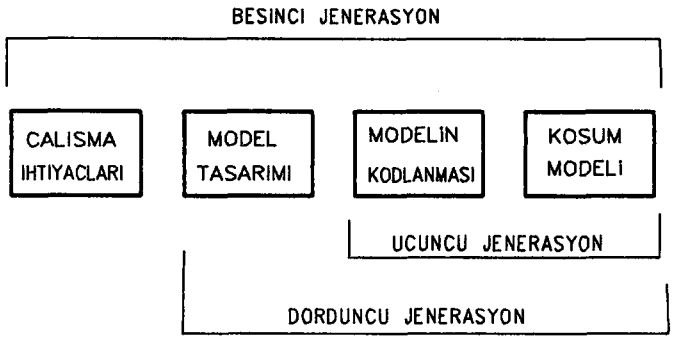
Dördüncü jenerasyon dilleri, bir çok yazılım kategorisini içeren bir şemsiye terimidir. Burada en azından üç temel alan vardır. Bunlar sunuş dilleri (geçerli soru, doğal soru, raporlama, grafik, vb.); özel fonksiyonlarda odaklaşmış özel diller (tablolama, modelleme, analiz, simülasyon vb.) ve tüm uygulamaları oluşturmak için (veri tabanı yöneticileri) verilerin elde edilmesi, modifikasyon ve tanımlama ile ilgili uygulama üreticileridir. Bu diller, son kullanıcı araçları olarak piyasada bulunmalarına rağmen, bu dillerin öğrenilmesi gerektiği bir

gerçektir. Bunlardan sadece bir kaçı programcı gerektirmeyen (tablolama, LOTUS'taki gibi) araçlardır. Bunların bir çoğu kullanıcı eğitimi gerektirmektedir. Gerçekte 4.jenerasyon dilleri, programcılar için mükemmel bir prodüktivite aracıdır.

Beşinci jenerasyon dilleri ise, kendilerini 4.jenerasyondan ayıran bir çok özelliklere sahiptirler. Bunlar doğal dil komutlarının kullanılmasıyla, öğrenmenin, kullanmanın ve desteğin kolaylaştırılması, dökümantasyonun basılması, güncelleştirmenin veya değişikliklerin kolaylaştırılması, bilgisayardan biligisayara aktarımın kolaylaştırılması ve tasarıma bağlı geçici hafızanın kullanılmasıdır. Ayrıca, programcı olmayanlar kendi uygulamalarını geliştirdiklerinde, beşinci jenerasyon dilleri programlama işini tamamen otomatik olarak yapabilmektedirler. Doldurulacak formları, menüler, ikonlar veya karşılıklı sorular şeklinde sunan sistemler gerekmektedir. Arzulanan uygulama, kullanıcıya daha sonra ihtiyaç duymadan, kullanıcının cevaplarını verebilecek niteliktedir. Başka bir değişle, 5.jenerasyon simülasyon sistemleri, 4.jenerasyonda geliştirilen araçları entegre etmektedir ve simülasyon modelleme uzmanı ile beraber, uzman programcının bilgisini elde etmektedir.

Beşici jenerasyon yazılımın etkisini incelemede diğer bir yol ise Şekil 3.17'nin göz önüne alınmasıdır /6/. 3.jenerasyon dilleriyle, ASSEMBLER veya makina diline nazaran, simülasyon modellerinin yazılması ve işletilmesi daha kolaydır. Ancak, modelleme tecrübesi ike beraber programlama uzmanının servisini elde etmek zordur. Bu günün 4.jenerasyon simülasyon dillerinin gelişmesiyle, simülasyonun kullanımı ve kabul edilebilirliği, ileriye doğru büyük bir adım atmıştır. 4.jenerasyon dilleri, sadece program yazmayı kolaylaştırmayıp aynı zamanda model tasarım safhalarında, bir dünya görüşü ve kavramsal rehberlik sağlamaktadır. Ne yazık ki, bugünkü simülasyon, kullanıcının amaçlarını, hedeflerini ve ihtiyaçlarını uygun veri kullanılarak doğru deney tasarımı altında işleterek ve uygun olarak analiz ederek, uygun bir modele dönüştürmek için büyük çapta uzmanlık gerektirmektedir. Beşinci jenerasyon uzman simülasyon sisteminin amacı, kullanıcının ilgili

sistemi, amaçları, hedefleri ve cevap formlarını doğal anlamlı bir dilde tanımlamasını sağlamak ve gersini uzman sisteme bırakmaktır.



Şekil 3.17. Programlama Dillerinin Rolü

Dördüncü ve beşinci jenerasyon simülasyon sistemlerinden, birbirlerine geçiş başlamıştır. Etkileşimli grafik model kurulmasının ve veri girdilerinin artan kullanımı; grafiksel ve animasyonlu çıktı analizi, modelleme, deneysel ve çıktı analizi yapılarının ayrılması, dilin içinde çok sayıda istatistiksel analizin kullanılması gibi tüm adımlar, gerekli ilk adımlar olmaktadır ve ardışık olarak devam etmektedir. Fakat çok yakın bir gelecekte bazı yeni ve radikal olarak farklı simülasyon sistemlerinin gelişimini görmek mümkün olabilir. Yapısal sınırlamalardan dolayı, üçüncü jenerasyon dillerinin geliştirilmesi mümkün değildir.

Mevcut olarak kullanılan simülasyon dilleri, prosedüre yönelik veya komuta yöneliktir. Yani, bir program çözüme erişmek için gerekli adımları açık olarak belirtmelidir. Uzman sistem geliştirilmesinde genelde kullanılan diller, LISP ve PROLOG, nitelik olarak birbirlerinden farklıdır. Orjinal LISP (List Processing) fonksiyonel bir dildir. PROLOG (Programming in Logic) ise açıklayıcı veya tanımlayıcı bir

dildir. PROLOG dilinde, sorular halinde elemanlar ile ilgili deyimler ve kurallarla sadece problemin tanımlanması gerekir. Problemin tanımlanması yeteri kadar hassas ise, problem kolaylıkla çözülebilir. Tekrarlı fonksiyonları destekleme kabiliyeti de (bu fonksiyonlara bağlı olarak tanımlanan) önemlidir. Programlar ve veriler, sembolik işletimi kolaylaştırabilecek güçlü ve üniform bir mekanizma yardımıyla her iki dilde de oluşturulabilir. Bunların dilbilgisi ve yorumlama yapısı, programcıya diğer programlarla beraber çok çeşitli şekilde verilerin (sayılar, kelime parçaları, karakter dizileri ve listeler) işletilmesini sağlayan programların ortaya çıkarılmasına ve programın davranışını değiştirmeye imkan vermektedir.

Simülasyon için PROLOG ve/veya LISP benzeri dillerin kullanılmasından elde edilen avantajlar aşağıda verilmiştir :

- Genel prosedüre yönelik anlamsal ağlara (semantiklere) ilave olarak, mantığa bağlı açıklayıcı anlamsal ağlar sağlamaktadır.

- Programlar ve veriler, form olarak belirlidir ve bu yüzden kolaylıkla işletilebilir.

- Prosedürlerin elemanları diğer programlama dillerinde olduğu gibi girdi ve çıktı parametreleri olarak sabit değildirler ve prosedürler çok sayıda girdi ve çıktıya sahip olabilir.

- Geriye izleme, verilen bir problemin çözümüne ait bir kümenin bulunmasında güçlü bir araçtır. Bu aynı zamanda yüksek düzeyli bir iterasyon formu şeklinde ortaya çıkmaktadır.

- Temel elemanalar (alanlar, değişkenler ve terimler) klasik programlama dillerinde kullanılan dizi ve kayıtlardan daha üstün genel ve esnek bir veri yapısını sağlamaktadır.

4. KUYRUK MODELLERİ İÇİN SİMÜLASYON AMAÇLI BİR UZMAN SİSTEMİN TASARLANMASI

Bekleme hattının diğer bir ismi de kuyruktur. Kuyruk sistemi ise bir bekleme hattını içeren bir sistemdir. Kuyruk teorisi, analizcilerin kuyruk sistemlerinin davranışlarını tanımlamasına yarayan bir yönetim bilim dalıdır.

Kuyruk problemlerine olan yaklaşımlar üç sınıfta toplanmaktadır. İlk yaklaşımsa, sistem tasarımı deneme-yanılma esasını gözönüne almaktadır. Bu yaklaşım, basit sistemler için maliyeti etk,leyici niteliktedir. Birkaç alternatif incelendikten sonra, optimum bir tasarım belirlenir. Ancak, incelenen sistem daha karmaşık olduğunda, deneme yanılma prosedürleriyle optimal bir tasarımın bulunması oldukça zorlaşmaktadır. Bu yaklaşım, yeni bir binaya kaç tane asansör konacağını belirlemede olduğu gibi ilk tasarım değişikliklerinin zor veya imkansız olduğu yeni sistemlerin tasarlanmasında kullanışlı değildir.

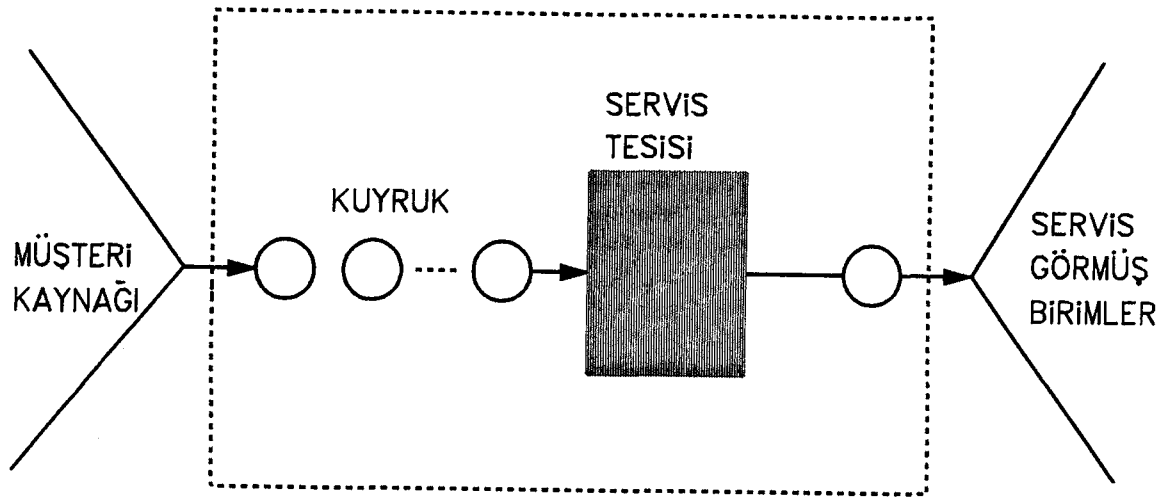
Diğer bir yaklaşım ise, analiz için analitik metotların kullanılmasını kapsamaktadır. Standart matematik ve istatistiksel metotlarla sistemin davranışını tanımlayan ifadeler geliştirilmiştir. Bu ifadeler, maliyet faktörleriyle birleştirildiğinde analist, sistemin tasarımı hakkında bir fikir sahibi olabilir. Bu yaklaşım mümkün olan her yerde kullanılabilir.

Ancak birçok kuyruk sistemlerinin analitik metotlarla analiz edilmesi oldukça zordur. Bu durumda analist, nümerik bir analiz metodu yani bir simülasyon yaklaşımını kullanmalıdır. Kuyruk sistemlerine simülasyon yaklaşımı ile sistemin performansı hakkında daha gerçekçi tahminler yapılabilir. Bu bölümde açıklanan ise kuyruk modellerinin bilgisayarla simülasyonunda, simülasyon programlarının (modellerinin) otomatik olarak oluşturulmasını gerçekleştirecek olan simülasyon amaçlı bir uzman sistemim tasarlanmasıdır.

4.1. KUYRUK SİSTEMLERİNİN GENEL YAPISI

Bütün kuyruk modelleri, müşterilerin bir servis tesisine gelişlerini içermektedir. Servis tesisinde istenilen servisi bekleme ve servisi alma için belirli bir zaman geçer. Müşteriler genellikle insan olarak düşünülmelerine rağmen, tamircide tamir edilen arabalar, üretimin bir sonraki sonraki safhasına geçen yarı mamuller, inmek için bekleyen uçaklar veya bilgisayarda işlenen veriler de müşteri olabilirler. Bir servis tesisi, berber veya kuaför gibi bir tek kişi olabildiği gibi, bir ameliyat ekibi gibi birden fazla kişiden de oluşabilir.Bir servisçi, kutu içecek veren, kısımları birleştiren, veri işleyen bir makina olabildiği gibi, havaalanı pisti veya petrol rafineri tesisi gibi kompleks bir tesis de olabilir.

Şekil 4.1'de /8/ görüldüğü gibi bir kuyruk sistemi 4 kısımdan oluşmaktadır. Bunlar potansiyel müşteri kaynağı (talep birimleri), kuyruk, servisçi, servis görmüş talep birimleridir. Uygun modellerin formülasyonundan önce bunların bazı özelliklerinin ve karakteristiklerinin dikkate alınması gerekmektedir. Burada potansiyel müşteri kaynağı, kuyruk ve servisçiler detayla incelenmektedir. Genelde servis görmüş talep birimlerinin sistemi terkettiği kabul edilir.



Şekil 4.1. Kuyruk Sistemi

Günlük hayatta sık sık kuyruk sistemleriyle karşılaşılmaktadır. Örnek olarak, bir uçakla seyahat esnasında, seyahat sonuna kadar çok sayıda kuyruğa girilir ve çıkılır. İlk olarak, bilet acentasındaki servisi elde etmek için beklenir. Daha sonra acenta, istenilen uçuşta yer olup olmadığını kontrol etmek üzere bir kuyruğa girer. Bilet alındıktan sonra, uçuşun yapılacağı kapı önünde bilet kontrolü için beklenir. Sonra, uçağa gitmek için beklenir ve uçağa binilir. Bundan sonra uçak, kalkmak için bekleyen uçak kuyruğunun bir elemanı olur. Uçak varış noktasına geldiğinde ise, inmek için havaalanı üzerinde tur atar. Uçak indikten sonra, çıkış kapısını arar ve yolcular uçaktan çıkmak için yeni bir kuyruğa girerler. Son olarak, bagajların alınması ve taksiye binilmesinde de bir kuyruk oluşacaktır.

Yönetim biliminde veya yöneylem araştırmasında, bir bekleme hattı, kuyruk olarak adlandırılır. Araştırma alanı olarak kuyruk teorisi, yöneylem araştırması metodolojisinin en zengin alanlarından birisidir. Spesifik durumları gösteren modellerin sayısı, yıllara bağlı olarak artmıştır, ve yeni modeller bugün bile ortaya çıkmaktadır. Kuyruk teorisi, ilk kantitatif metodlardan birisidir. Bu teorinin orijinal eseri, ismi matematiksel kuyruk modellerinde kullanılan, çok geniş ihtimal dağılım sınıfıyla ilgili olan Danimarkalı telefon mühendisi A.K. Erlang tarafından, 1905 yılında 1909 sayfa halinde yayınlanmıştır /24/. Erlang'ın karşılaştığı temel problem, yeni tasarlanan ve kullanılmaya başlanan teçhizatla ilgili olarak telefon servisine ait düzensiz taleplerin etkisini belirlemekti. Erlang, servis verme zamanı değişirken, telefon şirketinin acil ve etkin servis (servis tesisi) verme kabiliyetine bağlı olarak, bilinmeyen geliş (telefon konuşmaları) dağılımının etkisini ölçmüştür /25/.

Kuyruk teorisinin uygulanacağı örnekler birbirinden farklı olmak üzere çok sayıdadır. Kuyruk içeren sistemlerin tasarımında, tahmini bekleme zamanı, kuyruk uzunluğu vb. gibi değerlerin tahmin edilmesinde kuyruk teorisi veya dijital simülasyon kullanılırsa, kuyruk elemanlarının (veya talep birimlerinin) kuyrukta bekleme zamanları azalır. Kuyruk uygulamaları için bir örnek liste Tablo 4.1'de verilmiştir.

Tablo 4.1. Kuyruk Sistemi Uygulamaları

SİSTEM	TALEP BİRİMLERİ	SERVİSÇİLER
Paralı yol	Otomobiller,kamyonlar,vb.	Gişe
Atölye	Tezgahlar	Bakım işçileri
Atölye	İşler	Tezgah merkezleri
Bakım tesisi	Tezgahlar	Bakım işçisi
Resepsiyon	İnsanlar	Resepsiyon memuru
Garaj	Kamyonlar	Tamirci
Hastane	Hastalar	Hemşireler
Ambar	Paletler	Vinçler
Havaalanı	Uçaklar	Pist
Üretim hattı	Kutular	Paketleyici
Ambar	Siparişler	Siparişi alan
Yol	Arabalar	Trafik ışığı
Süpermarket	Müşteriler	Kasa
Çamaşırhane	Kirli eşyalar	Çamaşır makinaları
Muayenehane	Hastalar	Doktor,hemşire,lab.
Bilgisayar sistemi	Programlar,işler,mesajlar	Bilgisayar
Kayak merkezi	Kayakçılar	Telesiyej
Liman	Gemiler	Liman tesisleri
Kayıt bürosu	Öğrenciler	Danışmanlar
Mahkeme	Davalar	Duruşma
Lokanta	Müşteriler,siparişler	Garsonlar,mutfak,masalar
Öğretmen ofisi	Öğrenciler	Öğretim görevlisi
Bilgisayar	İşler	CPU,disk,disketler
Bilet gişesi	Futbol seyircileri	Gişe görevlisi
Testere atölyesi	Odunlar	Testereler
Toplu Taşıma	Yolcular	Otobüsler,trenler
Telefon	Aramalar	Operatör

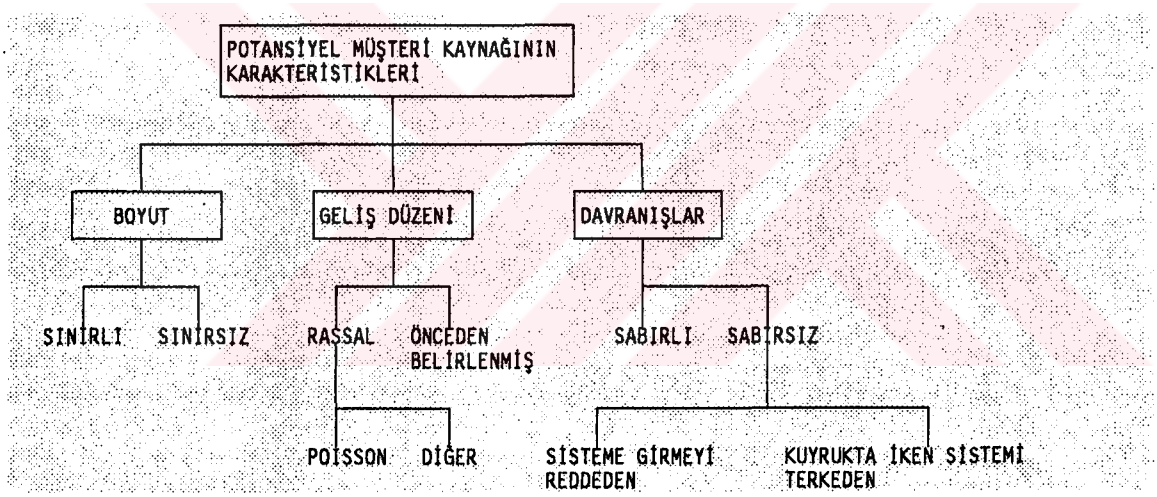
Kuyruk teorisi uygulandığında, yönetimin amacı genelde iki tür maliyeti minimize etmektir. Bunlar servis maliyeti ve bekleme zamanı maliyetleridir. Kuyruk teorisi, kuyruk sisteminin istatistiksel olarak açıklamasını yaptıktan sonra, analizci işletmenin amaçlarını en iyi

karşılamak için sistemi tasarlamak üzere, bekleyen müşterilerin maliyetleriyle, çeşitli hizmet verme maliyetlerini tahmini olarak belirler /26/.

4.1.1. Potansiyel Müşteri Birimlerinin Karakteristiği

Şekil 4.2'de /26/ görüldüğü gibi, girdi kaynağı olarak da belirtilen potansiyel müşteri kaynağı, hangi tip kuyruk modelinin uygulanacağına karar verilmesinde önemli olan 3 karakteristiğe sahiptir. Bunlar:

(1) Potansiyel müşteri kaynağının boyutu
(2) Kuyruk sistemindeki gelişlerin düzeni
(3) Talep birimlerinin davranışları

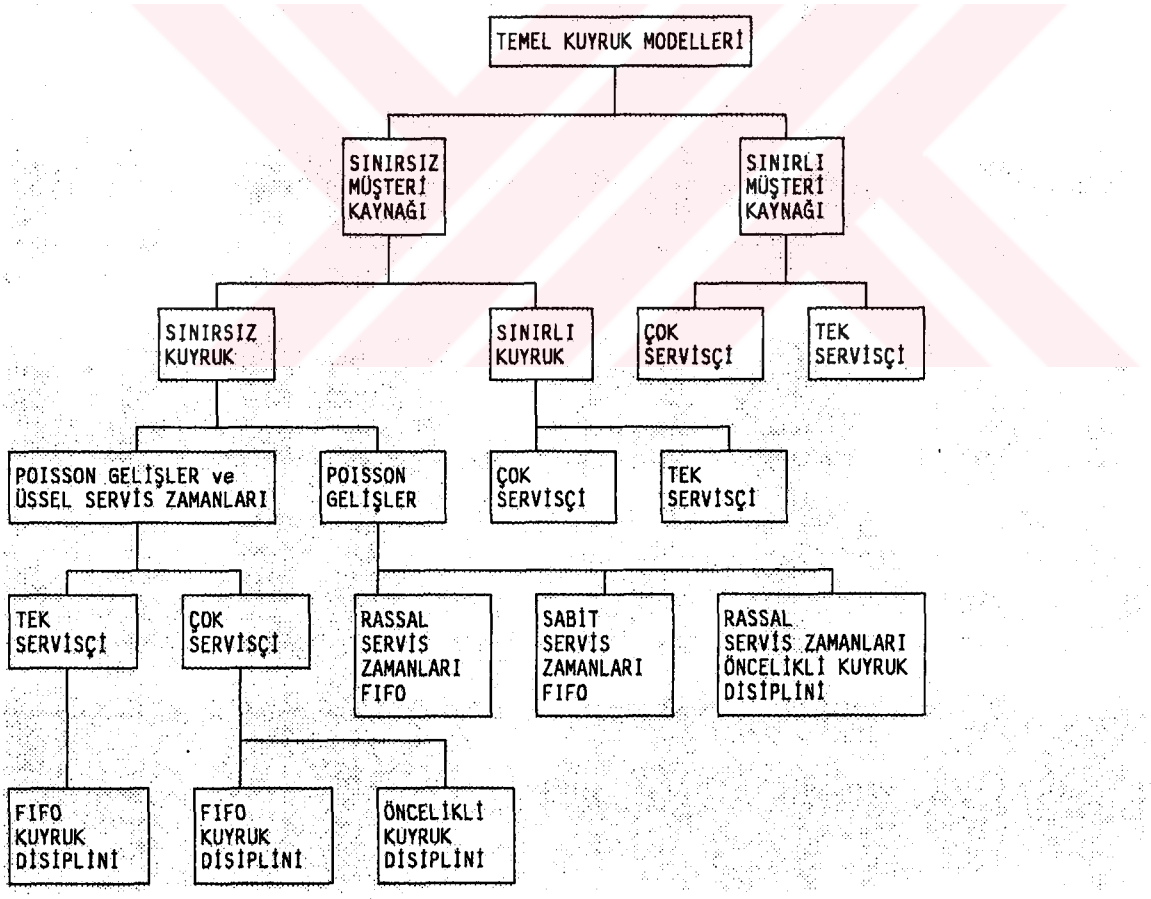


Şekil 4.2. Potansiyel Müşteri Kaynağının Karakteristikleri

1. Potansiyel Müşteri Kaynağının Boyutu : Bu faktörün kuyruk modellerinin seçiminde çok büyük önemi vardır. (Sınırlı ve sınırsız potansiyel müşteri kaynağıyla ilgili alternatifler Şekil 4.3'de /26/ gösterilmiştir). Potansiyel müşteri kaynağının sonsuz sayılabildiği kuyruk sistemleri genelde analitik modellemeye daha yatkındır. Kuyruk sistemlerinde sonsuz potansiyel müşteri kaynağına örnek olarak paralı yoldaki otomobiller ve bir hastanenin acil servisindeki hastalar gösterilebilir. Potansiyel müşteri kaynaklarının çok sınırlı olduğu

sistemlere uygulanabilen kuyruk modellerinin oluşturulması oldukça zordur.Sonlu veya sınırlı kaynağa örnek olarak, bozulduğunda servisi tarafından tamir edilen üç kişisel bilgisayar ve belirli bir ders için öğretim görevlisinin çalışma saatlerinde odasının önünde bekleyen öğrencilerden oluşan kuyruk sistemlerini örnek gösterebiliriz.

Müşteri kaynağının bir elemanı veya elemanları servis görüyor iken, bir gelişin olasılığı görülür bir ölçüde değişiyorsa bu bir sonsuz potansiyel müşteri kaynağıdır. Bir potansiyel müşteri kaynağında üç talep birimi varsa ve biri servis görüyorsa, sisteme bir başka gelişin olasılığı belirli bir ölçüde azalır çünkü potansiyel müşteri kaynağının boyutu %33.3 azalmıştır. Genelde, kuyruk modeli uygulamalarında 200'den fazla bir potansiyel müşteri kaynağı boyutu sonsuz olarak kabul edilir.



Şekil 4.3. Temel Kuyruk Modelleri

2. Kuyruk Sistemindeki Gelişlerin Düzeni : Kuyruk modelleri, genelde müşterilerin sisteme teker teker geldiklerini kabul ederlerse de, bazı modeller partiler halinde gelişlere izin verirler. Şehirlerarası bir otobüsün yoldaki bir mola yerine gelmesi halinde oluşan müşteri gelişleri, ikinci tip modele örnek gösterilebilir. İncelenen bir peryottaki müşteri gelişlerinin aynı dağılıma uyduğu çoğu kuyruk modelleri tarafından kabul edilir. Ancak hemen hemen tüm bekleme hatlarında meşgul peryotlar olacağından bu her zaman doğru değildir. Bu nedenle belirli bir model sadece dolu saatler için, diğer bir model de sadece boş peryotlar için uygun olabilir. Servis düzeninin müşteri gelişlerinden bağımsız olduğu genellikle kabul edilse de, bu bazı tip servis tesisleri için doğru olmayabilir. Örneğin, bir süpermarket kasa memuru, sırada bekleyen üç dört kişi olduğunda, sırada kimse olmadığı zamandan daha hızlı çalışabilir /24/.

Talep birimleri, kuyruk sistemine, önceden belirli bir plana göre veya rastgele bir düzende gelirler. Bir doktorun muayenehanesinde bekleyen hastalar örneğinde olduğu gibi gelişlerin planlı olduğu durumlarda analitik kuyruk modelleri uygun değildir. Eğer gelişler rastgele ise gelişler arası sürelerin olasılık dağılımının belirlenmesi gereklidir. Gelişler arası sürenin olasılık yoğunluk fonksiyonunun üssel fonksiyon olduğu durumlarda da talep birimlerinin sisteme Poisson prosesine göre geldikleri matematiksel olarak kanıtlanmıştır.Kuyruk sistemlerinde Poisson gelişlerine çok sık rastlanır. Belirli bir zaman aralığındaki gelişlerin önceki zaman aralıklarındaki gelişlerden bağımsız olduğu durumlar Poisson gelişleri olarak kabul edilir. Bu temel özelliğe göre, gelecek bir olayın olasılığı, sistemin önceki durumlarından bağımsız olarak yalnız sistemin şimdiki durumuna bağlıdır.

Poisson olasılık yoğunluk fonksiyonu t zaman peryodundaki n gelişlerin olasılığını verir. Poisson olasılık yoğunluk fonksiyonunun matematiksel eşitliği şöyle verilir:

$$P_n(t) = \frac{e^{-\lambda t}(\lambda t)^n}{n!}$$

n = gelişlerin sayısı

t = zaman aralığının boyutu

λ = birim zamandaki ortalama geliş hızı

Bir çok kuyruk sistemlerinin Poisson prosesine uyan rassal gelişleri olmasına rağmen, gelişler arası zamanların üssel olmayan bir düzende dağılması da mümkündür.Bu nedenle, herhangi bir kuyruk problemine nasıl yaklaşılacağına karar verilirken, gelişler arası sürenin dağılım ve parametrelerinin belirlenmesi gereklidir.

3. Talep Birimlerinin Davranışı : Potansiyel müşteri birimlerinin dikkate alınması gereken son karakteristiği talep birimlerinin davranışıdır. Talep birimleri sabırlı olanlar ve olmayanlar olarak 2 şekilde sınıflandırılabilir. Sabırlı bir müşteri veya talep birimi sistemin durumunu dikkate almadan kuyruk sistemine gelen ve sistemde bekleyen kişidir. Sabırlı olmayan bir talep birimi, sisteme girmekten vazgeçer veya servis görmeden sistemi terkeder. Genelde kuyruk modellerindeki talep birimlerinin çok sabırlı olduğu kabul edilir.

4.1.2. Kuyruk Uzunluğunun Özelliği

Kuyrukların bu karakteristiği genelde potansiyel müşteri kaynağının boyutuna ve bazen de potansiyel müşteri kaynaklarının davranışına bağlıdır. Kuyruk modellerinin uygulanmasında, kuyruk sınırlı veya sınırsız olabilen maksimum uzunluğuyla karakterize edilir. Limitasyonlar genelde müşteri davranışına ya da kuyruk için boş kalan yere göre nitelendirilir. Sınırlı kuyrukları olan kuyruk sistemleri için çok az seçenekler vardır. Genelde, eğer talep birimlerinin kuyruğun uzunluğuna bakmadan kuyruğa girdikleri varsayılırsa, bir analitik kuyruk modelinin bu sisteme uygulanması olasılığı yükselir.

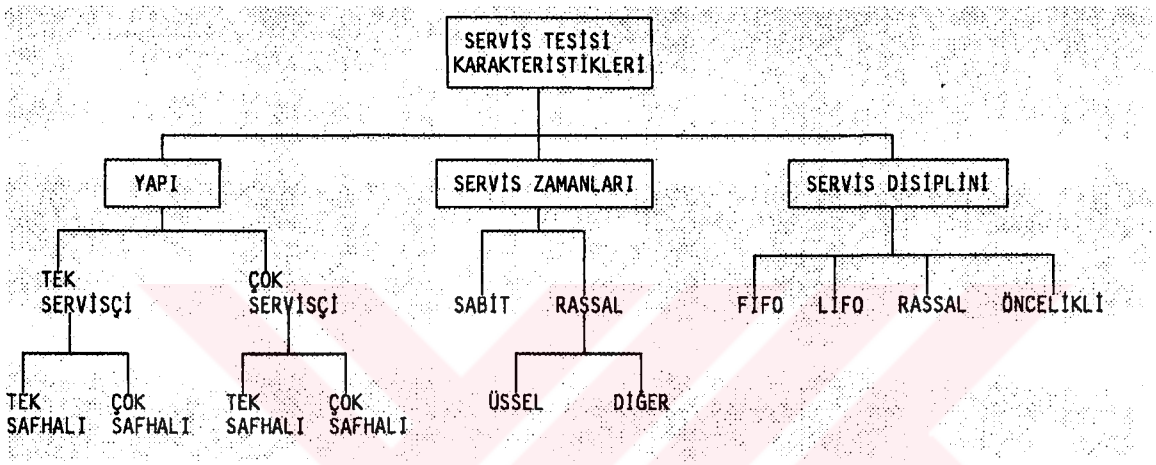
4.1.3. Servis Tesisinin Karakteristikleri

Şekil 4.4'de görüldüğü gibi, servis tesisinin 3 temel özelliği / / vardır /26/. Bunlar:

(1) Kuyruk sisteminin yapısı,

(2) Servis zamanlarının dağılımı,

(3) Servis disiplinidir.



Şekil 4.4. Servis Tesisinin Karakteristikleri

1. Kuyruk Sisteminin Yapısı : En basit kuyruk sistemi, belirli bir anda bir müşteriye hizmet veren tek servisçili bir tesisi kapsamaktadır. Bu sistemde, ilk gelen müşteri servis görürken, sonra gelen müşteri kuyrukta beklemek zorundadır. Tek servisçili, tek safhalı bu kuyrukta, gerekli tüm servisler, müşteriler sistemi terketmeden önce tamamlanır. Bekleme hattının kendisi, sinema gişesi önünde oluşan kuyrukta olduğu gibi fiziksel bir müşteri kuyruğu olmak zorunda değildir. Bekleme hattı, servis sırasının belirli veya belirsiz olduğu (mağazalarda olduğu gibi fiziksel kuyrukların oluşması mümkün olmayan yerlerde bir sıraya göre) bir müşteri grubu şeklinde olabilir. Burada müşteriler, fiziksel olarak bir araya gelmezler. Örneğin, çok sayıda araştırmacı (müşteri) geniş coğrafik alanlara dağılmış bilgisayar terminallerde işlemlerini yapmalarına rağmen, hepsi merkezi bir bilgisayar sisteminde

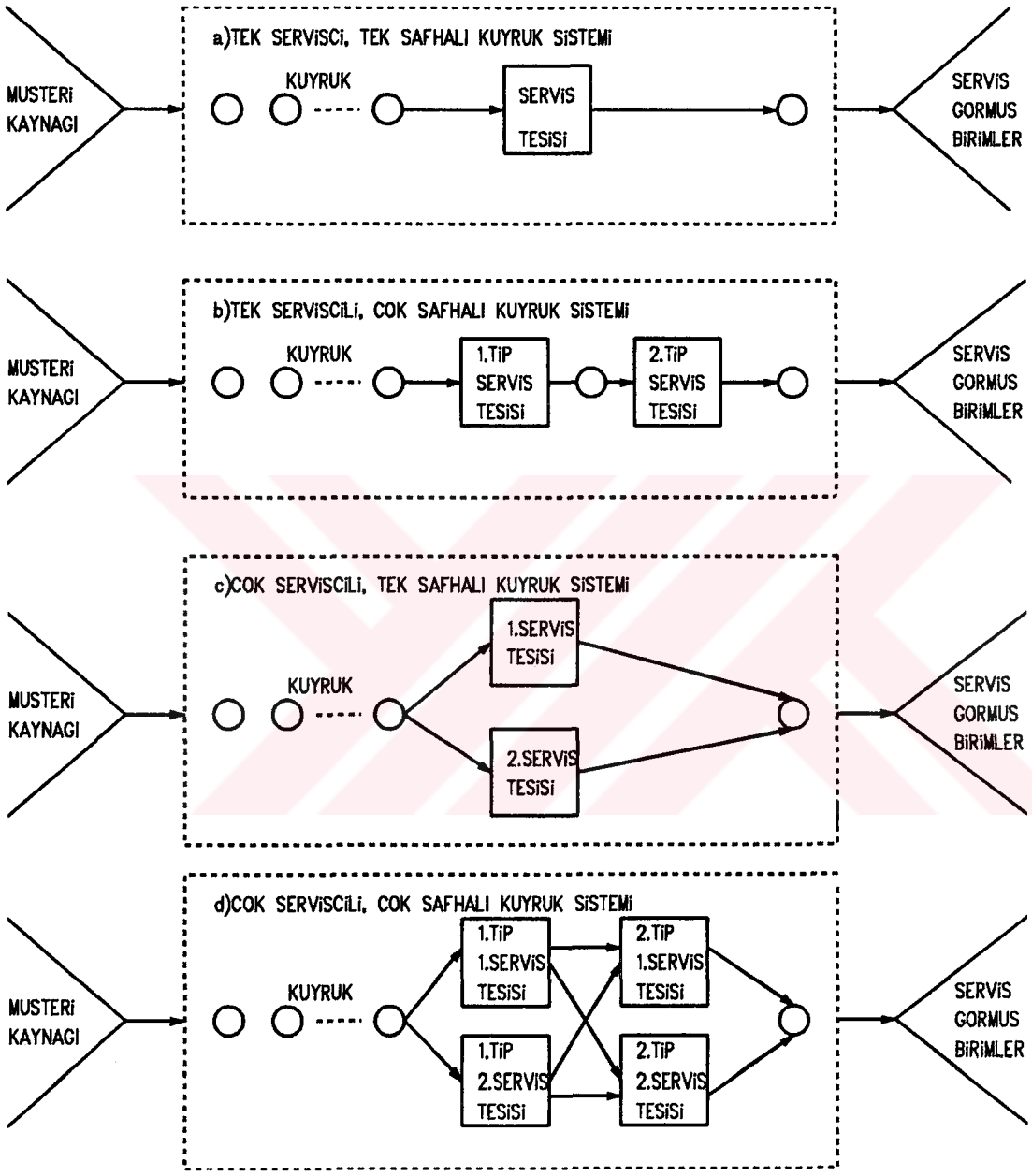
toplanabilirler veya havada binlerce kilometrelik alana dağılmış olan uçaklar büyük bir havaalanına inmek isteyebilirler.

Birden fazla servis tesisinin bulunduğu, çok servisçili, tek safhalı kuyruk sistemi ise ilk sisteme göre oldukça karmaşıktır. En basit durumda, her tesis belirli bir servis verir, tek bir bekleme hattı oluşur ve ilk müşteri boş olan ilk servisçiye gider. Çok sayıda banka, müşterilerin tek bir hatla vezneye gittikleri böyle bir sistemi kullanmaktadır. Gelen müşterilerin, bir servisçiyi seçmeleri ve ayrı bir servisçi hattında beklemeleri durumunda oldukça farklı karakteristikler uygulanır.

Kuyrukların incelenmesinde diğer bir yol ise, müşterinin, sistemi terketmeden önce geçtiği servis noktalarının veya safhalarının ele alınmasıdır. Bu kuyruk sistemlerinin en basiti, tek servisçili, çok safhalı kuyruk olarak belirtilir. Burada, sistemi terketmeden önce, her müşteri iki veya daha fazla türde servis görmektedir. Bu durum, bir mağazadan çadır gibi büyük hacimli bir parçanın satın alınması esnasında ortaya çıkabilir. Öncelikle, siparişinizi söylemek için satış memurunu beklemek zorundasınız. İkinci olarak, satış memuru bankaya telefon ederek kredi kartınızın limitini kontrol eder ve son olarak da, paketleme alanından çadırınızı almak istediğinizde üçüncü bir kuyruk oluşacaktır. Böyle bir kuyruk, üretimin farklı kademelerinde yarı mamaullerin işlem beklediği üretimde uygulanabilir.

En karmaşık kuyruk sistemi ise, çok servisçili, çok safhalı kuyruktur. Öyle bir sistem, hastaneye kolayca uygulanabilir. Öncelikle, hasta kabul bölümünde sıraya girilerek kayıt yaptırılır. Yapılan kayıda göre ilgili servis birimleri önünde kuyruğa girilir. Muayeneden sonra, reçeteye göre eczane önünde kuyruğa girilir ve daha sonra hastaneden çıkılır.

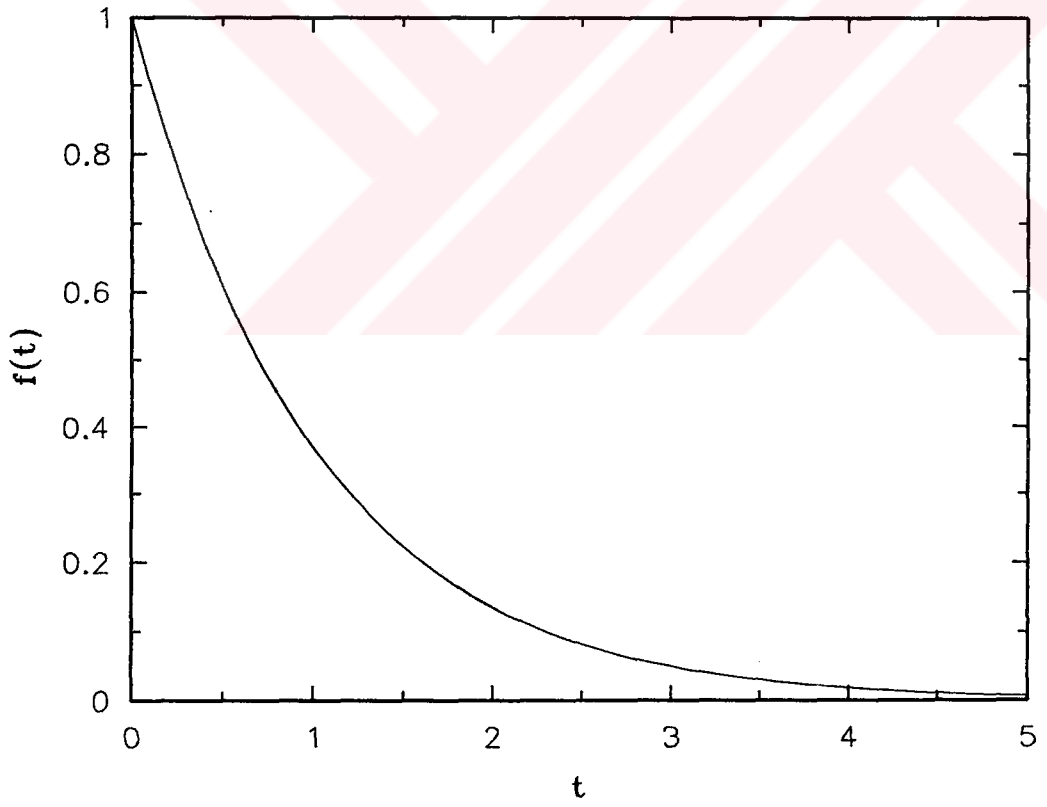
Yukarıda açıklanan kuyruk yapıları genel olarak Şekil 4.5'de gösterilmiştir /26/.



Şekil 4.5. Farklı Kuyruk Sistemlerinin Yapısı

2. Servis Zamanlarının Dağılımı :τServis zamanları sabit ve rassal olarak iki şekilde dağılmıştır. Eğer servis zamanı rassal bir değişken ise, analizcinin bu rassal değişkenin dağılımını belirlemesi gerekir. Bir çok kuyruk durumunda servis zamanları üssel olarak dağılmıştır. Bu durumlarda uygulanabilecek bir kuyruk modeli bulma olasılığı daha yüksektir.

Servis zamanları üssel dağılmış ise, göreceli olarak uzun olan bir servisin olasılığı daha az olur. Örnek olarak telefon konuşmalarının uzunluğunun üssel dağıldığı gösterilebilir. Bu durum Şekil 4.6'da gösterilmiştir /26/.



Şekil 4.6. $f(t)= \mu e^{-\mu t}$ (Üssel Dağılımın) Grafiği

3. Kuyruk Disiplinleri : Bir kuyruk sisteminin en önemli özelliği, müşterilere verilecek servis sırasıdır. Müşterilerin servise giriş sırasına kuyruk disiplini adı verilir. Temelde, aynı kuyruk yapısı için geliştirilen matematiksel modeller, uygulanan disipline bağlı olarak farklılık gösterebilir.

En yaygın kuyruk disiplini, müşterilerin geliş sırasına göre servis aldığı fiziksel FIFO (İlk gelen, ilk çıkar) disiplinidir. Satış mağazaları ve devlet daireleri genelde bu sistemi kullanmaktadır. Birçok başarılı kuruluşlar, gelen müşterilere bir numara vererek FIFO'yu kullanmaktadır. FIFO, burada incelenen kuyruk modellerinin tümüne uygulanmaktadır.

Daha az yaygın olan diğer bir disiplin ise, LIFO (son giren, ilk çıkar) disiplinidir. Asansörlerin kullanılması gibi LIFO'nun uygulanabileceği bazı durumlar mevcuttur. Çok sayıda kargo taşıma durumunda, son yüklenen parça ilk olarak çıkartılır. Bu durumlara ait kuyruk şekilleri pek önem taşımamaktadır. Ancak, aşağıdaki örnek için LIFO disiplinin uygulanması bir zorunluluktur.

Açık bir atölyede bulunan haddehaneye çelik ingotlar gelmektedir. Bu ingotlar geldikleri anda oldukça sıcaktır ve zamanla yapılarında sıcaklık değişimi olmaktadır. Bu nedenle, haddelenmeden önce yüksek bir ısı dağılımı ve haddeleme işlemi için yüksek bir sıcaklık gerektiğinden, ingotlar ön ısıtma fırınlarına yerleştirilmelidir. İngotlar geldiklerinde fırınlar doluysa, beklemek zorunda kalacaklardır. Bu esnada ingotlar beklerken soğuyacaklardır. İngotlar ne kadar çok beklerlerse, fırınlarda da o kadar uzun kalmaları gerekecektir.

Enerji maliyetini minimize etmek için, hattaki en son (en sıcak) ingot ön ısıtma fırınına ilk önce yerleştirilmelidir. Böylece, LIFO disiplini kolayca uygulanmış olur.

Diğer bir kuyruk disiplini ise, SIRO (rassal sırada servis) disiplinidir. Bir sinemada, arada içecek dağıtan eleman SIRO'nun en göze çarpan örneklerinden birisidir. SIRO ile ilgili en önemli bekleme hattı

durumu, telefon konuşmasında ortaya çıkmaktadır. Hat boşken gelen telefon konuşması, yapılan ilk konuşmadır. Buradaki bekleme hattını, "meşgul" sinyali almış olan kişiler olarak düşünebiliriz. Ancak, bu konuşmalara ait hiç bir kayıt tutulmamaktadır. Bu yüzden, hangi konuşmanın ilk olarak yapılacağını belirlemek için bir yöntem yoktur ve arayan kişinin konuşması için tek yol, numarayı tekrar çevirmektir. Servis sırası bu nedenle rassal olarak gözönüne alınabilir. Modern telefon cihazlarında ise, tüm arayan kişiler konuşma yapılana kadar beklemeye alınır. Böyle sistemlerde genellikle FIFO disiplini uygulanır.

Bunlarla beraber çok çeşitli kuyruk disiplinleri de mevcuttur. Diğerleri üzerinde önceliğe sahip müşterilerin bulunduğu durumlar oldukça yaygındır. Örneğin, bir hastanenin acil servisinde, durumu daha ciddi olan hastalara ilk olarak bakılır. Bilgisayar sistemleri, önceliklere göre fonksiyonlarını yerine getirir ve bu yüzden en yüksek öncelikli işler zamanlarına bakılmadan diğerlerinden önce yapılır.

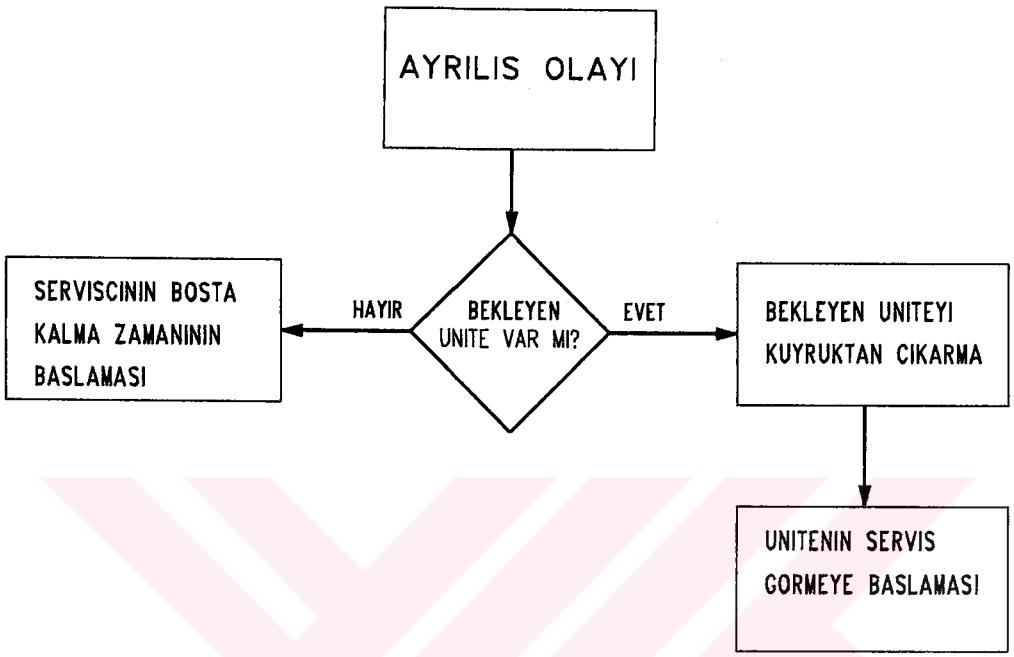
4.2. KUYRUK SİSTEMLERİ SİMÜLASYONUNUN TEMEL YAPISI

Kuyruk sistemlerinin simülasyonuna geçmeden önce, sistem durumu, olaylar ve saat zamanı kavramlarını açıklamak gereklidir. Sistem durumu, sistem içindeki ünitelerin adedi ve servisçinin meşgul veya boş olma durumudur. Olay, sistemin durumu içinde ani bir değişikliğe neden olan durumların kümesidir.

Tek kanallı bir kuyruk sisteminde, sistemin durumuna etki edebilecek sadece iki mümkün olay vardır. Bu iki olay, sistemin içine bir ünitenin girmesi (Geliş Olayı) veya bir ünitenin servisinin tamamlanmasıdır (Ayrılış Olayı). Kuyruk sistemi, servisçi, servis gören ünite (eğer biri servis görüyorsa) ve kuyruktaki diğer ünitelerden (eğer bekleyen ünite varsa) oluşmaktadır.

Eğer servis henüz tamamlanmış durumda ise simülasyon, Şekil 4.7'de /8/ verilen akış diyagramında gösterildiği biçimde ilerler. Şekil

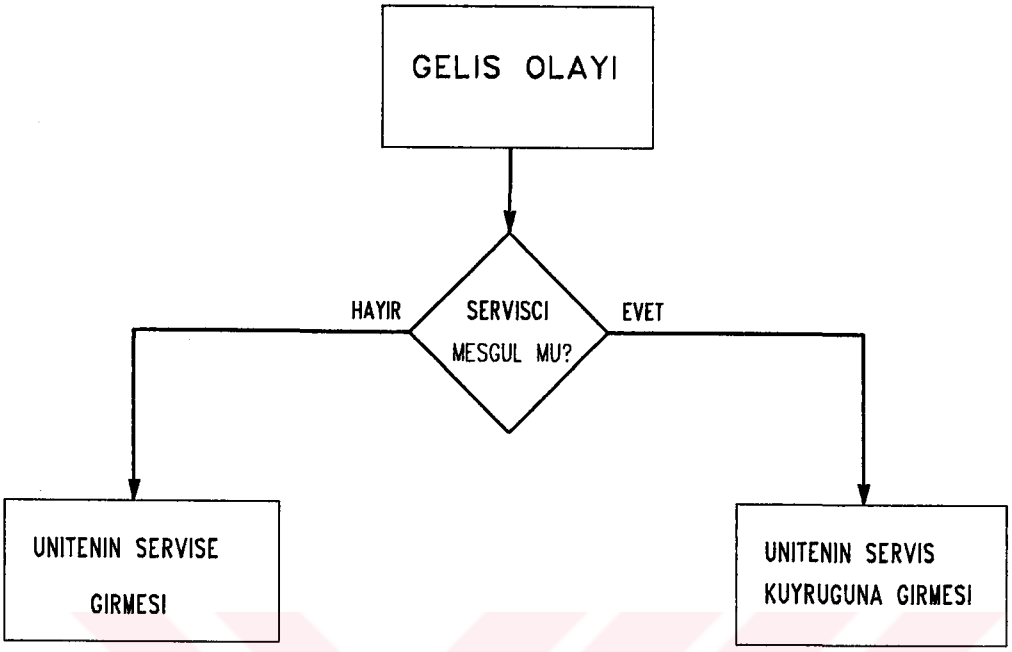
4.7'deki servisçi, sadece mümkün iki şarta sahip olacaktır. Bu şartlar, servisçinin meşgul olması veya boşta kalmasıdır.



Şekil 4.7. Henüz Tamamlanan Servisin Akış Diyagramı

İkinci olay, sisteme bir ünite girdiği zaman meydana gelir. Bu durumun akış diyagramı Şekil 4.8'de gösterildiği gibidir /8/. Sisteme bir ünite geldiğinde, servisçi meşguldur veya boştadır. Daha önceki ünite servisteyse, servisçi meşgul durumdadır. Bu durumda gelen ünite, servis görmek için kuyruğa girer.

Bu iki olay, kuyruk simülasyonları için temel kavramlardır. Sistem giriş ve sistemden çıkış olayları olmadan sistemin değerlendirilmesi söz konusu değildir. Bu açıdan, bu iki olayın ne zaman başlayıp, ne zaman bittiği kesin olarak saptanmalıdır.



Şekil 4.8. Bir Ünitenin Geliş Durumundaki Akış Diyagramı

Gelen ünitenin izlediği, davranış biçimi, Şekil 4.9'da görülmektedir. Eğer servisçi meşgul ise, gelen ünite kuyruğa girer. Eğer servisçi boşta ve kuyruk boş durumda ise, gelen ünite servise girer. Eğer servisçi boşta ve kuyruk boş değil ise gelen ünitenin hemen servise girmesi mümkün değildir.

		Kuyruğun Durumu	
		Boş Değil	Boş
Servisçinin Durumu	Meşgul	Kuyruğa Giriş	Kuyruğa Giriş
	Boş	Mümkün Değil	Servise Giriş

Şekil 4.9. Bir Ünitenin Geliş Durumundaki Davranışları

Bir servisin tamamlanmasından sonra, servisçi boşta kalabilir veya bir sonraki ünite ile meşgul olur. Kuyruk durumuna göre bu iki sonucunun ilişkileri Şekil 4.10'da görülmektedir. Eğer kuyruk boş değil ise, diğer ünite servise girecektir ve servisçi meşgul durumda olacaktır ve eğer servis tamamlandığında kuyruk boş ise servisçi de boşta kalacaktır. Bu iki olasılık, Şekil 4.10'de taralı bölgelerde gösterilmektedir. Servis tamamlandığı zaman, eğer kuyruk boş ise, servisçinin meşgul olması mümkün değildir. Benzer şekilde, servis tamamlandıktan sonra, kuyruk boş değil ise, servisçinin boş olması mümkün degildir.

		Kuyruğun Durumu	
		Boş Değil	Boş
Servisçinin Durumu	Meşgul		Mümkün Değil
	Boş	Mümkün Değil	

Şekil 4.10. Servis Tamamlandıktan Sonra Servisçinin Mümkün Durumları

Şimdi, simüle edilmiş bir zaman içinde oluşan olayların nasıl tanımlandığını inceleyelim. Kuyruk sistemlerinin simülasyonu, genellikle gelecekte ne olacağının belirlenmesi için olay listesinin muhafazasını gerektirir. Bir olay listesi, kuyruk sistemi içindeki her ünite için meydana gelen farklı tip olayların zamanlarını gösterir. Bu zamanlar, simülasyon süresi içinde meydana gelen olayların belirlendiği "saatte" kaydedilmektedir. Simülasyonda olaylar, çoğunlukla rassal olarak meydana gelir. Rassallık, belirsizlikleri belirlemek üzere gerçek yaşamı yansıtmaktadır. Örneğin, bir sonraki müşterinin yazar kasaya ne zaman geleceği veya bir banka veznedarının bir işlemi ne kadar sürede tamamlayacağı önceden kesin olarak bilinmemektedir.

Gerçek hayatı yansıtan rassallık, "rassal sayıların" kullanılmasıyla mümkün olmaktadır. Rassal sayılar, üniform ve (0,1) aralığında bağımsız olarak dağılırlar. Rassal dijitler (haneler),

{0,1,2,....,9} kümesinde üniform olarak dağıtılmaktadır. Rassal dijitler, her rassal sayı için uygun dijit sayısını seçerek ve seçilen değerin soluna desimal noktayı yerleştirerek, rassal sayıların oluşturulmasında kullanılmaktadır. Uygun dijit sayısı girdi amacı için kullanılan girdilerin doğruluğuyla sağlanmaktadır.

Rassal sayılar, çeşitli yöntemlerle türetilebilir. Sayılar, bir küme prosedürü kullanarak türetilirse, bu sayılar hayali-rassal sayılar olarak adlandırılırlar. Metod bilindiğinden, simülasyondan önce hangi sayı sırasının oluşacağını bilmek mümkündür.

4.3. SIMAN DİLİ İLE MODELLEME

SIMAN modelleme yapısı, Zeigler ve Ören /27/ tarafından geliştirilen sistem teorisine dayanmaktadır. Bu yapı içinde, sistem modeli ve deneysel yapı arasında temel bir ayırım söz konusudur. Sistem modeli, sistemin statik ve dinamik karakteristiklerini tanımlamaktadır. Buradaki deneysel yapı, spesifik çıktıların elde edilmesinde, modelin işletileceği deneysel şartları tanımlamaktadır. Herhangi bir model için, çıktı kümeleri halinde deneysel yapılar oluşabilir. Model yapısını ve deneysel yapıyı iki farklı elemana ayırarak, sadece deneysel yapının değiştirilmesiyle farklı simülasyon deneyleri oluşturulabilir.

Bu yapı içinde, üç farklı modelleme türüne bağlı eleman modelleri, tek sistem modeli halinde birleştirilebilir. Kesikli değişen sistemler için modelin tanımlanmasında prosese ya da olaya yönelik modelleme kullanılmaktadır.

Sistem modeli ve deneysel yapı verildiğinde, SIMAN simülasyon programı simülasyon boyunca oluşan model durum birimlerini kaydeden bir çıktı dosyası oluşturur. Çıktı dosyasındaki veriler, histogramların ve tabloların oluşturulması gibi farklı veri analizleri için kullanılabilir.

SIMAN dilinde, simülasyon programının geliştirilmesi ve çalıştırılmasından sonra veri analizi ve görüntüleme fonksiyonu gerçekleştirilir.

SIMAN modelleme yapısı ile modelleme iki türde olmaktadır. Bunlar:

(1) Prosese Yönelik Modelleme
(2) Olaya Yönelik Modelleme

Prosese yönelik modellemede model, fonksiyonel operasyonları veya blok diyagram şeklinde sistemleri tanımlayarak kurulur. Blok diyagramı, gecikme zamanı ve kuyruklar gibi spesifik proses fonksiyonlarını temsil eden bir blok sırasıdır.

Kesikli değişen sistemler için ikinci bir modelleme ise, olaya yönelik modellemedir. Bu modelleme türünde, blok diyagram elemanlarının sayısı arttırılmaktadır. Olay elemanları, her olay zamanında ortaya çıkan gezer birimleri tanımlamak için matematiksel-lojik deyimleri içeren FORTRAN dilinde yazılmış alt programlardan oluşmaktadır. Buradaki alt programlar, SIMAN alt program kütüphanesinde bulunmaktadır.

SIMAN Yazılımının Yapısı : Şekil 4.11'de gösterildiği gibi, bir SIMAN simülasyonu üç farklı faaliyete ayrılır /28/. Bunlar, sistem modelinin geliştirilmesi, deneysel yapının oluşturulması ve veri analizidir. Bu üç faaliyet içinde, SIMAN yazılımı dört veri dosyasını (model dosyası, deney dosyası, program dosyası, çıktı dosyası) kullanan beş işlemciden oluşmaktadır. Bu işlemcileri şöyle sıralayabiliriz:

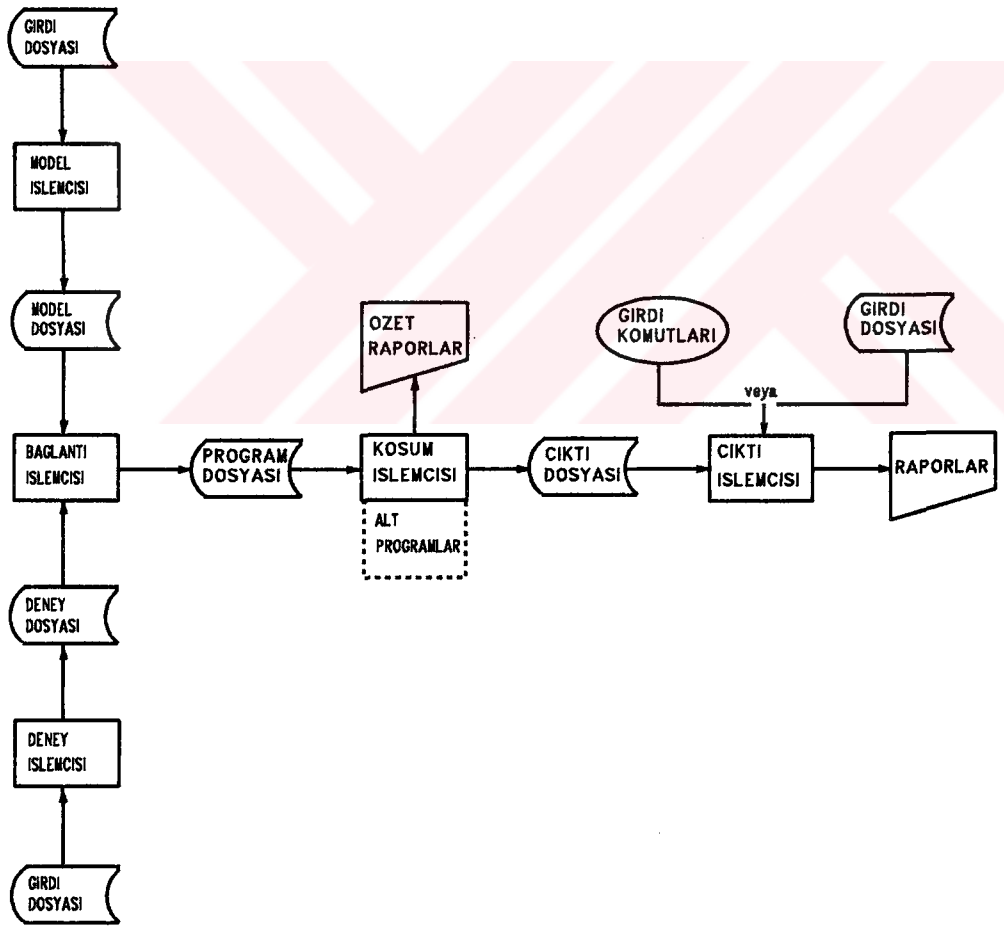
1. Model işlemcisi, blok diyagram modeli oluşturmak için kullanılmaktadır. Oluşturulan veri dosyası, model dosyası adını almaktadır.

2. Deney işlemcisi, sistem modelindeki deneysel yapıyı tanımlamak için kullanılmaktadır. Oluşturulan veri dosyası, deney dosyası adını almaktadır.

3. Bağlantı işlemcisi, program dosyasını oluşturmak üzere model dosyasını ve deney dosyasını birbirine bağlar.

4. Program dosyası, simülasyonu çalıştıran ve sonuçları çıktı dosyasına yazan koşum işlemcisi için bir model girdidir. Eğer sistem modeli, bir olay veya sürekli model içeriyorsa, simülasyona başlamadan önce, FORTRAN alt programlar koşum işlemcisi ile birleştirilmelidir.

5. Çıktı işlemcisi, çıktı dosyasında bulunan verileri analiz etmek etmek, düzenlemek ve göstermek için kullanılmaktadır.



Şekil 4.11. SIMAN Yazılımının Genel Yapısı

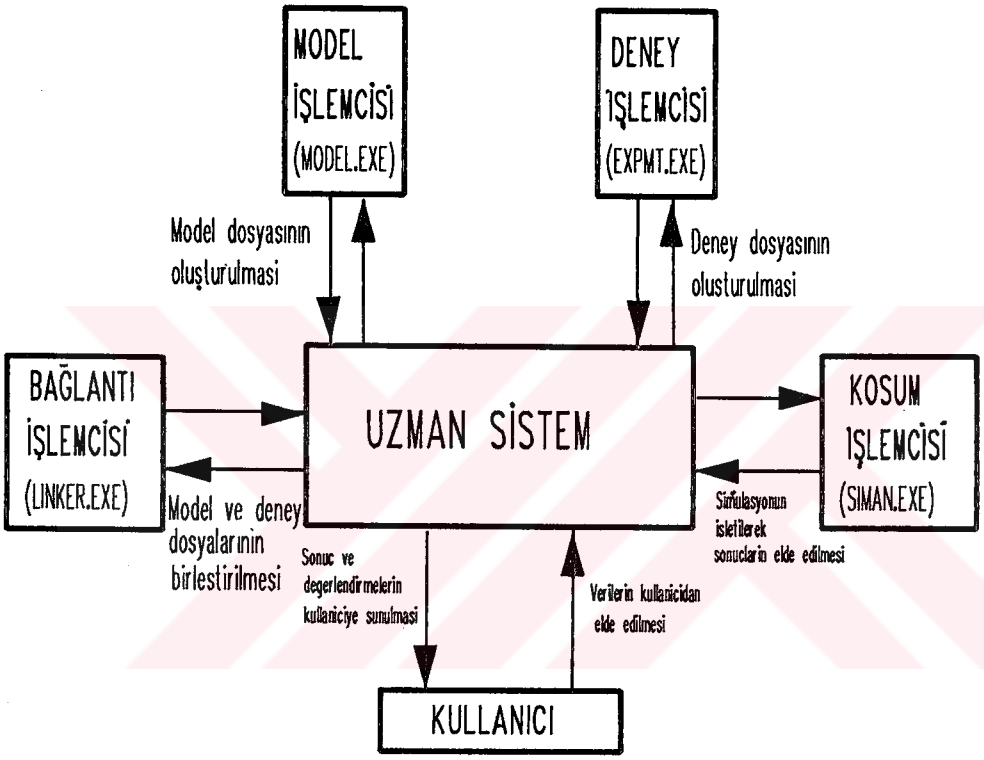
SIMAN yazılımı içindeki bu beş bağımsız işlemci, simülasyonun farklı fonksiyonel faaliyetlerini birbirinden ayırarak modelleme yapısını basitleştirmektedir. Beş işlemciden sadece bir tanesi bir kez çalıştırıldığından ve bilgisayar hafızasından tasarruf edildiğinden dolayı bu şekildeki bir ayırım oldukça pratik olmaktadır. Ayrıca, daha sonraki kullanımlar için, model dosyalarının, deney dosyalarının, program dosyalarının ve çıktı dosyalarının disk üzerine yazılması ve saklanması da mümkündür.

4.4. KUYRUK MODELLERİNİN SİMÜLASYONU İÇİN BİR UZMAN SİSTEMİN GELİŞTİRİLMESİ

Bir simülasyon çevriminin çeşitli kısımlarının mevcut yapay zekâ ve uzman sistem tekniklerinin kullanılarak gerçekleştirilebileceği daha önce belirtilmişti. Aynı teknikler kuyruk modellerinin smülasyonuna uygulanarak, kuyruk modelleri simülasyonuna tüm çevrim boyunca otomatik destek sağlanır.

Burada gerçekleştirilen sistem yapısı, bir simülasyon paket programı ile kullanıcı arasına bir uzman sistemin yerleştirilmesidir. Bu yapı, 3.bölümde uzman sistemler ile simülasyonun birleştirilmesi konusunda Şekil 3.16'da şematik olarak gösterilerek açıklanmıştır. Bu tip bir yaklaşımın kullanılması ile simülasyonu yapılacak alan hakkında bilgisi olan fakat simülasyon paket programı hakkında bilgisi olmayan bir kullanıcının bile bir simülasyon çalışması yapması sağlanabilir. Bu tezde geliştirilen uzman sistemin, kullanıcı ve simülasyon programının otomatik olarak oluşturulacağı dil olarak seçilen SIMAN paket programı ile etkileşimi Şekil 4.12'de görülmektedir. Şekilden görüldüğü gibi uzman sistem verileri elde etmek ve daha sonra çıktı raporları ve değerlendirmelerini iletmek üzere, bir kullanıcı ile iletişim kurar. Kullanıcıdan verileri elde eden uzman sistem daha sonra, SIMAN paket programına ait işlemcilerle iletişim kurar. Uzman sistem, model işlemcisi (MODEL.EXE) ile iletişim kurarak model dosyasını, deney işlemcisi (EXPMT.EXE) ile iletişim kurarak deney dosyasını oluşturduktan sonra bağlantı işlemcisi (LINKER.EXE) ile iletişim

kurarak model ve deney dosyalarını birleştirip program dosyasının oluşturulmasını sağlamaktadır. Uzman sistem sonunda koşum işlemcisi (SIMAN.EXE) ile iletişim daha önce oluşturulmuş olan program dosyasını çalıştırarak yani simülasyonu işleterek sonuçları elde eder ve bir dosyaya aktarılmasını sağlar.



Şekil 4.12. Tasarlanan Uzman Sistemin Bir Kullanıcı ve SIMAN Paket Programı ile Etkileşimi

Üçüncü bölümde ayrıntılı olarak belirtildiği gibi simülasyon amaçlı bir uzman sistemde öncelikle problemin ayrıntılı bir tanımının elde edilmesi gerekmektedir. Bunun için de, üç yaklaşım kullanılabileceği gene aynı bölümde belirtilip açıklanmıştı. Bu üç yaklaşım doğal dil arabirimi, grafik arabirimi ve diyalog arabirimidir. Burada bu amaçla kullanılan yaklaşım bir diyalog arabirimidir. Bir diyalog arabirimi ile kullanıcıdan problemin ayrıntılı bir tanımı ve

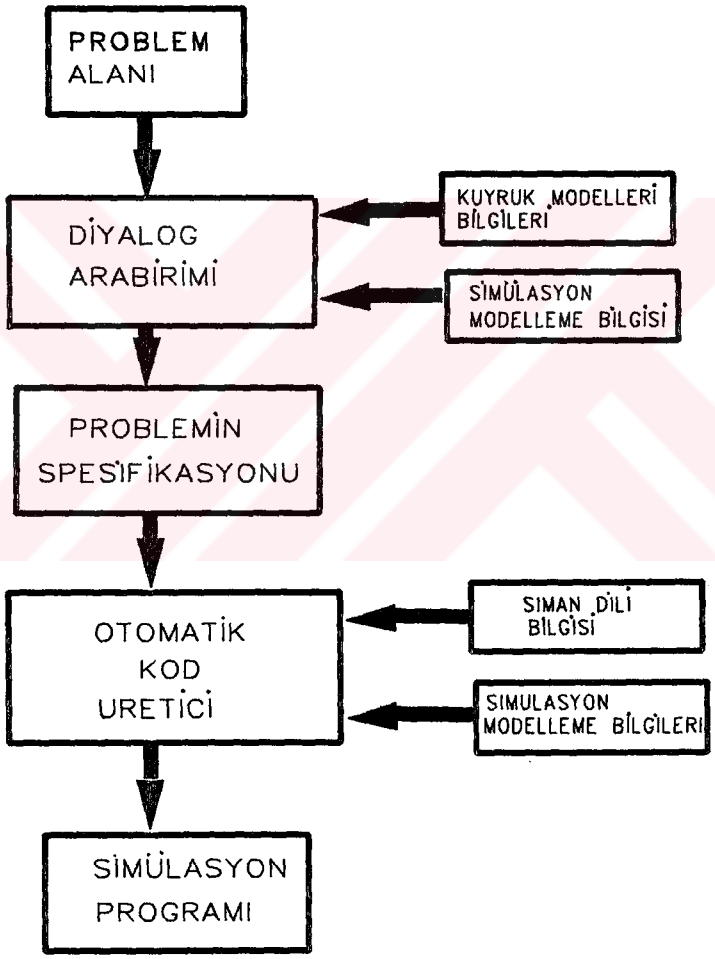
gerekli veriler tam ve eksiksiz bir şekilde elde edilerek, problemin spesifikasyonu oluşturulur. Problemin spesifikasyonu elde edilirken, kuyruk sistemleri bilgisi ve simülasyon modelleme bilgisi diyalog arabirimine destek verir. Daha sonra ise, bir kullanıcıdan elde edilen problem spesifikasyonu, simülasyon programının otomatik olarak oluşturulmasında bir veri tabanı olarak kullanılır.

Üçüncü bölümde açıklandığı gibi simülasyon programının otomatik olarak oluşturulmasında iki yaklaşım kullanılabilmektedir. Bu yaklaşımlardan biri, elde edilen problem spesifikasyonunun, bir otomatik kod üretici (yani simülasyon programını otomatik olarak yazan bir çıkarım mekanizması) tarafından yorumlanarak program haline dönüştürülmesidir. Diğer yaklaşım ise, daha önceden oluşturulmuş bir altprogramlar (makrolar) katoloğunda, problemin spesifikasyonuna göre gerekli makroların alınarak bunların bir kombinasyonunun oluşturulması ile simülasyon programının elde edilmesidir. Otomatik simülasyon yazılımını desteklemek için iki tür bilgi kullanılmaktadır. Bu bilgiler simülasyon modelleme bilgisi ve hedef simülasyon dili bilgisidir. Hedef simülasyon dili, simülasyon programının oluşturulması amaçlanan simülasyon dilidir. Bu tezde, otomatik olarak simülasyon programının oluşturulacağı simülasyon dili olarak SIMAN kullanılmaktadır. Burada simülasyon programının oluşturulmasında, bir otomatik kod üretici yani problem spesifikasyonunu yorumlayarak simülasyon programı haline dönüştüren bir çıkarım mekanizması kullanılmaktadır.

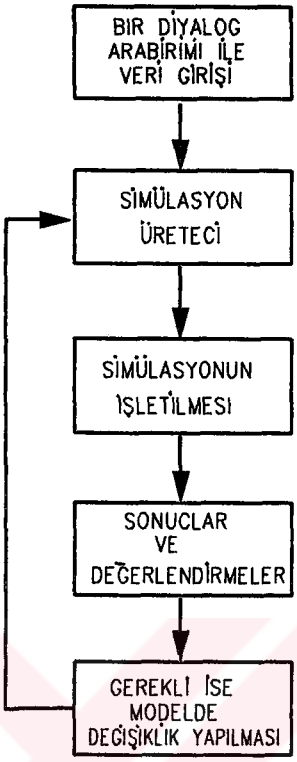
Bir diyalog arabiriminin tasarlanması ve bir de otomatik bir kod üretecinin geliştirilmesi ile bir simülasyon üreteci elde edilir. Kuyruk modellerinin simülasyonu için geliştirilen simülasyon üretecinin yapısı Şekil 4.13'de gösterilmektedir.

Daha sonra, bir simülasyon üreteci ile oluşturulan simülasyon programı, otomatik olarak çalıştırılarak simülasyon çıktı raporları elde edilmektedir. Ortaya çırkarılan bu sonuçlar da daha önceden tanımlanan kriterlere göre değerlendirilerek yorumlanır. Değerlendirmelerde kullanılan kriterler servis birimlerinin kullanım oranları, servis birimlerinde oluşan kuyruk uzunlukları, sistem süreleri ve bunlar gibi

kriterlerden oluşmaktadır. Örneğin daha önceden belirlenmiş bir kullanım oranına göre her servis birimi değerlendirilerek sonuçlar kullanıcıya bildirilir ve kullanıcıdan gerekirse modelde değişiklik yapmasına izin verilir. Meydana getirilen bu sistem yapısı Şekil 4.14'de şematik olarak gösterilmektedir. Bu durumda geliştirilen bu yapı kuyruk modellerinin simülasyonu yönelik simülasyon amaçlı bir uzman sistem olarak ortaya çıkmaktadır.



Şekil 4.13. Kuyruk Modelleri için Geliştirilen Simülasyon Üretecinin Yapısı



Şekil 4.14. Geliştirilen Simülasyon Amaçlı Uzman Sistemin Genel Yapısı

4.4.1. Bir Diyalog Arabiriminin Tasarlanması

Bir modelin kurulması için gerekli olan problem spesifikasyonunun yani problemin eksiksiz ve ayrıntılı tanımından oluşan verilerin elde edilmesi amacıyla, menülerden ve sorulardan oluşan bir diyaloğun geliştirilmesi gerekmektedir. Geliştirilen bu diyalog, simülasyon modeli kurulacak olan kuyruk sistemlerinin tüm karakteristiklerini kullanıcıdan elde edebilecek nitelikte olmalıdır.

Kuyruk modellerinin simülasyonu için, gereken verilerin elde edilmesi amacıyla tasarlanan diyalog şu şekildedir :

- Simülasyonu yapılacak olan sistemin adı nedir?

- Kuyruk sistemine kaç farklı türde müşteri gelmektedir?

  (Burada girilen değer birden büyükse, kullanıcıdan her müşteri tipinin kesikli dağılıma uygun olarak ortaya çıkma olasılığının girilmesi istenir)

- Kuyruk modelindeki safha sayısı kaçtır?

- Her safhadaki servisçilerin sayısı

- Kuyruk modelindeki gelişler hangi dağılıma uymaktadır?

[1] POISSON DAĞILIMI
[2] ÜSSEL DAĞILIM
[3] DÜZGÜN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABİT DEĞER

Dağılım tipini giriniz :

Eğer girilen değer 1 ise,

- POISSON dağılımına ait ortalama gelişler arası süre nedir?

Eğer girilen değer 2 ise,

- ÜSSEL dağılıma ait ortalama gelişler arası süre nedir?

Eğer girilen değer 3 ise,

- DÜZGÜN dağılımın minimum değeri nedir?

- DÜZGÜN dağılımın maksimum değeri nedir?

Eğer girilen değer 4 ise,

- NORMAL dağılımın ortalama değeri nedir?

- NORMAL dağılımın standart sapması nedir?

Eğer girilen değer 5 ise,

- SABİT değer nedir?

- Her müşteri tipinin uğradığı safhaların belirlenmesi

  (Eğer müşteri tipi birden büyükse, bu bilgiler istenir)

- Her servis birimi için servis sürelerinin dağılım özelliği ve belirlenen dağılım için gerekli olan parametrelerin girilmesi

  (Geliş dağılımını belirlemek için kullanılan menu ve sorular burada da kullanılır)

- Simülasyonun çalışma süresini girin

Bazı kuyruk modelleri için geliştirilen bu spesifik diyalog ile, kullanıcıdan, simülasyon için gerekli olan verilerin otomatik olarak elde edilerek problemin bir spesifikasyonunun oluşturulması sağlanır.

4.4.2. Simülasyon Amaçlı Uzman Sistemin Oluşturduğu Simülasyon Modellerindeki Varsayımlar

Geliştirilen sistem tarafından, üretilen simülasyon programları (modelleri) bazı varsayımları içermektedir. Kuyruk sistemlerinin simülasyon modellerinde kabul edilen varsayımlar aşağıda verilmektedir :

- Gelişler arası süre poisson, üssel, normal ve düzgün dağılımlardan herhangi birisi tarafından tanımlanabilmekte veya sabit bir değer olmaktadır.

- Her servis tesisi önündeki kuyruklar için sonsuz yeterlilikte alan vardır. Yani kuyruk uzunlukları sınırsızdır.

- Tüm servislerde servis disiplini olarak ilk gelen ilk servis görürü (FIFO) kuralı uygulanır.

- Servis süreleri poisson, üssel, normal, ve düzgün dağılımlardan herhengi birisi tarafından tanımlanabilmekte veya sabit bir değer olmaktadır.

- Paralel servislerde yani aynı işi yapan birden fazla servisçi olduğunda bir müşteri için kuyruk seçiminde uygulanan kural, bu servis safhasına gelen müşterinin, kuyruk uzunluğu en kısa

olan servis biriminin kuyruğuna girmesi, eğer servis birimlerindeki kuyruk uzunlukları birbirine eşitse o zaman müşterinin bu servis birimlerinden belirlenmiş olan birinin kuyruğuna girmesidir.

- Çok safhalı kuyruk modellerinde birden fazla müşteri tipi tanımlanabilmekte ve birden fazla müşteri tipi tanımlandığı zaman sisteme gelen bir müşterinin hangi türde olduğu ise kesikli dağılıma uygun olarak tanımlanabilmektedir.

Tasarlanan uzman sistem bu varsayımları göz önüne alarak simülayon programlarını (modellerini) oluşturmaktadır.

4.4.3. Geliştirilen Bilgisayar Programının Mantıksal Yapısı

Bu tezde geliştirilen bilgisayar programı (yani, simülasyon amaçlı bir uzman sistem), kuyruk modellerinin simülasyonu için otomatik olarak SIMAN simülasyon dilinde programlar oluşturmakta, daha sonra simülasyonu işleterek sonuçları değerlendirmektedir. Geliştirilen kuyruk modelleri simülasyon uzman sistemi **Turbo PASCAL 6.0** ile yazılmış olup bu program **SIMAN 3.5** dilinde simülasyon programları üretmektedir. Geliştirilen programın mantıksal yapısını gösteren akış diyagramı Şekil 4.15'te gösterilmektedir.

İlk olarak çözümlenecek olan kuyruk modeli ile ilgili tüm verilerin eksiksiz bir şekilde kullanıcıdan elde edilmesi gerekmektedir. Bu veriler gelişler ile ilgili bilgiler, servisler ile ilgili bilgiler ve kuyruk sisteminin yapısı ile ilgili bilgileri içermektedir. Bu veriler kullanıcıdan, bir diyalog arabirimi ile elde edilir. Bu program için geliştirilen diyalog arabiriminin yapısı daha önceki konuda anlatılmıştır.

Kuyruk modeli ile ilgili tüm veriler elde edilerek problemin spesifikasyonu oluşturulduktan sonra, bir otomatik kod üretici (yani

bir çıkarım mekanizması) ile kuyruk sisteminin SIMAN dilinde bir model programı oluşturulur.

SIMAN ile modelleme konusunda açıklandığı gibi, SIMAN ile simülasyon sisteminde iki program dosyasının oluşturulması gerekmektedir. Bu iki program dosyalarından birisi model program dosyası diğeri de deney programı dosyasıdır.

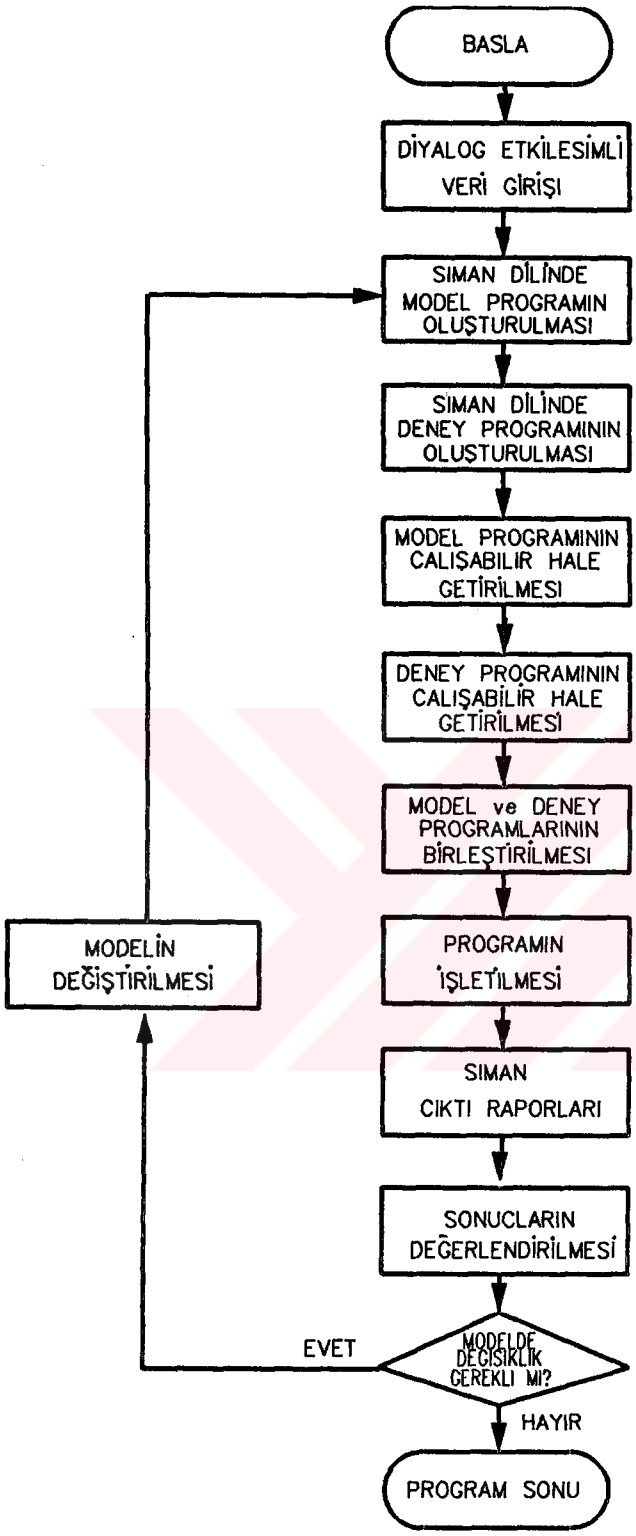
Model program oluşturulduktan sonra, kullanıcıdan elde edilen veriler bir deney programı oluşturma mekanizması tarafından yorumlanarak bu kez bir deney programı dosyası oluşturulur.

Uzman sistem daha sonra, kendisi tarafından oluşturulmuş olan model program dosyasının, SIMAN model işlemcisi tarafından derlenmesini gerçekleştirerek, model programının deney programı ile birleştirilebilir hale getirilmesinin sağlandığı yeni bir model dosyası oluşturur. Bu işlem gerçekleştirildikten sonra uzman sistem, bu kez benzer işlemleri deney programı dosyasına uygulayarak, gene uzman sistemin kendisi tarafından oluşturulmuş olan deney program dosyasının, SIMAN deney işlemcisi tarafından derlenmesini sağlayarak deney yapısının tanımlanmasını temin eder ve model program ile birleştirilebilir hale getirilen yeni bir deney dosyası oluşturur.

Uzman sistem daha sonra, SIMAN ile iletişim kurarak oluşturulmasını sağladığı yeni model ve deney dosyalarının, SIMAN bağlantı işlemcisi ile birleştirilmesini gerçekleştir. Bu şekilde de simülasyonu çalıştıracak olan esas program dosyası elde edilir.

SIMAN bağlantı işlemcisini kullanarak esas program dosyasının oluşturulmasını sağlayan uzman sistem daha sonra, SIMAN koşum işlemcisi ile iletişim kurarak, elde edilen programın işletilerek (RUN edilerek), simülasyonun çalıştırılmasını ve sonuçların bir dosyaya aktarılmasını gerçekleştirir.

Simülasyon sonucunda elde edilen SIMAN çıktı raporları, her bir servisçi için kullanım oranları ve her bir servis biriminde oluşan



Şekil 4.15. Geliştirilen Bilgisayar Programının (Simülasyon Amaçlı Uzman Sistemin) Mantıksal Yapısını Gösteren Akış Diyagramı

kuyruk uzunlukları ve bunlara ait istatistiksel bilgilerden oluşur. Buradaki istatistiksel bilgiler kullanım oranları ve kuyruk uzunluklarına ait ortalamalar, simülasyon süresi içinde gözlenmiş olan maksimum ve minimum değerler ile ortalamalara ait standart sapmalardan oluşur. Ayrıca çıktı raporlarında sistem süresi ve sistem süresine ait istatistiksel bilgiler de elde edilir. Birden fazla müşteri tipinin olduğu kuyruk sistemi için elde edilen çıktıda, her bir müşteri tipi için ayrı bir sistem süresi ve bu sistem sürelerine ait istatistiksel bilgiler elde edilir. Sistem süresine ait istatistiksel bilgiler, ortalama sistem süresi, simülasyon süresi içinde gözlenmiş olan maksimum sistem süresi ile minimum sistem süresi ve sistem süresinin standart sapmasıdır. Burada bir de simülasyon süresi içinde sistemde hizmet görerek sistemden ayrılan müşterilerin sayısı da bildirilmektedir.

Elde edilen bu çıktı raporu, uzman sistem tarafından kullanım oranlarına göre değerlendirilerek yorumlanır. Daha önceden belirlenmiş olan kullanım oranı göz önüne alınarak, her servis biriminin kabul edilebilir kullanım oranında olup olmadığı bir rapor halinde kulanıcıya sunulur. Ayrıca kullanım oranları histogramı da uzman sistem tarafından oluşturularak kullanıcıya sunulur.

Kullanıcıya sunulan bu bilgilerden sonra kullanıcıdan, modelde değişiklik yapmak isteyip istemediği öğrenilir. Modelde yapılabilecek değişiklikler, herhangi bir safhaya paralel bir servis biriminin eklenmesi veya herhangi bir safhadan paralel bir servis biriminin çıkartılmasını içermektedir.

Model değiştirildikten sonra ortaya çıkan bu yeni kuyruk yapısı için sistem Şekil 4.15'te görüldüğü gibi SIMAN dilinde simülasyon model programının oluşturulması safhasından çalıştırılarak işlemler tekrarlanır ve yeni elde edilen sonuçlar ve değerlendirmeler kullanıcıya sunulur.

Ortaya çıkan sonuçlar, kullanıcı için tatmin edici ise program sonuçlandırılarak, kabul edilebilir bir kuyruk yapısı oluşturulur veya

var olan bir kuyruk sisteminin performansı elde edilirek kullanıcının alacağı kararda ona destek sağlanır.

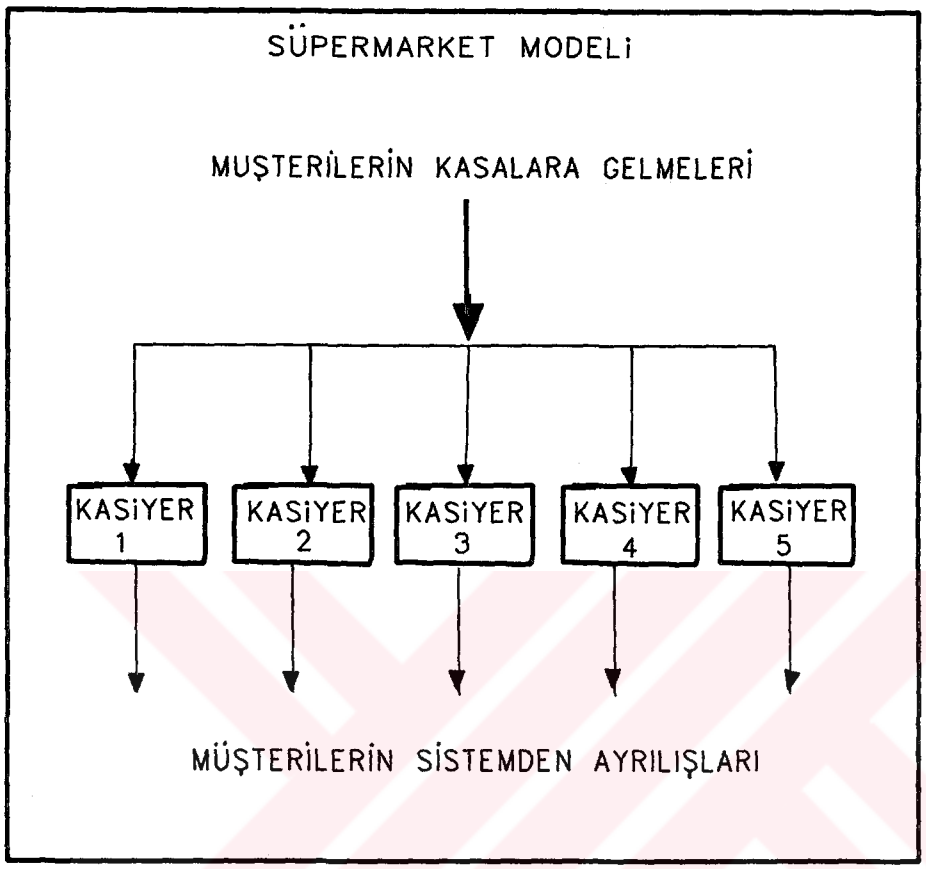
4.4.4. Geliştirilen Uzman Sistemin Yapay Bir Örnek Üzerinde Çalıştırılması

Yapay bir örnek olarak 5 kasiyere sahip olan bir süpermarket modeli ele alınmış ve bu süpermarket modelinde gelişlerin poisson ve servis sürelerinin de üssel dağılıma uyduğu kabul edilmiştir. Burada amaç, kasiyerlerin her birinin kullanım oranları, her bir servis tesisi yani yazarkasaların önünde oluşan kuyrukların uzunlukları ve müşteriler için ortalama sistem süresinin ortaya çıkarılması yani bu kuyruk sisteminin performans ölçülerinin elde edilmesidir. Kuyruk modelinin performans ölçülerinin elde edilmesi içinde bir simülasyon etüdünün yapılması istenmektedir.

Bu örnekte ele alınan süpermarket modelinin şematik bir görünümü, Şekil 4.16'da verilmektedir. Bu süpermarket modelinde, ortalama gelişler arası sürenin 1.2 dakika olduğu ve ortalama servis sürelerinin de 4.5 dakika olduğu ve bu ortalamanın bütün kasiyerler için geçerli olduğu kabul edilmiştir.

Ayrıca bu kuyruk modelinde yapılan varsayımlarda, kasalar önünde oluşan kuyruklar için sonsuz uzunlukta alan olduğu yani kuyruk uzunluklarının sınırısız olduğu ve kasaların önüne gelen bir müşterinin, kasaların önünde oluşan kuyruk uzunluklarına bakarak, en kısa kuyruk uzunluğuna sahip olan kasanın kuyruğuna girdiği ve eğer kasaların önünde oluşan kuyruk uzunlukları birbirlerine eşitse o zaman müşterinin Şekil 4.16'da görülen 1 numaralı yazarkasanın kuyruğuna girdiği kabul edilmektedir.

Süpermarket modelinin performans ölçülerinin elde edilmesi için yapılacak olan simülasyon çalışması, bu tezde geliştirilen kuyruk modelleri simülasyon uzman sistemine yaptırılarak istenen tüm performans ölçüleri, uzman sistem tarafından elde edilerek kullanıcıya sunulmuştur.



Şekil 4.16. Süpermarket Modeli Örneğinin Şematik Olarak Görünümü

Burada uzman sistem, öncelikle bir kullanıcıdan, bir diyalog arabirimiyle problem ile ilgili tüm verileri elde etmiş ve daha sonra süpermarketin simülasyonu gerçekleştirmiştir. Uzman sistem, simülasyon için SIMAN 3.5 simülasyon dilini kullanmaktadır. Kuyruk modelleri simülasyon uzman sistemi, sorunun SIMAN ile çözümü için gerekli olan tüm programları otomatik olarak oluşturmakta ve simülasyonu işleterek sonuçları ve değerlendirmeleri kullanıcıya sunmaktadır.

Uzman sistem tarafından oluşturulan MODEL program

```
BEGIN;
          CREATE:NP(6,1):MARK(1);
          ASSIGN:A(1)=1;
SAFHA11   PICKQ,SNQ,SRV11:SRV11:SRV12:SRV13:SRV14:SRV15;
SRV11     QUEUE,1;
          SEIZE:SERVIS1;
          DELAY:EX(1,1);
          RELEASE:SERVIS1:NEXT(RAPOR);
SRV12     QUEUE,2;
          SEIZE:SERVIS2;
          DELAY:EX(2,1);
          RELEASE:SERVIS2:NEXT(RAPOR);
SRV13     QUEUE,3;
          SEIZE:SERVIS3;
          DELAY:EX(3,1);
          RELEASE:SERVIS3:NEXT(RAPOR);
SRV14     QUEUE,4;
          SEIZE:SERVIS4;
          DELAY:EX(4,1);
          RELEASE:SERVIS4:NEXT(RAPOR);
SRV15     QUEUE,5;
          SEIZE:SERVIS5;
          DELAY:EX(5,1);
          RELEASE:SERVIS5:NEXT(RAPOR);
RAPOR     TALLY:A(1),INT(1):DISPOSE;
END;
```

Uzman sistem tarafından oluşturulan DENEY programı

```
BEGIN;
PROJECT,SUPERMARKET,KUMSU;
DISCRETE,1500,1,5;
PARAMETERS:
          1, 4.500:
          2, 4.500:
          3, 4.500:
          4, 4.500:
          5, 4.500:
          6, 1.200;
RESOURCE:
          1,SERVIS1,1:
          2,SERVIS2,1:
          3,SERVIS3,1:
          4,SERVIS4,1:
          5,SERVIS5,1;
TALLIES:
          1,TIP1 SIS_SURESI;
DSTAT:
          1,NR(1),SERV1 KULLANM:
```

```
            2,NR(2),SERV2 KULLANM:
            3,NR(3),SERV3 KULLANM:
            4,NR(4),SERV4 KULLANM:
            5,NR(5),SERV5 KULLANM:
            6,NQ(1),SERV1 KUYRUGU:
            7,NQ(2),SERV2 KUYRUGU:
            8,NQ(3),SERV3 KUYRUGU:
            9,NQ(4),SERV4 KUYRUGU:
            10,NQ(5),SERV5 KUYRUGU;
REPLICATE,1,0,480;
END;
```

Simülasyonun sonuçları

SIMAN Summary Report

Run Number 1 of 1

Project: SUPERMARKET
Analyst: KUMSU
Date : 1/ 1/2000

Run ended at time .4800E+03

Tally Variables

Number	Identifier	Average	Standard Deviation	Minimum Value	Maximum Value	Number of Obs.
1	TIP1 SIS_SURESI	241.00040	143.63790	1.23488	478.94060	424

Discrete Change Variables

Number	Identifier	Average	Standard Deviation	Minimum Value	Maximum Value	Time Period
1	SERV1 KULLANM	.85435	.35275	.00000	1.00000	480.00
2	SERV2 KULLANM	.83118	.37460	.00000	1.00000	480.00
3	SERV3 KULLANM	.76032	.42689	.00000	1.00000	480.00
4	SERV4 KULLANM	.63362	.48181	.00000	1.00000	480.00
5	SERV5 KULLANM	.56011	.49637	.00000	1.00000	480.00
6	SERV1 KUYRUGU	.50807	.76344	.00000	3.00000	480.00
7	SERV2 KUYRUGU	.41851	.67548	.00000	3.00000	480.00
8	SERV3 KUYRUGU	.32661	.57732	.00000	3.00000	480.00
9	SERV4 KUYRUGU	.25617	.56625	.00000	3.00000	480.00
10	SERV5 KUYRUGU	.14728	.39245	.00000	2.00000	480.00

Run Time : 1 Second(s)
Stop - Program terminated.

```
KABUL EDİLEBİLİR KULLANIM ORANI      : %75

1. safhadaki 1. servisçinin kullanım oranı yeterlidir.
1. safhadaki 2. servisçinin kullanım oranı yeterlidir.
1. safhadaki 3. servisçinin kullanım oranı yeterlidir.
1. safhadaki 4. servisçinin kullanım oranı YETERLİ DEĞİLDİR.
1. safhadaki 5. servisçinin kullanım oranı YETERLİ DEĞİLDİR.
```

5. GELİŞTİRİLEN UZMAN SİSTEM İLE YAPILAN BİR UYGULAMA

Geliştirilen uzman sistem bazı kuyruk modellerinin simülasyon modellerini yani programlarını otomatik olarak oluşturmakta ve simülasyonu işleterek sonuçları ve değerlendirmeleri kullanıcıya sunmaktadır.

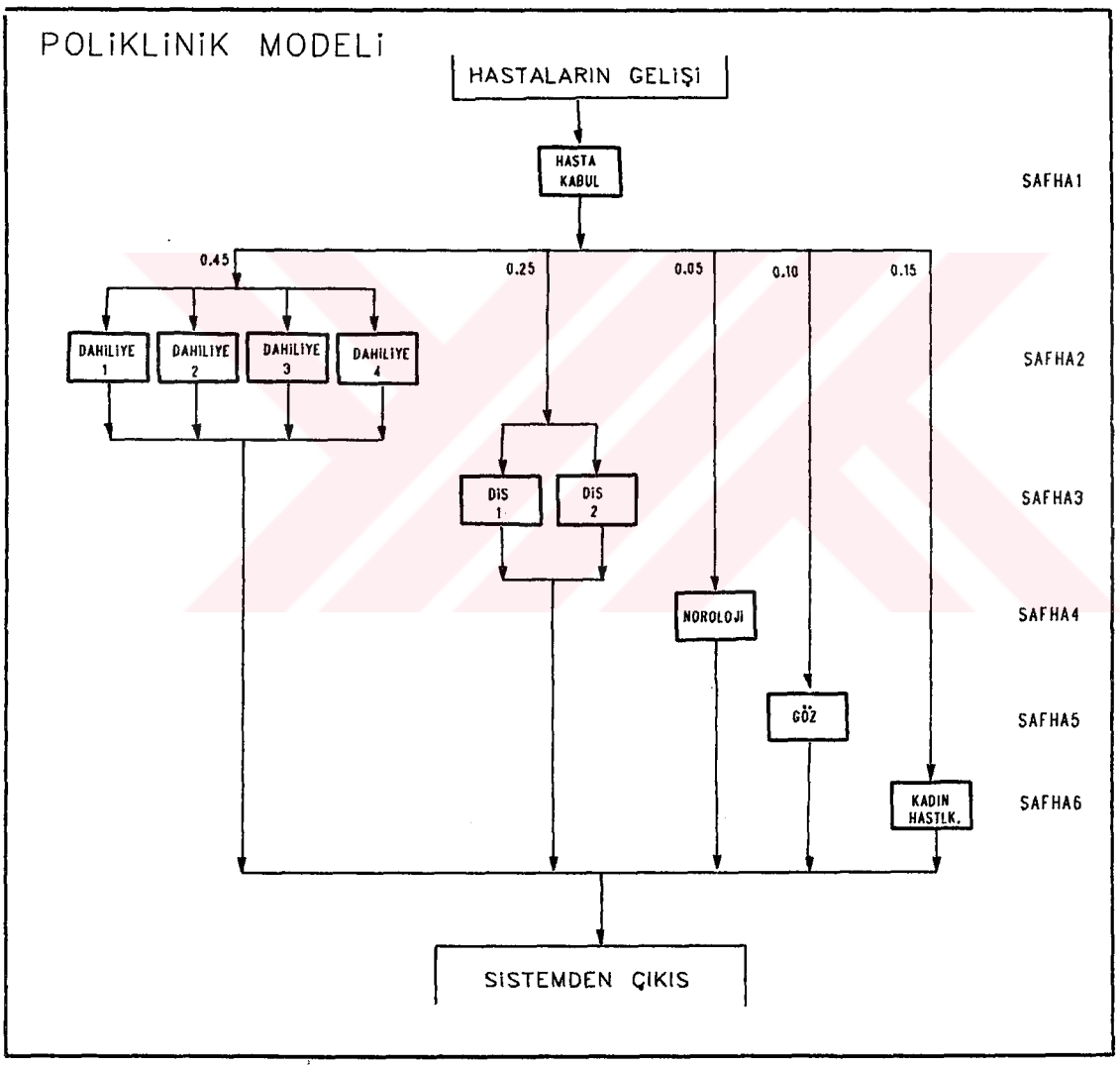
Tasarlanan uzman sistem, en basit yapıdaki tek servisçili ve tek safhalı kuyruk modeli, daha karmaşık olan tek serviçili ve çok safhalı kuyruk modeli, çok servisçili ve tek safhalı kuyruk modeli ve çok daha karmaşık yapıdaki çok servisçili ve cok safhalı kuyruk modeli olmak üzere birçok kuyruk modelini çözebilmektedir. Burada sözü edilen kuyruk modellerinin şematik bir görünümü daha önce verilmiştir (Şekil 4.5.).

Burada belirtilen kuyruk yapılarına ilaveten uzman sistem, çok safhalı kuyruk modellerinde, birden fazla müşteri grubu (çeşidi veya tipi) için de çözüm yapabilmekte ve böylece temel kuyruk modellerini belli bir ölçüde aşabilmektedir. Birden fazla müşteri tipi sözü ile anlatılmak istenen, çok safhalı bir kuyruk modeline gelen müşterilerin birbirlerinde farklı bir rota izleyerek sistemi terk etmesidir. Örneğin, üç safhalı bir kuyruk sistemi ve bu sisteme gelen iki farklı müşteri tipi ele alındığında, sisteme gelen müşterilerden birinci tipte olanlar, üç safhaya da uğrayarak sistemi terk etmekte ama buna karşın, ikinci tipteki müşteriler ise sadece üçüncü safhaya uğrayarak sistemi terk etmesi durumu, birden fazla müşteri tipi sözü ile anlatılmak istenendir. Böylece uzman sistemin, çok daha farklı kuyruk modellerinin simülasyonunu yapabilmesi sağlanabilmektedir.

POLİKLİNİK MODELİ

Burada, kuyruk modelleri simülasyon uzman sistemi gerçek bir sistem üzerinde çalıştırılmıştır. Gerçek bir sistem olarak bir poliklinik sistemi ele alınmıştır.

Ele alınan poliklinikte toplam 5 farklı hastalıkla ilgili bölümler ve birde hasta kabul bölümü bulunmaktadır. Poliklinikte yer alan bölümler, dahiliye, nöroloji, diş, göz ve kadın hastalıkları bölümleridir. Poliklinikte, dahiliyede 4, dişte 2, göz, nöroloji ve kadın hastalıklarında birer doktor olamak üzere toplam 9 doktor ve birde hasta kabulde bir görevli bulunmaktadır. Poliklinik modelinin şematik olarak görünümü Şekil 5.1'de verilmektedir.



Şekil 5.1. Poliklinik Modelinin Şematik Olarak Gösterimi

Polikliniğe gelen hastalar önce hasta kabul bölümüne uğramakta ve oradan da hastalıkları ile ilgili bölüme sevk edilmektedirler.

Bu poliklinik, bir çok safhalı ve çok servisçili kuyruk modelinin gerçek hayattaki bir görünümüdür. Poliklinik modelinde her bir bölüm ayrı birer safhadır. Çünkü her bölümde yapılan işler farklıdır. Dolayısıyla bu poliklinik modeli Şekil 5.1'de de görüldüğü gibi 6 safhadan oluşmaktadır. Ayrıca bu poliklinik modelinde birden fazla müşteri tipi olarak belirtilen durum da söz konusudur. Yani polikliniğe gelen herhangi bir hasta önce hasta kabule uğramakta ve oradan da kendi hastalığıyla ilgili bölüme gitmektedir. Bu poliklinikte 5 farklı bölüm olduğundan, polikliniğe 5 farklı hasta grubu gelmektedir.

Burada amaç, poliklinik modelinin simülasyonu yapılarak, bu polikliniktekı doktorların ortalama kullanım oranları, herbir servis biriminin önünde oluşan ortalama kuyruk uzunlukları ve her hasta grubu için ortalama sistem süresi ve işletim süresi boyunca her hasta grubuna ait gözlenen hasta sayılarının ortaya çıkarılmasıdır. Burada simülasyon geliştirilen uzman sisteme yaptırılımış ve SIMAN programları ile sonuç ve değerlendirmeler elde edilmiştir.

Bu poliklinik modeli, teorik verilerle çalıştırılmıştır. Poliklinik modelinde gelişlerin poisson dağılımına ve servis sürelerinin de üssel dağılıma uyduğu kabul edilmiştir. Her bir servis biriminin üssel dağılımına ait ortalama süreleri Tablo 5.1'de verilmektedir. Gelişler için poisson dağılımına ait ortalama gelişler arası sürenin değeri 1.5 dakika olarak alınmıştır. Ayrıca polikliniğe gelen hastaların, %45'nin dahiliye, %25'nin diş, %5'nin nöroloji, %10'nun göz ve %15'nin de kadın hastalıklarına sevk edildiği kabul edilmiştir.

Poliklinik modeli için bir kullanıcı tarafında bir diyalog arabirimi ile yapılan veri girişi EK'te verilmiştir.

Tablo 5.1. Ortalama Servis Süreleri

BÖLÜM	ORTALAMA SERVİS SÜRESİ (dak.)
HASTA KABUL	1.0
1. DAHİLİYE	12.5
2. DAHİLİYE	14.5
3. DAHİLİYE	12.5
4. DAHİLİYE	11.5
1. DİŞ	23.0
2. DİŞ	27.0
NÖROLOJİ	19.0
GÖZ	21.0
KADIN HASTALIKLARI	25.0

Uzman sistem tarafından oluşturulan MODEL program

```
BEGIN;
          CREATE:NP(11,1):MARK(1);
          ASSIGN:A(2)=DP(12,1);
KONTR11   BRANCH,1:
          IF,A(2).EQ.1,SAFHA11:
          IF,A(2).EQ.2,SAFHA11:
          IF,A(2).EQ.3,SAFHA11:
          IF,A(2).EQ.4,SAFHA11:
          IF,A(2).EQ.5,SAFHA11:
          ELSE,KONTR12;
SAFHA11   QUEUE,1;
          SEIZE:SERVIS1;
          DELAY:EX(1,1);
          RELEASE:SERVIS1:NEXT(KONTR12);
KONTR12   BRANCH,1:
          IF,A(2).EQ.1,SAFHA12:
          ELSE,KONTR13;
SAFHA12   PICKQ,SNQ,SRV12:SRV12:SRV13:SRV14:SRV15;
SRV12     QUEUE,2;
          SEIZE:SERVIS2;
          DELAY:EX(2,1);
          RELEASE:SERVIS2:NEXT(KONTR13);
SRV13     QUEUE,3;
          SEIZE:SERVIS3;
          DELAY:EX(3,1);
          RELEASE:SERVIS3:NEXT(KONTR13);
SRV14     QUEUE,4;
          SEIZE:SERVIS4;
          DELAY:EX(4,1);
          RELEASE:SERVIS4:NEXT(KONTR13);
SRV15     QUEUE,5;
          SEIZE:SERVIS5;
```

```
          DELAY:EX(5,1);
          RELEASE:SERVIS5:NEXT(KONTR13);
KONTR13   BRANCH,1:
          IF,A(2).EQ.2,SAFHA13:
          ELSE,KONTR14;
SAFHA13   PICKQ,SNQ,SRV16:SRV16:SRV17;
SRV16     QUEUE,6;
          SEIZE:SERVIS6;
          DELAY:EX(6,1);
          RELEASE:SERVIS6:NEXT(KONTR14);
SRV17     QUEUE,7;
          SEIZE:SERVIS7;
          DELAY:EX(7,1);
          RELEASE:SERVIS7:NEXT(KONTR14);
KONTR14   BRANCH,1:
          IF,A(2).EQ.3,SAFHA14:
          ELSE,KONTR15;
SAFHA14   QUEUE,8;
          SEIZE:SERVIS8;
          DELAY:EX(8,1);
          RELEASE:SERVIS8:NEXT(KONTR15);
KONTR15   BRANCH,1:
          IF,A(2).EQ.4,SAFHA15:
          ELSE,KONTR16;
SAFHA15   QUEUE,9;
          SEIZE:SERVIS9;
          DELAY:EX(9,1);
          RELEASE:SERVIS9:NEXT(KONTR16);
KONTR16   BRANCH,1:
          IF,A(2).EQ.5,SAFHA16:
          ELSE,RAPOR;
SAFHA16   QUEUE,10;
          SEIZE:SERVIS10;
          DELAY:EX(10,1);
          RELEASE:SERVIS10:NEXT(RAPOR);
RAPOR     TALLY:A(2),INT(1):DISPOSE;
END;
```

Uzman sistem tarafından oluşturulan DENEY programı

```
BEGIN;
PROJECT,POLIKLINIK,KUMSU;
DISCRETE,1500,5,10;
PARAMETERS:
          1, 1.000:
          2, 12.500:
          3, 14.500:
          4, 12.500:
          5, 11.500:
          6, 23.000:
          7, 27.000:
```

```
          8, 19.000:
          9, 21.000:
          10, 25.000:
          11, 1.500:
          12, 0.45,1, 0.70,2, 0.75,3, 0.85,4,
           1.00,5;
RESOURCE:
          1,SERVIS1,1:
          2,SERVIS2,1:
          3,SERVIS3,1:
          4,SERVIS4,1:
          5,SERVIS5,1:
          6,SERVIS6,1:
          7,SERVIS7,1:
          8,SERVIS8,1:
          9,SERVIS9,1:
          10,SERVIS10,1;
TALLIES:
          1,TIP1 SIS_SURESI:
          2,TIP2 SIS_SURESI:
          3,TIP3 SIS_SURESI:
          4,TIP4 SIS_SURESI:
          5,TIP5 SIS_SURESI;
DSTAT:
          1,NR(1),SERV1 KULLANM:
          2,NR(2),SERV2 KULLANM:
          3,NR(3),SERV3 KULLANM:
          4,NR(4),SERV4 KULLANM:
          5,NR(5),SERV5 KULLANM:
          6,NR(6),SERV6 KULLANM:
          7,NR(7),SERV7 KULLANM:
          8,NR(8),SERV8 KULLANM:
          9,NR(9),SERV9 KULLANM:
          10,NR(10),SERV10 KULLANM:
          11,NQ(1),SERV1 KUYRUGU:
          12,NQ(2),SERV2 KUYRUGU:
          13,NQ(3),SERV3 KUYRUGU:
          14,NQ(4),SERV4 KUYRUGU:
          15,NQ(5),SERV5 KUYRUGU:
          16,NQ(6),SERV6 KUYRUGU:
          17,NQ(7),SERV7 KUYRUGU:
          18,NQ(8),SERV8 KUYRUGU:
          19,NQ(9),SERV9 KUYRUGU:
          20,NQ(10),SERV10 KUYRUGU;
REPLICATE,1,0,480;
END;
```

Simülasyon sonuçları

SIMAN Summary Report

Run Number 1 of 1

Project: POLIKLINIK
Analyst: KUMSU
Date : 1/ 1/2000

Tally Variables

Number	Identifier	Average	Standard Deviation	Minimum Value	Maximum Value	Number of Obs.
1	TIP1 SIS_SURESI	22.53504	16.94810	1.36212	99.45319	144
2	TIP2 SIS_SURESI	138.72850	72.53139	4.97105	271.50340	45
3	TIP3 SIS_SURESI	24.98384	13.30340	1.98479	47.77222	14
4	TIP4 SIS_SURESI	84.73696	39.57657	12.89240	165.99040	29
5	TIP5 SIS_SURESI	79.44405	59.91656	7.34610	185.93430	18

Discrete Change Variables

Number	Identifier	Average	Standard Deviation	Minimum Value	Maximum Value	Time Period
1	SERV1 KULLANM	.69875	.45880	.00000	1.00000	480.00
2	SERV2 KULLANM	.92762	.25912	.00000	1.00000	480.00
3	SERV3 KULLANM	.89599	.30527	.00000	1.00000	480.00
4	SERV4 KULLANM	.84288	.36392	.00000	1.00000	480.00
5	SERV5 KULLANM	.79538	.40342	.00000	1.00000	480.00
6	SERV6 KULLANM	.98089	.13690	.00000	1.00000	480.00
7	SERV7 KULLANM	.96825	.17534	.00000	1.00000	480.00
8	SERV8 KULLANM	.46876	.49902	.00000	1.00000	480.00
9	SERV9 KULLANM	.95678	.20335	.00000	1.00000	480.00
10	SERV10 KULLANM	.89856	.30191	.00000	1.00000	480.00
11	SERV1 KUYRUGU	1.09520	1.77808	.00000	9.00000	480.00
12	SERV2 KUYRUGU	.89861	.80235	.00000	3.00000	480.00
13	SERV3 KUYRUGU	.76692	.77868	.00000	3.00000	480.00
14	SERV4 KUYRUGU	.48819	.67308	.00000	3.00000	480.00
15	SERV5 KUYRUGU	.44120	.67579	.00000	2.00000	480.00
16	SERV6 KUYRUGU	12.47777	6.55099	.00000	23.00000	480.00
17	SERV7 KUYRUGU	11.98207	6.60185	.00000	23.00000	480.00
18	SERV8 KUYRUGU	.19443	.45230	.00000	2.00000	480.00
19	SERV9 KUYRUGU	4.61369	1.92790	.00000	8.00000	480.00
20	SERV10 KUYRUGU	9.26633	9.23495	.00000	28.00000	480.00

Run Time : 3 Second(s)

```
KABUL EDİLEBİLİR KULLANIM ORANI      : %65

1. safhadaki 1. servisçinin kullanım oranı yeterlidir.
2. safhadaki 1. servisçinin kullanım oranı yeterlidir.
2. safhadaki 2. servisçinin kullanım oranı yeterlidir.
2. safhadaki 3. servisçinin kullanım oranı yeterlidir.
2. safhadaki 4. servisçinin kullanım oranı yeterlidir.
3. safhadaki 1. servsiçinin kullanım oranı yeterlidir.
3. safhadaki 2. servisçinin kullanım oranı yeterlidir.
4. safhadaki 1. servisçinin kullanım oranı YETERLİ DEĞİLDİR.
5. safhadaki 1. servisçinin kullanım oranı yeterlidir.
6. safhadaki 1. servisçinin kullanım oranı yeterlidir.
```

6. SONUÇ

Yapay zekâ ve özellikle uzman sistemlerinin simülasyon ile birleştirilmelerinin sonucu kazanılan bilgiler, çıkarılan teknikler ve elde edilen teorik ve pratik sonuçlar, simülasyonun değişik alanlardaki uygulamalarını ilginç bir hale getirmiştir.

Klasik olarak bir simülasyon çalışmasında, problemin spesifikasyonunu oluşturma ve daha sonra genel amaçlı bir bilgisayar dili veya özel amaçlı bir simülasyon dili kullanılarak, simülasyon modelini veya programını oluşturma işi bir simülasyoncu tarafında yerine getirilmektedir. Simülasyoncu daha sonra, modeli geçerli hale getirip, denemeler yaparak sonuçları elde etmekte ve elde edilen sonuçları bizzat kendisi analiz ederek yorumlamaktadır. Oysa bir simülasyon çevriminin bütün bu kısımları mevcut yapay zekâ ve uzman sistem teknikleri kullanılarak otomatik olarak yapılabilir. Böylece simülasyon etüdü yapılacak olan alan hakkında bilgi sahibi olan fakat bir simülasyon dilini bilmeyen bir kimsenin de, rahatlıkla bir simülasyon çalışması yapabilmesi sağlanabilmektedir. Uzman simülasyon sistemlerinin geliştirilmesindeki temel amaç, mühendisler, bilim adamları ve yöneticilerin, bir simülasyon bilgisine ve simülasyon programlama diline gerek kalmadan simülasyon çalışmalarını, kolay ve doğru bir şekilde yapabilmelerini sağlamaktır.

Uzman simülasyon sistemleri, klasik simülasyon sistemlerinden bazı yönlerden farklılık göstermektedirler ve birtakım avantajlara sahiptirler. Bu tez çalışması sonucunda ortaya çıkarılan ve açıklanan temel farklılıklar aşağıda maddeler halinde verilmektedir :

- Klasik bir simülasyon ile uzman simülasyon sistemi arasındaki temel fark, modelin kurulması ve işletilmesinde ortaya çıkmaktadır. Uzman bir simülasyon sistemi, bir arabirim ile kullanıcıdan, problemin spesifikasyonunu elde ederek bu

spesifikasyona göre problemin simülasyon modelini (programını) otomatik olarak kurar.

- Yapay zekâ tabanlı bir sistemde, veri tabanı, bilgi tabanı ve kontrol yapısı birbirlerinden ayrıdırlar yani bağımsızdırlar ve her biri diğerini etkilemeden kolayca değiştirilebilir.

- Yapay zekâ tabanlı bir uzman simülasyon sistemi ideal olarak, kendi bilgilerinden ve tecrübelerinden yeni şeyler öğrenebilmeli ve gerektiğinde kendisini değiştirebilmelidir. Genelde, yapay zekâ tabanlı uzman simülasyon sistemleri, teorik olarak mümkün olmasına rağmen bunu tam olarak başaramamaktadır.

- Uzman simülasyon sistemleri spesifik bir alanda olmalıdır. Genel amaçlı bir uzman simülasyon sistemi çoğunlukla geçerli değildir, zira bu durumda spesifik bilgi tabanından uzaklaşıldığından, kontrol mekanizmalarının çok karmaşık olmaları söz konusudur.

Bu tezde geliştirilen simülasyon amaçlı uzman sistem prototipi, bazı kuyruk modellerinin SIMAN dilinde simülasyon modellerini kurmakta, simülasyonu işletmekte ve elde edilen sonuçları ve sonuçların değerlendirilmelerini kullanıcıya sunmaktadır. Uzman sistem problem için gerekli olan veriyi kullanıcıdan bir diyalog arabirimi ile elde etmektedir. Uzman sistemin bilgi tabanı içerisinde ise, SIMAN komutları ve bu komutları problemin spesifikasyonuna göre uygun bir formda kodlayacak olan kontrol yapısı mevcuttur.

Kuyruk modelleri simülasyon uzman sistemi, simülasyon paket programı ile kullanıcı arasında olup, paket programı bilmeyen bir kullanıcının bile bir simülasyon çalışması yapmasına olanak verir.

Simülasyonun işletimi sonunda oluşturulan çıktılardan elde edilen bilgiler şunlardır :

- Ortalama sistem süresi ve bu süreye ait istatistikler,

- Her bir servis biriminin ortalama kullanım oranları ve bu oranlara ait istastistikler,

- Her bir servis birimi önünde oluşan ortalama kuyruk uzunluğu ve bunlara ait istatistikler,

- İşletim süresi boyunca sistemden hizmet görerek ayrılan müşterilerin sayısı,

- Elde edilen istatistikler, ortalamalara ait standart sapmalar ve işletim süresi boyunca gözlenmiş olan maksimum ve minimum değerleri de kapsamaktadır,

- Kullanıcıya ayrıca, her bir servis biriminin kabul edilebilir kullanım oranında olup olmadığı da sunulur.

Simülasyon ve uzman sistemlerin birbirleriyle olan ilişkilerinde göz önüne alınması gereken diğer bir hususda, sadece simülasyonun uzman sistem metoduna ihtiyaç duymaması aynı zamanda uzman sistemlerin de simülasyon tekniğine ihtiyaç duymasıdır. Yapay zekâ ve uzman sistemlerin mevcut kullanımaları incelendiğinde, bu kullanımların bazılarında karar için bir zaman boyutunun olmadığı görülür. Bunun nedeni ise yapay zekâ uzmanları, henüz zamanın nasıl kullanılacağını tam olarak bilmemektedirler. Simülasyonun gerçek gücünden birisi, zamana göre olayları ve etkilerini tahmin etme ve projelendirme özelliğidir. Uzman sistemin kullanımı, simülasyon metodunun birleştirilmesine ve kullanılmasına kadar sınırlı kalmaktadır. Bu yüzden, her iki alan da birbirlerinden yararlandığından, simülasyon ve uzman sistemlerin birleştirilmesi kaçınılmaz olmaktadır.

Daha ileri düzeyde geliştirilecek olan uzman simülasyon sisteminin amacı, kullanıcının ilgili sistemi, amaçları, hedefleri ve cevap formlarını doğal anlamlı bir dilde tanımlamasını sağlamak ve gerisini uzman sisteme bırakmak olmalıdır.

KAYNAKLAR

(1) FEIGENBAUM, E.A. ve P. McCORDUCT (1983), The Fifth Generation, Reading, Massachusetts, Addision.

(2) WATSON, H.J. ve J.H. BLACKSTONE JR. (1989), Computer Simulation, John Wiley & Sons.

(3) IGNIZIO, J.P. (1991), Introduction to Expert Systems, McGraw Hill, New York.

(4) BAYAZIT, E.N. ve M. KAVAKLI (1992), " Uzman Sistem Metotları-1 ", Bilgisayar Magazin, sayı 14, sayfa:60-66.

(5) PAYNE, E.C. VE R.C. McARTHUR (1990), Developing Expert Systems, John Wiley & Sons New York.

(6) SHANNON, R.E., R. MAYER ve H.H. ADELSBERGER (1985), " Expert Systems and Simulation ", Simulations, 44, no:6 (June), sayfa:275-284

(7) BAYAZIT, E.N. ve M. KAVAKLI (1992), " Uzman Sistem Metotları-2 ", Bilgisayar Magazin, sayı 15, sayfa:57-74.

(8) BANKS,J. ve J.S. CARSON (1984), Discrete-Event System Simulation, Prentice-Hall, Inc.

(9) SCHMIDT,J,W, ve R.E. TAYLOR (1970), Simulation and Analysis of Industrial Systems, Irwin, Homewood.

(10) HILLER, F.S. ve G.J. LIEBERMAN (1980), Introduction to Operations Research, 3rd.Ed., Holden-Day.

(11) NAYLOR,T.H. ve D. GATTIS, (1976) " Corporate Plannig Models ", California Management Review, sayfa:69-78, Summer.

(12) CONWAY, R.W. (1962), Some Tactical Problems in Simulation Methods, Memo RM-3244-PR, Rand Corporation, October 1962.

(13) SCHROER, B.J. ve F.T. TSENG (1989), " An Intelligent Assistant for Manufacturing System Simulation ", Vol.27, no:10, sayfa:1665-1683.

(14) HEIDORN, G.E. (1974), " English as a Very High Level language for Simulation Programming ", SIGPLAN Notice, 9, sayfa:91-100.

(15) FORD, D.R. ve B.J. SCHROER (1987), " An Expert Manufacturing Simulation System ", Simulations, 48, sayfa:193-200.

(16) KHOSHNEVIS, B. ve A.P. CHEN (1986), " An expert Simulation Model Builder ", Intelligent Simulation Environment, 17, sayfa:129-132

(17) HADDOCK, J. ve R.P. DAVIS (1985), " Building a Simulation Generator for Manufacturing Cell Design and Control ", In Annual Internatiol Endustrial Engineering Spring Conference Proceeding, Los Angales, California, sayfa:237-244.

(18) BRAZIER, M.K. ve R.E. SHANNON (1987), " Automatic Programming of AGVS simulation models ", In Proceedings of the 1987 Winter Simulation Conference, Atlanta, Georgia, sayfa:703-708

(19) DİNÇMEN, M., M.B. DURMUŞOĞLU, Z. AYAĞ ve T. ERTAY (1991), " Üretim Sistemlerinde Benzetim Amaçlı Uzman Sistemler ", Sanayide Bilgisayar Kullanımı ve Otomasyonu 90-91, İTÜ-KOSGEB.

(20) O'KEEFE, R. (1986), " Simulation and Expert Systems - A Taxonomy and Some Examples ", Simulation, 46:1, sayfa:10-16.

(21) DOUKIDIS, C.I. ve R.J. PAUL (1985), " Research into Expert systems to Aid Simulation Model Formulation ", J. Opl. Res. Soc.,36:4, sayfa:319-325.

(22) HADDOCK, J. (1987), " An Expert system Framework Based On a Simulation Generator ", Simulation, 48:2, sayfa:45-53.

(23) MICHE,D. (1980), " Knowlegde-Based Systems ", University of Illinois at Urbana-Champaign, report 80-1001.

(24) LAPIN, L.L. (1991), Quantitative Methods For Business Decisions, Harcourt Brace Jovanovich, Inc.

(25) DELMAR, D. (1982), Operational and Industrial Management, McGraw Hill Co.

(26) COOK, T.M. ve R.A. Russel, Introduction To Management Science, Prentice Hall,Inc., 1981

(27) ZEINGLER, B.P. ve T.I. ÖREN (1979), "Concepts for Advanced Simulation Methodologies", Simulation, No.32,3, sayfa:69-82.

(28) PEDGEN, C.D. (1982), Introduction to SIMAN, System Modelling Corporation, State College, Pa.

(29) MURRAY, K.J. ve S.V. SHEPPARD (1988), " Knowledge-Based Simulation Model Specification ", Simulations, 50:3, sayfa:112-119.

(30) ZHANG, S.X., B.J. SCHROER ve F.T. TSENG (1989), " An Automatic Programming Approach to Simulating Prelaunch Activities ", Simulations, 59:2, sayfa:23-29.

ÖZGEÇMİŞ

Tufan DEMİREL, 1966 yılında İstanbul'da doğdu. İlkokulu, Harun Reşit İlkokulu'nda (1977), ortaokulu, Fikirtepesi Ortaokulu'nda (1980) ve lise öğrenimini de Kabataş Erkek Lisesi'nde tamamladı (1983). 1984 yılında Yıldız Üniversitesi Endüstri Mühendisliği Bölümü'e girerek, 1989 yılında Endüstri Mühendisi olarak mezun oldu ve 1990 yılında da Yıldız Üniversitesi (bugünkü adıyla Yıldız Teknik Üniversitesi) Fen Bilimleri Enstitüsü'nde Endüstri Mühendisliği Dalı'da Yüksek Lisans öğrenimine başladı. Halen aynı üniversitede Endüstri Mühendisliği Bölümü Yöneylem Araştırması Anabilim Dalı'nda Araştırma Görevlisi olarak çalışmaktadır.

EK

Poliklinik Modelinin Veri Girişi

YILDIZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ENDÜSTRİ MÜHENDİSLİĞİ BÖLÜMÜ

UZMAN SİSTEM

Kuyruk Modelleri Simülasyon Uzmanı

Tasarımcı : Tufan DEMİREL

Simüle edilecek sistemin adı nedir ? POLIKLINIK

POLIKLINIK sistemine kaç farklı tipte müşteri gelmektedir ? 5

1. tip müşterinin gelme olasılığı nedir ? 0.45
2. tip müşterinin gelme olasılığı nedir ? 0.25
3. tip müşterinin gelme olasılığı nedir ? 0.05
4. tip müşterinin gelme olasılığı nedir ? 0.10
5. tip müşterinin gelme olasılığı nedir ? 0.15

POLIKLINIK modelindeki safha sayısı kaçtır ? 6

---

1. safhadaki sevisçi sayısı kaçtır ? 1
2. safhadaki sevisçi sayısı kaçtır ? 4
3. safhadaki sevisçi sayısı kaçtır ? 2
4. safhadaki sevisçi sayısı kaçtır ? 1
5. safhadaki sevisçi sayısı kaçtır ? 1
6. safhadaki sevisçi sayısı kaçtır ? 1

```
POLIKLINIK modelindeki gelişler hangi dağılıma uymaktadır ?

[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABİT DEĞER

Dağılım tipini giriniz : 1

Poisson dağılımına ait ortalama gelişler arası süre nedir ? 1.5
```

```
SİSTEME GELEN HER MÜŞTERİ TİPİNİN UĞRADIĞI SAFHALARIN BELİRLENMESİ

1.tip müşteri 1.safhaya uğruyor mu (E/H) ? E
1.tip müşteri 2.safhaya uğruyor mu (E/H) ? E
1.tip müşteri 3.safhaya uğruyor mu (E/H) ? H
1.tip müşteri 4.safhaya uğruyor mu (E/H) ? H
1.tip müşteri 5.safhaya uğruyor mu (E/H) ? H
1.tip müşteri 6.safhaya uğruyor mu (E/H) ? H
```

```
SİSTEME GELEN HER MÜŞTERİ TİPİNİN UĞRADIĞI SAFHALARIN BELİRLENMESİ

2.tip müşteri 1.safhaya uğruyor mu (E/H) ? E
2.tip müşteri 2.safhaya uğruyor mu (E/H) ? H
2.tip müşteri 3.safhaya uğruyor mu (E/H) ? E
2.tip müşteri 4.safhaya uğruyor mu (E/H) ? H
2.tip müşteri 5.safhaya uğruyor mu (E/H) ? H
2.tip müşteri 6.safhaya uğruyor mu (E/H) ? H
```

```
   SİSTEME GELEN HER MÜŞTERİNİN UĞRADIĞI SAFHALARIN BELİRLENMESİ

3.tip müşteri 1.safhaya uğruyor mu (E/H) ? E
3.tip müşteri 2.safhaya uğruyor mu (E/H) ? H
3.tip müşteri 3.safhaya uğruyor mu (E/H) ? H
3.tip müşteri 4.safhaya uğruyor mu (E/H) ? E
3.tip müşteri 5.safhaya uğruyor mu (E/H) ? H
3.tip müşteri 6.safhaya uğruyor mu (E/H) ? H
```

```
   SİSTEME GELEN HER MÜŞTERİ TİPİNİN UĞRADIĞI SAFHALARIN BELİRLENMESİ

4.tip müşteri 1.safhaya uğruyor mu (E/H) ? E
4.tip müşteri 2.safhaya uğruyor mu (E/H) ? H
4.tip müşteri 3.safhaya uğruyor mu (E/H) ? H
4.tip müşteri 4.safhaya uğruyor mu (E/H) ? H
4.tip müşteri 5.safhaya uğruyor mu (E/H) ? E
4.tip müşteri 6.safhaya uğruyor mu (E/H) ? H
```

```
  SİSTEME GELEN HER MÜŞTERİ TİPİNİN UĞRADIĞI SAFHALARIN BELİRLENMESİ

5.tip müşteri 1.safhaya uğruyor mu (E/H) ? E
5.tip müşteri 2.safhaya uğruyor mu (E/H) ? H
5.tip müşteri 3.safhaya uğruyor mu (E/H) ? H
5.tip müşteri 4.safhaya uğruyor mu (E/H) ? H
5.tip müşteri 5.safhaya uğruyor mu (E/H) ? H
5.tip müşteri 6.safhaya uğruyor mu (E/H) ? E
```

```
1. safhadaki 1. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 1
```

```
2. safhadaki 1. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 12.5
```

```
2. safhadaki 2. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 14.5
```

```
2. safhadaki 3. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 12.5
```

```
2. safhadaki 4. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DAĞILIM

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 11.5
```

```
3. safhadaki 1. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DAĞILIM

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 23
```

```
3. safhadaki 2. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 27
```

```
4. safhadaki 1. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 19
```

```
5. safhadaki 1. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 21
```

```
6. safhadaki 1. servis için servis süresi dağılımı nedir ?


[1] POISSON DAĞILIMI
[2] USSEL DAĞILIM
[3] DUZGUN DAĞILIM
[4] NORMAL DAĞILIM
[5] SABIT DEĞER

Dağılım tipini giriniz : 2

ÜSSEL dağılımına ait ortalama servis süresi nedir ? 25
```

```
Simülasyonun çalışma süresini giriniz          :480

Kabul edilebilir kullanım oranını giriniz  (%)   :65
```