YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANAYİ TESİSLERİNDE KAYNAK YERLEŞİMİNDE KULLANILAN UZMAN SİSTEMLER İÇİN KURAL VE ALGORİTMALARIN SAPTANMASI

Elek.Müh. Seyhun YENİPINAR

F.B.E. Elektrik Mühendisliği Anabilim Dalında
hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Doç. Dr. Galip CANSEVER

İSTANBUL , 1996

İÇİNDEKİLER

IV. SONUÇ ÇIKARIM MEKANİZMASI

V. MONTAJ HATLARINDA KAYNAK YERLEŞİMİNDE KULLANILAN, KURALLARIN VE ALGORİTMALARIN TAYİNİ İÇİN BİLGİ TABANLI SİSTEMİN OLUŞTURULMASI

TEŞEKKÜR

20. Yüzyılın şu son yıllarında uzmana duyulan ihtiyaç ve bunun paralelinde verilen önem gün geçtikçe artmaktadır. Günümüzde, bilgisayarların da insanlığın vazgeçemediği bir araç haline gelmesiyle birlikte, uzman gereksiniminin bilgisayarlar tarafından karşılanması gündeme gelmiştir. Böylelikle uzman sitemler'de literatürdeki yerini almıştır. Gelişmeler onu gösteriyor ki, uzman sistemler de insanoğlunun vazgeçemediği unsurlar arasınada yerini alacaktır.

Böyle zevkli bir konuyu araştırma fırsatını veren ve yardımlarını esirgemeyen hocam Doç.Dr Galip Cansever beye teşekkürlerimi borç bilirim

EKİM 1996 SEYHUN YENİPINAR

ÖZET

Uzman sistemler, uzmanların problem çözme ve karar verme işlemleri doğrultusunda hareket etmeye çalışan bilgisayar programlarıdır.

Uzman sistemlerin üç ana modülü vardır. Bunlar; Bilgi tabanı, çıkarım mekanizması ve kullanıcı ara yüzüdür.

Bilgi tabanı uzman tarafından sağlanan problem hakkındaki özel bilgilerin saklandığı kısımdır. Burada kurallar, kavramlar problemin gerçekleri ve ilişkiler bulunmaktadır. Bilgileri bilgi tabanında nasıl kayda döktüğümüz ve bilgi gösterim tekniklerinin konusudur ki bu bölüm 2'de incelenmiştir.

Sonuç çıkarım mekanizması, uzmanın sonuçlandırmasından sonra modellenen bilgi işlemcisidir. Mekanizma problem hakkındaki bilgilerle, bilgi tabanındaki saklanan bilgileri eşleyerek sonuç ve tavsiyeler üretir. Buna genellikle uzman sistem shell'i ya da kısaca shell denilmektedir. Bu mekanizmayı nasıl dizayn etmemiz gerektiği sonuç çıkarım tekniklerinin konusu olup bölüm 4'de incelenmektedir.

Kullanıcı arabirimi, sistem kullanıcısının kuralları ve gerçekleri girmesine ve sistem hakkında sorular sormasına imkan verir.

Uzmanlar çalıştıkları kurumlar için değerli kaynaklardır. Onlar yenilikçi fikirler üretebilir, çok zor problemleri çözebilir ya da verimli bir şekilde rutin işleri yapabilirler. Ama yapılan bunca işi bir uzman sistem ile gerçekleştirmek istenmesinin bir takım sebepleri vardır. Bunlar:

Uzman sistemlerin uzmanlar gibi belirli mesai saatleri yoktur. Devamlı çalışabilirler. Ayrıca diğer bilgisayar programları gibi uzman sistemlerin bir nüshasını kolaylıkla temin edebilirler.

İnsan uzmanlar devamlı değildirler. Ölüm, emeklilik ya da iş transferleri gibi durumlarda uzmanlardan yararlanmak mümkün olmaz.

Uzman sistemler, insan uzmanlığının ürettiğinden daha istikrarlı, tekrar edilebilir sonuçlar üretebilir. Bir uzman ise duygusal faktörlerden dolayı belirli aynı durumlar için farklı kararlar alabilir

Uzman sistemlere verilen önem, saydığımız ve diğer birçok avantajlarının bir sonucu olarak, gün geçtikçe artmaktadır.

SUMMARY

Expert system is a computer program designed to model the problem-solving ability of a human expert. There are three main modules that expert system contains. These modules are:

The knowledge-base contains highly specialized knowledge on the problem area as provided by the expert. It includes problem facts, rules, concepts and relationships. How we code this knowledge in the knowledge-base is the subject of knowledge representation techniques and is covered in chapter 2.

The inference mechanism is the knowledge processor whichis modeled after the expert's reasoning the mechanism works with available information on agiven problem, coupled with the knowledge stored in the knowledge base, to draw conclusions or recommendations. How we design the mechanism is the subject of inference techniques and is covered in chapter 4.

User interface gives user the oportunity to enter the rules and facts.

Experts represent a valuable resource for any organisation. They can offer creative ideas, solve difficult problems, or efficiently perform routine tasks. But there are some reasons capturing this talent in an expert system:

Like any machine, an expert system continues to churn long after the expert's workday. Like any computer program, we can cheaply duplicate it.

Human expertise is perhisable. Through death, retirement, or job transfer, an organization can lose the talents of an expert.

An expert system produces more consistent results than a human expert. Human decision making is influenced by many factors that might impact performance. In an emergency situation, the expert might forget some important piece of knowledge.

The speed at which an expert solves a problem is also influenced by many factors. But the speed of an expert system is consistent.

Human experts tendto be expensive. But expert systems are relatively inexpensive.

As a result of advantages of expert systems, we can say that the importance of expert system is growing day by day.

BÖLÜM 1.
UZMAN SİSTEMLER

1. 1 Giriş

Uzman sistemlerin tanımına literatürde değişik kaynaklarda rastlamaktayız. Uzman sistem için yapılan tanımları inceledikten sonra genel bir tanıma ulaşalım:

Uzman sistemler, uzmanların problem çözme ve karar verme işlemlerinin doğrultusunda hareket etmeye çalışan bilgisayar programlarıdır. Uzman sistemlerin amacı, spesifik problemleri çözmek, sunmak ve çözümü detaylı bir şekilde terimlerle kullanıcıya açıklamaktır.

Uzman sistemler; uzman insanın dizayn, oluşturma, planlama, teşhis etme, yorumlama, özetleme, ve tavsiye etme gibi yapabileceği türden faaliyetleri yapmak için oluşturulmuş bilgisayar programlarıdır.

Uzman sistem, bir problem alanındaki bir veya daha fazla uzmanın bilgi ve becerisini kullanan bilgisayar sistemidir. Uzman sistemler, kullanıcılara faydalı çıkarımlar yapmak için uzmanların problem çözme deneyimlerini kullanır.

Yukarıdaki tanımlardan anlaşıldığı gibi Uzman sistemler bilgisayar programlarından başka birey değildir. En önemli özelliği genellikle "zeki" tarzda işlem yapacak şekilde hazırlanmış olmalarıdır. Bu sistemde amaç, bilgisayarda uzman insanı kopyasını sağlamaktır. Bunu sağlayacak programların işlevi uzman insanın yaptığı işlevler olacaktır. Bu işlevlerin uzman insanın düşünme prosesini oluşturduğunu söyleyebiliriz.

Probleme ait veriler ve problem tanımlanmasında yardımcı olacak bilgiler kara kutunun girdilerini oluşturmakta, problemin çözümü ve karar verme ise çıktıları gösterecektir. Kara kutu ise uzman insanın kendisidir. Fakat şimdiye kadar uzman insanın fonksiyonların tamamını yerine getiren bir uzman sistem programı geliştirilememiştir. Tüm bu söylenenler ışığında Uzman sistemlerin tanımı şöyle verilebilir;

Uzman sistemler; tavsiye analiz etme, kategorilere ayırma, haberleşme, danışma, dizayn, teşhis etme, açıklama, araştırma, tahmin, tanımlama, yorumlama, doğruluğunu kanıtlama, öğrenme, yönetme, planlama, programlama, test etme ve öğrenme fonksiyonlarını yerine getiren programlardır. Bu programlar, normalde çözümleri için insanın gerektiği düşünülen problemleri ele alırlar (Durkin, J., 1994)

1.2 Uzman Sistemlerin Oluşturulmasında Rol Oynayan Elemanlar

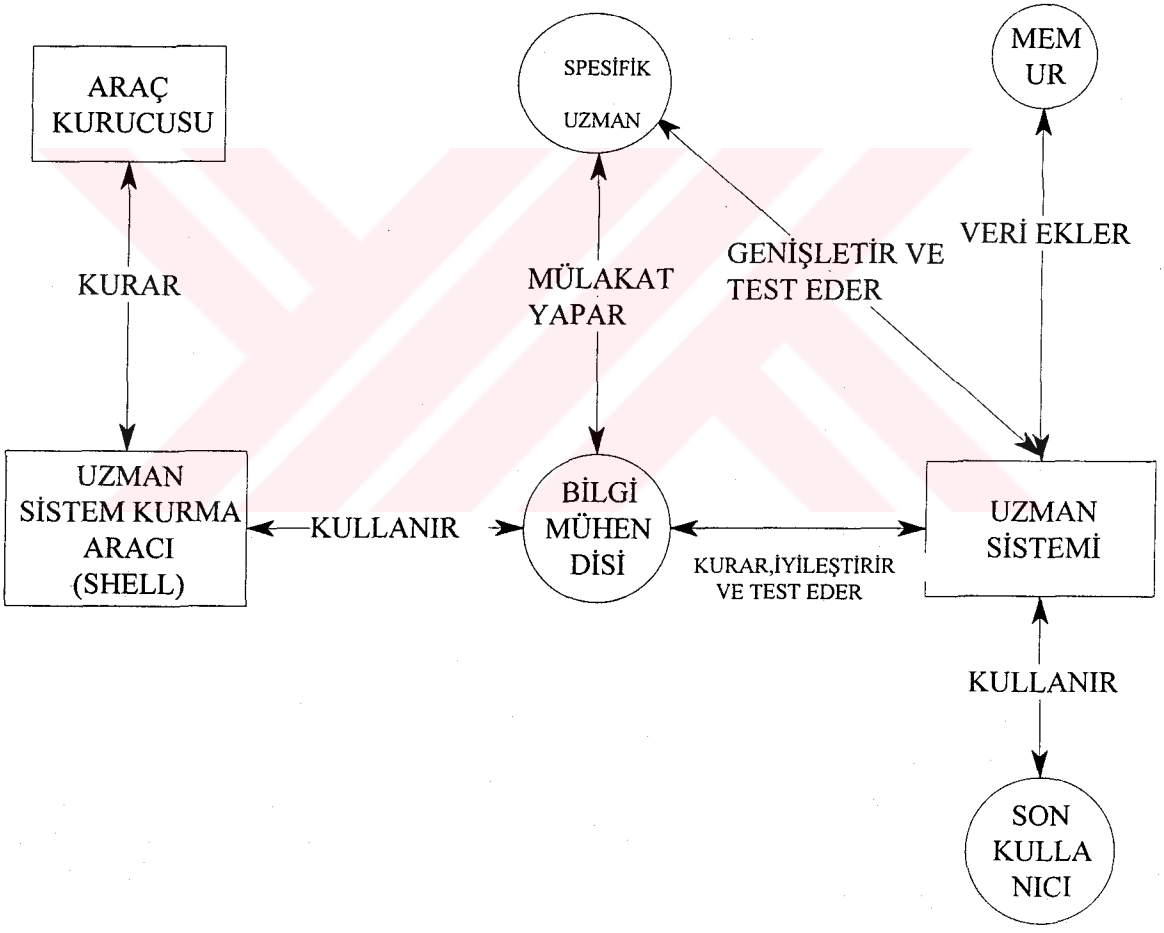
Bir uzman sistemin kurulmasında rol oynayan elemanlar şunlardır:

Uzman sistem, insan uzman, bilgi mühendisi,uzman sistem kurma araçları ve kullanıcılarıdır. Bunların temel fonksiyonları ve birbiri ile olan ilişkileri şekil 1.1'de özetlenmiştir.

Uzman sistem ilgili alandaki problemleri çözen bilgisayar yazılım ve programların bütünüdür. Hem problem çözme kısmını ve hem de bir destek kısmını içerdiği için onu sadece bir program görmekten öte bir sistem olarak nitelendirilir. Bu destek yapısı kullanıcının ana programla ilişki kurmasına yardımcı olur. Bu destek yapısı Tablo 1.1 de belirtilen elemanları içerebilir. Bu elemanların fonksiyonları tabloda ayrıca belirtilmiştir.

Bu aşamada uzman sistemin oluşmasında rol oynayan elamanların fonksiyonlarını incelemek gerekmektedir.

Spesifik uzman (Domain expert):İlgilendiği konu veya alan hakkındaki düşüncelerini çok net bir şekilde açıklayabilen o alandaki problemlere iyi çözümler üretmek için genel bir düşünceye sahip bilgili bir kişidir. Uzman, etkin bir çözüm bulmak için ustalığını ve tecrübenin kendisine kazandırdığı kısa yolları kullanır. Uzman sistem bu problem çözme stratejilerini modeller. Konuları tanımlar ve başarı performansını belirler.



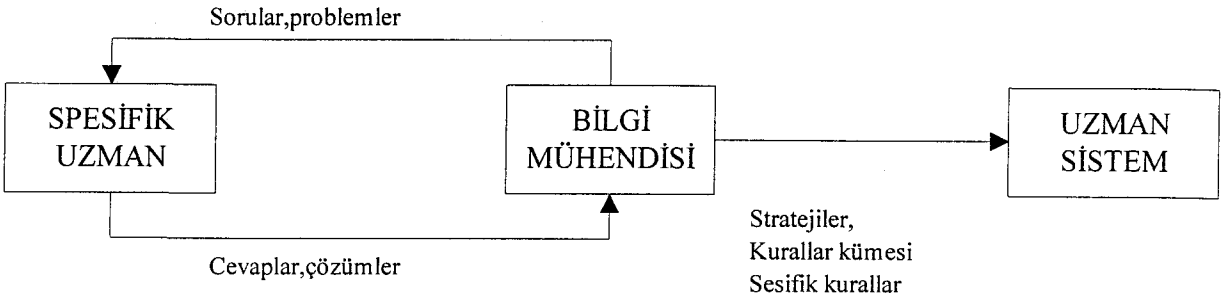
Şekil 1.1 Uzaman sistemim kurulmasında rol oynayan elemanlar

Tablo 1.1. Bir uzman sistemde olabilecek destek yapı elemanları.

DESTEK YAPI ARAÇLARI	YARDIMI ALANLAR	FONKSİYONUN TÜRÜ
Kompleks hata ayıklama yardımları	Uzman Sistem Kurucusu	Programın kodunu test etme ve değerlendirme
Tashih araçlar	Uzmanlar	Uzman sistemin içinde bilgi ve veriyi değiştirmek
Gelişmiş grafik araçları	Kullanıcı	Sistem çalıştıkça bilgi girişi ve okuma

Bilgi mühendisi (Knowledge engineer): Bilgisayar bilimi ve yapay zeka konusunda temel bilgisi olan, uzman sistemlerin nasıl oluşturulacağını bilen bir insandır. Bilgi mühendisi organizasyonda uzmanlarla görüşmeler yapar, bilgiyi organize eder, uzman sistemde bilginin nasıl temsil edilmesi gerektiğine karar verir ve kodların yazılımında programcılara yardım eder. Bilgi mühendisi uzmanlarla olan mülakatında problemin çözümü için gerekli prosedürleri stratejileri ve kuralları elde etmeye çalışır. Daha sonra bu bilgiyi uzman sistem içinde kurmaya çalışır. Bu bilgi transferi şekil 1.2'de gösterilmiştir.

Bir sistemi dizayn eden ve yürüten aynı zamanda uzmanlarla etkileşimli olan klasik bir software mühendisi ile karşılaştıracak olursak, bilgi mühendisi uzmanlarla çok daha fazla zaman harcar. Bilgi mühendisleri uzmanın düşünme prosesi ile çok daha fazla ilgilidirler. Sistemi geliştirip bitirene kadar uzmanla kontak halinde olmaya çalışırlar.



Şekil 1.2 Bilgi mühendisliği: Bilginin bir uzmandan bir bilgisayar programına transferi.

Projenin veya şirketin büyüklüğüne bağlı olarak, birden fazla bilgi mühendisi olabilir. Bu mühendisleri kapasitelerine göre bir çok gruba ayırabiliriz. Örneğin, bunlardan biri tüm yönetim ve dizayn sorumluluğunu üstlenirken diğerleri uzmanla günübirlik görüşmelerini sürdürürler. Ayrıca yüksek seviyeli mühendislerden oluşan bir grup ta kodları bir makinaya girmek için daha önceden sorumluluğu alabilirler. Basitleştirmek için bu mühendislerden her biri tüm bilgi mühendisinin rollerini yerine getireceğini varsayarak, bir tek bilgi mühendisinden bahsedeceğiz. dolayısıyla , bilgi mühendisleri, uzmandan bilgiyi elde etmede, bilgi içeren bir uzman sistemin prototipini kurmada ve daha sonra da sistemi geliştirmek için uzmanla birlikte çalışan uzmanlardır. Şekil 1.3'te bilgi mühendisinin rollerini özetlemektedir.

Uzman sistem kurma aracı (The expert systems building tool): Uzman sistem oluşturmada bilgi mühendisi veya programcı tarafından kullanılan programlama dilidir Bu araçlar, problemlerin çözümü için kullanıcılara yardımcı olan bilgi sistemlerinin kurulmasında bilgi mühendislerine imkan sağlar. Uzman sistem kurma araçları konvansiyonel (klasik) programlama dillerinden farklılık gösterir. Bu araçlar yüksek seviyeli kavramları ve kompleksliliği göstermede uygun yöntemler sağlar. Yapay zeka terminolojisinde Araç (tool) terimi hem programlama diline hem de uzman sistemin oluşturulmasında kullanılan destek yapısına karşılık gelir.

Kullanıcı (The user): Geliştirilen uzman sistemi kullanan kişidir. Kullanıcı;

- yeni mineral kaynakların keşfedilmesinde yardımcı olarak sistemi kullanan bir bilim adamı olabilir,

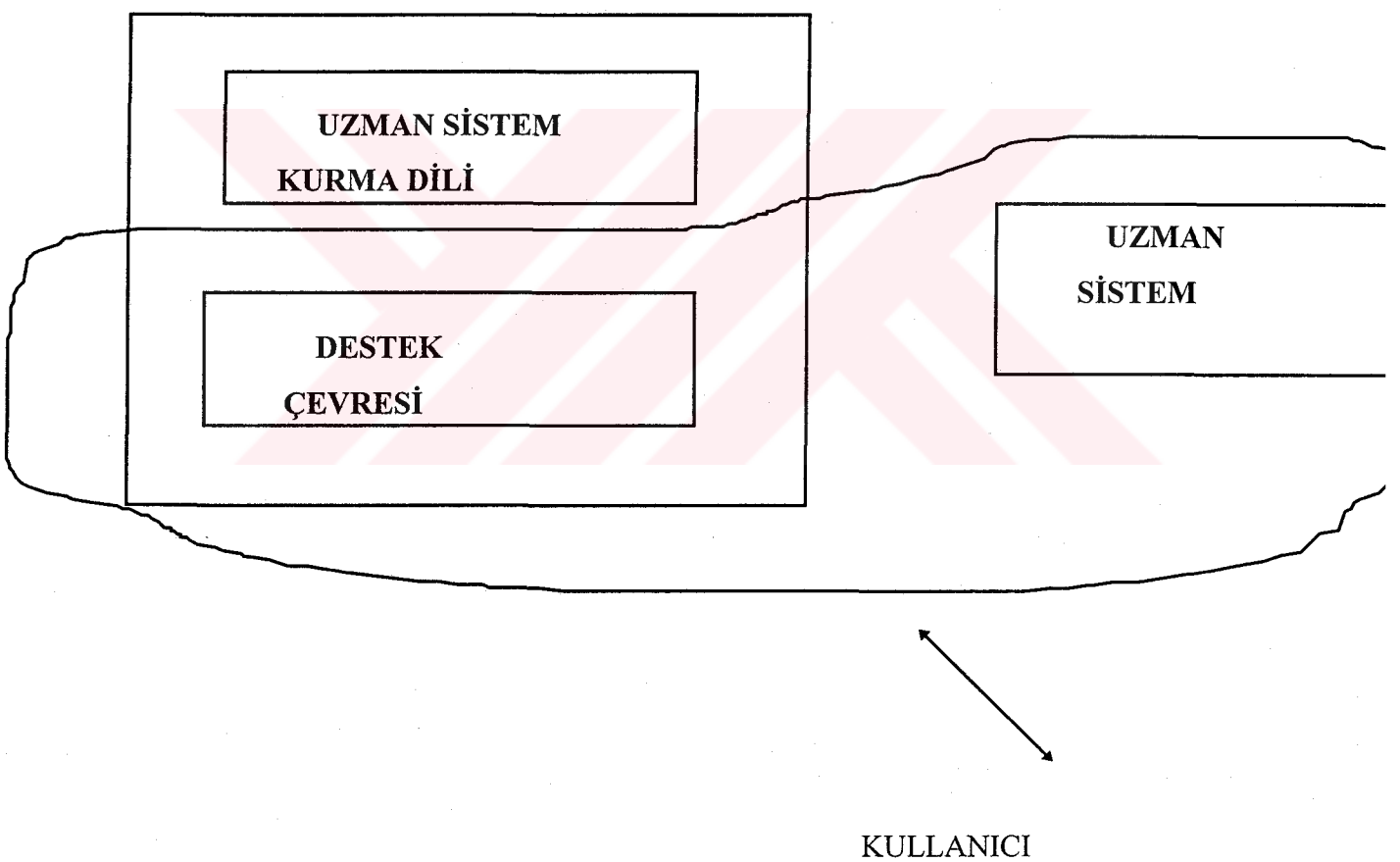
-Herhangi bir vakayı saptamada uzman sistemi yardim aracı olarak kullanan bir hakim veya savcı olabilir.

Kullanıcı terimi azda olsa belirsiz bir terimdir. genellikle son kullanıcı (end user) manasına karşılık gelir. Son kullanıcı aslında uzman sistemin geliştirilip kendisinin kullanımına sunulan kişidir. Bizim kullandığımız manada ise uzman sistemi kullanan herkese karşılık gelecektir. Bu açıklamaya göre şekil 1.1'den de anlaşılacağı üzere kullanıcı;

- Uzman sistem oluşturma dilini inceleyen, hataları gideren, bir araç kurucusu (tool builder) olabilir.
- Tavsiye için sisteme güvenen bir son kullanıcı ,
- Sistemde mevcut bilgiyi saflaştıran bir spesifik olabilir,
- Sisteme veri ekleyen memur personel den biri olabilir.

Şekil 1.3 Bilgi mühendisinin rolü

Uzman sistemi oluşturmak için kullanılan araç ile uzman sistemin kendisi arasındaki farkı anlamak çok önemlidir. Uzman sistem oluşturma aracı hem sistemde içerilen bilgiye erişmek ve temsil etmek için kullanılan programlama dilini hem de destek yapısını kapsar. Destek yapısı olarak tanımladığımız programlar, kullanıcıların uzman sistemin problem çözme kısmı ile daima etkileşimli olmalarına yardımcı olurlar. Bilgisayarla uğraşan bilim adamları bile bu farklılığı birbirine karıştırabilirler. Aynı zamanda, destek yapısı, tamamlanmış uzman sistemin bir parçasıdır. Kullanıcı bu destek yapısı yoluyla uzman sistemle ilişki kurduğu için, karışıklığın nasıl oluşabileceğini görmek çok kolaydır. Şekil 1.4 bunu ifade etmektedir.



Şekil 1.4 Uzman Sistem kurma aracı uzman sistem değildir.

1.3 Neden Gerçek Uzmanlar Yerine Yapay Uzmanlar Kullanılıyor?

Aklımıza şöyle bir soru gelebilir:Geçmişte olduğu gibi insan deneyi ve tecrübelerine güvenmek varken neden uzman sistemler geliştiriliyor?

Uzman sistem yerine yapay uzmanlıktan (yapay deneyim ve tecrübelerden) yararlanmanın birçok sebepleri vardır. Bu avantajların bazıları Tablo1.2'de özetlenmiştir.

Yapay uzmanlığın avantajlarından biri sürekli olmasıdır. Oysa insan Uzmanlığının performansı çok hızlı bir şekilde azalır. Bir uzman ilgili olduğu problem alanlarında profesyonelliğini devam ettirebilmesi için sürekli olarak o konularda uygulamalar yapmak zorundadır. Önemli derecede atıl kalma durumu uzmanın performansını ciddi bir şekilde etkiler. Bilgi kullanılmazsa kaybolur deyimi işte burada geçerli olmaktadır. Fakat bu durum yapay uzmanlık için söz konusu değildir. Bir kere bu deneyim kazanılıp kodlandı mı her zaman için geçerlidir. Tâbi ki hafızada depolamayla ilgili kazalar hariç. Bu yapay uzmanlığın performansı onun kullanımıyla ilgili değildir.

Tablo 1.2. İnsan ve yapay uzmanlığın karşılaştırılması.

AVANTAJLARI	
İNSAN UZMANLIK	**YAPAY UZMANLIK**
• Zamanla bozulur	• Süreklidir
• Transferi zordur	• Transferi kolaydır
• Dokümanı zordur	• Dokümanı kolaydır
• Tahmin edilemez	• İstikrarlı
• Pahalı	• Kolayca sahip olunabilir

Yapay uzmanlığın diğer bir avantajı ise transfer işleminin veya yeni bir tanesini elde edilmesinin kolay olmasıdır. Bir insandan diğerine bilgi transferi, eğitim veya bilgi mühendisliği olarak kabul ettiğimiz çok büyük bir çaba isteyen, uzun dönemli ve pahalı bir prosestir. Yapay uzmanlığın transferi ise çok basit bir kopyalama işleminden başka bir şey değildir (Jakson P., 1986).

Yapay uzmanlığın doküman haline getirilmesi çok daha kolaydır . İnsan uzmanlığın dokümantasyonu ise son derece zor ve çok zaman alır. Yapay uzmanlığın dokümantasyonu ise nispeten kolaydır.

Yapay uzmanlık, insan uzmanlığının ürettiğinden daha istikrarlı, tekrar edilebilir sonuçlar üretebilir. Bir uzman ise duygusal faktörlerden dolayı belirli aynı durumlar için farklı kararlar alabilir. Örneğin, bir uzman insan bir kriz durumunda zaman baskısı (kısıtlılığı) veya stresten dolayı çok önemli bir kuralı kullanmayı unutabilir.

Yapay uzmanlığın son bir avantajı maliyetinin düşük olmasıdır. Uzman insanlar,özellikle üst seviyeli kişiler ,çok nadir bulunurlar. Dolayısıyla çok pahalıdırlar. Çok büyük miktarda maaş istemekte ve almaktadırlar. Uzman sistemler nispeten pahalı değillerdir. Geliştirilmesi çok pahalı olmasına rağmen çalışması ucuzdur. Bunların çalışma maliyeti sadece programın çalışmasını sağlayan nominal bilgisayar maliyetidir. Bunların çok yüksek geliştirme maliyeti, düşük çalışma maliyeti ve sistemin kopyasının kolaca alınması ile kapatılmaktadır. Geliştirme maliyetinin yüksek olması ,uzman sistemlerin çok yüksek maaşlı bilgi mühendisleri tarafından ve uzman insanları yıllar süren çalışmalarıyla kurulmasından kaynaklanmaktadır.

1.4 Niçin Uzman İnsanı Dışlamıyoruz?

Yukarıdaki kısımda açıklandığı gibi ,madem yapay uzmanlık insan uzmanlığından o denli avantajlı ve uygun, niçin uzman insanları elimine edip yerlerine uzman sistemlerini ikame etmeyelim? Akla böyle bir sorunun gelmesi mümkündür. Yüksek seviyede bilgi ve beceriye sahip uzmanı elimine etmek mümkündür. Fakat çoğu durumlarda, orta seviyeli uzmanların sistemde tutulmasında fayda vardır. Uzman sistem bu kullanıcıların bilgi ve becerilerini artırmak için kullanılabilir.

Uzman insanı sistemden tamamen elimine etmemek için birçok iyi sebepler vardır. Bu sebepler yapay uzmanlığın dezavantajlarını ifade etmektedir. Bu dezavantajların bazıları tablo 1.3'te verilmiştir. Uzman sistemler iyi çalışmasına rağmen, uzman yapay olandan daha iyi çalıştığı önemli alanlar vardır. Fakat bu yapay zekanın temel bir sınırlamasını göstermez,sadece YZ sanatının mevcut durumunu gösterir.

Tablo 1.3. İnsan ve yapay uzmanlığın karşılaştırılması

AVANTAJLARI	
İNSAN UZMANLIĞI	**YAPAY UZMANLIK**
• Yenilikçi	• Yenilikçilikten yoksun
• Uyumlu	• Açıklamaya ihtiyaç duyar
• Duyu uzmanlık sahibi	• Sembolik girdi
• Geniş alanlı	• Dar alanlı
• Sağduyu bilgisine sahip	• Teknik bilgiye sahip

İnsan umanlığının avantajlı olduğu alanlardan bir tanesi yenilikçi olmasıdır. İnsanlar en zeki programlardan dahi daha çok yenilikçi ruha sahiptir. Bir uzman insan bilgiyi yeniden organize eder ve yeni bilgiyi sentezlemek için bu bilgiyi kullanır. Oysa bir uzman sitem bir anlamda yenilikçi ruhundan yoksun, rutin yönde hareket eder. İnsanlar problem çözümünde hayalci yaklaşımlar beklenmeyen olayları ele alırlar. Bu yaklaşımlar arasında ,tamamen birbirinden farklı problem alanlarındaki durumlar için benzerlikleri tanımlamakta bulunmaktadır. Zeki programlar olarak tanımladığımız uzman sistemler bu fonksiyonları yapmakta az başarılıdırlar.

İnsan uzmanlığın yapay olana daha üstün olduğu alanlardan bir diğeri ise öğrenme özellinin olmasıdır .Değişen şartlara uyum sağlarlar. Dolayısıyla yeni şartlara uygunluk sağlamak için stratejilerini ayarlarlar. Uzman sistemler özellikle yeni kavramlar veya yeni kurallar öğrenmede becerikli değillerdir. Bu programlar son derece basit alanlarda iyi çalışmaktadırlar Fakat detaylı gerçek dünya problemleri ve onların kompleksliği ile karşılaştıklarında iyi çalışmamaktadırlar.

İnsanlar kompleks sezgisel verilerden doğrudan faydalanabilirler. Bu sezgisel verilerini görme, işitme veya koklamayla ilgisi olsun veya olmasın fark etmez. Fakat uzman sistemler fikirleri ve kavramları temsil eden sembolleri kullanır. Sezgisel verilerin sistem tarafından anlaşılabilmesi için sembollerle ifade edilmesi gerekmektedir. Yani sezgiselden sembole doğru bir transformasyon gerekmektedir. Tabi ki bu işlem sonucu bilginin çok az bir kısmı kaydedilmiş olabilir. Örneğin, karşımızda bir resmin varolduğunu düşünelim. Bu resmi nesneler ve aralarındaki ilişkiler seti şeklinde açıklarken birçok bilgiyi göz ardı etmek zorunda kalmaktayız. Çok eski bir deyim bu durumu kısa fakat net bir şekilde açıklamaktadır:bir resim bin kelimeye bedeldir. Bu da sembolik ifade gerektiren uzman programların dezavantajlarını ifade etmektedir.

Son olarak diyebileceğimiz diğer bir hususta ,ister uzman insanlar olsun ister uzman olmayan insanlar olsun, hepsi de yaşam tecrübesinin kendilerine kazandırdığı bilgiye sahiptirler (commonsense knowledge - sağduyu bilgisi). Bu da dünya olayları hakkındaki çok geniş bir yelpaze ifade etmektedir. Bu bilgi hemen hemen herkes tarafından bilinir ve kullanılır. Bu bilginin çokluğundan dolayı, bu bilgileri özellikle uzman sistem gibi bir zeki bilgisayar programın içinde yapılaştırmanın hiçbir kolay yolu yoktur.

Bu sağduyu bilgisi bir kişinin neyi bildiğinin yanısıra neyi bilmediğinin bilincinde olmayı da içerir. Bunu bir örnekle açıklayalım:Örneğin daha önce oturduğumuz evin telefon numarası sorulursa ,önceden bildiğimiz için, ezberden söylemeye çalışırız. Fakat bize Türkiye'nin Cumhurbaşkanının evinin telefon numarası sorulursa, hemen bilmediğimizi söyler, kafamızı kurcalamayız. Eğer bize Fatih Sultan Mehmet in telefon numarası kaçtı diye bir soru yöneltilirse, onu zamanında telefon icat edilmediği için, bu soru bilgi tabanındaki bilgi ve kuralları cevabı olmadığını biliriz. Oysa uzman bir sisteme ,cevablayamıyacağı ve cevabının olmadığı böyle sorular sorulursa ,insanların bu özelliğini taşımadığı için, soruların cevaplarının neden bulunmadığı konusunda hemen bir yargıya varamıyacaktır. Bunun yerine bir çözüm için bilgi ve kuralları aramakta çok zaman harcayacaktır. Daha kötüsü çözüm bulunmadığı zaman bilgisinin yetersiz olduğunu düşünebilir ve bilgi tabanını tamamlamak için ek bilgi (additional information) isteyecektir.

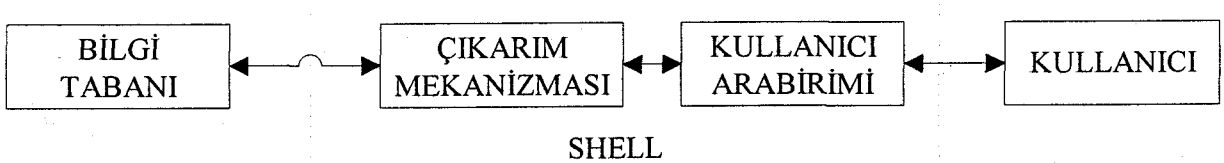
Dolayısıyla, bu sonuçlarla yapay uzmanlığın genel kabulü ile ilgili diğer sonuçlar dikkate alınırsa, uzman sistemler genellikle bir danışman olarak veya gerek bir uzman için gerekse uzman olmayan birisi için yardımcı olma mahiyetinde kullanılırlar.

1.5 Bir Uzman Sistemin Genel Yapısı

Ivan Bratko bir uzman sistemin temel yapısının üç ayrı elemandan oluştuğunu belirtmiştir (Beerel A.J. 1987). Şekil 1.5' de gösterildiği gibi, sistemin üç elemanı:

- bir bilgi tabanı,
- bir çıkarsama mekanizması ,
- bir kullanıcı interface (arabirimi)

Bir bilgi-tabanı uygulama alanı için spesifik bir anlam taşıyan bilgiden oluşmaktadır. Bu bilgi ,bu spesifik alanla ilgili gerçekleri ,ilişkileri veya olayı tanımlayan kuralları içerir. Ayrıca problemlerin çözümü için gerekli metotları ,höristikleri ve fikirleri de içerebilir. Sonuç çıkarım mekanizması bilginin aktif olarak nasıl kullanılacağını bilmektedir. Kullanıcı interface ,kullanıcı ve sistem arasındaki pürüzsüz haberleşmeyi temin eder. Sonuç çıkarım mekanizması ve kullanıcı ınterface'ini birlikte bir eleman olarak düşünmek mümkündür. Buna genellikle uzman sistem shell'i veya kısaca shell denilmektedir.



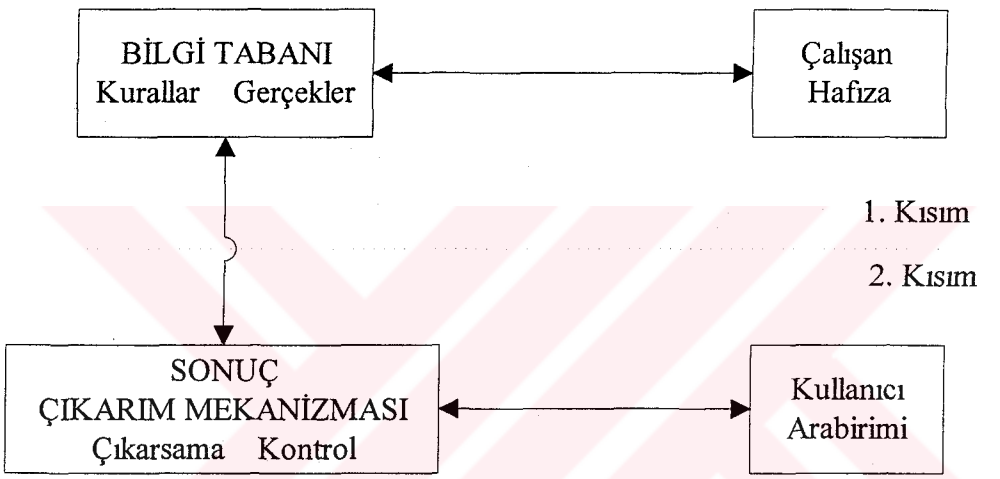
Şekil 1.5 Uzman Sistemlerin Genel Yapısı

Paul harmon bir eleman daha eklemekte ve sistemi iki ana kısma ayırmaktadır. Çalışan Hafıza olarak isimlendirdiği bu yeni bilgi tabanıyla etkileşim halindedir. İki ana kısmı oluşturan elemanları ise şöyle gruplandırmaktadır:

I.kısım: Bilgi tabanı ve çalışan hafizadan oluşmakta,

II.kısım: Çıkarsama mekanizması ,eklenebilecek tüm alt sistemler ve interface' lerden oluşmaktadır.

Bu açıklamaya göre şekil 1.5'deki gösterim şekil 1.6'deki gibi olacaktır.



Şekil 1. 6 Yapısı iki kısma ayrılmış bir uzman sistem

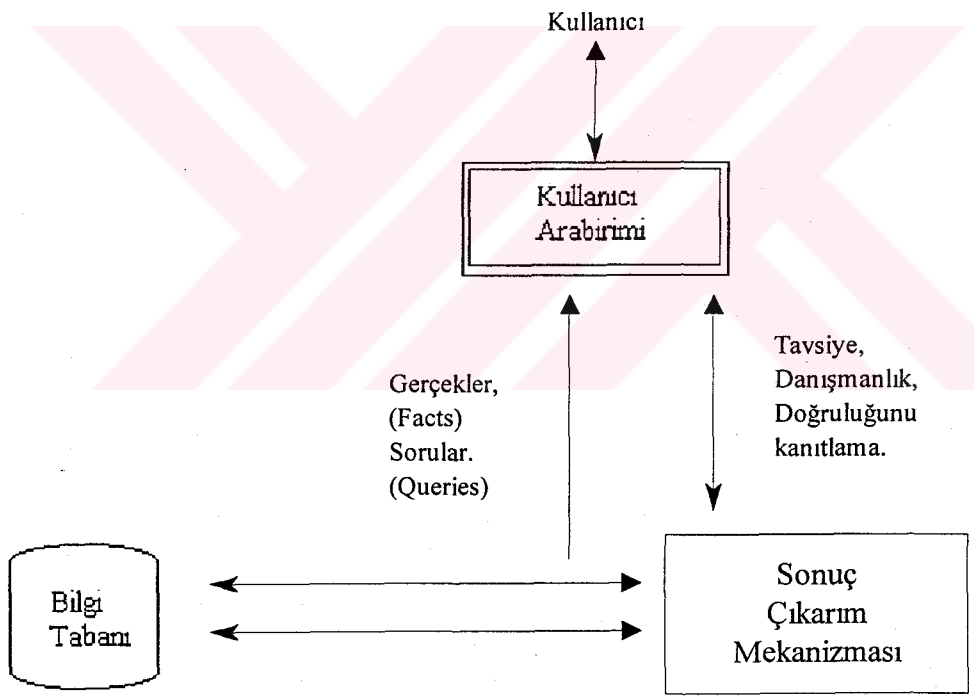
Uzman Sistemin Yapısına Eklenebilecek Alt Sistemler

Bugünkü uzman sistemlerin yapıları ,bilginin nasıl temsil edilmesi gerektiği ve bir bilgisayar bazlı sistemle desteklenmiş zeki karar verme konularının nasıl yürütülmesi gerektiği gibi sorular karşısında bilgi mühendislerinin düşüncelerini yansıtmaktadır.

Özel bir alandaki pratik deneyimler ve lâboratuardaki denemeler doğrultusunda ,bilginin nasıl temsi edileceği ve nasıl kullanılacağı daha çok şey öğrenilmektedir. Böylece temel yapısal prensipler anlaşılır hale gelmeye başlıyor.

Şekil 1.7 basit bir uzman sistemin yapısını göstermektedir. Bu sistemin yapısı spesifik bilgisayar donanımından bağımsızdır. bilgisayar donanım seçimi için belirleyici faktörler arasında;

- bilgi veri-tabanının büyüklüğü,
- sistemin arzu edilen cevap verme hızı ,
- geliştirme yapısı (development environment) ve
- kullanıcı arabirimi için karmaşıklığın seviyesi sayılabilir.



Şekil 1.7 Basit bir Uzman Sistemin yapısı

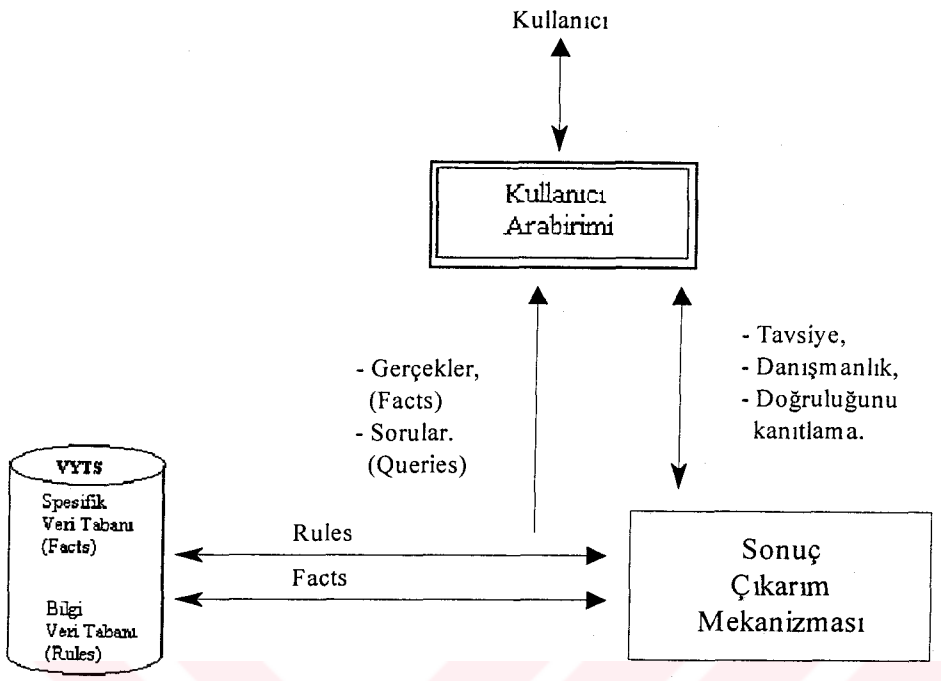
Kullanıcı arabirimi, sistem kullanıcısının kuralları ve gerçekleri girmesine ve sistem hakkında sorular sormasına kan sağlar. Bunun karşılığında , kullanıcı isteklerine cevaplar temin eder. Ayrıca sistem ile kullanıcı arasındaki diğer tüm haberleşmelerde desteği sağlar.

Bilgi-tabanı, özel bir konuda bir uzman insanın bilgisini ,kodlanmış bir formda ,ihtiva eder. Bu kodlamada işlemi,bilginin okunması ve anlaşılması için genellikle kolay olur. Bilgi tabanları uzmanlar tarafından geliştirilmesine rağmen ,içerdikleri bilgilerin konu hakkında bilgisi olan herkes tarafından anlaşılabilir olması
istenir.

Sonuç Çıkarım Mekanizması, yeni gerçeklere ulaşmak için bilgi tabanı kullanıcı tarafından kendisine sağlanan bilgiyi kullanır. İşlevi ise ,bir uzmanın çıkarımcı düşünce proseslerinin bir nevi simülasyonu olarak kabul edilebilir. Sistem tarafından üretilen yeni gerçekler sistem kullanıcısına tavsiye edici olmalıdır.

Bu basit uzman sistemimizi genişletmek mümkündür. Anlamayı kolaylaştırmak için genişletmeyi adım adım gerçekleştireceğiz. En yaygın olan bir genişletme ,bilgi tabanının;

- bir bilgi veri- tabanı ve
- bir spesifik veri-tabanı olmak üzere iki bölüm altında genişletilmesidir. Bu iki veri tabanı bir Veri-Tabanı Yönetim sistemi (VTYS) tarafından yönetebilir. Şekil 1.8 u genişletmeyi açıklamaktadır.



Şekil 1.8. Bilgi tabanının bilgi veri tabanı ve spesifik veri tabanı olarak genişletilmesi ve VTSY'inin eklenmesi.

Bilgi veri-tabanı, özel bir konunun birleşenlerinin davranışları hakkındaki kurallardan oluşur. Bu kurallar uzman sistem tarafından kullanılabilir olacak şekilde formatlanmış olmalıdır. Birkaç kural örneği verelim ;

"An accounting item is a current asset.

it is an asset,and.

it is liquid vithin one year"

"X can drive :-

X is legal driver ,and.

X can get a car.

X is legal driver :-

X has a licence ,and.

X has a licence still good.

X has a licence :-

X is old enough ,and.

X knows how to drive ,and.

X passed the test.

X can get a car :-

X owns a car ,or

X can borrow a car ,or.

X can rent a car

Bilginin bu kural formatında gösterilmesinden dolayı ,uzman sistemlere bazen kural-bazlı sistemlerde denmektedir .

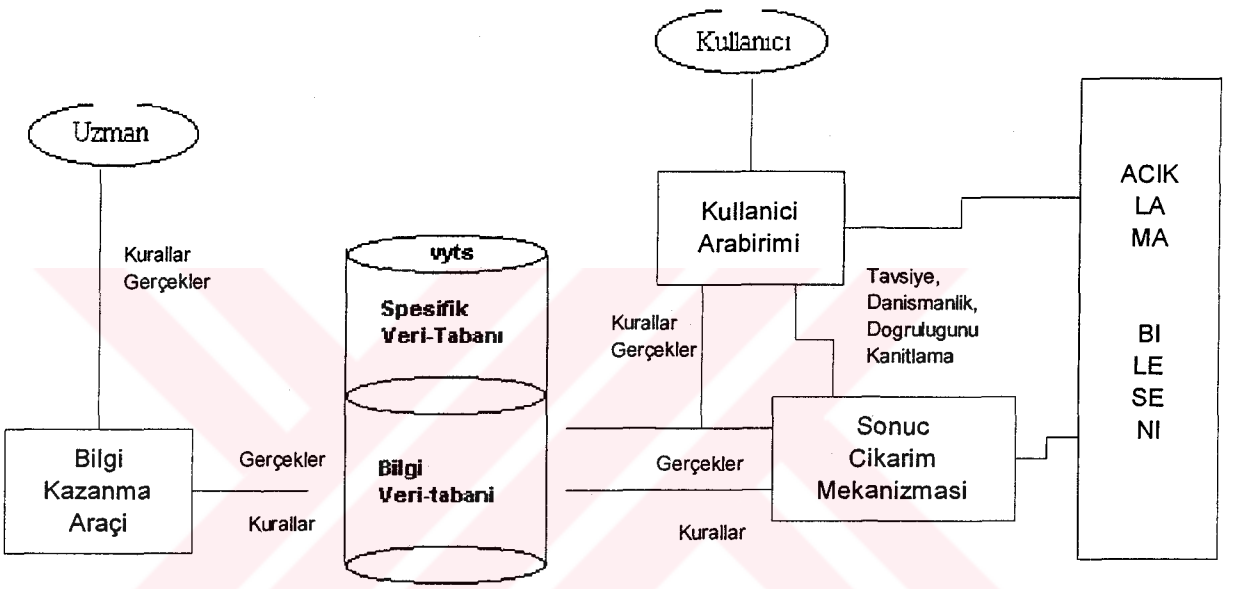
Spesifik veri- tabanı, uzman sistemin konusu hakkındaki gerçeklerden oluşmaktadır.

Çoğu uzman sistemlerdeki bilgi ,sistemin çalışma periyodu içerisinde yenilenmiş ve genişlemiş olacaktır. Bu işlem kullanıcıyı sistemin mükemmel bir hizmet sağladığı konusunda emin kılmak için gerçekleştirir .Birçok uzman sistemler bu yenileme prosesini yerine getiren bir alt birime sahiptirler. Bu birime ve alt sistem grubuna Bilgi Kazanma Aracı (BKA) -Knowledge Acquistion Facility - denilmektedir .BKA ,bilginin uzman insanlardan bilgisayar bazlı uzman sistemlere transferini kolaylaştırmak için daha çok spesifik uzman tarafından kullanılır. Şekil 1.9 eklenen BKA ile uzman sistem görülmektedir.

BKA ,bilginin kurallar ve gerçekler şeklinde elde edilmesi amacıyla uzman insanla bir diyalog ortamı sağlar. Bu alt departman daha sonra kuralları bilgi veri-tabanı ve gerçekleri de spesifik veri-tabanına yerleştirir.

Çoğu uzman sistemlerin fonksiyonlarından birisi doğruluğu kanıtlama fonksiyonudur. bu fonksiyon kullanıcılara, sistemin ürettiği cevap veya tavsiyenin doğruluğunu kanıtlamasını uzman sistemden isteme imkanını sağlar. Uzman sistemler böyle bir cevaba veya tavsiyeye nasıl ulaştığını açıklamak suretiyle doğruluğunu kanıtlar. Bu çıkarımı

açıklama fonksiyonu önemli olduğundan, bazı uzman sistem tasarımcıları sistemin yapısında ayrı bir açıklama bileşeni geliştirmişlerdir. Bu alt sistem şekil 1.9'da ilave edilmiştir. Uzman sistemle etkileşim halinde iken, istenildiği zaman, kullanıcı sistemden ulaştığı sunuca nasıl vardığını sorabilir. Açıklama aracı çok çabuk ve iyi formatlanmış bir açıklamayla cevap verecektir.

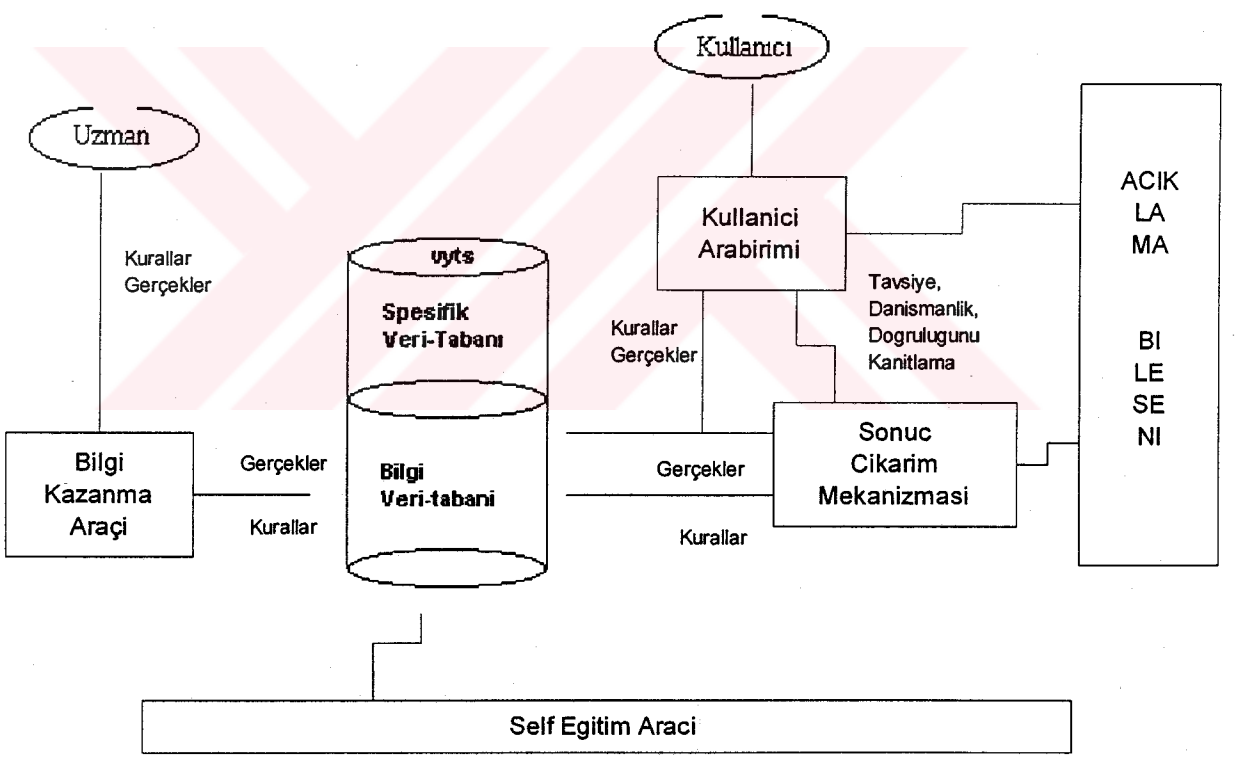


Şekil 1.9 Uzman sistem yapısına açıklama bileşeninin eklenmesi

Uzman insanlar daima deneyimlerini yenilerler ve ilerletirler. Geliştirilen yeni tavsiye ve elde edilen yeni bilgi uzmanın deneyimini artırır. Uzman böylece deneyimini self-eğitime dönüştürmüş olur. Self-eğitim uzma sistemler için diğer bir amaçtır. Uzman sistem çıkarım prosesdüründe yeni bir gerçeği (fact) yürüttüğü zaman, bu yeni gerçek kullanıcıye

sunulduğu gibi bilgi-tabanına da eklenebilir. Şekil 1.10 bir self eğitim aracı eklenmesiyle elde edilen uzman sistem görülmektedir.

Self-eğitim aracı, uzman sistemin sonuç çıkarım mekanizması tarafından gerçekleştirilen gerçekleri kabul eder ve bu yeni yürütülen gerçeklerle spesifik veri tabanında stoklanmış gerçekleri karşılaştırır. Eğer yeni geliştirilen gerçek spesifik veri tabanında bulunmuyorsa, veri tabanına eklemek için bir aday durumundadır. Self-eğitim aracı ayrıca bu gerçeğin eldeki problem için spesifik mi yoksa uzman sistemin konusu için genel olup olmadığını öğrenmeye çalışacaktır.



Şekil 1.10 Uzman sisteme self-eğitim aracının eklenmesi

1.7 Uzman Sistemlerin Uygulanabildiği Problem Tipleri

Uzman sistemler, birçok farklı problem tiplerini çözmek için geliştirilmişlerdir. Uzman sistemler, problem çözmede farklı yaklaşımlar kullanırlar. Bilgi, birçok farklı yöntemlerle gösterilebilir. Çıkarım için birçok farklı yaklaşımlar ve bir uzmanın faaliyetlerini sunmanın birçok farklı metotları vardır.

Uzman sistemlerin tablo 1.5'de gösterilen temel faaliyetleri tanımlamak kolay olmasına rağmen, bunları kullanarak mevcut uzman sistemleri kategorilere ayırmak yanlış olabilir. Çünkü birçok uzman sistem birden fazla faaliyeti verebilir. Sonuç olarak, araştırmacılar uzman sistemleri çözdükleri problem tipleriyle kategorilere ayırmayı daha uygun bulmuşlardır. Watermanın uzman sistem kataloğunda bahsettiği problem alanlarından bazıları tablo 1.6'da verilmiştir

Tablo 1.5 Problem çözme stratejisinin genel kategorisi

KATEGORİ	PROBLEM TANIMLARI
Yorum	Algılayıcı verilerden durum tanımlamalarına ulaşm
Tahmin	Verilen durumların olabilecek sonuçlarına ulaşmak
Teşhis	Gözlemlerden sistemin çalışmasındaki hataları ortaya çıkarmak
Dizayn	Nesnenin konfigürasyonu
Planlama	Faaliyetin dizayn edilmesi
Gözlem	Beklenen sonuçlar için gözlemlerin karşılaştırılmas
Hata ayıklama	Hatalar için çarelerin bulunması
Tamir	Belirlenen çarelerin yürütümü için planların yürürlüğe konması
Talimat, Eğitim	Öğrenci davranışlarının teşhisi
Kontrol	Bütün sistem davranışının kontrolü

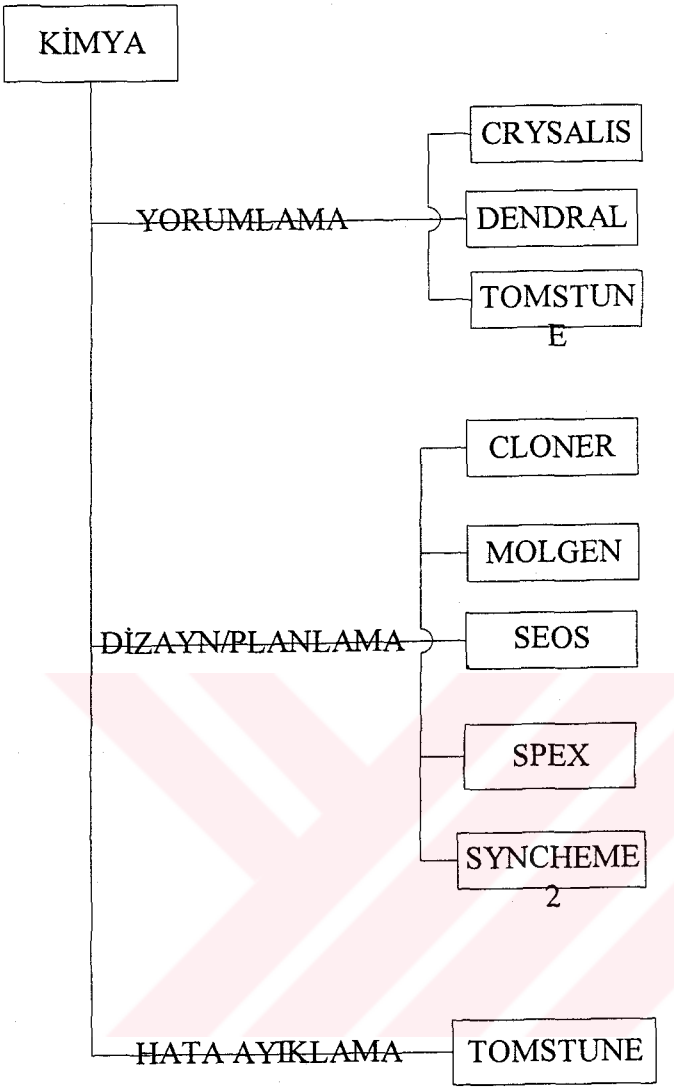
Tablo 1.6 Uzman sistemler için uygulama alanları

- Uzay teknolojisi
- Proses kontrol
- Fizik
- Askeri bilimler
- Meteoroloji
- Tıp
- Matematik
- Üretim
- Hukuk
- Bilgi yönetimi
- Tarım
- Kimya
- Bilgisayar sistemleri
- Elektronik
- Mühendislik
- Jeoloji

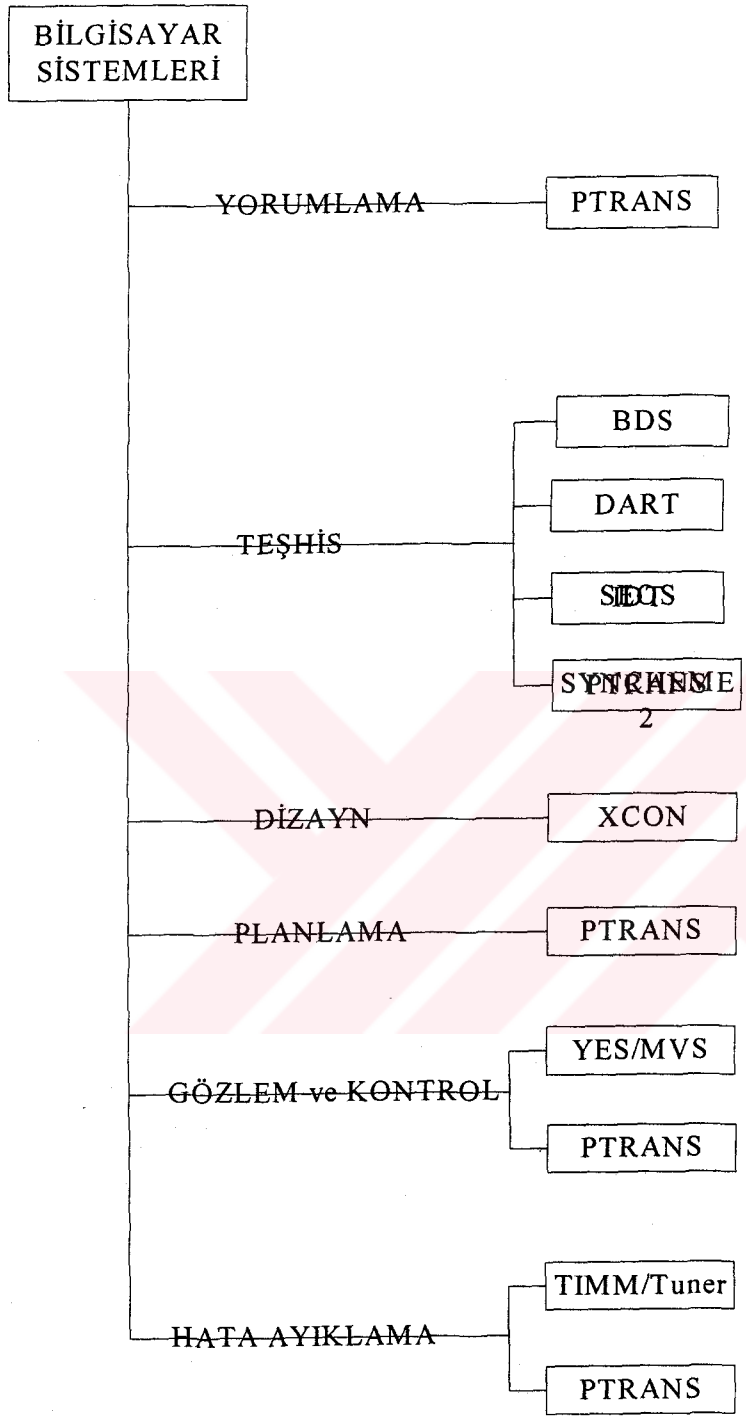
Bu problem alanlarından tıp alanı en popüler olanıdır. Şekil 1.11'den 1.15'e kadar, tablo 2.6'da en aktif beş alandaki seçilmiş uzman sistemleri kısaca tanımlar. Bu alanlar:

- **Kimya**
- **Bilgisayar sistemleri**
- **Elektronik**
- **Tıp**

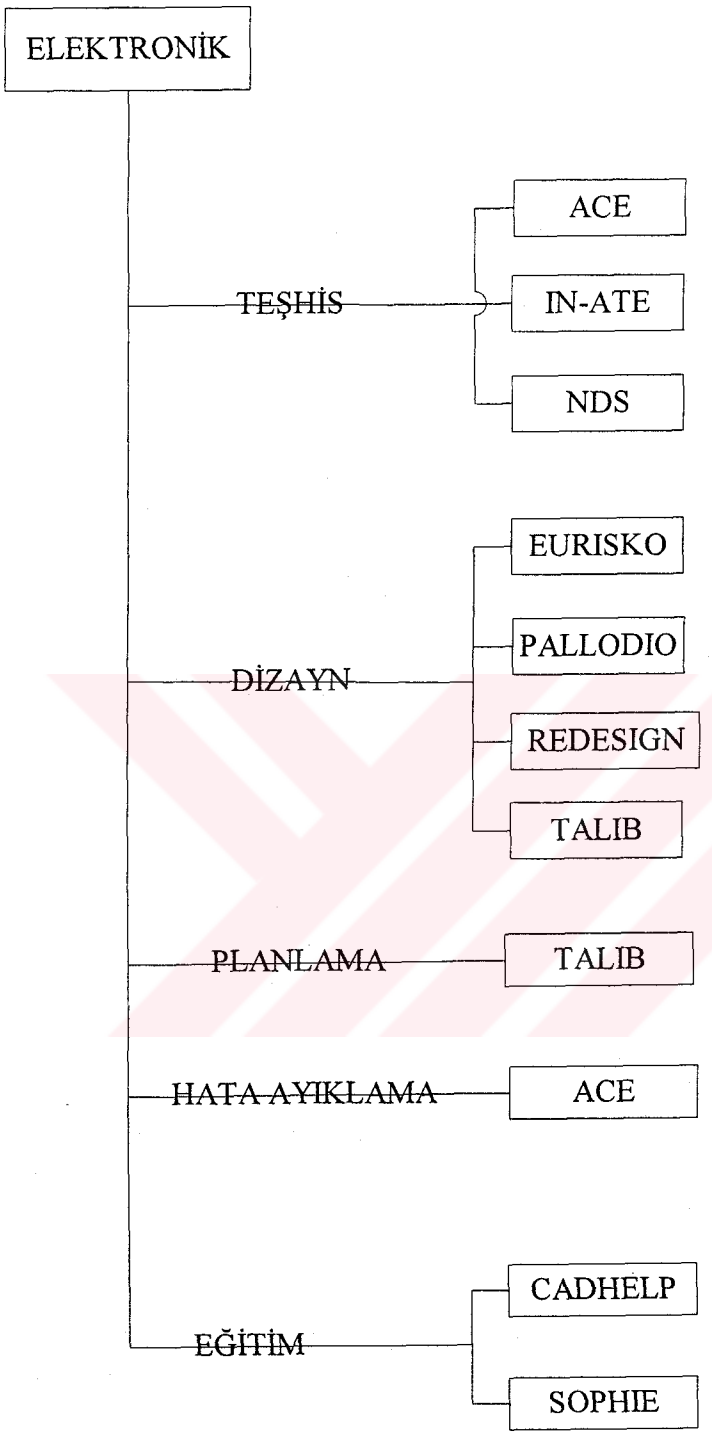
Askeri bilimle



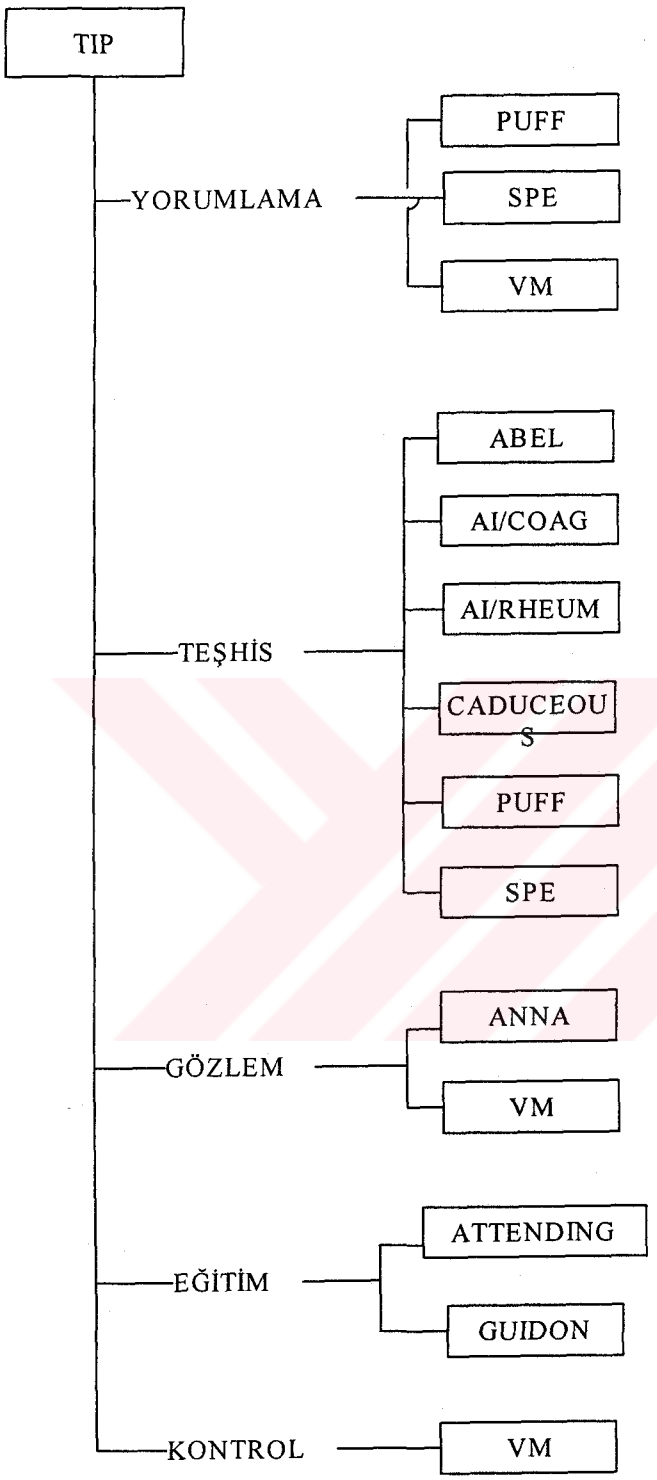
Şekil 1.11 Kimya alanından seçilmiş bazı uzman sistemler



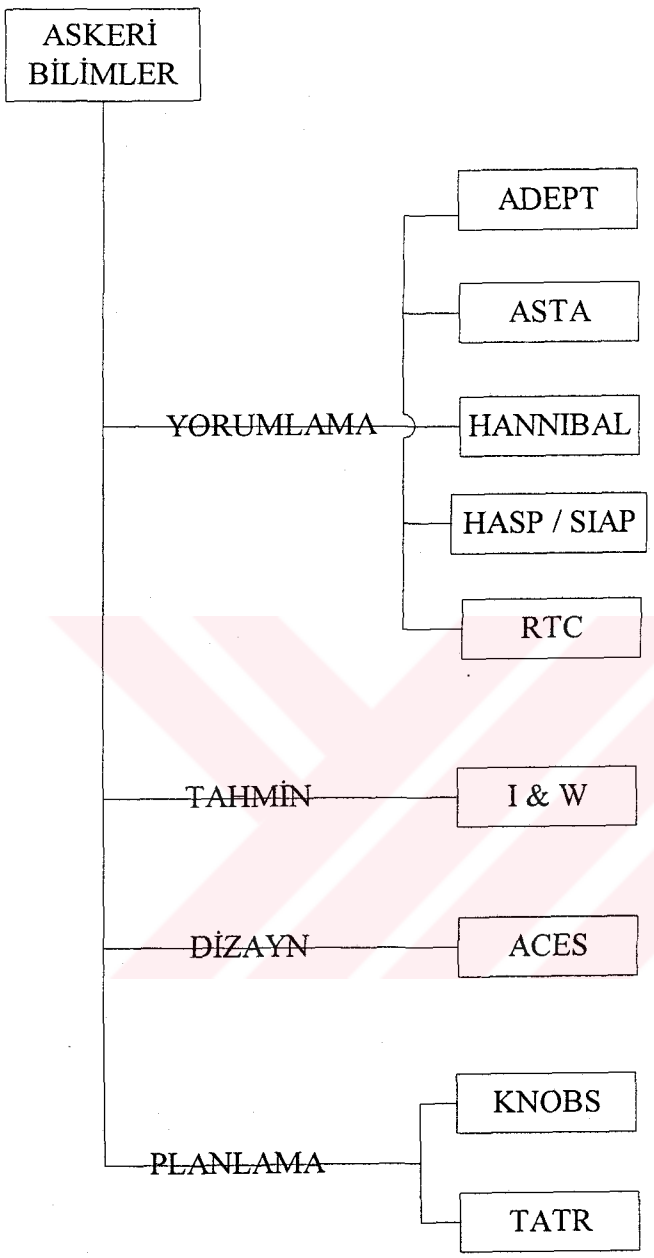
Şekil 1.12 Bilgisayar alanından seçilmiş bazı uzman sistemler



Şekil 1.13 Elektronik alanından seçilmiş bazı uzman sistemler



Şekil 1.14 Tıp alanında seçimli uzman sistemler



Şekil 1.15 Askeri bilimler alanından seçilmiş bazı uzman sistemler

BÖLÜM 2
BİLGİ ORGANİZASYONU VE GÖSTERİMİ

2.1.GİRİŞ

Bilgi kuvvettir. Bu tabir bilginin uzman için ne kadar önemli olduğunu göstermek için kullanılmıştır. Uzaman sistem performansının verilen bir problemin ne kadar sağlıklı bilgiye sahip olup olmadığıyla direkt olarak ilgili olduğu kabul edilmiştir (Durkin J., 1994).

Ama bilgi nedir? Bilgi, kişinin verilen bir konuyu anlama olayını izah eden soyut bir terimdir. Örneğin, tıp alanını anlama gibi. Bununla birlikte, uzman sistem oluştururken o konudaki uzmanın bilgisinin hepsinin alınmasına gerek yoktur. Daha ziyade konu hakkında yoğunlaşmış bilgilerin alınması hedeflenir. Bu bilgi birikimine uzman sistem dünyasında ihtisas(saha) bilgisi denir.

Başarılı bir uzman sistem geliştirilmesi yapılmak isteniyorsa, ihtisas (saha) bilgileri üzerinde yoğunlaşmak gerekir. Eğer konu genişse ve birçok başlık altında bilgi gerektiriyorsa, uzman sistemin performansı düşer.

Uzmandan ihtisas alanı hakkında bilgi aldıktan sonra, bunları uzman sistem için kodlamak gerekecektir. Bunu gerçekleştirebilmek ancak, uzmanın izlediği yola benzer bir şekilde problemin sistem tarafından çözebilmesini sağlamak amacıyla bilgiyi sistemde yapılandırmanın yolunu bulmak gerekir. Bilgi gösterimi olarak nitelendirilen bu olay bu bölümün konusunu teşkil etmektedir.

2.2.BİLGİ TİPLERİ

Bilim adamları, insanların problemleri nasıl çözdüğünü açıklamak amacıyla bir takım teoriler üretmişlerdir. Bu çalışma insanların kullana geldikleri genel bilgi tiplerini, onların bilgiyi akılda nasıl organize ettiğini ve bu bilgileri problemi çözerken nasıl kullandığını ortaya çıkarmıştır. Yapay zeka konusunda çalışan araştırmacılar bu çalışmanın sonuçlarını değişik bilgi tiplerini bilgisayarda en iyi şekilde temsil edecek tekniği geliştirmede kullanmışlardır (Duda R.O., 1981).

İnsanın bilgi organizasyonunu açıklayan tek bir teori ya da veriyi klasik bilgisayar programında yapılandırma için en iyi bir teknik olmadığı gibi, tek bir bilgi yapısı da ideal değildir. Bilgi mühendisinde en öneli sorumluluklarından biri de, verilen uygulama için en iyi bilgi temsil tekniğini seçmektir. Bunu yapabilmek için de değişik bilgi temsil tekniklerini ve bunların hangi tip bilgiyi en iyi temsil edeceklerini bilmek lazımdır. Şekil 2-1'de bilginin değişik tipleri görülmektedir.

Prosedürel bilgi, bir problemin nasıl çözüleceğini tarif eder. Bu tip bilgi bir şeyin nasıl yapılacağı konusunda yön gösterir. Kurallar, stratejiler ve prosedürler uzman sistem de kullanılan tipik prosedürel bilgi çeşitleri arasındadır.

Deklaratif bilgi, bir problem hakkında bilinenleri tarifler. Bunların arasında doğru ya da yanlış olarak da ifade edilen bilgiler olduğu gibi bir nesne ya da kavramı tamamen tarif eden yapılar da vardır.

Meta-bilgi, bilgi hakkında bilgiyi tarif eder. Bu tip bilgi problemi çözmek için gerekli diğer bilgileri almak için kullanılır. Uzmanlar bu bilgi çeşidini problemi çözme verimliliğini artırmak amacıyla kullanırlar.

Horisrik bilgi, genelde sığ bilgi olarak ta nitelendirilir ve prosesleri sonuçlandırmada yön gösteren bilgilerdir. Bu bilgiler tecrübe sonucunda elde edilmiştir ve bir uzmanın geçmiş problemleri çözmesi sırasında elde ettiği bilgilerin derlenmiş bir şekli diyebiliriz. Uzmanlar genelde problem hakkında derin bilgi diye nitelendirilen temel bilgileri edineceklerdir, (temel kurallar,fonksiyonel ilişkiler gibi) ve bunları problem çözmede yardımcı olabilmek için basit horistiklerle derlerler.

Yapısal bilgi, bilgi yapılarını tarif eder. Bu bilgi tipi, bir uzmanın probleminin zihni modelini tam olarak tarif eder. Tipik olarak bu tür bilgiler uzmanın kavramlar,alt kavramlar ve nesneler hakkındaki zihni modelini yansıtırlar.

BİLGİ TİPLERİ	
PROSEDUREL BİLGİ	KURALLAR STRATEJİLER PROSEDÜRLER
DEKLERATİF BİLGİ	KAVRAMLAR NESNELER GERÇEKLER
META-BİLGİ	DİĞER TİP BİLGİLER VE ONLARIN KULLANILDIĞI HAKKINDA BİLGİ
HORİSTİK BİLGİ	RULES OF THUMB
YAPISAL BİLGİ	KURAL GRUPLARI KAVRAM İLİŞKİLERİ

ŞEKİL 2-1. Değişik bilgi tipleri.

2.3.BİLGİ GÖSTERİM TEKNİKLERİ

Yapay zeka konusundaki araştırmacıların çalışmaları sonucunda, bilginin bilgisayarda temsil edilmesinin etkin yolları geliştirilmiştir (Swift K.G., 1987). Bu bölümde, uzman sistemlerin geliştirilmesi sırasında kullanılan çok genel 5 teknik incelenecektir.

- Nesne-Sıfat-Değer üçlüsü (O-A-V triplets)
- Kurallar (rules)
- Semantik şebekeler (semantik network)
- Çerçeveler (frames)
- Lojik ifadeler

2.3.1.NESNE-SIFAT-DEĞER ÜÇLÜSÜ

Öğrenme teorilerine göre, insanlar, bilgi organizasyonunda temel yapı blokları olarak gerçekleri (fact) kullanırlar. Gerçek dekleratif bilginin bir biçimidir. Olayı ya da problemi anlamayı sağlar.

Uzman sistemlerde, gerçekler çerçeveleri kısımlarını, semantik şebekelerin ve kuralların tariflenmesinde yardımcı olmak amacıyla kullanılırlar. Gerçekler aynı zamanda çok kompleks bilgi yapıları arasındaki ilişkiyi tarif etmek için ve bu yapıları problem çözme sırasında kontrol etmek amacıyla da kullanılırlar. Yapay zekada ve uzman sistemlerde bir gerçek genelde bir önerme ile ilgilidir ya da gösterilir.

Tanım: Önerme

doğru ya da yanlış olan ifade.

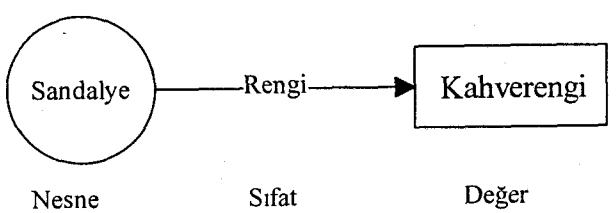
Önerme aşağıdaki gibi çok basit bir ifade de olabilir.

Önerme: Hava yağmurlu.

Uzman sistemlerde çalışan hafızaya ifadenin Boolean doğruluk değeri işlenir (doğru ya da yanlış) ve bu değer diğer bilgilerin prosesi sırasında da kullanılır.

Bir gerçek aynı zamanda bazı nesnelerin özel bir takım niteliklerini ifade etmek için de kullanılır. Örneğin "Topun rengi kırmızıdır" ifadesi kırmızı değerini topun rengine atar. Bu tip gerçekler Nesne-Sıfat-Değer üçlüsü (O-A-V triplets) olarak bilinir.

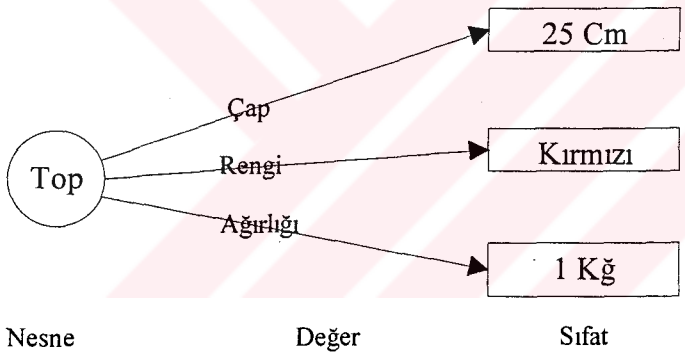
Nesne-Sıfat-Değer üçlüsü (O-A-V triplets) önermenin çok kompleks şeklidir. Verilen bir ifadeyi Nesne,Sıfat.Değer olmak üzere üç ayrı parçaya böler. "Sandalyenin rengi kahverengidir." ifadesini göz önüne alalım. Bu ifadeyi Nesne-Sıfat-Değer üçlüsü yapısında ifade etmek gerekirse, burada koltuk nesnedir, renk sıfattır, kahverengi ise değerdir. Bu özellik şekil 2.2'de gösterilmiştir.



Şekil 2.2 Nesne-sıfat-değer.

Nesne-Sıfat-Değer üçlüsünde gösterilen nesne araba yada masa gibi fiziksel nesneler olabileceği gibi, sevgi,üzüntü, gibi soyut kavramlar da olabilir. Sıfat ise problemin çözümünde önemli bir unsur olup nesnenin özelliğini bildirir. Nesnenin sıfatına atanan ise değerdir. Bu değer Boolean,numerik ya da dizi olabilir.

Uzman sistemlerin ele aldığı problemlerin çoğunda nesneler birden fazla önemli özelliğe sahiptirler. Bu tip örneklerde nesne için çoklu sıfat ve bunlara karşılık gelen değerler tariflenir. Şekil 2-3'de çoklu sıfat Nesne-Sıfat-Değer üçlüsü yapısına bir örnek görülmektedir. Bu bölümün sonunda semantik semantik şebekelerin ve çerçevelerinde nesneleri daha iyi tarif edebilmek için çoklu sıfat özelliğini kullandığını görülecektir.



Şekil 2.3 Çok değerli sıfatlar.

Tek Ve Çoklu Değerli Gerçekler

Bazı sıfatlar lojik olarak sadece bir değere sahip olduğu gibi, diğerleri de doğal olarak çoklu değerlere sahiptirler. Bunlara genelde tek değerli ve çoklu değerli gerçekler denir. Uzman sistem dizaynı süresince tek yad çoklu değerli N-S-D sıfatı tahsis edilmelidir. Bu da dızayncıya, problemin bilgisinin gösteriminde ek bir fleksibilite kazandırır.

Barometrenin basıncının okunması hakkında bir ifadenin gösterimini Nesne-Sıfat-Değer üçlüsünü kullanarak yaptığımızı göz önüne alalım. Burada nesne " barometre", sıfat ise "basınç okuma"dır. Sıfatın alabileceği değerler ise düşüyor,sabit ya da yükseliyor dur.

Şayet uzman sistemin barometre basıncı hakkında bilgiye ihtiyacı olursa aşağıdaki soruları soracaktır.

S: Lütfen baro metrik basınç hakkında bilgi veriniz,

- Düşüyor.
- Sabit.
- Yükseliyor.

C: Düşüyor.

Mantıken baro metrik basınç yukarıdaki üç değerden birini alabilir. Bu durumda tek değerli NSD'yi kullanmanız gerekir.

İnsan eğitimi hakkında bir örneği göz önüne alacak olursak; nesne "kişi"dir ve sıfat "eğitim seviyesi"dir. Bununla birlikte bir kişi birden fazla eğitim derecesine sahip olabilir (lise,üniversite gibi). Bu durumda çoklu değerli NSD kullanmak gerekir.

Örneğin;

S: Lütfen eğitim seviyelerini seçiniz.

- Yüksek okul.
- Lise.
- Üniversite.

C: Lise

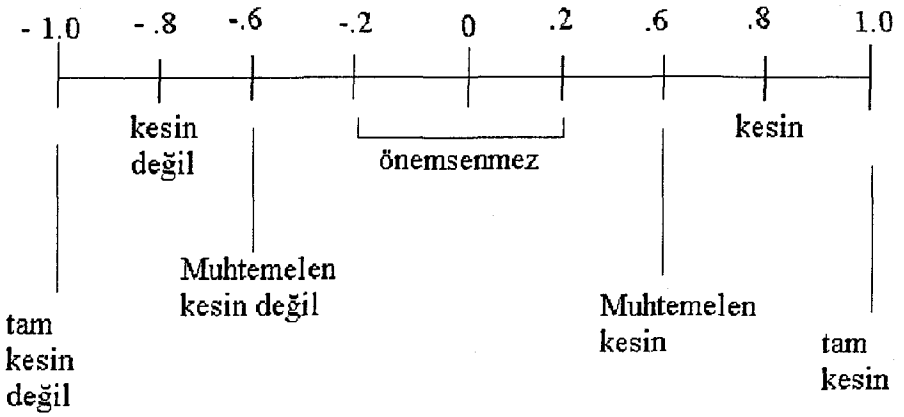
Üniversite.

Hem tek hem de çoklu değerli Nesne-Sıfat-Değer üçlüsü gerçekleri önemli bir özelliği paylaşırlar. Kullanıcı listeden bir değer seçtiği zaman, sistem bu değeri çalışan hafızaya **DOĞRU** olarak ve diğer tüm değerleri **YANLIŞ** olarak girer. baro metrik basınç sorusunu göz önüne alalım. Şayet kullanıcı "düşüyor" şıkkını seçtiği zaman sistem sadece bu gerçeği bilmiyor aynı zamanda basıncı **yükselmediğini** ve **sabit kalmadığını** da biliyor.

Kesin Olmayan Gerçekler

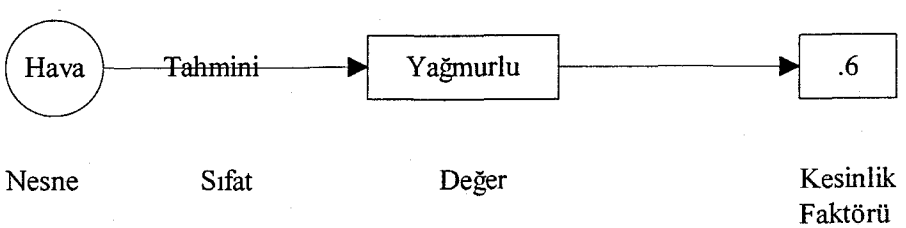
Dünyamızda sadece siyahlar ve beyazlar yoktur. Bu yüzden, her zaman bir olayın doğru mu ya da yanlış mı olduğunu tam bir kesinlikle bilemeyebiliriz. Daha ziyade olay hakkında bir dereceye kadar inancımız vardır. Bu özellik günlük konuşmalarda "olabilir","belki" gibi kelimelerle ifadesini bulur. Örneğin birisi "Bugün muhtemelen yağış var." Diyebilir. İnsanlar böyle ifadeleri yorumlamakta az da olsa güçlük çekerler. Uzman sistemlere ise bu tür ifadeleri yorumlama kabiliyeti kazandırılmıştır.

Uzman sistemlerde kesin olmayan bilgileri yorumlayabilmek için kullanılan klasik bir metot **Kesinlik Faktörüdür**. Kesinlik Faktörü (KF) ifadeye inanma derecesini gösteren nümerik bir değerdir. Şekil 2-4'de ifadenin nitel tarifinde bu değerin nasıl yerleştiğini gösterilmektedir. Kesinlik Faktörü MYCIN üzerindeki çalışmalar sırasında ortaya çıkmıştır ve kesinlik teorisi olarak formülize edilmiştir (Shortliffe 1975).



Şekil 2-4 MYCIN'da kullanılan kesinlik faktörleri.

Şekil 2.4'de görüldüğü gibi Kesinlik Faktörü -1 ile +1 değerleri arasında değişmekte olup, -1 ifadenin kesinlikle yanlış olduğunu +1 ise kesinlikle doğru olduğun göstermektedir. Bu iki uç noktanın arasındaki değerler ise ifadeye inanmanın ya da inanmamanın derecesini göstermektedir. Örneğin bir önceki örneği ele alacak olursak "Bugün muhtemelen yağış var." İfadesine kesinlik faktörü olarak 0.6'yı atayabiliriz. (Bkz Şekil 2.5).



Şekil 2.5 Hava tahmini kesinlik faktörü

Fazi Gerçekler

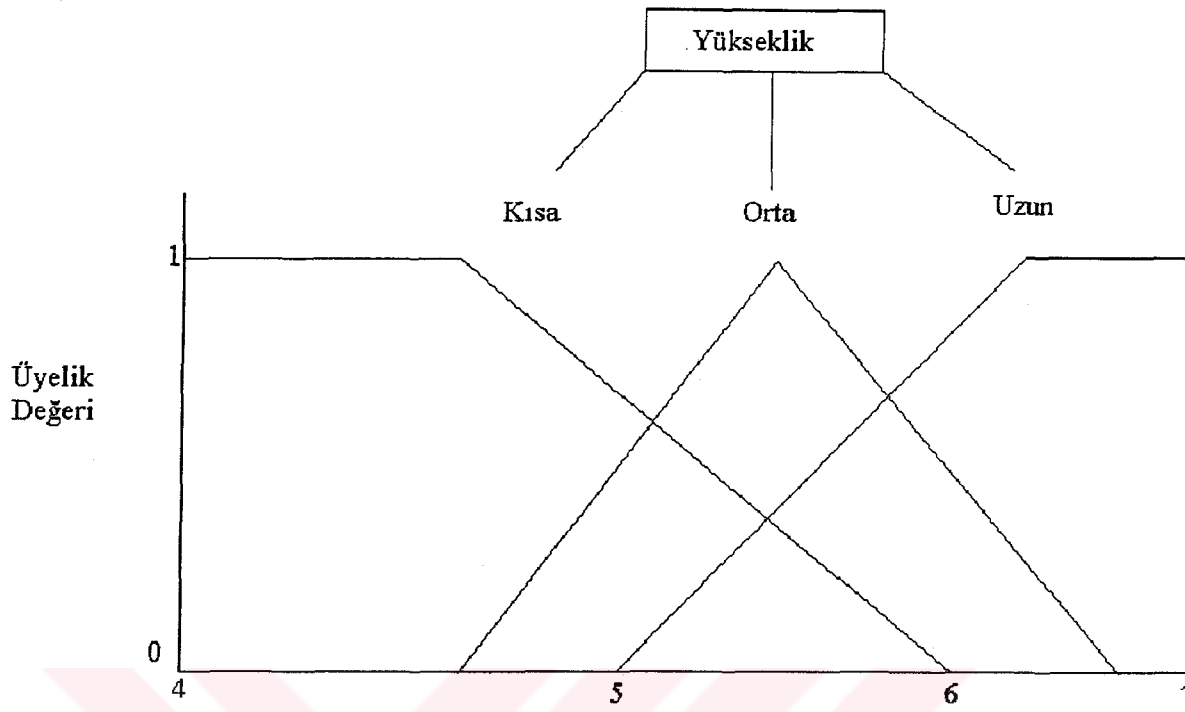
Kesin olmayan ifadelerin uzman sistem dünyasına girmesinin sebebi, doğal dilde bulunan belirsiz ifadelerin gösterim gerekliliğinden kaynaklanmaktadır. Örneğin, "Ali uzun boyludur." İfadesini göz önüne alalım. İfade, *uzun boylu* kelimesinin kullanılmasından dolayı belirsizdir. İnsanlar bu tür belirsiz terimleri yorumlamak ve sonuçlandırmakta güçlük çekerler. Bununla birlikte, bilgisayarında bu konuda yardıma ihtiyacı vardır.

Fazi lojik diye bilinen çalışma alanında yardım mümkündür (Zadeh 1965). Fazi lojik, belirsiz ifadelerle gösterim ve sonuçlandırma imkanını sağlamaktadır. Belirsiz ifadeler fazi gruplar halinde gösterilirler. Örneğin şekil 2.6'da bir insanın boy uzunluğunu tarif den fazi gruplar halinde değişik sıfatlar görülmektedir.

Bu şekilde insan boyunu, üyelik değeri (derecesi) adı verilen bir sayı ile sınıflandıran üç fazi grup görülmektedir. Üyelik değer sıfır ile bir arasında bir değer olup, bir fazi gruba ait verilen bir yüksekliğe olan inanma derecesini yansıtmaktadır. Örneğin 5.5 feet uzunluğundaki bir kişi 1 üyelik değeri ile orta uzunlukta insanların grubuna dahil edilir ve aynı zamanda kısa ve uzun insanlar grubuna 0.25 değerlerle dahil edilir.

Fazi gruplar oluşturmaya ek olarak fazi lojik fazi kurallar yazmaya da izin verilir. Bir fazi kural hem IF hem de THEN grubunu içerir. Aşağıdaki örneği göz önüne alalım;
IF kişinin boyu uzunsa,
THEN kişinin kilosu fazladır.

Bu kuralda yapılan çıkarımlar sonucunda, kişinin kilosu kısmında ağır ibaresi belirecektir.



Şekil 2.6 Boy uzunluğu üzerine fazi gruplar

2.3.2 KURALLAR

Kullanıcı tarafından sağlanan gerçekler bir uzman sistemin çalışmasında çok öemlidir. Bu bilgiler sayesinde sistem dünyanın hali hazırdaki durumunu anlar. Bununla birlikte sistemin, verilen bir problemi bu gerçekler dahilinde akıllıca çözebilmesini sağlayan ek bilgiye sahib olması gerekir. Uzman sistemin dizaynında genellikle kullanılan ve bu ek bilgiyi sisteme kazandıran yapıya kural denmektedir.

Kurallar prosedürel bilginin bir şekildir. Verilen bir bilgi ile bazı fiilleri/işleri birleştirir. Bu fiil, icra edilmesi gereken bir prosedür ya da çalışan hafızaya eklenen bir bilgi olabilir. Bu anlamda bir kural problemin nasıl çözüleceğini tarif eder.

Kuralın yapısal lojik olarak, IF kısmında bulunan bir ya da daha fazla şartı THEN kısmındaki sonuçlara birleştirir.
Örneğin;

IF arabanın rengi kırmızı ise,
THEN arabayı satın alırım.

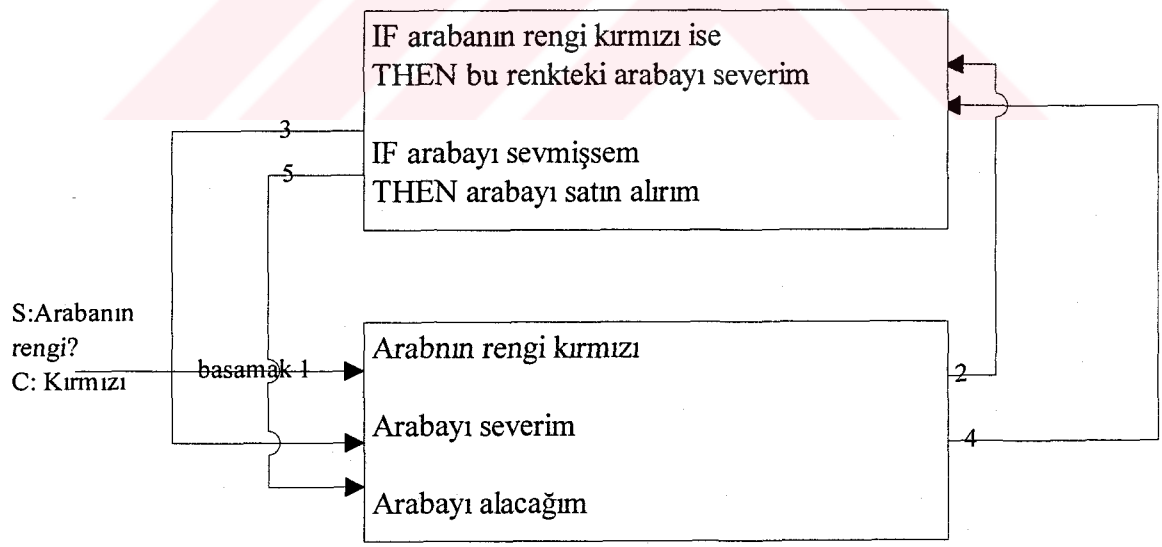
Bu basit örnekte verilen bir arabanın rengi kırmızı ise kural arabayı alırım sonucunu çıkarır.

Genelde, bir kural AND, OR ya da ikisinin kombinasyonu ile şartları birleştirerek çoklu şarta sahip olabilirler. Bu ifadenin sonucu tek bir ifade olabileceği gibi AND ile birleştirerek bir kombinasyonu da olabilir. Kural aynı zamanda şartların bir ya da bir kaçı YANLIŞ, olduğu zaman doru olacak ELSE ifadesini de içerebilir. Aşağıdaki örnekte genel kural yapısını görebiliriz.

IF saat 10'u geçmisse,
AND bugün haftasonu değilse,
AND evdeysem,
OR patronum beni arayıp işe geç kaldığımı söylediyse,
THEN ben işe geç kalmışımdır,
ELSE işe geç kalmamışımdır.

Kural-tabanlı uzman sistemde, uzmanlık bilgileri kurallar grubu içinde tutulur ve sistemin bilgi tabanına girilir. Sistem bu kuralları çalışan hafızadaki bilgileri kullanarak problemi çözer. Kuralın IF kısmındaki bilgi çalışan hafızadaki bilgi ile dengelenmişse, sistem kuralın THEN kısmındaki fiili gerçekleştirir. Bu olay gerçekleştiği zaman, kural ateşlenir ve THEN kısmı çalışan hafızaya eklenir. Yeni ifadenin çalılan hafızaya eklenmesi diğer bazı kurallarında ateşlenmesine sebebiyet verebilir. Şekil 2.7 bu prosesin bir örneğini göstermektedir.

Proses, sistemin kullanıcıya arabanın rengini sormasıyla başlar. Sistem "Kırmızı" cevabını alır ve bu gerçeği çalışan hafızaya ekler (basamak 1). Bu girilen gerçek ile birinci kuralı şartı uyar (basamak 2). Bu karşılaştırma kuralın sonucunun ateşlenmesine sebep olur ve "Arabayı severim" çalışan hafızaya eklenir (basamak 3). Bu yeni bilgi ikinci kuralın şartına uyar (basamak 4). Bu da kuralın ateşlenmesine sebep olur ve "Arabayı alacağım" gerçeği çalışan hafızaya eklenir (basamak 5). Bu noktada başka hiçbir kural göz önüne alınmaz böylelikle proses işlemi biter.



Şekil 2.7 Kural tabanlı operasyon

PROSEDÜRÜ ÇALIŞTIRMAK

Bir kural, yeni sonuçlara varmanın yanında, başka operasyonlar da gerçekleştirebilir. Bu işlem aşağıda gösterileceği gibi basit bir hesaplama da olabilir.

IF dikdörtgenin alanı gerekli ise

Alan = Uzunluk * Genişlik

Bu kural, şartında gösterilen bilgi, çalışan hafızaya eklendiği zaman ateşlenir.

Daha kompleks operasyonlar gerçekleştirebilmek amacıyla, çoğu kural tabanlı sistemler harici bir programa ulaşacak şekilde dizayn edilmiştir. Bu program, , veri tabanı, C programlama dili gibi, klasik tip yazılım olabilir. Örnek olarak aşağıdaki kuralları göz önüne alalım. Bu kurallar hiç bir dilde koda dökülmemiştir, sadece açıklama amacıyla sunulmaktadır.

1.Prosedürel Programlama

Kural 1

IF dizayn yeni bir kutuyu gerektiriyorsa,
AND ADET = paketlenecek birimlerin adeti ise
AND ÖLÇÜ = birimlerin ölçüsü ise

COMPUTE_BOX_VOLUME'ı çağır

AND ADET ve ÖLÇÜ'yü gönder,
AND HACİM değerini al.

2.Tablolama
Kural 2
IF OCAK SATIŞLARI gerekli ise
SATIŞLAR'ı aç
AND OCAK_SATIŞLARI = B7

3. VERİTABANI
IF Fabrika tesisatında acil bir durum varsa
AND NAME = Smith

TELEPHONE aç
AND NAME,NAME_FIELD'ı bul
AND TELEPHONE = TELEPHONE_FIELD

KURAL 1, dizayn işlemi sırasında yeni bir kutu gerekli olduğu zaman kutunun hacmini tayin eder. Kural, kutu içinde paketlenecek birimlerin ölçüsü ve adedi verildiği durumda, kutunun hacmini hesaplayan ve çıkış değeri olarak bunu veren COMPUTE_BOX_VOLUME programını çalıştırır. Bu program herhangi bir klasik programlama dilinde yazılabilir.

KURAL 2, ocak ayı satış miktarlarını tayin eder. Kural bu bilgiyi satışlar tablosunu B7 hücresinden alır.

KURAL 3, saha amiri Bay Smith'in telefon numarasını acil bir durum ortaya çıktığı zaman bulur. Bu bilgileri TELEPHONE veri tabanında NAME_FIELD'de Smith ismini yerleştirerek ve burada da TELEPHONE değişkenini atayarak TELEPHONE_FIELD'de bulur.

Genelde harici programlarda ki bilgilere ulaşılabilir ya da değiştirilebilirdir. Bu özellik uzman sistem dizaynı sırasında esnekliği artırır. Hali hazırda mevcut veri tabanlarındaki ve tablolardaki birçok bilgiler kullanılabilir ya da alternatif olarak uzman sistemin akıllı seçim yapması sırasındaki bilgi de değiştirilebilir.

KURAL TİPLERİ

Aşağıdaki listede anlatılacağı üzere kuralları değişik tipteki bilgilerin gösteriminde kullanılabilir.

İlişki:

IF Akü bitmişse,
THEN Araba çalışmayacaktır.

Tavsiye:

IF Araba çalışmıyorsa,
THEN Taksi tut.

Direktif:

IF Araba çalışmıyorsa,
AND Yakıt sisteminde bozukluk yoksa,
THEN Elektrik sistemini kontrol et.

Strateji:

IF Araba çalışmıyorsa,
THEN İlk önce yakıt sistemini, daha sonra elektrik sistemini kontrol et.

Horistik:

IF Araba çalışmıyorsa,
AND Araba Ford ise,
THEN Karbüratörü kontrol et.

Kurallar aynı zamanda, genelde *problem çözme model* olarak adlandırılan, problem çözme stratejilerinin doğasına göre de katogorize edilebilir. Aşağıdaki liste, ortak modellerin bazılarında bulunan tipik kuralları göstermektedir.

Yorum Problemi:

IF Direncin gerilimi 2 Volt'dan düşük ise,
AND Q1'in kollektör gerilimi 1 Volt'dan düşük ise,
THEN Pre-Amp bölümü normal sınırlar içerisindedir.

Dizayn Problemi:

IF Şu anki yapılacak iş güç kaynağı tayin etmek ise,
AND Kabine içinde güç kaynağının yeri biliniyorsa,
AND Kabine içinde güç kaynağı için uygun yer varsa,
THEN Güç kaynağını kabine içine yerleştir.

Değişken Kurallar

Bazı uygulamalarda aynı işlemi bir grup benzer nesneler için de yapmak gerekebilir. Dizaynçı her bir nesne için ayrı bir kural yazabilme imkanına sahip olsa da, bu yaklaşım çok verimsiz olup sistemin işlemesini zorlaştırır. Serdar Yeşil isimli bir işçinin emekli olabileceği durumu sonuçlandıran aşağıdaki örneği göz önüne alalım.

IF Serdar Yeşil bir işçiyse,
AND Serdar Yeşil 65 yaşından büyükse,
THEN Serdar Yeşil emekli olabilir.

Bu kural Serdar Yeşil'in şirketin bir işçisi olup olmadığını ve 65 yaşından daha büyük olduğuna bakarak emekli olabileceği sonucuna varıyor. Eğer aynı kontrol diğer kişiler için de yapılmak isteniyorsa benzer kuralın her bir kişi için yazılması gerekmektedir. Bu çok verimsiz bir yol olmakla birlikte şayet temel bazı bilgi değişiklikleri olursa, örneğin emeklilik yaşı 65 den 60'a düşerse, gerekli değişiklikleri her bir kural için yapmak gerekir.

Bu tip problemlere maruz kalmamak için güçlü bir *örnek-eşleme* (pattern-matching) kural sunan bilgisayar dili ya da uygun bir shell kullanmak lazımdır.

Biraz önce verdiğimiz emeklilik örneğini, bu kez değişken kullanarak tekrar ele alalım.

IF ?X bir işçiyse,
AND ?X 'in yaşı > 65 ise,
THEN ?X emekli olabilir.

Bu kural çalışan hafızada iki şartın da uyduğu durumları tarar. Eğer iki şart da gerçekleşirse, kural bu kişinin emekli olabileceği sonucuna varır. Bu kuralın nasıl işlediğini görelim.

Farzedelimki çalışan hafızada aşağıdaki bilgiler mevcut.

Serdar bir işçi. Serdar'ın yaşı 67.
Haluk bir işçi. Serdar'ın yaşı 70
Savaş bir işçi. Serdar'ın yaşı 60.

Birinci şart bilgileri tarar ve kişi bir işçi ise ismini ?X değişkenine atar. İkinci şart aynı kişinin yaşının 65 den büyük olup olmadığını kontrol eder. Eğer her iki şart da yerine getirilmişse kural bu kişinin emekli olacağı sonucunu çıkarır ve bu yeni bilgiyi çalışan hafızaya ekler. yukarıda verilen bilgiler sonucunda çalışan hafızaya aşağıdaki iki bilgi eklenir.

Serdar emekli olabilir. Haluk emekli olabilir.

Kesin Olmayan Kurallar

Kesin olmayan gerçekler olduğu gibi, kesin olmayan kurallar da vardır. Örneğin;

IF Enflasyon oranı yüksekse,
THEN Aşağı yukarı faiz oranları yüksektir.

Bu durumlarda uzman sistemin birleştirme işleminde ne kadar güvenilir olduğunu göstermek için *Kesinlik Faktörü* (KF) kullanılabilir. Bir önceki kural aşağıdaki gibi yazılabilir.

IF Enflasyon oranı yüksekse,
THEN Faiz oranları yüksektir. KF = 0.8

Burada, enflasyon yükselirken faiz oranlarının yüksek olması tam bir kesinlik içermediğinden KF=0.8 atanmıştır.

Meta Kurallar

Uzmanlar bir problemi çözerken, problem çözümünde kendini yönlendirecek bilgiye ihtiyaç duyarlar. Bu bilgi saha uzmanını karakterize etmesini beklediğimiz bilgiden farklıdır. Daha ziyade, bu tip bilgiler problemin en iyi şekilde çözülebilmesi için saha konusunda hali hazırda mevcut olan bilgilerdir. Bu bilgilere meta-bilgi denir.

Meta-bilgi, konsultasyonu yönlendirmek için meta-kural'da kullanılmaktadır. Meta-kural ise diğer kuralların nasıl kullanılacağını tarif eden kurallardır. Meta-kural, yeni bilgiler içermekten ziyade, sahaya özel kuralların kullanım stratejilerini düzenler. Örneğin,

IF Araba çalışmıyorsa,
AND Elektrik sisteminde bozukluk yoksa,
THEN Yakıt sistemi ile ilgili kuralı uygula.

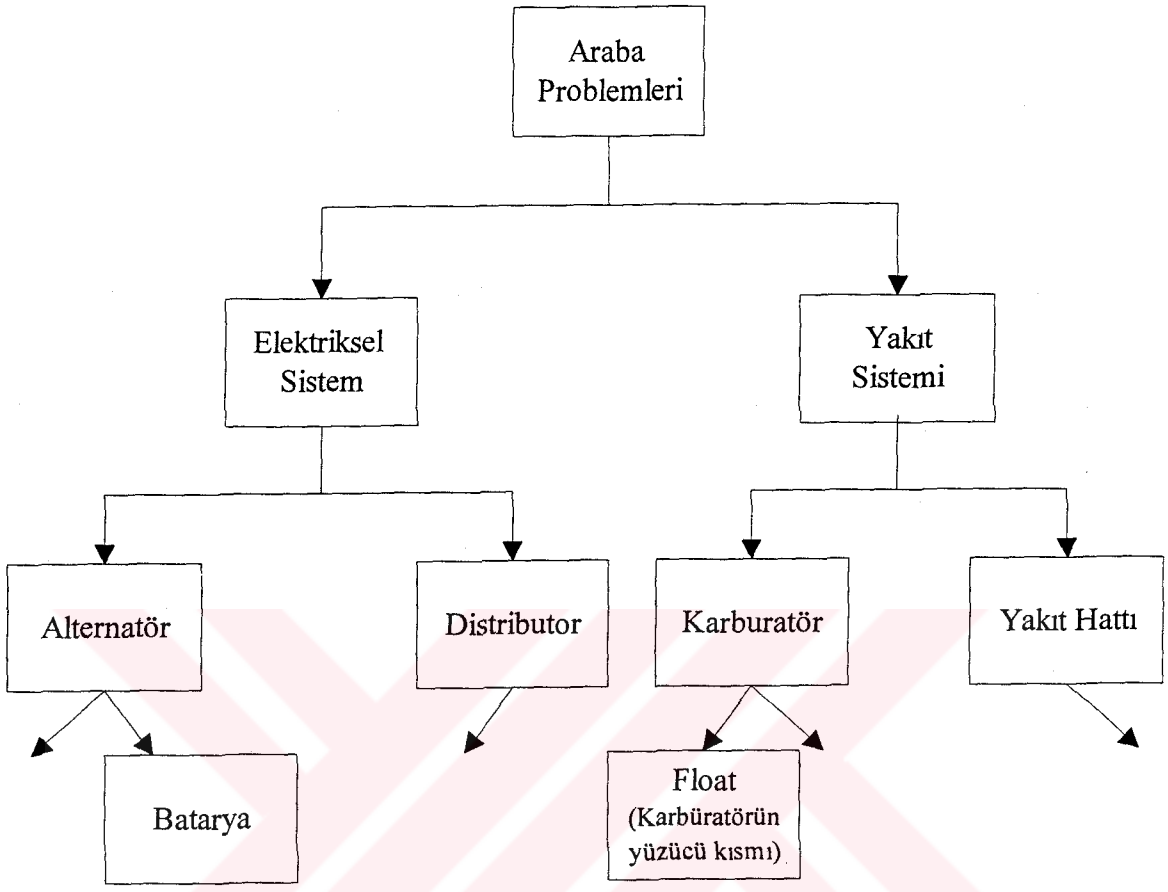
Bu otomobil arıza teşhis meta-kuralı, eğer elektrik sisteminde hiçbir arıza yoksa, sistemi yakıt sistemini kontrol etmeye yönlendirir.

Kural Grupları

Uzman, verilen bir probleme elde ettiği tecrübe ile bir çok kural gruplarını uygular. Bir grup kural verilen bir probleme uygulanabilir olurken, diğer bir problem için kullanışsız olabilir. Örneğin, bir otomobilde elektrik ve yakıt sistemi için ayrı ayrı kural grupları olabilir. Kişi elektrik ile ilgili problemle karşılaştığı zaman, geçmiş tecrübelerinden, otomobilin elektrik sistemi üzerinde çalışırken elde ettiği kuralları kullanır.

Verilen kural grubu, uzmanın verilen konudaki kabiliyetini yansıtır. Değişik kural gurupları arasında seçim yapmak ve bunları kullanmak uzmanın yeterliliğini gösterir. Kurallar tek başına uzman sonuçlandırması için yeterli değildir. Bu kuralların ne zaman ve nasıl uygulanacağı konusunda da bir strateji şarttır.

Kural gruplarının kullanımı ve organizasyonunu anlayabilmek için şekil 3.8'i göz önüne alalım. Bu şekil otomobil mekaniğinin kullanabileceği değişik kural gruplarını göstermektedir.



Şekil 2.8 Otomobil hata araştırma kural grupları.

Şekil 2.8'de her bir blok, bloğun etiketi ile ilgili kural grubunu temsil etmektedir. Üst seviyedeki bloklar genel problemin çözümü için soyut kural gruplarını içermektedir. Bu bloklar sistemin sonuçlandırma mekanizmasını, kural grupları tarafından daha detaylı tetkiklerin icra edildiği alt bloklara yönlendirirler.

En üst seviyedeki blokta "Araba problemleri" sistemi ya "elektrik sistemine" ya da "yakıt sistemini" tetkik etmeye yönlendirmektedir. Bu kural grubu, elektriksel ve yakıta ait

problemleri birbirinden ayırarak belirtiler üzerinde çalışır. Bu alt kategoriler altında "Alternatör" ya da "distribütör" gibi daha detaylı kural kategorileri mevcuttur. "Elektriksel sistem" modülündeki kurallar bu iki alt kurallardan hangisinin izleneceğini tanımlamakla sorumludurlar. En alt seviye bloklar olan "Batarya" ve "Float" çok özel test bilgileri ile çalışarak arabnın problemini teşhis eden kurallar içermektedir.

Bu modüler yapı problem çözümünde, gerektiğinde uygu kuralları kullanıldığı tepeden aşağı problem çözüm yaklaşımını gerektirmektedir.Uzman sistem, daha önce bahsedilen meta-kurallarını kullanarak, verilen kural grubunun kontrolunu yapmadan geçebilir.

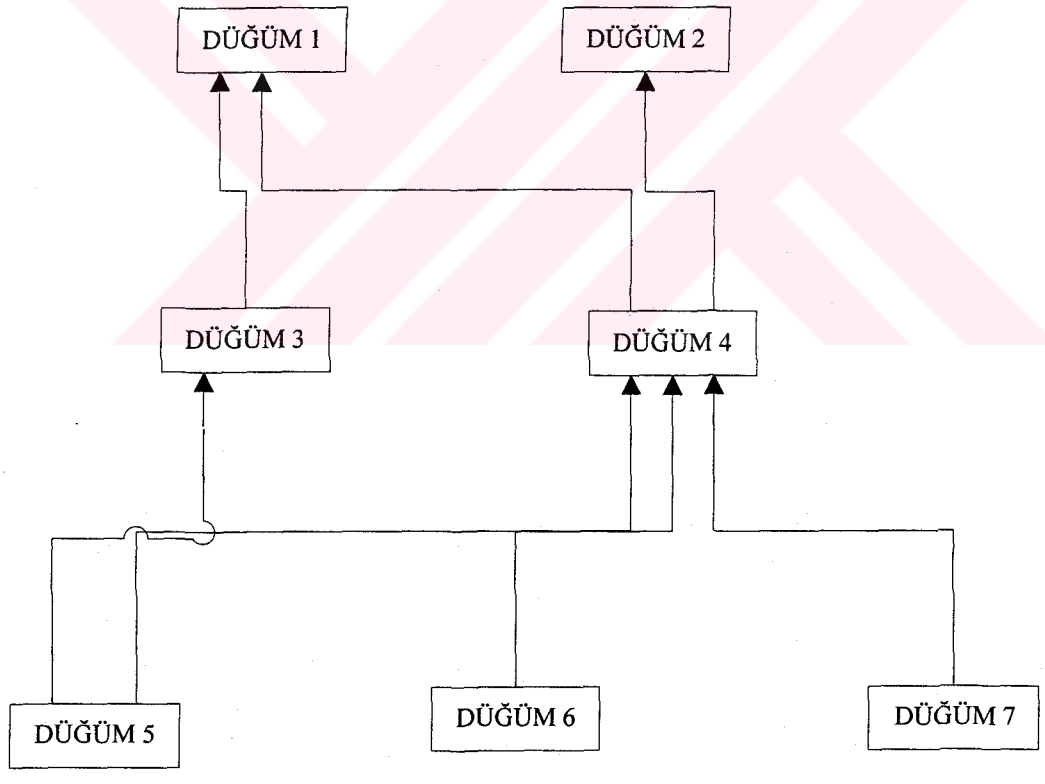
Bu dizaynın temel avantajı, kompleks problemi çözmede doğal bir yaklaşım getirmesidir. İnsanlar genelde parçala ve fethet (divide and conquer) felsefesiyle, kompleks problemleri küçük, kolay halledilebilir alt problemlere ayırarak çözerler.

Bu yolun diğer bir avantajı ise, sistemin geliştirilmesi ve bakımını çok kolaylaştırmasından kaynaklanmaktadır. Uzman sistem geliştirme sırasında, aynı anda sadece bir modül üzerinde yoğunlaşılabilir. Diğer modüllerden bağımsız olarak modül kurulup, test edilebilir. Bu özellikte dizayncıya herhangi bir anda sadece bir iş üzerinde yoğunlaşma kolaylığını kazandırmaktadır. Aynı zamand, projenin modifiye edilip yeni versiyonun ortaya çıkarılması sırasında harcanacak efor da kaldırılmış olacaktır.

Bu sistemle birlikte başka teknik bir avantaj da sunulmaktadır. Her bir modül ayrı uzman sistemler olabileceğinden, değişik bilgi gösterim teknikleri ve çıkarım stratejilerini sisteme integre etme imkanını da sağlamaktadır.

2.3.3 SEMANTİK ŞEBEKELER

Semantik şebekeler grafiksel bir bilgi gösterim metodudur. Bilgi gösterim metodunun bir şebeke yapısı üzerinde tanımlanmasında kullanılırlar. Başlangıçta insan zihninin psikolojik metotları olarak kullanılması için geliştirilen bu şebekeler, şu anda Uzman Sistemler için standart bir gösterim metodu haline gelmiştir. Bir şebeke düğümler olarak tanımlanan nesnelerin bir bütünüdür. Düğümler birbirlerine oklarla veya bağlayıcılarla birleştirilirler. Şebekedeki düğümler nesnelere, kavramlara veya olaylara karşılık gelir. Oklar gösterilecek bilginin çeşidine göre bir çok şekilde tanımlanabilir. Bu tanımlamalarda kullanacağımız ifadeler orijinalliği ve esnekliğinden dolayı İngilizce asılları muhafaza edilmiştir. Gösteride kullanılan en yaygın bağlayıcılar is-a ve has-a dirler. Şekil 2.9'de semantik şebekenin yapısı görülmektedir.

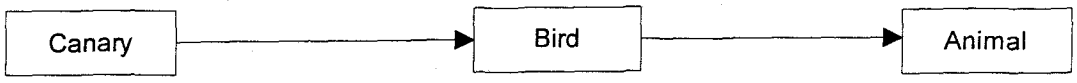


Şekil 2.9 Bir semantik şebekenin temel yapısı.

Semantik Şebekeye Bir Örnek

Aşağıda verilen çok basit gösterimde düğümler arasındaki bağlantı çok yaygın olarak kullanılan is-a ve is-an ile sağlanmıştır. Bu gösterimde ele alınan bilgi:

- Bird is an animal. (Kuş bir hayvandır)
- Canary is a bird. (Kanaya bir kuştur)



Bu iki ifadeden bir üçüncü bir ifade çıkar ki " Canary is an animal" ifadesi gösterimde kendiliğinden çıkar. Bunu ayrıca göstermek gerekmez. Bu şebekelerin faydalarından birisi de gereksiz işlemleri azaltmak ve kompleksliği gidermektir. Şebekenin alt kısmında alan nesneler bir üstlerinin sahip oldukları haklara da sahip olmaktadırlar. Bu örneğimizde kuşun hayvan olduğu ifade edilmekte, fakat kanaryanın hayvan olduğu söylenmemektedir. Ancak kanaryanın bir kuş olduğu ifade edildiği için kanaryanın da hayvan olduğu kolaylıkla anlaşılmaktadır.

Şekil 2.10'da verilen şebeke örneğini ele alırsak, düğümler arasında bağlayıcı olarak kullanılan yönlendirilmiş okların isimleri 'Has-a', 'is-a', 'is-a-value-of', 'is-in-charge-of' ve 'works-for' şeklindedir.

Düğümlerin ve bağlantıların (okların) nasıl isimlendirileceği konusunda herhangi mutlak sınırlayıcılar yoktur. Düğümler somut olabileceği gibi soyut anlam da taşıyabilirler.

Fakat bununla beraber hem düğümler hem de bağlantılar için bazı tipik sınıflandırmalar vardır.

Düğümler

Düğümler aşağıdaki kategorilerde sınıflandırılabilirler.

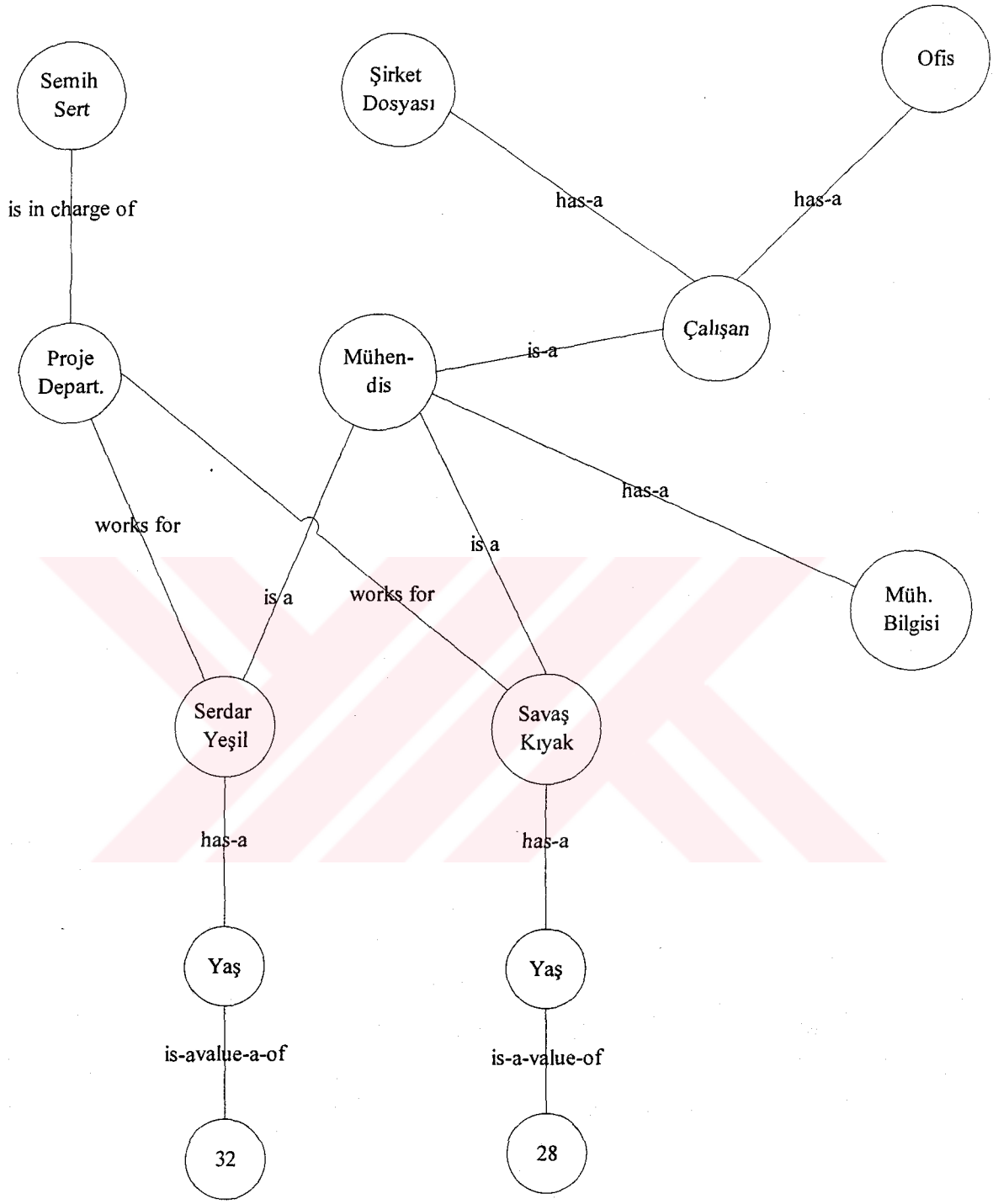
1. Nesneler genellikle düğümlerde temsil edilirler. Şekil 2.10 daki örnekte Serdar Yeşil, Savaş Kıyak ve Semih Sert fiziksel nesnelerdir. Düğümlerde soyut nesneler de bulunabilir. Örneğin 32 sayısı soyut bir nesnedir. Saydığımız her iki nesne grubuna girmeyen fakat ikisini arasında kalan nesnelerde vardır. Yani kavramsal varlıklar olabilir. Gösterimdeki 'proje' departmanı bir insan ismi olan Semih Sert'den çok soyut, fakat bir 32 sayısından ise daha az soyut olan bir nesnenin ismidir. Bu da bir düğüm olara gösterilir.

2. 'Çalışan' ve 'mühendis' gibi genel terimler düğümlerde yerleştirilirler. Bu terimler bireysel nesnelerin kategorilerine karşılık gelir.

3. Sıfatlar iki kategoride düşünülebilir. Bazı sıfatlar nümerik değere sahiptirler. Bazıları ise değerlere gereksinim duymazlar. Örneğimizdeki 'yaş' değer isteyen sıfatlara bir örnektir. 'Mühendislik bilgisi' veya 'ofis' sırasıyla 'mühendis' ve 'çalışanın' sıfatlarıdır. Nesneler ve sıfatlar arasındaki farklılıklar daima net değildir.

Daha genel söylenecek olursa, değer gerektirmeyen sıfatlar, genel terimlerle ilişki mümkün olabilen en yüksek soyutlama seviyesine çıkacaktır. Şebekede görüldüğü gibi, her bir çalışan bir şirket dosyasına sahiptir. Dolayısıyla her bir mühendis de çalışan olduğu için birer şirket dosyasına sahiptir. Bu şirket dosyasına sahip olmayı, tek tek şirketin her bir çalışan nesnesine bağlamak mümkün olacaktır.

Diğer taraftan, bir değer gerektirdiği için, 'yaş' soyutluluk hiyerarşisinde yükselemeyecektir. Yine her bir çalışanın bir yaşa sahip olmasına rağmen, yaşları aynı değildir. Oysa, yukarıda açıklamaya çalıştığımız olayda, işçilerin sahip olduğu şey hepsi için aynı idi. Burada ise yaş çalışanlar sınıfından sayılar sınıfına bir fonksiyondur.



Şekil 2.10 Bir semantik şebeke

Bağlayıcılar

Düğümler arasını birleştiren yönlendirilmiş bağlayıcılar (veya oklar) aşağıda belirtilen kategorilerden birisine girerler:

1. Has-a bağlayıcıları, <sıfat düğümlerini> ⟶ <genel terim düğümlerine veya nesne düğümlerine > bağlarlar. Bunlar genel bir terimin veya bir nesnenin bir sıfata sahip olduğunu ifade ederler. Genel bir terimin bir sıfata sahip olduğuna örnek olarak;

- bir çalışanın bir ofise sahip olduğu,

- bir mühendisin mühendislik bilgisine sahip olduğu, sayılabilir. Bir nesnenin bir sıfata sahip olduğuna örnek ise;

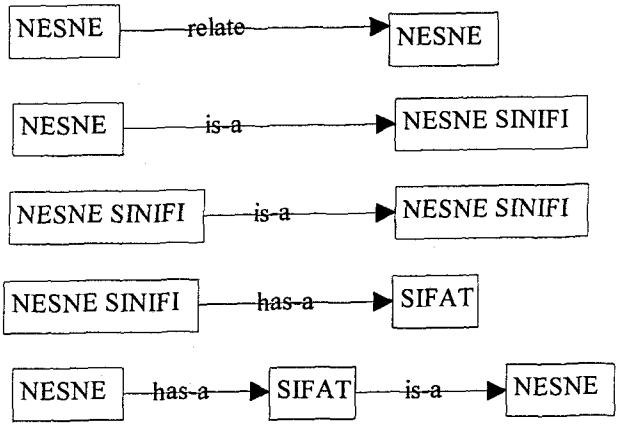
- Savaş Kıyak 28 yaşındadır, olarak verilebilir.

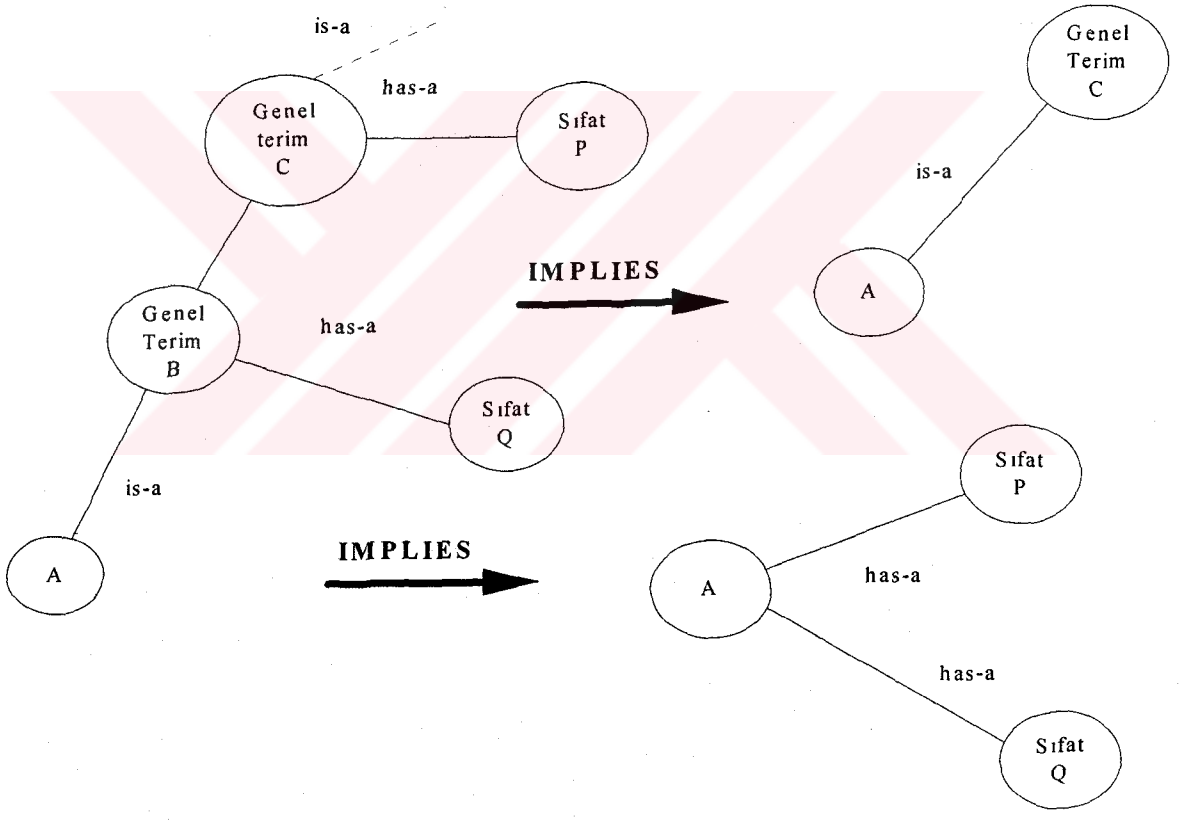
2. Is-a bağlayıcıları ise <nesne düğümleri> ile <genel terim düğümleri> veya <genel terim düğümleri> ile <diğer genel terim düğümleri> arasındaki ilişkileri gösterir. İlk durumda, ilişki üyelik kümesine benzerdir. İkinci durumda ise, bir alt küme ilişkisidir. Buma şebekeden örnek verelim: Savaş Kıyak bir mühendistir. Buna mühendislik kümesinin bir üyesi olduğu söylenebilir. Mühendislerin hepsi ise çalışandır. Dolayısıyla mühendisler kümesi çalışanlar kümesinin bir alt kümesidir.

3. Bunlarda başka bağlayıcılar da olabilir. Şekil 2.9'daki gösterimde, Serdar Yeşil ile proje departmanı arasındaki bağ iki nesne arasındaki ilişkiyi gösterir. Bu aradaki ilişkiler works-for ilişkisiyle bağlanmıştır. Benzer bir şekilde, is-in-charge-of iki nesne arasındaki diğer bir ilişkiyi gösterir.

a)

Yapılar **Örnekler**





Şekil 2.11 Bazı semantik şebekelerin özellikleri a) Yaygın bağ/düğüm yapıları b) is-a ve has-a yapısı

Önemli iki bağlayıcı grubu daha vardır. Olaylar veya durumlar arasındaki ilişkileri gösteren sebepsel bağlayıcılar bir çok bilgi tabanında bulunurlar. Örneğin; caused-by bağlantısı. Bir nesnenin diğerinin bir parçası olduğunu ifade etmekte yaygın olarak kullanılan is-part-ofilşkisi de mevcuttur. (The engine is part of the car)

Yaygın olarak kullanılan ilişki/dğüğüm yapıları ve hiyerarşisi şekil 2.11'da görülmektedir.

Semantik Şebeke Gösterim Avantajları

Şebeke gösterimlerinin avantajlarını şöyle sıralamak mümkündür:

1- Esneklik, bu gösterimlerin en önemli avantajlarından birisidir. Gerekirse yeni düğüm ve bağlantılar (ilişkiler) eklenebilir. Düğümlerin ve bağlantıların manaları konusunda sınırlamanın olması doğal bir şekilde bilgi eklemek çok kolaydır.

2- <Düğüm: is-a bağlantısı : düğüm> ve <Düğüm: has-a bağlantısı : düğüm> ingilizcede en yaygın yapılara karşılık gelir. Örneğin;

- Canary is a bird <nesne> is-a <genel terim>
- A bird is an animal <genel terim> is-a <genel terim>
- An animal has a skin <genel terim> has-a <sıfat-vasıf>

3- diğer bir özellik inheritance-miras özelliğidir. Buradaki miras'ın manası, bir düğümün kendisine bağlantısı bulunan diğer düğümün özelliklerine sahip olma yeteneğine karşılık gelir. Daha önce belirtildiği gibi, bilgiyi mümkün olan en soyut seviyede saklamak mümkündür. Alt seviyelerdeki düğümler üst seviyedeki düğümlerden is-a bağlantılarıyla özelliklere sahip olurlar. Şekil 2.9'da mühendislerin şirket dosyalarına ve ofislere sahip olduklarını söyleyebiliriz. Is-a bağlantıları yardımıyla, hem Serdar Yeşil hem de Savaş

Kıyak mühendislik bilgisine, şirket dosyalarına ve ofislere sahip olduklarını çıkarabilmekteyiz. Niteliklerin gereksiz tekrarlarını minimuma indirmenin yanı sıra, bu miras özelliği ingiizcede doğal bir çıkarım yöntemine karşılık gelir.

Bilgi gösterim şeması insan bilimi alnında Yapay Zeka araştırmasıyla yakından ilgilidir. Böylece semantik şebekenin kavramları, düğümler, bağlayıcılar ve miras hiyerarşi özellikleri, insanı bilgiyi nasıl sakladıkları konusundaki mevcut yapay zeka araştırmasıyla bağlantılıdırlar.

Şekil 2.12 Colins ve Quiallian'ın hafıza üzerinde çalışırken bir deneyde kullandığı semantik şebeke örneğini göstermektedir. Şebeke, farklı nesnelerle bağlantısı olan farklı niteliklerin hiyerarşik bir gösterimidir. Bu hiyerarşik düzenin farklı seviyelerine atanmış olan nitelikler, lâboratuar çalışmalarında deneye tabi tutuldu. Öğrencilere farklı ilişkiler hakkında sorular soruldu. Örneğin; bir öğrenciye şöyle sorular soruldu:

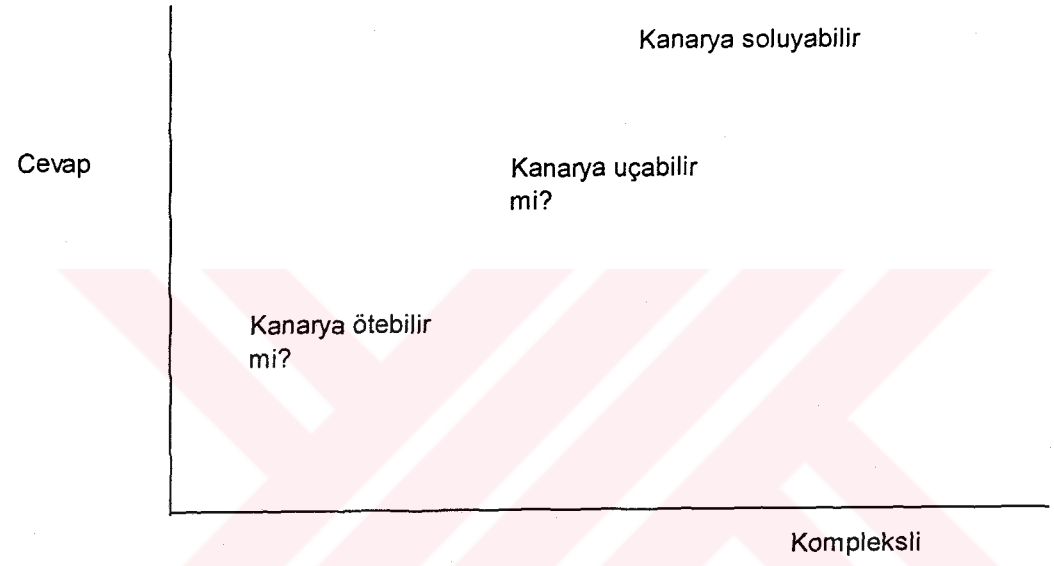
- Is a canary a bird? (Kanarya bir kuş mudur?)
- Can a canary sing? (Kanarya ötebilir mi?)
- Can a canary fly? (Kanarya uçabilir mi?)

yapılan dikkatli alışmalar sonucunda, öğrenciler'in "Kanarya uçabilir mi?" sorusuna karşılık cevap sürelerinin "Kanarya ötebilir mi?" sorusuna karşılık gelen cevap süresinden çok daha uzun olduğu gözlenmiştir. Colins ve Quiallian, insanların bilgiyi mümkün olabilen en soyut seviyede ele aldıklarını ispatlayarak, farklılıkları cevaplama süresi içinde açıklarlar. Böylece,

- Canaries fly (Kanaryalar uçar)
- Sparrows fly (Serçeler uçar)

gibi gerçekleri hatırlamaya çaba göstermek yerine, çok basit bir şekilde, hepsinin kuş olduklarını, kuşların uçtuklarını, tüyleri var olduğunu ve yumurta bıraktıklarını hatırlarız.

sŞekil 2.12 a) Semantik hiyerarşisi.



Şekil 2.12 b) Cevaplama süresi.

Şebeke Gösterimlerin Dezavantajları

Şebeke gösterimlerinin bazı faydaları yanında bazı dezavantajları da vardır. Dezavantajları aşağıdaki gibi sıralanabilir.

1- Yapısal unsurların esnekliği ve basitliği faydalı olduğu kadar, farklı semantik kategoriler arasındaki farklılığın gösteriminin zor olduğunu da belirtirler. Örneğin, <düğüm: link: düğüm> yapısı gereğince (<genel terim> is-a <genel terim>) yapısı tamamen aynı şekilde gösterilirler. Bu, iki yapı arasındaki farklılıkları çalıştırır. Farklılık, çıkarımların yürütümünde kullanılabilecek şekilde gösterilirler. Basit semantik şebekeler bu farklılıkları ortadan kaldırır.

2- Semantik şebekeler, özellikle stok medyası olarak etkindirler. Ancak kısa sürede istisnaların oluşması semantik şebekelerin kompleks bir yapıya sahip olmalarına neden olur. Şekil 3-4'de çalışan ve ofis'i birbirine bağlayan "has-a " bağlantısı çalışan oldukları belirtilen Serdar Yeşil, Savaş Kıyak,....., lerin ofislere sahip olduklarını ifade eden bilgileri saklanmasında "is-a" bağlantısı hiyerarşisi uygun ve etkin yöntemdir. Bununla beraber istisnaların bulunduğunu farz edersek, bunların ayrı ayrı kodlanması gerekmektedir. Örneğin Serdar Yeşil'in özel bir niteliği olarak, bir has-a bağlantısı ile saklanmasını gerektirir. Böylece hiyerarşik yapıda Serdar Yeşil'in ofis mülkiyeti ile ilgili özellik bloke edilmiş olur. Bir kaç istisna olduğu sürece, semantik şebeke avantajlı bir stoklama gösterimi olabilir.

2.3.4 ÇERÇEVE KULLANARAK BİLGİNİN GÖSTERİMİ

Bir bilgi tabanınıda gerçekleri ilişkileri göstermenin diğer bir yolu da çerçeveler kullanmaktır. Yapay zeka alnında, çerçeve (frame) terimi yaygın kavram ve durumları göstermenin özel bir yöntemine karşılık gelir. Çerçeve fikrini çıkaran Marvin Minsky çerçevi şöyle tanımlamaktadır:

" Bir çerçeve, standartlaştırılmış bir durumu (örneğin:belirli bir oturma odasında bulunmak veya bir çocuğun doğum günü partisine gitmek gibi) göstermek için bir veri yapısıdır. Herbir çerçeveye bağlı olan birçok çeşit bilgi vardır. Bu bilgierin bir kısmı, çerçevenin nasıl kullanılacağı ile ilglidir. Bir kısmı, bir kullanıcının bir sonraki aşamad ne olacağını tahmin edebilmesi ile ilgilidir. Bazıları ise, bu tahminler gerçekleşmediği taktirde ne yapılması gerektiğinin açıklamasıyla ilgidir."

Bir çerçeve, verilen bir nesneyi tanımlayan bir nitelikler grubundan oluşmaktadır. Her bir nitelik **slot** adı verilen alanlarda saklanır. Bir nesne ile ilgili tüm bilgiyi ifade edecek bir çok slotlar vardır. Dolayısıyla slotlar, niteliklerde olduğu gibi, değerler içerebilirler. Slotlar, aynı zamanda, **default değelreri** diğer çerçevelere bağlantı bağlantı kuran **göstergeleri, kurallar kümesini** veya niteliklere ait değerlerin değiştirilmesi veya hangi değerlerin olmasını belirleyecek **prosedürleri** de içerebilir.u ek özelliklerin olması, çerçeveleri N-S-D üçlü gösteriminden farklılaştırır. Bir açıdan, çerçveler bilginin daha zengin gösterimlerine imkan sağlar. Diğer bir açıdan ise çok daha kompleks ve daha basit olan N-S-D üçlü kural sistemlerinden çok daha zor geliştirilirler.

Slotların sahip olduğu bilgiyi değiştirmek ve yenilerini atamak için bir çok prosedürlere sahip olduğu bilinmektedir. Slotlara ilişkilendirilen üç tane faydalı prosedür tipi vardır. Bunlar :

1. Ekleme prosedürü: (if-added-procedure) Slota yeni bilgi girildiğinde çalışır.
2. Silme prosedürü: (if-removed-procedure) Slottan bilgi silinirken çalışır.
3. Gereksinim prosedürü: (if needed procedure) Bir slottan bilgi gereksinimi duyulursa çalışır. Fakat bu durumda slot boştur.

Çerçevelerin kullanımı bir örnek üzerinde anlaşolabilir. Şekil 2,13'te bir uzun-dönemli kredi için hazırlanmış bir çerçeveyi ifade etmektedir. Çerçeve nesnenin özel bir durumu için değerlerinin girildiği slotlardan oluşmaktadır. Şekilde verilen çerçeve tipi, bir nesnenin tipi olarak, slotlar da kredinin nitelikleri şeklinde düşünülebilir. Slotlarda yer alan her bir sıfatın karşısına onun belli bir değeri girilmiştir.

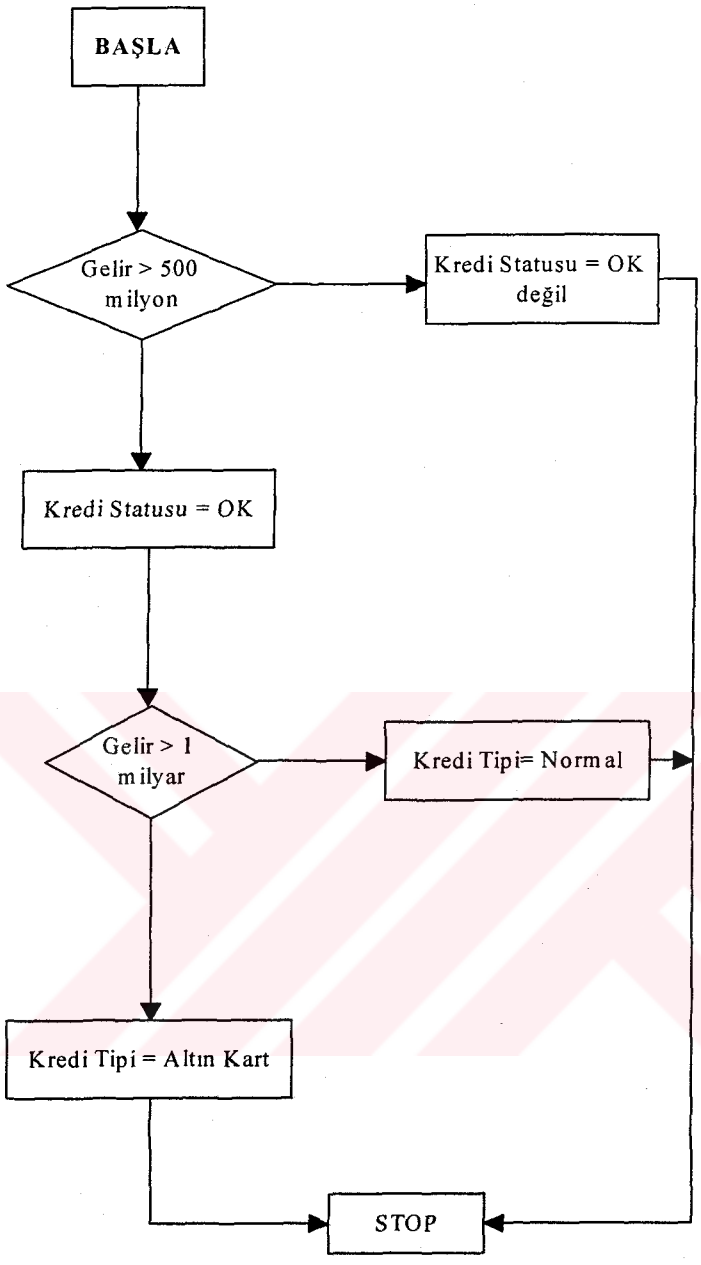
Çerçeve gösterimi aşağıda belirtilen durumlarda N-S-D üçlü gösterimlerinden daha çok gelişmişlerdir:

1. Çerçeve diğer çerçevelere dallanmak için göstergelere sahip olabilir. Bu göstergeler semantik şebeke gösterimlerinin tipik bir bağlantı çeşidi olan **is-a** bağlantı hiyerarşilerini gösterebilirler. Şekil 3-16'da "uzun dönem kredisi bir kredi çeşididir" ifadesi is-a (....dir) link bağlantısı söz konusudur.
2. Uzmanlar, belirli bir durumun gerçekleri hakkında tam bir bilgiye sahip olmayabilir. Bir çok durumlarda uzmanlar, kaybolan bilginin nesne için tipik olduğunu farzedeceklerdir. Çerçeve ise, bunun tanımlama bölümünde kurulmasına imkan verir. Şekil 2.13'te, kredinin kredinin faiz oranının belirtilmesi gerekmektedir. Eğer belirtilmemişse bu alan boş kalmayacak, daha önceden belirtilmiş faiz oranı geçerli olacaktır. Bu değer tanımlanmış bir değişken olabileceği gibi, aynı zamanda tek başına bir çerçeve de olabilir. Çerçeveler başka bir yerdeki çerçeveye referans verebilir. Herhangi bir aşamada bilinen tüm bilgiler eklenmesiyle önceki değer pasifleştirilebilir.

BİLGİLER

Çerçeve tipi	Uzun Dönem	
Çeşiti	Kredi	
SLOTLAR		**TANIMLAR**
Kredi No	1971	Nümerik
Miktarı	3.000.000.000.-	Parasal Değer
Keşide Tarihi	17/01/1996	Tarih
Vade Tarihi	17/01/2004	Tarih
Kreditör	Vakıf Bank	Yazı
Faiz Oranı	%85	Yüzde Default: mevcut oran
Ödeyebilirlilik	Evet	Değeri: Evet, Hayır
Geri Ödeme Şekli	Faiz	Değeri = 1 veya 2 1= sadece faiz 2= faiz + anapara

Şekil 2.13 Bir uzun dönem kredisinin çerçeve gösterimi.



DEKLERATİF

If Gelir > 1 milyar then kredi tipi = Altın Kart

If Gelir =< 1 milyar and Kredi Statusu= OK then Kredi Tipi = Normal

If Gelir >= 500 milyon then Kredi Statusu = OK else Kredi Statusu = OK değil

Şekil 2.14 Prosedürel ve dekleratif bilgi.

önermesi ise "Doğru" değeri almaktadır. Diğer lojik kurallar çıkarsamalara imkan sağlar. Eğer X'in değeri doğru olduğu biliniyorsa ve X⟶Y yi gerektiriyorsa (X IMPLIES Y), Y'nin de doğru olduğuna varabiliriz.

Yüklem esaslı lojik gösterim: Önermeler dayalı lojik sisteme göre çok genişlemiş bir sistemdir. Bu sistemin temel birimi nesnedir. Nesneler hakkındaki durumlar **predicate** şeklinde tanımlanır.

Yapay zeka dili olan PROLOG Predicate-lojik esaslıdır. Predicate esaslı sistem, lojik-bazlı fikirlerin bir yazı formunda kolayca belirtilmesi için geliştirilmiştir. Bir nesne hakkındaki çıkarsamanın nasıl yapıldığını basit bir şekilde açıklığa kavuşturan bir yöntemdir. Aritmetik yapıdan çok daha basittir. Prolog, doğal dil yapısına çok benzeyen kolayca anlaşılabilir bir yazım kuralın asahip olduğundan, predicate lojik yazım kuralının basitleştirilmiş bir şeklidir. Dolayısıyla, PROLOG lojik esaslı bir programlama dilinin geliştirilmesinde bu yazım kuralı avantajına sahiptir.predicate lojik gösterimde önceki cümledeki gereksiz tüm kelimeler elimine edilir. Daha sonra, cümledeki ilişki saptanır ve ilişkiyle ilgili tüm nesneler tespit edilir. Cümle nesneleri ilişkiden sonra gruplandırmak suretiyle transfer edilir. Lojik gösterimdeki yazım kuralını şöyle gösterebiliriz.

İlşki(nesne1,nesne2,...nesneN)

Bu ilşkiye nesneler ilişkinin üzerlerinde etkilediği elemeanlar şekline dönüşürler. Bu lojik sistemle ilgili örnekler tablo 2.1'de verillmiştir. Bu gösterm aslında ingilizce esaslıdır. Ancak bunu belli ölçülerde Türkçede ifade etmek mümkündür. Tabloda Türkçe cümleler bu lojik yapıda gösterilmeye çalışılmıştır. Ancak bir uzman sistemde bilgiyi ingilizce dil

yapısın agöre hazırlamak daha faydalı olacaktır. İngilizce dilinin kullanılmasındaki avantaj, dil yapısını Türkçeye göre daha basit olşudur. Aşağıdaki örnekleri inceleyerek İngilizce dil yapısını nedenli yararlı olabileceğine bakalım:

-Elma güzeldir.	Yüklem → -dir.
Apple is nice.	Yüklem →is.
-Kalem kısadır.	Yüklem → -dır.
Pencil is short.	Yüklem →is.
-Hava kötüdür.	Yüklem → -dür.
Weather is bad.	Yüklem →is.

Görüldüğü gibi Türkçedeki duruma göre değişen basit yüklemlerin (-dir, -dır, -dür gibi) İngilizcede tek bir yükleme (is) karşılık gelmektedir.

Doğal Dilde Cümle	Predicate-Lojik Yazım Kuralı
* Elma güzeldir.	* güzel(elma)
* Hava kötüdür.	* kötü(hava)
* Hoca çalışkan olan if öğrencileri sever.	* sever(hoca,öğrenci) Çalışkan (öğrenci)

Tablo 2-1. Doğal dildeki cümlelerin lojik dildeki ifadeleri.

BÖLÜM 3

UZMAN SİSTEMLERİN GELİŞTİRİLMESİ

3.1. Uzman Sistemin Geliştirilmesi

Uzman sistemler normalde bilgi mühendisleri tarafından geliştirilirler. Bilgi mühendisleri, bir uzman kişiden bilginin alınmasında ve bir uzman sisteme transfer etmede profesyonel olan kişilerdir. Bu kişiler, bilginin uzmandan alınmasında, sistemin kurulmasında ve organizasyonda kullanıma hazır hale getirilmesinde uzmandırlar. uzman sistemler, az çok birbirinden bağımsız altı aşamada geliştirilirler:

Aşama 1: Uygun bir problemin seçimi,

Aşama 2: Prototip bir sistemin geliştirilmesi,

Aşama 3: Tam bir uzman sistemin geliştirilmesi,

Aşama 4: Sistemin değerlendirilmesi,

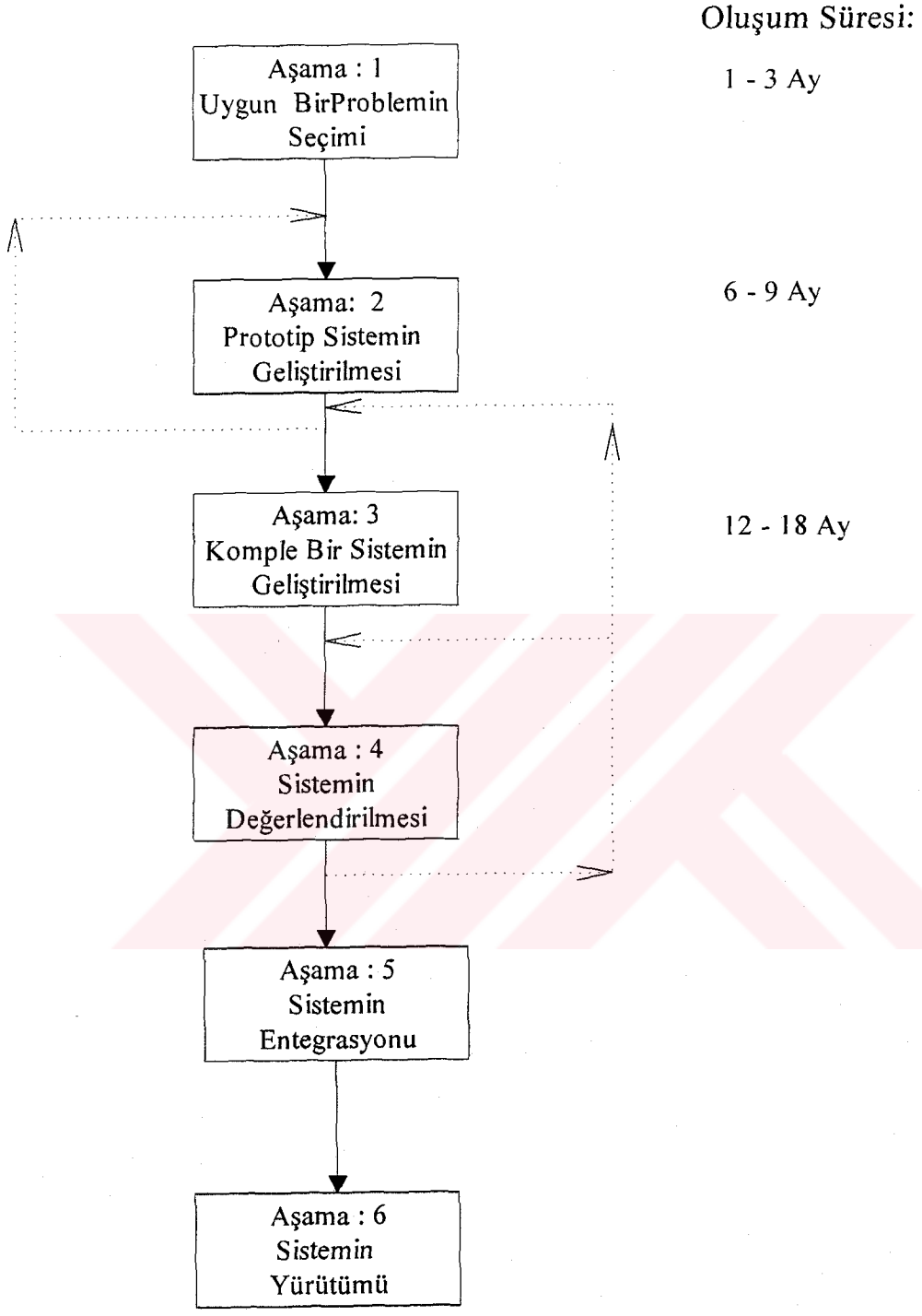
Aşama 5: Sistemin entegrasyonu, ve

Aşama 6: Sistemin çalıştırılması

Bu geliştirme sırası, tamamen sabit değildir. Daha sonraki geliştirmeler, bir önceki kararları etkileyebilir ve önceki çalışmalar tekrar gözden geçirilebilir. Adımların ve faaliyetlerin sırası, ideal bir projenin nasıl ilerlediği konusunda bir bilgi vermek içindir. Herbir adım detaylıca incelenecektir. Şekil 3.1. de bir uzman sistemin gelişme aşamaları görülmektedir.

3.1.1. Uygun Bir Problemin Seçimi

Problemin tanımlanması uzman sistem geliştirme prosesinin en zor aşamasıdır. Sistemin başarıya ulaşması bu aşamaya bağlıdır. Eğer tanımlama çok genel ise, sistem yönetilemeyen ve test edilemeyen bir durum alır ve tutarlı bir kurallar kümesi elde etmek mümkün olmaz. Eğer tanımlama çok dar tutulmuşsa,



Şekil 3.1 Bir uzman sistemin geliştirilmesi

sistem ne maliyet yönünden etkindir ne de doğruluğu kanıtlanabilir. Bilgi mühendisi ve uzman kişi ilk karşılaştıkları anda problemi tanımlarlar. Problem uzman sistemin eğimini saptadığı için, bu ilk aşama son derece önemlidir. Problem formal bir şekil kazandıktan sonra, yazılı bir tanımı yapılır ve tekrar gözden geçirilmeye başlanır. Bu gözden geçirme safhası, tanım üzerindeki tüm şikayetler cevaplanan ve tüm değişiklikler yapılana kadar devam etmelidir. Bu formal tanımlamanın bir maliyet/kar analizini içermesi de önemlidir. Bir uzman sistemin maliyet/kar analizi ile klasik sisteminki arasında hiçbir farklılık yoktur. Şekil 3.2 de görülen basit bir föy, sistemin gerçek karını değerlendirmekte kullanılabilir. Standart kural ise, herbir geliştirilmiş kural için 5-12 işçi saatı harcanabileceğidir. Küçük bir sistem, genellikle 100-400 kuraldan oluşurki maliyeti de $ 20000 ila $ 60000 arasında değişmektedir. Orta büyüklükteki bir sistem, ortalama olarak 400-3000 kuraldan oluşur ve maliyetler ise $ 60000 ila $ 500000 arasında değişir. Büyük bir sistem için düşünecek olursak, 3000'den fazla kural ihtiva eder ve maliyetler $ 1000000 ila $ 5000000 arasında değişecektir.

Proje:.....,

1. Mevcut Maliyetler:

Fonksiyon	Saat/ay	TL/saat	Maliyet/ay
.....	.....	.....	.....
.....	.....	.....	.....
A.Mevcut bakım maliyetleri	.....	.....	.....
Toplam(1+1A):	.....	.....	.....

2. Tahmin edilen maliyet:

Fonksiyon	Saat/ay	TL/saat	Maliyet/ay
.....	.....	.....	.....
.....	.....	.....	.....

3. Planlanan tasarruf (Toplam maliyet- tahmini maliyet)

Aylık: ..... Yıllık: .....

4. Geliştirme Maliyeti:

A) Bilgi kazanma: .....

B) Bilgi kodlama: .....

C) Klasik programlama: .....

İsim	Konu	Saat	Maliyet
.....	.....	.....	.....
.....	.....	.....	.....

Toplam Geliştirme maliyeti(A+B+C): .....

5. Sermaye:

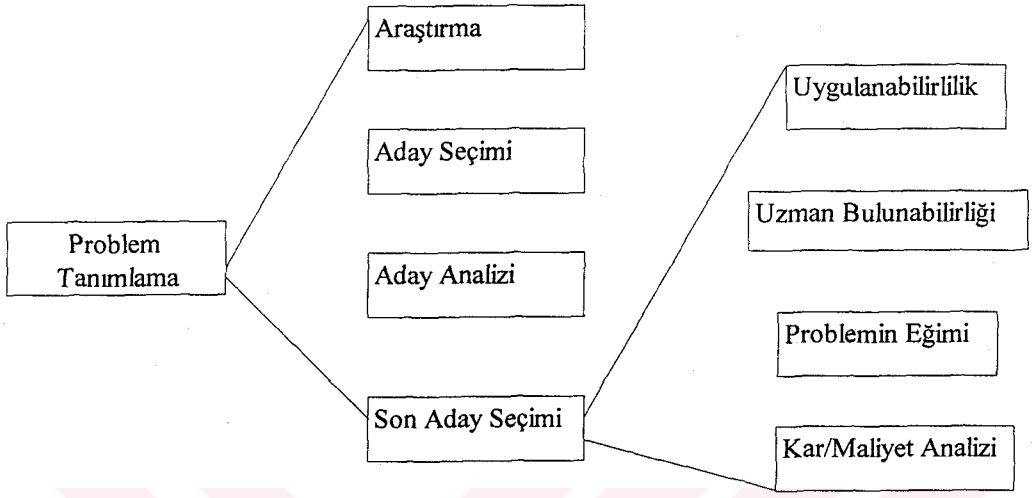
Parça	Miktar	Fiyat
.....	.....	.....

6.Aylık Bakım: .....

$$\text{Aylık Başabaş Noktası: } \frac{\text{Toplam Geliştirme + Sermaye}}{\text{Aylık tasarruf+ Aylık bakım}}$$

Şekil 3.2 Basit bir maliyet/kar analizi

Bu açıklamalara göre, uygun bir problemin seçimi, birçok faaliyetleri içermektedir. Bu faaliyetler, spesifik bir uzman sistemin geliştirilmesine karar vermek için önem taşırlar. Bu faaliyetleri şekil 3.3'te gösteridiği gibi aşamalara ayırmak mümkündür.



Şekil 3.3 Uzman sistemler için problemin tanımlanması

*** Problem alanı ve spesifik konunun tanımlanması**

Doğru problemin seçimi, geliştirme çalışmalarının en kritik kısmını oluşturur. Bununla ilgili teknoloji hala sınırlıdır. Eğer uygun olmayan bir problem seçilirse, hiç kimsenin nasıl çözüleceğini bilmediği problemlerin dizaynı için harcanan tüm çalışmalar boşa gidebilir. Veya, uygun olmayan bir problem, getirisi maliyetinden çok az olan bir uzman sistemle sonuçlanabilir. En kötüsü, kullanıcılar tarafından kullanılamaz olmasıdır.

Çok yakın zamanda, ana problemlerin çözümü için dizayn edilmiş uzman sistemlerin geliştirilmesi sözkonusu olacaktır. Bunun yanında, küçük problemlerin çözümü için dizayn edilmiş birçok küçük bilgi sistemleri de geliştirilecektir. Başlangıç geliştirme maliyetlerinin çok büyük olmasından dolayı, büyük sistemler, genellikle, getirisi büyük ve geri-ödemesi hızlı olan seçilmiş problemler üzerinde yoğunlaşmalıdır. Büyük sistem

geliştiricileri, mevcut araçlara uyum sağlayan problemlerin seçimiyle uğraşmalıdır. Ayrıca, belirli bir uzmanlığın ekonomik olarak otomatikleştirilmesiyle de uğraşacaklardır.

Diğer yandan, küçük sistemler, bilgi mühendisliğinin uygulamalarını ve araçların faydalılığını göstermek için geliştirilebilirler. Bazı sistem geliştiricileri, özellikle çaba gerektiren fakat özel araçlarla çözüme çok hızlı bir şekilde uyum sağlayan küçük konuları seçeceklerdir. Dolayısıyla küçük sistemler için projeler, öncelikle özel bir araçla kolay çözülüp çözülmediğine göre seçileceklerdir.

Uygun bir alanın ve konunun seçimi, uzman sistemin kullanılacağı alanların incelenmesiyle başlar. Eğer bir konunun yerine getirilmesi için gerekli olan bilgi statik, nümerik ve kolaylıkla raporlara dökülebiliyorsa, o zaman, bu alandaki problemlerin çözümü için en iyi yol klasik, algoritmik bilgisayar programları olacaktır. Bununla beraber, eğer konunun performansı, subjektif, değişken, sembolik veya kısmen kurallardan oluşan bilgiye dayanıyorsa, bu alan sezgisel bir yaklaşım sağlayan uzman sistemler için iyi bir uygulama alanıdır. Bazı küçük sistemler, klasik tekniklere uyum sağlayan problemlerin çözümü için de geliştirilebilir.

Uzman sistemlerin çok iyi bir şekilde çözebildiği konuların alt gruplarını tanımlamada yardımcı olan bazı hususlar aşağıda belirtilmiştir. Uygun konular:

- Konu dar bir ihtisas alanında olması gerekir,
- Tamamen sağduyu bilgisine dayanmayan konular,
- Duyu hisleri (koklama, görme, tatma,...) gibi sinyaller gerektirmeyen konular. Geçerli olan konular sembolleri gerektiren konulardır.
- Nümerik, matematiksel işlemlerden daha çok karar vermeye dayanan bilgi konuları,
- Çözümü için yeterli bilgisi bulunan konular,
- Bilgi yapısını, lojik kural bazlı programlamaya aktarabilecek bir uzmanı bulunan problemler,

- Uzman insan için ne çok zor ne de çok kolay olmayan konular. Uzmanın birkaç saatten az olmayan ve birkaç haftadan fazla olmayan bir zaman diliminde çözebileceği problemler,
- Mümkün olduğu kadar net bir şekilde tanımlanan konular,
- Değerlendirilebilen sonuçlara sahip olan konular.

Problem alanı ve konusunun tanımlanması aşamasında, cevap aranması gereken sorular şunlar olabilir:

- Problem nedir?
- Organizasyonun geçmişte buna benzer çözülmüş problemleri varmı? Varsa nasıl çözülmüştür?
- Eğer problem organizasyon için yeni ise, ne gibi çözümler tanımlanabilir?
- Problem daha detaylı ele alınabilir mi?
- Uzman sistem teknolojisi, bu problemlere nasıl bir çözüm sağlayacaktır?
- Uzman sistem ne yapacaktır? Sistemin amaçları nelerdir?
- Uzman sistem bu amaçlara nasıl ulaşacaktır?

Problem alanı ve konusu saptandıktan sonra, çalışma sahasında bir daraltma yapılır. Bundan sonra, cevaplandırılması gereken sorular arasında aşağıdakiler sayılabilir:

- Problemin hangi kısmını çözmeye çalışıyoruz? Veya problemin tümünü mü çözmeye çalışıyoruz?
- Özet olarak, son sistemimiz ne olacak?
- Diğer teknolojiler bu probleme uygulanabilir mi?
- Proje üzerinde çalışacak bir uzmana sahip miyiz?
- Uzman sistem teknolojisi problemimize etkin bir çözüm getirecek mi?

- Sistem ne çeşit tavsiyelerde bulunacaktır?
- Bu tavsiyeleri kime yapacaktır?

*** Uzmanlığı aktarmak için istekli bir uzmanın bulunması:**

Daha önce belirttiğimiz gibi, uzman sistemler, bir uzmandan spesifik bilginin alınması ve bu bilginin bir sisteme aktarılmasıyla geliştirilirler. Küçük sistemler (ve bazı çok büyük sistemler), birden fazla uzmanın bilgisini içerebilirler. Fakat çoğu sistemler tek bir uzmanın bilgi ve stratejilerini yansıtırlar. Dolayısıyla, bir uzman sistemin kurulmasında doğru bir uzmanın bulunması anahtar aşamadır. Bilginin, uzmandan alınması da bilgi mühendislerinin görevidir. Doğru uzmanın seçimi konusunda, bir bilgi mühendisi şunu söylemektedir: "İstediğiniz uzman, şirketin size vermek için istediği kişidir". Sistemin prototip ve diğer geliştirme aşamalarında, bilgi mühendisi ve uzman beraber çalışacaklardır. Bilgi mühendisi, bilginin yapısının oluşturulmasında uzmana yardım eder, problemlerin çözümünde kullanılacak önemli kavram ve kuralları belirler ve formülize eder.

Böylece, uzman olan kişi kendi uzmanlığını açıklarken zorlanmıyacaktır. İlk görüşmelerinde, hem bilgi mühendisi hem de uzman, başarılı bir şekilde nasıl etkileşimli olabileceklerine karar vereceklerdir. Bu beraberlikler belki en az bir yıl sürecektir. Dolayısıyla, her ikisinin de sempatik ilişkiler kurması çok önemlidir.

*** Problem için prototip bir yaklaşımın tanımlanması**

Uzman, problemin nasıl işleneceğini açıkladıkça, bilgi mühendisi bir ileri aşama için harekete geçecektir. Bildiği birçok uzman sistem geliştirme aletleri üzerinde çalışacaktır. Uzmanlığı, bilgi gösterimleri ve çıkarım stratejileri şeklinde karakterize eder. Böylece, bu konu üzerinde bir fikir yürütmeye başlar.

*** Çalışmanın maliyet/kar analizinin yapılması**

Bir problemin, uzman sistemin geliştirilmesi için uygun olduğu anlaşıldıktan sonra, sistemin maliyetleri ve karları düşünülmesi gerekir. Maliyet, bilgi mühendisinin yanısıra uzmanın harcadığı zamanı da içerir. Eğer problem ciddi bir problem ise, hem mühendis hem de uzman en az bir yılını verecektir. Bunun dışında, uzman sistem için gerekli olan donanım ve yazılım maliyetleri de sözkonusudur. Bu maliyetleri dört ana grupta toplayabiliriz:

– Donanım maliyetleri,

– Yazılım maliyetleri,

– Geliştirme maliyetleri ve

– Bakım maliyetleri.

Bu maliyetlerin dengelenmesi ancak bilgi sisteminin getirileriyle olacaktır. Bu getiriler, azaltılmış maliyetleri, artan verimliliği, iyileştirilmiş üretim ve servisleri veya yeni ürün ve servislerin geliştirilmesini içerebilir. Herhangi bir sistemin nispi maliyet ve karları, sistemin geliştirme maliyetini ne kadar sürede amorti edeceğini saptar. 1984 yılından sonra, büyük uzman sistemler geliştiren çoğu şirketler, hem maliyet hem de getiri açısından çok yüksek olan, fakat geri ödeme süreleri çok kısa olan, projeleri seçmektedirler. Uzman sistem geliştirme araçları, diğer bileşenlerden arındırıldıkça, daha az maliyetli fakat geri ödeme süreleri biraz daha uzun olan projelere doğru bir eğilim beklenebilir. Tablo 3.1. de birçok uzman sistem geliştirme seviyelerinde gerekli kaynakların bazıları verilmiştir.

Tablo 3.1. Bilgi-bazlı uzman sistemlerin geliştirilmesi

	SİSTEMİN TİPİ		
DÜŞÜNÜLEN KAYNAKLAR	Küçük	Büyük	Çok büyük
Kurallar	50-350	500-3000	10.000
Mevcut Araçlar	mümkün	mümkün	olabilir
Geliştirilmesi için Ggerekli yıl	1/4-1/2	1-2	3-5
Proje maliyeti ($)	40.000-60.000	500.000-1.000.000	2-5 milyon

*** Spesifik bir geliştirme planının hazırlanması :**

Bilgi mühendisi, aşağıdakileri gerçekleştirebileceğine karar verdikten sonra, alt geliştirme çalışmalarına rehberlik edecek spesifik bir plan hazırlamak için hazır hale gelir:

- Spesifik konunun bir uzman sistem tarafından yürütülebileceği,
- Uzman sistemin mevcut bir araçla kurulabileceği,
- Uygun bir uzmanın varolduğu,
- Amaçlanan performans kriterinin mantıklı olduğunu ve
- Maliyet ve geri ödeme süresinin müşteri tarafından kabul edilebilir olduğu.

Bilgi mühendisinin hazırlayacağı bu plan, sistemin lojik yapısını - temel çıkarımını- vermeli ve geliştirme süresince ele alınacak adımları belirtmelidir. Ayrıca, gerek maliyetleri ve beklenen sonuçları da içermelidir.

3.1.2. Prototip Sistemin Geliştirilmesi

Uzman sistem üzerindeki çalışma, ilk olarak, bilgi mühendisi ile uzmanın bir prototip oluşturmak için birlikte çalışmasıyla başlar. Prototip sistem, bir uzmanın uzmanlığıyla ilgili olgularının, ilişkilerinin ve çıkarım stratejilerinin nasıl kodlanacağı ile ilgili varsayımları test etmek için dizayn edilmiş bir uzman sistemin küçük bir versiyonudur. Bu sistem, aynı zamanda, uzman insanı geliştirme süresince aktif kılma açısından bilgi mühendisine avantaj sağlar. Böylece, ful bir sistemin geliştirilmesinde gerekli olan çaba konusunda, uzmanın sözünü kazanmış olur. Prototip bir sistemin geliştirilmesi aşağıdaki faaliyetleri içerir:

- Problem alanı ve konusu hakkında bilgi edinme,
- Performans kriterlerinin belirlenmesi,
- Bir uzman sistem kurma aracının belirlenmesi,
- İlk yürütmenin geliştirilmesi,
- Küçük örnekler üzerinde sistemin test edilmesi, ve
- Komple bir uzman sistem için, detaylı bir taslağın geliştirilmesi.

*** Problem alanı ve konusu hakkında bilgi edinme**

Bu aşama, bilgi mühendisinin, uzmanın alanı ve konusu hakkında olabilecek herşeyi öğrenmek için yoğun bir çalışmaya girmesiyle başlar. Bilgi mühendisi, uzmanla geniş bir etkileşime girmeden önce, problem alanına yabancı kalmaması için, dökümanları gözden geçirir ve ilgili kitapları okur. Bilgi mühendisi, kendisini uzmanla konuşmaya hazır görünce, konuyu daha detaylı tanımlamak için ilk diyalogu başlatır. Aynı zamanda, bilgi

mühendisi, uzmanın sezgisel yapıdaki kararlarını formüle etmek ve çıkarım stratejilerini açıklamak için gerekli bilgileri öğretmeye çalışılır.

Bilgi mühendisi, uzmandan, konu hakkında şu ana kadar çözdüğü birkaç problem örneğini açıklamasını talep eder. Uzman bu durumlarla ilgili tüm dökümanları birleştirir. Bilgi mühendisi, uzmanın her bir duruma nasıl yaklaştığını dinler ve her bir özel duruma bir çözüm geliştirmek için, adım adım prosedürü temin eder. Bilgi mühendisi, uzmandan normal sesle düşünmesini ve her bir kararın arkasında yatan çıkarım işlemlerini açıklamasını talep eder. Ek olarak, bilgi mühendisi, uzmandan, çıkarımını doğruluğunu kanıtlamasını da isteyebilir.

Mümkün olduğu kadar, uzmanın kullandığı çıkarım işlemleri, kurallar seti halinde tekrar formüle edilir. Bu işlem, uzmanın analitik çalışmasını açıklığa kavuşturmakta bilgi mühendisine yardımcı olur. Ayrıca, uzmana, düşüncelerini *i-then* (şart-sonuç) kuralları şeklinde nasıl formüle edeceğini de gösterir. Uzmanın problem çözme ve sezgisel ifadelerinin incelenmesi, bilgi mühendisinin, çıkarımda çok önemli olan gerçekleri ve ilişkileri tanımlamasına yol açacaktır.

Bilgi mühendisi, uzmanın problem çözme stratejileri ve sezgisel yaklaşımları hakkında bilgi sahibi oldukça, benzer sezgisel kural ve stratejilerin uzman sistemlerde nasıl yerleştirileceği üzerinde çalışacaktır. Bilgi yapılarını ve çıkarım stratejilerini, bilgi mühendisleri tarafından iyi bilinen kategorilerden birisi içinde sınıflandırmak için bazı sorular soracaktır. Bilgi mühendisinin sorabileceği sorular arasından bazıları aşağıda verilmiştir:

- Bilgi dağınık ve yetersiz mi? veya gereksiz ve çok mu?
- Gerçeklere ve kurallara atanmış belirsizlik faktörleri var mı?
- Problemin yorumu, zaman içinde oluşan olaylara bağlı mıdır?
- Konu hakkındaki bilgi nasıl elde edildi?

- Bilgiyi saptamak için, cevaplanması gereken ne gibi sorular vardır?
- Gerçekler güvenilir, doğru ve kesin midir? Yoksa, güvenilmez, doğru olmayan ve değil midir?
- Bilgi, çözülecek problem için, tutarlı ve tamamlanmış mıdır?

*** Performans kriterinin belirlenmesi**

Uzmanın yaptığı işlevler tam olarak saptandıktan sonra, bilgi mühendisi, prototip sistem için performans kriterini saptamaya başlayacaktır. Performans kriteri açık ve net bir şekilde belirlenmelidir. Sistemden, belki, 4-5 spesifik durumlarda uzmanın ulaştığı aynı sonuçlara ulaşması beklenecektir. Belki de, henüz ele alınmamış 4-5 özel durumlarda, farklı 4-5 uzmanın ulaştığı aynı sonuçlara ulaşması beklenecektir. Kriter ne olursa olsun, bilgi mühendisinin çalışmasını başarılı bir şekilde tamamladığını kanıtlayacak bir testin yapılabilmesi için kriterin belirlenmesi gerekir.

Aynı zamanda, spesifik bir performans kriterinin saptanması, bilgi mühendisinin dikkatini, başlangıç şartlarına ve sistemden üretilmesi gereken son çıktılar üzerine toplayacaktır.

*** Bir uzman sistem kurma aracının seçimi :**

Bilgi mühendisi, bilgiyi işlemekte kullanılan çıkarım stratejilerini ve uzmanın sahip olduğu tüm bilginin yapısını kavradıkça, prototip sistemin geliştirilmesinde kullanılacak mevcut uzman sistem araçlarına karar verecektir. Prototipin çıkartılmasında en önemli sonuç, seçilen aracın uygunluğunun bir testidir.

Bu aşamada, mevcut tüm ticari uzman sistem araçları ile ilgili bir çalışma yapılır. Birçok uzman sistem geliştirme aracı bulunmasına rağmen, belirli gereksinimler doğrultusunda bu alan daralır. Örneğin, sistemimiz 400 kuralın altında küçük bir sistem olacaksa ve hız faktörü önemli değilse, seçilecek shell normal PC'lerde çalışmalıdır. Eğer sistem 1000'den fazla kuraldan oluşacak ve hız çok önemli ise, özel bir sistem veya büyük makinalar gerecektir. Bunlar özel programlama dillerine sahiptir. Dolayısıyla, bu durumda shell'ler

elimine edilmiş olmaktadır. Bu aşamada ele alınması gereken sorulardan bazıları aşağıda verilmiştir:

- Piyasada, hangi uzman sistem geliştirme araçları mevcuttur?
- Hangi bilgi gösterim şemaları, uzman sistem shell ürünlerine uygundur?
- Hangi çıkarım ve kontrol mekanizmaları, mevcut uzman sistem shell'leri tarafından kabul edilmektedir?
- Kullanıcı-sistem arabirimi nasıldır? Kullanımı kolay mıdır?
- Şirket bu ürünler için yardımda bulunacak mı?
- Shell'ler hangi donanımlar da çalışmaktadır?
- Şirkette shell'in çalışabileceği herhangi bir dananım var mıdır?
- Shell'in sağlayacağı fonkksiyonlara karşı, maliyeti ne kadardır?
- Shell'in dökümantasyon ve eğitim bileşeni var mıdır?

Projenin gereksinmeleri ile mevcut araçlar karşılaştırılır ve proje için bir veya daha çok uzman sistem geliştirme aracı seçilir. Bu seçim prosesi, aşağıdaki sorulara cevap vermelidir:

- Hangi araçlar, gerekli bilgi gösterim şemalarına destek sağlar?
- Hangi araçlar, gerekli çıkarım tekniklerine destek sağlar?
- Hangi araçlar, grafik gibi gerekli araçları sunar?
- Hangi araçlar, diğer proje gereksinimlerinin tamamına veya bir çoğuna cevap verir?
- Hangi araçlar, proje bütçesine uygundur?

Hiçbir araç proje gereksinimlerini tamamen karşılamıyorsa, o zaman üç alternatif vardır:

(1) Proje gereksinimlerini en iyi karşılayan bir veya daha azla araç kullan,

(2) Kendi uzman sistem geliştirme aracını geliştir,

(3) Şu anda projeden vazgeç, bir yıl sonra tekrar ele almaya çalış. Çünkü, her yıl birçok yeni uzman sistem geliştirme aracı dizayn edilmektedir. Bunlardan, belki, en az biri proje gereksinimlerine uygun olabilir.

*** İlk yürütmenin geliştirilmesi :**

Bir araç seçildikten ve ilk örnek çok iyi analiz edildikten sonra, uzman sistemin prototip bir versiyonunu geliştirilmeye başlar. Elde edilen prototip sistem, daha sonra diğer örnekler üzerinde test edilir. Her bir durum icra edildikçe, bilgi mühendisi ve uzman, sistemin çıkarımını gözler ve kuralların beklenildiği gibi çalışıp çalışmadığı tartışılır. Sonuç olarak, belgi tabanı, uzmanın bilisi, sezgisel kuralları ve çıkarım stratejilerine uygun olarak tekrar gözden geçirilir.

Dolayısıyla, bilgi mühendisi, probleme uyan ve en azından, genel olarak, prototip sistemi tatmin edecek bir araç seçecektir. Bir prototip modellemenin amacı, uzman sistemin en son şekline ulaşmak değildir. Fakat aşağıda belirtilen elemanların geliştirilmesi açısından önem taşır:

– bir uzman sistem kurma aracı,

– uzman bilgisinin gösterimi, ve

– konuya uygun çıkarımlar yürütmek için bir strateji.

*** Küçük örnekler üzerinde sistemin test edilmesi :**

Bilgi mühendisi, prototip sistemi kurduktan sonra, birçok örnek çalışma üzerinde sistemin çalışmasını kontrol eder. Bu kontrol aşamasında uzman da bulunur. Bu testlerin iki önemli fonksiyonu vardır. Birincisi, uzman bilgisinin gösteriminde kullanılan formalitelerin, örneklerde ele alınan konulara uygun olup olmadığının saptanmasına imkan sağlar. İkincisi ise, uzmandan sağlanan bilginin bir uzman sistem tarafından nasıl kullanıldığını görmeye imkan sağlar. Birinci özellik bilgi mühendisliğiyle ilgili olup, ikincisi uzman kişiyi

ilgilendirmektedir. Böylece, uzman sistemin test edilme aşamasında aktif bir rol aldığı için, bilginin elde edilme işleminde daha çok içli-dışlı olur. Geliştirmenin bir sonraki aşamasında, sistemin performansının ayarlanmasında, uzmanın sistemle etkileşimli olması isteneceğinden, uzmanın bu rolü çok önemlidir.

*** Komple bir uzman sistem için detaylı bir taslağın geliştirilmesi :**

Prototip sistem, fonksiyonunu iyi bir şekilde yerine getirdiği zaman, uzman ve bilgi mühendisi, komple bir sistemin geliştirilmesinde nelerin gerekli olacağını belirlemede iyi bir aşamaya gelirler. Nesne ve sıfatların orijinal seçimi güç ise, tanımlanmalıdır. Komple bir uzman sistemin oluşturulmasında gerekli olan sezgisel kuralların toplam sayısı konusunda tahminler yapılmalıdır. Bu aşamada, performans kriteri daha kesin bir şekilde ifade edilebilir. Tüm bu bilgiler, bir plan, program ve bütçe ile beraber belli bir taslak dahilinde yapılaştırılılar. Bu taslak dökumanı, sistemin geliştirilmesine rehberlik yapacaktır.

3.1.3. Komple Bir Uzman Sistemin Geliştirilmesi

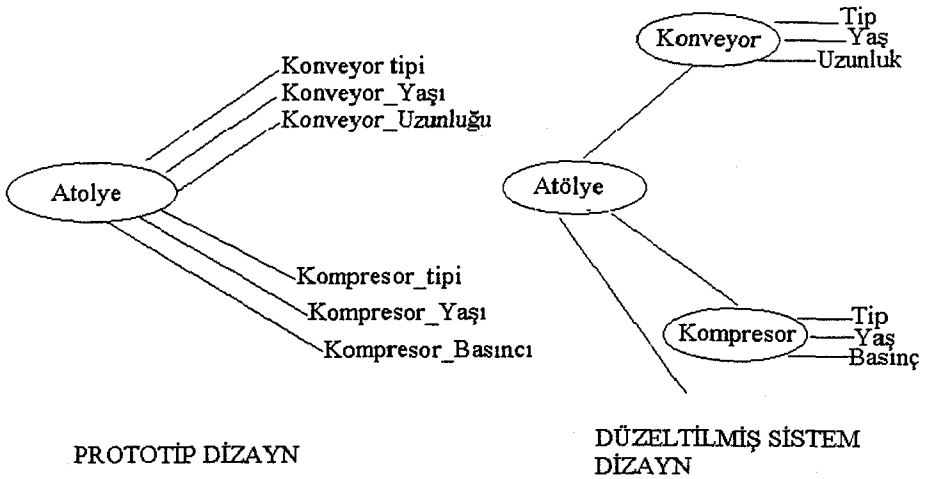
Herkes, prototip sistemin arzu edildiği gibi çalışacağına ve komple bir sistem için hazırlanan tasladığın, belirlenen performans kriterini gerçekleştiren bir uzman sistemle sonuçlanacağına kanaat getirdikten sonra, bilgi mühendisi ve uzman komple bir sisteme ulaşmak için prototip sistemi genişletmeye başlarlar. Komple bir uzman sistemin geliştirilmesi aşağıdaki faaliyetleri içine alır:

- Komple sistemin öz yapısının yürütümü,
- Bilgi tabanının genişletilmesi,
- Kullanıcı arabiriminin eklenmesi (Interface),
- Sistemin performansının gözlenmesi.

*** Komple sistemin öz yapısının yürütümü:**

Bilgi mühendisleri arasında çok kullanılan bir atasözü vardır: "Prototipten uzaklaşmak genellikle en iyi yoldur". Bilgi mühendisliği araçları kısa bir zaman diliminde hızlı bir modelleme sağlar. Böylecle, bu aşamada, bilgi tabanının temel taslağı üzerinde tekrar düşünmek gerek. Bununla, özel bir aracı terkettiğimiz manasına gelmez. Burada ifade etmek istediğimiz şey, sistemde içerilecek nesne ve niteliklerin tam listesinin mühtemelen biraz değişeceğidir. Hiyerarşik ilişkilerin tekrar düzenlenmesi gerekebilir. Sergisel kurallarda ele alınan çıkarımın gerçek şekli, uzmanın bilgi ve problem çözme stratejilerinin en iyi bir şekilde nasıl temsil edilebileceği doğrultusunda tekrar tanımlanabilir.

Prototip aşaması sırasında, ciddi problemlerin farklı bir araca gereksinim duyduğu ortaya çıkabilir. Eğer böyle bir durum varsa, bu aşama tekrarlanmalıdır. Bununla beraber, genellikle modelleme başarıyla sonuçlanır. Fakat kural ve gerçeklerin başlangıç gösterimleri tekrar ele alınması gerekbilir. Bunu bir örnek üzerinde açıklayalım. Merkez nesnesi üretim atölyesi etrafında yapılaştırılmış prototip bir Fabrika Bakım Danışman sisteminin geliştirildiğini düşünelim. Bu atölyenin farklı bir çok konveyör şeritleri ve bir çok kompresör birimlerinine sahip olduğunu varsayalım. Modellemenin başlangıcında, bilgi mühendisi ve uzmanın konveyörler ve kompresörler hakkındaki genel bilgilerin "ara nesne" olarak gösteriminin avantajlarını düşünmemiş olabilirler. Bunun yerine hem kompresör hem de konveyörleri atölyenin nitelikleri olarak göstermiş olabilirler. Bu safhada, problem üzerinde çalışıldığı zaman, bilgi mühendisi ve uzman, konveyörlerin ve kompresörlerin birer nesne olarak gösterilmesi gerektiğini ortaya çıkarabilirler. Şekil 3.4'te bir başlangıç prototip taslağın daha güzel bir gösterime nasıl dönüştürülebileceğini gösterilmektedir.



Şekil 3.4 Düzeltilmiş sistem dizayn

*** Bilgi tabanının genişletilmesi :**

Üçüncü aşamanın en önemli faaliyeti, çok büyük miktarlarda ek höristik, sezgisel kuralların eklenmesi işlemidir. Bu ek sezgisel ifadeler, özel bir durumun ele alınmasında daha çok kurallar sağlayarak sistemin derinliğini artırır. Aynı zamanda, uzman ve bilgi mühendisi, ek alt problemleriyle ve uzmanlık alanlarıyla ilgili kuralların içerilmesine karar verebilirler. Bu da, sistemin genişliğini artırır.

*** Kullanıcı arabiriminin eklenmesi :**

Uzman sistemin temel yapısı kurulduktan sonra, sistemin bilgileri kullanıcıya aktarabileceği bir ara biriminin geliştirilmesi ve eklenmesine çalışılır. Burada dikkat edilmesi gereken en önemli husus, kullanıcının sistem mantığını kolaylıkla izleyebilmesidir. Dolayısıyla bu konudaki çalışmalar, yapılan açıklamalar ve kullanılan deyimler üzerinde yoğunlaşacaktır.

Sistem, kullanıcının araştırma yapmasını kolaylaştırmalı ve doğal olmalıdır. Özellikle, grafik gösterimler faydalı olabilir. Sistemin çkarım işleminin izlenmesine imkan sağlayan bu gösterimler, sistemin pazarlanmasında anahtar bileşen görevini yapabilir.

*** Sistemin preformansının gözlenmesi :**

İyi bir araç, sistem çıkarımını denetlemek için bazı örnek durumların çalıştırılmasına imkan veren bir bilgi mühendisliği arabirimidir. Böyle bir araç, aynı zamanda, sistem çalışırken uzmanın sorular sormasına da imkan sağlar. Bu açıdan, uzmanlar, kuralların sisteme girişi hakkında yeterli bilgiye sahip olmuşlardır. Böylece, bu durum, bilgi mühendisinin sistemin kontrolünü uzmana transfer etme prosesinin başlangıcıdır. Bilgi mühendisinin desteği olmaksızın sistemin yürütümünün devamlılığı sağlanmış olacaktır. Bu da, sistemin performansını gösterir.

3.1.4. Sistemin Değerlendirilmesi

Uzman ve bilgi mühendisi, uzman sistemin tamamlandığına kanaat getirdikten sonra, prototip aşamasında kararlaştırılan performans kriterine karşı sistem test edilmelidir. Bu aşama, aynı zamanda, diğer uzmanların sistemi görmeleri için davet edildiği ve yeni problemlerin denendiği aşamadır.

Bir uzman sistemin değerlendirmeye çalışmak beraberinde bazı yeni problemleri de getirebilir.Değerlendirme için bazı belirli teknik özellikler kolaylıkla uygulanabilir. Örneğin; cevaplama oranı, ana hafıza kullanımı. Fakat bir uzman sistem olarak, sistemin başarısını ölçmek çok zordur. Problemlerin bir kısmı, uzman insanın değerlendirilme alanının birçok zorluklarla yüklü olmasından doğmaktadır. Bu problemler kendilerine bir model teşkil eden sistemlere transfer edilirler.Değerlendirme konusunda bazı belirli yaklaşımlar yapılabilir:

(1) Sistem bir test olaylar serisiyle değerlendidirilir. Bu olaylar yapılaştırılmış ya da tesadüfi olarak alınabilir. Uzmanlar tarafından bu olaylara karşılık gelen cevaplar hazırlanır. Böylece cevapları içeren bir kontrol cevap kümesi oluşmuş olur. Daha sonra, sistemin

performansı bu kontrol cevap kümesiyle karşılaştırılabilir. Buradaki problem uzmanların farklı olmalarıdır.

(2) Sistemin değerlendirilmesi realite karşısında bir test olarak yapılabilir. Analizleri bilinen gerçek problemler sisteme yüklenir. Eğer doğru cevaba ulaşılırsa sistem başarılı, aksi halde sistem başarılı değildir. Eğer sistem probabilistik tavsiyeler vermeyi gerektiriyorsa veya içeriyorsa, zorluklar ortaya çıkar. Bu sefer de, sistemin başarı oranı uzman insanların başarı oranlarlarıyla karşılaştırılabilir.

(3) Daha az formal olan testler ise, uzman sistemin geliştirilmesinde rol almayan bir uzman insan gerektirecektir. Bu uzman bilgi tabanını kontrol eder, sorular sorar ve verilen cevapları değerlendirir. Bu daha subjektif ve bilimsel olmayan bir yapıya sahiptir. Bu nedenle, bu yönteme başvurulmayabilir. dar anlamda, bu metod bilimsel olmamasına rağmen, değerlendirmede değer anlayışları sağlarlar. Ayrıca, uzman sistemlerin değerlendirilmesinde neden aynı eğilim kabul edilmiyor gibi bir soruya herhangi bir sebep yoktur.

Değerlendirmeyle ilgili ana problemlerden bir tanesi, sistem tarafından kullanılması gereken çok miktardaki lojik rotalardır. Bunlardan sadece bir kısmı test edilebilir. Eğer sistem belirsizlik altında çıkarım yapıyorsa, durum biraz daha komplekstir. Bu sınırlama, ancak, uzman sistemlerin kurulmasında hatasız ve doğru geliştirme tekniklerinin kullanılmasıyla giderilebilir. Dolayısıyla, hiyerarşik ve modüler geliştirmeye özel dikkat göstermek, yardımcı ve faydalı olur.

Değerlendirmede dikkat edilmesi gereken hususlar şunlardır:

- Sistem doğru sonuçlar için doğru tavsiyeler veriyor mu?
- Kullanıcının istediği bilginin seviyesi ve verilen tavsiye açısından, sistem son-kullanıcıların bilgi seviyesiyle uyum sağlıyor mu?

- Sistem, yan etkileri olmaksızın değişikliklerin yapılabileceği bir şekilde mi tasarlandı?
- Kullanıcılar açısından sistemin çalışması kolay mı?
- Kullanıcı, sistemin sağladığı tavsiyeleri anlayabilmekte ve kullanabilmekte mi?
- Spesifik performans kriteri sağlanmış mı?
- Sistem, uzman insanın kabul ettiği tavsiyeler mi veriyor?
- Sistem, sonuca arzu edilen zaman içinde mi ulaşıyor?

3.1.5. Sistemin Entegrasyonu

Uzman sistem geliştirmede bir sonraki aşama, uzman sistemin işletileceği çalışma ortamına entegre edilmesini ve sistemi kullanacak kişilere eğitimin verilmesini gerektirir.

Entegrasyon işlemiyle, bir şirketin içinde mevcut sistemlerle yeni bir uzman sistem çalışması yapmak için gerekli olan prosedürlerin tümü kasdedilmektedir. Bu işlemle uzman sistemde önemli değişiklikler yapılması ifade edilmemektedir. Sistem kullanıma hazır hale getirildikten sonra, eğer çok önemli değişiklerin yapılması gerekli olduğu saptanırsa, bilgi mühendisi veya mühendis grubu modelleme veya geliştirme aşamasına geri gider ve gerekli duyduğu değişiklikleri yapar. Sistemin kodunu değiştirmesini bilen birisi de bu işlemleri yapabilir. Entegrasyonla, uzman sistem ve çalıştığı yapı arasında bağlantılar geliştirmeyi ifade etmekteyiz. Bir uzman sistemin entegre işlemi genellikle, uzman ve sistemin kullanıcılarıyla ilgilenen sistem personeli tarafından gerçekleştirilir. Bu aşamayla ilgili faaliyetler şunlar olabilir:

- Teknoloji transferi için düzenleme,
- Sistemin hızı ve kullanışını iyileştirmek için, diğer veritabanlarıyla, diğer araçlarla veya diğer bir donanım ile ara bağlantısının kurulması.

*** Teknoloji transferi için düzenleme :**

Uzman sistem hazır olduğu zaman, bilgi mühendisi sistemin nasıl kullanılacağı ve devamlılığının nasıl sağlanacağı konusunda uzmanlardan, kullanıcılardan ve sistem personelinden emin olmalıdır. Bilgi mühendisi, bilgi ve teknoloji know-how'unun transferini gerçekleştirdikten sonra, sistemi kullanıcıların eline teslim ederek projeden çekilmelidir.

Her şirketin yapısı, personelin uzman sistem kullanımına hazırlamaktan sorumlu olan kişilerden farklı bir şey isterler. Burada uzman sistem ile uzman arasında bir mücadele vardır. sistemin faydalı tavsiyeler vereceği konusunda uzman ikna edilebilse, sistemi daha çabuk kabul edeceklerdir. Uzmanların sistemin kullanışlılığına ikna edilebilmesi, ancak her bir şirket uzmanının sisteme yeni problemler sunması ve sistemin nasıl çalıştığını görmesiyle mümkündür. Kabülün önemli bir yönü, uzman sistemin, uzman insanların yerine geçme yöntemi olmaktan çok, çaba isteyen konulardan uzmanları rahatlatacak bir yardım yöntemi olarak gösterilmesidir.

Uzman olmayan personelin ikna edilmesi, bir şirket bünyesine sunulan herhangi yeni bir sistemle ilişkili problemlerin tümünü içerecektir. Başarı, iyi bir planlamaya, iyi iletişimler kurmaya, bu değişiklikle ilgili uygun üstünlüklere ve sistemin desteğine bağlıdır.

*** Sistemin diğer veri tabanları, araçları veya diğer bir donanım ile ara bağlantısının sağlanması :**

Entegre işleminin diğer bir amacı, uzman sistem ile mevcut veritabanları ve şirketin diğer sistemleriyle ara bağlantısının sağlanmasıdır. Bir uzman sistem, kendisine girdi sağlayacak diğer donanım ve araçlardan bilgi toplama ihtiyacı duyabilir. Diğer amaçlardan biride, sistemin daha kullanışlı, daha hızlı ve etkin çalışmasını sağlamak için zamana bağlı faktörlerde iyileştirme yapmaktır. sistem alışılmamış farklı bir yapıda çalışacaksa, donanımın fiziksel özelliklerinde iyileştirme yapmakta bir amaçtır.

3.1.6. Sistemin Devamlılığın Sürdürülmesi

Sistem, organizasyon yapısına yerleştirildikten sonra, devamlılığın sürdürülmesi gerekir. Bu devamlılığın en önemli kısmı, bilgi tabanına yeni gerçeklerin ve kuralların girilmesidir. Bunlar sistem personeli tarafından gerçekleştirilebileceği gibi bilgi mühendisine de gereksinim duyulabilir. Sistem, organizasyon problem hakkında daha çok bilgi edindikçe bilgitabanı genişlemelidir. Nihayetinde, sistem, problemin çözümünde yardımcı olacaktır.

BÖLÜM 4

SONUÇ ÇIKARIM MEKANİZMASI

4.1 Sonuç Çıkarım Prensipleri ve Metodları

Spesifik bir problemin nitelikleri ve spesifik bir alanın genel kuralları verilmiş ise, ne gibi sonuçlar üretilebilir? Sonuçlara ulaşma prosesi, statik bir proses değildir. Bu proses sağlam, doğru çıkarım metodlarının uygulanmasını gerektirir.

Sonuç çıkarım mekanizması, spesifik alan bilgitabanını yeni sonuçlar elde etmek için ele alınan problemin özel gerçeklerine uygular. Prensipler, çıkarım mekanizmasının çalışmasını idare ederler. Bu prensiplerin bazıları, izin verilebilen çıkarsama tipleriyle ilgilidir. En katı çıkarsama prensipleri lojik olarak doğrulanırlar. Yani, biz ilk veriyi doğru (TRUE) olarak vermişsek, prensibin kullanılmasıyla elde edilen sonuçlar da TRUE değerini almalıdır. Bazı çıkarım prensipleri probabilistik veya belirsiz sonuç çıkarımı ile ilgilidir.

Kural bazlı bir sistemde, en yaygın çıkarım prensibi, "Modus ponendo ponens" (veya Modus ponens) tarafından yapılan lojiksel türetimdir. Bu çok eskiden bilinen bir çıkarım prensibidir. Modus ponens'in basit bir örneğini şöyle verebiliriz:

Verilen Kural : If Hava yağmurlu,

Then Yer ıslaktır.

Verilen Gerçek : Hava Yağmurludur.

Çıkarılan Yeni Gerçek: Yer ıslaktır.

Daha az bilinen fakat benzer bir çıkarım "Modus tollendo tollens" (veya modus tollens)'tir. Aynı örneği ele alırsak;

Verilen Kural : If Hava yağmurlu,

Then Yer ıslaktır.

Verilen Gerçek : Yer ıslak değildir.

Çıkarılan Yeni Gerçek: Hava yağmurlu değildir.

Bu modus tollens, çoğu formal sistemlerde, lojik olarak, çıkarımın eşdeğeri şeklinde gösterilebilir. Fakat bununla beraber, bazı uzman sistemler sadece modus ponens'ten faydalanırlar. Bu da, verilen kural ve gerçekler setinden belirli ve faydalı sonuçlar çıkaramadıklarını ifade eder.

Boolean cebiri için yaygın çıkarım prensipleri şekil 4.2. de verilmiştir. (a) ve (b) şıkları daima düşünülen formlardır. (c) ve (d) biraz önemsizdirler. Basit gerçeklerden kompleks önermeler üretirler. (e) şıkkı ise kararlılık (resolution) prensibi olarak bilinir. ve kompleks önermeler setinden basit önermeler çıkarmakta kullanılır. (f) ise, iki ifade arasında bulunan zıtlığı elimine eder ve kompleks önermelerden basit önermeler elde eder. Bütün uzman sistemler çıkarım prosesinde bunların bazılarını veya hepsini ya da eşdeğerlerin kullanacaktır.

Uzman sistemlerde kullanılan birçok sonuç çıkarım prensipleri vardır. Bunlardan en önemli üç prensip şunlardır:

* Modus ponens

* Kararlılık (Resolution)

* Belirsizlik hakkında sonuç çıkarım

(Reasoning about uncertainty)

Prensipler		Örnekler
(a) Kural	if A then B	if yaratığın tüyleri varsa then yaratık bir kuştur.
Gerçek	A	yaratığın tüyleri vardır.
Yeni Gerçek	B	yaratık kuştur.
(b) Kural	if A then B	if yaratığın tüyleri varsa then yaratık bir kuştur.
Gerçek	not B	yaratık bir kuş değildir.
Yeni Gerçek	not A	yaratığın tüyleri yoktur.
(c) Gerçek	A	o bir balıktır.
Gerçek	B	ağırlığı 10 kg'dır.
Yeni Gerçek	A and B	o bir balıktır ve ağırlığı 10 kg'dır.
(d) Gerçek	A	yağmur yağıyor.
Yeni Gerçek	A or B	yağmur yağıyor veya kar yağıyor.
(e) Gerçek	(not A) or B	ya solungaçları yoktur veya o bir kuştur.
Gerçek	(not B) or C	ya o bir balık değildir veya o suda yaşar.
Yeni Gerçek	(not A) or C	ya solungaçları yoktur veya o suda yaşar.

(f) Gerçek (not A) or B ya solungaçları yoktur veya

Gerçek A o bir balıktır.

solungaçları vardır.

Yeni Gerçek B o bir balıktır.

Şekil 4.2. Boolean cebiri için bazı çıkarım prensipleri a) Modus ponens (b) Modus tollens (c) birleştirme (d) karşıtını birleştirme (e) Kararlılık (d) tezatlığı elimine etme

4.1.1. Modus Ponens

Bilgi sistemlerinde kullanılan en yaygın çıkarım stratejisi "modus ponens" olarak bilinen lojik-kural yapısına dayanan bir uygulamadır. Bu prensibe çok yakından bağlantılı olan diğer bir prensip "modus tollens"tir. Çoğu uzman sistemlerin anlaşılmasında bu prensipler temel teşkil etmektedirler. Bu temel teşkil etme özellikleri, if-then kurallarına bu prensiplerin uygulanmasıyla belli bir alandaki bazı çıkarımların en iyi bir şekilde elde edilmesinden kaynaklanmaktadır. Modus ponens, ayrıca, insanların sonuç çıkarım şeklini de yansıtır.

Sonuç çıkarım prensiplerindeki en önemli katı özellik, prensiplerin doğru şartlara uygulandığında doğru sonuçlara ulaşmasıdır. Sonuç çıkarımla ilgili bazı prensipler ve önermeler şekil 4.2'de verilmiştir. Acaba bunların gerekli olan bu katı özelliği (doğru sonuca ulaşma özelliği) gerçekleyeceğinden nasıl emin olabiliriz? Bunu cevaplamak için, lojik bağlantıların (and, or, not) manaları üzerinde durmak gerekir.

Bu konu ile ilgili DOĞRU/YANLIŞ (TRUE/FALSE-T(F) kullanımı şekil 4.3 (a) da verilmiştir. Şekilde yer alan tablolar, önce A ve B'nin birlikte T ve F kombinasyonlarına bakılır, daha sonra lojik bağlantıyla birleştirilen A ve B'nin değerinin saptanmasıyla elde edilmişlerdir. Örneğin;

if A$\longrightarrow$T and B$\longrightarrow$F then "A or B" $\longrightarrow$T değerini alır.

Şekildeki tablo (a), (if-then', 'or' ve 'not' lojik ifadelerin manalarını tanımlamaktadır. (b) ise, modus ponens'in neden geçerli bir çıkarım prensibi olduğunu gösterir. Burada doğruluk tabloları, herbir şart ve sonuca T veya F değerini atamak için kullanılır. A be B'ye mümkün olabilen tüm T ve F kombinasyonları atanır. Daha sonra, doğruluk değeri herbir şart ve sonuç için hesaplanır. Gerekli olan özellik, "doğru şartlar doğru sonuçları verir"dir. Şekildeki tabloda yer alan (*) işareti, bulunduğu satırda tüm şartların doğru olduğu durumu gösterir. Burada sonuç TRUE'dür.

a)

A	B	if A then B	A and B	A or B	not A	not A or B
T	T	T	T	T	F	T
T	F	F	F	T	F	F
F	T	T	F	T	T	T
F	F	F	F	F	T	T

b)

A	B	if A then B	A	B
* T	T	T	T	T
T	F	F	T	F
F	T	T	F	T
F	F	F	F	F

c)

A	B	C	not A or B	not B or C	not A or C
*T	T	T	T	T	T
T	T	F	T	F	F
T	F	T	F	T	T
T	F	F	F	T	F
*F	T	T	T	T	T
F	T	F	T	F	T
*F	F	T	T	T	T
*F	F	F	T	T	T

Şekil 4.3 Doğruluk tabloları a) Lojik bağlayıcılar için bir doğruluk tablosu.b) modus ponens için doğruluk tablosu. c) Kararlılık prensibi için doğrulukl tablosu.

4.1.2. Kararlılık (resolution)

Şekil 4.2. de kararlılık prensibi verilmiştir. Şekil 4.3. (c) de bunun doğru bir çıkarım prensibi olacağı gösterilmektedir. Yine (*) ile işaretlenen satırlar, şartları doğru (T) olan durumları temsil etmektedir. Bu satırların herbirinde sonuç TRUE'dür.

Kararlılık metodu, bir önermenin if-then kuralları ve diğer gerçekler seti tarafından kanıtlanıp kanıtlanmayacağı bulmada sistematik bir yöntemdir. Bu metodun temeli kararlılık prensibidir. Pratikteki basit bir örneği üzerinde duralım. Bunun için aşağıdaki lojik ifadelerin varsayıldığını düşünelim:

(1) if Hava yağmurlu then Yol kaygandır.

if A then B

(2) if Yol kaygandır. then Araba kazaları artar.

if B then C

(3) Hava yağmurludur.

A

olduğunu ve aşağıdaki gerçeğin doğru olup olmadığını çıkarmaya çalıştığımızı arzu edelim:

(4) Araba kazaları artar.

C

Kararlılık metodu dört aşamadan oluşur:

– İlk olarak, tüm if-then durumları -or- durumları olarak tekrar yazılır.

if A then B ile (not A) or B

ifadelerinin aynı olduğu şekil 4.3 (a) da gösterildi. Böylece (1) ve (2) lojik ifadeler aşağıdaki forma dönüşür:

(5) Ya hava yağmurlu değildir veya yol kaygandır.

not A or B

(6) Ya yol kaygan değildir veya araba kazaları artar.

not B or C

– ikinci aşama olarak; önermenin negatifi test altına alınır. Yani önerme (4)'ün negatifinin doğru olduğu kabul edilir:

(7) Araba kazaları artmaz.

not C

– Üçüncü aşama olarak; (5) ve (6) nolu önermelere kararlılık prensibi uygulanır. (Bak: şekil 4.2 (c) ve (8) elde edilir:

(8) Ya hava yağmurlu değildir veya not A or

Araba kazaları artar. C

Şimdi tezatlığı elimine etme prensibini (Bak: şekil 4.2. (f) (3) ve (8) nolu ifadelere uygulayalım ve araba kazalarının arttığını ispatlamaya çalışalım. (3) nolu ifadeyi tekrar alalım. Sonuçta (9) nolu ifade elde edilir:

(3) Hava yağmurludur. A

(9) Araba kazaları artar. C

– son aşama da, (9) ve (7) arasında bir çelişmenin varlığı gösterilir:

Araba kazaları artar ve araba kazaları artmaz.

C and not C

Dolayısıyla (7)'deki varsayım yanlış (F) olmalıdır.

Yani;

Araba kazaları artar. C kanıtlanmış olur.

Kararlılık metodu daima bu dört aşama stratejisini izler. Üçüncü aşama çok sayıda ve kompleks olabilir. Uzman sistemler açısından bu metod aşağıdaki özelliklerden dolayı çok önemlidir:

(a) kolaylıkla otomatikleştirilebilir ve

(b) Gerçekler ve kurallar setinden hareket ederek hedeflere ulaşmaya çalışan uzman sistemler, ulaşılacak hedefin negatifinin verilmesi ve bu metodun kullanılmasıyla kolaylıkla amacına ulaşabilir.

Kararlılık metodu, çok karmaşık olmasına rağmen predicate hesabında çok iyi bir şekilde ifade edilebilir. Predicate lojik yapı, bilgi-tabanı için standart gösterim tekniklerinden birisi olduğundan metodun bu özelliği çok önem taşır.

4.1.3. Belirsizlik Hakkında Sonuç Çıkarma

Şimdiye kadar ele aldığımız prensip ve metodlar, tam belirlilik altında doğruluk ve yanlışlık durumlarıyla ilgilidir. Daima doğru bir önermeden doğru bir sonuca ulaştıran prensiplerle ilgilidirler. Fakat, insan uzmanlığının çoğu belirsizlik altında yapılan çıkarımlarla ilgilidir. Bu belirsizlik birçok şekilde olabilmektedir. Bazen:

- önermelerin tam belirlilik altında bilinmediği,
- sonuçların tam belirlilik altında çıkartılamadığı,
- kuralların kendi aralarında belirli olmadığı durumlar olabilmektedir. Zaten, belirsizlik olduğu için insanlar uzman kullanmaya ihtiyaç duymaktadırlar.

Klasik programlamada, program aşamasından önce, tüm gerekli bilgilerin elde edilmiş olması gerekir. Bilgi programlamada ise bu durum o kadar gerekli değildir. Bu programlamada, bazı bilgilerin unutulduğu veya bilinmediği durumlarla ilgilenilmektedir. Dolayısıyla çıkarım mekanizmasının bu eksik bilgiyi işleme özelliğine sahip olması gerekir. Bilinmeyen bilginin işlenmesi, ancak kuralların şartlarını değerlendirmekte gerekli olan bilginin bulunmadığı durumlarda kuralları geçersiz kılmakla mümkündür. Buradan, sonucun tamamen şartın yapısına bağlı olduğu ortaya çıkar. Eğer bir şartta if cümlecikleri birbirlerine AND-lojik deyimi ile bağlanmışsa, tüm bu cümlecikler TRUE değerini almaları gerekir ki kural başarıyla gerçekleşsin. Bu durumda, eğer kullanıcı şartın herhangi bir kısmına "bilinmiyor" cevabını verse, kural geçersiz olacaktır. Eğer şart cümlecikleri birbirleriyle OR-lojik deyimiyle bağlanmış olsa, bilinmeyen bir bilgi kuralın başarılı olarak gerçekleşmesine engel olmaz. Yani, kuralın şart bölümündeki cümleye kullanıcı "bilinmiyor" dese bile kural başarılmış olabilir.

Bu belirsizlik hakkında sonuç çıkarmada üç durumdan bahsetmek mümkündür. If A Then B varsayımımız üzerinde bu üç durumu belirtmeye çalışalım:

(1) A'nın doğru olup olmamasında bir belirsizlik sözkonusu olabilir.

(2) A şartı tam belirli olabilir. Fakat sonuç kısmının oluşması konusunda belirsizlik olabilir.

(3) Varsayımın doğru olup olmamasında bir belirsizlik olabilir.

Bu üç durum için belirsizliği içeren gerekli format şekli şöyledir:

(1) muhtemelen A probably A

(2) if A then muhtemelen B if A then probably B

(3) muhtemelen (if A then B) probably (if A then B)

Aynı zamanda, bir kural yapısında bu durumların hepside bulunabilir. Daha önce verdiğimiz örneği bu üç duruma adapte etmeye çalışalım:

(1) if Hava yağmurludur,

then muhtemelen yol kaygandır.

if A then muhtemelen B

(2) if Yol kaygandır,

then muhtemelen kazalar artar.

if B then muhtemelen C

(3) Hava yağmurludur. A, Böylece;

(4) muhtemelen kazalar artar. muhtemelen C.

Uzman sistemlerde, belirsizlik konusunda önemli olan iki husus vardır. İlki: belirsiz bilginin nasıl temsil edileceği, ikincisi; sonuç çıkarımda belirsizliğin nasıl işleneceğidir. Belirsizliğin temsil edildiği yöntem, aynı zamanda, sonuç çıkarmada kullanılacak yöntem

üzerinde etki edecektir. Belirsizliği temsil etme yöntemlerinden biri olan ve bu konuda önem taşıyan belirlilik faktörlerinin kullanımıdır.

4.1.4. Belirlilik Faktörleri

Bu metod MYCIN ve diğer birçok uzman sistemlerde kullanılan metodtur. Her bir önermeye (veya bir nesnenin bir niteliğine verilen değere) -1, +1 arasında değişen gerçek sayılarda bir belirlilik faktörü atanır. -1 belirlilik faktörü önermenin sağlanamaz olduğunu gösterir. Yani tam belirsizdir. + 1 de önerme sağlanır. O ise önemsememeyi gösterir.

Belirsizlik faktörlerini kullanarak çıkarımın yapılabileceği farklı yöntemler vardır. Bu metodlardan biri, aşağıdaki işlemleri gerektiri:

– Eğer önermeler birbirine .and. ile bağlanmışsa, iki önermeden oluşan bir ifadenin belirlilik faktörü, bu iki önermenin belirlilik faktörlerinin minumum olanına eşit olacaktır.

– Eğer önermeler .or. ile bağlanmışsa, ifadenin belirlilik faktörü, ifadeyi oluşturan önermelerin belirlilik faktörlerinin maksimum olanına eşit olacaktır.

– if-then kuralının sonucunun belirliliği ise, sonucun belirliliği ile şart kısmın belirliliğinin çarpımı olacaktır.

Kural, normalde, şart kısmıyla ilgili başlangıç belirlilik faktörüne sahiptir. Bu belirlilik faktörünün altında kural başarılmayacaktır. MYCIN'da bu belirlilik faktörünün başlangıç değeri o.2 olarak alınmıştır.

Bu açıklamaları bir örnek üzerinde ifade edelim. Aşağıdaki örneği ele alalım:

A (CF = 0.6)

B (CF = 0.4)

if A and B then C (CF = 0.9)

Buna göre,

A and B (CF = min (0.6, 0.4) = 0.4)

C (CF = 0.9 * 0.4 = 0.36) olur.

4.1.5. Gereklilik ve Yeterlilik Faktörleri

MYCIN'nın kurallarındaki belirlilik faktörleri, şartın bir sonuç için ne kadar yeterli olduğunu gösterir. Uzmanlığın birçok alanlarında, bir sonuç için bir şartın olmayışı sonucun gerçekleşemeyeceği şeklinde kabul edilir. Dolayısıyla, önermenin ilk kısmını oluşturan şart kısmı, sonuç için gerekli olmaktadır. Yani, yeterlilik faktörünün yanısıra gereklilik faktörüde sözkonusudur. Bir uzman sistem olan PROSPECTOR her iki faktörü de içeren kurallara sahiptir.

Belirsizlik alanlarında diğer bir yaklaşım da bulanık kümelerin kullanımıdır. Bu konuya girilmeyecektir.

4.2. Çözüm Arama Teknikleri

Çıkarım mekanizmasının ikinci bileşeni, çıkarım kontrol mekanızmasıdır. Bu mekanizmanın işleyişinde birçok kontrol stratejileri kullanılır. Bu çıkarım kontrol stratejileri açısından önemli olan bir husus kullanacakları arama teknikleridir. Dolayısıyla, çıkarım kontrol stratejilerine girmeden önce arama tekniklerini incelemek gerekir.

Mümkün bir çözüm bulmak için birçok arama teknikleri vardır. En önemli ve en yaygın kullanılan tekniklerden sadece ikisi üzerinde duracağız. Bunlar;

– Önce-Derinlemesine arama tekniği

(Depth-first search technique)

– Önce-Enlemesine arama tekniği

(Breadth-first search technique)

Bir arama tekniğinin performansının değerlendirilmesi çok karmaşık olabilir. Gerçekten de, aramaların değerlendirilmesi YZ çalışmalarının büyük bir kısmını oluşturmaktadır. Bununla beraber, bu tekniklerin değerlendirilmesinde en önemli iki temel ölçü vardır:

(1) Arama tekniğinin bir çözüme ne kadar hızlı ulaştığı,

(2) Arama tekniğinin iyi bir çözüme ne kadar hızlı ulaştığı.

Örneğin çözüm yolunun uzunluğu ve ayrık düğümlerin sayısı bir tekniğin hızlılığını saptar. Ölü bir çözümden geri dönüş epey zaman harcayacaktır. Arzu edilen teknik, az geri dönüş yapan teknik olacaktır.

Diğer önemli bir husus, optimal bir çözüm bulmakla iyi bir çözüm bulmak arasında bir farklılığın varolduğunu bilmektir. Optimal bir çözüm bulmak için geniş bir arama sözkonusudur. Oysa iyi bir çözüm bulmak demek, sınırlar arasında kalan bir çözüm bulmak demektir. Bu durumda daha iyi bir çözümün olup olmadığı önemli değildir.

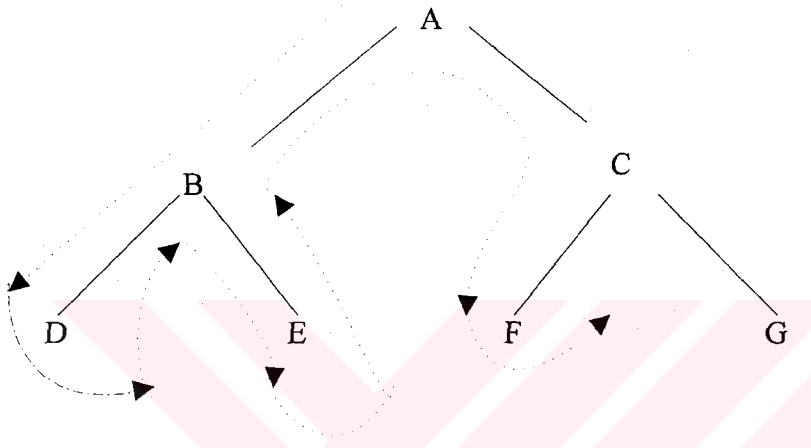
Son olarak, bir metodun diğer bir metodtan daha iyi olduğunu söylemek zordur. Herbir tekniğin diğerinden daha iyi çalıştığı belirli spesifik durumları vardır. Şunu da bilmekte fayda var: problemi tanımlama şekli bazen uygun bir arama yöntemini seçmekte yardımcı olacaktır.

4.2.1. Önce-Derinlemesine Arama Tekniği

Bu arama tekniğinde, amaca giden herbir mümkün rotanın sonucuna gidilir. Daha sonra diğer rota incelenmeye çalışılır. Bunu daha iyi anlamak için şekil4.4.tekiağaç yapısını inceleyelim. F, bizim hedefimizi göstersin.

Önce derinlemesine arama tekniği A B D E A C F'nin sırasına göre hareket edecektir. Görüldüğü gibi, bu arama tekniğinin, hedefe ulaşacağı kesindir. Çünkü geniş bir arama sözkonusudur. En kötü durum G'nin amaç olma durumudur. Eğer G amaç olsaydı, teknik

tüm düğümleri taramış olacaktır. Önce derinlemesine arama tekniği, sezgisel ifadeler kullanılmadığı zaman izlenebilen en iyi yöntem olabilir. Bu da Turbo Prolog'un kullandığı arama tekniğinin bu olduğu ortaya çıkmaktadır.



Şekil 4.4 Önce derinlemesine arama tekniği

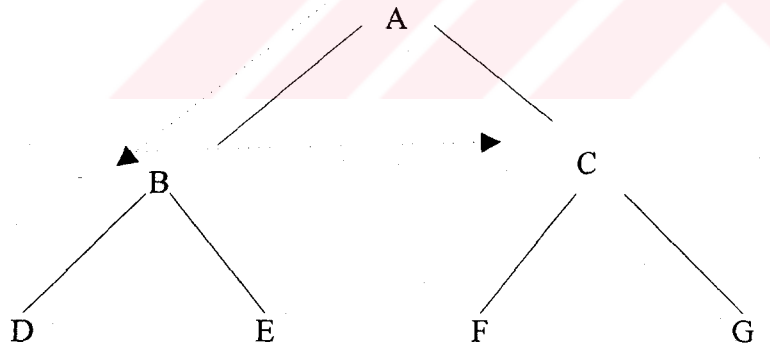
4.2.2. Önce-Enlemesine Arama Tekniği

Önce enlemesine arama tekniği, önce derinlemesine arama tekniğinin zıttıdır. Bu yöntem, başlama noktasından bir kademe aşağıya inmekte ve bu kademedeki tüm düğümleri incelemektedir. Eğer amaca ulaşılmazsa bir sonraki kademeye inmekte ve oradaki düğümleri incelemeye başlamaktadır. Şekil 4.5'te hedefimizin C olduğunu varsayalım.

Bu gösterimde, A B C düğümlerinin incelendiği görülmektedir. Bu teknik te, diğer teknikte olduğu gibi, bir çözüm bulacağı muhakkaktır. Eğer bir çözüm varsa, geniş aramayla onu mutlaka bulacaktır.

İncelediğimiz her iki tekniği, katı ve kör teknik olarak tanımlayabiliriz. Her yöntem de, izlenen yolun doğru olup olmadığı konusunda herhangi bir tahmin yapmamaktadır. Her ikisinde de, belli bir düğümden diğer bir düğüme belli bir hiyerarşik yapıya göre hareket etmeye dayanarak bir çözüm arama vardır. Akla şöyle bir soru gelebilir: Acaba bu iki tekniği geliştirmenin yolu yokmudur?

Bilindiği gibi, aramanın doğru bir yönde ilerlemesinde ihtimaliyeti artıran kurallar vardır. Bu basit kurallara "sezgisel kurallar" veya "höristikler" adı verilir. Spesifik uygulamalar için hazırlanmış bilgi bazlı programlara bu kuralları eklemek mümkün olabilir. Ancak genel amaçlı sezgisel arama teknikleri oluşturmak imkansızdır. Çoğu höristik arama teknikleri, problemin bazı bölümlerinin minimizasyonu veya maksimizasyonu üzerine dayandırılmıştır.



Şekil 4-5 Önce enlemesine arama tekniği

4.3. Çıkarım Kontrol Stratejileri

Bir çıkarım yapmakta kullanılan prensip veya metod ya yeni bir gerçeği verecek veya probabilistik bir sonuca ulaşacaktır. Uzman sistem, danışmanlık görevini gerçekleştirmesi için birçok çıkarımlar yapmak zorundadır. Bu çıkarımlar gelişigüzel (random) yapılamaz. Sistem çıkarımda bulunmak için bilgi alanını araştıracaktır. Bu araştırmada bilgi-tabanı taranırken hangi aşamada hangi çıkarımların yapılacağına karar vermek için bazı model veya stratejilere gereksinim vardır. İşte çıkarım kontrol stratejilerinin amacı bunu sağlamaktır. Kullanılan strateji uzman sistemin performans özelliklerini saptayacaktır.

"Belirlilik" ve "Belirsizlik" gibi iki belirgin kategorinin ötesinde çıkarım mekanizmasını oluşturan iki temel kontrol yöntemi vardır. Bu yöntemler:

* İleriye dönük zincirleme yöntemi

(Farward chaining method)

* Geriye dönük zincirleme yöntemi

(Backward chaining method)

Bu yöntemlerin farklılıkları, mekanizmanın amaca nasıl ulaşmaya çalıştığıyla ilgilidir. Dolayısıyla, bu yöntemler, çıkarmalar yapmak için izleyecekleri rota açısından farklılaşırlar.

4.3.1. İleriye Dönük Zincirleme Metod

Çıkarım mekanizması, hedef olarak kabul edilen bir terminal noktasına ulaşana kadar, bir şebekenin mantıksal AND'leri ve OR'ları doğrultusunda ilerler. Bu ilerleme

işleminde, kullanıcılar tarafından sağlanan bilgiyi kullanır. Bundan dolayı, bu metoda bazen veri-kaynaklı metod (data-driven method) ta denir. Çıkarım mekanizması mevcut bilgiyi kullanarak bir hedefi bulamazsa daha fazla bilgi isteyecektir. Nesneyi veya hedefi tanımlayan kurallar kendisine rehberlik eden yolu oluşturacaktır. Dolayısıyla, amaca ulaştıracak olan yöntem, kuralların tümünü tatmin etmesi gerekir. Böylece, Böylece, ileriye dönük metodla, çıkarım mekanizması bazı bilgilerle çalışmaya başlayacak ve bu bilgiye uyan bir amacı bulmaya çalışacaktır.

İleriye dönük zincirleme metodunun nasıl çalıştığını bir örnekle anlatmaya çalışalım. Arabamızın çalışma sisteminde bir problemin varolduğunu ve problem konusunda bir fikir sahibi olmak için bu konuda uzman olan bir araba teknisyenine telefon ettiğimizi varsayalım. Teknisyen bizden iyi gitmeyen bu durumu tanımlamamızı sormaktadır. Bizde aşağıdaki açıklamaları yapmaktayız:

- arabamızın güç kaybettiğini,
- kilitleme yaptığını,
- ateşleme sistemini kapatmamıza rağmen motorun çalışmasına devam ettiğini ve

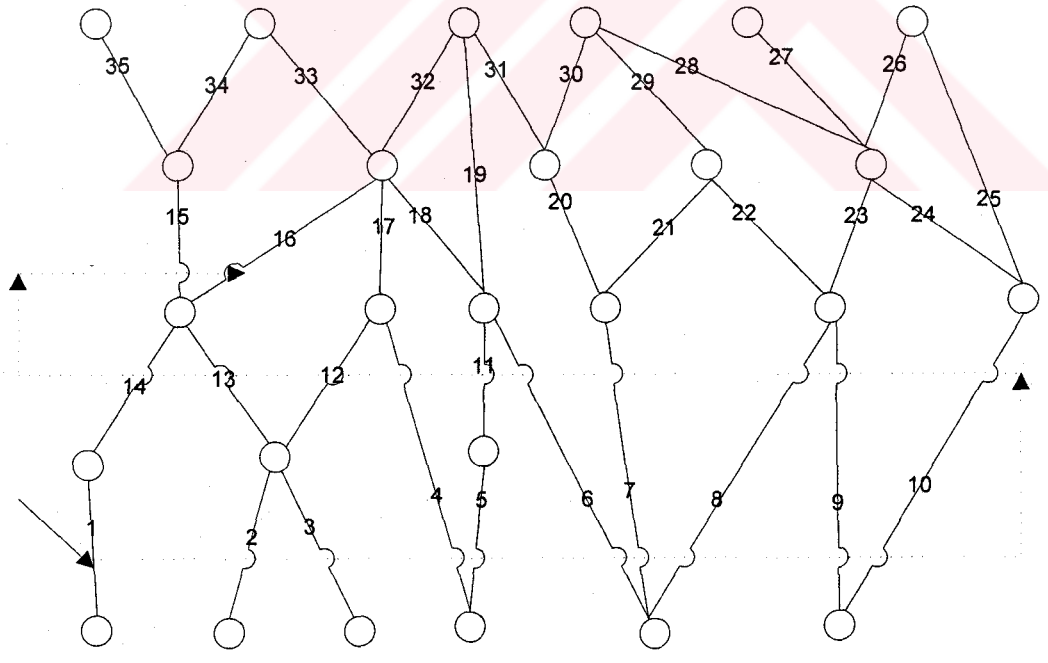
aylardır arabanın çalıştırılmadığını bildirmekteyiz. Teknisyenimiz büyük bir ihtimalle, bu bilgileri kullanarak, arabamızın ayarlanmaya ihtiyacı olduğunu söyleyecektir.

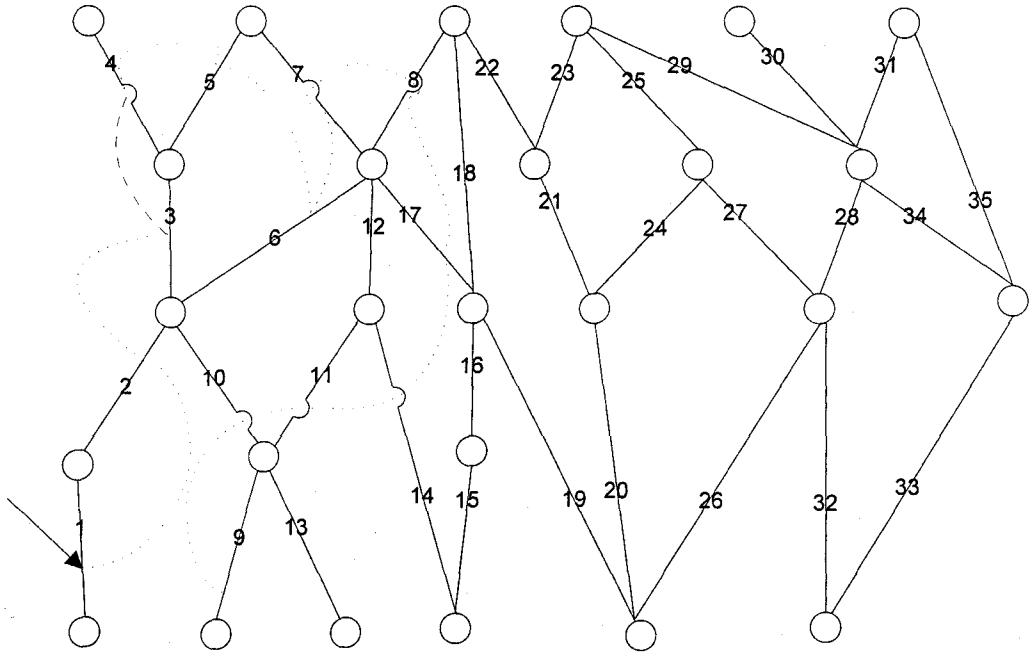
Bu tekniği daha genel bir şekilde açıklamaya çalışalım. İleriye dönük zincirleme metodu, bilgitabanında tanımlanmış gerçekleri alır, kuralları kullanarak yeni gerçekler bulmaya çalışır. Elde edilen yeni gerçekler daha sonra spesifik bilgi tabanına eklenir ve strateji başka yeni gerçekler bulmaya çalışır. Veri kaynaklı metod denmesinin sebebi de budur. Dolayısıyla bu işlemleri yapan bu strateji, modus ponens uygulamalarında çok iyi işlemektedir. Herhangi bir anda, birden fazla sonuç elde etmek için birden fazla kural kullanmak mümkün olabilir. Çıkarım kontrol stratejisi, karmaşıklığı çözmek için, test altına alınan kurallar arasından karar vermelidir. Bunu yapmanın basit bir yolu, kuralları

nmaralamak ve en düşük numaralanmış kuraldan başlamaktır. Çok karmaşık karar teknikleri daha büyük sistemlerde kullanılır.

Şekil 4.6 da, kurallar, şartlar ve sonuçlar arasındaki bağlantı, bilginin bir ara gösterimi kullanılarak gösterilmiştir. Sistem çalışmaya başlamadan önce, daha düşük numaralı düğüm/düğümleri saptamalıdır. Kurallar şekil 4.6 (a) da olduğu gibi numaralanmış ve kontrol tekniği tatmin edilebilen en düşük numaralı kuralı seçmek ise, sistem kural tabanını önce enlemesine arama tekniğiyle taramaya başlayacaktır. Diğer bir yandan, kurallar şekil 4.6 (b) deki gibi numaralanmış ise kural tabanı önce derinlemesine arama tekniğiyle taranacaktır.

İleriye dönük çıkarım, problemin çözümlerin sayısının yönetilemediği -yani çok miktarda çözümün bulunduğu- fakat problemin ilk durumunu tanımlayan veri parçalarının sayısının çok olmadığı durumlarda kullanılır. Strateji, verilerden çalışmaya başlamak ve uygun sonuca doğru gitmektir. Uygulamada bu özellikteki uzman sistemler, birkaç temel kısıtlayıcıların verildiği planlamayla ilgili sistemlerdir.





b)

Şekil 4.6 İleriye doğru zincirleme metod (a) Önce enlemesine, (b) Önce derinlemesine

4.3.2. Geriye Dönük Zincirleme Metod

Bu metod ileriye dönük zincirleme metodunun tam tersidir. Bu metodla çıkarım genellikle bir hipotez veya nesne ile başlamakta ve bunu desteklemek veya reddetmek için bilgi isteyecektir. Örneğin, Turbo Prolog örnek paketinde mevcut olan basit hayvan (animal) uzman sistemi bu metodu kullanır. Bu metoda bazen amaç veya nesne kaynaklı metodta denilmektedir. Çünkü bir nesne ile başlamakta ve onu kanıtlamaya çalışmaktadır.

Bu metodun nasıl çalıştığını anlamak için, bilgisayarımızın aniden durduğunu varsayalım. İlk hipotezimiz güç (enerji)'nin gittiği şeklinde olsun. Bunu kontrol etmek için, vantilatörü dinleriz. Vantilatorun çalışmasını duyarsak, bu hipotezi reddeder ve başka birini düşünürüz. İkinci hipotezimiz yaazılım hatasından dolayı durduğu olsun. Bu ihtimalin doğru veya yanlış olduğunu görmek için, bilgisayarı kapatıp yeniden açarız. Sistemi okuyup

bilgisayarın çalıştığını görürsek ikinci hipotezimiz doğru çıkar. Böylece problem çözülmüş olur.

Geriye dönük zincirleme metodu, şimdiye kadar en çok bilinen çıkarım kontrol stratejisidir. MYCIN, PROSPECTOR ve tüm temel sistemlerde bu metod kullanılır. Bu metotta hedef seçilmekte ve sistem bu hedefi kanıtlamak için gerekli gerçekleri oluşturmaya çalışır. Gerekli gerçeklerin herbiri bir alt amaç olmakta ve sistem bu sefer bu alt amacı kanıtlayan gerçekleri oluşturmaya çalışır. Bu işlem temel bir önerme bulunana kadar devam eder. Temel bir önerme, başka hiçbir gerçeğe bağlı olmayan bir alt-alt-... amaç olacaktır.

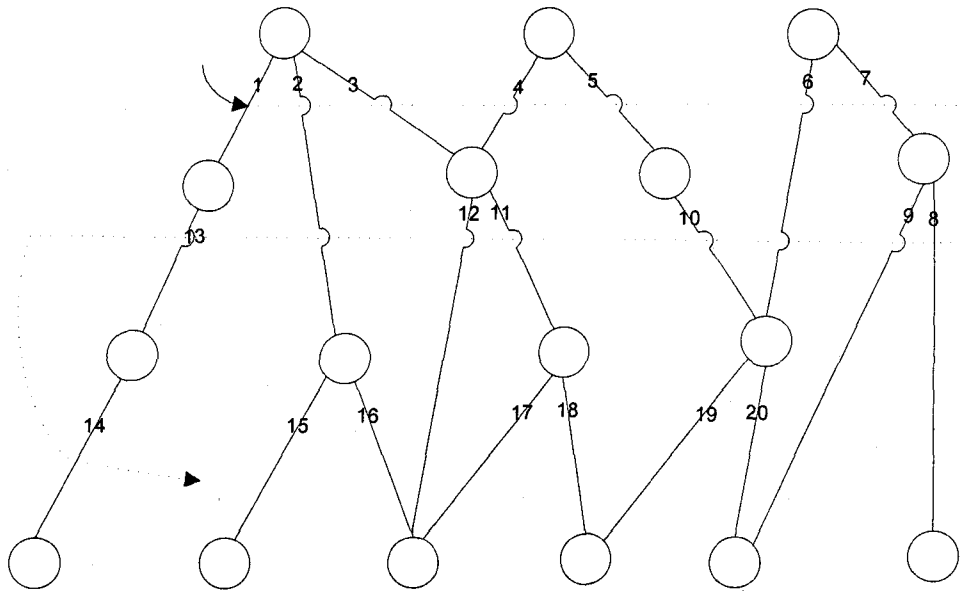
Burada üç durumdan bahsetmek mümkündür:

– Gerekli olan önerme veritabanının içindedir. sistem, başlangıç hedefi kanıtlamak için gerekli olan diğer gerçekleri, eğer mümkünse, bu veritabanının içine yerleştirecektir;

– Önermenin negatifi veritabanının içindedir. Sistem bu durumda, başlangıç hedefi, eğer mümkünse, farklı bir rotadan kanıtlamak zorunda olacaktır.

– Ne önermenin kendisi ne de negatifi kurulmamıştır. Yani yapılaştırılmamıştır. Bu son durumda, uzman sistemin kullanıcıdan bilgi alış-verişi yapması normaldir. Sağlanan cevaplar, daha sonra, çıkarımda kullanılabilir.

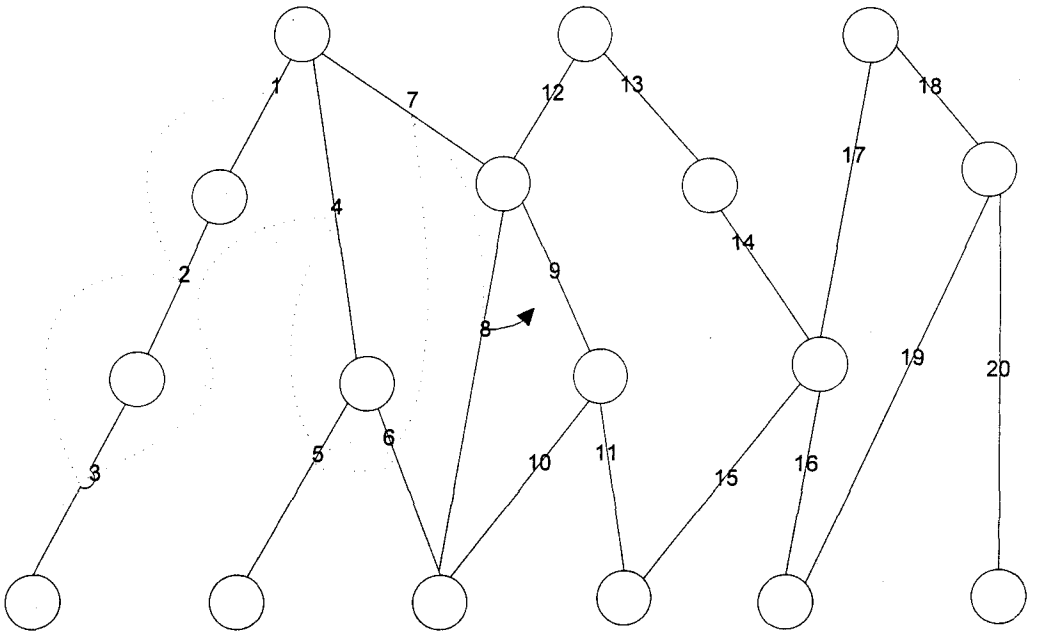
Önce enlemesine ve önce derinlemesine arama teknikleri arasında bir farklılığın bulunduğunu tekrar irdeleyelim. Bu durum, şekil 4.3'da gösterilmiştir. Önce derinlemesine stratejide, bir hedef seçilmekte ve bu hedefin alt hedefleri saptanmaya çalışılmaktadır. ana hedefi oluşturmak için alt hedelerin gerekli olduğu kabul edilir. Önce enlemesine stratejide ise, daha detaylı önermeler kurulmadan önce, bir aşamada bir çok hedef gözönünde bulundurulur.



Şekil 4.7.a) Önce enlemesine geriye doğru zincirleme metodu

Çoğu geriye dönük zincirleme yöntemini kullanan uzman sistemler önce derinlemesine arama stratejisini kullanır. Bu stratejiye göre, sistem bir hipotezi ele alır ve hipotezi kabul veya reddetmek için detaylı bilgiyi gerektirir. Eğer hipotez kanıtlanmazsa, başka bir hipotezi ele alır. Önce enlemesine stratejide, sistem farklı hipotezler arasında atlamalar yapar. Dolayısıyla, daha spesifik soruya geçmeden önce, genel bir seviyede sorular sorar.

Bir uzman ile -örneğin bir doktor- yapılan bir görüşmede, hasta, başlangıç aşamasında şu bilgiyi verir: "Istahsızlığın yanısıra bir haftadır kendimi yorgun hissediyorum ve bir de boğaz ağrısı var". Doktor bu başlangıç bilgiyi kullanarak en uygun bir hedef (burada teşhis) seçer ve geriye dönük çıkarımla bunu kanıtlamaya çalışır. Bunun için, laboratuar testleri yapar, özel sorular sorar, refleksleri kontrol eder,... vb. Bu yaklaşım şekli oldukça çok iyi bilinmektedir.



Şekil 4.7.b) Önce derinlemesine geriye doğru zincirleme metodu

Bu ve buna benzer örnekler başlangıçta ileriye dönük zincirleme metodu daha sonra geriye dönük zincirleme metodu kullanacak şekilde karakterize edilebilirler. Modern uzman sistemlerde, bu iki çıkarım kontrol stratejilerini birlikte kullanmak yaygınlaşmıştır. PROSPECTOR'un MYCIN'a göre avantajlarından biri, bu karışık stratejiyi kullanmasıdır.

BÖLÜM 5
MONTAJ HATLARINDA KAYNAK YERLEŞİMİNDE KULLANILAN, KURALLARIN VE ALGORİTMALARIN TAYİNİ İÇİN BİLGİ TABANLI SİSTEMİN OLUŞTURULMASI

5.1 Giriş

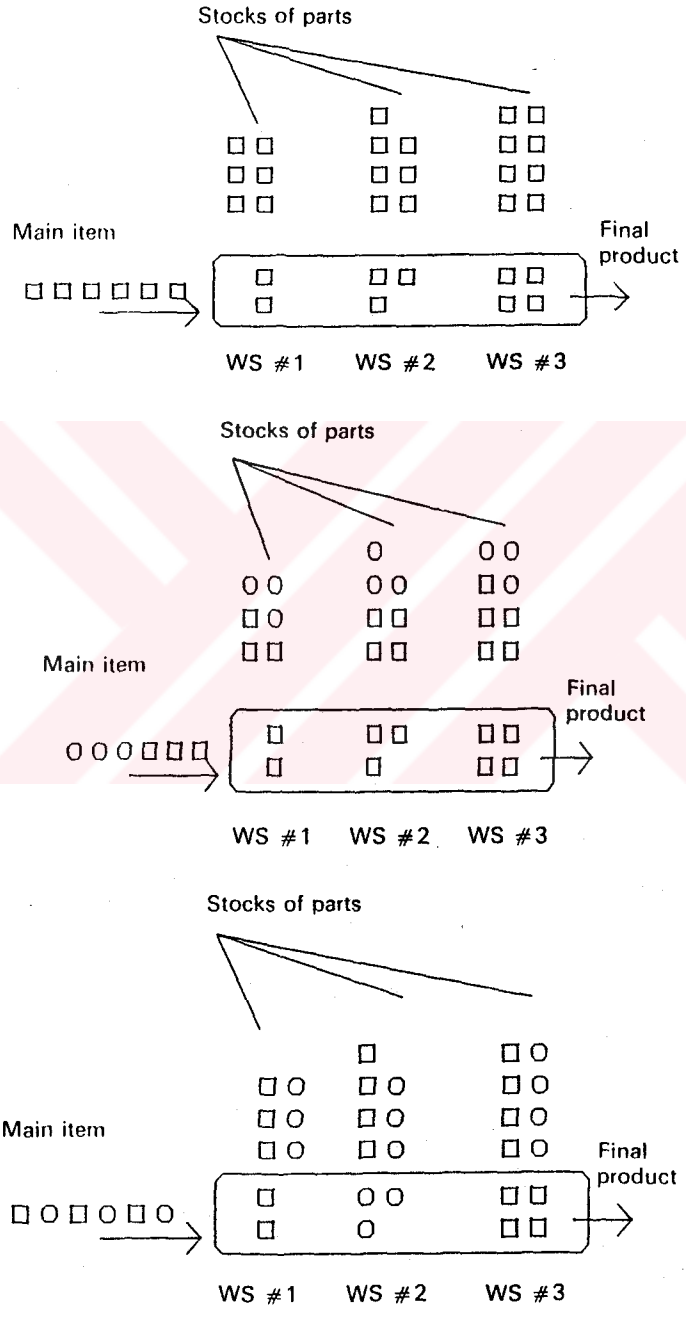
Endüstride montaj hatları 1920'lerin ilk zamanından itibaren kullanıla gelmektedir. Ford, otomotiv endüstrisinde ilk olarak montaj hattını kuran firmadır. Böylelikle fabrikaların verimlilikleri, önceden kullandıkları üretim teknikleri ile karşılaştırıldığı zaman, önemli bir ölçüde arttığı görülmüştür.

Montaj hattı kavramı geniş ölçüde kabul görmüş ve daha sonra diğer endüstri dallarında da kullanılmaya başlanmıştır.

Montaj hattının dizaynı, zamanla artan ürün çeşidi ve müşterilere sunulan opsiyonlar karşısında çok kompleks bir iş olmaya başlamıştır. Kompleks hale gelmeye başlayan montaj hattı dizaynında bu sorunu ortadan kaldırabilmek için çoklu-model (multi-model) ve karışık-model (mixed-model) yaklaşımları kullanılmaya başlanılmıştır. Tek model montaj hatlarında bir hat sadece bir model için kullanılmaktadır (şekil 1-a), çoklu model montaj hattında ise modeller gruplaşır ve yığınlar halinde işlenirler. Karışık model hatlarında ise grubun miktarı modellerin sıralanmasında önemli rol oynamaktadır (şekil 1-c). Karışık model montaj hatları genellikle JIT (Just In Time) felsefesiyle çalışan işletmelerde kullanılır.

Bazı endüstrilerde, makinelerde ve cihazlar için ek bir yatırım gerekeceğinden montaj hattı her bir modele istenmeyebilir. Diğer bazılarında ise her bir model için bir tane hat kurmak çok fleksibil olabilir öyle ki burada kullanılanlar hafif aletler olabileceği gibi robotlar kullanıldığında da çok değişik operasyonlar gerçekleştirilebilmektedir.

Bu şartlar altında hatların ve kullanılacak iş istasyonların sayısında hiç bir kısıtlamaya gidilmemektedir.



Şekil 5-1. Montaj hatlarının operasyonları a) tek-model montaj hattı b) çoklu model montaj hattı c) karışık model montaj hattı

5.2 Bilgi Tabanlı Programlama Sistemleri

Bilgi tabanlı programlama, uzman sistemler için yeni uygulama alanlarından bir tanesidir. Hali hazırda tamamlanmış sitemlerin sayısı çok da fazla değildir. Bununla birlikte araştırmacılar arasında bu alana karşı git gide artan bir ilgi meydana gelmeye başlamıştır. Bir çok bilgi tabanlı programlama sistemleri, simülasyon modülleri, operasyon araştırma teknikleri, programlama algoritmaları gibi değişik alt sistemlerin kombinasyonları olarak geliştirilmiştir.

Miller, programların otomatik üretimi için horistik tabanlı bir sistemi yarı otomatik bir üretim ortamında geliştirdi (1984). Bu yaklaşım olası programların ağaçlarının üretiminde kullanılmıştır. Bruno, Elia ve Laface, fleksibil bir ortamda programlanan parçalar için uzman programlama sistemi geliştirmişlerdir (1986). Ben-Arieh'de otomatik üretim için simülasyon alt sistemli bilgi tabanlı bir uzman sistem önermişlerdir (1986). Shen ve Chang ise fleksibil üretim ortamları için iki adet gerçek zaman programlama algoritmaları geliştirmişlerdir. Bu çalışmada planlama ve programlama fonksiyonlarının entegrasyonunu ve bir algoritmanın verimliliğini artırmak için bilginin kullanılmasını önermişlerdir.

Shaw (1986)' da bir planlama problemini belli sayıda alt problemlere dekompeze eden yapay zeka yaklaşımını tariflemiştir. Bir çok kaynak sınırları kombinasyonları ve horistik kurallar esnek üretim sistemleri (FMS) programlamasında kullanılmıştır. Marton ve Smunt (1986) esnek üretim sistemleri problemlerini çözen 4 kademeli hiyerarşik yapı geliştirmişlerdir. Dagli ve Cihangirli (1988) esnek üretim ortamında programlama amacıyla prototip bir uzman sistem geliştirmişlerdir. Burada programlama, küçük alt problemlere bölerek kolaylaştırılmıştır.

Chang (1985) karar destek sistemi geliştirmiştir. Erchler ve Esquiaral 1986'da mağaza programlama sistemi sunmuşlardır.

O'connar(1984) Digital şirketinde ISA ve IMACS' ı geliştirdiler. ISA müşteri siparişlerini programlama amacıyla kullanılmıştır. IMACS ise kapasite planlaması ve envanter (mal stoku) yönetiminde kullanılmak üzere dizayn edilmiş bir uzman sistemdir. Fukuda,Tekeda ve Hyashi (1986) da üretim kontrolu üzerine bir uzman sistem geliştirmişlerdir. Parunak (1987) YAMS adında yapay zeka metodolijisini anlatan bir planlama ve kontrol sistemi geliştirmiştir.

5.3 Problemin Tarifi

Analiz edilen sistem, montaj operasyonlarından meydana gelmiş olup, tüm bu operasyonlar manuel olarak ya da hafif ve düşük maliyetli diye tabir edilen basit aletler kullanılarak veya robotlar tarafından yapılmaktadır.

Herhangi bir zamanda birden fazla montaj hattı aktif halde olabilir. Hatların ve iş istasyonların sayıları zaman içinde değişebilir. Bununla birlikte hatta bir model atandığı zaman o modelden geride kalan diğer birimlerinden bitirilmesine kadar(o modelin montaj işlemi bitinceye kadar), iş istasyonlarının sayısı (operatörler,robotlar) değişmeden kalır. Bir modelin birden fazla hatta simultane olarak montajına müsaade edilmemektedir.

T = 0 zamanında R adet kaynak ile monte edilecek n tane model bulunmaktadır. Montaj hatlarının sayısı kaynakların adeti ile sınırlandırılmıştır (her bir modele bir kaynak tahsis edebilmek amacıyla). Böylelikle problemi şöyle tarif edebiliriz:

Problem , n adet değişik modeli sistemdeki montaj hatlarının sayısı ve bu montaj hattında bulunan iş istasyonların sayısı T zaman periyodu içinde değişirken, kullanabilir kaynakların sayısı R sabit olduğu durumda montaj hatlarına atamaktır.

Bu sınırlamalar matematiksel olarak aşağıdaki gibi gösterilebilir.

$1 <= NL_t <= R$ $t=1,2,3,.....,T$

$0 <= WS_{i,t} <= R$ $t=1,2,3,.....,T\ i=1,2,3,.....,n$

$$\sum_{i=1}^{n} WS_{i,t} <= R \qquad t= 1,2,3,.....,T$$

Burada T kabul edilen planlama süreci, n modellerin sayısı, R kaynakların toplam sayısı, N_{lt} t anındaki montaj hatlarının sayısını, WS_{it} ise t zamanında i modeline tahsis edilen kaynakların sayısını göstermektedir.

Halihazırda iki durum mevzubahistir:

1. n>R modellerin sayısı kaynakların toplam sayısından daha fazladır. Bu durumda t = 0 anında tüm modeller işleme sokulmayacak demektir. Bazı modeller, diğer modellerin montajı bitip kaynakların müsait duruma geldikten sonra işleme sokulabilmektedir. Bu anlamda, net etkinin çoklu modellerinkine benzer olduğu kabul edilir.

2. n <= r modellerin sayısı, kaynakların toplam sayısından daha az ya da eşittir. Matematiksel olarak, tüm modelleri işlemeye t=0 anında başlanabilir. Bu yüzden aynı anda,değişik modeller üzerinde çalışan bir kaç hat bulunabilir. Tüm ürünleri son haliyle aynı anda ya da çok kısa bir zaman periyodu içinde elde etmek mümkün olacağından, net sonucun karışık model hatlarınınkine benzer olduğu kabul edilebilir. Bununla birlikte, bu durum hat verimliliği dikkate alındığı zaman en iyi alternatif olmayabilir. Gerçek dünya çalışma ortamında çoğu durumlar birinci örnekte temsil edildiği gibidir.

Burada montaj hattının verimliliği adına önerilmiş 6 adet kural bulunmaktadır. Bunlar RESMAX, RESMIN, FMAX, FMIN, TOTMAX ve TOTMIN'dir. Bu kuralların performansı ise iki kriter altında 6 değişik üretim programlama politikası dahilinde değerlendirilmiştir. Üretim programlama politikalarının hedefi mümkün olan en yüksek verimliliği her bir ürün için elde etmektir.

5.4 Performans Ölçüleri

Bu kuralların performansı, programlama araştırmalarında genellikle kullanılan iki kriter dahilinde ölçüldüler.

1. Ortalama akış zamanı: Bir modelin sistemde kalma süresi akış zamanı olarak bilinmektedir. Başka bir deyimle, bir modelin başlangıçtan, montaj tamamlana kadar geçen süre olarak da tanımlanabilir. Tüm modeller için ortalaması hesaplanır ve ölçü olarak kullanılır.

2. Makespan: Son modelin sistemde kalma süresi makespan olarak tariflenir.

5.5 Arka Plan

Bu bölümde ilk olarak, çevrim zamanına karşı hat verimliliği ve çevrim zamanına karşı iş istasyonlarının sayısı analiz edilerek, bu incelemenin sonucuna dayanarak öncelik matrisi oluşturulmuştur. Öncelik matrisi, geliştirilen programlama kuralları temel alınarak oluşturulur.

5.5.1 Hat Verimliliğine Karşı Çevrim Zamanı

Çevrim zamanı (cycle time), her iş istasyonunun kendisine atanan operasyonu tamamlayabilmesi için tahsis edilen zamandır. İstasyon zamanı (station time) (yani atanan operasyonu yapabilmek için gerekli zaman) çevrim zamanını geçemez. Şayet tüm istasyonların istasyon zamanı, çevrim zamanına eşitse hattın verimliliği %100 demektir. Bununla birlikte pratikte bu değer genellikle %100 den daha düşüktür.

İstasyon ve hat verimliliği aşağıdaki gibi hesaplanabilir.

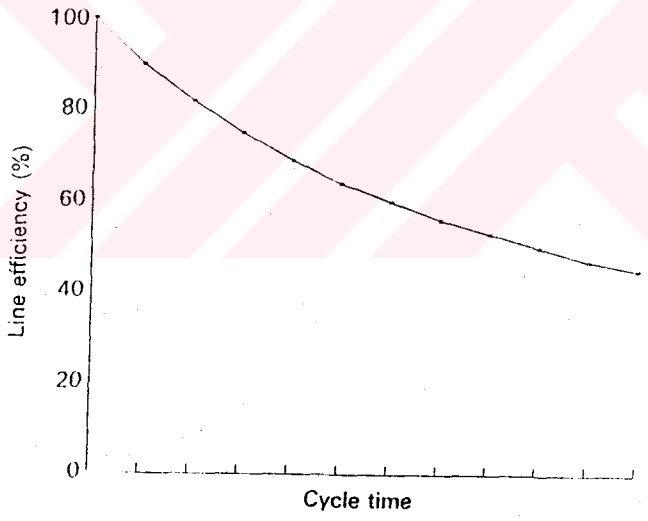
$SE_i = ST_I/CT \quad LE=\Sigma^{ns} SE_I/NS$

Burada SE_i iş istasyonu i nin verimliliğini, St_i aynı istasyonunun istasyon zamanı, CT çevrim zamanı, LE hat verimliliği ve ns iş istasyonlarının sayısını belirtmektedir. Şekil 2'de gösterildiği gibi iş istasyonu sayısı sabit olduğu durumda çevrim zamanı artarken hat verimliliği azalmaktadır.

5.5.2 Çevrim Zamanına Karşı iş istasyonlarının Sayısı

Yukarıdaki açıklamalardan da çok net bir şekilde anlaşılacağı üzere, iş istasyonlarının sabit bir değerinde maksimum hat verimliliği minimum çevrim zamanıyla mümkündür. İş istasyonlarının sayısı değiştiği zaman, buna karşılık gelen başka bir en iyi çevrim zamanı değeri bulunur. Bu ilişki şekil 5-3'de gösterilmiştir.

Çevrim zamanı CT_1 olduğu zaman, iş istasyonlarının buna karşılık gelen adeti ise NS_5 'dir. Çevrim zamanı artarken, iş istasyonlarının sayısı aynı kalmaktadır, böylelikle hat verimliliğini çevrim zamanı [CT_2 -e] oluncaya kadar düşürmüş oluruz. Çevrim zamanı CT_2 değerini alır almaz gerekli olan istasyon sayısı NS_4 olur.



Şekil 5-2. Hat verimliliği ile çevrim zamanı arasındaki ilişki

Aynı davranış diğer değerler için de gözlenebilir. Bu yüzden çevrim zamanı ile iş istasyonları arasındaki ilişki basamak fonksiyon şeklindedir. Alternatif hat konfigurasyonları $NS_1, NS_2, \ldots\ldots NS_k$ için en iyi çevrim zamanları (en yüksek hat verimliliği) sırasıyla CT_k, CT_{k-1} ,....., CT_1 dir. Ve bu değeri de konumuzun ilerleyen kısmında açıklayacağımız öncelik matrisini oluşturmada kullanacağız.

5.5.3 Öncelik Matrisi

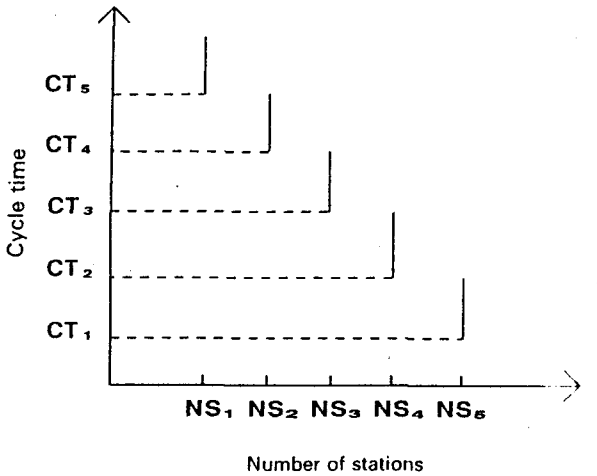
Öncelik matrisi aşağıdaki basamaklar sonucunda oluşur:

1. Tüm i ve j ler için Ct_{ij} bulunur.
2. Tüm i ve j ler için LE_{ij} hesaplanır.
3. Tüm i'ler için verimliliğin azalma sırasına göre Le_{ij} sıralanır $Le_{i[1]} >= Le_{i[2]} \ldots\ldots >= Le_{i[k]}$
4. Tüm i ve j'ler için $Le_{i[j]}$'lere karşılık gelen $Ns_{i[j]}$ değerleri belirlenir.

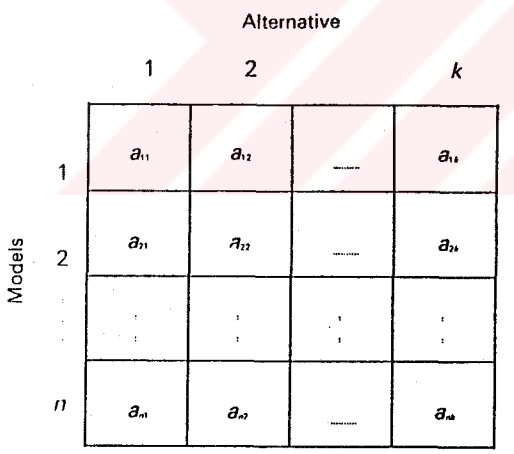
Tüm i ve j ler için $a_{ij} = Ns_{i[j]}$ dir.

Burada k, alternatif hat konfigürasyonlarının sayısını, i modeli için j alternatif kullanıldığında $Le_{i[j]}$ hat verimliliğini, i modeli için j alternatif kullanıldığında Ct_{ij} en iyi çevrim zamanını (en yüksek hat verimliliği), $Le_{i[j]}$ ise i modeli için j.ninci en yüksek hat verimliliğini, $Ns_{i[j]}$ i modeli için j.ninci en yüksek hat verimliliğine karşılık gelen iş istasyonlarının sayısını, a_{ij} öncelik matrisinin elemanını, n ise modellerin sayısını göstermektedir.

Her bir ürün için, istasyonlarının alabileceği en iyi değer (alternatifler arasında en yüksek hat verimliliğine karşılık gelen) öncelik matrisinin birinci sütununda yer almaktadır. Aynı şekilde ikinci iş istasyonun alabileceği en iyi ikinci değer 2. sütunda yer almaktadır. Öncelik matrisinin genel formu şekil 5-4'de gösterilmiştir.



Şekil 5-3. Hat verimliliği ile istasyon sayısı arasındaki ilişki



Şekil 5-4. Öncelik matris.

5.6 Kurallar

Bu çalışmadaki kurallar, modelleri montaj hattına atarken öncelik matrisini göz önüne almıştır.

REMAX KURALI (Maksimum Kaynak)

Seçme kriteri (SC_{ij})bir modeli monte etmek için gerekli olan kaynakların (iş istasyonlarınının) sayısıdır sayısıdır ($Sc_{ij}= a_{ij}$). En yüksek kaynak adetine sahip modellere (en büyük Sc_{ij} değeri) öncelik verilir. Bu kurallar makespanı (yani en son modelin hatta kalma süresi) minimize etmek amacı gözetildiği zaman idealdir.

RESMIN KURALI (Minimum Kaynak)

Seçme kriteri (Sc_{ij}) bir modeli monte etmek için gerekli olan kaynakların (iş istasyonlarınının) sayısıdır ($Sc_{ij}= a_{ij}$). En düşük kaynak kullanan modellere öncelik verilir. Bu kural akış zamanını azaltmak için kullanılır.

FMAX (Hatta maksimum kalma süresi)

Seçme kriteri modelin hatta kalma süresi olup, çevrim süresiyle talebin çarpımı ile hesaplanır ($Sc_{ij}= Ct_{ij}*D_i$).Hatta kalma süresi en yüksek olan modele öncelik veriir Bu kural makespan değerini minimize etmek amacında olduğumuz zaman tercih edilir ve bir ürünü üretmek için gerekli zamanı göz önüne alır.

FMIN (Hatta Minimum Kalma Süresi)

Seçme kriteri modelin hatta kalma süresi olup, çevrim zamanıyla talebin çarpımı ile bulunur($Sc_{ij}= Ct_{ij}*D_i$). Hatta en düşük kalma süresi olan modellere öncelik verilir. Bu kural

akış zamanını minimize etmek istenildiği zaman idealdir. Çünkü en düşük işleme zamanı (SPT) kuralına çok benzer bir şekilde işlemektedir.

TOTMAX (Maksimum Toplam Zaman)

Seçme kriteri, her bir model için toplam montaj süresidir. Çevrim zamanı, talebi ve kaynakların sayısını çarparak hesaplayabiliriz($Sc_{ij} = Ct_{ij}*D_i*a_{ij}$). Toplam montaj süresi en yüksek olan modellere daha yüksek öncelik verilir. Makespan değerini düşürmek istenildiği durumlarda bu kural en iyi uygulanır.

TOTMIN (Minimum Toplam Zaman)

Seçme kriteri, her model için toplam montaj süresidir ve çevrim süresi talep ve kaynakların sayısını çarparak hesaplanabilir ($Sc_{ij} = Ct_{ij}*D_i*a_{ij}$). Montaj süresi en düşük olan modele daha yüksek öncelik verilir. Bu kural, akış zamanını minimize etmek istenildiğinde kullanılması tavsiye edilir.

5.7 Program Üretim Teknikleri

Gecikmeme, gecikme ve gecikme durma bu çalışma da kullanılan 3 temel program üretme tekniğidir. Bir model, şayet gecikmeme tekniği kullanılırsa kullanılabilir kaynaklardan oluşmuş hattı kullanarak ve tercihan en yüksek hat verimliliği konfigrasyonu ile işlenir. Gecikme tekniği kullanıldığı zaman, bir model öncelik matrisinin ilk sütunundaki [a_{i1}] modelinin kaynak gereksinimleri karşılanana kadar tutulur. Bu yüzden gecikme programları için öncelik matrisinin sadece birinci sütunu kullanılır. Bununla birlikte, birinci sütun tamamlandıktan sonra hala daha birtakım kullanılabilir kaynaklar varsa ve diğer bazı modeller monte edilecekse, bu kaynaklar modellerin kaynak gereksinimlerini karşıladıkları sürece kullanılırlar. Gecikme durma tekniği bir modelin, birinci sütun için gerekli olan kaynak mevcut oluncaya kadar ve bu kaynaklar diğer modellere tahsis edilmeyerek bekletilmesi noktasında gecikme tekniğinden ayrılır. Program

üretme teknikleri ile ilgili algoritmaların akış diyagramı konunun ilerleyen kısmında verilecekti

5.8 Algoritmalar

Bu kısımda 6 adet program üretme tekniği için, her birinin geniş açıklamaları da yapılarak, algoritmalar verilmiştir.

Kullanılan notasyonlar

t zaman sayacı

j sütun (alternatif) sayacı

k alternatiflerin sayısı

S atanmamış modellerin grubu

U geçici olarak göz önüne alınmayan S'ler

R_t t zamanında kullanılabilir kaynakların sayısı

SC_{ij} i modeli ve j alternatifi için seçme kriterinin değeri

i model sayacı

Gecikmeme Algoritması

Bir model için araştırma işlemi, öncelik matrisinin birinci sütununda başlar ve birinci sütundaki hat konfigurasyonu en yüksek verimliliğe sahip oluncaya kadar devam eder.

RESMIN, FMIN ve TOTMIN kuralları için minimum SC_{ij} değerleri seçilirken, RESMAX, FMAX ve TTMAX kuralları için de Sc_{ij} 'nin maksimum değerleri seçilir.

r modeli min(max) Sr_l değeri ile elde edildiği durumda mevcut olan kaynaklar ile bu modelin kaynak gereksinimleri karşılaştırılır. Şayet kullanılabilir yeterli derecede kaynak

varsa bu model ar_1 kaynağa atanır. Daha sonra bu model araştırmaların dışında tutulur ve kaynaklar bu modele tahsis edilen miktarda azaltılır. Şayet kullanılabilir kaynakların sayısı sıfır'a düşmüşse, zaman bir sonraki bitirme zamanına ilerletilir ve kullanılabilir kaynakların sayısı artırılır. Aksi taktirde algoritma tüm modellere kaynak atanıp atanmadığını kontrol eder.

Şayet yeteri kadar kaynak mevcut değilse model geçici olarak başka bir U grubuna yerleştirilir ve başka bir model için araştırma başlar. Şayet atanabilecek uygun bir model bulunamazsa, algoritma bir sonraki sütuna geçer (bu işlem hat verimliliğini düşürdüğü için istenmeyen bir durumdur) ve geçici olarak U'ya yerleştirilen tüm modeller yeniden S'ye yerleştirilir. Yeni bir model için araştırma yeni bir sütunda başlar.

Bir defa tüm sütunlar algoritma tarafından gözden geçirilir ve mevcut kaynakların kalan hiçbir modelin atanması için yeterli olmadığı sonucuna varılırsa, bundan sonra zaman bir sonraki tamamlama zamanına iletilir, kaynaklar ayarlanır gruplar yeniden düzenlenir ve araştırma birinci sütundan başlayarak bir kez daha yapılır. Bu algoritmanın akış diyagramı şekil 5-5'de gösterilmiştir.

Modifiye Edilmiş Gecikmeme Algoritması

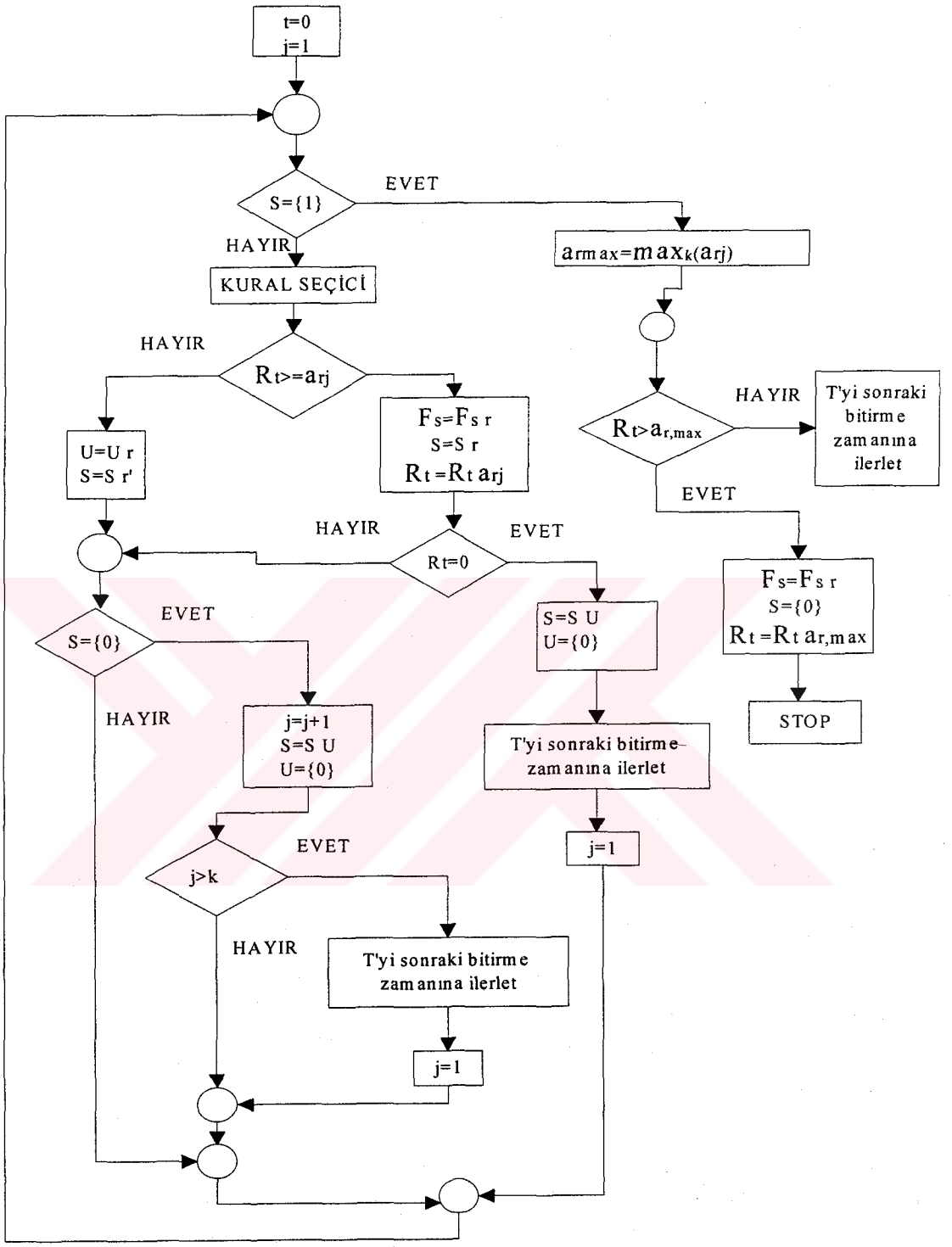
Gecikmeme algoritmasının modifiye edilmiş versiyonu, S grubunda en son tayin edilmemiş model için maksimum kaynak konfigurasyonu kullanılması dışında orijinal forma çok yakındır (benzerdir). Model için başlama zamanı gerekli max. Kaynak hazır oluncaya kadar tehir edilir. Bu algoritma için akış diyagramı şekil 5-6'da gösterilmiştir.

Gecikme Algoritması

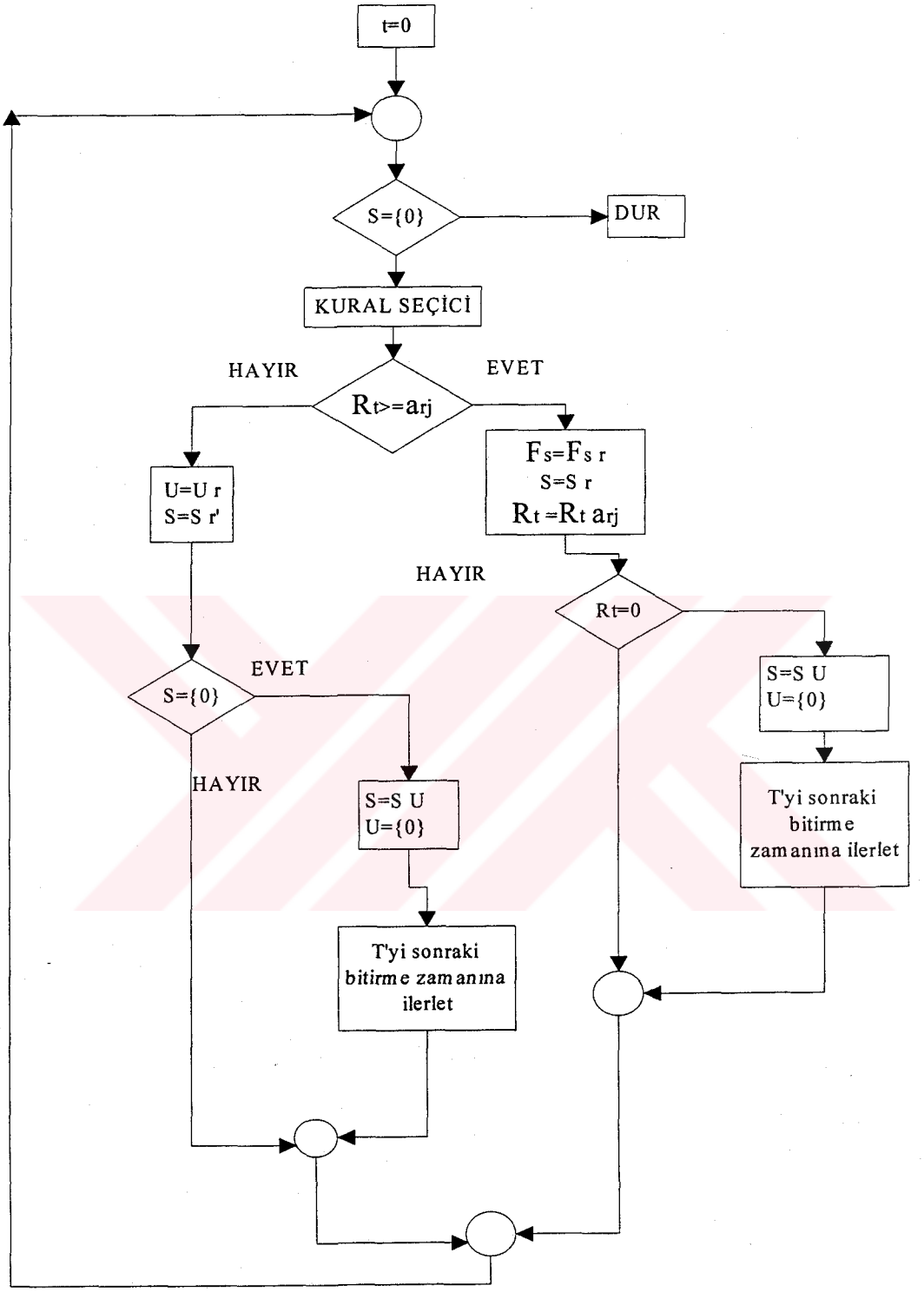
Bir model için araştırma her zaman öncelik matrisinin birinci sütunu göz önünde tutularak yapılır.

R modeli min (max) Sr_l değeri ile elde edildiği zaman, kullanılmaya uygun olan kaynaklar ile bu modelin kaynak gereksinimleri karşılaştırılır. Şayet yeterli derecede kaynak varsa bu model ar_l adet kaynağa atanır. Daha sonra model S grubundan çıkartılır ve kullanılabilir kaynakların sayısı aynı oranda azaltılır. Şayet kullanılabilir kaynakların sayısı sıfır'a düşerse, zaman bir sonraki bitirme zamanına ilerletilir ve kullanılabilir kaynaklar artırılır. Aksi halde, algoritma tüm modellerin atanıp atanmadığını kontrol eder.

Şayet yeterli kullanılabilir kaynak yoksa, model geçici olarak başka bir U grubuna yerleştirilir ve başak bir model , için araştırma başlar. Eğer atanacak fizibil bir model yoksa algoritma zamanı bir sonraki tamamlama zamanına ilerletir ve serbest bırakılan kaynakları sayısı oranında kullanılabilir kaynaklar artırılır. Geçici olarak S grubundan U'ya yerleştirilen modeller tekrar yerlerine iade edilirler ve yeni modeller için araştırma tekrar başlar. Bu algoritmanın akış diyagramı şekil 5-7'de gösterilmiştir.



Şekil 5-6. modifiye edilmiş gecikmeme algoritması için akış diyagramı



Şekil 5-7. Gecikme algoritması için akış diygramı

Modifiye Edilmiş Gecikme Algoritması

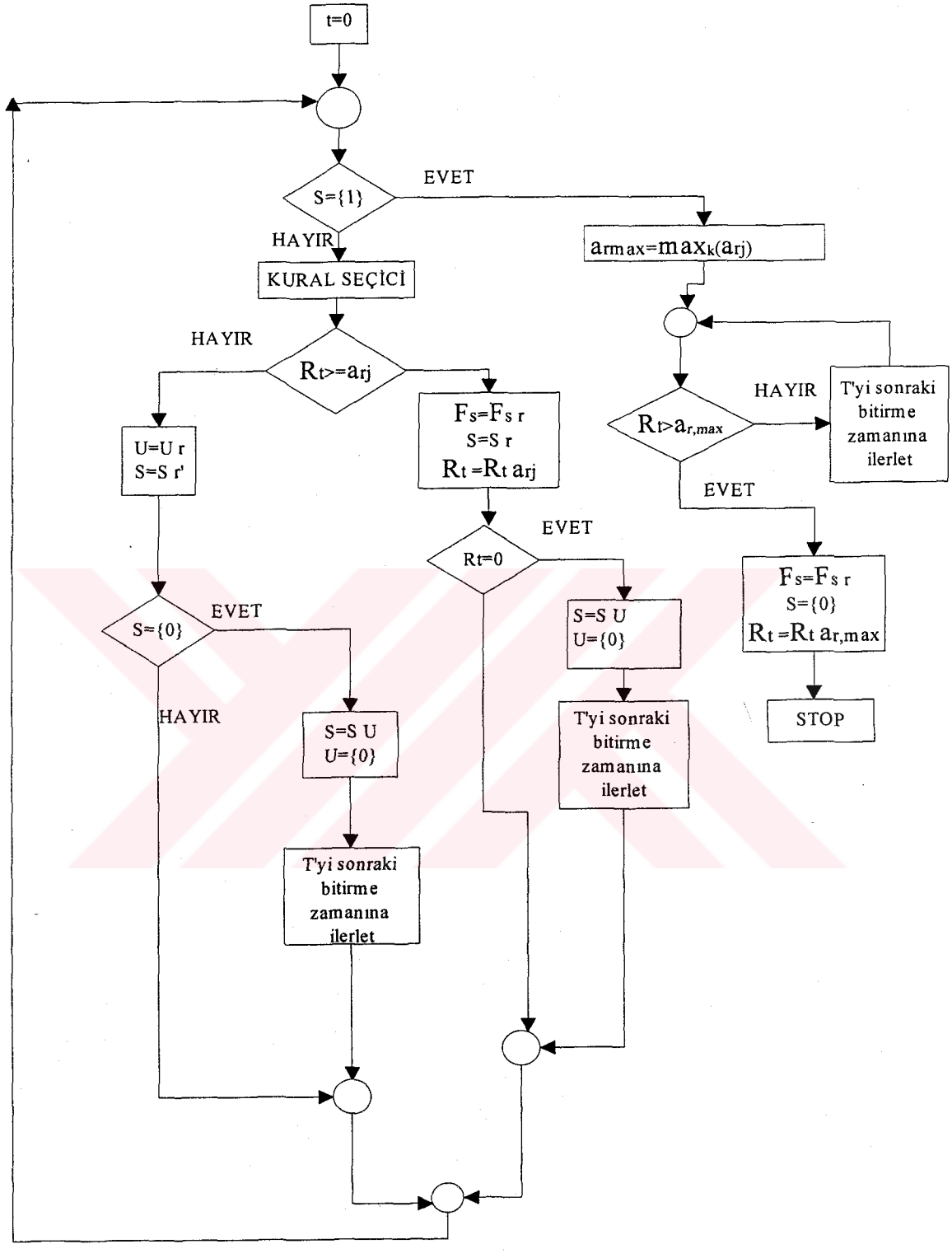
Gecikme algoritmasının modifiye edilmiş versiyonu orijinal formunun benzeridir. Bununla birlikte,birinci sütunun kaynak gereksinimini kullanma üzerindeki sınırlamalar S grubunda atanmamış modeller için ortadan kaldırılır. Hat verimliliğine bakılmaksızın maksimum kaynak konfigürasyonu kullanılır. Modelin başlangıç zamanı, gerekli olan maksimum kaynaklar kullanılabilir duruma gelinceye kadar ertelenir. Algoritmanın detayları şekil 5-8'de gösterilmiştir.

Gecikme\durma Algoritması

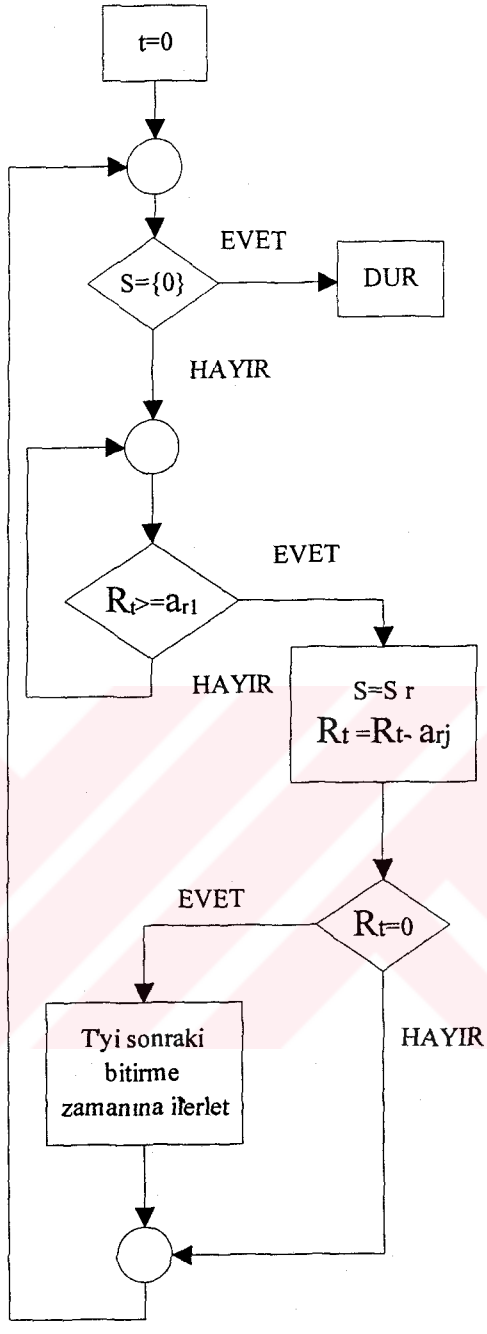
Bu teknikte de model için araştırma , daima öncelik matrisinin birinci sütunu göz önünde tutularak yapılmaktadır. Gecikme algoritmasıyla, bu algoritma arasında ki fark, şayet seçilmiş model kullanılabilir kaynaktan daha fazlasına ihtiyaç duyuyorsa başka bir model için araştırma başlar, ve seçilen model atanana kadar başka hiç bir model atanmaz. Zaman bir sonraki tamamlama zamanına iletilir ve kullanılabilir kaynağın sayısı artar.
Kullanılabilir kaynaklar ile gerekli kaynakların sayısı bir kez daha karşılaştırılır. Şekil 5-9'da algoritmaların tüm adımları gösterilmiştir.

Modifiye Edilmiş Gecikme\durma Algoritması

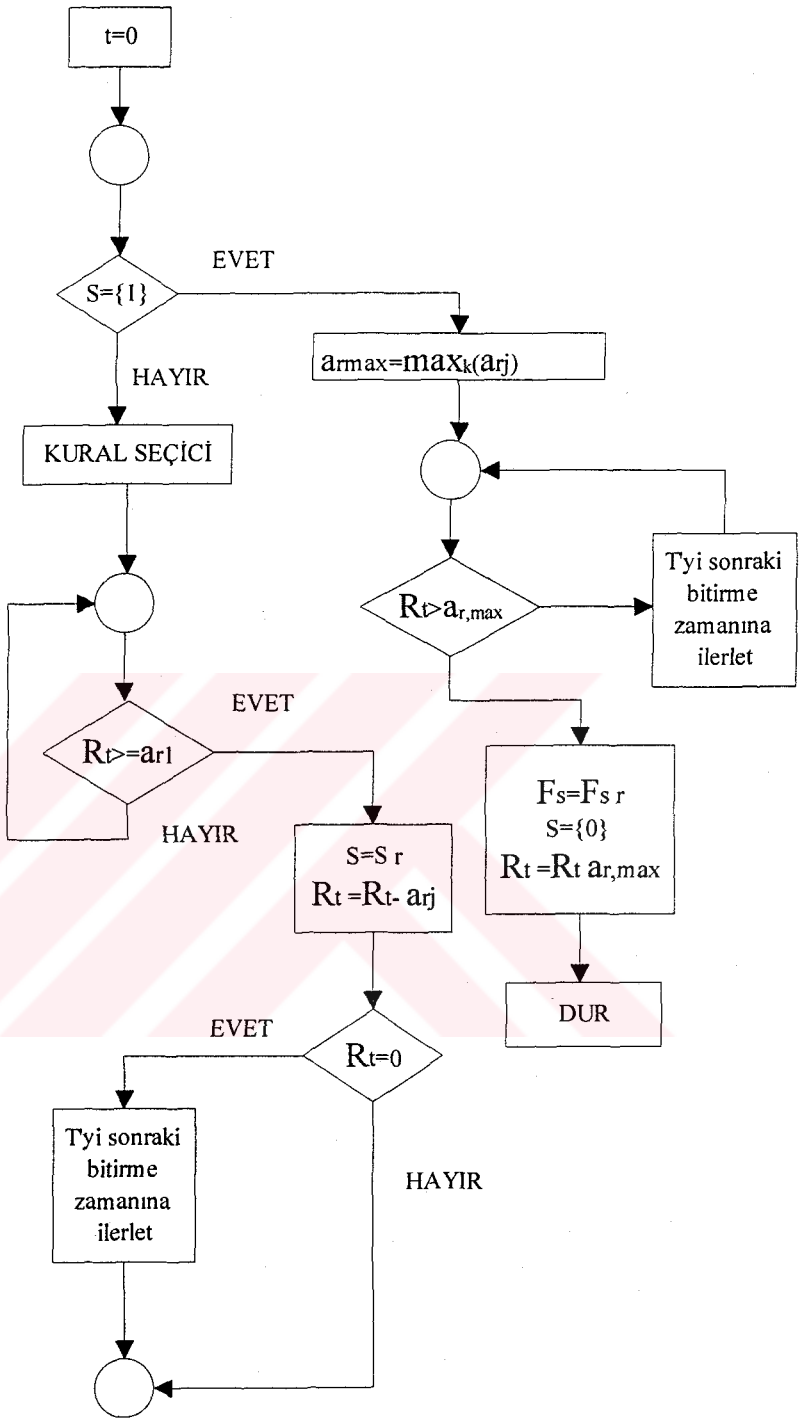
Gecikme\durma algoritmasının modifiye edilmiş versiyonu, modifiye edilmiş gecikme algoritmasına benzemektedir. Son atanmamış model için maksimum kaynak konfigürasyonu, hat verimliliği dikkate alınmaksızın kullanılmaktadır. Modelin başlama zamanı gerekli olan max kaynaklar kullanılabilir duruma gelinceye kadar tehir edilir. Prosesin tüm akış diyagramı şekil 5-10'da mevcuttur.



Şekil 5-8. Modifiye edilmiş gecikme algoritması için akış diyagramı.



Şekil 5-9. Gecikme\durma algoritması için akışdiyaramı



Şekil 5-10. Modifiye edilmiş gecikm\edurma algoritması için akış diyagramı

5.9 Bir Örnek

Bilgi tabanının kuralları, geniş deneyimlerle elde edilmiş tecrübi sonuçlara dayanarak tanımlamıştır. Deneyler sırasında göz önünde bulundurulan faktörler aşağıdaki gibidir.

1) modellerin sayısı (2 seviye)

a) 5 model

b) 10 model

2) talep (3 seviye)

a) düşük, 75 birim tüm modeller için,

b) orta, 100 birim tüm modeller için,

c) yüksek, 125 birim tüm modeller için.

3) alternatiflerin sayısı (2 seviye)

a) 2 alternatif

b) 3 alternatif

4) verimlilik (2 seviye)

a) 1.0,0.90 (iki alternatif); 1.0,0.90,0.80 (üç alternatif)

b) 1.0,0.80 (iki alternatif); 1.0,0.85,0.70 (üç alternatif)

5) öncelik matrisi (2 seviye)

a) bir model için 1 adetten 3 adete kaynak (uniform dağıtım)

b) bir model için 1 adetten 5 adete kaynak (uniform dağıtım)

Öncelik matrisini üretirken, aynı model için aynı kaynak değerinin tekrarlanmasına izin verilmemiştir. Bu yüzden bir model için kaynak tanımlamasının her yapılışında uniform dağılım gözden geçirilmektedir.

6) çevrim zamanı (6 seviye)

a) 1-5 dakika her birim (uniform dağıtım)

b) 1-10 dakika her birim (uniform dağıtım)

c) 3-5 dakika her birim (uniform dağıtım)

d) 4-10 dakika her birim (uniform dağıtım)

e) 6-10 dakika her birim (uniform dağıtım)

Yukarıda bahsedilen uniform dağıtım en yüksek hat verimliliği için çevrim zamanını elde etmek adına kullanılmıştır. Diğer alternatifler için geriye kalan çevrim zamanı değerleri hat verimlilikleri ve öncelik matrisinin kaynak şekilleri göz önünde tutularak hesaplanmıştır.

7) kaynak seviyeleri (7 seviye)

a) öncelik matrisinde maksimum kaynak seviyesi = x

b) x+1

c) x+2

d) x+3

e) x+4

f) x+5

g) x+6

Program Çalıştırıldığında bize altı adet soru soracaktır. Bu altı adet soruya aldığı cevaba göre program, amacı olan "combination" 'ı bulacaktır. Şimdi biz programın çalışmasını bir örnek ile ele alalım ve vereceği sonucu bulalım.

1- "Ürün sayısını giriniz". Burada sorulan model sayısıdır. Legal değerler 5-10'dur. Biz örneğimiz için **5** değerini seçelim

2- "Alternatiflerin sayısını giriniz". Burada alternatif hat konfigurasyon alternatifleri olup iki seviyelidir ve 2 veya 3 değerini alabilmektedir. Biz **3** değerini seçelim.

3- "Amacınızı giriniz" sorusuna makespan ya da akış zamanı cevabını verebiliriz. Örneğimizde **akış zamanı'** nı seçelim.

4- "Çevrim zamanı değerini en iyi hangisi temsil eder" sorusuna dakika olarak çevrim zamanının yoğunlaştığı aralığı seçmemiz lazımdır. Örneğimiz için **1-5** aralığını seçelim.

5- "Hangi kaynak seviyesi durumu en iyi temsil eder" sorusu için kullanılan iş istasyonlarının sayısı girilir. Legal değerler 1-3 ve1-5'dir. Biz **1-3** 'ü seçelim.

6- Verimlilik değeri de yüksek olarak girelim.

Bu değerlerin sonuçu da bize program aşağıdaki sonucu verir.

combination= FMIN rule and DELAY STOP MOD. Alg.

SONUÇ

İnsanoğlu daima gereksinimlerini giderme adına yeni ürünleri geliştirme yolunda olmuştur. Azmi ve becerisi sonucunda bunları ortaya çıkarmada başarılı olmuştur. Her şeyin otomasyona uğradığı asrımızda, zekânın da otomasyona bağlanması fikri kaçınılmaz hale gelmiştir. Bunda en önemli sebeplerden biri de birçok konuda uzmana olan ihtiyacın artmasıdır. Uzmanın zaman zaman pahalı olması , istihdam etmedeki zorluklar, sürekli olmaması , yani ölüm gibi, hastalık gibi durumlarda ya tam verimli ya da hiç istifade edilmemesi gibi dezavantajları kaldırıp yüksek kalitede, istikrarlı, kolayca sahip olunabileni cevap zamanı çok daha kısa olan, hem de tehlikeli çevrelerde çalışması kolay olan, insan uzman gibi karar verebilen sistemlere olan ihtiyaç uzman sistemlerin geliştirilmesinde en önemli faktörlerdendir. Böylelikle rekabetin çok fazla olduğu, Her şeyin hızının arttığı günümüzde bu rekabet şansı çok daha fazla artırılmış olur.

Uzman sistemlerin yararını sayarken, insan uzmanların tamamen gereksizliği ya da bu sistemlerin gelmesiyle birlikte insan uzmanların yerini birebir alabilecekleri sonucu çıkmamalıdır. Zira uzman sistemler kusursuz sistemler değillerdir. Onların çalışmasıyla birlikte bile uzman insanlara ihtiyaç vardır. Çünkü uzman sistemler özellikle yeni kavramlar veya yeni kurallar öğrenmede becerikli değillerdir. Ayrıca insan uzmanlar, yaşam tecrübelerinin kendilerine kazandırdığı bilgiye sahiptirler.

Uzman sistemlerin , uzman insanları da elimine etmeden, gerekliliği konusunda varılan sonuç ve bu sistemlerin önemliliği, artık tartışılmaz gerçek olan bir asırda bulunduğumuz şu günlerde artık ülkemizde de bu konuya gereken önem verilmesi elzem olmuştur. İdrak edilen bu gerçek sonucunda Türkiye 'mizde de pratiğe yönelik uzman sistemlerin geliştirilmesi beklenmektedir

KAYNAKLAR

Beerel A.C., Expert Systems :Strategic Implications And Applications, Newyork Halsted Press 1987

Dagli C.H., Prototype Expert System For Job Scheduling In Flexible Manufacturing, 1988.

Duda R.O., Knowledge Based Expert Systems, Come Of Age Byte Vol6, No:9 1981

Durkin J., Expert Systems Design And Development, Macmillan Publishing Company 1994

Epton J., Expert Systems And Optimisation,UNICOM 1994

Frost R., Introduction to Knowledge Based Systems, Riverside, Macmillan Inc 1986

Jackson P., Introduction To Expert System, Addison Wesley Publishing 1986.

Kelly R.V., Practical Knowledge Engineering: Creating Successful Commercial Expert Systems, Modern Art Newyork, 1991

Kidd A.L., Knowledge Acquisiion For Expert Systems Newyork, Plenum Press, 1987

Swift K.G., Knowledge Based Design For Manufacture, Prentice Hall Inc, 1987

Ringland G.A, Approaches To Knowledge Represantition Newyork, John Willey And Sons Inc 1988

Yazdani M., Building An Expert System, İn Expert Systems: Princeples An Case Studies Chapmann & Hall, Newyork 1989.

EK

```
goal=combination.


/*------------------------------ Syntax ----------------------
-----*/

prefix preferred.
prefix number.
prefix cycle.
prefix priority.
prefix desired.
postfix products.
postfix alternatives.
postfix objective.
postfix time.
postfix matrix.
prefix of.


/*-------------------- multivalued functions ------------------*/

multivalued(combination).
multivalued(desired objective).
multivalued(number of products).
multivalued(number of alternatives).
multivalued(cycle time).
multivalued(priority matrix).
multivalued(efficiency).
multivalued(case).


/*-------------------- questions and legal answers ----------------*/

enumeratedanswers(X).

question(number of products) = ['Enter Number of Products'].
legalvals(number of products) = [5, 10].

question(number of alternatives) =
        ['Enter Number of Alternatives'].
legalvals(number of alternatives) = [2, 3].
```

```
question(preferred objective) = ['Enter Your Objective'].
legalvals(preferred objective)= [makespan, flowtime].


question(cycle time) = ['Which one represents the cycle time values the best?'].
legalvals(cycle time) = [1-5, 1-10, 3-5, 4-10, 6-10].


question(priority matrix)=
    ['Which resource levelrepresents the situation the best?'].
legalvals(priority matrix) = [1-3, 1-5].

/* question(efficiency) =
   ['Which efficiency level represents the situation the best?'].
legalvals(efficiency) = [high, low].

/*---------- Generalizations and Unknowns ----------*/

if number of products= 5 and
  number of alternatives = 2
then case = 1.

if number of products= 5 and
  number of alternatives = 3
then case = 2.

if number of products= 10 and
  number of alternatives = 2
then case = 3.

if number of products= 10 and
  number of alternatives = 3
then case = 4.

if preferred objective = X
then desired objective = X.

if preferred objective is unknown
then desired objective = makespan and
    desired objective = flowtime.
if number of products is unknown
then number of products = 5 and
    number of products = 10.
```

```
if number of alternatives is unknown
then number of alternatives = 2 and
     number of alternatives = 3.

if cycle time is unknown
then cycle time = 1-5 and
     cycle time = 1-10 and
     cycle time = 3-5 and
     cycle time = 4-10
     cycle time = 6-10.

if priority matrix is unknown
then priority matrix = 1-3 and
     priority matrix = 1-5.

if efficiency is unknown
then efficiency = high and
     efficiency = low.


/*------------ 5 products, 2 alternatives, makespan ---------*/

if ((case = 1) or
   (case = 2 and priority matrix = 1-3)) and
   desired objective = makespan and
   cycle time = 3-5
then combination = ' TOTMAX rule and  DELAY MOD Alg'.

if case = 1 and
    desired objective = makespan and
   cycle time = 6-10 and
   priority matrix = 1-3
then combination = ' RESMAX rule and  DELAY STOP MOD Alg'.

if case = 1 and
   desired objective = makespan and
   cycle time = 6-10 and
   priority matrix = 1-5 and
   efficiency = high
then combination = ' RESMAX rule and  NONDELAY MOD Alg' and
     combination = ' RESMAX rule and  DELAY MOD Alg' and
     combination = ' RESMAX rule and  DELAY STOP MOD Alg'.

if case = 1 and
```

desired objective = makespan and
cycle time = 6-10 and
priority matrix = 1-5 and
efficiency = low
then combination = ' RESMIN rule and NONDELAY MOD. alg' and
combination = ' TOTMAX rule and NONDELAY MOD. alg'.

if case = 1 and
desired objective = makespan and
cycle time = 1-10 and
priority matrix = 1-3
then combination = ' RESMAX rule and NONDELAY MOD. alg' and
combination = ' RESMAX rule and DELAY MOD Alg'.

if case = 1 and
desired objective = makespan and
cycle time = 1-10 and
priority matrix = 1-5 and
efficiency = high
then combination = ' RESMAX rule and DELAY MOD Alg' and
combination = ' FMAX rule and DELAY Alg' and
combination = ' FMAX rule and DELAY MOD Alg'.

if case = 1 and
desired objective = makespan and
cycle time = 1-10 and
priority matrix = 1-5 and
efficiency = low
then combination = ' TOTMAX rule and DELAY MOD Alg'.

if case = 1 and
desired objective = makespan and
cycle time = 4-10
then combination = ' TOTMAX rule and DELAY MOD Alg'.

if case = 1 and
desired objective = makespan and
cycle time = 4-10 and
priority matrix = 1-3
then combination = ' RESMAX rule and DELAY MOD Alg'.

if case = 1 and
desired objective = makespan and
cycle time = 1-5 and

```
  priority matrix = 1-5
then combination = ' FMAX rule and  NONDELAY MOD. alg' and
    combination = ' FMAX rule and  DELAY Alg' and
    combination = ' FMAX rule and  DELAY MOD Alg' and
    combination = ' TOTMAX rule and  DELAY Alg' and
    combination = ' TOTMAX rule and  DELAY MOD Alg'.
if case = 1 and
  desired objective = makespan and
  cycle time = 1-5 and
  priority matrix = 1-3
then combination = ' RESMAX rule and  NONDELAY MOD. alg' and
    combination = ' RESMAX rule and  DELAY MOD Alg' and
combination = ' RESMAX rule and  DELAY STOP MOD Alg'.


/* -------- 5 products, 3 alternatives and makespan ----------*/

if case = 2 and
  desired objective = makespan and
  cycle time = 3-5 and
  priority matrix = 1-5
then combination = ' FMIN rule and  DELAY MOD Alg' and
    combination = ' FMIN rule and  DELAY STOP MOD Alg'.

if case = 2 and
  desired objective = makespan and
  cycle time = 1-10
then combination = ' TOTMAX rule and  DELAY Alg' and
    combination = ' TOTMAX rule and  DELAY MOD Alg'.

if case = 2 and
  desired objective = makespan and
  cycle time = 1-10 and
  priority matrix = 1-5
then combination = ' FMAX rule and  NONDELAY MOD. alg' and
    combination = ' FMAX rule and  DELAY Alg' and
    combination = ' FMAX rule and  DELAY MOD Alg'.

if case = 2 and
  desired objective = makespan and
  cycle time = 4-10 and
  priority matrix = 1-3
then combination = ' RESMAX rule and  DELAY MOD Alg' and
    combination = ' TOTMAX rule and  DELAY MOD Alg'.
```

```
if case = 2 and
   desired objective = makespan and
   cycle time = 4-10 and
   priority matrix = 1-5
then combination = ' RESMAX rule and  NONDELAY MOD. alg' and
     combination = ' RESMAX rule and  DELAY MOD Alg'.

if case = 2 and
     desired objective = makespan and
   cycle time = 1-5 and
   priority matrix = 1-5
then combination = ' FMAX rule and  DELAY MOD Alg' and
     combination = ' FMAX rule and  DELAY STOP Alg'.

if case = 2 and
     desired objective = makespan and
   cycle time = 1-5 and
   priority matrix = 1-3
then combination = ' FMAX rule and  NONDELAY MOD. alg' and
     combination = ' FMAX rule and  DELAY Alg' and combination = '
     TOTMAX rule and  DELAY Alg' and combination = ' TOTMAX rule
     and  DELAY STOP Alg'.


/*-------- 10 products, 2 alternatives and makespan ----------*/

if case = 3 and
    desired objective = makespan and
   cycle time = 1-10 and
   priority matrix = 1-3
then combination = ' TOTMAX rule and  NONDELAY MOD. alg'.


if case = 3 and
   desired objective = makespan and
   cycle time = 1-10 and
   priority matrix = 1-5
then combination = ' FMAX rule and  DELAY Alg' and
     combination = ' FMAX rule and  DELAY MOD Alg' and
     combination = ' FMAX rule and  DELAY STOP Alg' and
     combination = ' FMAX rule and  DELAY STOP MOD Alg'.

if case = 3 and
   desired objective = makespan and
```

(cycle time = 4-10 or
cycle time = 1-5)
then combination = ' FMAX rule and DELAY Alg' and
combination = ' FMAX rule and DELAY MOD Alg'.

if case = 3 and
desired objective = makespan and
cycle time = 4-10 and
priority matrix = 1-3
then combination = ' FMAX rule and NONDELAY MOD. alg'.

if case = 3 and
desired objective = makespan and
cycle time = 1-5 and
priority matrix = 1-3
then combination = ' TOTMAX rule and DELAY Alg' and
combination = ' TOTMAX rule and DELAY MOD Alg'.

if case = 3 and
desired objective = makespan and
cycle time = 1-5 and
priority matrix = 1-5 and
efficiency = high
then combination = ' TOTMAX rule and NONDELAY alg' and
combination = ' TOTMAX rule and NONDELAY MOD. alg'.

if case = 3 and
desired objective = makespan and
cycle time = 3-5 and
priority matrix = 1-3
then combination = ' RESMAX rule and NONDELAY alg' and
combination = ' RESMAX rule and DELAY Alg'.

if case = 3 and
desired objective = makespan and
cycle time = 3-5 and
priority matrix = 1-5
then combination = ' TOTMAX rule and NONDELAY MOD. alg'.

if case = 3 and
desired objective = makespan and
cycle time = 6-10 and
priority matrix = 1-3

```
then combination = ' TOTMAX rule and  NONDELAY MOD. alg' and
    combination = ' TOTMAX rule and  DELAY MOD Alg'.

if case = 3 and
  desired objective = makespan and
  cycle time = 6-10 and
  priority matrix = 1-5
then combination = ' FMIN rule and  DELAY Alg' and
    combination = ' FMIN rule and  DELAY MOD Alg'.

if case = 3 and
  desired objective = makespan and
  cycle time = 6-10 and
  priority matrix = 1-5 and
  efficiency = high
then combination = ' FMAX rule and  DELAY Alg' and
    combination = ' FMAX rule and  DELAY MOD Alg'.

if case = 3 and
  desired objective = makespan and
  cycle time = 6-10 and
  priority matrix = 1-5 and
  efficiency = low
then combination = ' RESMAX rule and  NONDELAY MOD. alg' and
    combination = ' RESMAX rule and  DELAY MOD Alg'.


/*------ 10 products, 3 alternatives and makespan ----------*/

if case = 4 and
  desired objective = makespan and
  cycle time = 1-10
then combination = ' FMAX rule and  DELAY Alg' and
    combination = ' FMAX rule and  DELAY MOD Alg'.
if case = 4 and
  desired objective = makespan and
  cycle time = 1-10 and
  priority matrix = 1-5 and
  efficiency = high
then combination = ' TOTMAX rule and  DELAY Alg' and
    combination = ' TOTMAX rule and  DELAY MOD Alg'.

if case = 4 and
  desired objective = makespan and
  (cycle time = 3-5 or
```

cycle time = 6-10) and
priority matrix = 1-5
then combination = ' FMAX rule and DELAY Alg' and
combination = ' FMAX rule and DELAY MOD Alg'.

if case = 4 and
desired objective = makespan and
cycle time = 3-5 and
priority matrix = 1-3
then combination = ' TOTMAX rule and NONDELAY alg'.

if case = 4 and
desired objective = makespan and
cycle time = 6-10 and
priority matrix = 1-3
then combination = ' FMAX rule and DELAY Alg'.

if case = 4 and
desired objective = makespan and
cycle time = 4-10 and
priority matrix = 1-3
then combination = ' FMAX rule and DELAY Alg' and
combination = ' FMAX rule and DELAY MOD Alg' and
combination = ' RESMAX rule and NONDELAY MOD. alg' and
combination = ' RESMAX rule and DELAY MOD Alg'.

if case = 4 and
desired objective = makespan and
cycle time = 4-10 and
priority matrix = 1-5 and
efficiency = high
then combination = ' RESMAX rule and NONDELAY MOD. alg'.

if case = 4 and
desired objective = makespan and
cycle time = 4-10 and
priority matrix = 1-5 and
efficiency = low
then combination = ' RESMAX rule and DELAY MOD Alg'.

```
if case = 4 and
  desired objective = makespan and
  cycle time = 1-5 and
  priority matrix = 1-3
then combination = ' RESMAX rule and  NONDELAY alg' and
    combination = ' RESMAX rule and  DELAY Alg' and
    combination = ' FMAX rule and  DELAY Alg' and
    combination = ' FMAX rule and  DELAY STOP Alg'.

if case = 4 and
  desired objective = makespan and
  cycle time = 1-5 and
  priority matrix = 1-5
then combination = ' TOTMAX rule and  NONDELAY alg' and
    combination = ' TOTMAX rule and  DELAY Alg' and
    combination = ' TOTMAX rule and  DELAY MOD Alg'.


/*--------- 5 products, 2 alternatives and flowtime ----------*/

if case = 1 and
  desired objective = flowtime and
  cycle time = 3-5 and
  priority matrix = 1-5
then combination = ' FMIN rule and  DELAY MOD Alg'.

if case = 1 and
  desired objective = flowtime and
  cycle time = 3-5 and
  priority matrix = 1-3
then combination = ' TOTMIN rule and  DELAY Alg' and
    combination = ' TOTMIN rule and  DELAY MOD Alg' and
    combination = ' TOTMIN rule and  DELAY STOP Alg' and
    combination = ' TOTMIN rule and  DELAY STOP MOD Alg'.

if case = 1 and
  desired objective = flowtime and
  cycle time = 3-5 and
  priority matrix = 1-3 and
  efficiency = high
then combination = ' FMIN rule and  DELAY Alg' and
    combination = ' FMIN rule and  DELAY MOD Alg' and
    combination = ' TOTMAX rule and  DELAY MOD Alg'.
```

if case = 1 and
desired objective = flowtime and
cycle time = 6-10 and
priority matrix = 1-3
then combination = ' FMIN rule and DELAY MOD Alg' and
combination = ' FMIN rule and DELAY STOP MOD Alg'.

if case = 1 and
desired objective = flowtime and
cycle time = 6-10 and
priority matrix = 1-5 and
efficiency = high
then combination = ' TOTMAX rule and NONDELAY MOD. alg'.

if case = 1 and
desired objective = flowtime and
cycle time = 6-10 and
priority matrix = 1-5 and
efficiency = low
then combination = ' FMIN rule and NONDELAY alg' and
combination = ' FMIN rule and NONDELAY MOD. alg' and
combination = ' FMIN rule and DELAY STOP MOD Alg' and
combination = ' RESMAX rule and NONDELAY MOD. alg' and
combination = ' TOTMIN rule and NONDELAY MOD. alg'.

if case = 1 and
desired objective = flowtime and
cycle time = 1-10 and
priority matrix = 1-3
then combination = ' RESMIN rule and DELAY Alg' and
combination = ' RESMIN rule and DELAY MOD Alg' and
combination = ' RESMIN rule and DELAY STOP Alg' and
combination = ' RESMIN rule and DELAY STOP MOD Alg' and
combination = ' TOTMIN rule and DELAY Alg' and
combination = ' TOTMIN rule and DELAY MOD Alg' and
combination = ' TOTMIN rule and DELAY STOP Alg' and
combination = ' TOTMIN rule and DELAY STOP MOD Alg'.

```
if case = 1 and
   desired objective = flowtime and
   cycle time = 1-10 and
   priority matrix = 1-5 and
   efficiency = high
then combination = ' RESMIN rule and  NONDELAY MOD. alg' and
     combination = ' TOTMIN rule and  DELAY Alg' and
     combination = ' TOTMIN rule and  DELAY MOD Alg' and
     combination = ' TOTMIN rule and  DELAY STOP Alg' and
     combination = ' TOTMIN rule and  DELAY STOP MOD Alg'.

if case = 1 and
   desired objective = flowtime and
   cycle time = 1-10 and
   priority matrix = 1-5 and
   efficiency = low
then combination = ' FMIN rule and  DELAY MOD Alg' and
     combination = ' FMIN rule and  DELAY STOP MOD Alg'.

if case = 1 and
   desired objective = flowtime and
   cycle time = 4-10 and
   priority matrix = 1-3
then combination = ' FMIN rule and  DELAY MOD Alg' and
     combination = ' FMIN rule and  DELAY STOP MOD Alg'.

if case = 1 and
    desired objective = flowtime and
   cycle time = 4-10 and
   priority matrix = 1-5
then combination = ' FMIN rule and  NONDELAY MOD. alg' and
     combination = ' FMIN rule and  DELAY MOD Alg'.

if case = 1 and
   desired objective = flowtime and
   cycle time = 1-5 and
   priority matrix = 1-3
then combination = ' FMIN rule and  DELAY STOP MOD Alg'.

if case = 1 and
   desired objective = flowtime and
   cycle time = 1-5 and
   priority matrix = 1-3 and
   efficiency = low
```

then combination = ' RESMAX rule and NONDELAY MOD. alg' and
combination = ' RESMAX rule and DELAY MOD Alg' and
combination = ' RESMAX rule and DELAY STOP MOD Alg' and
combination = ' FMIN rule and NONDELAY MOD. alg'.

if case = 1 and
desired objective = flowtime and
cycle time = 1-5 and
priority matrix = 1-5
then combination = ' RESMIN rule and DELAY Alg' and combination = '
RESMIN rule and DELAY MOD Alg' and
combination = ' RESMIN rule and DELAY STOP Alg' and combination
= ' RESMIN rule and DELAY STOP MOD Alg'.

/*-------- 5 products, 3 alternatives and flowtime ----------*/

if case = 2 and
desired objective = flowtime and
cycle time = 3-5 and
priority matrix = 1-3
then combination = ' FMIN rule and NONDELAY MOD. alg' and
combination = ' FMIN rule and DELAY MOD Alg' and
combination = ' TOTMIN rule and DELAY Alg' and
combination = ' TOTMIN rule and DELAY MOD Alg'.

if case = 2 and
desired objective = flowtime and
cycle time = 3-5 and
priority matrix = 1-5
then combination = ' RESMIN rule and DELAY Alg' and
combination = ' RESMIN rule and DELAY MOD Alg' and
combination = ' RESMIN rule and DELAY STOP Alg' and
combination = ' RESMIN rule and DELAY STOP MOD Alg' and
combination = ' TOTMIN rule and DELAY Alg' and
combination = ' TOTMIN rule and DELAY MOD Alg' and
combination = ' TOTMIN rule and DELAY STOP Alg' and
combination = ' TOTMIN rule and DELAY STOP MOD Alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 6-10 and
  priority matrix = 1-3 and
  efficiency = high
then combination = ' RESMAX rule and DELAY MOD Alg' and
    combination = ' FMIN rule and NONDELAY MOD. alg' and
    combination = ' FMIN rule and DELAY MOD Alg' and
    combination = ' TOTMIN rule and NONDELAY MOD. alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 6-10 and
  priority matrix = 1-3 and
  efficiency = low
then combination = ' RESMAX rule and DELAY MOD Alg' and
    combination = ' TOTMIN rule and NONDELAY MOD. alg' and
    combination = ' TOTMIN rule and DELAY Alg' and
   combination = ' TOTMIN rule and DELAY MOD Alg' and
    combination = ' TOTMIN rule and DELAY STOP Alg' and
    combination = ' TOTMIN rule and DELAY STOP MOD Alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 6-10 and
  priority matrix = 1-5
then combination = ' FMIN rule and DELAY MOD Alg' and
    combination = ' FMIN rule and DELAY STOP MOD Alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 1-10 and
  priority matrix = 1-3
then combination = ' TOTMIN rule and DELAY MOD Alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 1-10 and
  priority matrix = 1-5
then combination = ' FMIN rule and DELAY MOD Alg' and
    combination = ' FMIN rule and DELAY STOP MOD Alg'.

```
if case = 2 and
  desired objective = flowtime and
  cycle time = 4-10 and
  priority matrix = 1-3
then combination = ' FMIN rule and  DELAY Alg' and
    combination = ' FMIN rule and  DELAY MOD Alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 4-10 and
  priority matrix = 1-5
then combination = ' TOTMIN rule and  DELAY Alg' and
    combination = ' TOTMIN rule and  DELAY MOD Alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 1-5
then combination = ' FMIN rule and  DELAY STOP MOD Alg'.

if case = 2 and
  desired objective = flowtime and
  cycle time = 1-5 and
  priority matrix = 1-5
then combination = ' TOTMIN rule and  DELAY Alg' and
    combination = ' TOTMIN rule and  DELAY MOD Alg'.

   /* -------- 10 products, 2 alternatives and flowtime ---------*/
if case = 3 and
  desired objective = flowtime and
  priority matrix = 1-5
then combination = ' TOTMIN rule and  DELAY Alg' and combination
   = ' TOTMIN rule and  DELAY MOD Alg'.

if case = 3 and
  desired objective = flowtime and
  not(cycle time = 1-10) and
  priority matrix = 1-3
then combination = ' TOTMIN rule and  DELAY Alg' and
    combination = ' TOTMIN rule and  DELAY MOD Alg'.

if case = 3 and
  desired objective = flowtime and
```

```
  not(cycle time = 1-5) and
  priority matrix = 1-3
then combination = ' TOTMIN rule and  DELAY STOP Alg' and
    combination = ' TOTMIN rule and  DELAY STOP MOD Alg'.

if case = 3 and
  desired objective = flowtime and
  cycle time = 1-5 and
  priority matrix = 1-3
then combination = ' FMIN rule and  NONDELAY MOD. alg' and
    combination = ' FMIN rule and  DELAY MOD Alg' and combination
    = ' TOTMIN rule and  NONDELAY alg' and combination = ' TOTMIN
    rule and  NONDELAY MOD. alg'.


/*-------- 10 products, 3 alternatives and flowtime ----------*/

if case = 4 and
  desired objective = flowtime and
  cycle time = 3-5 or
  cycle time = 6-10 or
  cycle time = 4-10
then combination = ' TOTMIN rule and  DELAY Alg' and
    combination = ' TOTMIN rule and  DELAY MOD Alg'.




if case = 4 and
  desired objective = flowtime and
  cycle time = 6-10 and
  priority matrix = 1-5
then combination = ' FMAX rule and  DELAY Alg' and
    combination = ' FMAX rule and  DELAY MOD Alg' and
    combination = ' FMIN rule and  NONDELAY MOD. alg'.

if case = 4 and
  desired objective = flowtime and
  cycle time = 3-5 and
  priority matrix = 1-3 or
  (priority matrix = 1-5 and
  efficiency = low)
then combination = ' TOTMIN rule and  DELAY STOP Alg' and
    combination = ' TOTMIN rule and  DELAY STOP MOD Alg'.
```

ÖZGEÇMİŞ

Doğum tarihi : 17 Ocak 1971

Doğum Yeri : Kahramanmaraş

Eğitim : 1982-1989 Kahramanmaraş Anadolu Lisesi

: 1989-1993 Yıldız Teknik Üniversitesi Elektrik Mühendisliği Bölümü

: 1993-1994 Marmara Üniversitesi Management Information Systems (MIS) Lisansüstü programı

: 1993-1996 Yıldız Teknik Üniversitesi F.B.E Elektrik Mühendisliği A.B.D

Çalıştığı Kurum: BELBİM İstanbul Belediyeleri Bilgi İşlem San. ve Tic. A.Ş'de proje mühendisi olarak çalışmaktadır.