BİLGİSAYAR PROGRAMCILARININ PROGRAM GELİŞTİRME DENEYİMİ KAZANMALARINDA KARAR TABLOLARININ KULLANIMI

Ali GÜNEŞ*

Özet

Bu çalışmada bilgisayar programcılarının program geliştirme süreci aşamalarında deneyim kazanmalarında kullanılmak üzere karar tablolarından destek amaçlı olarak nasıl yararlanılabileceği üzerinde durulmuştur. Çalışmada geliştirilen bir yazılım ile programcı, önce problemle ilgili bir karar tablosu oluşturmakta ve sadeleştirme çalışması yapmaktadır. Daha sonra Quine-McCluskey Algoritması kullanılarak başlangıçtaki söz konusu tablo sadeleştirilmektedir. Bu işlemin ardından, bu sadeleştirilmiş olan tablo, programcıya kendi yaptığı sadeleştirmelerle karşılaştırmak üzere verilmektedir. Böylece programcının ele alınan problem alanı hakkında ne kadar bilgili olduğu, yapılan analiz ve tasarımın problemi ne kadar kapsadığı konusunda kendini sınaması olanağı ortaya çıkmaktadır. Bu çalışmada geliştirilen algoritma kullanıldığında, programcının problem alanı hakkındaki ve yazılım geliştirme konusundaki yetkinliğini sınaması ve geliştirmesi olanağı sağlanabilecektir.

Abstract

In this paper, we have studied how to get advantages of decision tables with the purpose of programming education in order to use in software development process cycles. A software was developed where a programmer can create and simplify a decision table related with the problem. Then the starting decision table is simplified by using Quine-McCluskey Algorithm. The decision table that was simplified with Quine-McCluskey Algorithm is given to the programmer in order to compare the

---

**Prof. Dr., İstanbul Aydın Üniversitesi Mühendislik-Mimarlık Fakültesi.*

table with the user's own simplifications. Thus the programmer can test his or her knowledge about the problem space, and can test how much analysis and design comprises the problem. With the help of the software developed in this work, the programmer can test and improve his or her level of knowledge about the problem space and the software development process.

1. Giriş

Bu çalışmada yazılım analiz ve tasarım aşamalarında süreç/işlem modelleme yöntemlerinden biri olan karar tablolarının, deneyim kazanma amaçlı olarak belirli bir yazılım kullanılarak hazırlanması ve özellikle ortaya büyük bir tablo çıktığında bu tablonun nasıl daha kolay sadeleştirilebileceği konusunda çalışılmıştır. Programcının yapacağı tablo sadeleştirmeleri ile Quine-McCluskey Algoritması kullanılarak yapılan sadeleştirilmeler karşılaştırılarak konu üzerindeki deneyim kazanma becerisine katkıda bulunulmuştur. Ortaya çıkan nihai tablo, programcının ele alınan yazılım problemine ne kadar hakim olduğu konusunda geri bildirimler verebilecektir. Böylece, gerek eğitsel amaçlı, gerekse gerçek uygulamalarda programcının hata yapma olasılığı en az indirilmeye çalışılmıştır.

2. Karar Tablosu Kullanımı

Karar tabloları başlangıçta ağırlıklı olarak yazılım kodu geliştirmede kullanılan bir yöntem olarak geliştirilmiştir(Moreno, 2000; Veinott, 1966). Buradaki esas amaç, bir yazılım probleminin çözümüne yönelik geliştirilen karar tablosunun yazılım koduna dönüştürülmesiydi (Al-Koraishi, 1989; Dwyer, 2007; Reinwald, 1967). Günümüze kadar, bir karar tablosunu belirli bir programlama dili ile koda çeviren yazılımlar da geliştirilmiştir (Dwyer, 2008). Ancak zaman içinde karar tabloları daha başka birçok alanda da kullanılmaya başlanmış ve bu çalışmalarla ilgili olarak 1982–2000 yılları arasında 1000' e yakın yayın ortaya çıkmış olup, söz konusu karar tablosu çalışmaların dağılımı Tablo 1'de gösterilmiştir (Moreno, 2000).

Tablo 1'de görüldüğü gibi, yapılan karar tablosu uygulamalarının büyük kısmı yazılıma yöneliktir.

Karar Tablosu Uygulama Alanı	**Yüzde**
Uzman sistemler, karar destek sistemleri, yapay us	34
Programlama	10
Kontrol sistemleri	8
Karar formları	7
Uygulamalı matematik	7
Yazılım geliştirme	5
Algoritma geliştirme	5
Tıp	4
Elektronik devre tasarımı	3
Veritabanı	3
Üretim sistemleri	3
CASE araçları	2
Robotics	2
Diğer	7

Tablo 1. 1982–2000 yılları arasında yapılan karar tablosu çalışmaları

VKarar tablosu, bir karar probleminin tablo şeklinde gösterimi olup, ele alınan konudaki olası tüm durumları sergilemesi ve yüksek düzeyde tutarlılık sağlaması nedeniyle yazılım mühendisliği için önemli bir araçtır (Vanthienen, 1994). Problem alanı içindeki her bir olası koşul durumunun gerçekleşmesi halinde hangi işlemin yapılacağı tabloda sütunlar şeklinde ifade edilir. Problem alanıyla ilgili tüm olası koşullar koşul seçeneklerini, bu olası koşullara bağlı yapılması gereken işlemler ise işlem girişlerini meydana getirir. Tüm bu yapı, yazılımcının problemin çözümü konusunda geliştireceği tasarım sürecini kısaltır ve tasarım üzerindeki hakimiyeti arttırır (Vanthienen, 1994). Örneğin, 6 kişinin akış şeması kullanarak bir yılda yaptıkları dosya bakımı konusundaki bir yazılım tasarımı, karar tablosu kullanılarak 4 kişiyle dört haftada gerçekleştirilmiştir (Vanthienen, 1994).

Bir karar tablosu Şekil 1'de gösterildiği gibi, koşul girişleri, işlem girişleri, koşul seçenekleri (kurallar) ve işlemler olmak üzere dört kısımdan meydana gelmektedir.

Problem Alanı	Kurallar
KOŞUL GİRİŞLERİ	KOŞUL SEÇENEKLERİ
İŞLEM GİRİŞLERİ	İŞLEMLER

Şekil 1. Karar Tablosu Bileşenleri

Bir karar tablosunda koşulların ve işlemlerin gösterimleri değişik biçimlerde olabilir (CODASYL, 1982; Vanthienen, 2001). Bu çalışmada koşulların sadece Evet veya Hayır olarak gerçekleştiği ve olası herhangi bir koşul kümesinin diğerlerinden bağımsız olduğu ve sadece tek bir işlemi tanımladığı türde (if... then... else .... benzeri) karar tabloları ele alınacak ve karar tablosu terimi bu tür yapılar için kullanılacaktır. Belirli bir koşul seçeneği grubu ile buna bağlı tek bir işleme kural denir. Doğal olarak kural sayısı koşul seçeneği sayısı kadardır. Kurallar 0'dan başlayarak maksimum kural adedi kadar ardışık sırada numaralandırılır.

Bir karar tablosunda, koşul kümesinde "n" adet koşul olması durumunda, eğer bu koşullar sadece "Evet" (E) veya "Hayır" (H) ile sonuçlanabilecek ise, bu durumda tüm kurallar, koşulların olası her türlü sonucunu kapsayacak miktarda sonuç uzayı 2^n genişliğindedir. Yani eğer 3 adet koşul var ise, tüm olası sonuçları kapsayan 8 (2^3) adet, eğer 5 adet koşul var ise, kural miktarı 32 (2^5) olacaktır. Bu durum karar tablosu kullanımını kısıtlayan en önemli özelliklerden biridir. En azından tablo büyüdüğünde ele alınması ve kullanımı pratik olmaktan uzaklaşmaktadır. Bu nedenle karar tablolarının oluşturulması konusunda yazılımlar da geliştirilmiştir (Vanthienen, 2001; Wets, 1994).

Karar tablosundaki her bir koşul, genellikle problem alanındaki bir değişkene, en yaygın haliyle de, girdi-işlem-çıktı türü tasarımlarda bir girdiye karşılık gelir. Karar tablosundaki işlem ise, koşul seçeneklerinden herhangi birinin gerçekleşmesi durumda yapılacak işlem veya uygulanacak yordamı ifade eder. Bir karar tablosunda kullanılan koşul sayısı arttıkça, ifade edilebilecek olası tüm koşul durumlarını

kapsayan koşul uzayı da çok genişleyecek, sadece tablonun görünümü değil, hazırlanması, kullanımı ve en önemlisi sadeleştirmesi de en azından görsel nedenlerle zorlaşacaktır.

Bir karar tablosunda ne gibi koşul girişi olabileceği ve olası tüm koşullara bağlı olarak hangi işlemlerin yapılabileceği tamamen ilgili karar tablosunun ele alındığı konuya yani probleme bağlıdır. İyi bir analiz çalışması yapılmış ise, doğru bir karar tablosu ortaya çıkabilecektir. İşlem girişi miktarı bir karar tablosunun büyüklüğünü doğrudan etkilemez. Ancak, başlangıç karar tablosunun müstakbel koda dönüşmesi oldukça karmaşık olup, zaten tekrar eden veya gerçekleşmesi mümkün olmayan koşul bileşenleri nedeniyle anlamsız da olabilir. Tüm bu sakıncaları gidermek için, bir karar tablosu başlangıçtaki haliyle bırakılmaz, olabildiğince sadeleştirilir. Bu çalışmada sadece söz konusu sadeleştirme işlemi üzerinde durulacaktır. Yapılan sadeleştirme çalışması ile programcının ilgilendiği problem alanını daha iyi tanıması ve en uygun ve doğru kodun ortaya çıkmasına, yani yazılım geliştirme sürecine eğitsel katkı sağlanmış olacaktır.

3. Karar Tablosunda Sadeleştirme Problemi

Karar tablosu kullanımı ile yazılımcı problem alanıyla ilgili olarak analiz ve tasarım konusunda aşağıdaki avantajları elde edebilecektir (Vanthienen, 2001; Merlevede, 1991).

- Çalışma tam ve eksiksiz olabilecektir.
- Çalışmada tutarlık artacaktır.
- Çalışmada tekrarlar olmayacaktır.
- Çalışmada hata en aza inecektir.

Tüm bu üstünlüklerine karşılık, ele alınan problem alanındaki koşul ve koşul seçenekleri (kurallar) sayısının çokluğu karar tablosunun kullanımını zorlaştırmaktadır. Bu nedenle karar tablolarının sadeleştirilmesi oldukça önemlidir. Çünkü söz konusu karar tablosu daha sonra müstakbel bir koda dönüştürülecektir. Bu nedenle karar tablosu sadeleştirilirse yukarıda listelenen avantajlar sürdürebilecek, ayrıca aşağıdaki olanaklar da sağlanabilecektir (Vanthienen, 1994; Shwayder, 1975; Colomb, 1990; Cragun, 1987; Wets, 1997).

- Kodun işlem süresi kısalacaktır.
- Kodu geliştirme süresi kısalacaktır.
- Kodun ihtiyaç duyacağı bellek miktarı azalacaktır.
- Koşul ve işlem miktarı azalacağından, kodun okunabilirliği artacaktır.

Yukarıda listelenen nitelikte yazılım geliştirmek için deneyimli olmak kuşkusuz ilk koşuldur. Bu süreci hızlandırmak için kullanılabilecek araçlardan biri de, bu çalışmada ele alınan yöntemle geliştirilen algoritmayı kullanarak, problem alanının başlangıçta görülenden daha basit ve sade biçimde ortaya koyabilme olanağını, daha kısa sürede görülmesini sağlamaya yöneliktir. Bilgisayar programcılık eğitimi alan kişiler bu çalışmaları ne kadar sık ve farklı problem alanı üzerinde yaparlarsa deneyim kazanmaları da o kadar kısalabilir. Ancak bu doğal olarak uzun zaman alacaktır. Bu çalışmada ele alınan yöntemle söz konusu zamanın kısaltılması ve kullanılabilecek örneklerin çoğaltılması hedeflenmiştir. Bilgisayar programcılığı eğitimi alan kişi problem alanını çok iyi kavradığında, kendi öngördüğü sadeleştirme ile bu çalışmadaki yöntemle kısa sürede elde edilen sadeleştirme sonucunu karşılaştırma olanağı bulabilecektir. Bu sayede problem alanını ne kadar iyi kavramış olduğunu ve hangi sadeleştirmelerin de mümkün olabildiğini görme olanağı olacaktır. Tüm bu faydalar ile bu çalışmada geliştirilen yazılım, yazılım geliştirme eğitiminde yarar sağlayacaktır.

Bilgisayar programı geliştirme sürecinde yapılabilecek sadeleştirmelerin önemini aşağıdaki örnekle de açıklayabiliriz:

$\sqrt{9} + \sqrt{4}$ yerine kısaca 5 kullanmak yukarıdaki sadeleştirme açıklamalarını destekler niteliktedir. Bu örnekte de sadeleştirme yapılmadığı takdirde daha uzun kod, daha fazla işlem süresi vb. etkilerin daha fazla olacağı ortadadır.

Karar tablolarının algoritmik olarak sadeleştirilmesi konusunda birçok deneme yapılmış ancak biri hariç çok başarılı sonuçlar alınmamıştır. Bunun en önemli nedeni her bir karar tablosunun söz konusu problem alanıyla çok yakından ilişikli olması, birkaç tablo için geliştirilen genellemenin, bir başka tabloda geçersiz olmasıdır. Yani sadeleştirme işlemi ele alınan konuya yakından bağlı olup, sadeleştirmenin derinliği tamamen programcının deneyimine ve ele aldığı problemi ne kadar iyi analiz ettiğine

de bağlıdır. Zaten bu çalışmanın da amacı, ele alınan problemin tanımını ve yeni tasarımı eksiksiz gerçekleştirme konusunda programcıyı eğitmeye yardımcı olmaktır.

4. Karar Tablosu Sadeleştirme Yöntemi

Eğer bir karar tablosu, Karnaugh tablosu sadeleştirmesinde olduğu gibi, bir boolean ilişkisi şeklinde ele alınıp, kurallar birbirinden bağımsız biçimde değerlendirilebilir ise, karar tablosu tıpkı bir mantık devresi sadeleştirmesi gibi ele alınabilir (Shwayder, 1975; Vastianos, 2001). İzleyen kesimde bir karar tablosunun Quine-McCluskey Algoritması ile sadeleştirilmesindeki yöntem kısaca ele alınacaktır.

Mantıksal tasarımda basit boolean fonksiyonların sadeleştirilmesinde kullanılan en temel algoritma Karnaugh haritası algoritmasıdır. 6'dan fazla değişken içeren fonksiyonların görsel olarak Karnaugh haritasında sadeleştirilmesi ya da bu algoritmanın bir programlama dili aracılığı ile bilgisayar ortamına aktarılması oldukça zordur. Bu nedenden dolayı Karnaugh algoritmasına alternatif bir algoritma Quine-McCluskey algoritması geliştirilmiştir (Vastianos, 2001). Quine-McCluskey Algoritması asal bileşenlerin (prime implicant) bulunmasına ve sadeleştirilmesine sistematik bir yaklaşım getirir. Aynı zamanda, Quine-McCluskey Algoritmasının, bir programlama dili ile modellenmesi daha kolaydır. Quine-McCluskey Algoritmasının heuristik bir yaklaşım yerine kesin sonuç üretmesi, elimizdeki verileri sadeleştirmek için en büyük tercih sebebidir.

Quine-McCluskey Algoritması $A + \overline{A} = 1$ kanununun sadeleşecek ifadeye tekrar tekrar uygulamasıdır. A bir boolean değişken ise doğruluk tablosu ele alındığında A, 0 veya 1 değerlerinden birini alabilir. $A + \overline{A}$ ifadesi; A'nın 0 değeri için $0 + 1 = 1$ ve A'nın 1 değeri için $1 + 0 = 1$ sonucunu verecektir. Algoritmanın hedefi fonksiyondaki bütün mintermleri temsil eden minimum sayıda terim bulmaktır. Bunun için öncelikle asal bileşenler bulunur. Daha sonra temel bileşenler bulunur. Son olarak kalan temel bileşenleri temsil eden minimum küme seçilir ve sonuca ulaşılır. Örneğin Tablo 2'de verilen dört koşullu karar tablosunu ele alalım.

											0	1	2	3	4	5
Koşul 1																
Koşul 2																
Koşul 3																
Koşul 4																
İşlem 1																
İşlem 2																
İşlem 3																

Tablo 2. Sadeleştirme için örnek bir karar tablosu

Programcı yukarıdaki tabloyu inceleyerek sadeleştirme işlemini yapmalıdır. Örneğin, tablodaki ilk sekiz kuralda koşul1 "H" görünmektedir. Programcı ilk 8 kuralı sadeleştirerek Tablo 3'de görülen karar tablosunu oluşturmuştur. Programcı yazılımı kullandıktan sonra ortaya çıkan nihai karar tablosunu (Tablo 4) kendi sadeleştirdiği karar tablosuyla karşılaştırarak, yapmış olduğu sadeleştirme işlemindeki eksikliklerini görebilecektir.

				0	1	2	3	4	5
Koşul 1									
Koşul 2									
Koşul 3									
Koşul 4									
İşlem 1									
İşlem 2									
İşlem 3									

Tablo 3. Programcının Tablo 2'deki karar tablosu üzerindeki sadeleştirme işleminden sonra ortaya çıkan karar tablosu

Tablo 2'deki H'ler 0, E'ler 1 olarak dönüştürülüp, bu tablo Quine-McCluskey Algoritması ile sadeleştirildiğinde, kuralların en son durumu aşağıdaki gibidir.

0, 1, 2, 3, 4, 5, 6, 7 □ 0 – – –
10, 11, 14, 15 □ 1 – 1 –
8, 9, 12, 13 □ 1 – 0 –

Yukarıdaki sadeleştirme sonucu, aynı işleme karşılık gelen kuralların listesi olup, bunların karar tablosu şeklindeki nihai gösterimi Tablo 4'te verilmiştir. Görüldüğü gibi ilk 8 kural sadeleşerek tek bir kurala (H,-,-,-)indirgenmektedir. Benzer şekilde 10,11,14 ve 15 nolu kurallar ile 8,9,12 ve 13 nolu kurallar aynı işleme karşılık geldiklerinden sadeleştirme sonucu birer kural, sırasıyla (E,-,H,-), (E,-E,-) elde edilmektedir. Sonuç itibariyle 16 kural ile başlanan karar tablosu 3 kurala indirgenmiştir.

	0	**8**	**10**
Koşul 1	H	E	E
Koşul 2	–	–	–
Koşul 3	–	H	E
Koşul 4	–	–	–
İşlem 1	X		
İşlem 2			X
İşlem 3		X	

Tablo 4. Tablo 2'nin sadeleştirilmiş durumu

Doğal olarak sadeleştirmedeki başarı oranı doğrudan doğruya problemin yapısına, yani kuralların içerdiği değerle ilgilidir. Ancak amaç program geliştirme süreci eğitiminde problem alanının başlangıçta verildiği halinin nereye kadar sadeleştirilebileceği becerisine katkı sağlamak, böylece gereksinme, analiz ve tasarımın altyapısı eğitimini pekiştirmeye katkı sağlamaktır. Bu çalışmada kullanılan yöntem her problemde mutlaka büyük oranda sadeleştirme sağlamaz. Bununla beraber, yukarıda ele alınan sadeleştirme yöntemi genel olarak başarılı sonuçlar vermektedir. Örneğin Tablo 5'te farklı sayıdaki koşul adedi (A) ve tüm bu koşullar için aynı işlemle sonuçlanan kural adedi (B) girildiğinde, sadeleştirme sonucu ortaya çıkan kural adedi (C) ve sadeleştirme oranı verilmiştir (Vastianos, 2001).

A	B	C	Sadeleştirme Oranı (%)
4	12	5	58
6	40	12	70
64	64	18	72

Tablo 5. Sadeleştirme başarı örnekleri

5. Geliştirilen Yazılım

Program geliştirme sürecinin gerek çözümleme, gerekse tasarım aşamalarında deneyim kazanılmasında karar tablolarının kullanımı için BAK isimli bir yazılım geliştirilmiştir. Bu yazılımın temel amacı, programcıya üzerinde çalıştığı konuyu ne kadar iyi çözümlediği ve tasarımın ne kadar geçerli olduğu konusunda etkileşimli olarak yardımcı olmaktır. Böylece yazılım geliştirme sürecindeki, tamlık, tutarlılık, doğruluk ve tekrarlardan arındırılmış nitelikte bir yazılım geliştirilmesi becerilerinin pekiştirilmesine katkı sağlanabilecektir. BAK yazılımının çalışma yöntemi tıpkı karar tablosu kullanımı metodolijisindeki gibidir. Buna göre, programcı öncelikle problem alanıyla ilgili her türlü çalışmayı yapar. Konuyu her yönüyle öğrenmeye ve anlamaya çalışır. Programcı BAK'ı kullanmaya hazır olduğunu kabul ettikten sonra:

1. Adım: BAK menüsünden "Karar Tablosu" seçilir. Bunun ardından "Yeni", "Aç", "Sakla" ve "Sil" seçenekleri de gelmektedir. İlk kez başlarken "Yeni" seçeneği seçilir. Ayrıca çalışmanın herhangi bir aşamasında menünün diğer seçenekleri de uygulanabilir.

2. Adım: BAK menüsünden "Koşullar" seçilir. Bunun ardından, "Koşul ekle", "Koşul sil" ve "Koşul içerik değiştir" seçenekleri gelir. Öncelikle "Koşul ekle" seçilip teker teker tüm koşullar girilir. Değiştirilecek veya iptal edilecek koşul girişleri var ise, daha sonra bu menüden gereken değişiklikler yapılabilir.

3. Adım: BAK menüsünden "İşlem Girişleri" seçilir. Bunun ardından "İşlem ekle", "İşlem sil" ve "İşlem değiştir" seçenekleri gelir. Öncelikle "İşlem ekle" seçilip teker teker tüm işlemler girilir. Değiştirilecek veya iptal edilecek işlem girişleri var ise, daha sonra bu menüden gereken değişiklikler yapılabilir.

4. Adım: BAK menüsünden "Tüm koşul seçeneklerini belirle" seçilir. Bu işlemin ardından 2. adımdaki koşul adedine bağlı olarak olası tüm koşul seçenekleri, örneğin, Tablo 2'deki gibi, kurallar olarak belirlenir. BAK yazılımıençok sekiz adet koşul girişine yani toplam olarak 256 adet koşul kuralına kadar çalışabilir. Bu miktar, eğer iyi bir analiz çalışması yapılmış ise, birçok en alt düzey alt sistemin (modülün) problem alanını özelliklerini karşılar.

5. Adım: BAK menüsünden "Kural gir" seçilir. Bu durumda 4. adımda belirlenmiş tüm koşul seçenekleri teker teker ekrana gelir ve programcı bunlar için hangi işlemin uygulanması gerektiğine karar verir. Böylece kurallar belirlenmiş olur. Programcı gereksiz veya anlamsız bulduğu bazı kuralları boş geçebilir. Doğal olarak eğer koşul miktarı çok ise, bu adım biraz uzun sürebilecektir. Ancak istenildiği zaman 1.adımdaki "Sakla" ile tablo o haliyle saklanabilir, "Aç" ile de daha sonra kullanıma devam edilebilir.

6. Adım: BAK menüsünden "Sadeleştir" seçildiğinde, boş işlem içeren tüm kurallar tablodan çıkartılır. Daha sonra menüden "Silerek sadeleştir" seçilir. Bu işlemin ardından "Kural no gir" gelir ve programcı iptal edilecek kuralı numarasını girer. Bu menüde bir de "İçerik sadeleştir" seçeneği vardır. Bu seçildiğinde "Başlangıç-bitiş kural no gir" mesajı gelir. Verilen kural numaralarına ilişkin daha önceden girişi yapılmış olan koşul seçenekleri ve ilgili işlemler ekrana gelir. Programcı sadeleştirmeyle ilgili gereken değişiklikleri yapar ve tabloyu 1. adımdaki menüyü

kullanarak saklar veya 6. adımda devam eder. Bu adımdaki "Sadeleştirme bitti" seçeneği ile tablonun son hali saklanır.

7. Adım: BAK menüsünden "Otomatik sadeleştir" seçildiğinde, Quine-McCluskey Algoritmasıyla tablo sadeleştirilir. Programcı, "Sadeleştirilmiş tabloyu göster" seçeneğini seçtiğinde, sadeleştirme sonucu oluşan, örneğin Tablo 4'teki gibi, tüm kurallar ekrana getirilir. "Karşılaştır" seçeneği seçildiğinde ise, yazılımcının 6. adımda kendisinin yaptığı sadeleştirilmiş tablo ile Quine-McCluskey yöntemiyle sadeleştirilen tablo sekizerlik kurallar halinde yan yana ekranda gösterilir. Böylece programcı hangi kurallarda daha fazla sadeleştirme olanağı olabildiğini, problem konusunda eksik ve yanlış anlamaları görme olanağı bulur.

6. Uygulama ve Sonuç

Bu çalışmada, program geliştirme sürecinin analiz ve taşarım aşamalarında yardımcı bir araç olarak kullanılabilecek karar tablolarından programcılık deneyimi kazanılmasında nasıl yararlanılabileceği üzerinde durulmuştur. Bu amaçla BAK isimli bir yazılım geliştirilmiştir. Söz konusu yazılım kullanılarak, programcı problem alanıyla ilgili belirlediği tüm koşullar ve bunlara bağlı işlemleri bir karar tablosu şeklinde ifade eder ve bunu sadeleştirmeye çalışır. Programcının yapacağı sadeleştirmenin derinliği, kendisinin problem alanı hakkındaki bilgi düzeyi ve deneyimi ile yakından ilişkili olacaktır. Böylece analiz çalışmasını ne kadar doğru yaptığını görebilecek, en uygun tasarımı ortaya koyabilecektir. Programcının bu çalışması, Quine-McCluskey Algoritması kullanılarak yapılan tablo sadeleştirmesi ile karşılaştırıldığında, programcı problem alanı konusundaki yetkinliğini görebilme olanağına kavuşabilecektir. Bu çalışmada ele alınan yöntem, test amaçlı çeşitli karar tablolarında denenmiş ve sadeleştirmede başarılı sonuçlar alınmıştır. Sadeleştirme ile problem alanının içindeki ilişkiler çok daha iyi görülebilecektir. Bu sayede, programcı adaylarının problem alanlarını analiz etme ve tasarlama becerilerine, doğal olarak da etkin program geliştirme becerilerine olumlu katkı sağlanmış olacaktır. Programcı problem alanını çok iyi kavramış ve bunu karar tablosuna doğru olarak yansıtabilmiş ise, sadeleştirme sonucu ortaya çıkan tablo en kısa program koduna aday olacaktır. Bu çalışmada geliştirilen BAK yazılımı daha da geliştirilerek, sadece program geliştirme deneyimi kazanılmasında değil, doğrudan program geliştirme sürecinde de kullanılabilir.

KAYNAKÇA

Al-Koraishi F., "Decision table preprocessor for PASCAL", Dissertation Univ. Wales, Swansea, UK,1989.

CODASYL, A Modern Appraisal of Decision Tables, Report of the Decision Table Task Group, ACM, New York, 1982.

Colomb R., Chung C., "Very fast decision table execution of propositional expert systems", Eighth National Conference on Artificial Intelligence, MIT Press, Cambridge, MA, USA, Vol.2, p.671-676, 1990.

Cragun B., Steudel H., " A decision table based processor for checking completeness and consistency in rule based expert systems", Int. J. Of Man Machine Studies 26, p.633–648, 1987.

Cyril G. Veinott, "Programming decision tables in FORTRAN, COBOL or ALGOL", Communications of the ACM, v.9 n.1, p.31–35, Jan. 1966.

Dwyer Barry, "How to Write Cope Decision Tables",

http://www.cs.adelaide.edu.au/users/dwyer/COPE-MAN.html (Erişim tarihi Kasım 2007).

Dwyer Barry, "Experience With The Cope Decision Table Processor",

http://www.cs.adelaide.edu.au/users/dwyer/Cope.html (Erişim tarihi Ocak 2008).

Lewis T. Reinwald, Richard M. Soland, "Conversion of Limited-Entry Decision Tables to Optimal Computer Programs II: minimum storage requirement", Journal of the ACM (JACM), v.14 n.4, p.742–756, Oct. 1967.

Merlevede P, Vanthienen J., "A structured approach to formalization and validation of knowledge", Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert System Programs, IEEE Comput. Soc. Press, Los Alamitos, CA, USA, p. 149–158, 1991.

Moreno Garcia A.M., M.Verhelle & J.Vanthienen, "An Overview of Decision Table Literature 1982–2000", Fifth International Conference on Artificial Intelligence and

Emerging Technologies in Accounting, Finance and Tax, November 2–3, 2000, Huelva, Spain.

Shwayder Keith, "Combining Decision Rules in a Decision Table", Communications of the ACM, 18(8), p:476–480, 1975.

Vanthienen J. "A more general comparison of the decision table and tree", Communications of the ACM, Vol 37, Nr. 2, p.109–113, 1994.

Vanthienen J., Dries E., "Illustration of a decision table tool for specifying and implementing knowledge base systems", International Journal on Artificial Intelligence Tools, Vol 3, Nr 2, p.262-288, 1994.

Vanthienen Jan, "Ruling the Business: About Busines Rules and Decision Tables", Vandenbulcke J. & Snoeck M. (Eds), New Directions in Software Engineering, Leuven University Press, Leuven, p.103–120, 2001.

Vastianos George, "Boolean functions' minimisation software based on the Quine-McCluskey method", Encoder Magazine, Seattle Robotics Society, Issue Number 40, May/July 2001.

Wets, G. & Vanthienen J., "Interfacing Decision Tables with Knowledge Acquisition Formalisms", The Second World Congress on Expert Systems (WCES), January 10-14, 1994, Lisbon, Portugal.

Wets G., Vanthienen J., Piramuthu S., "Extending a tabular knowledge-based framework with future selection", Expert Systems With Applications, 13(2): 109-119, Aug. 1997.