İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi Yıl:4 Sayı:7 Bahar 2005/1 s.103-121

YAZILIM GELİŞTİRME PROSESİNDE KISITLAR TEORİSİNİN DÜŞÜNCE SÜREÇLERİNİN KULLANILMASI

Gülşen AKMAN[*] Çağın KARAKOÇ[**]

ÖZET

Kısıtlar teorisi her sistemde, o sistemin performansını etkileyen en az bir kısıtın olduğunu ve sistemdeki darboğazların bu kısıttan kaynaklandığını savunan bir yönetim felsefesidir. Teorinin genel ilkeleri hem imalat hem de hizmet sistemlerinin performansını geliştirmek için başarılı bir şekilde uygulanabilir. Bu çalışmada kısıtlar teorisi yöntemlerinin yazılım geliştirme prosesinin performansını iyileştirmek için nasıl kullanılabileceğinin gösterilmesi amaçlanmaktadır. Bunun için, yazılım geliştirme prosesindeki darboğazların tespiti ve giderilmesi ve prosesin performansının artırılması konusunda kısıtlar teorisinin düşünce süreçlerinin, yani sorun çözme araçlarının kullanımına ilişkin örnek bir uygulama gerçekleştirilmiştir.

Anahtar Kelimeler: Kısıtlar Teorisi, Düşünce Süreçleri, Yazılım Geliştirme Prosesi

USING OF THINKING PROCESSES OF TOC (THEORY OF CONSTRAINTS) AT SOFTWARE DEVELOPMENT PROCESS

ABSTRACT

Theory of constraint (TOC) is a management philosophy that advocate that every system has at least one constraint which limits the system from achieving high level performance. General principles of TOC can be applied successfully to improve performance of manufacturing and service organizations. Aim of the study is to demonstrate how methods of TOC is used to improve performance od sofware development process. Therefore, to define and to eliminate bottlenecks in sofware development process, and to increase performance of the process , a study that explain using of thinking processes of TOC, that is problem solving methods, is carried out.

Keywords: Theory of Constraints, Thinking Processes, Software Development Process

* *Kocaeli Üniversitesi, Mühendislik Fakültesi, Endüstri Müh. Böl., Kocaeli,* gulsenakman@yahoo.com
**Kocaeli Üniversitesi, Mühendislik Fakültesi, Endüstri Müh. Böl., Kocaeli,* caginkarakoc@yahoo.com

1. GİRİŞ

Ürün geliştirme ve yenilik özellikle son yıllarda çeşitli pazarlarda rekabet edebilmenin en temel anahtarları olarak tanımlanmaktadır. Günümüzde pazarlarda tahmin edilebilir ürün geliştirme sürelerine ve daha düşük maliyetlere sahip, daha yüksek kaliteli ve yüksek performanslı ürünlere olan talep artmaktadır. Yeni ürün geliştirme faaliyetlerinden en iyi performansı sağlamak için ürün geliştirme prosesinin etkin ve verimli bir şekilde yönetilmesi gerekmektedir. Ancak, yeni geliştirilen ürünlerin başarısız olma olasılığı her zaman mevcuttur ve getirdiği başarısızlık maliyeti de oldukça yüksektir. Bunları önlemek için gerekli önlemler daha prosesin başında alınmalıdır (Büyüközkan ve Feyzioğlu, 2003). Bu noktada kısıtlar teorisi, ürün geliştirme prosesinin etkinliğini artırmayı ve gerekli önlemleri almayı sağlayacak bir yöntem olarak karşımıza çıkmaktadır.

Kısıtlar teorisi, hem imalat hem de hizmet sistemlerinde başarılı bir şekilde kullanılan örgütsel bir yönetim felsefesidir (Stein,1997). Kısıtlar teorisi, herhangi bir sistemin performansının artırılması için, sistem performansını olumsuz yönde en çok etkileyen faktörün bulunması, yönetilmesi ve ortadan kaldırılması konusunda oluşturulmuş yönetim felsefeleri, disiplinleri ve sektörlere özel en iyi uygulamaları içeren bir felsefedir (Goldratt ve Cox,1992).

Bu çalışmanın amacı, ürün geliştirme sürecinde ortaya çıkabilecek engelleri ortadan kaldırmak için kısıtlar teorisi metodlarını kullanarak sürecin verimli olmasını sağlamak, performansını artırmak ve böylece kısıtlar teorisinin uygulama alanını genişletmektir. Örnek uygulama yazılım geliştirme prosesi için gerçekleştirilmiştir. Yazılım sektöründe ürün ömürleri çok kısadır. Bu yüzden yazılım sektöründe faaliyet gösteren firmalar için zaman ve rakiplerine göre daha hızlı davranmak başarı için en önemli faktördür. Bu çalışmada, yazılım geliştirme sürecinin etkinliğini ve hızını olumsuz yönde etkileyen engellerin tespiti ve ortadan kaldırılması için, kısıtlar teorisinin düşünce süreçlerinden faydalanılmıştır. Böylece hem zaman hem de paradan tasarruf edilmesinin sağlanması ve firmalara yazılım geliştirme sürecinin performansını artırmaları için yol gösterilmesi hedeflenmiştir.

2. KISITLAR TEORİSİ KAVRAMI VE TANIMI

Kısıtlar teorisi (KT), 1980'lerin başında Dr. Eliyahu M. Goldratt tarafından geliştirilen bir yönetim sistemi felsefesidir ve temel çıkış noktası, bir firmanın performansını kısıtların belirlediği ve her sistemin en az birkaç tane kısıta sahip olduğudur (Ruhl, 1996). Atwater ve Gagne'e göre (1997) kısıt "bir sistemin hedefi

ile ilgili olarak, performansı sınırlayan her şey" olup, kısıtlar teorisi, "sistemdeki kısıtların yönetilmesi yoluyla gelişmeye odaklanan bir yönetim yaklaşımıdır (Atwater ve Gagne, 1997). KT'nin temel noktası ise; geleneksel düşüncenin tersine, her kısıtın aslında birer ilerleme fırsatı olmasıdır. KT, kısıtları pozitif olarak değerlendirir, çünkü kısıtlar bir sistemin performansını tanımlarlar ve sistem kısıtlarının aşama aşama ortadan kaldırılması sistemin performansını artırır (Spencer ve Cox, 1995).

KT, şemalar çizerek, kısıtların ve problemlerin esas nedenlerini bulmak ve problemi ortadan kaldırmak için adımlar geliştirmeyi kapsar (Rudy, 1996). Amaca ulaşmayı engelleyen noktaları belirlemeyi ve bu noktaları ortadan kaldırmak için gerekli değişiklikleri uygulamayı sağlayan bir yönetim disiplinidir (Schucavage, 1995). KT'nin kısıt odaklı yaklaşımı mantıksal ve pragmatiktir. Kısıtları tanımlamak ve belirlemek organizasyonlara katkılarını artırmalarına yönelik olarak en hızlı ve düşük maliyetli çözümler sunar. KT yaklaşımı bir organizasyonu yönetirken hem bugünü sağlam temellere dayandırma hem de yarına ışık tutma olanağı verir.

KT'ye göre endüstriyel bir örgütün amacı şimdi ve gelecekte para kazanmaktır. KT, bir organizasyonun performansını doğru bir şekilde ölçmek için iki tür ölçüt kullanır; finansal ölçütler, operasyonel ölçütler. Finansal ölçütler, net kar, yatırım getirisi (ROI) ve nakit akışından oluşmaktadır. Operasyonel ölçütler ise, katkı (throughput), envanter ve faaliyet giderleridir. Organizasyonlar para kazanma amacını gerçekleştirmek için katkıyı artırmalı ve eş zamanlı olarak envanter ve faaliyet giderlerini azaltmalıdır (Kendall, 1998; Verma, 1997; Goldratt ve Cox, 1992).

Teorinin temelinde; sistem kısıtlarını tanımlama ve bu kısıtların üretim süreçleriyle eşzamanlı olarak nasıl çalışabileceğinin kararını vermek yatar (Rand, 2000). Sistem kısıtları, çalışanlar, makineler, şirket politikaları ve şirketi etkileyen kurallar olabilir. Kısıt, sistemin amacını gerçekleştirme sürecinde daha yüksek performansa ulaşmasını sınırlandıran herhangi bir şey olarak tanımlanabilir .

KT'nin temelinde beş ana ilke bulunmaktadır (Womack ve Flowers, 1999)

1. Tüm sistem ve prosesler birbirine bağlı olayların bir dizisidir; zincire benzerler.
2. Tüm sistemler bünyelerinde en az bir kısıt barındırırlar. Kısıt, zincirin en zayıf noktası ya da süreçteki darboğazdır.
3. Kısıt performansındaki herhangi bir iyileşme doğrudan sistemin bütününde performans artışını sağlar. Zayıf noktanın güçlendirilmesi ile, zincirin tamamı daha güçlü hale gelir. Darboğaza doğru akışın arttırılması ile de, sistemin çıktı miktarı artar.

4. Kısıtlar nedenlerine göre sınıflandırılabilir. Kısıtların çoğu organizasyonel kurallar, eğitim ve ölçülerdir. Bunlar politik kısıtlardır. Diğer kısıtlar ise kaynak ve pazar kısıtlarıdır.
5. Kısıtlı olmayan bir kaynak veya süreçte yapılacak herhangi bir iyileştirme sistem performansını etkilemeyecektir.

Her sistemin en az bir kısıtı olduğu ve sistem performansının da bu kısıt tarafından yönetildiği gerçeğinden hareketle organizasyonun verimliliğini ve karlılığını artırmak için bu kısıtlara odaklanılması gerekir. Temel olarak, bir sistemin karşılaşabileceği iki tür kısıt bulunmaktadır. Fiziksel kısıtlar (malzeme, makine, teçhizat, insan, talep) ve politik kısıtlar (firma faaliyetlerini aksatacak veya engelleyecek politikalar, prosedürler, kurallar ve yönetim metodları) (Rahman, 1998). Organizasyonlar genel olarak çok az fiziksel kısıtla karşılaşırlar. Karşılaşılan kısıtların büyük bir kısmı politiktir. Goldrat'a göre bir organizasyondaki kısıtların %99'u politik kısıtlardır (Goldratt, 1998). Bu gibi kısıtlar, hatalı ve modası geçmiş karar verme sistemlerinin ürünüdür.

Kısıtların yönetilmesi ve ortadan kaldırılması için, KT beş adımlı bir gelişme süreci izler. Bu süreç şu adımları içermektedir (Gardiner vd., 1994, Siha, 1999).

1. Sistem kısıtlarını tanımlamak; sistemde ne tür kısıtların olduğu bilinmesine rağmen, bir sistem yüksek bir performans gösteremeyebilir, bu nedenle kısıtlar tespit edilerek uygun kontrol mekanizmaları tasarlanmalıdır.
2. Kısıtların nasıl işletileceğine karar vermek, kısıtları olan sistemin performansının nasıl arttırılacağına karar vermek; Örneğin sistemin içindeki fiziksel kısıtlar en karlı ürünleri üretmek için programlanmalıdır.
3. Başka bütün her şeyi 2. adımdaki karara bağımlı kılmak ; kısıt olmayan durumlar sistemin maksimum performansını sınırlandırmaz. Kararları etkileyen kısıtlar kararları etkilemeyen durumlardan daha çok önceliğe sahip olmalıdır.
4. Kısıtları ortadan kaldırmak; İlk üç adım tamamlandıktan sonra, sistem performansındaki gelişmeler bir kısıtın değişmesini veya ortadan kalkmasını gerektirir. Örneğin darboğaz yaratan bir makinanın kapasitesini artırmak sistemin kapasitesini artırır.
5. Adım 1'e geri dönmek; Bir kısıt ortadan kalktıktan sonra, yeni bir sistem kısıtı incelenmelidir. Yeni bir kısıt tanımlamak için Adım 1'e dönülür.

KT'nin bu temel ilkeleri çeşitli üretim sistemlerinde başarılı bir şekilde uygulanmıştır. Üretim ve hizmet sistemleri farklı özelliklere sahip olmalarına rağmen bu ilkeler hizmet sistemlerinde de başarılı bir şekilde uygulanabilir (Siha, 1999)

KT, kısıtların ortadan kaldırılmasına yönelik olarak sistemin ana problemleri üzerine yoğunlaşan, alternatif çözümler sunan çeşitli araçlar kullanmaktadır. Bu araçlara bütün olarak düşünce süreçleri adı verilmektedir. Düşünce süreçleri (DS), sistemin performansını sınırlandıran kısıtın incelenmesi, çözüm önerilmesi, çözümlerin önkoşullarının bulunması ve uygulanması sırasında karşılaşılabilecek güçlüklerin DS yöntemleri kullanılarak ortadan kaldırılmasını içerir (Köksal ve Karşılıklı, 2000).

Yapılan araştırmalar, organizasyonel değişim prosesinin başarılması en zor proses olduğunu göstermektedir. Gerekli değişimleri kolaylıkla ve başarılı bir şekilde gerçekleştirebilmek için "düşünce süreçleri" yaklaşımı geliştirilmiştir. DS'nin amacı, bir organizasyonun mevcut durumunu geliştirmek için gerekli faaliyetleri tanımlamak, belirsiz durumlara çözüm üretmektir (Stein, 1997). Düşünce süreçlerinin temelinde 3 soru bulunmaktadır.

Ne değişecek? Bir organizasyonun/sistemin geliştirilmesi değişimi gerektirir, fakat değişim her zaman gelişmeyle sonuçlanmaz. Bazen kötü sonuçlara neden olabilir. Değişim sadece doğru bileşene odaklandığında gelişme ile sonuçlanır. Bu nedenle neyin değiştirileceğini belirlemek çok önemlidir (Choe ve Herman, 2004). Bu soru ile, organizasyonun performansını artırmayı ya da performansını geliştirmeyi engelleyen, yanlış politikalar ve etkenler tespit edilir. Bunun için sonuç-neden-sonuç tekniği kullanılır. Burada kısıt olarak ifade edilen durumlar, arzu edilmeyen sonuçlardır. Ne değişecek sorusu, kısıtlar teorisi uygulanarak organizasyonel bir kısıtın, yani performansı engelleyen temel problemin tanımlanmasına öncülük eder. Bu amaçla mevcut gerçeklik ağacı yöntemi kullanılır.

Neye dönüşecek? Bu aşamada kök problem için mantıklı, basit ve pratik çözümler araştırılır. Kısıtlar teorisi, gerçek dünyada basit çözümlerin, problemleri ortadan kaldırma gücüne sahip olduğunu vurgulamaktadır. Bu aşamada amaçlanan, çözüm üretmeyen politikaların neye dönüşmesi gerektiğini belirlemektir. Bunun için, buharlaşan bulut ve gelecek gerçeklik ağacı yöntemleri kullanılır.

Dönüşüm nasıl gerçekleştirilecek? Bu aşamada, çözümün nasıl gerçekleştirileceği sorusunun yanıtı aranır. Bunun için, ön gereksinim ve geçiş ağacı kullanılır.

Bu sorular problem çözme tekniklerinin de esasını oluşturur. Bu soruları cevaplamak için temel olarak neden-sonuç diyagramlarına dayanan araçlar kullanılır. Düşünce süreçlerinde sorular, amaçlar ve kullanılan yöntemler Tablo 1'de görülmektedir.

Tablo 1.DS'de Kullanılan Yöntemler (Rahman, 1998)

JENERİK SORULAR	AMACI	YÖNTEMLER
Ne değişecek?	Temel problemlerin tanımlanması	Mevcut gerçeklik ağacı
Neye Dönüşecek?	Basit ve pratik çözümler geliştirmek	Buharlaşan Bulut Gelecek Gerçeklik ağacı
Dönüşüm Nasıl gerçekleşecek?	Çözümlerin uygulanması	Ön gereksinim ağacı Geçiş ağacı

DS araçları değişimi yönetmek amacıyla kullanılırlar. Tablo 1'de de görüldüğü gibi beş temel araçtan oluşurlar. Bunlar; mevcut gerçeklik ağacı, buharlaşan bulut, gelecek gerçeklik ağacı, ön gereksinim ağacı, geçiş ağacı olarak sınıflandırılır.

Mevcut Gerçeklik Ağacı (MGA) : Düşünce süreçlerinin uygulanmasındaki ilk adım istenmeyen etkilerin listelenmesi ve bunlara göre mevcut gerçeklik ağacının oluşturulmasıdır. MGA bir sistemin mevcut durumunu analiz etmek ve problemleri daha iyi anlamak için oluşturulur ve sistemin performansını azaltan istenmeyen etkilere sahip temel problemleri tanımlar (Pfeifer ve Tillmann, 2003). MGA, istenmeyen etkiler ve onların sonuçları arasındaki neden-sonuç ilişkilerini gösteren bir diyagramdır. Amaç, problem yaratan kök nedeni bulmaktır. Öncelikle kök neden bulunur ve ortadan kaldırılır. Böylece istenmeyen etkiler yok olur (Stein, 1997). MGA, istenmeyen bir sonuçtan temel nedene ulaşıncaya kadar birbirine bağlı nedenlerin ve sonuçların oluşturduğu bir çözüm yöntemidir (Özer, 2001).

Buharlaşan Bulut : İstenmeyen sonucu ortadan kaldırmak için önerilen çözümlerle temel ve ön gereksinimlerin tanımlandığı, çözümler arasındaki çatışmanın ortaya konduğu ve bu çatışmanın yok edilmesi için enjeksiyonun yapıldığı araçtır (Özer, 2001). Bu araç, tek bir problemin ayrı olarak ele alınmasını, karşılaşılan çatışmaların ve varsayımların belirlenmesini ve çözüm amacıyla incelenmesini içerir (Köksal ve Karşılıklı, 2000). Buharlaşan bulut yöntemi problemin yaşandığı mevcut durumdan

arzulanan gelecek duruma geçişte, problemlerin ortadan kaldırılmasına katkıda bulunarak etkili bir köprü görevi görmektedir (Davies vd., 2005)

Gelecek Gerçeklik Ağacı (GGA) : Geleceği hayal ederek canlandırmak ve tahmin etmek için kullanılan bir araçtır. Gelecek gerçeklik ağacı (GGA), mevcut sistemde yapılacak değişiklikler ile meydana gelebilecek sonuçlar arasındaki neden sonuç ilişkisini gösterir. GGA, bir what-if uygulamasıdır. GGA, bir organizasyon için strateji, vizyon veya bir planın resminin görülmesini sağlar (Mcmullen, 1998). Önerilen değişimin yararlarını, doğuracağı olumsuz etkileri ve bu etkilerin nasıl ortadan kaldırılacağını belirlemeye çalışır.

Ön Gereksinim Ağacı : Çözüm fikrinin önündeki tüm engellerin üstesinden gelmek için gerekli olan ikincil çözüm kümelerinin oluşturulması için mantıksal bir yol sunar (Özer, 2001). Amacı, büyük bir hedefe ulaşmak için ihtiyaç duyulan ara adımların tümünün tanımlanmasına yardımcı olmaktır (Rizzo, 2001). Ön gereksinim ağacının geliştirilmesi arzulanan sonuçlara ulaşmayı engelleyen lokal engelleri, durumları ve ihmalleri tanımlar ve bu engelleri ve değişime direncin üstesinden gelmeyi sağlayacak yeni hedefleri ve amaçları belirler (Davies vd., 2005).

Geçiş Ağacı (GA): Geçiş ağacı (GA), amaca ulaşmak için gerekli faaliyetlerin tanımlanmasında kullanılır. Arzu edilmeyen sonucun tanımlanmasından, değişimin tamamlanmasına kadar adım adım süreçleri ortaya koymak için tasarlanmış bir sebep-sonuç zinciridir (Özer, 2001). GA, satış, tahmin, çizelgeleme, yeni ürün geliştirme prosesleri gibi mevcut ve yeni oluşturulan süreçlerin belgelenmesinde, yeni yerleşim düzeninin oluşturulması, işletme süreçlerinin geliştirilmesi, stratejik hareketlerin belirlenmesi durumlarındaki değişimlerin zamanında yapılmasında kullanılmaktadır (Mcmullen, 1998).

3. YAZILIM SEKTÖRÜ VE YAZILIM GELİŞTİRME PROSESİ

Ulrish'in de ifade ettiği gibi tasarım ve yeni ürün geliştirme problemleri fiziksel ürünler, yazılım ve hizmet alanlarında ortak özellikler taşımaktadır (Ulrish, 2001). Son yıllarda bilgisayar yazılımlarının etkili kullanımı iş dünyasında başarılı olmanın en önemli belirleyicilerinden biri olmuştur (Büyüközkan ve Feyzioğlu, 2003).

Bilgi toplumunun kalbi olarak adlandırılan yazılım sektörü, ileri teknoloji sektörünün temelini oluşturmaktadır. Yüksek ürün ve proses yeniliği oranları, yüksek bilgi yoğunluğu, hızla kısalan ürün ömürleri, teknoloji yaşam döngüleri ve global pazara yayılmış bir değer zinciriyle karakterize edilebilir (Nambisan, 2002).

Yazılım sektörü hem ürün hem de hizmet sektörü arasında geçiş sağlamakta ve başarılı bir şekilde yer almaktadır.

Yazılım firmaları teknoloji esaslı firmalardır, çünkü geliştirdikleri veya dış kaynaklardan elde ettikleri ileri teknolojik bilgiyi yeni teknik çözümler üretmede kullanırlar. Bu anlamda yazılım firmalarının teknolojik kapasiteleri ve kaynakları, ürün geliştirme süreci için kritik önem taşımaktadır (Igel ve Islam, 2001). Yazılım sektöründe ürün ömrü, genellikle aylarla ve haftalarla ölçülebilir. Çok kısa ürün ömürleri, çok kısa firma ömrü ile sonuçlanabilir (Nambisan, 2002).

Bu özellikleri ile yazılım firmaları, ayakta kalabilmek ve varlıklarını devam ettirebilmek için, sürekli olarak yeni ürün yaratma ve bu ürünleri üretip müşteriye teslim etme süreçlerini geliştirme yeteneğine sahip olmalıdırlar (Igel ve Islam, 2001). Teknolojik kaynak ve kapasite oluşturma yeteneği, girişimci yazılım firmaları için çok önemlidir. Yazılım ürünleri geliştirmek için gerekli stratejiler müşteri ihtiyaçlarına odaklanma, teknik üstünlüğe önem verme, girişimci olmayı devam ettirme ve kıt kaynaklara odaklanmayı içerir.

Yazılım geliştirme prosesi, iş gereksinimlerinin yazılım ürünlerine dönüşmesi için yapılması gerekli olan faaliyetlerin tanımlanması ve bu gereksinimlerin bir yazılım ürününe dönüştürülmesidir (Schwalbe, 2000; Boutquin vd., 2000).

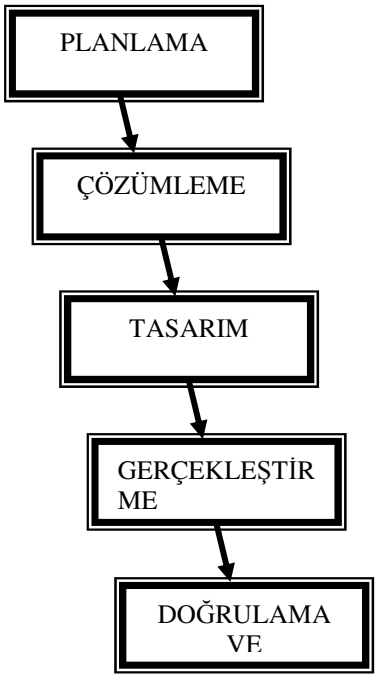
Yazılım geliştirme, yeni ürün geliştirmenin özel bir durumudur. Yazılım üretimi ve geleneksel üretim arasında önemli farklılıklar bulunmaktadır. Yazılım geliştirme tekrarlayan bir proses değildir. Ürün geliştirme ekibinin üyelerinin kişisel özellikleri ürünün kalitesini önemli bir şekilde etkilemektedir. Yazılım geliştirme prosesinin bir aşamasındaki başarısızlık olasılığı önceki adımlara son derece bağımlıdır (Örneğin çözümleme aşaması iyi bir şekilde tanımlanmamışsa, geliştirme aşamasında başarısızlık kaçınılmazdır) (Büyüközkan ve Feyzioğlu, 2003).

Yazılım geliştirme prosesini yalnızca bir kod yazma faaliyeti olarak düşünmemek gerekir. Proses, geliştirilen yazılım için test etme, bakım, dokümantasyon, kullanıcı eğitimi gibi faaliyetleri de kapsamaktadır (Balaban, 2004). Bir çok yazılım geliştirme metodu olmasına rağmen, bu çalışmada en yaygın kullanıma sahip olan Şelale (Waterfall) Metodu kullanılmıştır. Şelale Metodu bilinen en eski yazılım ürünü geliştirme tekniğidir.

Şelale metodu yazılım ürünü geliştirilmesinde doğrusal bir yaklaşım izler. Bu metotta yapılacak işler listelenerek safhalara ayrılır. Uygulama adım adım her safhanın tamamlanması ile gerçekleştirilir. Herhangi bir safhada yapılacak işlemler tamamlanmadan bir sonraki safhaya geçilmez. Uygulamanın herhangi bir adımında

kesinlikle kontrol veya test amaçlı geriye dönüş yapılmaz. Adından da anlaşılacağı gibi kayalardan aşağı akan suyun tekrar yukarı çıkamaması gibi adım-adım yaklaşım izlenir (Balaban, 2004). Şelale metoduna göre yazılım geliştirme prosesi beş aşamalı bir proses olarak tanımlanabilir (Şekil 1); planlama, çözümleme, tasarım, gerçekleştirme, doğrulama ve geçerleme (Arifoğlu ve Doğru, 2001).

Planlama aşaması, üretilecek yazılımla ilgili olarak personel ve donanım gereksinimlerinin çıkarıldığı fizibilite çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.



Şekil 1. Yazılım Geliştirme Prosesi (Arifoğlu ve Doğru, 2001)

Çözümleme aşaması, yazılım işlevleri ile gereksinimlerin ayrıntılı olarak çıkarıldığı aşamadır. Bu aşamada temel olarak mevcut sistemde var olan işler incelenir, temel sorunlar ortaya çıkarılarak, yazılımın yapabilecekleri vurgulanır. Temel amaç yazılım mühendisi gözüyle mevcut yapıdaki işlerin ortaya çıkarılması ve doğru olarak algılanıp algılanmadığının belirlenmesidir.

Tasarım; çözümleme aşamasından sonra belirlenen gereksinimlere yanıt verecek yazılım ya da bilgi sisteminin temel yapısının oluşturulması çalışmalarıdır. Bu çalışmalar, mantıksal tasarım ve fiziksel tasarım olarak iki gruba ayrılır. Mantıksal tasarımda, mevcut sistem değil, önerilen sistemin yapısı anlatılır. Olası örgütsel değişiklikler önerilir. Fiziksel tasarımda ise yazılımı içeren bileşenler ve bunların ayrıntıları yer alır.

Gerçekleştirme, Tasarım aşaması biten yazılımın kodlama, sınama ve kurma çalışmalarının yapıldığı aşamadır.

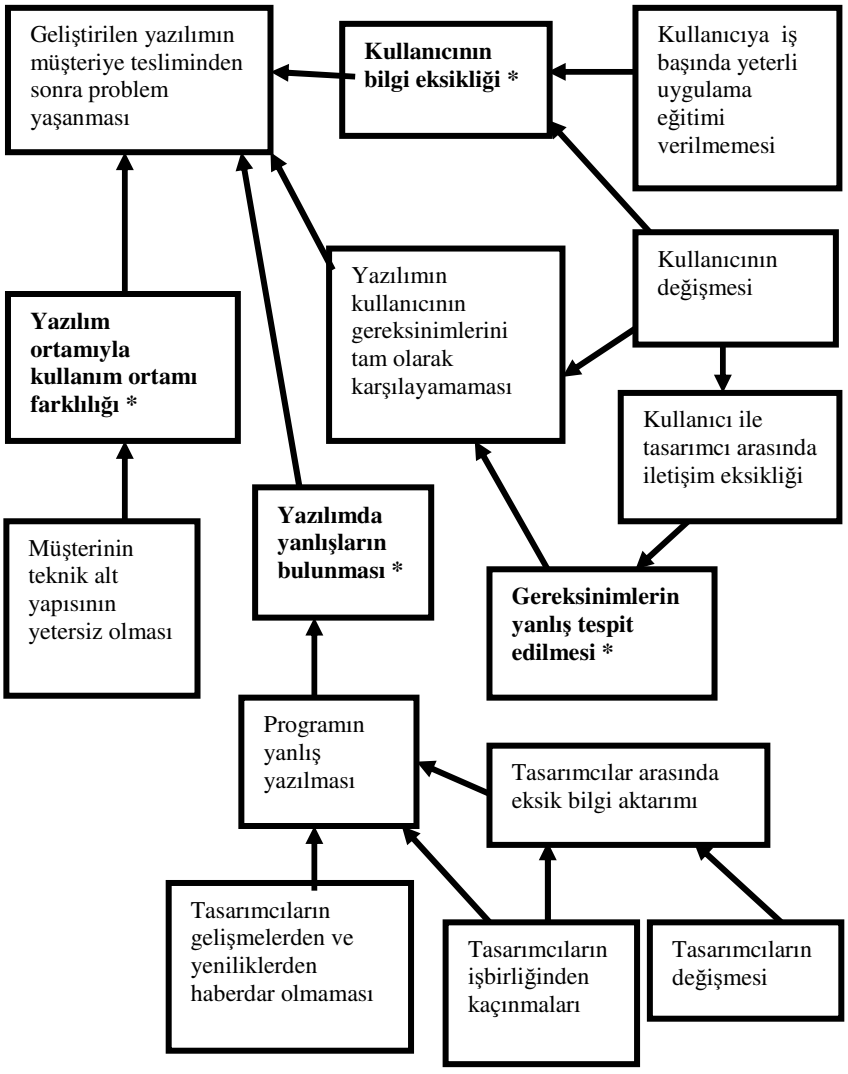
Doğrulama ve geçerleme; İşletime alınan yazılımla ilgili olarak, hata giderme ve yeni eklentiler yapma aşamasıdır. Bu aşamada doğrulama ile ürünü kullanacak kişilerin isteklerinin karşılanıp karşılanmadığı, geçerleme ile, ürünün içsel niteliğine ilişkin izleme ve denetim etkinlikleri gerçekleştirilir.

4. YAZILIM GELİŞTİRME PROSESİNDE KISITLAR TEORİSİ DÜŞÜNCE SÜREÇLERİNİN KULLANIMI

Bu çalışmada, beş yazılım firmasından yazılım geliştirme faaliyetlerinde karşılaşılan problemler konusunda yüz yüze görüşme yöntemiyle bilgi toplanmıştır. Bu bilgiler ışığında, yazılım geliştirme faaliyetlerinde firmaların yazılımın müşteriye tesliminden sonra bir takım problemlerle karşılaştıkları tespit edilmiştir. Yazılım geliştirme prosesi incelendiğinde, en çok yaşanan problemin, geliştirme ihtiyaçlarının belirlenmesi aşamasında ortaya çıkan aksaklıklardan dolayı olduğu görülmüştür. Bu çalışmada, problemi ortaya çıkaran ana nedeni bulmak ve çözmek için Kısıtlar Teorisinin düşünce süreçleri kullanılmıştır.

Düşünce süreçlerinin uygulanmasında ilk adım, istenmeyen etkilerin listelenmesi, sonra mevcut gerçeklik ağacının (MGA) oluşturulmasıdır. Daha önce de sözü edildiği üzere MGA, istenmeyen etkiler ve onların sonuçları arasındaki neden-sonuç ilişkisini gösteren bir diyagramdır. Amaç, kök nedeni bulmaktır. Bu neden bulunup ortadan kaldırıldığında istenmeyen etkiler yok olur. Yazılım prosesinde karşılaşılan ana kısıtlar; kullanıcının bilgi eksikliği, yazılımın kullanıcı ihtiyaçlarını tam olarak karşılayamaması, yazılım çevresiyle kullanıcı çevresinin farklı olması, kullanıcı ihtiyaçlarının doğru olarak belirlenememesi, yazılımda bir takım hatalar yapılması olarak sıralanabilir. Şekil 2'de mevcut gerçeklik ağacında da geliştirilen yazılımın müşteriye tesliminden sonra problem yaşanmasının kök nedenleri ortaya konmuştur. (*) işaretiyle belirlenmiş olan sorunlar problemi ortaya çıkaran temel sorunlardır.

Düşünce süreçlerinin ikinci adımı, buharlaşan bulutun oluşturulmasıdır. Buharlaşan bulut tekniği mevcut kök nedeni ortaya çıkaran problemlerin çözümlerinin ortaya konulmasında kullanılır.



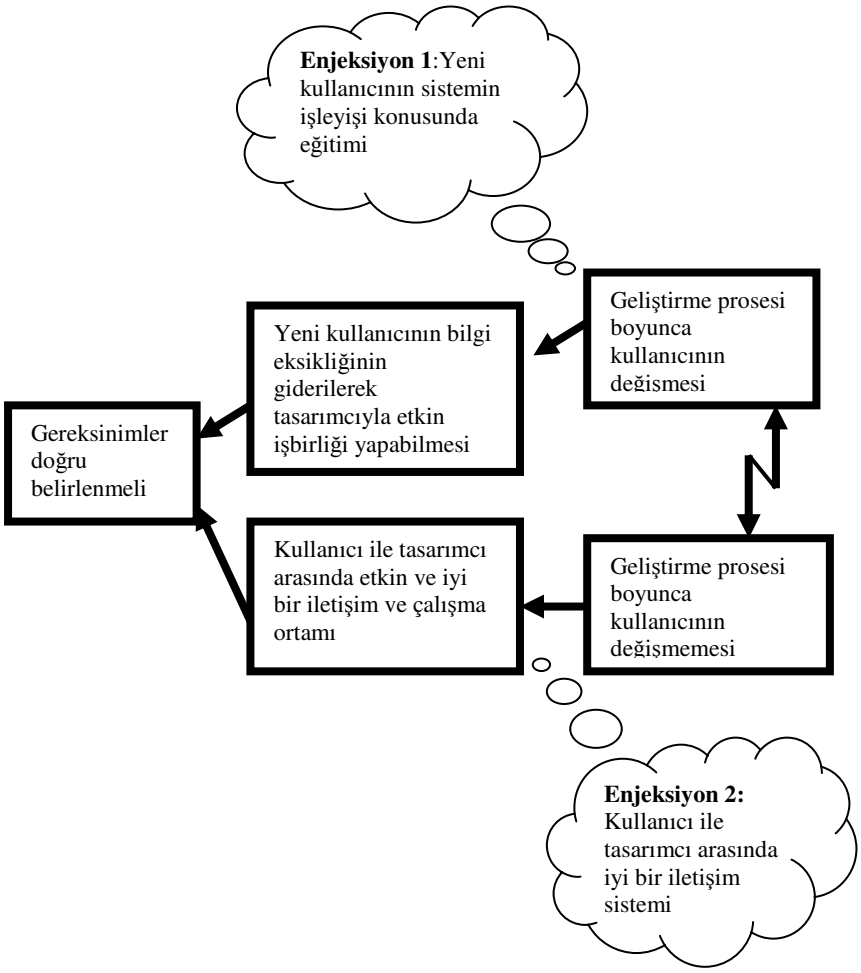
Şekil 2. Mevcut Gerçeklik Ağacı

Bu teknikte çatışma ortaya konduktan sonra, bu çatışmanın yok edilmesi için enjeksiyon(lar) önerilir. Şekil 3'deki buharlaşan bulut, yazılım geliştirme prosesinde karşılaşılabilecek çatışmalardan bir tanesini ve çözüm için olası enjeksiyonları göstermektedir. Burada yaşanan çatışma, istenen yazılımın geliştirilmesi sürecinde kullanıcıların değişmesi ya da değişmemesidir. Bu çatışmanın ortadan kaldırılmasına yönelik enjeksiyonlar, etkin bir iletişim sisteminin oluşturulması ve kullanıcıların iş başında eğitimlerine gereken önemin verilmesidir. Bu enjeksiyonlar uygulanarak kök nedenin ortadan kalkması sağlanabilir.

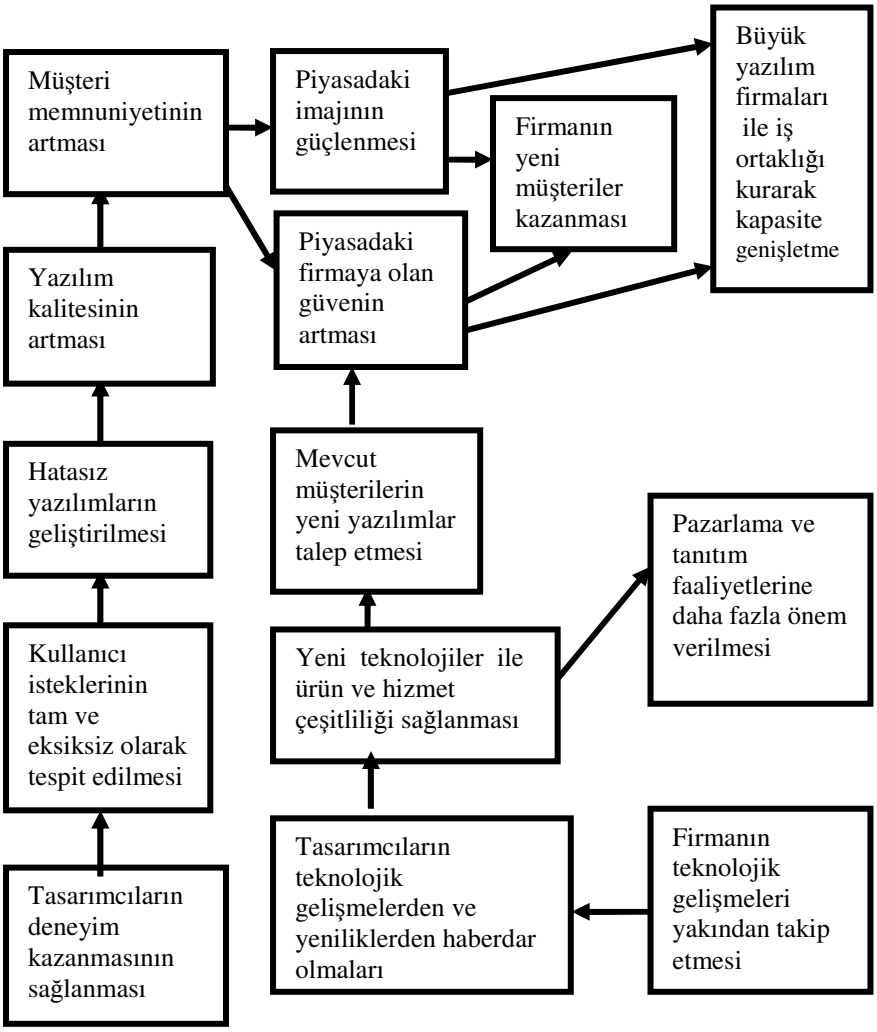
Buharlaşan bulut ile, daha önce de bahsedildiği gibi, istenmeyen sonucu ortadan kaldırmak için çözüm önerileri, temel ve ön gereksinimler tanımlanır. Çözümler arasındaki çatışmalar belirlenerek bu çatışmaların giderilmesi için enjeksiyon yapılır (Özer, 2001). Bu araç, tek bir problemin ayrı olarak ele alınmasını, karşılaşılan çatışmaların ve varsayımların belirlenmesini ve çözüm amacıyla incelenmesini içerir (Köksal ve Karşılıklı, 2000).

Sonraki aşama Gelecek Gerçeklik Ağacının (GGA) uygulanmasıdır. Bu aşamanın amacı, gerçekleştirilmek istenen bir durumun beklenen en iyi sonuçlara (İstenen Etki - Desirable Effect - DE) ulaştıracağının doğrulanmasıdır. Şekil 4'de bu çalışmaya ait gelecek gerçeklik ağacı görülmektedir. GGA'nda, Şekil 3'deki buharlaşan bulutta bahsedilen tasarımcı ve kullanıcı arasındaki işbirliği ile gereksinimlerin tam ve doğru belirlenmesi sonucunda hatasız yazılımlar geliştirilebilir, bu da yazılım kalitesini artırır, ürün ve hizmet çeşitliliği sağlanır, bunların sonucunda müşteri memnuniyeti artar. Böylece firmanın uzun vadede piyasadaki imajı güçlenebilir ve güvenilirliği artabilir, piyasadaki güvenilirlik artışıyla birlikte yeni müşteriler kazanılabilir, firma performansı ve kapasitesi artırabilir.

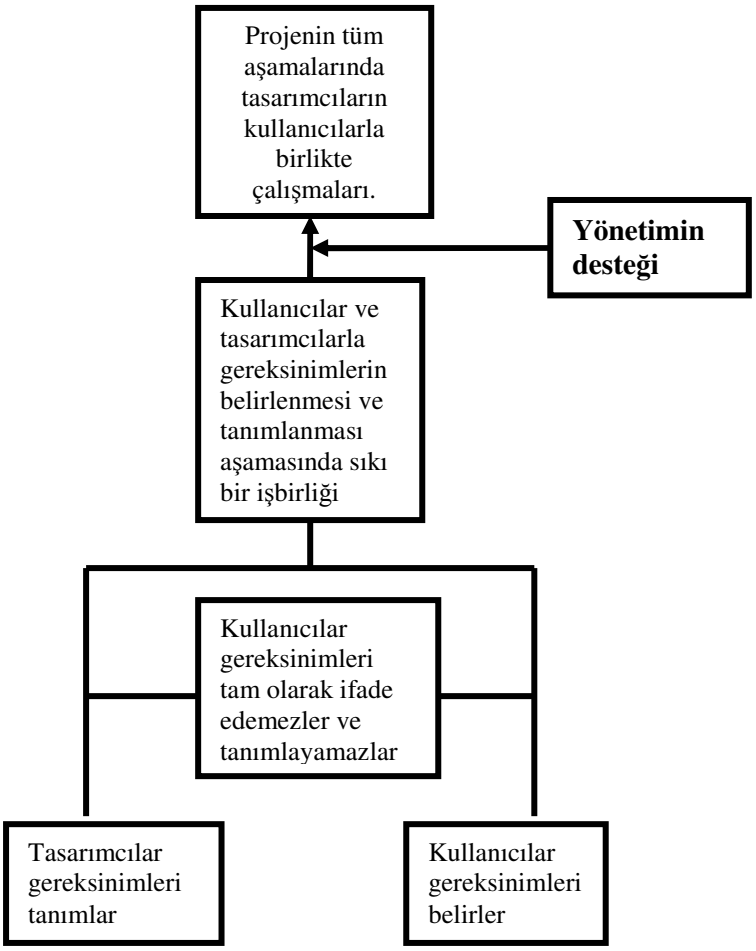
Bir sonraki aşamada değişimleri hayata geçirebilmek ve çözüm kümeleri oluşturabilmek için ön gereksinim ve geçiş ağaçları kullanılır. Ön gereksinim ağacıyla, çözüme ulaşmamızı engelleyen durum tanımlanır. Bu çalışmada çözüme ulaşmamızı engelleyen durum, Şekil 5'deki ön gereksinim ağacında görüldüğü gibi kullanıcıların gereksinimlerini tam ve doğru olarak tanımlayamamalarıdır. Bu engeli ortadan kaldırmak için kullanıcılar ve tasarımcılar gereksinimleri belirler, her iki grup arasında sıkı bir işbirliği kurularak ve üst yönetimin de desteğiyle projenin tüm aşamalarında birlikte çalışarak gereksinimler doğru bir şekilde tanımlanabilir.



Şekil 3. Buharlaşan Bulut

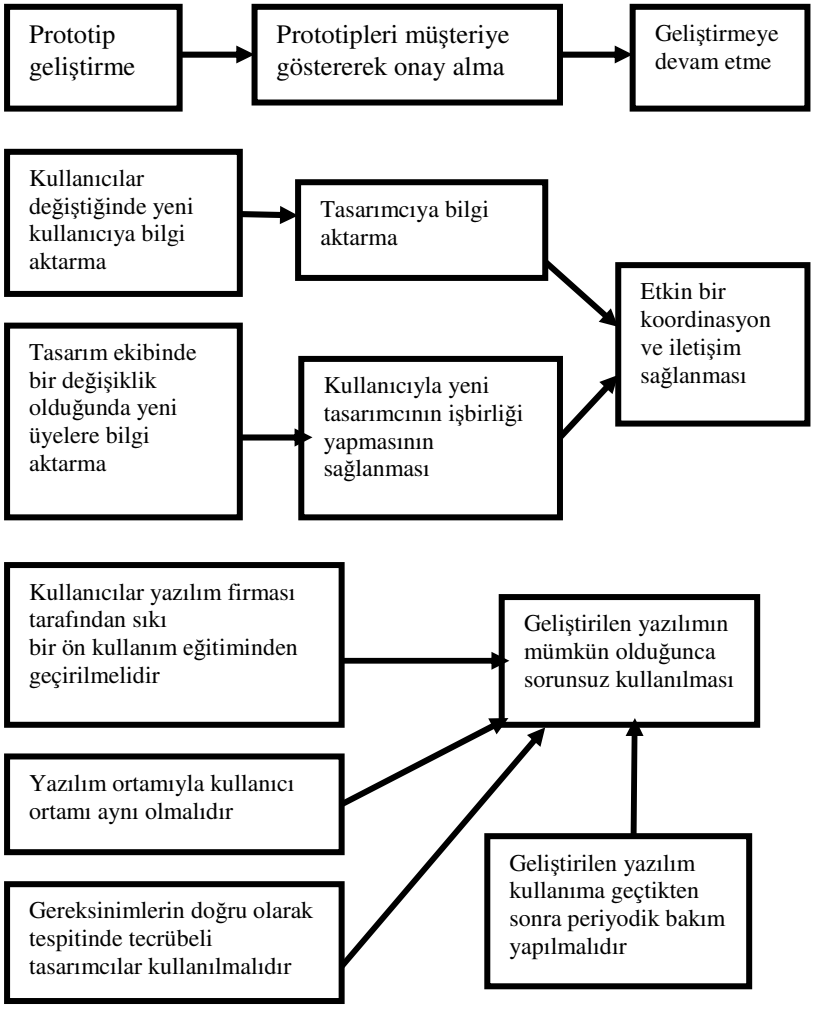


Şekil 4. Gelecek Gerçeklik Ağacı



Şekil 5. Ön Gereksinimler Ağacı

Geçiş ağacı ise, detaylı bir şekilde planlanan ana amaçlara ulaşmak ve bunları uygulamaya geçirmek için gerekli faaliyetlerin tanımlanmasında kullanılır. Bu şekilde aktif ve etkin bir iletişim ve işbirliği sağlanabilir ve geliştirilen yazılım problemsiz kullanılabilir.



Şekil 6. Geçiş Ağacı

Şekil 6'da verilmiş olan geçiş ağacında görüldüğü üzere, yazılımın kullanıcı ihtiyaçlarını tam olarak karşılayabilmesi için tecrübeli tasarımcılar kullanılmalı, kullanıcılar değiştiğinde yeni kullanıcıya bilgi aktarımı yapılmalı, tasarımcıya bilgi aktarılarak etkin bir koordinasyon sağlanmalıdır. Prototip geliştirip müşteri onayı

alınarak geliştirmeye devam edilmelidir. Tasarımcı ve kullanıcı ortamlarının aynı olması, kullanıcıların iyi ön kullanım eğitiminden geçirilmeleri ve geliştirilen yazılıma, kullanıma geçildikten sonra periyodik bakım uygulanması ile sorunsuz bir kullanım süreci mümkün olabilir.

5. SONUÇLAR

Bir çok yazılım şirketi günümüzde yoğun bir rekabetle karşı karşıya kalmaktadırlar. Bu firmaların ürünleriyle ilgili olarak karşılaştıkları en önemli problemlerden biri, yazılım geliştirildikten ve müşteri firmada kullanılmaya başlandıktan sonra, geliştirilen yazılımın müşteri isteklerini tam olarak karşılayamadığının tespit edilmesidir. Bu sorun genellikle son aşamada, geliştirilen yazılım kullanıcıya teslim edildikten sonra ortaya çıkmaktadır. Bu durum hem yazılımı geliştiren firmada harcanan emeğin ve zamanın hem de müşteri firmanın parasının ve zamanının boşa gitmesine neden olmaktadır. Kısıtlar teorisi felsefesi, firmanın başarısını engelleyen durumları ortaya koyan, firmaların faaliyetlerinde en zayıf noktayı belirleyebilen, kendi sistematiğiyle firmadaki bu zayıf noktayı güçlendirmeyi sağlayarak firmanın başarısını artıran ve firmanın başarısını engelleyen durumları uyararak kendi rekabet gücünde şıçrama yapmasını sağlayan yöntemleri içerir.

Kısıtlar teorisinin düşünce süreçleri - ki bunlar mevcut gerçeklik ağacı, buharlaşan bulutlar, gelecek gerçeklik ağacı, ön gereksinimler ağacı ve geçiş ağacı- yazılım firmalarında yazılım geliştirme prosesinin daha iyi ve dikkatli yönetilmesini sağlayarak ortaya çıkacak problemleri önceden tespit etmeye imkan sağlayacak bir tekniktir.

Bu çalışmada, yazılım geliştirme prosesinde geliştirilen ürünlerin en önemli başarısızlık nedenlerinden biri olan müşteri gereksinimlerinin tam olarak anlaşılamamasının altında yatan temel sebepler tespit edilmiş ve kısıtlar teorisinin düşünce süreçleri uygulanarak bu kısıtı ortadan kaldıracak şekilde çeşitli öneriler getirilmiştir. Bu önerileri düşünce süreçlerinin son aşaması olan geçiş ağacında görmek mümkündür. Ürün geliştirme sürecinde en önemli kısıtların politik kısıtlardan kaynaklandığı tespit edilmiştir.

Özellikle politik kısıtlardan kaynaklanan problemlerin giderilmesinde düşünce süreçleri, çok etkin olarak kullanılabilecek yöntemlerdir. Bu çalışma ile, kısıtlar teorisinin uygulama alanının genişletilmesine katkıda bulunulduğu ve gelecek çalışmalara yol gösterici rolü üstlenildiği düşünülmektedir.

KAYNAKÇA

Arifoğlu, A., Doğru, A., (2001), Yazılım Mühendisliği, Ankara, SAS Bilişim Yayınları.

Atwater, B. and Gagne M., (1997) "The Theory Of Constraints Versus Contribution Analysis for Product Mix Decisions", Journal of Cost Management, (11/1), 6-15.

Balaban,E.,www.isletme.istanbul.edu.tr/ogrelem/balaban/ProjeYönetimi-horosanli.htm.[16.12.2004].

Boutquin, P., Poremsky, D. and Slovak, K., (2000), Beginning Visual Basic 6 Application Development, UK, Wrox Press.

Büyüközkan, G. And Feyzioğlu, O.,(2003) "A Fuzzy-logic-based Decision Making Approach for New Product Development", International Journal of Production Economics, (90/1), 27-45.

Choe, K. and Herman, S. (2004), "Using Theory of Constraints Tools to Manage Organizational Change : A Case Study of Euripa Labs", International Journal of Management & Organisational Behaviour, 8 (6), 540-558 ISSN 1440-5377

Davies, J., Mabin, V.J. and Balderstone, S.J., (2005), The Theory of Constraints: A Methodology Apart?—A Comparison with Selected OR/ Msmethodologies, Omega, Article in Press, www.sciencedirect.com, 1-19

Gardiner, S., Blackstone J. and Gardiner L., (1994) "The Evolution of The Theory of Constraints", Industrial Management, (May/June),13-16.

Goldratt, E. and Cox, J.,(1992) The Goal: A Process of Ongoing Improvement, Great Barrington, MA, North River Press.

Goldratt, E., (1998) The Goal / Amaç, İstanbul, Profilo Dağıtım A.Ş.

Igel, B. and Islam, N., (2001), "Strategies for Service and Market Development of Entrepreneurial Software Designing Firms, Technovation, (21/3),157-166.

Kendall, G., I., (1998) Securing the Future Strategies for Exponential Growth Using the Theory of Constraints, Florida, CRC Press.

Köksal, G. ve Karşılıklı, U.K., (2000) "Kısıtlar Teorisi ve Toplam Kalite Yönetimi Yoluyla Etkin Performans Yönetimi", 9. Ulusal Kalite Kongresi, 21-22 Kasım.

Mcmullen, T., B., (1998), An Introduction to The Theory of Constraints Management System, Florida, CRC Press.

Nambisan, S., (2002) "Software Firm Evolution and Innovation-orientation", Journal of Engineering and Technology Management, (19/2)141-165.

Özer, G., (2001), "Dünya Sınıfı Bir Sistem ve Yönetim Yaklaşımı: Kısıtlar Teorisi ve Katkı Muhasebesi", Verimlilik Dergisi, (2), 7-29.

Pfeifer, T. and Tillmann, M., (2003), "Innovative Process Chain Optimization – Utilizing the Tools of TRIZ and TOC for Manufacturing", First presented at the European TRIZ Association TRIZ Futures Conference, Aachen, Germany, November 2003.

Rahman, S., (1998) ,"Theory of Constraints: A Review of The Philosophy and Its Applications", International Journal of Operations and Production Management," (Vol.18), 336-355.

Rand, G.K., (2000), Critical Chain: The Theory of Constraints Applied to Project Management, International Journal of Project Management, (18/3),173-187,

Rizzo,T., (2001), Theory of Constraint, http://www.rogo.com/cac/rizzo11.html.

Rudy, Y., (1996), Theory of Constraints, http://www.chief.co.il/toc/toc.html

Ruhl, J. M., (1996), "An Introduction to The Theory of Constraints", Journal of Cost Management, (10) 2, Summer, 43-48.

Schucavage, D., (1995), Apıcs Toc Faq, http://www.rogo.com/cac/FAQ.html .

Schwalbe, K., (2000), Information Technology Project Management, Course Technology Thomson Learning, Canada.

Siha, S. (1999), "A Classified Model for Applying the Theory of Constraints to Service Organizations", Managing Service Quality, (9/4), 255-264.

Spencer, M.S. and Cox.J.F., (1995) "Master Production Scheduling Development in a Theory of Constraint Environment", Production and Inventory Management Journal, First Quarter, 8-15.

Stein, R, E., (1997) The Theory of Constraints, Second Edition, Marcell Decker Inc., USA.

Verma, R., (1997), "Management Science, Theory of Constraints / Optimized Production Technology and Local Optimization", Omega, International Journal of Management Science, (25/2).

Womack, D. and Flowers S., (1999), "Improving System Performance : A Case In The Aplication Of The Theory of Constraints", Journal of Healthcare Management, (September/October), 397-405.

Ulrish, K.T., (2001), "Introduction to the Special Issue on Design and Development", Management Science, 47 (1), v-vi.