Etkin Yeniden Kullanım: Yazılım Ürün Hatlarında Değişkenliğin Modellenmesi

Baris Can Kasikci[1], Semih Bilgen[2]
[1] Kurumsal Teknoloji, Siemens A.Ş, İstanbul,
[2] Orta Doğu Teknik Üniversitesi Elektrik Elektronik Mühendisliği Bölümü
[1] e-posta: baris.kasikci@siemens.com
[2] e-posta: semih-bilgen@metu.edu.tr

Özetçe

Bu bildiri, yazılımın yeniden kullanımında ön alıcı *(proactive)* bir yaklaşım olan yazılım ürün hatlarında değişikliklerin modellenmesine, bu kapsamda izlenebilirliğin artırılarak yeniden kullanım olanaklarının iyileştirilmesine yönelik gerçekleştirilen çalışmaları ele almaktadır. Bu kapsamda, bilinen yazılım ürün hattı yaklaşımlarına eklenmek üzere "kaygı" kavramı ortaya atılmaktadır. Çalışmanın bir başka özgün katkısı da ürün hatlarında değişkenliğin modellenmesi için bağlamdan bağımsız gramer (BBG) kullanımı önerisidir. Ürün hattı varlıkları bünyesinde bulunan değişkenliğin izlenebilirliğini sağlayan "kaygı" kavramının sağladığı yararlar birtakım değerlendirme ölçütleri uyarınca sınanmıştır. Bununla birlikte "kaygı"ların sağlayabileceği yararlar ve uygulanmaları sonucunda elde edilen verilere de değinilecektir.

1. Giriş

Yeniden Kullanım (YK) konusu, yazılım mühendisliği araştırmalarında yoğun olarak ele alınmaktadır. YK'nın temel amacı, yeniden kullanılacak varlıkların ortak özelliklere sahip sistemlerde birçok kez kullanabilmesini sağlamaktır. Yeniden kullanılan varlıkların boyut ve kapsamları, YK'nın yararlarını doğrudan etkilemektedir. Bir tamsayı değişkenin yeniden kullanımı pek bir yarar getirmezken, gereksinimlerin, tasarımların, kaynak kodun, test varlıklarının ve sistem özelliklerinin hepsinin yeniden kullanımı, YK'nın anlamını ve etkisini artırmaktadır. Bu düzeyde bir YK, yazılım ürün hatları yaklaşımları ile mümkündür. Yazılım ürün hatları, en genel anlamıyla, ortak özellikleri yoğun olan bir ürün ailesindeki ortak ve değişken varlıkları açık bir biçimde ortaya koyan; bu varlıklarının kullanımı ile ürün ailesine ait yeni sistemlerin geliştirilmesini sağlayan yapılardır.

Bu tür yaklaşımların verimli olabilmeleri için öncelikli koşul ürün ailesinin ait olduğu alanın kapsamının belirlenmesi ve ortak özelliklerin kullanılarak alandaki sistemlerin tümünde bulunması beklenen çatının oluşturulmasıdır. Değişkenliğin modellenmesi ise ürün hattı geliştirmede bir sonraki aşamadır. En önemli üretkenlik kazanımı, ürün hattı varlıklarının değişkenliği kullanılarak yeni sistemlerin geliştirilmesi ile elde edilir. Bilinen ürün hattı yaklaşımları, ürün hattındaki sistemler arasındaki değişkenliği özellik modelleri kullanarak tanımlamaktadır.

Ancak [1]'de de belirtildiği gibi, değişkenliğin özellik modelleri ile izlenmesi kısıtlamalar ve yetersizlikler getirmektedir. Bu durumu ortadan kaldırmak için [2]'de ortaya atılan "karar" kavramı kullanılarak değişkenlikler ürün hattını oluşturan farklı soyutlama seviyelerinde modellenmiş ve izlenebilirlik sağlanmıştır. Bu çalışmamızda ise, "karar" kavramı genişletilerek "kaygı" kavramı önerilmekte, "kaygıların" biçimsel modellemesi bağlamdan bağımsız gramer kullanımı ile gerçekleştirilmektedir.

Önerilen yaklaşımın bilinen ürün hattı yaklaşımlarına eklenmesi ile elde edilecek yararları tartmak üzere birtakım değerlendirme ölçütleri ortaya atılmaktadır. Bu değerlendirme ölçütleri ışığında "kaygı" kavramının var olan bir ürün hattı yaklaşımına uygulanması gösterilmekte, elde edilen sonuçlar tartışılmaktadır.

2. Yeniden kullanım, yazılım ürün hatları ve değişkenlik modelleme

Bu bölümde yeniden kullanım, ürün hatları ve değişkenlik modelleme konularıyla ilgili literatür özetlenerek değerlendirilmektedir.

2.1. Yazılımın yeniden kullanılabilirliği

Yazılımın yeniden kullanabilirliği, geçmişte gerçekleştirilmiş olan yazılım geliştirme etkinliklerinde elde edilen varlıkların veya bilginin yeni sistemler geliştirmede kullanılması olarak tanımlanmaktadır [3]. YK' ya yönelik ilk önemli yaklaşım, yazılımın sanayileşmesi çerçevesinde bileşen fikrinin ortaya atılmasına dayanmaktadır [4]. Buna göre verimli YK'nın sağlanması için bileşenlerin doğruluk, dayanıklılık ve başarım gibi ölçütlere göre sınıflandırılıp kullanıcılarına sunulması gerekir.

Geleneksel YK iki ana sınıfta toplanabilir. Bunlar kaynak kod ve kavramsal YK'dır. Kaynak kodun YK'yı gerçek anlamda saf kodun yeniden kullanımı veya kütüphane, çatı gibi daha düzenli yapılar aracılığı ile gerçekleştirilebilmektedir. Sadece kodun yeniden kullanımının, çalışan yetkinliği ve bilgi seviyesine bağımlılığı artırabilecek olması ve dolayısıyla kaynakların verimsiz kullanımına yol açması beklenebilir.

Daha soyut YK ise kavramsal düzeyde gerçekleşmektedir. Bu YK'a ilk örnek, yazılım tasarım kalıpları ile getirilmiştir [5]. Yazılım tasarım kalıpları tekrar karşılaşılan yazılım sorunlarıyla ilgili elde bulunan sorun, koşullar, çözüm ve sonuç topluluğunu kayıt altına almaktadır.

Yazılım tasarım kalıplarının soyutluk anlamında bir üst düzeyinde ise mimari tasarım kalıpları bulunmaktadır. Bu

kalıplar geliştirilen uygulamanın temel yapısını tanımlamaktadır ve birden çok mimari görünümü kapsamaktadırlar. Bu nedenle geliştirilen sistemler açısından önemli ve geliştirmenin ilk safhalarında ele alınması gereken bir konumdadırlar.

2.2. Yazılım ürün hatları

Yazılım ürün hatları geliştirmenin temel amacı değişken müşteri isteklerini karşılayabilmektir. Özelleşmiş ürünlerin daha düşük maliyetle ve daha hızlı üretilmesi yazılım ürün hatlarının ana hedefidir. Bunu sağlamak ise YK kapsamını genişletmekten ve bunu sağlayan süreçleri sistematik bir hale dönüştürmekten geçmektedir. Bununla birlikte bu süreçleri destekleyen araçlar ve ortam da yazılım ürün hattı oluşturulması kapsamındadır.

2.2.1. Sistem özellikleri ve özellik modelleri

Özellik *(feature)* bir şeyin benzerlerinden veya başka şeylerden ayrılmasını sağlayan nitelik olarak tanımlanmaktadır[1]. Çeşitli sistemlerin benzer ve değişken özellikleri, bu sistemleri ortak özelliklerin olduğu gruplara toplamakta veya birbirlerinden ayırt etmede kullanılabilir.

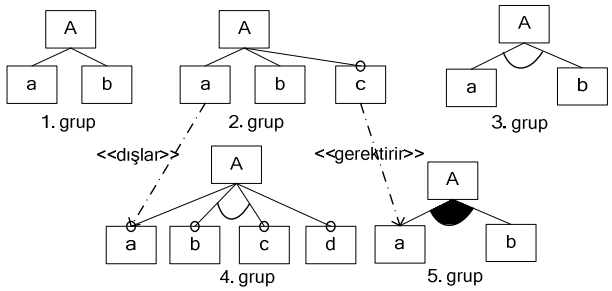
Yazılım ürün hatlarına ait sistemlerin yoğun ortak özelliklere sahip olması beklenmektedir. Bununla birlikte bu sistemleri birbirlerinden ayıran değişkenliğin de ürün hattı yaklaşımı tarafından ifade edilmesi gerekmektedir. Yazılım ürün hatlarında ortaklık ve değişkenliği ele alırken alışılagelmiş yazılım geliştirme yaklaşımlarında rastlanmayan, sistemleri özellikleri bakımından sınıflandıran yöntemler benimsenmektedir. Bunu ilk gerçekleştiren yaklaşım FODA[2]'dır[6]. Gene FODA, özelliklerin ve bu özellikler arasındaki ilişkilerin tanımlandığı modeller ile ürün hattındaki ortaklığı ve değişkenliği ifade etme yoluna gitmiştir. Özgün FODA özellik yaklaşımın ardından, çeşitli farklı yaklaşımlar ortaya atılmıştır. [7]'de özellik modelleme yaklaşımlarına ilişkin kapsamlı bir araştırma yayımlanmıştır.

Bu çalışmada da değişkenlik bilgisi taşıyan yapılar arasındaki ilişki, özellik modellerine benzer şekilde gösterilmiştir. Esinlenilen özellik modeli gösterimleri Şekil 2-1'de özetlenmiştir.

Şekil 2-1'de, büyük harf ile gösterilen yapılar değişkenlik veya ortaklıkların ait olduğu kavramsal topluluğu, bir diğer ifade ile değişkenlik veya ortaklık noktalarını ifade etmektedir. Küçük harfler ise değişkenlik ve ortaklıkların aldıkları gerçek değerler, yani özelliklerin kendileridir.

[1] www.tdk.gov.tr, son olarak 21.04.2009 tarihine kadar geçerlidir.

[2] FODA, Feature Oriented Domain Analysis, yani Özellik Yönelimli Alan Analizi anlamına gelmektedir.

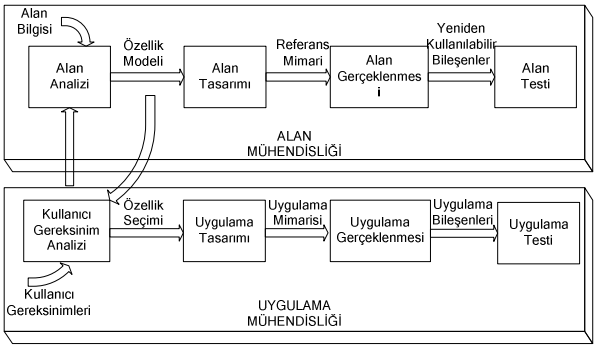


Şekil 2-1 Özellik modeli ve Özellik İlişkileri

Şekil 2-1'deki gruplara kısaca değinilecek olursa; 1. grup, a ve b olmak üzere iki ortak özelliği; 2.grup, gene a ve b ortak özelliklerini ve isteğe bağlı c özelliğini; 3. grup, a ve b alternatif özelliklerini; 4. grup, tümü isteğe bağlı olan ve b ve c'nin alternatif olduğu dört adet özelliği; son olarak da 5. grup, "a veya b" ilişkisini –mantıksal "veya"- göstermektedir. Bunun yanı sıra özellikler arasında "dışlar" ve "gerektirir" ilişkileri de bulunabilmektedir.

2.2.2. Yazılım ürün hattı süreçleri

Ürün hatlarına dayalı yazılım geliştirme süreç yapısı farklı yaklaşımlar incelendiğinde benzerlikler göstermektedir. Şekil 2-2'de görülen yapı da [8,9]'da ele alınan ürün hattı süreç yapılarının bir karması olarak algılanabilir.



Şekil 2-2 Yazılım Ürün Hattı Yaklaşımlarında Genel Süreç Yapısı

Buna göre, ürün hattı süreci, alan ve uygulama mühendisliği olmak üzere iki alt süreçten oluşmaktadır. Alan mühendisliği sırasında öncelikle alan bilgisinin kullanımı ile alan analizi gerçekleştirilir. Bu aşamada alan kapsamı belirlenir, alandaki ortaklıklar ve değişkenlikler ortaya konulur. Alan tasarımı çıktısı olarak alana ait sistemlerin uyum sağlaması beklenen yapıları ve kuralları tanımlayan bir referans mimari oluşturulur. Alan gerçeklenmesi, alana özel yeniden kullanılabilir ayrıntılı bileşenlerin tasarlanması ve gerçeklenmesi işlemidir. En son aşamada ürün hattı test altyapısı oluşturulur.

Uygulama mühendisliği aşamasında da benzer aşamalardan, kullanıcı gereksinimleri göz önüne alınarak geçilir ve alan mühendisliği alt sürecindeki değişkenlik çözümlenerek somut ürün elde edilir. Burada önemli olan bu etkinliklerin ardışık olmayışıdır. Alan mühendisliği çalışmalarının uygulama

mühendisliği çalışmalarından gelen geri beslemeye göre uyarlamalara ve geliştirmelere uğraması beklenmektedir.

Tarihsel gelişime bakıldığında, ürün hattı yaklaşımlarından önce alan mühendisliği yaklaşımlarının ortaya atıldığı görülmektedir. Bununla birlikte çeşitli ürün hattı yaklaşımları değerlendirildiğinde ağırlığın alan mühendisliğine verildiği görülmektedir.

2.3. Ürün hattı yaklaşımları örnekleri

2.3.1. FODA [6]

FODA, özellik kavramı kullanarak ürün hattındaki değişkenlik ve ortaklıkları modelleyen ilk yaklaşımdır. Bununla beraber FODA esasen bir alan mühendisliği yöntemidir. Bağlam analizi, alan modelleme ve mimari modelleme FODA'da izlenen üç temel etkinliktir. Bağlam analizi aşamasında alan kapsamını belirlemek ve alanın diğer alanlarla ilişkisini göstermek için bağlam modelleri ve yapı diyagramları kullanılır. Alan analizi sırasında öğeler arası ilişki modeli, özellik modelleri, işlevsel modeller oluşturulur ve alan sözlüğü geliştirilir. FODA özellik modelleri "veya ilişkisi" dışında Şekil 2-1'de gösterilen ilişkilere benzemektedir. Bununla birlikte özellik modellerindeki özellikler dört katmana ayrılmıştır. Bunlar yetenekler, çalıştırma ortamı, alan teknolojisi ve uygulama teknikleri katmanlarıdır. Mimari modelleme aşamasında ise görev etkileşimleri ve modüller arası ilişkiler ortaya konmalıdır.

2.3.2. *FORM*[1] [9]

FODA yöntemini geliştirmek ve uygulama mühendisliğini de sürece katmak amacıyla FORM geliştirilmiştir. FORM'dan önce geliştirilen yöntemler, özellik analizini daha çok kullanıcı gereksinimleri düzeyinde ele almaktadırlar. FORM'da bu analizi kullanarak modül geliştirmeye odaklanılmaktadır. Bu bakımdan FODA'dan bir adım öteye gidip, mimari modellemeden sonra yeniden kullanılabilir bileşenleri modellemeye değinilmiştir. Mimari ve bileşen oluşturma sırasında çeşitli mühendislik ilkeleri tanımlanmıştır. Bu ilkelerin temelinde, modüllerin, özelliklerin kullanımı ile parametrize edilmesi ve özelliklerin sahip olduğu katmanlı yapıya uygun geliştirilmeleri gerektiği bilgisi yatmaktadır. Bunun yanı sıra uygulama mühendisliği alt süreci de yönteme dâhil edilmiştir.

2.3.3. PLUS[2]

PLUS, güncel ve geniş kapsamlı bir ürün hattı yaklaşımıdır. Süreci Şekil 2-2'de gösterilen süreç modeline oldukça benzemektedir. Gereksinim modelleme, analiz modelleme, tasarım modelleme, bileşen geliştirme ve test aşamalarını hem alan hem de uygulama mühendisliği alt süreçlerinde işletmektedir. Getirdiği temel yenilik, alan gereksinimleri, özellik modelleri, bileşenler arasındaki ilişkiler gibi ürün hattı kavramlarını, kullanım durumu diyagramları, sınıf diyagramları, işbirliği diyagramları gibi UML gösterimleriyle ifade etmesidir.

[1] FORM, Feature Oriented Reuse Method, yani Özellik Yönelimli Yeniden Kullanım Yöntemi anlamına gelmektedir.
[2] PLUS, Product Line UML based Software Engineering, yani UML tabanlı Ürün Hattı Mühendisliği anlamına gelmektedir.

2.4. Değişkenliğin modellenmesi ve izlenebilirliği

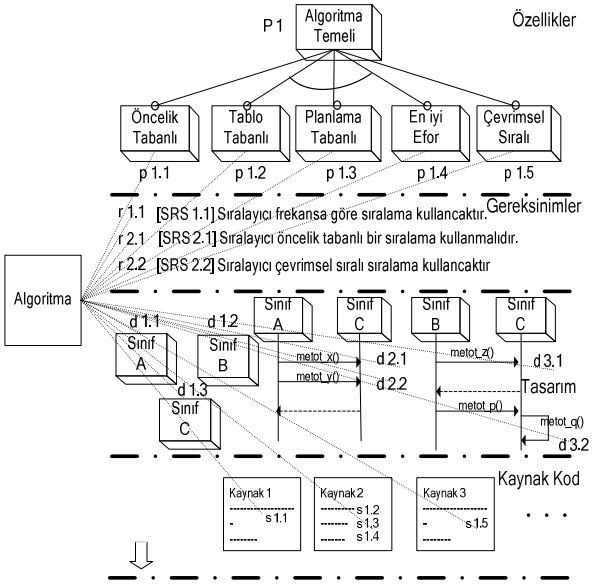
FODA alan analizinin temeline özellikleri koymaktadır. Bununla birlikte alan içindeki değişkenliğin modellenmesini ve izlenebilirliğini sağlamak için ürün hattını oluşturan varlıklar ve özelliklerin ilişkisini kurmaktadır. FORM da benzer bir yaklaşıma gitmiştir, FODA'dan öteye giderek ürün hattı bileşenlerinin parametrizasyonunu da özellikler aracılığıyla gerçekleştirmektedir. PLUS'da ise kullanım durumları ile özellikler arasındaki izlenebilirliğin yanı sıra özellikler ile özelliği gerçekleyen yazılım nesneleri arasındaki bağlantılar da kurulmuştur.

2.3 numaralı bölümde değinilen ürün hattı yaklaşımlarının ortak yönü değişkenliğin modellemesinde ve izlenebilirliğin sağlanmasında özellikleri kullanmalarıdır. Ancak [8]'de belirtildiği gibi, ürün hattındaki değişkenliğin yönetilmesi, ürün yöneticilerinin, yazılım mimarlarının, geliştiricilerin ve test sorumlularının girdilerine bağlıdır. Buna ek olarak, [1]'de de özelliklerin değişkenlik modellemede ölçeklenebilirlik, tutarlılık ve en önemlisi izlenebilirlik bakımından eksikleri olduğu belirtilmiştir. [11]'da ise özellik modellerinde değişkenlikle birlikte ortaklığın da bulunmasının değişkenlik modellenmesini karışıklaştırdığı, buna ek olarak, farklı varlıklar arasındaki izlenebilirliği sadece özellikler kullanarak kurmanın zorluğu vurgulanmıştır.

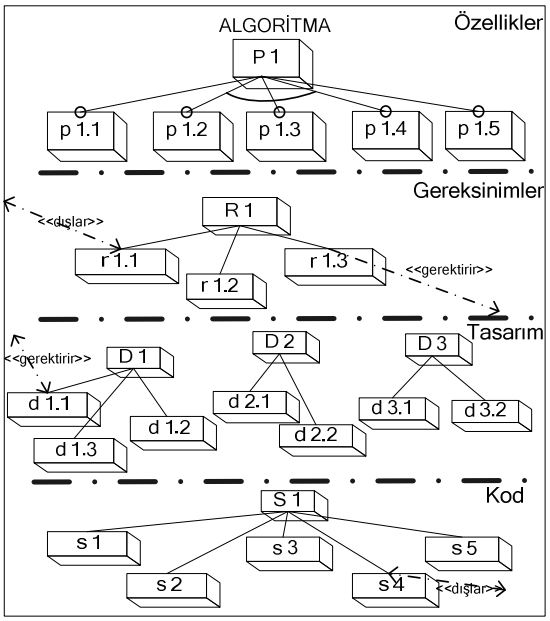
Bu nedenlerle değişkenliğin salt özelliklerle modellenmesi yerine daha geniş kapsamda ele alınması gerekmektedir. Bunu sağlamak için [11]'da dikey değişkenlik modelleme tekniği, [2]'de ise "karar" kavramı ortaya atılmıştır. Bu çalışmada ortaya konan "kaygı" kavramının temelinde yatmakta olduğundan ve değişkenliğin izlenebilirliğine daha çok değindiğinden, aşağıda kararlar kısaca incelenecektir.

2.4.1. *Kararlar* [2]

Kararlar, özellik modellerindeki değişkenlik noktalarının daha kapsamlı halleridir. Kararlar, değişkenlik ifade etmekle birlikte birden fazla değişkenlik noktasını ele almaktadırlar. Bunu da [1]' de anlatıldığı gibi değişkenliği ürün hattına ait olan farklı düzeydeki varlıkları aynı karar altına toplayarak gerçekleştirmektedirler. Adından da anlaşılabileceği gibi bir kararın uygulama mühendisliği sırasında değişkenlikten kurtulup son değerini alması gerekmektedir. Kararı ifade eden genelde bir evet-hayır sorusudur, ancak daha karmaşık durumlarda çeşitli koşulların bulunduğu bir anlatım tercih edilmektedir. Şekil 2-3'de "algoritma" adlı kararın yeniden kullanılabilir bir görev sıralayıcı bileşeni bünyesinde barındırdığı ve izlenebilirliğini sağladığı görülmektedir.



Şekil 2-3 Algoritma Kararı ve bağlantılı olduğu ürün hattı varlıkları



Şekil 3-1 Algoritma Kaygısı

3. Kaygılar ve ürün hatlarındaki değişkenliğin modellenmesi

3.1. Kaygılar[2]

Kararlar ürün hattı değişkenliğini yeni bir boyutta ele alıp farklı katmanlardaki ürün hattı varlıklarına bağlantı kurmayı önermiştir [1,2]. Ancak kararlar farklı katmanlardaki değişkenliğin her katmanda ayrı modellenmesine değinmemiştir. Bunun sağlanması, değişkenlik bilgisinin farklı katmanlara yayılmış hali "dikey", münferit bir katmandaki doğasının ise "yatay" olarak modellenmesi ile gerçekleştirilebilir.

Kaygı, bir ürün hattındaki değişkenliği dikey ve yatay hiyerarşilerde modelleyen ve varlıklar arasındaki gerekli ilişkileri kuran değişkenlikler topluluğudur. Kaygı modeli ise, ürün hattındaki değişkenliği kaygılar aracılığı ile ifade eden modeldir. Kaygıların kullanımı, mevcut bir ürün hattı yaklaşımında değişkenliğin modellenmesi aşamasında kaygı modelinin geliştirilmesi ile mümkündür.

Kaygılar bu çalışma kapsamında, dört farklı katmandaki ürün hattı varlıklarını ele almaktadır. Bunlar ürün özellikleri, gereksinimler, yazılım tasarımı ve kaynak koddur. Bu katmanlar bütünleşim sağlanan ürün hattı yaklaşımına bağlı olarak çoğaltılabilir veya azaltılabilir (örneğin test varlıkları bu katmanlara eklenebilir.) Şekil 2-3'de gösterilen ürün hattındaki değişkenliklerin bir kaygı ile nasıl ifade edildiği Şekil 3-1 'de görülmektedir.

Değişkenlik bilgisi bir kaygı içerisinde simgesel ismi ile gösterilmektedir. Değişkenlik noktaları 3.2 numaralı bölümde açıklanacağı gibi büyük, değişkenlikler ise küçük harflerle gösterilmektedir. Değişkenlik noktalarının ve değişkenliklerin hangi ürün hattı varlıklarına denk geldiklerini görmek için Şekil 2-3'deki varlıklarla ilişkilendirilen simgelere başvurulmalıdır. Simgelendirmede sırasıyla P,R,D,S harfleri, özellikler, gereksinimler, tasarım ve kaynak kod için kullanılmaktadır. Değişkenlik noktaları bu harflerin sonuna sayılar getirilerek ifade edilir (D1, D2...). bu harflerin başlarına sayı getirilemez çünkü bu notasyon 3.2 bölümünde anlatıldığı üzere farklı bir anlama gelmektedir. Değişkenlikler de değişkenlik noktalarındaki sayı kademelerinin bir alt düzeyinde numaralandırılırlar. Buna göre D1 değişkenlik noktasının altındaki değişkenlikler d1.1, d1.2... şeklinde adlandırılmalıdırlar. İsimlendirmede kullanılan numaralandırma çok kritik olmamakla beraber, model çapında değişkenlik noktalarının birbirinden sistematik biçimde ayırt edilebilmeleri gereklidir.

Değişkenlik noktaları ve değişkenliklerin gösteriminde kullanılan notasyon Şekil 2-1'deki gibidir. "İçerir" ve "dışlar" bağlantıları kaygı bünyesindeki değişkenlik bilgisi taşıyan varlıklarla diğer bir kaygıdaki değişkenlik elemanları arasında kurulmalıdır. Bir kaygı içerisinde bu ilişkilerin gösterilmesi gerekiyorsa bunun için içerir ve dışlar kullanılmaması önerilir. Bunun nedeni kaygı içerisinde "içerir" ilişkisinin Şekil 2-1'deki değişkenlik ilişkileri ile kolayca gösterilebilecek olması ve "dışlar" ilişkisinin aynı kaygı altında toplanan değişkenlik elemanları arasında bulunmasının beklenmemesidir.

3.2. Kaygılar ve bağlamdan bağımsız diller

Dikey değişkenlik modellemede ürün hattı varlıkları arasındaki izlenebilirliğin yazılım geliştirme araçları tarafından sağlanması gerektiği söylenmiştir[11]. İzlenebilirliğin kurulmasının yararlı olması beklenmekle birlikte, bunu

sağlayacak araçların, değişkenliğin biçimsel olarak ifade edilmesine ihtiyaç duymaları beklenir. Bu sayede değişkenliği modelleyen araçlar gerçekleme ve doğrulamayı matematiksel bir temele oturtabileceklerdir. Burada akla gelen soru şudur: Değişkenlikler ve değişkenlik noktaları arasındaki ilişkiler nasıl modellenmelidir ki, bu bilgi sistem gerçekleme zamanında korunsun ve işe yarasın?

Değişkenliğin modellenmesinde uygun bir çözüm, bağlamdan bağımsız dillerin (BBD) kullanımıdır. Özellik modellerinin modellenmesinde gramerlerin kullanımı önerisi ortaya atılmış[13], ancak BBD kullanılmamış, bunun yerine özellik ağacını göstermek için özel dönüşümler ve notasyonlar tanımlamıştır. [14]'de ise döngüsel bir gramer kullanılarak– birden fazla(+) ve sıfırdan fazla(*) yapıları ile isteğe bağlı özellikleri göstermek için gramere özel bir notasyon eklenmiştir. Bununla birlikte BBD kullanımının özellik modellerini –dolayısıyla değişkenliği– ifade etmekte yetersiz kaldığı öne sürülmüştür. Bu noktaya daha sonra değinilecektir.

BBD bilindiği üzere üreysel *(non-terminal, generic)* simgeleri sonuçsal *(terminal, non-generic)* bir ya da daha çok simgeye dönüştüren, BBG'nin tanımladığı kuralların oluşturduğu dildir. Buna göre A bir üreysel simge, α da herhangi bir üreysel ve/veya sonuçsal simge dizisi ise, BBG, $A \rightarrow \alpha$ gibi üretim kurallarından oluşur (Daha fazla bilgi için örneğin [12]'e bakılabilir.) Böyle bir gramerin tanımladığı BBD ise, o gramer kapsamındaki kurallar ile türetilebilecek tüm sonuçsal simge dizilerini içeren kümedir.

BBG kuralları, 3.1 numaralı bölümdeki değişkenlik noktaları ve değişkenlikler için kullanılan adlandırmayı anlamlandırmaktadır. Değişkenlik noktaları üreysel (büyük harfler), değişkenlikler ise sonuçsal simgelerle (küçük harfler) ifade edilmektedir. Sonuçsal simgeler BBG ile tanımlanan BBD'nin alfabesini oluşturmaktadırlar. Bu durumda sonuçsal simgelerden oluşan bir dizgi ürün hattına ait bir sistemi ifade etmekte kullanılır. Bu dizginin geçerli bir sistemi ifade edebilmesi için BBG kurallarına uyması gerekmektedir. Şekil 2-1'deki özellikler arasındaki ilişkilerin BBG ile ifadesi, kaygılarda bulunan değişkenlik noktaları ve değişkenlikler arasındaki ilişki için kullanılabilir. Buna göre Şekil 2-1'deki a-e arasındaki her bir ilişkiye karşılık gelen ifade 3.1–3.5 arasındaki üretim kurallarıyla verilmiştir. Bu ifadelerin değişkenlik arasındaki hangi ilişkilere karşılık geldiği 2.2.1 numaralı bölümdeki Şekil 2-1'e gönderme yapılarak açıklanmıştır[1].

$$A \rightarrow a\ b \tag{3.1}$$

$$\begin{aligned} A &\rightarrow a\ A1 \\ A1 &\rightarrow b\ A2 \\ A2 &\rightarrow c \mid \Lambda \end{aligned} \tag{3.2}$$

$$A \rightarrow a \mid b \tag{3.3}$$

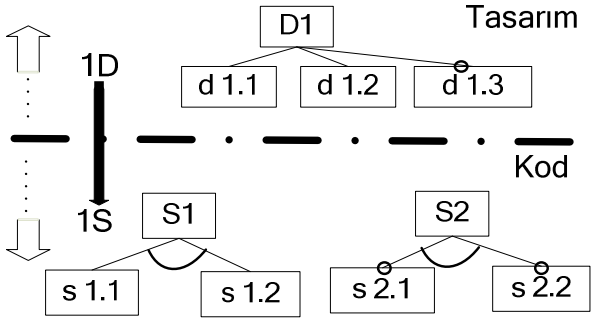
$$\begin{aligned} A &\rightarrow a \mid A1 \mid d \mid \Lambda \\ A1 &\rightarrow b \mid c \mid \Lambda \end{aligned} \tag{3.4}$$

[1] (3.1)'deki ifade ortaklığı tanımlamaktadır ve bir değişkenlik modelinde yer alması beklenmemektedir. Buna rağmen Şekil 2-1 ile bütünsellik açısından dâhil edilmiştir.

$$A \rightarrow a \mid b \mid a\ b \tag{3.5}$$

Her hangi bir üretim kuralını ifade edebilmek ve yeni bir kademe üretim kuralı yaratmak için, üreysel simgelerin sonuna sayılar getirilerek ((3.1)'de A1 gibi veya gerektiğinde bir alt kademeye inilerek (A1.1)) yenileri oluşturulabilir. Bu şekilde bir gösterim zorunluluktan ya da yalın bir ifade elde etmek için tercih edilebilir. Bununla birlikte isteğe bağlı değişkenlikler için [13] veya [14]'deki gibi özel bir notasyon değil, $A \rightarrow \Lambda$ şeklinde boş dizgi üretimini gösteren ek bir üretim kuralı kullanılmaktadır.

Kaygılar içerisindeki değişkenlik modellenirken gündeme gelen bir diğer konu da farklı kaygı soyutlama seviyeleri arasındaki geçişin BBG ile nasıl ifade edileceği konusudur. Şekil 3-2'de tasarım ve kaynak kod katmanları belirtilen kısmi bir kaygı görünmektedir.



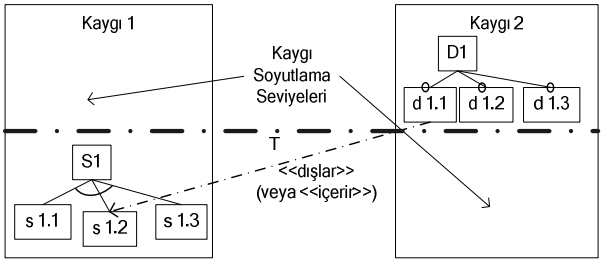
Şekil 3-2 Kaygı katmanları arasında geçiş

Bölüm 3.1'de yasaklanan gösterim iki katman arasındaki geçişi sağlamak için kullanılan 1D-1S üreysel simge çiftidir. Geçiş çifti simgelerinin sayısı tercihe bağlıdır, önemli olan sistemin kaygı modeli çapında tek olmalarıdır. Şekil 3-2'deki kısmi kaygıyı ve katmanlar arası geçişi ifade eden üretim kuralları 3.6'da verilmiştir.

$$\begin{aligned} D1 &\rightarrow d1.1\ d1.2\ D\ 1.1\ 1D \\ D\ 1.1 &\rightarrow d1.2 \mid \Lambda \\ 1D &\rightarrow 1S \mid \Lambda \\ 1S &\rightarrow S1 \mid S2 \mid S1\ S2 \mid \Lambda \\ S1 &\rightarrow s1.1 \mid s\ 1.2 \\ S2 &\rightarrow s2.1 \mid s2.2 \mid \Lambda \end{aligned} \tag{3.6}$$

Bu kurallardan katmanlar arası geçişi sağlayan $1D \rightarrow 1S \mid \Lambda$ kuralındaki $1D \rightarrow \Lambda$ üretimi, sistem gerçekleme zamanında kaygının ilişkili olduğu tüm katmanlarındaki değişkenliğin çözümlenmemesi özgürlüğünü sağlamak için konulmuştur. Bu sayede kısmi sistem belirtimleri ile de yetinilebilmektedir. Bu anlamda kaygılar, değişkenliğin istendiği ve gerekli görüldüğü ölçüde modellenmesi özgürlüğünü de sunmaktadırlar.

Değinilmesi gereken son nokta ise BBG'lerin özellik modellerini, dolayısıyla da değişkenlik modelleri olan kaygı modellerini ifade etmekteki yetersiz kalabilecekleri görüşüdür [14]. Bunun asıl nedeni olarak içerir ve dışlar ilişkilerinin gösteriminin zorluğu öne sürülebilir. Durumu açıklamak için Şekil 3-3'te d1.1 ile s1.2 değişkenlikleri arasındaki mevcut olan dışlar –veya içerir– ilişkisini değerlendirelim.



Şekil 3-3 Kaygılar arası ilişkiler

Eğer bu dışlar ilişkisi BBG ile ifade edilmek istenseydi, (3.7)'deki üretim kuralının tümleyen kümesindeki tüm kuralların bu gramerde bulunması gerekirdi. Bunun nedeni BBG'lerin hangi üretim kurallarına izin verilmediğini değil ancak hangilerine izin verildiğini gösterebilmesidir.

$$D1 \rightarrow d1.1T \mid d1.2 \mid d1.3 \mid \Lambda$$
$$T \rightarrow s1.2 \qquad (3.7)$$

"İçerir" ilişkisi modellenmesi daha kolay olmakla birlikte kaygılar arasında geçişi sağlayan üreysel simgelerin –Şekil 3-3 T gibi- çoğalmasına neden olacağı için, "içerir", "dışlar" ile aynı kapsamda değerlendirilmiştir. Bununla birlikte [14]'de, "A, B veya C veya D'yi içerir" gibi karmaşık üretim kurallarının da BBG ile ifade edilmesinin mümkün olmadığı, salt içerir ve dışlar ilişkilerinin de değişkenlik modelleme açısından yetersiz olduğu belirtilmiştir. Bütün bu sıkıntıları göğüslemek adına kaygıların modellenmesinde kaygılar arası içerir, dışlar veya karmaşık içerir-dışlar ilişkileri, kaygılar içindeki değişkenliğin modellenmesinden ayrı tutulmuştur.

Bu ilişkilerin ifade edilmesi kuramsal olarak bakıldığında bir Turing makinesi, yani bu kuralların işletildiği bir algoritma ile çok daha kolayca gerçekleştirilebilir. Bunun için C programlama dilindeki derleme işleminden önceki adıma benzer bir önişlemenin, ürün hattına ait bir sistemi ifade eden dizgiyi, gramerin kapsamında bulunmayan ilişkilere uygunluk bakımından incelemesi mümkündür. Öncelikle, bu önişlemci motoru, içerme ve dışlama ile ilgili kuralları denetler. Ardından, BBG ile kaygılar içindeki ilişkiler modellenir ve oluşan BBD'ye ait bir ayrıştırıcı ile önişleme aşamasını geçen dizgiler değerlendirmeye alınır. Bu iki aşamalı denetimi geçen her dizgi ürün hattına ait geçerli bir sistemi tanımlıyor olacaktır.

3.3. Kaygılar ve yararları

Bu çalışmanın temel amaçlarından birisi, değişkenliğin farklı soyutlama seviyelerinde ele alınıp izlenmesidir. Bunu sağlayan kaygılarla destekli ürün hattı yaklaşımının daha az (karşılaştırmak için özellik soyutlama seviyesinden bakılırsa) gerçeklenebilir sistemi ifade eden bir model oluşturması beklendiği, 4. bölümde ele alınacaktır. Bu bir dezavantaj gibi görünse de, kaygıların bütünleştirildiği bu yaklaşımla üretilebilen sistemler gerçekten üretilebilir sitemlere çok daha yakındırlar. Bu birleşimler gerçeğe çok daha yakın sistemleri ifade etmektedirler. Kaygılar, özelliklerden daha derinde bulunan değişkenlik ilişkilerini su yüzüne çıkartarak daha yalın ve doğru bir ürün hattı değişkenlik modeli elde edilmesini sağlamaktadır. Amaç, gerçeklenebilir sistemleri daha yalın bir şekilde ifade eden bir kaygı –dolayısıyla değişkenlik- modeline sahip olmaktır.

İkinci yarar ise birinci ile doğrudan ilişkilidir. Ürün hatlarındaki yaygın sorunlardan biri, özelliklerin çatışmaları sorunudur. Bu sorun belirli birtakım özelliklerin aynı anda sistemde var olması durumunda sistemin doğru çalışmasına kötü etki edecek veya engelleyecek etkenlerin ortaya çıkması olarak tanımlanabilir. Kaygıların, değişkenliği, birçok soyutlama seviyesinde ele alarak daha doğru yönetmeyi ve böylece ürün hattı yaklaşımda özellikler arasında bulunması gereken, ancak salt özellik modelleme ile tespit edilemeyen ilişkileri ortaya çıkartmasının, özellikler arası çatışma sorununu da azaltması beklenmelidir.

4. Kaygılar için değerlendirme ölçütleri

Bölüm 3.3'te belirtilen yararların gerçekten sağlanıp sağlanmadığını ölçmek için bir takım değerlendirme ölçütleri tanımlanmalıdır. Bu ölçütler genel olarak ürün hattı yaklaşımlarını karşılaştırmada kullanılabilirler.

İlk önerilen ölçüt, denklem (4.1) ile verilen kabul edilebilirlik oranı olarak tanımlanmaktadır. Kabul edilebilirlik oranı (PR), uzmanlar ve kullanıcılar tarafından kabul gören sistemlerin (Na), ürün hattı değişkenlik modelleme yaklaşımı kullanılarak ifade edilebilecek tüm sistemlere (Nt) oranıdır. Bu oran ideal durumda 1'dir.

$$PR = Na / Nt \qquad (4.1)$$

Bu oranın sağladığı bilgi değerli olmasına karşın, bu bilgiyi elde etmek zor olabilmektedir. Bunun nedeni, Na değerini belirleyebilmek için kullanıcı veya alan uzmanlarının ciddi işgücüne ihtiyaç duyulmasıdır. Bununla birlikte bu oran değişkenlik modellemeyi farklı ele alan iki ürün hattı yaklaşımını karşılaştırmada da zorlukla kullanılabilir. Durumu bir örnek ile açıklamak için salt FORM ile çıkartılan bir değişkenlik modelini –model 1– FORM'A kaygıların bütünleştirilmiş hali ile çıkartılan bir model ile -model 2– karşılaştırmak gerekmektedir. Model 2'de model 1'e kıyasla çok daha fazla değişkenlik noktası –dolayısıyla değişkenlik– olması beklenmektedir. Bu durumda hem Na hem de Nt değerleri artacaktır. Ancak PR, model 2'deki bilgi ile hesaplanırsa, model 1 bilgileri ile hesaplanacak orana göre nasıl bir değer alacağını (az ya da çok) kestirmek mümkün olmayabilir. Bunun nedeni model 2'deki değişkenlik bilgisinin model 1'dekine göre ne kadar ve nasıl fazla olup olmayacağının bilinemeyişidir.

Ancak iki farklı modelde de değişkenliğin ürün gerçekleme zamanında alacağı değeri belirlemek için aynı değişkenlik bilgisi kullanılırsa, iki yaklaşım karşılaştırılabilir olacaktır. Bu durumda karşılaştırma yapabilmek için model 2'de, sadece kaygılardaki özellikler katmanı kullanılırsa, iki modelde de kabul gören sistem sayısı aynı olacaktır (Na). Karşılaştırmanın sağlıklı olması için, modelleme yaklaşımı kullanılarak ifade edilebilecek sistem sayısı (Nt) hesaplanırken model-2'de kaygıların özellik katmanı arasındaki ilişkiler dikkate alınmalıdır. Eğer model 1 ve 2 için hesaplanan PR değerlerinin ($PR_1=Na_1/Nt_1$, $PR_2=Na_2/Nt_2$) birbirine oranı

hesaplanırsa ve $Na_1 = Na_2$ olduğu dikkate alınırsa denklem (4.2)'deki bağıl kabul edilebilirlik oranına (RPR) ulaşılmaktadır.

$$RPR_{1,2} = \frac{PR_1}{PR_2} = \frac{Nt_2}{Nt_1} \qquad (4.2)$$

Bu değer 0'a ne kadar yakın olursa model 2'nin değişkenliği model 1'den o kadar daha iyi modellediği söylenebilir.

İkinci değerlendirme ölçütü, bölüm 3.3'teki kaygı yararlarında da belirtildiği gibi ilk yarar ve dolayısıyla ilk ölçüt ile yakından ilişkilidir. Yanlış veya eksik değişkenlik modelleme, ürün hatlarında karşılaşılan özelliklerin çatışması sorununu doğurmaktadır. Kaygıların, özellik modelinde ilk anda fark edilemeyen ilişkileri ortaya çıkartarak özellikler arası ilişkilerdeki çatışmaları azaltması beklenmektedir. Değişkenlik modellerinde özellikler arası çatışmaların önüne "dışlar" ile geçilir. Bu nedenle aynı değişkenlik noktaları ve değişkenler için bir ürün hattı yaklaşımı daha çok "dışlar" ilişkisi tespit edebiliyorsa, bu yaklaşımın özellik çatışmalarını engellemekte daha başarılı olduğu söylenebilir.

Ürün hattı yaklaşımları karşılaştırılırken, "dışlar" ilişkilerindeki yüzdesel artış, yaklaşımların özellik çatışmalarını yönetme yeteneklerini karşılaştırmakta kullanılabilir. Buna göre özellik çatışmasını engellemedeki bağıl yüzdesel artış (A), denklem (4.3)'deki gibi tanımlanmaktadır.

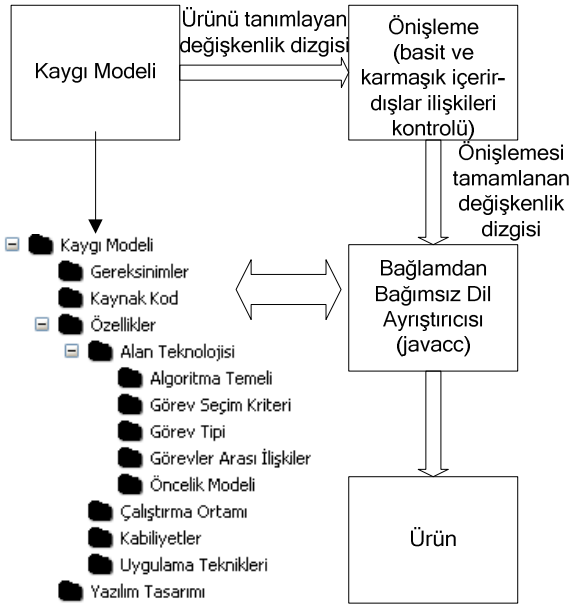
5. Kaygıların uygulanması

Kaygıların uygulanıp yararların değerlendirilmesi amacıyla, 4. bölümde de değinildiği gibi, "FORM" (1. yaklaşım) ve "kaygıların bütünleştirildiği FORM" (2. yaklaşım), ürün hattı yaklaşımları kullanılarak gerçek zamanlı işletim sistemlerinde sıralama algoritmaları alanının ele alındığı bir çalışma yapılmıştır. Kaygılar ve FORM'un birlikte kullanıldığı yaklaşım, temelde bölüm 3.2'de anlatılan sürecin gerçekleştirmesi olarak algılanabilir. Kaygıların uygulamada kullanıma alınmasının nasıl sağlandığı Şekil 5-1'de gösterilmiştir.

$$A_{1,2} = \frac{En_2 - En_1}{En_1} \times 100 \qquad (4.3)$$

4.3'deki değerlere yakından bakılacak olursa,
$A_{1,2}$: Model 1 ve 2 arasındaki bağıl özellik çatışması engellemedeki artış yüzdesi,
En_1: Model 1'deki "dışlar" ilişkisi sayısı,
En_2: Model 2'deki "dışlar" ilişkisi sayısı
olduğu görülür.

Bu değer, ürün hattındaki değişkenliğin tamamının modellenmesi göz önünde bulundurularak da hesaplanabilir. Ancak ilk ölçüt için de belirtildiği gibi, ürün hattı yaklaşımlarını karşılaştırmak söz konusu olduğunda iki yaklaşımda da aynı soyutlama seviyesindeki değişkenlikleri ele almak doğru olacaktır.



Şekil 5-1 Kaygıların Değişkenlik Modellemede Kullanımı

Kaygı modelindeki seçimlerin yapılması için uygun ve basit bir yapı olan ağaç yapısı tercih edilmiştir. Bu yapı kullanılarak ürünü tanımlayan değişkenlikler belirlenir. Arka planda da değişkenliği tanımlayan dizgi yaratılmış olur. Bu dizgi öncelikle "içerir" ve "dışlar" ilişkilerine göre (karmaşık/basit) ürünü tanımlayan dizgiyi kontrol eder. Bu ön işlemden geçen dizgi, kaygı modeli kullanılarak oluşturulan BBD ayrıştırıcısına verilir. Bu ayrıştırıcıyı oluşturmak için *javacc* aracı kullanılmıştır[1]. Eğer bu ayrıştırıcı da dizgiyi BBG'nin kurallarına uygun buluyorsa artık bu dizgi ürün hattına ait gerçek bir sistemi ifade etmektedir.

Uygulama sırasında, 2. yaklaşımda, kaygılar değişkenlik modellemede anlatılan şekilde, FORM'un diğer öğeleri de değiştirilmeden kullanmıştır. Karşılaştırma yapılan 1. yaklaşımda ise salt FORM –dolayısıyla özellik modeliyle değişkenlik modelleme– kullanılarak aynı alan ele alınmıştır.

6. Tartışma

Örnek alan üzerinde kaygıların etkilerinin değerlendirildiği çalışma halen devam etmekle birlikte, kaygı seviyelerinden ikisinin değerlendirilmesi – özellikler ve gereksinimler – tamamlanmıştır. Buna göre, gereksinimlerin salt özellik modellemeye göre ne oranda daha fazla özellikler arası ilişki ortaya çıkarttığı ve özellik çatışmasını azalttığı değerlendirme ölçütleriyle görülebilir. Buna göre, kaygılarla desteklenmeyen FORM yaklaşımı örnek alanda toplam 19 değişkenlik noktası ve 50 değişkenlik ortaya çıkmıştır. FORM kullanıldığı durumda N_{t1} 18911232, En_1 63 değerini, FORM ve kaygılar kullanıldığı durumda da N_{t2} 2285568, En_2 96 değerini almıştır

[1] https://javacc.dev.java.net/, son olarak 21.04.2009 tarihine kadar geçerlidir.

Bu değerlere göre, $RPR_{1,2}$ ve $A_{1,2}$ değerleri (6.1) ve (6.2)'deki gibidir[1].

$$RPR_{1,2} = 0.12 \quad (6.1)$$

$$A_{1,2} = 52.4\,\% \quad (6.2)$$

4. bölümdeki açıklamalar ışığında, kaygıların kısmi olarak dahi değişkenlik modellemede kullanılmasının yararları açıkça görülmektedir. Beklenti, kaygı modeli tamamlandığında bu değerlerin gene kaygılar lehine değişmesi yönündedir. Ancak şuna da dikkat çekilmelidir ki, sadece gereksinimlerin kaygılar aracılığı ile değişkenlik modellemede kullanılması bile, değişkenlik modelinin ciddi anlamda daha gerçekçi olmasına katkı sağlamaktadır. Buna dayanarak, eksiksiz kaygı modeli geliştirmek için ayrılması gereken kaynak ve işgücü kaygı modeli oluşturmayı cazip kılmıyorsa, kısmi kaygı modeli geliştirmenin de faydalı olacağı söylenebilir.

7. Sonuçlar

Yazılım ürün hatlarında değişkenliğin modellenmesi ürün hattının ifade ettiği alanı doğru yansıtabilmesi bakımından önemli bir aşamadır. Değişkenlik modellemede kullanılacak yapıların bu önemli işlevi eksiksiz ve sistematik bir biçimde yerine getirebilmesi ürün hattının başarısı bakımından kritik öneme sahiptir.

Bu çalışma, değişkenliğin modellenmesi için kaygı kavramını ortaya atmaktadır. Kaygılar ürün hattındaki değişkenliği, farklı soyutlama katmanlarında ele almaktadır. Böylece, farklı varlıklara dağılmış değişkenlik bilgisini modellemektedirler. Buna ek olarak, yine bu çalışmada, kaygıların BBG kullanımı ile modellenmesi önerilmiştir. Kaygı kullanarak sistem tanımlama için iki aşamalı bir yaklaşım benimsenmiştir. Buna göre kaygı bünyesindeki değişkenliklerden oluşan bir dizgi ile aday bir sistem tanımlanması sağlanır. Bu dizgideki değişkenlikler arasındaki karmaşık ve basit içerir/dışlar ilişkileri bir ön işleme programı ile kontrol edilir. Sonrasında ise kaygı modelindeki diğer ilişkilerden türetilmiş BBD ayrıştırıcısı yardımı ile bu dizgi dilin kurallarına uygunluk bakımından değerlendirilir. Bu iki aşamayı geçen dizgiler ürün hattına ait bir sistemi tanımlamaktadırlar.

Bu yönteme ek olarak, ürün hatlarının yararlarını belirlemede kullanılabilecek iki yeni değerlendirme ölçütü önerilip, kaygıların geçerliği bu ölçütlerle değerlendirilmiştir.

Kaygıların sistem bütünündeki değişkenliğe değinmeleri nedeni ile sistem tanımlamada alana özel bir dilin altyapısını oluşturabileceği düşünülmektedir. Bununla birlikte kaygı modelinin, otomatik kod üretimine ve üretici programlamaya *(generative programming)* girdi oluşturabileceği öngörülmektedir. Gelecekteki çalışmalar, ilk olarak, kaygıların kullanımı ile üretici programlama arasında sistematik bir bağ kurmaya odaklanacaktır.

[1] Bu değerler, kaygı belirtimlerini işleyen ön işlemci girdisi üzerinden, bu çalışma kapsamında geliştirilen bir hesaplama aracı ile belirlenmiştir.

8. Kaynakça

[1] Kathrin Berg, Judith Bishop, Dirk Muthig, *Tracing Software Product Line Variability – From Solution to Problem Space*, Proceedings of SAICSIT 2005

[2] Muthig Dirk, Atkinson Colin, *Model-Driven Product Line Architectures; Lecture Notes In Computer Science; Vol. 2379 archive*, Proceedings of the Second International Conference on Software Product Lines 2002

[3] William B. Frakes ve Kyo Kang, *Software Reuse Research: Status and Future*, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 7, JULY 2005

[4] M. D. McIlroy, *Mass produced software components*, Proc. NATO Software Eng. Conf., Garmisch, Germany (1968) 138-155

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional Computing Series 1994

[6] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, A Spencer Peterson, *Feature Oriented Domain Analysis (FODA) Feasibility Study*, Technical Report CMU/SEI-90-TR-021, November 1990.

[7] K. Czarnecki, S. Helsen ve U. Eisenecker; *Staged configuration through specialization and multi-level configuration of feature models*. Software Process Improvement and Practice, 10(2):143–169, 2005.

[8] K. Pohl, G. Böckle ve Frank van der Linden, *Software Product Line Engineering: Foundations, Principals and Techniques*, Springer, Berlin Heidelberg New York, 2005.

[9] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Gerard Jounghyun Kim, Euiseob Shin, FORM: *A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures*, Annals of Software Engineering, 1998

[10] H. Gomaa, *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*, Addison Wesley Longman Publishing Co., Inc., 2004.

[11] A. Metzger ve P. Heymans. *Comparing feature diagram examples found in the research literature*. Technical report TR–2007–01, Univ. of Duisburg-Essen, 2007

[12] John Martin; *Introduction to languages and the theory of computation*; Mc Graw-Hill 2003

[13] M. de Jong ve J. Visser, *Grammars as Feature Diagrams*, CWI 2002

[14] Don Batory, Feature Models, *Grammars ve Propositional Formulas*, Software Product Line Conference 2005