Alloy Analyzer Kullanarak Emniyet Kritik Aviyonik Yazılım Gereksinimlerinin İncelenmesi

Burak Ata[1] Halit Oğuztüzün[2] Reşat Erhan Yüceer[3]

[1,3]Savunma Teknolojileri ve Mühendislik A.Ş., Mecnun Sokak No:58 Beştepe, 06510 Ankara, Türkiye
[2]Bilgisayar Mühendisliği Bölümü, ODTÜ, İnönü Bulvarı, 06531 Ankara, Türkiye

[1]e-posta: bata@stm.com.tr [2] e-posta: oguztuzn@metu.edu.tr
[3] e-posta: eyuceer@stm.com.tr

Özetçe

Bu bildiride aktarılan çalışmada bir küme emniyet kritik aviyonik yazılım gereksinimi ele alınmış ve bu gereksinimler Alloy Analyzer aracı ile analiz edilmiştir. Bu amaçla önce doğal dilde yazılmış olan gereksinimler mantıksal ifadelere çevrilmiştir. Ardından bu ifadeler Alloy Analyzer ile modellenmiş, tutarsızlıklar ve eksiklikler bakımından incelenmiştir. Karşılaşılan zorluklar ve çözümleri (modellemeye yönelik zorluklar gibi) aktarıldıktan sonra sonuçlar değerlendirilmiştir.

1. Giriş

1.1. Aviyonik Yazılım ve Gözden Geçirme Aktiviteleri

Aviyonik yazılım ürünlerinin herhangi bir uçakta kullanılmadan önce uçuşa elverişlilik onayı alması gerekmektedir. Bu onayı alabilmek için onayı verecek olan otorite tarafından belirlenmiş bazı standart ve/veya rehber dokümanlar ile uygunluk göstermek gerekir. Bu tarz hedefleri olan projelerde harcanan eforun büyük bir kısmı işin geliştirilmesi aşamasında değil geçerleme ve doğrulama aşamasında harcanır, çünkü amaç "doğru işin", "doğru olarak" yapıldığını ispat etmektir.

Geçerleme ve doğrulama aşamasında ortaya çıkan eforlar proje bütçesini de etkilemektedir. Örneğin Hilderman'a göre RTCA'in (The Radio Technical Commission for Aeronautics) yayımladığı "DO-178B, Software Considerations in Airborne Systems and Equipment Certification" rehber dokümanına uyumlu yazılım projesi geliştirmek %20 ile %40 civarında ek maliyet getirmektedir [1]. Bu rakamlar DO-178B uyumlu yazılım geliştirme konusunda deneyimli firmalar göz önünde bulundurularak hesaplanmıştır. Daha önce bu konuda tecrübesi olmayan firmalar için bu değerler %200 gibi rakamları bulmaktadır [1]..

Harcanması gereken efor ve maliyet hesapları için önemli bir girdi ise geliştirilecek yazılımın emniyet kritiklik seviyesidir. DO-178B A, B, C, D ve E olmak üzere beş emniyet seviyesi belirlemiştir [2]. Seviye A, emniyet kritiklik seviyelerinin en yükseği iken seviye E en düşüğüdür. Aviyonik yazılımın tamamı ele alındığında geliştirilen yazılımın ortalama %35'ının seviye A, %30'nun seviye B, %20'sinin seviye C, %10'unun seviye D ve geri kalanın da seviye E olarak geliştirildiği görülmektedir [1]. Emniyet seviyesinin maliyete etki etmesinin başlıca iki sebebi vardır; ilki karşılanması gereken hedeflerin kritiklik seviyesi ile paralel artmasıdır. İkinci sebep ise bağımsızlık isterleridir. DO-178B'ye göre kimi aktiviteler (özellikle doğrulama aktiviteleri) emniyet seviyelerine bağlı olarak bağımsız bir ekip tarafından karşılanmalıdır. Örneğin Emniyet seviyesi A olan bir yazılım bölümü için geliştirilen yazılımın ayrı (bağımsız) bir ekip tarafından doğrulanması gerekmektedir. *Tablo 1*'de DO-178B'de emniyet seviyelerine göre karşılanması gereken hedefler verilmiştir [2].

Emniyet Seviyesi	Hata Durumu	Amaç	Bağımsızlık Gerektiren
A	Ölümcül	66	25
B	Tehlikeli	65	14
C	Önemli	57	2
D	Az Önemli	28	2
E	Etkisiz	0	0

Tablo 1: Emniyet Seviyeleri ve DO-178B

Tablo 1'de verilen amaçların çoğu doğrulama aktivitelerine karşılık geldiği için doğrulama sürecinin maliyetinin yazılım geliştirme projesinin maliyetini aşması aviyonik projeleri için beklenen bir durumdur. Doğrulama süreçlerinde herhangi bir hatanın geç fark edilmesi, sorunun çözülmesi için harcanan zaman ve eforu arttırmaktadır. Bu sebeple birçok firma proje giderlerini minimuma indirmek ve proje takvimine uyabilmek için sorunlu kısımları mümkün olduğunca çabuk tespit etmeyi amaçlamaktadır. Bu kapsamda gözden geçirmeler oldukça önem kazanmaktadır. Hataları erken yakalayabilmek için yazılım gereksinimlerinin hazır olması ile birlikte diğer aktivitelere paralel olarak bir dizi gözden geçirme aktivitesi başlar; yazılım gereksinimleri gözden geçirmeleri, yazılım tasarımı gözden geçirmeleri, yazılım kod gözden geçirmeleri ve yazılım test durumu gözden geçirmeleri gibi.

Yazılım gereksinimleri gözden geçirmeleri, yazılım doğrulama aktiviteleri açısından ilk kontrol noktasıdır. Bu aktivite esnasında yazılım gereksinimleri, emniyet seviyesinin gerektirdiği bağımsızlık kriterine uygun şekilde gözden geçirilir. Bu esnada yakalanacak uygunsuzluklar büyük önem arz etmektedir. Bu uyumsuzluklar gözden geçirme sürecinde yakalanmadığı takdirde yanlış/eksik tasarım veya yanlış uygulama olarak ileriki aşamalarda fark edilecek ve geri dönüşü oldukça masraflı olacaktır. Öte yandan yazılım gereksinimleri başarılı bir şekilde gözden geçirilir ve projenin bundan sonraki aşamaları sağlam bir gereksinim kümesinin üzerine inşa edilirse daha sonraki aşamalarda yakalanacak hatalar asgari seviyeye indirilmiş olur.

Aviyonik yazılım projelerinin çoğunda özellikle üst seviye gereksinimler geliştirilirken doğal dil kullanılmaktadır. Doğal dilde hazırlanan bu gereksinimler daha sonra test edilebilirlik,

tutarlılık, tam olup olmama gibi kriterler göz önünde bulundurularak genellikle yazılım, test ve kalite mühendisleri tarafından gözden geçirilmektedirler. Gözden geçirme süreci otomatik olmadığından deneyimli bir ekip tarafından yürütüldüğünde bile bazı hataların yakalanamama riski hep mevcuttur. Masa başı gözden geçirmelerin bir diğer dezavantajı ise projenin en yoğun dönemlerinde birçok kalifiye elemanın eforuna ihtiyaç duymasıdır.

Bu bildiriyle açıklanan çalışmada tutarlılık ve tam olma kriterlerine göre gereksinim gözden geçirme sürecinin Alloy Analyzer adlı araç ile otomasyonu sorgulanmıştır. Alloy Analyzer MIT'deki Software Design Group tarafından geliştirilmiş bir model analizcisi olup birinci derece mantık üzerine kurulu basit yapısal bir modelleme dili olan Alloy ile hazırlanmış modelleri analiz etmek için kullanılmaktadır [3]. Çoğu yazılım gereksinimi birinci derece mantık ile ifade edilebildiği için seçilen aracın ihtiyacı karşılayabileceği düşünülmüştür. Karşılaştırmalı bir sonuç elde edebilmek için daha önceden masa başı gözden geçirmeleri tamamlanmış bir gereksinim kümesi belirlenmiş ve bu gereksinimler Alloy dilinde ifade edilerek Alloy Analyzer ile tutarlılık ve tam olma kıstaslarına göre analiz edilmiştir.

1.2. Alloy Analyzer Tanıtımı

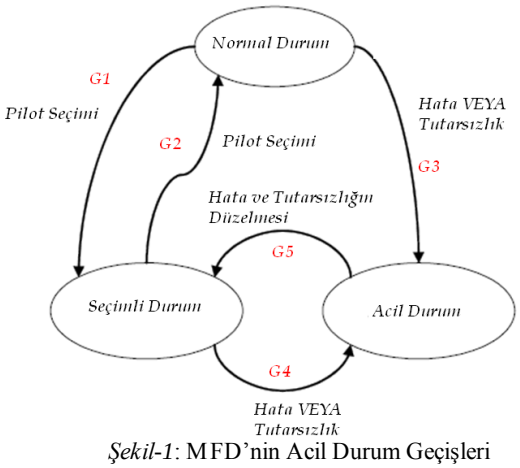
Alloy aracı Z gibi bildirimsel dilleri tam otomatik analizlere adapte edebilmek amacı ile geliştirilmiştir. (Z yüksek dereceli mantığı desteklemekte olduğundan, semantik anlamdaki otomatik analiz için elverişli değildir. Yüksek dereceli mantıkta niceleyiciler, kümeler, dolayısıyla fonksiyonlar ve ilişkiler üzerine çalışabilmektedir.) Aracın ilk tasarımları Professor Daniel Jackson'un öncülüğündeki MIT Software Design Group tarafından yapılmıştır. Alloy birinci derece mantıktan yararlanarak belirtimleri büyük Boolean ifadelere çevirebilmektedir. Daha sonra da bu ifadeler SAT (Boolean satisfiability problem) çözücüler tarafından otomatik olarak analiz edilmektedirler. Dolayısıyla mantıksal formüller araca girdi olarak verildiğinde, Alloy Analizer bu formülü karşılayacak modelleri bulmaya çalışmaktadır [3]. Bir başka deyişle, birinci derece mantık ile modellenebilecek herhangi bir tasarım Alloy Analizer ile analiz edilebilir.

2. Gereksinim Kümesinin Modellenmesi

2.1. Örnek Gereksinim Kümesi

Modern uçak kokpitlerinin çoğunda pilotlara seyr-ü sefer bilgilerini sunan ve acil durumlar hakkında bilgi veren çok fonksiyonlu ön panel göstergeleri (MFD: Multifunction Display) bulunmaktadır. Bu ön panel göstergelerinin kimi kendisine gelen veriyi aynen ekrana basarken, kimileri ise ham veri alarak işlemekte, arkasından da bu işlemlerinin sonuçlarını ekranlarında göstermektedir. İlk tip göstergelere işlevsiz MFD denirken, ikinci tip göstergelere akıllı MFD denmektedir. Alloy ile ifade edilmek üzere seçilen gereksinim kümesi akıllı MFD'nin acil durum mod geçişlerine dair işlemleri anlatan kısmıdır.

Gereksinimlerde *Şekil-1*'deki durum geçiş diyagramı açıklanmıştır. Akıllı MFD'ler normal durumda iken pilot tarafından seçilmiş bilgileri göstermektedirler. Öte yandan bir anormallik fark edildiğinde (örneğin, sensorlardan gelen verilerde tutarsızlığın veya hata durumunun fark edilmesi) göstergeler acil durum moduna geçmekte ve sadece uçuş kritik verileri göstermektedirler. Acil durum modunda kullanıcı girdileri mod değişimine yol açmamaktadır. Bir başka deyişle, MFD'ler bir kez acil durum moduna geçtikten sonra emniyet gereksinimleri nedeniyle pilotun başka bir moda geçmesine izin verilmemektedir. Acil durumda iken anormalliklerin düzelmesi durumunda MFD'ler seçimli moda geçmektedirler. Seçimli modda acil durum modundaki gibi sadece uçuş kritik veriler gösterilmekte ancak pilota mod değiştirme yetkisi verilmektedir. Diğer bir deyişle, MFD'ler seçimli modda iken herhangi bir sorun gözlemlenmediği durumda pilot istediği verileri ekranında görmek konusunda özgürdür. Ancak seçimli modda iken hata durumu ya da sensor verilerinde tutarsızlık fark edilmesi halinde acil durum moduna geçiş gerçekleşmelidir. Öte yandan MFD normal modda iken pilot istediği takdirde seçimli moda da geçebilmektedir. Bu seçenek pilota önemli verileri tek bir seçimle göz önüne getirebilme esnekliğini sağlamaktadır.



Şekil-1: MFD'nin Acil Durum Geçişleri

Şekil-1'de de gösterildiği gibi her geçiş için bir gereksinim yazılmıştır (G1, G2, G3, G4, G5 ve G6) ve incelenen gereksinim grubunda toplamda yedi adet gereksinim bulunmaktadır. Öncelikle doğal dilde yazılmış olan bu gereksinimler mantıksal önermelere çevrilmiştir. Aşağıda bir gereksinim için bu işlemin nasıl yapıldığına dair örnek verilmiştir.

Software_Req_003:
While in the *Normal* state **AND** a failure is detected the Flight Display Mode Control (FDMC) **SHALL** transition to the *Emergency* state.
[Uçuş Göstergesi Mod Kontrol yazılımı *Normal* modda ise **VE** bir hata durumu fark edildi ise Acil Durum moduna **geçmelidir.**]

Sistemde iki adet akıllı MFD vardır ve ***Software_Req_003*** her ikisi için de geçerlidir. Doğal dilde yazılan tek gereksinim her iki MFD için de yeterli olmasına rağmen önermelere geçişte her iki gösterge için ayrı gereksinimler yazmak gerekmektedir.

```
MFD1_Mode = Mode1Normal AND MFD1_Failure
= Detected -> MFD1_Mode '=MFDEmergency

MFD2_Mode = Mode2Normal AND MFD2_Failure =
Detected -> MFD2_Mode '=MFDEmergency
```

2.2. Alloy Uygulaması

Sistemin durumları MFD1 ve MFD2 olmak üzere iki küme değişkenler üzerinden modellenmiştir. Fark edilmiş hata durumları ya da pilot seçimleri gibi sistem girdileri ise *Şekil-2* ve *Şekil-3*'teki gösterildiği gibi aksiyonlar (events) olarak modellenmiştir.

```
abstract sig ModeControl {}
one sig MFDOptional,
MFDEmergency,MFDNormal extends
RMC{}

abstract sig Miscompare {}
one sig
Miscompare_Detected,Miscompare_Not
Detected,Cleared extends
Miscompare{}

abstract sig Failure {}
one sig
Failure_Detected,Failure_NotDetect
ed extends Failure {}

abstract sig MFDToggle{}
one sig Received,Not_Received
extends MFDToggle {}
```

Şekil-2: Aksiyonlar için Örnek Kod Parçacığı

Şekil-2'de MFD modu (`ModeControl`), tutarsızlık (`Miscompare`) ve hata (`Failure`) durumu ve pilot seçimleri (`MFDToggle`) tanımlanmıştır.

```
pred ifr3a(s1:System,
e:EventInstance)
{
      s1.mfd1_Mode = MFDNormal &&
Failure_Detected = e.mfd1.f
}
pred r3a(s2:System)
{
      s2.mfd1_Mode=MFDEmergency
}

pred ifr3b(s1:System,
e:EventInstance)
{
      s1.mfd2_Mode = MFDNormal &&
Failure_Detected = e.mfd2.f
}
pred r3b(s2:System)
{
      s2.mfd2_Mode= MFDEmergency
}
```

Şekil-3: İfadeler için Örnek Kod Parçacığı

Şekil-3'de ise gereksinimler ifadelerle modelle aktarılmaya başlanmıştır.

Her gereksinim modellenirken önerme ve sonuçtan oluşan iki adet Alloy kuralı kullanılmıştır. Sistemin tam olarak modellenmesini sağlamak için gereksinimlerde yazmamasına rağmen MFD1 ve MFD2'nin modlarının birbirinden bağımsız olmasını sağlayacak özel kurallar tanımlanmıştır.

Alloy Analyzer'in temelinde yatan analiz yöntemi örnek model oluşturmadır. Dolayısıyla araç, bir belirtimin her koşulda doğru olduğunu ispatlamak yerine, tüketici test yaparak belirtimi çürütmeye çalışmaktadır. Eğer iddianın çürütülebildiği bir durum bulunursa, bu durum karşı örnek olarak kullanıcıya sunulmaktadır. Eğer herhangi bir karşı örnek bulunmaz ise iddianın daha büyük bir test kümesi kullanılarak çürütülme ihtimali teorik olarak mevcuttur [3]. Ancak arama uzayının makul ölçülerde kısıtlandığı durumlarda karşı örnek üretilememesi modelin doğru olduğuna dair güven vermektedir. Model büyüklüğündeki kısıtlamalar, belirtimin özelliğine göre mühendislik sağduyusuyla belirlenir.

Sistem modellenirken gereksinimlere dair kural kontrolleri *Şekil-4*'deki örnekte de gösterildiği üzere belirtimler olarak tanımlanmışlardır. Bu noktadan sonra Alloy Analyzer'dan *Şekil-5*'deki kod parçasında olduğu gibi bu belirtimlere karşı örnek üretmesi istenmiştir. Örneğin *Şekil-5*'de daha önce *Şekil-4*'de tanımlanan `ruleCheck3a` ve `ruleCheck3b` kontrole gönderilmiştir.

```
assert ruleCheck3a
{
      all a:System,
b:next[a],e:EventInstance|
      ifr3a[a,e] => r3a[b]
}
assert ruleCheck3b
{
      all a:System,
b:next[a],e:EventInstance|
      ifr3b[a,e] => r3b[b]
}
```

Şekil-4: Belirtimler için Örnek Kod Parçacığı

```
check ruleCheck3a for 3

check ruleCheck3b for 3
```

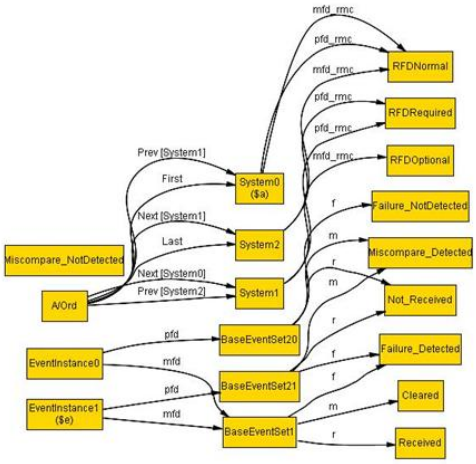
Şekil-5: Kontroller için Örnek Kod Parçacığı

3. Tartışma

3.1. Teknik Sonuçlar

Tüm sistem modellenip analiz edildiğinde araç kimi karşı örnekler bulmuştur. *Şekil-6*'da bu karşı örneklerden biri verilmiştir. Karşı örnekler incelendiğinde sorunun eldeki gereksinimlerin aynı anda meydana gelen olayları kapsamamasından oluştuğu ortaya çıkmıştır. Daha ayrıntıya inmek gerekirse birden fazla gereksinimin girdileri aynı anda karşılanırsa ne olacağı gereksinim kümesinde açıklanmamaktadır. Örneğin hata durumu yakalandığı anda pilot seçimi algılanırsa MFD'nin son modu ne olacaktır; acil

durum modu mu, seçmeli mod mu? Bu gereksinimler konunun uzmanı tarafından okunduğunda bu soruya -emniyetle ilgili olduğu için- acil durum modu cevabının verileceği aşikardır. Ancak bu sağduyu bilgisi olmaksızın, sadece gereksinimlerden yola çıkarak sistemin son moduna karar verebilmek mümkün değildir. Öte yandan sistemi inceleyen bir kişi en kötü ihtimalle MFD'nin kısa bir süre seçmeli moda geçeceğini ve hemen arkasından altı ya da yedi numaralı gereksinimler sayesinde acil durum moduna geçeceğini söyleyerek bu bulgunun önemsiz olduğunu ileri sürebilir. Ancak acil durumlarda pilotların tepki verebilmek için birkaç saniye gibi kısa sürelere sahip oldukları göz önünde bulundurulursa hatalı yorumlanarak yapılan bir uygulamanın sonuçlarının ağır olabileceği görülebilir.



Şekil 6: Alloy Analyzer Tarafından Raporlanan Bir Karşı Örnek (Alloy Analizer çıktısı)

Bu analizden çıkan bir diğer sonuç ise durum geçiş diyagramlarından gereksinimlere kırılım yapılırken zamanlamanın da göz önünde bulundurulmasının gereğidir. Benzer bir durum geçiş diyagramının, füze ateşleme sistemi için kullanıldığını varsayalım. Normal durumda iken pilotun tetiklemesi ile bir füzenin ateşleneceğini düşünelim. Öte yandan görev kontrol bilgisayarından bir uyarı geldiği takdirde pilotun ateşleme yetkisi iptal olsun. MFD mod geçişlerinde olduğu gibi bir durum geçiş diyagramının çizildiğini ve aynı yöntemlerle sadece geçişleri belirten ancak olayların zamanlamasını göz önünde bulundurmayan bir gereksinim kümesinin yaratıldığını varsayalım. Pilotun bir hedefe kilitlendiğini, ateşlemeyi tetiklediğini ve hemen hemen aynı zamanda görev kontrol bilgisayarından pilotun ateşleme yetkisinin iptal eden bir mesaj alındığını düşünelim; bu durumda füze ateşlenmeli midir, ateşlenmemeli midir? Görüldüğü gibi bir önceki örnekte çok da önemli etkisi yokmuş gibi gözüken bir eksiklik, sistematik bir eksikliğe dönüşür ve sistemin geneline yayılırsa sonuçları emniyet açısından oldukça tehlikeli olabilmektedir.

3.2. Efor Karşılaştırması

Modellenen gereksinim kümesi daha önce iki uzman tarafından gözden geçirilmiştir. Gözden geçirme kayıtları incelendiğinde ilk uzmanın inceleme için bir saat harcadığı, ikincisinin ise bir buçuk saat kadar harcadığı görülmüştür. Gözden geçirme toplantısı ise yaklaşık yarım saat kadar sürmüştür. Gereksinimlerin yazarının da cevap süresi göz önüne alındığında harcanan toplam efor beş ile altı adam-saat olarak hesaplanabilir. Bu durumda ortalamada bir gereksinimi gözden geçirmek için 1.16 adam-saat gibi bir efor harcanmıştır. Öte yandan bu gereksinimlerin Alloy'da modellenip analiz edilebilmesi için yaklaşık 40 adam saatlik efor harcanmıştır. Bu durumda gereksinim başına düşen gözden geçirme eforu 5.71 adam-saat olarak ortaya çıkmıştır.

Elbette örnek uzay oldukça küçük olduğu için yukarıda verilen bu rakamları istatistiksel olarak incelemek ve metodun verimliğini istatistiksel olarak sorgulamak uygun değildir. Bu detayda yargılara varmadan önce aynı yöntemin kullanıldığı daha büyük kapsamlı projelerden gelecek verilere ihtiyaç vardır. Dolayısıyla eldeki verileri anekdot düzeyindeki kanıt olarak değerlendirmek en doğrusudur. Metot için ilk sonuçlar incelendiğinde modelleme işleminin klasik gözden geçirme yöntemlerine nazaran daha çok efor gerektirdiğini görmek mümkündür. Öte yandan yedi gereksinimlik oldukça yalın bir küme incelendiğinde bile tutarsızlıklar ve eksiklikler yakalanmış olması analizin gücünü ortaya koymaktadır.

4. Sonuçlar

Bu makalede emniyet kritik aviyonik yazılım gereksinimlerinin gözden geçirme aktivitesinin otomasyonu için bir yöntem ve bu yöntemin pilot bir uygulaması aktarılmıştır. Otomasyon için MIT tarafından geliştirilen Alloy Analyzer adı verilen bir araç kullanılmıştır.

Sonuçlar incelendiğinde basit bir gereksinim kümesi ele alınmasına rağmen sağduyu bilgisinin gizli kalması ve durum geçiş diyagramlarından gereksinimlere geçerken zamanlamanın bir faktör olarak ele alınmaması nedeniyle ortaya çıkan eksiklikler bulunmuştur.

İstatistiksel manada yeterli veriye sahip olunmamasına rağmen en azından ilk çalışmalar ışığında yöntemin geleneksel gözden geçirme yöntemlerine göre daha maliyetli olduğunu söylemek mümkündür. Metrikler incelendiğinde maliyet farkının doğal dilde yazılan gereksinimlerin modellenmesi esnasında yapılan işlerden kaynaklandığı görülmektedir. Unutulmaması gereken önemli bir nokta bu fazladan eforun kimin tarafından harcandığıdır. Masa başı gözden geçirmelerinde, incelemeler projenin diğer parçaları ile ilgilenen ekip tarafından yapılmakta dolayısıyla sürecin uzaması durumunda diğer aktiviteleri de etkilemektedir. Ancak aynı gözden geçirme Alloy Analyzer konusunda uzman, tek işi gereksinimleri modellemek olan bir ekip tarafından gerçekleştirildiğinde diğer aktivitelere asgari etki ile modelleme işlemi gerçekleştirilebilir. Ayrıca, gereksinimlerde küçük değişiklikler yapılması halinde, bu değişikliklerin eldeki Alloy modellerine aktarılmasının nispeten kolay olması beklenir. Sonuç olarak, bu ilk veriyi göz önünde bulundurarak analizin en azından şimdiki hali ile sadece en üst emniyet seviyesine sahip olan, küçük fakat girift gereksinim kümeleri için denenmesini önermek mantıklı olacaktır.

Eforu azaltmak için izlenebilecek bir başka çözüm yolu ise doğal dilde yazılan gereksinimlerin modellenme işinin otomasyonunun yapılmasıdır. Lee'ye göre CNLP (Contextual Natural Language Processing) ve İki Seviye Gramer (TLG-

Two-level grammar) yardımı ile doğal dilde yazılan gereksinimlerin formel spesifikasyonlara otomatik olarak çevrilmesi mümkündür [4]. Benzer bir çalışma doğal dilde yazılan gereksinimlerin Alloy diline otomatik olarak çevrilmesi konusunda yapılabilir. Gereksinim dokümanında yer alan durum geçiş diyagramı gibi notasyonların Alloy diline otomatik çevrilmesi de mümkün görünmektedir. Bu yöndeki ilerlemeler, efor konusundaki engeli aşağı çekecektir.

5. Kaynakça

1. **Vance Hilderman ve Tony Baghai.** *Aviation Certification: A Complate Guide to DO-178B (Software) DO-154 (Hardware).* Avionics Communications Inc., 2007.

2. **DO-178B.** *Software Considerations in Airborne Systems and Equipment Certification.* : RTCA, Inc., 1992.

3. **Jackson, Daniel.** *Software Abstractions.* : The MIT Press, 2006.

4. **Lee, Beum-Seuk.** *Automated Convertion from a Requirements Document to an Executable Formal Specification.* 2003 .