T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI SİSTEMLER İÇİN YÜKSEK GÜVENİRLİKLİ ÇOKLU YAZILIM SENTEZİ

Nadir SUBAŞI

DOKTORA TEZİ
Kontrol ve Otomasyon Mühendisliği Anabilim Dalı
Kontrol ve Otomasyon Mühendisliği Programı

Danışman
Dr.Öğr.Üyesi İlker ÜSTOĞLU

Şubat, 2020

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI SİSTEMLER İÇİN YÜKSEK GÜVENİRLİKLİ ÇOKLU YAZILIM SENTEZİ

Nadir SUBAŞI tarafından hazırlanan tez çalışması 12.02.2020 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol ve Otomasyon Mühendisliği Anabilim Dalı Kontrol ve Otomasyon Mühendisliği Programı **DOKTORA TEZİ** olarak kabul edilmiştir.

Dr.Öğr.Üyesi İlker ÜSTOĞLU
İstanbul Teknik Üniversitesi
Danışman

Jüri Üyeleri

Dr.Öğr.Üyesi İlker ÜSTOĞLU, Danışman
İstanbul Teknik Üniversitesi _____

Dr.Öğr.Üyesi Muharrem MERCİMEK, Üye
Yıldız Teknik Üniversitesi _____

Doç.Dr. Tufan KUMBASAR, Üye
İstanbul Teknik Üniversitesi _____

Prof.Dr. Şeref Naci ENGİN, Üye
Yıldız Teknik Üniversitesi _____

Dr.Öğr.Üyesi Yavuz EREN, Üye
Yıldız Teknik Üniversitesi _____

Danışmanım Dr.Öğr.Üyesi İlker ÜSTOĞLU sorumluluğunda tarafımca hazırlanan Gerçek Zamanlı Sistemler İçin Yüksek Güvenirlikli Çoklu Yazılım Sentezi başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Nadir SUBAŞI

İmza

Dedicated to My Family
and
NNS

TEŞEKKÜR

Tez çalışmamda benimle bilgi birkimini paylaşan, ileride mesleki hayatımda bana yön verecek tecrübelerini aktaran, motive eden danışman hocam Dr.Öğr.Üyesi İlker ÜSTOĞLU'na,

Tez izleme jürimde bulunan ve eleştirileriyle yol gösteren, bilgisini, tecrübesini ve yardımlarını hiçbir zaman esirgemeyen Sayın Doç.Dr. Tufan KUMBASAR'a ve Dr.Öğr.Üyesi Muharrem MERCİMEK'e,

Tez jürimde bulunmayı kabul eden ve değerli fikir ve önerilerinden faydalandığım Prof.Dr. Şeref Naci ENGİN ve Dr.Öğr.Üyesi Yavuz EREN'e,

Deneyimleri ve ufkumu açan düşünceleriyle çalışmamı zenginleştiren değerli arkadaşlarım Dr.Öğr.Üyesi Tarık V. MUMCU ve Arş.Gör. Ufuk GÜNER'e,

Doktora çalışmalarım süresince sürekli fedakarlık yapan, beni teşvik eden ve bana destek olan çok kıymetli eşime,

Zaman zaman bilgisayarımı karıştırarak, dokümanlarımı karalayarak ve çoğunlukla baba ne yapıyorsun diyerek desteklerini esirgemeyen 8 yaşındaki ikizlerim Necati ve Nejat'a,

Hayatımın her anında desteklerini benden esirgemeyen çok değerli aileme,

SONSUZ TEŞEKKÜRLER

Nadir SUBAŞI

İÇİNDEKİLER

SİMGE LİSTESİ

Ω	Açısal hız vektörü
τ_w	Ağırlık torku
v_{ij}	Ağırlıklandırılmış Normalleştirilmiş Karar Matrisi
r	Aksaklık sayısı
R	Anlaşma Matrisi
$F(t)$	Arıza ihtimali
$f(t)$	Arıza yoğunluğu
O_I	Atalet koordinatını
I	Atalet matrisi
λ	Başarısızlık oranı
x_0	Başlangıç durumları
$y(t)$	Çıkış Sinyali
l^B_{y1}	Denge noktasından uzaklığı (en)
l^B_{y2}	Denge noktasından uzaklığı (boy)
Y_i	Doğru çıkış sayısı
ξ	Dönme yer değiştirmesi
$R^{B\rightarrow I}$	Dönüşüm Matrisi
E	Eşdeğerlik matrisi
G^I	Eylemsizlik çerçevesine göre yerçekimi vektörü
τ_c	Eyleyici torku
α	Geri devrilme açısı
O_y	Geri devrilme sapma momenti
$u(t)$	Giriş sinyali

m	Gövde çerçevesinin kütlesi
T_1^B	Gövde çerçevesinin merkezindeki kuvvet (en)
T_2^B	Gövde çerçevesinin merkezindeki kuvvet (boy)
$R(t)$	Güvenirlik fonksiyonu
$z(t)$	Hata oranı
C_i^*	İdeal çözümün yakınlığı
PTI	Kanıt test aralığı
A_{ij}	Karar Matrisi
$\mathscr{L}_2$	$\mathscr{L}_2$ Uzayı
Q	Kompozisyon sonuç sayısı
W_{ij}	Kriter Ağırlıkları
$A(t)$	Elverişlilik
M	Maliyet
C	Mantıksal matris için kompozisyon
A^-	Negatif ideal çözüm
S_i^-	Negatif ideal çözümden uzaklık
N_{ij}	Normalleştirilmiş Karar Matrisi
β	Paralel devrilme açısı
O_p	Paralel devrilme, yatma torku
T_2	Pervaneler tarafından üretilen boy tork kuvveti
T_1	Pervaneler tarafından üretilen en tork kuvveti
A^*	Pozitif ideal çözüm
S_i^*	Pozitif ideal çözüme yakınlık
$R^{y_1 \to B}$	Rotorların ters eğim etkisi (en)
$R^{y_2 \to B}$	Rotorların ters eğim etkisi (boy)
l_h	Sabitleme noktasından uzaklık (boy)
l_v	Sabitleme noktasından uzaklık (en)
ψ	Sapma Açısı
$Fr(t)$	Tamir dağılım fonksiyonu

$fr(t)$	Tamir yoğunluğu
TZ	Tarama Zamanı
$T(t)$	Toplam işlem süre
$\vec{\tau}$	Toplam tork vektörü
$T(z)$	Transfer Fonksiyonu
V	Versiyon
O_B	Vücut çerçevesinin koordinatı
θ	Yunuslama Açısı
ϕ	Yatma Açısı

KISALTMA LİSTESİ

BLDC	Fırçasız Doğru Akım Motorları (Brushless Direct Current Motors)
CENELEC	Avrupa Elektroteknik Standardizasyon Komitesi (European Committee for Electrotechnical Standardization)
E/E/PE	Elektrik / Elektronik / Programlanabilir Elektronik (Electrical / Electronic / Programmable Electronic)
FCV	Biçimlendirilmiş oy birliği seçici (The formalized consensus voting algorithm)
FMV	Biçimlendirilmiş çoğunluk seçici (The formalized majority voting algorithm)
FSA	Fonksiyonel Emniyet Değerlendirmesi (Functional Safety Assessment)
HFT	Ekipman Hatası Toleransı (Hardware Fault Tolerance)
IAR	İsveçli bir bilgisayar yazılımı (Ingenjörsfirman Anders Rundgren)
IEC	Uluslararası Elektroteknik Komisyonu (The International Electrotechnical Commission)
IEEE	Elektrik Elektronik Mühendisleri Enstitüsü (The Institute of Electrical and Electronics Engineers)
İHA	İnsansız hava aracı (Unmanned aerial vehicle)
KCG	Kalifiye Kod Üretici (Qualified Code Generator)
MCDM	Çok kriterli karar verme (Multiple Criteria Decision-making)
MISRA	Motor Sanayi Yazılım Güvenilirlik Derneği (Motor Industry Software Reliability Association)
MinCV	Minimizasyonlu oy birliği seçici (The consensus voting algorithm with minimization)
MinMV	Minimizasyonlu mutlak çoğunluk seçici (The majority voting algorithm with minimization)
MLV	Maksimum olasılık seçici (Maximum likelihood voting)

MTBF	Başarısızlıklar arasındaki ortalama süre (Mean Time Between Failures)
MTTF	Ortalama başarısızlık süresi (Mean Time To Failure)
MTTR	Ortalama tamir süresi (Mean Time To Repair)
NVP	N-versiyon programlama (N-version programming)
NVP-CV	NVP Oy birliği seçici (NVP Consensus Voter)
NVP-MV	NVP çoğunluk seçici (NVP Majority Voter)
PFD	Talep Üzerine Başarısızlık Olasılığı (Probability of Failure on Demand)
PID	Oransal integral türev (Proportional integral derivative)
RRD	Risk Azaltma Oranı (Risk reduction definition)
RTTI	Çalışma zamanı türü tanımlaması (Run-time type information)
SCADE	Emniyet Kritik Uygulama Geliştirme Ortamı (Safety Critical Application Development Enviroment)
SFF	Emniyet Hata Oranı (Safe Failure Fraction)
SIF	Emniyet Donanımlı Fonksiyon (Safety Instrumented Functions)
SIL	Emniyet Bütünlük Seviyesi (Safety Integrity Level)
SIS	Emniyet Donanımlı Sistem (Safety Instrumented System)
TOPSIS	İdeal çözümlere göre benzerlik yoluyla tercihi tekniği (Technique for Order Preference by Similarity to Ideal Solution)
UML	Birleşik Modelleme Dili (Unified Modelling Language)

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Gerçek Zamanlı Sistemler İçin Yüksek Güvenirlikli Çoklu Yazılım Sentezi

Nadir SUBAŞI

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Dr.Öğr.Üyesi İlker ÜSTOĞLU

Teknolojik gelişmeler ile birlikte, emniyet tedbirlerine duyulan ihtiyaç insan hayatının önemli bir parçası haline gelmiştir. Emniyet-kritiklik günlük yaşantıda sıklıkla karşılaşılan bir kavramdır. Bu çalışmada, emniyet-kritiklik hususunda sektördeki birçok standardın taban olarak kullandığı IEC61508 standardı ile bu standardın önerdiği yöntemler ve araçlar incelenmiştir.

Bölüm 1'de literatür taraması yapılarak tezin amacından ve literatürde görülen açık noktalardan bahsedilmiştir. Emniyet-kritik sistemler tanıtılarak IEC61508 standardının önemini vurgulayan literatür bilgileri değerlendirilmiştir.

Bölüm 2'de IEC61508 stardardının genel yapısı anlatılmış ve bu standarda ihtiyaç duyulma sebebi hakkında bilgi verilmiştir. Güvenlik ve emniyet kavramları üzerinde durulmuştur. Emniyet taktikleri incelenmiş ve standart ile ilişkilerinden söz edilmiştir. IEC61508 standardının 3. bölümü olan yazılım bölümü incelenmiştir. Emniyet-kritik bir uygulama içerisinde, IEC61508-3 bölümünde önerilen yöntemler ile, C/C++ kullanılarak oluşturulan kodlar, V-model metoduna göre PC-lint programı aracılığıyla analiz edilmiştir. IEC61508-3 standart parçasının önerilerinden olan senkron ve asenkron programlamalar hakkında bilgiler verilmiştir. Ayrıca standardın uygulanmasını önerdiği yöntemler benimsenerek yazılmış bir programlama dili olan SCADE Suit incelenmiştir.

Bölüm 3'de, IEC61508-3 standardında önerilen yöntemlerden biri olan NVP tekniği açıklanmış ve çeşitleri hakkında bilgi verilmiştir. n sayıda farklı versiyondan gelen

verileri analiz eden ve bu konuda karar veren, ayrıca NVP'nin başarısını belirleyen NVP seçicileri hakkında bilgi verilmiştir. SCADE programlama dili içerisinde örnek bir NVP uygulaması tasarlanmış ve davranışları incelenmiştir.

Bölüm 4'de, uygulama sahasında karşılaşılan sorunlardan biri olan, n versiyon sayısını oluştururken hangi n tanesinin seçilmesi gerektiği sorununa çözüm aranmıştır. MCDM yöntemlerinden biri olan TOPSIS yöntemi ile örnek bir veri seti üzerinde uygulama yapılmıştır.

Bölüm 5'de gerçek-zamanlı bir test ortamı oluşturulmuştur. Bu test ortamı, tilt-rotor ile dengede durması kontrol edilen bir yapıdadır. Matlab ortamında simule edilen kontrolcü, bu test ortamında gerçeklenmiştir. Ana motivasyon kaynağımız olan NVP-MV ile çoğunluk kararının sistemi kararsızlığa götürmesinin önüne nasıl geçilebilir sorusuna cevap aranmıştır. Geliştirilen algoritma ile bunun önüne geçilmiştir. Test senaryosunun sonuçları yorumlanmıştır.

Son olarak, Sonuç ve Öneriler bölümünde, çalışma boyunca elde edilen sonuçlar sunulmuştur. IEC61508 ve IEC61508-3 standartları açıklanmıştır. SCADE programlama ortamında NVP yöntemi ile tasarlanan emniyet-kritik uygulama sonuçları verilmiştir. TOPSIS algoritması kullanılarak hangi n versiyonunun seçileceği, tartışılmıştır. Son olarak ise gerçek-zamanlı bir uygulama üzerinde NVP-MV seçicisine göre daha yetenekli bir tasarım açıklanmıştır.

Anahtar Kelimeler: Gerçek-zamanlı sistemler, yüksek güvenirlik, çoklu yazılım sentezi, çok ölçütlü karar verme, fonksiyonel emniyet

ABSTRACT

The High-Reliability Multi-Version Software Synthesis For Real Time Systems

Nadir SUBAŞI

Department of Control and Automation Engineering
Doctor of Philosophy Thesis

Advisor: Assist.Prof.Dr. İlker ÜSTOGLU

With technological developments, the need for safety measures has become an important part of human life. Safety-criticality is a frequently encountered concept in daily life. In this study, the IEC61508 standard used by many other standards in the sector in terms of safety-critical issues and the methods and tools proposed by this standard are examined.

In Chapter 1, the literature review is done and the purpose of the thesis and the open points in the literature are mentioned. By introducing safety-critical systems, literature information emphasizing the importance of the IEC61508 standard was evaluated.

In Chapter 2, the general structure of the IEC61508 standard is explained and information is given about the reason why this standard is needed. Security and safety concepts were emphasized. Safety tactics were examined and their relationship with the standard was mentioned. The software part, which is the 3rd part of the IEC61508 standard, has been examined. In a safety-critical application, codes created using C/C++ were analyzed using the PC-lint program according to the V-model method using the methods proposed in the IEC61508-3 part. Information on synchronous and asynchronous programming, which is one of the recommendations of IEC61508-3 standard part, is given. In addition, SCADE Suit, a programming language written by adopting the methods recommended by the standard, was examined.

In Chapter 3, NVP technique, which is one of the methods recommended in the IEC61508-3 standard, is explained and information about its types is given.

Information was given about NVP selectors, which analyzed and decided on data from n different versions and also determined the success of NVP. A sample NVP application is designed in SCADE programming language and its behaviors are examined.

In Chapter 4, a solution was sought for one of the problems encountered in the field of application, which n to choose when creating the number of n versions. The application was made on a sample dataset with the TOPSIS method, which is one of the MCDM methods.

A real-time test environment has been created in Chapter 5. This test environment is in a structure that is controlled to balance with the tilt-rotor. The controller simulated in Matlab environment was implemented in this test environment. With our main motivation source, NVP-MV, an answer was sought to the question of how to prevent the majority decision from bringing the system to instability. This has been prevented with the developed algorithm. The results of the test scenario are interpreted.

Finally, the results obtained during the study are presented in the Results and Recommendations section. IEC61508 and IEC61508-3 standards are described. Safety-critical application results designed by NVP method in SCADE programming environment are given. Which n version to choose using the TOPSIS algorithm is discussed. Finally, a more capable design is explained over a real-time application than the NVP-MV selector.

Keywords: Real-time systems, high-reliability, n-version software synthesis, multi-criteria decision-making, functional safety

1
GİRİŞ

Günümüz teknolojilerinin gelişimi, beraberinde teknolojiyi güvenle kullanabilme ihtiyacını doğurmaktadır. Bunun farkında olan sistem mühendisleri, sistemlerini kurma ve güncelleme aşamasında sertifikasyonu olan ve onaylı ürünler seçmeye çalışmaktadırlar. Yazılım ve sistem aksaklıkları sistem çökmelerine neden olabilmektedir. Özellikle teknolojinin bize sağladığı avantajları düşündüğümüzde sistem çökmeleri çevre hasarları veya insan hayatlarına mal olabilmektedir. Bizlere bu sertifikasyon ihtiyacı Avrupa Elektroteknik Standardizasyon Komitesi (CENELEC) ve Uluslararası Elektroteknik Komisyonu (IEC), elektrikli, elektronik ve programlanabilir elektronik cihazlar tarafından üretilen koruma fonksiyonları dahil olmak üzere tüm uygulamalar için uluslararası bir standart oluşturmuştur. IEC 61508 standardı bu ihtiyaçtan doğmuştur. Üreticiler bu pazar talebine cevap verebilmek için bir yandan üretmeye çalışırken bir yandan da bu sertifikasyonu almaya çalışmaktadırlar. Bu standartlar o kadar karmaşık ve anlaşılması zordur ki işletme veya ürünün bu standarda uygunluğunu araştıran sertifikasyon firmalarının oluşmasına sebep olmuştur.

Standardın Bölüm 0'ı teknik rapor statüsündedir ve tamamen bilgilendiricidir. IEC standartlarında normatif bir gereklilik "zorunluluk" ile karşı karşıya kalır ve bu gereklilik belirli bir uygulamada geçerliyse, tanımlı gerekliliklere uyulmalıdır. "Olması gereken" ile bildirilen bir gereklilik bilgilendiricidir ve bir öneri olarak kabul edilebilmektedir, ancak standarttaki ilgili gerekliliklere uygunluk açısından normal değildir.

Standardın Bölüm 1, 2, 3'te tüm örnek oluşturan gereksinimler ve bilgilendirici detayları yer almaktadır. Bölüm 0, 5, 6 ve 7 numaralı bölümler örnek oluşturan gereklilikler içermemektedir. IEC 61508'in 1, 2, 3 ve 4 numaralı bölümleri IEC temel emniyet yayınlarıdır.

Bu sertifikasyon donanım cephesinde olduğu gibi yazılım cephesini de kapsamaktadır. IEC 61508 standardının 3. Bölümü yazılım güvenirliği ile ilgili bölümdür. Bu bölüm

örneklerden daha çok hangi yöntemlerin tavsiye edildiği bilgilerini barındırmaktadır.

Bu sertifikasyona uygun sistemlerin çökmemesi ve verdiği hizmette kesinti olmaması, çevreye veya insan hayatına zarar vermemesi şeklinde mükemmel sistemi tarif etmektedir. Ama bunun karşılığında sistemlerin oldukça maliyetli olması ve tasarlanmaları normalden çok daha zor olacaktır.

IEC61508 standardı içerisindeki eklerin incelendiği zaman senkron/asenkron programlama, NVP(N-versiyon programlama), V-model konuları öne çıkmaktadır. Uygulama tarafında ise maaliyet unsuru karşımıza çıkmaktadır.

1.1 Literatür Özeti

Standartlar her geçen gün önemini arttırmaktadır. Her gün üretilen ürünlerin emniyet ihtiyacı, can ve mal emniyeti ihtiyacının artması sonucu bu önem değerini daha da arttırmıştır.

IEEE, emniyet-kritik yazılımın tanımını "kabul edilmez riskler ile sonuçlanan sistemlere kullanılan yazılım olarak tanımlanır. Emniyet kritik yazılım çalışması ya da çalışmaması tehlikeli olacak durumlar için tehlikeli durumdan kurtarmaya ve kazanın ciddiyetini azaltmaya yönelik yazılımları içerir." [1] şeklinde yapmaktadır.

Avrupa Elektroteknik Standardizasyon Komitesi (CENELEC) ve Uluslararası Elektroteknik Komisyonu (IEC) elektrikli, elektronik ve programlanabilir elektronik cihazlar (E/E/PE) tarafından üretilen koruma fonksiyonları dahil olmak üzere tüm uygulamalar için uluslararası bir standart oluşturmuştur. IEC 61508 standardı bu ihtiyaçtan doğmuştur. IEC 61508 standardı, gerekli SIL(Emniyet Bütünlük Seviyesi) seviyesine ulaşmak için bir sistemin tasarımı, entegrasyonu, işletimi ve bakımı için gereksinimleri belirlemektedir [2].

Gelişen teknolojide artan emniyet-kritik yazılımın miktarına bakmak için, aviyonik sistemleri örnek alırsak, 1987'de piyasaya sürülen Airbus A320, yaklaşık 800.000 kod satırına [3] ve 1994'te piyasaya sürülen Boeing 777 yaklaşık 4 milyon kod satırına sahipti. 2009 yılında piyasaya sunulan Boeing 787, yaklaşık 7 milyon kod satırına sahiptir. Aviyonik sektöründe, sistemlerinde kullanılan yazılım miktarının her 10 yılda bir ikiye katlandığı anlaşılmaktadır [4].

Sağlık sektöründe, emniyet-kritik sistemlerin önemi ise Therac-25 olayı ile karşımıza çıkmaktadır. Radyasyon terapisinde kullanılan Therac-25 cihazı, içerisindeki bir yazılım hatası nedeniyle radyasyon tedavisi sırasında aşırı dozdan dolayı dört kişinin ölümüne sebep olmuştur [5]. Yapılan araştırma sonucu, kazaya neden

olan Therac-25 cihazı içerisinde yazılım geliştirilirken yazılım geliştirmenin temel ilkelerinin izlenmediği, testlerin yetersiz olduğu, gerekli belgelerin hazırlanamadığı ve kalite güvence işlemlerinin yapılmadığı tespit edilmiştir [6].

Geliştiricinin temel görevi, standartların gerekliliklerini anlamak ve bunları yazılıma çevirmektir. Standartlar karışık ve yanlış anlaşılmaya uygun metinler içermektedir. Yanlış anlamalar ve şartların yanlış uygulanması, sistematik başarısızlığın ana nedenlerinden biri olarak kabul edilir. Bu problemi çözmek ve standartlarına uygun bir yazılım oluşturmak için çeşitli kalite modelleri önerilmiştir [7]. Ayrıca literatür içerisinde bazı genelleştirmeler ile Emniyet Taktik Katalogları oluşturulmuştur [8].

Bir yazılım ürününü izlemek ve kontrol etmek için yazılım kalite kriterlerinin tanımı ve kullanımı bazı araştırmalarda ele alınmıştır [9, 10].

Standartta sunulan sistem hatalarının analizi ve mekanik sistemler için güvenirlik, ne yazık ki, karmaşık yapısı nedeniyle yazılım için geçerli değildir. Mekanik sistemlerin aksine, yazılım fizik yasalarına uygun olarak hareket etmez, bu nedenle yazılımın arızalanabileceği koşullar tahmin edilememektedir. Yazılım güvenirliğini, test etme veya güvenirlik analizi yoluyla belirlemek mümkün olmadığından, tasarım kalite güvencesi konsepti tanıtılmıştır [11]. Tasarım garantisi, gereksinimler, tasarım ve geliştirme sırasında meydana gelebilecek hataların kabul edilebilir bir sistem emniyet düzeyinde tutulmasını sağlamak için yapılan planlı ve sistematik eylemleri ifade etmektedir [12] Tasarım desteği yaklaşımı, yazılım geliştirme sırasında daha titiz gerçekleştirildiği varsayılırsa, yazılım daha az hata içermektedir [13]. Bu yaklaşımla, yazılımın kritiklik seviyesi arttıkça, uygulanan geliştirme ve doğrulama işlemleri miktar ve kalite açısından artmaktadır.

Standartta kendini kanıtlamış sertifikalı yazılım araçları kullanmayı önermektedir. Ham bir komut dosyasının (.cpp veya .h) programlama dilini makine diline dönüştürmek için IAR derleyici kullanılabilir [14].

MISRA (Motor Industry Reliability Association)' nın standartlar için kurallar rehberlik yapmaktadır. Yazılan emniyet-kritik kodların bu kural rehberine uygunluğu için ayrı bir yazılım gerekmektedir [15]. "PC-Lint for C/C++" bu yazılımlardan bir tanesidir [16].

Emniyet-kritik kodların C++ ile emniyet-kritik uygulama geliştirilmesi gerektiğinde; emniyet riskleri sebebiyle kalıplar, çoklu kalıtım, yeniden yorumlama, sürekli yönetim istisnaları ve çalıştırma zaman türü bilgisi(RTTI) gibi bazı C++ programlama dili işlevlerini PC-lint içerisinde devre dışı bırakılmalıdır [17].

Aviyonik sektöründeki IEC-61508 standardının referans ettiği DO-172B/C standardının kılavuzu emniyet-kritik aviyonik yazılım sistemlerinin emniyet kriterlerini belirleyen bir kodlama standardıdır. Kod okunabilirliği, sürdürülebilirliği, fonksiyon tasarımı, makroların kullanımı, lokal ve global veri kullanımı, düşük seviye gereksinimler ile kod arasındaki doküman izlenebilirliği, modül/fonksiyon/prosedür adlandırma ve şartlı olarak derlenmiş kod için birtakım sınırlandırmalar bu standardın dahilindedir ve kullanıcıya kılavuzluk yapması adına tanımlanmışlardır [11].

Donanımsal sorunları tasarım aşamasında UML diyagramları yarı resmi bir yöntem olarak tekil bir model olarak tanımlayarak çözülebilmektedir [18].

Gerçek-zamanlı emniyet-kritik sistemlerde yaşanan sorunların üstesinden gelebilmek için Esterel Firması SCADE (Safety Critical Application Development Environment) gerçek-zamanlı sistemler ve hayati tehlike arz eden gömülü yazılım uygulamalarının tasarımı için kullanılan bir tümleşik geliştirme ortamı geliştirmiştir [19].

Yazılımın yıkılmasını engellemek için gerekli önlemler alınarak, güvenirlik (reliability) artırılmaya çalışılmaktadır. Tüm çabalara rağmen, yazılımın giderilemeyen bazı aksaklıkları (faults) içermesi olasıdır. Belirli koşullar altında yazılım işletilirken, içerdiği aksaklıklar hataya (error) ve yıkılmaya dönüşmektedir. Güvenirliğin yüksek düzeyde olması amaçlanan yazılım sistemlerinde özellikle, emniyet-kritik (safety-critical) uygulamalarda, hataların neden olabileceği yıkılmaları önlemek için NVP (N-versiyon programlama), kurtarma bloğu (recovery block), hataya dayanıklı yazılım (fault-tolerant software) gibi yaklaşımlar önerilmektedir [20].

NVP, yazılım mühendisliğindeki, işlevsel olarak eşdeğer programların aynı başlangıç değerinden bağımsız olarak üretildiği bir yöntem veya işlemdir [21]. NVP kavramı, [22]' de programlama zorluğunun bağımsızlığı, programın iki veya daha fazla versiyonunda ortaya çıkan aynı yazılım hatalarının olasılığını önemli ölçüde azaltacağı varsayımı üzerine getirilmiştir. NVP günümüzde havacılık, ulaşım gibi birçok alandaki yazılımlara uygulanmıştır [22].

NVP fikrine göre, N farklı çalışma grubu tarafından N farklı yazılım versiyonu (veya modülü) geliştirilmelidir. Bu versiyonların giriş çıkış özellikleri aynıdır ve işlevsel olarak eşdeğerdir, yani aynı görevi çözerler. Bununla birlikte, çalışma gruplarına bağlı olarak, görev çözme yöntemleri farklı olabilmektedir.

NVP potansiyel uygulama alanları, nükleer sanayi, su altı ve yer altı araştırmaları, kimya endüstrisi, demiryolu kilitleme sistemleri, elektronik oylama, ileti gönderme sistemleri, web hizmetleri, intihal tespiti ve sıfır gün istismarların tespiti gibi sayılabilmektedir [20, 23–30]. Daha önce de belirtildiği gibi N versiyon programlama

yöntemi, aynı yazılım modülünün çeşitli versiyonlarının bağımsız ve paralel yürütülmesini içermektedir. Modülün tüm versiyonları aynı veriyi alır [31]. Tasarım çeşitliliği nedeniyle versiyonların farklı hesaplama sonuçları üreteceği ihtimali olduğundan, doğru ve yanlış sonuçların belirlenmesinde ortaya çıkan bir diğer sorun hangisinin geçerli olacağıdır.

Literatürde NVP tekniğinin başarılı uygulamaları arasında uzay [32, 33], demiryolu sinyalizasyon sistemleri [23], mesaj iletim sistemleri [24], e-oylama [34], intihal tespit algoritmaları [25] ve ağ hizmetleri [26, 29] yer almaktadır.

Ayrıca NVP tekniğindeki yazılım gereksinimleri literatürde [35–38] anlatılmıştır. NVP yaklaşımına göre, versiyonlar çeşitlilik şartına uymalıdır [36, 37, 39]. İdeal olarak, versiyonlar farklı programcılar tarafından farklı çerçevelerde farklı programlama dilleri ile geliştirilmelidir.

NVP yöntemi, işlevsel olarak eşdeğer modüllerde hataların oluşmasının çeşitli noktalarda gerçekleşebileceğini önermektedir; bu nedenle hatalar tespit edilebilir ve gerçek sonuç alınabilir [40]. Yazılım versiyonlarının sayısı ne kadar fazla olursa, N-versiyon yazılımının doğru şekilde çalıştırılma ihtimali de artmaktadır.

NVP yöntemin en büyük avantajı yazılım hata toleransını sağlamaktır [31]. Herhangi bir versiyonun başarısız olması durumunda bile, çalışan versiyonların geri kalanı sonuçlarını üretecek ve sistem çalışmaya devam edecektir. Bu durumda sistemin olağan işleyişi, yazılım geliştirme sırasında üretilen ve test aşamasında tanımlanmayan hatalara karşı sigortalı hale gelmektedir [41, 42].

NVP yöntemi, versiyonlardan birinin hatalarının sistemin çalışmasını bozmamasına ve güvenirliğin sıkı gerekliliklere sahip olmasını sağlar. Böylelikle, bu yaklaşım aktif olarak mühendislik uygulamalarında kullanılmakta ve yaklaşımın yazılım kontrol sistemlerinin ve bilgi işlemenin geliştirilmesinde yararlı olduğu kanıtlanmıştır. Mevcut test yöntemleri ve program doğruluğunun ispatlanması ile birlikte NVP yöntemini kullanması yüksek seviyede yazılım güvenirliğini garanti etmektedir [41, 43, 44].

Seçici şemaları ve orijinal verilerin gereklilikleri bakımından farklı olan seçiciler için pek çok algoritma bulunmaktadır [45–50]. NVP yazılımında değerlendirme ve doğru sonuç seçimi için çeşitli oylama algoritmaları kullanılmaktadır [35, 44, 46, 51–61]. Her bir algoritmanın kendi avantajları ve dezavantajları bulunmaktadır ve başarılı bir şekilde uygulanabilmesi için algoritma belirli olmalıdır. Ayrıca, sürümlerin verdiği veri setine bağlı olarak bazı algoritmalar etkisiz olabilir veya çıktıların doğruluğu hakkında bir karar verme fırsatı bulunmayabilir. Seçici algoritmalarının tümünde ortak özellikler içeren bir dizi algoritma vardır.

Standartlara ve çok modüllü bir NVP tasarımının uygulama tarafında emniyet-kritik sistem ihtiyacı ile beliren maliyet kaygısı ortaya çıkmaktadır. Bu sorun ile başa çıkmak için modern karar verme yöntemlerinden faydalanılmaktadır.

MCDM (çok kriterli karar verme) yöntemlerinden sıklıkla kullanılan yöntemlerin başında TOPSIS, MAUT, CP ve VIKOR gibi yöntemler bulunmaktadır [62].

Bu yöntemler objektif bir performans değerlendirmesi için birçok ölçüt dikkate alınarak karar almaya yardımcı olmaktırlar. TOPSIS yöntemi, MCDM' de yaygın kullanımından dolayı literatürde geniş çapta yer almaktadır [63].

MCDM yöntemlerinden biri olan TOPSIS'in temel mantığı, pozitif bir ideal çözüm ve negatif bir ideal çözüm elde ederek, ideal çözüme dayalı alternatiflerin sıralanmasına dayanmaktadır. İdeal çözüme nispeten yakın alternatiflerden başlayarak, bir dizi oluşturulmaktadır [64, 65].

Gerçek-zamanlı ve emniyet-kritik bir tilt-rotor sistem üzerinde uygulama düşünüldüğünde; tilt-rotor sistemler çerçevesi platformun sınırlamasına göre üç bağımsız eksen etrafında serbestçe döndürülebilmektedir. Bu nedenle, sistem üç serbestlik derecesine sahiptir. Sistem; yatma, yunuslama ve sapma eksenleri üzerinde hareketlidir. Sistemin matematiksel modeli doğrusal değildir. Sistemi kontrol etmek için PID kontrolcüsü seçilmiştir. PID kontrolcüsünün herhangi bir donanım arızası, sistemin performansını ve hatta kararlılığını etkileyecek istenmeyen kontrol sinyallerine neden olmaktadır.

Bu tarz sistemler için Feron' un önerisi [66] ile emniyet-kritik bir kontrolcü elde etmek için çevrimiçi bir Lyapunov stabilite analiz özelliğinin mimariye eklenmesi düşünülmüştür.

1.2 Tezin Amacı

Tez çalışmamızda özellikle IEC61508 standardının anlaşılması ve gereksinim analizlerinin daha açık bir şekilde aktarılması amaçlanmıştır. Standartta önerilen yöntemler ile emniyet-kritik kod geliştirme üzerine C/C++ ve SCADE ortamlarında çalışmaları açıklanabilmektedir. NVP versiyon seçiminde TOPSIS algoritması ile emniyet-kritik sistemler için en iyi versiyonların nasıl seçileceği elde edilmiştir. Gerçek-zamanlı emniyet-kritik bir sistem için kararsızlık dedektörü geliştirilerek, NVP-MV(NVP Majority Voter) çoğunluğun sistemi kararsızlığa götürme ihtimalinin ortadan kaldırılması sağlanmıştır.

1.3 Bulgular

Çalışmamızda özellikle IEC 61508 standardının daha anlaşılır ve uygulanabilir kodlar ile aktarılması gerçekleştirilmiştir. Emniyet-kritik sistemlerde kullanılan yöntemler, kurallar ve kullanılması önerilen araçlar ile emniyet-kritik kod geliştirme üzerine C/C++ ve SCADE ortamlarında standarda uygun kodlar yazılmıştır. Kodlar, emniyet-kritik bir sisteme uygulanmıştır. NVP versiyon seçiminde TOPSIS algoritması ile emniyet-kritik sistemler için en iyi versiyonlar seçtirilmiştir. Gerçek-zamanlı emniyet-kritik bir sistem için kararsızlık dedektörü geliştirilmiş, NVP-MV(NVP Majority Voter)'a yeni bir yaklaşım geliştirilmiştir. Çoğunluğun sistemi hataya veya yıkılmaya sürüklediği durumlarda azınlığı seçebilen bir seçici tasarım gerçekleştirilmiştir.

2
IEC 61508-3 STANDARDI

Günümüz teknolojileri, elektronik ve bu elektronik parçaları yöneten yazılımlar çerçevesinde inşa edilmiştir. Elektronik ve yazılımsal yıkılmalar(failure) yaşam-kritik(life-critical) olmayan uygulamalarda sorun teşkil etmezken, yaşam-kritik uygulamalarda örneğin yol taşıtları, demir yolu uygulamaları, endüstri uygulamaları, nükleer güç santralleri gibi yazılımsal yıkılmaların sonuçları trajediye dönüşmektedir.

Yazılımın yıkılmasını engellemek için gerekli önlemler alınarak, güvenirlik (reliability) artırılmaya çalışılır. Tüm çabalara rağmen, yazılımın giderilemeyen bazı aksaklıkları (faults) içermesi olasıdır. Belirli koşullar altında yazılım işletilirken, içerdiği aksaklıklar hataya (error) ve yıkılmaya dönüşecektir. Güvenirliğin yüksek düzeyde olması amaçlanan yazılım sistemlerinde, özellikle, emniyet-kritik (safety-critical) uygulamalarda, hataların neden olabileceği yıkılmaları önlemek için N-versiyon programlama (N-version programming), kurtarma bloğu (Recovery block), hataya dayanıklı yazılım (fault-tolerant software) sunmak için önerilen yaklaşımlardır [20].

Bu soruna karşı Avrupa Elektroteknik Standardizasyon Komitesi (CENELEC) ve Uluslararası Elektroteknik Komisyonu (IEC), elektrikli, elektronik ve programlanabilir elektronik cihazlar tarafından üretilen koruma fonksiyonları dahil olmak üzere tüm uygulamalar için uluslararası bir standart oluşturmuştur.

IEC, 1985 yılında, emniyet uygulamaları için kullanılacak ile programlanabilir elektronik sistemler için genel bir standart geliştirme uygulanabilirliğini değerlendirmek üzere bir görev grubu kurmuştur. Emniyetle ilgili yazılımlarla başa çıkmak için daha önce bir çalışma grubu kurulmuştur. Bu iki çalışma grubu, IEC 61508 (IEC 2000) olacak uluslararası bir standardın geliştirilmesinde iş birliği yapmıştır. Görev grubunun (emniyet uygulamaları için kullanılan programlanabilir elektronik sistemler) orijinal kapsamı, her türlü elektro-teknik tabanlı teknolojiyi (elektrik, elektronik ve programlanabilir elektronik sistemler (E/E/PE sistemleri)) içerecek şekilde genişletilmiştir.

1998-2000 döneminde IEC 61508'in 1 ile 7 bölümleri yayınlanmıştır. 2005 yılında IEC / TR 61508-0 (IEC 2005) yayınlanmıştır. Standardı güncellemek ve iyileştirmek için bir inceleme süreci 2002 yılında başlatılmış ve Nisan 2010'da IEC 61508 Sürüm 2'nin (IEC 2010a) yayınlanmasıyla tamamlanmıştır.

2.1 IEC 61508 Yapısı

IEC 61508'in genel başlığı "Elektriksel, elektronik ve programlanabilir elektronik (E/E/PE) emniyetle ilgili sistemlerin işlevsel güvenliği" şeklinde tanımlanmıştır. Bölümler Tablo 2.1'de gösterilmiştir.

Tablo 2.1 IEC 61508 Standardının Bölümleri

Bölüm	**Başlık**
0	Fonksiyonel emniyet ve IEC 61508
1	Genel Gereksinimler
2	(E/E/PE) emniyetle ilgili sistemler için gereklilikler
3	Yazılım gereksinimleri
4	Tanımlar ve kısaltmalar
5	Emniyet bütünlüğü seviyelerinin belirlenmesi için yöntem örnekleri
6	IEC 61508-2 ve IEC 61508-3 uygulanmasına ilişkin rehber
7	Tekniklere ve ölçümlere genel bakış

Standardın Bölüm 0'ı teknik rapor statüsündedir ve tamamen bilgilendiricidir. IEC standartlarında normatif bir gereklilik "zorunluluk" ile karşı karşıya kalır ve bu gereklilik belirli bir uygulamada geçerliyse, tanımlı gerekliliklere uyulmalıdır. "Olması gereken" ile bildirilen bir gereklilik bilgilendiricidir ve bir öneri olarak kabul edilebilir, ancak standarttaki ilgili gerekliliklere uygunluk açısından normal değildir.

Bölüm 1, 2, 3'te tüm normatif gereksinimler ve bazı bilgilendirici gereksinimler yer almaktadır. Bölüm 0, 5, 6 ve 7 numaralı bölümler normatif gereklilikler içermemektedir. IEC 61508'in 1, 2, 3 ve 4 numaralı bölümleri IEC temel emniyet yayınlarıdır.

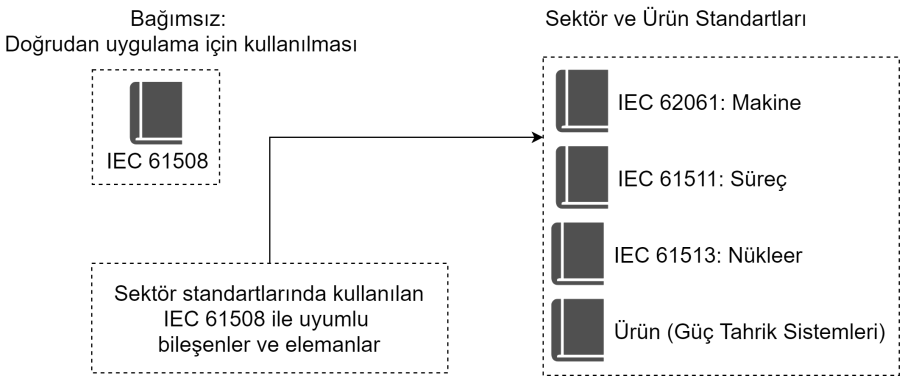
IEC teknik komitelerinin sorumluluklarından biri, uygulanabilir olduğu her yerde, IEC 61508'i, temel bir yayın rolü olarak kendi sektöründe veya E/E/PE emniyetle ilgili sistemleri olan kendi sektörlerinin veya ürün standartlarının hazırlanmasında IEC kapsamını kullanmaktır.

IEC 61508 hem bağımsız bir standarttır hem de sektör ve ürün standartlarının temeli olarak kullanılabilmektedir. Bir diğer rolü de süreç, nükleer, demiryolu endüstrileri, makine ve güç tahrik sistemleri için standartlar geliştirmek için kullanılmıştır. Tüm

sektörlerde E/E/PE emniyetle ilgili sistemlerin ve ürünlerin gelişimini etkilemiş ve etkilemeye devam edecektir. Bu konsept Şekil 2.1'de gösterilmektedir.

IEC 61508'in bağımsız bir standart olarak uygulanması, standardın kullanımını içerir:

- Uygulama sektörü veya ürün standartlarının bulunmadığı veya uygun olmadığı durumlarda, E/E/PE emniyetle ilgili sistemler için bir genel gereksinimler kümesi olarak,
- Tüm sektörlerde kullanım için E/E/PE bileşenleri ve alt sistemleri tedarikçileri tarafından (örneğin, sensörlerin donanım ve yazılımı, akıllı aktüatörler, programlanabilir kontrolcüler),
- E/E/PE emniyetle ilgili sistemler için kullanıcı şartnamelerini karşılamak üzere sistem entegratörleri tarafından,
- Kullanıcılar tarafından, bu emniyet fonksiyonlarının performans gereklilikleri ile birlikte gerçekleştirilecek emniyet fonksiyonları açısından şartları belirleme,
- E/E/PE emniyetle ilgili sistemlerin "tasarlandığı gibi" emniyet bütünlüğünün korunmasını sağlamak,
- Emniyet yaşam döngüsü faaliyetlerinin değerlendirmesini yapmak için uygunluk değerlendirme ve belgelendirme hizmetleri için teknik çerçeveyi sağlamaktır.



Şekil 2.1 Bağımsız ve sektör / ürün standartları

IEC 61508'e dayalı ürün veya uygulama sektörü uluslararası standartları:

- sistem tasarımcılarına, sistem entegratörlerine ve kullanıcılara yöneliktir,
- sektöre özgü uygulamaları dikkate almak,

- amaçlanan kullanıcıları için anlayışı artırmak için sektörde geçerli olan terminolojiyi kullanmak,
- sektöre uygun belirli kısıtlamalar belirleyebilir,
- genellikle, alt sistemlerin tasarımı için IEC 61508'in gereksinimlerine dayanır.

IEC 61508 türetilen standartlar, örneğin, endüstriyel prosesler için standartları (61511 IEC), nükleer sektör (IEC61513), makine güvenliği (IEC 62061 ve ISO 13849) ya da demiryolu endüstrisidir(EN 50126 / EN 50128 / EN 50129).

2.2 IEC 61508'in Kapsamı

IEC 61508 başarısızlığı, kişilerin ve/veya çevre güvenliğini etkileyebilecek olan E/E/PE emniyetle ilgili sistemlerle ilgilidir. Bununla birlikte, başarısızlığın sonuçlarının ciddi ekonomik sonuçlara sahip olabileceği ve bu gibi durumlarda, ekipmanın veya ürünün korunmasında kullanılan herhangi bir E/E/PE sisteminin belirlenmesinde standardın kullanılabileceği kabul edilmiştir. Bunun önemli etkileri vardır, çünkü fonksiyonel emniyet ile tanımlanan IEC 61508'in, fonksiyonel performans parametresinin emniyet olmadığı, örneğin çevre koruma veya varlık koruması olduğu sistemlerin belirlenmesi ve uygulanması için kullanılabileceği anlamına gelmektedir.

IEC 61508'in temel özelliklerinden bazıları aşağıda verilmiştir.

- E/E/PE emniyetle ilgili sistemler ile ilgili olarak, ürün ve sektör uluslararası standartlarının geliştirilmesini sağlar. Bu hem uygulama sektörleri içinde hem de genelinde yüksek düzeyde tutarlılığa (örneğin; temel prensipler, terminoloji vb.) yol açmalıdır; bunun hem emniyet hem de ekonomik faydaları olacaktır.
- E/E/PE emniyet sistemleriyle ilgili sistemler için gerekli fonksiyonel emniyeti sağlamak için gerekli emniyet gereksinimleri spesifikasyonunun geliştirilmesi için bir yöntem sağlar.
- E/E/PE emniyetle ilgili sistemler tarafından uygulanacak emniyet işlevleri için hedef emniyet bütünlüğü seviyesini belirlemek için emniyet bütünlüğü seviyelerini (SIL'ler) kullanır.
- Emniyet bütünlüğü seviyesi gereksinimlerinin belirlenmesi için risk bazlı bir yaklaşım benimsemiştir.

- Emniyet bütünlüğü seviyelerine bağlı E/E/PE emniyetle ilgili sistemler için sayısal hedef arıza önlemleri belirler.
- Tek bir E/E/PE emniyetle ilgili sistem için talep edilebilecek tehlikeli bir arıza durumunda, hedef başarısızlık önlemlerine daha düşük bir sınır koyar. E/E/PE'de çalışan emniyetle ilgili sistemler için:
  - Düşük talep modu, alt sınır, tasarım işlevini istek üzerine yerine getirme ihtimalinin 10^{-5} arasında olması ihtimalinde ortalama olasılık olarak belirlenmiştir.
  - Yüksek talep veya sürekli çalışma modu olan alt sınır, saatte 10^{-9} arası ortalama tehlikeli bir başarısızlık sıklığına ayarlanır.

2.3 Fonksiyonel Emniyet Kavramı

Emniyet, mülke veya çevreye verilen zararın bir sonucu olarak doğrudan veya dolaylı olarak kabul edilemez bedensel yaralanma riski veya insanların sağlığına zarar vermeden kaçınma olarak tanımlanır.

Fonksiyonel Emniyet, girdilerine yanıt olarak doğru çalışan bir sisteme veya donanıma bağlı olarak genel emniyetin bir parçasıdır. Örneğin; yanıcı sıvı barındıran tankın içine yanıcı sıvının girmesini veya girmemesini sağlayan vanalı sistemin tankına eklenen sıvı seviye şalteri bir fonksiyonel emniyet örneğidir.

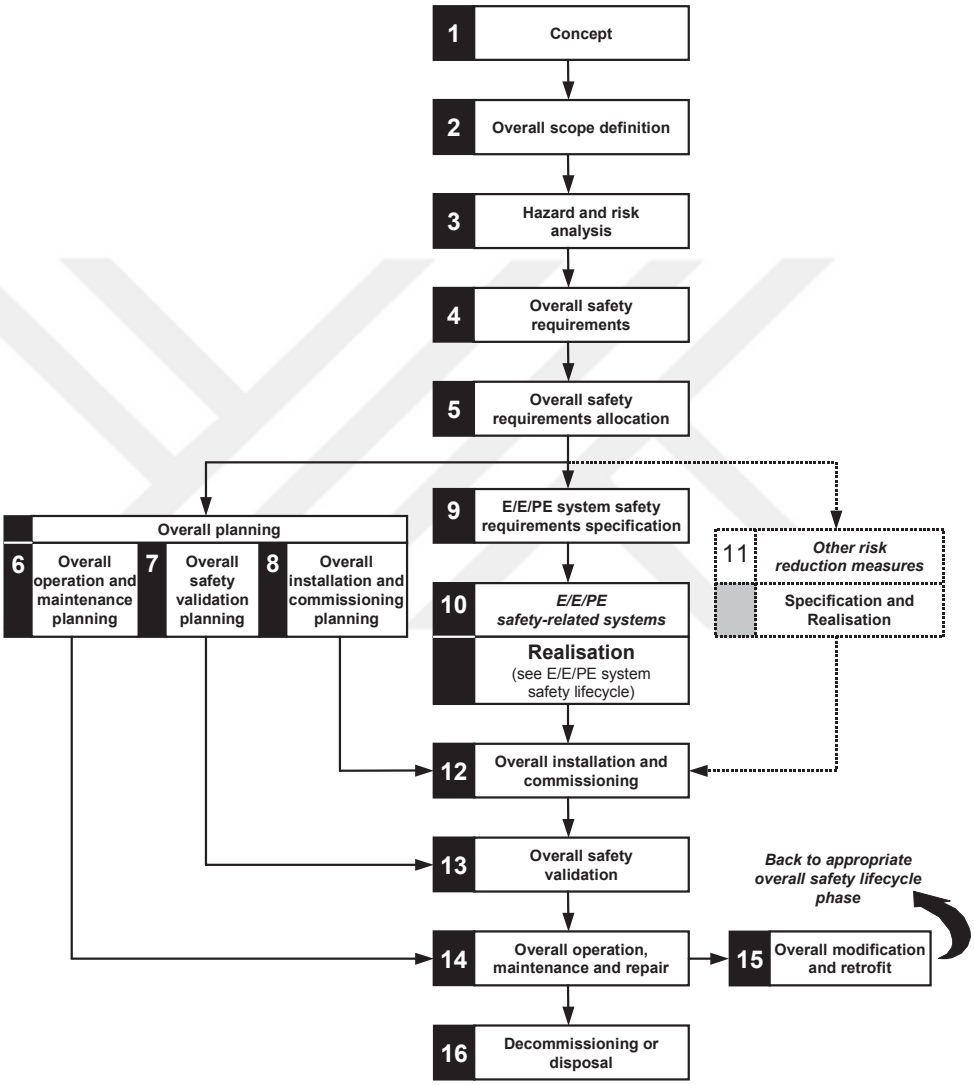
2.4 Fonksiyonel Emniyeti Sağlamak için Strateji

Fonksiyonel emniyeti sağlama stratejisi, aşağıdaki kilit unsurlardan oluşur:

- Fonksiyonel emniyet yönetimi
- Uygulanabilir emniyet yaşam döngüleri ile ilgili aşamalar için teknik gereklilikler
- Fonksiyonel emniyet değerlendirmesi (FSA)
- Kişilerin yetkinliği

IEC 61508, ilgili tüm aşamaların ele alınması için üç emniyet yaşam döngüsü kullanır:

- Genel Emniyet Yaşam Döngüsü (bkz. Şekil 2.2)
- E/E/PE Sistemi Emniyet Yaşam Döngüsü (bkz. Şekil 2.3)
- Emniyet-Kritik Yazılım Yaşam Döngüsü (bkz. Şekil 2.4).

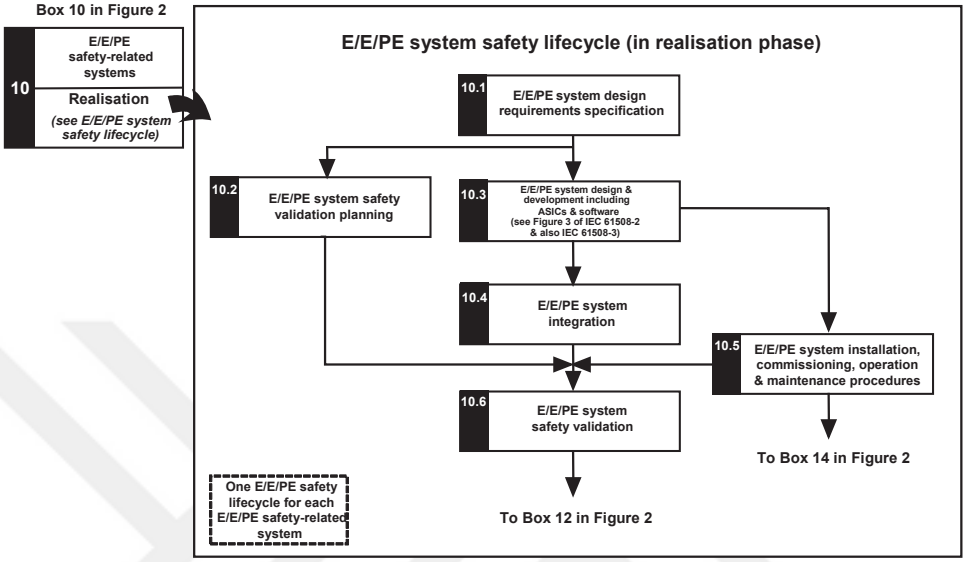


Şekil 2.2 IEC 61508'e göre genel emniyet yaşam döngüsü

E/E/PE emniyetle ilgili sistemler için gerekli emniyet bütünlüğünü elde etmek için gerekli tüm faaliyetleri sistematik bir şekilde ele almak için, IEC 61508, Şekil 2.3'te gösterilen Genel Emniyet Yaşam Döngüsünü teknik çerçeve olarak benimsemektedir. IEC 61508'de belirtilen Genel Emniyet Yaşam Döngüsü, standarda uygunluk iddiasında temel olarak kullanılmalıdır, ancak standardın her bir maddesinin amaçlarını ve gereksinimlerini karşılayan, Şekil 2.3'te verilenlere farklı bir Genel Emniyet Yaşam Döngüsü kullanılmaktadır.

Genel emniyet yaşam döngüsü aşağıdaki risk azaltma modeli kapsar:

- E/E/PE emniyetle ilgili sistemler
- Diğer risk azaltma önlemleri.



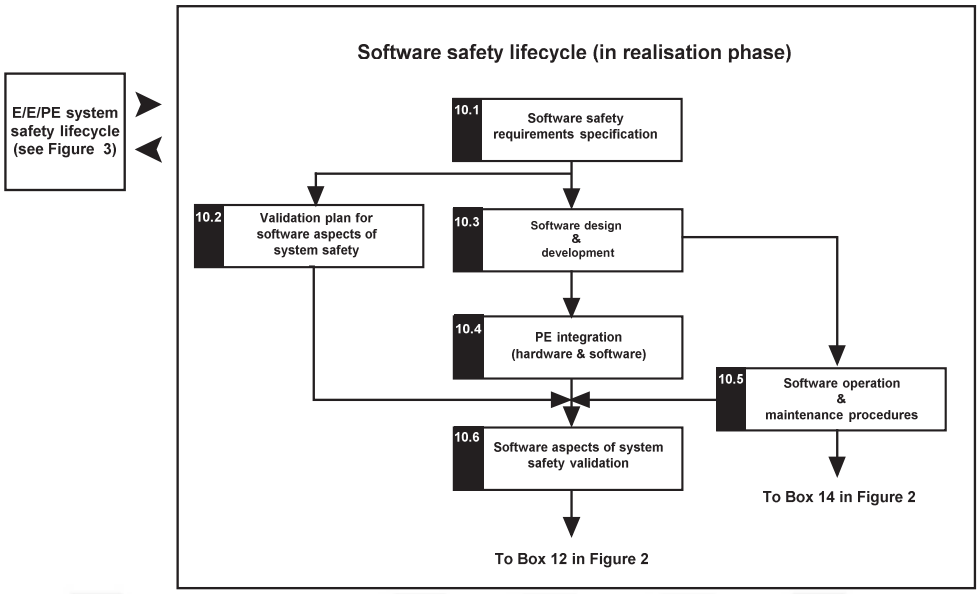
Şekil 2.3 IEC 61508'den E/E/PE sistemi emniyet yaşam döngüsü (gerçekleştirme fazı)

Genel emniyet yaşam döngüsünün E/E/PE emniyetle ilgili sistemlerle ilgili kısmı genişletilmiştir ve Şekil 2.3'te gösterilmiştir. Bu, E/E/PE Sistem Güvenliği Yaşam Döngüsü olarak adlandırılır ve IEC 61508-2 için teknik çerçeveyi oluşturur. Emniyet-Kritik Yazılım Yaşam Döngüsü, Şekil 2.4'te gösterilmiştir ve IEC 61508-3 için teknik çerçeveyi oluşturmaktadır.

Genel E/E/PE Sistem Güvenliği ve Emniyet-Kritik Yazılım Yaşam Döngüsü rakamlarının gerçekliğin basit görünümleri olduğunu ve belirli fazları veya fazlar arasındaki tüm yenilikleri göstermediğini kabul etmek çok önemlidir. Bununla birlikte, yineleme, Genel E/E/PE Sistem Güvenliği ve Emniyet-Kritik Yazılım Yaşam Döngülerinde gelişimin önemli ve hayati bir parçasıdır.

Planlama fazları genel ürün güvenliğini içermektedir. Bunların arasında kavram tanımı ve kapsamı bulunmaktadır. Ayrıca bir tehlike ve risk analizi ile elde edilen de emniyet gereksinimleri ve bileşenlere ait Emniyet Bütünlük Düzeyleri (SIL) tahsisi de bu bölümde tanımlanmaktadır.

Planlama aşamaları sırasında; işletme, bakım, montaj ve tanımlı emniyet doğrulaması planlanmaktadır. Emniyet yaşam döngüsünde önemli bir adım, ürün uygulama



Şekil 2.4 IEC 61508'den Yazılım sistemi emniyet yaşam döngüsü (gerçekleştirme fazı)

aşamasıdır. Bu aşamada, donanım ve yazılım uygulanması aşağıdaki gibi alt kısımlardan oluşur.

- Gereksinimler belirtimi : Ürün için emniyetle ilgili işlevlerin tam belirtimi, bu işlevlere SIL tahsis edilmesi ve bu işlevler için risk azaltma önlemlerinin belirtilmesi.
- Doğrulama planlaması : Sistemin belirtilen emniyet gereksinimlerine göre nasıl doğrulanacağına dair bir plan hazırlanması.
- Tasarım ve geliştirme : Emniyet gereksinimlerine göre emniyet-kritik yazılımı / donanımının tasarımı ve uygulanması.
- Entegrasyon : Emniyetle ilgili komple bir ürün oluşturmak üzere geliştirilen yazılım / donanım kuruluşlarının entegrasyonu / montajı.
- Çalıştırma ve bakım : Geliştirilen yazılımın / donanım ürününün doğru çalıştırma eylemleri.

Fonksiyonel emniyet yönetimi, doğrulama ve fonksiyonel emniyet değerlendirmesi ile ilgili faaliyetler, Genel E/E/PE Sistem Güvenliği ve Emniyet-Kritik Yazılım Yaşam Döngülerinde gösterilmemiştir. Bu durum, emniyet yaşam döngüsü faaliyetlerinin karmaşıklığını azaltmak için yapılmıştır. Bu faaliyetlerin emniyet yaşam döngülerinin ilgili aşamalarında uygulanması gerekecektir.

Sistemlerin emniyetli olması için, tüm alt işlevleri emniyetle gerçekleştirmeleri gerekir. Fonksiyonel emniyet, sistemin güvenliğinin bir parçasıdır ve sistemin girişlerine göre doğru çalışmasına bağlıdır. Örneğin; sıcaklık sensörü aşırı ısınmaya karşı koruyucu bir cihazdır. Bu cihaz motorun aşırı ısınmasından korumak için kullanılmaktadır. Motorun sıcaklığı normal çalışma seviyesinin üzerine çıktığında, emniyet cihazı kapatır ve motoru durdurur. Motorun bir sensör kullanılarak durdurulması "Fonksiyonel emniyet" örneğidir.

Tüm tehlikeli durumlar ve risk analizi, sistemlerin, ekipmanın çevre için ve kendileriyle ilişkili oldukları kişi veya kişiler için neler oluşturabileceği üzerine belirlenir. Bu analiz, tüm olası tehlikeler için fonksiyonel emniyetin gerekli olup olmadığını belirler. Gerekirse, tasarım aşamasında her tehlikeli durumda fonksiyonel emniyeti sağlamak için önlemler alınmalıdır.

Fonsiyonel emniyeti sağlamak, tehlikeli durumları ortadan kaldırmak için kullanılan yöntemlerdendir. Daha da önemlisi, tehlikeli durumların azaltılması, ortadan kaldırılması ve devam eden bir emniyet tasarım aşamasında gerçekleştirilir.

Belirli bir eylem veya eylemler gerçekleştirerek tehlikenin uygun düzeyde belirlendiği sistemleri tanımlamak için bir emniyet konsepti kullanılır. Bu eylemler emniyet prosedürleridir. İki tür emniyet prosedürü vardır. Bunlar:

- Eylemler için emniyet gereksinimleri (Eylemlerin ne yaptığı)
- Emniyet bütünlüğü gereksinimleri (Emniyet prosedürlerini uygun şekilde yürütme yeteneği)

Operasyonlar için emniyet gerekleri, tehlike analizi; emniyet bütünlüğü gereklilikleri risk değerlendirmesi ile ortaya çıkartılmaktadır.

Fonksiyonel güvenliğin bir örneği olarak, döner bir demir bıçağı ve koruma kılıfı olan bir makineyi ele alınabilir. Makinenin rutin temizliği kılıf çıkarılarak yapılmaktadır. Koruyucu kılıf dahili bir kilitleme sistemine sahiptir, bıçaklar durur çünkü kılıf açıkken motor elektrik devresinden ayrılır ve operatör temizleme işlemini güvenle gerçekleştirebilmektedir.

Emniyet için, tehlike durum analizleri ve risk değerlendirmeleri gereklidir. Bunlar:

1. Tehlikeli Durum analizi, dönen bıçakların temizliğinin neden olduğu tehlikeleri tanımlar. Bu durumda, koruyucu kılıf 8 mm'den fazla açıksa, acil durum

frenlemesi etkinleştirilmeli ve makine durdurulmalıdır. Daha ileri analiz, kılıfı açtıktan sonra makineyi 1 saniye içinde durduracaktır.

2. Risk Değerlendirme analizi, emniyet prosedürlerinin etkinliğini ve başarısını belirler. Bir risk değerlendirmesinin amacı, emniyet bütünlüğünü sağlamak için emniyet prosedürlerinin tehlikeli bir durumla ilişkili riskin kabul edilemez değerini aşmamasını sağlamaktır.

Bu eylemi yapmamak operatörün zarar görmesine neden olmaktadır. Risk, koruyucu kılıfın açılma sıklığı ile de ilişkilidir. Gerekli SIL emniyet bütünlüğü seviyesi, yaralanmanın ciddiyetine ve tehlikeli durumun ortaya çıkma sıklığına bağlı olarak artmaktadır.

2.5 SIL Kavramı

SIL kavramı bir sistemin SIF güvenirliğini belirleyen kavramdır. SIF, herhangi bir işlem sırasında oluşabilecek tehlikeli bir durumu tespit etme ve önleme işlevidir. Örneğin, SIF; aşırı ısınma, bir sıcaklık sensörü kullanarak bir motorun sıcaklığını ölçme ve motoru durdurma işlemidir. SIF işlevi belirli bir işlemi kontrol eder. Her işlem ayrı bir SIF işlevi gerektirir. Ancak, tüm SIF işlevleri için bir SIS sistemi oluşturulur.

SIS, tüm sistemi izleyen ve tehlikeli durumlarda güvenliğini sağlayan bir kontrol sistemidir. Ek olarak, “Fonksiyonel Emniyet” terimi, sistemdeki tüm SIF fonksiyonlarının başarılı bir şekilde yerine getirildiğini ve bunun sonucunda işlem sırasında ortaya çıkan risklerin kabul edilebilir bir seviyeye indirildiğini ifade eder.

SIL sertifikası SIF’e göre uygulanır. SIL sertifikası, SIF işlevinin güvenirlik aralığını tanımlar. Güvenirlik aralığı "Talep Üzerine Başarısızlık Olasılığı"(PFD) ile ifade edilir ya da bunun tersi olan "Risk Azaltma Oranı"(RRD) ile belirtilir. SIL seviye tanımlaması, IEC 61508’de belirtildiği gibi Tablo 2.2’da gösterilmiştir [2].

Tablo 2.2 Güvenirlik-PFT-RRD SIL Değerlendirmesi

SIL Seviyesi	Güvenirlik	PFT	RRD
SIL 4	%99.99	%0.01 ve %0.001	10000 ve 100000 arasında
SIL 3	%99.9 ve %99.99	%0.1 ve %0.01	1000 ve 10000 arasında
SIL 2	%99 ve %99.9	%1 ve %0.1	100 ve 1000 arasında
SIL 1	90% ve %99	%10 ve %1	10 ve 100 arasında

SIL sertifikası, sistem arızası olasılığına bağlı olarak dört düzeyde tanımlanır. SIL seviyesi ne kadar yüksek olursa, sistemin çökme olasılığı o kadar düşük ve performansı da o kadar yüksek olur. Ancak, SIL seviyesi arttıkça, sistem maliyetleri, bakım

maliyetleri artar ve sistem daha karmaşık hale gelir. Ekipman Hatası Toleransı(HFT) emniyet fonksiyonunun kalitesini gösterir.

HFT 0: Tek kanal kullanım: Tek hata emniyet kaybına neden olabilmektedir.

HFT 1:İleri versiyon: emniyet kaybına neden olacak, aynı anda oluşan en az iki donanım hatası olmalıdır.

HFT 2:Çift ileri versiyon: emniyet kaybına neden olacak, aynı anda oluşan en az üç donanım hatası olmalıdır.

Kesin Hata Oranı SFF değeri, kesin cihaz hatalarının oranını belirtir. Örneğin, 79% SFF değeri olan bir cihazda 100 hatanın 79'u dışındaki hatanın cihaz emniyet fonksiyonuna etki etmediği anlamına gelmektedir. SFF değeri HFT ile beraber kullanılır. Bu iki değer göz önünde bulundurularak hangi cihazın kullanılması gerektiği belirlenir. Bu değerlendirme Tablo 2.3'de gösterilmiştir.

Tablo 2.3 IEC61508'e göre SFF, HFT ve SIL Seviye Değerlendirmesi

SFF	**HFT**		
	0	**1**	**2**
<%60	-	SIL1	SIL2
%60-%90	SIL1	SIL2	SIL3
%90-%99	SIL2	SIL3	SIL4
>%99	SIL3	SIL4	SIL4

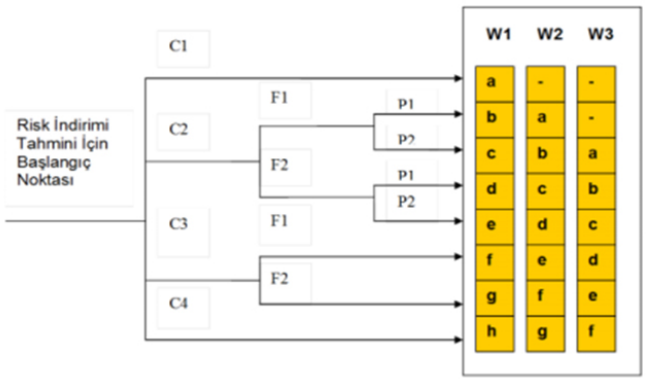
Sistemde gerekli SIL seviyesi, karşılık gelen riskin değeri ve istenmeyen bir olay esnasında sistemin ihtiyaç duyduğu koruma seviyesi ile belirlenir. Endüstriyel uygulamalar için, SIL 4 seviyesi olan sistemler çok karmaşık ve uygun maliyetli değildir. Bu sistemlerin uygulanması için uygun değildir.

SIL sertifikası, tüm sistemle ilişkili emniyet seviyesidir. Sistemdeki herhangi bir eleman veya bileşen için SIL seviyesi yoktur. Sertifikalı bir SIL sistemi, kullanım için uygun veya uygun değil olarak sınıflandırılır. Ek olarak, sadece sistemde kullanılan elemanların veya bileşenlerin SIL seviyesine karşılık gelmesi, sistemi SIL seviyesine uygun hale getirmez. Sistemi herhangi bir SIL seviyesine uygun hale getirmek için tüm sistem kontrol edilmelidir.

Tesis veya bir kısmı için gerekli olan SIL seviyesi, sayısal yöntemle veya nitel yöntemle hesaplanabilir. Sayısal yöntemde, risk olasılığı hesaplanır ve gerekli SIL seviyesi belirlenir. Karşılaştırılabilir durumların hata katsayılarının analizini kullanmak, uygun veritabanını kullanmak veya uygun tahmin yöntemlerini kullanımı için hesaplanır. Sayısal yöntem hesaplama detayları IEC 61508:5 bölümünde açıklanmıştır.

Nitel yöntem basitleştirilmiş bir model olup, tehlike türlerine göre gereken SIL seviyesinin belirlenmesinde yardımcı olmaktadır.

Nitel yöntem, tehlike türlerine göre gereken SIL seviyesini belirlemeye yardımcı olan basitleştirilmiş bir modeldir. Bu nitel yöntem Şekil 2.5'de gösterilmiştir.



-	Güvenlik İhtiyacı Yok		
a	Özel Güvenlik İhtiyacı Yok		
b, c	SIL1	d	SIL2
e, f	SIL3	g	SIL4
h	Tek Bir Emniyet sistemi yeterli değil		
Hasarın Derecesi			
C1	Küçük Zarar		
C2	Bir veya Daha Fazla Kişi İçin Ciddi Kalıcı Zarar		
C3	Birkaç Kişinin Ölümü		
C4	Birçok Kişinin Ölümü		
Bir kişinin tehlikeli bölgede kalış süresi			
F1	Kişi Tehlikeli Bölgede nadiren bulunur		
F2	Kişi Tehlikeli Bölgede sıklıkla bulunur		
Tehlikeden kaçınma			
P1	Tehlikeden Kaçınmak bazı durumlarda mümkün		
P2	Tehlikeden Kaçınmak zorlukla mümkün		
Olma olasılığı			
W1	Yüksek Olasılık		
W2	Orta Olasılık		
W3	Küçük Olasılık		

Şekil 2.5 SIL seviyesi belirlenmesinde kullanılan nitel yöntem

Emniyet sistemlerinde iki tür hata vardır; rasgele ve sistematik arızalar. Rasgele hatalar, sistem parçaları arası iletim esnasında oluşmaz. Ekipmanın bazı kısımlarında meydana gelen kısa devre, bağlantı kesme ve değer yanlışlığı gibi faktörlerden dolayı rasgele hataların gerçekleşmesi söz konusudur. Sistematik hatalar iletim sırasında cihazda oluşurlar. Bunlar genellikle tasarım, mühendislik ve yapılandırma hatalarıdır. Örnekler arasında yazılım hataları, uygun olmayan cihaz aralığı seçimi sayılabilir.

IEC 61508 standardı, gerekli SIL seviyesine ulaşmak için bir sistemin tasarımı, entegrasyonu, işletimi ve bakımı için gereksinimleri belirlemektedir [2].

2.6 Güvenirlik Teorisi

Kullanıcılar kullandıkları sistemlerden ve makinelerden güvenirlik beklemektedirler. Mühendisler günümüzde çeşitli makine ve sistemlerin plan, üretim ve çalıştırılması ile görevlendirilmektedirler. Güvenirliği sistem ve makinelerin kıyaslanmasında kullanılabilir. Ama bu kıyaslama mantıksal (güvenilmez) 0 veya (güvenilir) 1 ile değil, % yüzde olarak oransal anlamlı hale gelmektedir.

Güvenirlik, başlarda deneyimlere göre ortaya çıkarken, şimdilerde mühendislik ve işletme yönetiminin birçok alanında kullanılan bir konu başlığı haline gelmiştir. Belirli senaryolar oluşturularak sistem veya makinelerin hangi durumlarda kullanım dışı kalacağı ve bu olayın oluşturacağı olumsuzluklar tanımlanmaya çalışılır. Ayrıca tasarım maliyeti ile istenen güvenirlik düzeyi arasındaki dengenin kurulması üzerinde çalışılmaktadır.

Risk değerlendirmesinin ana yapı taşlarından biri, önceki bilgi ve incelenen sistem deneyiminin varlığıdır. Ancak, incelenen sistem yeni ve daha az bilinen bir sistemse, bu amaç için geliştirilen risk analizi yöntemleri uygulanmalıdır. Sistem güvenliğini analiz etmek için kullanılan risk değerlendirme yöntemleri genellikle güvenirlik teorisini ve olasılık hesaplamalarını kullanır.

2.6.1 Güvenirlik

Güvenirlik, genel olarak bir bileşenin belirli zaman dilimi ve belirli durumlarda tanımlı görevi yerine getirme ihtimalidir. Bir başka deyişle güvenirlik analizi, sistemin parça ve birimlerinin bozulma oranlarının analizidir.

Güvenirlik (dependability) analizin temel parçaları şu şekildedir:

- **Kullanılabilirlik (availability):** Herhangi bir zamanda sistemin düzenli çalışabilme, kendisinden istenen hizmeti verebilme olasılığı,
- **Güvenirlik - Doğruluk (reliability):** Belirli bir diliminde gerekli hizmeti doğru şekilde verebilme olasılığı,
- **Güvenlik (security):** Sistemin dış etkenlere karşı dayanbilme ölçüsü,
- **Emniyet (safety):** Sistemin canlılara ve çevresine zarar vermeden çalışabilme ölçüsüdür.

Her sistem kendisini oluşturan alt sistemlerin etkileşimlerinin oluşturduğu bir ağdır. Bu alt sistemler birbirilerini etkileyebildiği gibi sistemin ana karakteristiğini de etkileyebilmektedirler. Lusser Teoremine [67] göre sistem başarısı, sitemin kendisini oluşturan alt sistemlerin başarılarının çarpımına eşit olmaktadır. Lusser teoremi (2.1) 'de gösterilmiştir.

$$R_x = R_1 \cdot R_2 \cdot R_3 \ldots R_n \tag{2.1}$$

Sistemlerde süreçlerin iyileştirilmesi için Lusser teoreminde kabul edilen yedi varsayım vardır. Bunlar:

1. Sistem performansının başarılı olması alt sistemlerin her birinin performanslarının başarılı olduğunu göstermez.
2. Alt sistemlerden sistem performansını düşürenin bulunması gerekmektedir. Bu alt sistem bir cihaz olabileceği gibi yönetim algoritması da olabilmektedir.
3. Sistemi parçalayarak, parçaların başarılarının arttırılmasının ardından parçaların bir araya getirilerek sistemin bütününün başarılı hale getirilmesi mümkündür.
4. Sistemin başarısız alt sistemleri yerine başka bir bölümünün iyileştirilmesi, sistemi bütün olarak geliştirmeye etkisi olmaz.
5. Sistemde yaşanan istenmeyen durumların pek çoğuna bazı ana problemler sebep olmaktadır.
6. Bu ana problemler genellikle açıkça anlaşılır değildir. Ana problemler istenilmeyen etkiler yoluyla görülebilir olmaktadır.
7. Sistemin her alt sisteminde başarıyı en üstte tutmak sistemin başarısını en üstte tutmak anlamına gelir.

Bir sistemde veya cihazda meydana gelen arızalar, engellenmezse ikincil arızalar oluşur ve sistemin yıkılmasına sebep olur. Bu yıkılmalar ciddi boyutta insan ve çevresel zararlar içerebilir. Bu yıkılmaları oluşturan sorunları anlayabilmek için güvenirlik analizinin yapılması gerekmektedir. Bu; tasarım aşamasında hatalardan kaçınmak, kullanım sırasındaki hataları tanımlamak ve düzeltmek ve işletme hatalarından kaynaklanan hasarı sınırlamak suretiyle başarılabilir. Sistem güvenirlik analizi:

- Güvenirlik kriterlerinin tanımı ve değerlendirilmesi,

- Kriterleri belirli bir seviyeye çıkarmak için sistemin hatalı alt sistemlerinin belirlenmesi,
- Sistemin gereksiz kısımlarının sistemden sökülmesi,
- Çeşitli tasarım seçeneklerinin karşılaştırılması,
- Koruma ve gözetim için gerekli ek donanımın belirlenmesi,
- Önleyici bakım ve ölçüm periyotlarının belirlenmesi sağlanır.

Güvenirlik; sistemin amaçlanan işlevlerini belirtilen standartlara uygun olarak belirli bir zaman aralığında yerine getirme olasılığıdır. Güvenirlik fonksiyonu $R(t)$, sistem ve parçalarının t zamanında kendi işlevlerini yerine getirebilme olasılığıdır. Sistemde (t) aralığında belirlenen güvenirlik fonksiyonu sadece olasılıktır. Güvenirlik genellikle $R(t)$ olarak tanımlanır.

Bahsi geçen R güvenirlik, t ise zaman aralığıdır. Arıza ihtimali $F(t)$, bir sistem veya sistem parçasının (t) zaman aralığından önce arızalanma ihtimalini (2.2)'de gösterilmiştir.

$$F(t) = 1 - R(t) \tag{2.2}$$

$F(t)$ dağılım fonksiyonunu, t'den küçük rastgele değişkenlerinin testlerindeki olasılığı (2.3)'de belirlenir.

$$F(t) = \int_{-\infty}^{t} f(t) \cdot dt \tag{2.3}$$

$f(t)$, hata olasılığının negatif zaman türevi olan (2.3)'deki olasılık yoğunluğunun bir fonksiyonu olarak tanımlanır. Hata oranı, ilk hatanın $t = 0$ zamanından sonra ortaya çıkma olasılığıdır. İlk hatanın oluşma olasılığı (2.4) kullanılarak hesaplanır.

$$f(t) = -\frac{dR(t)}{dt} \tag{2.4}$$

Bir sistem hatası olarak ifade edilen bir diğer önemli değişken, hata oranına $z(t)$ bağlıdır. Hata oranı (2.5) kullanılarak hesaplanır.

$$z(t) = -\frac{f(t)}{R(t)} \tag{2.5}$$

Sabit bir başarısızlık oranı λ olarak kabul edilirse, güvenirlik işlevi (2.6) olur. Denklem, güvenirlik işlevi genel halidir.

$$\begin{aligned} z(t) &= \lambda \\ R(t) &= e^{-\lambda t} \end{aligned} \tag{2.6}$$

- **Seri Sistemler**

  Mühendislik sistemleri genellikle çeşitli bileşenlerden oluşur. Tüm bileşenler çalışıyorsa bir çalışan bir seri sistemden bahsedebiliriz. $R1(t)$, $R2(t)$, $R3(t)$ ve benzeri alt sistemin güvenirliği kabul edilirse tüm sistemin güvenirlik olasılığı, $R_{sistem}(t)$ olmakta (2.7) ile hesaplanmaktadır.

  $$R_{sistem}(t) = R_1(t) \cdot R_2(t) \cdot R_3(t) \ldots \tag{2.7}$$

  Seri sistemler için, tüm sistemin hata fonksiyonu (2.8) ile hesaplanmaktadır.

  $$F_{sistem}(t) = 1 - R_{sistem}(t) = F_1(t) + F_2(t) + F_3(t) \ldots \tag{2.8}$$

- **Paralel Sistemler**

  Paralel sistem, tüm bileşenler arızalandığında çalışmayan bir sistemdir. Örnek olarak; tüm aydınlatma sistemi, birkaç aydınlatma elemanından oluşmaktadır. Aydınlatma görevini yerine getirememesi için tüm aydınlatma elemanlarının arızalanması gerekmektedir. Tüm sistemin hata fonksiyonu, (2.9) denklemi kullanılarak hesaplanmaktadır.

  $$F_{sistem}(t) = F_1(t) \cdot F_2(t) \cdot F_3(t) \ldots \tag{2.9}$$

Güvenirlik yöntemlerinin uygulanması, onarımlar için harcanan sürenin yanı sıra, sistem ve/veya ilgili bileşenlerin arıza verilerinin bilgisine ihtiyaç duyar. Bazı operatör türleri için veriler de gerekli olabilir. Veriler doğrudan veri bankalarından veya teknik literatürden toplanabilir. Veriler mevcut değilse, derecelendirmeler kullanılabilir. Doğruluk seviyesi, analizin uygulanmasına büyük ölçüde bağlıdır.

Hesaplamalar, ileri matematiksel yöntemler gerektirebilir. Veri toplama ve değerlendirmede zaten birçok zorluk mevcuttur. Bir sorun da teknolojinin hızlı gelişimidir. Yeni bileşen sürümleri o kadar kısa aralıklarla üretilmektedir ki güvenirlikleri hakkında zamana bağımlı yeterli saha bilgisi mevcut değildir. Sistemin

kurulumuna ve meydana gelen hata türlerine bağlı olarak farklı istatistiksel modeller uygulanabilir. Bunlardan bazıları üstel dağılım, normal dağılım ve weibull dağılımıdır.

Diğer önemli kavramlar, başarısızlıklar arasındaki ortalama süre (MTBF) ve ortalama başarısızlık süresidir (MTTF). Bu iki kavram benzer, ancak aynı değildir. MTBF, yenilenmiş bir malzeme grubu veya sistemi için geçerlidir. Bu, toplam çalışma süresinin işlemini yerine getirememe sayısına oranı ile elde edilen zamandır. MTTF aksine, tamir edilemeyecek sistemler için uygulanmaktadır.

2.6.2 MTTF, MTTR, MTBF ve Kullanılabilirlik

Sistemlerin veya makinelerin gelecekteki davranışları net olmadığından, güvenirlik göstergeleri yalnızca olasılık yöntemler kullanılarak tahmin edilebilir. Güvenirlik ve kullanılabilirlik olarak tanımlanan olasılıklar, güvenirlik analizinin temel değerleridir. Güvenirlik analizlerinde; arızalar arası ortalama süresi (ortalama çalışma süresi) MTTF, ortalama tamir süresi (ortalama başarısızlık süresi)MTTR, MTBF arızalar arasındaki ortalama süre, tanımlı bir zaman aralığındaki arıza sayısı, başarısızlık maliyeti vb. temel güvenirlik kriterleri kullanılır. Bu kavramlar Şekil 2.6'da görselleştirilmiştir. Güvenirlikte rastgele bir değişken olan zaman, yani başarısız olma süresidir ($x = t$).

- **MTTF - Arızaya Kadar Geçen Ortalama Süre (Mean Time To Failure)**

  Ortalama başarılı işlev zamanı sistemin gerekli tüm işlevlerini yerine getirdiği zaman dilimidir. Sistem kullanılabilirliğini ve güvenirliğini arttırır. Arıza yoğunluğu fonksiyonu (2.10) ile gösterilir.

$$MTTF = E(t) = \int_0^\infty f(t) \cdot dt = \int_0^\infty R(t) \cdot dt \tag{2.10}$$

- **MTTR - Ortalama Tamir Süresi (Mean Time To Repair)**

  Sistemin arızalardan dolayı işlevini yerine getiremediği ortalama süreyi ifade eder. Bu süre, tamir için gereken süre ve tamircinin sisteme müdahale etmeye ihtiyaç duyduğu parçaların temin süresi de dahildir. Diğer açıdan, hatalı bir cihazın tamir süresi de tipik çalışma süresi gibi rastlantısaldır. Tamir işlemi için rastlantı değişkeni $E(t)$ $fr(t)$ Tamir yoğunluğu fonksiyonu, $Fr(t)$ tamir dağılım fonksiyonu hesaplanabilmektedir. $E(t)$, (2.11) ile belirlenmektedir.

$$MTTR = E(t) = \int_0^\infty t \cdot fr(t) \cdot dt \tag{2.11}$$

- **MTBF - Arızalar Arasındaki Ortalama Süre (Mean Time Between Failure)**

  Arıza Arasındaki Ortalama Süre (MTBF), sistemin tüm bölümlerinin görevlerini yerine getirdiği belli bir süre için geçen ortalama süredir. MTBF kavramı güvenirlik literatüründe çok yaygındır. Onarılabilir ve hasarlı parçaları değiştirerek çalışabilecek bir sisteme uygulanabilmektedir. $T(t)$ toplam işlem süre, r aksaklık sayısı olarak tanımlandığında MTBF (2.12) ile ifade edilmektedir.

$$MTBF = \frac{T(t)}{r} = MTTF + MTTR \tag{2.12}$$

  MTBF'nin onarılabilir sistemlerine uygulanabilir olduğuna dikkat edilmelidir. Bu durumda MTBF, ortalama yaşam beklentisini ifade etmektedir.

- **Kullanılabilirlik (Availability)**

  Kullanılabilirlik, sistemin belirtilen özelliklere göre çalışacağı sürenin ölçüsüdür. Sistem sürekli çalışma sırasında yüksek kullanılabilirlikle tasarlanmalıdır. Kullanılabilirlik konsepti, sürekli çalışabilen ve onarılabilen sistemler için geliştirilmiştir. Sistemin iki olası durumu bulunur: Çalışma süresi ve Onarım süresi. Kullanılabilirlik; bir sistemin tüm işlevlerini belirli bir t zamanında gerçekleştirilebilme olasılığı olarak ifade edilir. Kullanılabilirlik; güvenirlik ve sürekliliğin bir kombinasyonudur. Zaman içerisinde kullanılabilirlik gelişimi ve ortalama kullanım seviyesi ile ilişkili olarak iki kullanılabilirlik kavramı kullanılmaktadır:

  – Tahmin edilebilir anlık kullanılabilirlik,

  – Ortalama kullanılabilirlik.

Şekil 2.6 Sistem çalışma ve arıza döngüsü

Sistem işletilirken, sadece ortalama kullanılabilirlik matematiksel olarak hesaplanabilir. Bu oran, etkin işleyiş süresinin, çalışma süresi ve tamir sürelerinin

toplamına oranıdır. Sistemin kullanılabilirliği, sistemin işlemeyen ve işleyen süresi cinsinden $A(t)$ (2.13) ile hesaplanabilektedir.

$$A(t) = \frac{MTBF}{MTBF + MTTR} = \frac{\text{Çalışan Süre}}{\text{Çalışan Süre + Tamir Süresi}} \tag{2.13}$$

Korunabilirlik (Maintainability)

Korunabilirlik; sistemde bir arıza varsa, arızayı en kısa sürede düzeltip işleve devam edebilme olasılığıdır. Korunabilirlik, sorun giderme süresi (MTTR) (2.14) ile ifade edilir. Sistem korunabilirlik bilgilerinin, bakım prosedürlerinde yer alması için sistem tasarımcısı tarafından hazırlanması gerekmektedir.

$$M(t) = 1 - e^{-\frac{t}{MTTR}} \tag{2.14}$$

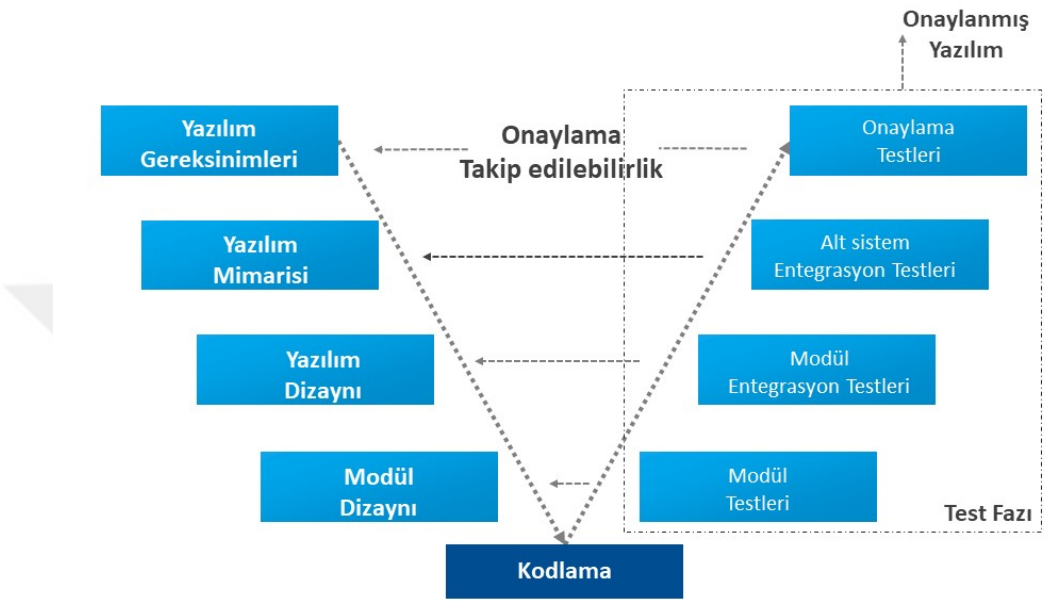
Arıza süresi, yukarıda belirtildiği gibi beş ana aşamadan oluşur. Sorun tespiti, sorun giderme işlemlerinin süresi, tamir operatörünün bilgisine ve kabiliyetine bağlıdır. Değiştirilecek ürünün elde edilmesi ve tamir operatörünün atanması gibi faktörlerden kaynaklanan gecikmeler, işletmenin yönetimsel yapısında yaşanan sorunlardır.

2.7 IEC 61508-3 Ekleri

Standardın 3. Bölümü, yazılımın ne kadar emniyetli olabileceğinin detaylı bir tanımını sunar. Günümüzde, emniyet-kritik sistemlerde kullanılan yazılım miktarı önceki dönemlere göre önemli ölçüde artmıştır. Aviyonik sistemlere örnek olarak, 1987'de piyasaya sürülen Airbus A320, yaklaşık 800.000 kod satırına ve 1994'te piyasaya sürülen Boeing 777 yaklaşık 4 milyon kod satırına sahipti. 2009 yılında piyasaya sunulan Boeing 787, yaklaşık 7 milyon kod satırına sahiptir [3]. Bu verilerden görülebileceği gibi, emniyet-kritik sistemlerinde kullanılan yazılım miktarı artmaktadır. Aviyonik sektöründe, sistemlerde kullanılan yazılım miktarının her 10 yılda bir ikiye katlandığı öne sürülmektedir [4].

Yazılım hataları, yazılım geliştirme sırasında ortaya çıkan sistematik hatalardan kaynaklanır. Yazılım hataları, fiziksel parçalarda görülebilen kaza sonucu ortaya çıkan arızalardan farklıdır. Kullanım sonucu oluşan aşınma ve yıpranma nedeniyle ortaya çıkmazlar. Yazılım geliştirmeye yönelik emniyet kriterleri de, sistematik hataları önlemek amacıyla yazılım geliştirme süreçlerine yönelik belirli gereksinimleri ve kısıtlamaları tanımlamak için geliştirilmiştir [68]. Bu yaklaşıma göre, bir sistem emniyet analizi yapılır ve yazılımın çalışacağı sistem için emniyet-kritik

düzey belirlenir ve daha sonra bu emniyet-kritik seviyesini sağlamak için standardın gerektirdiği yazılım yaşam döngüsü süreçleri belirlenir. Sertifika yetkilileri, emniyet-kritik yazılımın, ilgili standardın gerektirdiği yazılım yaşam döngüsü süreçlerine uygun olarak geliştirildiğini doğrular. Şekil 2.7'de, sıklıkla kullanılan V-modelini görebilirsiniz.



Şekil 2.7 IEC 61508-3 Yazılım geliştirme yaşam döngüsü - V model

Emniyet-kritik bir sistemin parçası olan ve bir arıza durumunda sistem emniyetini etkileyen yazılıma emniyet-kritik yazılım denir. Yazılım, insanlık tarafından üretilen en karmaşık ürünlerden biri olarak sınıflandırılmaktadır. Tüm büyük giriş verilerinin yazılımı, özellikle büyük sistemlerde test edilerek doğrulanması mümkün değildir; bu nedenle, kaç test yapıldığına bakılmaksızın, yazılımın hatasız olduğu garanti edilememektedir. Günümüzde, emniyet-kritik sistemlerde kullanılan yazılımların sayısı da artmakta, bu nedenle üretilen yazılımın istenen emniyet düzeyini karşıladığından emin olmak daha da zorlaşmaktadır.

Sistem hatalarının analizinde mekanik sistemler için emniyet analizi kolaylıkla yapılabilmekte iken, karmaşık yapısı nedeniyle yazılım için bu analiz geçerli değildir. Mekanik sistemlerin aksine, yazılım fizik yasalarına uygun olarak hareket etmemektedir. Yazılımın arızalanabileceği koşullar tahmin edilemez. Yazılım güvenirliğini test etme veya güvenirlik analizi yoluyla belirlemek mümkün olmadığından, tasarım kalite güvencesi konsepti tanıtılmıştır [11]. Tasarım garantisi; gereksinimler, tasarım ve geliştirme sırasında meydana gelebilecek hataların kabul

edilebilir bir sistem güvenliği düzeyinde tutulmasını sağlamak için yapılan planlı ve sistematik eylemleri ifade eder [12]. Tasarım desteği yaklaşımı, yazılım geliştirme sırasında daha titiz işlemlerin gerçekleştirildiği varsayılırsa, daha az hata yazılım içinde olacaktır [13]. Bu yaklaşımla, yazılımın kritiklik seviyesi arttıkça, uygulanan geliştirme ve doğrulama işlemleri miktar ve kalite açısından artmaktadır.

Radyasyon terapisinde kullanılan Therac-25 cihazı kullanırken yaşanan olay istenmeyen yazılım hatasına örnektir. Therac-25, içerisindeki bir yazılım hatası nedeniyle radyasyon tedavisi sırasında aşırı dozdan dolayı dört kişinin ölümüne sebep olmuştur [5]. Araştırmada, kazaya neden olan Therac-25 cihazı içerisinde yazılım geliştirilirken yazılım geliştirmenin temel ilkelerinin izlenmediği, testlerin yetersiz olduğu, gerekli belgelerin hazırlanamadığı ve kalite güvence işlemlerinin yapılmadığı tespit edilmiştir [6].

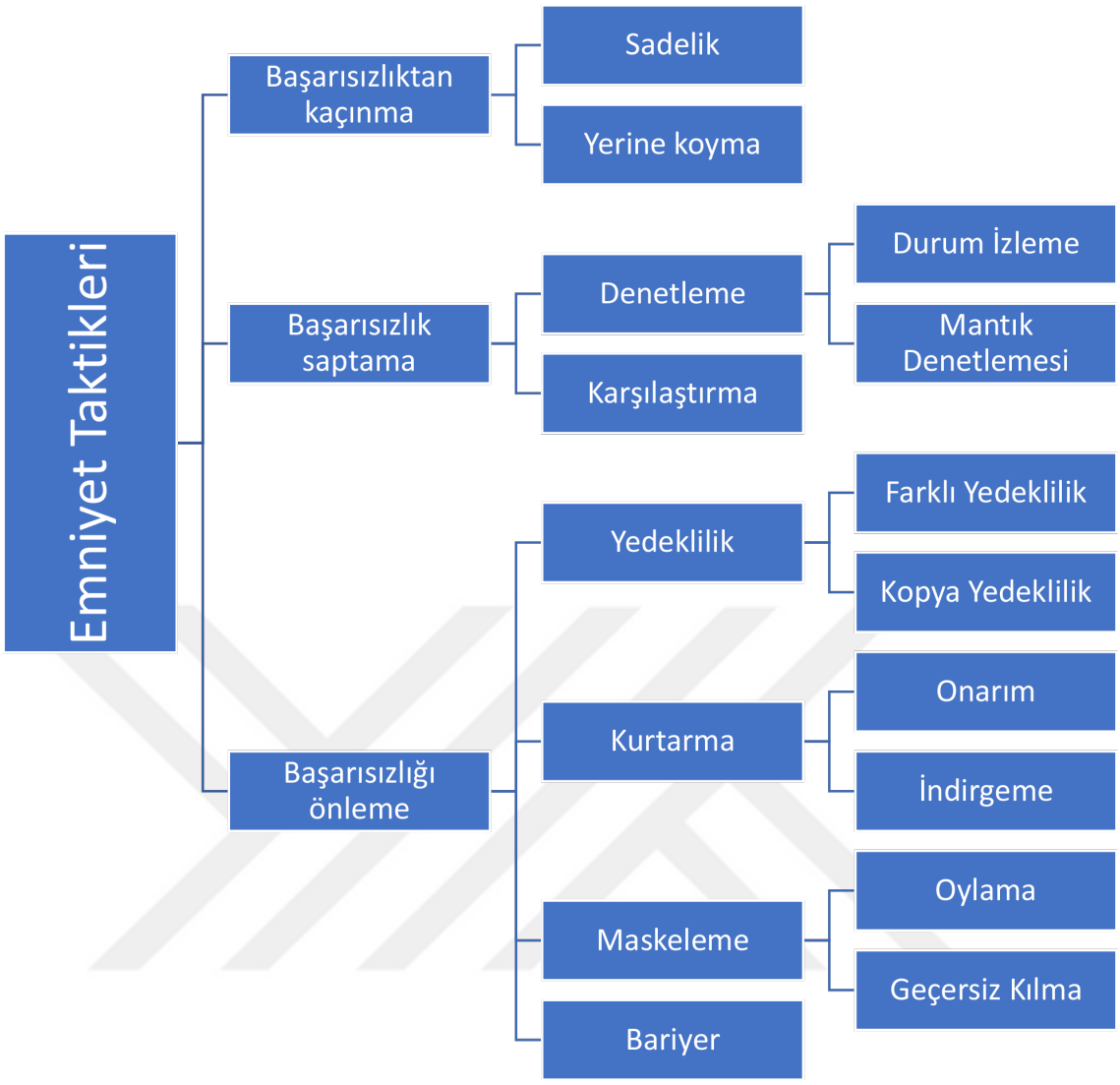
2.7.1 IEC 61508 Emniyet Yaşam Döngüsüne göre Emniyet Taktikleri

Emniyet taktikleri, ortak mimari çözümleri ve bunların genel sistem emniyeti üzerindeki etkisini tanımlar. Bu emniyet taktikleri, emniyet sertifikasyon tasarım kararları ile ilişkili değildir. IEC 61508, emniyet standardından mimari tasarım kararları ile ilgili bilgileri çıkararak emniyet taktiklerini yeniden ele alınmıştır. Emniyet taktik kullanımının emniyet-kritik sistemlerin farklı gelişim aşamalarındaki etkisini tanımlamak için bu bilgi genelleştirilmiştir.

Literatür içerisinde bu tarz genelleştirmeler ile Emniyet Taktik Katalogları oluşturulmuştur [8]. Bu kataloglara göre emniyet kavramı Şekil 2.8'deki gibi gruplandırılmıştır.

Bu taktikler, literatürde açıklanan emniyet mimarilerinden çıkarılmıştır. Başarısızlıktan kaçınma, Başarısızlık saptama ve Başarısızlığı önleme başlıklarında toplanmıştır. Her taktik "Amaç", "Açıklama", "Emniyet Yaşam Döngüsü Etkisi" ve "İlgili IEC 61508 Yöntemleri" üzerine etkilere göre incelenmiştir. Standardın bölümleri ile emniyet taktikleri arasındaki bağlantıları bulmak için standardı incelediğinde, standardın 7. bölümünde başlayarak, emniyet alanında sıklıkla uygulanan tekniklerin bir araya gelmesi sağlanmıştır. Bu teknikler, teknik amaçlar ile taktik amaçlar arasında benzerlikler bularak emniyet taktikleriyle eşleştirilmiştir. Standardın 7. bölümünde tanımlanan teknikler, "İlgili IEC 61508 Yöntemleri" bölümü için referans alınmıştır.

Standardın 7. bölümünde tanımlanan teknikler yine standardın 2. ve 3. bölümlerinde belirtilmektedir. Taktiklerin emniyet yaşam döngüsü üzerindeki etkilerini bulmak için referanslar incelenmiştir.



Şekil 2.8 Emniyet taktikleri

Bölüm 2'de açıklanan emniyet yaşam döngüsü, aşamaları doğrudan emniyet taktikleri tarafından etkilenen ve aşağıdaki özellikleri içermektedir. Bunlar; yetenekler: tasarım ve geliştirme, entegrasyon, işletme ve bakım, değiştirme, muayene ve emniyet doğrulamasıdır. Taktiklerin etkileri "Emniyet yaşam döngüsüne etki" bölümünde anlatılmıştır.

Emniyet taktikleri için IEC 61508 standardını benimsemek için yukarıda belirtilen yaklaşıma ek olarak, baştan sona emniyet taktikleriyle olan bağlantılara kadar standartların 1-6 bölümleri tekrar gözden geçirilmiştir.

2.7.1.1 Başarısızlıktan kaçınma

Sadelik ve Yerine Koyma bu başlıktaki taktiklerdir. Eğer bu seçenek mümkünse, genellikle arıza tespit ve arıza tutma taktiklerine göre tercih edilirler, çünkü diğer taktiklerden bağımsızdırlar ve diğer emniyet yaşam döngüsü aşamaları için ek bir yük oluşturmazlar.

- **Sadelik**
  - *Amaç:*

    Sistemi mümkün olduğu kadar basit tutarak arızalardan kaçınılması.
  - *Açıklama:*

    Sadelik, sistem karmaşıklığını azaltır. Yapılandırma yöntemlerini veya gereksiz işlevleri kesmeyi içermektedir ve sistem öğelerini düzenler veya temel emniyet işlevlerine indirgeyerek tehlikeleri ortadan kaldırır. Karmaşık sistemlerdeki basitleştirilmiş acil durdurma anahtar sistemi bir sadelik taktiği uygulamasıdır.
  - *Emniyet Yaşam Döngüsü Etkisi:*

    Bu taktik, sistem karmaşıklığındaki bir azalma veya hatta sistem işlevselliğinde bir azalma nedeniyle güvenlik yaşam döngüsünün her aşamasında çalışmalarını azaltır. Ayrıca diğer emniyet taktiklerinin önerdiği karmaşık yapılar konusunda çelişmektedir. Tanımlama aşamalarında sistemin anlaşılabilirliğini arttırmaktadır (IEC 61508-3:F). Tasarım ve Geliştirme aşamasında, IEC 61508-3:7.4.2.2, 7.4.3.6, 7.4.2.6 ve 7.6.2.2'de gerekli olan sistem geliştirmeyi kolaylaştırmaktadır. Bununla birlikte, bu taktik aynı zamanda sistemin gelişimine kısıtlamalar getirmektedir. Örneğin, IEC 61508-3:7.4.4.13, güvenli ve kanıtlanmış komutların kullanımıyla gerçekleştirilen ayarlamalar programlama dili komut setinin sınırlandırılmasını gerektirmektedir.
  - *İlgili IEC 61508 Yöntemleri:*

    IEC 61508-7:B.2.1 yapısal şartname, B.3.2 yapısal tasarım, C.2.7 yapısal programlama, E.3 yapısal açıklama yöntemi, C.4.2 programlama dili komut seti, C.4.2 asenkron yapıların sınırlılığı, E.5.13 yazılım karmaşıklığı kontrolcüsü
- **Yerine Koyma**
  - *Amaç:*

    Daha güvenir bileşenlerin kullanılmasına rağmen arızalardan kaçınılması.

– *Açıklama:*

Bileşenler ya da yöntemlerden birinin daha yüksek bir etkiye sahip olduğu diğer bileşenler ya da yöntemler ile değiştirilmesidir. Donanım ve yazılım için, emniyet alanında kanıtlanmış olan mevcut bileşenlerin kullanılması anlamına gelmektedir.

– *Emniyet Yaşam Döngüsü Etkisi:*

Yazılım veya donanım bileşenlerinin değiştirilmesi emniyet tehlikesi analizinin yeniden yapılmasını gerektirmektedir. Bununla birlikte, bu sertifikalar için dokümantasyon veya dokümantasyon bilgilerini yeniden kullanarak sertifika bileşenlerini azaltmak için önceden geliştirilmiş bileşenler veya üçüncü taraf bileşenlerle değiştirilmektedir. Daha güvenli bileşenler kullanılırsa, değiştirme işlemi donanım veya üçüncü parti donanım maliyetlerini artırabilmektedir. Örneğin, bir SIL3 bileşeni satın almak genellikle SIL2 bileşeni satın almaktan daha pahalıdır.

– *İlgili IEC 61508 Yöntemleri:*

IEC 61508-7:B.3.3 kanıtlanmış bileşenlerin kullanımı, B.5.4 Alan tecrübesi, C.2.10 kanıtlanmış / onaylanmış yazılım öğelerinin kullanımı, E.20 onaylanmış genel amaçlı çekirdek uygulaması, E.35 onaylanmış özel amaçlı çekirdek uygulaması, E.41 deneme ile kendini kanıtlamış devrelerin kullanımı, C.4.3 Sertifikalı takımlar ve derleyiciler, C.4.4 kendini kanıtlamış araçlar ve derleyiciler, E.4 kendini kanıtlamış araçlar, E.42 kendini kanıtlamış üretim süreci, E.28 kendini kanıtlamış sentez araçlarının uygulanması, E.29 kendini kanıtlamış kütüphanelerin uygulanması.

2.7.1.2 Başarısızlık saptama

Her emniyet taktiği, bir çeşit yedeklilik ve yedek bilgisinin test edilmesini gerektirmektedir. Denetleme taktikleri, bir sistemi kontrol edilmesi için çeşitli bilgiler sunmaktadır. Karşılaştırma taktiği, tüm bağımsız bilgileri veya sistemleri karşılaştırmaktadır.

- **Denetleme > Mantık Denetlemesi**

  – *Amaç:*

  Mantıksız sistem çıkışlarının veya durumlarının tespiti.

  – *Açıklama:*

  Mantık Denetlemesi taktiği, bir sistem durumu veya değerinin, sistem spesifikasyonunda tanımlanabilen veya sistemin iç yapısı veya yapısı

hakkındaki bilgiye dayanan geçerli bir aralıkta olup olmadığının kontrol edilmesidir. Bir Mantık Denetlemesi örneği, sistem çalışma zamanı sırasında belleğin uygun çalışıp çalışmadığını kontrol eden bir hafıza testidir. Hafızaya veri yazıldıktan sonra aynı veriyi okunabilmesi anlamına gelmektedir. Hafıza farklı şekilde davranırsa, arızalar tespit edilmektedir.

– *Emniyet Yaşam Döngüsü Etkisi:*

Mantıklı sistem çıkışları ve durumları belirtilmelidir(örneğin, ön koşulların sistem giriş aralığını sınırladığı IEC 61508-3:C.2, C.3a). Bu değer aralığı, yalnızca belirtilen değer aralığının kontrol edilmesi gerektiğinden sistemin kontrol edilmesine yardımcı olabilir (IEC 61508-3: C.2). Emniyet doğrulaması için, Mantık Denetleme'nin emniyet işlevselliğini doğrulamak için başka bir uygulama sağladığı veya tasarımdaki bir dereceye kadar sistematik hataları tespit ettiği söylenebilir (IEC 61508-6: D.1.4).

– *İlgili IEC 61508 Yöntemleri:*

IEC 61508-7: A.1.2 bayrak izlenmesi, A.2.7 sinyal kontrolü, A.3.1-A.3.3 kendi kendine test, A.4.1-A.4.4 denetleme denetimi, A.5.1-A.5.5 RAM denetimi, A.6.1 denetleme kalıbı, A.7.1 tek bit donanım desteği, A.7.2 çok bitli donanım desteği, A.7.4 denetleme kalıbı ile denetim, A.9 programın mantıksal ve zamanlama takibi, C.3.3 tescillenmiş programlama, C.5.3 arayüz yönetimi, C.4.1 sağlam yazılı programlama dili.

• **Denetleme > Durum İzleme**

– *Amaç:*

Amaçlanan sistem çıkışlarından veya durumlarından sapmaları tespit edilmesi.

– *Açıklama:*

Durum izleme, bir sistem değerinin daha güvenilir, ancak genellikle daha az doğru referans değeriyle karşılaştırıldığında uygun bir aralıkta olup olmadığının kontrol edilmesidir. Referans değeri, çalışma zamanında, sistem giriş değerlerine dayanabilen ve spesifikasyondan önceden bilinmeyen (mantık denetlemesi gibi), fazladan bir bölüm tarafından hesaplanır. İnternet üzerinden zaman senkronize edilmesi gereken ve senkronize zamanın dahili bir saatle karşılaştırılmasını sağlayan sistem örnek verilebilir.

– *Emniyet Yaşam Döngüsü Etkisi:*

Referans değerini sağlayan ek bir eleman uygulanmalıdır. Genel olarak, "Durum İzleme" taktikleri "Mantıksal Denetleme" taktiklerinden sistem

açısından daha ağırdır. "Durum izleme" sistemi rasgele hatalardan korumaktadır.Bununla birlikte, emniyet işlevselliğini izlemek için farklı bir uygulama kullanılıyorsa, sistematik uygulama veya tasarım hataları da tespit edilebilmektedir(IEC 61508-6:D.1.4).

– *İlgili IEC 61508 Yöntemleri:*

IEC 61508-7:A.1.1 çevrimiçi izlemeyle arıza tespiti, A.6.4 izlenen çıktılar, A.8.2 voltaj kontrolü, A.9 zamansal ve mantıksal program izleme, A.12.1 referans sensörü, A.13.1 İzleyicisi.

- **Karşılaştırma**

  – *Amaç:*

  Yedek sistem çıkışlarının tutarsızlıklarının tespiti.

  – *Açıklama:*

  Karşılaştırma taktiği, arızaları tespit etmek için tamamen yedekli alt sistemlerin çıktıları eşitliğini test eder. Karşılaştırma taktiği genellikle yedeklilik taktiğinin kullanımını gerektirir. Adım kilitli modunda çalışan çift çekirdekli bir işlemcide, aynı yazılımı her iki çekirdek üzerinde de çalıştırıp ve çıktılarını her döngüden sonra karşılaştırması bu taktiğe örnektir.

  – *Emniyet Yaşam Döngüsü Etkisi:*

  Alt sistemleri karşılaştırmak için çalışma zamanında kaynak gerektiren ek bir elemanın uygulanması gerekmektedir.

  – *İlgili IEC 61508 Yöntemleri:*

  IEC 61508-7:A.1.3 karşılaştırıcı, A.6.5 giriş karşılaştırıcısı/seçim

2.7.1.3 Başarısızlığı önleme

Başarısızlığı önleme, başarısızlık saptama tarafından tanınan hataların nasıl ele alınacağını açıklar. Maskeleme ve Bariyer taktikleri, arızaların alt sistemler arası yalıtımını ile ilgilenirken Kurtarma taktikleri, arızaların giderilmesi ile ilgilenmektedir. Yedeklilik taktikleri, diğer bazı taktikler için gerekli olan çoklu sistemleri sağlar.

- **Yedeklilik > Farklı Yedeklilik**

  – *Amaç:*

  Spesifikasyon veya uygulamadaki arızaların tespit edilmesi veya maskelenmesi ve rastgele donanım arızalarına izin veren yedek sistemin tanıtılması.

– *Açıklama:*

Spesifikasyona veya uygulama seviyesine farklı yedeklilik uygulanabilir. Uygulama düzeyinde farklı yedeklilik kullanan bir sistemde, yedek bileşenler aynı şartnameden bağımsız olarak geliştirilen farklı uygulamaları kullanır. Spesifikasyon seviyesindeki farklı yedeklilik, bir adım daha ileri gider ve yedek parçalar için gerekli şartnamelerinde farklı ekipler tarafından ayarlanması gerekmektedir.

– *Emniyet Yaşam Döngüsü Etkisi:*

Farklı yedeklilik, sadelik taktiğine son derece aykırıdır, çünkü ek olarak sunulan fazladan sistemler, sistem işlevselliğine katkıda bulunmayan (şartname, uygulama, doğrulama, değiştirme, gibi) çok fazla çaba gerektirmektedir. Yedekli sistemler kullanılıyorsa, emniyet doğrulaması için, bariyer taktiğinin uygulanmasıyla elde edilebilecek sistemlerin birbirinden bağımsız olduğu gösterilmelidir(IEC 61508-1:7.6.2.7). Yedek donanım sistemleri emniyet açısından daha kolay test edilebilir, çünkü donanım arıza toleransı olmayan bir sistem için, emniyet-kritik bir fonksiyon hesaplanmadan önce her seferinde tanımlama denetimleri yapılmalıdır. Bu gereksinim yedek donanım sistemleri için daha esnektir (IEC 61508-2: 7.4.4.1.4, 7.4.4.1.5, 7.4.4.2.1, 7.4.5.3).

– *İlgili IEC 61508 Yöntemleri:*

IEC 61508-7:A.7.6 veri yedekliliği, A.13.2 çoklu aktüatörün çapraz takibi, B.1.4 çeşitli donanımlar, C.4.4 çeşitli programlama.

• **Yedeklilik > Kopya Yedeklilik**

– *Amaç:*

Kazara donanım arızalarını tespit eden veya maskeleyen yedekleme sistemlerini uygulanması.

– *Açıklama:*

Kopya Yedeklilik, aynı uygulamanın fazladan bir sisteme eklenmesi anlamına gelir. Yedekli sistemler aynı işlevselliği destekler, aynı ekipmanı ve aynı yazılım uygulamasını çalıştırılmasıdır. RAID1 veri depolama teknolojisi kopya yedeklilğe örnektir.

– *Emniyet Yaşam Döngüsü Etkisi:*

Kopya yedeklilik, donanım yükleme ve değiştirme için birden fazla çaba gerektirir. Yedekli sistemler kullanılıyorsa, sistemlerin birbirinden bağımsız olduğunu göstermek gerekir, bu da emniyet doğrulaması için bariyer taktikleri uygulanarak gerçekleştirilebilir (IEC 61508-1: 7.6.2.7). Yedek

donanım sistemleri emniyet açısından daha kolay denetlenebilir, çünkü donanım arıza toleransı olmayan bir sistem için, emniyet-kritik bir işlev hesaplanmadan önce her seferinde tanımlama denetimleri yapılmalıdır. Bu gereksinim yedek donanım sistemleri için daha esnektir (IEC 61508-2: 7.4.4.1.4, 7.4.4.1.5, 7.4.4.2.1, 7.4.5.3).

– *İlgili IEC 61508 Yöntemleri:*

IEC 61508-7:A.2.1 Yedek donanım denetimleri, A.2.5 izlenebilir yedeklilik, A.3.5 yazılım ile çift taraflı veri karşılaştırması, A.4.5 kopyalama engeli, A.6.3 çoklu çıkış, A.7.3 tümleşik donanım yedekliliği, A.7.5 aktarım yedekliliği.

- **Kurtarma > Onarım**

  – *Amaç:*

  Arızalı sistemi tamamen işlevsel bir duruma döndürülmesi.

  – *Açıklama:*

  Sistem arızası durumunda, sistemin çalışma durumunu manuel veya otomatik olarak tamamen geri yüklenmesi.

  – *Emniyet Yaşam Döngüsü Etkisi:*

  Fonksiyonel emniyeti sağlayan tüm yedeksiz donanım elemanları için bir onarma veya indirgeme taktiği gereklidir(IEC 61508-2:7.4.8.2). Bununla birlikte, kendi kendini düzeltme sistemleri gibi karmaşık kurtarma sistemleri standart (IEC 61508-3:A.2) tarafından önerilmemektedir ve onaylama işlemlerini daha karmaşık hale getirmektedir.

  – *İlgili IEC 61508 Yöntemleri:*

  IEC 61508-7:C.3.9 hata düzeltmesi, C.3.10 dinamik yeniden yapılandırma.

- **Kurtarma > İndirgeme**

  – *Amaç:*

  İndirgeme, bir sistemin hatasında sistemin hala temel fonksiyonel-emniyetinin sürdürüldüğü, işleme düzeyinin azaldığı bir duruma getirilmesi.

  – *Açıklama:*

  İndirgeme sistemleri temel bir emniyet işlevselliği tanımlar. Sistemler bu emniyet işlevselliğini ve ek kritik olmayan işlevleri sürdürür. Bir hata durumunda, sistem tekrar ana emniyet işlevselliğini indirgediği bir moda geri döner. İndirgeme taktiğinin sıklıkla uygulandığı bir örnek, otomasyon sistemleridir. Bu sistemler emniyet-kritik süreçleri kontrol eder

ve genellikle bu işlemleri bir GUI'de görselleştirir. Sistem çok az kaynağa sahipse (örneğin işlem süresi), sistem GUI hizmetini durdurur ve yalnızca emniyet-kritik işlemlerini kontrol etmek için temel işlevlerine odaklanır.

– *Emniyet Yaşam Döngüsü Etkisi:*

Sistem için indirgeme yöntemleri belirtilmelidir(IEC 61508-2:7.2.3.2) ve fonksiyonel emniyet sağlayan tüm yedeksiz donanım elemanları için bir onarım veya indirgeme taktiği gerekmektedir(IEC 61508-2:7.4.8.2). İndirgeme, emniyet doğrulama çabasını azaltabilir, çünkü sadece indirgeme mekanizması ve çekirdek fonksiyonel emniyetin doğrulanması gerekir.

Ayrıca taktikler, bir hata durumunda iyi tanımlanmış davranışı uygulayabilmek için standardın gerekliliklerine uygundur (IEC 61508-2: 7.2.3.2, IEC 61508-3: 7.2.3.2).

– *İlgili IEC 61508 Yöntemleri:*

IEC 61508-7:A.8 gerilim beslemesi hata işleme, C.3.8 indirgenmiş fonksiyonlar.

IEC 61508-7:A.8 hata işleme, C.3.8 indirgenmiş eylemler.

• **Maskeleme > Oylama**

– *Amaç:*

Arızanın diğer sistemlere yayılmadığından emin olmak için bir alt sistem arızasını maskelemek.

– *Açıklama:*

Oylama başarısızlığı şeffaflaştırır. Taktik, arızayı onarmaya çalışmaz, ancak yedekli alt sistemlerden doğru bir sonuç seçerek arızayı gizler. Çıktı değerlerinin çoğunluğuna karar verir.

– *Emniyet Yaşam Döngüsü Etkisi:*

Oylama yapmak için yedeklilik taktiği kullanılmalı ve seçici unsuru uygulanmalıdır. Operasyon sırasında bir oylama sisteminin alt sistemleri onarılabilir, çünkü bir alt sistem tamir edilirse genel sistem hala çalışabilir(IEC 61508-6:B.3.1). Ancak, oylama sistemleri sonuçları sadece karşılaştıran ve sonuçlardan herhangi biri farklıysa güvenli bir durum sağlayan sistemler kadar güvenli değildir(IEC 61508-6 B.3).

– *İlgili IEC 61508 Yöntemleri:*

IEC 61508-7:A.1.4 seçmen, A.6.5 giriş karşılaştırması/oylama.

- **Maskeleme > Geçersiz Kılma**
  - *Amaç:*

    Bir alt sistemin arızasını maskeleyerek, arızanın diğer sistemlere yayılmamasını sağlama.
  - *Açıklama:*

    Geçersiz kılma taktiği, sistem çıktısını güvenli bir duruma zorlar. Örneğin, kapatıldığında güvenli bir durumda olan bir sistemimiz varsa, sistem çıktısı hakkında şüphelerimiz varsa sistemi kapatmak için Geçersiz kılma taktiklerini uygulayabiliriz (örneğin, bir çıktı geçerliliği kontrolü başarısız olduğunda). Bu senaryoda, sistem çıkışının güvenli çıkış değeri ile geçersiz kılınması, sistemin kullanılabilirliğini azaltır. Kullanılabilirliği azaltmayan ve Oylama taktiği ile yakından ilgili olan Geçersiz kılma taktiğinin diğer bir formu, bir alt sistemi veya bir çıktı durumunu diğerini tercih ederek, yedekli alt sistemlerin çıktısını seçer.
  - *Emniyet Yaşam Döngüsü Etkisi:*

    Tercih edilen bir sistem çıkış durumu tanımlanmalı ve geçersiz kılma mekanizması uygulanmalıdır. Geçersiz kılma sistemlerinin doğrulanması daha kolaydır, çünkü emniyet-arıza ilkesine uyarlar (IEC 61508-1: 7.10.2.6).
  - *İlgili IEC 61508 Yöntemleri:*

    IEC 61508-7:A.1.3 karşılaştırıcısı, A.1.5 boş akım prensibi, A.6.5 girdi karşılaştırma / oylama, A.8.1 emniyet kapamalı aşırı gerilim koruması, A.8.3 emniyet kapamalı ile sistem kapanması.

    IEC 61508-7:A.1.3 kıyaslaştırıcı, A.1.5 serbest akış prensibi, A.6.5 giriş karyaslaması / seçici, A.8.1 emniyetli aşırı gerilim koruması, A.8.3 emniyetli sistem kapatması.
- **Bariyer**
  - *Amaç:*

    Alt sistemin diğer alt sistemlere maruz kalmasından korunması.
  - *Açıklama:*

    Bariyer taktiği, alt sistemler arasındaki istem dışı sızıntılardan korunmak için bir mekanizma sağlar. Bariyer uygulamak için alt sistemler arasındaki ara yüzlerin analiz edilmesi ve belirtilmesi gerekir. Bu ara yüzler çalışma zamanında, zaten mevcut olan güvenilir bir mekanizma olan güvenilir bileşen bariyer tarafından kontrol edilir. Bariyer taktiğine örnek olarak, farklı görevler arasındaki iletişimi kontrol eden ve kısıtlayan

bir bellek koruma birimi verilebilir. Bariyer taktikleri, alt sistemler arasında yanlışlıkla oluşan sızıntılara karşı koruma sağlayan bir mekanizma sağlar. Bariyer uygulamak için alt sistemler arasındaki arayüzler analiz edilmeli ve tanımlanmalıdır. Bu arayüzler çalışma zamanında güvenilir bir bariyer bileşeni, zaten var olan güvenilir bir mekanizma ile izlenir. Bir bariyer taktiği örneği olarak, farklı görevler arasındaki iletişimi izleyen ve kısıtlayan bir bellek koruma birimi verilebilir.

– *Emniyet Yaşam Döngüsü Etkisi:*

  Alt sistemler arasındaki arayüzlerin belirtilmesi gerekir. IEC 61508-3:8.3.1'e göre, emniyetle ilgili olmayan fonksiyonlar, emniyetle ilgili fonksiyonlardan, bariyer taktiği ile ayrılmalıdır. Alt sistemler arasındaki arabirimler belirtilmelidir. IEC 61508-3:8.3.1'e göre, emniyetli olmayan fonksiyonlar bariyer taktikleri kullanılarak emniyetle ilgili fonksiyonlardan ayırt edilmelidir. Ayrıca sistemi yapılandırarak, Sadelik taktiğine yardımcı olunabilir(IEC 61508-2:7.2.2.1). Bariyer, modüler sertifikasyon ve emniyet modifikasyonuna izin verir ve alt sistemlerin birbirini etkilemediği kanıtlanırsa, analizle etkili bir şekilde gösterilmesi gereken doğrulama sürecini azaltabilir (IEC 61508-3:C.8, F).

– *İlgili IEC 61508 Yöntemleri:*

  IEC 61508-7:A.11 enerji hatlarının bilgi hatlarından ayrılması, B.1.3 emniyet fonksiyonlarının emniyet dışı fonksiyonlardan ayrılması, B.3.4 modülerlik, C.2.8 bilginin saklanması / kapsüllemesi, C.2.9 modüler programlama, E.12 modülerleştirme, C.3.11 zamana dayalı mimari.

2.8 IEC61508-3 ile Emniyetli Yazılım Geliştirme

Standartlar her geçen gün önemini arttırmıştır. Her gün üretilen ürünlerin emniyet ihtiyacı, can ve mal güvenliği ihtiyacının artması sonucu bu önem değerini arttırmaktadır. Ancak standartlar karışık ve yanlış anlaşılmaya uygun metinler içermektedir. Standartlara ve sertifikalara uymak, havacılık, ulaştırma, nükleer ve petrokimyasallar gibi emniyet açısından kritik uygulamalar için yazılım geliştirmede en önemli adımdır. Geliştiricinin temel görevi, standartların gerekliliklerini anlamak ve bunları yazılıma çevirmektir. Yanlış anlamalar ve şartların yanlış uygulanması, sistematik başarısızlığın ana nedenlerinden biri olarak kabul edilir. Bu problemi çözmek ve standartlarına uygun bir yazılım oluşturmak için çeşitli kalite modelleri önerilmiştir [7].

Bu modeller, her gereksinim için uygun adımları atmayı ve bu çalışmanın odaklandığı C ++ dili de dahil olmak üzere tüm programlama dillerini desteklemeyi amaçlamaktadır. Bir yazılım ürününü izlemek ve kontrol etmek için yazılım kalite kriterlerinin tanımı ve kullanımı bazı araştırmalarda ele alınmıştır [9, 10].

Bu bölüm, yazılım sertifikası için IEC 61508 fonksiyonel emniyet standardının üçüncü bölümünü ele almaktadır. Bu standart, emniyet-kritik öneme sahip tüm uygulamalar için standarttır ve E/E/PE emniyet sistemleri için fonksiyonel emniyet sürecini gösterir. Tüm endüstri güvenliği standartları nükleer alanda IEC61513, demiryolu alanında IEC62279, EN50126, EN50128, EN50129, makine sektöründe IEC62061, otomotiv sektöründe ISO 26262 ve kontrol sürecinde IEC 61511'e dayanmaktadır. Yukarıda da belirtildiği gibi, çoğu standartta olduğu gibi, bu standart bazı yerlerde anlaşılması ve yorumlanması zor olabilecek metinler içerir. Birçok standardın şartlarının bir açıklama getirmemesi gözlemlenmiştir. Örnek: 7.4.5.3'te "Yazılım modülerlik, test edilebilirlik ve emniyetli modifikasyon yeteneğine sahip olacak biçimde üretilmelidir." ifadesi yazılımın nasıl modülerlik sağlayacağını göstermez, özel kurallar tanımlamaz. Ayrıca, yazılım geliştirme sürecinin ilk aşamasında, test zorluklarını öngörmek ve bu zorlukların farkında olmak için test edilmesi gerektiği de göz önünde bulundurulmalıdır. Örneğin, nihai ürünü, savunmacı programlama gibi emniyet-kritik programlama teknikleriyle test etmek zor olmaktadır.

Bu bölüm ayrıca IEC 61508-3'te tanımlanan kurallara ve kullanımlarına da atıfta bulunur ve SIL3 yazılımı emniyet bütünlüğü standardının şiddetle tavsiye ettiği gereklilikleri vurgulamaktadır.

2.8.1 SIL 3 Gereksinimleri

SIL3 düzeyinde bir yazılım emniyet seviyesini elde etmek için, IEC 61508'in Tablo A.3'teki tekniklerin bazıları önerilmektedir [69]. Bu tablo, Tablo 2.4'de gösterilmektedir.

Tablo 2.4 Desteklenen Araç ve Programlama Dilleri

Teknik / Ölçüm	
1	Uygun programlama dili
2	Güçlü yazılmış programlama dili
3	Dil alt kümesi
4a	Sertifikalı araçlar
4b	"Kullanımda kanıtlanmış" araçlar

Ayrıca IEC61508'de yer alan Tablo A.4'te ise bu teknikler detaylandırılmıştır. Tablo 2.5 bu detaylandırılmış teknikleri barındırmaktadır.

Tablo 2.5 Detaylı Tasarım

Teknik / Ölçüm	
1a	Yapılandırılmış yöntemler
1b	Yarı-biçimsel yöntemler
1c	Biçimsel tasarım ve arındırma yöntemleri
2	Bilgisayar destekli tasarım araçları
3	Savunmaya dayalı (defensive) programlama
4	Modüler yaklaşım
5	Tasarım ve kodlama standartları
6	Yapısal programlama
7	Güvenilir/ doğrulanmış yazılım modülleri ve elemanları kullanma
8	Yazılım emniyet gereksinim şartnamesi ve yazılım tasarımı arasında ileri izlenebilirlik

2.8.1.1 Gereksinimlerin C++ ile gerçeklenmesi

Kodların C++ ile emniyet-kritik uygulama geliştirilmesi gerektiğinde; emniyet riskleri sebebiyle kalıplar, çoklu kalıtım, yeniden yorumlama ve sürekli yönetim istisnaları, çalıştırma zaman türü bilgisi(RTTI) gibi C++ programlama dili işlevleri devre dışı bırakılmalıdır [17]. Ham bir komut dosyasının (.cpp veya .h) programlama dilini makine diline dönüştürmek için sertifikalı veya kanıtlanmış araçlara sahip olmanısı gerekmektedir. Bu kanıtlanmış araçlardan biri IAR'dır [14].

Tablo 2.5'deki tasarım kuralları, kaynak kodunun nasıl emniyetli yapıldığını açıklar. Örneğin, otomotiv sektöründeki emniyet-kritik uygulamalar için bir kod geliştirilecekse, MISRA tarafından C/C++ programlama dili için getirilen yazılım geliştirme kuralları izlenmelidir. Başka bir deyişle, sadece IEC 61508-3'e değil, aynı zamanda MISRA'ya uymak gerekmektedir. MISRA, sadece otomotiv sektöründe değil, havacılık, iletişim, tıp elektroniği, savunma, demiryolu vb. olmak üzere birçok sektör için kod geliştirenler tarafından yaygın olarak kabul edilen bir model haline gelmiştir [70]. Bu standart sürekli yazılımın modüler yapıda tasarlanması gerektiğini vurgular. Modülerlik nedeniyle kod karmaşıklığını indirgemektedir. Elbette modüllerin giriş ve çıkışları kontrol edilmelidir. Taşma ve/veya tanımsız kalmalardan kaçınılmalıdır. Bir modül sertifika ile onaylanmışsa, gereksinimlerin değişmediği çeşitli emniyet-kritik uygulamalarda kullanılabilmektedir.

1. Yapısal yöntemler ve Yarı resmi yöntemler

   Standart, SIL3 seviyesinde emniyet-kritik uygulamada yarı resmi yöntemler kullanılamadığında, resmi yöntemlerin kullanılması gerektiği belirtilmektedir. C++ 'da emniyet-kritik yazılım geliştirmek için yarı resmi yöntemler uygulanabilir. Yarı resmi yöntemler, yazılım gereksinimlerinden etkin kod almak için şablonlar veya çizelgeler kullanır. UML diyagramlarının kullanılması, gereksinimlerin yerine getirilmesini ve emniyet-kritik yazılımına dönüştürülmesini kolaylaştırmaktadır.

2. Savunmaya dayalı programlama ve modülerlik

   Savunmaya dayalı programlama tekniği, yazılım tasarım döngüsünün içerisinde yazılım hatalarını erken teşhisi için kullanılan bir programlama tekniğidir. Savunmaya dayalı olmayan programlamaya örnek Şekil 2.9 verilmiştir.

```
1  int i;
2   while(i != 20)
3   {
4      cout<<"i değişkeni= "<<i<< endl;i++;
5   }
```

Şekil 2.9 Savunmaya dayalı olmayan programlama örneği

Savunmaya dayalı programlamaya örnek olarak Şekil 2.10 kod bloğu verilebilir.

```
1  int i (0);
2  while (i < 20)
3  {
4      cout << "i değişkeni ="<<i<< endl; i++;
5  }
```

Şekil 2.10 Savunmaya dayalı programlama örneği

Yukarıdaki örneklerde savunmaya dayalı ve dayalı olmayan programlama arasındaki fark gösterilmektedir.

Savunmaya dayalı olmayan programlamada *i* değişkenine ilk tanımlaması yapılmamıştır. Ayrıca *while* döngüsünde bellekteki olası bir bozulma döngüyü sonsuz döngü haline dönüştürebilir.

Yukarıdaki örneklerle, savunmaya dayalı ve savunmaya dayalı olmayan programlama örnekleri ile iki teknik arasındaki farka dikkat çekilmektedir.

Savunmaya dayalı programlamada, aynı kod bu tekniğe uygun olarak yazılmıştır. Bu tekniğe göre, düzenli aralıklarla giriş/çıkış işaretlerini ve kayıtları güncelleme, aynı girişin birkaç kez okunup ortalamasını alma, kullanılmayan

belleği bir komutla doldurma ve program sayacının bilinen en son duruma sıfırlanması gibi teknikler bulunmaktadır. Sistemin durmasını engellemek için sertifikalı bir durum izleyici kullanılabilir. Donanım birimlerinden biri olan tarayıcıyı örnek olarak verirsek; tarayıcıyla iletişim kurmak için, tarayıcıya iletişim modülü C++ ile yazılan kod Şekil 2.11 gösterilmiştir.

```
1  class tarayici
2  {
3      public:
4      Init_tarayici()
5      Write_Data();
6      Read_Data();
7          ...
8      private:
9          ...
10 }
```

Şekil 2.11 Savunmaya dayalı olmayan programlamada tarayıcı iletişim modülü

Yukarıdaki kod bloğu, tarayıcı sınıfı tanımlanmış olan tarayıcıyla iletişim için bir koddur.

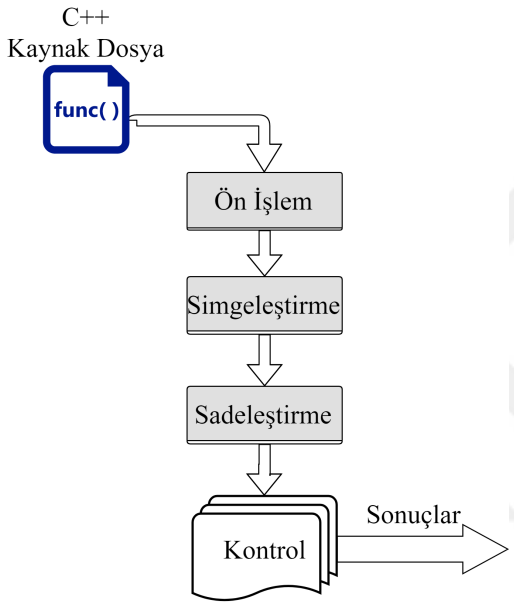
Tek bir tarayıcının olduğunu varsayarsak, ancak burada iki nesne başlatılmıştır. Böylece nesneler aynı anda okunabilir ve yazılabilir. Tarayıcı belirtilen komutlara doğru şekilde yanıt veremez. Bu sorunun basit çözümü, yalnızca bir tarayıcı nesneye sahip olunmasıdır. Bu sorun, UML diyagramlarını yarı resmi bir yöntem olarak tekil bir model olarak tanımlanarak Şekil 2.12 gibi çözülmelidir [18].

```
1  class tarayici: public Devices
2  {
3      public:
4          static tarayici& Instance(void);
5      private:
6          tarayici(void);
7          tarayici (tarayici const& );
8          tarayici& operator =(tarayici const&);
9          ~ tarayici();
10 }
```

Şekil 2.12 Savunmaya dayalı programlamada tarayıcı iletişim modülü

Yukarıdaki kod bloğu, kullanıcının birden çok nesne oluşturmasını önlemektedir. Varsayılan yapıcı yöntemi, kopya kurucu yöntemi ve kopya ataması işlevi, kullanıcının yöntemlere erişmesini önleyerek özel olarak tanımlanmaktadır. Bir tarayıcı nesnesini tanımlamanın tek yolu "Instance" yöntemini kullanmaktır. "Instance", birden çok nesne oluşturulmasını önleyen statik bir yöntemdir. Ek

olarak, tarayıcı sınıfı bir aygıt alt sınıfıdır. Böylece yeni cihazlar kaydedebilir ve silebilir. Kendi standartlarına göre kullanılması gereken programlama dillerine ek olarak, emniyet-kritik yazılım kodları da uygun standartlara yazılmalıdır. MISRA'nın bu konuda birçok kuralı bulunmaktadır, bu nedenle standartlara ve standarda uyumlarını kontrol etmek için ayrı bir yazılım gerekir [15]. "PC-Lint for C/C++" bu yazılımlardan bir tanesidir [16]. Tüm statik kod analiz edici programların genel mantığı, Şekil 2.13'de ifade edilmektedir. Böylece kodun hataları, eksiklikleri ve yanlışlıkları kontrol edilmektedir. Ayrıca, belirli PC-Lint hata mesajlarını göz ardı edilebilmesi seçeneği mevcuttur.



Şekil 2.13 Statik örnek kod

C++ güçlü bir programlama dili olduğundan, PC-Lint "typedef" komutunun kötüye kullanımını algılamaktadır. C++ statik kod analizi örneği, Şekil 2.14 verilmiştir.

```
1  class Base {
2      char* mp;
3      int mCount;
4      public:
5          Base(int aCount = 0) { mCount = aCount;
6          if(mCount > 0) mp = new char(mCount]; }
7          void printRunType() { cout « "Base"; }
8          -Base() { delete mp; }
9      };
10 class Derived : public Base {
11     public:
12         void printRunType() { cout « "Derived"; }
13     };
```

Şekil 2.14 C++ statik kod analizi

PC-Lint'in analizi ise Şekil 2.15 verilmiştir.

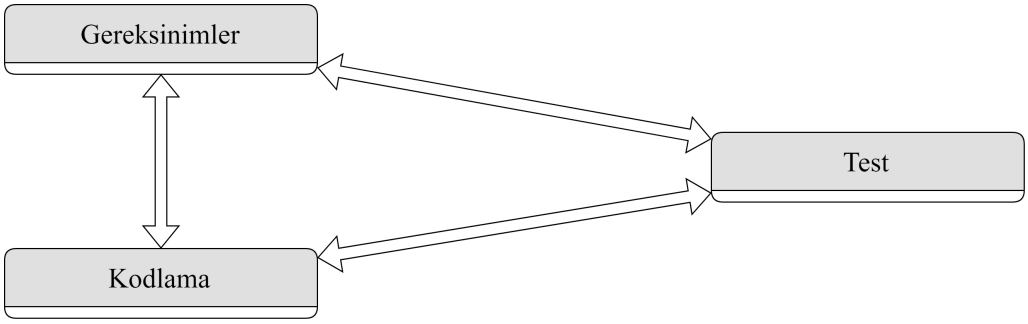
```
1  file.cpp:5 I 1732 new in constructor for class Base which has no assignment operator
2  file.cpp:5 I 1733 new in constructor for class Base which has no copy constructor
3  file.cpp:5 I 737 Loss of sign in promotion from int to unsigned int
4  file.cpp:5 W 1541 Member Base::mp (line 4) possibly not initialized by constructor
5  file.cpp:7 W 424 Inappropriate deallocation (delete) for 'new[]a data.
6  file.cpp:11 W 1511 Member hides non-virtual member Base: :pnntRunType(void) (line 8)
7  file.cpp:12 W 1569 base class destructor for class 'Base' is not virtual
```

Şekil 2.15 PC-Lint kod analizi

Ek olarak, PC-Lint C++ yerel değişkenlerinin aldığı değerleri izler ve muhtemel sıfıra bölünme hatalarını tespit eder.

Nesne yönelimli bir programlama dili olan C++ 'da, iki modülü birbirinden ayırmak için, sınıflar ve ad alanları kullanılmalıdır. Emniyet-kritik yazılımında, sistemde bulunan tüm güvenli donanım ürün sürücüleri olmalıdır. Bütün dijital sinyaller farklı bir alan adı içinde barındırılmalıdır. Emniyet-kritik yazılımı modüler olmalı, her modül kendi emniyet fonksiyonuna sahip olmalıdır. Kullanıcı, bu bölümlenmiş yapıdan kullanılacak modülü seçebilir. Ek olarak, modüler yaklaşım, modülün gelecekteki genişlemesinin ön hazırlığını oluşturmaktadır.

Standardın anahtar şartlarından biri, gelişmiş izlenebilirlik ihtiyacıdır. Emniyet-kritik yazılım için tipik geliştirme döngüsü istemlerle başlar. Yazarlar ve kod geliştiriciler arasında isterleri yanlış anlama sorunu yaygındır. Test personeli derhal kodu test etmelidir. Özel yazılım araçlarıyla daha fazla izlenebilirlik sağlanabilir. Gereksinimler değişirse, bu durum test personeline ve geliştiriciye bir geri bildirim sistemi aracılığıyla iletilmelidir. Bu ilke Şekil 2.16 gösterilmiştir. İstek yazanlar, isteklerini geliştiriciye ve test personeline bildirmelidir. Günümüzde yazılım araçları bu görevi yerine getirmektedir.



Şekil 2.16 İleri izlenebilirlik

3. Dinamik analiz

   Emniyet-kritik yazılım titizlikle test edilmeli ve belgelendirilmelidir. Test şartname belgeleri olarak adlandırılan bu belgeler, bir test gereksinimleri yol haritası ve kod entegrasyonu içermelidir. Burada, C++ programlama

dilinin sonsuz döngüler, işaretçiler gibi kritik noktalarda dikkatli olunmalıdır. Ayrıca, yazılım testi bileşen testiyle bitmemekte, değişken kapsam kriterleri kullanılmalıdır. Test edenler kodu değiştirmez ve geliştiriciden değiştirmesini ister. Bu işlem pahalıdır, çünkü yazılım geliştirme döngüsü için çok geç kalınmıştır. Emniyet-kritik yazılımının test edilebilirliğini geliştirmek için geliştirici, her sınıfın özellikleri için "Get" ve "Set" yöntemlerini tanımlaması gerekmektedir. "Kapsülleme", bu yöntemlerin kullanılmasından etkilenmemektedir.

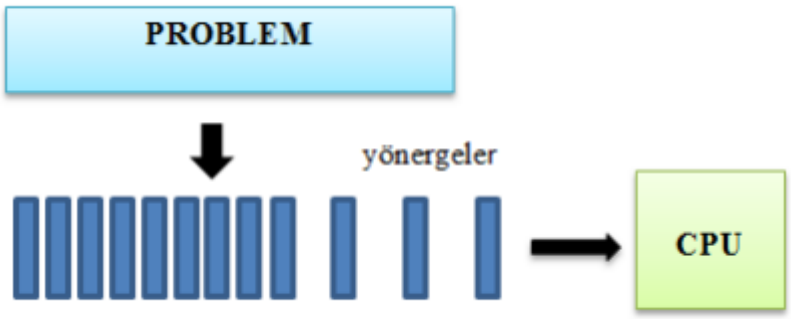
Emniyet-kritik yazılımının standartlar ışığında geliştirilmesi ve entegrasyonu oldukça zordur. Geliştirici genellikle emniyet isterlerini yanlış anlamaktadır. Sonuç olarak, farklı yazılım kalitesi modelleri oluşturmak zorunda kalınmıştır. Bu bölüm, IEC 61508-3 ve SIL3 emniyet-kritik yazılımının gerekliliklerini ele almıştır. Geliştirme desteği olarak IAR derleyicisi ve üretilen C++ kodunun etkinliğini kontrol etmek için "PC-Lint" ve "MISRA C++" kullanılmıştır. Tekil modelin entegrasyonu için basit kod örnekleri sunulmuştur.

2.9 Senkron/Asenkron Programlama ve SCADE

Bilişim sektörünün geldiği nokta ve buna bağlı olarak ihtiyaçların her geçen gün hızla artması birçok problemi de beraberinde getirmektedir. Bilgi teknolojileri sektörü de bu konuda pastadan en fazla dilimi alarak ilk sıraya oturmaktadır. Günümüzde gerek kurumsal gerekse akademik çalışmalarda kullanılan veri boyutlarının artması son kullanıcılardaki performans olgusunu ilk sıraya yerleştirmiştir. Artık geliştirilen bir sistemin kararlı ve hatasız çalışmasının yanında en kısa sürede en iyi sonucu elde etmek de isteklerin başında gelmektedir. Hız konusu, üretim sektörü, mühendislik hesapları, askeri simülasyon projeleri ve hava tahmin hesaplamaları için hayati öneme sahiptir. Paralel hesaplamanın kullanıldığı birçok alan vardır. Bunlara örnek olarak:

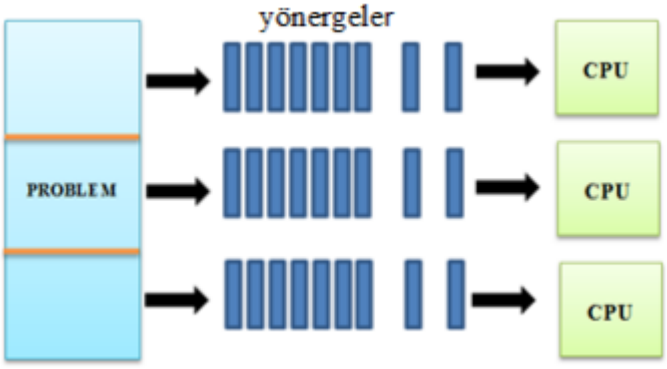
- Hava tahmini
- Fizik-Nükleer hesaplamalar, Parçacıklar, Yüksek Basınç
- Uzay Kara-deliklerin modellenmesi
- Askeri Simülasyonlar
- Kimya-Moleküller
- Elektrik-Elektronik-Devre Tasarımı, Mikro-elektronik
- Veri tabanları-veri madenciliği alanları verilebilir.

Gündelik hayatta kullanılan yazılımların çok büyük bir çoğunluğu seri olarak çalışmaktadır. Bu yazılımlar, bir bilgisayarda birden çok işlemci olsa bile tek işlemciyi kullanırlar. Seri sistemlerde her işlem bir diğerinin ardından gerçekleştirilir. Yani bir işlem süresinde sadece bir işlem gerçekleştirilmektedir. Şekil 2.17'de yer alan görsel bu durumu özetlemektedir.



Şekil 2.17 Seri hesaplama

Paralel hesaplamada ise mevcut hesaplama kaynaklarının yani bir bilgisayarda yer alan tüm işlemcilerin aynı anda kullanılabilmesi söz konusudur. Paralel sistemlerde bir problem için geliştirilen yönergeler eş zamanlı olarak farklı işlemciler üzerinde çalıştırılmaktadır. Şekil 2.18'de yer alan görsel bu durumu özetlemektedir.



Şekil 2.18 Paralel hesaplama

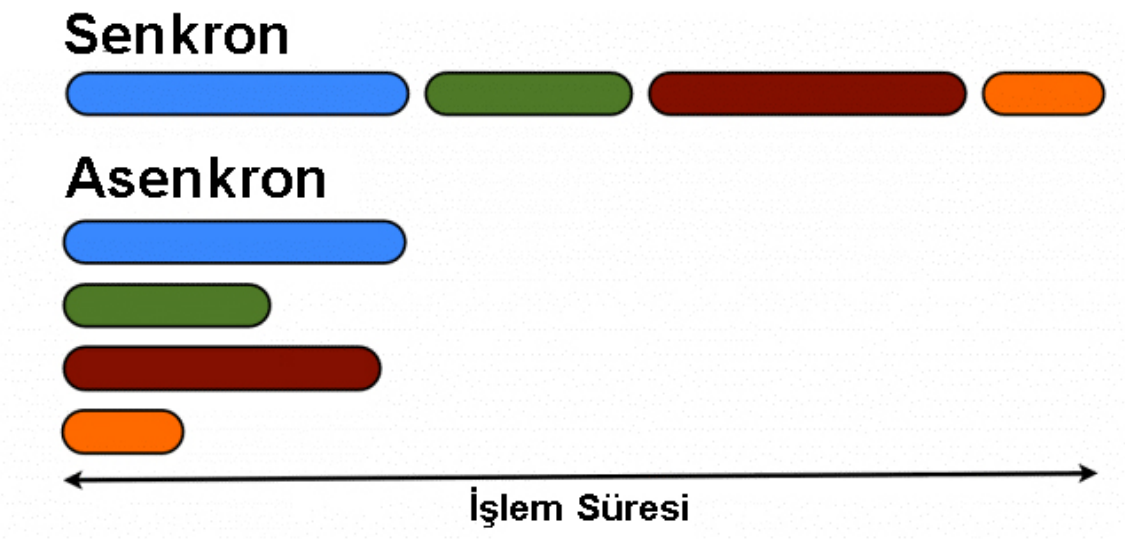
Paralel hesaplamadaki en önemli kavramlar adımlar ve süreçlerdir. Süreç, çalışan bir sistem üzerinde koşan programın kendisidir. Her süreç belirli bir adres uzayına sahiptir. Farklı süreçler farklı adres blokları üzerinde yürütülmekte ve sistem kaynaklarını kullanabilmektedirler. Adımlar ise süreçler içerisinde yer alırlar ve yer aldıkları sürecin adres havuzu içerisinde hareket ederler. Adımlar adres uzayı, bellek ve açık dosyalar gibi sistem üzerindeki kaynakları kullanabilirler.

2.9.1 Senkron ve Asenkron Programlama

Senkron ve asenkron programlama uygulanacak olan kod parçacığının eş zamanlı veya eş zamansız koşturulması anlamına gelmektedir. Bu iki programlama tekniği arasında farklar vardır.

Senkron Programlama, her işlemin tamamlanmasını bekler, bundan sonra yalnızca bir sonraki işlemi yürütür. Asenkron Programlama, her işlemin tamamlanmasını beklemek yerine, tüm işlemleri ilk başta yürütür. Her işlemin sonuçları, tüm sonuçlar elde edildikten sonra değerlendirmektedir.

Kullanım olarak eğer farklı zaman aralıklarında 4 adet işleminiz Şekil 2.19'daki gibi olsun. İşleminiz çok yavaş gerçekleştiriliyor ise ağır işlemse, örneğin veritabanınından büyük veri sorgulamak olsun, asenkron programlama ile arka planda gerçekleştirilmesi sağlanabilir.

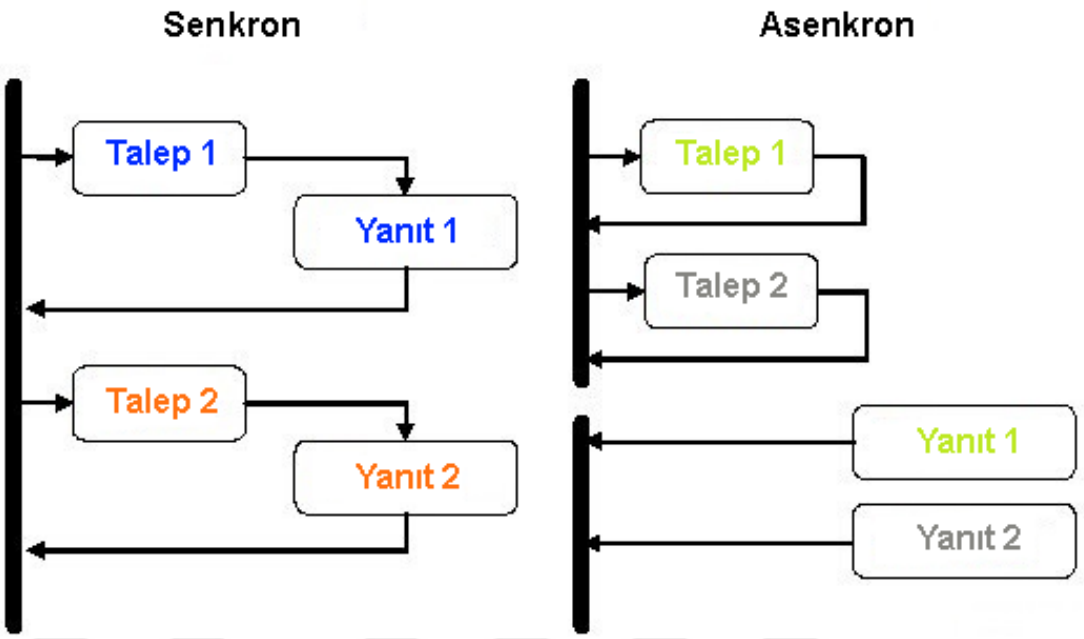


Şekil 2.19 Senkron ve asenkron programlama işleme örneği

Asenkron şekilde, bazı ilerleme göstergelerini kullanıcıya gösterebilirken, arka planda daha yavaş biçimde işlem devam edecektir. Bu GUI uygulamaları için ideal bir senaryodur. Şekil 2.20'de görüldüğü gibi talepler(Request) sonrası oluşan yanıtlar(Response) beklenmeden program akışına devam ediyorsa Asenkron programlama eğer cevaplar beklenerek bir sonraki adıma geçiliyorsa senkron programlama ismini almaktadır.

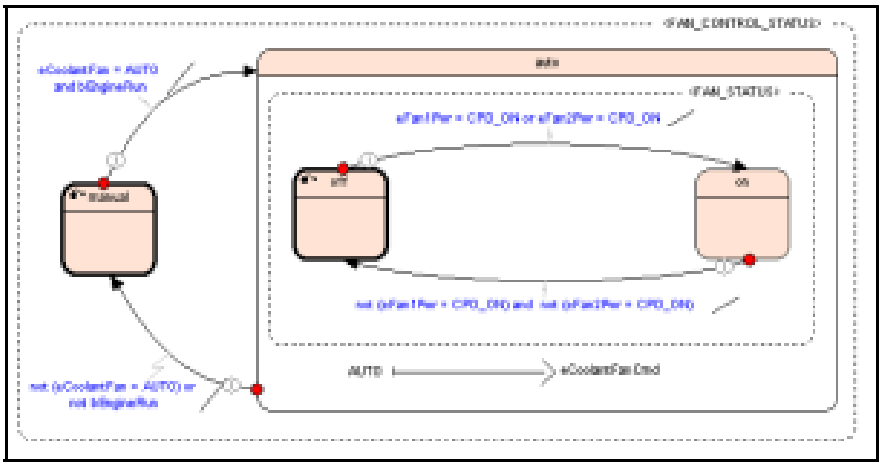
2.9.2 Scade

N-versiyon Programlama ve IEC 61508 standardı araştırmacıları emniyet-kritik uygulama geliştirme aşamasında SCADE programına itmektedir. SCADE(Safety-Critical Application Development Environment) programı Esterel

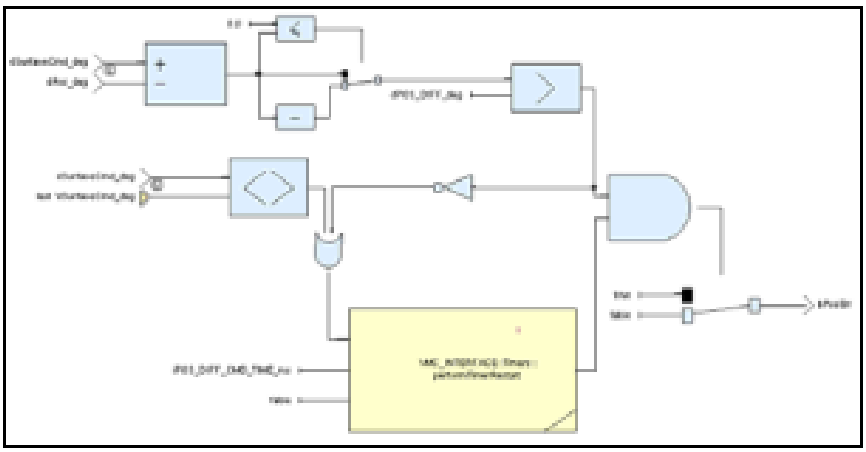


Şekil 2.20 Senkron ve asenkron programlama çalışma prensibi

Technologies'in bir ürünüdür. Program aviyonik, ulaştırma ve enerji endüstrileri için emniyet-kritik olan entegre yazılımların geliştirilmesinde yaygın olarak kullanılan model tabanlı bir yazılım geliştirme ortamıdır. SCADE aracıyla geliştirilen yazılım bileşenleri için yazılımın sağlaması gereken durumlar ve durumlar arasındaki geçiş koşulları durum makineleri üzerinden modellenmekte ve istenilen işlemler için fonksiyonlar ise blok diyagramlar aracılığı ile oluşturulur. Bir durum makinesi örneği Şekil 2.21'de, örnek bir blok diyagram ise Şekil 2.22'de gösterilmiştir.



Şekil 2.21 Örnek durum diyagramı



Şekil 2.22 Örnek blok diyagramı

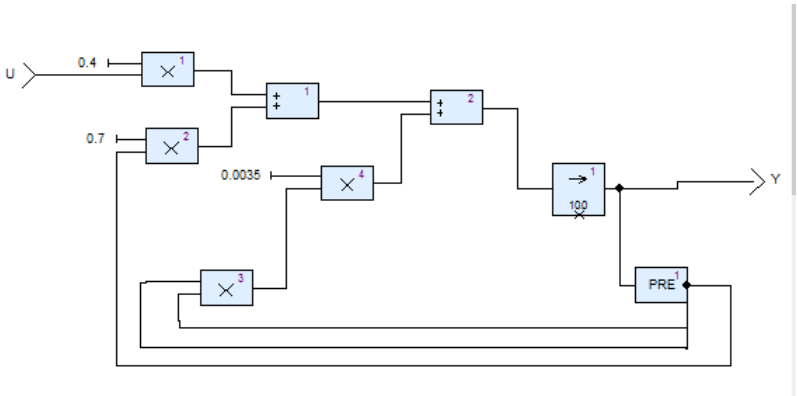
Geliştiriciler, Scade tarafından oluşturulan kodun güvenirliği ile, uzun sürede çözülmeye çalışılan programlama hatalarını düzeltmek ve geleneksel programlama yöntemlerindeki programlama dilini etkin bir şekilde kullanmak için çok fazla enerji harcamamaktadırlar.

2.9.2.1 Scade İle Emniyet-Kritik Bir Kontrolcü Tasarım

Scade Suite çalışma ortamını daha iyi tanıyabilmek için bir ısıtıcı tarafından kontrol edilen sistemi incelenmiştir. Sistemin r referans sıcaklığına cevabı (2.15)'daki gibidir:

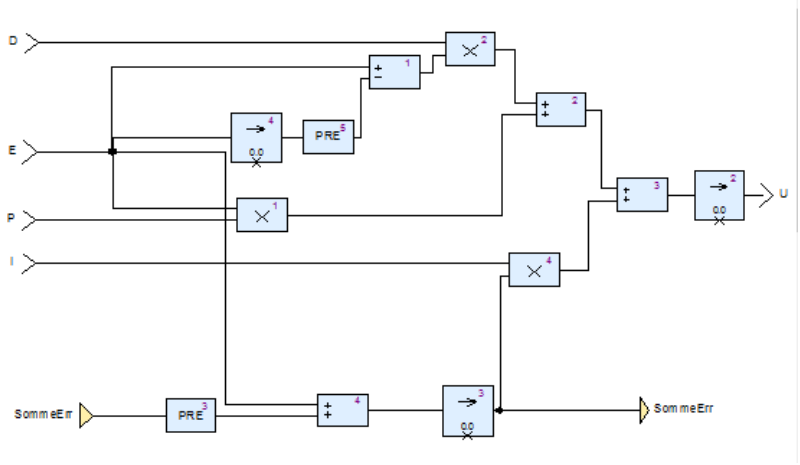
$$u_t = 0,7y_t - 0,0035y_t^2 + 0,4u_t \tag{2.15}$$

Scade Suite içerisinde yeni bir proje açıldıktan sonra operatör eklenmelidir. İlk operatör ısıtıcı operandınız olan "kazan" eklenmeli ardından kazan içerisinde u(giriş) ve y(çıkış) eklenerek Şekil 2.23'te kazanın matematiksel modeli oluşturulmuştur.



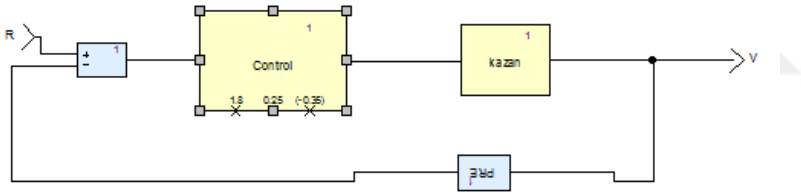
Şekil 2.23 Sistem Diyagramı

Ardından PID kontrolcü için, hata için e, geçmiş hata değeri için $somerr$, PID Katsayıları ve U çıkış olarak eklenmiş ve Şekil 2.24'de belirtilen bağlantılar yapılmıştır.



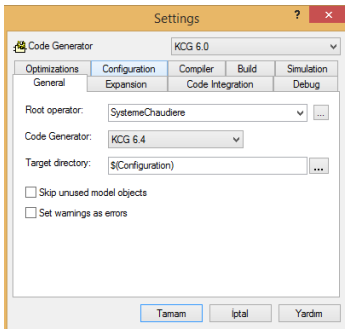
Şekil 2.24 PID Kontrolör Tasarımı

Böylece sistem ve kontrolcümüzü oluşturulmuştur. Bu iki operandı birbirine bağlamak ve gözlemleyebilmek için yeni bir operatör içerinde oluşturulan her iki operatör sürüklenerek eklenmiştir. R giriş ve Y çıkış eklenerek Şekil 2.25 oluşturulmuştur.



Şekil 2.25 Kapalı Çevrim Tasarım

Kod üreticisi ayarlanarak simule edilmiştir. Bu ekran Şekil 2.26'de gösterilmiştir. Bu işlem sonunda derlenmiş olan sistemimizin C kodunda emniyet-kritik kaynak kodu proje klasörü içerisinde oluşturulmuştur.



Şekil 2.26 Kod üreticisi

3
N-VERSİYON PROGRAMLAMA

Emniyet-kritik uygulamalar standartlara uygun şekilde oluşturmak için önceki bölümde anlatılan IEC 61508:3 standardına göre NVP (N Versiyon Programlama) önerilen yöntemlerin arasında yer almaktadır.

Bu yöntem ilk Avizienis tarafından öne sürülmüştür [20]. N-versiyon programlama (veya N-versiyon yazılımı) fikrine göre, N farklı çalışma grubu tarafından N farklı yazılım versiyonu (veya modülleri) geliştirilmelidir. Bu versiyonların giriş çıkış özellikleri aynıdır ve işlevsel olarak eşdeğerdir, aynı görevi çözerler. Ayrıca, çalışma gruplarına bağlı olarak görev çözme yöntemleri farklı olabilir.

Literatürde N-versiyon programlama tekniğinin başarılı uygulamalar arasında uzay [32, 33], demiryolu sinyalizasyon sistemleri [23], mesaj iletim sistemleri [24], e-oylama [34], intihal tespit algoritmaları [25], ağ hizmetleri [26, 29] yer almaktadır.

Ayrıca N versiyon Programlama tekniğindeki yazılım gereksinimleri literatürde [35–38] anlatılmıştır. Bu çalışmalar ışık tutmuştur ki, geliştirilecek yazılımların farklı çalışma grupları tarafından, farklı yazılım dilleri kullanılarak, farklı yazılım geliştirme ortamlarında çalışmaları gerektiği sonuçları ortaya çıkmıştır.

N-versiyon yaklaşımına göre, versiyonlar çeşitlilik şartına uymalıdır [36, 37, 39]. İdeal olarak, versiyonlar farklı programcılar tarafından, farklı programlama dilleri kullanılarak ve farklı çerçeveler kullanılarak geliştirilmelidir. Aynı görevi çözmek için farklı yöntem ve algoritmaların gerçekleştirilmesi, bağımlı arızalardan kaçınılmasına olanak tanıyacaktır. Başka bir deyişle, N versiyon programlama yöntemi yazılım tasarım hatalarını aşmaz ancak tüm sistemi tehlikeli duruma düşürmeye karşı korur.

N versiyon programlama yöntemi, işlevsel olarak eşdeğer modüllerde hataların oluşmasının çeşitli noktalarda gerçekleşebileceğini önermektedir, bu nedenle hatalar tespit edilebilir ve gerçek sonuç alınabilir [40]. Yazılım versiyonlarının sayısı ne kadar fazla olursa, N-versiyon yazılımının doğru şekilde çalıştırılma ihtimali de artmaktadır. Bu yöntemin en büyük avantajı yazılım hata toleransını sağlamaktır

[31]. Herhangi bir versiyonun başarısız olması durumunda bile, çalışan versiyonların geri kalanı sonuçlarını üretecek ve sistem çalışmaya devam edecektir. Bu durumda, sistemin olağan işleyişi, yazılım geliştirme sırasında üretilen ve test aşamasında tanımlanmayan hatalara karşı sigortalıdır [41, 42]. N versiyon programlama yöntemi, versiyonlardan birinin hatalarının sistemin çalışmasını bozmamasına ve güvenirliğin sıkı gerekliliklere sahip olmasını sağlar. Bu bağlamda, bu yaklaşım aktif olarak mühendislik uygulamasında kullanılmaktadır ve yazılım kontrol sistemlerinin ve bilgi işleminin geliştirilmesinde yararlı olduğu kanıtlanmıştır. Mevcut test yöntemleri ve program doğruluğu ispatlaması ile birlikte N-versiyon programlama yöntemini kullanmak, yüksek seviyede yazılım güvenirliğini garanti eder [41, 43, 44].

N-versiyon programlamanın potansiyel uygulama alanlarını, nükleer sanayi, su altı ve yer altı araştırmaları, kimya endüstrisi, demiryolu kilitleme sistemleri, elektronik oylama, ileti gönderme sistemleri, web hizmetleri, intihal tespiti ve sıfır gün istismarların tespiti gibi sayabiliriz [20, 23–30]. Daha önce de belirtildiği gibi N versiyon programlama yöntemi, aynı yazılım modülünün çeşitli versiyonlarının bağımsız ve paralel yürütülmesini içerir. Modülün tüm versiyonları aynı veriyi alır [31]. Tasarım çeşitliliği nedeniyle versiyonların farklı hesaplama sonuçları üreteceği ihtimali olduğundan, doğru ve yanlış sonuçların belirlenmesinde ortaya çıkan bir diğer sorun hangisinin geçerli olacağıdır.

3.1 NVP Seçiciler

N farklı yazılım versiyonun çıktıları, karşılaştırma için seçici(voter) olarak bilinen başka bir birime gönderilir. Seçici, N farklı yazılım versiyonun çıktılarını alır ve seçici algoritmasına bağlı olarak nihai bir karar verir. Seçici algoritmaları, N versiyon yazılımında N farklı versiyon sonuçlarını değerlendirmek için kullanılır. Hangi sonucun doğru olduğunu belirlerler ve bir sonraki modülün girişine iletirler veya kullanıcıya geri dönerler. Dolayısıyla, seçici algoritması, N-versiyon yazılımının en önemli bileşenidir çünkü tüm hesaplama prosesinin çıktısını belirler.

Seçici şemaları ve orijinal verilerin gereklilikleri bakımından farklı olan seçici için pek çok algoritma bulunmaktadır [45–50]. Her bir algoritmanın kendi avantajları ve dezavantajları vardır ve başarılı bir şekilde uygulanması için belirli koşullar mevcuttur. Ayrıca, sürümlerin verdiği veri setine bağlı olarak bazı algoritmalar etkisiz olabilir veya çıktıların doğruluğu hakkında bir karar verme fırsatı bulamayabilir. Seçici algoritmalarının tümünde ortak özellikler içeren bir dizi algoritma vardır. Seçici algoritmaların sınıflandırılması, çıkışların karşılaştırılmasına ve benzerliklerine yönelik 2 ana grupta incelenebilir.

3.1.1 Karşılaştırmaya Dayalı Seçici Algoritmaları

Bu algoritma sınıfını incelerken, uygulamanın temel özellikleri aşağıdaki şekilde kaydedilmiştir:

1. Algoritmaların uygulanmasının kalbi, özdeş ve yanıltıcı cevapların olasılığının ihmal edilebilir olduğu varsayımını ortaya koymaktadır [47]. Aynı ve yanlış cevaplar, birden fazla versiyonun hatalı sonuçlar doğurduğu durumlarda verilir.
2. Sürümler tarafından döndürülen sonuçların tamamı, "doğru" alt kümesine ve "yanlış" çıktıların bir alt kümesine ayrılmıştır. "Doğru" cevaplar setinin seçilmesi, sonuçların tümünün karşılaştırılmasına dayanmaktadır.
3. Sürümlerin sonuçlarını karşılaştırırken çıktıların eşdeğerliği kavramı kullanılır. Çıkış değerleri, tolerans değeri olarak adlandırılan sabit sayıdan daha fazla farklı değilse, bir sürümün çıktısı başka bir sürümün çıkışına eşdeğerdir.
4. Kural olarak, eşdeğer çıktılar dizisi doğru sonuç olarak kabul edilir. Bu set, sürümlerin sonuçlarının tümünden seçilir.
5. Sürümlerin doğru sonuç kümesini seçmek, anlaşma matrisini kullanarak veya kabul eden sürümlerin alt kümeleri kullanılarak yapılır.

Çıktı verilerinin karşılaştırılmasına dayanan oy verme algoritmaları, biçimlendirilmiş ve biçimlendirilmemiş oylama algoritmalarından oluşan iki alt sınıfa ayrılabilir.

3.1.2 Çıkış Verilerinin Benzerliğine Dayalı Seçici Algoritmaları

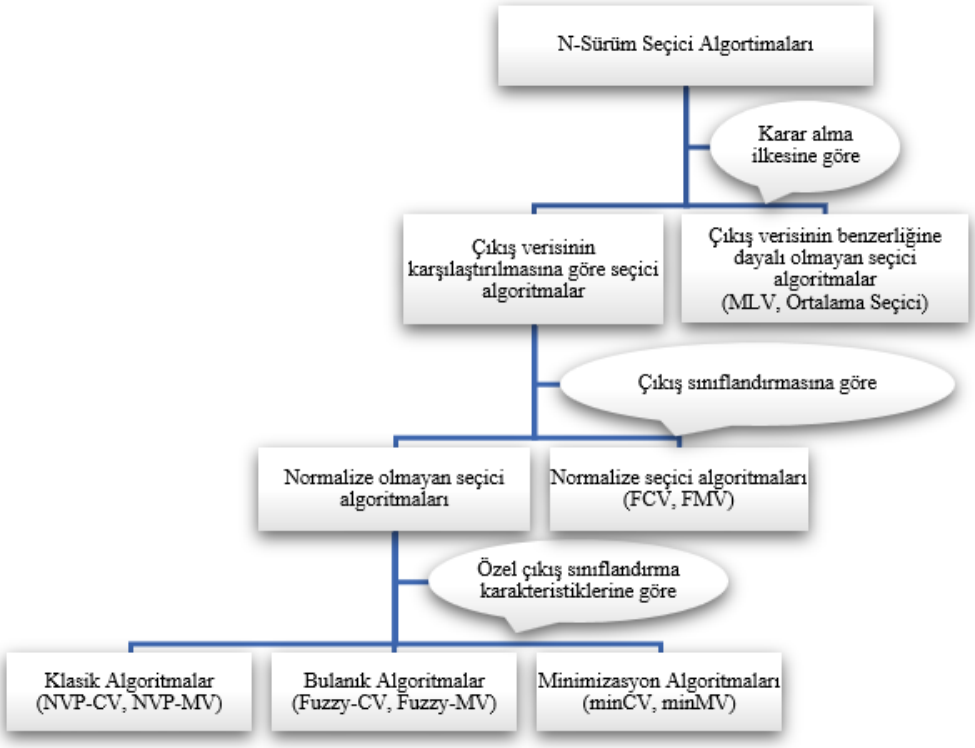
Bu algoritmaların temel karakteristik özelliği, değerlendirmelerinin sonucunun, sürümlerin çıkışlarının nasıl tehlikeye atıldığına bağlı olmamaktadır. Maksimum olasılık seçici algoritmaları (MLV) ve Ortalama seçici algoritmaları bu algoritma sınıfına dâhildir.

3.2 NVP Seçici Algoritma Sınıflandırılması

NVP yazılımında değerlendirme ve doğru sonuç seçimi için aşağıdaki oylama algoritmaları kullanılır [35, 44, 46, 51–61]:

1. Mutlak çoğunluk seçici algoritma (çoğunluk oyuyla N sürüm programlama, NVP-MV)
2. Oy birliği seçici algoritması (oybirliğiyle N-versiyonu programlama, NVP-CV)

3. Bulanık çoğunluk seçici algoritma
4. Bulanık oy birliği seçici algoritma
5. Minimizasyonlu mutlak çoğunluk seçici algoritma (MinMV)
6. Minimizasyonlu oy birliği seçici algoritma (MinCV)
7. Biçimlendirilmiş çoğunluk seçici algoritma (FMV)
8. Biçimlendirilmiş oy birliği seçici algoritma (FCV)
9. Maksimum olasılık seçici algoritma
10. Ortalama seçici algoritma



Şekil 3.1 Nvp seçici yazılımda algoritma sınıflandırılması

N sürüm yazılımında kullanılan seçici algoritmalarının sınıflandırılmasını Şekil 3.1 göstermektedir. Sınıflamanın, oy verme algoritmalarının karar alma ilkesine, çıktı verisinin sınıflandırılmasına ve çıktı veri sınıflandırmasının belirli özelliklerine dayandığını göstermektedir.

3.3 N Versiyon PID kontrolör Tasarımı

N-versiyon programlama yazılımın güvenirliğini garanti altına almak için kullanılan yöntemlerden birisidir. Bir sistemin yedekliliğine ve çeşitliliğine dayanan iki temel stratejiye sahiptir. Emniyet-kritik yazılım geliştirme anlamında bir sistem için tanımlanan n-Versiyon PID kontrolcüler, aktif olan kontrolcünün arızalanması sonucu devre dışı kalmasıyla yeni kontrol kazançlarını seçici aracılığıyla elde ederek sistemin kararlılığını garanti eder. Bu çalışmada model oluşturmak, emniyetli kod üretmek ve simülasyon sonuçlarını görmek için ANSYS SCADE Suite programı kullanılmıştır. Programın genel özellikleri ve yetkinlikleri kısaca anlatılmış, kullanılan operatörler, sabitler detaylı olarak açıklanmıştır. Gömülü sistemlere entegre edilebilecek olan emniyet-kritik C kodu üretilmiştir. Benzetim yapmak amacıyla transfer fonksiyonu olarak 2 serbestlik derecesine sahip bir helikopterin yunuslama açısının transfer fonksiyonu seçilmiştir. 3 farklı PID kontrolcüsü sistemi farklı performanslar ile kararlı yapacak şekilde seçilmiştir.

3.3.1 Giriş

N-versiyon programlama (NVP), yazılım mühendisliğindeki, işlevsel olarak eşdeğer programların aynı başlangıç değerinden bağımsız olarak üretildiği bir yöntem veya işlemdir [21]. N-versiyon programlama kavramı, [22]'de programlama zorluğunun bağımsızlığı, programın iki veya daha fazla versiyonunda ortaya çıkan aynı yazılım hatalarının olasılığını önemli ölçüde azaltacağı varsayımı üzerine getirilmiştir. NVP günümüzde havacılık, ulaşım gibi birçok alandaki yazılımlara uygulanmıştır [22].

ANSYS SCADE (Safety Critical Application Development Environment) gerçek-zamanlı sistemler için hayati tehlike arz eden gömülü yazılım uygulamalarının tasarımı için kullanılan bir tümleşik geliştirme ortamıdır [19].

Model tabanlı tasarım; SCADE Suit Editor kullanıcıya metinsel ve grafiksel olarak modelleme imkânı verir. SCADE Suit sistem isterlerini açık ve esnek bir şekilde yönetmeyi sağlamaktadır. SCADE grafik tasarımı yapmak amacıyla "Display", modellerin dinamik doğrulamasını yapmak için "Test" ve bu uygulamaların yaşam döngüsünü yönetmek için de "Life Cycle" adlı yan programlara sahip model tabanlı programlama aracıdır [19]. SCADE Suite, uçuş kontrolü, otomatik pilot, iniş takımı, kokpit ekranları gibi havacılık uygulamalarında, nükleer enerji santralleri gibi enerji uygulamalarında, raylı sistemlerde ve diğer endüstriyel emniyet-kritik uygulamalarda kullanılmaktadır.

SCADE Suite bir model bazlı yazılım geliştirme aracıdır. SCADE Suite Editor ortamında oluşturulan model SCADE Suite KCG Code Generator'a gönderilerek kod üretilir.

SCADE Suite KCG Code Generator, emniyet standartlarında belirlenen yapıda C ve ADA kodu üretir. Üretilen kodlar Tablo 3.1 'da belirtilen standartlar ile sertifikalandırılabilmektedir.

Tablo 3.1 SCADE Suite ve uyumlu sertifikasyonlar [71]

Standart ve Uyumlu Seviye	**Kullanım Alanı**
DO-178B/C A Seviye	Havacılık ve Savunma
IEC 61508 SIL 3 Seviye	Ulaşım ve Endüstriyel
EN 50128 SIL 3/4 Seviye	Demiryolu
IEC 60880	Nükleer
ISO 26262	Otomotiv

Emniyet-kritik yazılımın tanımı öznel olmasının yanında IEEE tarafından, "kabul edilmez riskler ile sonuçlanan sistemlere kullanılan yazılım olarak tanımlanır. Emniyet kritik yazılım çalışması ya da çalışmaması tehlikeli olacak durumlar için tehlikeli durumdan kurtarmaya ve kazanın ciddiyetini azaltmaya yönelik yazılımları içerir [1]."

DO-172B/C kılavuzu emniyet-kritik aviyonik yazılım sistemlerinin emniyet kriterlerini belirleyen bir kodlama standarttır. Kod okunabilirliği ve sürdürülebilirliği, fonksiyon tasarımı, makroların kullanımı, lokal ve global veri kullanımı, düşük seviye isterler ile kod arasındaki doküman izlenebilirliği, modül ve fonksiyon ya da prosedür adlandırma ve şartlı olarak derlenmiş kod için bir takım sınırlandırmalar bu standardın dahilindedir ve kullanıcıya kılavuzluk yapması adına tanımlanmışlardır [11].

3.3.1.1 Operatörler

SCADE Suit yüksek seviyeli isterler için fonksiyonel birim olan operatörlerin yardımıyla kullanıcının grafiksel olarak tasarım yapmasına imkân verir. Operatörler, ön tanımlı, kullanıcı tanımlı, içe aktarılan ve kütüphane operatörleri olacak şekilde dört çeşittir. Ön tanımlı operatörler temel matematiksel işlemleri, karşılaştırma, mantıksal, yapısal, zaman, seçim, bitsel ve yüksek dereceli operatörleri içeren program içerisinde "shortcuts" bölümünde yer alırlar.

Kullanıcı tanımlı operatörler, diğer kullanıcı tanımlı operatörlerden ve ön tanımlı operatörlerden oluşan veri akış denklemleridir. İçe aktarılan operatörler SCADE dilinin anlamlılığını arttıracak harici *C* kodlarından oluşur. Kütüphane operatörleri ise SCADE'in standart kütüphanelerini içermektedir.

3.3.1.2 Veri Tipleri ve Sabitler

SCADE Suit model içerisinde çeşitli veri tiplerinde işlem yapılmasını sağlar. Gereken durumlara çok fazla veri tipini tanımlamak mümkündür. Bu veri tipleri, ön tanımlı, kullanıcı tanımlı ve içe aktarılan olmak üzere üç çeşittir. Ön tanımlı veri tipleri int, bool, float, char şeklindedir. Kullanıcı tanımlı veri tipleri 3 farklı biçimde ele alınır. Eşit veri tipleri; model okunabilirliğini arttırmak amaçlı kullanılırlar. (Örneğin, Speed=float). Listelenmiş veri tipleri (örneğin, color=enum{blue, green,red}). Yapısal veri tipleri; veri yapıları ve diziler yapısal veri tipidir [19]. Bu çalışmada kullanılan değişkenlerin ya da sabitlerin tümünün veri tipi ön tanımlı olup float64'dür.

İçe aktarılan veri tipleri içe aktarılan operatörlerdeki veri tiplerini tanımlamak için kullanılırlar. Sabitler, sisteme ait sabit değerleri veri tipleriyle birlikte girmek için kullanılmaktadır.

Bir sistem içerisinde sürekli kullanılacak sabit değerler Tablo 3.2'de görüldüğü gibi bu bölümde yazılmaktadır. Bu yazım programın okunabilirliği açısından avantaj sağlamaktadır.

Tablo 3.2 Sabitler

İsim	Veri tipi	Değer
Kd1	float64	1.0
Kd2	float64	5.0
Kd3	float64	3.0
Ki1	float64	20.0
Ki2	float64	250.0
Ki3	float64	250.0
Kp1	float64	120
Kp2	float64	120
Kp3	float64	50
TfDen	float64 ^ 4	[1, (-1.705), 1.098, (-0.256)]
TfNum	float64 ^ 3	[0.002968, (-0.001825), 0.000259]
UpLimInt	float64	1000000.0

3.3.2 N-Version PID Kontrol

3.3.2.1 Çağrı Çizgesi

Oluşturulan model operatörlerinin ağacını çağrı çizgesi ile gösterilebilir. Burada tüm operatörleri içeren "OverallSystem" operatörü, kapalı çevrim sistemi gösteren son operatördür.

1. OverallSystem
    (a) nPID
        i. PIDBlock [3]
            A. linear::Derivative
            B. linear::Gain [3]
            C. linear::IntegrTrapez
        ii. Voter
            A. math::Abs [3]
            B. math::Mean [3]
            C. math::Mean3
    (b) TransferFunction
        i. SumDelay

3.3.2.2 PIDBlock Operatörü

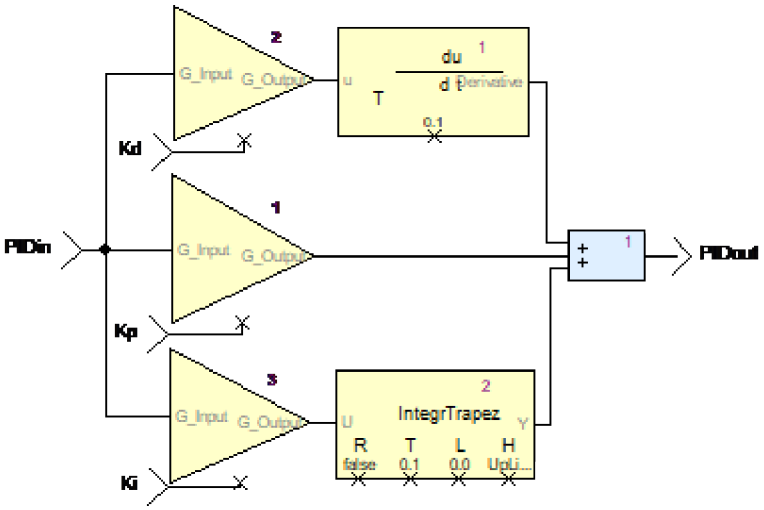
SCADE Suite Editor'ün sahip olduğu kütüphaneler ve ön tanımlı operatörler kullanılarak temel bir PID modeli Şekil 3.2'deki gibi oluşturulmuştur. Burada "$PIDin$" giriş, "$PIDout$" çıkış "Kp", "Kd", "Ki" sabitleri ise "hidden input" olarak ayarlanmıştır. Türev, integral ve kazanç blokları ise "liblinear" kütüphanesi içerisinden alınmıştır. Bu bloklara ait değerler Tablo 3.3 ve Tablo 3.4'de verilmiştir.

Tablo 3.3 Derivative operatörünün girişleri

Sıra	İsim	Değer
1	TimeCycle	0.1

Tablo 3.4 IntegrTrapez operatörünün girişleri

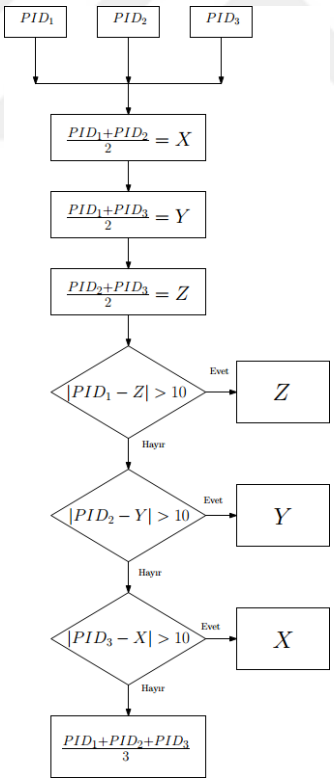
Sıra	İsim	Değer
1	Reset	false
2	TimeCycle	0.1
3	LowLimit	0
4	HiLimit	UpLimInt



Şekil 3.2 PIDBlock operatörünün SCADE suite modeli

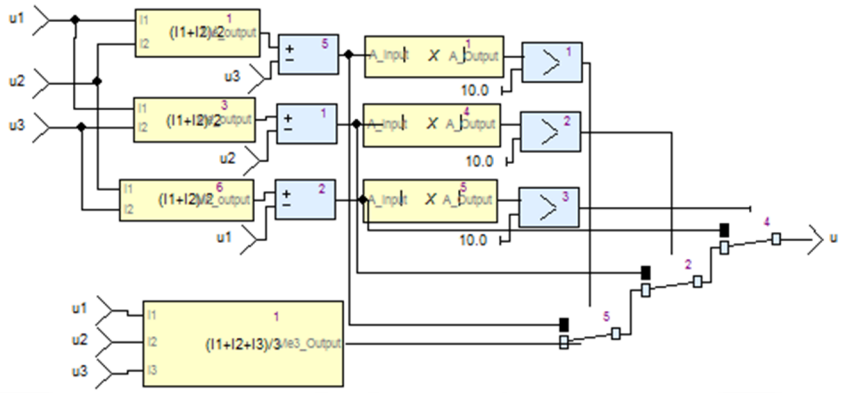
3.3.2.3 *Voter* Operatörü

3 farklı PID bloğundan gelen değerleri arasından Şekil 3.3'de verilen akış diyagramına göre seçim yapıp sisteme ait transfer fonksiyonuna gönderen basit bir algoritma seçici operatör olarak tasarlanmıştır. Seçici *Voter* operatörü bir fonksiyon olarak çalışmıştır.



Şekil 3.3 *Voter* akış diyagramı

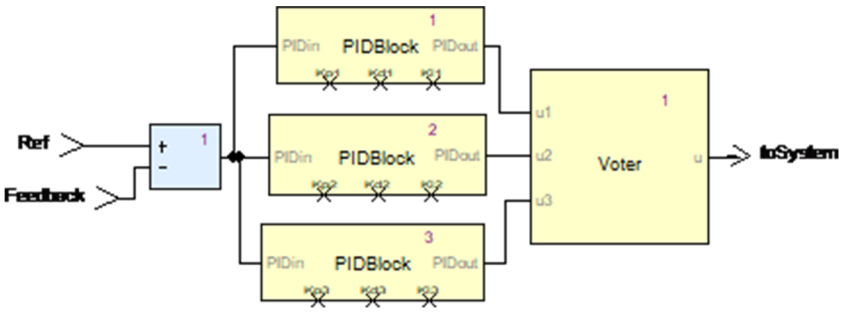
Bu algoritma SCADE Suite Editor ortamında Şekil 3.4'deki gibi modellenmiştir. Burada mutlak değer almak için kullanılan "*Abs*" operatörü ve aritmetik ortalamaları alan "$Mean$" ve "$Mean3$" operatörleri "$libmath$" kütüphanesinden alınmıştır. Bu operatör 3 farklı PID bloğundan aldığı $u1$, $u2$ ve $u3$ girişlerine ve u çıkışına sahiptir.



Şekil 3.4 $Voter$ operatörünün SCADE suite modeli

3.3.2.4 nPID Operatörü

"PIDBlock" operatörünü kullanarak "nPID" operatörü Şekil 3.5'deki gibi oluşturulmuştur. Bu operatör "Ref" referans girişine ve "$Feedback$" negatif geri besleme girişine, "$toSystem$" çıkışına sahiptir.



Şekil 3.5 nPID operatörünün SCADE Suite modeli

Oluşturulan nPID modeli SCADE Suite KCG Code Generator kullanılarak Şekil 3.6 görülen "nPID operatörüne ait üretilen C kodu" üretilmiştir.

```c
#include "kcg_consts.h"
#include "kcg_sensors.h"
#include "nPID.h"
/* nPID/ */
void nPID(inC_nPID *inC, outC_nPID *outC)
{
  /* _L21=(PIDBlock#1)/ */
  PIDBlock(inC->Ref, Kp1, Kd1, Ki1, &outC->Context_PIDBlock_1);
  /* _L25=(PIDBlock#2)/ */
  PIDBlock(inC->Ref, Kp2, Kd2, Ki2, &outC->Context_PIDBlock_2);
  /* _L26=(PIDBlock#3)/ */
  PIDBlock(inC->Ref, Kp3, Kd3, Ki3, &outC->Context_PIDBlock_3);
  outC->toSys = /* _L24=(Voter#1)/ */
    Voter(
      outC->Context_PIDBlock_1.PIDout,
      outC->Context_PIDBlock_2.PIDout,
      outC->Context_PIDBlock_3.PIDout);
}
#ifndef KCG_USER_DEFINED_INIT
void nPID_init(outC_nPID *outC)
{
  outC->toSys = kcg_lit_float64(0.0);
  /* _L26=(PIDBlock#3)/ */ PIDBlock_init(&outC->Context_PIDBlock_3);
  /* _L25=(PIDBlock#2)/ */ PIDBlock_init(&outC->Context_PIDBlock_2);
  /* _L21=(PIDBlock#1)/ */ PIDBlock_init(&outC->Context_PIDBlock_1);
}
#endif /* KCG_USER_DEFINED_INIT */
#ifndef KCG_NO_EXTERN_CALL_TO_RESET
void nPID_reset(outC_nPID *outC)
{
  /* _L26=(PIDBlock#3)/ */ PIDBlock_reset(&outC->Context_PIDBlock_3);
  /* _L25=(PIDBlock#2)/ */ PIDBlock_reset(&outC->Context_PIDBlock_2);
  /* _L21=(PIDBlock#1)/ */ PIDBlock_reset(&outC->Context_PIDBlock_1);
}
#endif /* KCG_NO_EXTERN_CALL_TO_RESET */
```

Şekil 3.6 nPID operatörüne ait üretilen C kodu

3.3.2.5 Transfer Fonksiyonu

Bu test edilecek sistem için "$\sum$" 2 serbestlik derecesine sahip bir helikopterin yunuslama $(pitch, 0)$ hareketini ifade eden transfer fonksiyonu olarak seçilmiştir. [72]'de verilen durum uzay denklemi kullanılarak sistemin transfer fonksiyonu elde edilmiştir. Aynı şekilde bu transfer fonksiyonu için seçilen kontrolcü katsayıları da bu sisteme ait katsayılardır. "In" sistemin girişini, "y" ise sistemin çıkışını ifade etmektedir. "$TfNum$" ve "$TfDen$" transfer fonksiyonuna ait dizileri içeren sabitlerdir. Sistemin ayrık zamanlı transfer fonksiyonu $T(z)$ ise (3.1)'de verilmiştir.

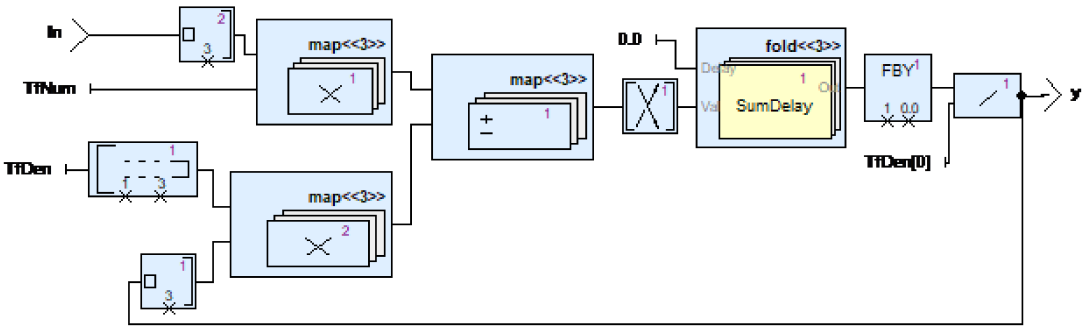
$$T(z) = \frac{\theta(z)}{V_{m,p}(z)} = \frac{0.002968z^2 - 0.001825z + 0.000259}{z^3 - 1.705z^2 + 1.098z - 0.256} \tag{3.1}$$

Burada çıkış $\theta(z)$ yunuslama açısını, giriş $V_{m,p}(z)$ yunuslama hareketini sağlayan motora uygulanan gerilimi ifade etmektedir.

(3.1)'de verilen transfer fonksiyonunu ifade eden SCADE Suite modeli Şekil 3.7'da verilmiştir.

Bu model "$liblinear$" kütüphanesindeki "$TransferFcnND << Ns, Ds >>$" operatörü

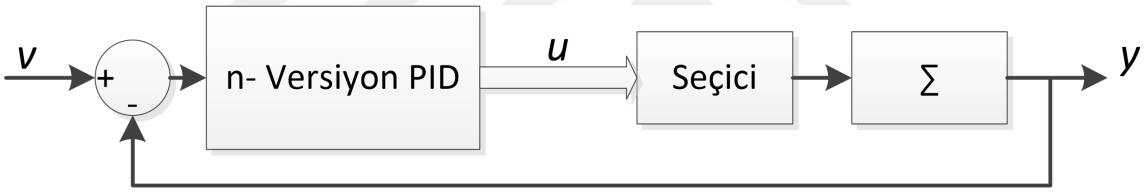
temel alınarak yeniden oluşturulmuştur.



Şekil 3.7 TransferFunction operatörünün SCADE Suite modeli

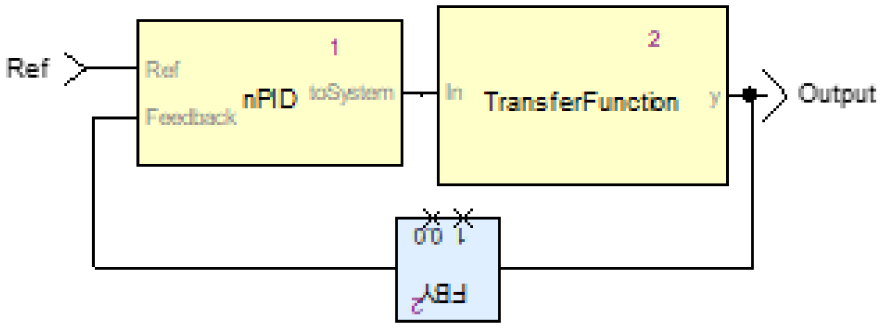
3.3.2.6 Tüm Sistem

Bir "*publicnode*" olan "*OverallSystem*" operatörü, sistemin kapalı çevrimi olarak tanımlanmıştır. Şekil 3.8'de gösterildiği gibi sistem, negatif geri beslemeli kapalı çevrim bir sistem olarak tanımlanmıştır.



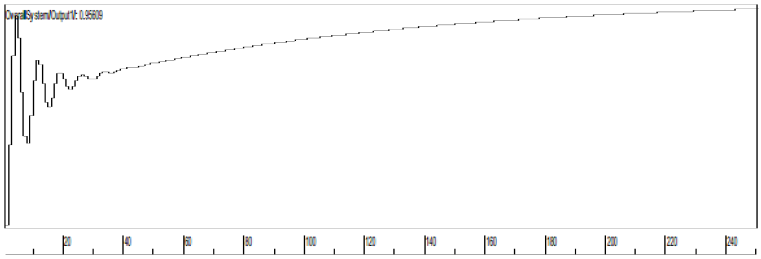
Şekil 3.8 Tüm sisteme ait blok şeması

nPID ve TransferFunction operatörleri ve negatif geri beslemenin başlangıç değerinin sıfır olarak tanımlanması için "Followed By" ön tanımlı operatörü kullanılmıştır. Sistemin gösterimi OverallSystem operatörü üzerinden simule edilmiştir. OverallSystem operatörün girişi Ref ve Çıkışı ise Output olarak tanımlanmıştır. OverallSystem operatörü Şekil 3.9'de gösterildiği gibi modellenmiştir.



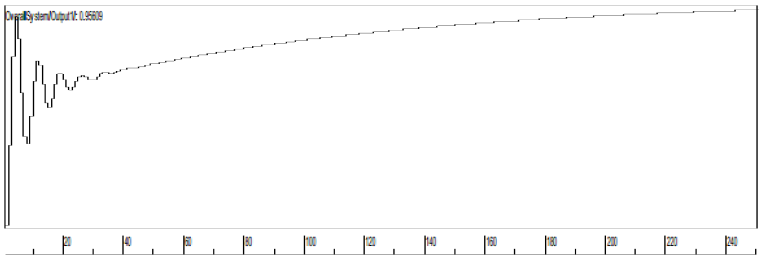
Şekil 3.9 OverallSystem operatörünün SCADE Suite modeli

Şekil 3.10'da 1. PIDBlock operatörünün sahip olduğu katsayılarla hesaplanan kapalı çevrim sistemin cevabı verilmiştir. Şekil 3.10 incelendiğinde sistemin kararlı olduğu fakat çok geç bir sürede sistemin istenilen referans değerine oturduğu görülmektedir.



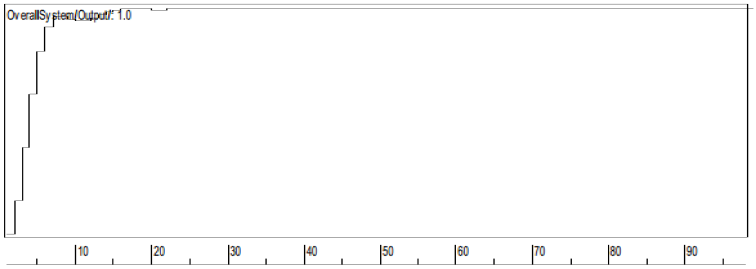
Şekil 3.10 PID1 kontrolcüsü için sistemin birim basamak cevabı

Şekil 3.11'de 2. PIDBlock'a ait sistemin kapalı çevrim cevabı verilmiştir. Görüldüğü gibi sistem kararlıdır ve bir miktar aşım yaptıktan sonra referans değerine oturmaktadır.



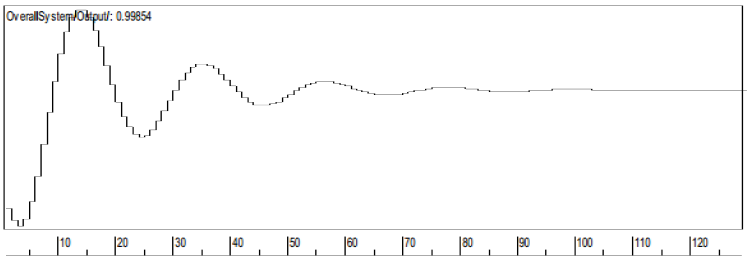
Şekil 3.11 PID2 kontrolcüsü için sistemin birim basamak cevabı

Şekil 3.12'de 3. PIDBlock'a ait sistemin kapalı çevrim cevabı verilmiştir. Bu cevap oturma zamanı ve aşım oranı gözetildiğinde bu sistem için en ideal cevaptır.



Şekil 3.12 PID3 kontrolcüsü için sistemin birim basamak cevabı

Sonuç olarak *voter* algoritmasının seçtiği PID katsayılarıyla sistemin cevabı Şekil 3.13'de görüldüğü gibidir. Burada voter 2. ve 3. PIDBlock operatörlerinin hesapladığı değerin ortalamasını alarak PID kazanç katsayılarını hesaplamıştır. Sistemin kararlı olduğu ve bir miktar aşım yaptığı görülmektedir.



Şekil 3.13 *Voter* algoritmasının ürettiği kazanç değerleriyle sistemin birim basamak cevabı

Voter operatörünün ideal kazanç değerini seçebilmesi veya hesaplayabilmesi adına, algoritma içerisinde karşılaştırma yaptığı değer sabit tutulmayıp, farklı bir fonksiyon kullanılarak hesaplanıp *voter* operatörü akıllı hale getirilebilecektir. Ayrıca *voter* algoritması, sistemin çıkışına ve çıkışın bir ve iki çevrim önceki değerine bakarak sistemin kararlılığı hakkındaki bilgi üretebilir. Bu bilgiye dayanarak sistemi farklı kanallar deneyerek kararlı duruma getirebilir.

4
ÇOK ÖLÇÜTLÜ KARAR YÖNTEMİ - TOPSIS

IEC 61508 Standardında tavsiye edilen yöntemlerden biri olan NVP yazılımda tasarım çeşitliliği ihtiyacını karşılamak amacıyla, aynı yazılım modülünün birden fazla sürümünü kullanır. Ancak hangi alternatiflerin tercih edilmesi gerektiği ve sürüm sayısının belirlenmesi problemi tam olarak çözülmüş değildir. Öte yandan, tasarım aşamasında yapılan hatalar sistemin güvenirliği, projenin tamamlanma süresini ve bütçesini önemli oranda etkilemektedir.

Seçicilerin kararının yanında hangi sürümlerin seçileceği kısmında ise karar verme sistemleri kullanılır. MCDM (Çok kriterli karar verme) yöntemlerinin amacı aldığı her türlü karar için doğru ve en uygun alternatifi seçmektir. Alınan kararların güvenirliği veya nesnelliği tasarım aşamasında çok önemli bir rol oynamaktadır. Kararlar yüzeysel olarak ve yalnızca tahmine dayalı olarak alınırsa, NVP tasarımının başarısızlığına sebep olacaktır. Bu nedenle klasik karar alma yöntemlerinin modern karar verme yöntemleriyle değiştirilmesi maliyet sisteminin güvenirliliği (reliability) ve kullanılabilirliği (availability) açısından avantajlı olacaktır. Çoklu kriter karar verme yöntemlerinden biri olan TOPSIS tasarım aşamasında doğru kararların alınmasında önemli bir rol üstlenmektedir. Bu çalışmada NVP tekniğinde hangi alternatiflerin tercih edilmesi gerektiği problemine TOPSIS (ideal çözümlere göre benzerlik yoluyla tercihi tekniği) algoritması yardımıyla çözüm bulunabilir.

Modern karar verme yöntemlerinin en önemlilerinden biri olan TOPSIS, daha objektif bir performans değerlendirmesi için birçok ölçüt dikkate alarak karar almaya yardımcı olmaktır. TOPSIS yöntemi, MCDM'de yaygın kullanımından dolayı literatürde geniş çapta kullanılmaktadır [63].

4.1 TOPSIS Algoritması

Çoklu kriter karar verme yöntemi MCDM, modern karar biliminin önemli bir parçasıdır. Toplum, ekonomi, askeri, yönetim vb. alanlara yoğun bir şekilde uygulanmıştır ve son on yılda giderek daha fazla ilgi görmektedir.

MCDM, sonlu bir nitelik kümesine göre uygulanabilir alternatiflerin ayrı bir setinden en uygun alternatif veya alternatifleri bulmayı içerir.

MCDM, karar bilimi alanında, sistemin sıcak bir araştırma konusu haline gelmektedir. Mühendislik, yönetim bilimi ve operasyon araştırmalarını kapsar ve çeşitli alanlara yoğun bir şekilde uygulanmıştır. Değerlendirme dizininin ağırlığı çeşitli birçok çalışma yapılmıştır.

MCDM yöntemlerinden sıklıkla kullanılan yöntemlerin başında TOPSIS, MAUT, CP ve VIKOR gibi yöntemler bulunmaktadır [62].

Çoklu karar verme prosedüründen biri olan TOPSIS, artan emniyet-kritik sistem ihtiyacı ile beliren maliyet kaygısı için çözüm önerisi olarak kullanılabilir. Bu nedenle, önemi günden güne artan bu yöntemle ilgili birçok çalışma vardır. Bu yöntemin temel mantığı, pozitif bir ideal çözüm ve negatif bir ideal çözüm elde ederek, ideal çözüme dayalı alternatifler sıralanmasına dayanmaktadır. İdeal çözüme nispeten yakın alternatiflerden başlayarak, bir dizi oluşturulmaktadır [64, 65].

4.1.1 Karar Matrisinin Normalleştirilmesi

Karar matrisi, karar verici tarafından üretilmesi gereken bir matristir. Bu matris mXn olacaktır. Karar verici, karar noktalarını satırlara ve etki faktörlerini ise sütunlara yerleştirmiştir. Bu matris (4.1)'da gösterilmiştir.

$$A_{ij} = \left\{a_{ij}\right\}_{\substack{i=1\cdots n \\ j=1\cdots m}} \tag{4.1}$$

Ardından o kriterlerin kareleri toplamının kareköküne bölünerek normalleştirilmesi gerçekleştirilir. Bu işlem (4.2) yardımıyla hesaplanır.

$$N_{ij} = \frac{a_{ij}}{\sqrt{\sum_{i=1}^{m} a_{ij}^2}} (i = 1, ..., m \text{ ve } j = 1, ..., n) \tag{4.2}$$

4.1.2 Normalleştirilmiş Karar Matrisinin Ağırlıklandırılması

Normalleştirilmiş matristeki her değer, w_{ij} ağırlık ile işlem görür. Ağırlıklandırma işlemi, TOPSIS yönteminin öznel tek yönünü bize vermektedir. Çünkü ağırlıklandırma değerleri kriterin önemine göre gerçekleşir. Bu ağırlıklar W_{ij} toplamı 1'e eşit olmalıdır. Böylece $\sum_{j=1}^{n} w_j = 1$ olacaktır. Normalleştirilmiş matris sayesinde oluşturulan n_{ij} değerleri, ağırlıklı normalize edilmiş bir matris (V Matris) vermesi için w_{ij} ağırlıkları

ile çarpılır. Matris (4.3) ile bulunabilir.

$$v_{ij} = w_i n_{ij} \;\; i = 1, 2, ..., m \;\; ve \;\; j = 1, 2, ..., n \tag{4.3}$$

4.1.3 Pozitif Ve Negatif İdeal Çözümlerin Belirlenmesi

Ağırlıklandırılmış normalize edilmiş matris (matris V) elde edildiğinde, sorunun yapısına bağlı olarak hedefin maksimize edilmesi şartıyla her bir sütunun maksimum değerleri belirlenir. Bu maksimum değerler bizim ideal çözüm değerlerimizdir. Daha sonra her sütun için minimum değerler elde edilir. Bunlar ise ideal çözümün negatif değerleridir. Amacımız minimize etmekse, elde edilen değerler tersine çevrilir. İdeal çözüm değerleri için göstergeler (4.4) ve (4.5) tarafından verilir.

$$A^* = \{(\max_i v_{ij} | j \in J), (\min_i v_{ij} | j \in J')\} \;\; i = 1, 2, ..., m \tag{4.4}$$

$$A^* = \{v_1^*, v_2^*, v_3^*, ..., v_j^*, ..., v_n^*\} \tag{4.5}$$

Negatif ideal çözüm değerleri elde edebilmek için (4.6) ve (4.7)'den yararlanılır.

$$A^- = \{(\min_i v_{ij} | j \in J), (\max_i v_{ij} | j \in J')\} \; i = 1, 2, 3, ...m \tag{4.6}$$

$$A^- = \{v_1^-, v_2^-, v_3^-, ..., v_j^-, ..., v_n^-\} \tag{4.7}$$

Bu denklemlerdeki J değeri fayda ölçütleri J' ise zarar ölçütleri olarak tanımlanmaktadır.

4.1.4 Uzaklık Değerlerinin Hesaplanması

Her seçenek arasındaki mesafe, ideal çözümün n.elemanı için (4.8) ile bulunmaktadır.

$$S_i^* = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^*)^2}, i = 1, 2, ...m \tag{4.8}$$

Negatif ideal çözümden uzaklığı ise (4.9) ile hesaplanmaktadır.

$$S_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2}, i = 1, 2, ...m \tag{4.9}$$

4.1.5 İdeal Çözüme Göre Yakınlığın Hesaplanması

İdeal ve ideal olmayan noktalara olan mesafeler, her karar noktasının ideal çözüme olan yakınlığını hesaplamak için kullanılır. İdeal çözümün yakınlığı C_i^* ile gösterilir. C_i^*, $0 <= C_i^* <= 1$ aralığındadır. Ayrıca $C_i^* = 1$, pozitif ideale yakınlık olarak mutlak çözümdür. İlgili karar noktasının negatif ideal çözümü, $C_i^* = 0$, ilgili karar noktasının ideal negatif çözüme mutlak yakınlığını belirtir. C_i^* (4.10) ile hesaplanmaktadır.

$$C_i^* = \frac{S_i^-}{(S_i^* + S_i^-)}, C_i^* \in [0, 1], i = 1, 2, 3, ..., m \tag{4.10}$$

4.1.6 Yakınlık Değerlerinin Hesaplanması

Hesaplanan yakınlık değerler büyükten küçüğe sıralanır. Göreceli olarak büyük olan C_i^* yakınındaki diğer seçenekleri seçmek, yüksek yakınlık derecesine göre reyting ve sıralama ile sonuçlanacaktır.

4.2 TOPSIS İle NVP Uygulaması İçin En iyi Alternatiflerin Seçimi

Sistem tasarımcısı m alternatiften p kriter için en iyisini seçmek için TOPSIS algoritmasını kullanabilmektedir. Emniyet-kritik uygulamalarda ise NVP en iyi çözümü bulabilmek için iki durum söz konusudur. Birinci durum TOPSIS algoritması sonucunda ortaya çıkan en iyi alternatifin N kere kullanılmasıdır (Tekrarlı Durum). İkinci durum ise algoritmanın belirlediği ilk N adet alternatifin tercih edilmesidir (Tekrarsız Durum).

N-sürüme geçildiğinde için tekrarsız durum için 220 kere, tekrarlı durum için ise 1748 kere p adet kriterin yeniden hesaplanması gerekir. Kriterlerden biri, örneğin maliyet ise N-sürüm durumunda, yeni maliyet kriter kullanılacak olan alternatiflerin maliyetlerinin toplamıdır. Bir diğer örneği verecek olursak CPU tarama hızı kriterlerden biri ise N-sürümde yeni CPU tarama hızı kriteri kullanılacak alternatiflerin en büyüğüdür.

p kriterlerinin N-sürüm için yeniden hesaplanması sonucu elde edilen alternatifler,

TOPSIS algoritmasının koşturulması sonucu N-sürümde kullanılacak olan alternatifler sıralanmaktadır.

Sistem tasarımcısı 12 tane alternatif içerisinden en iyiyi bulmaya çalışmaktadır. Bu seçim esnasında kriter olarak; alternatiflerin ücretleri, PTI (Proof Test Interval), Güvenirlik, Kullanılabilirlik ve tarama zamanı belirlenmiştir. Tasarımcı bu 5 kriter üzerinden 12 alternatiften, sürüm adedi 3 olmak üzere tekrarlı ve tekrarsız durum için en iyi 3 tanesini belirlemek istemektedir. Tablo 4.1 tasarımda kullanılacak 12 alternatif ve kriterler görüntülenmiştir.

Tablo 4.1 Alternatifler ve kriterler

Alternatifler ve Kriterler					
	M	**PTI**	**G**	**K**	**TZ**
Ağırlıklar	0,15	0,25	0,15	0,3	0,15
A1	10000	6	0,987578	0,999989	40
A2	15000	6	0,983471	0,999989	20
A3	12000	12	0,951229	0,999954	30
A4	14500	24	0,984963	0,999993	15
A5	16000	24	0,98895	0,999985	20
A6	12500	12	0,983471	0,999954	34
A7	10000	6	0,97531	0,999966	19
A8	20000	24	0,98895	0,999987	20
A9	23000	12	0,987879	0,999996	21
A10	18000	6	0,987578	0,999989	17
A11	14000	24	0,984733	0,999986	18
A12	13000	12	0,967216	0,999966	14

Buradaki sistem kriterleri olarak:

M: Maliyet Kullanılacak alternatifin maliyet değeri olarak kullanılmaktadır.

Tasarımcı alternatif seçiminde Türk Lirası olarak seçilmiştir.

PTI: İki ardışık prova testinin başlatılması arasındaki zaman aralığı, kanıt test aralığı olarak adlandırılır ve T ile gösterilir. Aralığın uzunluğu, sisteme bağlı olarak birkaç gün ila birkaç yıl arasında değişebilir. Karşılık gelen test frekansı, test aralığının tersidir (diğer bir deyişle, 1 / T). Prova test aralığı, normalde, bir tehlikeli tahmin edilemeyen arıza (Dangerous, Undetected Failure) süresinden daha kısa bir sürede olmak üzere seçilir.

Tasarımcı alternatif seçiminde 6,12 ve 24 aylık sürelerden seçim yapmıştır.

Sistem alternatiflerinin kriter seçimlerinde sistem elemanlarının iki istatistiksel değeri olan MTTR ve MTTF üzerinden güvenirlik ve kullanılabilirlik kriterleri hesaplanır. Bu iki değer sarısıyla:

- **MTTR**: Plansız olarak gerçekleşmiş olan bir arızanın ortalama süresidir. Bir sistem elemanının plansız olarak işlev göremez halden tekrar işlevini yerine getirebilir duruma gelişinin ortalama değeridir.
- **MTTF**: Plansız olarak gerçekleşmiş olan arızaların ortalama oluşma süresidir. Bir sistem elemanının plansız olarak işlev göremez halden tekrar işlevini yerine getirebilir duruma gelişinin ortalama değeridir.

G: Güvenirlik sistemin herhangi bir zaman dilimi içinde gerekli hizmetleri doğru biçimde verebilme olasılığıdır. 0-1 arasında yer almaktadır. $G(t)$ (4.11) ile hesaplanmaktadır.

$$G(t) = e^{-t\frac{1}{MTTF}} \tag{4.11}$$

K: Kullanılabilirlik herhangi bir zamanda sistemin çalışır durumda olması ve istenen hizmetleri verebilmesi demektir. 0-1 arasında yer almaktadır. $K(t)$ (4.12) ile hesaplanmaktadır.

$$K(t) = \frac{MTTF}{MTTF + MTTR} \tag{4.12}$$

TZ: Tarama Zamanı alternatifin hizmeti tamamlama süresi olarak belirlenmiştir. Tasarımcı alternatif seçiminde saniye olarak seçmiştir.

TOPSIS Algoritmasına göre tasarımcının seçtiği on iki alternatiften en iyiyi seçmek için normalize karar matrisi elde edilmelidir. Elde edilen matris Tablo 4.2'de gösterilmiştir.

Tablo 4.2 Normalize karar matrisi

	M	**PTI**	**G**	**K**	**TZ**
A1	0,189	0,109	0,291	0,289	0,489
A2	0,283	0,109	0,289	0,289	0,244
A3	0,226	0,218	0,280	0,289	0,367
A4	0,274	0,436	0,290	0,289	0,183
A5	0,302	0,436	0,291	0,289	0,244
A6	0,236	0,218	0,289	0,289	0,416
A7	0,189	0,109	0,287	0,289	0,232
A8	0,377	0,436	0,291	0,289	0,244
A9	0,434	0,218	0,291	0,289	0,257
A10	0,340	0,109	0,291	0,289	0,208
A11	0,264	0,436	0,290	0,289	0,220
A12	0,245	0,218	0,285	0,289	0,171

Algoritmanın sonraki basamağında ise her kriteri ağırlıklar ile çarparak ağırlıklanmış karar matrisi elde edilir. Elde edilen matris Tablo 4.3'de gösterilmiştir.

Tablo 4.3 Ağırlıklanmış karar matrisi

	M	**PTI**	**G**	**K**	**TZ**
A1	0,0283	0,0273	0,0436	0,0866	0,0733
A2	0,0424	0,0273	0,0434	0,0866	0,0367
A3	0,0340	0,0546	0,0420	0,0866	0,0550
A4	0,0410	0,1091	0,0435	0,0866	0,0275
A5	0,0453	0,1091	0,0437	0,0866	0,0367
A6	0,0354	0,0546	0,0434	0,0866	0,0623
A7	0,0283	0,0273	0,0430	0,0866	0,0348
A8	0,0566	0,1091	0,0437	0,0866	0,0367
A9	0,0651	0,0546	0,0436	0,0866	0,0385
A10	0,0509	0,0273	0,0436	0,0866	0,0312
A11	0,0396	0,1091	0,0435	0,0866	0,0330
A12	0,0368	0,0546	0,0427	0,0866	0,0257

Algoritmanın sonraki adımlarında ise pozitif ve negatif ideal çözümlere uzaklıkları hesaplanır, bu değerler toplanarak her bir alternatif için her kriterin ideal çözüme uzaklıkları toplanıp başarı değeri elde edilir. Bu başarı tablosu Tablo 4.4'de gösterilmiştir.

Tablo 4.4 İdeal çözüme yakınlık başarı değerleri

ALTERNATİF	İDEAL ÇÖZÜME YAKINLIK BAŞARISI
A1	0,2799666
A2	0,3398243
A3	0,4211388
A4	0,8827165
A5	0,8194876
A6	0,3875329
A7	0,3927846
A8	0,7479279
A9	0,3977538
A10	0,3434730
A11	0,8753556
A12	0,5280855

Bu ideal çözüme yakınlık başarı değeri hesaplandıktan sonra sıralama yapıldığında en başarılı alternatiften en başarısız alternatife doğru sıralanmış listeyi elde etmiş oluruz. Bu sıralama Tablo 4.5'de gösterilmiştir.

Tablo 4.5 Alternatiflerin sıralama sonuçları

SIRA NO	İDEAL ÇÖZÜME YAKINLIK BAŞARISI	ALTERNATİF
1	0,8827165	A4
2	0,8753556	A11
3	0,8194876	A5
4	0,7479279	A8
5	0,5280855	A12
6	0,4211388	A3
7	0,3977538	A9
8	0,3927846	A7
9	0,3875329	A6
10	0,3434730	A10
11	0,3398243	A2
12	0,2799666	A1

NVP için seçilecek üç alternatifin oluşturacağı yeni sistem için kriterlerin bu üç alternatif için yeniden düzenlenmesi gerekmektedir.

- **M**: Maliyet kriterinin hesaplanması için kullanılacak olan üç alternatifin maliyetlerinin toplamıdır.
- **PTI**: Bu kriter için kullanılacak olan üç alternatifin minimumu seçilmelidir.

- **G**: Güvenirlik kriteri için yeni sistemin güvenirliği (4.13) ile hesaplanır.

$$G = 1 - ((1 - G_1)(1 - G_2)(1 - G_3)) \tag{4.13}$$

- **K**: Kullanılabilirlik kriteri için yeni sistemin kullanılabilirliği K (4.14) ile hesaplanır.

$$K = 1 - ((1 - K_1)(1 - K_2)(1 - K_3)) \tag{4.14}$$

- **TZ**: Bu kriter için kullanılacak olan üç alternatifin maksimumu seçilmelidir

Bu sıralamaya göre tekrarlı yani alternatiflerden üç alternatifin de Alternatif4 seçilmesi öngörülebilmektedir. Fakat Tablo 4.6 'te görüldüğü gibi aynı seçilen alternatiflerde Alternatif11 seçimi ideal çözüme yakınlık başarısı daha yüksektir.

Tablo 4.6 Tekrarlı durum için ideal çözüme uzaklık sıralama tablosu

SIRA NO	İDEAL ÇÖZÜME UZAKLIĞI	SEÇİLECEK ALTERNATİF	SEÇİLECEK ALTERNATİF	SEÇİLECEK ALTERNATİF
1	0,919506112	A11	A11	A11
2	0,918963607	A4	A4	A4
3	0,916992247	A4	A11	A11
4	0,916992247	A11	A4	A11
5	0,916992247	A11	A11	A4
6	0,914471594	A4	A4	A11
7	0,914471594	A4	A11	A4
8	0,914471594	A11	A4	A4
9	0,901486800	A5	A11	A11
10	0,901486800	A11	A5	A11
11	0,901486800	A11	A5	A5
12	0,899188288	A4	A5	A11

Tekrarsız durum alternatiflerin üçlü kombinasyonları içinde yeni kriter değerlerine göre yeniden hesaplatılması ve sıralaması sonucu yeni sıralama listesi Tablo 4.7 gibi oluşmuştur.

Tablo 4.7 Tekrarsız durum için ideal çözüme uzaklık sıralama tablosu

SIRA NO	İDEAL ÇÖZÜME UZAKLIĞI	SEÇİLECEK ALTERNATİF	SEÇİLECEK ALTERNATİF	SEÇİLECEK ALTERNATİF
1	0,924794594	A4	A5	A11
2	0,904954316	A4	A8	A11
3	0,897630123	A5	A8	A11
4	0,895206705	A4	A5	A8
5	0,379022222	A4	A11	A12
6	0,371806481	A5	A11	A12
7	0,371173322	A4	A5	A12
8	0,366970269	A8	A11	A12
9	0,366404539	A4	A8	A12
10	0,364762844	A5	A8	A12
11	0,360996988	A9	A11	A12
12	0,360482859	A4	A9	A12

Tekrarlı ya da tekrarsız olarak N adet alternatifin projenin tasarım aşamasında ortaya konulabiliyor oluşu gerek proje takvimine uyum gerekse proje bütçesinin hedeflenen düzeyde kalabilmesi adına çok büyük fayda sağlamaktadır. Bu sonuca ulaşırken çoklu kriter karar verme yöntemlerinden biri olan TOPSIS algoritmasından da yararlanılmıştır. Bu m adet alternatif içinden en iyi olanın NVP yöntemine geçildiğinde tekrarlı ya da tekrarsız durumlar için bazen en iyi olmadığı gösterilmiştir. Bunun neticesinde emniyet-kritik yazılım geliştirme süreçleri için etkin bir yöntem ortaya konulmuştur. Ayrıca, tekrarlı durumlarda donanımsal olarak aynı alternatifin N kere seçilmesi ortak nedenli arıza durumunda tüm alternatiflerin yitirilme olasılığını üreteceğinden aynı donanım içinde muhakkak farklı yazılımların kullanılması gerektiği sonucu ortaya çıkmıştır.

5
BİR GERÇEK-ZAMAN UYGULAMASI:TILT-ROTOR SİSTEMİ

Bu bölümde NVP'nin yeni bir şekilde nasıl kullanılacağı açıklanmıştır. NVP ile, bir sonraki işlem için aynı kontrol cihazının birkaç sürümü kullanılmıştır. Bununla birlikte, NVP-MV, sistemi dengesiz bir konfigürasyona sokmak için oy kullanabilir. Örneğin, NVP-MV insansız hava aracının (İHA) düşmesine neden olabilir.

Bu bölümdeki çalışma, NVP çerçevesinin, yine de istikrarlı bir sisteme yol açacak olan azınlıktan girişi seçmesine izin verir. NVP, bu tür girişleri geçersiz olarak işaretleyen bir kararsızlık dedektörü ile birleştirir.

NVP-MV algoritması ayrıntılı olarak açıklanmaktadır. Gerçek-zamanlı bir deney için, bir tilt-rotor stabilizasyon platformu oluşturulmuş ve burada sistemin matematiksel modeli verilmiştir. Sistem üç serbestlik derecesine sahiptir. Bu nedenle; yatma(Roll), yunuslama(Pitch) ve sapma(Yaw) eksenleri arasında serbestçe hareket edebilmektedir. Platform her dönme ekseni için orantı-integral-türev (PID) kontrolörlerine sahiptir. Genel kayıp olmadan, NVP-MV yapısı sadece yatma ve yunuslama kontrol cihazlarına uygulanmaktadır. Ayrıca, NVP-MV algoritması ile sisteme bir stabilite kontrol özelliği eklenerek modifiye edilmiştir. Deneysel sonuçlar ve ulaşılan yorumlar ilerleyen alt bölümlerde tartışılmıştır.

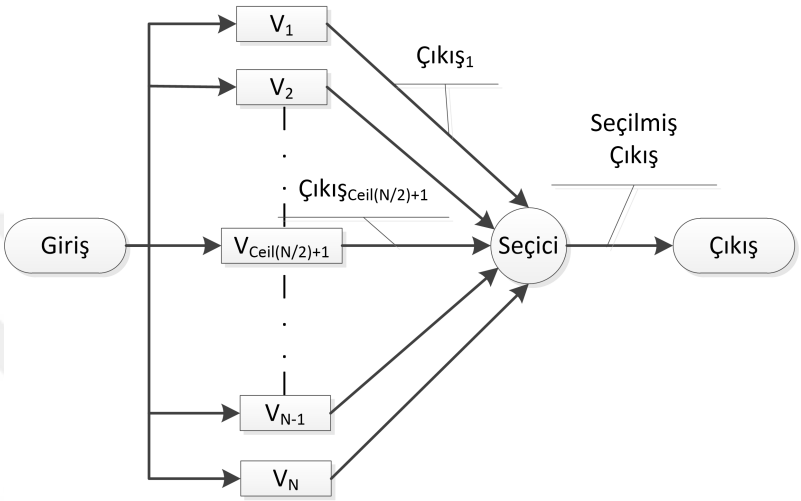
5.1 N-versiyon Programlama

NVP problemlerinde sunulan oylama algoritmaları, başlangıç verilere ve çalışma programına bağlı olarak farklık göstermektedir. Bir veri seti için en uygun oylama algoritmasını seçmek çok önem arz etmektedir. Bununla birlikte, algoritmaların uygulanması verilerin eşdeğer kendi alt kümelerine bölünmesini gerektirmektedir [32, 35, 73].

NVP tekniği, bağımsız olarak tasarlanmış N program versiyonlarının oluşturduğu mimari yapıdır. Bu mimarinin matematiksel tanımı (5.1)'de verilmiştir.

$$(V_1, V_2, \cdots, V_{\lceil N/2 \rceil + 1}, \cdots, V_{N-1}, V_N) \tag{5.1}$$

NVP algoritmasının çıktısının, eğer en azından ($\lceil N/2 \rceil 1$) versiyonları aynı çıktı üzerinde anlaşıyorsa, güvenilir olduğu kabul edilir [74]. Bu Şekil 5.1'de gösterilmiştir.



Şekil 5.1 Basitleştirilmiş NVP-MV diyagramı

NVP-MV için Anlaşma Matrisi

Doğru çıktı setini seçmedeki en kritik nokta, anlaşma matrisi *R*'nın yaratılması ve analiz edilmesidir. *N* sürüm sayısı olan *N*x*N* boolean matrisidir ve çıktılar arasındaki eşitliği yansıtır. *R* unsurları (5.2) ile belirlenir.

$$r_{ij} = \begin{cases} 1, & |\, x_i - x_j \,| \leq \varepsilon \\ 0, & |\, x_i - x_j \,| > \varepsilon \end{cases} \tag{5.2}$$

(5.2)'deki i ve j sırasıyla *R*'nin satır ve sütunları tanımlamaktadır. x_i ve x_j versiyonların çıktılarını tanımlamaktadır. ε ise eşik toleransıdır.

R anlaşma matrisi aşağıdaki terimler geçerlidir. *R* üzerindeki eşitlik ilkesi yansıma (5.3), simetri (5.4) ve geçiş (5.5) özellikleri tanımlanmıştır.

$$r_{ii} = 1, \forall i, \tag{5.3}$$

$$r_{ij} = r_{ji}, \forall i \neq j, \tag{5.4}$$

$$\text{if } r_{ik} = 1 \text{ and } r_{kj} = 1 \text{ then } r_{ij} = 1, \forall i, j. \tag{5.5}$$

Boolean bileşimlerinin R üzerindeki amacı, denklik ilişkisinin bulunduğu uygun bir forma dönüştürmektir. Genel olarak, Boolean matrisleri için kompozisyon çalışmaları (5.6)'de tanımlanmıştır. A ve B iki matris olmak üzere ve onların giriş değerleri 0 veya 1'dir. Ardından A ve B matrislerinin Boolean birleşimi (5.6)'de tanımlanmıştır.

$$C = A \circ B, \text{where } c_{ij} = \bigoplus_{k=1}^{N} (a_{ik} \otimes b_{kj}), \tag{5.6}$$

Denklemde $\oplus$ ve $\otimes$ Boolean VE / VEYA operatörleridir.

R anlaşma matrisindeki (5.3), (5.4) ve (5.5) eşdeğerlik ilişkisinin yerine getirilmesi için R tutarındaki Boolean kompozisyonlarının uygulanması gerekir. Bu ilke (5.7) ile birlikte gerçekleştirilir

$$E = R^1 \cup R^2 \cup \cdots \cup R^Q, 1 \leq Q \leq N-1, \tag{5.7}$$

Q Boolean kompozisyonunun sonuç sayısı, N ise sürüm sayısıdır. Oluşturulan Matris sonucu tatmin edici değilse (5.8) denklemi ile Boolean Kombinasyonu yenilenir.

$$E^2 = R \cup R \circ R \tag{5.8}$$

(5.8) ile yeniden oluşturulan Boolean kombinasyon matrisi sonucu tatmin edici sonuç barındırmıyorsa (5.9) denklemi kullanılır.

$$E^3 = R \cup R \circ R \cup R \circ R \circ R = E^2 \cup R^3. \tag{5.9}$$

NVP-MV Algoritması:

N sürümünden her birinin bağımsız olduğunu ve her sürüm tarafından oluşturulan çıktı değerlerinin $x_1, x_2, \cdots, x_N$ tarafından belirtildiğini varsayalım. ε tolerans değerini seçtikten sonra, algoritmanın adımları uygulanır.

1. R anlaşma matrisini (5.2) kullanarak oluşturun.

2. R üzerindeki eşdeğerlik ilişkisini (5.3), (5.4) ve (5.5) koşulları altında analiz edilir. Eğer uygun ise Adım 4'e gidilir. Değilse ise Adım 3'e gidilir.

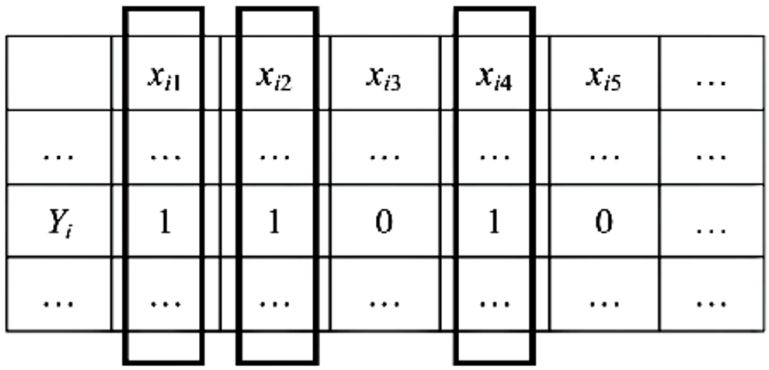
3. (5.7), R için eşitlik oranı (5.3), (5.4) ve (5.5) 'e kadar olana kadar yapılmaz.

4. Doğru çıkış seti tanımlanmalıdır. R'ın her satırında, elemanların sayısı belirlenir. Y_i, i satırındaki öğelerin sayısını gösterir. Böyle bir satır için i ile (5.10) ile uygulanır.

$$Y_i \geq \left\lceil \frac{N+1}{2} \right\rceil \tag{5.10}$$

   (5.10) içerisindeki $\lceil . \rceil$ tavan oparatörüdür.

Şekil 5.2 sürümlerin sonuçlarının nasıl seçildiğini ve A ile doğru sonuçların alındığını gösterir.

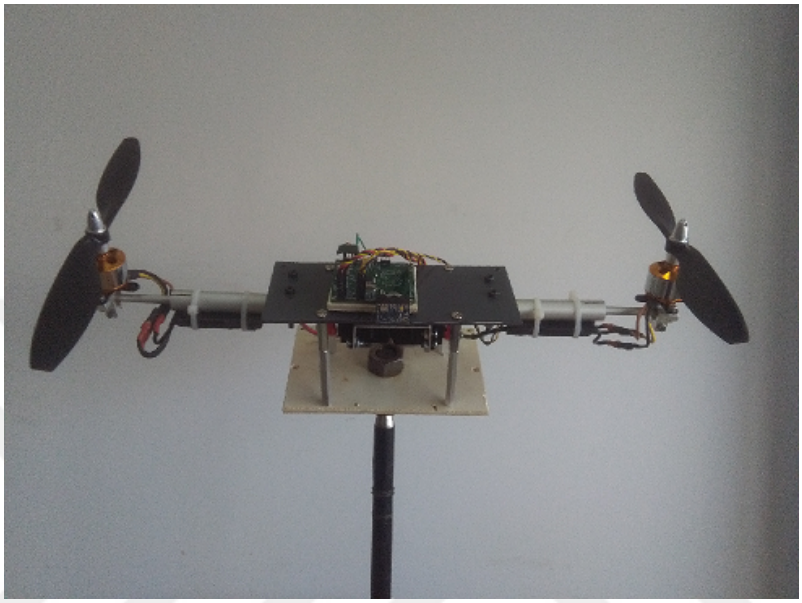


Şekil 5.2 R'dan doğru cevapların seçimi [51]

5.2 Tilt-Rotor Sistemi ve Kontrolcü Yapısı

Sistemin mekanik yapısı iki ana bölümden oluşmaktadır. Biri sabit bir taşıyıcı, diğeri ise sabit taşıyıcı üzerine monte edilmiş çift eğimli rotor sistemidir. Tilt-Rotor çerçevesi platformun sınırlamasına göre üç bağımsız eksen etrafında serbestçe döndürülebilmektedir. Bu nedenle, sistem üç serbestlik derecesine sahiptir ve yatma, yunuslama ve sapma eksenleri üzerinde hareketlidir. Tilt-Rotor sistemi, fırçasız doğru akım (BLDC) ve aktüatör olarak servo motorlar içermektedir [75]. Servolar yunuslama ve sapma momentlerinden sorumludur ve BLDC motorları yatma eksen kontrolünde kullanılmaktadır. Şekil 5.3 sistemi kontrol altında gösterilmektedir. Sistemin matematiksel modeli doğrusal değildir. Bu

gerçeklemede, denetleyici tasarımını çok daha konforlu hale getiren doğrusal bir sistem yaklaşımı gerçekleştirilmiştir [76]. Kontrolcü bir PID kontrolcüsü seçilmiştir. PID kontrolcüsünün herhangi bir donanım arızası, tesisin performansını ve hatta kararlılığını etkileyecek istenmeyen kontrol sinyallerine neden olmaktadır. Bu sorunun üstesinden gelmek için NVP-MV tabanlı bir yapı göz önünde bulundurulmuştur.



Şekil 5.3 Test platformunu

5.2.1 Matematiksel Modeli

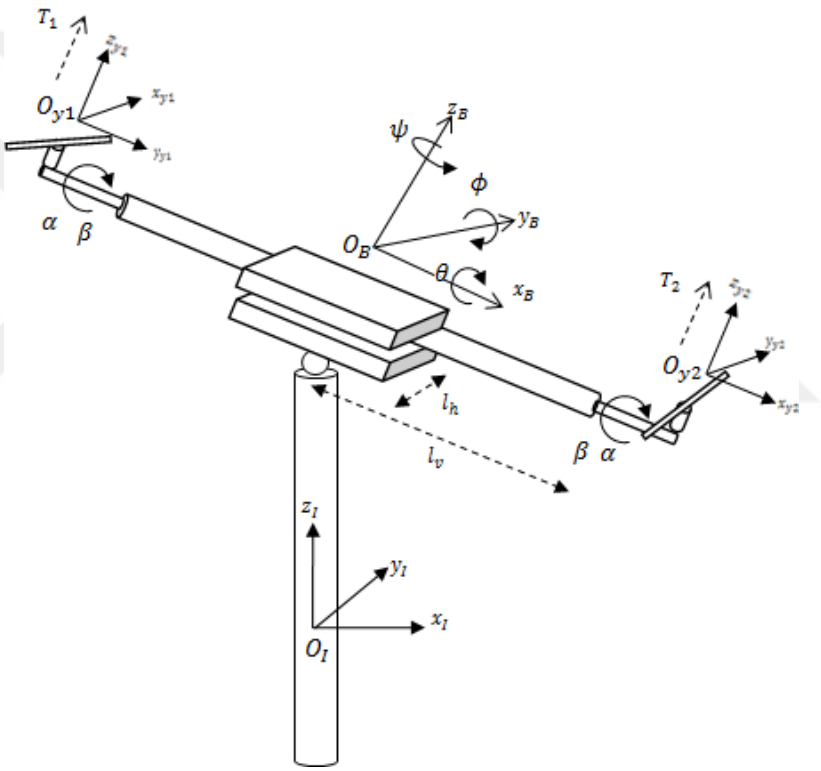
Sistemi stabilize etmek için yatma, yunuslama ve sapma torkları kullanılmaktadır. Sistemin yatma kontrolü için, BLDC motor hız farkı kullanılır. Servolar yunuslama ve sapma torklarını sağlamakta ve BLDC motorlarından yatma torku ile ortaya çıkan itme kuvvetini değiştirilmesi ile sağlamaktadır. Modelleme için, platformun çerçeveleri aşağıdaki şekilde tanımlanmıştır; tilt-rotor kısmı atalet çerçevesi ve sabit taşıyıcı gövde çerçevesidir. Ayrıca, (5.11) atalet ve vücut çerçevesinin koordinatını belirtilmektedir.

$$\begin{aligned} O_I &= \{x_I, y_I, z_I\} \\ O_B &= \{x_B, y_B, z_B\} \end{aligned} \tag{5.11}$$

Tilt mekanizması nedeniyle, BLDC motorları kendi yapıları bulunmaktadır. Geri devrilme sapma momentine neden olur ve O_{y1} ve O_{y2} değerleri ile temsil edilmektedir. O_p ile gösterilen paralel devrilme, yatma torku üretmektedir. Denklemler (5.12)'de gösterilmektedir.

$$\begin{aligned} O_{y_1} &= \{x_{y_1}, y_{y_1}, z_{y_1}\} \\ O_{y_2} &= \{x_{y_2}, y_{y_2}, z_{y_2}\} \\ O_p &= \{x_p, y_p, z_p\} \end{aligned} \tag{5.12}$$

Sistem eksenlerinin ayrıntılı açıklaması Şekil 5.4'de görülmektedir. α, β ise sırasıyla geri ve paralel devrilme açılarını tanımlamaktadır.



Şekil 5.4 Platformun atalet ve gövde Yapısı

l_v, l_h sırasıyla sabitleme noktasından en ve boy olarak uzaklıklardır. Pervaneler tarafından üretilen tork kuvvetleri, T_1 ve T_2 ile temsil edilmektedir. Pervanelerin ana güven kuvvetini oluşturduğunu unutmayın.

Paralel ve geri devrilme, yunuslama ve sapma torklarını üretmektedir. Yatma torku, rotorların farkından elde edilmektedir. Dönme yer değiştirmesi $\xi = \{\theta, \phi, \psi\}$ olarak tanımlanmaktadır. Sistemin doğrusal olmayan dönüş dinamikleri Newton-Euler yöntemi kullanılarak elde edilebilmektedir. Bu yöntem (5.13)'de gösterilmektedir.

$$\vec{\tau} = I\dot{\vec{\Omega}} + \vec{\Omega} \times (I\vec{\Omega}) \tag{5.13}$$

Denklemde, $\vec{\tau}$ toplam tork vektörüdür, I atalet matrisini ve Ω ise açısal hız vektörüdür. Toplam tork (5.14)'de görüldüğü gibi, jiroskopik, baskı ve ağırlık torklarını içermektedir.

$$\vec{\tau} = \vec{\tau}_c + \vec{\tau}_g + \vec{\tau}_w \tag{5.14}$$

Bununla birlikte, modelin derecesini azaltmak için, eğim ile üretilmiş jiroskopik torklar dikkate alınmamaktadır. Dönüş dinamikleri nedeniyle, gerekli dönüşüm matrisleri (5.15) ile tanımlanmaktadır.

$$R^{B\to I} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{5.15}$$

$R^{B\to I}$, gövdenin iç çerçeve dönüşümüne tanımlanmakta ve $c = cos$ ve $s = sin$ ile yerine konulduğunda (5.16) elde edilmektedir.

$$R^{y_1\to B} = \begin{bmatrix} c_\alpha & 0 & s_\alpha \\ 0 & 1 & 0 \\ -s_\alpha & 0 & c_\alpha \end{bmatrix}, R^{y_2\to B} = \begin{bmatrix} c_\alpha & 0 & -s_\alpha \\ 0 & 1 & 0 \\ s_\alpha & 0 & c_\alpha \end{bmatrix} \tag{5.16}$$

$R^{y_1 \to B}$, $R^{y_2 \to B}$, rotorların ters eğim etkisini göstermektedir. Rotorların yunuslama değişiminin gövde çerçevesi dönüşüm matrisi (5.17)'deki gibi tanımlanmaktadır.

$$R^{p \to B} = \begin{bmatrix} c_\beta & 0 & -s_\beta \\ 0 & 1 & 0 \\ s_\beta & 0 & c_\beta \end{bmatrix} \tag{5.17}$$

Bu bağlamda, dönüşüm matrislerini kullanarak, gövde çerçevesinin merkezindeki kuvvet (5.18)'deki gibi tanımlanmaktadır.

$$\begin{cases} T_1^B = R^{y1 \to B} R^{p \to B} T_1 \\ T_2^B = R^{y2 \to B} R^{p \to B} T_2 \end{cases} \tag{5.18}$$

Eyleyici torku ise (5.19)'deki gibi hesaplanmaktadır.

$$\tau_c = l_{y1}^B \times T_1^B + l_{y2}^B \times T_2^B \tag{5.19}$$

Denklemde, l_{y1}^B, l_{y2}^B, denge noktasından uzaklığı temsil etmektedir. $l_{y1}^B = [l_h, -l_v, 0]^T$, $l_{y2}^B = [-l_h, -l_v, 0]^T$ Ağırlık torku, gövde çerçevesi üzerindeki ağırlık merkezinden uzaklığı sayesinde sağlanır ve (5.20) gibi tanımlanmaktadır.

$$\tau_w = l_w^B \times R^{I \to B} m G^I \tag{5.20}$$

Denklemde, l_w^B, ağırlık merkezinin denge noktasından uzaklığıdır ve $l_w^B = [0, -l_v, 0]^T$ olarak tanımlanır. m, gövde çerçevesinin kütlesidir ve G^I, eylemsizlik çerçevesine göre yerçekimi vektörüdür.

Dolayısıyla, denklemler (5.13), (5.19) ve (5.20) türevlerini alarak, sistemin doğrusal olmayan dinamiği (5.21)'deki denklemlerle modellenebilmektedir.

$$\begin{aligned} I_{xx}\ddot{\phi} &= \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) - (T_1 + T_2) l_h s_\beta s_\alpha - (T_1 - T_2) l_h c_\beta c_\alpha \\ I_{yy}\ddot{\theta} &= \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - (T_1 + T_2) l_v c_\alpha c_\beta - (T_1 - T_2) l_v s_\beta s_\alpha + c_\psi c_\theta l_v m g \\ I_{zz}\ddot{\psi} &= \dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) - (T_1 + T_2) l_v s_\beta c_\alpha - (T_1 - T_2) l_v s_\alpha c_\beta + s_\theta l_v m g \end{aligned} \tag{5.21}$$

Denge noktası etrafında doğrusal bir model elde etmek için dinamik denklemlere doğrusal bir yaklaşım uygulanmaktadır. Yatma, yunuslama ve sapma yer

değiştirmelerinin ve hızlarının tümü sıfıra eşitlenmiştir. Böylece doğrusal denklemler sağlanması için üç alt sistem tanımlanabilmektedir. Yatma dengelenmesi için $\alpha = \beta = \theta = \psi = 0$'a eşitlenip Yatma denkleminin basitleştirilmiş hali (5.22)'de verilmiştir.

$$I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) - (T_1 - T_2)l_h \tag{5.22}$$

Kontrol sinyali $u_1 = (T_1 - T_2)$ olarak tanımlanırsa ve $\delta\phi$, $\delta\dot{\phi}$ değerindeki küçük sapmalar için, yatma dinamiğinin doğrusal yaklaşımı (5.23)'deki gibi tanımlanabilmektedir.

$$I_{xx}\delta\ddot{\phi} = -u_1 l_h \tag{5.23}$$

Yunuslama dinamikleri için, $\alpha = \psi = \phi = 0$ dengesini varsayarsak, yunuslama denklemi (5.24)'deki gibi tanımlanabilmektedir.

$$I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - (T_1 + T_2)l_v c_\beta + l_v mgc_\theta \tag{5.24}$$

Denklemde, kontrol sinyali $u_2 = (T_1 + T_2)c_\beta$ olarak tanımlanmışsa ve $\delta\theta$, $\delta\dot{\theta}$ 'ın küçük sapması için, denklemin sadeleştirilmiş hali (5.25)'de gösterilmiştir.

$$I_{yy}\delta\ddot{\theta} = -u_2 l_v + l_v mg\delta\theta. \tag{5.25}$$

Sapma dinamikleri için, $\beta = \theta = \phi = 0$ olduğu varsayılırsa, sapma denklemi (5.26)'deki gibi tanımlanabilmektedir.

$$I_{zz}\ddot{\psi} = \dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) - (T_1 + T_2)l_v s_\alpha \tag{5.26}$$

Denklemde, kontrol sinyali $u_3 = (T_1 + T_2)s_\alpha$ olarak tanımlanmışsa ve $\delta\psi$, $\delta\dot{\psi}$'ın küçük sapması için, denklemin sadeleştirilmiş hali (5.27)'de gösterilmiştir.

$$I_{zz}\delta\ddot{\psi} = -u_3 l_v \tag{5.27}$$

Sistemin durumları (5.28)'de tanımlanmıştır.

$$x_1 = \delta\dot{\theta} \qquad x_2 = \delta\dot{\phi} \qquad x_3 = \delta\dot{\psi} \tag{5.28}$$

Sistemin sadeleştirilmiş doğrusal modeli (5.29) ve (5.30)'da gösterilmiştir.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{5.29}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} \frac{l_v mg}{I_{yy}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} \frac{-l_h}{I_{yy}} & 0 & 0 \\ 0 & \frac{-l_v}{I_{xx}} & 0 \\ 0 & 0 & \frac{-l_v}{I_{zz}} \end{bmatrix} u$$
$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x \tag{5.30}$$

Burada sistem parametreleri Tablo 5.1'de gösterilmiştir. Başlangıç durumları ise (5.31)'de gösterilmiştir.

Tablo 5.1 Test ortamının parametre değerleri

Parametreler	**Değerler**
I_{xx}	32x10$^{-3}kgm^2$
I_{yy}	102x10$^{-3}kgm^2$
I_{zz}	72x10$^{-3}kgm^2$
m	134x10^{-2}kg
l_h	30x10^{-2}m
l_v	90x10^{-2}m
g	$9.8m/s^2$

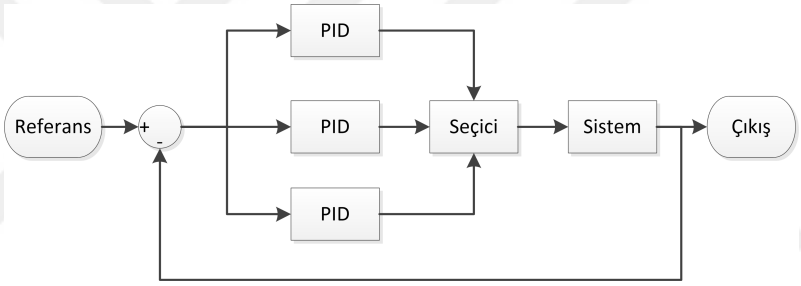
$$x_0 = \begin{bmatrix} 40^{\circ}(Yunuslama^{\circ}) & 0^{\circ}(Yatma^{\circ}) & 0^{\circ}(Sapma^{\circ}) \end{bmatrix} \tag{5.31}$$

Doğrusallaştırılmış model, sistemin PID denetleyicisi gibi düşük dereceli denetleyiciler kullanılarak düzenlenebileceğini göstermektedir. Kontrolör transfer fonksiyonu (5.32) 'de gösterilmektedir.

$$PID = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}} \tag{5.32}$$

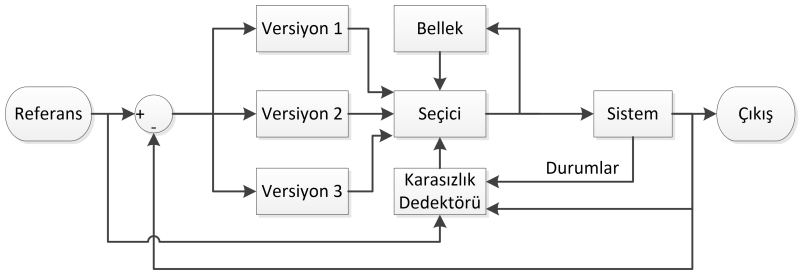
5.2.2 Kontrolcü Yapısı

Bu bölümde, geliştirilmiş bir NVP-MV algoritması tilt-rotor insansız hava aracı için gerçek-zamanlı bir kontrolcü tasarımı sunulmuştur. Tatmin edici performans ve kapalı döngü kararlılığı sağlamak için PID kontrolcüsü seçilmiştir. Genel olarak, NVP-MV algoritması 3te2 yapısındadır. Yani üç versiyondan iki versiyon kabul edilirse çoğunluk seçmeninin bu kararı doğru çıktı olarak kabul etmesi anlamına gelmektedir. Bu genel yaklaşım Şekil 5.5'de gösterilmiştir.



Şekil 5.5 NVP-MV PID ile kontrol edilen sistem

Bu çalışmada, seçmenlerin, kontrolörlerin dengeleyici kararlarını algılama anlamında daha akıllı olmasını sağlayan bir algoritma sunulmaktadır. Bu geliştirilmiş seçmen tasarımında, bir kararsızlık dedektörü ve önceki kararı saklayan bir hafıza uygulanmıştır. Böylece, NVP-MV seçmeni, kararın sistemi dengede tutup tutmayacağını bilmektedir. Yeni NVP-MV'nin temel blok diyagramı, Şekil 5.6'da gösterilmiştir.



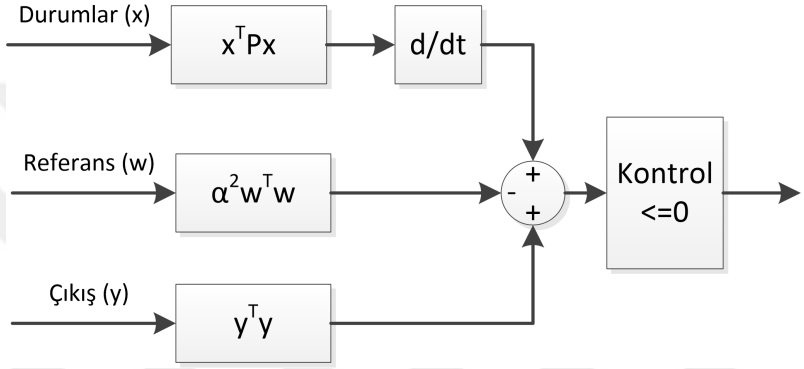
Şekil 5.6 Tasarlanan seçmen diyagramı

Kararsızlık detektörü; sistem çıkış değerine, sistem durumlarına ve kontrol edilen sistemin referansına ihtiyaç duymaktadır. Sistem kararlı olduğu zaman dedektör çıkışı

0'dır (Yanlış). Öte yandan, sistemin çıkışı değiştiğinde (kararsız), dedektörün çıkışı 1'dir (Doğru).

Feron [66], emniyet-kritik bir kontrolcü elde etmek için çevrimiçi bir Lyapunov stabilite analiz özelliğinin mimariye entegre edilebileceğini önermiştir. Böylelikle seçmenlere kararsızlık dedektörü olarak adlandırılan bir özellik eklenmesi geliştirilmiştir.

Girdi-çıktı kararlılığı için hem depolama hem de tedarik fonksiyonlarının oluşturulması gerekmektedir. Şekil 5.7, tilt-rotor sisteminin $\mathscr{L}_2$ kazanç dengesi prensibini (5.33)'de gösterilmiştir.

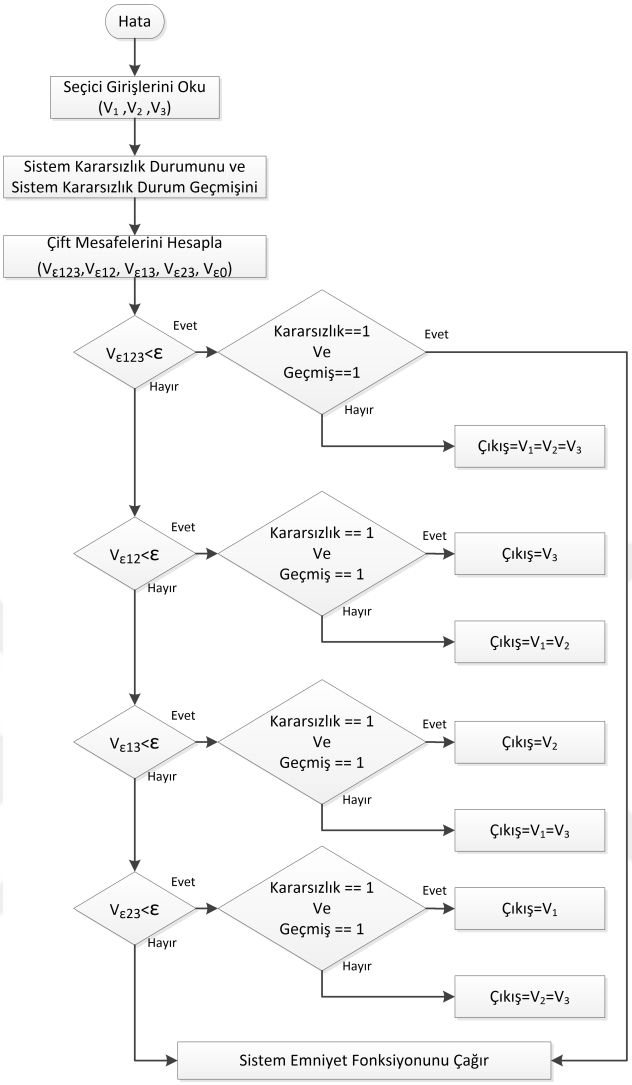


Şekil 5.7 Karasızlık dedektörü

$$\dot{V}(x) - \alpha^2 w^T w + y^T y \leq 0 \tag{5.33}$$

(5.33)'de w birim tepe gürültüsü, y ise sistemin çıkışı olarak tanımlanmaktadır. Burada, $x^T P x$ (ikinci dereceden Lyapunov benzeri) depolama fonksiyonunu temsil etmektedir. P matrisi ve α seçimi çaba gerektiren bir bölümdür. Bu çalışmada, alan bilgilerinden ve deneme yanılma simülasyonları sonrası uygun P matrisi ve α değeri seçilmiştir.

Önerilen geliştirilmiş NVP-MV Algoritması, Şekil 5.8'de gösterilmiştir.



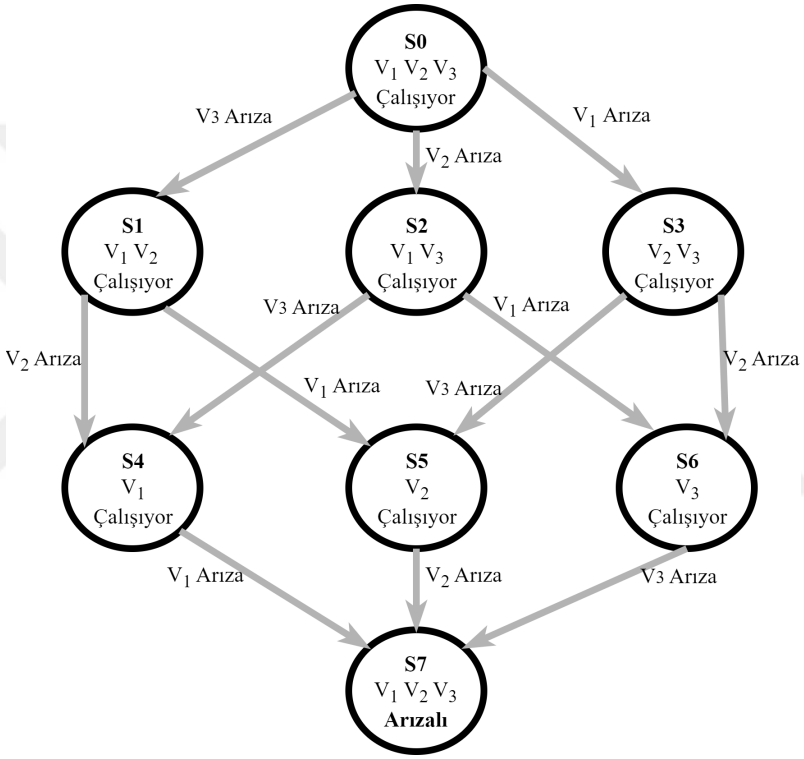
Şekil 5.8 Tasarlanan seçmen algoritması

5.2.3 Deneysel Sonuçlar

Geliştirilmiş NVP-MV tabanlı PID kontrolcüsünün performansı, bir tilt-rotorlu sistem üzerinde yapılan simülasyonlarla değerlendirilmiştir. Bu bölümde, simülasyon senaryoları ve yüksek kullanılabilirliğe sahip PID kontrolcü tasarımı açıklanmaktadır. Simülasyon ortamı olarak, gerçek-zamanlı davranışa ve sistemin matematiksel modeline dayanan Matlab R2017b Simulink modeli kullanılmıştır. Simülasyon senaryosunda, sistem yatma ve yunuslama açıları için üçer tane PID kontrolcü mevcuttur.(Sapma açısı ihmal edilmiştir.) Her PID kontrol cihazı parametresi, Ziegler-Nichols yöntemi kullanılarak hesaplanmıştır. Gerçek-zamanlı deney sonuçları verildiği için simulasyon sonuçları bu bölümde verilmemiştir. Simülasyon süresi 40 saniye olarak seçilmiştir ve her 5 saniyede bir veya birden fazla PID parametresi sistemi dengesiz yapan değerlerle değiştirilmektedir. 5 saniyelik aralıkların

seçilmesinin nedeni, kontrolcülerin sistemi dengeye getirebilmesi için 3 saniyenin yeterli olmasındandır.

Sistemi kararlı kılan PID parametreleri için PID'nin sağlığı 1 (Doğru) olarak tanımlanmıştır. Aksi takdirde, sistemin kararsızlığına neden olan PID parametreleri 0 (Yanlış) olarak tanımlanmıştır. Tablo 5.2 ve Şekil 5.9, senaryonun simülasyon detayları hakkında bilgi vermektedir. Şekil 5.9 olası durumları ve geçişleri açıklayan için bir Markov Diyagramı gösterilmiştir. Senaryoda, ortak olası sebepler, iki veya üç kontrolörün aynı anda ortak bir nedenden dolayı arızalanma olasılığı göz ardı edilmiştir.



Şekil 5.9 Sistemin markov diyagramı

Tablo 5.2 Simulasyon Senaryosu

Durumlar	Kararlılık			Kararsızlık	NVP-MV	Tasarlanan	Zaman(sn)
	V1	V2	V3	Dedektör	Karar	Seçmen Kararı	
S0	1	1	1	0	1-2-3	1-2-3	0-5
S1	1	1	0	0	1-2	1-2	5-10
S2	1	0	1	0	1-3	1-3	10-15
S3	0	1	1	0	2-3	2-3	15-20
S4	1	0	0	1	2-3	1	20-25
S5	0	1	0	1	1-3	2	25-30
S6	0	0	1	1	1-2	3	30-35
S7	0	0	0	1	1-2-3	SF	35-40

$S0$ durumu, tüm sürümlerin sistemi dengelediğini ve $V_{\varepsilon 123} < \varepsilon$ olduğunu gösterir; bu, tüm sürümlerin ε tanımlı küçük bir sayı olan aralığında olduğu anlamına gelmektedir.

Öte yandan, örneğin $S4$ durumu (20^{th} ile 25^{th} saniye arasındaki durumdur), yalnızca Sürüm 1'in (PID 1) bir dengeleyebilir bir çıkışı vardır, Sürüm 2 (PID 2) ve Sürüm 3 (PID 3) sistemi kararsız kılacak çıkışlar üretmektedir. NVP-MV algoritmasına göre geçerli karar Sürüm 2 ve Sürüm 3'ün çıkışını seçecektir. Geliştirilen NVP-MV seçmeni ise derhal azınlığın Sürüm 1 olan kararına geçecektir. Seçmen, nihai kararı bir sonraki örnekleme süresi olan 0.01 saniye içerisinde Sürüm 1 kararına değiştirecektir. 25^{th} saniye ile 30^{th} saniye arası $S5$ senaryosuna göre, Sürüm 1 ve Sürüm 3 kontrolcü parametreleri sistemi kararlı hale getiremezken Sürüm 2 kapalı döngü sistemi kararlı hale getirebilmekteddir. Geliştirilen seçmen bu sorunu derhal tespit eder ve azınlık kararından yana tercihini değiştirmektedir. Son olarak, son durumu $S7$'yi analiz edersek, tüm sürümlerin kararsızlığa neden olduğu sonucuna varırız, bu nedenle bir emniyet fonksiyonu (SF) [77] çağrılmalıdır.

Gerçek-zamanlı deney için, bir denetleyici kartı oluşturulmuştur ve NVP-MV algoritmasını uygulamak için bir mikroişlemci kullanılmıştır. Atalet ölçümü için, kontrol panosuna 9 DOF sensör kartı eklenmiştir. Sensör kartında, bu eksenler boyunca atalet varyasyonlarını ölçmek için üç eksenli jiroskop, ivme ölçer ve manyetometre bulunmaktadır. Sensör birleştirme algoritması ve filtre kullanılarak sensör verilerinin güvenirliği artırılmıştır. Platformda, servolar ve BLDC'ler Darbe Genişliği Modülasyonu (PWM) sinyalleri ile kontrol edilmektedir.

Deneyde, BLDC'nin başlangıç PWM değeri 1200μ saniye ve kontrol edilebilir aralık ise 1280μ - 1380μ saniye aralığında tanımlanmıştır.

1280μ saniye PWM değeri, platformun yunuslama hareketi için temel güvenli değeri temsil etmektedir. Bu nedenle, yunuslama kontrolünün PID çıkışı, 0 ila 100 aralığında ayarlanmıştır. Aynı şekilde, yatma PID çıkış aralığı -20 ila 20 arasında sabitlenmiştir. Test platformundaki servolar, BLDC'lerin dikey pozisyonları için PWM orta noktalarına (1800μsaniye) yerleştirilmişlerdir.

Servo PWM çalışma aralığı, orta noktadan -50μ saniye ile +50μ saniye olarak tanımlanmıştır. Bu şekilde, servolar BLDC'ler için ±5 derece eğim açısı değişikliği sağlamaktadır. Ek olarak, PID kontrolörleri denge noktaları çevresinde ölü bant aralıklarına sahiptir.

NVP-MV algoritması, yatma ve yunuslama PID denetleyicileri için uygulanır. Her iki PID denetleyicisi NVP-MV algoritmasıyla aynı anda incelenir. Platform dengeleme noktası sıfıra eşit bir yatma ve yunuslama açısı olarak düzenlenmiştir. Bu nedenle, PID

kontrol cihazının istenen referans değeri, yatma ve yunuslama için de sıfıra ayarlanır. Başlangıçta, sistem yatma açısı olarak sıfıra, yunuslama açısı ise $-40°$ ile hizalama işlemi yapılır.

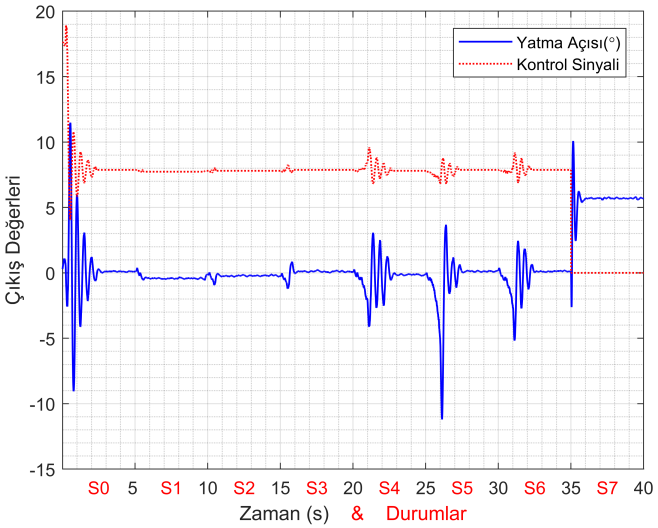
Deneyde, aynı parametrelere sahip üç ayrı PID kontrolcü, yatma ve yunuslama kontrolleri için tasarlanmıştır. PID kontrolcü parametreleri belirlenmiştir. Ayrıca, sistem kararsızlığına yol açabilecek PID parametreleri de aynı yöntem kullanılarak belirlenmiştir. Bu PID kontrolcü parametreleri Tablo 5.3'de gösterilmiştir.

Tablo 5.3 PID Parametreleri

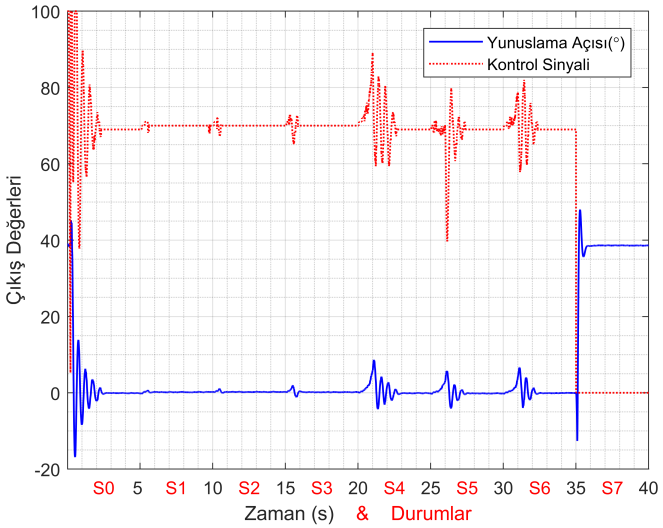
			Kararlı Değerler	**Kararsız Değerler**
Yatma	(P)	Oransal	3.2	0
	(I)	İntegral	0.9	500
	(D)	Türevsel	0.4	0
Yunuslama	(P)	Oransal	3.5	0
	(I)	İntegral	1.2	500
	(D)	Türevsel	0.6	0

Kontrol kartı sensörleri okumak ve PID çıkışlarını hesaplamak için 100Hz frekansına sahiptir. Bu nedenle, elektronik hız kontrol cihazlarının (ESC) ve servoların PWM sinyalleri her 10ms'de bir güncellenmektedir. Ayrıca, tüm sistem parametreleri her 10ms'de bir seri arabirim üzerinden izlenmektedir.

Şekil 5.10 ve 5.11 sırasıyla sistemin ve karşılık gelen kontrol sinyallerinin yatma ve yunuslama tepkisini gösterilmektedir.



Şekil 5.10 Yatma açısı senaryo cevabı

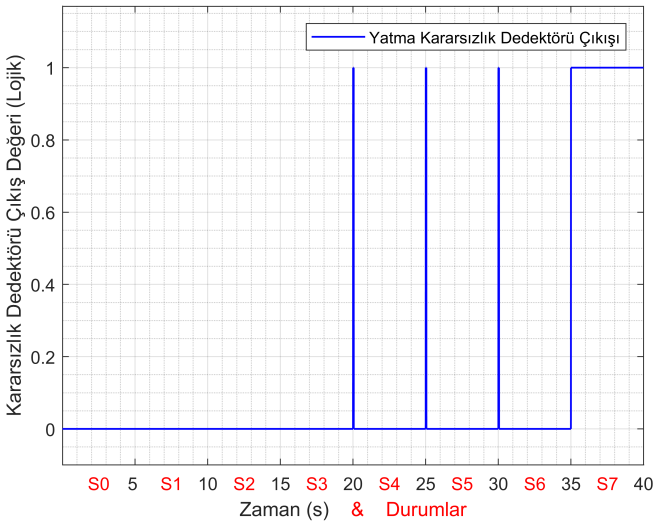


Şekil 5.11 Yunuslama açısı senaryo cevabı

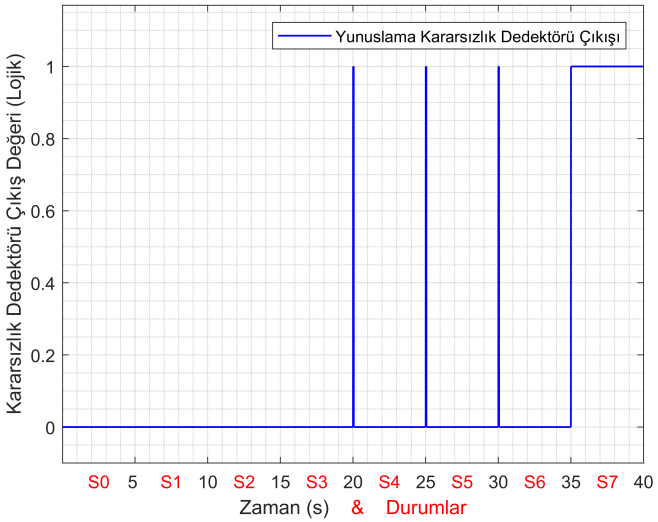
Şekil 5.10 ve 5.11'da, kontrol sinyali çıkışları ESC'lerin ve servoların PWM girişi ile ilişkilidir. Sistem çıktısı, sistemin yatma ve yunuslama açısını doğrudan temsil eder. Yatma kontrolü için, Yatma PID çıkışı eklenir ve BLDC'lerin karşılık gelen PWM değerinden çıkarılır. Öte yandan, servo yunuslama açısına kontrol eden PID çıktısı servoların eğim açısı ESC taban PWM değeri ile çarpımı, ile yeterli güvenceye gerekli güveni sağlamaktadır.

$S0$-$S3$ durumu için Şekil 5.10 ve 5.11'de, kararsızlık detektörü çıkışları 0'dır (Yanlış), çünkü her zaman iki kontrolör sistemi kararlı kontrol sisyali üretilmektedir. $S4$ durumu bize sadece Sürüm 1'in (PID 1) sistemi kararlı bir hale getirecek bir kontrol sinyalini ürettiğini, Sürüm 2 (PID 2) ve Sürüm 3'ün (PID 3) sistemi kararsız hale getirdiğini söylüyor, ancak çoğunlukta oldukları için NVP-MV Sürüm 2 ve Sürüm 3 çıkışlarını seçer. Geliştitirilmiş NVP-MV ürünümüz derhal Sürüm 1 olan azınlık kararına geçer. Seçmen, nihai kararı bir sonraki örnekleme süresinde 0.01'lik bir süre içinde değiştirdi. $S4$ durumu gibi, yalnızca $S5$ durumu sürüm (PID 2) ve yalnızca $S6$ durumu Sürüm 3 (PID 3) sistemi istikrarlı kılan dengeleyici bir kontrol sinyali üretmektedir.

Şekil 5.12 ve 5.13, Yatma ve Yunuslama kararsızlık dedektörlerinin çıktısı, Şekil 5.10 ve 5.11'in ile aynı zaman damgasıyla çizildirilmiştir.



Şekil 5.12 Yatma açısı kararsızlık dedektörü çıkışı



Şekil 5.13 Yunuslama açısı kararsızlık dedektörü çıkışı

$S0$, $S1$, $S2$ için $S3$ kararsızlık dedektörlerinin çıktısının "Doğru" olamayacağını belirtilmektedir. Çünkü çoğunluk seçmeni "Doğru" çift denetleyiciyi seçmektedir. Ancak $S4$, $S5$, $S6$ çoğunluk seçmeninin sistemi istikrarlı kılan denetleyiciyi seçemediğini belirtmiştir. Kararsızlık dedektörü "Doğru" cevabını verdiğinde, seçmen azınlık denetleyicinin çıktısı ile karar vermektedir.

Sistem $S7$ durumunda ise, tüm kontrolcüler sistemi kararlı kılabilecek bir kontrol sinyali üretememektedir. Kararsızlık dedektörlerinin çıktısı doğru sinyali üretmekte ve sistemin emniyet fonksiyonunu çağırmaktadır.

NVP-MV, bir yazılımın güvenirliğini artırmak için etkili bir yaklaşımdır ve doğru ve hatalı sürümlerin doğru karar verilmesini gerektirir. Bunu yapmak için, algoritmalar derecelendirmesini kullanarak, çok sayıda hesaplama sonucu kümesi arasından doğru cevabın seçilmesi gerekir. Ayrıca, NVP-MV, emniyet-kritik uygulamalar ve yazılım kalitesini artırmak için pratik bir yaklaşımdır. Bununla birlikte, NVP-MV yanlış bir karar seçerse; başka bir deyişle, çoğunluk hatalı bir çıktı üretir, bu durum sistemin kararsızlığına neden olmaktadır. Bu yazıda NVP-MV, seçmen sistemin kararlılığını kontrol edecek ve sistemi kararsız hale getirdiği takdirde çoğunluğun kazanmasına her zaman izin vermeyecek şekilde değiştirilmiştir. Fikir, deneysel bir kurulumda, tilt-rotorlu sistemde gösterilmiştir ve önerilen seçmenlerin başarısı incelenmiştir. İlerleyen çalışmalarda girişleri ağırlıklandırılmış bulanık seçmen algoritmaları üzerine çalışılması planlanmaktadır. Ayrıca, sistemin kararsızlık dedektörlerinin kararlarının da güvenir olabilmesi için çoklu kararsızlık dedektörü üzerine çalışılacaktır.

6
SONUÇ ve ÖNERİLER

Emniyet-kritik yazılım standartlarının entegrasyonu çok zordur. Geliştirici genellikle emniyet gereksinimlerini yanlış anlamaktadır. Bu durumu açıklayarak, farklı yazılım kalitesi modelleri oluşturmak zorunda kalınmıştır. Bu çalışmanın ilk bölümlerinde IEC 61508-3 standardına ve SIL3 emniyet-kritik yazılım gereksinimlerine odaklanılmaktadır. IAR derleyicisi kullanılmış, üretilen C++ kodunun etkinliğini kontrol etmek için ise PC-Lint ve MISRA C++ kullanılmıştır. Tekil model entegrasyonu için basit kod örnekleri verilmiştir.

NVP-MV, yazılımın güvenirliğini arttırmada etkili bir yaklaşımdır. SCADE programlama ortamı, emniyet-kritik uygulamaları oluşturmayı kolaylaştırmaktadır. Bu çalışmanın bölümlerinde NVP-MV ile emniyet-kritik bir seçici sistem geliştirilmiştir. Seçici karşılaştırma sırasında sabit bir değer kullanmıştır. İleriki çalışmalarda ise seçici operatörünün ideal kazanç değerini seçmesi veya hesaplaması için, algoritmadaki karşılaştırma değeri sabit değil, başka bir fonksiyon ile hesaplanabilecektir. Böylelikle seçici operatörü daha akıllı bir hale getirilebilir. Ayrıca, seçici algoritma, sistemin çıktısını ve çıktıların bir ve iki çevriminin önceki değerini inceleyerek sistemin kararlılığı hakkında bilgi üretebilecektir. Seçici, bu verilere dayanarak sistemin farklı versiyonlarını seçerek dengeleyebilecektir.

Çalışmanın bir dördüncü bölümünde ise TOPSIS algoritmasıyla seçilecek sürüm sorununa çözüm sunulmuştur. Projenin tasarım aşamasında tekrarlanan veya benzersiz N alternatiflerinin daha da geliştirilebilmesi gerçeği, proje bütçesine uyum sağlamanın yanı sıra proje bütçesinin istenen düzeyde tutulması açısından da büyük bir avantaj sunmaktadır. Bu sonucu elde etmek için, çok kriterli karar verme yöntemlerinden biri olan TOPSIS algoritması kullanılmıştır.

NVP yöntemine geçerken bazen bu m alternatiflerinin en iyisinin, tekrarlanan veya tekrarlanamayan durumlar için en iyisi olmadığı durumuna rastlanmıştır. Sonuç olarak, emniyet-kritik yazılım geliştirme süreçleri için etkili bir metodoloji tanıtılmıştır.

Tek tek incelendiğinde en başarılı olan versiyonun NVP içerisinde kullanılırken tekrarlayan şekilde kullanılmasının başarısız olduğu görülmüştür. Ortak bir hata arızası durumunda tüm alternatiflerin kaybedilmesi anlamına gelmektedir. Hatanın giderilebilmesi için aynı donanımda bile farklı yazılımların aynı anda kullanılmasıyla başarının elde edilebildiği görülmüştür.

NVP-MV, emniyet-kritik uygulamaların yazılım kalitesini iyileştirmeye yönelik pratik bir yaklaşımdır. Ancak, NVP-MV sistemi kararsızlığa götürecek bir karar verirse; başka bir deyişle, eğer çoğunluk hatalı bir çıkış ürettirilmesine sebep olursa, bu sistemin kararsızlığına yol açmaktadır. Çalışmamızın son bölümünde; NVP MV seçici, sisteminin kararlılığını kontrol etmek için değiştirilmiştir. Sistemi kararsız hale getirecekse çoğunluğun kazanmasına izin vermeyecektir. Fikir, tilt-rotor denge sistemindeki deneysel bir konfigürasyonda gösterilmiştir ve önerilen seçicilerin başarısı incelenmiştir.

Gelecek çalışmalarda, girişleri ağırlıklı olan bulanık seçmen algoritmalarının incelenmesi planlanmaktadır. Ayrıca, sistemin kararsızlık dedektörlerinin kararlarını daha güvenilir hale getirmek için birden fazla kararsızlık dedektörü ile çalışılması planlanmaktadır.

Kaynakça

[1] IEEE, *Standard glossary of software engineering terminology (ıeee std. 610.12-1990)*, 1990.

[2] IEC, *IEC 61508 : Functional safety of electrical/electronic/ programmable electronic safety-related systems*, 2005. DOI: 10.1049/ic:20050489.

[3] B. Aleksa J. Carter, "Boeing 777 airplane information management system operational experience," in *Proceedings of the Digital Avionics Systems Conference (DASC)*, vol. 1, 1997, pp. 21–27, ISBN: 0-7803-4150-3. DOI: 10.1109/DASC.1997.635042.

[4] A. S. Ian Moir, *Aircraft systems*, 9. 2013, vol. 53, pp. 1689–1699, ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[5] D. Birsch, "Moral responsibility for harm caused by computer system failures," *Ethics and information technology*, vol. 6, no. 4, pp. 233–245, 2004. DOI: 10.1007/s10676-005-5609-5.

[6] N. G. Leveson C. S. Turner, "An Investigation of the Therac-25 Accidents," *Computer*, vol. 26, no. 7, pp. 18–41, 1993. DOI: 10.1109/MC.1993.274940.

[7] A. Mayr, R. Plösch, M. Saft, "Towards an operational safety standard for software: Modelling IEC 61508 part 3," in *Proceedings - 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, ECBS 2011*, 2011, pp. 97–104, ISBN: 9780769543796. DOI: 10.1109/ECBS.2011.8.

[8] C. Preschern, N. Kajtazovic, C. Kreiner, *et al.*, "Catalog of safety tactics in the light of the ıec 61508 safety lifecycle," in *Proceedings of VikingPLoP 2013 Conference*, 2013, p. 79.

[9] I. Sommerville, *Software engineering*. Addison-wesley, 2011.

[10] R. Ploesch, H. Gruber, C. Koerner, M. Saft, "A Method for Continuous Code Quality Management Using Static Analysis," *2010 Seventh International Conference on the Quality of Information and Communications Technology*, pp. 370–375, 2010.

[11] L. Rierson, *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*. CRC Press, 2013.

[12] A. Landi M. Nicholson, "ARP4754A/ ED-79A - Guidelines for Development of Civil Aircraft and Systems - Enhancements, Novelties and Key Topics," *SAE Int. J. Aerosp.*, vol. 4, no. 2, pp. 871–879, 2011, ISSN: 19463855. DOI: 10.4271/2011-01-2564.

[13] L. Rierson, *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*. 2013, p. 610, ISBN: 1439813698.

[14] I. E. Workbench, *Iar systems ab*.

[15] J. Anderson, "Using software tools and metrics to produce better quality test software," in *Proceedings AUTOTESTCON 2004.*, IEEE, 2004, pp. 293–297.

[16] J. Gimpel, "Software that checks software: The impact of pc-lint," *IEEE software*, vol. 31, no. 1, pp. 15–19, 2014.

[17] M. Schreiber, E. Delic, A. Hayek, J. Börcsök, "Concept for a sıl3 middleware encapsulating safety-related aspects of applications for an 8051-based sıl3 multi-core system-on-chip," in *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, 2013, pp. 81–84.

[18] R. Erich Gamma, R. Helm, J. Vlissides, "Design patterns: Elements of reusable object-oriented software," *AddisonWesley Professional*, 1994.

[19] E. Technologies, "Getting started with scade suite," Tech. Rep., 2013, pp. 5–6.

[20] A. Avizienis, "The N-Version Approach to Fault-Tolerant Software," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 12, pp. 1491–1501, Dec. 1985. DOI: 10.1109/TSE.1985.231893.

[21] Liming Chen A. Avizienis, "N-VERSION PROGRAMMINC: A FAULT-TOLERANCE APPROACH TO RELlABlLlTY OF SOFTWARE OPERATlON," in *Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'.*, p. 113, ISBN: 0-8186-7150-5. DOI: 10.1109/FTCSH.1995.532621.

[22] A. Avizienis, "The methodology of n-version programming," *Software fault tolerance*, vol. 3, pp. 23–46, 1995.

[23] E. Dincel, O. Eris, S. Kurtulan, "Automata-based railway signaling and interlocking system design," *IEEE Antennas and Propagation Magazine*, vol. 55, no. 4, pp. 308–319, 2013. DOI: 10.1109/MAP.2013.6645212.

[24] B. Chitsaz M. Razzazi, "Non-Blocking N-Version Programming for Message Passing Systems," Tech. Rep., 2012, pp. 345–348.

[25] F. Zhang, Y.-C. Jhi, D. Wu, P. Liu, S. Zhu, "A first step towards algorithm plagiarism detection," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis - ISSTA 2012*, New York, New York, USA: ACM Press, 2012, p. 111, ISBN: 9781450314541. DOI: 10.1145/2338965.2336767.

[26] S. Aghaei, M. R. Khayyambashi, M. A. Nematbakhsh, "A fault tolerant architecture for web services," in *2011 International Conference on Innovations in Information Technology*, IEEE, Apr. 2011, pp. 53–56, ISBN: 978-1-4577-0311-9. DOI: 10.1109/INNOVATIONS.2011.5893867.

[27] M. S. Durmuş, O. Eriş, U. Yildirim, M. T. Söylemez, "A new voting strategy in Diverse programming for railway interlocking systems," *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering, TMEE 2011*, pp. 723–726, 2011. DOI: 10.1109/TMEE.2011.6199304.

[28] L. Nagy, R. Ford, W. Allen, "N-version programming for the detection of zero-day exploits," *IEEE Topical Conference on Cybersecurity Daytona Beach Florida USA*, no. April, 2006.

[29] E. Nourani, *A New Architecture for Dependable Web Services Using N-Version Programming*. IEEE, Mar. 2011, vol. 2, pp. 333–336, ISBN: 978-1-61284-839-6. DOI: 10.1109/ICCRD.2011.5764144.

[30] O. Cetinkaya D. Cetinkaya, "Verification and validation issues in electronic voting," *The Electronic Journal of e-government*, vol. 5, no. 2, pp. 117–126, 2007.

[31] Y. S. Dai, M. Xie, K. L. Poh, S. H. Ng, "A model for correlated failures in N-version programming," *IIE Transactions (Institute of Industrial Engineers)*, vol. 36, no. 12, pp. 1183–1192, 2004. DOI: 10.1080/07408170490507729.

[32] a. Avizienis, "A fault tolerance infrastructure for high-performance COTS-based computing in dependable space systems," *10th IEEE Pacific Rim International Symposium on Dependable Computing, 2004. Proceedings.*, no. June 2000, p. 7695, 2004. DOI: 10.1109/PRDC.2004.1276585.

[33] M. Yatsynovich, P. Litvinko, K. Kovalev, A. Bozhko, "Evaluation of interference signal from the space allocated abonent stations," in *2012 22nd International Crimean Conference" Microwave & Telecommunication Technology"*, IEEE, 2012, pp. 255–256.

[34] S. D. Liburd 1. Liburd, Soyini (Soyini Denise), "An N-Version Electronic Voting System," May 2004.

[35] M. Xie, C. Xiong, S.-H. H. Ng, "A study of N-version programming and its impact on software availability," *International Journal of Systems Science*, vol. 45, no. 10, pp. 2145–2157, Oct. 2014, ISSN: 14645319. DOI: 10.1080/00207721.2013.763299.

[36] R. Peng, Y. F. Li, W. J. Zhang, Q. P. Hu, "Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction," *Reliability Engineering and System Safety*, vol. 126, pp. 37–43, 2014, ISSN: 09518320. DOI: 10.1016/j.ress.2014.01.004.

[37] K. Salako L. Strigini, "When does 'diversity' in development reduce common failures? Insights from probabilistic modeling," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 2, pp. 193–206, Mar. 2014, ISSN: 15455971. DOI: 10.1109/TDSC.2013.32.

[38] S. Młynarski, R. Pilch, M. Smolnik, M. Szkoda, J. Szybka, "Evaluation Of The Safety Integrity Level (SIL) Due To The Guidelines Of EN 61508 And With The Use Of Markov Processes," *Journal of KONBiN*, vol. 35, no. 1, Jan. 2015, ISSN: 2083-4608. DOI: 10.1515/jok-2015-0041.

[39] S. Lee, X. Bao, T. Zhao, "A safety-critical software development strategy based on theory of diverse design," *The Proceedings of 2011 9th International Conference on Reliability Maintainability and Safety*, pp. 694–699, 2011. DOI: 10.1109/ICRMS.2011.5979354.

[40] X. Cai, M. R. Lyu, M. A. Vouk, "An experimental evaluation on reliability features of N-version programming," in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, vol. 2005, 2005, pp. 161–170, ISBN: 0769524826. DOI: 10.1109/ISSRE.2005.7.

[41] S. Chatterjee, R. B. Misra, S. S. Alam, "N-version programming with imperfect debugging," *Computers and Electrical Engineering*, vol. 30, no. 6, pp. 453–463, Sep. 2004, ISSN: 00457906. DOI: 10.1016/S0045-7906(04)00025-4.

[42] D. R. P. Williams, "Study of the warranty cost model for software reliability with an imperfect debugging phenomenon," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, no. 3, pp. 369–381, 2007, ISSN: 13000632.

[43] B. Littlewood, P. Popov, L. Strigini, "N-version design Versus one Good Version of the DISPO Technical Report of the same title) Software Diversity as a way of achieving high reliability of software," 2000.

[44] B. Parhami, "Approach to component based synthesis of fault tolerant software," *Informatica (Ljubljana)*, vol. 25, no. 4, pp. 533–543, 2001, ISSN: 03505596.

[45] A. Kumar K. Malik, "Voting Mechanisms in Distributed Systems," *IEEE Transactions on Reliability*, vol. 40, no. 5, pp. 593–600, 1991. DOI: 10.1109/24.106783.

[46] A. Armoush, F. Salewski, S. Kowalewski, "Recovery Block with Backup Voting: A new pattern with extended representation for safety critical embedded systems," in *Proceedings - 11th International Conference on Information Technology, ICIT 2008*, 2008, pp. 232–237, ISBN: 9780769535135. DOI: 10.1109/ICIT.2008.60.

[47] M. Vouk, D. McAllister, D. Eckhardt, K. Kim, "An empirical evaluation of consensus voting and consensus recovery block reliability in the presence of failure correlation," *Journal of Computer and ...*, no. 919, 1993.

[48] G. Latif-Shabgahi, J. M. Bass, S. Bennett, "A taxonomy for software voting algorithms used in safety-critical systems," *IEEE Transactions on Reliability*, vol. 53, no. 3, pp. 319–328, 2004. DOI: 10.1109/TR.2004.832819.

[49] A. Mohamed M. Zulkernine, "Improving reliability and safety by trading off software failure criticalities," in *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, 2007, pp. 267–274, ISBN: 0769530435. DOI: 10.1109/HASE.2007.44.

[50] S. Yacoub, "Analyzing the behavior and reliability of voting systems comprising tri-state units using enumerated simulation," *Reliability Engineering and System Safety*, vol. 81, no. 2, pp. 133–145, Aug. 2003, ISSN: 09518320. DOI: 10.1016/S0951-8320(03)00074-7.

[51] R. Y. Tsarev, M. Durmuş, I. Üstoglu, V. Morozov, "Classification of voting algorithms for n-version software," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1015, 2018, p. 042 060.

[52] P. Lorczak, A. Caglayan, D. Eckhardt, "A theoretical investigation of generalized voters for redundant\nsystems," *[1989] The Nineteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*, 1989, ISSN: 07313071. DOI: 10.1109/FTCS.1989.105617.

[53] R. Scott, J. Gault, D. McAllister, "Fault-Tolerant Software Reliability Modeling," *IEEE Transactions on Software Engineering (TSE)*, vol. 13, no. 5, pp. 582–592, 1987. DOI: 10.1109/TSE.1987.233463.

[54] B. Parhami, "Design of reliable software via general combination of N-version programming and acceptance testing," *Software Reliability Engineering, 1996. Proceedings., Seventh International Symposium on*, pp. 104–109, 1996, ISSN: 10719658. DOI: 10.1109/ISSRE.1996.558714.

[55] B. Parhami, "Voting Algorithms," *IEEE Transactions on Reliability*, vol. 43, no. 4, pp. 617–629, 1994, ISSN: 15581721. DOI: 10.1109/24.370218.

[56] H.-S. L. H.-S. Lee, "Aggregation of fuzzy opinions under group decision making\nenvironment," *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297)*, vol. 1, pp. 279–285, 2001, ISSN: 01650114. DOI: 10.1109/FUZZ.2001.1007275.

[57] R. Zwick, E. Carlstein, D. V. Budescu, "Measures of similarity among fuzzy concepts: A comparative analysis," *International Journal of Approximate Reasoning*, vol. 1, no. 2, pp. 221–242, 1987. DOI: 10.1016/0888-613X(87)90015-6.

[58] Z. Aghajani M. A. Azgomi, "A majority voter for intrusion tolerant software based on N-version programming techniques," *2009 International Conference on Innovations in Information Technology, IIT '09*, pp. 304–308, 2009. DOI: 10.1109/IIT.2009.5413780.

[59] K. S. Yim, V. Sidea, Z. Kalbarczyk, D. Chen, R. K. Iyer, "A fault-tolerant programmable voter for software-based N-modular redundancy," in *IEEE Aerospace Conference Proceedings*, 2012, ISBN: 9781457705564. DOI: 10.1109/AERO.2012.6187253.

[60] R. Y. Tsarev, M. S. Durmuş, I. Üstoglu, V. A. Morozov, "Application of majority voting and consensus voting algorithms in N-version software," *Journal of Physics: Conference Series*, vol. 1015, no. 4, p. 042 059, May 2018, ISSN: 1742-6588. DOI: 10.1088/1742-6596/1015/4/042059.

[61] B. Parhami, "Threshold voting is fundamentally simpler than plurality voting," *International Journal of Reliability, Quality and Safety Engineering*, vol. 01, no. 01, pp. 95–102, 1994, ISSN: 0218-5393. DOI: 10.1142/S021853939400009X.

[62] M. Islamoglu, M. Apan, A. Oztel, "An Evaluation of the Financial Performance of REITs in Borsa Istanbul: A Case Study Using the Entropy-Based TOPSIS Method," *International Journal of Financial Research*, vol. 6, no. 2, 2015, ISSN: 1923-4031. DOI: 10.5430/ijfr.v6n2p124.

[63] M. Behzadian, S. Khanmohammadi Otaghsara, M. Yazdani, J. Ignatius, "A state-of the-art survey of TOPSIS applications," *Expert Systems with Applications*, vol. 39, no. 17, pp. 13 051–13 069, 2012. DOI: 10.1016/j.eswa.2012.05.056.

[64] D. Wei, Y. Deng, Y. Li, Y. Zhang, S. Tang, "Multi-criteria safety evaluation index with topsis under uncertain environment," *ICIC Express Letters, Part B: Applications*, vol. 3, no. 1, pp. 83–89, 2012, ISSN: 21852766.

[65] M. Özdemir, "Topsis," *BF YILDIRIM, E. ÖNDER (Editör). İşletmeciler, Mühendisler ve Yöneticiler için Operasyonel, Yönetsel ve Stratejik Problemlerin Çözümünde Çok Kriterli Karar Verme Yöntemleri*, pp. 133–153, 2014.

[66] T. Wang, R. Jobredeaux, E. Feron, "A graphical environment to express the semantics of control systems," Aug. 2011. arXiv: 1108.4048.

[67] R. Lusser, "Reliability through safety margins," ARMY ROCKET and GUIDED MISSILE AGENCY REDSTONE ARSENAL AL, Tech. Rep., 1958.

[68] I. Dodd I. Habli, "Safety certification of airborne software: An empirical study," *Reliability Engineering and System Safety*, vol. 98, no. 1, pp. 7–23, 2012, ISSN: 09518320. DOI: 10.1016/j.ress.2011.09.007.

[69] "Functional safety of electrical/electronic/ programmable electronic safety-related systems," International Electrotechnical Commission, Geneva, CH, Standard, 2010.

[70] A. Kumar B. K. Sarkar, "Best programming practices using mısra-c and its compliance,"

[71] E. Technologies, "Qualifiable/certified scade suite kcg 6.4," Tech. Rep., 2013.

[72] I. Quanser, "Quanser 2-DOF Helicopter Laboratory Manual," 2011.

[73] N. Looker, M. Munro, J. Xu, "Increasing web service dependability through consensus voting," in *Proceedings - International Computer Software and Applications Conference*, vol. 2, 2005, pp. 66–69, ISBN: 0769522092. DOI: 10.1109/COMPSAC.2005.88.

[74] Y. A. Batawi O. A. Abulnaja, "Accuracy evaluation of arabic optical character recognition voting technique: Experimental study," *IJECS: International Journal of Electrical & Computer Sciences*, vol. 12, no. 1, pp. 29–33, 2012.

[75] H. Hu, C.-H. Jiang, K.-Y. Cai, W. E. Wong, A. P. Mathur, "Enhancing software reliability estimates using modified adaptive testing," *Information and Software Technology*, vol. 55, no. 2, pp. 288–300, 2013, ISSN: 0950-5849. DOI: http://dx.doi.org/10.1016/j.infsof.2012.08.012.

[76] A. Sanchez, J. Escareno, O. Garcia, R. Lozano, "Autonomous hovering of a noncyclic tiltrotor uav: Modeling, control and implementation," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 803–808, 2008.

[77] C. Easton, "Safety ıntegrity verification of legacy systems," *Measurement and Control*, vol. 42, no. 6, pp. 185–189, 2009.

Tezden Üretilmiş Yayınlar

Makale

1. ***Subaşı, N.***, Güner U., Üstoğlu İ.,"N-version programming approach with implicit safety guarantee for complex dynamic system stabilization applications" Measurement and Control (2020).

2. ***N. Subaşı***, İ. Üstoğlu, and M. S. Durmş, “Iec61508-3 ile emniyetli yazılım geliştirme”, Uluslararası Bilgi Güvenliği Mühendisliği Dergisi, vol. 3, no. 1, pp. 1–5.

Konferans Bildirisi

1. ***N. Subasi*** and I. Ustoglu, "Determining the best alternative for n-version programming based systems," 2018 6th International Conference on Control Engineering Information Technology (CEIT), pp. 784–790, 2018.

2. K.K. Candir, ***N. Subasi*** and I. Ustoglu, "ANSYS SCADE ile N-Versiyon PID Kontrolör Tasarımı," 2017 Otomatik Kontrol Türk Milli Komitesi Ulusal Toplantısı (TOK 2017), pp. 305-310,2017.