İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

FONKSİYONEL GÜVENLİK KAPSAMINDA ELEKTRİK MOTORU TAKVİYELİ DİREKSİYON SİSTEMİNİN MODEL TABANLI YAZILIMININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Cengiz AYDIN

Makine Mühendisliği Anabilim Dalı

Otomotiv Programı

TEMMUZ 2022

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

FONKSİYONEL GÜVENLİK KAPSAMINDA ELEKTRİK MOTORU TAKVİYELİ DİREKSİYON SİSTEMİNİN MODEL TABANLI YAZILIMININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Cengiz AYDIN
(503181703)

Makine Mühendisliği Anabilim Dalı

Otomotiv Programı

Tez Danışmanı: Dr. Öğr. Üyesi Osman Taha ŞEN

TEMMUZ 2022

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 503181703 numaralı Yüksek Lisans / Doktora Öğrencisi Cengiz AYDIN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "FONKSİYONEL GÜVENLİK KAPSAMINDA ELEKTRİK MOTORU TAKVİYELİ DİREKSİYON SİSTEMİNİN MODEL TABANLI YAZILIMININ GELİŞTİRİLMESİ" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Dr. Öğr. Üyesi. Osman Taha ŞEN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Özgen Akalın** İstanbul Teknik Üniversitesi	.....
	Doç. Dr. Tarkan Sandalcı Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : 03 Haziran 2022
Savunma Tarihi : 04 Temmuz 2022

Eşime,

ÖNSÖZ

Otomotiv endüstrüsi insan hayatındaki hareketliliğin önemli bir kısmını sağlayarak refah düzeyine doğrudan katkı vermektedir. Ancak ulaşılan yüksek hızlarla birlikte yaşanan kazalarda bir o kadar ölümcül olmaktadır. Bu nedenle sektör, trafik kurallarından, üretim standartlarına kadar bir çok yasal yükümlülükle iç içe girmiştir. Bu tez kapsamında otomotiv elektrik/elektronik sistemlerinin güvenliği ele alınarak, ISO 26262 Kara Araçları Fonksiyonel Güvenlik Standardı incelenmiş ve örnek bir çalışma ile pekiştirilmiştir.

Bu çalışmada bana destek olan tez danışmanım Dr. Öğr. Üyesi Osman Taha Şen'e ve katkı sağlayan tüm arkadaşlarıma teşekkür ederim.

Ayrıca manevi desteğini her an hissettiğim aileme ve eşime teşekkür ederim.

Haziran 2022

Cengiz Aydın
Yazılım Geliştirme Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

ASIC **:** Application Specific Integrated Circuit (Uygulamaya Özel Tümleşik Devre)

ASIL **:** Automotive Safety Integrity Level (Otomotiv Güvenliği Bütünlük Seviyesi)

AUTOSAR **:** AUTomotive Open System ARchitecture (Otomotiv Açık Sistem Mimarisi)

CPU **:** Central Process Unit (Merkezi İşlem Birimi)

E/E **:** Elektrik/Elektronik

EKÜ **:** Elektronik Kontrol Ünitesi

EMC **:** Electromagnetic Compatibility (Elektromanyetik Uyumluluk)

EMI **:** Electromagnetic Interference (Elektromanyetik Girişim)

ESD **:** Electrostatic Discharge (Elektrostatik Boşalma)

FTTI **:** Fault Tolerant Time Interval (Hataya Dayanıklı Zaman Aralığı)

HARA **:** Hazard Analysis and Risk Assesment (Tehlike Analizi ve Risk Değerlendirmesi)

ISO **:** International Organization for Standardization (Uluslararası Standartlar Teşkilatı)

MC/DC **:** Modified Condition/Decision Coverage (Değiştirilmiş Koşul/Karar Kapsamı)

NHTSA **:** National Highway Traffic Safety Administration (Ulusal Karayolu Trafiği Güvenliği İdaresi)

PMSM **:** Permanent Magnet Synchronous Motor (Sabit Mıknatıslı Senkron Motor)

RTE **:** Real Time Environment (Gerçek Zamanlı Ortam)

SAE **:** Society of Automotive Engineers (Otomotiv Mühendisleri Birliği)

SBC **:** System Basis Chip (Sitem Tabanlı Çip)

QM **:** Quality Management (Kalite Yönetimi)

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

FONKSİYONEL GÜVENLİK KAPSAMINDA ELEKTRİK MOTORU TAKVİYELİ DİREKSİYON SİSTEMİNİN MODEL TABANLI YAZILIMININ GELİŞTİRİLMESİ

ÖZET

Karayolları ulaşımında yaşanan kazalar nedeniyle her yıl yüz binlerce kişi hayatını kaybetmekte, sakat kalmakta veya ağır yaralanmalar geçirmektedir. Bu nedenle otomotiv üreticileri geliştirdikleri teknolojilerle insan hayatını koruyucu önlemler almayı hedef haline getirmektedir. Ayrıca markalaşma açısında da güvenilir olarak anılan markaların değeri ve prestiji diğer üreticilere göre oldukça yüksektir. Bu kapsamda elektronik kontrol ünitelerinin otomotiv sektöründe kullanımının artmasıyla birlikte pek çok elektronik güvenlik sistemi hayatımıza girmiştir.

Peki bu hayatlarımızı emanet ettiğimiz elektrik elektronik sistemlerin güvenliğinden nasıl emin olabiliriz? İhtiyacımız olduğunda bu sistemler sağlıklı bir biçimde fonksiyonlarını yerine getirebilecek mi? Bu şüpheler sadece elektronik güvenlik sistemleri için değil tüm elektronik kontrol üniteleri için geçerlidir. Amacı hayatımızı kolaylaştırmak olan bir elektronik kontrol ünitesindeki bir arızanın hayatımızı tehlikeye atmayacağından nasıl emin olabiliriz? Bu tehlikelere karşı hiçbir önlem almadan bu sistemleri kullanmak oldukça risklidir. Tabi ki de tüm riskleri ortadan kaldırmak mümkün değildir ancak bu sistemleri tasarlayan mühendislerden iyi bir risk yönetimi yaparak güvenli sistemler meydana getirilmesi beklenir. Risk ettiğimiz şey insan hayatı olduğunda, risk yönetimini standartlaştırmak kaçınılmaz olmaktadır. İşte bu nedenle 2011 yılında ISO 26262 – Kara Araçları Fonksiyonel Güvenlik Standardı hayatımıza girdi.

Bu çalışmada, otomotiv sektöründe çalışarak veya akademide yayınlar yaparak sektöre katkı yapanlar için ISO 26262 – Kara Araçları Fonksiyonel Güvenlik Standardı hakkında bilinmesi gerekenleri özetlemek, süreç hakkında bilgiler paylaşmak ve sektörel kabul görmüş yaklaşımları incelemektir.

Tezin literatür taraması sürecinde sadece ISO 26262 – Kara Araçları Fonksiyonel Güvenlik Standardı ile ilgili yapılan çalışmaları referans vermekle yetinilmemiş, sektörel olarak kabul edilen mimari ve konseptler de incelenmiştir.

Uygulama kısmında bir konsept oluşturabilmek için referans bir elektronik kontrol sistemine ihtiyaç duyuldu. Bunun için popüler elektronik kontrol sistemlerinden EPS (Electric Power Steering) seçildi ve sistem hakkında temel bilgiler paylaşıldı.

Yeterli temellerin oluşmasıyla detay tasarım konularına giriş yapıldı. Konsept aşamasında ve fonksiyonel güvenlik mühendisliğini ilgilendiren kısımlarda NHTSA (National Highway Traffic Safety Administration) tarafından yayınlanan “Functional Safety Assesment of a Generic Electric Power Steering System with Active Steering and Four-Wheel Steering Features” çalışması referans alınarak alıntılar yapıldı. Potansiyel tehlikeler ve riskler tanımlandı. HARA (Hazard Analysis and Risk

Assesment süreci açıklanarak ASIL'ler (Automative Safety Integrity Level) tanımlandı.

Konsept belirlendikten sonra, sistem seviyesinde fonksiyonel güvenlik gereksinimlerini sağlayacak teknik yaklaşımlardan bahsedilerek güvenlik mekanizmalarına kısaca değinildi. Güvenlik mekanizmalarının sistem seviyesinde doğrulanması için standart tarafından önerilen yöntemler açıklandı. Sistem seviyesi sadece yazılım geliştirme sürecine geliştirilecek fonksiyonları tanımlamakla değil aynı zamanda sistem entegrasyonu ve kabul testleri süreçlerinde de aktif rol almaktadır. Bu nedenle geliştirilen yazılımın, performans, tutarlılık, zamanlama, arayüz entegrasyonu, yetkinlik ve sağlanlık gibi kriterleri test ederken standart tarafından önerilen yöntemler incelendi.

Yazılım geliştirme sürecinde, iş akış organizasyonunun belirlendiği V-Döngüsüne değinildi. Modelleme ve yazılım geliştirme sürecinin bütünlüğünün önemi vurgulanarak seçilen geliştirme methodları için şartnamelerin belirlenmesinin sisteme katacağı değerler vurgulandı.

Yazılım test sürecinde MiL (Model in the Loop), SiL (Software in the Loop), PiL (Processor in the Loop), HiL (Hardware in the Loop), Gerçek donanımla test gibi farklı test ortamlarında yapılan testlerin farklılıkları geliştirme sürecine katkıları açıklandı.

Uygulama bölümünde, EPS sistemi direksiyon sensör modülü için yazılım gereksinimleri belirlendi. Mimari tasarım hakkında bilgi vermek adına Data Flow Diagram, Sequence Diagrams ve State Machine Diagramları paylaşıldı. Yazılım birim tasarım bölümünde Model Based Software Design teknikleri kullanılarak geliştirme süreci açıklandı. Yazılım birim doğrulama bölümünde statik kod analizi, dinamik testler, test koşum ortamları, kapsama analizleri gibi teknikler gösterildi

Model seviyesinde doğrulama yapabilmek için simülasyon ortamı oluşturuldu. Sistemin test edilebilmesi için EPS sistemi MATLAB/Simulink Simscape aracı kullanılarak modellendi ve kontrol algoritmaları sisteme entegre edildi. Farklı kullanım senaryoları için test senaryoları olışturuldu ve sistem sonuçları paylaşıldı.

Son olarak elde edilen çıktılar değerlendirilerek gelecek çalışmalar için fikirler verildi. Elektrik/Elektronik sistemlerde fonksiyonel güvenlik kopseptinin gelişmekte olduğuna vurgu yapılarak çalışma tamamlandı.

MODEL-BASED SOFTWARE DEVELOPMENT OF ELECTRIC MOTOR ASSISTED STEERING SYSTEM WITHIN THE SCOPE OF FUNCTIONAL SAFETY

SUMMARY

Every year, hundreds of thousands of people lose their lives, become permanently disabled or suffer serious injuries due to accidents in road transportation. For this reason, automotive manufacturers aim to take protective measures for human life with the technology which they have developed. In addition, the prestige of the brands that are known safe and reliable in terms of marketing are quite high compared to other manufacturers. In this context, with the increasing use of electronic control units in the automotive sector, many electronic safety systems have entered our lives. Systems such as ABS (Anti-Lock Braking System), EPS (Electronic Stability Program), Airbag can be given as examples of these developments.

So how can we be sure of the safety of the electrical and electronic systems to which we entrust our lives? Will these systems be able to function properly when we need them? These doubts are valid not only for electronic safety systems but also for all electronic control units. How can we be sure that a malfunction in an electronic control unit whose purpose is to make our life easier will not endanger our lives? It is quite risky to use these systems without taking any precautions against these dangers. Of course, it is not possible to eliminate all risks, but engineers who design these systems are expected to create safe systems by making good risk management. When what we risk is human life, it is inevitable to standardize risk management. That's why in 2011, ISO 26262 – Road Vehicles Functional Safety Standard came into our lives.

In this study, for those who contribute to the sector by working in the automotive sector or making publications in the academy, to summarize what should be known about the ISO 26262 - Rand Vehicles Functional Safety Standard, to share information about the process and to examine the accepted approaches industrially.

During the literature review process of the thesis, not only the studies on ISO 26262 – Land Vehicles Functional Safety Standard were given as reference, but also the architecture and concepts accepted as industry were examined.

As a result of the research, AUTOSAR (Automotive Open System Architecture), where almost all automotive manufacturers meet on a common ground, was examined. In the thesis, the relationship between ISO 26262 and AUTOSAR was mentioned. In this section, it was decided to focus on the application level rather than examining the fundamentals of basic software design in depth.

Another important topic Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units is mentioned. Although the E-Gas monitoring concept was prepared for electronic control units used in diesel and gasoline engines, it was mentioned that since the concept is based on solid foundations, it can also form a basis for different systems designed.

To create a concept in the application part, a reference electronic control system was needed. For this, EPS (Electric Power Steering), one of the popular electronic control systems, was selected and basic information about the system was shared.

The basis of the study is established with standards, architecture and concepts. In addition to these, necessary information was shared in the vocabulary section so that technical terms related to the standard do not cause confusion.

With the formation of sufficient foundations, detailed design issues were introduced. In the concept phase and in the parts related to functional safety engineering, quotations were made with reference to the "Functional Safety Assessment of a Generic Electric Power Steering System with Active Steering and Four-Wheel Steering Features" published by NHTSA (National Highway Traffic Safety Administration). In the item definition section, the boundaries of the EPS system are drawn and the components with which it interacts are determined. Potential hazards and risks have been identified. HARA (Hazard Analysis and Risk Assessment) process was explained and ASILs (Automotive Safety Integrity Level) were defined. Functional Safety Concept was created, and safe situations were determined. In order not to disperse the subject during the implementation process, functional safety requirements were listed by examining only the steering system sensors signals instead of examining a whole complex system.

After the concept was determined, technical approaches that would provide functional safety requirements at the system level were mentioned, and safety mechanisms were briefly mentioned. The methods recommended by the standard for system-level verification of safety mechanisms are explained. The system level not only defines the functions to be developed into the software development process, but also takes an active role in system integration and acceptance testing processes. For this reason, while testing the criteria such as performance, consistency, timing, interface integration, competence and reliability of the developed software, the methods recommended by the standard were mentioned.

During the software development process, the V-Cycle used to explain the workflow organization was mentioned. Emphasizing the importance of the integrity of the modeling and software development process, attention was drawn to the value that determining the specifications for the selected development methods would add to the system. In the steps of the software development process, specification of software safety requirements, software architectural design, software unit design and implementation, software unit verification, software integration and verification, testing of the embedded software, the approaches recommended by the standard are listed and comments are made.

During the software testing process, the differences of the tests made in different test environments such as MiL (Model in the Loop), SiL (Software in the Loop), PiL (Processor in the Loop), HiL (Hardware in the Loop), testing with real hardware, and their contribution to the development process were explained.

In the application section, the selected safety requirement set for the EPS steering system input signals was examined and the appropriate sensor module was determined. The technical information documents of the selected sensors are shared in the appendix. Necessary software requirements for security mechanisms have been determined. The software architecture that will provide the determined software requirements and concept was evaluated. Data Flow Diagram, Sequence Diagrams and State Machine Diagrams were shared to provide information about architectural design.

In the software unit design section, the development process was explained using Model Based Software Design techniques. Information was given about the

specifications and coding standard controls applied for Model Based Software Design. Automatic code generation and relationships between model and code and traceability processes are shown.

In the software unit validation section, techniques such as static code analysis, dynamic tests, test run environments, coverage analyzes were demonstrated. An example error and solution suggestion were shared while performing static code analysis. Coverage analysis was made for a sample state machine and dynamic test environment was created in this process.

A simulation environment was created to validate at the model level. In order to test the system, the EPS system was modeled using the MATLAB/Simulink Simscape tool and control algorithms were integrated into the system. Test scenarios were created for different use-case scenarios and system results were shared.

Finally, the obtained outputs were evaluated and ideas for future studies were given. The study was concluded by emphasizing that the functional safety concept in electrical/electronic systems is developing.

1. GİRİŞ

1.1 Motivasyon

Günümüzde, premium bir otomobil, 2000'den fazla fonksiyonu gerçekleştirebilmek için birden fazla veri yolu ile birbirine bağlanan 70'e kadar EKÜ'yü içermektedir [1]. Toplamda yaklaşık 100 milyon satır kod ile premium segment araçlar, modern savaş uçakları ve ticari uçaklardan daha fazla yazılım kodu taşımaktadır [2]. Sistemlerin sayısı ve karmaşıklığı arttıkça, birçok gömülü sistemin güvenlik, fonksiyonellik, yaşam döngüsü, işletme açısından gereksinimleri artmakta ve yüksek düzeyde güvenilirlik ve sağlamlık talep edilmeye devam edilmektedir [3]. Yaklaşık 15 yıl öncesine kadar gömülü sistemler temel olarak, elektromanyetik uyumluluk, elektrik testleri, çevresel şart testleri ve saha testleri olmak üzere 4 temel test türü ile doğrulanmaktaydı [4]. Ancak bu testlerde sistematik hataların tespiti, rastgele oluşan donanımsal arızaların risk değerlendirmeleri bir standarda bağlı olarak yönetilmiyordu. 2011 yılında ISO 26262 – Kara Araçları Fonksiyonel Güvenlik Standardının ilk versiyonunun yayınlanmasıyla bu kavramlar otomotiv elektroniğine girdi. Standart ile ürün geliştirme sürecindeki dokümantasyon arttırılarak paydaşlar arasındaki süreç takip edilebilir hale getirilerek iletişim güçlendirildi. Standard ürün yaşam döngüsü ve ürün geliştirme süreci için bir çerçeve çizmektedir. Bu tezin motivasyonu, ISO 26262 – Kara Araçları Fonksiyonel Güvenlik Standardı ile sektörel olarak kabul edilen tasarım mimarileri ile ilişkilerini göstererek farkındalık yaratmaktır.

2020 yılında otomotivdeki elektronik bileşenler toplam araç maliyetinin yaklaşık %35'ini oluşturuyordu. Teknolojideki gelişmelerle birlikte otomotiv elektronik bileşenlerinin 2030 yılına kadar toplam araç maliyetinin yaklaşık %50'sini oluşturması beklenmektedir [5]. Böylesine büyük bir maliyet kaleminde yapılacak iyileştirmenin etkisinin de büyük olacağından, otomotiv üreticileri elektronik komponent geliştirme sürecini geliştirecek yöntemler aramaktadır. Yazılım geliştirme sürecinde model tabanlı tekniklerin popülerliği her geçen gün artmaktadır. Hemen hemen her mühendislik disiplininde, sistem karmaşıklığını yönetmek için modeller

kullanılmaktadır [3]. Model tabanlı yazılım geliştirme, paydaşlar arasındaki anlaşılırlığı ve iletişimi geliştirir [6]. Modeller koddan soyutlanmış daha üst bir seviyeden bir bakış açısı sağlayarak farklı disiplinlerden çalışanların yazılım geliştirme sürecine daha aktif katkı verebilmesini sağlamaktadır. Model tabanlı ve nesne yönelimli programlama tekniklerinin ana motivasyonu, yeni uygulamaların geleneksel yaklaşımlardan çok daha az çabayla, sadece mevcut parçaları birleştirerek oluşturulabilmesidir [7].

Bunlara ek olarak model tabanlı yazılım geliştirme araçları model seviyesinde test ve simülasyon ortamları sağlayarak hataların erken tespitine olanak sağlar ve ürün geliştirme sürecindeki dokümantasyonları otomatize edecek ara yüzler sağlayarak verimliliği arttırmaktadır.

Tüm bu nedenlerle tezin uygulama bölümünde model tabanlı yazılım geliştirme aracı MATLAB Simulink kullanıldı.

1.2 Araştırma Sorunsalı

Tezin amacı, örnek olarak seçilmiş otomotiv gömülü sistemi için geliştirme sürecinde uygulanabilecek standartlar ve konseptler hakkında bilgileri aktarmaktır. Çalışma içerisinde değinilecek ve cevaplanacak sorular aşağıda listelenmiştir.

- 2011 yürürlüğe giren ISO 26262 Kara Araçları Fonksiyonel Güvenlik Standardı nedir ve günümüz otomotiv elektroniğine etkileri nelerdir?
- ISO 26262 Kara Araçları Fonksiyonel Güvenlik Standardına uygun ürün geliştirme süreci nasıl bir organizasyon gerektirir?
- 26262 Kara Araçları Fonksiyonel Güvenlik Standardının sektörel bazda kabul görmüş mimari ve konspetlere etkileri nelerdir?
- Model tabanlı yazılım geliştirme tekniğinin, geleneksel yazılım geliştirme methodlarına göre avantajları ve dezavantajları nelerdir?
- 26262 Kara Araçları Fonksiyonel Güvenlik Standardının, geleneksel prototipten ürüne geliştirme süreci üzerine etkileri nelerdir?

Bu soruları daha iyi açıklayabilmek için bölüm 2’de literatürde bulunan bilgiler paylaşıldı, bölüm 3’te süreç ve konsept detaylı bir şekilde anlatıldı. Son olarak bölüm 4’te örnek bir çalışma seçilmiş ve uygulama örneği yapıldı.

2. LİTERATÜR ARAŞTIRMASI

Bu bölümde literatürde bulunan, fonksiyonel güvenlik standartları, kullanılan mimariler, programlama dilleri, araçlar ve teknolojiler hakkında bilgi verilecektir. Tezin uygulama bölümünde sıkça bahsedilecek bu konular hakkında temel bilginin sağlanması amaçlanmıştır. Bunlara ek olarak, referans alınan çalışmalardan da bahsedilecektir.

2.1 ISO 26262 Kara Araçlarında Fonksiyonel Güvenlik Standardı

Uluslararası Standardizasyon Örgütü (ISO) tarafından ilk baskısı 2011 yılında yayınlanan ISO 26262 – Functional Safety – Road Vehicles standardı, maksimum brüt ağırlığı 3500 kg olan seri üretim binek otomobillerde kullanılan elektrikli ve/veya elektronik E/E sistemlerin fonksiyonel güvenliğine yönelik uluslararası bir standarttır. 2018 yılında revize edilerek mopedler hariç bütün karayolu araçlarını kapsayacak şekilde kapsamı genişletildi. Şekil 2.1'de ISO 26262 standardının içerdiği konular paylaşıldı.

Araçlarda kullanımı artan E/E sistemler ve komplex yazılımlar teknolojik karmaşıklığı beraberinde getirmiştir. Sistematik ve rastgele donanım arızalarından kaynaklanan risklerin değerlendirmesi ve önlemlerin alınması fonksiyonel güvenlik kapsamında ele alınmaktadır. ISO 26262 kara araçlarında fonksiyonel güvenlik standart serisi, seri üretim araçların yaşam döngüsü boyunca uygulanması gereken süreçler ve minimum gereksinimleri belirleyerek oluşacak riskleri azaltmayı amaçlamaktadır.

ISO 26262'nin Hedefleri:

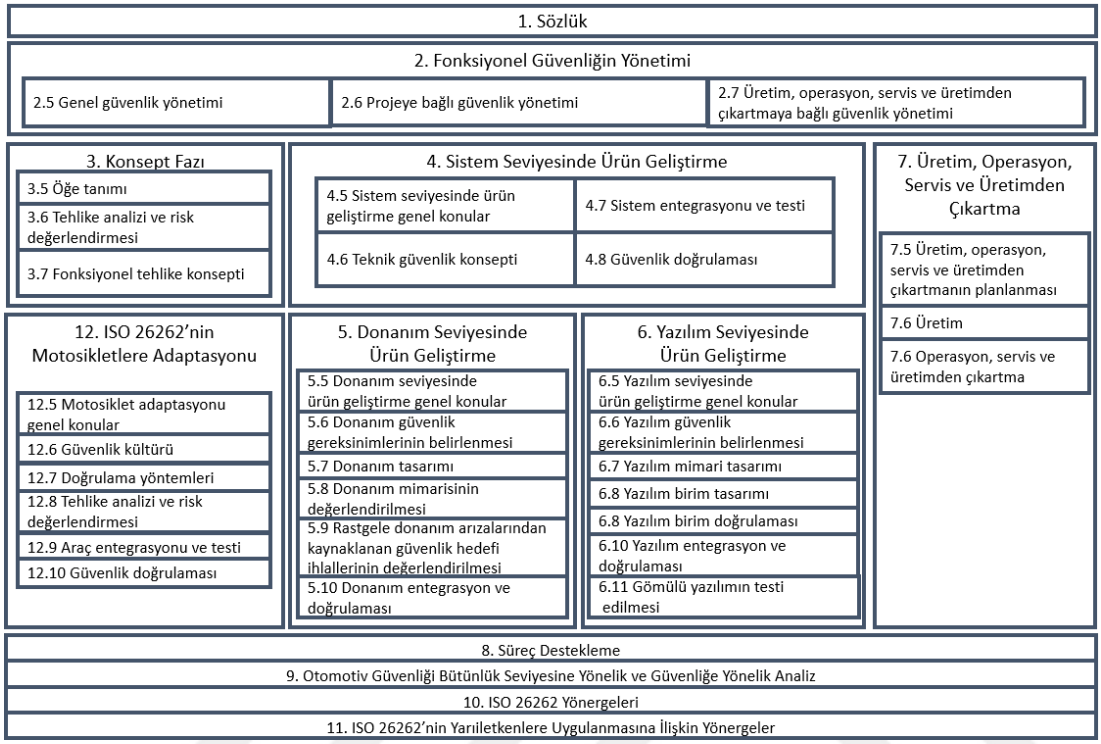
-Otomotiv güvenlik yaşam döngüsü için bir referans sağlamak ve yaşam döngüsü aşamalarında gerçekleşrtirilecek geliştirme, üretim, işletim, servis ve hizmetten çıkartma gibi faaliyetleri desteklemek,

-Risk sınıflarının (Automotive Safety Integrity Level , ASIL) belirlenmesi için otomotive özgü risk tabanlı bir yaklaşım sağlamak,

-Makul olmayan risklerden kaçınmak için ASIL seviyelerine bağlı olarak minimum gereksinimleri belirlemek,

-Tüm geliştirme süreçleri (yönetim, tasarım, devreye alma, doğrulama ve sağlama) için gereksinimleri sağlamak,

-Müşteri ve tedarikçiler arasındaki gereksinimleri sağlamaktır.



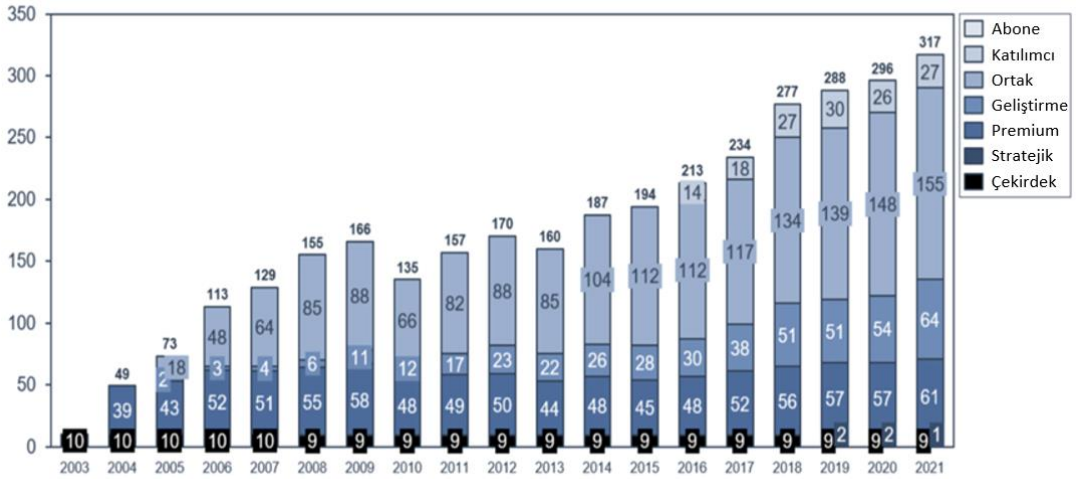
Şekil 2.1 : ISO 26262 standardı içeriği [8].

Sonuç olarak ISO 26262 Fonksiyonel Güvenlik standardının temel amacı riski en aza indirmektir. Bunu sağlamak için düzenli olarak fonksiyonel güvenlik kapsamında oluşturulan çıktılar değerlendirilmekte ve teftiş edilmektedir. Sistematik ve rastgele donanım hataları meydana geldiğinde sistemin güvenli durumlarda kalarak, diyagnostik oluşturması ve fonksiyonları monitör ederek hataları tolere etmesi beklenir.

2.2 AUTOSAR

AUTomotive Open System Architecture (AUTOSAR), 2003 yılında BMW, Robert Bosch GmbH, Continental AG, Daimler AG, Siemens VDO ve Volkswagen ile başlayan, daha sonrasında Ford Motor Company, Groupe PSA, Toyota ve General Motors'un da katılımıyla çekirdek ortakların bir araya gelerek otomotiv sektöründe kullanılan elektronik kontrol ünitelerinde açık ve standartlaştırılmış bir yazılım

mimarisi oluşturmayı hedefleyen bir ortaklıktır. AUTOSAR'ın kurulumunda yer almış ve farklı şekillerde görev almış firmalrın yıllara göre dağılımı Şekil 2.2'de paylaşıldı.



Şekil 2.2 : AUTOSAR katılımcılarının dağılımı [9].

Teknik açıdan bakıldığında, standardın motivasyonu aşağıdaki maddelerce özetlenebilir:

- Artan fonksiyonalite ile E/E sistemlerin karmaşıklığını yönetebilmek.
- Ürün modifikasyonu, iyileştirme ve güncellemelerindeki esnekliği arttırmak.
- Üretim hattında veya sonrasında çözümlerin ölçeklenebilirliğini[1] arttırmak
- E/E sistemlerin kalite ve güvenilirliğini arttırmak.
- Erken tasarım aşamalarında hataların tespit edilebilmesini sağlar.

AUTOSAR mimarisi 5 katmana ayrılır. Her katman bir üst seviyeye geçildikçe yazılımın soyutlanma seviyesini arttırarak AUTOSAR arayüzleri üzerinden haberleşir. Bu katmanların birbiri ile etkileşimini gösteren mimari Şekil 2.3'de görülmektedir.

Microdenetleyici Soyutlama Katmanı: Bu katman mikrodenetleyiciye doğrudan erişimi olan sürücüleri (Driver) içerir. Microdenetleyiciye bağımlıdır. Temel amacı üst katmanları mikrodenetleyiciden bağımsızlaştırmaktır.

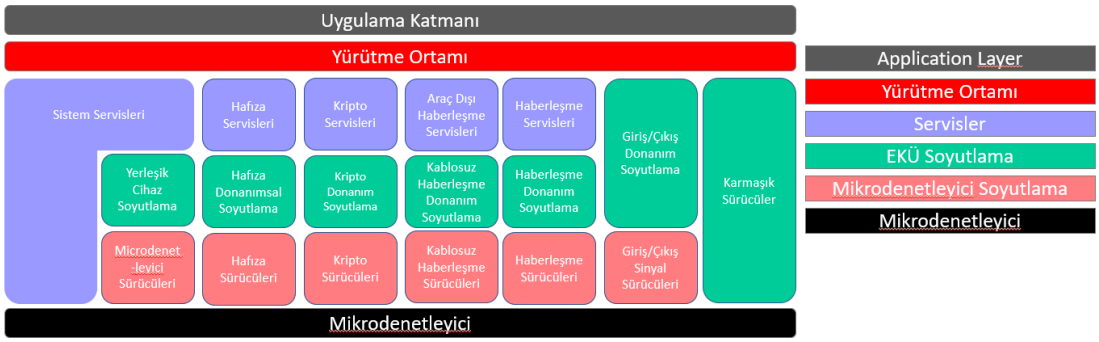
EKÜ Soyutlama Katmanı: Bu katman çevre birimlerin (peripheral) CPU'ya nasıl bağlandığını soyutlayan yazılım modülleri olan işyeliyicileri (handler) içerir. Bunun haricinde AUTOSAR'da standartlaşmamış enjektör sürücü gibi harici cihazların sürücüleri de bu katmanda yer alır.

[1] Büyüyen, gelişen, artan isteklere yanıt vermesi gereken bir sistemin, çalışmanın, işlemin veya yazılımın bu isteklere cevap verme, yönetme ve sorunlarla başa çıkmak yeteneğidir.

Servisler Katmanı: Servis katmanı uygulama ve diğer yazılımlar için gerekli işletim sistemi, ağ iletişimi, hafıza yönetimi, tanı servisleri gibi temel yazılım modüllerini içermektedir.

AUTOSAR Çalıştırma Ortamı: Bu katman uygulama katmanı ile temel yazılım katmanı arasındaki köprüdür. Bu köprü sayesinde uygulama katmanı ve temel yazılımların yanı sıra uygulama katmanındaki farklı komponentler de veri alışverişi yapabilmektedir.

Uygulama katmanı: Geliştirilen yazılımın bulunduğu katmandır. Tamamen donanımdan soyutlanmıştır.



Şekil 2.3 : AUTOSAR katmanlı yazılım mimarisi [9].

"AUTOSAR Katmanlı yazılım mimarisiyle bir otomobilde ya da bir EKÜ'de kullanılacak yazılım, donanımdan (EKÜ'den) bağımsız bir şekilde geliştirilebilir ve başka projelere aktarılabilir. Bunun yanında iletişim, hata tanımlama gibi alt yapısal ve tüm otomotiv üreticilerinin ortak kullandığı yazılımlar, standart yazılım pakedi olarak yazılım üreticileri tarafından temin edilir, otomotiv üreticileri ve yan sanayi tedarikçileri ise sadece bu yazılım paketlerini belirli masaüstü programlarını kullanarak yapılandırır. (Configuration)"

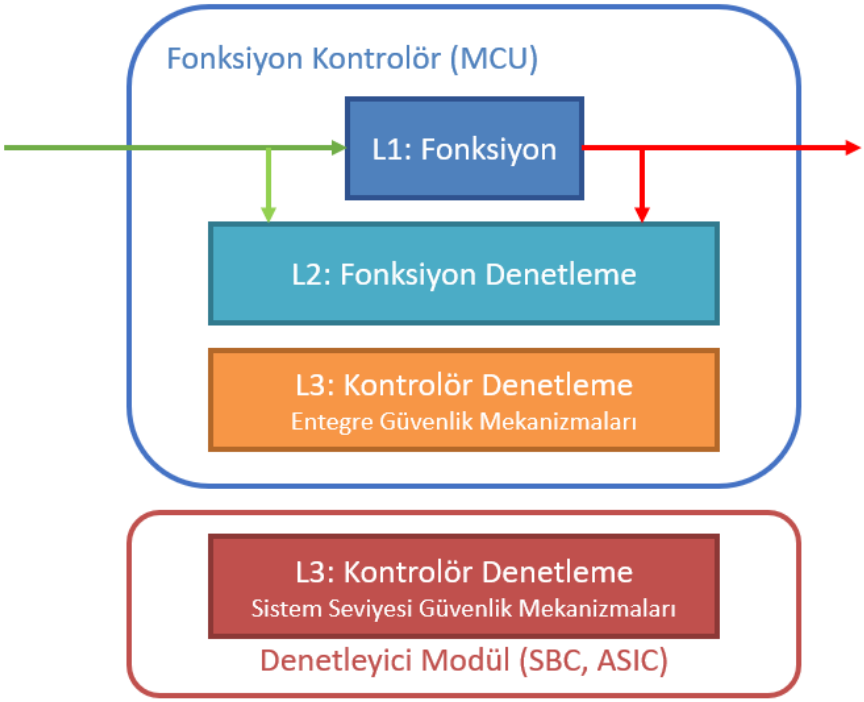
Ortaklığın mottosu "Cooperate on standards, compete on implementation" (Standartlarda işbirliği, uygulamada rekabet) 'tir. Temel geliştirmelerde ortak çalışarak yüksek standartları yakalayabilmekte ve enerjilerini uygulama alanında yapacakları inovatif gelişmelere harcamaktadırlar.

2.3 E-Gas

Otomotiv sektöründeki yüksek standartlar ve elektronik kontrol ünitelerinin birbirine entegre bir araç ağı içerisinde çalışmasından dolayı yüksek izlenebilirliği gereksinim

haline getirdi. Bir grup Alman otomotiv üreticileri bu ortak problemi çözmede bir markalaşma farkı görmemesi üzerine bu konu üzerinde birlikte çalışmaya başladı. Böylelikle standartlaştırılmış bir motor kontrol ünitesi izlenebilirlik konseptini geliştirdiler. BMW AG, Daimler AG, Volkswagen AG, Porsche AG ve AUDI AG'nin bir araya gelmesiyle kurulan ortaklık 97 yılında bu standardın ilk versiyonunu yayınladı. 2013'te yayınlanan 5.5 sürümünden itibaren E-GAS konsepti ISO 26262'ye uyumlu hale gelmiştir.

EGAS konsepti motor kontrol üniteleri için geliştirilmiş olsa da izlenebilirlik konsepti diğer uygulamalara da adapte edilebilmektedir. İzlenebilirlik konsepti 3 seviyeden oluşmaktadır. Bu seviyelerin hafızadaki yerleri ve kullandıkları hafıza alanları birbirinden ayrı olmalı ve alt seviyeler üst seviyelerin alanlarına erişememelidir. Mimarinin katmanlar arasındaki etkileşimi Şekil 2.4'de paylaşıldı.



Şekil 2.4 : E-Gas 3 katmanlı denetim konsepti [10].

Birinci seviye fonksiyon seviyesidir. Bu seviyede motor kontrol fonksiyonları, komponent denetleme, ve diyagnostik oluşması halinde gerekli sistem tepkilerini vermektedir.

İkinci seviye fonksiyon denetleme seviyesidir. İkinci seviye, belirlenen giriş ve çıkış sinyallerini izleyerek mantıksal işlemlerle birinci seviyeyi denetler. Birinci seviyedeki hatayı tespit ettiğinde sistem reaksiyonlarını tetikler.

Üçüncü seviye kontrolör denetleme seviyesidir. Mikrodenetleyicinin doğru çalıştığını sistematik olarak test eden bağımsız bir ASIC (application specific integrated circuit) veya başka bir kontrolör kullanılmaktadır. Kullanılan bağımsız entegre veya kontrolör, mikrodenetleyiciye belirli zaman aralıklarında sorular sorar ve cevaplarını bekler. Eğer mikrodenetleyici beklenilen cevabı veremezse belirlenen sistem reaksiyonları uygulanır. Bunun haricinde ikinci seviye program akışı da bu seviyede denetlenmektedir. Son olarak ikinci ve üçüncü seviye tarafından kullanılan tüm çevre birimlerin initializasyonu ve periyodik diagnostik kontrollerini gerçekleştirir.

Otomotiv elektroniğinde artan karışıklığı sistematik bir şekilde ele alabilmek için, fonksiyonel güvenliği, fonksiyonaliteden bağımsız korumalı bir katmanda ele almak avantaj sağlamaktadır. Birinci katmanda oluşan bir hatanın, ikinci katmanda da meydana gelmesi düşük bir olasılıktır. Bu nedenle, risk olasılığını azaltmak için katmanlı yapılar kullanmak oldukça önemlidir. Bu yaklaşım tüm otomotiv sistemlerinde kullanılmaya elverişli bir yaklaşımdır.

2.4 MISRA C

MISRA C, Motor Industry Software Reliability Association (MISRA) tarafından geliştirilen C dilinde otomotiv sektörü yazılım geliştiricileri için hazırlanmış bir şartnamedir. Her ne kadar otomotiv hedeflenerek geliştirilse de, havacılık, telekomünikasyon, tıbbi cihazlar, savunma sanayii gibi bir çok sektör tarafından da kabul görmüştür. ISO 26262-6:2018 bölüm 6 yazılım seviyesinde ürün geliştirme bölümünde kodlama ve modelleme şartname gereksinimleri açıklandığı bölümde MISRA C referans şartname olarak paylaşılmış ve kullanılması durumunda Çizelge 2.1’de paylaşılan gereksinimlerin büyük bir bölümünün sağlanacağı ifade edilmiştir.

Çizelge 2.1 : Yazılım birimi tasarımı ve uygulaması için tasarım ilkeleri [8]

Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
Alt programlarda ve fonksiyonlarda tek giriş ve tek çıkış noktası	++	++	++	++
Oluşturulma sırasında online testler yapılsa dahi dinamik obje ve değişkenlerin kullanılmaması	+	++	++	++
Değişkenlerin başlangıç durumlarının belirlenmesi	++	++	++	++
Değişken isimlerinin çoklu kullanılmaması	++	++	++	++
Global değişkenlerin kullanımından kaçınılması, kullanılması durumunda gerekçelendirilmesi	+	+	++	++
İşaretçi kullanımının kısıtlanması	+	++	++	++
İstemsiz veri tipi dönüştürülmemesi	+	++	++	++
Gizli veri akışı ve kontrol akışının olmaması	+	++	++	++
Koşulsuz atlamaların olmaması	++	++	++	++
Özyinelemelerin olmaması	+	+	++	++

Ayrıca AUTOSAR 4.3 Genel Yazılım Spesifikasyonu(Link verilebilir kaynaklara eklenecek), BSW Modülü uygulaması C dilinde yazılmışsa, MISRA C:2012 Standardına uygun olmasını gerektirir.

MISRA C'nin her bir şartname maddesi zorunlu, gerekli ve tavsiye olarak sınıflandırmıştır. Zorunlu maddelere her zaman uyulmalıdır. Gerekli maddelere uyulmalı ancak herhangi bir nedenden dolayı uyulamaması durumunda deviasyon dokümanına eklenmeli ve yazılım mühendisinin sistemin güvenliğini göz önünde bulundurduğuna ve kuraldan sapmanın olumsuz bir etkisi olmayacağına dair kanıt sunulmalıdır. Tavsiye sınıfındaki maddeler iyi uygulama teknikleri olarak kabul edilir ve standart uyumluluğu bakımından daha az resmidir.

İster C dilinde yazılmış olsun ister otomatik kod oluşturma yöntemleri ile elde edilsin MISRA C standardına uygun kod geliştirmek;

- Derleyici farklılıklarından oluşabilecek hatalardan kaçınmayı
- Hata yapmaya meyilli fonksiyonları ve yapıları kullanmaktan kaçınmayı
- Sürdürülebilir ve debug edilebilir kod oluşturmayı
- Karmaşıklığı sınırlandırmayı sağlamaktadır.

2.5 Referans araştırmalar

Bu bölümde, literatür araştırması sırasında tez konusu ile ilgili dikkat çekici olan araştırmalar konu başlıklarına göre kısaca açıklanacaktır. Bu çalışmalar tezin yazılması için elde edilen bilgi birikimine referans oluşturmaktadır.

Kirovskii ve Gorelov (2019) tarafından yapılan sürücü yardımcı sistemlerinin güvenliği hakkında yaptıkları çalışmada, elektronik sistemlerin güvenliği konusunda araştırmacılar için kafa karışıklıklarına neden olabilecek kavramların arasındaki farkları açıklamaktadır. Fonksiyonel güvenlik (ISO 26262 – Functional Safety), siber güvenlik (SAE J3061 - Cybersecurity) ve amaçlanan işlevselliğin güvenliği (ISO PAS 21448 – Safety of the Intended Functionality) birbiri ile ilişkili ancak farklı standartlardır. Fonksiyonel güvenlik sistemin iç arızalarına, siber güvenlik sistemi kötüye kullanmak için bilinçli yapılan dış etkenlere, amaçlanan işlevsel güvenlik ise çevresel ve kasıtlı olmayan yapılan dış müdahalelere karşı alınan önlemlerle sağlanmaktadır.

Knopf (2019) tarafından yapılan hibrit-elektrikli araçların otonom sürüş fonksiyonları için sistem güvenlik analizlerini konu alan çalışmasında, Tehlike Analizi ve Risk Değerlendirmesinde (HARA – Hazard Analysis and Risk Assesment) kullanılan teknikleri açıklayarak ve uygulama örnekleri yaparak kavranmasını sağlamaktadır. Ön Tehlike Tahlili (PHA – Preliminary Hazard Analysis), Tasarım Hata Türleri ve Etkileri Analizi (DFMEA – Design Failure Mode and Effects Analysis), Sistem Elemanı Arıza Analizi (SEFA – System Element Fault Analysis), Sistem Teorik Süreç Analizi (STPA – System Theoretic Process Analysis), Tehlike ve İşletilebilirlik (HazOp – Hazard and Operability) gibi yöntemler kullanılarak otomotiv güvenliği bütünlük seviyelerinin (ASIL – Automotive Safety Integrity Levels) belirlenmesinde kullanılmaktadır. Gnaniah (2019) çalışmasında Knopf'tan farklı olarak Hata Ağacı Analizi (FTA – Failure Tree Analysis) tekniği ve iterarif HARA süreçlerine değinmektedir.

Tikar ve Ansari (2021) tarafından yapılan çalışmada EPS sistemi için fonksiyonel güvenlik standardına uyumluğunu ele alarak bir konsept çalışma yaparak sürecin tamamını açıklamaktadır. EPS sistemi için yapılan çok daha kapsamlı fonksiyonel güvenlik değerlendirmeleri literatürde bulunmasına ve bu çalışmanın gereksinim setlerinin çok kısıtlı olmasına rağmen ISO 26262 Fonksiyonel Güvenlik Standardının sürecini açıklamaktadır.

Paulsen (2021) tarafından benzinli ve dizel motor elektronik kontrol üniteleri için standartlaştırılmış E-Gas denetleme konseptinin (E-Gas monitoring concept) medikal cihazlarda kullanımı ile ilgili yaptığı çalışmada otomotiv sektöründe kabul görmüş bir yapının farklı uygulamalarda da kullanılabileceğini önermektedir. Bu yaklaşımla E-Gas konseptini referans alan denetleme yapıları farklı otomotiv kontrol sistemlerinde de kullanılarak standartlaşma sağlanabilecektir. Nagabhushan ve Nadibail (2019) tarafından yapılan çalışmada otonom ağır vasıta araçların hareket kontrolü için denetleme konseptlerinin tasarımına değinilmiştir. Bu çalışmada da E-Gas denetleme konseptinde açıklanan 3 katmanlı denetleme mimarisi temel alınmaktadır.

Selic (2012) uygulamada model tabanlı yöntemlerin benimsenmesine ilişkin farklı perspektiflerden görüşlerini açıkladığı çalışmasında model tabanlı yaklaşımların yazılım geliştiricilerin üretkenliğini ve kalitesini arttırdığından bahsetmektedir. Ancak model tabanlı geliştirme tekniklerinin potansiyeline hala ulaşmadığını bununla birlikte geleneksel yöntemlerle uzmanların yetkinliğine dayanan geliştirme tekniklerinin günümüz ihtiyaçlarını karşılamaktan uzak olduğunu dile getirmektedir.

Broy, Kirstan, Krcmar, Schätz ve Zimmermann (2011) model tabanlı tasarımın otomotiv endüstrinde gömülü sistem yazılımı gelişme sürecine faydalarını değerlendirdikleri çalışmalarında model tabanlı geliştirmenin önemli maliyet tasarrufu sağlayabileceğini ancak bunun ancak amacına yönelik iyi seçilmiş bir araç ve rol paylaşımları ile mümkün olacağını belirtmektedir.

Conrad (2012) çalışmasında ISO 26262 fonksiyonel güvenlik standardının amaç ve gereksinimlerinin model tabanlı geliştirilen yazılımlar için de geçerli olduğunu ve oluşturulacak standart bir süreç ve taslakların hazırlanmasının gerekliliklerinden bahsetmektedir. Daha önce de bahsedildiği gibi amaca uygun model tabanlı yazılım aracı seçmenin önemi bir kez daha görülmüş oldu. Seçilen araç sürecin dokümantasyonunu ve otomasyonunu destekleyici özelliklere sahip olması verimliliği arttıracaktır.

Holtmann, Meyer ve Meyer (2011) çalışmasında sistematik bir model tabanlı geliştirme süreci hakkında yaklaşımlarını paylaşmaktadır. Otomotiv sektöründe kullanılan Automotive SPICE (Automotive Software Process Improvement and Capability dEtermination) standardı örnek bir geliştirme süreci tanımlar. Ancak bu süreçte kullanılacak dil hakkında bir bilgi vermez sadece sürece odaklanır. Çalışmayı gerçekleştiren araştırmacılar önerilen yöntemin model tabanlı geliştirme için referans bir Automotive SPICE modeli olarak tanımlamaktadır.

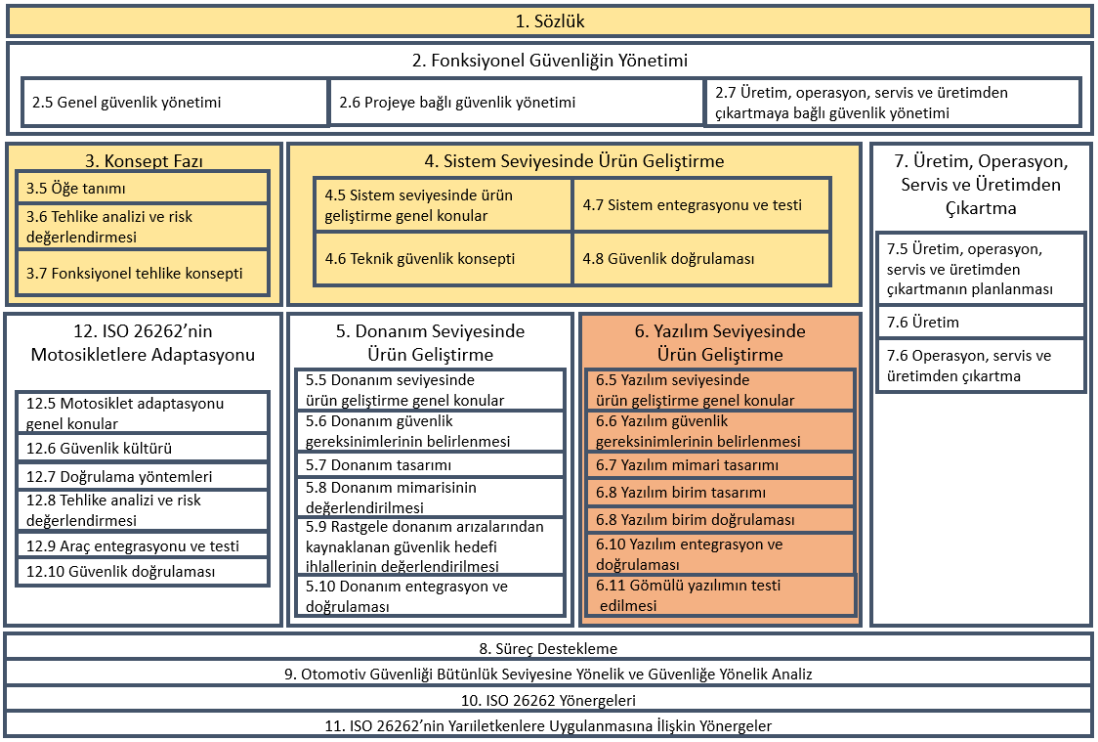
Liliegard ve Nilsson (2014) çalışmasında model tabanlı test sürecini değerlendirmektedir. Model tabanlı geliştirme yaklaşımının faydalarını ve eksikliklerini açıklayarak yazılım test sürecine hangi durumlarda katkı sağlayacağını açıklamaktadır. Araştırmacılar gereksinimleri kategorilere ayırarak incelemelerini detaylandırmıştır.

Hiremath ve Isha (2019) çalışmasında Sabit Mıknatıslı Senkron Motorlu (PMSM – Permanent Magnet Synchronous Motor) EPS sisteminin modellenme ve simülasyonunu gerçekleştirmektedir.

Bu tez kapsamında EPS sisteminin doğrula işlemi simülasyon ortamında yapılması hedeflenmektedir. Bu nedenle Hiremath ve Isha'nın modeli çalışma kapsamında modellenecek fiziksel sistem için bir referans oluşturacaktır.

3. KONSEPT

ISO 26262 ürün yaşam döngüsünün tamamını içeren birçok disiplini ilgilendiren geniş kapsamlı bir standart olmasından dolayı standardın tamamı bu tez kapsamında incelenmeyecektir. Bu tez kapsamında emniyet kritik model tabanlı yazılım geliştirme süreci incelenecek ve model tabanlı yazılım geliştirme sürecini etkileyen diğer süreçler hakkında genel bilgilendirmeler yapılacaktır. Süreci iyileştirmek için sürecin doğru bir şekilde anlaşılması büyük önem taşımaktadır. Bu başlık altında tezin uygulama bölümünde kullanılacak bilgiler hakkında bir temel oluşturmaktır. Şekil 3.1'de tez kapsamında incelenecek bölümler renklendirildi. Sarı renkler bilgi vermek amacıyla değinilen bölümleri gösterirken, kırmızı renkli bölüm detaylı incelenecek olup uygulama bölümünde verilen örneklerde yazılım seviyesinde olacaktır.



Şekil 3.1 : ISO 26262 standardı içeriğinde incelenecek kısımlar [8].

3.1 Elektrik Takviyeli Direksiyon Sistemi (EPS: Electric Power Steering)

Direksiyon sistemi, kullanılan aracın sürücü tarafından belirlenen yörüngede hareket edebilmesini sağlayarak geçmişten günümüze otomotiv sektörünün temel yapı taşlarından birisi olmuştur. İlk araçların kullanılmasıyla hayatımıza giren direksiyon sistemi yıllar içerisinde önemli gelişmeler geçirmiştir. Takviyeli direksiyon

sistemlerinin gelişmesi ile sürücüye hem ergonomik bir kullanıma sahip olmuş ve aracın kontrol edilebilirliği iyileştirilmiştir. Takviyeli direksiyon sistemleri kronolojik olarak Hydraulic Power Assisted Steering (HPAS), Electro-hydraulic Power Steering (EHPS), Electric Power Steering (EPS) yıllar içerisinde sektördeki yerlerini almışlardır. Tezin uygulama kısmında EPS sistemi ele alınmıştır.

EPS sistemler elektronik olarak programlanabilir olmasıyla şasi ve süspansiyon sistemleri geliştiren mühendislere daha geniş bir tasarım alanı oluşturmasının yanında sürekli çalışan bir pompaya ihtiyaç duymaması nedeniyle yakıt ekonomisi ve emisyon iyileştirmelerini de beraberinde getirmiştir. Seri üretim bir araçta ilk defa 1988 yılında Suziki Cervo[11] modelinde uygulanmıştır. Ancak ilk yıllarda yavaş hızlarda tehlikeden kaçınma hareketleri sırasında elektrik motorunun doğal olmayan davranışı nedeniyle diğer otomotiv üreticileri tarafından benimsenmesi zaman aldı. Özellikle son yıllarda EPS sistemi otonom sürüşün temel teknolojilerinden birisi olarak hayatımızda yerini almıştır.

EPS sistemi arızası durumunda takviye gücü ortadan kalkmakta ve sürücünün yoğun eforu gerekmektedir. Buna rağmen mekanik bağlantı nedeniyle kontrol edilebilirlik tamamen kaybolmamaktadır. Sürüş sırasında beklenmedik şekilde takviye gücünün ortadan kalkması yüksek hızlarda ölümcül kazalara neden olabileceğinden dolayı emniyet kritik parçalardan sayılmaktadır. Tüm bu bilgiler ışığında EPS sisteminin tezin konseptine uygun olduğu görülmektedir.

3.2 Sözlük

Bu bölümün amacı, tez içerisinde kullanılan teknik terim ve tanımların okuyucu için açıklık kazandırmasıdır. Bu bölüm içerisinde ISO 26262-1 Vocabulary bölümünde açıklanan terimlerden alıntılar ve açıklamalar bulunmaktadır.

Fonksiyonel güvenlik: E/E sistemlerin arızalı davranışından kaynaklanan tehlikeler nedeniyle oluşan riskin makul seviyede tutulması durumudur.

Tehlike analizi ve risk değerlendirmesi: Tehlike olaylarını tanımlama ve kategorize etmek ve makul olmayan risklerden kaçınmak için ilgili güvenlik hedeflerini ve ASIL'leri belirleme yöntemidir.

ASIL: Makul olmayan bir riskten kaçınmak için uygulanacak öğenin veya unsurun gerekli ISO 26262 gereksinimlerini ve güvenlik önlemlerini belirten dört seviyeden biridir. D en katı ve A en az katı seviyeyi temsil eder.

Yazılım mimarisi: Yazılımı oluşturan yapı taşlarının, sınırlarını ve arayüzlerini tanımlamaya olanak vererek, bu yapı taşlarına ilgili gereksinim atamalarının yapılmasına olanak veren yapının temsilidir.

Hata toleransı zaman aralığı: Bir komponentte arıza meydana gelmesinden sonra olası tehlikeli olayın meydana gelmesine kadar olan minimum zaman aralığıdır.

Fazlalıklı kullanım: Gerekli bir işlevi yerine getirmek veya bilgiyi temsil etymek için yeterli olacak araçların çoklu kullanımıdır.

3.3 Konsept Fazı

ISO 26262 Fonksiyonel güvenlik standardı 3. bölümü konsept aşamasını içermektedir. Bu bölümde fonksiyonel güvenlik standardı kapsamında geliştirilecek ürünün tanımı yapılarak hangi amaca hizmet edeceği, hangi elektronik komponentlerle etkileşime gireceği vb. konular ele alınarak sınır şartları belirlenmektedir.

Karayolları aracında kullanılacak komponentin güvenli olma durumu konsept aşamasında belirlenen sınır şartları çerçevesinde belirlenmektedir. Standardın özünde, geliştirilen yazılım kendi tanımlaması içerisinde hatasız olmalıdır. Güvenli yazılım geliştirmek için, güvenlik kültürü ve geliştirilen ürün hakkında uzmanlık kilit unsurdur.

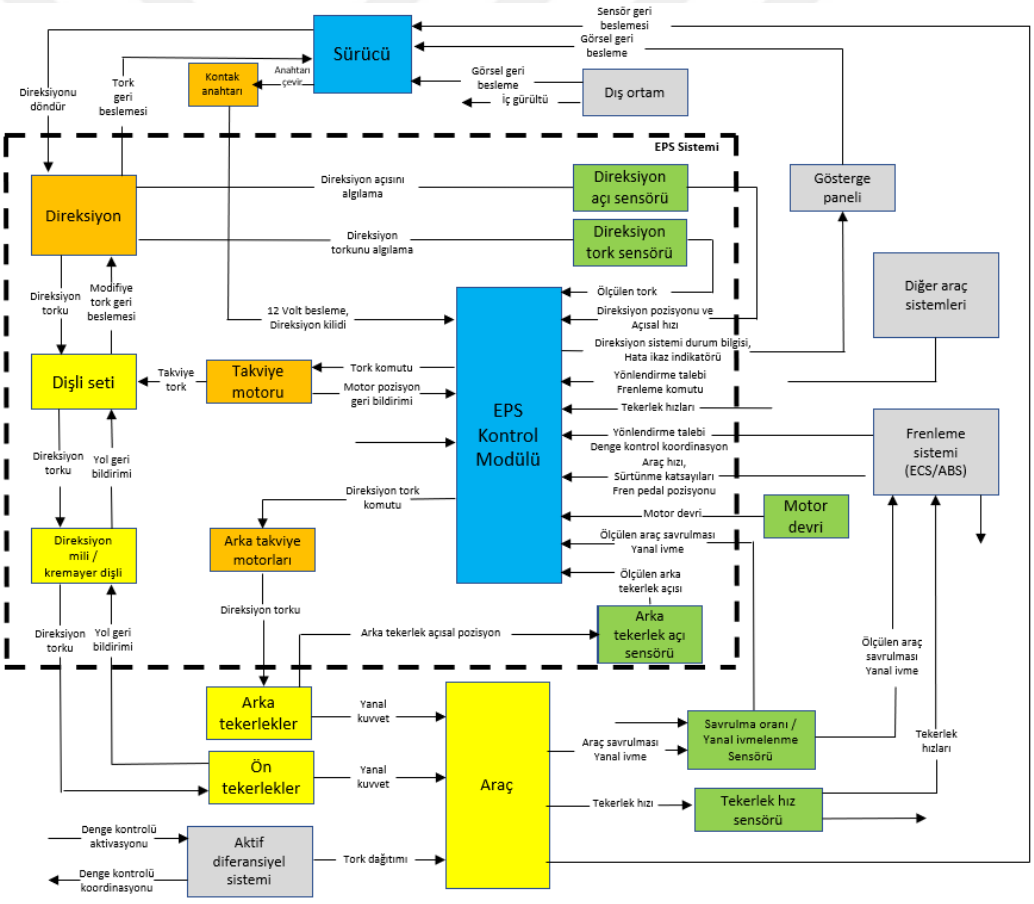
Konsept aşamasının çıktıları;

- Öğe tanımı
- Tehlike analizi ve risk değerlendirmesi
- Fonksiyonel güvenlik konseptidir.

3.3.1 Öğe tanımı

Öğe tanımı bölümünde geliştirilen komponentin fonksiyonalitesini, sürücü, çevre şartları, diğer elektronik komponentler ile bağımlılık ve etkileşimleri açıklanmalıdır. Referans alınan komponent diyagramını Şekil 3.2’de paylaşıldı. Böylelikle gelecek aktivitelerde tutarlı ve kapsamlı bir değerlendirme yapmak mümkün olacaktır.

Öğe tanımı bölümünde ilk olarak bakılması gereken kaynaklar yasal gereksinimler, ulusal ve uluslararası standardlardır. Bu araştırma sonunda temel gereksinimler belirlenir. Yasal gereksinimler belirlendikten sonra geliştirilen teknolojinin araç seviyesinde sağlayacağı fonksiyonlar ve operasyonel modlar belirlenmelidir. Fonksiyonlar için performans, kalite ve fonksiyonalitenin sağlanacağı şartlar(çevresel şartlar vb.) belirlenerek gereksinimler oluşturulmalıdır. Ayrıca bu fonksiyonların araç üzerinde bulunan diğer sistemler üzerine etkileri detaylıca incelenmeli ve oluşabilecek potansiyel hataların hangi sistemleri etkileyeceği açıkça belirtilmelidir. Buna ek olarak etkileşim içerinde bulunduğu sistemler ve geliştirme aşamasında ele alınan sistemin sınır şartları açık olmalıdır. Bir başka önemli konu ise algılayıcı ve eyleyicilerin kapasiteleri sistemin fonksiyonlarını yerine getirebilecek şartlarda belirlenmesi için gereksinimler oluşturulmalıdır.



Şekil 3.2 : Komponent diyagramı üzerinde sistem sınırlarının gösterimi [12].

3.3.2 Tehlike analizi ve risk değerlendirmesi (HARA)

Tehlike analizi ve risk değerlendirmesi, elektronik sistemin tamamının veya bir kısmında meydana gelen arıza sebebiyle güvenlik hedeflerinin sağlanmasına neden

olacak tehlikelerin belirlenmesi ve sınıflandırılmasıdır. Belirlenen tehlikelerin sınıflandırılması, güvenlik hedeflerinin ihlali durumunda meydana gelecek kayıpların büyüklüğüne göre belirlenmektedir. Değerlendirmenin rasyonel ve tutarlı bir şekilde yapılabilmesi için ISO 26262 Kara Araçları Fonksiyonel Güvenlik Standard'ında ASIL (Automotive Safety Integrity Level) sevileri açıkça belirlenmiştir. ASIL; şiddet (Severity), maruz kalma (Exposure), kontrol edilebilirlik (Controllability) parametrelerine bağlı olarak hesaplanmaktadır.

İlk olarak kazaya sebebiyet verebilecek tehlikelerin açıkça belirlenmesi gerekmektedir. Electric Power Steering sisteminde meydana gelebilecek tehlikeli durumlara örnek olarak, istemsizce aracın yanal hareketi veya yetersiz araç yanal hareketi verilebilir. Bu tehlike durumlarını rasyonel bir şekilde değerlendirmek için aracın operasyonel durumunun çerçevesi çizilmesi gerekmektedir. Oluşan tehlike, aracın sürüş şartlarına bağlı olarak farklı sonuçların meydana gelmesine sebebiyet verebilmektedir. EPS sistemindeki bir hata nedeniyle fonksiyonalitesinin devre dışı kalması sonucu oluşan tehlikenin büyüklüğü yol şartlarına veya manevra durumuna göre değişiklik gösterebilmektedir. EPS sisteminin iyi yol koşullarında, manevra halinde değilken devre dışı kalması ile, karlı bir havada yanal bir manevra yaparken devre dışı kalması arasında farklı sonuçlar meydana gelecektir. Bu nedenle risk değerlendirmesi yapılırken operasyonel durum, seyir hızı, trafik yoğunluğu, görüş mesafesi, yaya yoğunluğu, kara yolu tipi, yol koşulları ve sürüş manevrası gibi değişkenlere bağlı olarak incelenmelidir. Her sürüş senaryosu kendi içerisinde değerlendirilmelidir. Bu çalışmada değerlendirilen operasyonel durumlar ve çevresel şartlar Çizelge 3.1'de paylaşıldı.

Çizelge 3.1 : Operasyonel durumlar ve çevresel şartlar listesi [12].

Operasyonel Durumlar ve Çevresel Şartlar	
Araç Hızı	Çok yüksek hız (130km/s < V)
	Yüksek hız (100km/s < V < 130km/s)
	Orta hız (40km/s < V < 100km/s)
	Düşük hız (V < 40km/s)
	Durağan (V = 0km/s)
Trafik	Yoğun
	Seyrek
Görüş Mesafesi	Düşük
	Yüksek
Yaya Mevcudiyeti	Yok sayılabilir
	Seyrek
	Yoğun
Yol Tipi	Park alanı
	Şehiriçi
	Şehirlerarası
Yol Durumu	Kaygan
	İyi
Sürüş Manevrası	Düz yolda ilerleme
	Harekete başlangıç
	Geri yönde ilerleme
	Trafikte duruş kalkış
	Rampaya giriş ve çıkış

Tehlikeler ve operasyonel şartlar belirlendikten sonra risk değerlendirmesi yapabilmek mümkündür. Bu değerlendirmenin yapılması sırasında fonksiyonel güvenlik mühendislerine rasyonel ve tutarlı bir değerlendirme yapabilmesi için çerçeve çizilmiştir. Değerlendirme yapan mühendisler kazanın şiddeti, gerçekleşme olasılığı ve tehlike anında kontrol edilebilirliğini birer parametre olarak ele almaktadır. Bu parametrelerin seviyeleri Çizelge 3.2, Çizelge 3.3, Çizelge 3.4'te verilmektedir.

Şiddet (Severity): Meydana gelen kaza sonucu oluşabilecek kayıpların büyüklüğü ele alınarak yapılan bir değerlendirmedir. Kaza sonucu oluşacak kayıpların büyüklüğüne göre S0 S1 S2 S3 seviyeleri arasından bir seçim yapılmaktadır.

Çizelge 3.2 : Şiddet seviyeleri [8].

	S0	S1	S2	S3
Açıklama	Yaralanmasız	Hafif yaralanma	Ağır yaralanma	Ölümcül yaralanma

Bu seviyeler arasından uygun olanı seçebilmek için AIS (Abbreviated Injury Scale), ISS (Injury Severity Score) gibi değerlendirme yöntemleri kullanılmaktadır.

Maruz Kalma (Exposure): ISO 26262 Kara Araçları Fonksiyonel Güvenlik Standard'ına göre maruz kalma, analiz edilen arıza moduyla tehlikeli olabilecek operasyonel durumun çakışması durumudur. Maruz kalma olasılığı RxT formülü ile hesaplanabilir. Burada ürünün yaşam döngüsü boyunca operasyonel durumun oluşma oranıdır. T ise, arızanın algılanmadığı süredir. Bu çalışmada hedef pazardan alınacak istatiksel veriler kullanılabilmektedir.

Çizelge 3.3 : Maruz kalma seviyeleri [8].

	E0	E1	E2	E3	E4
Açıklama	İhmal edilebilir	Çok düşük ihtimal	Düşük ihtimal	Orta ihtimal	Yüksek ihtimal

Kontrol Edilebilirlik: ISO 26262 Kara Araçları Fonksiyonel Güvenlik Standard'ına göre kontrol edilebilirliği, muhtemelen dış önlemlerin desteğiyle, ilgili kişilerin zamanında tepkileri yoluyla belirli bir zarar veya hasarı önleme yeteneğidir. Kontrol edilebilirlik, sürücünün aracın kontrolünü elinde tutma, yeniden kazanma olasılığının veya çevredekilerin tehlikeden kaçınmasına katkıda bulunabilme olasılığıdır. Bu değerlendirme için örnek bir tablo ISO 26262-3 ek B B.6 tablosunda bulunmaktadır.

Çizelge 3.4 : Kontrol edilebilirlik seviyeleri [8].

	C0	C1	C2	C3
Açıklama	Genel olarak kontrol edilebilir	Basitçe kontrol edilebilir	Normalde kontrol edilebilir	Zor kontrol edilebilir veya kontrol edilemez

Çizelge 3.5'e göre, yukarıda bahsi geçen risk değerlendirme sürecinde kullanılan parametrelere bağlı olarak her bir tehlikeli olay için ASIL seviyesi belirlenmelidir.

Çizelge 3.5 : ASIL belirleme çizelgesi [8].

Önem Sınıfı	Maruz Kalma Sınıfı	Kontrol Edilebilirlik Sınıfı		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Standart tarafından tanımlanan ASIL A, ASIL B, ASIL C ve ASIL D olmak üzere 4 adet ASIL seviyesi bulunmaktadır. ASIL A en düşük ASIL D en yüksek güvenlik bütünlüğü seviyesidir. QM (Quality Management) olarak tanımlanan tehlikeler bir güvenlik gereksinimi gerektirmez.

Çizelge 3.6'da belirlenen operasyonel durum için ASIL seviyelerinin nasıl belirlendiğine dair bir örnek paylaşılmıştır. Bu örnek NHTSA (National Highway Traffic Safety Administration) tarafından 2018 yılında yayınlanan jenerik bir EPS'nin fonksiyonel güvenlik değerlendirmesinden alınmıştır. Aynı çalışma için EPS sisteminin farklı tehlikeler için ASIL seviyeleri Çizelge 3.7'de paylaşıldı.

Çizelge 3.6 : ASIL belirleme örneği [12].

Araç seviyesinde tehlike	Potansiyel yetersiz araç yanal hareketi		
Operasyonel durum	Yoğun trafiğe sahip, iyi görüş mesafesinde ve iyi yol koşullarında, kırsal yollarda orta hızda sürüş durumu		
Potansiyel kaza senaryosu	Yoldaki başka bir araca veya bir nesneye çarpılması		
ASIL değerlendirilmesi	Önem	S3	Başka bir araç ile çarpışma sonucu önden veya yanal darbe sonucu yolcu bölmesinde deformasyon oluşması.
	Maruz kalma	E4	Belirtilen sürüş senaryosu sıklıkla meydana gelmektedir. (Aracın yaşam döngüsünün %10'undan fazlasıdır.)
	Kontrol edilebilirlik	C2	Bu durum aracın düşük hızı ve bir dereceye kadar direksiyonun kullanılabilirliği nedeniyle normalde kontrol edilebilirdir.
Atanan ASIL seviyesi	C		

Çizelge 3.7 : NHTSA'ya göre belirlenen tehlikeler ve ASIL atamaları [12].

	Tehlike	ASIL
H1	Potansiyel istenmeyen araç yanal hareketi	D
H2	Potansiyel yetersiz araç yanal hareketi	C
H3	Potansiyel direksiyon desteği kaybı	B
H4	Artan arka tekerlek sürtünmesi nedeniyle sürücünün komutlarına potansiyel olarak azaltılmış tepki verme	A

3.3.3 Fonksiyonel güvenlik konsepti

Fonksiyonel güvenlik konseptinin amacı güvenlik hedeflerine ulaşabilmek adına fonksiyonel güvenlik gereksinimlerinin oluşturulmasıdır. Güvenlik gereksinimleri oluşturulurken;

- Güvenlik hedeflerine uygun olarak fonksiyonel ve kısıtlanmış fonksiyonların davranışlarının belirlenmesi
- Güvenlik hedeflerine uygun olarak ilgili hataların zamanında tespiti ve kontrolü ile ilgili kısıtlamaların belirlenmesi
- Komponent, sürücü veya harici önlemler tarafından oluşan hataların hedeflenen toleranslar içerisinde tespiti ve önlenme stratejilerinin belirlenmesi
- Fonksiyonel güvenlik gereksinimlerinin mimari tasarıma veya harici önlemlere tahsis etmek
- Fonksiyonel güvenlik konseptini doğrulamak ve doğrulama kriterlerini belirlemektir

NHTSA tarafından yapılan çalışmadan jenerik bir EPS'nin güvenlik hedefleri Çizelge 3.8'deki gibidir.

Çizelge 3.8 : Güvenlik hedefleri [12].

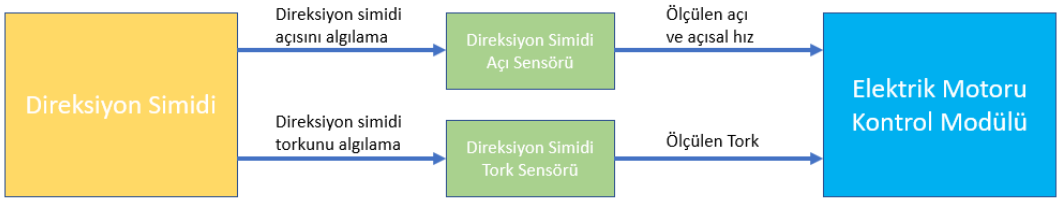
	Güvenlik Hedefleri	ASIL
SG 1	EPS sistemi, tüm araç koşullarında herhangi bir yönde istenmeden kendi kendine dümenlemeyi önlemelidir.	D
SG 2	EPS sistemi, tüm araç çalışma koşullarında doğru seviyede direksiyon desteği sağlamalıdır.	C
SG 3	EPS sistemi, tüm araç çalışma koşullarında istenmeyen direksiyon desteği kaybını önlemelidir.	B
SG 4	EPS sistemi, tüm araç çalışma koşullarında arka tekerlek sürüklenmesini önlemelidir.	A

Fonksiyonel güvenlik gereksinimleri belirlenirken aşağıdaki stratejiler değerlendirilmelidir.

- Hatadan kaçınma
- Arıza tespiti, arıza kontrolü ve oluşan hataların sonuçları
- Güvenli duruma geçiş stratejileri
- Hata toleransları
- Arıza durumunda işlevselliğin bozulması ve bunun sürücü ikazları ile ilişkisi
- Riske maruz kalma süresini kabul edilebilir bir süreye indirmek için gereken sürücü ikazlarının tepki sürelerinin belirlenmesi
- Kontrol edilebilirliği arttırmak için gerekli sürücü uyarılarının belirlenmesi
- Kontrol edilebilirliği arttırmak için ihtiyaç duyulan sürücü ikazlarının zamanlama gereksinimlerinin nasıl karşılanacağı ve hataya dayanıklı zaman aralığının nasıl karşılanacağının belirlenmesi
- Birden fazla kontrol talebinin aynı anda uygun olmayan şekilde uygulanması durumunda tehlikenin önlenmesi ve azaltılması

ISO 26262 fonksiyonel güvenlik kapsamında makul olmayan bir risk ile karşılaşıldığında güvenli bir durum tanımlaması yapılması gerekmektedir. Güvenli durum, tüm fonksiyonalitenin aktif olduğu, fonksiyonalitenin kısıtlandığı güvenli çalıştırma modu veya fonksiyonalitenin tamamen devre dışı kalması olabilir. Bu güvenli durumlar arasındaki geçiş stratejileri teknik güvenlik konseptinde daha detaylı incelenecektir.

EPS, direksiyon sistemi sensörleri, servo motor, haberleşme sistemi, arayüz sistemi gibi birçok donanımsal sistemin bir arada çalışması ile işlevini yerine getirmektedir. Fonksiyonel güvenlik kapsamında rastgele donanım ve sistematik hatalardan kaçınmak için yazılacak gereksinimler her bir sistem özelinde ele alınabilir. Tez kapsamında bütün sistemlerin ele alınması yerine, yöntem üzerine odaklanabilmek adına direksiyon sistemi sensör sinyalleri incelenecektir. Bu nedenle direksiyon sistemi sensörleri Şekil 3.3’te paylaşdı.



Şekil 3.3 : Direksiyon sistemi sensörleri [12].

Araç seviyesinde fonksiyonel güvenlik sistemlerinin ortak kullandığı sensörler veya bu sensörden alınan verilerin başka bir elektronik kontrol ünitesi ile paylaşılması durumunda değerlendirmeler sistemin risk değerlendirmesi birlikte değerlendirilmelidir. Örnek olarak, fren pedal sensörünün hem ABS (Anti-Lock Braking System) hem de içten yanmalı motor elektronik kontrol ünitesi cruise control fonksiyonunda kullanılması buna örnek verilebilir. Bu çalışmada kullanılan sensörler sadece EPS sistemi kapsamında değerlendirilmiştir. Değerlendirme sonucunda NHTSA'nın belirlediği minimum gereksinimler Çizelge 3.9'da paylaşıldı.

Çizelge 3.9 : Direksiyon sensörleri fonksiyonel güvenlik gereksinimleri [12].

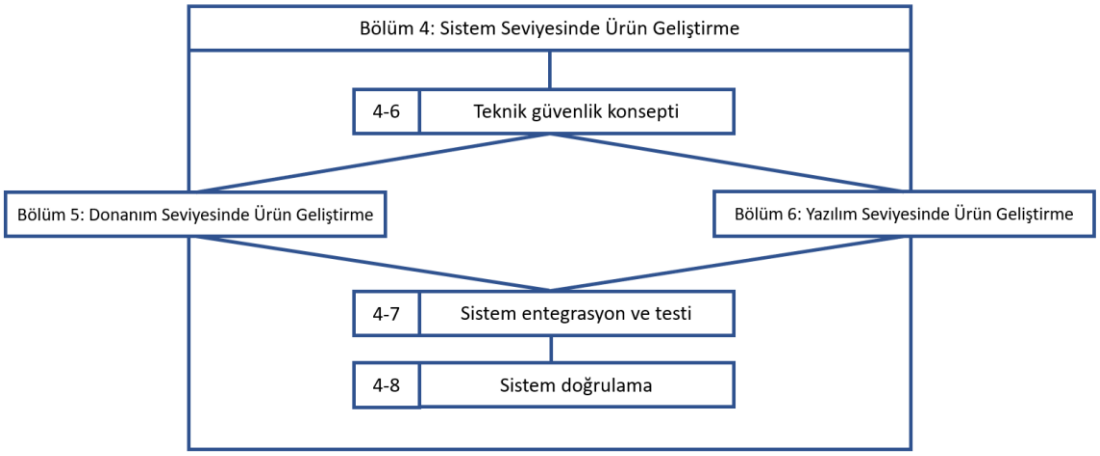
Gereksinim Numarası	Güvenlik Hedefi	ASIL Seviyesi	Güvenlik Gereksinimleri
2.1	1,2,3,4	No ASIL	Sürücünün girdisinden kaynaklanan direksiyon simidi torku, aracın kullanılabilir ömrü boyunca tüm çalışma koşullarında burulma çubuğunda tutarlı bir burulma oluşturmalıdır. (Bu gereksinim ISO 26262 kapsamında değildir ancak standarda uyumluluk incelemesinin bir parçası olacaktır.)
2.2	1,2,3	B, C, D	Direksiyon simidi tork sensörü, direksiyon milindeki burulmayı ölçmek içindir. Sensör ölçümünün doğruluğu ve geçerliliği sağlayacak yetkinliğe sahip olmalıdır.
2.3	1,2,3	B, C, D	Tork sensörünün burulma çubuğunun bükülmesini bir elektrik sinyaline dönüştürme yöntemi doğrulanmalıdır.
2.4	1,2,3,4	A, B, C, D	EPS sistemi açıkken tork sensörü giriş voltajı, yüksek ve düşük voltaj limitleri kontrol edilmelidir. Giriş voltajının aralık dışında olması durumunda EPS sistemi DSB ms içerisinde Güvenli Durum 3' geçmelidir.
2.5	1,2,3,4	A, B, C, D	Tork sensörü tarafından yapılan tork ölçümü, EPS kontrol modülüne iletilmeli ve geçerlilik, doğruluk ve rasyonellik açısından kalifiye olmalıdır.
2.6	1,2,3,4	A, B, C, D	Tork sensörü verileri sürekli olarak izlenecek ve EPS sistemi, sürücüye yalnızca tork sensörü verileri EPS kontrol modülü tarafından doğru şekilde okunduğunda güç desteği sağlanmalıdır.
2.8	1,2,3,4	A, B, C, D	Tork sensörü çıkış sinyali, izleme devresindeki bir tork sensörü veri analiz ünitesi tarafından analiz edilmelidir.
2.9	1,2,3,4	A, B, C, D	Tork sensörünün sağlığı ve verilerin makullüğü, çalışan tüm araç koşullarında izlenmeli ve onaylanmalıdır.
2.10	1,2,3,4	A, B, C, D	Güvenlik hedefini ihlal eden bir arıza durumunda, tork sensörü arızayı EPS kontrol modülüne iletmelidir.
2.11	1,2,3	QM, A, B	Tork sensörü, elektromanyetik uyumluluk EMC/EMI, ESD, kontaminasyon, tek olay etkileri ve diğer çevresel koşullardan kaynaklanan güvenlik ile ilgili arızalar için tanımlamaya sahip olmalıdır.

Gereksinim Numarası	Güvenlik Hedefi	ASIL Seviyesi	Güvenlik Gereksinimleri
2.12	1,2,3,4	No ASIL	Bir güvenlik hedefinin ihlaline yol açan tüm tek noktalı tork sensörü donanım arızaları, arıza tespit aralığı içerisinde tespit edilmeli ve FTTI içerinde azaltılmalıdır. Bir arıza durumunda sistem ilgili güvenlik durumuna geçecektir.
2.13	2	A	Direksiyon simidi açı sensörü, direksiyon simidi girişinden kaynaklanan direksiyon simidi açısını ölçmek içindir ve ölçüm metodu kalifiye olmalıdır.
2.14	2	A	Direksiyon açısının elektriksel dönüşüm yöntemine göre doğrulanmalıdır.
2.15	2	A	Direksiyon simidi açı sensörü tarafından yapılan açı ölçümü, EPS kontrol modülüne iletilmelidir.
2.16	2	A	Güvenlik Hedefi 2'yi ihlal edebilecek gizli hataları azaltmak için açı sensörü verileri düzenli aralıklarla kontrol edilmelidir.
2.17	2	A	Güvenlik Hedefi 2'yi ihlal eden bir arıza durumunda, direksiyon simidi açı sensörü, arızayı EPS kontrol modülüne iletmelidir.
2.18	2	A	Açı sensörü, EMC/EMI, ESD, kontaminasyon, tek olay etkileri ve diğer çevresel koşullardan kaynaklanan güvenlikle ilgili arızaları teşhis etmelidir.
2.19	2	A	Güvenlik hedefi 2'nin ihlaline neden olabilecek tüm tek nokta direksiyon açı sensörü donanım arızaları, arıza tespit aralığı içerisinde tespit edilmeli ve FTTI içerisinde azaltılmalıdır. Arıza durumunda sistem sarı ışıklı sürücü ikaz uyarısı vermelidir.

3.4 Sistem Seviyesinde Ürün Geliştirme

Sistem seviyesinde uygulanması gereken aktiviteler Şekil 3.4'te verilmiştir. Teknik güvenlik konsepti, teknik güvenlik gereksinimleri ile sistem mimari tasarımından türetilmektedir. Teknik güvenlik gereksinimleri, sistem elementlerine veya spesifik teknolojilere atanmaktadır.

Teknik güvenlik konsepti geliştirildikten sonra yazılım ve donanım seviyesinde ürün geliştirme süreçleri gerçekleştirilir. Geliştirilen yazılım ve donanımın sistem entegrasyonu yapılarak güvenlik hedeflerine ulaşıp ulaşılmadığı test edilir.



Şekil 3.4 : Sistem seviyedinde ürün geliştirme süreci [8].

3.4.1 Teknik güvenlik konsepti

Teknik güvenlik konseptinin amaçları;

- Teknik güvenlik gereksinimlerinin uygulanabilmesi için gerekli olan sistem elemanlarının ve arayüzlerin işlevselliği, bağımlılıkları, kısıtlamaları, ve özelliklerin belirlenmesi
- Sistem elemanları ve arayüzlerinde uygulanacak güvenlik mekanizmalarının teknik güvenlik gerekliliklerini belirlemek.
- Üretme, işletme, servis, hizmetten çıkarma sırasında sistemin ve elemanların güvenliğine ilişkin gereklilikleri belirlemek.
- Teknik güvenlik gereksinimlerinin sistem düzeyinde işlevsel güvenliği sağlamaya uygun olduğunu ve işlevsel güvenlik gereksinimleriyle tutarlı olduğunu doğrulamak.

- Güvenlik gerekliliklerini karşılayan ve güvenlikle ilgili olmayan gerekliliklerle çelişmeyen bir sistem mimari tasarımı ve teknik güvenlik konsepti geliştirmek
- Üretim ve servis için hataları önlemek amacıyla sistem mimari tasarımını analiz etmek ve gerekli güvenlikle ilgili özel karakteristikleri belirlemek
- Sistem mimari tasarımının ve teknik güvenlik konseptinin ilgili ASIL'e göre güvenlik gereksinimlerini karşılamaya uygun olduğunu doğrulamaktır.

3.4.1.1 Güvenlik mekanizmaları

Teknik güvenlik mekanizmaları sistemin kendisinin veya etkileşime girdiği çevre elemanların hatalarını tespit ederek, fonksiyonel güvenlik gereksinimlerini ihlal edebilecek durumları önlemek ve etkisini azaltması beklenen yapılardır. Güvenlik mekanizmaları sistemi tanımlanan güvenli halde bulumasını sağlayacak veya hatalar oluştuğunda tehlikenin şiddetini ve arıza altındaki sistemin kontrol edilebilirliğini arttırmak için belirlenen sürede ilgili güvenli durumlar arasındaki geçişi sağlayarak gerekli kısıtlamaları uygulamak ve kullanıcıyı uyarmaktır.

Sadece rastgele donanımsal arızalar gizli hatalar (latent fault) çerçevesinde kabul edilebilir. Bunu önlemek adına açılış, kapanış sekanslarında ve periyodik olarak valf, lamba gibi elemanların düzgün çalıştığına dair testlerden geçmesi beklenir. Testlerin sıklığı, HARA'da yapılan değerlendirmelere göre arızanın sebebiyet vereceği potansiyel kazanın şiddeti, arıza oluştuğunda kontrol edilebilirliği ne kadar etkilediği ve arızanın oluşma olasılığı göz önünde bulundurularak belirlenir.

Bu çalışmada kullanılacak 2 güvenli durum yeterli görüldü.

Güvenli durum 1: Direksiyon yardımı düşük hızlarda kullanılabilir, ancak yüksek hızlarda devre dışı bırakılır.

Güvenli durum 2: Tüm direksiyon yardımı devre dışı.

3.4.1.2 Teknik güvenlik gereksinimlerinin atanması

Teknik güvenlik gereksinimleri, bir donanım (uygulama özel tümleşik devreler, kompleks algılayıcıların korumalı yapıları, proje özelinde geliştirilen devreler vb.) veya yazılım (uygulama ve temel seviyedeki yazılımlar, standartlaşmış protokoller vb.) ile sağlanabilir. Bu gereksinimlerin hangi teknik yapılar tarafından karşılanacağı

belirlenmeli ve ileride paralel yürütülecek yazılım ve donanım seviyesinde ürün geliştirme adımlarına atamaların yapılması beklenir.

3.4.1.3 Doğrulama

Teknik güvenlik gereksinimlerinin, doğruluğu, bütünlüğü ve tutarlılığı doğrulanmalıdır. Doğrulama sürecinde fonksiyonel güvenlik standardı tarafından önerilen yöntemler Çizelge 3.10 ve Çizelge 3.11'de paylaşılmıştır.

Çizelge 3.10 : Sistem seviyesinde doğrulama yöntemleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Denetim	+	++	++	++
1b	Gidiş yolu inceleme	++	+	0	0
2a	Simülasyon	+	+	++	++
2b	Sistem prototipleme ve araç üstü testleri	+	+	++	++
3	Sistem mimari tasarım analizi	Çizelge 3.11'e bakınız.			

Çizelge 3.11 : Sistem seviyesinde doğrulama yöntemleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Tümdengelim analizi	0	+	++	++
1b	Tümevarım analizi	++	++	++	++

Donanım ve yazılım seviyesindeki geliştirme süreçleri tamamlandıktan sonra geliştirilen teknolojilerin sistem seviyesinde doğrulanması gerekmektedir. Doğrulama işlemi belirlenen test senaryolarının bir seri test sürecine tabi tutulmasıyla gerçekleştirilir. Test senaryoları farklı perspektiflerden ele alınarak oluşturulabilir. Fonksiyonel güvenlik standardı tarafından önerilen yöntemlerin listesi Çizelge 3.12'de paylaşılmıştır.

Çizelge 3.12 : Entegrasyon testleri için test senaryosu oluşturma methodları [8].

Yöntem		ASIL A	ASIL B	ASIL C	ASIL D
1a	Gereksinim analizi	++	++	++	++
1b	İç ve dış arayüzlerin analizi	+	++	++	++
1c	Yazılım donanım entegrasyonu için denklik sınıflarının oluşturulması ve analizi	+	+	++	++
1d	Sınır değerlerinin analizi	+	+	++	++
1e	Bilgi ve tecrübeye dayalı hata tahmini	+	+	++	++
1f	Fonksiyon bağımlılık analizi	+	+	++	++
1g	Ortak limit koşullarının, sekansların ve bağımlı arıza kaynaklarının analizi	+	+	++	++
1h	Çevresel koşulların ve operasyonel kullanım durumlarının analizi	+	++	++	++
1i	Saha deneyimine bağlı analiz	+	++	++	++

Test senaryolarını oluşturmak için farklı yöntemler olduğu gibi yapılan testler de kendi arasında farklı hedef gruplarına göre ayrılmaktadır. Teknik güvenlik gereksinimlerinin doğru uygulanması, doğru performans, tutarlılık ve zamanlama hedeflerine ulaşılması, arayüzlerin tutarlı ve doğru şekilde çalışması, geliştirilen güvenlik mekanizmalarının etkinliği ve dayanıklılığı (donanım ve yazılım seviyesinde robustness) gibi konular farklı test kriterleri oluşturmaktadır. Çizelge 3.13'te gereksinim, Çizelge 3.14'de performans, tutarlılık ve zamanlama, Çizelge 3.15'te arayüz, Çizelge 3.16'da güvenlik mekanizmaları yetkinlik, Çizelge 3.17'de sağlamlık testleri olmak üzere donanım yazılım entegrasyonu seviyesinde fonksiyonel güvenlik standardı tarafından önerilen test yöntemleri paylaşılmıştır.

Çizelge 3.13 : Gereksinimlerin doğru uygulanma testleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Gereksinime dayalı test	++	++	++	++
1b	Hata enjeksiyon testi	+	++	++	++
1c	Bitişik test	+	+	++	++

Çizelge 3.14 : Performans, tutarlılık ve zamanlama test yöntemleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Bitişik test	+	+	++	++
1b	Performans testi	+	++	++	++

Çizelge 3.15 : Arayüz test yöntemleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Dış ara yüz testi	+	++	++	++
1b	İç ara yüz testi	+	++	++	++
1c	Ara yüz tutarlılık kontrolü	+	++	++	++

Çizelge 3.16 : Güvenlik mekanizmalarının yetkinlik tetsleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Hata enjeksiyon testi	+	+	++	++
1b	Hata tahminine dayalı test	+	+	++	++

Çizelge 3.17 : Sağlamlık test yöntemleri [8].

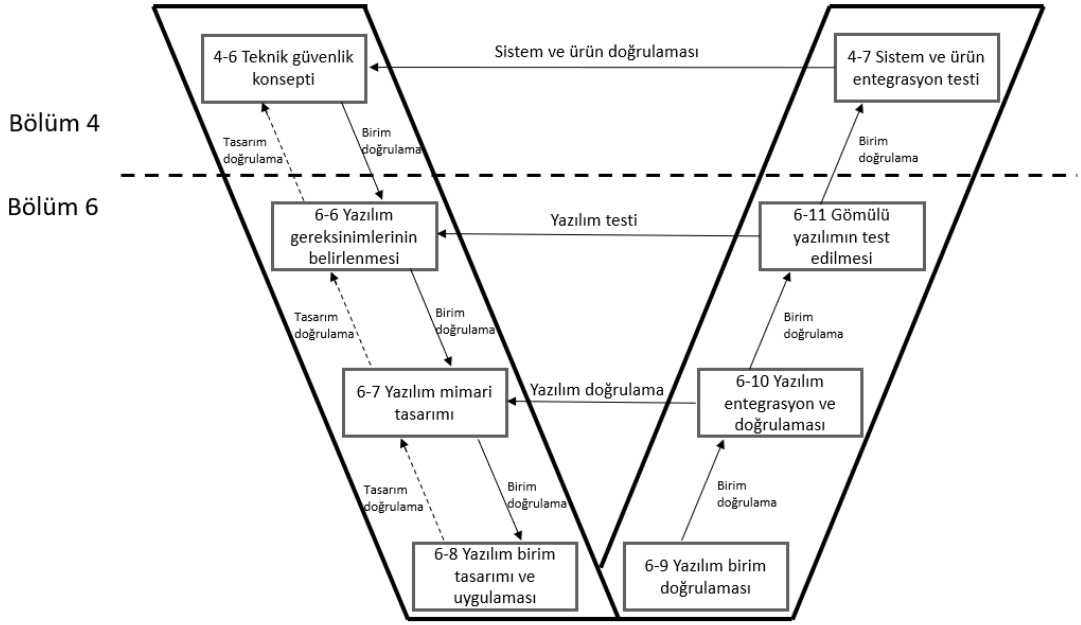
	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Kaynak kullanım testi	+	+	+	++
1b	Stres testi	+	+	+	++

Yukarıdaki yöntemlerin yazılım donanım entegrasyon testi sürecinde yapılacak testler için verildiğine dikkat edilmelidir. Bunlara ek olarak sistem entegrasyon ve araç entegrasyon seviyeleri için de standart içinde kapsamlı bilgi verilmiştir. Çalışmanın amacı fonksiyonel güvenlik standardı hakkında genel bir fikir vermek ve farkındalık yaratmak olduğundan, tekrara düşmemek adına sadece yazılım donanım entegrasyon seviyesindeki tablolar paylaşıldı.

Belirlenen test yöntemleri için doğrulama planı ve doğrulama yöntemlerinin açıklandığı entegrasyon ve test stratejisi oluşturulmalıdır. Test strateji dokümanında ele alınan testler raporlanmalı ve paylaşılmalıdır.

3.5 Yazılım Seviyesinde Ürün Geliştirme

Bu bölümün amacı yazılım geliştirme sürecinde uygun yöntem ve ortamın sağlanmasıdır. Yazılım geliştirme süreci sistem seviyesinden yazılım ekibine atanan teknik güvenlik gereksinimlerini yazılım seviyesinde değerlendirerek geliştirme sürecini başlatır. Geliştirdiği yazılımın sistem seviyesindeki entegrasyon testlerine destek vererek sürecini tamamlar. Geliştirme sürecinde donanım ekibi ile iletişim sürecin sağlıklı yürütülebilmesi için kilit unsurdur. Bu süreci desteklemesi için donanım-yazılım arayüz dokümanı bir gerekliliktir. Sürecin genel akışını gösteren diyagram Şekil 3.5'te paylaşılmıştır.



Şekil 3.5 : Yazılım seviyesinde V döngü süreci [8].

Geliştirme sürecinde kullanılacak yazılım dili, programlar amacına uygun seçilmeli, modülerlik, soyutlama ve encapsulation (veri ve işlemi bir arada içeren bağımsız modüller yaratma) desteklenmelidir. Proje genelinde yazılımın bütünlüğünün sağlanması, okunabilirliği arttırmak gibi amaçlar gözetilerek yönergeler hazırlanmalıdır. Şartname hazırlama kriterleri Çizelge 3.18'de paylaşıldı.

Çizelge 3.18 : Modelleme ve kodlama şartnamelerinde kapsanacak konular [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Düşük karmaşıklığın sağlanması	++	++	++	++
1b	Dil alt kümelerinin kullanımı	++	++	++	++
1c	Strong typing[2] sağlanması	++	++	++	++
1d	Defansif uygulama tekniklerinin kullanılması	+	+	++	++
1e	Güvenilir tasarım ilkelerinin kullanımı	+	+	++	++
1f	Belirsizliğe yer vermeyen grafiksel gösterimlerin kullanımı	+	++	++	++
1g	Stil kılavuzlarının kullanımı	+	++	++	++
1h	İsimlendirme kurallarının uygulanması	++	++	++	++
1i	Koşut zamanlılığa dikkat etmek	+	+	+	+

3.5.1 Yazılım gereksinimleri

Teknik güvenlik gereksinimleri ve sistem mimari tasarımına göre, fonksiyonel güvenliği sağlayacak yazılım için gereksinimler tanımlanır.

Yazılım gereksinimleri aşağıdaki karakteristik özellikleri taşımalıdır.

- Kesin, belirsizliğe yer vermeyen
- Anlaşılabilir
- Tekil
- Kendi içinde tutarlı
- Uygulanabilir
- Doğrulanabilir
- Gerekli (bir amaca hizmet için, yazılımsal karşılığı olan)
- Uygulamadan bağımsız. (Sadece neyin gerekli ve yeterli olduğu belirtilmeli, uygulamayı kısıtlamamalıdır.
- Eksiksiz. (Müşteri gereksinimlerini tamamen karşılamalıdır.)
- Uygun olma. (Yasal veya endüstriyel standartları ihlal etmeyecek şekilde olmalıdır.)

[2] 1c konusunun amacı, dilin doğasında bulunmayan güçlü yazım ilkelerini empoze etmektir

Yazılım gereksinimlerinin yukarıdaki özelliklerinin sağlanması için yazılırken doğal dil (natural language) kullanılmalı ve Çizelge 3.19'da paylaşılan tablodaki gereksinimlere dikkat edilmelidir.

Çizelge 3.19 : Güvenlik gereksinim belirlenmede kullanılabilecek notasyonlar [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Gereksinim yazımında resmi olmayan[3] gösterimler kullanmak	++	++	+	+
1b	Gereksinim yazımında yarı resmi [4]gösterimler kullanmak	+	+	++	++
1c	Gereksinim yazımında resmi gösterimler kullanmak	+	+	+	+

Yazılım gereksinimleri güvenlik yaşam döngüsü boyunca başka bir gereksinimle karışmaması için her bir gereksinim eşsiz olmalıdır. Bu birçok farklı şekilde başarılabilir. Gelişmiş ticari gereksinim yönetimi araçları kullanılarak veya yazılım gereksinimi yazma yönergesi ile sağlanabilir.

Yazılım gereksiniminin durum bilgisi (taslak, kabul edildi, incelendi vb.) ve ASIL bilgisi gibi öznitelikleriyle birlikte saklanmalıdır.

3.5.2 Yazılım mimarisi

Yazılım mimarisinin amacı, tüm yazılım komponentleri arasındaki etkileşimi tanımlamaktır. Bu yazılımın statik hem de dinamik yapısını kapsamaktadır. Yazılım mimarisi geliştirirken kullanılacak notasyon, anlaşılır tutarlı, basit, doğrulanabilir olma gibi özellikleri karşılaması beklenir. Standardda önerilen yöntemler Çizelge 3.20'de paylaşıldı.

[3] Daha yüksek seviyeli güvenlik gereksinimleri (ör. güvenlik hedefleri, işlevsel ve teknik güvenlik gereksinimleri) için doğal dil ve diğer resmi olmayan gösterim türleri en uygun biçimlerdir

[4] Yarı resmi gösterim, denklemler, grafikler, diyagramlar, akış şemaları, zamanlama diyagramları ve diğer birçok gösterim biçimi (örneğin UML® ve SysML™) gibi matematiksel veya grafik öğelerle desteklenen doğal dili kullanarak gereksinimleri formüle eder. Örnekler, modele dayalı teknikleri ve doğal dilde gereksinim cümleleri için şablonları ve kontrollü kelimeleri uygulama içerir.
Kesin donanım ve yazılım davranışlarının ve yeteneklerinin belirlenebildiği daha düşük seviyeli güvenlik gereksinimleri için, daha fazla netlik nedeniyle yarı resmi gösterimler daha uygundur. Ancak burada bile yarı formal tekniklerin her gereksinim için kullanılması mümkün veya gerekli olmayabilir.

Çizelge 3.20 : Yazılım mimarisi belirlemede kullanılabilecek notasyonlar [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Doğal dil	++	++	++	++
1b	Resmi gösterim	++	++	+	+
1c	Yarı resmi gösterim	+	+	++	++
1d	Resmi olmayan gösterim	+	+	+	+

Mimari yazılım gereksinimlerinin sağlanacağı birimler arasında iki yönlü takip edilebilirliği sağlamalıdır. Tasarlanan her bir birim en az bir yazılım gereksinimi ile eşleşmelidir. Ayrıca tüm yazılım gereksinimleri ilişkili olduğu yazılım birimlerine atanarak tam kapsama sağlanmalıdır.

Fonksiyonel güvenlik standardına göre Çizelge 3.21'de paylaşılan tablodaki tasarım ilkeleri dikkate alınarak tasarım yapılmalıdır.

Çizelge 3.21 : Mimari tasarım ilkeleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Yazılım bileşenlerinin uygun hiyerarşik yapısı	++	++	++	++
1b	Yazılım bileşenlerinin kısıtlı büyüklüğü ve karmaşıklığı	++	++	++	++
1c	Arayüzlerin büyüklüğünün sınırlanması	+	+	+	++
1d	Her yazılım bileşeninin güçlü uyumu	+	++	++	++
1e	Yazılım bileşenleri arasındaki bağlantının esnekliği	+	++	++	++
1f	Uygun zamanlama planlaması	++	++	++	++
1g	Interruptların[5] kısıtlı kullanımı	+	+	+	++
1h	Yazılım bileşenlerinin uygun uzamsal izolasyonu	+	+	+	++
1i	Paylaşılan kaynakların uygun yönetimi	++	++	++	++

Mimari elemanlarının statik tasarımı, hiyerarşik yapı, veri türü ve özellikleri, yazılım komponentleri ve gömülü yazılım ile arayüzleri, evrensel değişkenleri (global variables) içermektedir. Diğer yandan dinamik tasarım ise, kodun mantıksal bir çerçevede akması, verilerin işlenmesi, evrensel değişkenler üzerinden akan verilerin zamanlaması ve zamansal kısıtlamaları açıklar.

[5] Kesme (**interrupt**) işlemi, bilgisayarın merkezi işlem biriminin hiç hesapta olmayan bir olayın etkisiyle, normal olarak yapmakta olduğu işi bırakarak, kesmenin gösterdiği olaya geçici olarak yönelmesi için kullanılır.

Yazılım gereksinimde genel olarak belirlenen hata tespiti ve hata düzeltme gereksinimlerinin sağlanması için geçerliliği kabul görmüş yöntemler bulunmaktadır. Bu kapsamda hata tespiti için kullanılabilecek güvenlik mekanizmalarına örnekler verilmiştir.

- Giriş-Çıkış verilerinin aralık kontrolleri (Range check)
- Mantıklılık kontrolü (Plausibility check)
- Veri hatalarının tespiti (Detection of data errors)
- ASIC (Uygulamaya Özel Tümleşik Devre) veya bekçi köpeği (Watchdog) vazifesi gören başka bir yazılım ile mantıksal ve zamansal denetleme
- Yürütülen programın zamansal denetlenmesi
- Tasarımda çeşitli yedeklilik (Redundancy)
- Ortak hafızada paylaşılan kaynaklara erişimin donanımsal veya yazılımsal denetlemelerin uygulanması

Hata düzeltme için ise aşağıdaki yöntemler kullanılabilir.

- Güvenli duruma erişmek için devre dışı bırakma
- Statik kurtarma mekanizmaları (hata sonrası uygun bir şekilde güvenli duruma geçme mekanizmalarını kapsar.)
- Hatanın ani olumsuz etkilerini aza indirmek için yavaşlatılmış bozulma sağlama
- Homojen yedeklilik (Homogenous redundancy)
- Çeşitli yedeklilik (Diverse redundancy)
- Hata düzeltme algoritmaları
- Güvenlik ile ilgili kullanılan kaynaklara yazılım ve donanım erişim izinleri

3.5.3 Yazılım birim tasarımı

Bu bölüm yazılım mimarisinde tanımlanan birimlerin kodunun yazıldığı adımdır. Mimari referans alınarak kodlama standartlarına ve yönergelere uygun şekilde kod geliştirilir. Yazılan kod, önceki tasarım adımlarıyla tutarlı, kolay anlaşılabilir, sürdürülebilir ve doğrulanabilir olmalıdır. Karmaşıklıktan uzak, basit parçalara bölünmüş şekilde kod geliştirmek hem mikrodenetleyicinin kaynaklarını hem de mühendislik kaynaklarının optimize edilmesine yardımcı olur. Çizelge 3.22'de verilen tasarım kriterlerinin büyük bir kısmı MISRA C kurallarına uygun kod geliştirilmesi ile sağlanmaktadır.

Çizelge 3.22 : Yazılım birim tasarım ve uygulama ilkeleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Alt programlarda ve fonksiyonlarda tek giriş ve tek çıkış noktası	++	++	++	++
1b	Oluşturulma sırasında online testler yapılsa dahi dinamik obje ve değişkenlerin kullanılmaması	+	++	++	++
1c	Değişkenlerin başlangıç durumlarının belirlenmesi	+	+	++	++
1d	Değişken isimlerinin çoklu kullanılmaması	+	+	++	++
1e	Global değişkenlerin kullanımından kaçınılması, kullanılması durumunda gerekçelendirilmesi	+	+	++	++
1f	İşaretçi kullanımının kısıtlanması	+	+	++	++
1g	İstemsiz veri tipi dönüştürülmemesi	+	+	++	++
1h	Gizli veri akışı ve kontrol akışının olmaması	+	++	++	++
1i	Koşulsuz atlamaların olmaması	+	++	++	++
1j	Özyinelemelerin olmaması	+	++	++	++

3.5.4 Yazılım birim doğrulama

Bu aşamada testler yapılarak geliştirilen kodun, yazılım gereksinimlerini ihlal etmeden beklenen fonksiyonu gerçekleştirmesi beklenir. Bunun için bir doğrulama planına ihtiyaç vardır. Doğrulama planı oluşturulurken uygulanacak yöntem veya yöntemlerin yeterliliği, kodun karmaşıklığı, eğer bu çalışma var olan bir sistemin iyileştirilmesi adına yapılıyorsa geçmiş deneyimler ve doğrulama sürecinde kullanılan yöntemlerin olgunluğuna bağlı olarak belirlenir. Bu kapsamda kabul görmüş yöntemlerin listesi Çizelge 3.23'te paylaşılmıştır.

Çizelge 3.23 : Yazılım birim doğrulama yöntemleri [8].

Yöntem		ASIL A	ASIL B	ASIL C	ASIL D
1a	Walk-through[6]	++	+	0	0
1b	Pair programming[7]	+	+	+	+
1c	Denetim	+	++	++	++
1d	Yarı resmi doğrulama	+	+	++	++
1e	Resmi doğrulama	0	0	+	+
1f	Kontrol akış analizi	+	+	++	++
1g	Veri akış analizi	+	+	++	++
1h	Statik kod analizi	++	++	++	++
1i	Soyut yoruma bağlı statik analiz	+	+	+	+
1j	Gereksinime bağlı test	++	++	++	++
1k	Arayüz testi	++	++	++	++
1l	Hata enjeksiyon testi	+	+	+	++
1m	Kaynak kullanım değerlendirmesi	+	+	+	++
1n	Model ve kod arasında arka arkaya karşılaştırma testi (Eğer model tabanlı yazılım geliştiriliyorsa)	+	+	++	++

İdeal bir test ortamında, test edilecek yazılıma bir girdi seti verilerek sonuçlar alınır. Bu sonuçların gereksinimleri ihlal etmemesi durumunda doğrulama süreci başarıyla tamamlanır. Testin amacına göre farklı girdi setleri oluşturulur. Bu girdi setlerine test senaryoları da denilir. Test senaryoları oluşturulurken Çizelge 3.24'teki yöntemler kullanılabilir.

Çizelge 3.24 : Yazılım birim doğrulama için test senaryosu oluştuma yöntemleri [8].

Yöntem		ASIL A	ASIL B	ASIL C	ASIL D
1a	Gereksinim analizi	++	++	++	++
1b	Eşdeğerlik sınıflarının oluşturulması ve analizi	+	++	++	++
1c	Sınır değerlerinin analizi	+	++	++	++
1d	Bilgi ve deneyime dayalı hata tahmini	+	+	+	+

Yapılan testler sonucu hiçbir gereksinim ihlali elde etmemiş olunabilir, ancak bu sistemin hatasız olduğu anlamına gelmeyebilir. Test edilen yazılımın ne kadarının kapsandığını ölçmenin yöntemleri vardır. Bu yöntemler Çizelge 3.25'te verilmiştir.

[6] Geliştirilen kodun, ekip içerisinde baştan sona akışının açıklanması, gösterimi süreci işlemine verilen isimdir.
[7] İki yazılım gelişricinin bir bilgisayarda çalışmasıdır. Çalışanlardan biri yazılımı geliştirirken diğer yazılımcının diğer yazılımcıyı denetleme ve yönlendirmesi işlemidir.

Çizelge 3.25 : Yapısal kapsama metrikleri [8].

	Yöntem	ASIL A	ASIL B	ASIL C	ASIL D
1a	Statement coverage[8]	++	++	+	+
1b	Branch coverage[9]	+	++	++	++
1c	MC/DC (Modified condition/decision coverage)[10]	+	+	+	++

Doğrulama süreci benzer yöntemlerin farklı seviyelerde tekrarlanmasından oluştuğundan yazılım entegrasyon ve gömülü yazılım test süreçleri detaylı açıklanmaması uygun görüldü.

Testlerin hangi yöntemlerle yapıldığı kadar test ortamının da süreçte önemli bir yeri vardır. Test ortamı bir model, C kodu, geliştirme kiti veya gerçek donanımla sağlanıyor olabilir.

MiL (Model in the Loop): Model tabanlı geliştirme araçları yazılım geliştirme mühendislerine kendi arayüzlerinde C kodu oluşturmadan fonksiyonları modellemesine olanak sağlamaktadır. Daha sonrasında model kullanılarak C kodu üretilebilmektedir. Geliştirme model seviyesindeyken model tabanlı geliştirme araçları sağladıkları simülasyon ortamında geliştiricilere test yapabilme imkânı sağlamaktadır. Bu seviyede yapılan testler MiL olarak adlandırılmaktadır. Model ve kod arasında arka arkaya karşılaştırma testi (back-to-back comparison test between model and code) model ve otomatik kod üretimi sonrasında oluşabilecek farklılıkların tespiti için yapılır böylelikle modelleme ve kod arasındaki ilişki analiz edilebilir. Burada dikkat edilmesi gereken durum finalde kullanılacak mikrodenetleyici için derleme yapılarak bu testin yapılması daha doğru olacaktır. Bunun nedeni derlenen kodun komut kümesindeki farklılıktan dolayı tepkilerde değişiklikler meydana gelebilme ihtimalidir.

SiL (Software in the Loop): Otomatik kod üretimi veya geleneksel yöntemlerle elde edilen C kodunun bilgisayarda işletilerek test yapılmasına verilen isimdir.

---

[8] Statement coverage, tüm ifadelerin kaynak kodda en az bir kez yürütülmesini içeren bir beyaz kutu test tekniğidir.

[9] Branch coverage, bir kod tabanındaki tüm dalların testler tarafından uygulanıp uygulanmadığını gösteren bir ölçümdür. Bir "dal", bir karar ifadesinden sonra kodun alabileceği olası yürütme yollarından biridir.

[10] Modified condition/decision coverage (MC/DC), son derece kritik sistemleri test etmek için yazılım testlerinde kullanılan bir yöntemdir. MC/DC, diğer koşulları sabit tutarken her koşulun tüm olası durumlarının test edilmesini gerektirir.

PiL (Processor in the Loop): C kodunun projede kullanılacak işlemciye sahip bir geliştirme kiti için derlenerek geliştirme kitinde çalıştırılması ile yapılan testtir.

HiL (Hardware in the Loop): C kodunun, gerçek zamanlı işlem yürütebilen bir simülatör yardımı ile gerçek fiziksel bağlantılar ile çalıştırılmasıdır. Burada haberleşme arayüzlerin analizi yapılabilmektedir. Bu aşamaya kadar yapılan işlemler için zaman bir kısıt değilken bu test ile gerçek zamanlı tepkiler izlenebilir olmaktadır.

Gerçek donanım: Donanım tasarım süreci içerisinde ilk prototipin üretilmesinden sonra, gerçek donanım kullanılarak yapılan testlerdir. Eğer sistematik olarak ilerlenmiş ise hataların temel kaynağı donanım devrelerinden kaynaklanacaktır.

Test süreçlerin adım adım sistematik olarak ilerlemesindeki temel amaç her bir seviyenin ayrı ayrı doğrulanması sebebiyle hata ayıklama sürecinin sadeleştirilmesidir. İlk testin doğrudan gerçek donanım üzerinde yapılması durumunda anormal tepkilerin kaynağının bulunması zor ve zaman alıcı olacaktır. Ayrıca yazılım geliştirme ve donanım tasarım faaliyetleri paralel yürütülen süreçler olduğu unutulmamalıdır. Testlerin başlanması için bir prototipi beklemek ciddi zaman kaybına sebebiyet verecektir.

4. UYGULAMA

Bu bölümde konsept kısmındaki bilgiler ışığında direksiyon sistemi sinyalleri özelinde geliştirme adımları uygulanacaktır. İlk olarak 3 katmanlı denetleme mekanizması çerçevesinde projenin nasıl ele alınacağı açıklanacaktır. Bu adımda teknik güvenlik konsepti de açıklanmış olacaktır. Sonrasında fonksiyonel güvenlik gereksinimlerinden yazılım gereksinimleri türetilecektir. Yazılım gereksinimleri ve teknik güvenlik konsepti referansında yazılım mimarisi belirlenecektir. Uygulama örneği olacak fonksiyonlar geliştirilecek ve fonksiyonlar için birim doğrulamaları yapılarak uygulama bölümü tamamlanacaktır.

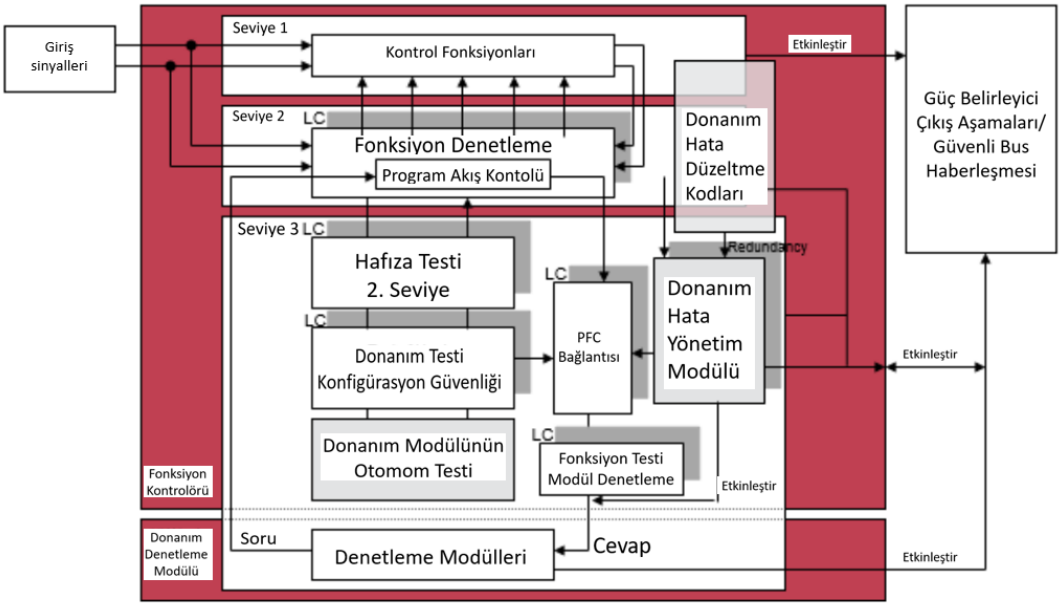
4.1 3 Katmanlı Denetleme Mekanizması

3 Katmanlı denetleme konsepti, EGAS siteminin izleme seviyeleridir. Fonksiyonel güvenlik süreçlerinin daha iyi organize edilebilmesi için süreci 3 seviyeye ayırır.

- 1.Seviye: İşlevselliğin bulunduğu katmandır. EPS sistemi özelinde, sensör bilgileri ile hesaplanacak yardımcı kuvvetin büyüklüğünün hesaplandığı ve kontrol algoritmaları kullanılarak elektrik motorunu süren ve takviye kuvvetin uygulanmasını sağlayan seviyedir. Bu seviye fonksiyonel güvenlik standardı kapsamında ASIL seviyesi QM olarak ele alınır. Bu nedenle bu seviye için güvenlik mekanizmalarından bahsetmek mümkün değildir.
- 2.Seviye: İşlev denetleme katmanıdır. 1. Seviye tarafından hesaplanan gerekli değerleri izleyerek anormal durumlar değerlendirilir. Sistematik ve rastgele donanımsal arızalar denetlenir, anormal sistem çıktıları karşısında güvenli durumlara FTTI (Fault tolerant time interval) olarak adlandırılan hata toleranslı zaman aralığında geçiş sağlanır.
- 3.Seviye: Kontrolör denetleme katmanıdır. Bu katman kontrolörün yazılım donanımda uygulanabilirliğini denetlemektedir. Bu seviye iki yapıdan oluşmaktadır. İlki mikro denetleyicinin donanım hata yönetim modülleri (Örnek olarak Memory Protection Unit, Cyclic Redundancy Check vb.) tarafından sağlanır. İkinci yapı ise denetleyici modül tarafından sağlanmaktadır. İşlevselliği sağlayan mikrodenetleyiciye sistematik sorular sorarak yürütülen programın doğruluğunu denetleyen ek bir modüldür.

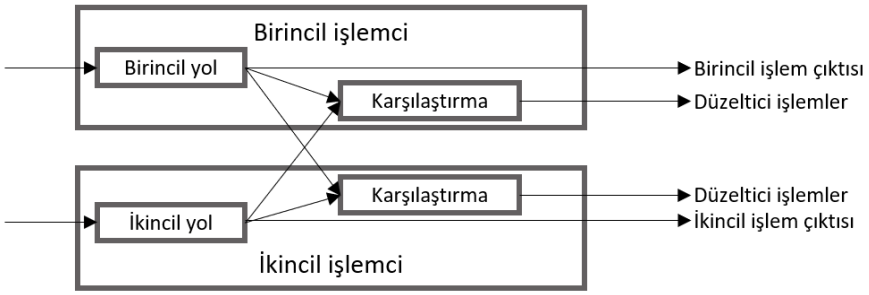
Denetleyici modül SBC (System Basis Chip) ASIC (Application Specific Integrated Circuit) veya basit bir mikro denetleyici kullanılarak sağlanabilir.

Bu seviyelerin hafıza alanları ve birbiri ile etkileşime geçmesi kısıtlanmıştır. Birbirinden bağımsız denetleme mekanizmalarının olması rastgele veya sistematik hatalara karşı bağışıklığı arttırmaktadır. Mimarinin katmanlarını açıklayan genel görünümü Şekil 4.1'de paylaşıldı.



Şekil 4.1 : Lockstep 3 katmanlı denetleme konsepti genel görünümü [13].

Çift çekirdekli Lockstep'e sahip 3 katmanlı denetleme konseptinin genel görünümü Şekil 4.2'de verilmektedir.



Şekil 4.2 : Fazlalıklı işlemci kullanımıyla yazılım karşılaştırması [8].

Çift çekirdekli Lockstep'e sahip işlemciler, karşılıklı olarak veri alışverişinde bulunur. Her birimde verilerin karşılaştırılması yapılarak tespit edilen farklılıklar arıza mesajı oluşturur. Bir işlem için beklenen zamanlama ve gerçek zamanlama arasındaki fark

jitter olarak tanımlanır. İşlemcilerde oluşabilecek jitter, haberleşme gecikmeleri vb. etkilerden korunmak için lockstep yapısı kullanılır.

4.2 Yazılım gereksinimlerinin belirlenmesi

NHTSA (National Highway Traffic Safety Administration) tarafından EPS sistemi geliştirilme aşamasında destek sağlamak amacıyla yayınlanan "Functional Safety Assessment Of a Generic Electric Power Steering System With Active Steering and Four-Wheel Steering Features" çalışmasından alınan güvenlik gereksinimleri incelendi. Çalışmanın amacı süreci açıklamak olduğu için referans çalışmanın sadece direksiyon sistemi sinyallerini ilgilendiren gereksinimler Çizelge 4.1'de listelendi.

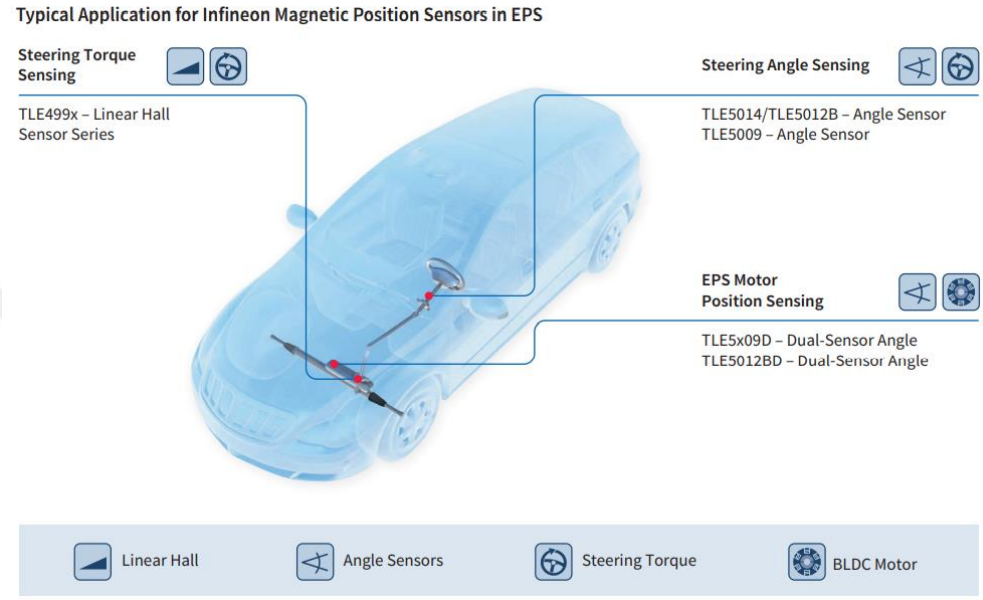
Çizelge 4.1 : EPS güvenlik gereksinimleri.

Yazılım Gereksinim Numarası	Güvenlik Hedefi	ASIL	Güvenlik Gereksinimi
1.15	1, 2	C, D	Yedekli elemanlar kullanılıyorsa, bunlar ortak arızalara karşı doğrulanmalıdır.
1.16	1, 2	C, D	Yedekli eleman kullanılır ve bunlardan birisinde arıza oluşursa, EPS sistemi FTTI zamanı içerisinde güvenli durum 1*'e geçmeli ve sarı renkli sürücü uyarısı verilmelidir.
1.17	1, 2	C, D	Yedekli eleman kullanılır ve bunların hepsinde arıza oluşursa veya tek eleman kullanılır ve arızalanırsa, EPS sistemi FTTI zamanı içerisinde güvenli durum 2*'e geçmeli ve kırmızı renkli sürücü uyarısı verilmelidir.
2.2	1, 2, 3	B, C, D	Direksiyon sistemi tork sensörü, direksiyon simidine uygulanan kuvvete karşılık burulma çubuğunun dönmesini ölçmek içindir. Bu değerin doğruluğu ve geçerliliğini sağlayacak nitelikte olmalıdır.
2.3	1, 2, 3	B, C, D	Tork sensörünün ölçümünün elektriksel sinyale dönüştürme yöntemi doğrulanmalıdır.
2.4	1, 2, 3, 4	A, B, C, D	Tork sensörü, yüksek ve alçak voltaj değerleri EPS sistemi aktifken denetlenmeli ve arıza durumunda güvenli durum 2*'e DSB ms içerisinde geçilmelidir.

Yazılım Gereksinim Numarası	Güvenlik Hedefi	ASIL	Güvenlik Gereksinimi
2.5	1, 2, 3, 4	A, B, C, D	Tork sensörü tarafından yapılan ölçüm, EPS kontrol modülüne iletilerek, geçerlilik, doğruluk ve rasyonellik açısından kalifiye olacaktır.
2.6	1, 2, 3, 4	A, B, C, D	Tork sensörü verileri sürekli olarak incelenecek, sadece tork verisi EPS kontrol modülü tarafından doğru bir şekilde alındığında güç desteği sağlanacaktır.
2.8	1, 2, 3, 4	A, B, C, D	Tork sensörü çıktısı, izleme devresindeki veri analiz ünitesi tarafından analiz edilecektir.
2.9	1, 2	C, D	Tork sensörünün sağlığı tüm çalışma koşullarında denetlenmeli ve doğrulanmalıdır.
2.10	1, 2, 3, 4	A, B, C, D	Tork sensöründe güvenlik hedefini ihlal eden bir hatada EPS sistemine bilgi verilecektir.
2.11	1, 2, 3	QM, A, B	Tork sensörü, elektromanyetik uyumluluk EMC/EMI, ESD, kontaminasyon, tek olay etkileri ve çevresel koşullardan kaynaklanan arızalar ile ilgili arıza teşhisine sahip olmalıdır.
2.13	2	A	Direksiyon simidi açı sensörü, direksiyon simidi girişinden kaynaklanan açı değişimini ölçmek içindir ve bu ölçümü yapabilecek nitelikte olmalıdır.
2.14	2	A	Direksiyon simidi açı sensörü ölçümünün elektriksel sinyale dönüştürme yöntemi doğrulanmalıdır.

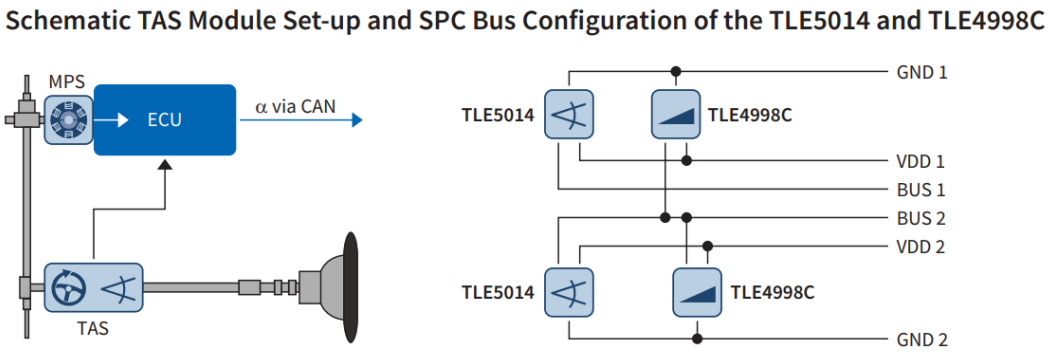
Yazılım Gereksinim Numarası	Güvenlik Hedefi	ASIL	Güvenlik Gereksinimi
2.15	2	A	Direksiyon simidi açı sensörü tarafından yapılan açı ölçümü, EPS kontrol modülüne iletilmelidir.
2.16	2	A	Güvenlik hedefi 2’yi ihlal edebilecek gizli hataları önlemek için açı sensörü verileri düzenli aralıklarla kontrol edilmelidir.
2.17	2	A	Güvenlik hedefi 2'yi ihlal eden bir arıza durumunda, direksiyon simidi açı sensörü, arızayı EPS kontrol modülüne iletecektir.
2.18	2	A	Direksiyon simidi açı sensörü, elektromanyetik uyumluluk EMC/EMI, ESD, kontaminasyon, tek olay etkileri ve çevresel koşullardan kaynaklanan arızalar ile ilgili arıza teşhisine sahip olmalıdır.
2.19	2	A	Güvenlik hedefi 2'nin ihlaline yol açan tüm tek nokta direksiyon açı sensörü donanım arızaları, arıza tespit aralığı içinde tespit edilecek ve FTTI içinde azaltılacaktır. Arıza durumunda sistem sarı ışıklı sürücü uyarısı verecektir.

Sektördeki yarı iletken ve sensör üreticileri otomotiv uygulamalarına özel kendi donanımsal çözümlerini sunmaktadır. Infenion Technologies AG, ISO 26262 Kara Araçları Fonksiyonel Güvenlik standardına uygun çözümler üretmektedir. Bu çalışmada kullanılan konfigürsyon, Infenion Technologies AG EPS çözümü referans alınmış ve sensör yerleşimleri Şekil 4.3'te paylaşılmıştır.



Şekil 4.3 : EPS'de kullanılan sensörler ve yerleşimleri [14].

Özellikle, SPC arayüzü, birleşik Tork-Açı-Sensörü (TAS) modüllerinde bir bus hattı üzerinde açı sensör(ler)inin ve lineer Hall sensör(ler)inin bağlanmasına izin verir. Ayrı tork sensörü ve açı sensörü modüllerine sahip geleneksel tasarımlarla karşılaştırıldığında, bu konfigürasyon kablolama maliyetini düşürür ve modül alanından tasarruf sağlar. TAS modülünün şematik gösterimi Şekil 4.4'te paylaşılmıştır.



Şekil 4.4 : Tork-açı sensör (TAS-Torque-Angle Sensor) modülü [14].

Yukarıda listelenen güvenlik gereksinim maddeleri incelenerek yazılım gereksinimleri türetildi.

1.15 güvenlik gereksiniminde fazlalık eleman kullanımı durumunda ortak arızalardan kaçınılması gerektiği belirtildi. Yedekli eleman kullanmak, elemanın arızası durumunda sistemin güvenli durumda kalması için fayda sağlar. Yüksek hızla sürüşte iken tork sensöründe bir arıza meydana gelmesi durumunda EPS fonksiyonları devre dışı kalmasına sebebiyet vereceğinden yedekli sensör kullanmak sistem güvenliği için tercih edildi.

Aşağıda listelenen durumlar ortak arızaya sebebiyet vermektedir.

- Aynı tasarım tekniğini kullanmak
- Aynı donanımı kullanmak
- Aynı yazılımı kullanmak
- Aynı arayüze sahip eleman kullanmak
- Aynı çevresel koşullara elemanları yerleştirmek

Bu gereksinim seçilen ISO 26262 standardına uyumlu bir topoloji ve donanım seçildiği için gereksinim donanım ve temel yazılım seviyesinde karşılanmaktadır. AUTOSAR mimarisine uygun tasarım yapıldığı için donanım temel yazılım seviyesinde soyutlanarak uygulama katmanına sensör sağlık durumu iletilmelidir. Uygulama seviyesinde sensör arıza bilgisini alabilmek için arayüz bilgilerine ihtiyaç vardır.

Aşağıda 1.15, 1.16, 1.17 fonksiyonel güvenlik gereksinimlerinden Çizelge 4.2’deki yazılım gereksinimleri türetilmiştir.

Çizelge 4.2 : Yazılım gereksinimleri grup 1

Yazılım Gereksinim Numarası	Yazılım Gereksinimi	ASIL	Durum
1.1	1. direksiyon simidi pozisyon sensörü sağlık durumu bilgisi “SteeringPosSens1HealthStatus” değişkeni ile uygulama seviyesine 1ms aralıklarla paylaşılmalıdır. SteeringPosSens1HealthStatus = 1 direksiyon pozisyon sensörü 1 HEALTY SteeringPosSens1HealthStatus = 0 direksiyon pozisyon sensörü 1 UNHEALTY	D	Taslak
1.2	2. direksiyon simidi pozisyon sensörü sağlık durumu bilgisi “SteeringPosSens2HealthStatus” değişkeni ile uygulama seviyesine 1ms aralıklarla paylaşılmalıdır. SteeringPosSens2HealthStatus = 1 direksiyon pozisyon sensörü 2 HealthStatus_T.HEALTY SteeringPosSens2HealthStatus = 0 direksiyon pozisyon sensörü 2 HealthStatus_T.UNHEALTY	D	Taslak
1.3	1. direksiyon simidi tork sensörü sağlık durumu bilgisi “SteeringTorqSens1HealthStatus” değişkeni ile uygulama seviyesine 1ms aralıklarla paylaşılmalıdır. SteeringTorqSens1HealthStatus = 1 direksiyon tork sensörü 1 HealthStatus_T.HEALTY SteeringTorqSens1HealthStatus = 0 direksiyon tork sensörü 1 HealthStatus_T.UNHEALTY	D	Taslak
1.4	2. direksiyon simidi tork sensörü sağlık durumu bilgisi “SteeringTorqSens2HealthStatus” değişkeni ile uygulama seviyesine 1ms aralıklarla paylaşılmalıdır. SteeringTorqSens2HealthStatus = 1 direksiyon tork sensörü 2 HealthStatus_T.HEALTY SteeringTorqSens2HealthStatus = 0 direksiyon tork sensörü 2 HealthStatus_T.UNHEALTY	D	Taslak

Yazılım Gereksinim Numarası	Yazılım Gereksinimi	ASIL	Durum
1.5	“SteeringPosSens1HealthStatus” ve “SteeringPosSens2HealthStatus” sinyallerinden sadece birinin sağlıksız olması durumunda RedundancySafetyMech1 = SafetyMech_T.ACTIVE olmalıdır.	D	Taslak
1.6	“SteeringPosSens1HealthStatus” ve “SteeringPosSens2HealthStatus” sinyallerinden ikisinin biden sağlıksız olması durumunda RedundancySafetyMech2 = SafetyMech_T.ACTIVE olmalıdır.	D	Taslak
1.7	“SteeringPosSens1HealthStatus” ve “SteeringPosSens2HealthStatus” sinyallerinden ikisinin biden sağlıklı olması durumunda RedundancySafetyMech1 = SafetyMech_T.INACTIVE ve RedundancySafetyMech2 = SafetyMech_T.INACTIVE olmalıdır.	D	Taslak
1.8	“SteeringTorqSens1HealthStatus” ve “SteeringTorqSens2HealthStatus” sinyallerinden sadece birinin sağlıksız olması durumunda RedundancySafetyMech1 = SafetyMech_T.ACTIVE olmalıdır.	D	Taslak
1.9	“SteeringTorqSens1HealthStatus” ve “SteeringTorqSens2HealthStatus” sinyallerinden ikisinin biden sağlıksız olması durumunda RedundancySafetyMech2 = SafetyMech_T.ACTIVE olmalıdır.	D	Taslak
1.10	“SteeringTorqSens1HealthStatus” ve “SteeringTorqSens2HealthStatus” sinyallerinden ikisinin biden sağlıklı olması durumunda RedundancySafetyMech1 = SafetyMech_T.INACTIVE ve RedundancySafetyMech2 = SafetyMech_T.INACTIVE olmalıdır.	D	Taslak

2.2 ve 2.13 güvenlik gereksinimleri ölçüm yönteminin niteliği ile ilgili, 2.3 ve 2.14 ölçümün elektriksel sinyale dönüştürülme methoduyla ilgili olduğu görülmüştür. Bu gereksinimler seçilen donanımın yetkinliği ile ilgili olup yazılım seviyesinde bir gereksinim ile ilişkili değildir. 2.4 güvenlik gereksinimi ise beklenilen değerler dışında yüksek veya alçak voltajın okunması gibi kontroller güvenilirlik kontrolü (plausibility check) kapsamında değerlendirilir. Ancak seçilen sensör doğrudan bir voltaj çıkışı değil kendi iç denetimlerinden geçirdiği veriyi haberleşme protokolüne uygun bir şekilde iletmektedir. Detay için ekte paylaşılan sensör teknik dokümanı incelenebilir.
2.6 güvenlik gereksiniminde veri iletişiminin sağlıklı olması ile ilgilidir. Haberleşme protokolünün sağlıklı bir şekilde sağlanamaması durumunda temel yazılım seviyesinde sensör arızası olarak değerlendirilir. Bu bilgi uygulama seviyesine sensör sağlık bilgisi ile aktarılabilir.
2.8 güvenlik gereksinimi sensörün donanımsal veya yazılım seviyesinde veri analizinin yapılmasından bahsetmektedir. TLE 4998 sensörü bunu donanımsal olarak sağlamaktadır.
2.9, 2.10, 2.11 güvenlik gereksinimleri verinin sağlıklı iletilmesi için yapılan denetimlere yöneliktir. Çevre şartlarından etkilenmesi durumlarının denetlenmesi ve hata sinyallerinin oluşturulması ile ilgilidir. Çevresel şartlar sıcaklık, elektromanyetik etkilere karşı sensörün bağışıklığı ve hata denetimiyle ilgilir. Bunlar donanımsal olarak sensörün tümleşik devresi tarafından yapılmaktadır. Sensör standartlaştırılmış SAE J2716 tarafından belirlenmiş “Single Edge Nibble Transmission” protokolüne uygun olarak ilgili hataları oluşturmaktadır. Bu çalışmada yazılımın uygulama seviyesi gereksinimleri belirleneceği ve bu işlemlerin temel yazılım seviyesinde donanım soyutlama sağlanacağından dolayı bu gereksinimler sensör sağlığı için atanan değişkenlerin işlendiği temel yazılım katmanına atanmalıdır.
2.15 gereksiniminde direksiyon simidi pozisyon sensörünün EPS kontrol modülüne iletilmesi belirtilmiştir. Buna benzer bir gereksinimin direksiyon simidi tork sensörü bilgileri için de gereklidir. 2.16 gereksinimde gizli hataların etkisini azaltmak için belirli aralıklarla direksiyon simidi pozisyon sensörünün verilerinin belirli aralıklarla denetlenmesini istemektedir. Bu güvenlik gereksinimlerini sağlayabilmek için hata tolerans zaman aralığını aşmayacak şekilde örnekleme yapılarak, güvenlik mekanizmalarının çıktılarına debounce (elektrik akımındaki küçük dalgalanmaları

kaldırma) algoritması uygulanması ile sağlanabilir. Çizelge 4.3'deki gereksinimler bu değerlendirmeler sonucunda yazılmıştır.

Çizelge 4.3 : Yazılım gereksinimleri grup 2.

Yazılım Gereksinim Numarası	Yazılım Gereksinimi	ASIL	Durum
1.11	1. direksiyon simidi pozisyon sensörü değeri "SteeringPosSens1" 1 ms aralıklarla uygulama seviyesi ile paylaşılmalıdır.	A	Taslak
1.12	2. direksiyon simidi pozisyon sensörü değeri "SteeringPosSens2" 1 ms aralıklarla uygulama seviyesi ile paylaşılmalıdır.	A	Taslak
1.13	1. direksiyon simidi tork sensörü değeri "SteeringTorkSens1" 1 ms aralıklarla uygulama seviyesi ile paylaşılmalıdır.	D	Taslak
1.14	2. direksiyon simidi tork sensörü değeri "SteeringTorkSens2" 1 ms aralıklarla uygulama seviyesi ile paylaşılmalıdır.	D	Taslak
1.15	"RedundancySafetyMech1" sinyaline debounce algoritması uygulanmalıdır.	D	Taslak
1.16	"RedundancySafetyMech2" sinyaline debounce algoritması uygulanmalıdır.	D	Taslak
1.17	"RationalitySafetyMech1" sinyaline debounce algoritması uygulanmalıdır.	D	Taslak

2.17 ve 2.18 güvenlik gereksinimleri donanımsal arıza sinyalleri tarafından sağlanmaktadır. İlgili sinyaller işlenerek temel yazılım geliştirme seviyesinde sensör arızası sinyalleri ile uygulama seviyesine aktarılabilir. Bu yöntem ile uygulama seviyesi ve donanım seviyesi temel yazılım katmanı kullanılarak soyutlanmıştır.

2.19 güvenlik gereksinimi tek nokta hatası (single point fault) ile ilişkilidir. Tek nokta hatası, bir elemanın arızası sonucu bütün sistemin işlevselliğini bozan hatalar olarak nitelendirilir. Direksiyon simidi pozisyon sensörü fazlalıklı kullanımı, farklı sensör berleme hatlarına sahip şekilde ve farklı fiziksel konumlara yerleştirilmesiyle bu gereksinim sağlanır.

2.5 güvenlik gereksinimi, sensörden alınan bilginin rasyonel denetimlerden geçmesini kapsar. Bunun için farklı sinyaller arasında mantıksal bir ilişki kurmayı gerektirir. Buna örnek olarak birbirinden bağımsız olarak ölçülen iki sensör değeri karşılaştırıldığında mantıksal bir tolerans aralığında birbiri ile tutarlı olmalıdır. Bu kapsamda tork sensörü ve pozisyon sensörleri de mantıksal denetimlerden

geçirilebilir. Şekil 4.4'te paylaşılan yazılım gereksinimleri mantıksal denetimleri kapsayıcı şekilde yazıldı.

Çizelge 4.4 : Yazılım gereksinimleri grup 3.

Yazılım Gereksinim Numarası	Yazılım Gereksinimi	ASIL	Durum
1.18	Direksiyon simidi tork sensörü "SteeringTorqSensValue" değeri sağlıklı "SteeringTorqSens1Value" ve "SteeringTorqSens2Value" değerlerinin ortalaması ile hesaplanmalıdır. Eğer değerlerden biri sağlıklı değil ise sağlıklı olan değer kabul edilmeli, ikisi de sağlıksız ise varsayılan "STEERING_TORQ_DEF_VAL" değeri kabul edilmelidir.	D	Taslak
1.19	Direksiyon simidi pozisyon sensörü "SteeringPosSensValue" değeri sağlıklı "SteeringPosSens1Value" ve "SteeringPosSens2Value" değerlerinin ortalaması ile hesaplanmalıdır. Eğer değerlerden biri sağlıklı değil ise sağlıklı olan değer kabul edilmeli, ikisi de sağlıksız ise varsayılan "STEERING_POS_DEF_VAL" değeri kabul edilmelidir.	A	Taslak
1.20	"SteeringPosSensValue" merkezi pozisyona DSB tolerans değerinden daha yakın iken "SteeringTorqSensValue" değeri DSB tolerans değerinden büyük olması durumunda RationalitySafetyMech1 = SafetyMech_T.ACTIVE, küçük olması durumunda RationalitySafetyMech1 = SafetyMech_T.INACTIVE olmalıdır.	D	Taslak
1.21	"SteeringTorqSensValue" merkezi pozisyona DSB tolerans değerinden daha yakın iken "SteeringPosSensValue" değeri DSB tolerans değerinden büyük olması durumunda RationalitySafetyMech1 = SafetyMech_T.ACTIVE, küçük olması durumunda RationalitySafetyMech1 = SafetyMech_T.INACTIVE olmalıdır.	D	Taslak

Bu bölüme kadar yazılan yazılım gereksinimleri güvenlik mekanizmalarının nasıl hesaplanacağı açıklandı. Bundan sonra oluşturulan güvenlik mekanizmalarına göre hata tolerans zaman aralığında güvenli durumlara geçme stratejisinin belirlendiği state

machine (durum makinesi) açıklanacaktır. Tanımlanan RedundancySafetyMech1, RedundancySafetyMech2 ve RationalitySafetyMech1 güvenlik mekanizmalarının gerçekleşmesi durumunda geçiş şartlarına bağlı olarak güvenli durum 1 ve güvenli durum 2 kararları alınacak ve bu durumlarda alınacak reaksiyonlar tanımlanacaktır. Bu kapsamda Çizelge 4.5'teki gereksinimler yazıldı.

Çizelge 4.5 : Yazılım gereksinimleri grup 4.

Yazılım Gereksinim Numarası	Yazılım Gereksinimi	ASIL	Durum
1.22	Başlangıç durumu "DefSST" olmalıdır. Bu durumda; SafeState1 = SafeState_T.INACTIVE SafeState2 = SafeState_T.INACTIVE olmalıdır.	D	Taslak
1.23	RedundancySafetyMech2 = SafetyMech_T.ACTIVE olduğunda "SST2" durumuna geçilmelidir. Bu durumda; SafeState2 = SafeState_T.ACTIVE olmalıdır.	D	Taslak
1.24	RedundancySafetyMech1 = SafetyMech_T.ACTIVE veya RationalitySafetyMech1 = SafetyMech_T.ACTIVE olduğunda "SST1" durumuna geçilmelidir. Bu durumda; SafeState1 = SafeState_T.WARNING olmalıdır.	D	Taslak
1.25	SST1 durumunda VehicleSpeed değeri EPSSpeedLimit değerini geçmeden, RedundancySafetyMech1 = SafetyMech_T.INACTIVE ve RationalitySafetyMech1 = SafetyMech_T.INACTIVE olduğunda DefSTT durmuna geçilmelidir. Bu durumda; SafeState1 = SafeState_T.INACTIVE olmalıdır.	D	Taslak
1.26	"SafeState1" durumunda iken VehicleSpeed değeri EPSSpeedLimit değerini geçmesi durumunda SafeState1 = SafeState_T.ACTIVE olmalı ve sistem kapatılana kadar kalıcı olmalıdır.	D	Taslak

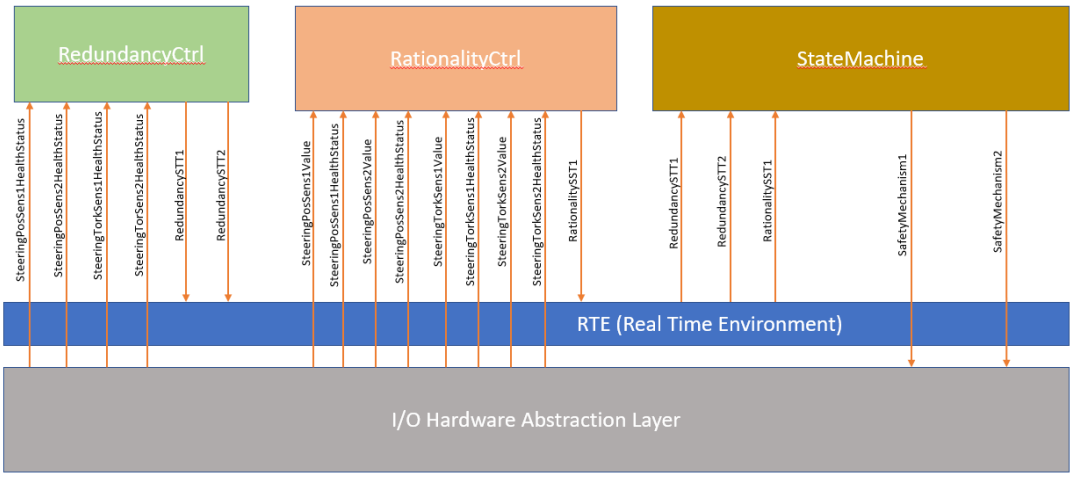
Tanımlanan gereksinimler mimari tasarım sürecinin referanslarını oluşturacaktır.

4.3 Yazılım Mimarisinin Belirlenmesi

Yazılım mimarisi, yazılım elemanlarının birbiri ile etkileşimlerini ve hiyerarşik yapılarını açıklayarak sistemin organizasyonunu tasvir eder. Komponentler arasındaki veri akışını sağlayan arayüzler gibi statik yönler ve kod akışı ve zamansal işlemler gibi dinamik yönler farklı diyagramlarla açıklanır.

Yazılım mimarisi geliştirilirken, microdenetleyicinin sahip olduğu kabiliyetlere ve hata denetleme mekanizmalarına hâkim olmak fonksiyonel güvenliği sağlamak içi önemlidir. Pratikte donanımsal güvenlik mekanizmaları, uygulama katmanından soyutlanmaktadır. Bunun temelinde geliştirilen uygulamaların donanımdan bağımsız hale getirilerek farklı uygulama ve donanımlarda kullanılabilirliğini arttırarak yeniden geliştirme masraflarından kaçınmaktır. Bu nedenle temel yazılım katmanı olarak adlandırılan katmanda zorunlu olmamakla birlikte sektörel seviyede kabul gören AUTOSAR mimarisi kullanılarak ortak bir taban oluşturuldu. Mikrodenetleyici üreticileri de bu yaklaşıma adaptasyon sağlamış MCAL(Microcontroller Abstraction Layer) fonksiyonlarını üreticiler ile paylaşılmaktadır. Bu nedenle temel yazılım katmanı genel olarak farklı üreticiler olmasına rağmen birbiri ile benzerlik gösterir. Bu nedenle, farklılığı yaratan bölüm uygulama seviyesindeki mimarilerden kaynaklanır. Bu çalışmada da uygulama seviyesini açıklayıcı diyagramlar paylaşıldı.

İlk olarak, komponentler arası veri akışını açıklayıcı Veri Akış Diyagramı (Data Flow Diagram) Şekil 4.5'te paylaşıldı.



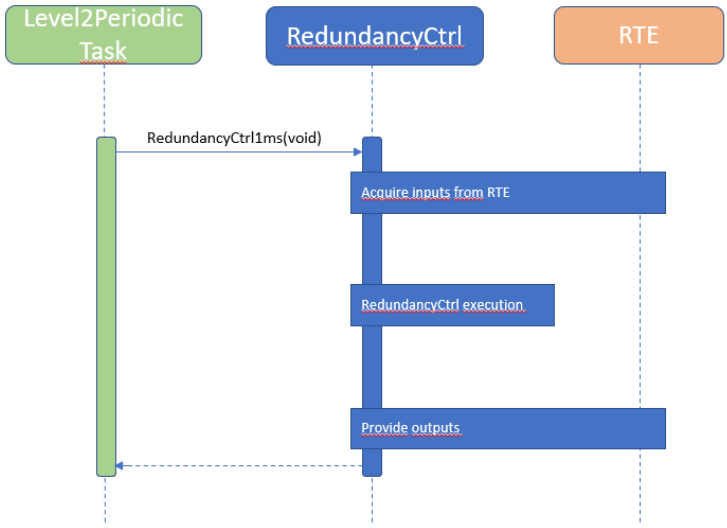
Şekil 4.5 : Veri akış diyagramı (Data flow diagram).

Bu diagram ile temel yazılım seviyesinden okunan ve yazılan verilerin akışı ve komponentler arasındaki etkileşim belirlenmiştir. Gelişmiş mimari tasarım araçlarında

arayüz daha fazla bilgiyi destekleyecek şekilde tasarlanmıştır. Sinyallerin veri tipleri gibi ek bilgileri de içerir. Bu bilgiler kullanılarak komponentlerin giriş ve çıkış sinyalleri tanımlanır.

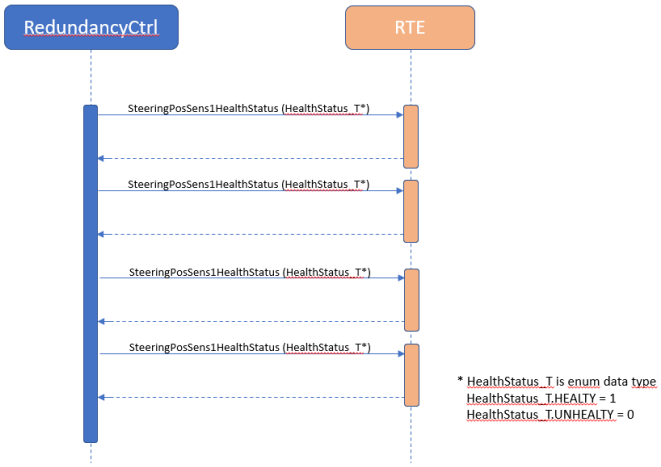
Bir başka destekleyici diyagram ise sekans diyagramlarıdır (Sequance Diagram). Sekans diyagramları ile çağırılacak fonksiyonlar ve bu fonksiyonların yapacağı işlemlerin yazılım birimlerine kırılımı yapılır. Yazılım birimlerine ilgili yazılım gereksinimleri atanır. Yazılım gereksinimlerinin mimari seviyede iki yönlü takibinin yapılması için profesyonel yazılımlardan destek alınır. Aşağıda RedundancyCtrl komponentinin sekans diyagramları paylaşıldı.

Şekil 4.6'daki sekans diyagramı 1 milisaniyelik periyodik görev süresinde bir çağırılan RedundancyCtrl1ms fonksiyonu olduğunu, bu fonksiyonda kullanılan giriş sinyallerinin RTE'den okunduğunu sonrasında fonksiyonun görevini yerine getirdiğini, son olarak da fonksiyon çıkış sinyallerinin RTE ile paylaşıldığı anlaşılmaktadır.



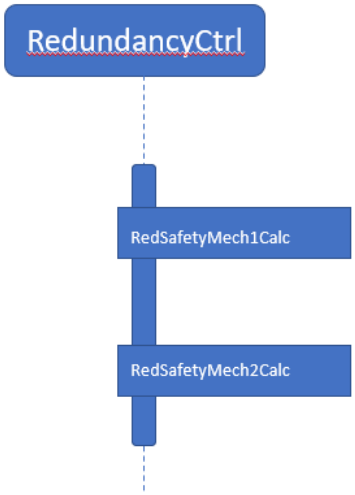
Şekil 4.6 : Sekans diyagramı örneği 1.

Şekil 4.7'deki sekans diyagramında RedundancyCtrl1ms fonksiyonunun "Acquire inputs from RTE" bölümünün içeriği gösterilerek giriş sinyalleri açıklanmaktadır. Giriş sinyallerinin RTE'den okunuş sırası ve veri tipi bilgileri paylaşılmaktadır.



Şekil 4.7 : Sekans diyagramı örneği 2.

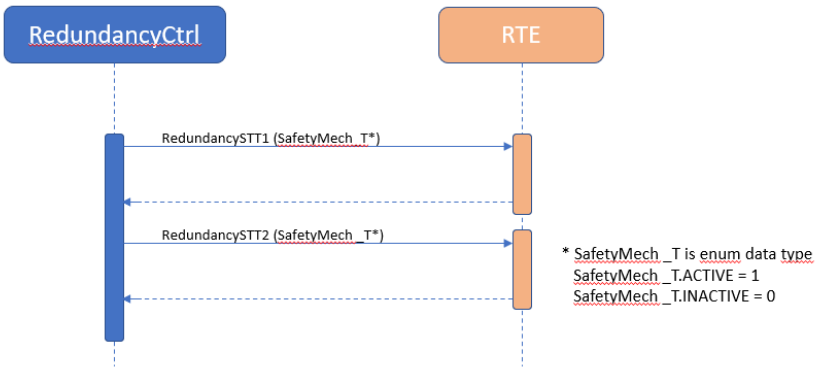
Şekil 4.8'deki sekans diyagramında RedundancyCtrl1ms fonksiyonunun "RedundancyCtrl execution" bölümünde yürütülecek yazılım birimleri tanımlanmaktadır. Eğer bir yazılım mimari tasarım programı kullanılıyor ve gereksinim yönetimini destekliyorsa, yürütülecek yazılım birimlerine yazılım gereksinimleri atanarak iki yönlü takip edilebilirlik sağlanır. Örnek olarak paylaşılan fonksiyon RedSafetyMech1Calc ve RedSafetyMech2Calc yazılım birimlerinden oluşmaktadır.



Şekil 4.8 : Sekans diyagramı örneği 3.

Şekil 4.9'da paylaşılan sekans ile hesaplanan değerlerin RTE'e yazılma sekanksı paylaşılmaktadır.

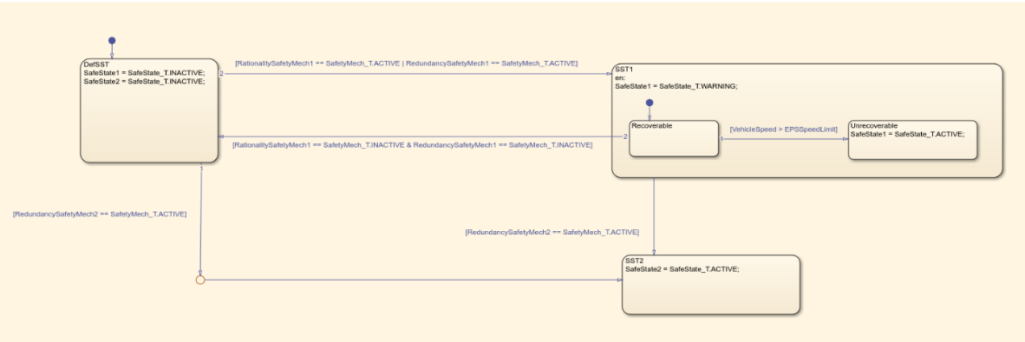
Şekil 4.9'da paylaşılan sekans diyagramında, ilk sekans diyagramında gösterilen "Provide outputs" kısmının detayları gösterilmektedir. Bu diyagramda RedundancyCtrl fonksiyonu içerisinde hesaplanan değerlerin RTE'a yazılma sırası ve veri tipi bilgileri açıklanmaktadır.



Şekil 4.9 : Sekans diyagramı örneği 4.

Sekans diyagramları kodun dinamik yapısını açıklamak için kullanılır. Böylelikle geliştirilen fonksiyonun yazılım birimlerine kırılımı yaplmaktadır. Yazılım birimlerinin, yazılım gereksinimleri ile ilişkileri ve yazılım birimlerinin yürütülme sırası açıklanmaktadır. Bu bilgiler detay tasarım kısmında kodun geliştirilmesi sürecinde kullanılacaktır.

Durum makineleri, sistem mod yönetimi ve arıza yönetimi gibi karmaşık sistemleri açıklamak için kullanılan etkili bir yöntemdir. Durumlar arasında geçişler belirlenen koşullara göre sağlanır ve durum bilgisine göre reaksiyonlar yürütülür. Şekil 4.10'da güvenlik mekanizmalarının yürütüldüğü StateMachine fonksiyonunun durum makinesini paylaşıldı.



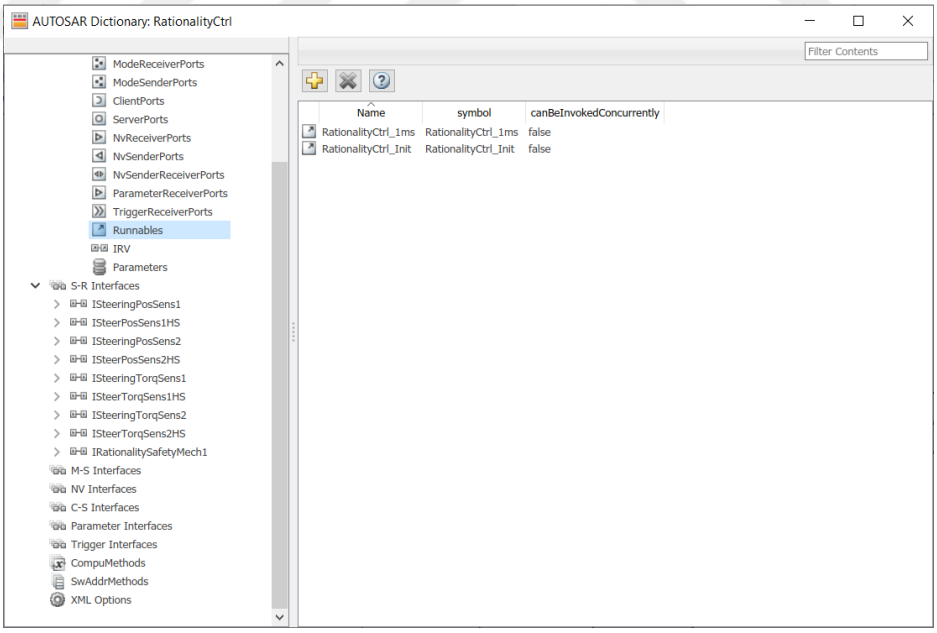
Şekil 4.10 : Durum makinesi diyagramı (State machine).

Otomotiv elektronik kontrol sistemlerini geliştirmek geniş bir ekip ve iş birliği gerektirir. Komponentler geliştirilirken ekip içi iletişimi yüksek tutmak ve geliştirme sürecinde referans alınacak bir mimari tasarıma sahip olmak tüm ekibin uyum içerisinde çalışmasını sağlamaktadır. Mimari tasarım kapsamında paylaşılan diyagramlar ve bilgiler ile fonksiyon detay tasarım süreci için bir çerçeve çizmektedir.

4.4 Yazılım Birim Tasarımı

Bu bölümde yazılım geliştirme sürecinde dikkat edilmesi gereken noktalara değinilerek uygulama örnekleri paylaşılacaktır. Konu akışını bölmemek adına detay tasarımlar Ek A'da paylaşıldı.

Yazılım gereksinimleri ve yazılım mimarisi, yazılım birim tasarımı sürecinde yapılacak geliştirmeler için bir çerçeve çizerek belirsizlikleri ortadan kaldırmaktadır. İlk olarak geliştirilecek fonksiyonun statik yönleri tanımlanarak sürece başlanır. Daha önce mimari tasarım bölümünde paylaşılan veri akış diyagramı incelendiğinde runnable (RTE tarafından tetiklenebilen, komponent tarafından sağlanan bir sıra işlemler dizisir.) ve RTE arayüzleri Şekil 4.11'de açıkça görülmektedir.



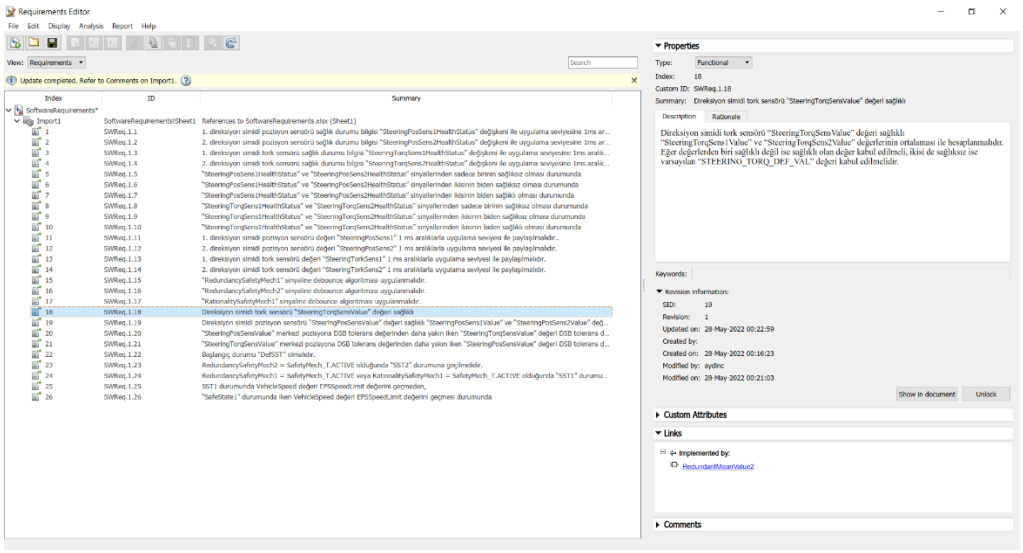
Şekil 4.11 : AUTOSAR kütüphanesi.

AUTOSAR tanımlamaları yapıldıktan sonra yazılım birimleri arasındaki sinyal ve parametre tanımlamaları yapılır. Kullanılan sinyallerin yazılım birimleri arasındaki ilişkileri sekans diyagramları referans alınarak oluşturulur. Son olarak yazılım

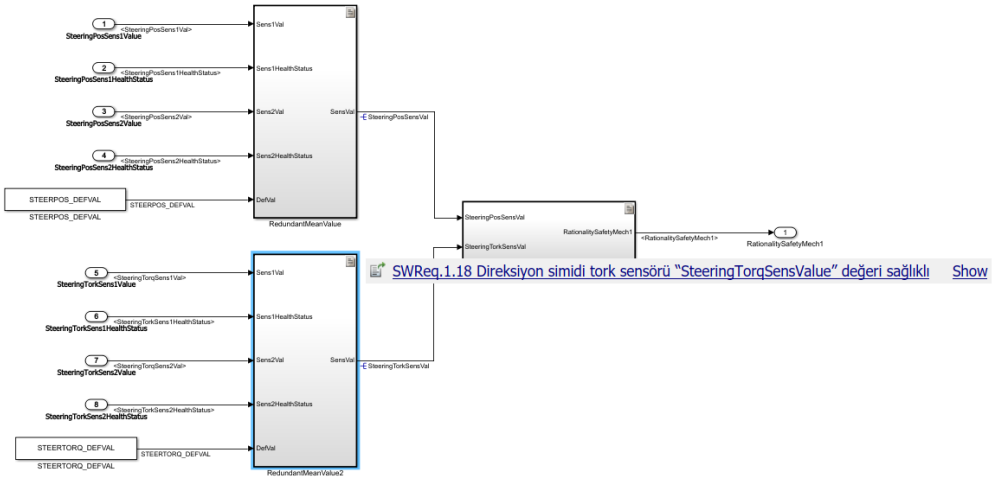
birimlerinin yerine getirmekle sorumlu oldukları fonksiyonlar yazılım gereksinimlerini sağlayacak şekilde geliştirilir. İyi bir yazılım geliştirmenin temelinde ihtiyaçların açıkça belirlenmesi ve bu ihtiyaçlar ve kısıtlamalar çerçevesinde yazılım gereksinimlerinin belirlenmesi yatar. Kafa karışıklığına neden olmayan anlaşılabilir açık gereksinimler kod kalitesini arttırır ve ileride hata düzeltme süreçlerini kısaltır.

Günümüzde otomotiv sektöründe modern bir araçta ortalama 100 milyon satır kod bulunmaktadır. [notepad] Böylesine karmaşık bir sistemde gereksinimlerin takibi oldukça zorlaşmaktadır. Bunun için iki yönlü takip edilebilirliği arttırmak için, gereksinim, mimari, model seviyesinde tasarım, C kodu ve test süreçlerinde mümkün olduğunca takip edilebilirliği arttıracak araçlar kullanılması bir gereklilik haline gelmektedir. Uygulama bölümünde lisans problemleri nedeniyle profesyonel gereksinim yönetim modülleri ve mimari tasarım programında örnek paylaşılamamaktadır. Buna rağmen yazılım birim tasarımı sürecinde kullanılan model tabanlı yazılım geliştirme aracında bir uygulama yapılarak genel yapı hakkında bilgi verilmektedir.

Yazılım gereksinimlerinin belirlenmesi sürecinde açıklanan gereksinimler bir Excel formatında toplandı. Toplanan gereksinimler programa yüklenerek ilgili yazılım birimlerine atandı.Bu gereksinimler Şekil 4.12'de görüldüğü üzere MATLAB ortamına aktarıldı. Böylelikle model seviyesinde yapılan bir işlemin dayandığı gereksinim kontrol edilmek istendiğinde veya gereksinimler incelenirken, yapılan işlemin kod seviyesinde nasıl uygulandığı incelenmek istediğinde kolayca ulaşılabilir ve raporlanabilir.



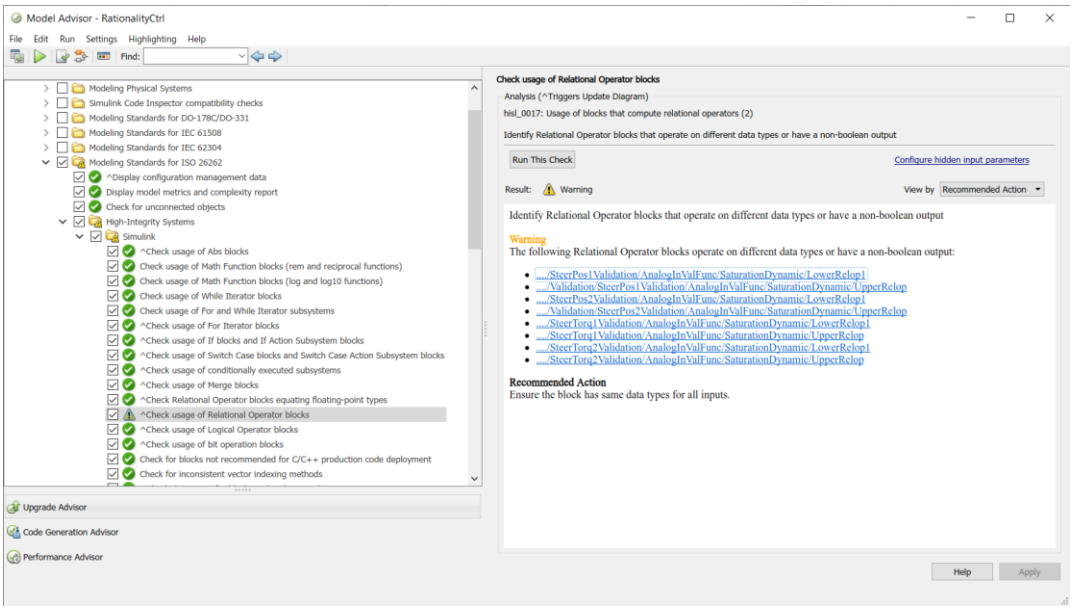
Şekil 4.12 : Gereksinim yönetimi aracı arayüzü.



Şekil 4.13 : Gereksinim - model bağlantısı gösterimi.

Şekil 4.13'te gereksinimlerin model ile ilişkilendirilmesi görsel olarak paylaşıldı.

Yazılım geliştirme sürecinde farklı yazılım geliştiricilerin kod kalitelerinde standardı yakalayabilmek için ve proje bütününde tutarlılığı ve okunabilirliği sağlayabilmek adına şartnameler belirlenir. Model tabanlı yazılım geliştirme sürecinde kullanılan programın standart şartnameleri kullanılmalı veya kurum içi standartlar oluşturulmalıdır. Firmaların oluşturduğu modelleme şartnameleri ve süreçleri fikri mülkiyet haklarından dolayı paylaşılamamaktadır. Bunun yerine kullanılan MATLAB Simulink programı tarafından müşterilerine kendi şartnamelerinin temellerini oluşturması adına paylaşılan standart kontroller Şekil 4.14'te paylaşıldı. Örnek olarak RationalityCtrl modelinde sinyalin limitleme işlemi yapılırken uyarı alındı. Alınan uyarı Şekil 4.14'te görülmekte ve Şekil 4.15'te model seviyesinde karşılık geldiği yer verilmektedir.



Şekil 4.14 : Modelleme standardı kontrolleri.

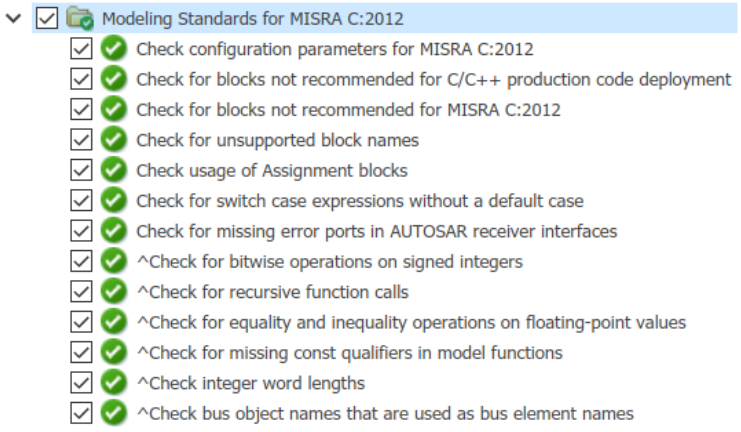
Örnekteki uyarı incelendiğinde birbirinden farklı veri tipleri arasında ilişki operatörü (relational operator) kullanımından kaynaklandığı görüldü. İlişki operatörleri iki değer arasında karşılaştırma işlemlerinde kullanılan bir operatördür. Farklı veri tipleri arasında yapılan karşılaştırmalar kullanılan derleyici seviyesinde farklı tepkilere neden olabilmektedir. Geliştirilen uygulamanın her işlemci ve derleyicide farklılığa sebebiyet vermemesi için güçlü veri tipi (strong data typing) sağlanmalıdır. Aşağıdaki şekilde görüldüğü üzere single veri tipine sahip bir veri double veri tipi ile sınırlandırılırken ilişki operatörü kullanımına sebebiyet vermektedir.

Şekil 4.15 : Güçlü veri tipi kontrolü uyarısıın model seviyesinde gösterimi.

Problemin çözümü için SteerPosSens1Value sinyali ve STERRPOS_LIM limit değerleri aynı veri tipinde tanımlanmıştır ve yapılan kontroller ile doğrulanmıştır. Şekil 4.16'da paylaşılan görselde kontrollerin tamamının başarılı bir şekilde geçtiği görülmektedir.

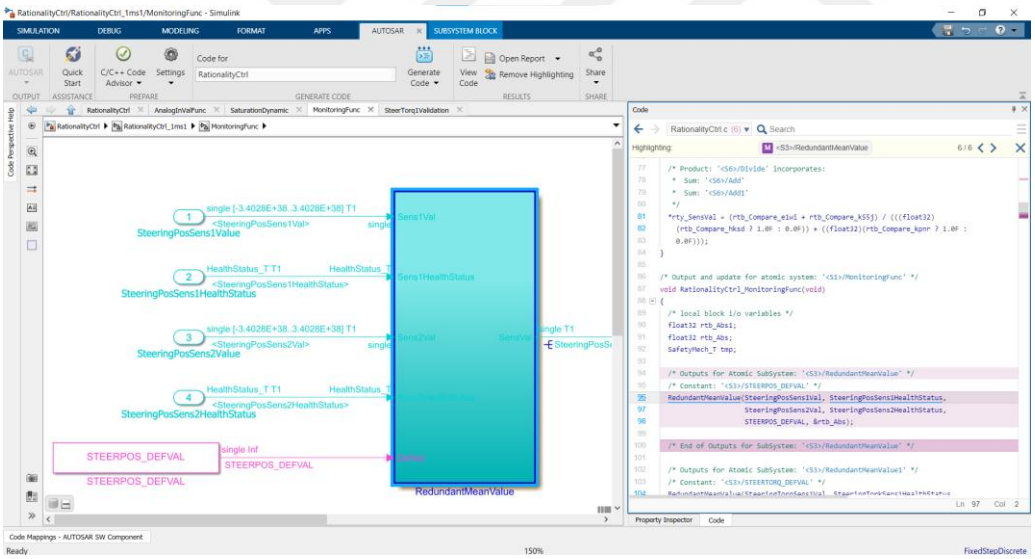
Şekil 4.16 : Simulink standart modelleme şartnamesi.

Modelleme seviyesindeki kontrollere ek olarak, otomatik kod üretimi sürecinde standardın sağlanması için de bir sıra kontrol gerekmektedir. Kullanılan program otomatik kod üretirken kullanılan dil için seçilen standarda uygun bir şekilde çıktı verebilmesi bu kontrollere bağlıdır. Şekil 4.17'de model seviyesinden MISRA C standardına uygun C kodu üretilebilmesi için model seviyesinde uyulması gereken kurallar paylaşıldı. Şekilde görüldüğü üzere RationalityCtrl fonksiyonu için yapılan kontroller başarılı şekilde geçildi.



Şekil 4.17 : Simulink modelleme seviyesindeki MISRA C şartnamesi.

Gerekli kontroller sağlandıktan sonra modelden otomatik kod üretilir. Şekil 4.18'de üretilen kod ile model arasındaki takip edilebilirlik süreci gösterilmektedir. Model ve kod arasındaki ilişkiler gözlemlenerek üretilen kodun istenilen gereklilikleri karşılayıp karşılamayacağı denetlenir. Otomatik kod üretimi süreci kullanılan program özelinde uzmanlık gerektirir. Gelişen teknoloji ile optimizasyon ayarları ve konfigürasyonlara bağlı olarak kendi iç dinamikleri kavranmalıdır. Yetkinliğe ulaşıldıktan sonra üretilen kod dönüşüm süreci manipüle edilerek istenilen yapıda kod üretilebilmektedir. Otomatik kod üretiminin avantajlarından birisi ise verilen argümanlara bağlı olarak tutarlı kod üretimidir. Bir diğer önemli nokta ise kullanılan donanımın sahip olduğu işlemci ve mimariye yönelik kod üretimi süreçlerini desteklemeleridir. Bir yazılım geliştiricinin yıllar boyunca kazanacağı deneyim, geliştiricinin bir işlemci mimarisine adaptasyonunu sağlayacaktır. Ancak farklı projelerde ihtiyaçlara göre farklı mimarilere göre optimize edilmiş kod yazabilmek ve bunu tutarlı bir şekilde devam ettirebilmek oldukça zordur. Bu nedenle uzun vadede otomatik kod üretimi sürecine uyum sağlamak faydalıdır.



Şekil 4.18 : Otomatik üretilen kod ve model ilişkisi.

4.5 Yazılım Birim Doğrulama

Yazılım birim doğrulama sürecinde amaç, yazılım birimlerinin gereksinimlerle tutarlı bir şekilde tasarlandığını bir sıra testten geçirerek doğrulamaktır. Yazılım birim testlerinde hangi doğrulama yöntemlerinin kullanılacağı test doğrulama planında belirlenir. Doğrulama yöntemleri daha önceki bölümlerde detaylı şekilde paylaşıldı. Her yazılım birim doğrulama tekniği kendi içinde farklı odaklara sahiptir. Bu nedenle

mümkün olduğunca kodu farklı testlerden geçirerek ileride tespit edilecek hataların önüne geçmektir. Daha yüksek seviyelerdeki testlerde bu hataların tespit edilmesi durumunda kod güncellemeleri ve yeniden tüm test süreçlerini yeniden uygulamak gerekmektedir.

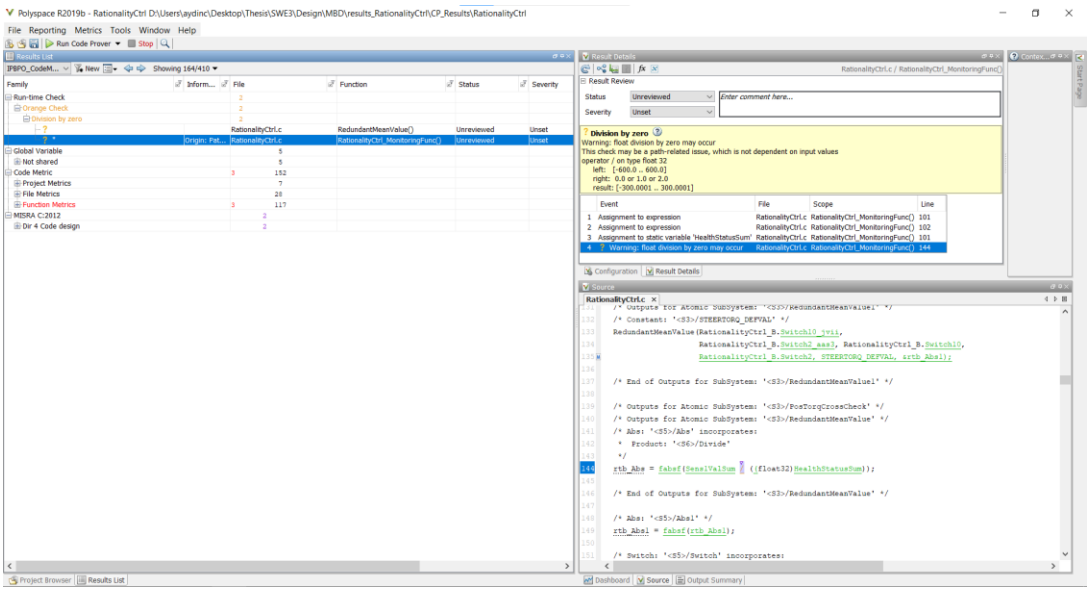
Yapılan testlerde kodun gereksinimler tarafından belirlenen işlevi yerine getirdiğinin kanıtı olacak doğrulama kriterleri belirlenmelidir. Yapılan testler boyunca kod hiçbir doğrulama kriteri ihlal edilmeden çalışması durumunda doğrulama sürecinin başarılı bir şekilde tamamlandığı kabul edilerek raporlar oluşturulur.

Test planı geliştirilirken ürün geliştirmenin bir süreç olduğu ve testler sonucunda bir hata bulunması sonucunda döngülerin tekrar edeceği ve benzer testlerin yeniden yapılması gerekeceği dikkate alınmalıdır. Hata düzeltme, performans iyileştirme gibi kodda yapılan değişikliklerden sonra kodun davranışında farklılıklar meydana gelebileceği kabul edilerek regresyon testleri yapılır. Regresyon testlerinin amacı fonksiyonun kritik alanlarının yapılan değişiklikler sonrasında da aynı çalıştığının doğrulanması için yapılır. Detaylı test süreçlerine başlamadan önce regresyon testlerinde büyük bir hatanın olup olmadığı doğrulandıktan sonra diğer süreçleri başlatmak zaman kazanılmasını sağlar.

Uygulama bölümünde, kod işlemci üzerinde çalıştırılmadan denetlemeler ve gözden geçirmelerle yapılan testler yerine uygulamalı örneklere odaklanılacaktır.

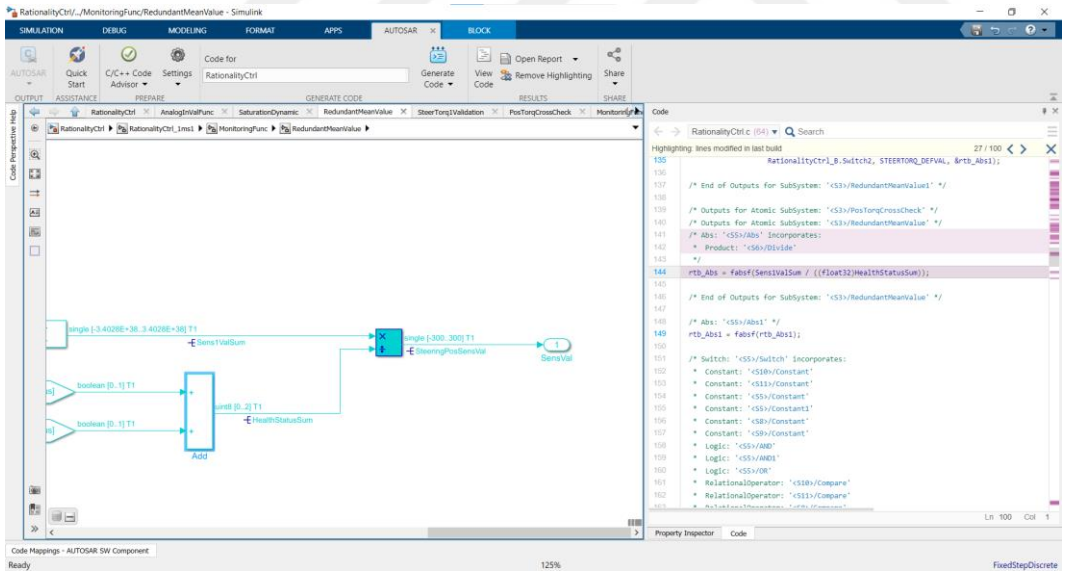
İlk olarak, statik kod analizi örneği paylaşılacaktır. Statik kod analizi, program yürütülerek yapılan dinamik analizlerin aksine, herhangi bir yürütme işlemi yapılmadan kaynak kodunda bulunan hataların ve standartlara uygunluğunu test etmek için bir bilgisayar programı tarafından yürütülen testtir.

Şekil 4.19’da statik kod analizinde bulunan sıfıra bölünme hatası tespit edildi. IEEE 754 floating point aritmetik standardına göre float veri tipinde yapılan sıfıra bölünme işlemleri inf veya -inf sonuçlarını vermektedir. Ancak bu durum uygulamada beklenmedik hatalara sebebiyet verebilmektedir. MISRA C Directive 4.1 “Run-time failures shall be minimized” maddesi kapsamında sıfıra bölünme işleminden kaçınılması gerektiği belirtilmektedir.



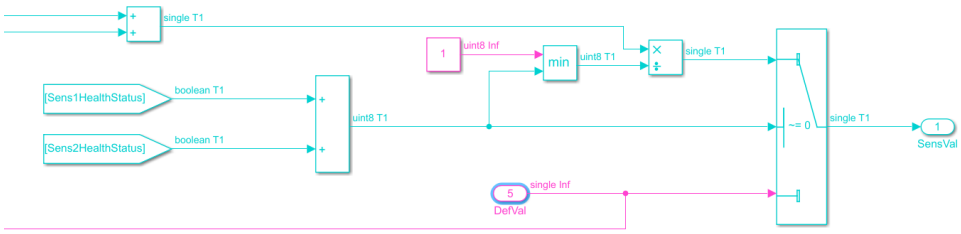
Şekil 4.19 : Statik kod analizi hata örneği.

Kullanılan programın iki yönlü takip edilebilirliği sağlaması bu gibi durumlarda oldukça faydalıdır. İlgili kod satırına gidilerek model seviyesindeki hata kaynağına ulaşılabilmektedir. Şekil 4.20'de koddaki hatanın model seviyesinde ilişkili olduğu işlem gösterilmektedir.



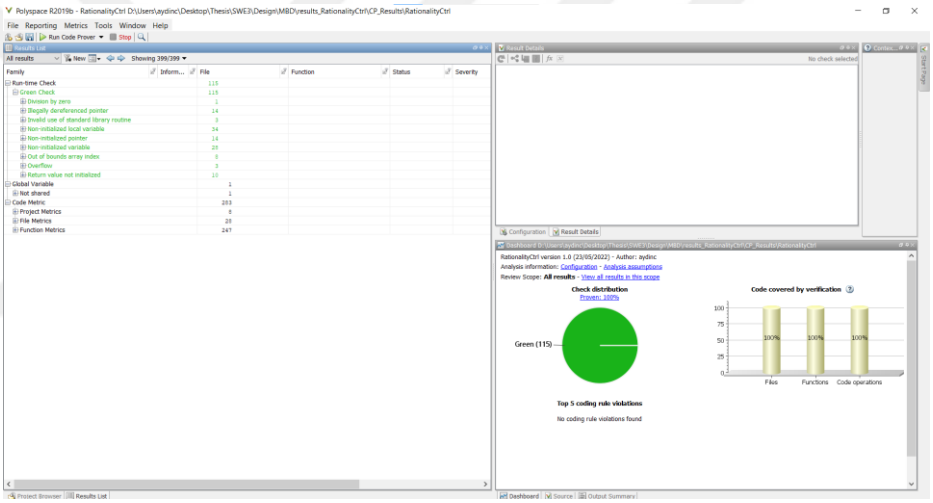
Şekil 4.20 : Statik kod analizi hatasının model seviyesinde tespiti.

Yapılan kontroller sonucunda Sens1ValSum sinyalinin HealthStatusSum sinyaline bölünmesi durumundan kaynaklandığı fark edildi. Bunu önlemek için farklı savunma amaçlı algoritmalar (defensive algorithms) kullanılabilir. Örnek bir yaklaşım Şekil 4.21'de paylaşıldı.



Şekil 4.21 : Sıfıra bölünme hatası için model seviyesinde çözüm önerisi.

Bölüm işleminde kullanılan, bölen sinyal sınırlandırılarak sıfıra bölünme durumu ortadan kaldırılmış, bölenin sıfır olması durumunda da varsayılan değer koşulla sağlanmıştır. Böylelikle sıfıra bölünmeden kaynaklanacak hataların önüne geçilmiştir. Yapılan değişiklikler sonucu yürütülen statik kod analizi sonucu Şekil 4.22'de görülmektedir.

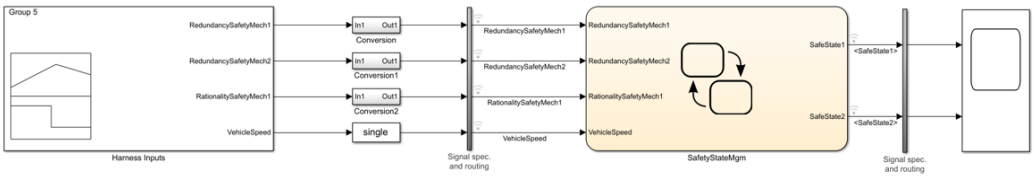


Şekil 4.22 : Başarılı tamamlanan statik kod analizi örneği.

Statik kod analizinde çalışma-zamanı kontrollerine (run-time check), kodlama standartları ve proje özelinde belirlenen kod metriklerine uyumluluk da kontrol edilebilmektedir.

Dinamik davranışların test edilmesi sürecine de bir örnek vermek adına StateMachine fonksiyonu içinde kullanılan durum makinesi yazılım birimi için Değiştirilmiş Koşul/Karar Kapsamı (MC/DC – Modified Condition/Desicion Coverage) testi paylaşıldı.

Testin yürütülebilmesi için bir test koşumunun yapılacağı ortamı (test harness) oluşturulmalıdır. Test senaryoları üretmenin ve sonuçları görüntülemenin kullanılan programa göre farklı yöntemler kullanılabilir. Örnek bir test ortamı Şekil 4.23'te paylaşıldı.

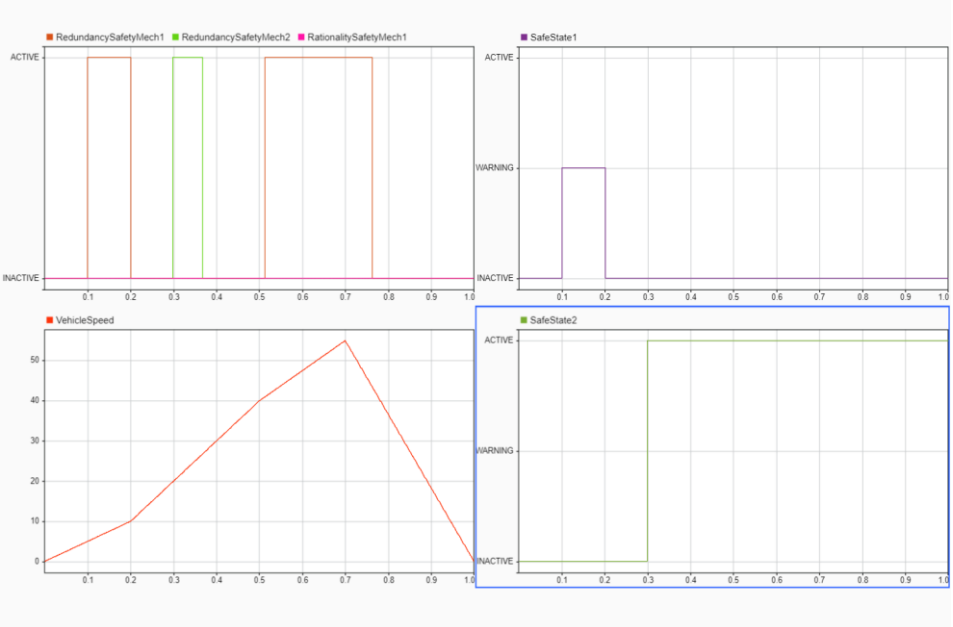


Şekil 4.23 : Test koşum ortamı örneği.

MC/DC, diğer koşulları sabit tutarken her koşulun tüm olası durumların test edilmesini gerektiren bir kapsama testidir. Ayrıca bireysel bir koşuldaki değişikliğin sonucu değiştirdiği gösterilmelidir. MC/DC gerekliliklerini yerine getirmek aşağıdaki koşullar sağlanmalıdır;

- Programın her çıkış ve giriş noktası, test durumlarından en az birinde yürütülmeli
- Her karar, olası tüm sonuçlar için test edilmeli
- Bir karardaki her koşul, tüm olası durumlar için test edilmeli
- Soncu bağımsız olarak değiştirmek için her bir koşul gösterilmeli

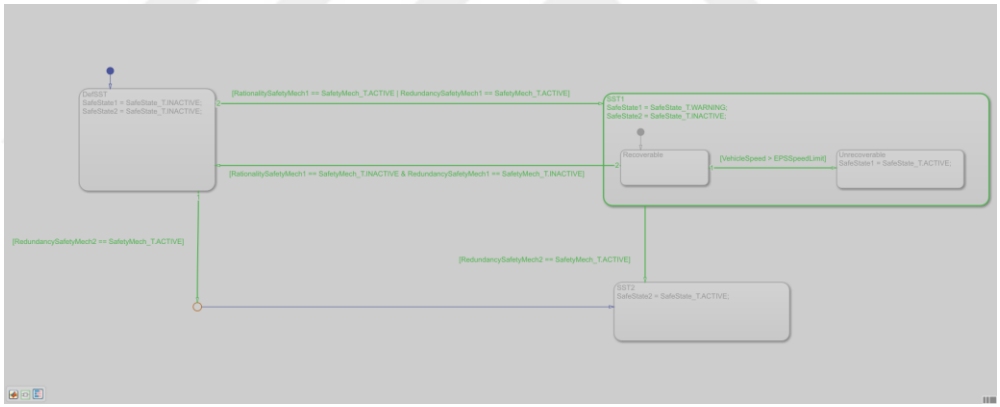
Şekil 4.24'te oluşturulan bir test senaryosunun sinyalleri ve bu sinyallere karşı sistem cevapları görülmektedir. Bu örnekte, görüntüleme amacıyla oluşturulan güvenlik mekanizma sinyalleri ve araç hızı bilgisine bağlı olarak güvenli durum geçişleri incelenmektedir.



Şekil 4.24 : MC/DC testi giriş ve çıkış sinyalleri.

Yürütülen test senaryosu sonucunda durumlar arasında koşullara bağlı olarak geçişler sağlandı ve buna bağlı durumlar incelenebilir. Sağlanmayan koşullar için farklı test sonuçları oluşturularak tam kapsama sağlanabilir. Kapsama testinde tam kapsamaya oluşmak fonksiyonun doğru çalıştığı anlamına gelmez, kapsama sonuçlarına göre sonuçlar incelenmeli veya doğrulama kriterlerine göre değerlendirilmelidir. MC/DC testi ASIL D seviyesi yazılım birimleri için şiddetle tavsiye edilmektedir. Kapsama testi sonuçları Şekil 4.25 ve Şekil 4.26'da paylaşıldı.

Şekil 4.25 : Geliştirme sürecinde MC/DC kapsama testi.



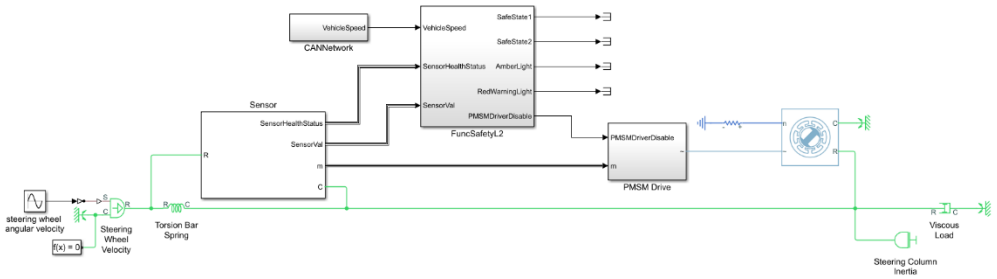
Şekil 4.26 : Tamamlanmış MC/DC kapsama testi.

4.6 Simülasyon

Simülasyon modellemek, gerçek dünyadaki sorunları dijital ortama aktararak güvenli ve verimli bir şekilde çözmemizi sağlar. Kolayca doğrulama yapmak, birimler arası iletişim ve anlaşılabilirliği arttıran önemli bir analiz yöntemidir. Farklı disiplinler arasında simülasyon modellemek, karmaşık sistemler hakkında bilgi sahibi olunmasını sağlayarak çözüm sürecini iyileştirir.

Yazılım birimi tarafından geliştirilen fonksiyonların sistem seviyesine aktarılmasında maliyetli düzenekler kurulmadan önce başvurulması gereken bir yöntemdir. Sistemsel bir hatayı erken fark etmek, kurulan düzeneğin modifikasyonundan kaynaklanacak

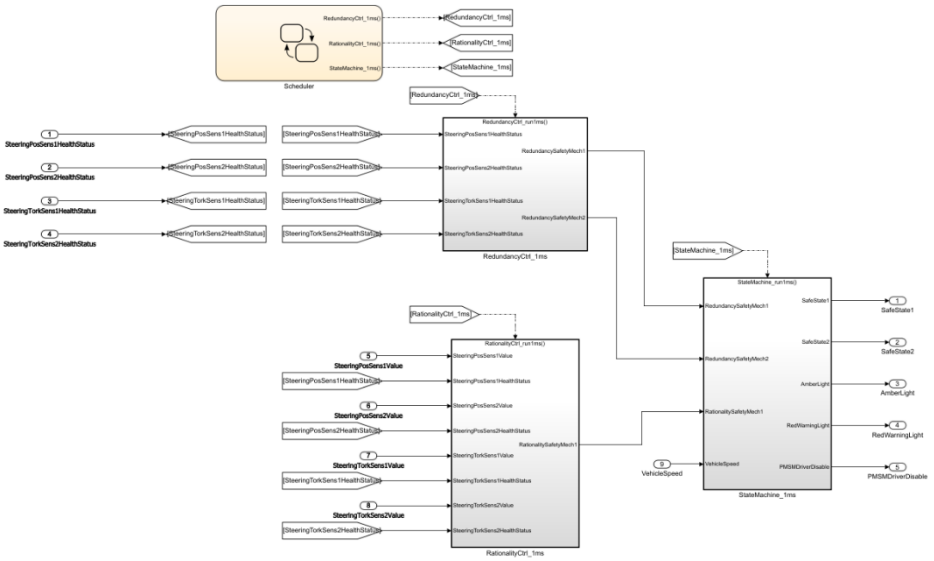
maliyetleri düşürür. Bu nedenle geliştirilen güvenlik fonksiyonlarını denemek amacıyla simülasyon modeli oluşturuldu. Oluşturulan modelin üst seviyeden alınmış bir görseli Şekil 4.27'de paylaşıldı.



Şekil 4.27 : EPS simülasyon modeli.

Kurulan simülasyon modelinde, MATLAB/Simulink Simscape aracı kullanılarak fiziksel sistemler modellenmiştir. Sistemde fiziksel olarak direksiyon mili, burulma mili gibi direksiyon sistemini elemanları ve Kalıcı Mıknatıslı Senkron Motor (PMSM – Permanent Magnet Synchronous Motor) kullanılmıştır. Burulma mili üzerinden alınan pozisyon ve tork değerlerine bağlı olarak PMSM sürücü direksiyon miline uygulanan kuvveti yükseltmektedir.

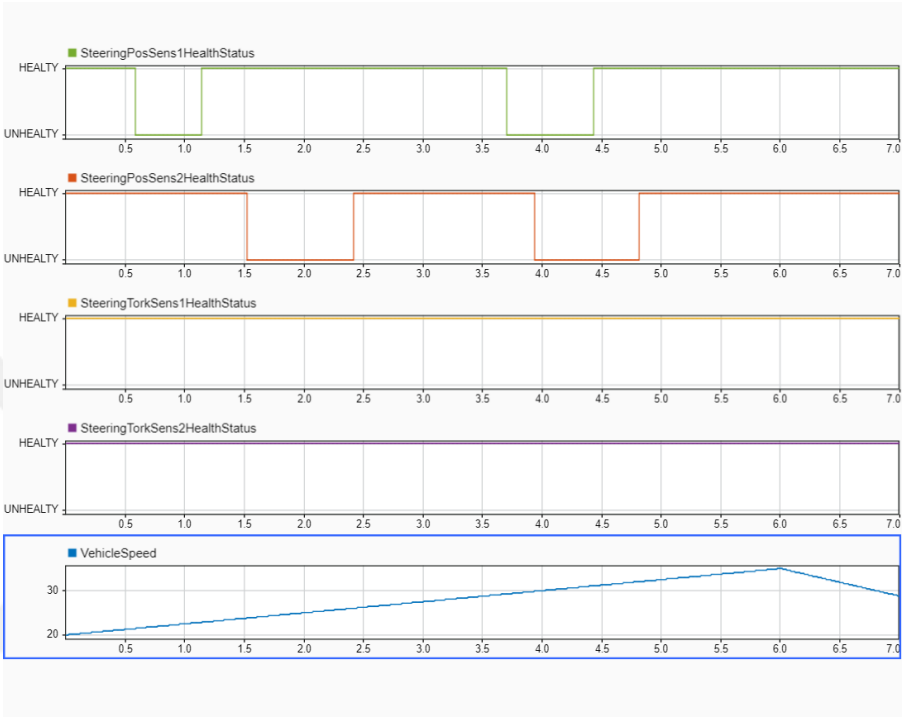
Bu simülasyonun yapılmasının amacı fonksiyonel güvenlik fonksiyonlarının görevini beklenilen şekilde yerine getirebildiğini doğrulamaktır. 3 katmanlı denetleme mekanizmasında bahsedildiği gibi 2.seviye fonksiyonlar incelenecektir. RTE katmanını modelleyebilmek adına fonksiyonların çağrılarını (function call) üreten bir durum makinesi kullanıldı. Oluşturulan model Şekil 4.28'de paylaşıldı.



Şekil 4.28 : 2. seviye görüntüleme fonksiyonlarının simülasyon modeli.

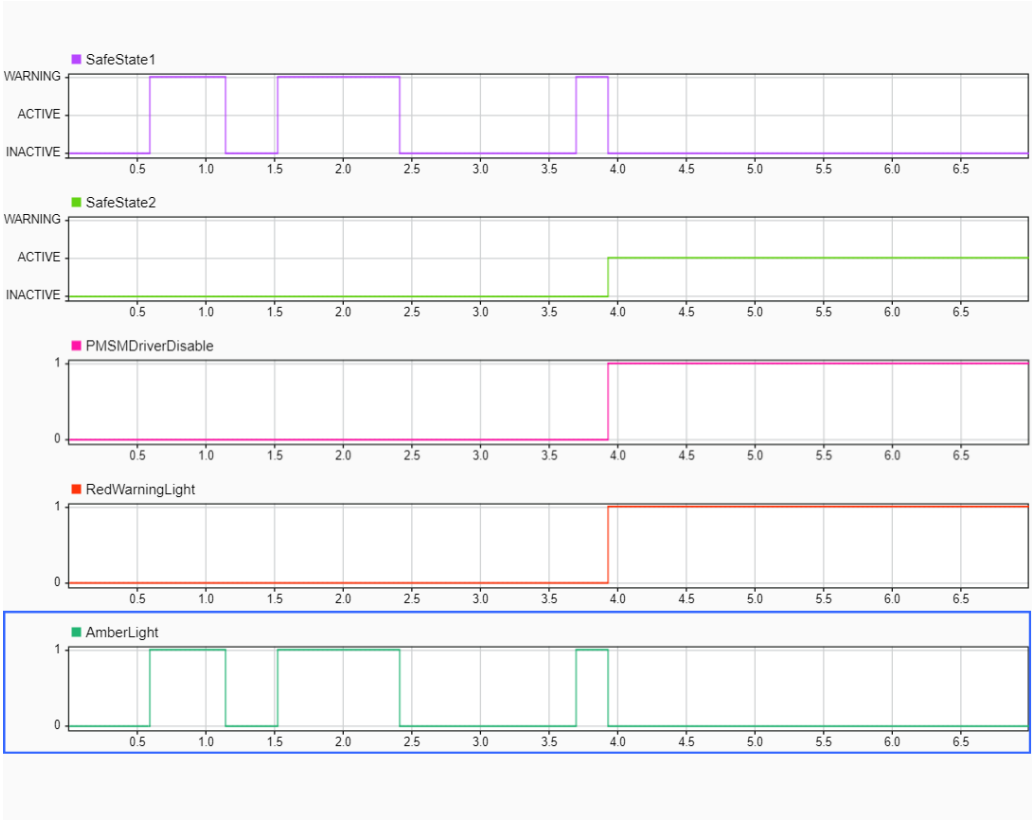
Sistem farklı test senaryolarında çalıştırıldı ve doğrulandı. Süreci açıklamak adına örnek bir test senaryosu ve sistem tepkileri paylaşıldı.

Örnek test senaryosunda, fazlalıklı pozisyon sensörü kullanımında sensörlerin farklı zamanlarda donanımsal arızalara maruz kalması ve iki sensörde birden donanımsal arıza olması durumları incelendi. Şekil 4.29'daki görsel ile sistemde kullanılan sensörlerin durum bilgileri paylaşıldı.



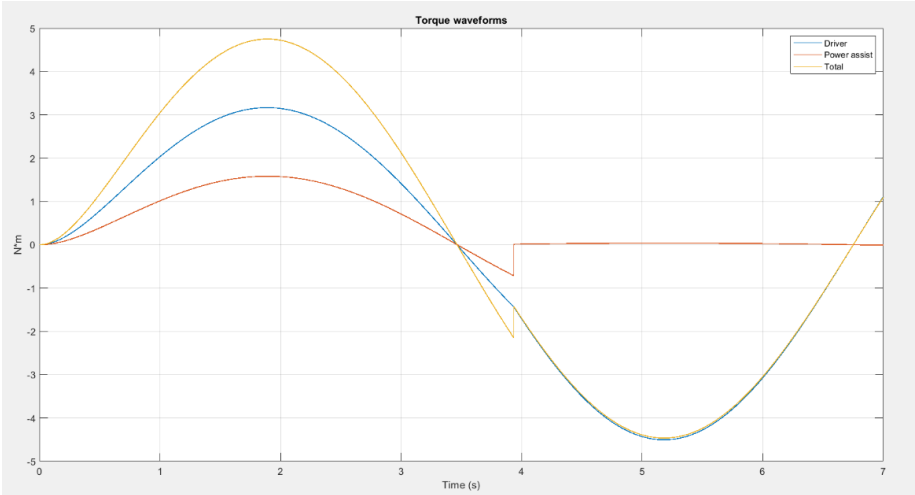
Şekil 4.29 : Simülasyon test senaryosu sinyalleri.

Sensör durum bilgilerine bağlı olarak inceleme yapıldığında beklenen sonuçlar, pozisyon sensörlerinin sadece birisinde arıza olduğu durumlarda sarı ikaz ikonu için sinyal üretilmesi, iki sensörün arıza yaptığı durumlarda sistemin kalıcı hata üreterek PMSM sürücüsünü devredışı bırakmasıdır. Bir sensörün arıza verdiği durumda araç hızı limit değeri aşmadığından sadece ikaz verecek PMSM sürücüsü devredışı kalmayacaktır. Sistem tepkilerinden beklenilen sonuçların alındığı Şekil 4.30'da görülmektedir.



Şekil 4.30 : Simülasyon test sonuçları.

Sistem cevaplarının modellenen sisteme etkilerini incelemek için sürücünün direksiyon sistemine uyguladığı tork, PMSM motorunun sağladığı güç yardımı ve direksiyon sistemine uygulanan toplam tork değerleri ile Şekil 4.31'dekigrafik oluşturuldu. PMSM sürücüsü devre dışı bırakma sinyali oluştuktan sonra PMSM motorundan elde edilen gelen dönme momentin kesildiği ve sadece sürücünün uyguladığı torkun sistemde kaldığı görülmektedir.



Şekil 4.31 : EPS tork sinyalleri

5. SONUÇ VE ÖNERİLER

Bu çalışmada elektronik sistemlerin otomotiv endüstrisinde artan kullanımıyla birlikte gün geçtikçe önemi artan elektronik sistemlerin güvenliği üzerinde durarak ISO 26262 Kara Araçları Fonksiyonel Güvenlik standardının önemine vurgu yapıldı. Fonksiyonel güvenlik standardının, otomotiv sektörünün büyük paydaşlarının ortak çalışması ile neredeyse sektörel standartlar haline gelmiş AUTOSAR, E-Gas gibi mimarilere etkileri incelendi. Bu çalışma otomotiv elektroniği ve güvenliği hakkında bilgi sahibi olmak isteyen araştırmacılar ve sektöre yeni adım atan kişiler için kullanılan standartlar, süreçler ve teknik terimler hakkında bilgi vermeyi amaçlamaktadır.

İlk olarak, uygulamada kullanılacak hedef bir elektronik kontrol sistemi seçildi. Uygulama sürecinde konu takibini kolaylaştırmak amacıyla seçilen sistemin temel çalışma prensipleri hakkında bilgiler paylaşıldı.

Daha sonrasında yönetimsel konuları tezin kapsamı dışında bırakarak, Amerikan Ulusal Otoyol Trafik Güvenliği İdaresi (NHTSA - National Highway Traffic Safety Administration) taradından genel EPS sistemleri için yayınlanan fonksiyonel güvenlik değerlendirmesi referans alınarak, tehlike analizi ve risk değerlendirmesi, fonksiyonel güvenlik konsepti, teknik güvenlik konsepti gibi kritik konular açıklandı.

Bunun üstüne, sistem ve yazılım geliştirme süreçlerini açıklamak için, sektörde oldukça popüler olan V-Döngüsü hakkında bilgiler verildi. Verilen bilgiler doğrultusunda uygulama örnekleri gerçekleştirildi.

Tüm bunlara ek olarak model tabanlı yazılım geliştirme ve otomaik kod dönüşümü gibi güncel teknolojiler hakkında bilgiler sunuldu.

Son olarak, gelecek çalışmalara değinilecek olursa, özellikle insan kontrolünü tamamen ortadan kaldıracak otonom sürüş sistemlerindeki gelişmeler fonksiyonel güvenlik standartlarının gelişmesini destekleyeceği öngörülmektedir. Kara araçları fonksiyonel güvenlik standardının ilk versiyonunun yayınlama yılı 2011 olduğundan, yayınlanan tezler oldukça güncel tarihli ve bu kadar kapsamlı bir konu için nicelik olarak yetersizdir. Gereksinim yönetimi, uygulama seviyesinde mimari yaklaşımlar kendi başına tez konusu olabilecek derinliktedir. Standardın hala gelişmekte ve sektör ihtiyacının yüksek olduğu düşünüldüğünde yapılacak akademik araştırmalar hem

standardın geliştirilmesine hem de konudaki uzman açığının giderilmesini sağlayacaktır.

KAYNAKLAR

[1] **Broy, M.** (2006). Challenges in automotive software engineering, in: *Proceedings of the 28th International Conference on Software Engineering*. pp. 33-42

[2] **Charette, R. N.** (2009). This Car Runs on Code.

https://spectrum.ieee.org/this-car-runs-on-code

[3] **Shorky, M. ve Hinchey, M.** (2009). Model-Based Verification of Embedded Software. IEEE Computer. 42. 53-59. 10.1109/MC.2009.125.

[4] **Bringmann, E. ve Kramer, A.** (2008). Model-Based Testing of Automotive Systems. 485-493. 10.1109/ICST.2008.45.

[5] **Grand View Research.** (2019). Automotive Electronics Market Size, Share & Trends Analysis Report By Component (Electronic Control Unit, Sensors, Current Carrying Devices), By Application, By Sales Channel, By Region, And Segment Forecasts, 2021 - 2028 (Rapor No: 978-1-68038-357-7)

[6] **Broy, M., Kristan, S., Krcmar, H. & Schätz, Bernhard** (2011). What is the Benefit of a Model-Based Design of Embedded Software Systems in the Car Industry?. doi: 10.4018/978-1-4666-4301-7.ch017.

[7] **C. Bunse, H. Gross and C. Peper,** (2007). "Applying a Model-based Approach for Embedded System Development," 33rd EUROMICRO Conference on Software Engineering and Advanced Applications 2007, pp. 121-128, doi: 10.1109/EUROMICRO.2007.18.

[8] **ISO** (2018). ISO 26262 - Road Vehicles – Functional

[9] **AUTOSAR** (2022). Partnership History of AUTOSAR. https://www.autosar.org/about/history/

[10] **Grosmann, M., Hirz, M. & Fabian, J.** (2016). Efficient application of multi-core processors as substitute of the E-Gas (Etc) monitoring concept. 913-918. doi: 10.1109/SAI.2016.7556089

[11] **Nakayama, T. & Suda, E.** (1994). The Present And Future Of Electric Power Steering *International Journal of Vehicle Design, Volume 15, Issue 3/4/5, 1994, p. 243-54.*

[12] **Becker, C., Nasser, A., Attioui, F., Arthur, D., Moy, A., & Brewer, J.** (2018). Functional safety assessment of a generic electric power steering system with active steering and four-wheel steering features (Report No. DOT HS 812 575). Washington, DC: National Highway Traffic Safety Administration.

[13] **EGAS Workgroup** (2015). "Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units." Version 6.0. *https://nanopdf.com/download/standardized-e-gas-monitoring-concept-for-gasoline-and_pdf*

[14] **Infenion** (2015). Sensor Solutions for Automotive, Industrial and Consumer Applications.

https://www.infineon.com/dgdl/Infineon-Sensor_Solutions_for_Automotive_Industrial_and+Customer_Appl_BR-2015.pdf?fileId=5546d4614937379a01495212845c039f

[15] **Peter K.** (2012). The Automotive Standard ISO 26262, the Innovative Driver for Enhanced Safety Assessment & Technology for Motor Cars, Procedia Engineering, Volume 45, Pages 2-10, ISSN 1877-7058

[16] **Piper, T**. (2015). Assessing and Enhancing Functional Safety Mechanisms for Safety-Critical Software Systems. (Doktora tezi). Darmstadt, Technische Universität.

[17] **Layal, V.** (2016). Analysis and Specification of an AUTOSAR based ECU in compliance withISO 26262 Functional Safety Standard.

[18] **Stolte, T., Bagschik, G. & Maurer, M.** (2016). Safety goals and functional safety requirements for actuation systems of automated vehicles. IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 2191-2198, doi: 10.1109/ITSC.2016.7795910.

[19] **Kristian, B. & Heisel, M. & Frese, T. & Hatebur, D.** (2013). A structured and model-based hazard analysis and risk assessment method for automotive systems. 2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013. 238-247. 10.1109/ISSRE.2013.6698923.

[20] **Menzel, T., Bagschik, G. & Maurer, M**. (2018). Scenarios for Development, Test and Validation of Automated Vehicles. 10.1109/IVS.2018.8500406.

[21] **Knopf, M. D.** (2019). Comprehensive concept-phase system safety analysis for hybrid-electric vehicles utilizing automated driving functions. Department of Mechanical Engineering, Colorado State University.

[22] **Kirovskii, O., & Gorelov, V.** (2019). Driver assistance systems: analysis, tests and the safety case. ISO 26262 and ISO PAS 21448. IOP Conference Series: Materials Science and Engineering.

[23] **Petrescu, L. & Cazacu, E. & Petrescu, M**. (2019). Failure Mode and Effect Analysis in Automotive Industry: A Case Study. The Scientific Bulletin of Electrical Engineering Faculty. 19. 10-15. 10.1515/sbeef-2019-0014.

[24] **Chaari, M.** (2015). Formalization and Model-Driven Support of Functional Safety. (Doktora tezi). Technische Universität München.

[25] **Gnaniah, R.** (2019). Functional Safety Assessment for Advanced Driver Assistance System. (Yüksek Lisans Tezi). Politecnico di Torino.

[26] **Hoxha, Y. & Tarıq, M.** (2020). Development toolchain for Vehicle Electrification. (Yüksek Lisans Tezi). Politecnico di Torino.

[27] **Tikar, S.S. & Ansari, A.**, "Compliance of ISO 26262 Safety Standard for Electric Power Steering System," SAE Technical Paper 2021-26-0025, 2021, doi:10.4271/2021-26-0025.

[28] **Paulsen, B., Henn, S., Männel, G. & Rostalski, P. (2021).** Functional Safety Concept EGAS for Medical Devices. Current Directions in Biomedical Engineering. 7. 739-742. 10.1515/cdbme-2021-2189.

[29] **Nagabhushan, S. & Nadibail, A.** (2019). Design of monitoring concepts for motion control of autonomous heavy vehicles. (Yüksek Lisans Tezi). Chalmers University of Technology.

[30] **Selic, B.** (2012). What will it take? A view on adoption of model-based methods in practice. Software & Systems Modeling. 11. 10.1007/s10270-012-0261-0.

[31] **Conrad, M.** (2012). Artifact-Centric Compliance Demonstration for ISO 26262 Projects Using Model-Based Design. GI-Jahrestagung.

[32] **Holtmann, J., Meyer, J. & Meyer, M.** (2011). A Seamless Model-Based Development Process for Automotive Systems. 79-88.

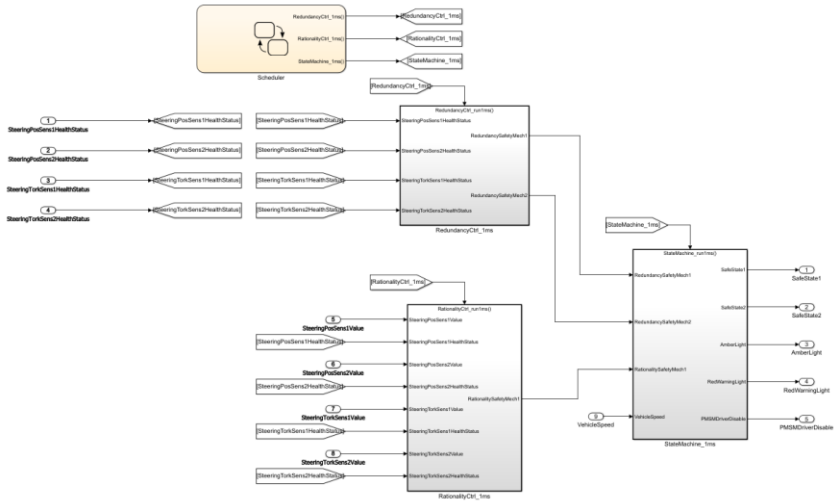
[33] **Liliegard, M. & Nilsson, V.** (2014). Model-Based Testing with Simulink Design Verifier. (Yüksek Lisans Tezi). Chalmers University of Technology.

[34] **Hiremath, R. & Isha, T.B.** (2019). Modelling and simulation of electric power steering system using permanent magnet synchronous motor. IOP Conference Series: Materials Science and Engineering. 561. 012124. 10.1088/1757-899X/561/1/012124.

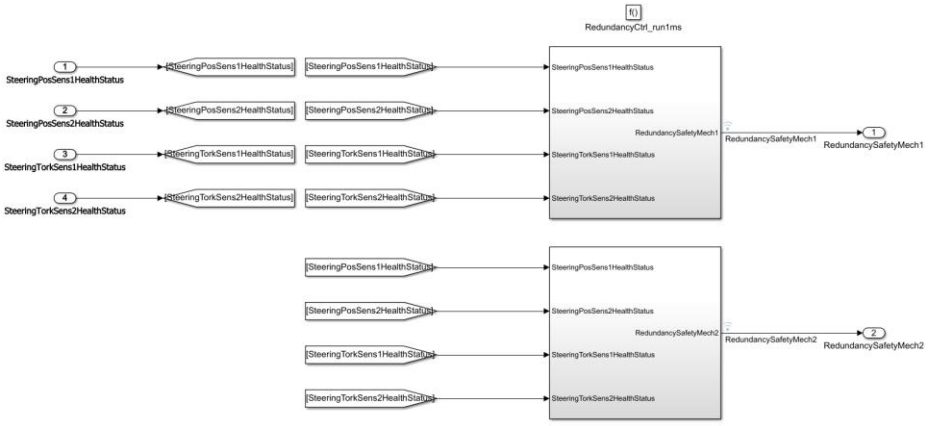
EKLER

EK A: Kontrol Modelleri

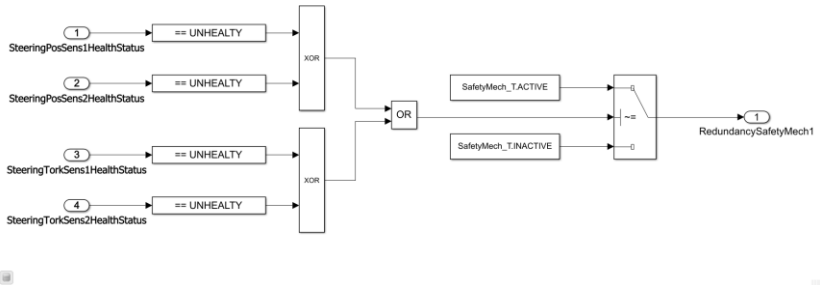
EK A



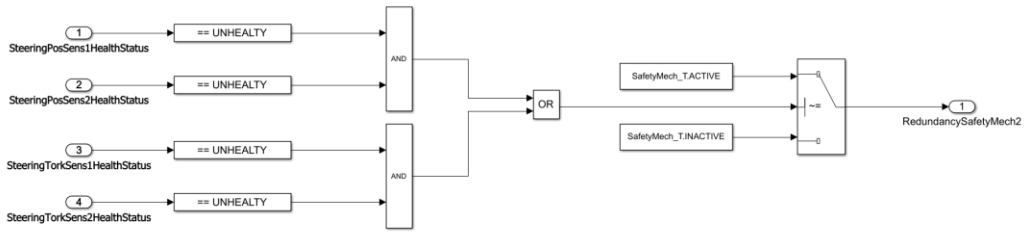
Şekil A.1 : Üst seviye kontrol modelinin ekran görüntüsü



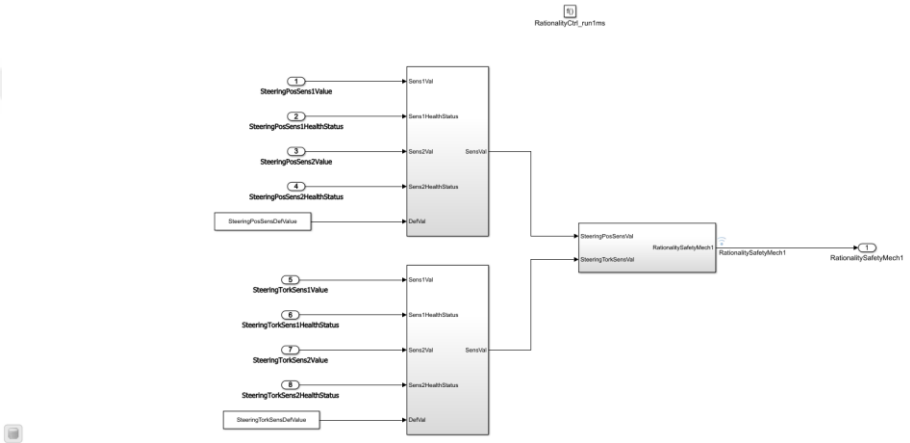
Şekil A.2 : RedundancyCtrl modeli ekran görüntüsü



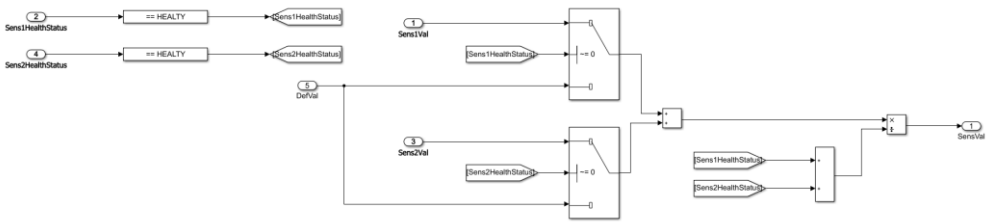
Şekil A.3 : RedundancySafetyMech1 sinyalinin hesabı



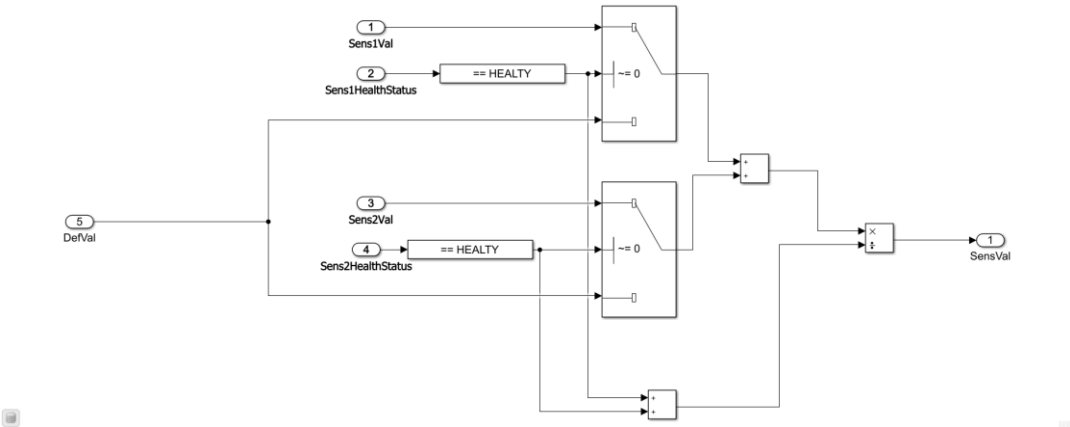
Şekil A.4 : RedundancySafetyMech2 sinyalinin hesabı



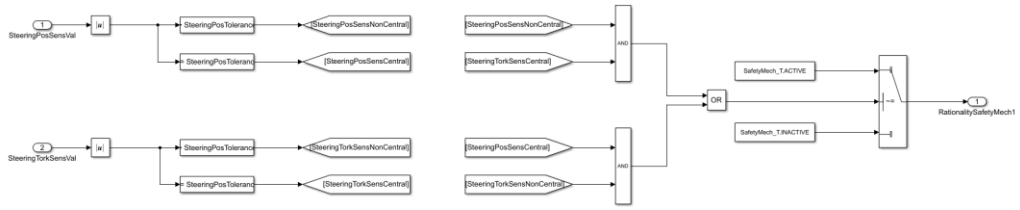
Şekil A.5 : RationalityCtrl modeli ekran görüntüsü



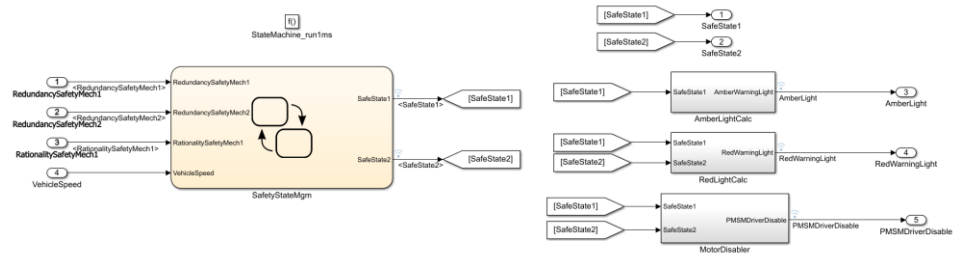
Şekil A.6 : Pozisyon sensörü sinyalinin hesabı



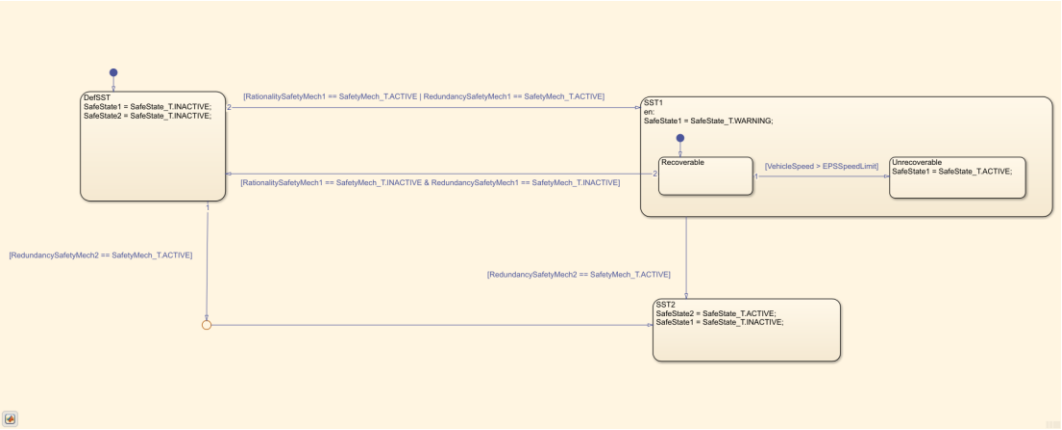
Şekil A.7 : Tork sensörü sinyalinin hesabı



Şekil A.8 : RationalitySafetyMech1 sinyalinin hesabı

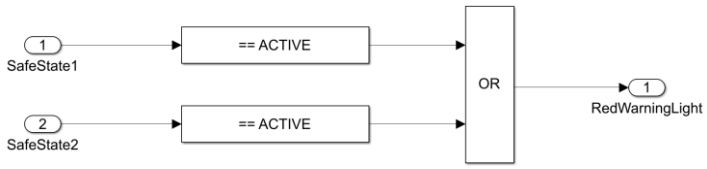


Şekil A.9 : StateMachine modeli ekran görüntüsü

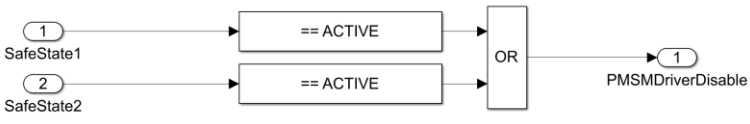


Şekil A.10 : Durum makinesi ekran görüntüsü

Şekil A.11 : Sarı ikaz ışığı hesabı

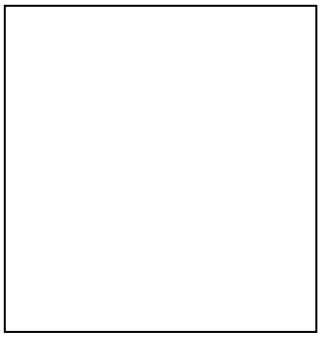


Şekil A.12 : Kırmızı ikaz ışığı hesabı



Şekil A.13 : Elektrik motoru deaktivasyon sinyali hesabı

ÖZGEÇMİŞ



Ad-Soyad : Cengiz Aydın

ÖĞRENİM DURUMU:

- **Lisans :** 2017, Yıldız Teknik Üniversitesi, Makine Fakülte, Makine Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2019-2021 yılları arasında Tümosan Motor ve Traktör A.Ş'de Kontrol Sistemleri Mühendisi olarak çalıştı.
- 2021 yılından itibaren Eldor Elektronik ve Plastik Mlz. Ürt. ve Tic. Ltd. Şti.'de Yazılım Geliştirme Mühendisi olarak çalışmaya başladı.

DİĞER YAYINLAR, SUNUMLAR VE PATENTLER: