

ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
MÜHENDİSLİK-MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ



VERİ TABANI YÖNETİM SİSTEMLERİ
Grup A

VALU3S V&V VERİ TABANI PROJESİ

152120181071 Muhammed Talha ŞAHİN
152120181058 Mine ÇAKIR
152120181068 Hüseyin Can ERGÜN
152120181049 Efekan SARGIN
152120181012 Yusuf Kenan AKSAN

FİRMA: İNOVASYON MÜHENDİSLİK

Yetkili: Cem BAĞLUM

☎ GSM: +90 222 229 07 10

✉ E-mail: bilgi@inovasyonmuhendislik.com

İçindekiler

1. GİRİŞ	4
1.1. Proje Tanım	4
1.2. Problem Tanım	4
1.2.1. IM-FIT Problem	4
1.2.2. UPPAAL Problem	5
2. GEREKSİNİMLER	5
2.1. IM-FIT İlk Rapordan Sonra Eklenen Gereksinimler	5
2.2. IM-FIT Gereksinimler	5
2.2.1. IM-FIT Fonksiyonel Gereksinimler	7
2.3. VALU3S V&V TOOL Gereksinimler	8
2.3.1. Uppaal Gereksinimler	8
2.3.2. ROSMonitoring Gereksinimler	9
2.4. Fonksiyonel Olmayan Gereksinimler	9
3. VERİ YAPISI	10
3.1 UPPAL/ROSMonitoring Varlık-İlişki Tabloları	10
3.1.1. Sistem	10
3.1.2. Uppaal Tablosu	10
3.1.3 Query Tablosu	11
3.1.4. RosMonitoring Tablosu	11
3.1.5. Config Tablosu	11
3.1.6. MonitorsOnline ve NodesOnline Tabloları	12
3.1.7. TopicOnline ve Publishers Tablosu	13
3.1.8. MonitorsOffline Tablosu	13
3.1.9. Property Tablosu	14
3.2. IM-FIT Varlık-İlişki Tabloları	14
3.2.1 Source Code Tablosu	14
3.2.2. Snippet Tablosu	15
3.2.3. Workload Tablosu	15
3.2.4. LineWithErrors Tablosu	15
3.2.5. Plan Tablosu	16
3.2.6. Execution Tablosu	16

3.2.7. LineToUse Tablosu	16
3.2.8. Fault Tablosu	17
3.2.9. Metric Tablosu.....	17
3.2.10. State Tablosu	17
3.2.11. Rosbag Tablosu	17
3.2.12. Mutant Tablosu.....	18
3.2.12. Report Tablosu	18
3.2.13. IM-FIT, ROS Monitoring, UPPAAL	19
3.2.14. tblFIPlan_tblExecution.....	20
3.3. Final ER Diyagramı	21
4. SORGULAR.....	22
5.VTYS ARAÇ KULLANIMI	23
5.1.Error Handling.....	23
5.2. Triggers	24
6. KULLANICI ARAYÜZÜ	24
6.1. IMFIT Arayüzü	24
6.2. Uppaal/RosMonitoring Arayüzü.....	27
6.2.1.UPPAAL Arayüzü.....	27
6.2.2.RosMonitoring Arayüzü.....	29
7.DİĞER ENTEGRASYONLAR.....	30
8. ÇALIŞMA ADAM SAAT DEĞERLENDİRME	31
8.1. Adam – Saat Dağılımı	31
8.2. Görevler.....	32
9. SONUÇ VE DEĞERLENDİRME	32
Tablo 1: Adam/Saat	31
Tablo 2: Görevler.....	32
Tablo 3: Özdeğerlendirme Tablosu	33

1. GİRİŞ

Veri Tabanı ve Yönetim Sistemleri dersi kapsamında yapılacak olan proje, yazılım doğrulama ve geçerleme işlemleri gerçekleştiren Valu3es projesi kapsamında olan veri tabanı projesini sunmaktır. Bu kapsamda İnovasyon Mühendislik şirketi tarafından geliştirilen IM-FIT ve VALU3S V&V araçları veri tabanı gerçekleştirilecektir. UPPAAL, veri türleriyle genişletilmiş, zamanlı otomat ağları olarak modellenen gerçek zamanlı sistemlerin modellenmesi, doğrulanması ve doğrulanması için bütünleşmiş bir araç ortamıdır. IM-FIT aracı kapsamında yazılım geçerleme ve doğrulama sürecinde kaynak kodların mutasyon kodları elde edilerek, yazılımın mutasyon kodları tarafından test edilmesi sağlanacaktır.

1.1. Proje Tanım

Yazılım doğrulama ve geçerleme işlemleri gerçekleştiren Valu3es projesi kapsamında olan veri tabanı projesinde, İnovasyon Mühendislik şirketi tarafından geliştirilen IM-FIT ve VALU3S V&V araçlarının veri tabanını gerçekleştirilecektir.

Veri tabanı projesinden önce kullanılacak olan araçların tanımı yapılacak olursa, IM-FIT aracı ile yazılım geçerleme ve doğrulama sürecinde kaynak kodların mutasyon kodları elde edilerek, yazılımın mutasyon kodları tarafından test edilmesi sağlanacaktır. Bu süreçte kullanıcının uygulamaya kaynak kodunun girmesi ardından istenilen hata çeşitlerine bağlı olarak kaynak kodun mutasyona uğratma işlemi yapılacaktır. Bu amaç sonucunda verilerin güvenli bir şekilde saklanması ve sonuçların raporlanması aşamasında veri tabanı kullanılacaktır.

UPPAAL, veri türleriyle genişletilmiş, zamanlı otomat ağları olarak modellenen gerçek zamanlı sistemlerin modellenmesi, doğrulanması ve doğrulanması için entegre bir araç ortamıdır. .xml ve .q uzantılı dosyaların içeriği veri tabanında tutularak UPPAL verilerinin tekrardan kullanımı sağlanacaktır.

1.2. Problem Tanım

1.2.1. IM-FIT Problem

Projemizin amacı, İnovasyon Mühendislik şirketi tarafından geliştirilen IM-FIT aracının kullanıcı tarafından girilen girdilerini saklayan, istenilen özelliklere göre girdilere kolay erişim sağlayan ve kullanıcının isteklerine göre oluşan sonuçları saklayarak kullanıcıya erişimini sağlayan bir veri tabanının oluşturulmasıdır.

Kullanıcı uygulamaya kaynak kodunu girerek, seçtiği hata çeşitleri ile kodunun istediği kısımları mutasyona uğratabilir. Bu kapsamda, kullanıcının kaynak kodu, seçtiği hata çeşitleri gibi

veriler saklanmalı ve kullanıcıya verilmesi gereken; metrik sonuçları, AST diyagramı, mutantların listesi gibi çıktılar rapor halinde kullanıcıya sunulmalıdır.

1.2.2. UPPAAL Problem

Modellenen sistemlerin tanımlamaları ve modelin yapısı xml formatında dışarı aktarılmaktadır. Bunun yanında sistemler üzerine uygulanan sorgular da .q uzantılı bir dosya içerisine ya da xml içerisinde kaydedilebilmektedirler. UPPAL verilerinin saklanması ve yeniden kullanılabilmesi için, .xml ve .q uzantılı dosyaların içeriğinin veri tabanına yazılması gerekmektedir. Bahsedilen bilgilerin, ilişkisel ve en verimli şekilde, veri tabanında saklanması gerekmektedir.

2. GEREKSİNİMLER

2.1. IM-FIT İlk Rapordan Sonra Eklenen Gereksinimler

Şirketten aldığımız geri dönüşler doğrultusunda yeni eklenen gereksinimlerimiz aşağıdaki maddelerde yer verilmiştir.

1. ReGex kodlarının veri tabanı içerisinde tutulması istenmektedir bu sayede IM-FIT içerisinde oluşturulan kod parçalarının satırlar içerisinde bulunabilmesi hedeflenmiştir.
2. Kullanıcının oluşturduğu kod parçalarına tıklandığında bilgi kutusunda başlık gösterilmesi istenmiştir. Bu doğrultuda kod parçalarına başlık verilip sistem içerisinde tutulması gerekmektedir.
3. Kod parçasının işlevinin tanımlanıp veri tabanı içerisinde tutulması istenilmiştir.
4. Kullanıcı iş yüklerinin IM-FIT içerisinde tanımlanıp tutulması gerekmektedir.
5. İş yükünde gerçekleşecek işlemin tanımlanması gerekmektedir.
6. Mutant satır oluşturulması ve tutulması gerekmektedir.
7. Orijinal satırın AST yapısının oluşturulması ve saklanması gerekmektedir.
8. Mutant satırın AST yapısının oluşturulması ve saklanması gerekmektedir. Bu sayede karşılaştırılma yapılabilmektedir.

2.2. IM-FIT Gereksinimler

IM-FIT aracının veri tabanının hedefi, kullanıcının kaynak kodunu ve uygulama üzerinde yaptığı seçimleri ve kullanıcının uygulama üzerinde yaptığı seçimler sonucunda elde edilen

çıktıları güvenli bir şekilde saklanmak, uygulama sonucunda elde edilen sonuç raporunu saklanmak ve kullanıcıya başarılı bir şekilde göstermektir.

1. IM-FIT içerisinde oluşturulan kod parçalarının isimleri tutulmalıdır. Böylelikle IM-FIT “Code Snippets” tablosuna kod parçasını "custom" özelliği kullanılarak eklenecektir.
2. Kod parçaları numaralandırılmalı ve veri tabanı içerisinde tutulmalıdır. Bu aşama “Code Snippets” tablosu için önem arz etmektedir.
3. ReGex kodlarının veri tabanı içerisinde tutulması istenmektedir bu sayede IM-FIT içerisinde oluşturulan kod parçalarının satırlar içerisinde bulunabilmesi hedeflenmiştir.
4. Kullanıcının oluşturduğu kod parçalarına tıklandığında bilgi kutusunda başlık gösterilmesi istenmiştir. Bu doğrultuda kod parçalarına başlık verilip sistem içerisinde tutulması gerekmektedir.
5. Kod parçasının işlevinin tanımlanıp veri tabanı içerisinde tutulması istenilmiştir.
6. Kullanıcı iş yüklerinin IM-FIT içerisinde tanımlanıp tutulması gerekmektedir.
7. İş yükünde gerçekleşecek işlemin tanımlanması gerekmektedir.
8. Oluşturulan hatalara isim verilmesi gerekmektedir bu sayede hatalar arayüz üzerinde yer alır.
9. Hata uygulanacak olan satırın bilgisi veri tabanı içerisinde tutulmalıdır.
10. Mutant satır oluşumu bilgisi veri tabanı içerisinde tutulmalıdır.
11. Orijinal satırın AST yapısının oluşturulması ve saklanması gerekmektedir.
12. Mutant satırın AST yapısının oluşturulması ve saklanması gerekmektedir. Bu sayede karşılaştırılma yapılabilir.
13. Çalıştırma planının isminin oluşturulması ve veri tabanı içerisinde tutulması gerekmektedir.
14. Çalıştırma işleminin hangi işletim sistemi üzerinde yapıldığı bilgisi tutulmalıdır.
15. Çalıştırma işleminin hangi Python sürümüyle yapılacağı bilgisi tutulmalıdır.
16. Çalıştırma işleminin hangi ROS sürümüyle yapılacağı bilgisi tutulmalıdır.
17. Çalıştırma işleminde Gazebo kullanılıp kullanılmayacağı bilgisi tutulmalıdır.
18. Seçilen FI plan bilgisi tutulmalıdır.
19. Çalıştırma işleminin kaç gigabayt RAM kullanılacağını bilgisi arayüzden seçilmeli ve tutulmalıdır.

20. Çalıştırma işleminde hangi metrik ve durumların kaydedileceğinin bilgisi veri tabanı içerisinde tutulmalıdır.
21. Kapsanmayan kodun tespitiyle elde edilen verinin analiz ve gösterim için tutulması gerekmektedir.
22. Mutant kod çalıştırıldığında zaman aşımı durumunu oluşturan verinin analiz ve gösterim için timeout metriği, runtime ve compile hatalarının tutulması gerekmektedir.
23. Mutant kodun yok sayıldığı durumun analiz ve gösterim için tutulması gerekmektedir.
24. Çalıştırma adımında gerçekleşen hatalar ve hangi hatanın ne kadar sayıda gerçekleştiğini gösteren verilerin karşılaştırma işlemi için tutulması gerekmektedir.
25. Çalıştırmalar sonucunda elde edilen mutasyon skoru verisinin gösterimi ve ileriki işlemler için tutulması gerekmektedir.
26. ROS senaryolarının gösterimi ve karşılaştırılması için. rosbag veri tipindeki dosyaların saklanması gerekmektedir.
27. V&V raporlarının oluşturulup veri tabanı içerisinde saklanması gerekmektedir.

2.2.1. IM-FIT Fonksiyonel Gereksinimler

1. Kullanıcı kaynak kodunu yükleyebilmelidir.
2. Kullanıcı kaynak kodunun bir parçasını yükleyebilmelidir.
3. Kullanıcı hangi hataları uygulamak istediğini seçebilmelidir.
4. Hatanın uygulanabileceği kod satırları saklanmalıdır.
5. Kullanıcının uygulamak istediği hatalar, uygulanacağı satırlarla birlikte saklanmalıdır.
6. Kullanıcı, hataları uygulanacağı satırlarla görüntüleyebilmelidir.
7. Kod parçacığı için uygulanacak hata planları saklanmalıdır.
8. Kullanıcı oluşturduğu hata planlarına erişebilmelidir.
9. Uygulama çalıştırma seçeneklerinin belirlenmesine izin vermelidir.
10. Çalıştırma planının ismi, işletim sistemi, python sürümü, ROS sürümü, Gazebo kullanılıp kullanılmayacağı, çalıştırılacak FI plan, hafıza boyutu çalıştırma aşaması için saklanmalıdır.
11. Kullanıcı çalıştırmak istediği FI plan'e uygulamanın çalıştırma penceresinden erişebilmelidir.
12. Kullanıcı çalışacak FI plan'in hangi metrikler için çalışmasını istediğini belirtebilir. Bu sebeple seçilen metrikler saklanmalıdır.

13. Çalıştırılan FI plan'ın sonuçları rapor oluşturması için saklanmalıdır. Bu sonuçlar mutasyon skoru, seçilen metriklerin ve durumların sonuçları, hatalar ve sayıları, AST diyagramları ve .rosbag uzantılı simülasyon senaryolarıdır.
14. Kullanıcı çalıştırılan FI plan sonuçlarına erişebilmelidir.
15. Uygulama monitoring kısmında FI plan sonuçlarını göstermelidir.
16. Çalıştırılan FI plan sonuçları pdf formatında saklanmalıdır.
17. Kullanıcı FI plan sonuçlarının raporuna erişebilmelidir.

2.3. VALU3S V&V TOOL Gereksinimler

1. Veritabanının bu parçası, Valu3s projesi kapsamında uygulanacak UPPAAL ve ROSMonitoring işlemlerinin verilerini saklamak/yönetmek amacı ile tasarlanmaktadır.
2. Uppaal üretilen her bir model .xml uzantılı olarak bir dosyada tutulmaktadır. Bu dosya içeriğinin veri tabanına aktarılması gerekmektedir.
3. Adı, oluşturulma tarihi, açıklaması vb bilgiler modele ait gereksinim kapsamındadır.
4. Cml içerisindeki tag'ler dikkate alınarak dosya içeriği uygun şekilde parse edilmelidir. - declaration -template(s) -system -queries vb. durumlar gerçekleştirilmelidir.
5. Kullanıcı daha önceden girilen verilere erişimi ve yeni veri yazma işlemlerini arayüz üzerinden yapabilmelidir.
6. Modelin veri tabanına yazılması veya veri tabanından okunması kapsamında bir ara yüz üzerinden .xml dosyanın okunması ve veri tabanına kaydedilmesi
7. Veri tabanındaki bir modelin okunması ve .xml model dosyasının oluşturulması işlemlerinin gerçekleştirilmesi sağlanacaktır.
8. Bir model sistemine ait bilgilerin, yazılması, okunması ve daha öncesinden yazılmış bir sistemin dosyalarının kullanıma hazır durumda oluşturulması.

2.3.1. Uppaal Gereksinimler

1. Uppaal üretilen her bir model .xml uzantılı olarak bir dosyada tutulmaktadır. Bu dosya içeriğinin veri tabanına aktarılması gerekmektedir.
2. Modelin adı, oluşturulma tarihi ve açıklaması gibi özellikleri veri tabanı içerisinde tutulmalıdır.
3. Xml dosya tipi içeriği uygun bir şekilde parse edilmelidir veya veri tabanı içerisinde bir bütün olarak daha sonra parsellenip arayüzde kullanmak üzere saklanmalıdır.
4. Veri tabanı içerisinde xml dosya tipi okunmalı ve yazılmalıdır.

5. Verifier kısmında yazılan özellikler ve doğrulama sonuçları saklanmalıdır. (Model ID - Özellik ID -Açıklaması -Sonuç vb.)

2.3.2. ROSMonitoring Gereksinimler

ROSmonitoring, ROS programları için, online ya da offline modlarda çalışabilen bir program takip uygulamasıdır. Bu uygulamanın çalışma (başlangıç) parametreleri bir .config dosyası içerisinde sisteme verilir. Veritabanında saklanması gereken bilgi .config dosyasının içeriğidir. .config dosyalarının okunması, yazılması ve kullanılabilir halde oluşturulması arayüz üzerinden yapılabilmelidir. Buradaki takip işlemleri UPPAAL başlığında bahsedilen ilgili modeller ile de bağlantılı olmalıdır.

RosMonitoring Gereksinim maddeleri aşağıda sunulmaktadır.

1. Runtime Verification sürecinde kullanılan içeriklerin veri tabanına aktarılması gerekmektedir.
2. config dosya içeriğinde yer alan tag'ler dikkate alınarak dosya içeriği uygun şekilde parse edilmelidir. Online ve offline olarak farklı iki .config dosya içeriği olduğu dikkate alınmalıdır. Doğrulama yapılacak modele ait bilgiler tutulmalıdır (Model ID, vb.).
3. Doğrulama sürecinde iki farklı yöntem (TL Oracle ve RML Oracle) kullanılabilir. Bu yöntemlerin her biri farklı specification language(sırasıyla TL ve RML) kullanır. Her bir özellik için bu iki dilde yazımı olduğu dikkate alınmalıdır.
4. Bir ara yüz üzerinden .config dosyanın okunması ve veri tabanına kaydedilmesi, veri tabanındaki bir içeriğin okunması ve .config dosyasının oluşturulması sağlanmalıdır.

2.4. Fonksiyonel Olmayan Gereksinimler

- ❖ Menü tasarımı göze hitap etmeli ve fonksiyonel olmalıdır.
- ❖ Menüdeki gerçekleştirilecek fonksiyonel işlemler için fonksiyonların en optimize şekilde çalışması gerekmektedir.
- ❖ Veri kaybı olabildiğince önlenmelidir ve olası hata veri girişleri engellenmelidir.
- ❖ Oluşturulan veritabanında beklenmeyen durumların, yazılımın hatalı olmaması gerekmektedir.
- ❖ Oluşabilecek hata durumlarında, veritabanının oluşan hataya uygun bir hata mesajı ortaya koyması beklenmektedir.

- ❖ Geliştirilen araç Ubuntu üzerinden yürütüldüğü için programın Ubuntu üzerinde daha iyi performans vermelidir.
- ❖ Ubuntu 16.04 – 18.04 – 20.04 sürümlerini desteklemelidir.
- ❖ Veritabanının kullanımı için PostgreSQL gereklidir.
- ❖ UPPAAL 4.0.15 sürümü desteklenmelidir.
- ❖ ROS Noetic Ninjemys sürümü desteklenmelidir.
- ❖ Kullanıcı daha önceden girilen verilere erişimi ve yeni veri yazma işlemlerini arayüz üzerinden yapabilmelidir.
- ❖ Okuma ve yazma işlemleri yetkisi bulunan kullanıcılara sunulacaktır.

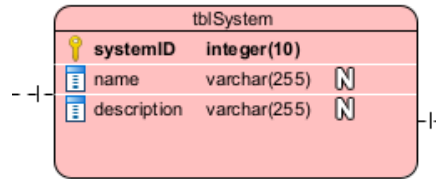
3. VERİ YAPISI

3.1 UPPAL/ROSMonitoring Varlık-İlişki Tabloları

3.1.1. Sistem

tblSystem isimli tablo ana kaynak görevi görüp doğrulanacak programların temel bilgilerini içerir. Birincil anahtar olarak *systemID* isimli veriyi tutmaktadır.

Buradan ikiye ayrılan modelimizin, bir tarafında UPPAAL ile alakalı veriler saklanırken diğer tarafında ROSMonitoring verileri saklanır. Ana bir tablo oluşturmamızın sebebi, aynı program üzerinde yapılacak farklı doğrulama işlemlerinin birbirleri ile bağlantılı olmasıdır.



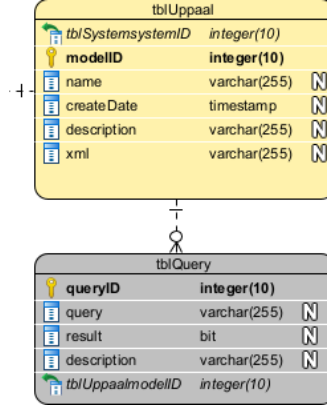
Şekil 1: Sistem Tablosu

3.1.2. Uppaal Tablosu

tblUppaal tablosu; UPPAAL tarafının ana tablosudur ve UPPAAL doğrulama sisteminde, bir programın doğrulama işlemini yürütmek için gerekli olan bilgileri içerir. UPPAAL üzerinde sistemler, modeller tasarlanarak işleme sokulur ve bu modeller UPPAAL tarafından xml formatında saklanır. Burada saklayacağımız ana verimiz de bu xml dosyasıdır. PostgreSQL, xml formatında veri saklanması desteklemektedir. Tasarladığımız sistem de bu özellikten faydalanarak xml dosyamızı olduğu gibi saklar. *tblUppaal*; birincil anahtar olarak *modelID*, yabancı anahtar olarak *tblSystem*'dan gelen *systemID*'yi tutar.

3.1.3 Query Tablosu

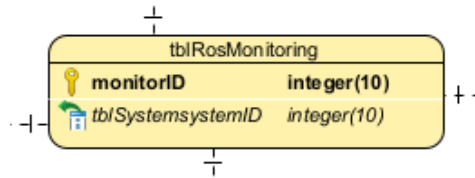
tblQuery, UPPAAL üzerinde sorgulama yapmak için kullanılan sorgu denklemlerini ve sonuçlarını saklamaktadır. Birincil anahtar olarak *queryID*, yabancı anahtar olarak ise UPPAAL modelleri ile bağlantı sağlanması için *modelID* verileri tutar. Sorgu sonuçlar sistem tarafından otomatik bir şekilde iletilememektedir. Bu sebeple sonuçları doldurma işleminin arayüz üzerinden yapılması planlanmıştır.



Şekil 2: Uppal Tabloları

3.1.4. RosMonitoring Tablosu

tblRosMonitoring, veritabanının bu kısmının geçiş tablosudur. ROSMonitoring üzerinde doğrulama yapmak için gerekli config dosyalarının içeriğine ve doğrulama sonuçlarına bağlantı sağlamaktadır. Birincil anahtar olarak *monitorID*, yabancı anahtar olarak *tblSystem* bağlantısı için *systemID* anahtarı saklamaktadır.

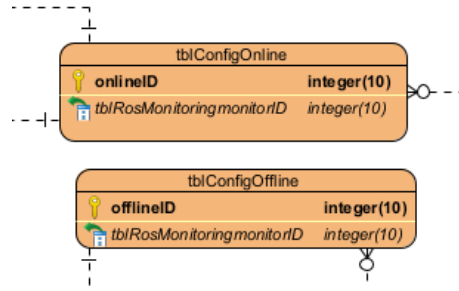


Şekil 3: RosMonitoring Tablosu

3.1.5. Config Tablosu

tblConfigOnline ve *tblConfigOffline*, ROSMonitoring üzerinde doğrulama işleminin başlatılması için gerekli dosyaların içeriklerini barındırmaktadırlar. Sırası ile *onlineID* ve *offlineID* olarak isimlendirilen birincil anahtarları barındırırlar. Yabancı anahtar olarak ise *tblRosMonitoring* tablosu ile bağlantı sağlanması için *monitorID* anahtarı kullanırlar. Online ve

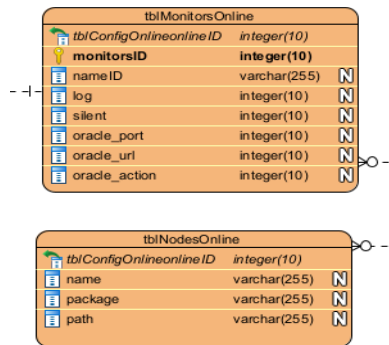
offline olarak ayırmamızın sebebi, config dosyalarının yapısal içeriğinin buna bağlı olarak değişmesidir. Bu iki tabloya bağlı olan diğer tablolar dosya içeriklerinin ayrıştırılmış ve veritabanında saklamaya uygun hale getirilmiş hallerini saklamak için kullanılmaktadırlar. Bu aşmada alternatif bir yol olan, dosya içeriklerini xml formatına getirip saklamak da düşünülmektedir. Verimliliği ve kullanılabilirliği konusunda değerlendirmeler yapılacaktır.



Şekil 4: Config Tabloları

3.1.6. MonitorsOnline ve NodesOnline Tabloları

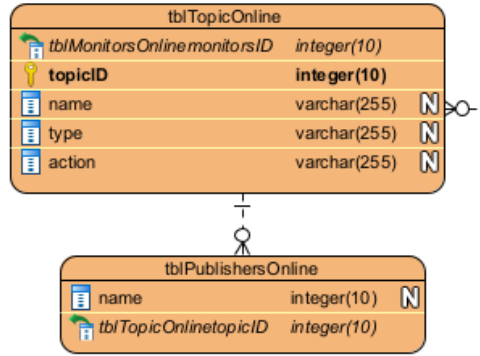
tblConfigOnline tablosu altında yer alan *tblNodesOnline*, config dosyası içerisinde yer alan node alanına ait bilgileri; *tblMonitorsOnline*, config dosyası içerisindeki monitör bilgilerini saklamaktadır. Bahsettiğimiz veriler çoğul olabileceği için ayrı tablolar içerisinde gömülmesi uygun görülmüştür. Üst tabloları ile bağlantı sağlanması *tblNodesOnline* ve *tblMonitorsOnline* içerisinde yabancı anahtar olarak, *tblConfigOnline*'dan gelen *onlineID* anahtarı barındırır. *tblMonitorsOnline* aynı zamanda kendi birincil anahtarı olan *monitorsID* verisini de tutar.



Şekil 5: ONline Config Alt Başlıkları

3.1.7. TopicOnline ve Publishers Tablosu

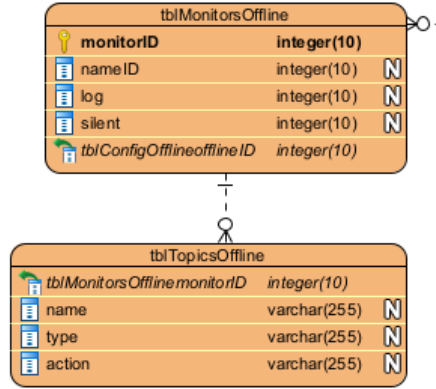
Config dosyasının monitor parçasının içerisinde de çoğul olarak yer alabilecek topicler ve topiclerin içerisinde çoğul olabilecek publisherlar da ayrı tablolar haline *tblTopicOnline* ve *tblPublishersOnline* isimleri ile oluşturulmuştur. *tblTopicOnline*, *tblMonitorsOnline*'ın birincil anahtarı olan *monitorsID*'yi yabancı anahtar olarak tutar. *tblPublishersOnline* ise *tblTopicOnline*'ın birincil anahtarı olan *topicID*'yi gerekli bağlantıyı sağlamak adına yabancı anahtar olarak içerisinde bulundurur.



Şekil 6: Monitors Online ALt Başlıkları

3.1.8. MonitorsOffline Tablosu

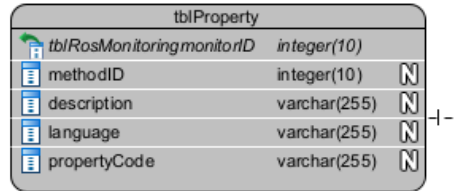
tblConfigOffline tablosunun alt tabloları ise, dosya içerisindeki çoğul olabilecek monitor ve monitorlerin altında çoğul bulunabilecek topicleri düzenli bir şekilde saklamak için, *tblMonitorsOffline* ve *tblTopicsOffline* isimleri ile oluşturulmuştur. *tblMonitorsOffline* doğrudan *tblConfigOffline* tablosuna bağlıdır dolayısıyla içerisinde yabancı anahtar olarak *offlineID* bulundurur. *tblTopicsOffline* ise *tblMonitorsOnline*'ın alt tablosudur. *tblMonitorsOffline*'ın birincil anahtarı olan *monitorID*, *tblTopicOnline*'ın yabancı anahtarıdır.



Şekil 7: MonitorsOffline Alt Başlıkları

3.1.9. Property Tablosu

tblProperty ise ROSMonitoring üzerinde yapılacak doğrulama işleminin sonuçlarını saklayacak şekilde tasarlanmıştır. tblRosMonitoring’den gelen *monitorID* yabancı anahtarını taşır. İçerisinde doğrulama için yazılan kod parçalarını, işlemin sonucunu ve açıklamasını barındırır. Doğrulama işlemi iki farklı dilde yapılabilmektedir (TL Oracle, RML Oracle). Tutulan kod parçalarının dili de bu tablo içerisinde saklanmaktadır.



Şekil 8: tblProperty Tablosu

3.2. IM-FIT Varlık-İlişki Tabloları

3.2.1 Source Code Tablosu

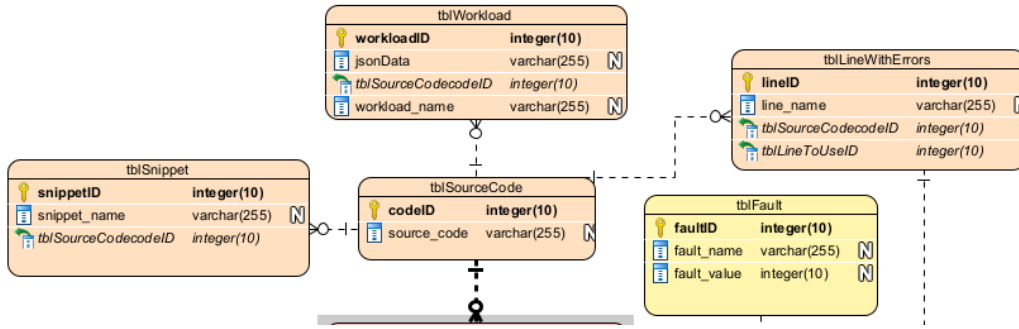
IM-FIT aracının yaptığı mutasyon kod üretme işleminde kullanıcının kaynak kodunu veri tabanında tutabilmesi Source Code adlı tablo oluşturuldu. Bu oluşturulan tablodaki codeID primary key (PK) olarak verildi. Foreign Key (FK) olarak ise Snippet, Workload, LineWithError ve Plan tablolarına verildi. Bunun sebebi Source Code tablosundaki kaynak koda bağlı olarak diğer tablolar da oluşacaktır.

3.2.2. Snippet Tablosu

Source Code ile Snippet arasında 1 to Many ilişkisi kullanıldı, çünkü kaynak kodun uygulanabileceği birden fazla Snippet vardır. Snippet'lar kodlardaki if, else, while return gibi olan yapılardır. Bu Snippet isimlerini snippet_name column'u altında saklanacaktır. Snippet'da PK snippetID olarak tanımlanmıştır. Source Code ile arasında codeID'leri FK ile birbirlerine bağlanmıştır.

3.2.3. Workload Tablosu

Workload'lar yani iş yükleri kaynak koddan oluştuğu için Source Code tablosu ile 1 to Many ilişkisi kuruldu. Ayrıca Workload tablosunda jsonData isimli column json dosyalarını tutabilmektedir. Workload tablosunun PK'i workloadID olup codeID'si Source Code'a FK ile bağlıdır.



Şekil 9: Source Code ve Diğer Tablo İlişkileri

3.2.4. LineWithErrors Tablosu

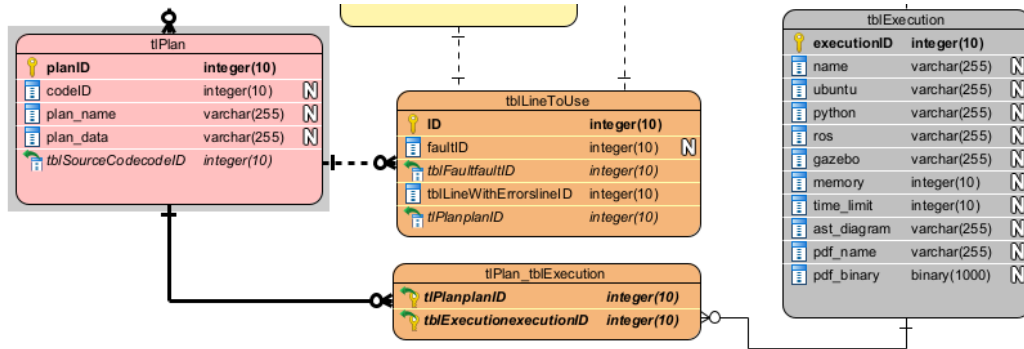
LineWithErrors tablosunda lineID PK olarak verilip, Source Code tablosu ile 1 to Many ilişkisi kuruldu, çünkü LineWithErrors tablosu, kullanıcının hata uygulaması yapmak istediği satırları tutabilecek bir tablodur. Bu seçilen hataları uygulamasını istediklerini line column'u altında saklayabilmekteyiz. Source Code'da bulunan codeID'si ile FK ile birbirine bağlıdır.

3.2.5. Plan Tablosu

Plan, kullanıcının oluşturduğu Workload, Snippet'lar sonucu oluşan kullanıcının daha sonrasında da kullanabileceği kaydettiği json dosyalarıdır. Plan tablosu Source Code tablosu ile 1 to Many ilişkisi ile bağlıdır, çünkü bir kaynak kodun kullanıcı birden fazla planı oluşturulabilir. Plan içerisinde planID PK olarak tanımlanmıştır. plan_name olarak daha öncesinde oluşturulan plan isimleri tutulup, plan_data'da ise oluşturulan planların json dosyaları bulunmaktadır. Aynı zamanda Plan ile Execution arasında Many to Many ilişkisi vardır. Bu 2 tablo arasındaki ilişki Plan_Execution tablosu altında id'leri foreign key (FK) ile birbirine bağlanmıştır.

3.2.6. Execution Tablosu

Aracı kullanan kişinin oluşturduğu proje adını, işletim sistemini, hangi python sürümü üzerinde çalıştığını, ROS ve Gazebo kullanıp kullanmadığı bilgisini, programın ne kadar RAM kullanacağı bilgisini ve seçilen planları saklayabilen bir tablodur. Plan ile Many to Many ilişkisi olup, Plan_Execution tablosu üzerinden gereken id'leri eşleştirilmiştir. Ayrıca programın son olarak bitiş durumunda ortaya çıkan AST diyagramını ve yapılan işlemler hakkında bilgilendirici bir pdf çıktısı da saklanacaktır. Plan ve Execution tablosu için oluşturulan Plan_Execution tablosu 2 tablonun gerekli ID'leri FK ile birbirlerine bağlamaktadır.



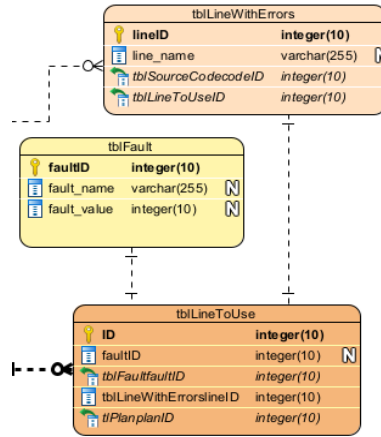
Şekil 10: Plan ve Execution Tabloları Arasındaki İlişki

3.2.7. LineToUse Tablosu

Kaynak kodun hata uygulanabilir satırlarıdır. Fault ve LineWithError tablosu ile arasında 1 to 1 ilişkisi vardır. Plan ile 1 to Many ilişkisi bulunmaktadır. Tabloda bulunan id'ler diğer tablolar ile birbirine bağlanmaktadır. LineWithError'da bulunan lineID'ler birbirlerine FK yardımıyla bağlanmıştır. Plan ile olan bağında ise planID'leri FK ile birbirlerine bağlandı.

3.2.8. Fault Tablosu

Kaynak koda uygulanacak hata çeşitlerinden oluşan bir tablodur. LineToUse ile arasında 1 to 1 ilişkisi vardır. LineToUse ile Fault arasındaki ilişkiden dolayı faultID'leri FK ile birbirlerine bağlıdır. fault_name column'u altında seçilen hata çeşidini tutmaktadır.



Şekil 11: Fault ve LineToUse Tablosu Arasındaki İlişki

3.2.9. Metric Tablosu

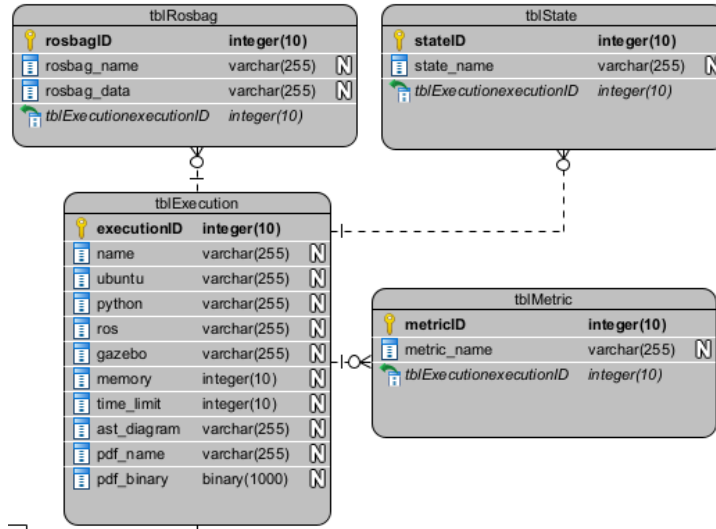
Kodun mutasyon yapılıp bitirilmesi durumunda ortaya çıkan Covered, Detected, Undetected gibi değerlerin saklandığı bir tablodur. metric_name bu metriklerin isimlerini saklamaktadır. Ayrıca Metric ile Execution arasında 1 to Many ilişkisi vardır. Metricde bulunan executionID ile Execution tablosundaki executionID birbirlerine FK ile bağlanmıştır.

3.2.10. State Tablosu

Son olarak ortaya çıkan koddaki Killed, Survived, No Coverage, Timeout gibi değerlerin saklandığı tablodur. State ile Execution tablosu arasında 1 to Many ilişkisi bulunmaktadır. executionID'leri FK ile birbirlerine bağlanmıştır.

3.2.11. Rosbag Tablosu

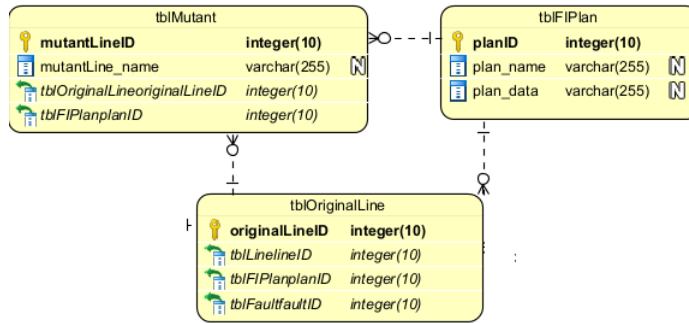
Programın sonucunda oluşan rosbag senaryolarının saklandığı bir tablodur. Oluşan rosbag dosyasının ismini ve oluşan dosyayı saklar. Execution tablosu ile 1 to Many ilişkisi bulunuyor. Bu değerleri Execution içerisinde bulunan değerlerden oluşturduğu için bu ilişki kurulmuştur. executionID'leri FK ile birbirlerine bağlanmıştır.



Şekil 12: State Rosbag ve Metricler arasında bulunan ilişkiler

3.2.12. Mutant Tablosu








Mutasyona uğratılacak satırların, mutasyona uğramış hallerini tutmak için tblMutant tablosu oluşturuldu. FIPlan içerisinde, satırın orijinal hali ve mutasyona uğramış hali bulunduğu için bu 3 tablo arasında 1-to-many bir ilişki kuruldu.



Şekil 13: Mutant Tablosu

3.2.12. Report Tablosu

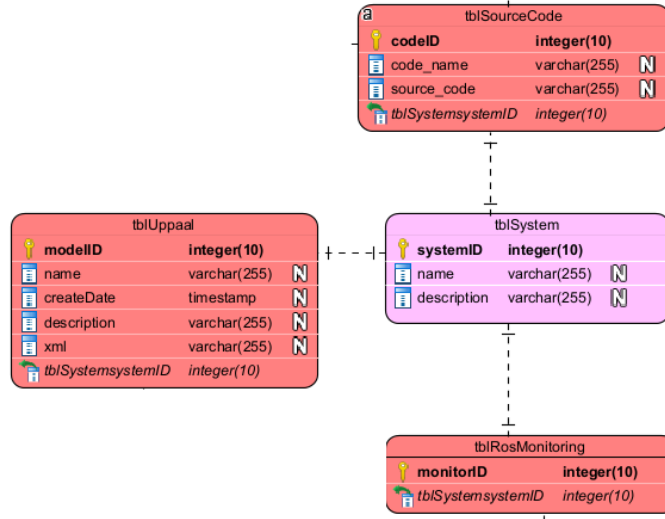
Metric, state gibi sonuçların yanında spesifik saklanması gereken sonuç değerlerini tutmak için bir tblReport tablosu oluşturuldu.

tblReport		
	reportID	integer(10)
	ast_diagram	varchar(255) N
	pdf_name	varchar(255) N
	pdf_binary	binary(1000) N
	mutation_score	double(10) N
	tblFIPlan_tblExecutiontblFIPlanplanID	integer(10)
	tblFIPlan_tblExecutiontblExecutionexecutionID	integer(10)

Şekil 14: Report Tablosu

3.2.13. IM-FIT, ROS Monitoring, UPPAAL

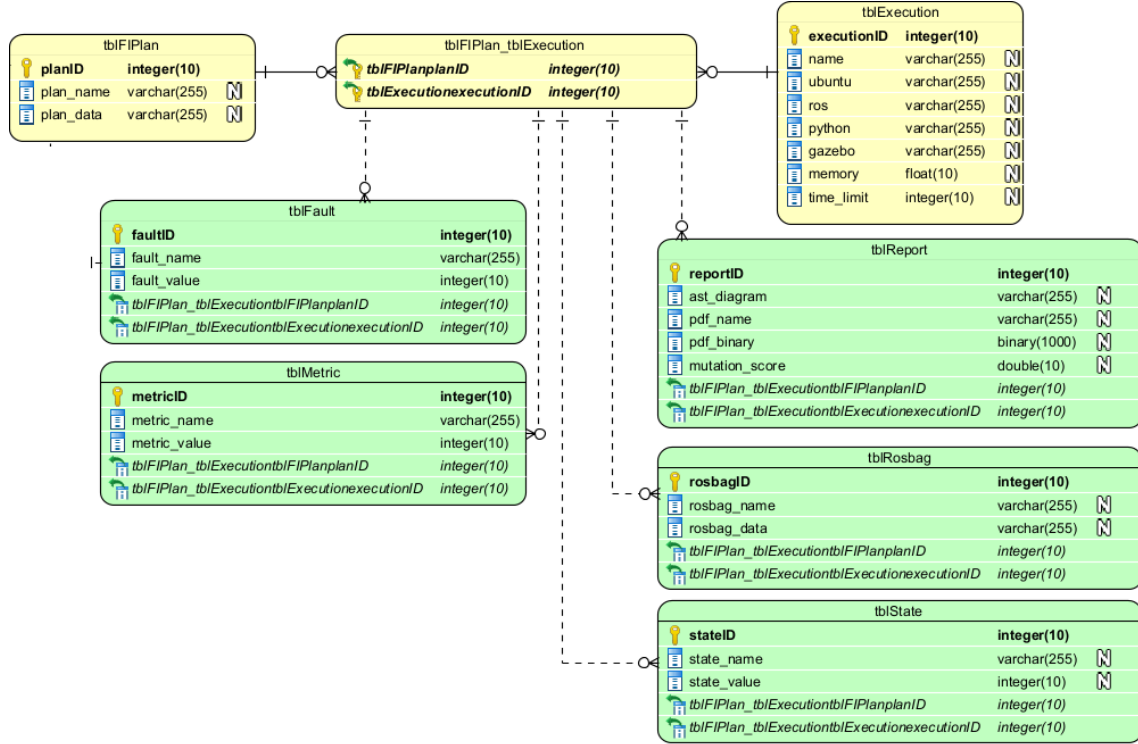
Ayrı olarak tasarlanan IM-FIT, ROS Monitoring ve UPPAAL diyagramları tblSystem tablosu üzerinden birleştirilerek tek bir ER diyagramı çizildi.



Şekil 15: IM-FIT ROS Monitoring Uppal Diyagram

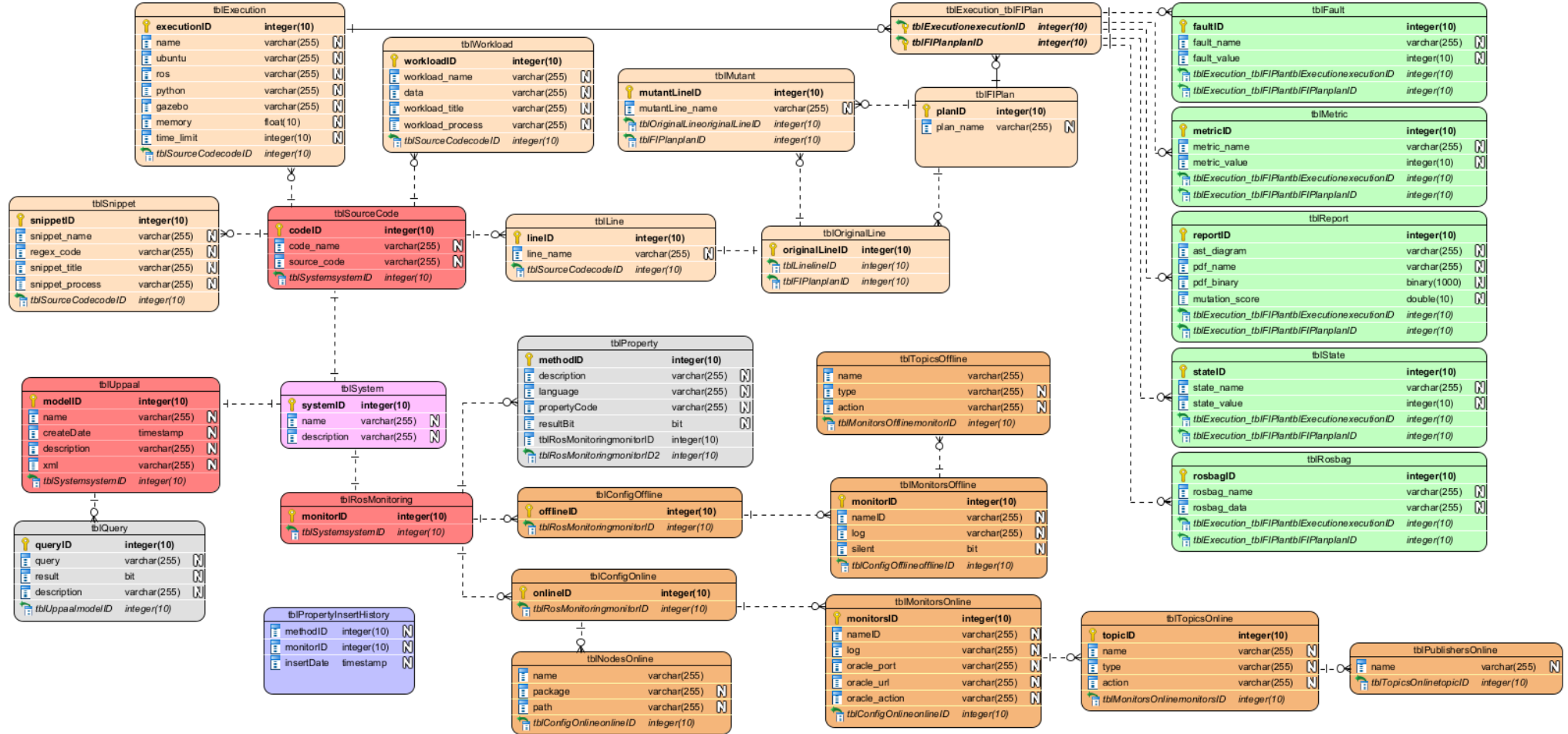
3.2.14. tblFIPlan_tblExecution

Saklanan sonuç değerleri hem FIPlan ile hem de Execution tablosu ile bağlantılı sonuçlar veridiği için tblState, tblMetric, tblRosbag, tblFault, tblReport FIPlan-Execution ara tablosuna bağlandı. Bir FI Plan birden çok execution ile çalıştırılabileceği için, aynı zamanda bir execution ın birden çok FI Plan'ı çalıştırılabileceği için sonuç değerlerinin bulunduğu tablolar da bu şekilde bir ilişki ile tblFIPlan_tblExecution tablosuna bağlandı.



Şekil 16: FIPlan ve Execution Tablosu

3.3. Final ER Diyagramı



4. SORGULAR

Veri tabanımızda ekleme, güncelleme, silinmesi ve çekilme işlemlerinin yapılması için SQL sorguları yazılmıştır. Bu bölümde de SQL içerisindeki karmaşık sorgularımıza yer verilmiştir.

UPPAAL/ROS Monitoring kısmımızda bulunan ‘tblmonitorsonline’ tablosuna aşağıdaki şekilde görülen değerlerin kaydı gerçekleştirilmektedir. Ardından otomatik oluşturan monitorsID değeri döndürülmektedir. Bu sayede monitorsID foreign key ine bağlı olan tabloları kolaylıkla oluşturabilmekteyiz. Bahsedilen sorgu Şekil 17 de gösterilmektedir.

```
cursor.execute(
    "INSERT INTO tblmonitorsonline(nameid, log, silent, oracle_port, oracle_url, oracle_action, onlineid)"
    "VALUES(%s, %s, %s, %s, %s, %s, %s) RETURNING monitorsID;",
    (monitor.id, monitor.log, str(0 if monitor.silent == "0" else 1), monitor.oraclePort, monitor.oracleUrl,
    monitor.oracleAction, str(onlineID))
)
```

Şekil 17: Uppaal/RosMonitoring tarafında gerçekleşen sorgu örneği

Aşağıda görülmekte olan IM-FIT sistemimizin içerisinde yer alan sorgumuz ‘tblFault’ tablosunda planID ve faultname değerlerine göre bir kaydın hali hazırda olup olmadığına bakmaktadır. Kayıt varsa hatanın değeri bir arttırılarak güncellenir eğer kayıt yoksa planID ve faultname değerleriyle ‘tblFault’ tablosuna bir hata girdisi gerçekleşir. Bahsi geçen sorgu Şekil 18 de gösterilmektedir.

```
cursor.execute("""UPDATE tblfault SET faultvalue = faultvalue +1 WHERE planid=%s AND faultname=%s
AND EXISTS(SELECT 1 FROM tblfault WHERE planid=%s AND faultname=%s);
INSERT INTO tblfault(planid ,faultname) select %s, %s WHERE NOT EXISTS
(SELECT 1 FROM tblfault WHERE planid=%s AND faultname=%s)""",
(planid, faultname, planid, faultname, planid, faultname, planid, faultname))
```

Şekil 18: IM-FIT tarafında gerçekleşen sorgu örneği

‘tablefiplan’ tablosuna bir insert işlemi gerçekleştirilirken sistemdeki bir kodun aynı isimde birden fazla FI planı olmaması gerekmektedir. Bu kontrolün gerçekleşmesi için orijinal satırların bulunduğu ‘tableoriginalline’ tablosu ‘tableline’ tablosuyla lineID foreign key değeri üzerinden birleştirilmektedir. Bu sayede ‘originalline’ tablosundaki değerler codID değerine göre alınabilmektedir. Bir FI planın içinde birden fazla orijinal satır bulunabilmektedir. Bu yüzden originallineid FI plan tablosuna foreign key olarak verilmektedir. Bu foreign key üzerinden tablefiplan ile tableoriginalline tabloları inner join özelliği kullanılarak birleştirilmektedir. Yapılan

birleştirme işlemleri sonucunda plan ismi ve codeID değerine göre bir kaydın var olup olmadığı kontrol edilir. Kayıt bulunmadığı takdirde FI plan tablosuna planname değeriyle birlikte veri tabanına eklenmesi gerçekleşir.

```
cursor.execute("""INSERT INTO tblfiplan (planname)
SELECT %s
WHERE NOT EXISTS(SELECT 1 FROM tbloriginalline
INNER JOIN tblfiplan ON tblfiplan.planid = tbloriginalline.planid
INNER JOIN tblline ON tblline.lineid=tbloriginalline.lineid
WHERE tblfiplan.planname=%s AND tblline.codeid=%s)""",(planname, planname, codeid, ))
```

Şekil 19: Diyagram üzerinde gerçekleşen 3'lü inner joinin uygulama sorgusu

5.VTYS ARAÇ KULLANIMI

5.1.Error Handling

Proje içerisinde kullanıcı tarafından girilen Property bilgilerinin geçerliliğini ve doğruluğunu kontrol edilmesinin hem kullanıcı kalitesi hem de projenin backend tarafı için verimli ve sağlıklı olacağı düşünülmüştür. Olası bir hata durumunda geçmiş kayıtlara dönüp kontrol edebilmemiz adına kullanıcı tarafından girilen property değerlerimiz 'tblPropertyInsertHistory' isimli tabloda tutulmaktadır. Bu doğrultuda veri tabanı üzerinde oluşturulmuş olan sorgumuz aşağıda verilmiştir.

```
CREATE OR REPLACE FUNCTION public.tg_propertyinsert_function()
RETURNS trigger
LANGUAGE 'plpgsql'
COST 100
VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
INSERT INTO tblPropertyInsertHistory(methodID, monitorID ,insertDate)
VALUES(NEW.methodID, NEW.monitorID, CURRENT_TIMESTAMP);
RETURN NULL;
exception
when connection_exception then
raise exception 'There is a problem related with connection!!!';
RETURN NULL;
END;
$BODY$;

ALTER FUNCTION public.tg_propertyinsert_function()
OWNER TO postgres;
```

Şekil 2010:Hata tespit aşaması için kullanılan SQL sorgumuz

5.2. Triggers

Projenin error handling kısmında tasarlanan sistemin işleyebilmesi için kullanıcı tarafından yeni eklenen property özelliklerinin algılanması gerekmektedir. Bu doğrultuda verileri sağlam ve doğru almak/depolamak için veri tabanında mevcut olan Trigger özelliği kullanılmıştır. Burda ‘After Insert’ özelliği tercih edilmiştir çünkü kullanıcı gerekli property değerlerini girdikten sonra tabloya yeni bir değer eklenip eklenilmediği kontrol edilmektedir. Eklenildiği saptandığında bu değerler zaman ve id değerleriyle birlikte veri tabanında kendilerine ayrılan tabloda tutulmaktadır. Trigger yapımız Şekil 21’de gösterilmektedir.

```
CREATE TRIGGER tg_propertyinsert
AFTER INSERT
ON public.tblproperty
FOR EACH ROW
EXECUTE FUNCTION public.tg_propertyinsert_function();
```

Şekil 21: Hata tespit için kullanılar trigger sorgumuz

6. KULLANICI ARAYÜZÜ

6.1. IMFIT Arayüzü

Bu bölümde arayüzümüzün IM-FIT tarafında bulunan özelliklerden bahsedilecektir.

Girilmek istenen sistemin adı ‘Enter ssystem Name’ başlığı altında bulunan bölüme kullanıcı tarafından girilmektedir. Giriş işleminin sağlanıp sağlanmadı hakkında kullanıcıya geri dönüt verilmektedir. Sistemde hangi kodların bulunduğu görülmek için ‘List Codes’ butonuna tıklanır. Listelenen kodlar ‘Source Code’ başlığı altındaki combo box da listelenmektedir. Combo box dan seçilen source code un sahip olduğu iş yükleri, workload combo boxu altında listelenmektedir. Ayrıca soruce code un sahip olduğu code snipped ları, snipped kısmında kullanıcıya listelenerek gösterilmektedir. Snippetların seçilmesinden sonra ‘Create Lines’ butonuna tıklanarak satırlar oluşturulur. Oluşturulan satırlar ‘lines’ kısmında listelenir. Anlatılan arayüz bölümü Şekil 22’de gösterilmektedir.

IM-FIT UPPAAL ROS

Enter System Name

system1

Enter

List Codes

Source Codes

Workloads

Snippets

Lines

Create Lines

Şekil 2211: IM-FIT arayüz snippet ve lines

Kullanıcının combo box dan seçtiği source code un içeriği, 'Source Code' başlığının altındaki kısımda kullanıcıya gösterilir. Source code un sahip olduğu iş yükü de combo boxdan seçildi takdirde 'Workload' başlığı altında kullanıcıya gösterilmektedir.

Information

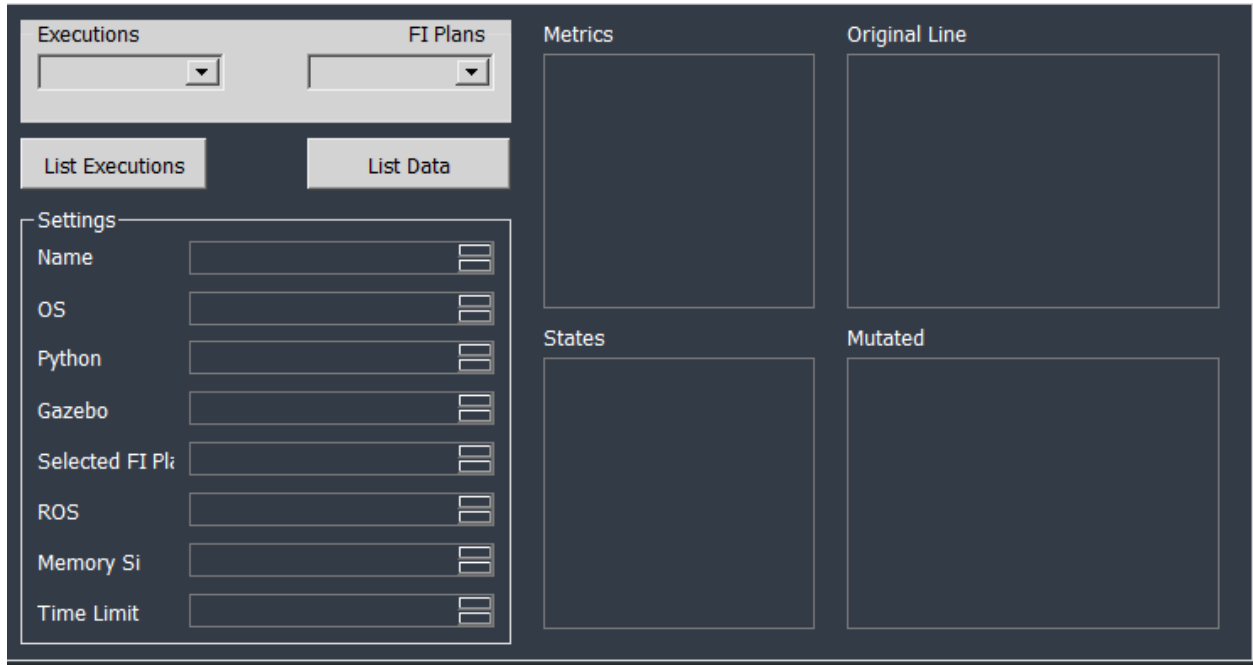
Source Code

Workload

Şekil 23: IM-FIT arayüz Source Code ve Workload

‘FI Plans’ combo box ından çalıştırılacak olan FI plan seçilir. Seçilmenin ardından FI planın içeriğinde bulunan satırın orijinal ve mutasyonlu hali ‘Original Line’ ve ‘Mutated’ başlıkları altında gösterilir.

‘List Executions’ butonuyla sistemin sahip olduğu çalıştırma seçenekleri sol sütte bulunan ‘Executions’ combo boxında listlenir. Bu bölümde seçilen çalıştırma seçeneğinin bilgileri ‘Settings’ başlığı altındaki parametrelerde gösterilmektedir. ‘List Data’ butonuna tıklanmasıyla seçilen FI planın ve seçilen execuionun çalışacağı metrikler ve durumlar ‘Metrics’ ve ‘States’ başlıkları altında listelenir. Bahsedilen özellikler Şekil 24’te gösterilmektedir.



Şekil 24: IM-FIT arayüz Executions ve FI Plans

‘Show Results’ butonuna tıklanmasıyla seçilen FI plan ve executionun çalıştırılması halinde oluşan sonuçlar listelenir. ‘Report’ başlığı altında daha önceden seçilmiş metriklerden ve durumlardan ne kadar sonuç alındığı, seçilen hatalardan kaçar tane olduğu ve mutasyon skoru kullanıcıya gösterilir. Ayrıca ‘AST’ başlığı altında sonuç olarak bir AST diyagramı gösterilmektedir. Bunların yanında çalıştırma işlemleri sonucu olarak Rosbag dosyaları da

gelmektedir. Bu dosyalar ‘Rosbag’ combo box ı altında listelenir. Seçilen dosya içeriği kullanıcıya gösterilir.

‘Open PDF File’ butonu ile oluşturulmuş olan rapor PDF’i açılarak kullanıcıya gösterilir. Daha sonra ‘Send Mail PDF File’ butonu ile girilen e-posta adresine rapor dosyası mail yolu ile gönderilmektedir. Bahsedilen arayüz kısmı Şekil 25’te gösterilmektedir.

The screenshot displays a web application interface with a dark blue header and a light blue body. The header contains the text 'Report' on the left and 'AST' in the center. The body is divided into three main sections: a large empty box on the left, a large empty box in the center, and a smaller box on the right labeled 'Rosbag' containing a dropdown menu. At the bottom, there is a row of buttons: 'PDF File :', 'Open PDF File', a text input field, 'Send Mail PDF File', and 'Show Results'.

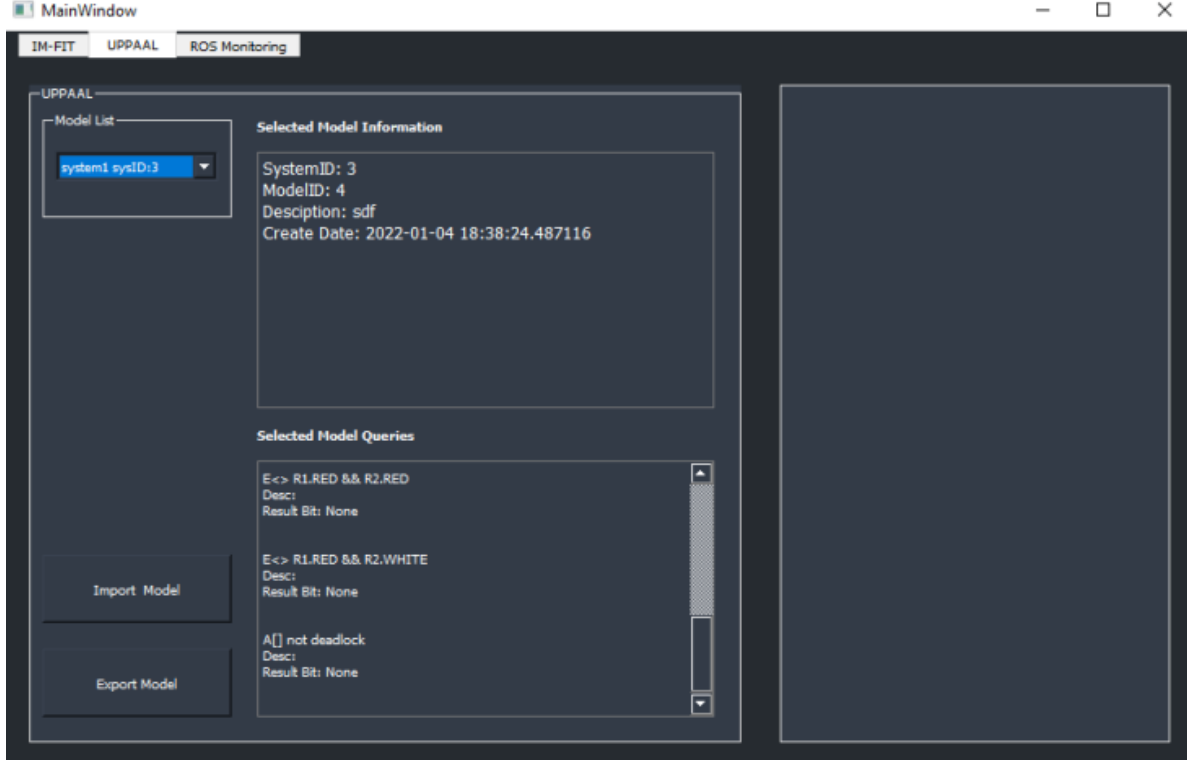
Şekil 25:IM-FIT arayüz Report ve AST

6.2. Uppaal/RosMonitoring Arayüzü

6.2.1.UPPAAL Arayüzü

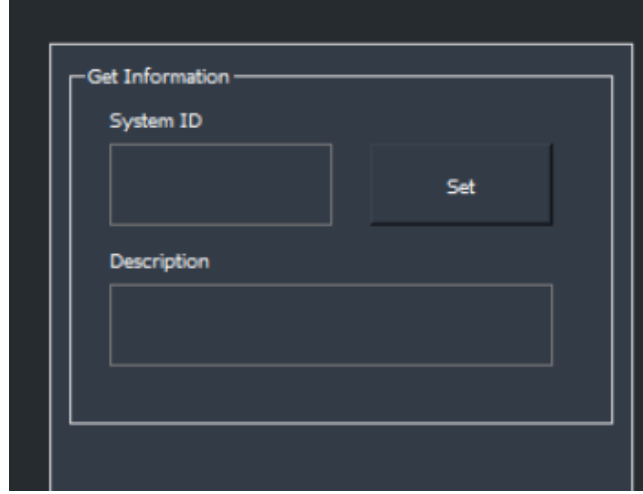
Arayüzümüz sekmeler yardımıyla üç ana başlıkta toplanmıştır. Bu sekmelerden 2 tanesi UPPAAL ve RosMonitoring için ayrılmıştır Şekil 26’da görülen görselimiz UPPAAL arayüzü için ayrılmıştır. Bu sekme içerisinde ‘Model list’ adlı bölümde seçilmiş olan xml dosyasının içerisinde bulunan modeller listelenmiştir ve liste içerisinde seçme işlemi yapılmaktadır.

Seilen model kendisine ayrılmıř ‘Selected Model Information’ adlı alanda kullanıcıya gsterilmiřtir. Burada modele zg zellikler yer almaktadır. Aynı řekilde ilgili modele zg bir diğerk zellik olan sorgular ‘Selected Model Queries’ bařlıđı altında gsterilmektedir.



řekil 26: Uppaal arayz penceresi

Arayz ierisine bařka bir UPPAAL dosyası eklenmek istediđinde panelin sol altında bulunan ‘Import Model’ tuřuna basılarak sađ tarafta yeni bir zellik aılması sađlanır. Bu zellik sayesinde dosyaya sistem numarası ve tanım zelliđi kazandırılmıř olur. Ardından kullanıcı dosyayı setikten sonra arayz zerinde gsterimi gerekleřir. Import tuřuna basıldıktan sonra gsterilen blm řekil 27 ‘de gsterilmiřtir.

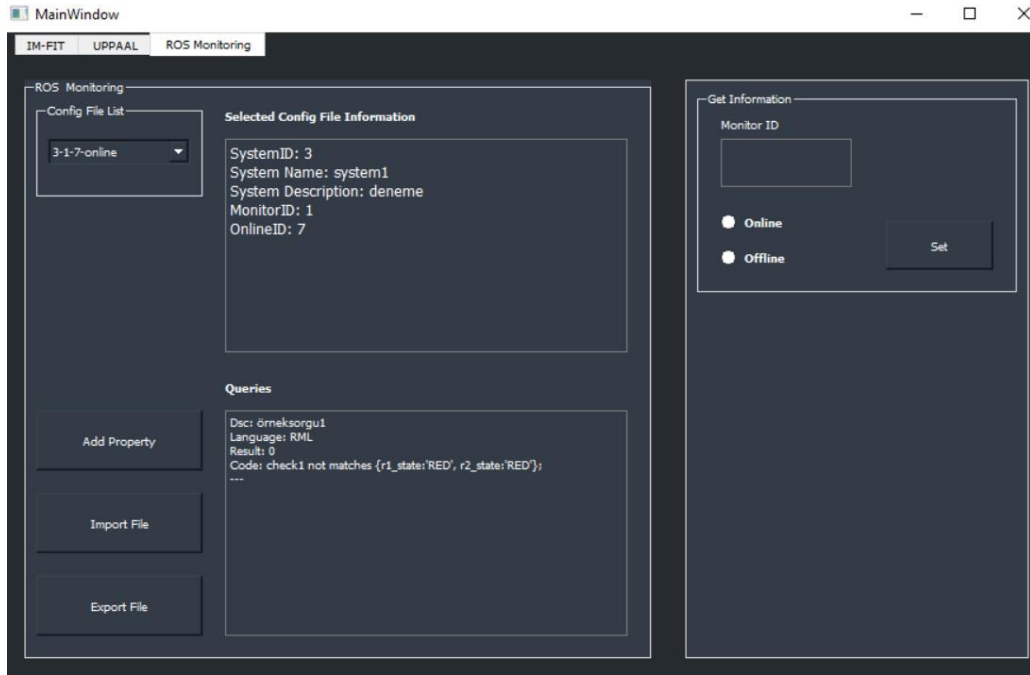


Şekil 27: Uppaal arayüzü Get Information bölümü

UPPAAL arayüz modelinin dosyayı dışarı aktarması 'Export Model' tuşuna basıldıktan sonra ekrana gelen pencere üzerinden konum seçilip onayladıktan sonra gerçekleşmektedir.

6.2.2. RosMonitoring Arayüzü

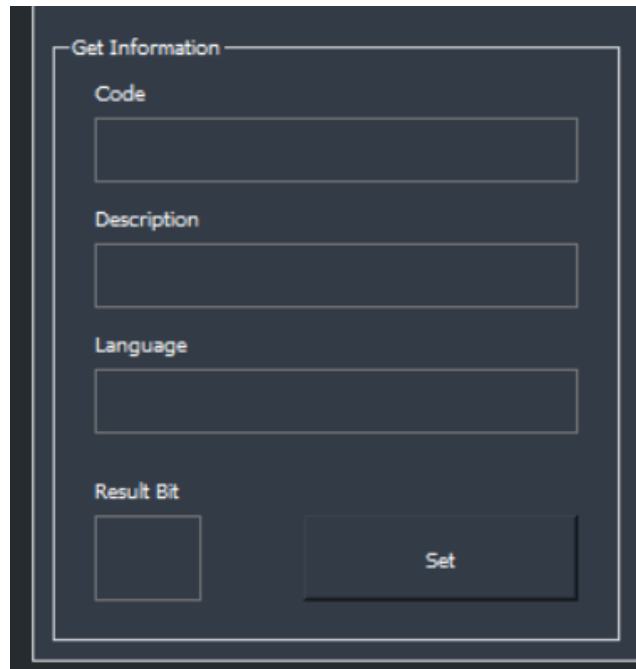
RosMonitoring arayüzümüz bir önceki sekmeye benzer görünümde tasarlanmıştır. Ekranın sol üstünde bulunan liste yardımıyla özelliklerini görmek istediğimiz dosya seçilerek 'Selected Config File Information' ve 'Queries' bölümlerinde gerekli dosya ile ilgili içerikler gösterilmektedir. ROS Monitoring sekmesi şekilde gösterilmektedir.



Şekil 28: Ros Monitoring arayüz penceresi

Kullanıcı arayüze ve veri tabanına yeni bir dosya eklemek istediği takdirde ‘Import File’ tuşuna basarak Şekil 28’in sağ üst kısmında görünen panelden dışarı aktarmak istediği monitörün tipini(online/offline) ve id numarasını belirleyerek dışarı aktarma işlemini gerçekleştirmektedir.

ROS Monitoring dosyasındaki monitörlere yeni bir özellik eklenmek istenildiğinde istenilen monitör seçildikten sonra ‘Add Propert’ tuşuna basılarak ekranın sağ alt bölümünde yeni bir bölümün açılması sağlanır. Bu bölüm içerisinde eklenmek istenen özelliğin nitelikleri yer almaktadır. Kullanıcı Şekil 29’da görülen bölümleri doldurduktan sonra ‘Set’ tuşuna basarak ekleme işlemini gerçekleştirmektedir.



The image shows a dark-themed dialog box titled 'Get Information'. It contains four text input fields: 'Code', 'Description', 'Language', and 'Result Bit'. The 'Result Bit' field is a checkbox. A 'Set' button is located at the bottom right of the dialog box.

Şekil 29: Ros monitoring dosya import

ROS Monitoring arayüz modelinin dosyayı dışarı aktarması ‘Export Model’ tuşuna basıldıktan sonra ekrana gelen pencere üzerinden konum seçilip onaylandıktan sonra gerçekleşmektedir.

7.DİĞER ENTEGRASYONLAR

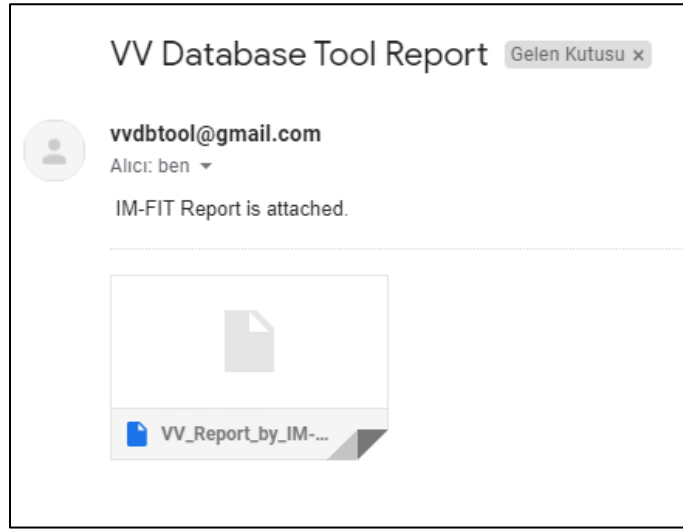
Proje kapsamında tasarladığımız tool şirketin ve kendi telifli programlarının verilerini sakladığı için sosyal medya bağlantısı eklenmemiştir. Bunun yerine veri tabanı içerisinde IM-FIT aracının çalışması sonucunda ulaşılan bilgiler PDF içerisinde tutularak istek dahilinde arayüz

içerisinde sağ aşağıda konumlanan Şekil 30'daki tuş yardımıyla kullanıcıya e-posta yoluyla aktarılmaktadır.

The image shows a dark-themed user interface element. On the left, the text 'E-Mail :' is followed by a white rectangular input field. To the right of the input field is a button with the text 'Send Mail PDF File' in a light gray font.

Şekil 30: Ros Monitoring PDF rapor e-posta yollama özelliği

Örnek e-posta gönderilmesine aşağıdaki görselde yer verilmiştir.



Şekil 31: arayüzün örnek e-posta yollama özelliği

8. ÇALIŞMA ADAM SAAT DEĞERLENDİRME

8.1. Adam – Saat Dağılımı

Projenin başından itibaren harcanan adam-saat değerleri.

Proje üyeleri	Adam Saat
Muhammed Talha ŞAHİN	112
Mine ÇAKIR	108
Hüseyin Can ERGÜN	105
Efekan SARGIN	110
Yusuf Kenan AKSAN	105
Toplam Harcanan Saat	540

Tablo 1: Adam/Saat

8.2. Görevler

Muhammed Talha ŞAHİN	<ul style="list-style-type: none">• İlişkilerin Kurulması• UPPAAL/Ros Monitoring Veri tabanı• UPPAAL/Ros Monitoring Veri tabanı Arayüz Entegrasyonu• UPPAAL/Ros Monitoring Veri tabanı Sorgu
Mine ÇAKIR	<ul style="list-style-type: none">• İlişkilerin Kurulması• Arayüz Tasarımı• UPPAAL /Ros Monitoring Veri tabanı Sorgu
Hüseyin Can ERGÜN	<ul style="list-style-type: none">• İlişkilerin Kurulması• UPPAAL/Ros Monitoring Veri tabanı• UPPAAL/Ros Monitoring Veri tabanı Arayüz Entegrasyonu
Efekan SARGIN	<ul style="list-style-type: none">• İlişkilerin Kurulması• IMFIT Veri tabanı• Genel ER Diyagramı• IM-FIT Veri Tabanı Sorgu
Yusuf Kenan AKSAN	<ul style="list-style-type: none">• Arayüz Tasarımı• IM-FIT Veri tabanı• IM-FIT Veri Tabanı Sorgu

Tablo 2: Görevler

9. SONUÇ VE DEĞERLENDİRME

Projenin başlangıcından itibaren, alınan ve güncellenen isterler doğrultusunda işlemler yapılmıştır. Geliştirme aşamasında verilen isterler haricinde, ders kapsamında uygulanması gereken yöntemler bulunduğu için isterlerin üzerine bazı ekstra tablolar (tblPropertyInsertHistory) ve fonksiyonellikler (çıktı raporunun mail olarak gönderilmesi) eklenmiştir.

Geliştirme süreci veri tabanı oluşturulması ve yönetimi, programa gömülecek SQL komutlarının oluşturulması, arayüz tasarımı olmak üzere üç ana başlıktan oluşmaktadır. Geliştirme sırası veri tabanı, gömülecek komutların oluşturulması, arayüz şeklindedir.

Veri tabanı oluşturulan araç gelişim aşamasında olduğu için belirsiz kalan noktalar olmuştur. Tasarım yapılırken ilerleyen aşamalarda gelebilecek güncellemelere açık yapılmıştır.

Özdeğerlendirme Başlığı	% (0-100)
Proje başındaki amaç-hedeflerde değişme oranı	50
Mevcut veriyapısı (tasarım, oluşturulan tablo vb) nin proje ister ve uygulama ihtiyaçlarını karşılama oranı	95
Mevcut Karmaşık sorguların tasarım ve tam entegre uygulamaya alınma oranı	100
Saklı Yordam, Hata Ayıklama ve Tetikleyicilerin tam entegre uygulamaya alınma oranı	100
Arayüzlerin tamamlanması ve tam entegre uygulamaya alınma oranı	80
Diğer Entegrasyonların (sosyal medya, e-mail/gsm, export/import) tam entegre uygulamaya alınma oranı	100
Projenin tam entegre tamamlanma oranı	90
Uygulamanın alanda kullanılabilirlik oranı	85
Sunumun; genel akışı, teknik içerik aktarım yöntemleri, grup çalışanlarının kendi içinde dengeli şekilde konu anlatımı, zamanı etkin kullanım vb açısından değerlendirmesi	100
Proje raporunun proje çalışmalarını yansıtmaya oranı	100

Tablo 3: Özdeğerlendirme Tablosu