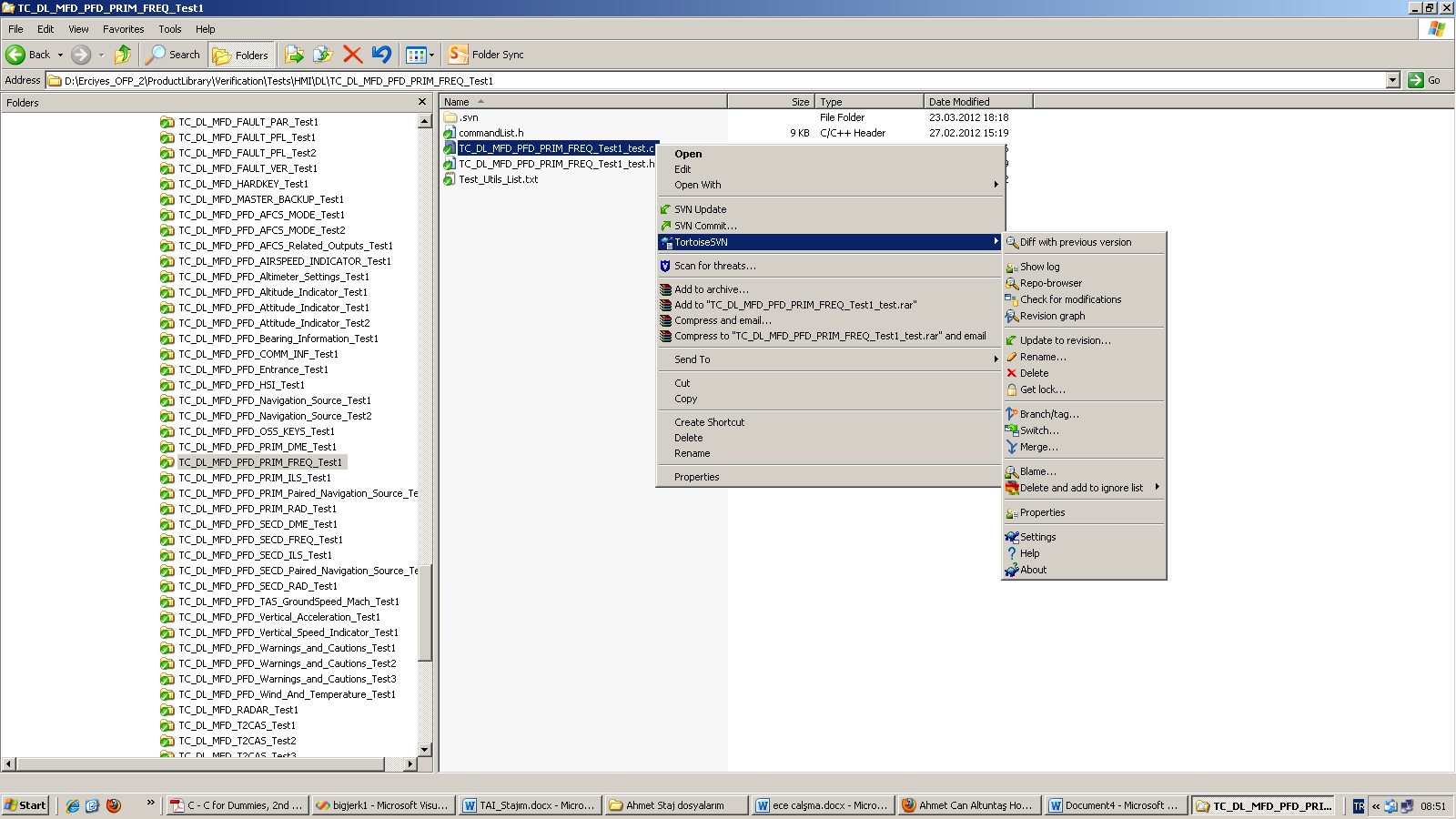
# 5. KULLANILAN ARAÇLAR VE TANITIMLARI

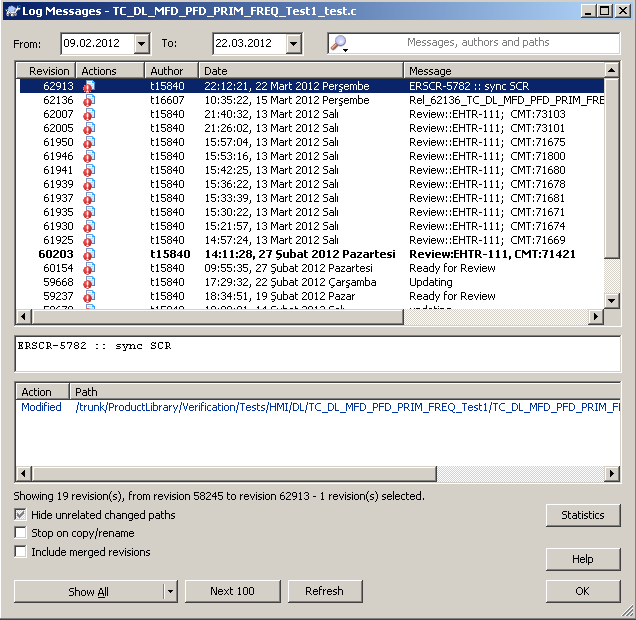
## 5.1- Tortoise SVN

Bu local area netwokinde dosya paylaşım sistemi gibi, bir sistem, (free open-source client for the Subversion control system), yani dosyaları ve directoryleri yönetmemizi sağlıyor. Bu programda windows gezginindeki gibi klasörleri görüyorsunuz, sağ tıkla gelen menüden tortoise işlemleri yapılabiliyor, zaten bulunduğu dosyaların yanındaki işaretlerden SVN’de bulunduğunuzu anlayabiliyorsunuz. SVN bir çok faydası sebebiyle tercih ediliyor, bunlardan bazılarını sıralamam gerekirse:

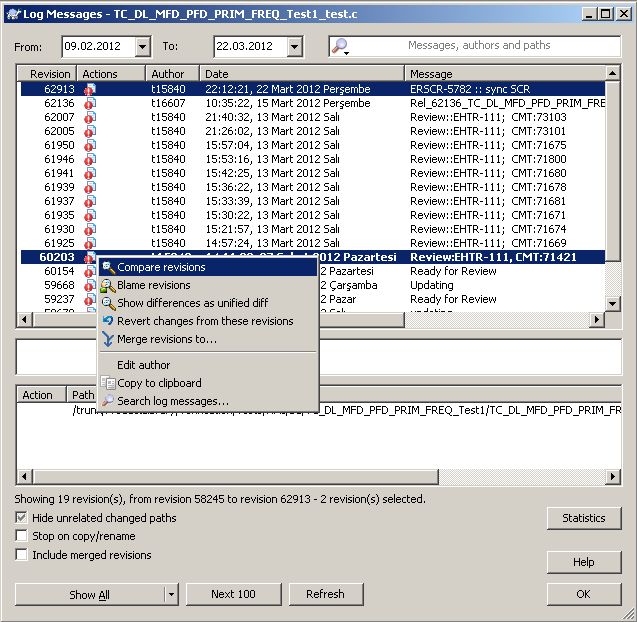
* Aynı anda bir çok kişi aynı dosyaya ulaşıp kullanabilir,
* Kullanıcılar eski dosyalarını ve değişikliklerini versiyon olarak yazabilir, böylece hata ve bug’ların bulunması ve düzeltilmesi daha kolay olmaktadır.
* Datanın historysinin ile birilkte hangi dosya, hangi parçaları, ne zaman ve kim tarafından değişmiş detaylıca görüntülenebilir.

Bu bir kaç işlemin yanı sıra, dosya temizleme, update etme, dosya kilitleme gibi bir çok işlem de yapılabilmektedir. Erciyes projesinde SVN Crucible ile birlikte kullanılıyor. Reviewer test kodunun dosyalarını SVN klasöründen aldıktan sonra, kendi ortamında build edip çalıştırabiliyor. Test yazarı değişiklikleri, commentlerle açıklıyor ve review eden hangi versiyonu inceliyorsa ona gore bakıp, yorumlara da bakarak gerekli düzenlemeler yapılmış mı diye kontrol ediyor. Bu detaylı tooldaki menüden, “show log” seçilip iki versiyona Ctrl ile sağ tıklayıp “Compare Revisions” tıklanınca farklı versiyonlardaki en ufak değişiklikleri renkli ve anlaşılabilir olarak detaylı görebiliyorsunuz, satır satır yanyana inceleme yapıyorsunuz, bunlar da reviewerin işini çok kolaylaştırmış oluyor. Kullandığınız dosyaların son versiyonlarını görüntülemek için “Update” etmeniz gerekebilir, update işlemi o klasörler altındaki tüm dosyaları son versiyonlarına günceller. Tekrar eski versiyonlara dönmek için “update item to revision” kısmından istediği versiyona klasörler geri çekilebilir.

**SVN Dosya Sistemi ve Menüsü**

****

**SVN’de Show Log menüsündeki Versiyonlar, tarihler, author ve commenti**

****

**SVN’de diff almak**

**SVN’de diff alırken kontrol ettiğimiz ekran**

(Kırmızılar değişiklik yapılan yeri göstermektedir, maviler eklenen veya değişen commentlere aittir.)

## 5.2- Crucible Code Review

Yazılım Test Departmanında test yazılımı yazan mühendislerin kodları SVN’nin de yardımıyla referans gösterilerek, browser tabanlı olan bu programa atılıyor. Burada genel anlamıyla anlatırsak, reviewer testi yazanın kodlarını inceler ve kodda bir hata var mı, tüm gereksinimler test edimiş mi, doğru sürümler kullanılarak mı oluşturulmuş şeklinde bir inceleme yapıyor. Reviewer hata bulduğu taktirde, Defect olarak comment giriyor, elindeki checkliste(checklistte olası hataların tipleri ve Numara değerleri var) göre bunu belirtiyor. Daha sonra testin yazarı bu commentleri inceliyor ve eğer gerçekten hata varsa veya kodu güncellemek gerekiyorsa sizinle aynı görüşte olduğunu bildiren “Agreed” yorumunu giriyor ve kodu söylediği versiyona gore düzeltiyor. Reviewer daha sonra bu versiyon ile önceki versiyon arasındaki değişiklikleri görmek için, (hatayı gerçekten düzeltmiş mi diye control ederken) svn’de o versiyonların arasında diff adını verdiğimiz, değişiklikleri yanyana gösteren bir programda açıyor. Eğer değişiklik yapılmışsa, reviewer yorumunun Status’unu “Open”dan “Verified”a çekiyor. Aksi taktirde yine hatayı yazıyor, test geliştirici tekrar düzeltiyor. Test yazarı , reviewer ile aynı fikirde değilse ve hata yoksa “Rejected” diye cevap veriyor, eğer reviewer ikna olursa o da son statusunu “Rejected” a çekiyor.

Alttaki tabloda bir comment alanının yapısı verilmiştir. Bu tablo sadece reviewer için geçerlidir. Test yazarı bu commentlere “Agreed” veya “Rejected” şeklinde cevap verir. Reviewer istediği taktirde bu comment alanını editleyebilir, eğer test yazarı cevap vermemişse istediği commentini silebilir.

|  |  |  |
| --- | --- | --- |
|  | Description | Possible values |
| **Defect** | If there should be a resolution as a result of this comment defect checkbox should be checked.  If this is selected the following attributes will be enabled. | Check or uncheck |
| **Type** | Type of the comment  ( See for ChecklistNA explanation) | Recommendation, Defect, ChecklistNA |
| **Checklist item** | Checklist item no | 1,2,3,4,5,6,7,8,9,… |
| **Status** | Status of the defect.  Open: Initial status value if there is no resolution or verification made on the defect  Verified: The resolution is verified by the reviewer  Rejected: The defect is rejected and no resolution is needed.  N/A: If this field is N/A it means that the checklist item entered is N/A and the comment entered is the explanation of this N/A checklist item. | Open, Rejected, Verified, N/A |

**Crucible Reviewer Comment Tablosu**

Test authoru herhangi bir hata bulduğunda ve değiştirmek istediğinde, bunu bir referans göstererek yapmak durumundadır. Bu nedenle reviewer’a gelip, o defectin bulunduğu yere girip bir comment girmesini isteyecektir.

Eğer verilen commentler sonucunda test authoru ve reviewer anlaşma sağlayamamışsa, bir görüşme ayarlanır ve birlikte toplantı yapılıp, bunun sonucunda bir karar alınıp ona gore bir yol izlenir. Bir comment çözüldüğü taktirde “Resolved” commentiyle beraber versiyon(revision) numarası da bu commentle beraber girilmelidir. Yazılan commentte yapılan değişikliğe ek bir değişiklik gerekiyorsa, yazar aynı yorumu reply ederek bunu farklı revisionlarda nasıl değiştirdiğini yazabilir. Reviewer “Resolved” commentine karşılık, diff kontrolü yapar gerekli değişiklikler yapılmışsa status’unu “Verified” a çeker.

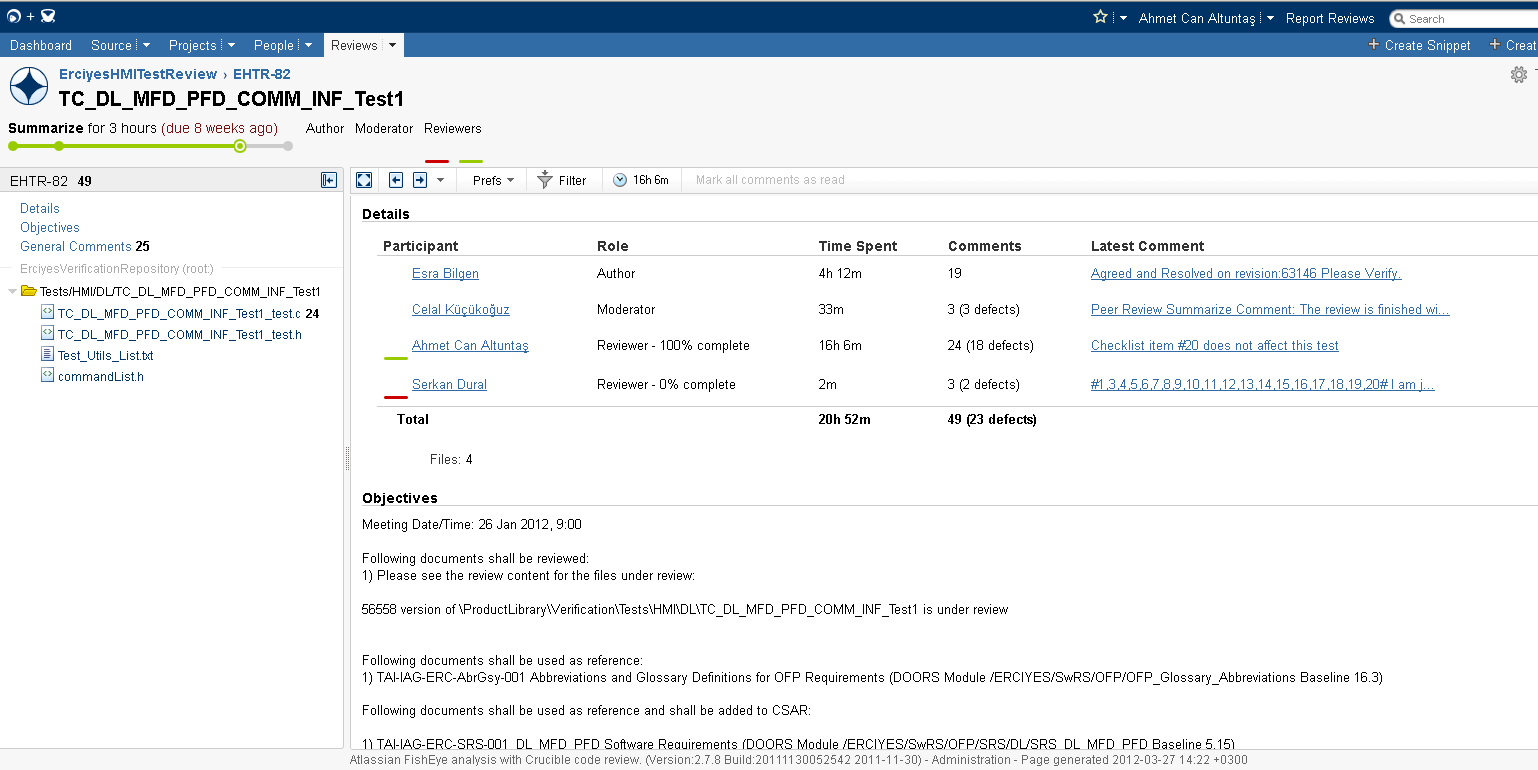
Yazar SVN’de yeni bir versiyon commit ederken bazı kurallara uygun davranmalıdır. SVN’deki commit comment alanına, o minor baselineda hangi reviewerin hangi commentlerini resolve ettiğini yazmalı, mümkün olduğunca aynı reviewerin birden fazla commentini aynı minor baseline içinde resolve etmeli, verification kolay mantıklı bölmeler şeklinde resolutionları sırayla yapmalı, bir çok yeri etkileyen resolutionları ayrı baselinelarda commit etmeli, Crucibleda ilgili defecte reply yaparak X versiyonunda çözüldü diye yazmalı.

Reviewer not applicable olan checklist itemleriyla ilgili bir general comment girmelidir. Bu commentin formatı

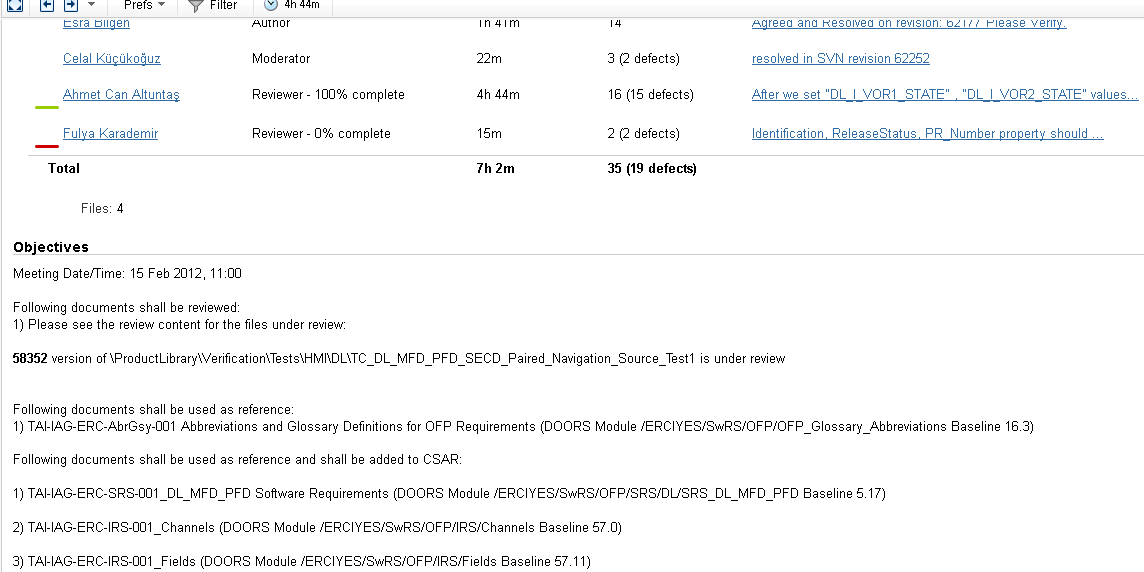
“ #1,2,3,4,5#

I am not responsible for these checklist items.” Şeklindedir. #..# formatı zorunludur, bu formata gore reviewer toolu tarafından parse edilmektedir.

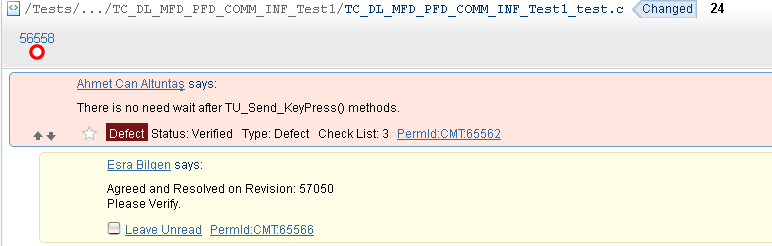
Eğer bir peer review gerekliyse, o Test case’nin moderator takım lideriyle tartışarak, var olan reviewi “abandoned” durumuna çeker ve yeni peer review yapılır, eğer gerekli değilse ona göre commentini girer. Tüm reviewerlar review işlemlerini tamamlamışlarsa moderator, bu test case’i summarize etmeye başlar, bir sorun olduğunda re-open ederek reviewi tekrar açabilir. Moderator, “open” statusunda yorum olmamasını, reviewerlerin N/A commentlerinin girilmiş olmasını, “No checklist item” commentlerinin doğruluğunu, TBD varsa ona açılmış bir SCR(Software Change Request) olup olmadığının doğruluğunu, tüm minor baseline difflerinin en az bir reviewer tarafından verify ediliğini kontrol etmek durumundadır. Uçtan uca diff alıp tüm commentleri tekrar trace etmek zorunda değil. Daha sonra test case’ini summarize ettikten sonra CM yetkilisi ve SQA yetkilisine reviewin bittiğine dair bir link gönderir ve onları bilgilendirir.

****

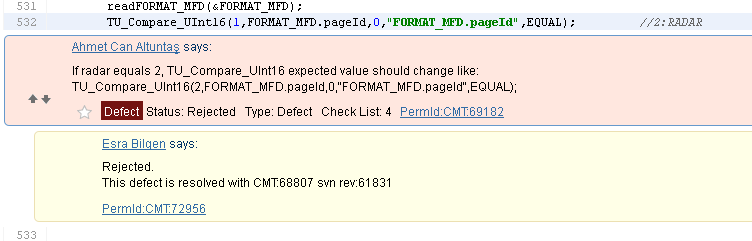
**Crucible’da bir test case’inin genel görüntüsü**

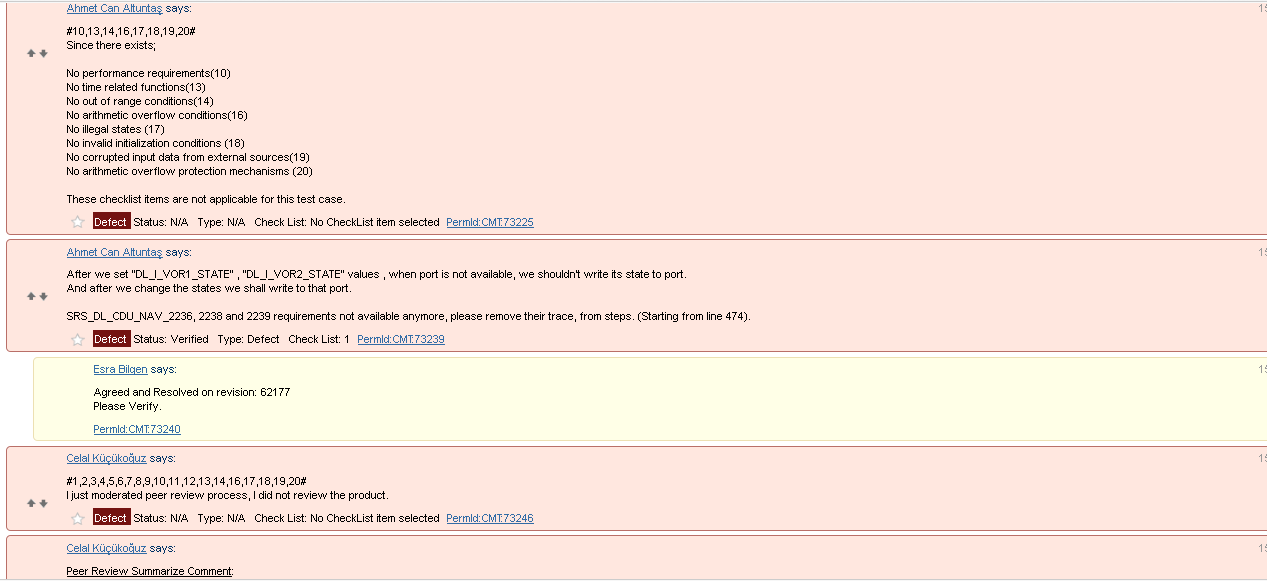
****

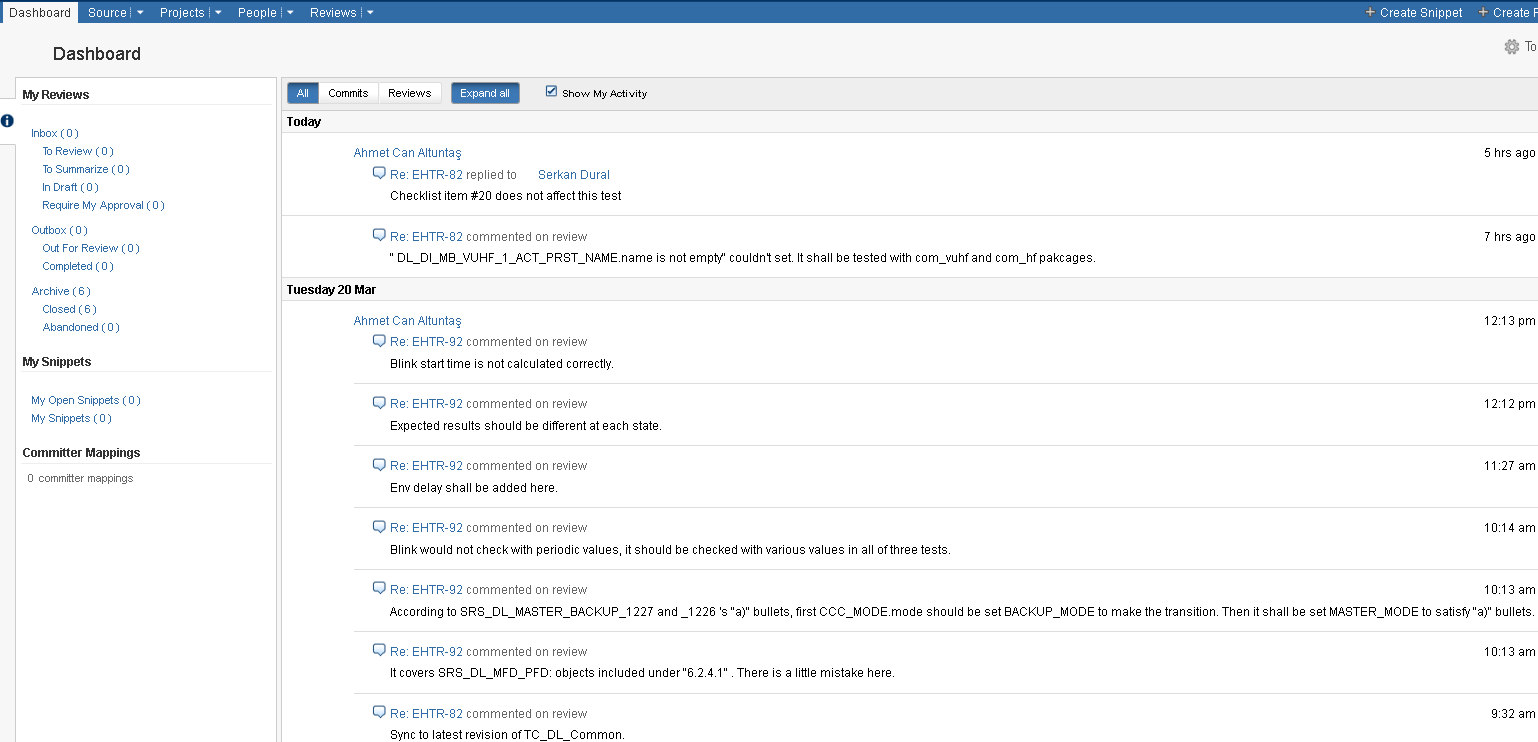
**Test Case Objektifleri ve Referans Dökümanları**

****

**Reviewer ve Author commentleri: Agreed and Resolved mesajı**

**Reviewer ve Author commentleri: Rejected**

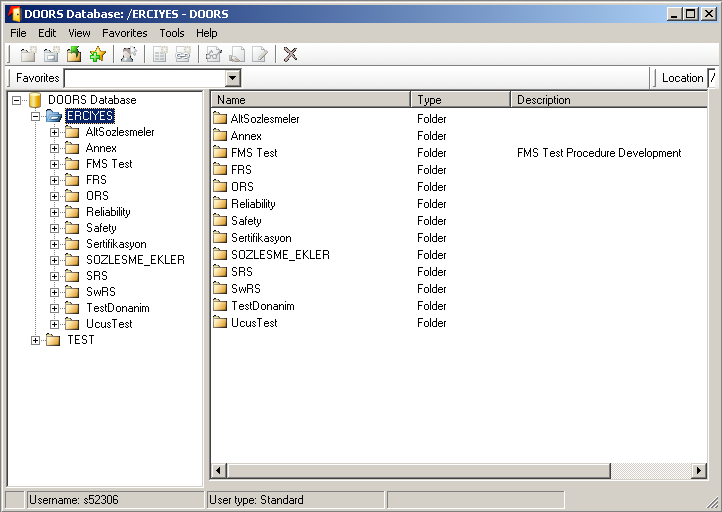
****

**N/A commenti ve Moderator commentlerine bir örnek**

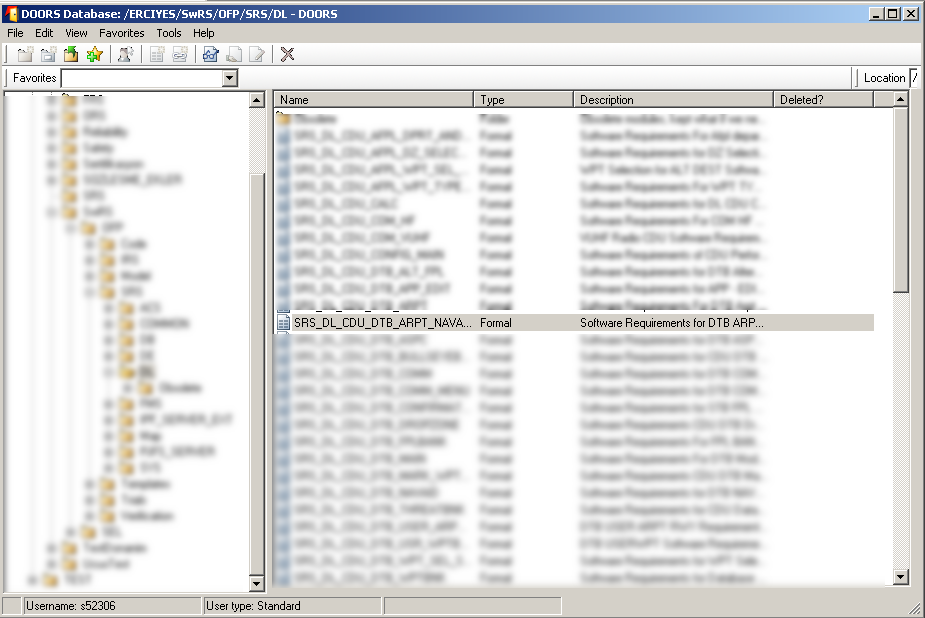
**Crucible Dashboard**

## 5.3- Telelogic DOORS Module

Bu modülde test kodu yazılacak olan partitionların gereksinimleri bulunmaktadır. SRS, IRS, OFP, FRS gibi gereksinim paketleri buradadır. Bu gereksinim paketlerinin altında partitionlara ait küçük gereksinim paketleri bulunmaktadır. Benim çalıştığım kısımda yazılım gereksinimleri vardı. Basitçe anlatırsak; sistem mühendislerinin yazdıkları gereksinimler daha sonra yazılıma uygun olacak şekilde yazılım mühendisleri tarafından yazılım gereksinimleri haline getirilir. Doors kullanımı basit ara yüzü sayesinde kolaylıkla yapılabilir.

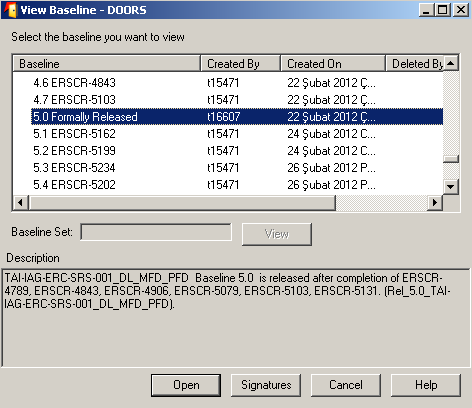
****

**Doorsa login olduktan sonraki paketlerin olduğu ekran**



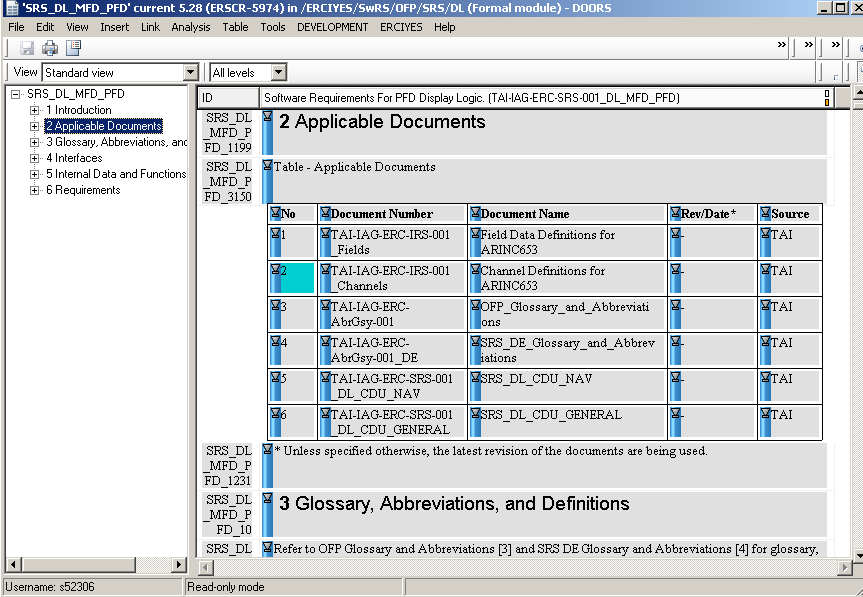
**SRS bölümüdeki farklı partitionlar için olan gereksinimlerin sıralandığı ekran**

Doors’a kullanıcı adı ve şifre ile giriş yapıldıktan sonra, gerekli gereksinim paketinin adresine gidilir ve o paket açılır. Daha sonra review edilen, paketin baseline’I (versiyonu) ile aynı olup olmadığına bakılır, eğer değilse File 🡪 Baseline 🡪 View seçerek istediğimiz versiyonu görüntüleyebiliriz. Var olan test caseleri en yüksek major baseline’a sync olmak durumundadır, buna gore test yazarları kodlarını değişen gereksinimlere uygun olacak şekilde kodlarını update ederler. (Major baseline: 1.0, 2.0 gibi sonunda sıfır olan kararlı küsüratı olmayan Formal olarak release edilmiş gereksinim versiyonlarına verilen addır.)

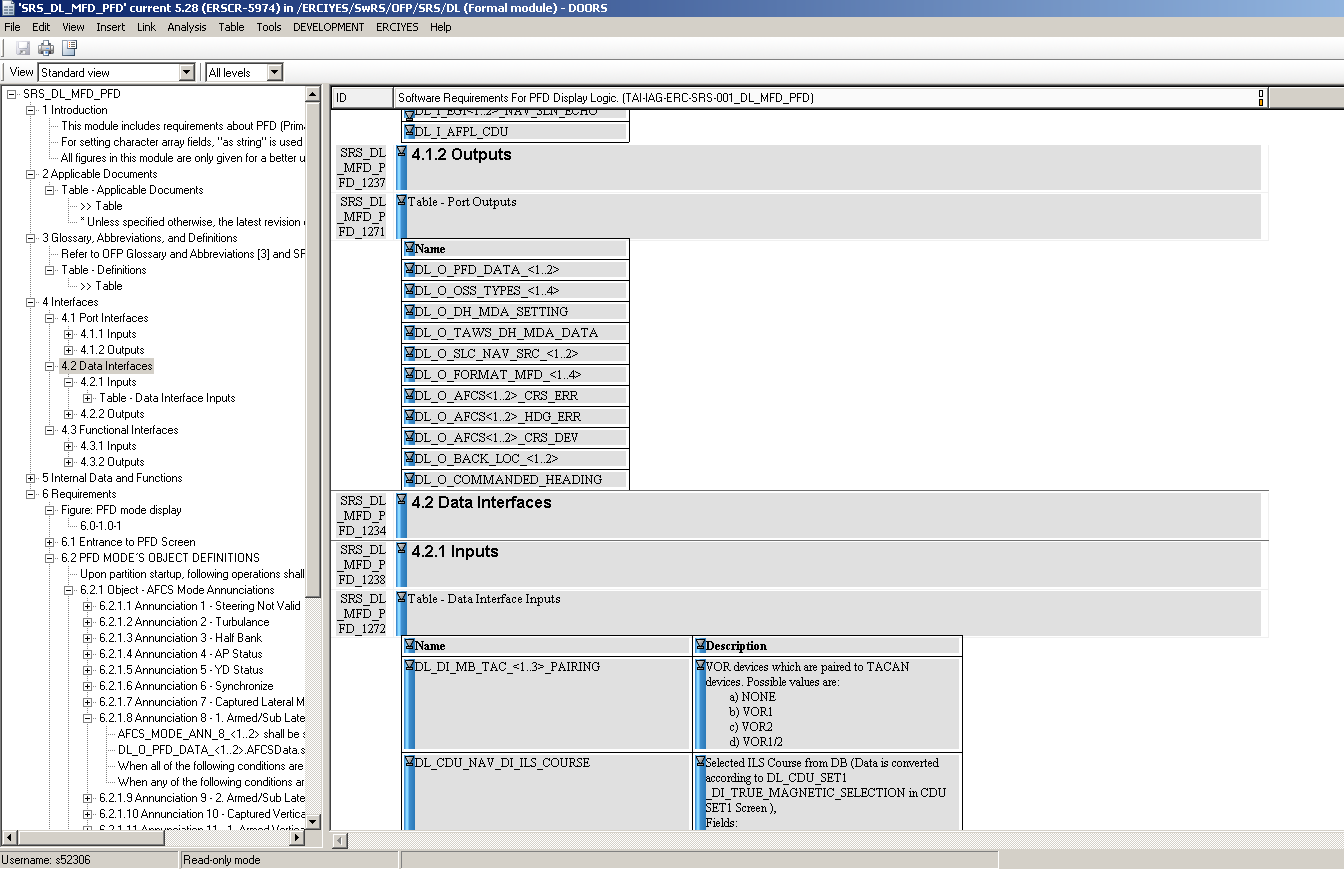
****

**Baseline seçme ekranı**

Baseline seçildikten sonra karşımıza alttaki gibi bir ekran çıkar. Introduction bölümünde o gereksinim paketinin kısa bir açıklaması yer alır. Applicable documents kısmında diğer paketlerle olan bağlantılarını ve yardımcı dökümanları görebiliriz. Üçüncü kısımda kısaltmaların anlamları ve bazı bilinmeyen sözcükler açıklanır. Interfaces kısmında input ve output portları, bunların detaylı bilgileri yer alır(altta bir görüntüsü mevcut). Beşinci kısımda internal datalar ve fonksiyonlarla ilgili açıklamalar ve gerekli bilgiler yer almaktadır (hangi modları olabildiği, default değerleri vs). Daha sonra esas işlemlerimizin yapıldığı ve gereksinimlerin bulunduğu kısım vardır.

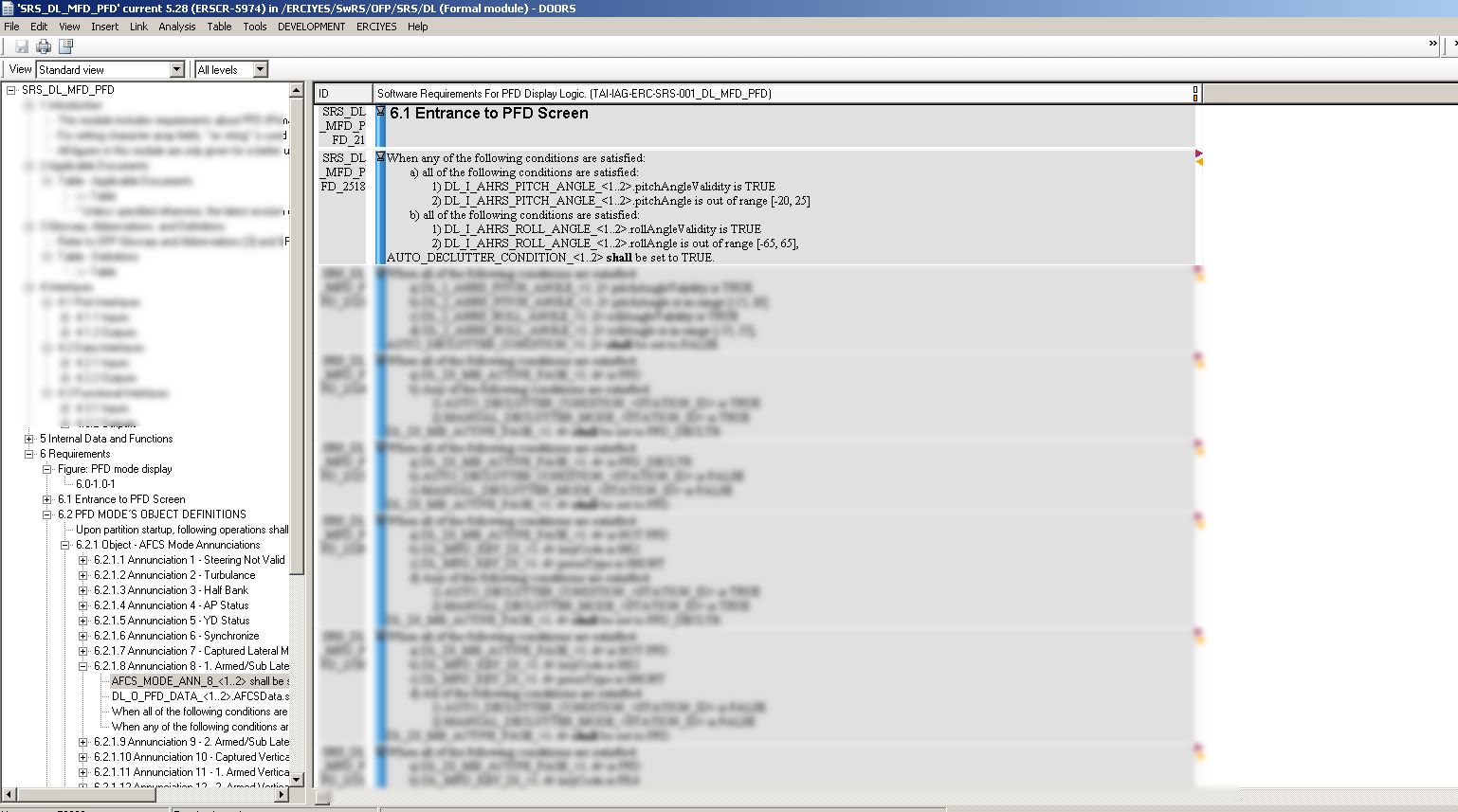
****

**Genel Gereksinim Ekranı Görüntüsü (ilk açıldığında)**

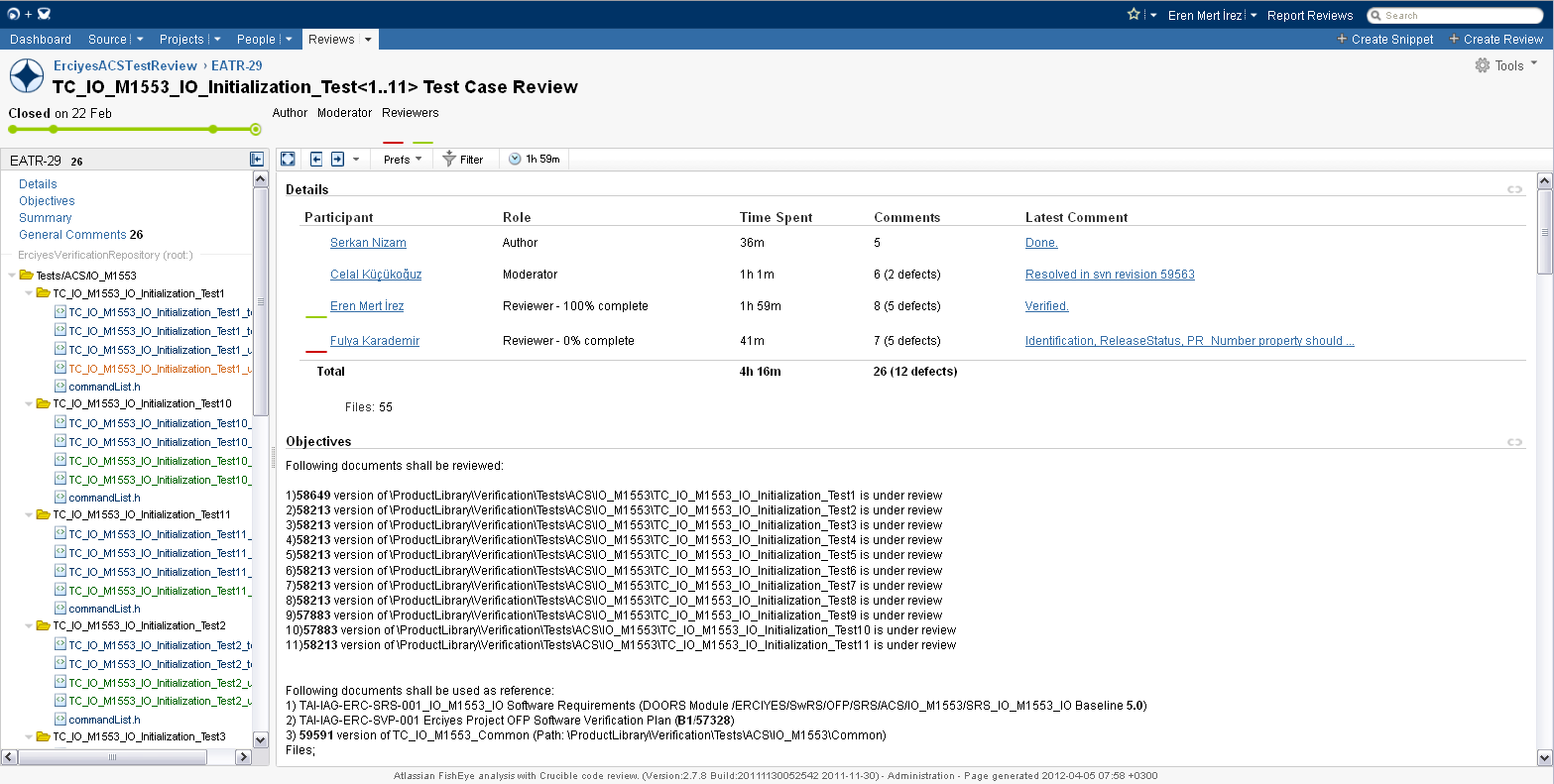
****

**Data Interfaces kısmındaki Input – Output portları ve açıklamaları**

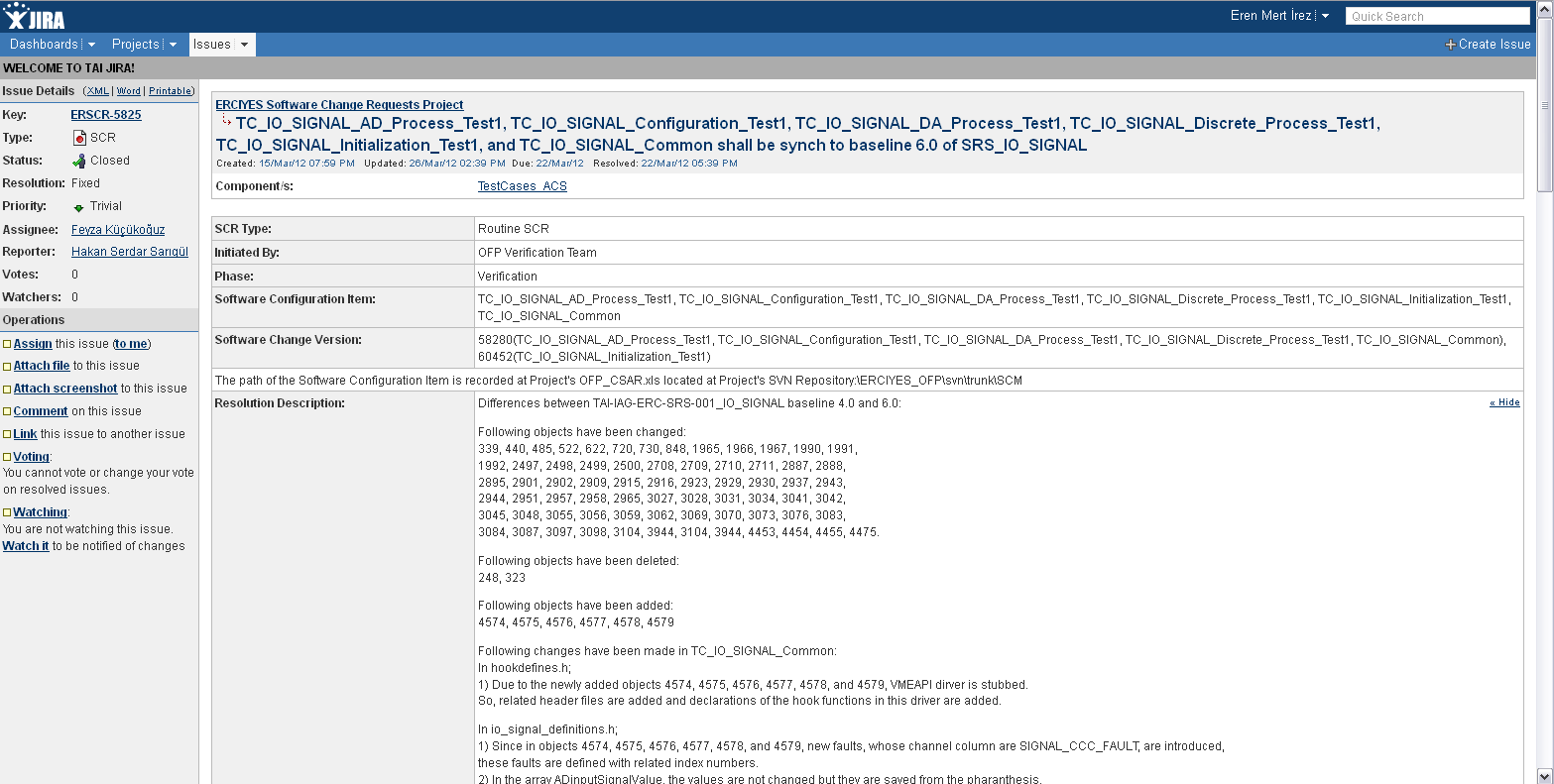
Bu gereksinim kısmında test kodu yazarları, gereksinim numaralarını trace edecek şekilde testlerini yazarlar. Reviewer de bu tracelere bakarak, checklist maddelerine gore bu gereksinimin complete(tam) ve correct(doğru) test edip edilmediğini kontrol eder. Ayrıca Channels ve Fields dökümanları adı verilen dökümanlar da incelenip, bu gereksinimlerde kullanılan port ve fieldların, rangelerinin, yönlerinin, port tiplerinin, okunması için gereken zamanlarının vs. doğru set edilmiş olması gereklidir. Reviewer ayrıca bunları da kontrol eder, hatalı gördüğü yerleri Crucible’da defect olarak, gerekli checklist maddesine gore belirtir.



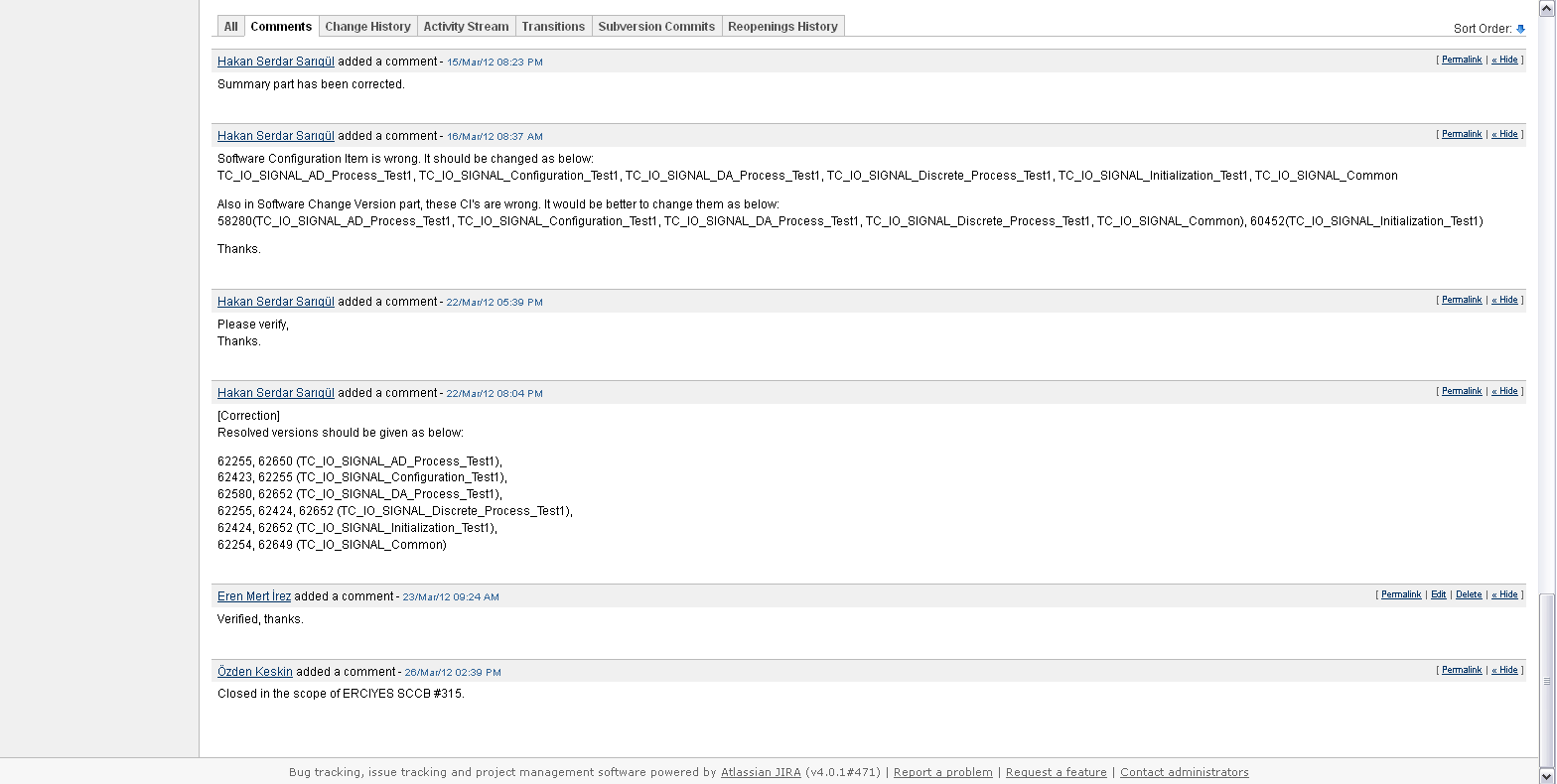
**Gereksinim Ekranı ve Gereksinimlerin görünümü, (sağdaki oklarla trace izlenebilir)**

****

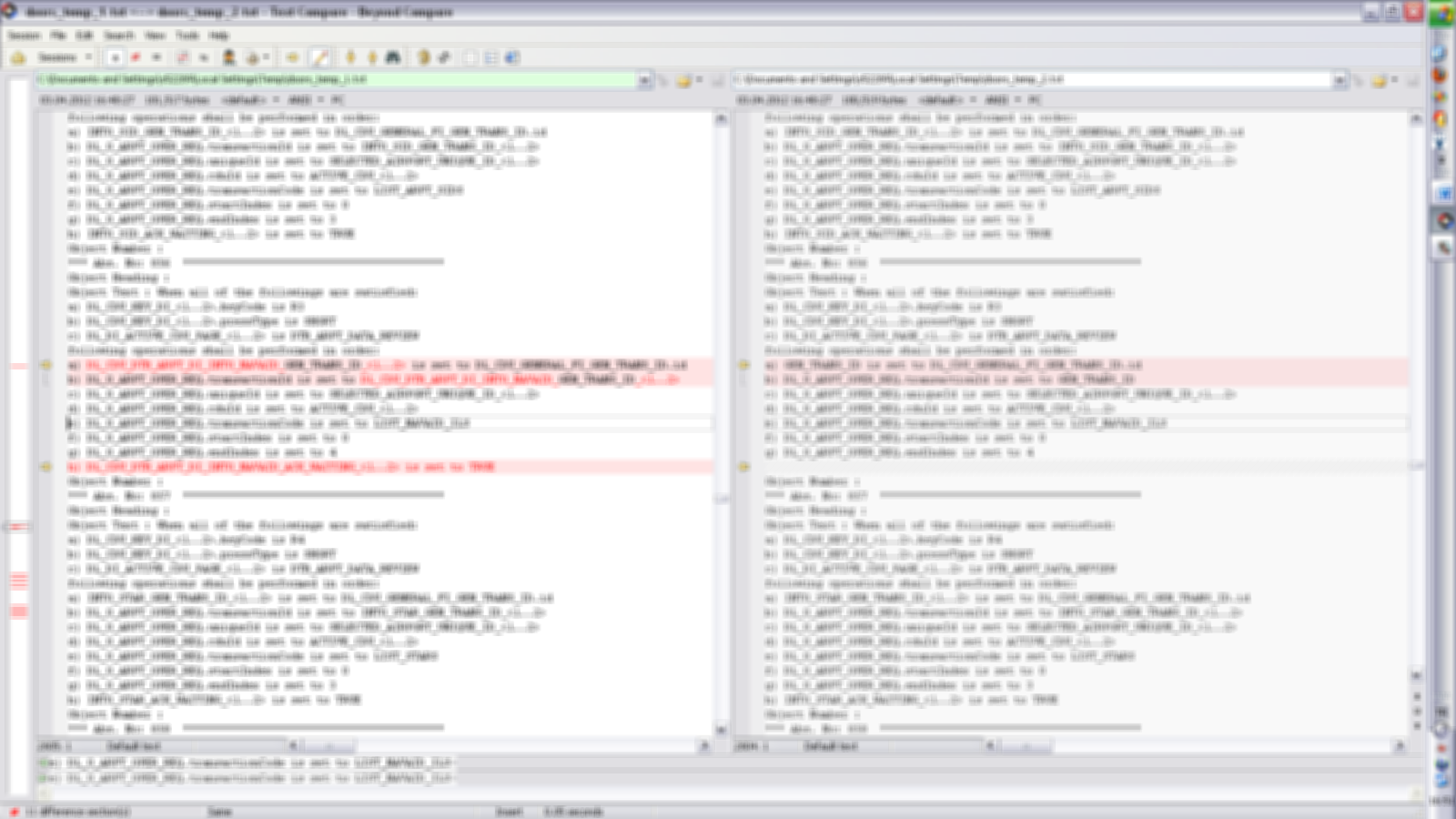
**Fotoğraf 1: Review’in Crucible üzerindeki görüntüsü.**

****

**Fotoğraf 2: SCR’ın JIRA üzerindeki görüntüsü.**

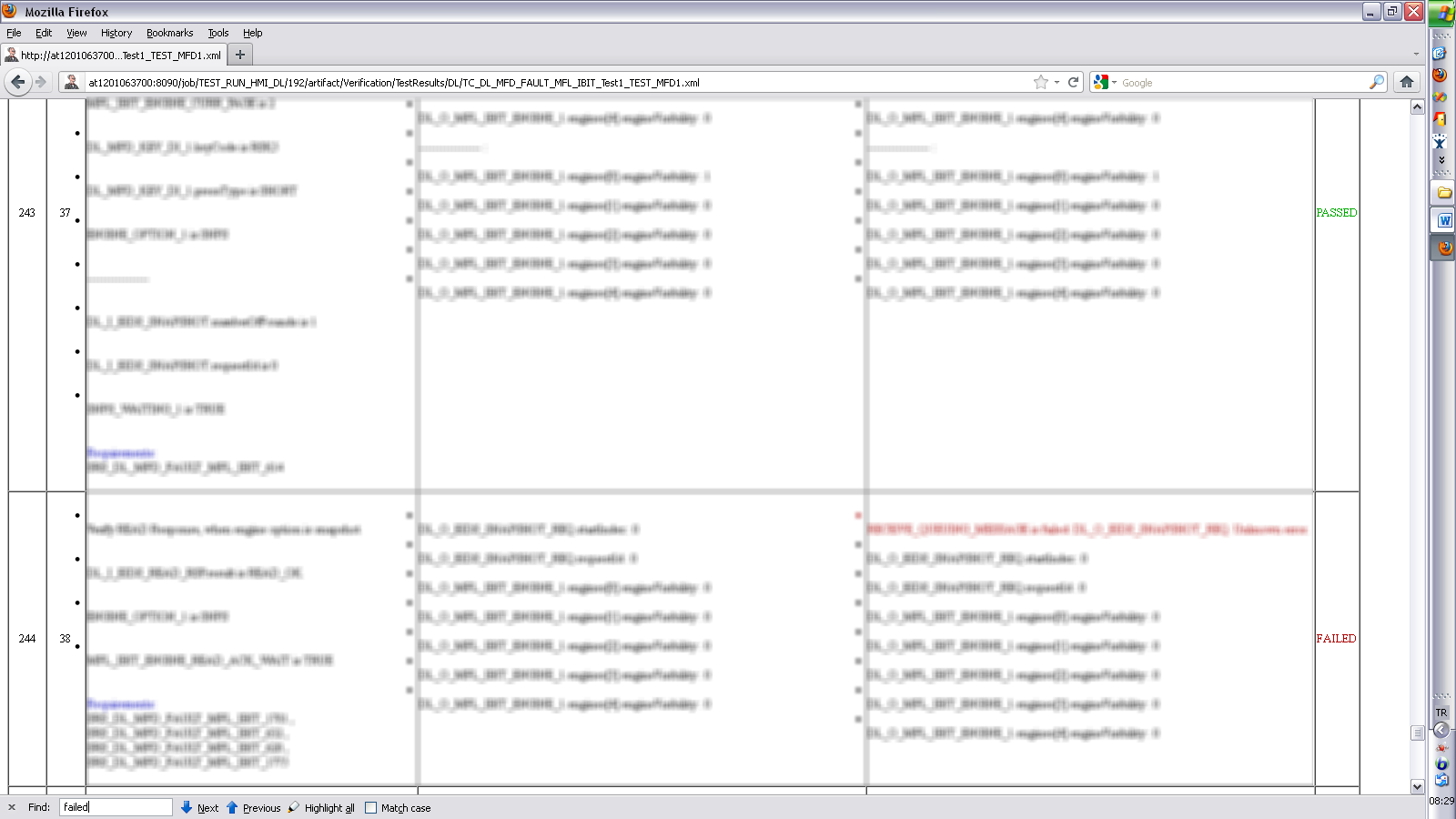
****

**Fotoğraf 3: Tarafımdan onaylanan bir SCR’ın görüntüsü.**

****

**Fotoğraf 4: İki gereksinim versiyonu arasındaki fark.**

**(Kod kısımlarının gizliliği sebebiyle kasıtlı olarak bulanıklaştırılmıştır.)**

****

**Fotoğraf 5: Hudson üzerinde hatalar ve görüntüleri.**

**(Kod kısımlarının gizliliği sebebiyle kasıtlı olarak bulanıklaştırılmıştır.)**