UNICASE

... kapsamlı bir CASE* aracı

* http://en.wikipedia.org/wiki/Computer-aided_software_engineering

Neden UNICASE?

- Yazılım geliştirme projelerinde yazılım mühendisliği modelleri merkezi bir yerde ve versiyon kontrolü yapılarak tutulmalıdır.
- Bahsedilen modeller gereksinimlerden, use-case ve UML modellerine, organizasyon modellerine kadar değişebilir.
- UNICASE ile tüm bu modeller birbiri ile ilişkilendirilebiliyor. Böylece yazılım geliştirme sürecini takip etmek ve yönetmek kolaylaşıyor.

… Neden UNICASE?

- Örneğin bir işlevsel gereksinimi gerçekleştirecek görevi tanımlayıp bu görevi ilgili gereksinime bağlayabiliyoruz. (Mesela “Kullanıcı yeni bir kural ekleyebilecek” işlev gereksinimini “Kural eklemenin gerçekleştirilmesi” görevine bağlayabiliyoruz. (Bu görevi kim yapacak vs. de yazılabiliyor.))
- Bu bağlantılar modellerin bir işbirliği içinde düzenlenmesini sağlıyor. (Mesela bir aktörün ismini değiştirdiğimizde aktörün ismi bulunduğu tüm modellerde değişiyor. Bu modellerden doküman elde ederken işlem otomatik olduğu için elle tek tek değiştirmeye gerek kalmıyor.)

... Neden UNICASE?

- Gereksinim analizi gibi dokümanlar müşteriye sunulacağı zaman UNICASE ile varolan modellerden model-tabanlı bir doküman oluşturmak mümkündür.
- Gereksinimlerin belirlenmesi, çözümleme yapılması, sistem ve nesne tasarımının yapılması, proje modeli üzerinde iterasyon planı, organizasyon yapısı, rational yönetim sadece bir araçla yapılabiliyor.
- Bu araç açık kaynaklı ve ücretsiz.
- ? Nightly build üstünden çalışıyor .

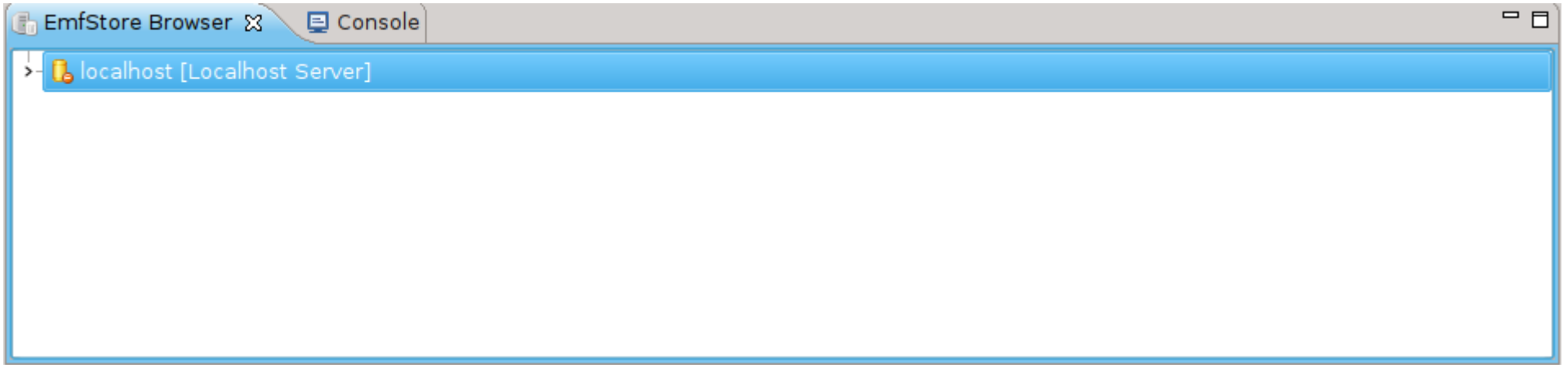
UNICASE Nedir?

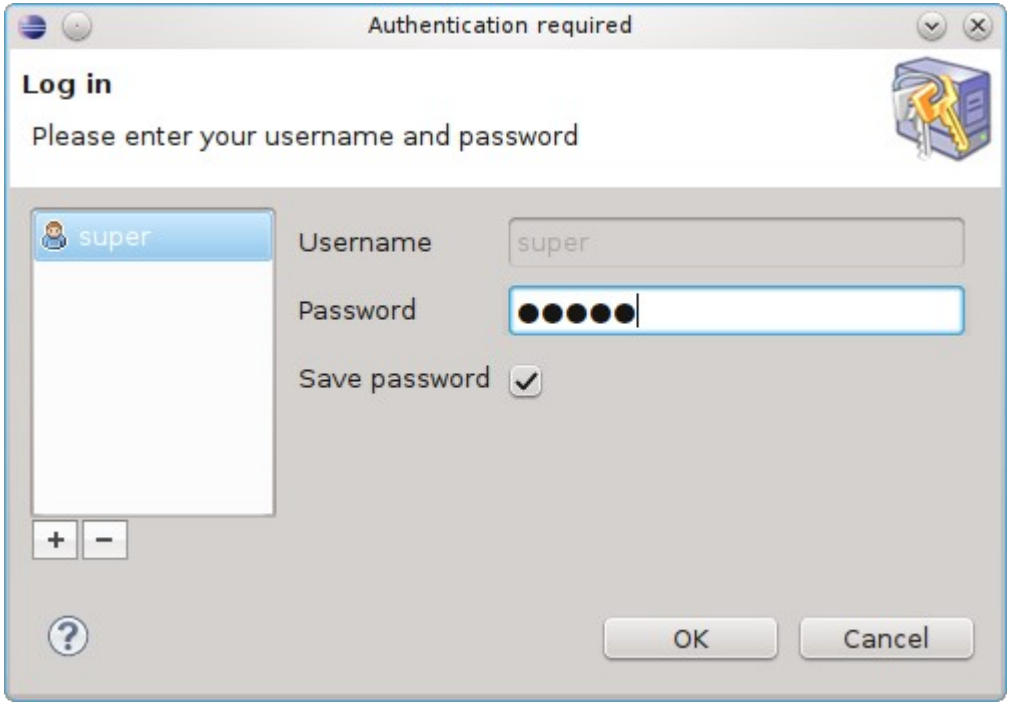
- UNICASE (Unified CASE) farklı geliştirme aktivitelerini birleştiren bir araçtır:
  - Gereksinimler
  - UML modelleri
  - Görev takvimi
  - Hata bildirimi
  - Toplantı yönetimi
  - ...

... UNICASE Nedir?

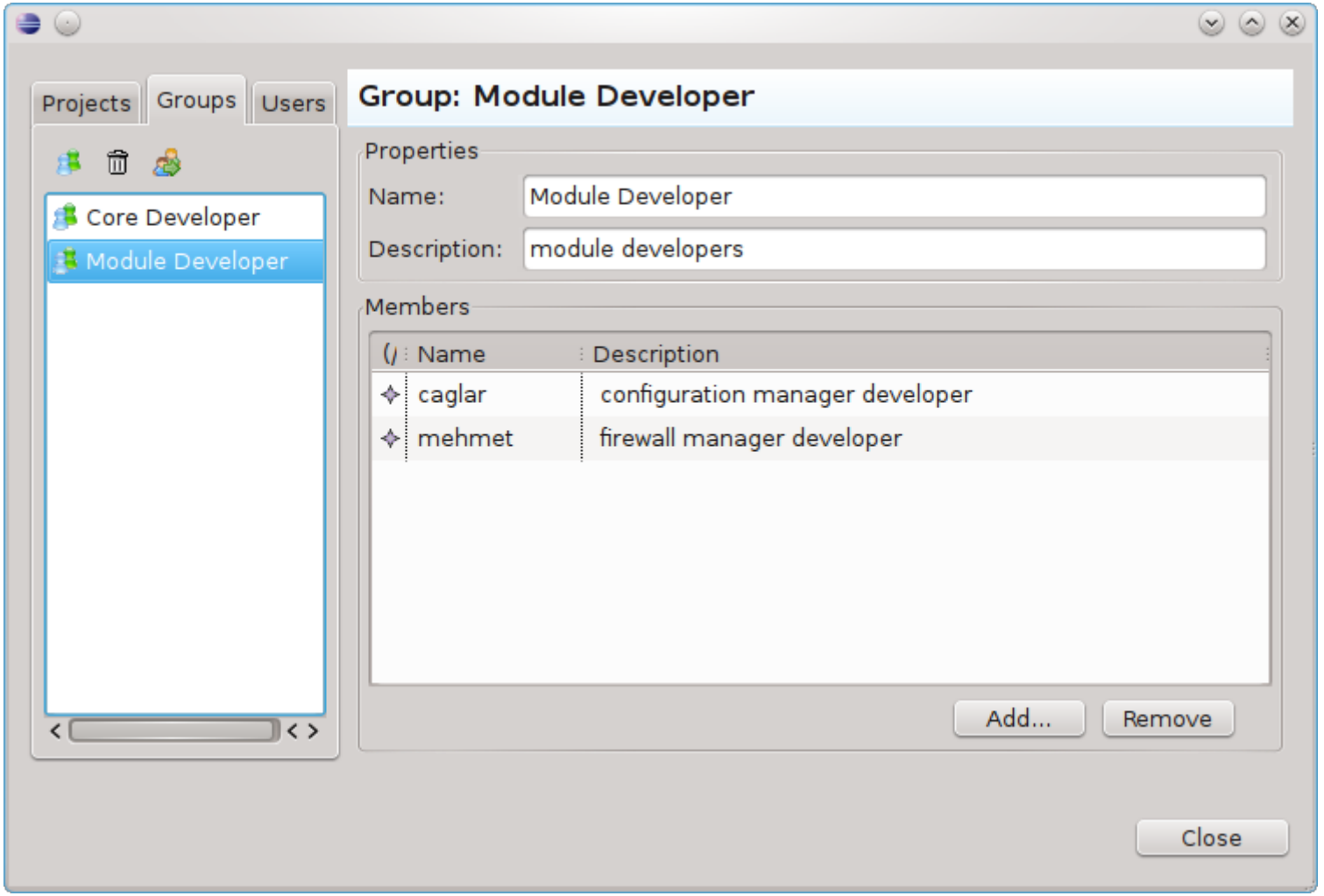
- UNICASE modelleri EMFStore denilen merkezi bir sunucuda tutulur.
- EMFStrore SVN gibi versiyon denetimi sağlar ama modeller için özelleşmiştir.
- UNICASE Eclipse tabanlı çalışır. (EMF ve GMF gerektirir.)
- Lisansı: Eclipse Public License v 1.0 (EPL)

EMF Store Browser & Login

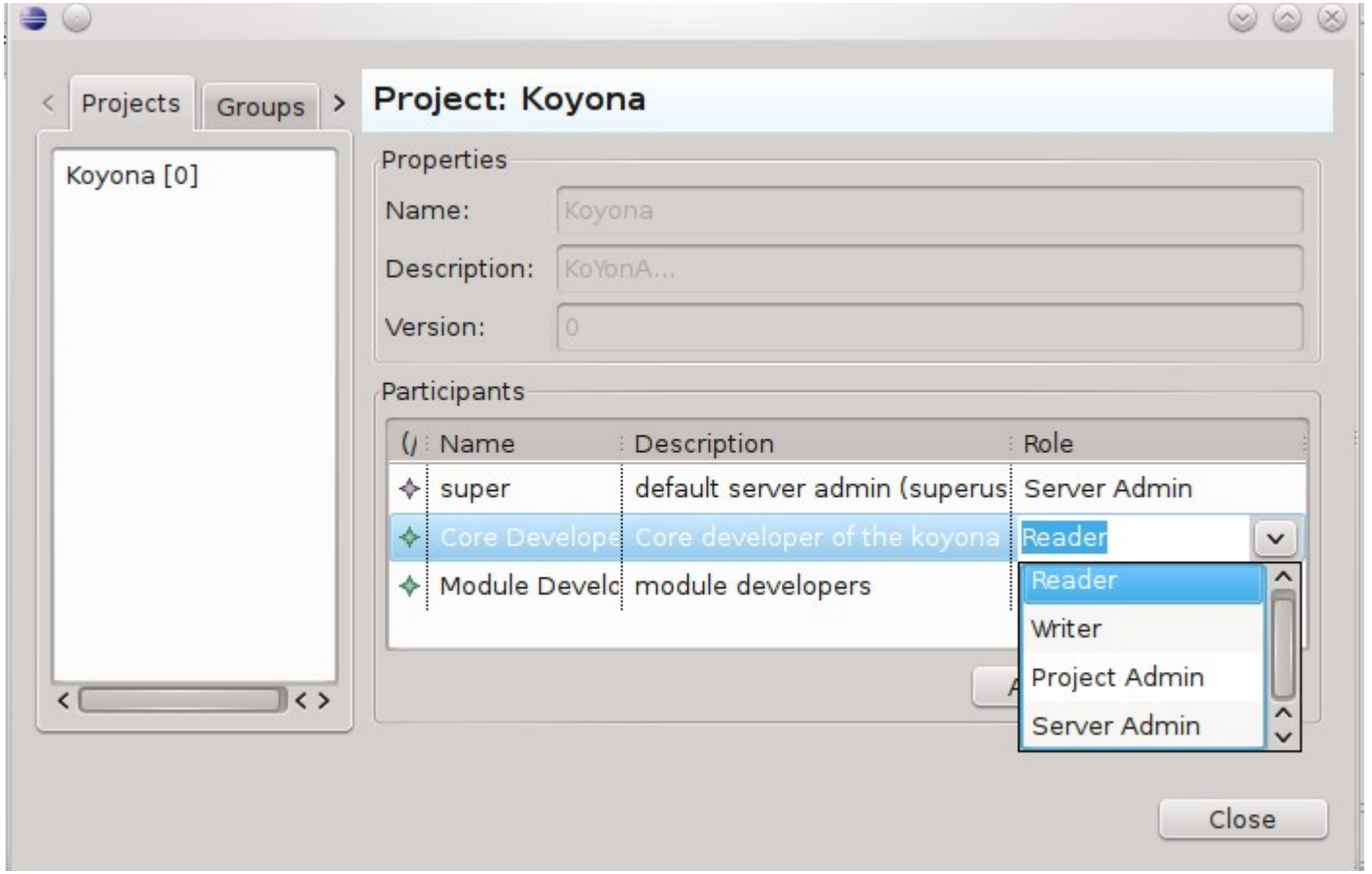




Proje / Grup ve Kullanıcılar



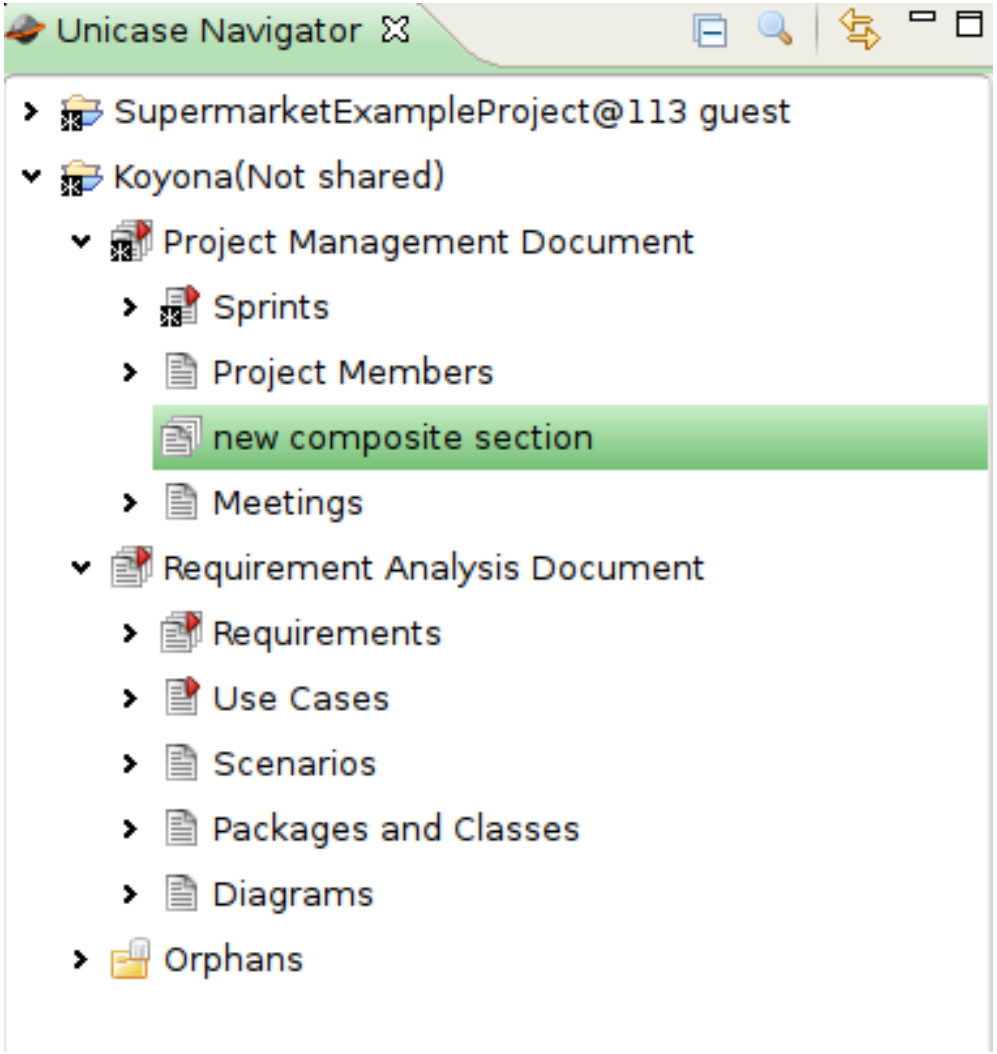
Grupların proje üzerindeki hakları



Kullanım

- Sunucu üstünde yeni bir proje oluşturuyoruz. Geliştiriciler sunucuya bağlanıp proje listesini alıyorlar. İlgili projede okuma/yazma gibi haklar tanımlandığını varsayıp o projeyi checkout (yerele kopyalama) etmeleri beklenir.
- Sunucudan çekilen projenin bir iskeleti olmalıdır. Bu iskeleti yönetici hazırlar. Geliştiriciler kendi geliştirdikleri alanları bu iskeletten bulup düzenlerler.
- İskelet hazırlama işi Leaf ve Composite denen kesimler tanımlanarak gerçekleştirilir. Composite(bileşke) kesimler composite ve leaf kesimleri içerebilir. Leaf kesimler ise “Model Element” denen nesneleri içerirler. Model Element bir diagram, aktör, hata, ya da bir görev tanımı olabilir. Bunlar ne anlama geliyor örneklerle görelim.

Navigator

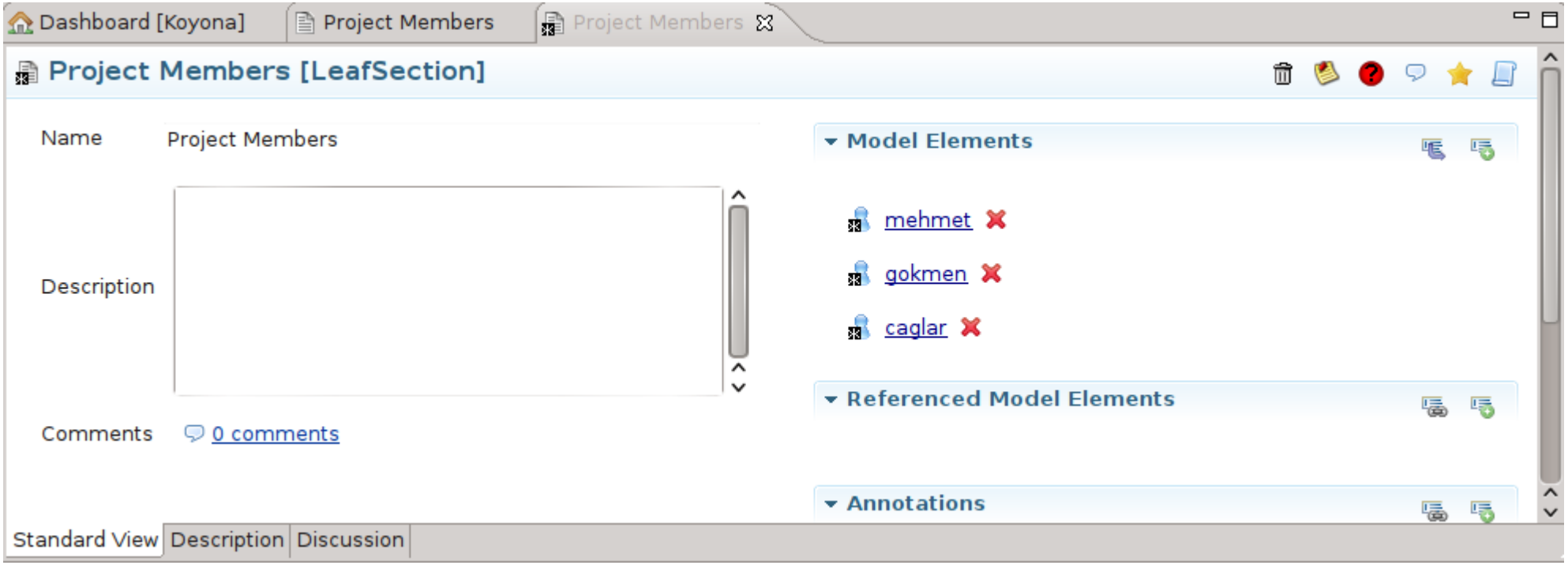

Unicase Navigator
SupermarketExampleProject@113 guest
Koyona(Not shared)
Project Management Document
Sprints
Project Members
new composite section
Meetings
Requirement Analysis Document
Requirements
Use Cases
Scenarios
Packages and Classes
Diagrams
Orphans

Proje Üyesi Ekleme

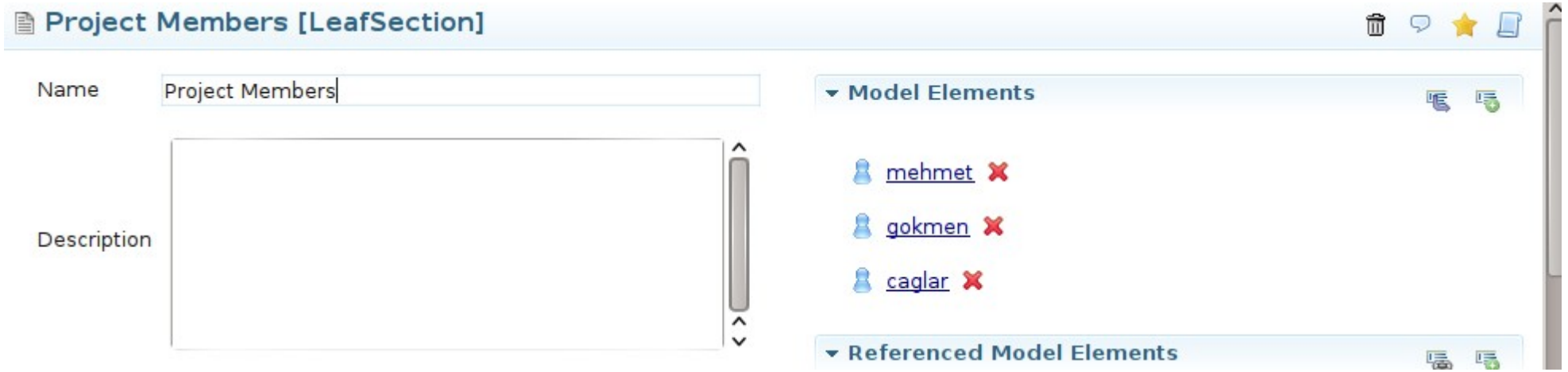
- Project Management Document diye bir composite section ekleyip bunun altına Project Members diye bir leaf section ekleyip proje üyelerini buraya yazıyoruz. Bu üyeler toplanıya çağrılmadan tutun da belirli bir görevin bu kişiye atanmasına kadar bir çok yerde kullanılıyor olacak.

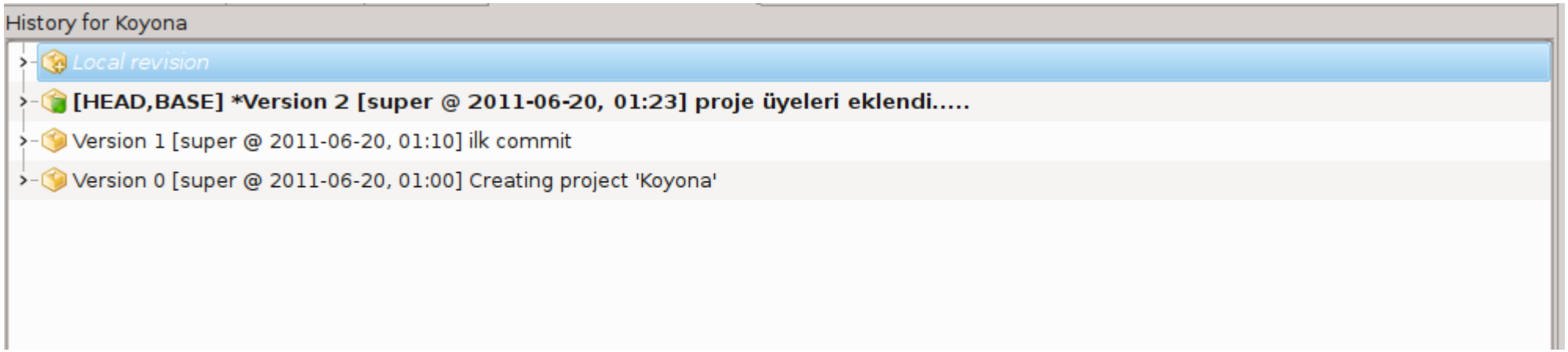
... Proje Üyesi Ekleme

- Kullanıcı ikonunun yanındaki siyah yıldız commit edilmediğini gösteriyor.

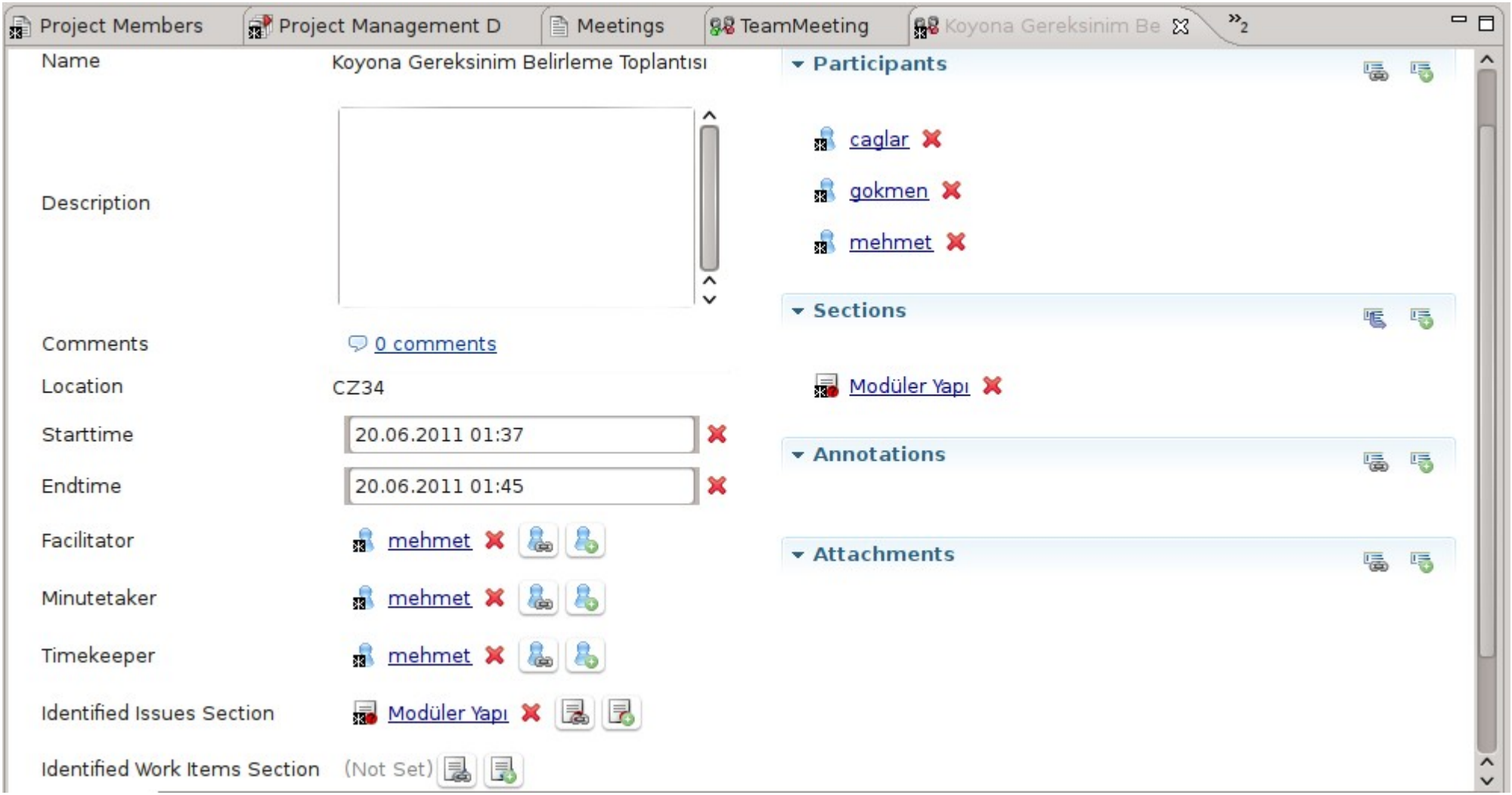


Commit sonrası ve History

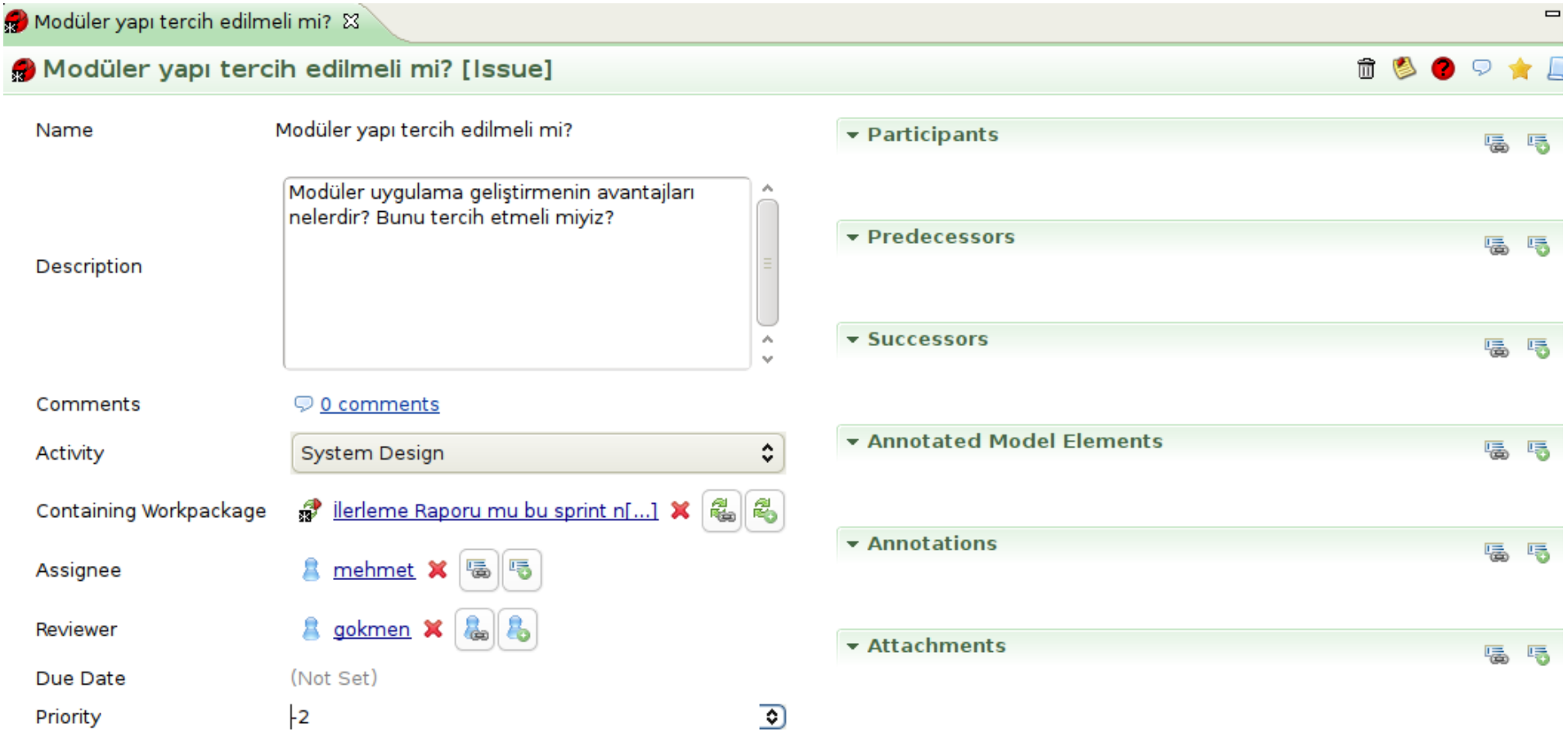




Toplantı Yönetimi

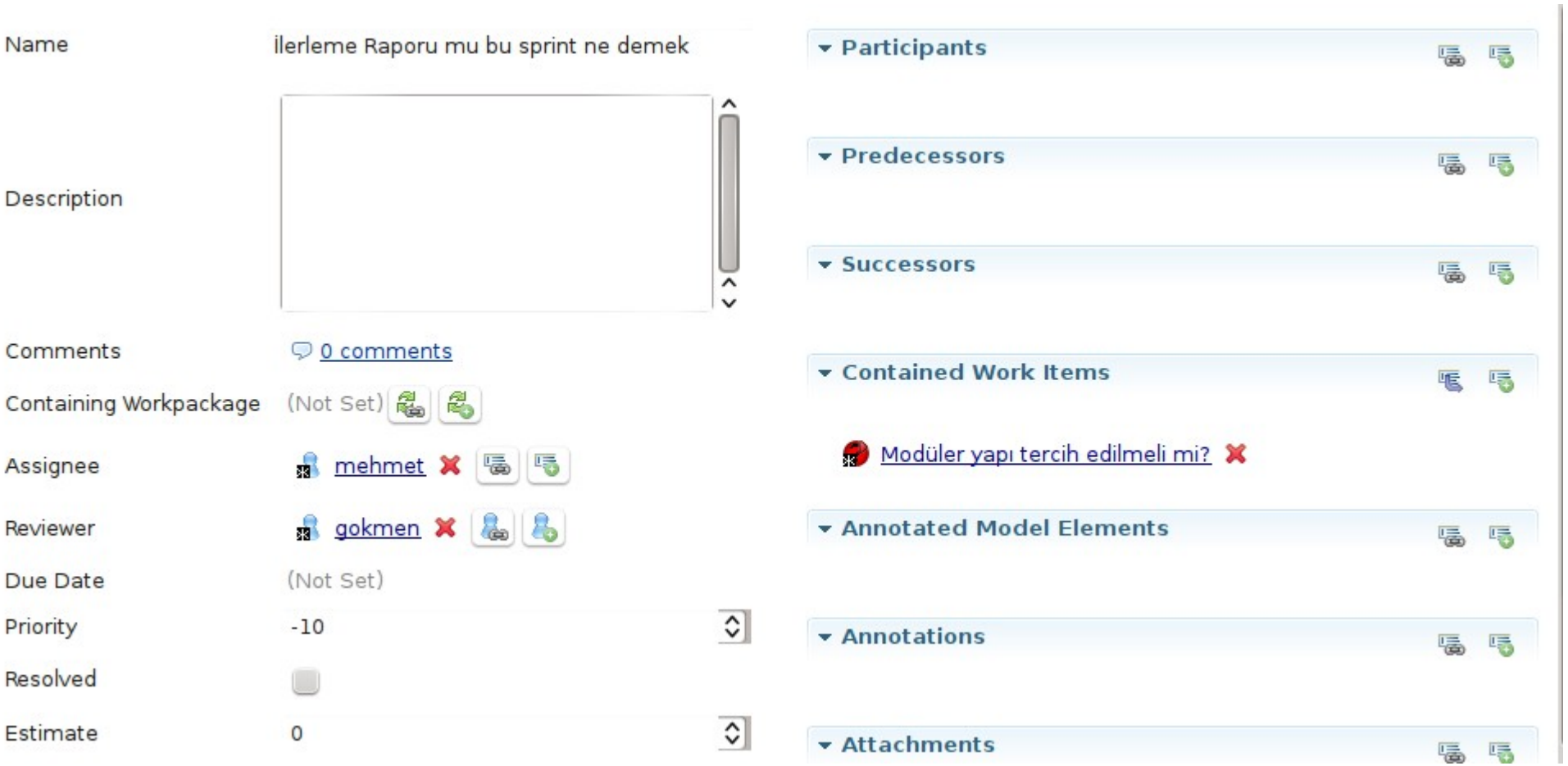


Issue Ekleme

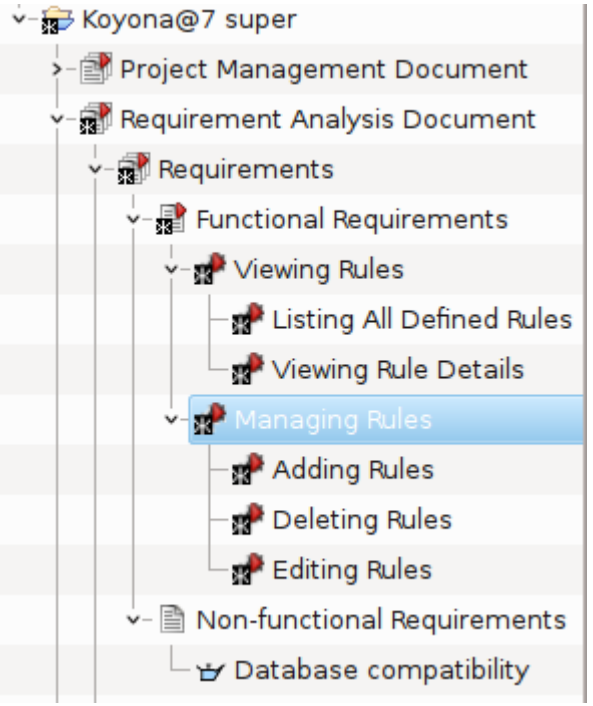


Sprint

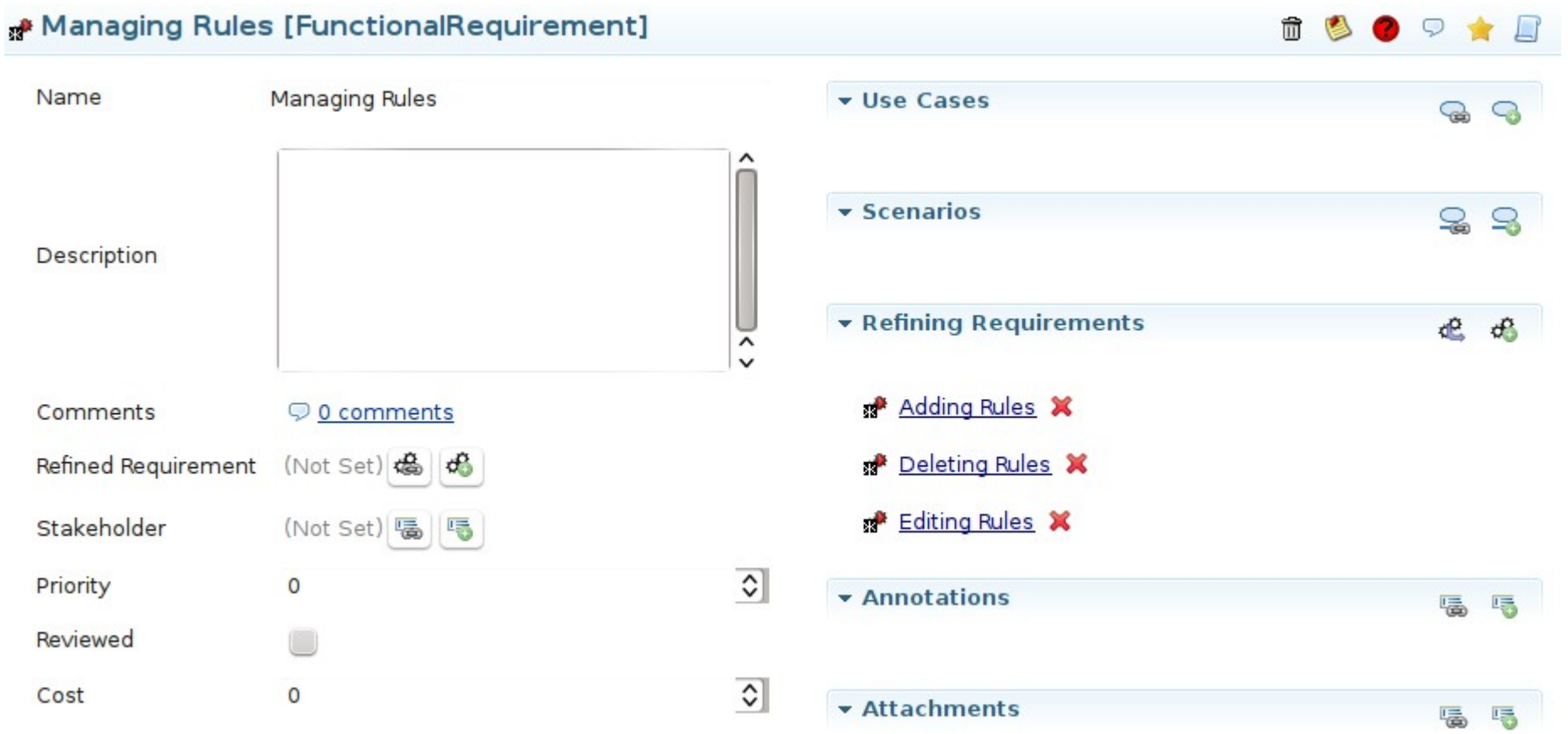
“Sprint; agile olarak geliştirilen sistemlerde belirli özellikleri içeren program parçalarının, yapılan işlerin demosu. Bir çeşit müşteri veya proje yönetimi gözden geçirmesi”



İşlevsel Gereksinim Ekleme

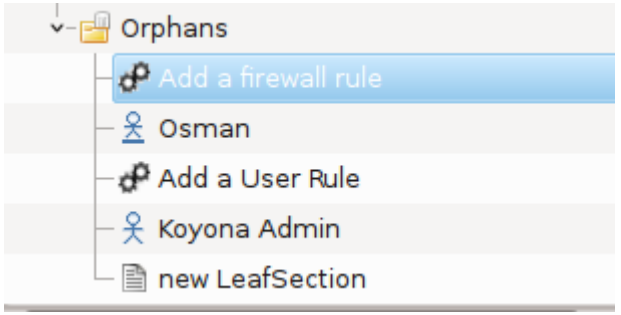


... İşlevsel Gereksinim Ekleme

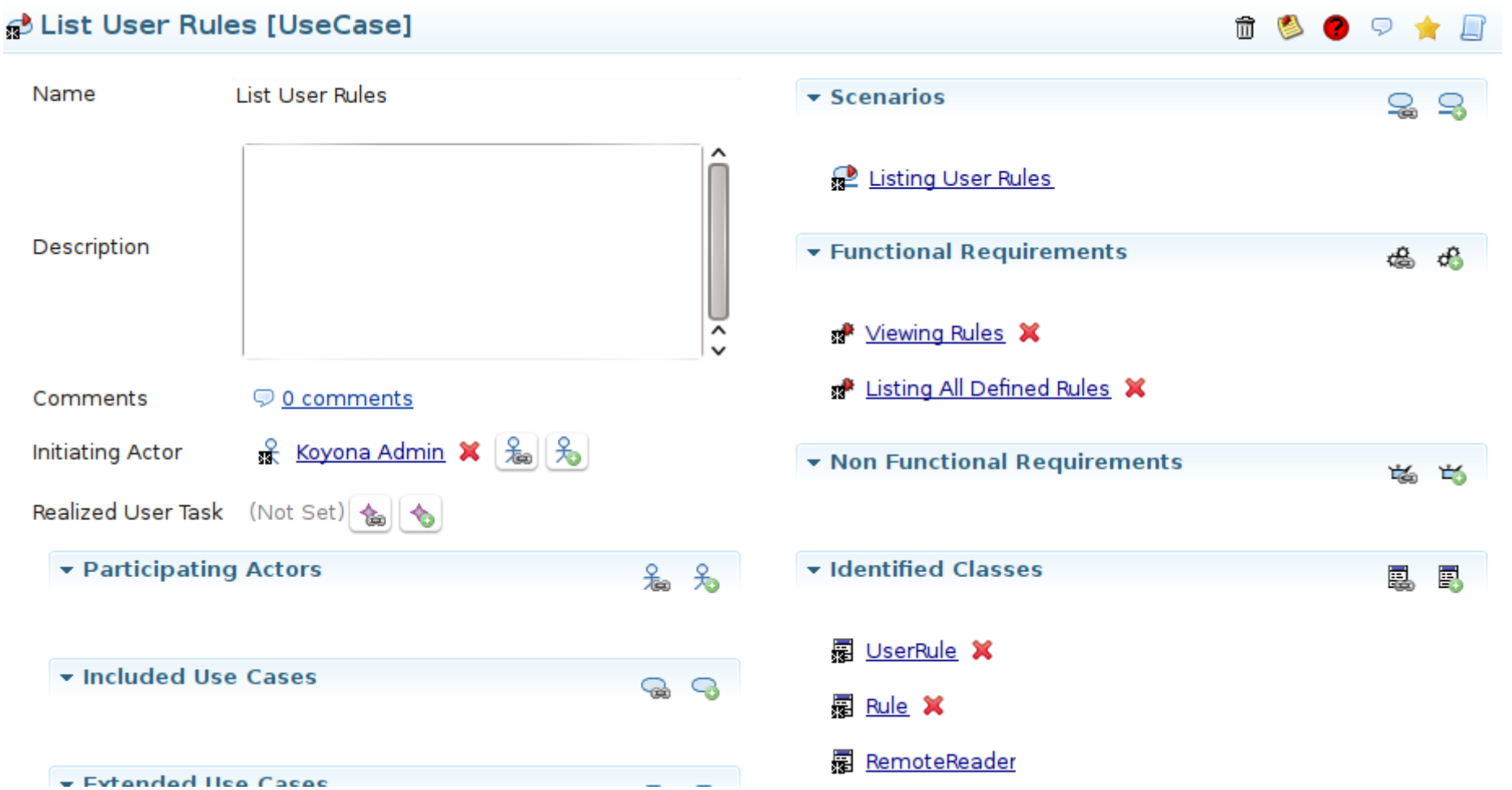


Orphans

Herhangi bir modelden silinen nesne sistemden doğrudan kaldırılmıyor; orphans dizini altında saklanıyor. Bunları tekrardan modellere dahil etmek mümkün.



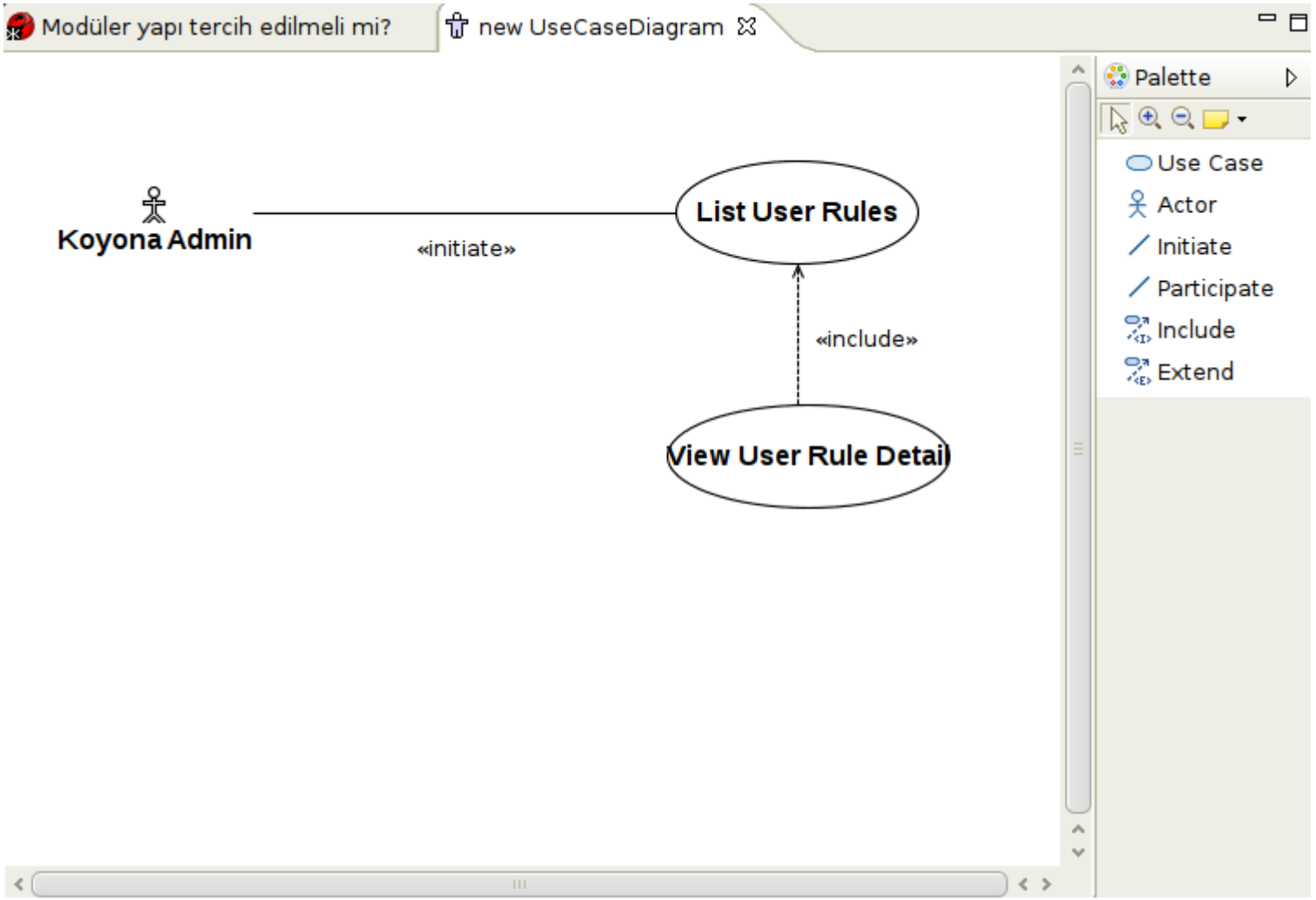
Use case


List User Rules [UseCase]
Name
List User Rules
Description
Comments
0 comments
Initiating Actor
Koyona Admin
Realized User Task
(Not Set)
Participating Actors
Included Use Cases
Extended Use Cases
Scenarios
Listing User Rules
Functional Requirements
Viewing Rules
Listing All Defined Rules
Non Functional Requirements
Identified Classes
UserRule
Rule
RemoteReader

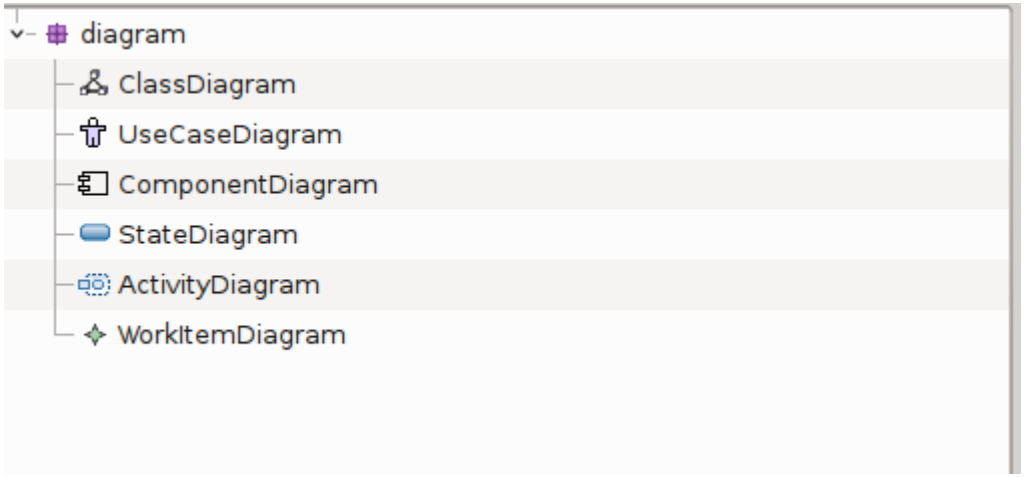
Use case için senaryo

Usecase diagram

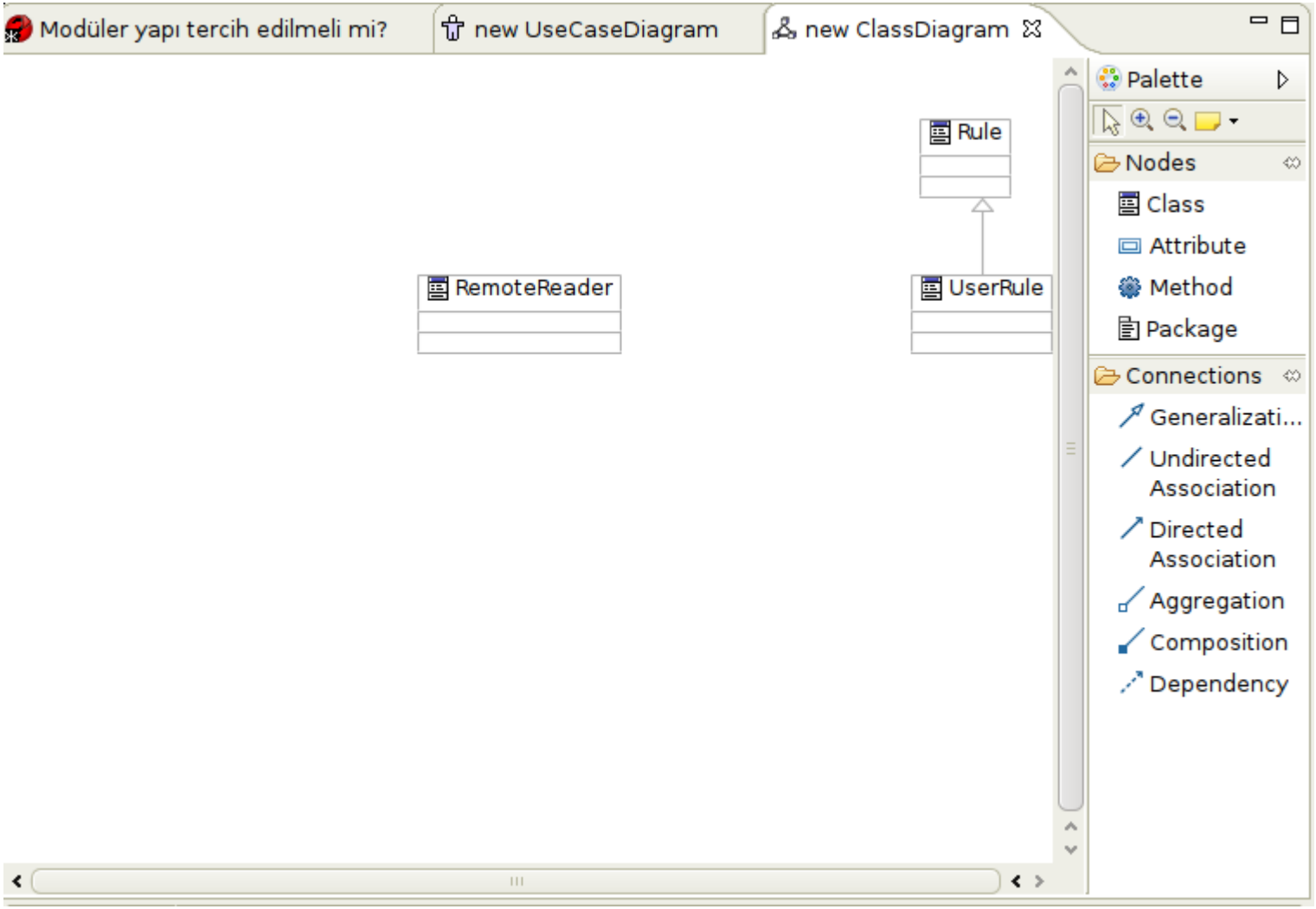
Tanımlanan nesneleri navigatordan sayfaya sürükleyerek diagram oluşturabiliyoruz.



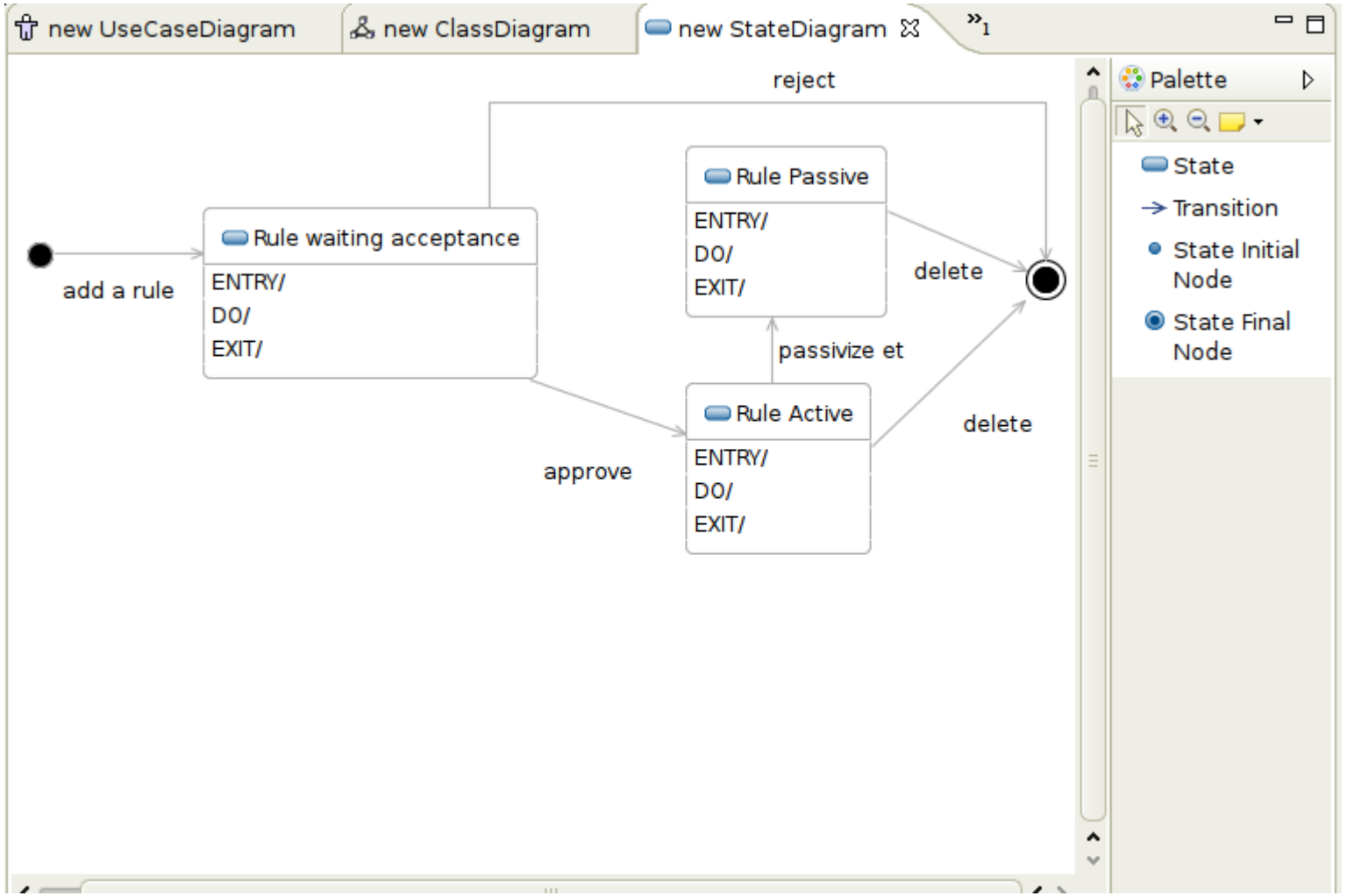
Diyagramlar



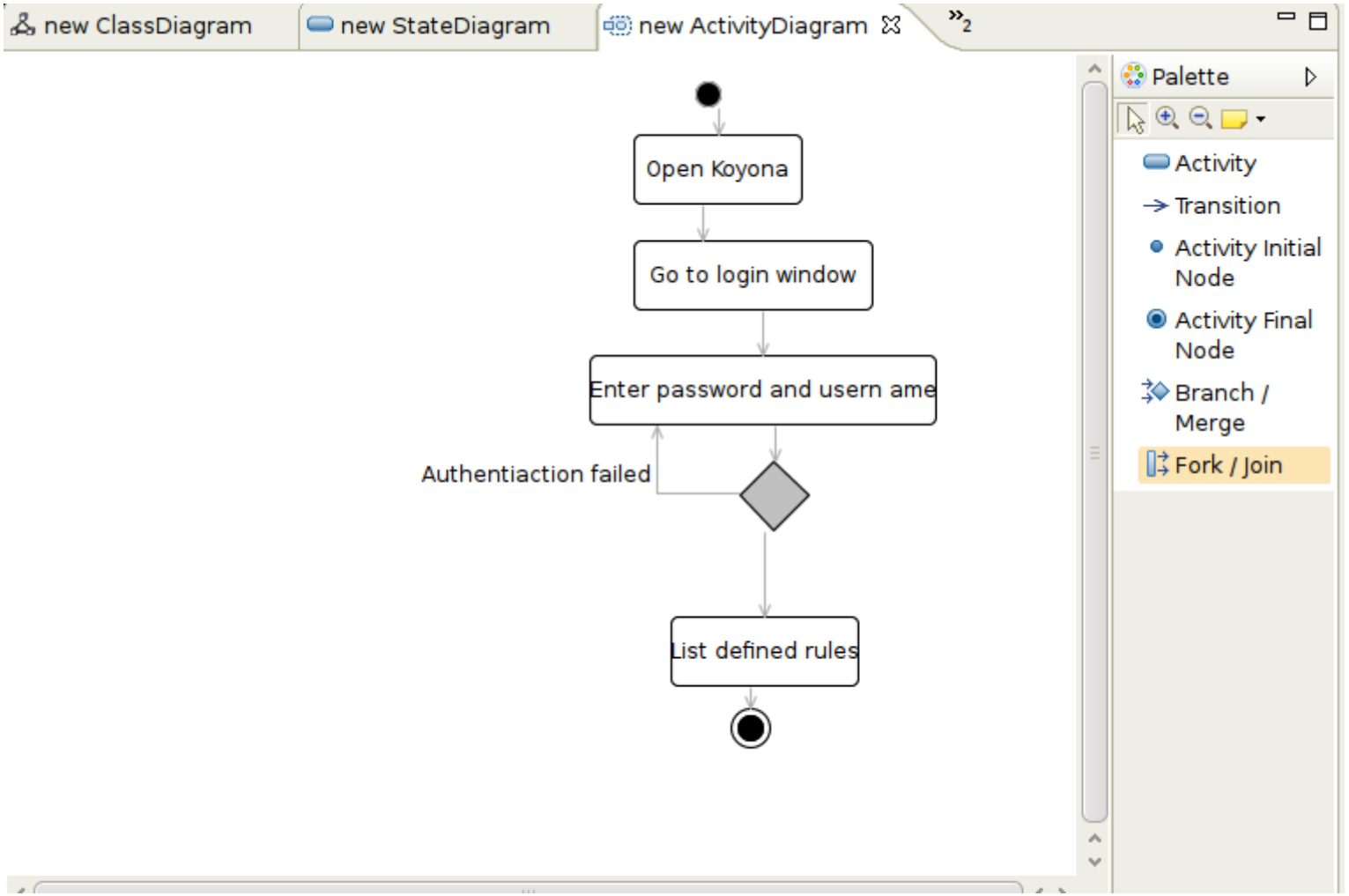
Class diagram



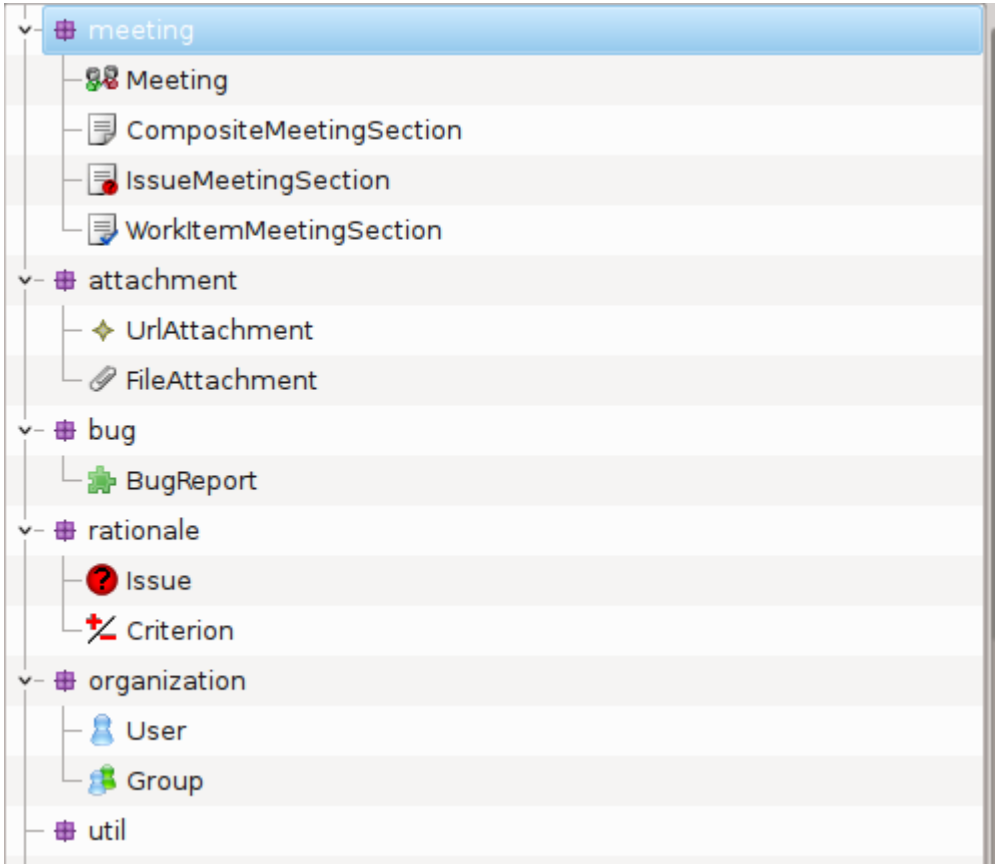
State diagram

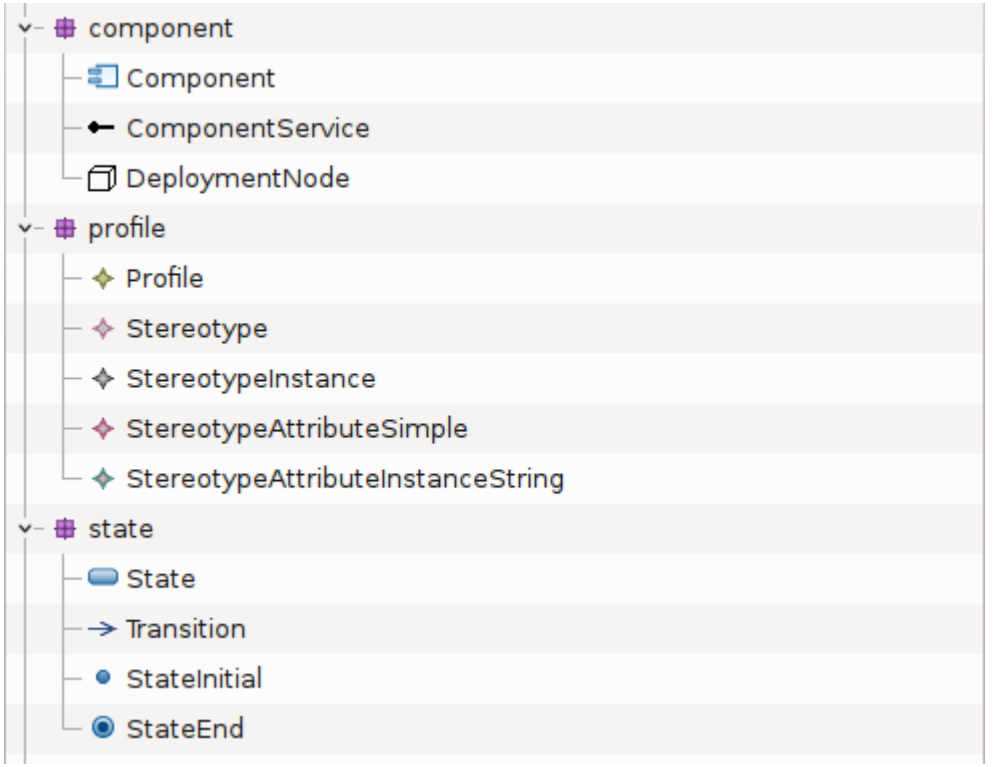


Activity diagram

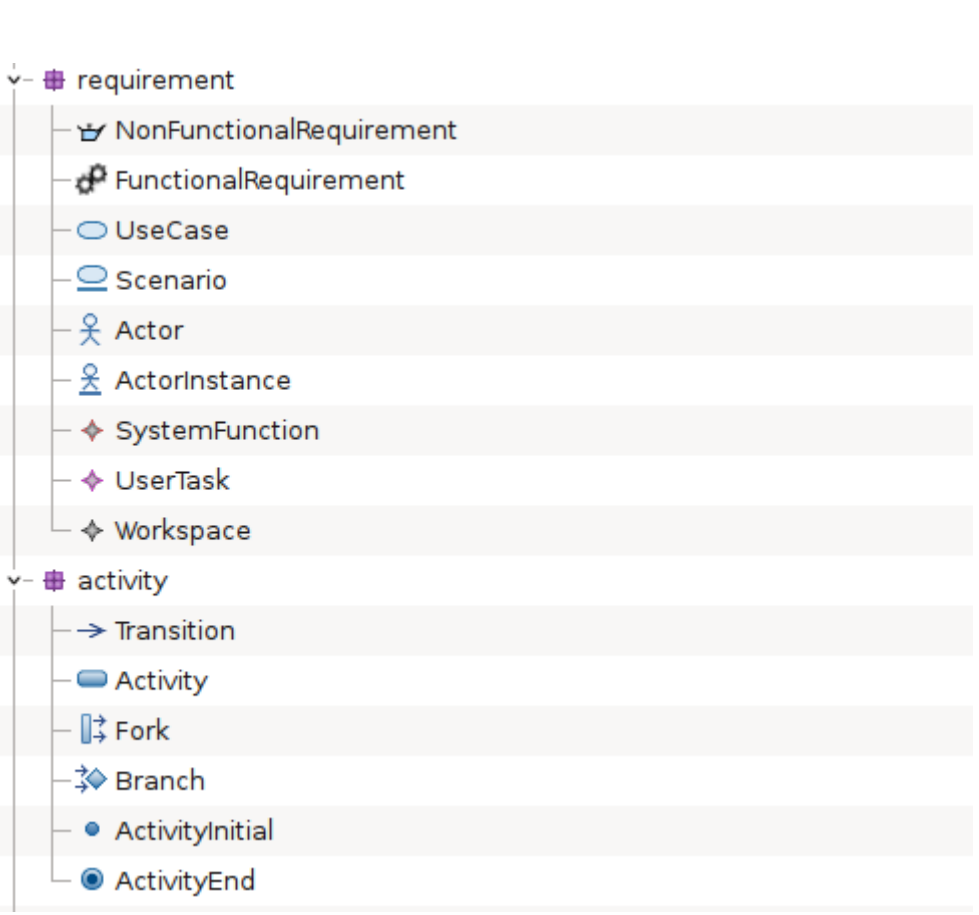


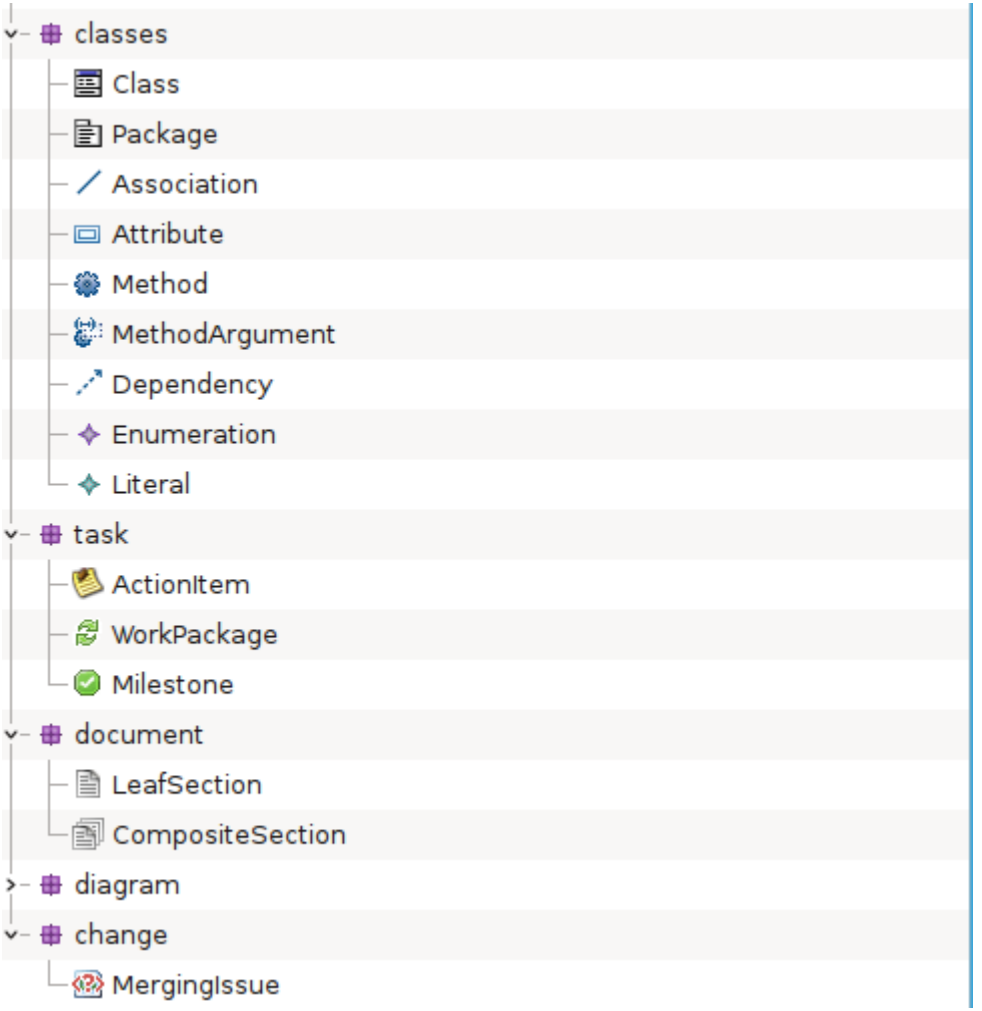
Model Elements





... Model Elements





Sequence & Collabration

- Bu diagramlar yok!!!
- *“Deployment diyagramı da yok. Ama bir şekilde component diyagramı ile birleştirebiliriz gibi görünüyor. (Component diyagramının içinde deployment node'u verilmiş)”*