Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ 313 Yazılım Mühendisliği

8. Gereksinimler için Modeller

Gereksinimler için Modeller

Bakış açısı analizi, senaryolar, use-case'ler vb. aracılığıyla gereksinimleri anlamaya çalışırken veya gereksinimleri analiz etmek ve belirtmek için **modeller** kullanılır.

Modeller, müşterinin anlayışı ile geliştiricilerin anlayışı arasında bir köprü sağlar.

Gereksinim analizi ve spesifikasyonların belirlenmesinde belirli bir görev için uygun aracın seçilmesi gerekir.

- Çeşitli araçlar ve teknikler seçilebilir.
- Bunlardan bazıları diğer derslerden aşina olduğunuz tekniklerdir.
- Her duruma uyan doğru bir teknik bulunmaz.

Modeller

Model, gerçekliğin basitleştirilmesidir

- Geliştirmekte olduğumuz sistemi daha iyi anlayabilmemiz için modeller oluşturuyoruz.
- Karmaşık sistem modelleri inşa ediyoruz çünkü böyle bir sistemi bütünüyle kavrayamayız.

Modeller gayri resmi veya resmi olabilir. Proje ne kadar karmaşıksa, resmi bir model o kadar değerli hale gelir.

BRJ

Modelleme İlkeleri

- Hangi modelleri kullandığımızın seçimi bir sorunu çözmek için hangi yolları izleyeceğimiz üzerinde etkilidir ve çözümü de şekillendirir.
- **Tek bir model yeterli değildir.** Her önemsiz olmayan sisteme, çoğu bağımsız modellerden oluşan küçük bir küme ile yaklaşılarak problem çözülür.
- Her model farklı hassasiyet seviyelerinde ifade edilebilir.
- Hassasiyet düzeyinden bağımsız olarak, iyi modellerin gerçeklikle bağlantılı olması önemlidir.

BRJ

Unified Modeling Language

UML, yazılım sistemlerini modellemek için standart bir dildir

- Gereksinimler ve uygulama arasında bir köprü görevi görür.
- Bir yazılım sisteminin tasarımını belirtmek ve belgelemek için bir araçtır.
- Süreç ve programlama dilinden bağımsız olması amaçlanmıştır, ancak özellikle nesne yönelimli program geliştirme için uygundur.

Rational Rose

Rational Rose, UML modelleri (diyagramlar ve belirtimler) oluşturmak ve yönetmek için IBM'e ait bir sistemdir.

Alternatifler:

- Modelio
- StarUML
- PlantUML

Modeller: UML'de Diyagramlar ve Spesifikasyonlar

UML'de bir model, bir **diyagram** ve bir **spesifikasyondan** oluşur.

- **Diyagram**, genellikle köşelerin (nesnelerin) ve yayların (ilişkilerin) bağlantılı bir grafiği olarak işlenen bir dizi öğenin grafiksel temsilidir.
- Her diyagram, diyagramın temsil ettiği modeli daha ayrıntılı olarak **belirten teknik belgelerle** desteklenir. Ancak derste sadece diyagramlara bakacağız.

Spesifikasyonu olmayan bir diyagram çok az işe yarar.

Veri Akışı Modelleri

Bir sistem üzerinden veri akışını göstermek için resmi olmayan bir modelleme tekniği.

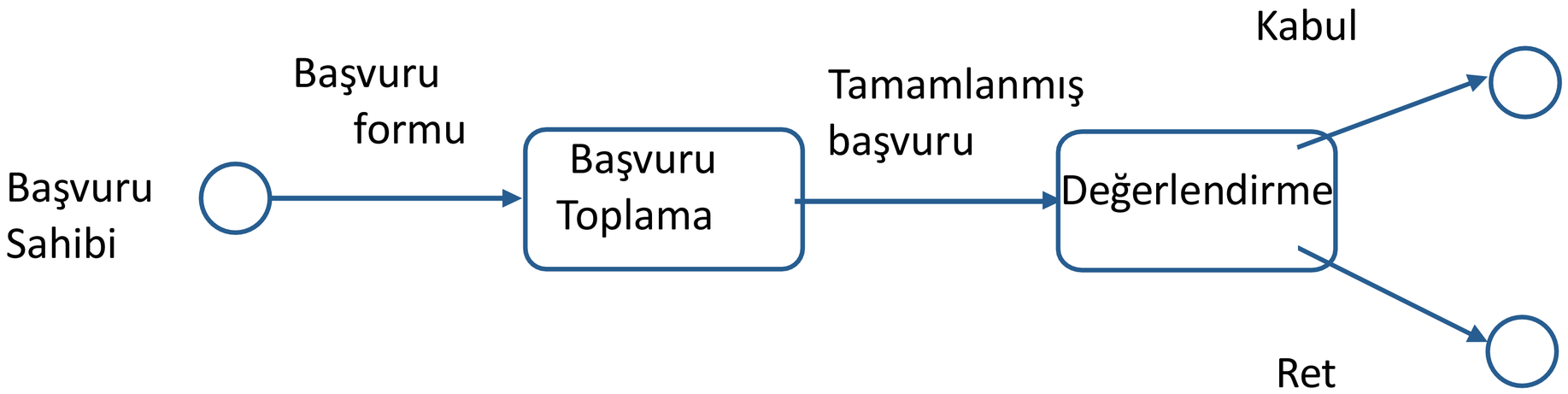
Harici varlıklar

İşleme adımları

Veri depoları veya kaynakları

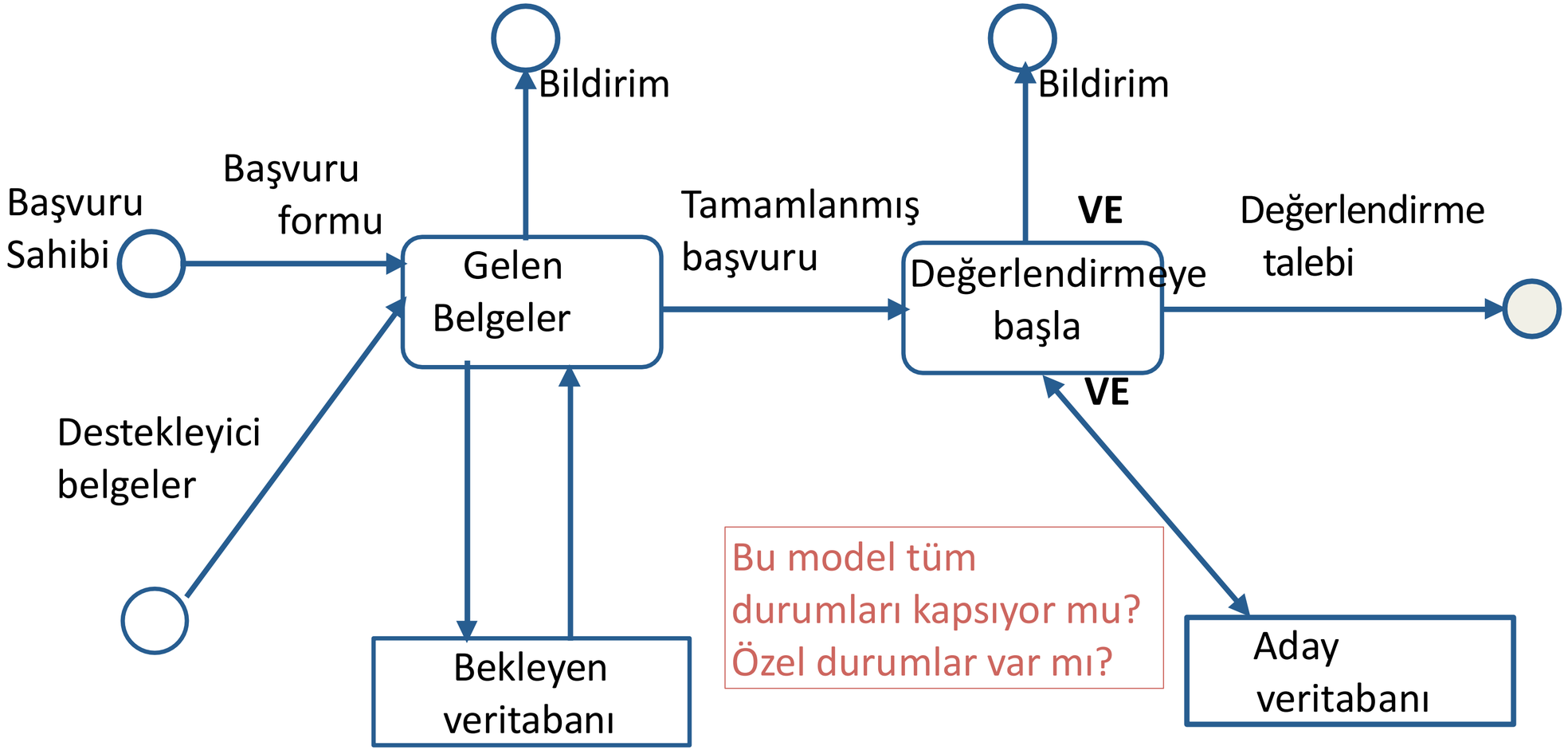
Veri akışları

Veri Akışı Modeli
Örnek: Özel Okul Kabulleri (İlk Deneme)



Akışı gösterir, ancak veriler nerede depolanır? Destekleyici bilgi var mı?

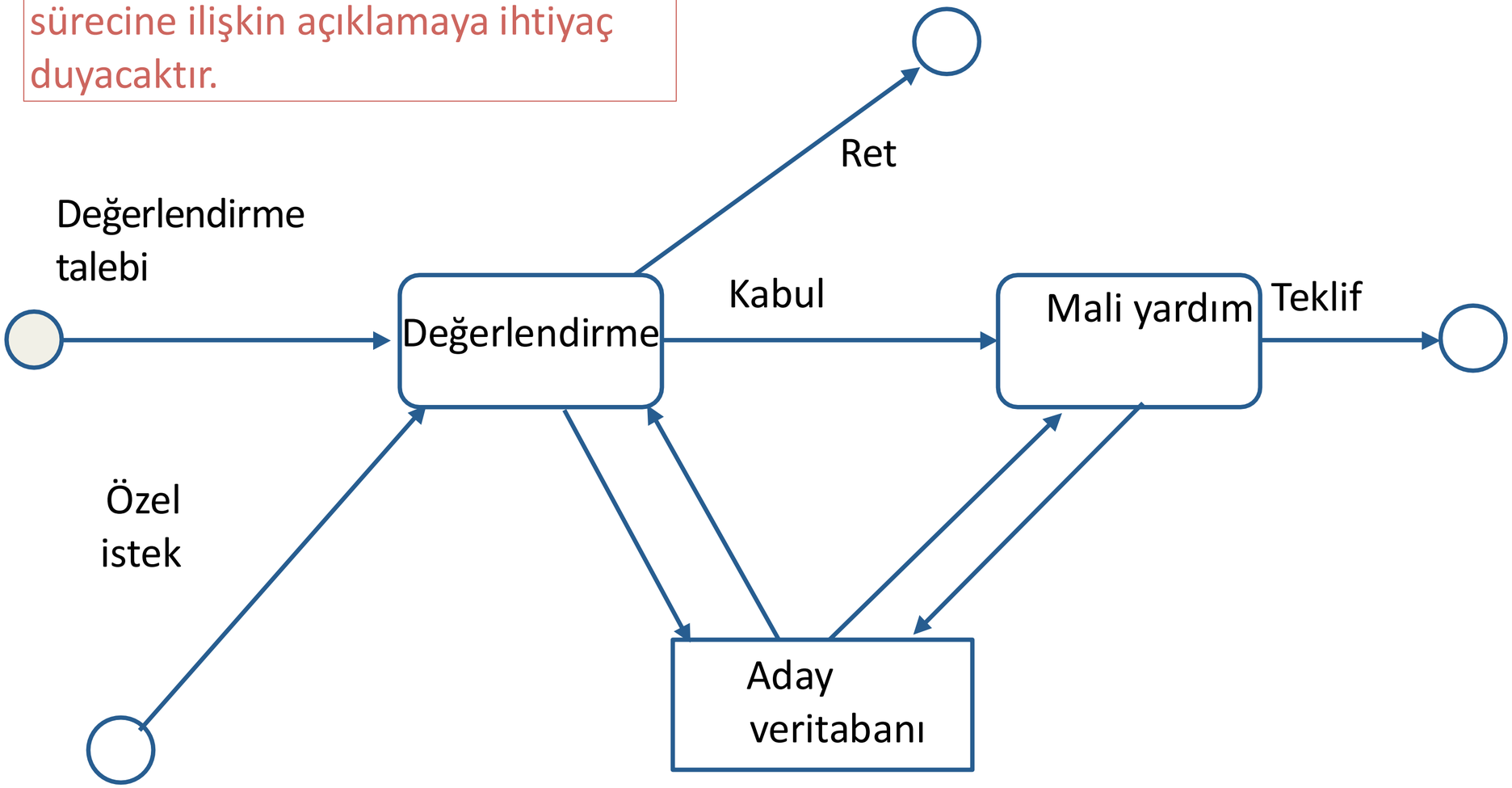
Veri Akışı Modeli
Örnek: Başvuru Toplama



Bu model tüm durumları kapsıyor mu? Özel durumlar var mı?

Veri Akışı Modeli
Örnek: Tamamlanan Başvuru

Gereksinimler, karar verme sürecine ilişkin açıklamaya ihtiyaç duyacaktır.



Karar Tablosu (Decision Table) Modeli

Üniversiteye Kabul Kararı

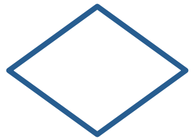
SAT > S1	T	F	F	F	F	F
GPA > G1	-	T	F	F	F	F
SAT between S1 and S2	-	-	T	T	F	F
GPA between G1 and G2	-	-	T	F	T	F
Accept	*X*	*X*	*X*			
Reject				*X*	*X*	*X*

Her sütun ayrı bir karar durumudur. Sütunlar soldan sağa doğru işlenir.
Kuralların spesifik ve test edilebilir olduğunu unutmayın.

Akış Şeması Modelleri

Bir sistemin bir parçasının mantığını ve verilerin bir sistem boyunca izlediği yolları göstermek için kullanılan resmi olmayan bir modelleme tekniği.

İşlem

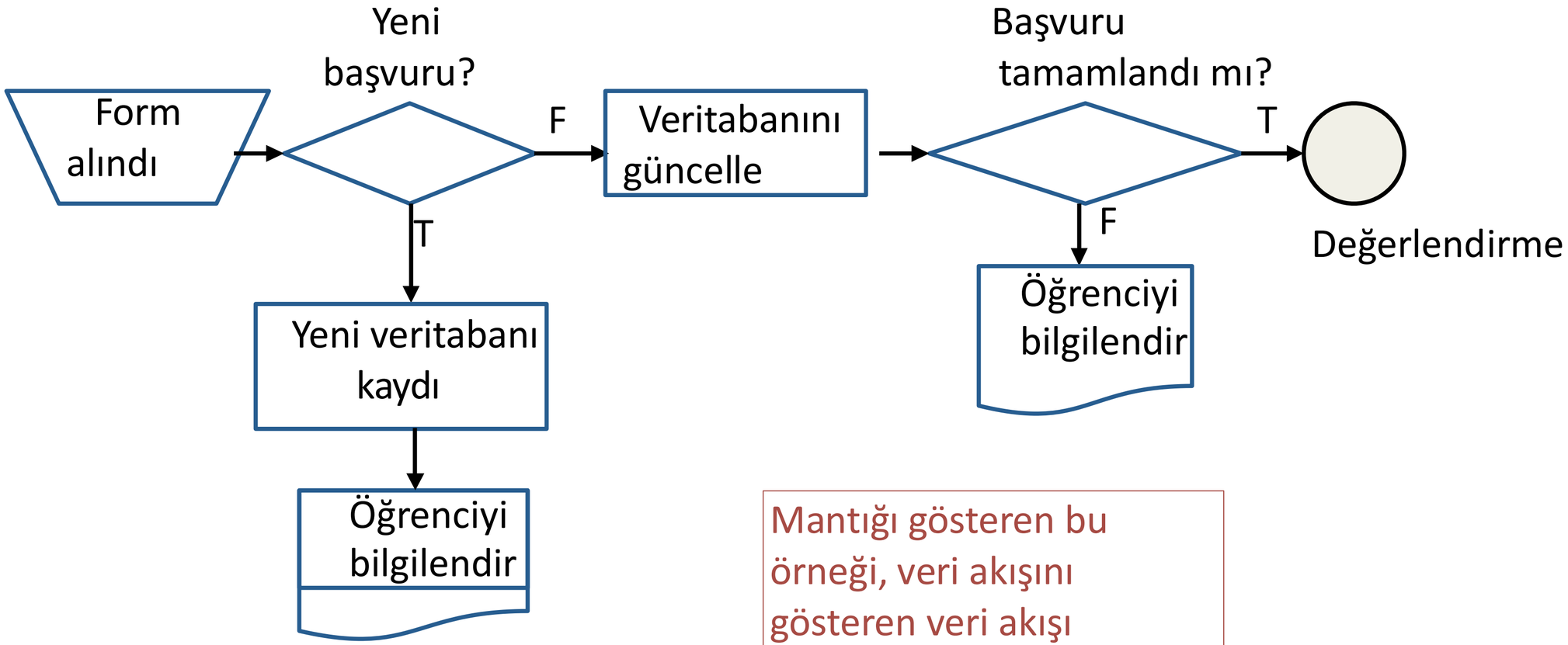


Karar

Manuel işlem

Rapor

Akış Şeması Modeli
Örnek: Üniversite Kabul Toplantısı



Mantığı gösteren bu örneği, veri akışını gösteren veri akışı modeliyle karşılaştırın.

Modelleme Araçları: Sözde kod (Pseudo-code)

Bir sistemin bir parçasının arkasındaki mantığı göstermek için gayri resmi bir modelleme tekniği.

Örnek: Üniversiteye Kabul Kararı

```
admin_decision (application)

  if application.SAT == null then error (incomplete)
  if application.SAT > S1 then accept(application)
  else if application.GPA > G1 then accept(application)
  else if application.SAT > S2 and application.GPA > G2
    then accept(application)
  else reject(application)
```

Sözde kod için kullanılan gösterim gayri resmi veya bir yazılım geliştirme kuruluşu tarafından kullanılan bir standart olabilir veya normal bir programlama diline dayalı olabilir. Önemli olan, yorumunun ilgili herkes tarafından anlaşılmasıdır.

Modelleme Araçları: Geçiş (Transition) Diyagramları

Sistem S_i **durum** (**states**) seti olarak tasarlanır.

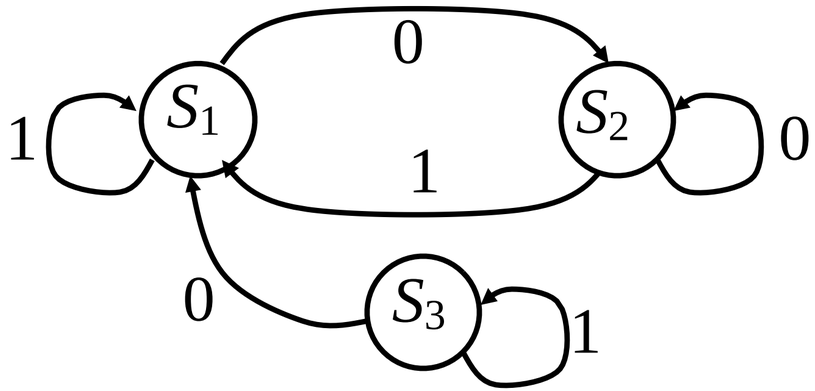
Geçiş, bir durumdan diğerine geçiştir.

Bir C_i **koşulunun** meydana gelmesi ile bir durumdan diğerine geçiş gerçekleşir.

Geçiş fonksiyonu:

$f(S_i, C_j) = S_k$

Örnek



Sonlu Durum Makine Modeli
Terapi Kontrol Konsolu

Örnek: Radyasyon Tedavisi Kontrol Konsolu

Operatörün kontrol konsolu için gereksinimler geliştirdiğinizi varsayalım. Bir görüşmede müşteri, operatörün makineyi çalıştırırken izlemesi gereken prosedürleri anlatmaktadır.

Siz de prosedürleri belirtmek için sonlu durum makine modeli kullanmaya karar verirsiniz.

Bu modelin amacı müşteriye gereksinimleri anladığınızı göstermektir ve geliştiriciler için prosedürleri belirtmektir.

Sonlu Durum Makine Modeli
Terapi Kontrol Konsolu: Senaryo

Senaryo

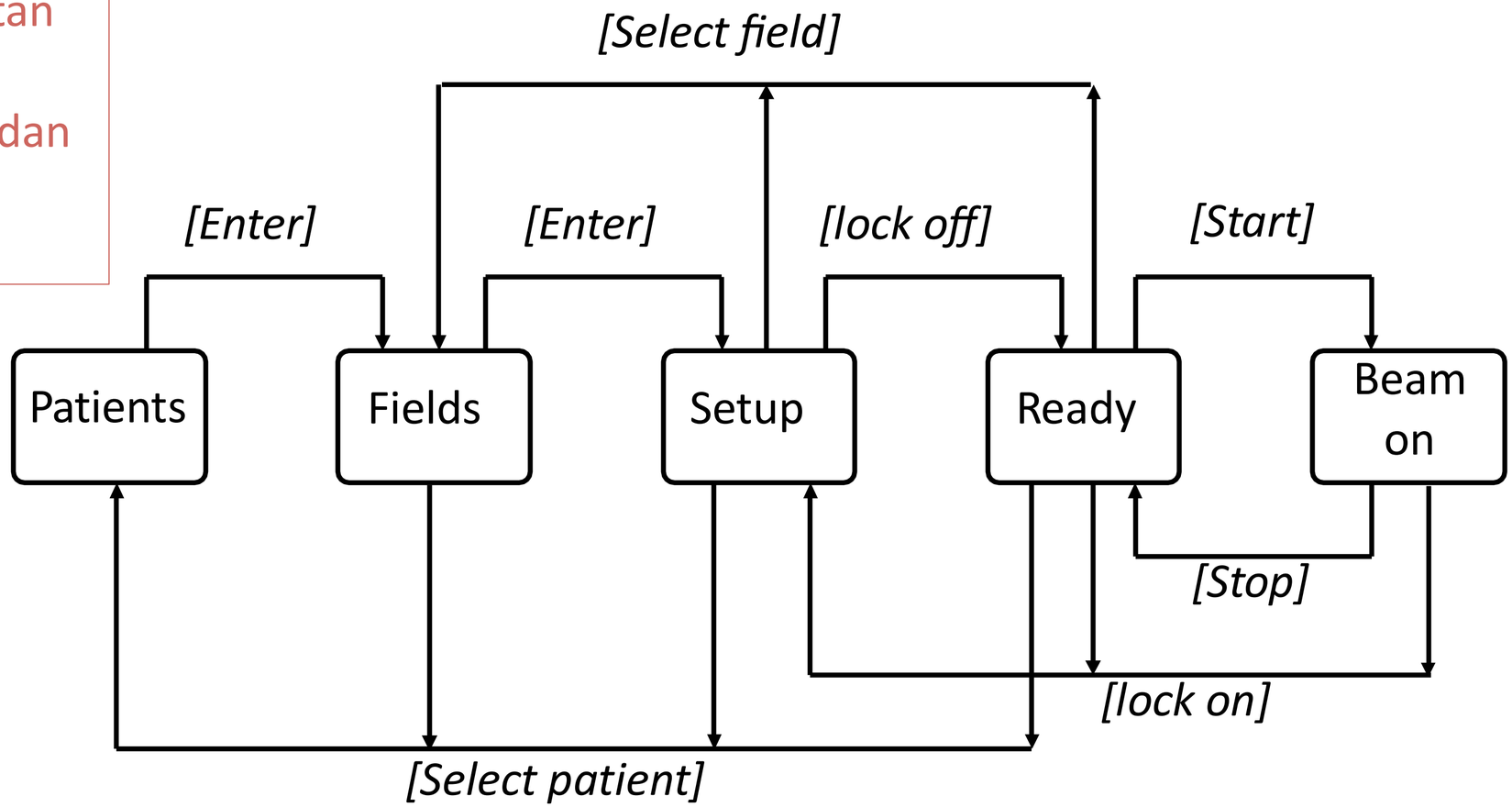
Müşteri aşağıdaki kaba senaryoyu sağlamaktadır.

"Hasta hazır hale getirilmeden önce kurulum yapılır. Operatör, hasta bilgilerini bir veri tabanından seçer. Seçim yapılınca ilgili hasta için onaylanan radyasyon alanlarının bir listesi görüntülenir. Operatör ilk alanı seçer böylece kurulum tamamlanır.

"Hasta artık hazırdır. Kilit makineden çıkarılır ve bu alana uygun dozlar uygulanır. Operatör daha sonra alan seçimine geri döner ve başka bir alan seçer."

Sonlu Durum Makine Modeli
Durum Geçiş Diyagramı

Her durumu ve geçiş müşteriyle tartışıldıktan sonra operatör konsolunun 5 durumdan birinde olduğu tespit edilir.

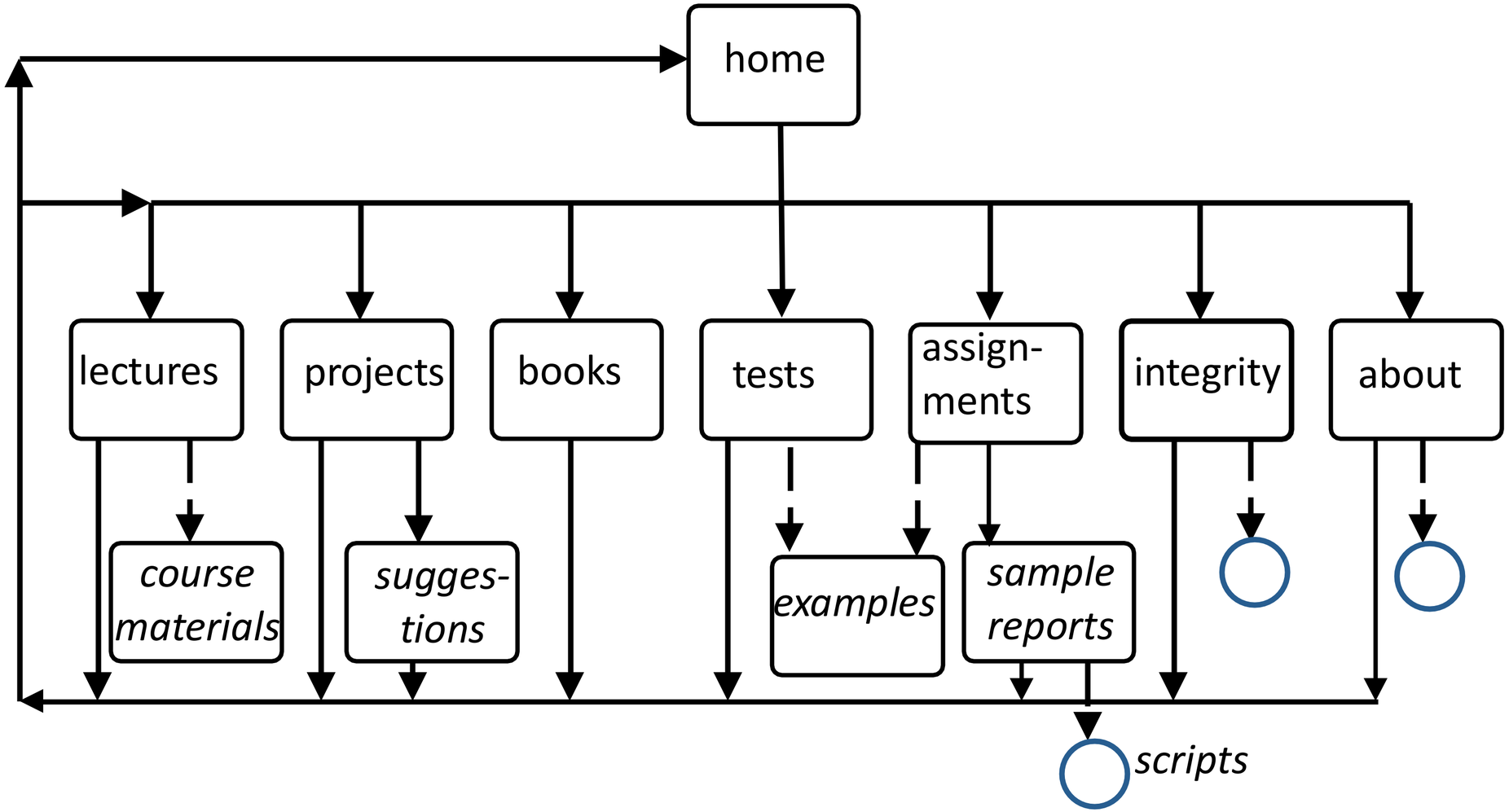


Sonlu Durum Makine Modeli
Durum Geçiş Tablosu

	Select Patient	*Select Field*	*Enter*	*lock off*	*Start*	*Stop*	*lock on*
Patients	___	___	Fields	___	___	___	___
Fields	Patients	___	Setup	___	___	___	___
Setup	Patients	Fields	___	Ready	___	___	___
Ready	Patients	Fields	___	___	Beam on	___	Setup
Beam on	___	___	___	___	___	Ready	Setup

Bu tablo, gereksinimlerin tanımlanması, program tasarımı ve kabul testi için kullanılabilir.

Kullanıcı Arayüzleri için Geçiş Diyagramı
Örnek: Bir Web Sayfası



Varlık-İlişki Modeli

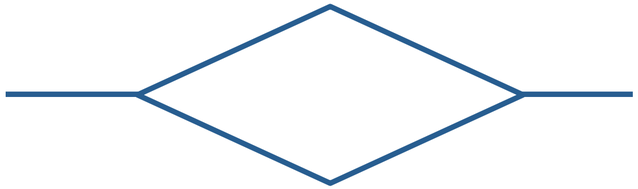
İlişkisel veri tabanlarında gereksinimler ve tasarımlar için standart metodolojidir.

- Varlıklar ve ilişkilerden oluşan bir veri tabanı
- Varlık-ilişki diyagramlarını görüntülemek ve değiştirmek için araçlar
- Veritabanını işlemek için araçlar (örneğin, veri tabanı tasarımına girdi olarak)

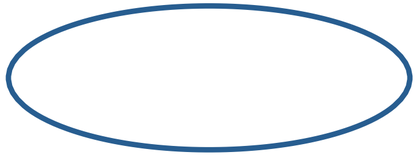
Varlık-ilişki modelleri hem gereksinim belirtimi hem de tasarım belirtimi için kullanılabilir.

Modelleme Araçları: Varlık-İlişki Diyagramı

Bir varlık (isim)



Varlıklar arasındaki ilişki (fiil)

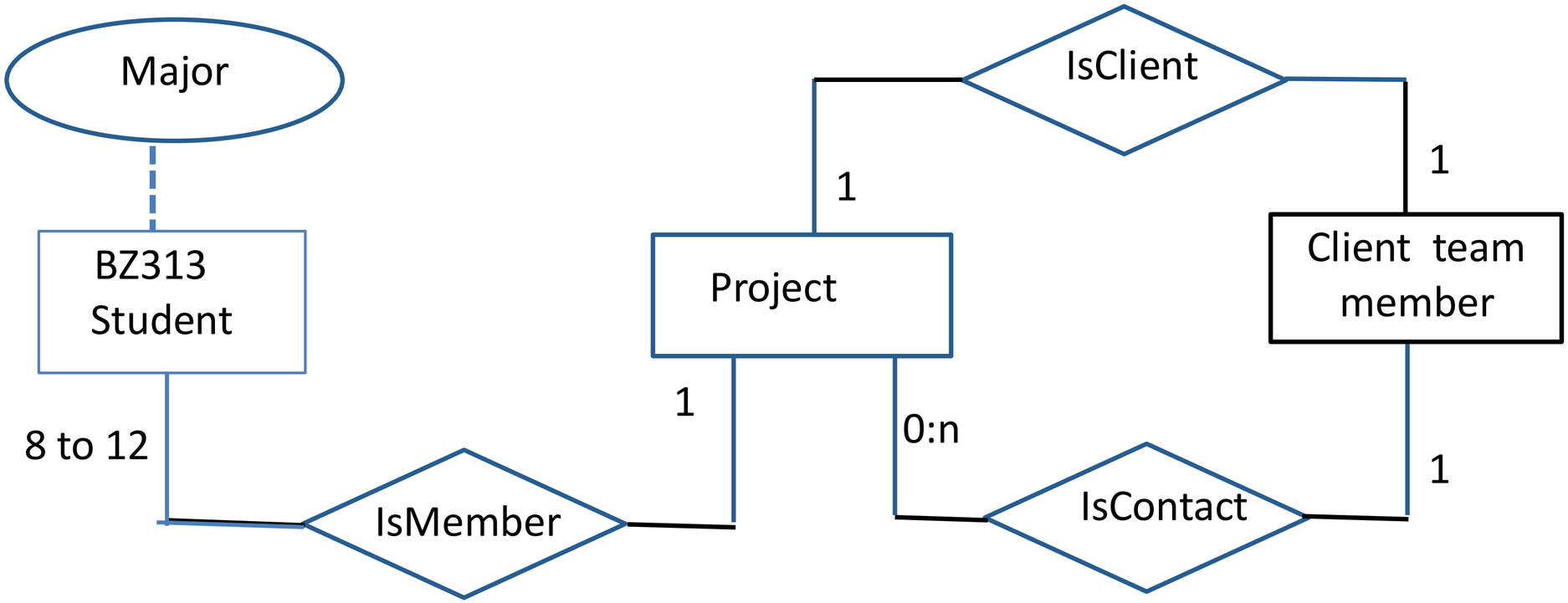


Bir varlık veya ilişki özniteliği

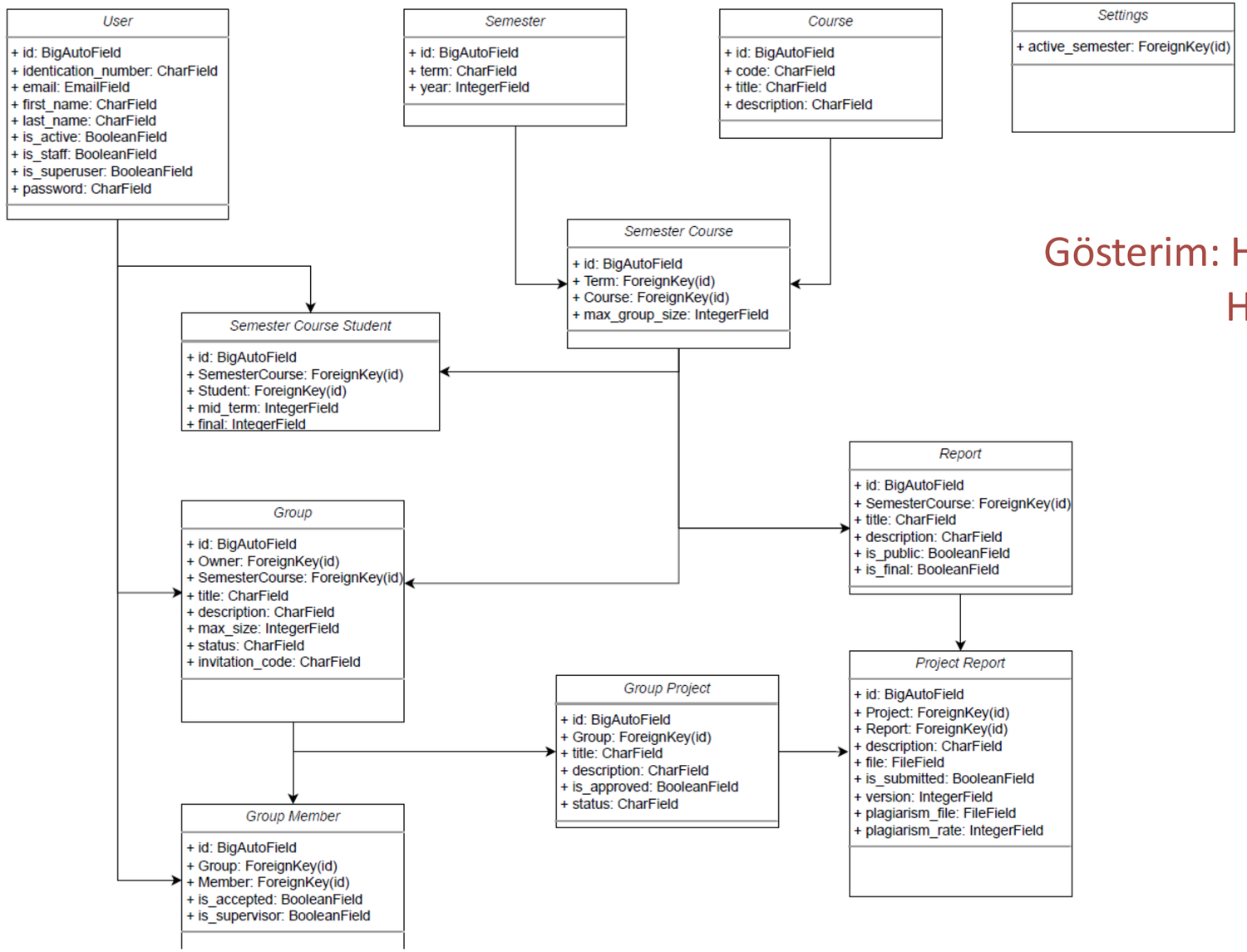
Not: Varlık-ilişki diyagramları için kullanılan çeşitli gösterimler vardır. Bu, Chen (1976) tarafından kullanılan gösterimdir.

Modelleme Araçları: Varlık İlişkisi Diyagramı

Örnek:



Bir Tasarım Aracı Olarak Varlık İlişkisi Diyagramı
Örnek: Web Verileri için Veritabanı Şeması



Gösterim: Her tablo bir varlığı temsil eder
Her ok bir ilişkiyi temsil eder

Gereksinimleri Prototipleme

Hızlı prototipleme, tüm modelleme yöntemlerinin en kapsamlısıdır

Gerekli sistemin temel parçalarının işlevselliğini gösteren bir sistem oluşturarak gereksinimleri belirleme yöntemidir.

Kullanıcı arayüzleri için özellikle değerlidir

Gereksinim Tanımı: Veri Sözlükleri

Veri sözlüğü, sistem tarafından kullanılan adların bir listesidir

- Ad (ör., "start_date")
- Kısa tanım (ör., "date" nedir)
- Nedir? (ör., integer, relation)
- Nerelerde kullanılır (örneğin, kaynak, kullanan, vb.)
- Bir sözlükle birleştirilebilir

Sistem geliştirildikçe gereksinimlerdeki veri sözlüğü, nihai dokümantasyonun bir parçası olabilecek sistem veri sözlüğünün temelini oluşturur.

Sınıf ve Nesne Modelleri Üzerine Bir Not

- Program tasarımında oldukça önemli olan sınıf ve nesne modelleri bazen gereksinim modellemede de önerilmektedir.
- Ancak sistem tasarımını çok fazla kısıtladığı için sınıf ve nesne modellemeyi gereksinim düzeyinde kullanmak zor olabilmektedir.
- Akış modelleri ve sonlu durum makinaları UML'ler tarafından desteklenmektedir ve gereksinimleri modellemek için daha faydalı olabilmektedir.

Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

BZ 313 Yazılım Mühendisliği

8. Gereksinimler için Modeller

Ders Sonu