



Hafta 3: Gereksinim Mühendisliği

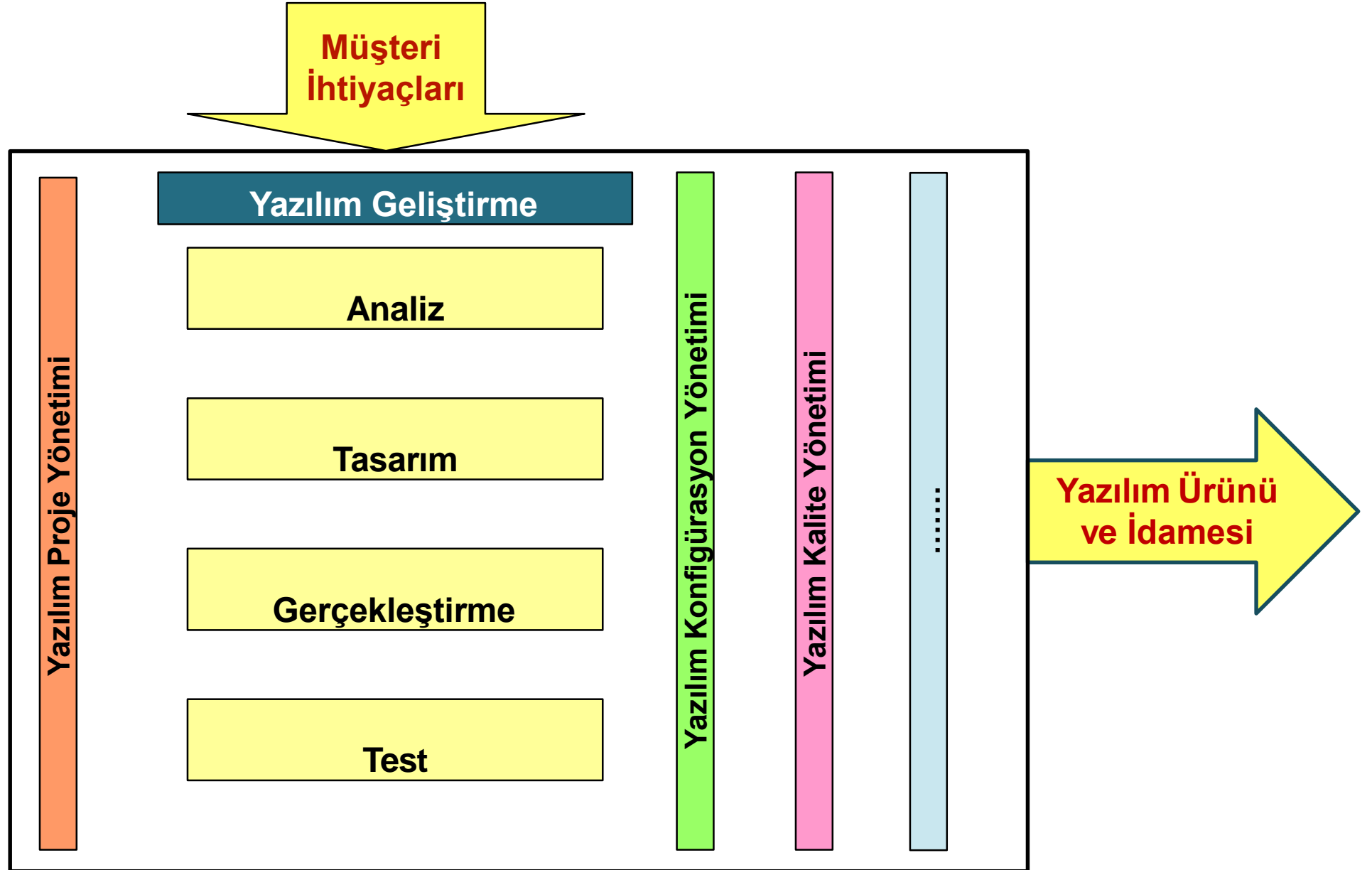
İÇERİK

- Gereksinim Mühendisliği Süreci
 - ▶ Gereksinim çıkarma, analiz, geçerleme, yönetim
- Gereksinimler
 - ▶ Gereksinim türleri nelerdir?
 - ▶ Gereksinim tanımlama
- İş Gereksinimleri Analizi ve Gereksinim Analizi İlişkisi
 - ▶ İş süreçlerini modelleme, gereksinim analizi

YAZILIM YAŞAM DÖNGÜSÜ

- Girdi: İş gereksinimleri (iş alanı bilgisini gerektirir)
- Yazılım ürününün geliştirilmesi ve yönetilmesi için izlenmesi gereken adımların bütünüdür
 - ▶ Analiz, tasarım, kodlama, test, ...
 - ▶ Proje yönetimi, kalite yönetimi, yapılandırma yönetimi, ...
- Yazılım ürününe mühendislik etkinliklerinin uygulanmasına olanak sağlar
 - ▶ Örnek: ISO/IEC 12207 Yazılım Yaşam Döngüsü Süreçleri
- Çıktı: Yazılım sistemi (çözüm alanı bilgisini gerektirir)

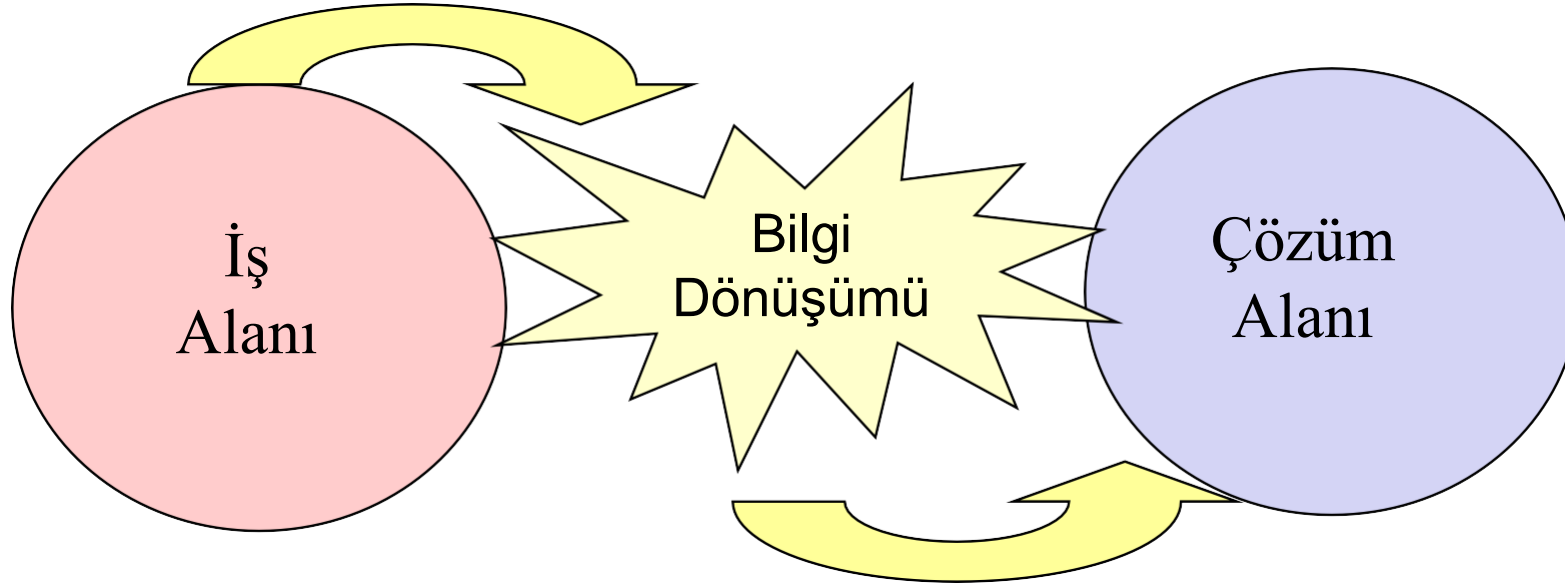
Yazılım Yaşam Döngüsü





GEREKSİNİM MÜHENDİSLİĞİ SÜRECİ

İŞ VE ÇÖZÜM ALANI BİLGİLERİ



*Yazılım sisteminin başarısında;
iş alanına ilişkin bilgi, en az çözüm alanına
ilişkin bilgi kadar önemlidir*



How the customer explained it



How the Project Leader understood it



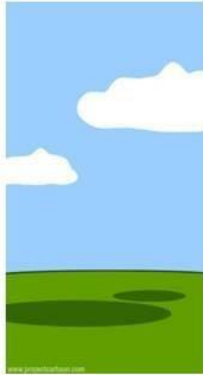
How the Business Consultant described it



How the Analyst designed it



How the Programmer wrote it



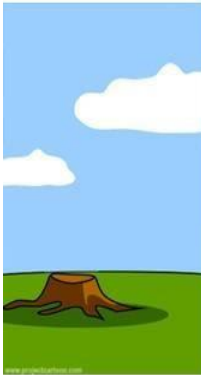
How the project was documented



What Operations installed



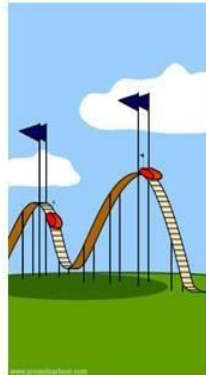
How it performed under load



How it was supported



What marketing advertised



How the customer was billed



What the customer really needed

*Farklı bilgi alanları,
farklı diller!!*

GEREK SINİM MÜHENDİSLİĞİ

- **Gereksinim mühendisliği**; müşterinin sistemden istediği servisleri ve sistemin altında çalışacağı kısıtları ortaya çıkarma ve tanımlama sürecidir.
- **Gereksinimler**; gereksinim mühendisliği süreci boyunca ortaya çıkan sistem servislerinin ve kısıtlarının bir tanımıdır.

GEREKSİNİM NEDİR?

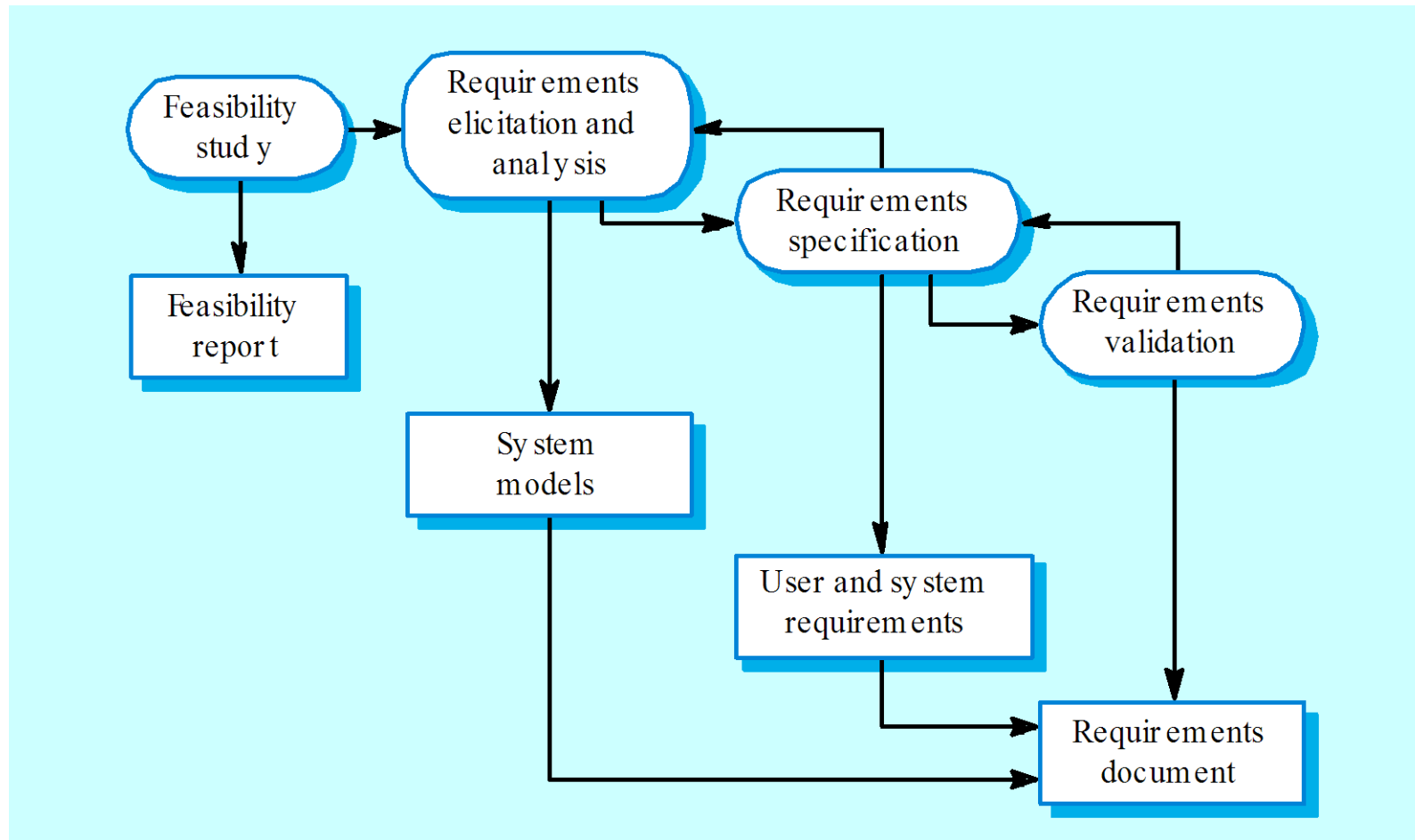
- Bir sistem servisinin / kısıtının üst seviyeli ifadesinden detaylı matematiksel tanımına kadar farklılık gösterebilir.
 - ▶ Sözleşme için teklife çağrı belgesine esas olabilir – farklı çözümlerin önerilmesine açık olmalıdır.
 - ▶ Sözleşmeye esas olabilir – detaylı olarak tanımlanmalıdır.
- Geliştirilecek sistemin sağlaması gereken bir durum veya yetenek
- Sistemin bir iş ihtiyacını karşılaması için göstermesi gereken özellik
- ...

GEREK SINİM MÜHENDİSLİĞİ SÜRECİ - I

■ Temel olarak aşağıdaki etkinlikleri içerir:

- ▶ Gereksinim çıkarma (“Requirements elicitation”)
- ▶ Gereksinim analizi (“Requirements analysis”)
- ▶ Gereksinim geçerleme (“Requirements validation”)
- ▶ Gereksinim yönetimi (“Requirements management”)

GEREKSİNİM MÜHENDİSLİĞİ SÜRECİ - 2



GEREKSİNİM MÜHENDİSLİĞİ SÜRECİ (3) - ETKİNLİKLERİN GELİŞİMİ

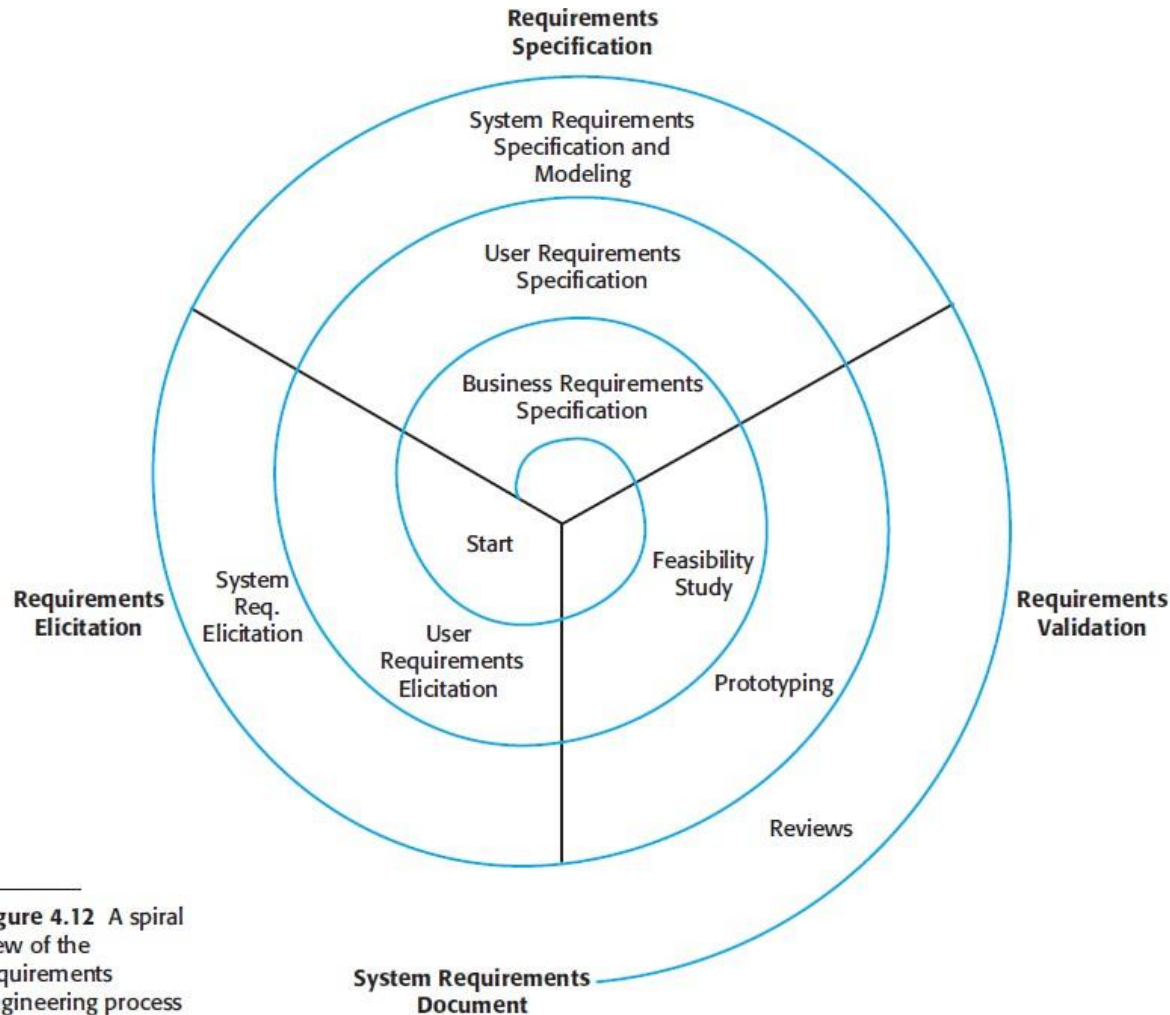


Figure 4.12 A spiral view of the requirements engineering process

Gereksinimler 3 seviyede çıkarılır, tanımlanır, geçerlenir:

- **İş (alan) gereksinimleri**
 - İş alanındaki yapı ve yürütülen işlere ilişkin tanım
 - *İş ortamı ve kısıtları ne, kurumsal birimler hangileri, bu birimlerin sorumlu olduğu işler nelerdir, halihazırda kullanılan sistemler var mı?*
- **Kullanıcı gereksinimleri**
 - İş alanında sistem desteği olacak kapsama ve bu kapsamda sistem ile kullanıcıları (insan ya da başka bir sistem) arasındaki etkileşime ilişkin genel tanım
 - *Sistem hangi iş gereksinimlerini destekleyecek, kullanıcılar sistem üzerinden hangi işlerini yürütecek, geliştirme kapsamı ne olacak?*
- **Sistem gereksinimleri**
 - Her bir kullanıcı gereksinimi için, kullanıcı ve sistem arasındaki etkileşimin detaylı tanımı (tipik olarak her kullanıcı gereksinimi, birden çok sistem gereksinimine detaylandırılır.)

KULLANICIVE SİSTEM GEREKSİNİMLERİ: ÖRNEK

User Requirement Definition

1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System Requirements Specification

- 1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.
- 1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
- 1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed, and the total cost of the prescribed drugs.
- 1.4 If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit.
- 1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

OLURLUK ÇALIŞMALARI (“FEASIBILITY STUDIES”)

- Olurluk çalışmasının amacı, sistemin gerçekleştirilip gerçekleştirilmeyeceğine karar vermektir.
- Bilgi toplamaya, değerlendirmeye ve raporlamaya dayanır.
 - ▶ Sistem kurumsal hedeflere hizmet edecek mi?
 - ▶ Sistem mevcut teknoloji ve bütçe sınırları içinde gerçekleştirilecek mi?
 - ▶ Sistem mevcut diğer sistemlere entegre edilebilecek mi?

GEREKSİNİM ÇIKARMA VE ANALİZİ (I)

- Uygulama (iş) alanını anlamayı ve sistemin bu alanı ne kapsamda destekleyeceğini bulmayı gerektirir.
 - ▶ İş alanında yürütülen işlevler nelerdir ve nasıl ilişkilidir?
 - ▶ Sistem bu işlevleri ne kapsamda ve hangi noktalarda destekleyebilir?
 - ▶ Sistemin destekleyeceği iş kapsamı için gereksinimler ve kısıtlar nelerdir?
- Son kullanıcılar, yöneticiler, alan uzmanları ve geliştiriciler; gereksinim çıkarma ve analiz etkinliklerine dahil olur. Tüm bu kişiler *sistemin paydaşları* (“*stakeholders*”) olarak isimlendirilir.

GEREKSİNİM ÇIKARMA VE ANALİZİ (2)

■ Yapılan hatanın düzeltilme maliyeti çok yüksek

- ▶ Bakım aşamasında 20 kat fazla
- ▶ Bulunan hataların %40'ı gereksinimlerden kaynaklanıyor

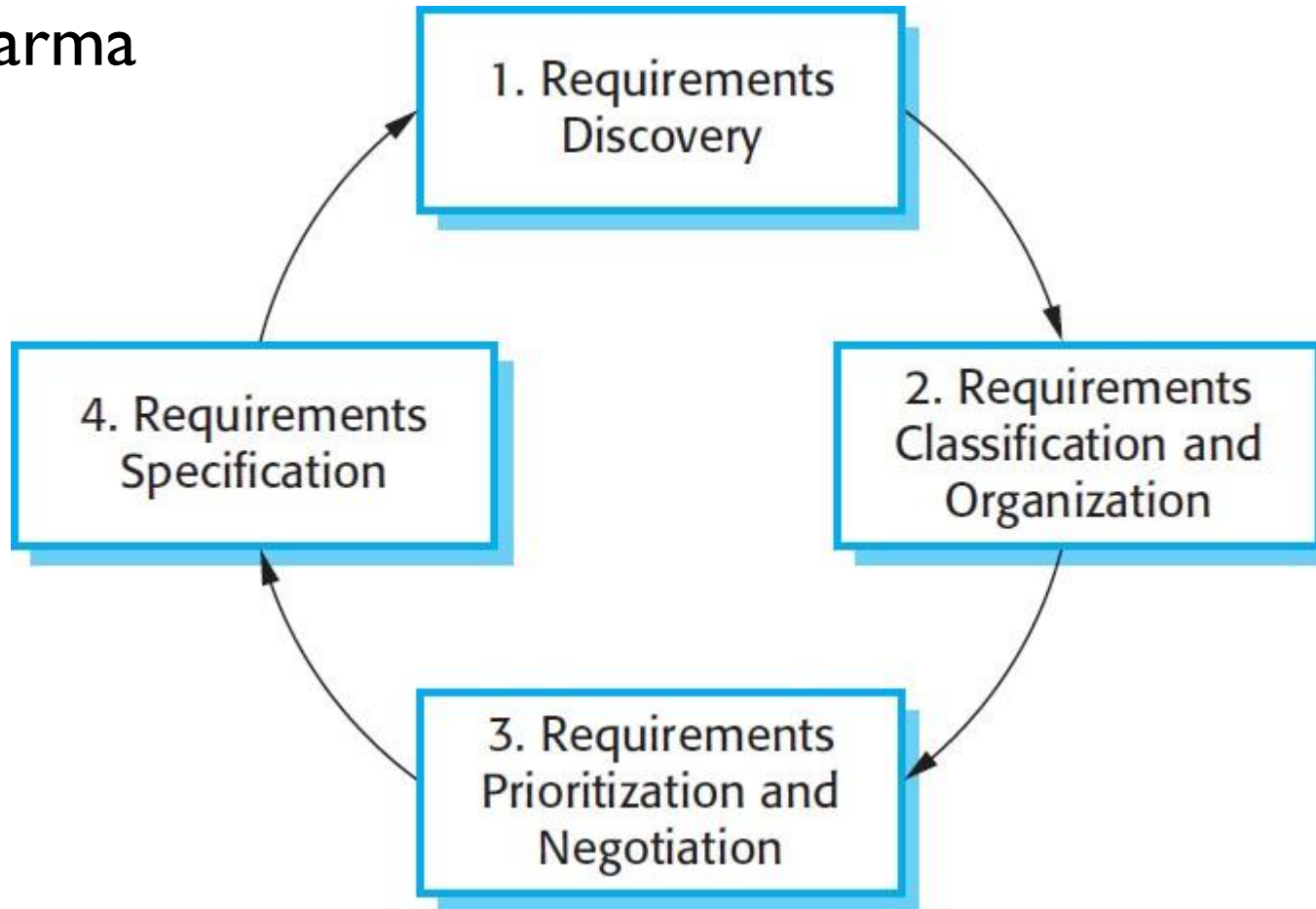
■ “Doğru” sistemi oluşturmak için

- ▶ Kurumsal ortam nedir ?
- ▶ Müşteri ne istiyor ?
- ▶ Bağlı bulunduğu sistemler var mı ?

■ Gereksinim tanımı iletişime esas

- ▶ Müşteri – teknik ekip
- ▶ Analiz uzmanları – tasarım uzmanları

Gereksinim Çıkarma Ve Analizi (3)



GEREKSİNİM ÇIKARMA VE ANALİZİ – ZORLUKLAR

■ Yazılım geliştirme çalışmalarının ön aşamaları

- ▶ Kimse birbirini tanımıyor, sistemi bilmiyor
- ▶ Yöntem, teknoloji, vb. yeni olabilir

■ Kullanıcı odaklı problemler yaşanabilir.

- ▶ Kullanıcılar ne istediklerini bilmeyebilir.
- ▶ Kullanıcılar isteklerini kendi terimleriyle ifade edebilir.
- ▶ Farklı kullanıcıların çelişen istekleri olabilir.

■ Farklı grupların beraber çalışmasını gerektiriyor.

- ▶ Kullanıcı grupları, analiz uzmanları, mimari/tasarım uzmanları

■ Kurumsal ve politik etkenler sistem gereksinimlerini etkileyebilir.

■ Analiz boyunca gereksinimler değişebilir; yeni kullanıcılar çıkabilir ve iş ortamı değişebilir.

GEREKSİNİM GEÇERLEME

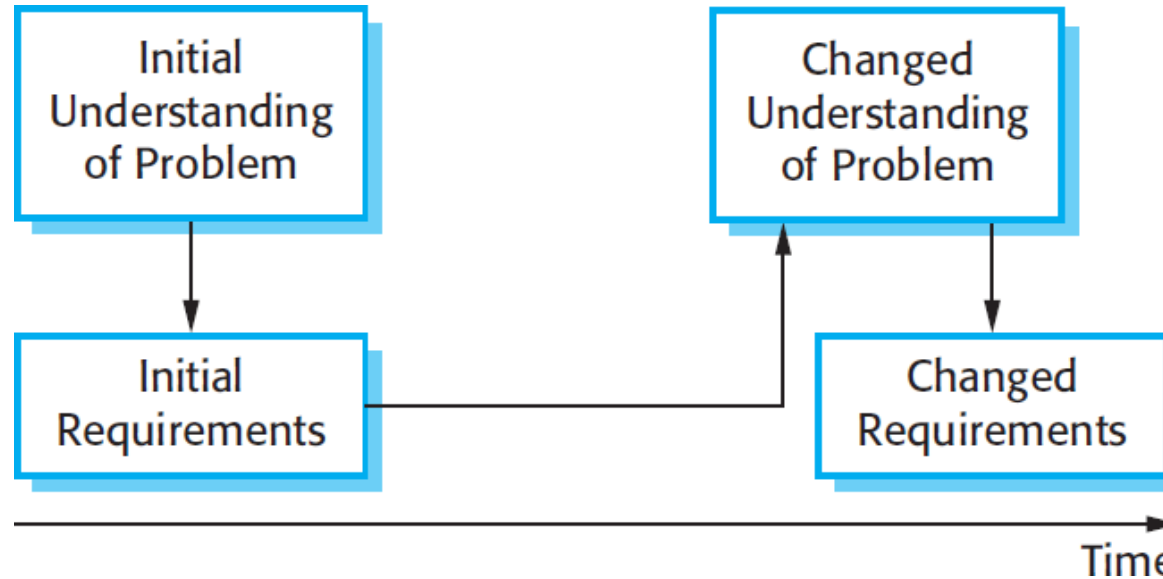
- Gereksinimlerin müşterinin istediği sistemi tanımladığını güvence altına almayı hedefler.
- Gereksinimlerden kaynaklanan bir hatayı sonradan düzeltmenin maliyeti yüksek olduğundan, geçerleme büyük önem taşır.
- Gereksinim geçerleme teknikleri:
 - ▶ **Gözden geçirme** – gereksinimlerin sistematik ve paydaşlara dayalı analizi
 - ▶ **Prototip oluşturma** – sistemin çalışan bir taslağı üzerinden kontrol
 - ▶ **Test durumu geliştirme** – sistemin test edilebilirliğini kontrol amacıyla gereksinimler için test durumu geliştirme

GÖZDEN GEÇİRME

- Gereksinimlerinin gözden geçirilmesi, hataların en aza indirilmesi, kalitenin güvencesi ve projenin başarısı açısından son derece önemlidir.
 - ▶ Gereksinim aşaması yazılım geliştirme sürecindeki en zayıf noktadır.
 - ▶ Yapılan hatalar ancak sürecin çok ilerleyen aşamalarında (büyük olasılıkla kabul testlerinde) ortaya çıkacaktır.
 - ▶ Analiz ekibi gerek kendi içinde gerekse sistemin kullanıcıları ile gereksinimleri gözden geçirmelidir.
- Gözden geçirme çalışmalarında hedefimiz, yazılım gereksinimlerinin aşağıdaki özellikleri taşıdığından emin olmaktır:
 - ▶ Tam ve doğru
 - ▶ Anlaşılabilir
 - ▶ Tutarlı
 - ▶ Test edilebilir
 - ▶ Diğer belgelerde tanımlanan iş/kullanıcı/sistem gereksinimlerine izlenebilir

GEREKSİNİMYÖNETİMİ

- Geliştirme boyunca değişen gereksinimlerin yönetilmesi etkinliğidir.
 - ▶ Geliştirme ilerledikçe (ve sisteme ilişkin anlayış iyileştikçe) yeni gereksinimler ortaya çıkabilir.
 - ▶ Farklı bakış açılarının gereksinimleri zaman içinde çelişebilir.



İZLENEBİLİRLİK




- Gereksinim Belgesinde yer alan her gereksinim, analizde kaynak olarak kullanılan belgelere ve sistem tasarımına izlenebilir olmalıdır.
- İzlenebilirlik aşağıdaki gibi sınıflandırılabilir:
 - ▶ Kaynağa olan izlenebilirlik
 - ◆ Gereksinimlerden, onu oluşturan paydaşlara ve kaynak belgelere
 - ▶ Gereksinimlere olan izlenebilirlik
 - ◆ Gereksinimlerden, bağımlı diğer gereksinimlere
 - ▶ Tasarıma olan izlenebilirlik
 - ◆ Gereksinimlerden tasarıma
- Gereksinimler ve diğer belgeler arasındaki ilişki bir matrisle tanımlanır.
 - ▶ Gözden geçirme sırasında bu matristeki bilginin doğruluğu, tamlığı ve tutarlılığı kontrol edilir.
 - ▶ İzlenebilirlik çift yönlü olarak sağlanmalıdır (geliştirme boyunca ileri ve geri).

İZLENEBİLİRLİK MATRİSİ – ÖRNEK: GEREKSİNİMLERDEN GEREKSİNİMLERE İZLENEBİLİRLİK

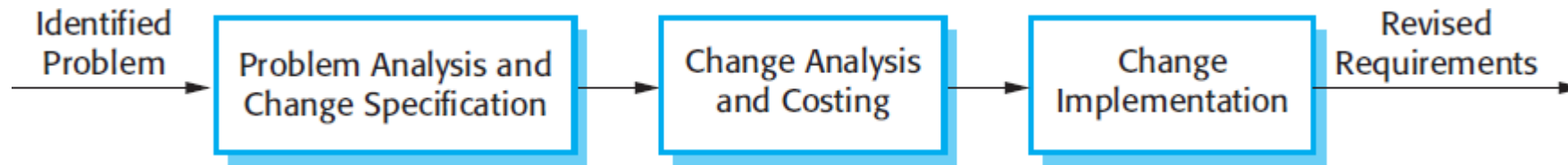
Req. id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			D		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

D: bağımlı ("dependent")
R: ilişkili ("relational")

GEREKSİNİMLERİN TEST DURUMLARINA İZLENEBİLİRLİĞİ

Requirement Traceability Matrix												
	Test Case ID 	TC_1	TC_2	TC_3	TC_4	TC_5	TC_6	TC_7	TC_8	TC_9	TC_10	# Test Cases for respective Requirement 
Req. ID 												
Req_1		✖		✖			✖					3
Req_2			✖			✖						2
Req_3				✖								1
Req_4					✖		✖					2
Req_5						✖		✖				2
Req_6							✖					1
Req_7						✖		✖				2
Req_8									✖			1
Req_9										✖		1
Req_10											✖	1

GEREKSİNİM DEĞİŞİKLİK YÖNETİMİ ADIMLARI



Genellikle bir CASE (Computer Aided Software Engineering) aracı ile desteklenir.



GEREKSİNİM TÜRLERİ VE GEREKSİNİM TANIMLAMA

KULLANICI VE SİSTEM GEREKSİNİMLERİ

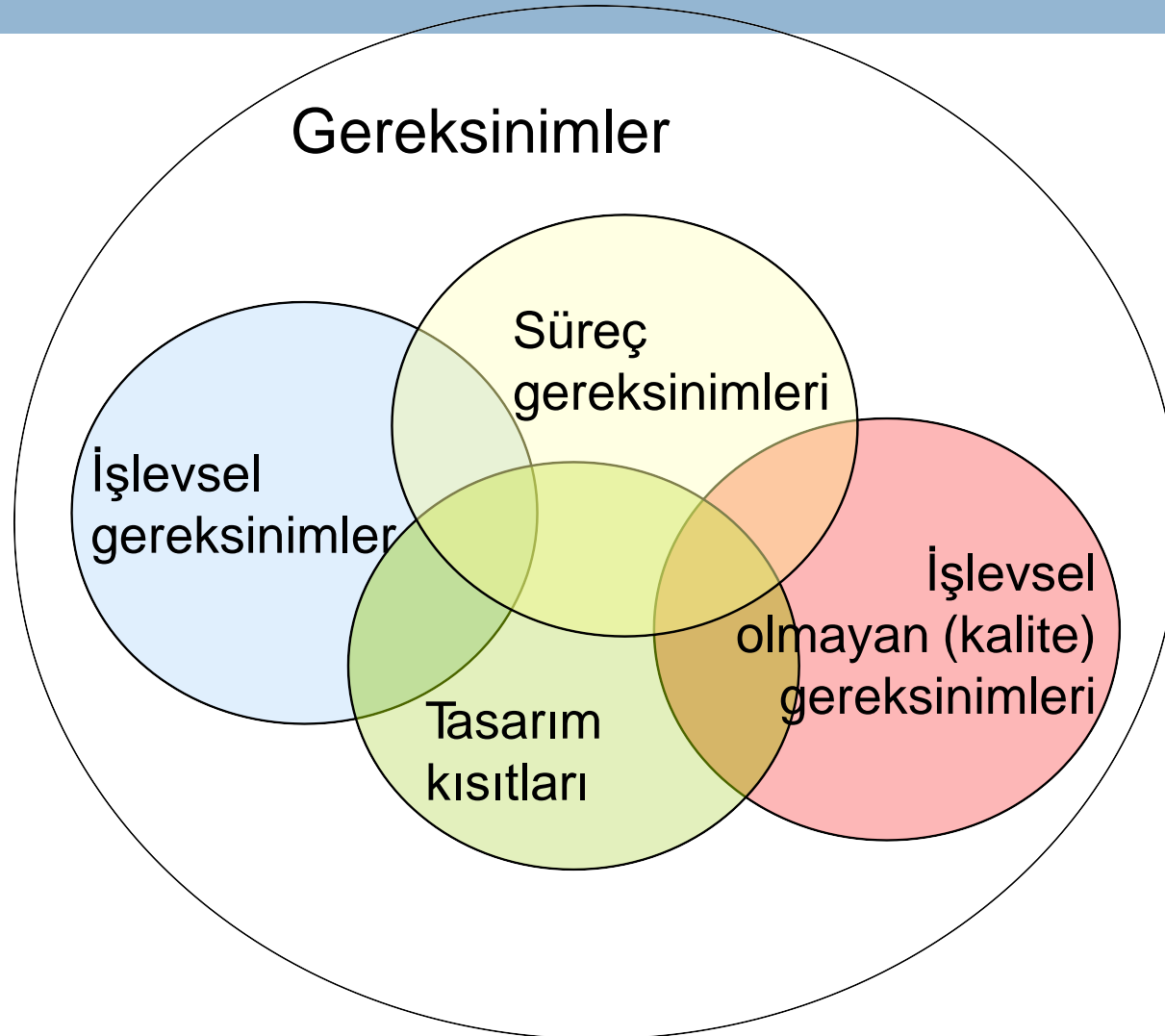
■ Kullanıcı gereksinimleri

- ▶ Doğal dilde ifade edilir.
- ▶ Sistemin sağlayacağı servislerin ve uyacağı kısıtların ifadesidir.
- ▶ Müşteriler için yazılır.

■ Sistem gereksinimleri

- ▶ Sistem işlevlerinin, servislerinin ve kısıtlarının detaylı tanımlarını içeren yapısal bir belge ile ifade edilir.
- ▶ Müşteri ve sağlayıcı arasındaki sözleşmenin bir bölümü olarak, sistemin neleri gerçekleştireceğini tanımlar.

GEREKSİNİM UZAYI



GEREKSİNİM TÜRLERİ (I)

■ İşlevsel gereksinimler

- ▶ Kullanıcıların sistem ile gerçekleştirmeyi istediği işlemler ve bu işlemlerin özellikleri
- ▶ Sistem “ne” yapacak ?
 - ◆ Girdilerin ve çıktıların tanımı
 - ◆ Girdileri çıktılara dönüştüren işlemlerin tanımı

GEREKSİNİM TÜRLERİ (2)

■ İşlevsel olmayan gereksinimler

► Ürün kalite kriterleri

- ◆ Performans, güvenilirlik, kullanılabilirlik vb.

- ◆ Örnek: Hatalar arası ortalama zaman

- “Bir sistem ya da sistem ögesi için hata oluşumları arasındaki ortalama zamandır.”

- “Tanımlı bir işletim süresince oluşan hatalar sayılarak ve işletim süresi bu hata sayısına bölünerek hesaplanır.”

- » $H.A.O.Z. = \text{işletim süresi} / \text{işletim süresince oluşan hata sayısı}$

- “Sistemin kesin kabul testi süresince ölçülen tüm sistem unsurları için hatalararası ortalama zamanı en az 60 (altmış) saat olacaktır.”

GEREKSİNİM TÜRLERİ (3)

■ Süreç gereksinimleri

- ▶ Geliştirme adımları ile ilgili istekler
- ▶ Tanımlı yaşam döngüsü, kalite güvence etkinlikleri, vb.

■ Tasarım kısıtları

- ▶ Tasarımı etkileyecek istekler
- ▶ Geliştirme ortamı (örnek: J2EE), ilişkisel veri tabanı, vb.

GEREKSİNİMLERİN BULANIKLIĞI

- Gereksinimler net olarak ifade edilmediği zaman problemler yaşanır.
- Bulanık gereksinimler geliştiriciler ve kullanıcılar tarafından farklı algılanabilir.
 - ▶ Örnek: “Sistem, kullanıcının doküman ambarındaki dokümanları okuması için, uygun görüntüleyiciler sağlamalıdır.”
 - ▶ “uygun görüntüleyiciler”:
 - ◆ Kullanıcı yorumu : her farklı doküman tipi için farklı kullanıcı
 - ◆ Geliştirici yorumu : her dokümanın içeriğini gösteren bir metin görüntüleyici

GEREKSİNİMLERİN TAMLIĞI VE TUTARLILIĞI

- Gereksinimler tam ve birbiriyle tutarlı olarak ifade edilmelidir.
 - ▶ **Tamlık** : Sistemin beklenen tüm özellikleri tanımlanmalıdır.
 - ▶ **Tutarlılık**: Sistemin tanımlanan özellikleri arasında çelişkiler bulunmamalıdır.
- Pratikte, doğal dilden kaynaklanan zorluklar sebebiyle, gereksinimleri tam ve tutarlı olarak ifade etmek çok kolay değildir.
- Tanımlanan gereksinimlerin ilgili tüm kişilerce [gözden geçirilmesi](#), tamlığı ve tutarlılığı büyük ölçüde sağlamanın en basit yoludur.

GEREKSİNİMLERİ YAZMAK İÇİN ÖNERİLER

- Standart bir biçim belirleyerek gereksinimleri tanımlarken kullanın.
- Her gereksinime bir numara verin.
- Doğal dili tutarlı olarak kullanın. Zorunlu ve seçimli gereksinimleri farklı kalıplarla ifade edin.
- Gereksinimlerin önemli kısımlarını ayırt etmek için farklı yazı tipi (büyük harf, alt çizme, farklı renk, vb.) kullanın.
- Bilgisayar terimlerini kullanmaktan kaçının.

GEREK SINİMLER VE TASARIM

- Gereksinimler, sistemin “ne” yapacağını tanımlar.
- Tasarım, sistemin tanımlanan gereksinimlerinin “nasıl” gerçekleştirileceğini belirtir.
- Pratikte, gereksinimler ve tasarım her zaman net olarak ayrılamayabilir.
 - ▶ Sistem mimarisi, gereksinimleri yapısalılaştırmak için tasarlanır.
 - ▶ Sistem işlevleri, tasarımı kısıtlayan diğer sistemlerle ilişki içinde gerçekleştiriliyor olabilir.
 - ▶ Müşteri tarafından, sistemin özel bir tasarıma uyması isteniyor olabilir.

Gereksinimlerin türlerine göre ayrı başlıklar altında tanımlanması, bu karışıklığı azaltacaktır.

DOĞAL DİLE İLE İLGİLİ PROBLEMLER

■ Muğlaklık (“Ambiguity”)

- ▶ Gereksinimler, okuyan herkes tarafından aynı yorumlanacak şekilde yazılmalıdır. Doğal dil muğlak ifadelerle açıktır.

■ Aşırı esneklik (“Over-flexibility”)

- ▶ Bir gereksinim, doğal dil ile çok farklı şekillerde ifade edilebilir.

■ Modülerliğin olmayışı (“Lack of modularisation”)

- ▶ Doğal dilin öğeleri, sistem gereksinimlerini yapısallaştırmak için yetersiz kalmaktadır.

Bu problemlere rağmen doğal dilin kullanılması, müşteri ve geliştirici arasındaki iletişim açısından önem taşımaktadır.

DOĞAL DİLE ALTERNATİFLER

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT. Now, use-case descriptions and sequence diagrams are commonly used.
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

YAPISAL DOĞAL DİL – ÖRNEK: FORM ESASLI TANIMLAMA

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: Safe sugar level

Description Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2), the previous two readings (r0 and r1)

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose – the dose in insulin to be delivered

Destination Main control loop

Action: CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requires Two previous readings so that the rate of change of sugar level can be computed. The

Pre-condition insulin reservoir contains at least the maximum allowed single dose of insulin.. r0 is

Post-condition replaced by r1 then r1 is replaced by r2

Side-effects None

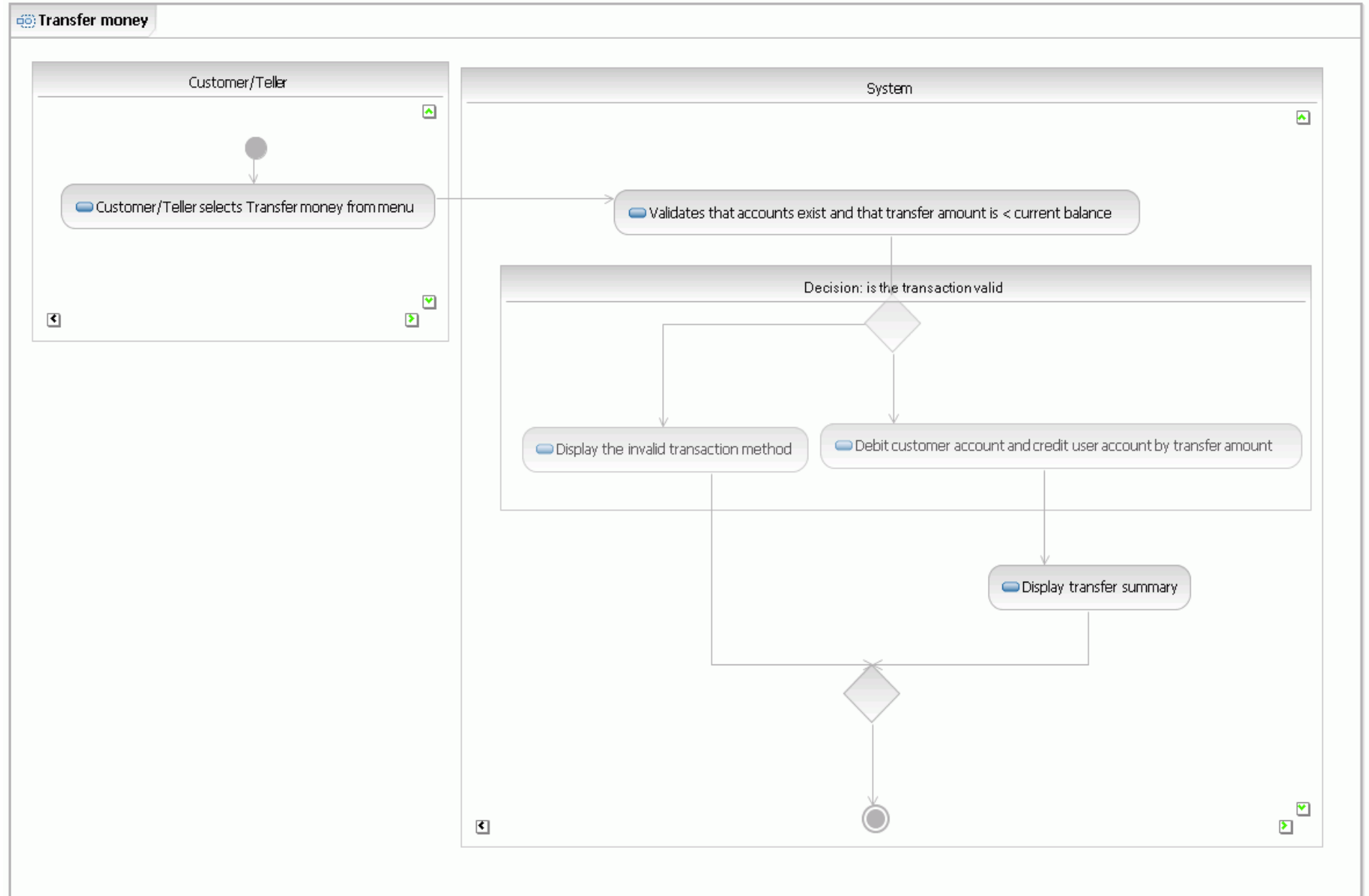
ARAYÜZ (“INTERFACE”) TANIMLAMA

- Geliştirilen birçok sistem, diğer sistemlerle birlikte çalışmak zorundadır. Birlikte çalışmayı sağlayacak arayüzler de gereksinimlerin bir parçası olarak tanımlanmalıdır.
- Tanımlanabilecek arayüz türleri:
 - ▶ Yordam arayüzleri
 - ▶ Değiş-tokuş edilecek veri yapılarının arayüzleri
 - ▶ Veri gösterimlerine ilişkin arayüzler
- Formal gösterimler, arayüz tanımlamaları için daha uygundur.

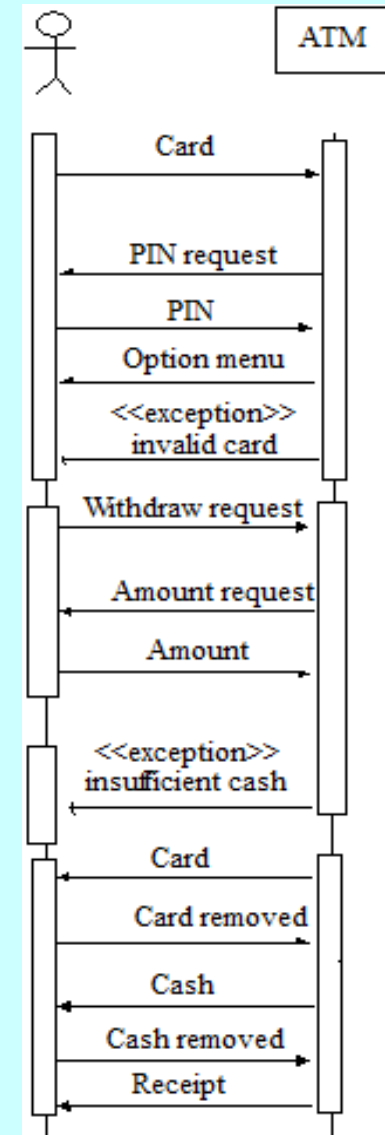
ARRAYÜZ TANIMLAMA – ÖRNEK: JAVA PDL ARRAYÜZ TANIMI

```
interface PrintServer {  
  
    // defines an abstract printer server  
    // requires:      interface Printer, interface PrintDoc  
    // provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter  
  
    void initialize ( Printer p ) ;  
    void print ( Printer p, PrintDoc d ) ;  
    void displayPrintQueue ( Printer p ) ;  
    void cancelPrintJob (Printer p, PrintDoc d);  
    void switchPrinter (Printer p1, Printer p2, PrintDoc d);  
} //PrintServer
```


Grafik Gösterim – Örnek: UML Etkinlik ("Activity") Diyagramı



Grafik Gösterim – Örnek: UML Ardıl İşlem (“Sequence”) Diyagramı



MATEMATİKSEL TANIMLAMA – ÖRNEK: Z GÖSTERİMİ

■ <http://archive.comlab.ox.ac.uk/z.html>

► *The Z notation is a formal specification notation based on set theory and predicate calculus.*

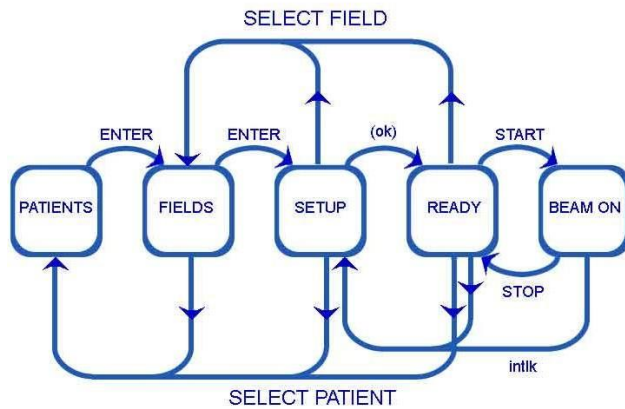


Figure 6.6: Therapy control cascade: state transition diagram

Graphic courtesy of kimberlybatteau.com

STATE ::= patients | fields | setup | ready | beam_on

EVENT ::= select_patient | select_field | enter | start | stop | ok | intlk

FSM == (STATE × EVENT) → STATE

no_change, transitions, control: FSM

control = no_change ⊕ transitions

no_change = { s: STATE; e: EVENT • (s, e) → s }

transitions = { (patients, enter) → fields,

(fields, select_patient) → patients, (fields, enter) → setup,

(setup, select_patient) → patients, (setup, select_field) → fields, (setup, ok) → ready,

(ready, select_patient) → patients, (ready, select_field) → fields, (ready, start) → beam_on, (ready, intlk) → setup,

(beam_on, stop) → ready, (beam_on, intlk) → setup }

GEREKSİNİM BELGESİ

- Sistemin sağlaması beklenen özelliklerin resmi tanımıdır.
- Hem kullanıcı gereksinimlerini, hem de sistem gereksinimlerini içermesi beklenir.
 - ▶ Bazı modeller bu ikisi için ayrı belgelerin oluşturmasını önermektedir.
- Mümkün olduğunca sistemin “ne yapacağını” tanımlamalı, “nasıl” yapacağı” detayına girmemelidir.

GEREKSİNİM BELGESİ TİPİK İÇERİĞİ

- Önsöz
- Giriş
- Tanımlar
- Kullanıcı gereksinimleri
- Sistem mimarisi
- Sistem gereksinimleri
- Sistem modelleri
- Sistem gelişimi
- Ekler
- Endeks

IEEE GEREKSİNİM BELGESİ STANDARDI

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- Gereksinim belgesi için, her sistem için özelleştirilebilecek genel bir yapı sunar.
 - ▶ “Introduction”
 - ▶ “General description”
 - ▶ “Specific requirements”
 - ▶ “Appendices”
 - ▶ “Index”

GEREKSİNİM ANALİZİ – KAZANÇLAR

■ Artan müşteri memnuniyeti

- ▶ Müşteri/kullanıcılar çalışmalara katılıyor
- ▶ İstekler tam ve doğru olarak tanımlanıyor

■ Artan yazılım kalitesi

- ▶ Gereksinimler doğru ve tam olarak tanımlanıyor
- ▶ Gereksinimleri tüm yazılım ekibi biliyor
- ▶ Gereksinim değişiklikleri en aza indirilebilecek

■ Etkin proje yönetimi

- ▶ Daha doğru tahminleme (takvim ve bütçe)
- ▶ Daha kolay izleme (gereksinimler esaslı)
- ▶ Daha düzgün iş atamaları (gereksinimler esaslı)

REFEREANSLAR

- Software Engineering (10th. Ed.); Ian Sommerville; 2015.
- Guide to Software Engineering Body of Knowledge (v3); 2014.
- Hacettepe Üniversitesi BBS-651, A. Tarhan, 2019.