Yazılım Geliştirmede Çevik Yöntemler

Hafta 6

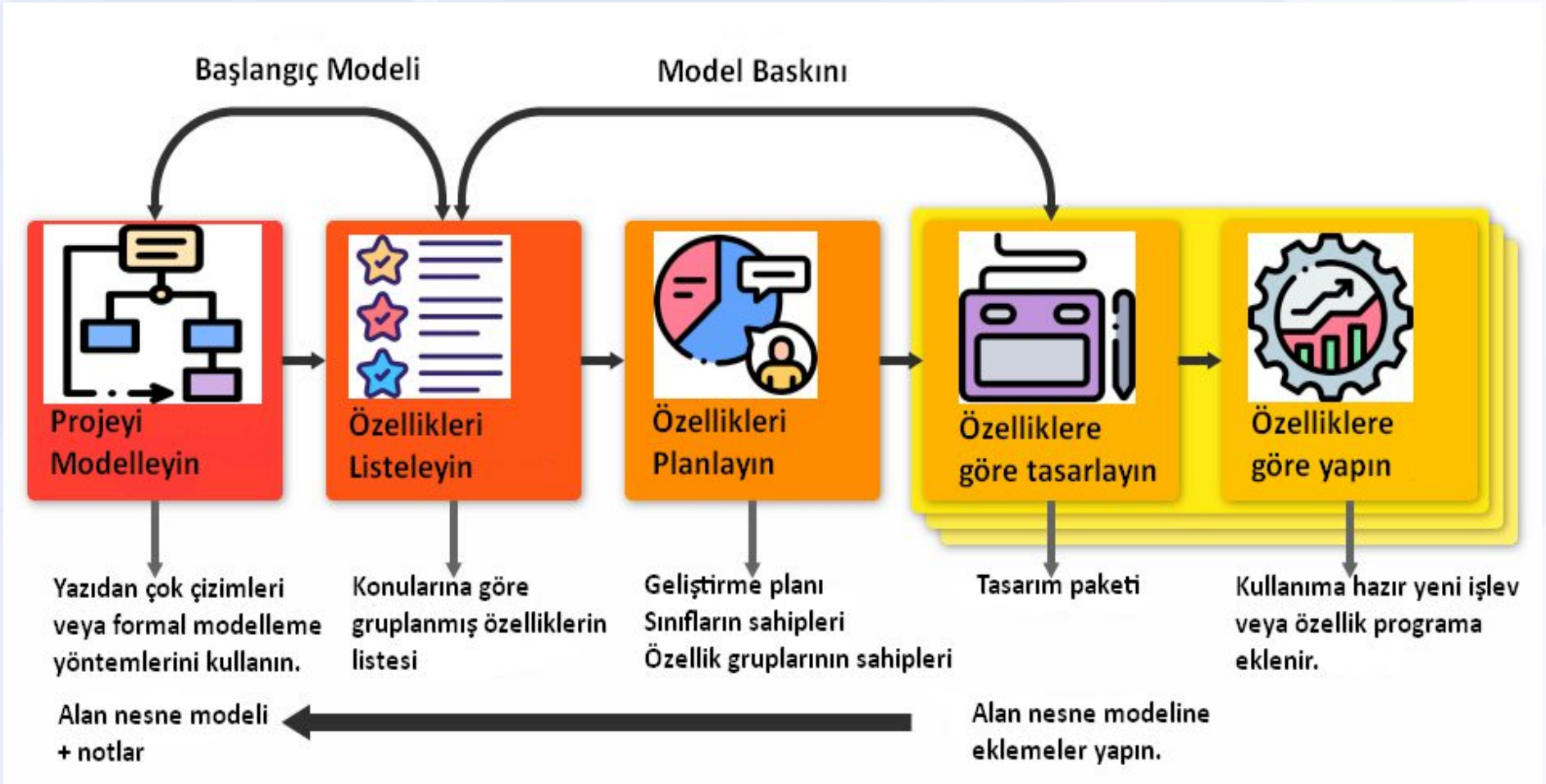
Özellik Güdümlü Geliştirme

Sibel Kuzgun Akın

Özellik Güdümlü Geliştirme

- *Feature Driven Development*
- Döngüsel ve artırımlı bir yazılım geliştirme yaklaşımıdır.
- Hafif (basit, az kurallı) ve çevik bir metodolojidir.
- Büyük ve uzun erimli projeler için uygundur.
- Birçok yazılımcının çalıştığı, kurumsal yazılım evlerinde uygulanabilir.
- En geç iki haftada, kod geliştirme döngüsü tamamlanmalıdır.
- Programa katılacak her bir özellik veya işlev için, bu işe en uygun kişilerden bir takım kurulur. Bunlar farklı görevleri üstenirler. Takım liderinin deneyim ve bilgisi, takımın başarısı için önemlidir. Analizi yapmak, diğerlerini yönlendirmek, takımındakilere danışmanlık vermek onun sorumluluğudur.
- Özellik: Eylem, sonuç ve nesneden oluşur. Programın yapacağı işlerden birinin tarifidir.
- Örnek: “İzin formunu çalışanın ilk amiri onaylasın.”

Özellik Güdümlü Geliştirme



Paydaşlar

Ana roller:

- Proje yöneticisi
- Baş mimar (Yazılım mimarı)
- Geliştirme yöneticisi
- Baş programcı
- Sınıf sahibi
- Alan uzmanı

Ne çok yönetici var! İşte o yüzden kalabalık ekiplerin çalıştığı kurumsal yazılım evlerine uygundur.

Yan roller:

- Alan yöneticisi
- Sürüm yöneticisi
- Dil uzmanı
- Üretim mühendisi *(Build engineer)*
- *Toolsmith* (Yazılım geliştirme araçları sorumlusu)
- Sistem yöneticisi
- Testçi
- Devreye alan *(Deployer)*
- Teknik yazar

Sorun Çıkarsa: Model Baskını

- İng.: *Model storming*
- XP'nin ayaküstü tasarım toplantısı gibidir. Tam gerektiği zaman (JIT: Just in Time) yapılır.
- Modelleme analiz veya tasarım için yapılabilir.
- Çözülmesi gereken bir sorun olduğunda, çözebileceğine inanan bir kişi birkaç ekip üyesini de yanına alıp, bir model geliştirir. Diğer yazılımcılar bu arada kendi işlerine devam eder.
- Çoğu zaman kağıtla kalemle veya tahtada tasarım yapılır.
- Kısa sürede (Yarım saatten az?) çözüm aranır.
- Analiz için ekran çizimleri veya tasarım için akış çizelgesi, UML, sınıf – sorumluluk- ortak çalışanlar kartları (XP'de kullanılıyordu) gibi yöntemler kullanılabilir.

JIT modeli neden işe yarar?

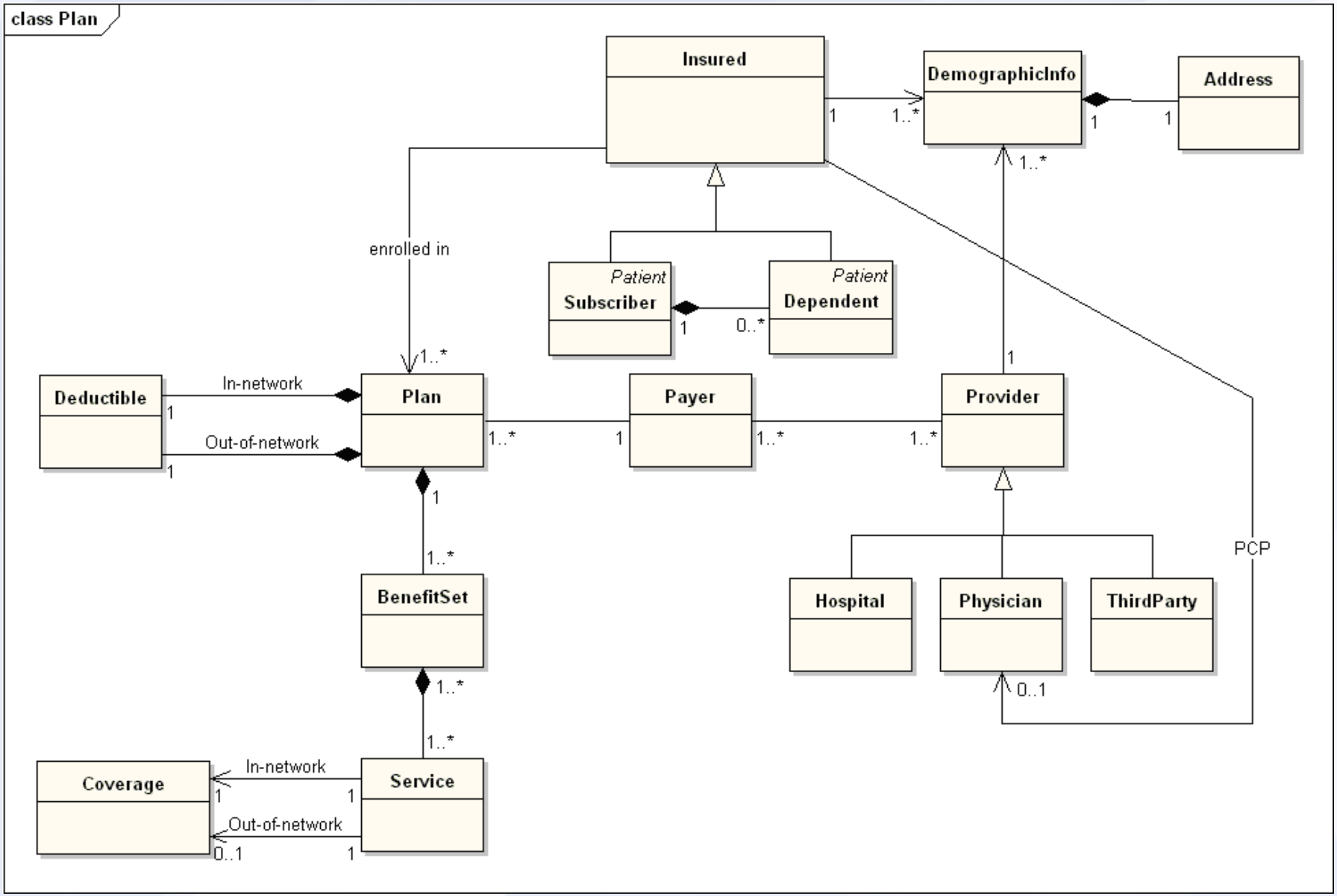
Tüm analizi projenin başında yapmak yerine, tam gerektiği zaman (JIT: Just In Time) analiz yapmak neden çoğu zaman daha yararlıdır?

- Proje ilerledikçe, alan bilginiz artar. Örneğin üç ay önce proje başladığında yapılamayan tasarımları yapabilecek hale gelirsiniz. Olayları daha iyi anlarsınız.
- Aynı zamanda yazılımın biten modüllerini teslim etmeyi sürdürdüyseniz, kullanıcılar da sistemi anlamaya başlar. Sorularınıza daha anlamlı yanıtlar verirler.
- Projenin en başında her şeyi modellemeye çalışmak zaman kaybıdır. İlk döngünün başında tabii ki ilk ihtiyaç analizi yapılır ve yazılım mimarisi ortaya konur. Ancak ayrıntılara dalmak için erkendir.

Alan-Nesne Modeli

- *“Domain object model”* veya “*domain model”*
- Kaynaklarda Alan Modeli olarak da geçiyor.
- Yazılım mühendisliğinde alan-nesne modeli üzerinde çalıştığınız alanın hem verilerinin, hem de nasıl davranacağının kavramsal tasarımıdır.
- Hangi alana ait sorun çözülecekse, o alanın keşfini ve açıklamasını içerir. Sonuçta ortaya çıkan alan nesne modeli, özelliklerin eklenebileceği toptan bir çerçeve sunar.
- Alan: Birbiriyle ilgili kavramlar, ilişkiler ve iş akışı bir araya toplanarak alanı oluşturur.
- Varlıklar (entities)
- Varlıklar ve aralarındaki ilişkiler (Entitiy – Relationship)

Alan Modeli Örneği: Sağlık Sigortası Planı



ER Çizelgeleri – Alan Nesne Modeli

- ER Çizelgeleri *(entity-relationship diagrams)*: Projede kullanılacak tüm gerçek dünya varlıklarını ve aralarındaki ilişkileri modellemek için kullanılırlar.
  - Veri tabanı tasarlamak için yararlıdırlar.
- Alan Nesne Modeli *(domain model)*: Projenin kapsadığı alanın ihtiyaçlarının analizidir.
  - Alan modeli veri yapılarının, veri akışının, işlevlerin, sınırlamaların ve kontrollerin çoğu zaman görsel anlatımıdır.
  - Alan modelini göstermek için UML kullanılıyorsa, sınıf diyagramı (class diagram) uygundur.

En İyi Uygulamalar

Müşteriye yarar sağlayan özellikler açısından bakarak, yazılım mühendisliğinin en iyi uygulamaları *(best practices)* kullanılır.

- Alan-nesne modeli veye alan modeli
- Özelliklerle geliştirme: Geliştirmesi iki haftadan uzun sürecek tüm işlevler parçalara ayrılmalıdır. Bu parçalara ayırma işi, her bir işlev, bir özellik olarak tanımlanacak kadar küçük olana kadar yapılır. Böylece doğru / istenen özelliklerin sisteme eklenmesi kolaylaşır.
- Bireysel sınıf (kod) sahipliği: Kodun birbirinden bağımsız parçaları veya kod grupları bir kişiye verilir. Örneğin nesne yönelimli programlama yapılıyorsa, bazı sınıflar bir kişiye ait olur. O kişi bu kodların tutarlı, performansı istenen düzeyde ve kavramsal bütünlüğe uygun olmasından sorumlu tutulur.
- Özellik takımları: Bir özellik takımı, küçük bir işi yapmak için birkaç kişinin anlık olarak bir araya gelmesiyle kurulur. Her tasarım kararında fikir alışverişi yapılır ve hangisinin yapılacağına karar vermeden önce, tasarım seçenekleri gözden geçirilir.

En İyi Uygulamalar

- İrdeleme *(inspection)*: Kodda hatalar çıktıkça, iyi kalitede tasarım ve koda ulaşmak için, bunlar irdelenir, masaya yatırılır...
- Ayarların yönetimi: Gelecekte iyileştirmeler yapacak takımlar için, bugüne kadar tamamlanan tüm özelliklerin kaynak kodunun bilinmesi ve sınıflarda yapılan değişikliklerin tarihçesinin tutulması. **Teknik dokümantasyon** bu metodolojide önemlidir.
- Düzenli aralıklarla kod üretme: Yazılım sistemindeki kodlar düzenli olarak derlenip / yüklenip sistem hep güncel tutulmalıdır. Bu güncel sistemin kullanıcılara gösterilmesine ve -varsa- entegrasyon hatalarının daha erken ortaya çıkmasına olanak sağlar. Herkes kendi sınıflarından sorumlu olduğu için, sınıfları bir araya getiririp entegrasyon testi yapmak önemlidir.
- İlerleme ve sonuçların görülebilirliği: Tamamlanan işler temelinde, projenin tüm seviyelerindeki ilerleme raporlanmalıdır. Bu raporlar sık, uygun ve sürekli olmalıdır. Ama bu yazılı iletişimin artmasına ve bürokrasiye de yol açar.

Kaynaklar

- Domain Modeling: What you need to know before coding
  https://www.thoughtworks.com/insights/blog/agile-project-management/domain-modeling-what-you-need-to-know-before-coding
- Systems and software engineering — Vocabulary International Standard ISO/IEC/IEEE 24765 First edition 2010-12-15
  https://cse.msu.edu/%7Ecse435/Handouts/Standards/IEEE24765.pdf
- Feature Driven Development and Agile Modeling
  https://agilemodeling.com/essays/fdd.htm
- Model Storming: An Agile Core Practice
  https://agilemodeling.com/essays/modelstorming.htm
- Domain Model
  https://en.wikipedia.org/wiki/Domain_model
- Process icons created by Freepik - Flaticon
  https://www.flaticon.com/free-icons/process