Günümüzde Yazılım Mühendisliği ve Yeni Açılımları

H. AKTÜRK
D. AYDIN
A. C. KINACI
T. E. KALAYCI

2007

Waterfall Modeli

- Waterfall modelinin yazılım mühendisliğinde kullanımı
- İki temel hatalı yaklaşıma sahip
- Süreç üzerinde iyileştirmeler
  - "Çöpe atmak için planla; zaten sonunda atacaksın"
  - Winton Royce tarafından gerçekleştirilen iyileştirmeler
- Nihayet waterfall modelinin yazılım mühendisliği için uygun bir yaklaşım olmadığı anlaşılmış.

The Mythical Man Month

- Yazılım mühendisliği metodolojilerine savaş açılmaya başlandığı 1975'lerde Brook bir kitap yazdı. Akademik temelli değil de piyasada yıllardır çalışan birisi olarak yazılım geliştirme sürecinin mühendislik odaklı değil insan odaklı bir süreç olduğunu iddia ediyordu. Brook'un bu kitabı yeni metodolojisinin temelini teşkil ettiği gibi diğer metodolojilerin çıkmasına da ön ayak oldu.

The Mythical Man Month

- Yazılım Geliştirme Projelerinde Temel Unsurlar
  - The Tar Pit
  - The Mythical Man-Month
  - Brook's Law
  - The Second System Effect
  - Neden Babil kulesi yıkılır
  - Bir adım ileri ve bir adım geri

1975'den Bu Yana

- IDE ve OO (Bilgisayar destekli araçlar ve Nesneye yönelik programlama)
  - Gerçekleştirilen IDE'ler ile her uzmanın kendi gerçekleştirdiği CASE tool'lar tarihe karıştı.
  - Nesneye yönelimli programlama ile sürece esneklik kazandırıldı.
- UML
  - İnsanlar en önemli faktörün insan ve aralarındaki iletişim olduğunu anladılar. Başarılı bir projenin sadece sözel bilgi paylaşımı ile değil bilgisayarda gerçekleştirilen işlemlerin iyi dökümante edilmiş olmasıyla ortaya çıktığı görüldü. OO programlamanın getirdiği avantajla profesyoneller Unified Modelling Language adı verilen bir sistemi ortaya attılar. Böylece bir yazılım sistemleri kendi notasyonlarına kavuşmuş oldular ve iletişim büyük oranda güç kazandı.

The Mythical Man-Month Methodology

- Brooks kitabında sadece bazı önemli noktalardan değil ayrıca yeni bir metodolojiden de bahsetmekteydi. Bu metodolojinin özü "Az fazladır" presibidir.
- Temel olarak beş bileşenden oluşur.
  - Proje Dizaynı — Kavramsal bütünlük
  - Proje Yapısı
  - Proje Gerçekleştirimi - "Plan to Trow One Away"
  - Proje İletişimi - "The Documentary Hypothesis"
  - Proje Organizasyonu – "Plan the Organization for Change"

The Mythical Man-Month Methodology

- "Kavramsal bütünlüğü gerçekleştirilmiş bir sistem hızlı yapılandırılıp test edilir"
- İki temel görüşe sahip:
  - Dizayn bir kişi veya küçük bir grup tarafından yapılmalıdır. Tüm komitenin dizayna çalışması felakete kapı açar.
  - Sistemin alt yapısını kuranlar ile gerçekleştirenler farklı kişiler olmalıdır.
- Şu anki yazılım projeleri kullanıcı yazılımları ve programcı yazılımlarıdır. Programcı yazılımlarının proje dizaynı ikinci görüşe zıttır.

The Mythical Man-Month Methodology

- Proje Yapısı
  - Fazla kişi çok iş demek değil belki az iş demektir. Brook bunun için iki parça çözüm önerisi getirir;
    - Büyük grupları küçük gruplara bölmek.
    - Her bir bileşen ve nesne bir gruba atanmalıdır.

The Mythical Man-Month Methodology

- Proje Gerçekleştirimi - "Plan to Trow One Away"
  - Proje geliştirdikçe sistem gereksinimleri değişir ve programcı problem kümesini daha iyi anlayabilir hale gelir. Bu yüzden Brook "önce planlama yap ama üzerine gitme vaz geç ve yeniden dizayn et" görüşünü savunuyordu.
  - Brook'un bu fikri daha sonra değişti. Bunda IDE'ler, application server'lar, yüksek seviyeli diller ve tekrar kullanımlı nesneler sayesinde yazılımın iteratif ve hızlı bir şekilde büyüme kaydedebileceği örneklerini görmesi etkili olmuştur.
  - Yine görüşünün değişmesinde gözüne çarpan birkaç metot etkili olmuştur.
    - Simulatorlar ve prototipler
    - Microsoft'un "Build Every Night" yaklaşımı

The Mythical Man-Month Methodology

- Proje İletişimi - "The Documentary Hypothesis"
  - Brook altı tane temel dökümantasyondan bahseder ve bunlar kim, ne, ne zaman, nerede ve ne kadar sorularına cevap verirler.
  - Brook bunun için bir proje çalışma kitabının hazırlanması gerektiğini böylece iletişimin ve doküman arşivinin sağlanacağından bahseder. Fakat internet ve web'in gelişmesiyle bir çok kullanışlı araçta çıkmıştır. Wiki'ler ve Web-tabanlı içerik yönetim sistemleri bunlara birer örnek olabilir. Ayrıca bu dokümanların hazırlanmasında UML'in önemini göz ardı etmemek gerektir.

The Mythical Man-Month Methodology

- Proje Organizasyonu – "Plan the Organization for Change"
  - "There is nothing in this world constant but inconstancy."
  - Sistemde ileriki aşamalarda değişime gidildiği gibi proje organizasyonunda da değişikliğe gidilmesi gerekir.
  - Yazılım geliştirmede kullanılan metot bir çok esnekliğe sahip olmalıdır. OO geliştirim ve surgical team yaklaşımı bu dizayn ve organizasyonel esnekliği sağlar.

Kavramsal Bütünlük

- Uygulamanın tutarlılığı ve kullanım kolaylığı
- Kullanıcının gözlemi önemlidir
- Çoğu yazılım takımlar tarafından geliştirilmektedir
- Büyük projeler, küçük projelerden farklı yönetilmelidir
- Temel amaç üründe kavramsal bütünlüğü sağlamak

Mimar kimdir?

- Ürünün genel akli modelini biçimlendirme
- Kullanıcılara nasıl kullanılacağını anlatma
- Kullanıcı ve geliştiriciler arasındaki bağlantığı sağlama
- İşlevsellik, başarım, maliyet, büyüklük ve zaman arasındaki dengeyi sağlama ve koruma

Mimar kimdir?

- Ürünü kullanıcının algıladığı şekliyle gerçekleştirini ayrılmalıdır
- Mimari ve gerçekleştirim ayrılmalıdır
- Büyük sistemler alt sistemler şeklinde gerçekleştirilmelidir
- Alt sistemler için bağımsız mimar ve tüm sistem için ana mimar gereklidir

Featuritis[1][2][?]

- Geniş kullanıcı kümesine sahip yazılımlarda her kullanıcıyı tatmin etmeniz gerekir
- Her istenen özelliği gerçekleştirirseniz uygulamanız özellik zengini (oburu?) olur
- Araç şişmanlar

[1] "Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away." — Antoine de Saint-Exupéry

[2] "Make everything as simple as possible, but not simpler." — Albert Einstein

Geleneksel çözüm

- Sorun: Hangi özellikleri öncelikli olarak gerçekleştirmeliyiz?
- Günümüzde bu sorun (bir nebze de olsa) çözülmüştür.
- Peki nasıl?
  - Hata ve değişiklik takip sistemleri
  - Geliştirici takımın öncelikleri
  - Kritik hataların öncelikleri
  - Oylama sistemi
  - Açık kaynak yazılım değişim yönetimi
  - "Benevolent Dictator for Life (BDFL[**?**])"

Diğer bir çözüm

- Eklenti mantığı
  - Eclipse[?] en önemli örnektir
  - Kullanıcılar kendilerine özgü bir ortamı eklentilerle sağlıyorlar
  - Kendinize özgü bir GNU/Linux[?] dağıtımı da bir başka örnektir.
  - Bir FLOSS[?] başarısı (Free/Libre/Open-Source Software)
  - Başarılı Örnekler: GNU/Linux, Apache, FreeBSD, JBoss, Eclipse,....[?]
  - Katedral ve Pazar karşılaştırması[?]

Çevik Yaklaşımlar

- Dijkstra'nın manifestosuna zıt olarak ortaya çıkmıştır[**?**]
- Çevik yaklaşımın anahtar katkısı 4M insan odaklı yaklaşımı açması ve genişletmesiydi.
- Yazılım geliştirme sürecindeki insan faktörünün anlaşılması ve üzerine gidilmesinin ve nesne yönelimli geliştirmenin üzerinde vurgu yapmışlardır.

Extreme Programlama

- Çok sayıda pratik içeren bir yöntemdir.[**?**]
- Agile manifestoya bağlı kalarak insan odaklı yaklaşıma sahiptir
- Geliştirme küçük adımlar şeklinde yapılmaktadır.
- İkili programlama yapılmaktadır.
- Kullanıcı hikayeleri belirtim birimidir.
- Başarıya ulaşması için çok sayıda pratiğin uygulanması gereklidir.

Crystal

- Alistair Cockburn tarafından kullanılır.
- "Proje tipine göre metodu ayarla"
- Temelinde yazılım geliştirmeyi yaratım ve iletişim gerektiren bir oyun olarak ele alır. Birincil hedef olarak işe yarar, çalışan yazılım üretmek, ikincil hedef olarak sonraki "oyun" için hazırlanmak
- Buna dayanarak farklı projelerde kişilerin farklı psikolojiye göre hareket etmeleri gerekir.

"Free/Libre/Open-Source Software"

- Her hangi bir yöntembilim profesyonellerine sahip değildir.
- Katedral ve Pazar[**?**] adlı manifesto temelini oluşturur.
- Bu tip projeler insan odaklıdır.
- Bütçe yoktur, kaynak kısıtlı bir oyun mantığı yoktur.

Model Tabanlı Yazılım Geliştirme

- Amaç soyutlamanın seviyesini arttırmak
- MDD için öncelikli motivasyon üretkenliği arttırmaktır.
  - Gerçekleştirilen işlevselliğin arttırılmasıyla geliştiricilerin kısa süreli üretkenliğini arttırır.
  - Ürünün eskimişlik oranını düşürerek geliştiricilerin uzun süreli üretkenliğini arttırır.
- Önemli olan bir özellik değişime duyarlılıktır. Dört temel değişim biçimi tanımlanmaktadır.
  - Personel
  - Gereksinimler
  - Geliştirme ortamları
  - Konuşlandırma ortamları

Model Tabanlı Yazılım Geliştirme

- MDA OMG tarafından geliştirilmiştir. UML tabanlı modellerin farklı platform ve soyutlama seviyelerine dönüştürülmesini açıklar[**?**].
- MDD için üç aşamalı bir çatıdır
  - Platform bağımsız model oluşturulması
  - Platforma özgü modele dönüştürme
  - Kodun oluşturulması

Model Tabanlı Yazılım Geliştirme

- MDA'ya ek olarak yazılım geliştirme sürecini otomatikleştiren araçlar vardır.
  - Rational Rose UML modelden kaynak kodu üretme
  - AndroMDA[**?**]
  - OpenMDX[**?**]
- MDA sayılan işlem model dönüşümleridir. Diller vardır.
  - QVT, ATL, VIATRA, GReAT, Tefkat, Kermeta ve MT[**?**]

Sorular?

Sorular?


http://en.wikipedia.org/wiki/Featuritis


http://en.wikipedia.org/wiki/BDFL


http://www.eclipse.org/


http://www.gnu.org/gnu/linux-and-gnu.html


http://en.wikipedia.org/wiki/FLOSS


How to Have a Successful Free Software Project http://portal.acm.org/citation.cfm?id=1032630.1032710


http://www.catb.org/~esr/writings/cathedral-bazaar/


http://www.agilemanifesto.org/


http://www.extremeprogramming.org


http://www.omg.org/mda/


http://www.andromda.org/

- http://www.openmdx.org/
- http://en.wikipedia.org/wiki/Model_Transformation_Language