Gelecek Fonksiyonel!

Chris Stephenson

Department of Everything
University of Life

created May 4, 2019

Not

Herşeyi tekrar etmek istemediğim için...
`https://chrisstephenson.org`

Yazılım ciddi iş...

Yazılım ciddi iş...

Yazılım ciddi iş...

Giriş

- İki defa oldu - toplam 346 ölü
- Uçak tam gazlı yere çakıldı.
- Pilotlar uçağı yukarı kaldırmaya çalışıyorlardı.
- *Yazılım* izin vermedi
- Kılavuzda uçağın bu "özelliği" yazılmıyordu
- "Undocumented Feature"

Yazılım ciddi bir iş

- Programların doğru çalışması
- Toplumsal sorumluluklarımız
- Kaynak israf etmemek
- Gereksiz sera gazlari üretmemek.
- İnsanlığa fayda sağlamak/zarar etmemek

What is Going to be Hot? (2017 yılında dediklerim...)

İki sene önce ne dedik?

What is Going to be Hot? (2017 yılında dediklerim...)

Functional Programming?

What is Going to be Hot? (2017 yılında dediklerim...)

- Haskell at Facebook
- Ocaml at Jane Street
- Scala in lots of places
- Clojure etc etc...
- node.js....interesting

What is Going to be Hot? (2017 yılında dediklerim...)

- Functions are first class values
- No state, no assignment
- Referential transparency
- Closures, Higher order programming, function transformations
- Optionally, laziness, powerful type systems

What is Going to be Hot? (2017 yılında dediklerim...)

- As a result...
- programs are up to ten times shorter..

Hep bir “yazılım sorunu” vardı...

Tarihi (ve kişisel) bir perspektif

Hep bir “yazılım sorunu” vardı...

- Proje beklenenden uzun sürer
- Program düzgün çalışmaz
- Program bakımı büyük bir yük olur
- Projelerin %50’si tamamlanmadan iptal olur

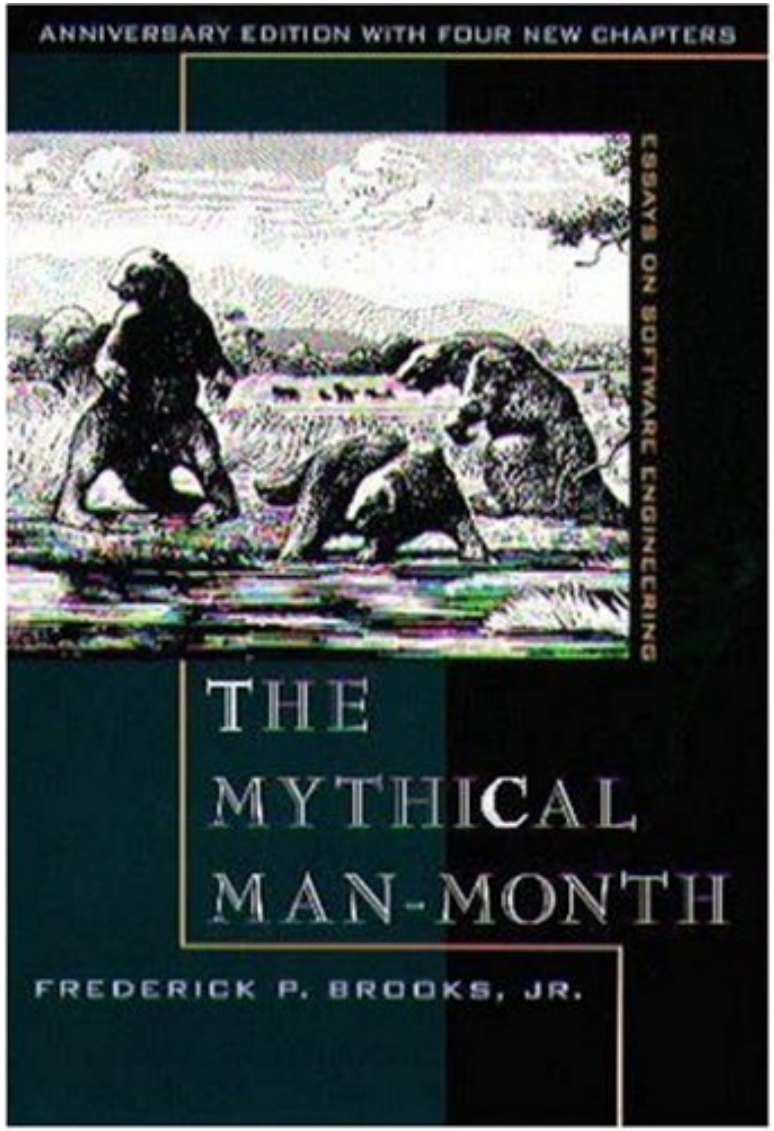
Tarihi (ve kişisel) bir perspektif

Hep bir “yazılım sorunu” vardı...

- Proje beklenenden uzun sürer
- Program düzgün çalışmaz
- Program bakımı büyük bir yük olur
- Projelerin %50’si tamamlanmadan iptal olur

Bakınız: “The Mythical Man Month” F P Brooks (1975!)

Tarihi (ve kişisel) bir perspektif



Tarihi (ve kişisel) bir perspektif

Her zaman "yazılım sorun" çözülmeye çalışılıyordu.

- "Yüksek Düzey Diller" (1950ler)
- "GoTo considered harmful" (1968)
- "Yapılandırılmış Programcılık" (1970ler)
- "Nesneye Yönelik Diller" (1980lar)
- "Design Patterns" (1994)

Çözümler neleri çözmeye çalışıyordu

- Kodun yazılmasın kolaylaştırmak
- Kodun taşınabilirliği
- Kodun azaltılması
- Kodun yeniden kullanılabilmesi
- Kodun okunabilirliğini iyileştirmek.
- “State” idare etmek

Tarihi (ve kişisel) bir perspektif

Her zaman begenmeyenler vardı.

- “Yüksek Düzey Diller” John Von Neumann
- “GoTo considered harmful” considered harmful
- ve devam...

Tarihi (ve kişisel) bir perspektif

"yazılım sorun" çözmek için....

- 1948-1975 makine dili.
- 1958-1970 Fortan, Cobol
- 1958 LISP (AI ve diller teorisi)
- 1974-1985 C, Pascal gibi diller
- 1980 Smalltalk
- 1991- C++. (ve evrimi)
- 1995- Java (ve evrimi)
- 1995- Javascript (ve evrimi)

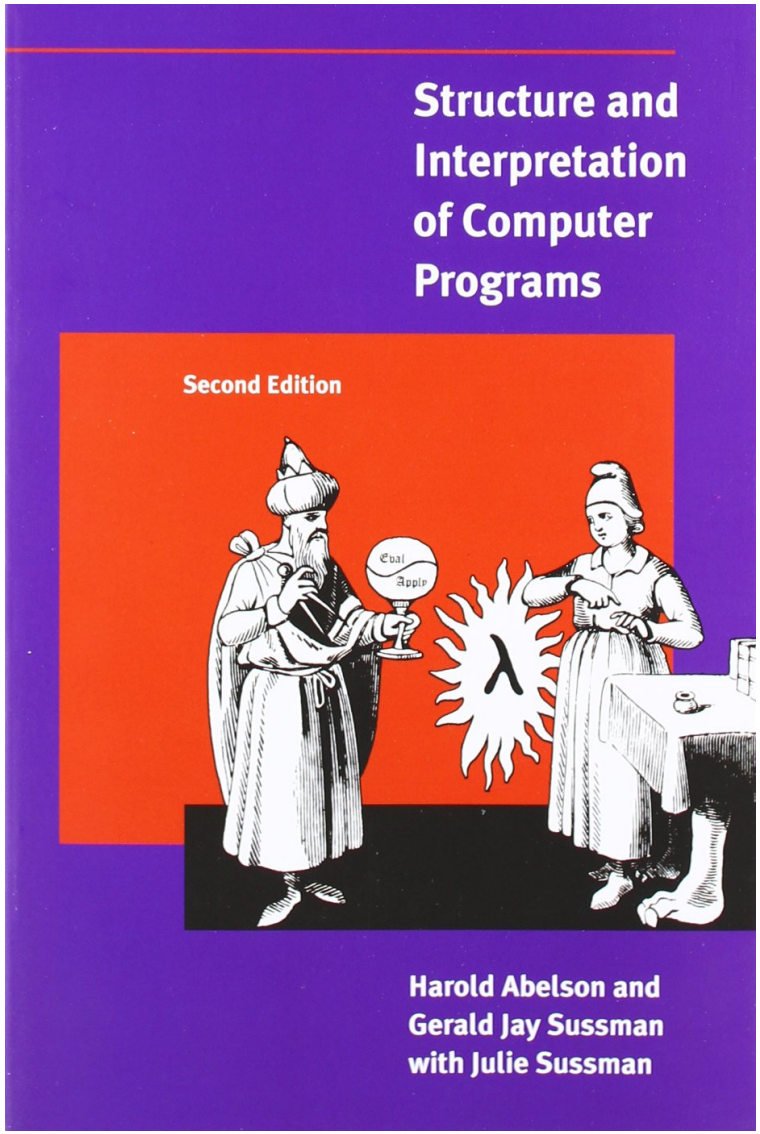
Kişisel deneyimim

- 1966-1979 makine dili.
- 1970-1974 AI için Prolog, POP-2, LISP
- 1980-1985 C, Pascal gibi diller, ilkel işletim sistemler
- 1985-1991 C, Pascal, MS Windows
- 1991-1999 C++. Windows NT,
- 1997 SICP! Scheme, Racket, Haskell

Kişisel deneyimim - sorunlar

- Tekrar aynı program yazmak
- C++ bloat
- C++ hafıza sızıntıları
- Java code bloat
- Java öğrenim sorunu

Tarihi (ve kişisel) bir perspektif



Fonksiyonel nereden çıktı?

Fonksiyonel baştan vardı...

- 1935: Çağdaş bilgisayar bilimleri Church-Turing Tez ile başlıyor.
- Turing - Turing Makinesi
- Church - λ-calculus

Fonksiyonel nereden çıktı?

Bazı λ-calculus cümleleri...

- x
- (x y)
- (λ x x)
- ((λ x x) a)
- (λ x (λ y x))
- ((λ x (λ y x)) a)
- (((λ x (λ y x)) a) b)
- (λ x (λ y y))
- (λ f (λ x (f(f(f x)))))

Fonksionel nereden çıktı?

λ-calculus ispat edilen önemli özellikleri..

- Church-Rosser niteliği - değerlendirme sırası sonucu değiştiremez.
- Bir ifade soldan değerlendirildiğinde bitmesi mümkünse biter.
- Bu ifadeleri değerlendirerek, bilgisayarla yapılabilen her hesap yapılabilir.
- λ-calculus basit cebirsel tipler eklenerse her değerlendirme biter. Yani sonsuz döngü olamaz.

Çoğu programlama dilinde olmayan bu nitelikleri önemli. Bundan dolayı bir dizi dil için λ-calculus referans olarak alınır.

Not. Ağustos ayın son haftasında Nesin Matematik Köyünde λ-calculus dersim var...

Fonksiyonel nereden çıktı?

1958 yılında LISP dili doğrudan λ-calculus'dan çıktı.

- LISP ilk "Yapay Zeka" ilkbaharında neredeyse herkesin ortak dili olur.

Hatırlatalım

Fonkisyonel nasıl birşey?

- Functions are first class values
- No state, no assignment
- Referential transparency
- Closures, Higher order programming, function transformations
- Optionally, laziness, powerful type systems

Bakmak anlatmaktan daha kolaydır

Fonksiyonel nasıl birşey?

Kullanacağım Racket dili LISP'in torunu. LISP'e benzer ve her ikisi λ-calculus'e benzer.
Bizim için önemli olan...

- fonksiyonları λ ile yazabiliriz, ama kolaylık olsun birden fazla parametre yazabiliriz, ve define kullanabiliriz
- Fonksiyon uygulamak böyle yapılır (f a b) - Fonksiyon f 'e a ve b sembollerin değerlerini parametre olarak veriliyor
- Önceden sembollerien tiplerini belirlemiyoruz. Sayılar var mantıksal değerler var stringler var, şimdilik bize yeter.
- Alternatif değerler arasında seçmek if .. else if .. else if ... else benzeyen bir cond ifademiz var.

Fonksiyonel nasıl birşey?

Şimdi elimize bu kadar basit bir dille ne kadar büyük bir güç geçtiğini görelim

- Her (tek değişkenli) denklemi çözen bir program yazalım
- yani herhangi bir f için $f(x) = 0$ denklemi doğru yapan x değerlerini bulan bir program yazalım.

Gerekli matematik bilgisi
Newton yöntemi: f fonksiyonun kökü için bir tahmin t ise, daha iyi bir tahmin, t_0, böyle bulunur

$$t_0 = t - \frac{f(t)}{f'(t)}$$

Gerekli matematik bilgisi Fonksiyon f in türevi:

$$\lim_{\delta x \to 0} \frac{f(x + \delta x) - f(x)}{\delta x}$$

Gerekli matematik bilgisi
Newton (Horner) polinom denklemi:

$$a_0+a_1x^1+a_2x^2\ldots a_nx^n = a_0+x(a_1+x(a_2+x(\ldots x(a_n)$$

Hadi biraz pratik yapalım

- Mecburen küçük programlar yazacağız.
- En az bir fikir elde edelim

Biraz Haskell. Haskell'in ekledikler arasında

- Farklı bir görünüm
- Algebraic statik tipler
- Tembellik

Bunları bir görelim....

Biraz Haskell. Hello Haskell World!
Bunu bir görelim....

Başka

Biraz Haskell. Algebraic types

- A Quick Sort

Bunu bir görelim....

Biraz Haskell. Tembellik

- Factoriallar? (Hepsi!)
- Fibonaççiler? (Hepsi!)
- Asal sayılar (Hepsi!)

Bunu bir görelim....

Başka

Bu makale güzel

Under consideration for publication in J. Functional Programming

The Genuine Sieve of Eratosthenes

Melissa E. O'Neill
Harvey Mudd College, Claremont, CA, U.S.A. (*e-mail:* `oneill@acm.org`)

Abstract

A much beloved and widely used example showing the elegance and simplicity of lazy functional programming represents itself as "The Sieve of Eratosthenes". This paper shows that this example is *not* the sieve, and presents an implementation that actually is.

The Genuine Sieve of Eratosthenes Melissa E. O'Neill, Journal of Functional Programming.

Klasik 1. sınıf C sorunu....

- Girilen iki numeranın toplamını hesaplayan bir C programı yaz
- Her 1. sınıf öğrenci gibi cevabını internetten alalım...

Yaklaşım önemli....
"Programlam dili sekterliğine mahal yok!"

Fonksiyonel yaklaşımın etkileri

- RESTful API - statelessness
- big data Map / Reduce
- Algebraic Types (Rust, Typescript)
- Higher Order Programming
- Serverless
- templates, interfaces, iterators, parametrised types

Fonksiyonel yaklaşımın etkileri

- Model View Controller
- Zeynep'in oyununa bir bakalım. (benimki de!)

Richard Stallman on LISP

"The most powerful programming language is Lisp. If you don't know Lisp (or its variant, Scheme), you don't know what it means for a programming language to be powerful and elegant. Once you learn Lisp, you will see what is lacking in most other languages."

Richard Stallman on LISP

"Lisp is no harder to understand than other languages. So if you have never learned to program, and you want to start, start with Lisp."

İlgin varsa...

Kaynak çok...hepsi online ve bedava

- htdp.org “How To Design Programs 2e”
- “Structure and Interpretation of Computer Programs” (google SICP)
- “Learn You a Haskell for a Great Good”
- Rust, Typescript ve saire
- Akademik: “Purely Functional Data Structures” Chris Okasaki

Caveat Emptor....

Görüşmek üzere...

- 5 Mayıs Ankara ACM ACS days
- 7 Mayıs ODTU bilgisayar günleri
- 11 Mayıs (saat 10) Özgür Yazılım Günleri, İstanbul Bilgi Üniversitesi "Rastgelelik, Güvenlik , Güvenilirlik ve Gelecek"

Caveat Emptor....

The End