

Nesneye Yönelik Yazılım Mühendisliđi (376)

Dr. Öğr. Üyesi Ahmet Arif AYDIN

No Silver Bullet !

- ❖ Yazılım problemleri
 - ❖ hatalı yönetim (*wrong software management*)
 - ❖ kaçırılan teslim tarihi (*missed schedules & deadlines*)
 - ❖ bütçe ve ödeme problemleri (*payment & budget problems*)
 - ❖ hatalı - kusurlu ürünler (*flawed products*)

- ❖ Bahsedilen problemler software projelerini canavarlara dönüştürebilir!
- ❖ Bu canavarı ortadan kaldırmak için bir sihirli değnek yok !
- ❖ **(NO SILVER BULLET) !**



No Silver Bullet !

- ❖ Yazılım problemlerini ortadan kaldırmak için yapılması gerekenler :
 - ❖ Replacing the demon and humors theories by the **germ theory**
 - ❖ **Stepwise progress** (aşamalı olarak ilerleme)
 - ❖ **Persistent** (*ısrarcı*) & **unremitting** (*aralıksız*) effort

Essence and Accidents

❖ ESSENCE

- ❖ *Yazılımın doğasından kalıtsal olarak devralınan ve görmezden gelinemeyen zorluklar (irreducible challenges that inherited from the nature of software)*

1. *complexity* (karmaşıklık)

2. *conformity* (uyum)

3. *changeability* (değişim) - constantly changing environments

4. *invisibility* (soyut olması- görünmezlik) (*un-visualizable, not tangible*)

❖ ACCIDENTS

- ❖ *Ürün (product) ve alan (domain) ile alakalı zorluklar*
- ❖ **Özel bir alan ve proje ile alakalı olarak ortaya çıkabilen problemler** (*product related , domain specific problems*)

Accidents (Alana Özel Problemler İçin)

1. High-level languages

- ❖ *productivity, simplicity, comprehensibility and reliability*
- ❖ **Makine dili ile bit seviyesinde işlem gerçekleştirilirken yüksek seviyeli diller veri tipleri, işlemler gibi soyut yapıları kullanarak problemin çözümünü sağlar.** (*Concrete machine program focused on bits, registers, conditions, branches and channels high-level (abstract) programs consist of conceptual constructs such as operations, data types, sequences and communication*)
- ❖ **Yüksek seviyeli diller düşük seviyeli dillerin karmaşıklığını azaltmıştır** (*High level languages reduced whole low level code complexity with abstract & conceptual constructs*)

Accidents (Alana Özel Problemler İçin)

1. High-level languages

2. Time sharing

- ❖ 1960'lı yıllarda ortaya çıkmıştır
- ❖ Programcıların ortak ve aynı anda kaynakları kullanmasını ve çalışmasını sağladığından daha yüksek kalite ve daha hızlı bir biçimdeki üretimi desteklemektedir.
- ❖ 'the distribution of a computing resource to many users via *multiprogramming* or *multitasking*' (<https://www.techopedia.com/definition/9731/time-sharing>)

Accidents (Alana Özel Problemler İçin)

1. High-level languages
2. Time sharing
3. Unified programming environments
 - ❖ Birleştirilmiş programlama ortamları
 - ❖ **Unix-Interlisp** dealt with accidental difficulties that arise from using individual programs together.
 - ❖ Solutions
 - ❖ integrated libraries
 - ❖ unified file formats
 - ❖ pipes and filters

Öz ile akalı umut verici çalışmalar

Promising attacks on Conceptual Essence

(1) *By versus build*

- ❖ yazılım dünyasında bir problemin çözümünü gerçekleştiren ve kullanılan bir ürünü **yeniden yazmak yerine satın almak** (*don't try to reinvent the wheel again*)

Öz ile akalı umut verici çalışmalar

Promising attacks on Conceptual Essence

(2)-a İhtiyaçların yenilenmesi(*Requirements refinements*)

- ❖ Kullanıcılar tarafından tam ve kesin olarak ihtiyaçları belirlenmesi imkansızdır. (*it is really impossible for a client to specify completely, precisely and correctly the exact requirements of product before using*)
- ❖ Ne geliştirileceğine tam olarak karar vermek (*Deciding exactly what to build*)
- ❖ detaylı teknik ihtiyaçları oluşturmak (*establishing detailed technical requirements*)

Öz ile akalı umut verici çalışmalar

Promising attacks on Conceptual Essence

(2)-b hızlı prototip geliştirilmesi (*rapid prototyping*)

- ❖ Bir prototip yazılım sistemi, önemli arayüzleri simüle eder ve beklenen sistemin ana işlevini yerine getirir ve istisnai görevleri yerine getiremez. (*A prototype software system simulates important interfaces and performs main function of expected system and can't handle exceptional tasks*)
- ❖ Kavramsal yapılar, doğru ve kusursuz bir şekilde bir defada gerçekleştirilemeyecek kadar karmaşıktır. (*Conceptual structures are too complex to be specified accurately and built faultlessly.*)
- ❖ Bir prototip tutarlılık ve kullanılabilirlik için belirli bir ürün için müşteriye bir test fırsatı sunar. (*A prototype gives a test opportunity for a specified conceptual structure to client for consistency and usability*).

Öz ile akalı umut verici çalışmalar

Promising attacks on Conceptual Essence

(2)-c hızlı prototip geliştirilmesi (*rapid prototyping*)

- ❖ Artan bir biçimde ve genelden özele yazılımın geliştirilmesi tavsiye edilmiştir.
- ❖ *Any software system should be grown by incremental development.*
- ❖ *A program fist must run that doubles efforts and encourage the team for development then will be fleshed out bit by bit. This approach is a **top-down design**, and allows easy backtracking and like prototype of a system (Harlan Mills))*

Öz ile akalı umut verici çalışmalar

Promising attacks on Conceptual Essence

(3) Great Designers!

- ❖ *Zayıf ve İyi tasarımlar* arasındaki fark tasarım metotlarının kullanımı ile alakalıdır (*Difference between good and poor design caused by "design methods"*)
- ❖ **Mükemmel ve iyi tasarımlar** arasındaki farkı ise **mükemmel tasarımcı beyinler** tarafından gerçekleştirilir (*Difference between great and good design comes from great designers (designing minds) because software construction is a "creative" process*)

Öz ile akalı umut verici çalışmalar

Promising attacks on Conceptual Essence

(3) Great Designers!

❖ Great designer!

- ❖ provide faster, smaller, simpler, cleaner products with less effort
- ❖ important as much as great managers
- ❖ not the most experienced ones

❖ Mükemmel Tasarımcılar nasıl yetiştirilir

- ❖ identifying top designers and assigning a career mentor for them
- ❖ providing opportunities to designers stimulate, interact, and communicate each other
- ❖ technology transfer and curriculum development.

No Silver Bullet: Hopes for the SILVER !

- (1) Ada programming language
- (2) Object-oriented programming
- (3) Artificial Intelligence
- (4) Expert System
- (5) Automatic Programming
- (6) Graphical Programming
- (7) Program Verification
- (8) Environments and Tools
- (9) Workstations

No Silver Bullet: Hopes for the SILVER !

(1) Ada programming language

- ➡ Yüksek seviyeli genel amaçlı programlama dili (*general purpose high-level-language of 1980's*)
- ➡ ADA dilinin modüler yapısı , soyut veri tipi , hiyerarşik ve yapısal programlama kavramları dilin kendisinden daha gelişmiştir. (*Ada's philosophy of modularization, ADT, and hierarchical structuring are more advanced than the language itself.*)

No Silver Bullet: Hopes for the SILVER !

(1) Ada programming language

(2) **Object-oriented programming**

- ➡ Soyut veri tipleri nesne oluşturmayı ve oluşturulan ve detayları gizlenen nesnelere erişimi kolaylığı sağlar (*Abstract Data Types allows object's type is defined by a name, a set of proper values and operations instead of its storage structure which should be hidden*)
- ➡ 1960 da geliştirilen Simula 67 nin hiyerarşik tip (*hierarchical types*) tanımlama özelliği
 - ➡ Inheritance (Kalıtım)
 - ➡ Polymorphism (Çok biçimlilik)

No Silver Bullet: Hopes for the SILVER !

- (1) Ada programming language
- (2) Object-oriented programming
- (3) **Artificial Intelligence**

- ➡ **Yapay zeka** insan zekasıyla çözülebilen problemlerin çözümünde bilgisayarların kullanılmasına imkan sağlamıştır. (*utilizing computer to solve problems which are only solved by human intelligence at earlier time*)
- ➡ Uzmanlar tarafından tanımlanan **sezgisel** ve **kural-tabanlı** programlamayı kullanır (*use of human expert defined heuristics or rule-based programming*)

No Silver Bullet: Hopes for the SILVER !

- (1) Ada programming language
- (2) Object-oriented programming
- (3) Artificial Intelligence
- (4) **Expert System**

- ➔ Uzman sistemler yapay zeka tabanlı olup önceden tanımlanan kurallara göre sisteme girilen bilgilere göre bir çözüm sunmaktadır (*generalized inference engine based on predefined rules, which takes input and sends feedback*)
- ➔ Its power come from "*ever-richer knowledge bases*" not from "*ever fancier inference mechanism*"

No Silver Bullet: Hopes for the SILVER !

- (1) Ada programming language
- (2) Object-oriented programming
- (3) Artificial Intelligence
- (4) Expert System

(5) Automatic Programming

- ❖ Bir programın belirlenen ihtiyaçlardan oluşturulmasıdır (*generation of a program to solve a problem from its specifications*)
- ❖ Euphemism for programming with high level languages (**David Parnas**)
- ❖ Sıralama ve diferansiyel denklemler üzerinde kullanılmıştır (*it was used on sorting and integrating differential equations*)

No Silver Bullet: Hopes for the SILVER !

- (1) Ada programming language
- (2) Object-oriented programming
- (3) Artificial Intelligence
- (4) Expert System
- (5) Automatic Programming
- (6) **Graphical Programming**

- ➡ Flowchart is a very poor abstraction of software structure because it is drawn after the design.
- ➡ Hardware technology needs improvement for graphical enhancements of software. Software is very difficult to visualize.

No Silver Bullet: Hopes for the SILVER !

- (1) Ada programming language
- (2) Object-oriented programming
- (3) Artificial Intelligence
- (4) Expert System
- (5) Automatic Programming
- (6) Graphical Programming
- (7) Program Verification**

- ➡ Program doğrulama geliştirilen programın hedeflenen gereksinimleri gerçekleştirip gerçekleştirmediğinin ortaya çıkarılma sürecidir (*process of to make sure that "the program meets its specifications"*)
- ➡ Test yükünü azaltır lakin hataları ortadan kaldırmaz (*it reduces testing load but can't eliminate errors*)

*Frederic P. Brooks,
No Silver Bullet:
Essence and Accident of Software Engineering , 1987*