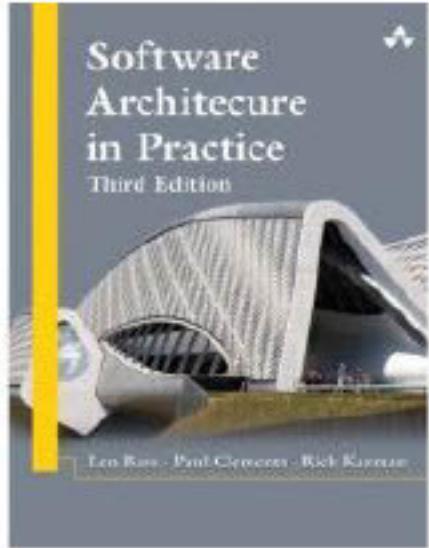


Chapter 1:

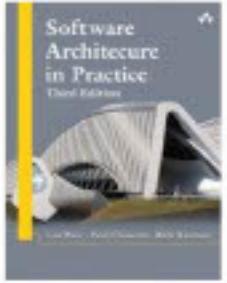
What is Software Architecture?

Good judgment is usually the result of experience. And experience is frequently the result of bad judgment. But to learn from the experience of others requires those who have the experience to share the knowledge with those who follow.

— Barry LePatner

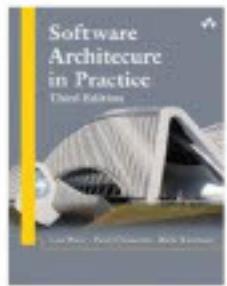


*Doğu hükm, genellikle tecrübebenin sonucudur.
Ve Tecrübe çoğu zaman kötü kararların sonucudur.
Fakat başkalarının tecrübelerinden öğrenmek,
tecrübeli olanların bilgiyi takip edenlerle
paylaşmasını gerektirir.*
-Barry LePatner.



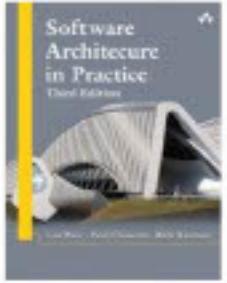
Chapter Outline

- What Software Architecture Is and
 - What It Isn't
- Architectural Structures and Views
- Architectural Patterns
- What Makes a “Good” Architecture?
- Summary



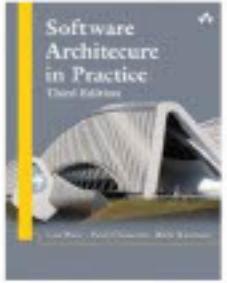
Chapter Outline

- Yazılım Mimarisi Nedir ve Ne Değildir?
- Mimarî Yapılar ve Görünümler
- Mimarî Şablonlar
- Mimariyi “İyi” Yapan Nedir?
- Özeti



What is Software Architecture?

*The software architecture of a system is
the set of structures
needed to reason about the system,
which
comprise software elements,
relations among them,
and
properties of both.*



What is Software Architecture?

software elements

relations among them

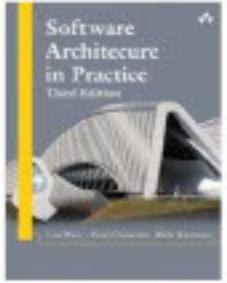
properties of both

Comprise a system

needed to reason about the system

the set of structures

The software architecture of a system

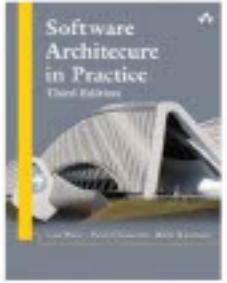


What is Software Architecture?

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

*yazılım öğeleri,
bunların aralarındaki ilişkiler ve
her ikisinin özelliklerince
oluşturulan bir sistem
hakkında mantık yürütütmek için
gerekli yapılar kümesi :*

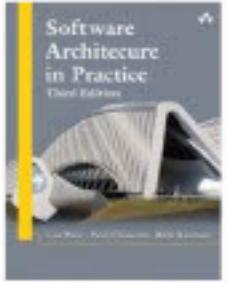
Bir sistemin yazılım mimarisi



What is Software Architecture?

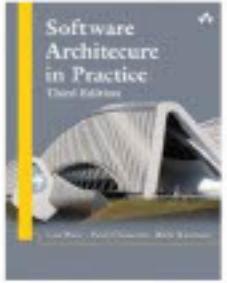
The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

*Bir sistemin yazılım mimarisi,
sistem hakkında
mantık yürütmek için gereklı olan ve
yazılım öğelerini,
onların aralarındaki ilişkileri ve
her ikisinin özelliklerini içeren
yapılar kümesidir.*



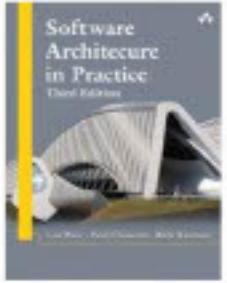
Definition

- This definition stands in contrast to other definitions that talk about the system’s “early” or “major” design decisions.
 - Many architectural decisions are made early, but not all are.
 - Many decisions are made early that are not architectural.
 - It’s hard to look at a decision and tell whether or not it’s “major.” Sometimes only time will tell.
- Structures, on the other hand, are fairly easy to identify in software, and they form a powerful tool for system design.



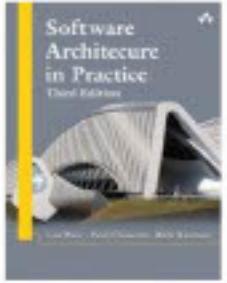
Definition

- Bu tanım, sistemin "erken" veya "büyük" tasarım kararlarından bahsededen diğer tanımlarla çelişir.
- Pek çok mimari karar erken alınır, ancak hepsi değil. (özellikle Çevik veya Spiral geliştirme projelerinde)
- Erken alınan kararların çoğu mimarı değildir.
- Bir karara bakıp “**esas**” olup olmadığını söylemek zordur. Bazen bunu sadece zaman gösterir.
- Öte yandan, **yapıların** yazılımda tanımlanması oldukça kolaydır ve sistem tasarıımı için güçlü bir araç oluştururlar.



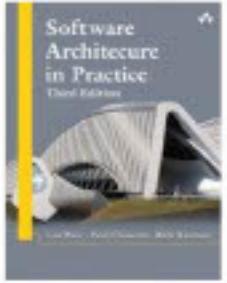
Architecture Is a Set of Software Structures

- A structure is a set of elements held together by a relation.
 - This is the first and most obvious implication of our definition.
- Software systems are composed of many structures, and no single structure holds claim to being the architecture.
- There are three important categories of architectural structures.
 1. Module
 2. Component and Connector
 3. Allocation



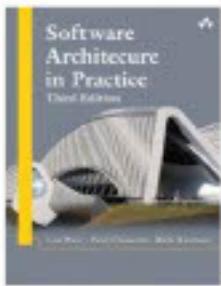
Architecture Is a Set of Software Structures

- Yapı, bir ilişki tarafından bir arada tutulan unsurlar kümesidir.
- Yazılım sistemleri birçok yapıdan oluşur ve hiçbir yapı tek başına mimari olma iddiasında değildir.
- Mimari yapıların üç önemli kategorisi vardır.
 - Modül
 - Bileşen ve Bağlayıcı (bağlayan)
 - Tahsis



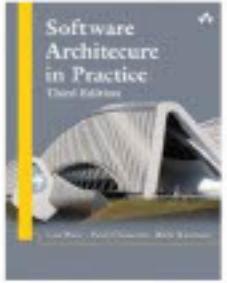
Module Structures

- Some structures partition systems into implementation units, which we call modules.
- Modules are assigned specific computational responsibilities, and are the basis of work assignments for programming teams.
- In large projects, these elements (modules) are subdivided for assignment to sub-teams.



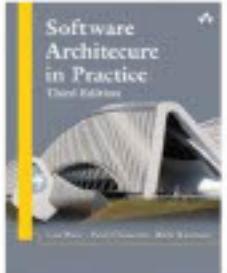
Module Structures

- In large projects, these elements (modules) are subdivided for assignment to sub-teams.
- For example, the database for a large enterprise resource planning (ERP) implementation might be so complex that its implementation is split into many parts. The structure that captures that decomposition is a kind of module structure, the module decomposition structure in fact.
- Another kind of module structure emerges as an output of object-oriented analysis and design—class diagrams.
- If you aggregate your modules into layers, you've created another (and very useful) module structure.
- Module structures are static structures, in that they focus on the way the system's functionality is divided up and assigned to implementation teams.



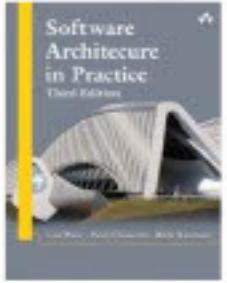
Module Structures

- Bazı yapılar sistemleri '[hayata geçirme birimleri](#)'ne ayırır ki biz ona [modül](#) diyoruz.
- Modüllere belirli [hesaplamalı](#) (computational) sorumluluklar atanır ve programlama ekipleri için [iş atamalarının/ödevlerinin](#) temelini oluşturur.
 - (Team A works on the database, Team B works on the business rules, Team C works on the user interface, etc.).
- Büyük projelerde, bu öğeler (modüller) alt ekiplere atanmak üzere [alt bölümlere](#) ayrıılır.



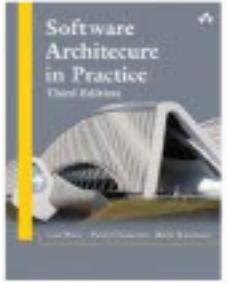
Module Structures

- Örneğin, büyük bir kurumsal kaynak planlama (ERP) uygulaması için veritabanı o kadar karmaşık olabilir ki, uygulama birçok parçağa bölünür. Bu ayrışmayı yakalayan yapı bir çeşit modül yapısıdır, aslında **modül ayrıştırma yapısıdır (module decomposition structure)**.
- Başka bir tür modül yapısı, nesne esaslı analiz ve tasarım sınıfı diyagramlarının bir çıktısı olarak ortaya çıkar.
 - class diagrams
- Modüllerinizi katmanlar halinde bir araya getirirseniz, başka (ve çok kullanışlı) bir modül yapısı oluşturmuş olursunuz.
- **Modül yapıları statik yapılardır**; sistem işlevlerinin bölünüp uygulama ekiplerine atanmasına odaklanır.



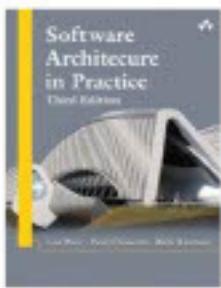
Component-and-connector Structures

- Other structures are dynamic, focus on the way the elements interact with each other at runtime to carry out the system's functions.
- We call runtime structures *component-and-connector (C&C) structures*.
- In our use, a component is always a runtime entity.
 - Suppose the system is to be built as a set of services.
 - The services, the infrastructure they interact with, and the synchronization and interaction relations among them form another kind of structure often used to describe a system.
 - These services are made up of (compiled from) the programs in the various implementation units – modules.



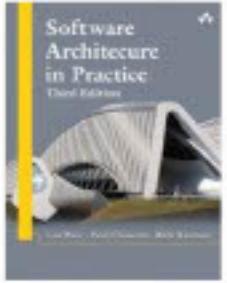
Component-and-connector Structures

- Diğer yapılar dinamiktir, sistemin işlevlerini yerine getirmek için **çalışma esnasında** öğelerin hangi yolla birbirleriyle etkileşimde bulunduğuna odaklanır.
 - Sistemin bir dizi hizmet olarak kurulacağını varsayılm.
 - Hizmetler, etkileşimde bulundukları altyapı, ve aralarındaki senkronizasyon ve etkileşim ilişkileri, genellikle bir sistemi tanımlamak için kullanılan başka tür bir yapıyı şekillendirir.
 - Bu hizmetler, çeşitli uygulama birimlerindeki - modüllerdeki **programlardan** (derlenen) oluşur.
- **Runtime** yapılarına bileşen ve bağlayıcı (**C&C**) yapıları diyoruz.
- Bizim kullanımımızda, bir bileşen her zaman runtime bir varlıktır.



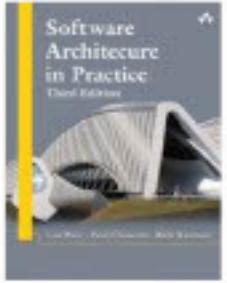
Allocation Structures

- A third kind of structure (allocation structures) describes the **mapping** from software structures **to** the system's organizational, developmental, installation, and execution environments.
 - organizational
 - developmental
 - installation
 - execution
- For example
 - Modules are assigned to teams to develop, and assigned to places in a file structure for implementation, integration, and testing.
 - Components are deployed onto hardware to execute.



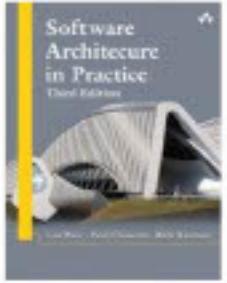
Allocation Structures

- Bir üçüncü yapı, Tahsis yapıları, yazılım yapılarını sistem ortamlarıyla örtüştürmeyi tanımlar.
 - Örgüt
 - Geliştirme
 - Kurulum
 - İcra
- Örneğin
 - Modüller, geliştirilmesi için ekiplere atanır ve geliştirme, entegrasyon ve test için dosya yapısında bir yere atanır.
 - Bileşenler, yürütülmek üzere donanıma yerleştirilir.



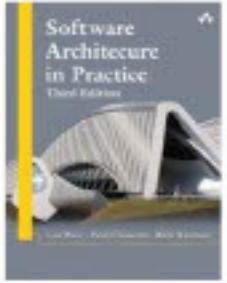
Which Structures are Architectural?

- A structure is architectural if it supports reasoning about the system and the system's properties.
- The **reasoning** should be about an **attribute** of the system that is **important to some stakeholder**.
- These include
 - **functionality** achieved by the system
 - the **availability** of the system in the face of faults
 - the difficulty of **making specific changes** to the system
 - the **responsiveness** of the system to user requests,
 - many others.



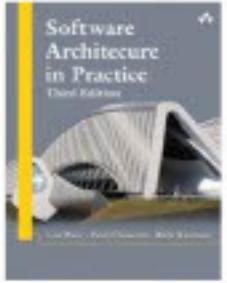
Which Structures are Architectural?

- Bir yapı, **sistem ve sistemin özellikleri** hakkında **akıl yürütme** destekliyorsa mimaridir.
- Bu akıl yürütme(muhakeme), sistemin bazı **"stakeholder"**larca önemsenen bir niteliği hakkında olmalıdır.
- Bunlar arasında
 - sistem tarafından sağlanan **İşlevsellik**
 - Arıza halinde sistemin **erişilebilirliği/kullanılabilirliği**
 - sistemde belirli değişiklikler yapmanın zorluğu
 - sistemin kullanıcı taleplerine **cevap verme kabiliyeti**,
 - Ve sair...
- Kalite Unsurları ?



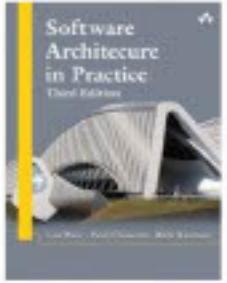
Architecture is an Abstraction

- An architecture comprises software elements and how the elements relate to each other.
 - An architecture specifically omits certain information about elements that is not useful for reasoning about the system.
 - It omits information that has no ramifications outside of a single element.
 - An architecture selects certain details and suppresses others.
 - Private details of elements—details having to do solely with internal implementation—are not architectural.
- The architectural abstraction lets us look at the system in terms of its elements, how they are arranged, how they interact, how they are composed, what their properties are that support our system reasoning, and so forth.
- This abstraction is essential to taming the complexity of an architecture.
- We simply cannot, and do not want to, deal with all of the complexity all of the time.



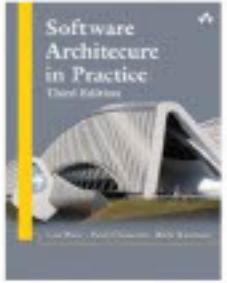
Architecture is an Abstraction

- Bir mimari, yazılım öğelerini ve bunların birbiriyle ilişki şeklini içerir.
 - Bir mimari, **sistem hakkında mantık yürütmek için bir faydası olmayan unsurlar** hakkındaki belirli bilgileri bilhassa gözardı eder.
 - Mesela, içince “z” harfi olan kaynak kodu satırlarının harf sayısına göre kısadan uzuna sıralanması bir yapı oluşturur ama bu ne ilginçtir ne de mimarî bir yapıdır.
 - Bir mimari bir ögenin dışarısında hiçbir sonucu olmayan bilgileri atlar.
 - Bir mimari, belirli ayrıntıları seçer ve diğerlerini bastırır.
 - Öğelerin özel ayrıntıları - **yalnızca iç uygulamayla ilgili ayrıntılar** - mimari değildir.



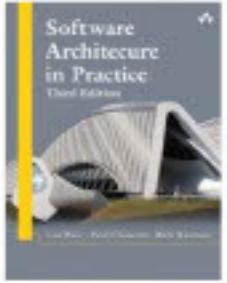
Architecture is an Abstraction

- Mimari soyutlama, sisteme,
 - öğeleri,
 - nasıl düzenlendikleri,
 - nasıl etkileşime girdikleri,
 - nasıl biraraya getirildikleri,
 - sistem mantığımızı destekleyen özelliklerinin neler olduğu vb. açısından
- bakiyi sağlar.
- Bu soyutlama, bir mimarinin karmaşıklığını uysallaştırmak için gereklidir.
- Biz karmaşıklığın hepsiyle ve sürekli uğraşamayız, uğraşmak da istemeyiz.



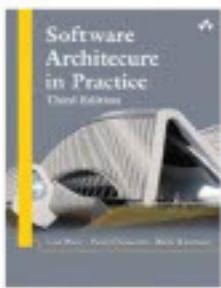
Every System has a Software Architecture

- Every system comprises elements and relations among them to support some type of reasoning.
- But the architecture may not be known to anyone.
 - Perhaps all of the people who designed the system are long gone
 - Perhaps the documentation has vanished (or was never produced)
 - Perhaps the source code has been lost (or was never delivered)
- An architecture can exist independently of its description or specification.
- Documentation is critical.



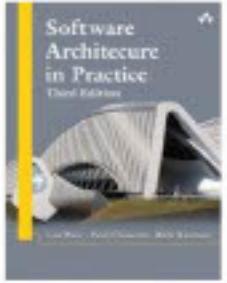
Every System has a Software Architecture

- Her sistem, bir tür muhakemeyi desteklemek için öğeleri ve aralarındaki ilişkileri kapsar.
- Ancak mimari kimse tarafından bilinmeyebilir.
 - Belki de sistemi tasarlayan tüm insanlar çoktan gitmiştir.
 - Belki belgeler kaybolmuştur (veya hiç hazırlanmamış)
 - Belki kaynak kodu kaybolmuştur (veya hiç teslim edilmemiştir)
- Bir mimari, tanımından veya şartnamesinden bağımsız olarak var olabilir.
- Belgelendirme hayatıdır.



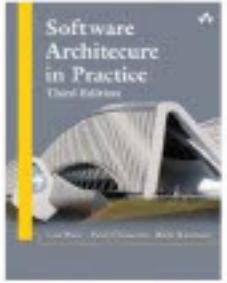
Architecture Includes Behavior

- The behavior of each element is part of the architecture insofar as that behavior can be used to reason about the system.
- This behavior embodies how elements interact with each other, which is clearly part of the definition of architecture.
- Box-and-line drawings that are passed off as architectures are not architectures at all.
 - When looking at the names of a reader may well imagine the functionality and behavior of the corresponding elements.
 - But it relies on information that is not present – and could be wrong!
- This does not mean that the exact behavior and performance of every element must be documented in all circumstances.
 - Some aspects of behavior are fine-grained and below the architect's level of concern.
- To the extent that an element's behavior influences another element or influences the acceptability of the system as a whole, this behavior must be considered, and should be documented, as part of the software architecture.



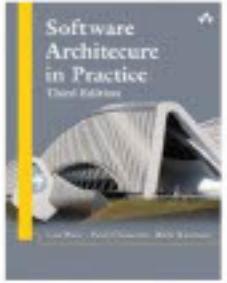
Architecture Includes Behavior

- Her bir ögenin davranışı,
 - bu davranış sistem hakkında mantık yürütmek için kullanılabilirliği ölçüde mimarının bir parçasıdır.
- **Bu davranış, unsurların birbirleriyle nasıl etkileşime girdiğini somutlaştırır ve bu, açıkça mimari tanımının bir parçasıdır.**
- Mimari olarak aktarılan kutu ve çizgi çizimler, mimari değildir.
 - Bir okuyucunun isimlerine bakarken, karşılık gelen öğelerin işlevsellliğini ve davranışını iyi bir şekilde hayal edebilirsiniz.
 - Ancak mevcut olmayan bilgilere dayanır ve yanlış olabilir!
- Bu, her unsurun tam davranışının ve performansının her koşulda belgelenmesi gerektiği anlamına gelmez.
 - Davranışın bazı yönleri ince ayrıntılara sahiptir ve mimarın ilgi düzeyinin altındadır.



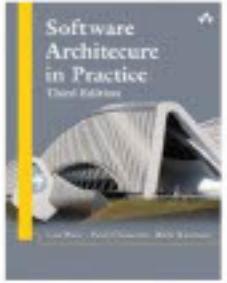
Architecture Includes Behavior

- Bir unsurun davranışı
 - başka bir unsuru etkilediği ölçüde
 - veya
 - bir bütün olarak sistemin kabul edilebilirliğini etkilediği ölçüde,
- yazılım mimarisinin bir parçası olarak değerlendirilmeli ve belgelenmelidir.



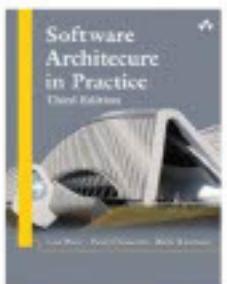
Physiological Structures

- The neurologist, the orthopedist, the hematologist, and the dermatologist all have different views of the structure of a human body.
- Ophthalmologists, cardiologists, and podiatrists concentrate on specific subsystems.
- The kinesiologist and psychiatrist are concerned with different aspects of the entire arrangement's behavior.
- Although these views are pictured differently and have different properties, all are inherently related, interconnected.
- Together they describe the architecture of the human body.
- So it is with software!

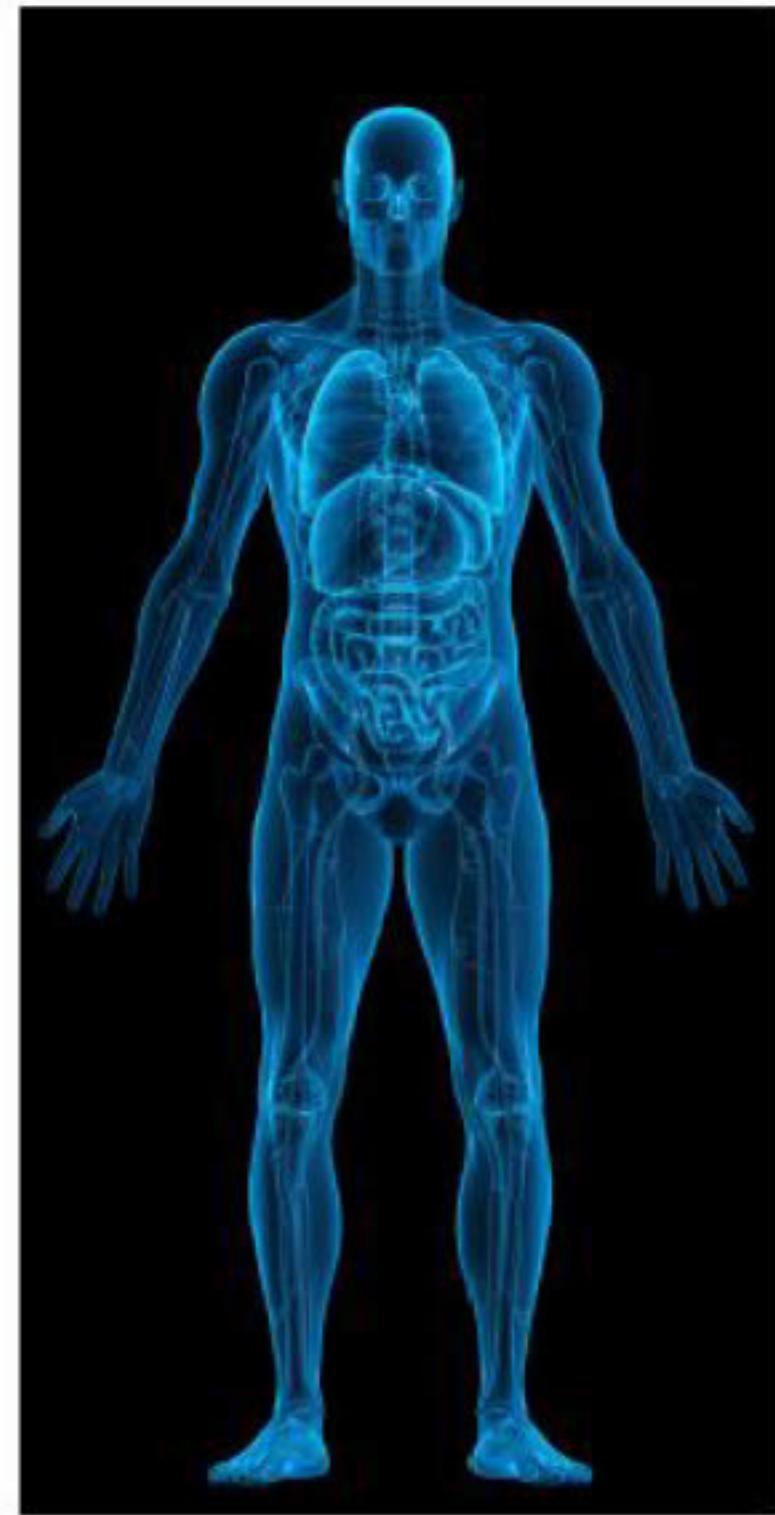


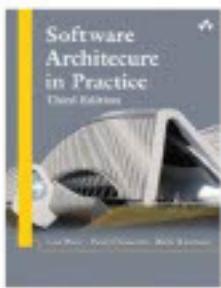
Physiological Structures

- Nörolog, ortopedist, hematolog ve dermatolog, insan vücudunun yapısı hakkında farklı görüşlere sahiptir.
- Göz doktorları, kardiyologlar ve ayak hastalıkları uzmanları belirli alt sistemler üzerinde yoğunlaşırlar.
- Kinesiyolog ve psikiyatrist, tüm düzenlemenin davranışının farklı yönleriyle ilgilenir.
- Bu görüşler farklı şekilde resmedilmiş ve farklı özelliklere sahip olsa da, **hepsi doğası gereği birbiriyle ilişkilidir ve birbiriyle bağlantılıdır.**
- Birlikte insan vücudunun mimarisini anlatıyorlar.
- Yani **yazılım ile!**



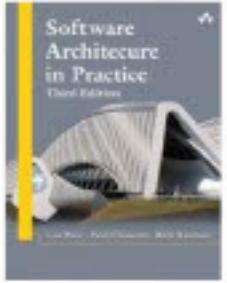
Physiological Structures





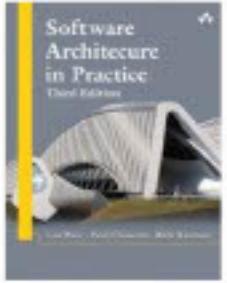
Structures and Views

- A *view* is a representation of a coherent set of architectural elements, as written by and read by system stakeholders.
 - A view consists of a representation of a set of elements and the relations among them.
- A *structure* is the set of elements itself, as they exist in software or hardware.
- In short, a view is a representation of a structure.
 - For example, a module *structure* is the set of the system's modules and their organization.
 - A module *view* is the representation of that structure, documented according to a template in a chosen notation, and used by some system stakeholders.
- Architects design structures. They document views of those structures.



Structures and Views

- Bir “view”,
 - sistem **stakeholder’ları** tarafından yazılan ve okunan şekliyle,
 - coherent bir mimari unsurlar kümesinin temsilidir.
- Bir “view”, bir dizi ögenin ve aralarındaki ilişkilerin temsilini ihtiva eder.
- Bir yapı yazılımda ve donanımda var olduğu haliyle, bir unsurlar kümesidir.



Structures and Views

- Kisaca bir “view” bir yapının temsiliidir.
 - Mesela bir modül yapısı sistem modüllerinin ve onların düzenlerinin kümesidir.
 - Bir modül “view” seçilen notasyondaki şablon'a göre dökümante edilen ve bazı sistem stakeholderları tarafından kullanılan bu yapının temsiliidir.
- Mimarlar yapıyı tasarlar. Bu yapının “view”larını dökümante ederler.