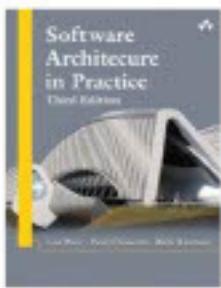


2

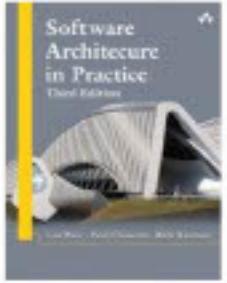
Software Architecture and Design

© Len Bass, Paul Clements, Rick Kazman,
distributed under Creative Commons
Attribution License



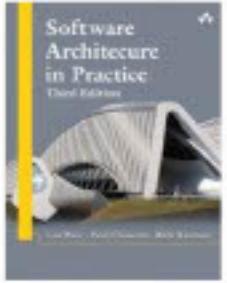
Module Structures

- Module structures embody decisions as to how the system is to be structured as a set of code or data units that have to be constructed or procured.
- In any module structure, the elements are modules of some kind (perhaps classes, or layers, or merely divisions of functionality, all of which are units of implementation).
- Modules are assigned areas of functional responsibility; there is less emphasis in these structures on how the software manifests at runtime.
- Module structures allow us to answer questions such as these:
 - What is the primary functional responsibility assigned to each module?
 - What other software elements is a module allowed to use?
 - What other software does it actually use and depend on?
 - What modules are related to other modules by **generalization or specialization (i.e., inheritance)** relationships?



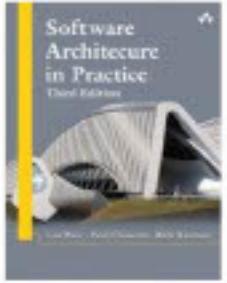
Module Structures

- Modül yapıları, sistemin, inşa edilmesi veya tedarik edilmesi gereken bir dizi kod veya veri birimleri olarak nasıl yapılandırılacağına ilişkin kararları içerir.
- Herhangi bir modül yapısında, öğeler bir tür modüllerdir (**hepsi uygulama (implementation) birimi** olan, belki, sınıflar veya tabakalar veya yalnızca işlevsellik bölümleri).
- **Modüller, tayin edilmiş işlevsel sorumluluk alanlarıdır;** Bu yapıarda, yazılımın koşma esnasında nasıl tezahür ettiğine dair vurgu azdır.



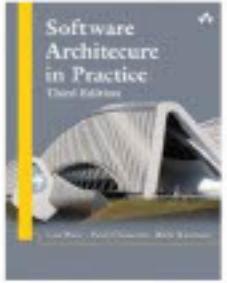
Module Structures

- Modül yapıları, aşağıdakilere benzer sorulara cevap vermemize imkân verir:
 - Her bir modüle atanınan birincil işlevsel sorumluluk nedir?
 - Bir modülün başka **hangi yazılım öğelerini kullanmasına izin verilir?**
 - Başka **hangi yazılımı** **gerçekte kullanır ve bunlara bağlıdır?**
 - Hangi modüller diğer modüllerle **genelleştirme veya özelleştirme (yani kalıtım)** yoluyla irtibatlandırılır?
(generalization or specialization (i.e., inheritance))



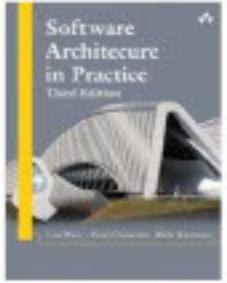
Component-and-connector Structures

- Component-and-connector structures embody decisions as to how the system is to be structured as a set of elements that have **runtime behavior (components)** and **interactions (connectors)**.
- Bileşen ve bağlayan yapıları, sistemin
 - Koşu davranışsı (bileşenler) ve etkileşimlerine (bağlayanlar) sahip bir öğeler kümesi
 - olaraknasıl yapılandırılacağına ilişkin kararları içerir.



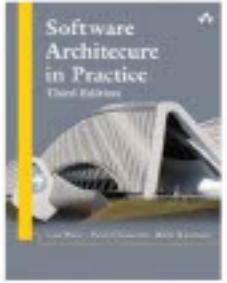
Component-and-connector Structures

- Elements are runtime components such as services, peers, clients, servers, filters, or many other types of runtime element
- Connectors are the communication vehicles among components, such as call-return, process synchronization operators, pipes, or others.
- Öğeler,
 - hizmetler, eşler(denkler), istemciler, sunucular, filtreler veya diğer birçok **koşu** ögesi türü gibi
 - koşu bileşenleridir.
- Bağlayanlar,
 - çağrı dönüşü, süreç senkronizasyon operatöleri, kanallar ve sair gibi
 - bileşenler arasındaki iletişim araçlarıdır.



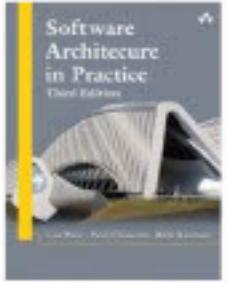
Component-and-connector Structures

- Component-and-connector views help us answer questions such as these:
 - What are the major executing components and how do they interact at runtime?
 - What are the major shared data stores?
 - Which parts of the system are replicated?
 - How does data progress through the system?
 - What parts of the system can run in parallel?
 - Can the system's structure change as it executes and, if so, how?
- Bileşen ve bağlayan görünümleri, aşağıdakilere benzer soruları cevaplamamıza yardımcı olur:
 - **Başlıca yürütme bileşenleri nelerdir ve koşulurken nasıl etkileşirler?**
 - **Başlıca paylaşılan veri depoları nelerdir?**
 - Sistemin hangi bölümleri kopyalanır (ikilenir)?
 - Veriler sistemde nasıl ilerler?
 - **Sistemin hangi bölümleri paralel olarak çalışabilir?**
 - Sistemin yapısı, yürütken değişebilir mi; değişirse nasıl değişir?



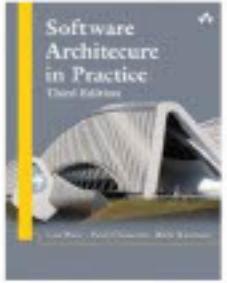
Component-and-connector Structures

- Component-and-connector views are crucially important for asking questions about the system's runtime properties such as performance, security, availability, and more.
- Bileşen ve bağlayan görünümleri, sistemin
 - koşu performansı, güvenlik, erişilebilirlik ve dahası hakkında
 - soru sormak için hayatı önemdedir.



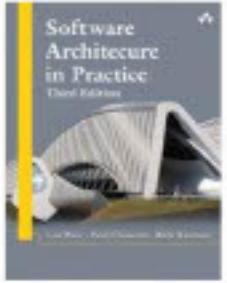
Allocation structures

- Allocation structures show the relationship between the software elements and elements in one or more external environments in which the software is created and executed.
- Tahsis yapıları,
- **yazılım öğeleri ile**
 - yazılımın oluşturulduğu ve yürütüldüğü
 - bir veya daha fazla harici ortamdaki
- **öğeler**
- **arasındaki ilişkiyi gösterir.**



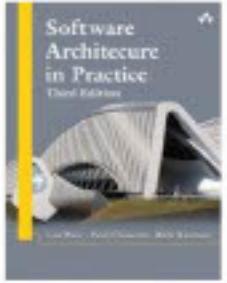
Allocation structures

- Allocation views help us answer questions such as these:
 - What processor does each software element execute on?
 - In what directories or files is each element stored during development, testing, and system building?
 - What is the assignment of each software element to development teams?
- Tahsis görünümleri, aşağıdakilere benzer soruları cevaplamamıza yardımcı olur:
 - Her bir yazılım ögesi **hangi işlemci** üzerinde çalışır?
 - Geliştirme, test etme ve sistem oluşturma sırasında her öge **hangi dizinlerde veya dosyalarda depolanır?**
 - **Geliştirme ekiplerine verilen** her bir yazılım ögesi ödevi nedir?



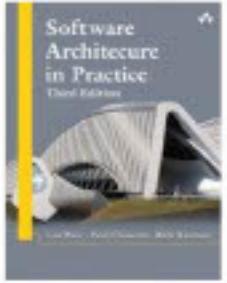
Structures Provide Insight

- Structures play such an important role in our perspective on software architecture because of the analytical and engineering power they hold.
- Yapılar,
 - sahip oldukları analitik ve mühendislik gücü nedeniyle
 - **yazılım mimarisine bakış açımızda çok önemli bir rol oynamaktadır.**



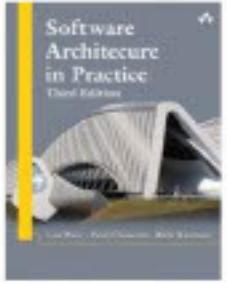
Structures Provide Insight

- Each structure provides a perspective for reasoning about some of the relevant quality attributes.
- Her yapı,
 - ilgili kalite özelliklerinden bazıları hakkında
 - mantık yürütmek için bir perspektif sağlar.



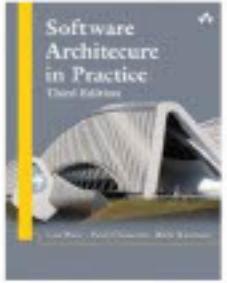
Structures Provide Insight

- For example:
 - The **module structure**, which embodies what modules use what other modules, is strongly tied to the **ease with** which a system can be **extended or contracted**.
- Örneğin:
 - Hangi modüllerin diğer hangi modülleri kullandığını içeren **modül yapısı**,
 - bir sistemin genişletilebilmesi veya daraltılmasının **kolaylığına**
 - sıkı sıkıya **bağlıdır**.



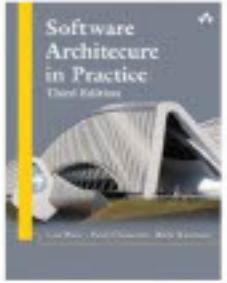
Structures Provide Insight

- For example:
 - The **concurrency structure**, which embodies parallelism within the system, is strongly tied to the ease with which a system can be made **free of deadlock and performance bottlenecks**.
- Örneğin:
 - Paralelliği sistem bünyesinde barındıran **eşanlılık yapısı**,
 - bir sistemin **kilitlenme ve performans darboğazlarından**
 - kurtarılma kolaylığına sıkı sıkıya bağlıdır.



Structures Provide Insight

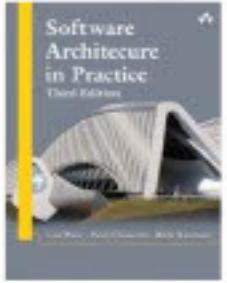
- For example:
 - The deployment structure is strongly tied to the achievement of performance, availability, and security goals.
 - And so forth
- Örneğin:
 - İntikal yapısı,
 - performans, erişilebilirlik ve güvenlik hedeflerine ulaşılmasına sıkı sıkıya bağlıdır.
- Ve benzeri..



Some Useful Module Structures

Decomposition structure

- The units are modules that are related to each other by the *is-a-submodule-of* relation.
- It shows how modules are decomposed into smaller modules **recursively** until the modules are small enough to be easily understood.
- **Ayrıştırma yapısı**
 - Birimleri, “**bir-alt-modülüdür**” ilişkisi ile birbirleriyle irtibatlı modüllerdir.
 - Modüllerin,
 - kolayca anlaşılabilenek kadar daha küçük modüllere
 - **tekrarlı** olarak nasıl ayrıstırıldığını gösterir.



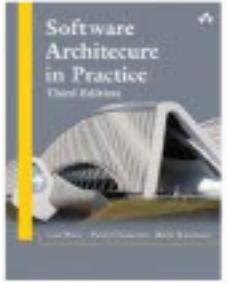
Some Useful Module Structures

- *Decomposition structure (cntd)*

- Modules often have products (such as interface specifications, code, test plans, etc.) associated with them.
- The decomposition structure determines, to a large degree, the system's modifiability, by assuring that likely changes are localized.

- Ayrıştırma yapısı (dvm)

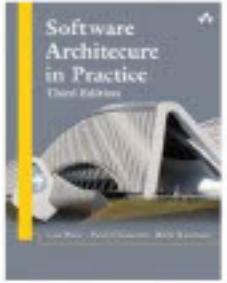
- Modüller genellikle kendileriyle eşlenik produktlere (arayüz şartnameleri, kod, test planları vb.) sahiptir.
- Ayrıştırma yapısı, büyük ölçüde,
 - olası değişikliklerin **yerelleştirilmesini** temin ederek
 - sistemin değiştirilebilirliğini belirler.



Some Useful Module Structures

- *Decomposition structure*

- This structure is often used as the basis for the development project's organization, including the structure of the documentation, and the project's integration and test plans.
- The units in this structure tend to have names that are organization-specific such as "segment" or "subsystem."
- **Ayrıştırma yapısı (dvm)**
 - Bu yapı, genellikle, geliştirme projesinin,
 - dokümantasyon yapısı, proje entegrasyonu ve test planlarını ihtiva eden
 - organizasyonu için temel olarak kullanılır.
 - Bu yapıdaki birimler, "segment" veya "alt sistem" gibi
 - kuruluşu has adlara sahip olma eğilimindedir.



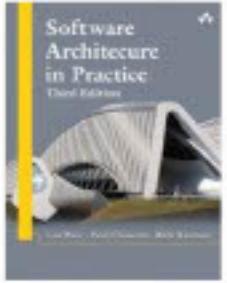
Some Useful Module Structures

- *Uses structure*

- The units here are also modules, perhaps classes.
- The units are related by the *uses* relation, a specialized form of dependency.

“Kullanır” yapıları

- Buradaki birimler aynı zamanda **modüller** belki de **sınıflardır**.
- Bu birimler **bağımlılığın** özel bir biçimini olan **“kullanır”** ilişkisi ile irtibatlıdır.



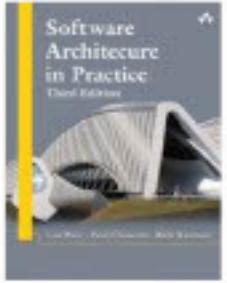
Some Useful Module Structures

- *Uses structure*

- A unit of software *uses* another if the correctness of the first requires the presence of a correctly functioning version (as opposed to a stub) of the second.

- “**kullanır**” yapısı

- İki yazılımdan biriminden ,
 - birinin doğruluğu,
 - diğerinin çalışan sürümünün (bir saplamanın aksine) doğruluğunu gerektiriyorsa,
 - biri diğerini kullanıyor demektir.



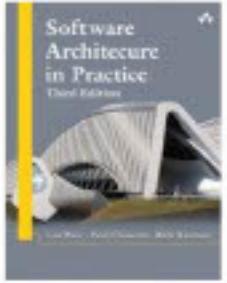
Some Useful Module Structures

- *Uses structure*

- The uses structure is used to engineer systems that can be extended to add functionality, or from which useful functional subsets can be extracted.
- The ability to easily create a subset of a system allows for incremental development.

- “**Kullanır**” yapısı

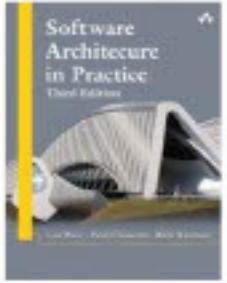
- Bu yapı,
 - işlevsellik eklenebilecek veya
 - faydalı işlevsel alt kümeler çıkarsanabilecek sistemler yapabilmek için kullanılır.
- Bir sistemin bir alt kümesini kolayca oluşturma yeteneği,
 - artırılmış geliştirmeye izin verir.



Some Useful Module Structures

Layer structure

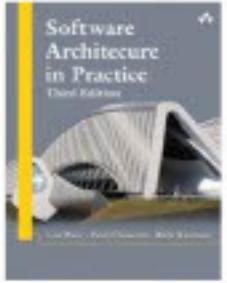
- The modules in this structure are called *layers*.
- A layer is an abstract “virtual machine” that provides a cohesive set of services through a managed interface.
- **Tabaka yapısı**
 - Bu yapıdaki modüllere tabakalar adı verilir.
 - Bir tabaka, yönetilen bir arabirim vasıtasıyla sıkı ve türdeş bir hizmet kümlesi sağlayan soyut bir "sanal makinedir".



Some Useful Module Structures

Layer structure

- Layers are *allowed to use* other layers in a strictly managed fashion.
 - In strictly layered systems, a layer is only allowed to use a single other layer.
- **Tabaka yapısı**
- Tabakaların
 - diğer tabakaları
 - sıkı bir yönetim tarzıyla
 - kullanmasına izin verilir.
 - Katı tabakalı sistemlerde,
 - bir tabakanın
 - yalnızca tek bir başka tabaka kullanmasına izin verilir.



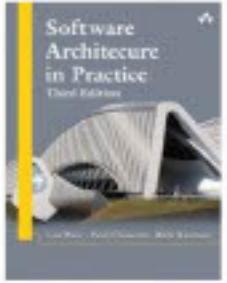
Some Useful Module Structures

- *Layer structure*

- This structure imbues a system with portability, the ability to change the underlying computing platform.

- **Tabaka yapısı**

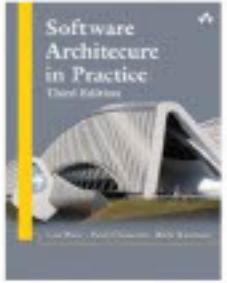
- Bu yapı bir sisteme
 - taşınabilirlik
 - yani
 - dayandığı bilgisayar zeminini değiştirme
 - yeteneği verir.



Some Useful Module Structures

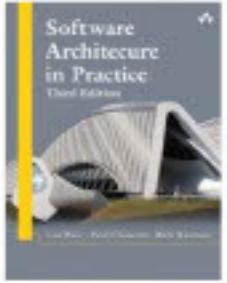
Class (or generalization) structure

- The module units in this structure are called *classes*.
 - The relation is *inherits from* or *is an instance of*.
-
- **Sınıf** (veya genelleştirme) yapısı
 - Bu yapıdaki modül birimlerine sınıflar denir.
 - İlişki,
 - “-den mirastır” veya “-in bir oluşudur”
 - dur.



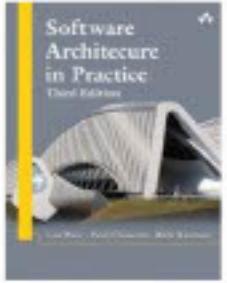
Some Useful Module Structures

- *Class (or generalization) structure*
- This view supports reasoning about collections of similar behavior or capability
 - e.g., the classes that other classes inherit from and parameterized differences
- **Sınıf (veya genelleştirme) yapısı**
- Bu yapı
 - benzer davranış veya yetenek hakkında
 - muhakeme yürütmemeyi destekler.
 - Diğer sınıfların miras aldığı ve parametreleştirtiği farklılıklar sınıfı.



Some Useful Module Structures

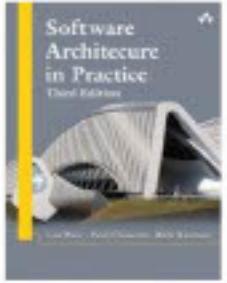
- *Class (or generalization) structure*
 - The class structure allows one to reason about reuse and the incremental addition of functionality.
- **Sınıf** (veya genelleştirme) yapısı
 - Sınıf yapısı,
 - birine,
 - yeniden kullanım ve işlevselliğin artırımı eklenışı üzerine
 - akıl yürütme izni verir.



Some Useful Module Structures

Class (or generalization) structure

- If any documentation exists for a project that has followed an object-oriented analysis and design process, it is typically this structure.
- **Sınıf** (veya genelleştirme) yapısı:
 - Nesneye dönük analiz ve tasarım sürecini takip eden bir projede eğer herhangi bir belge mevcutsa,
 - o, genellikle bu yapıdır.

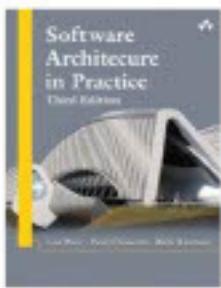


Some Useful Module Structures

- *Data model*
- The data model describes the static information structure in terms of data entities and their relationships.
 - For example, in a banking system, entities will typically include Account, Customer, and Loan.
 - Account has several attributes, such as account number, type (savings or checking), status, and current balance.

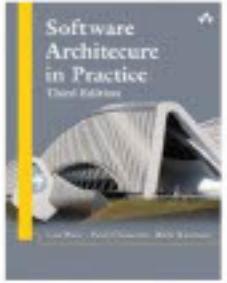
• Veri modeli

- statik bilgi yapısını ,
- veri varlıkları ve ilişkileri açısından tarif eder.
 - Mesela, bir bankacılık sisteminde varlıklar,
 - tipik olarak Hesap, Müşteri ve Krediyi içerir.
 - Hesap, hesap numarası, tür (tasarruf veya çek), durum ve cari bakiye gibi çeşitli niteliklere sahiptir.



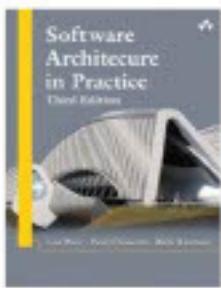
Some Useful C&C Structures

- The relation in all component-and-connector structures is attachment, showing how the components and the connectors are hooked together.
- The connectors can be familiar constructs such as “invokes.”
- Useful C&C structures include:
 - Service structure
 - The units are services that interoperate with each other by service coordination mechanisms such as **SOAP**.
 - The service structure helps to engineer a system composed of components that may have been developed anonymously and independently of each other.
 - Concurrency structure
 - This structure helps determine opportunities for parallelism and the locations where resource contention may occur.
 - **The units are components**
 - **The connectors are their communication mechanisms.**
 - **The components are arranged into logical threads.**



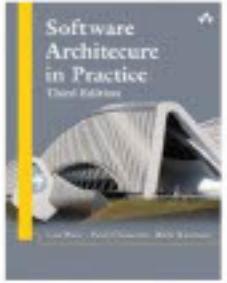
Some Useful C&C Structures

- The relation in all component-and-connector structures is attachment, showing how the components and the connectors are hooked together.
- The connectors can be familiar constructs such as “invokes.”
- Tüm bileşen ve bağlayan yapılarındaki ilişki,
 - bileşen ve bağlayanların birbirlerine nasıl bağlandığını gösteren
 - “ek”tir.
- Bağlayanlar, "**çağıırır**" gibi aşına yapılar olabilir.



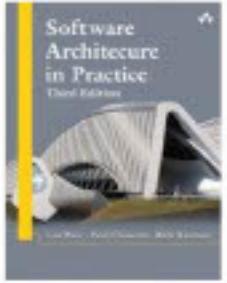
Some Useful C&C Structures

- Useful C&C structures include:
 - Service structure
 - The units are services that **interoperate** with each other by service coordination mechanisms such as SOAP.
 - The service structure helps to engineer a system composed of components that may have been developed anonymously and independently of each other.
 - **Hizmet yapısı**
 - Birimler , SOAP (simple object access protocol) gibi
 - hizmet koordinasyon mekanizmaları sayesinde
 - **birarada** çalışan servislerdir.
 - Hizmet yapısı,
 - anonim ve birbirinden bağımsız olarak geliştirilmiş olabilecek
 - bileşenlerden oluşan
 - bir sistem mühendisliğine yardımcı olur.



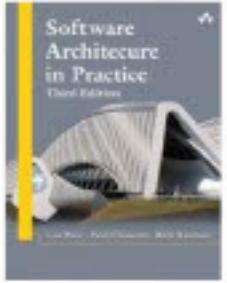
Some Useful C&C Structures

- Useful C&C structures include:
 - Concurrency structure
 - This structure helps determine opportunities for parallelism and the locations where resource contention may occur.
 - The units are components
 - The connectors are their communication mechanisms.
 - The components are arranged into logical **threads**.
 - **Eşanlılık yapısı**
 - Bu yapı,
 - paralellik fırsatlarını ve
 - kaynak çekişmesinin ortaya çıkabileceği yerleri
 - tesbit etmeye yardımcı olur.
 - Birimler bileşenlerdir.
 - Bağlayanlar onların iletişim mekanizmalarıdır.
 - Bileşenler mantıksal **silsiled**e düzenlenmiştir.



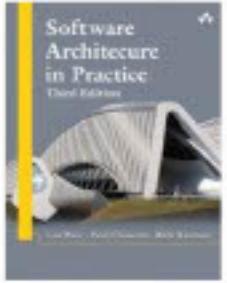
Some Useful Allocation Structures

- *Deployment structure*
 - The deployment structure shows how software is assigned to hardware processing and communication elements.
 - The elements are software elements (usually a process from a C&C view), hardware entities (processors), and communication pathways.
- **İntikal yapısı**
 - İntikal yapısı, yazılımın
 - donanım işleme ve iletişim öğelerine
 - nasıl atandığını gösterir.
 - Öğeler,
 - yazılım öğeleri (genellikle C&C görünümünden bir süreç),
 - donanım varlıkları (işlemciler) ve
 - iletişim yollarıdır.



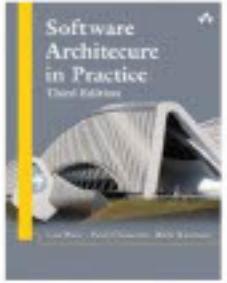
Some Useful Allocation Structures

- *Deployment structure*
 - Relations are allocated-to, showing on which physical units the software elements reside, and migrates-to if the allocation is dynamic.
 - This structure can be used to reason about performance, data integrity, security, and availability.
 - It is of particular interest in distributed and parallel systems.
- **İntikal yapısı**
 - İlişkiler , yazılım öğelerinin bulunduğu fiziki birimleri göstererek
 - “tahsis edilir” ve
 - tahsis dinamik ise “göçertilir” .
 - Bu yapı,
 - performans, veri bütünlüğü, güvenlik ve erişilebilirlik hakkında
 - mantık yürütmek için kullanılabilir.
 - Dağıtık ve paralel sistemlerde özel ilgi alanıdır.



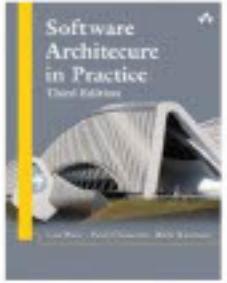
Some Useful Allocation Structures

- *Implementation structure*
 - This structure shows how software elements (usually modules) are mapped to the file structure(s) in the system's development, integration, or configuration control environments.
 - This is critical for the management of development activities and build processes.
- **Hayata geçirme yapısı**
 - Bu yapı,
 - yazılım öğelerinin (genellikle modüller)
 - sistemin geliştirme, birleştirme veya yapılandırma kontrol ortamlarındaki dosya yapılarıyla
 - nasıl örtüştürüldüğünü gösterir.
 - Bu,
 - geliştirme faaliyetlerinin
 - yönetimi ve inşa süreçleri için hayatıdır.



Some Useful Allocation Structures

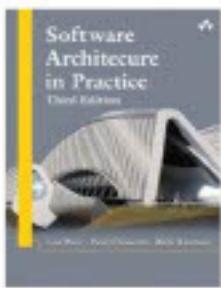
- *Work assignment structure*
 - This structure assigns responsibility for implementing and integrating the modules to the teams who will carry it out.
 - Having a work assignment structure be part of the architecture makes it clear that the decision about who does the work has architectural as well as management implications.
- **İş atama yapısı**
 - Bu yapı,
 - modüllerin hayat geçirme ve birleştirme sorumluluğunu
 - onu yapacak ekiplere yükler.
 - Bir iş atama yapısının mimarinin bir parçası olması,
 - işi kimin yapacağına ilişkin kararın
 - mimari ve idari sonuçları olduğunu açıkça ortaya koymaktadır..



Some Useful Allocation Structures

Work assignment structure

- The architect will know the expertise required on each team.
- This structure will also determine the major communication pathways among the teams: regular teleconferences, wikis, email lists, and so forth.
- **İş atama yapısı.**
- Mimar, her ekip için gerektirdiği uzmanlığı bilecektir.
- Bu yapı aynı zamanda ekipler arasındaki ana iletişim yollarını da belirleyecektir: düzenli telekonferanslar, wiki'ler, e-posta listeleri vb.



Relating Structures to Each Other

- Elements of one structure will be related to elements of other structures, and we need to reason about these relations.
 - A module in a decomposition structure may be manifested as one, part of one, or several components in one of the component-and-connector structures.
 - In general, mappings between structures are many to many.
-
- Bir yapıının unsurları,
 - diğer yapıların unsurlarıyla ilişkili olacaktır ve
 - bu ilişkiler hakkında mantık yürütütmeye ihtiyacımız vardır.
 - Bir ayrıştırma yapısındaki bir modül, bileşen ve bağlayan yapılarından biri, birinin bir parçası veya bir bileşenin birkaç bileşeni ve bağlayani olarak belirebilir.
 - Genel olarak, yapılar arasındaki örtüşmeler çoğu çoktur.