Architektura sw systému

KIV/ASWI 2017/2018

Obsah a cíl této části

- Co to je a jak vypadá architektura
  - statická struktura a dynamické aspekty celého systému
  - pohledy, vzory a dokumentace
- Dát návod jak popsat podstatné části návrhu
  - tak, aby výsledky analýzy šly jednoznačně implementovat v konkrétním programovacím jazyce a provozní platformě
  - tak aby implementaci mohlo provádět více lidí, aby byla efektivní (co do výkonu i tvorby) a výsledek byl snadno udržovatelný

Klíčová rozhodnutí v projektu

- (proč – vize)
- (co – požadavky)

- jak – architektura
  - technologie
  - struktura
  - pravidla, konvence

Architektura systému

The fundamental organization of a system … its components, their relationships, … the environment, and the principles guiding its design and evolution.

- Klíčové složky vnitřní organizace sw
  - jak je systém členěn, co/proč jednotlivé části dělají
  - technologie
  - struktura (hrubé členění)

- Pravidla pro celou implementaci (politiky, konvence)
  - prolínají všemi částmi struktury sw
  - vzory návrhu/implementace
  - mimo-funkční aspekty

Klíčová rozhodnutí na úrovni architektury

Stupně volnosti při tvorbě arch.

- Dáno: kontext
  - okolí systému => vazby, technologie, omezení; důvody
    - viz Enterprise architektura
  - stakeholders => úhly pohledu => aspekty architektury
    - navíc oproti vizi a požadavkům
    - aspekty: logická struktura, procesní pohled, varianty nasazení, datová integrace, bezpečnost, provoz a podpora, provozní infrastruktura, rozhraní, …
- Možnost volby: konstrukce v rámci konkrétní architektury
  - architectural principles – fundamentální přístup pro daný sw
  - struktura (prvky, vazby), konvence
  - **účel**: splnění požadavků + efektivita (sw run-time, náklady na vývoj), jednotnost, srozumitelnost (náklady na údržbu)

Výběr technologie, omezení

- Programovací jazyk, databáze, knihovny
- Použití frameworků a hotových komponent
  - make vs buy decision

- Faktory ovlivňující/omezující výběr technologie
  - smluvní podmínky
  - standardy (i lokální)
    - viz disciplína Enterprise Architecture
  - kontext
  - marketing
  - technické znalosti

Zdůvodnění arch. rozhodnutí

- Zdůvodnění
  - proč je architektura právě taková
  - klíčová funkčnost („architecturally significant use cases“)
  - vztah ke kontextu, omezujícím podmínkám, historii

Příklad: Web KIV OpenCms

Architektonické pohledy

Pohledy: dekompozice architektury

- Taylor et al (2010):
  - „A *view* is a set of design decisions related by common concern(s).“
  - „A *viewpoint* defines the perspective from which the view is taken.“

5.3 Selection of architectural viewpoints

An AD shall identify the viewpoints se

Each viewpoint shall be specified by

a) A viewpoint name,
b) The stakeholders to be addressed
c) The concerns to be addressed by t
d) The language, modeling techniqu

5.2 Identification of stakeholders and concerns

An AD shall identify the stakeholders considered by the architect in formulating the architectural concept for the system.

5.4 Architectural views

An AD shall contain one or more architectural views.

Each view in an AD shall correspond to exactly one viewpoint, as selected in accordance with 5.3.

Each view in an AD shall conform to the specification of its corresponding viewpoint.

Each view shall include:

a) An identifier and other introductory information, as defined by the using organization
b) Representation of the system constructed with the languages, methods, and modeling or analytical techniques of the associated viewpoint
c) Configuration information, as defined by the using organization

An architectural view may consist of one or more architectural models.

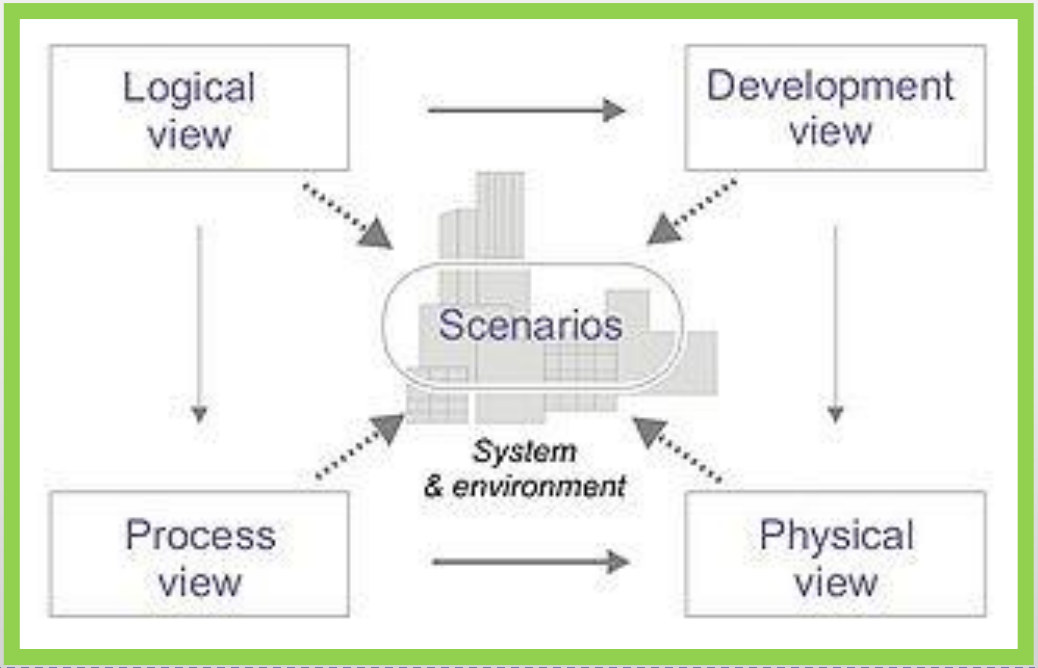
IEEE Std 1471

Přístup „4+1 pohled“

- Co a proč je významné
- RUP koncept (Kruchten)

P. Kruchten, “The 4+1 View Model of Architecture,” IEEE Software, vol. 12 (6), pp. 45-50, 1995.

- Obecný koncept:
  lokalita změn



logical view

Logické členění

- Motivace: monolitická aplikace nepřehledná

- Subsystém = skupina souvisejících prvků implementace tvořících funkční celek
  - funkčně soudržné (sada fčních požadavků)
  - často vázané na jednoho aktéra
- Balík
  - agregace prvků implementace - jmenný prostor
  - snaha o přehlednost a srozumitelnost
  - oddělení veřejných a privátních prvků implementace
  - závislosti indukované vazbami obsažených prvků

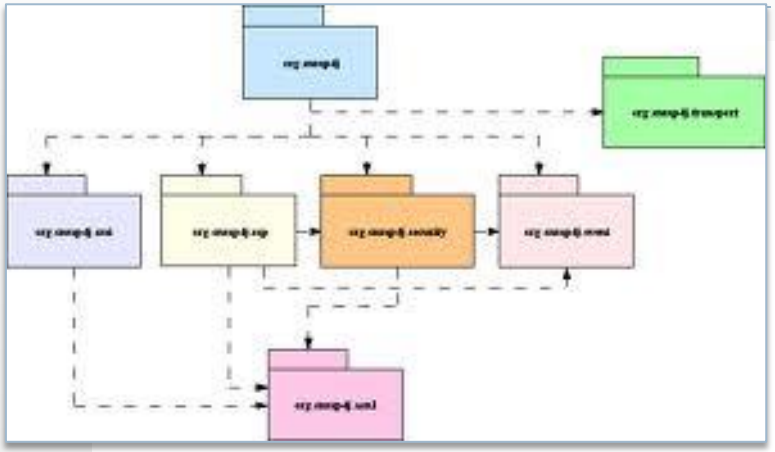
horizontální vrstvy vs. vertikální domény (finance, personalistika, řízení, utility, …)

Logické členění – balíky

- Motivace: logický objektový model velký
- Členění pro získání
  - přehledu o systému
  - rozdělení implementace mezi členy týmu
- Co je balík
  - skupina souvisejících tříd, tvoří organizační celek
  - mapování do jazyka (vytváří jmenný prostor)
  - hierarchické vnořování
- Třídy balíku
  - funkčně příbuzné, v jedné vrstvě aplikace nebo kdekoli

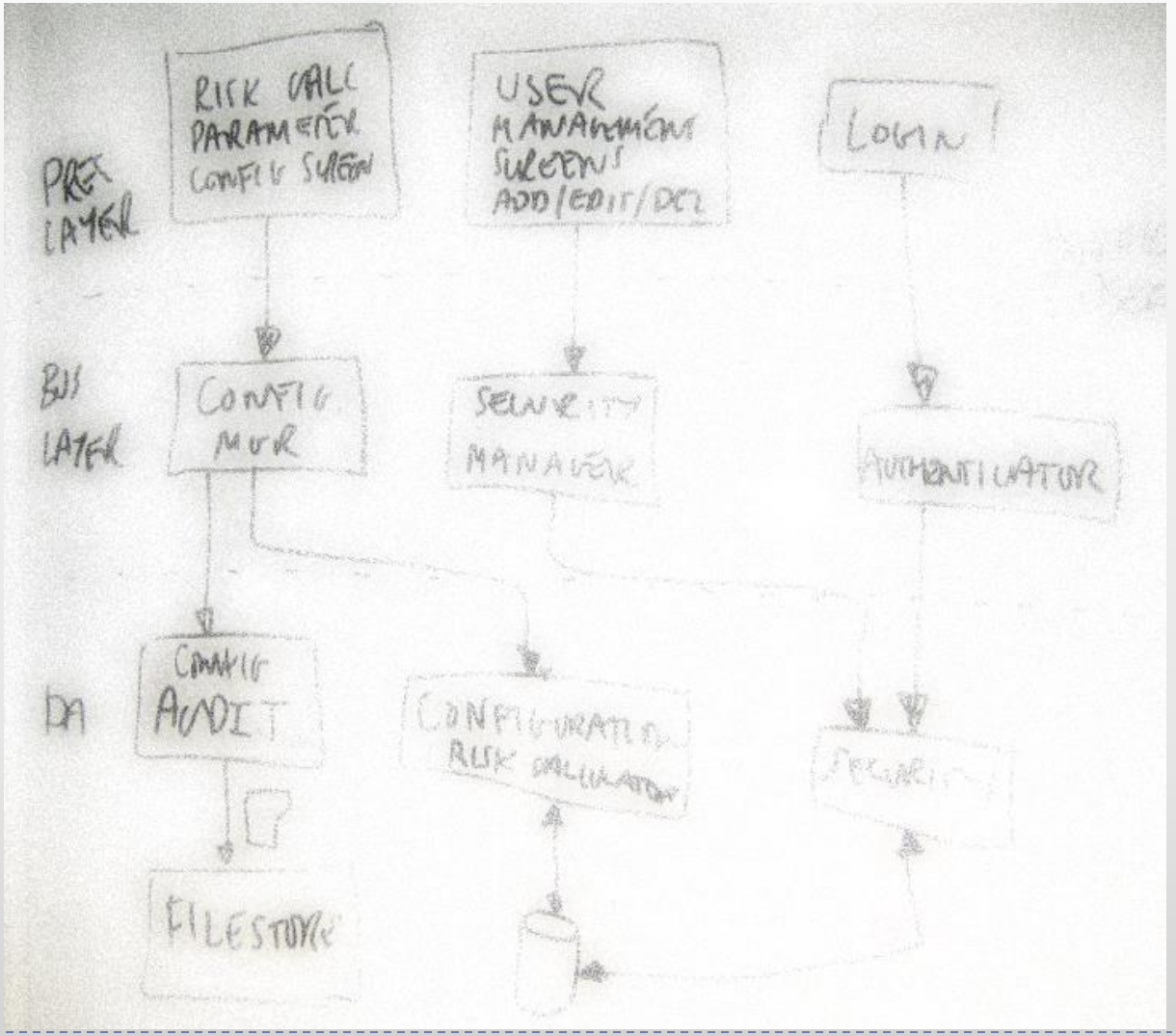
nezávisle na tom, zda daný jazyk zná koncept „package“

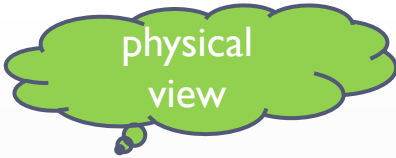
UML: Balíky a jejich vazby



„Boxes and arrows“ příklad

- (Brown 2009)





Fyzické členění

- Motivace: monolitická aplikace nepraktická

- Subsystém = skupina souvisejících prvků implementace tvořících funkční celek
  - funkčně soudržné (lokalita implementačních změn)
  - samostatná správa – nasazení, údržba, přístup
- Modul, komponenta, knihovna
  - celek vhodný pro samostatný vývoj, nasazení a údržbu
  - snaha o reuse = vícenásobnou použitelnost
  - => kvalitativní charakteristiky důležité
  - nutnost znát a spravovat závislosti

Poznámka: Jak mají vypadat moduly

D.Parnas: On the Criteria To Be Used in Decomposing Systems into Modules. Communications of the ACM, 15(12), 1972

- + kvalitativní charakteristiky dobrého návrhu
  - pravidla (information hiding, open-closed princip)
  - metriky (složitost, fan in/out)
- Komunikace mezi moduly: rozhraní a kontrakt

Různý výklad pojmu „komponenta“

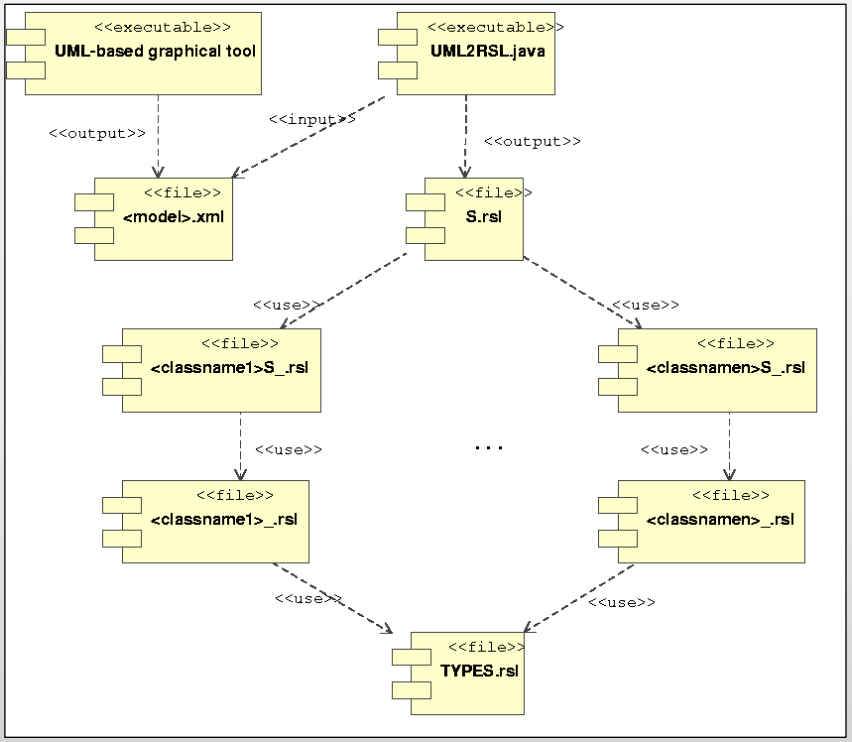
Komponentový přístup

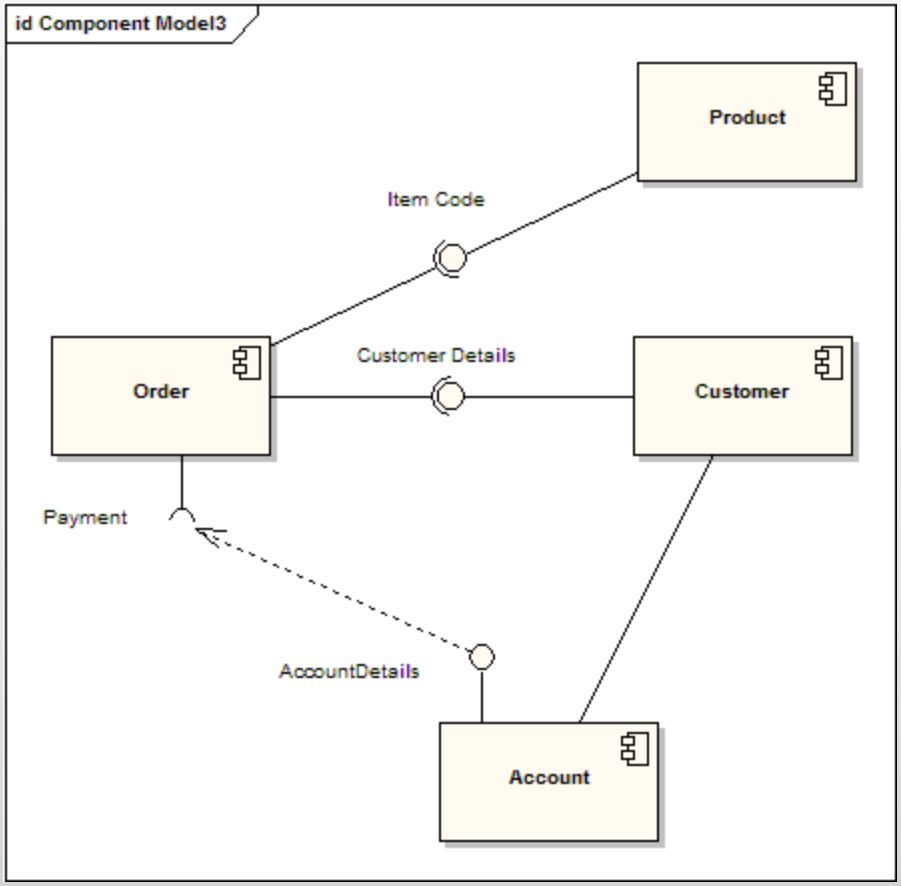
- Dotažení modularity, zapouzdření a rozhraní do konce
  - aplikace jako Lego skládačka (teoreticky)
- Komponenta
  - black-box – implementace „nedůležitá“ a „nedosažitelná“
  - rozhraní rozlišena na poskytovaná a vyžadovaná
    => vazby
  - explicitní specifikace rozhraní a vlastností
- Technologie
  - CORBA, EJB (částečně)
  - portlety, OSGi
  - Spring, PicoContainer

Inversion of Control (IoC) a Dependency Injection (DI)

UML: reprezentace komponent

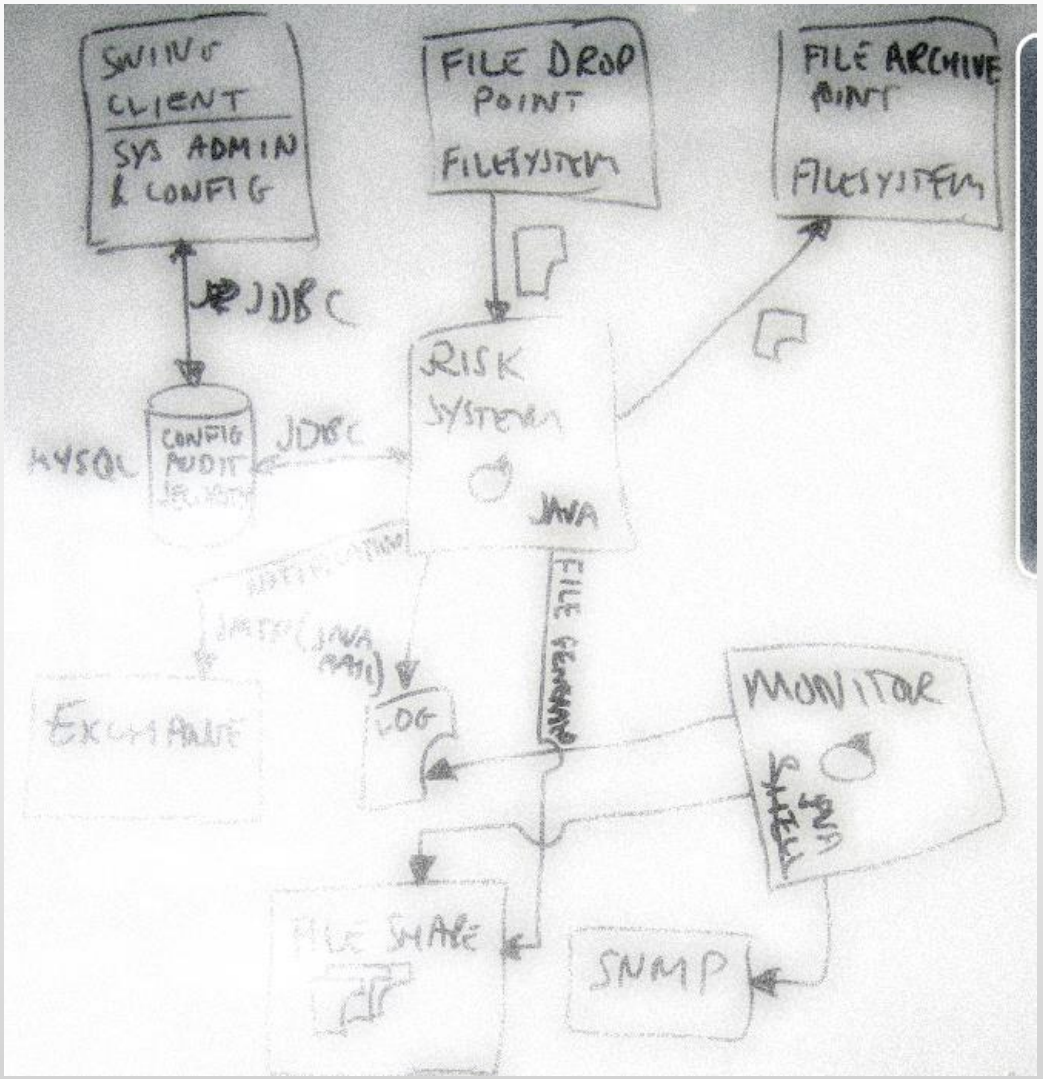
- Rozdíl UML1 a UML2





„Boxes and arrows“ příklad

- (Brown 2009)



development view

Od logické struktury k fyzické

- Realizace logické struktury se provádí v konkrétních technických artefaktech
- Tyto je potřeba vytvořit
  - jednotlivci, týmy, projekty
  - technologické aspekty
  - vývojářská infrastruktura

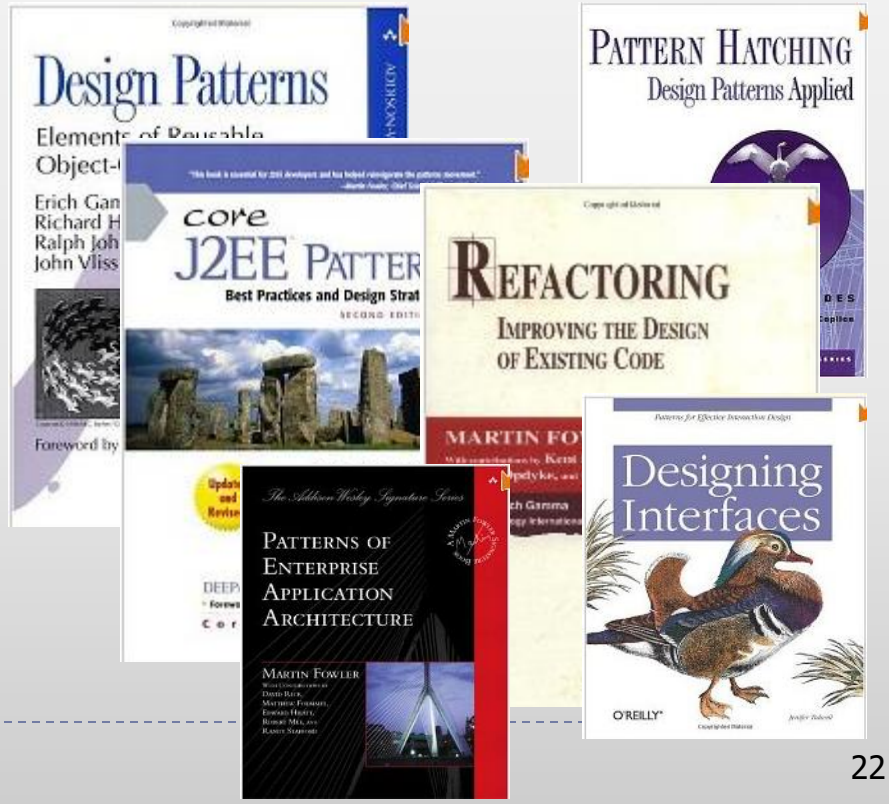
  => alokace prvků logické struktury do vývojových projektů

- Logický pohled: subsystémy, balíky => diagramy tříd
- Vývojový pohled: projekty, adresáře, sestavení
- Fyzický pohled: moduly, komponenty, knihovny

Konvence a politiky

- „Architectural policies“
  - obecná pravidla pro návrh v libovolné části aplikace
  - musí dodržovat všichni vývojáři

- Použité návrhové vzory
- Správa paměti
- Synchronizace, transakce
- Defenzivní programování
- Lokalizace (L10N, i18n)
- Dokumentace kódu
- …



Konvence: příklad

Pro ERA model a vytvářené tabulky platí následující pravidla:

- Všechny tabulky mají názvy VELKÝMI_PÍSMENY a prefix KIV_ následovaný zkratkou oblasti a jménem tabulky (např. KIV_PERS_AKTIVITY nebo KIV_PUBL_CLANKY).
- Jako primární klíč se používá `INT id`, atributy cizích klíčů začínají `FK_`.
- Každá ne-číselníková tabulka musí obsahovat atributy `zaznam_datum` (timestamp), `zaznam_autor` (FK do tabulky KIV_PERS_OSOBY) a `zaznam_aktivni` (boolean).
- Jmenné a popisné entity (viz níž
  respektive `popis_cz|en`.
- Rozlišení lokalizovaných atribut

4.2 Implementace vzoru Mediátor

V systému používáme pro mediátory rozhraní *IMediator*, ve kterém jsou definované jednotlivé řídící metody *handle* pro obsluhu požadavků z jednotlivých JSP stránek. Metoda *handle* správně vyhodnotí druh požadavku a tomu přizpůsobí další činnost. Toto rozhraní je implementováno abstraktní třídou *AbstractMediator*, která obsahuje základní utility metody (viz Obrázek 2). V současné podobě implementace má každý modul své rozhraní *IMediator* a to z důvodu různorodých potřeb[1] (může si zavést vlastní metody *handle* s potřebným počtem a typem parametrů). Je ale samozřejmě důležité nezavádět zbytečné metody *handle* a raději lépe analyzovat situaci a pokusit se využít již zavedené metody *handle*.

[1] Ideálně bychom měli dospět k tomu že IMediator a AbstractMediator jsou společné (v modulu Common) a každý modul má svou třídu/třídy *XyzMediator* oddědenou od *AbstractMediator*.

Procesní pohled

- Struktura paralelizace v implementaci
  - procesy, vlákna; způsob synchronizace
  - komunikace – synchronní/asynchronní
  - propustnost, škálovatelnost, odolnost (deadlock, fault)
  - podpora (OS, jazyk, knihovny)
- Vazba na strukturu nasazení (distribuované systémy)

- Procesní model (workflow) systému
  - alokace aktivit do modulů implementace
  - synchronizace, předávání artefaktů

Přístup „Více pohledů“

- Context, Constraints, Technology selection, Principles
- Views:
  - Functional (key UCs),
  - Process,
  - Non-functional,
  - Logical, Interface,
  - Design,
  - Infrastructure, Deployment, Operational,
  - Security,
  - Data
- Justification

[Zdroj: původní Coding the Architecture, http://www.codingthearchitecture.com/2009/10/27/software_architecture_document_guidelines.html]

Vazby mezi pohledy

- Dáno souvislostmi mezi prvky systému
  - vazbami artefaktů a logických celků
  - vazbami artefaktů mezi sebou
  - aktivitou či pasivitou artefaktů za běhu
  - nasazením artefaktů na provozní infrastrukturu
- Dáno postupem tvorby
  - priorita funkčností a omezení rizik ovlivňují postup vytváření
  - závislosti artefaktů definují postup skládání
  - připravenost infrastruktury ovlivňuje nasazení

Tvorba, validace a dokumentace architektury

Kdy a jak tvořit návrh architektury

- Členění architektury sw ovlivňuje plán a postup prací

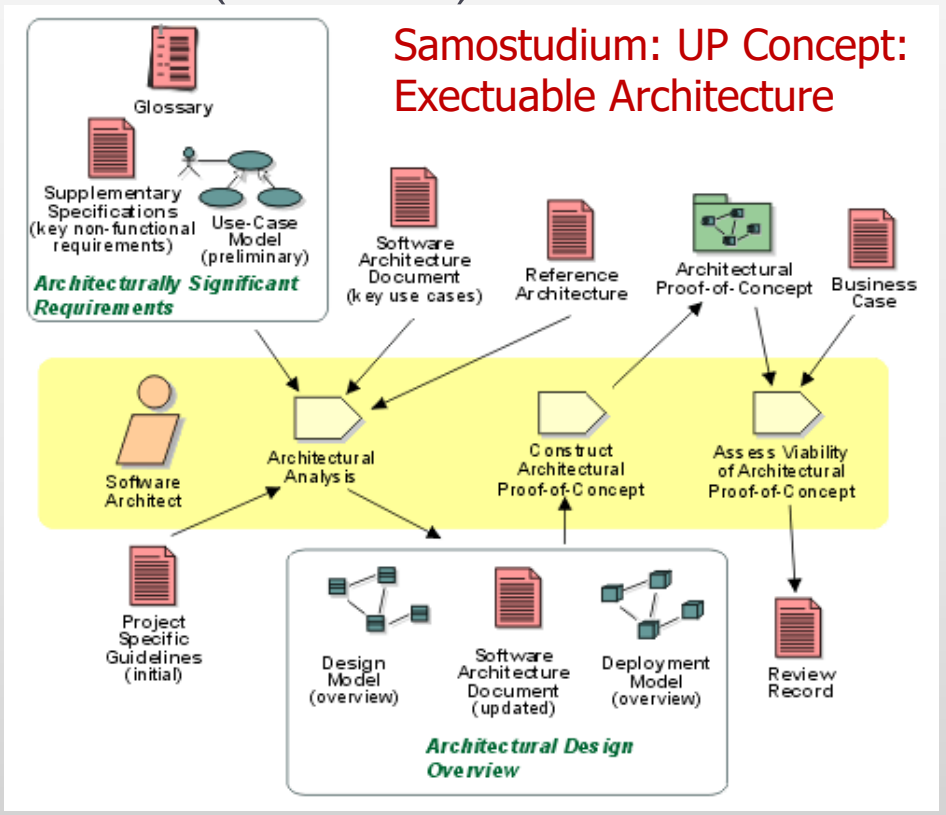
- Standardní projekty
  - možno odhadnout nebo stanovit předem
- Menší systémy
  - rozdělení na základě analýzy
- Velké projekty (analýza trvá dlouho)
  - nahrubo předem => rozfázování prací včetně analýzy
  - vodítko: aktéři, znalost struktury fyzického systému, možnosti reuse, vývojové kapacity
  - zpřesnění během činnosti návrhu a validace architektury

Samostudium: UP Task: Refine the Architecture

Validace architektury

- „Bude IOC, REL na tomto postavený splňovat LCO?“
  - executable architecture a její validace
  - architektonicky důležitá funkčnost (use cases)

- Mechanismy
  - návrh na základě klíčových use cases a mimofčních pož.
  - proof of concept implementace
  - oponentura



Dokumentace architektury

- Referenční architektura
  - dokument
  - kostra aplikace

Účel: porozumění, jak je produkt vytvořen a proč je tomu tak

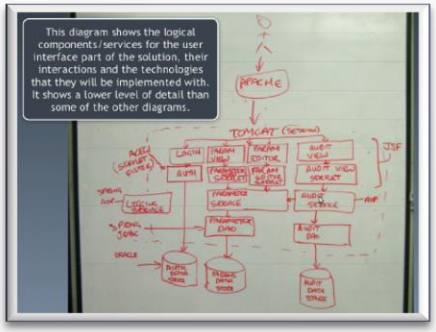
- Dokument
  - RUP Artifact: Software Architecture Document
  - IEEE Std 1471-2000

- Modely
  - UML: komponenty („physical view“), třídy, balíky („logical view“), interakce, stavový model („process view“)
  - ad-hoc diagramy (Visio, tabule)

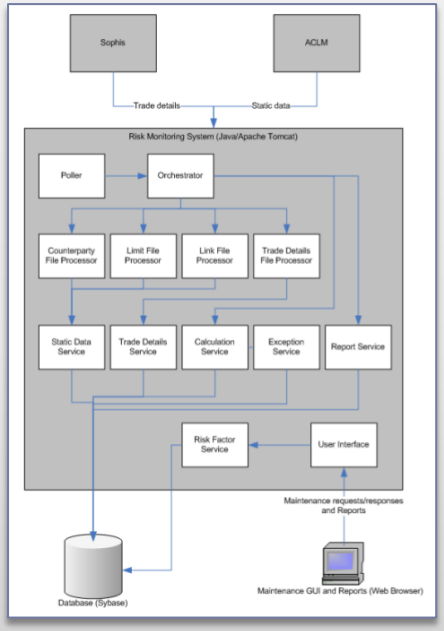
Samostudium: UP Checklist: Design

Příklad

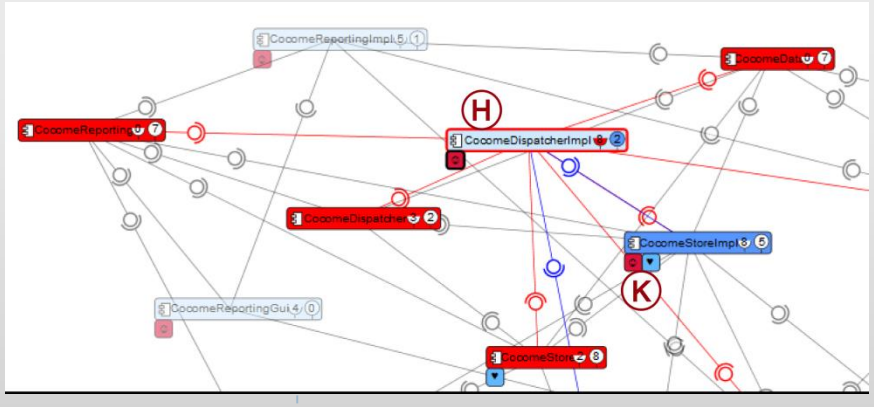
- Web KIV architektura modulů
- Tabule



- Visio



nástroje – architecture exploration

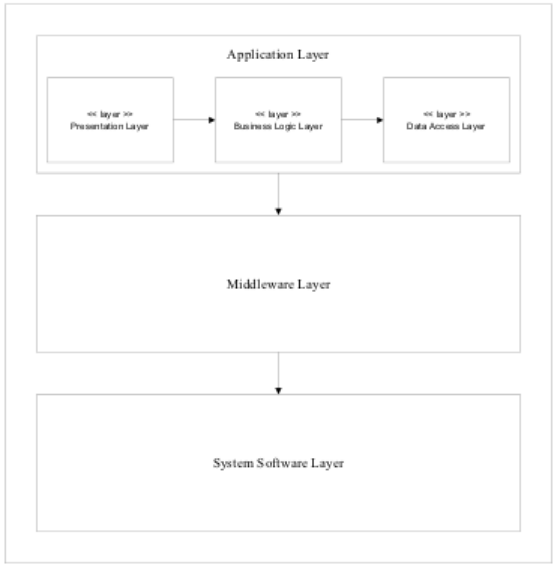


http://www.codingthearchitecture.com/presentations/yowsydney2014-software-architecture-sketches

Protipříklady

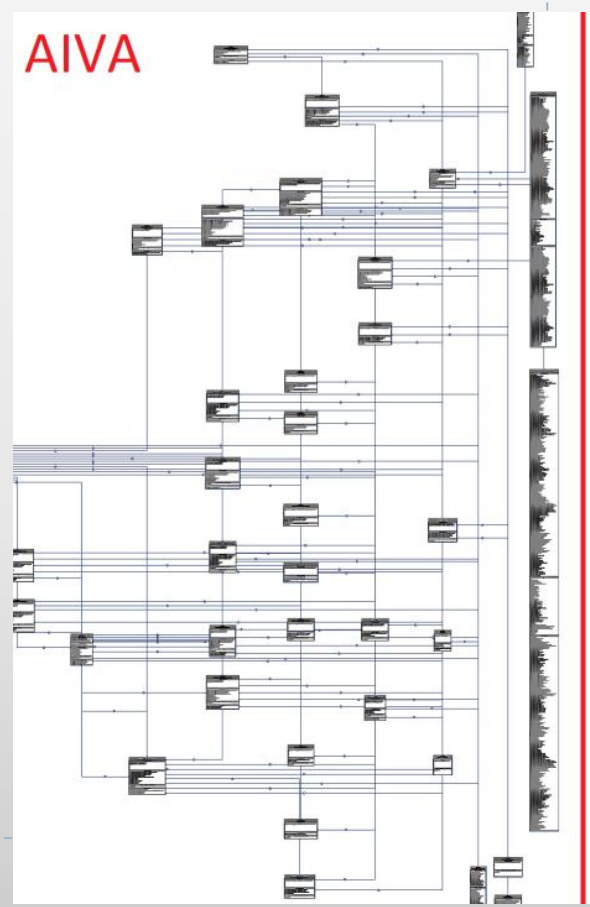
5.4.3 WUT Software Layers

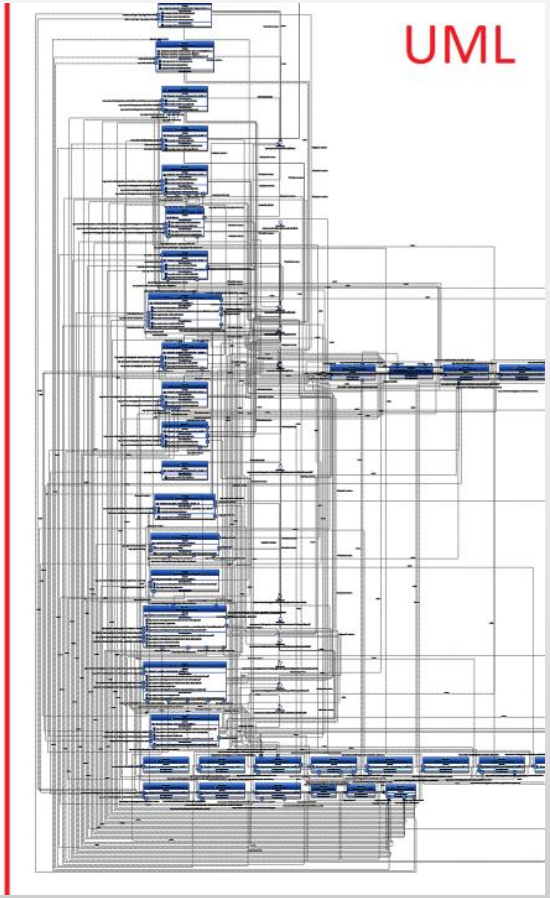
The WUT problem domain and solution space layers are graphically depicted in Figure 3. Note the directional dependencies between the layers within the problem domain as well as between the application layer and the Middleware and System Software layers.



zbytečné

nečitelné





Architektonické styly

Stavební prvky architektury

- (týká se strukturální části arch.)

- Základní prvky
  - komponenty, služby
  - rozhraní
  - konektory, filtry a sdílená data
- Vyšší úrovně
  - vrstvy
  - subsystémy

Architektonické styly

- Zavedené zvyky a standardy

"An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them."

Nulová varianta

Vrstvená architektura

- Delegování na podřízené
- Komunikace se sousedy

- Vrstvy např.
  - prezentace / řízení / doména / business služby / technická infrastruktura / knihovní třídy / systémové

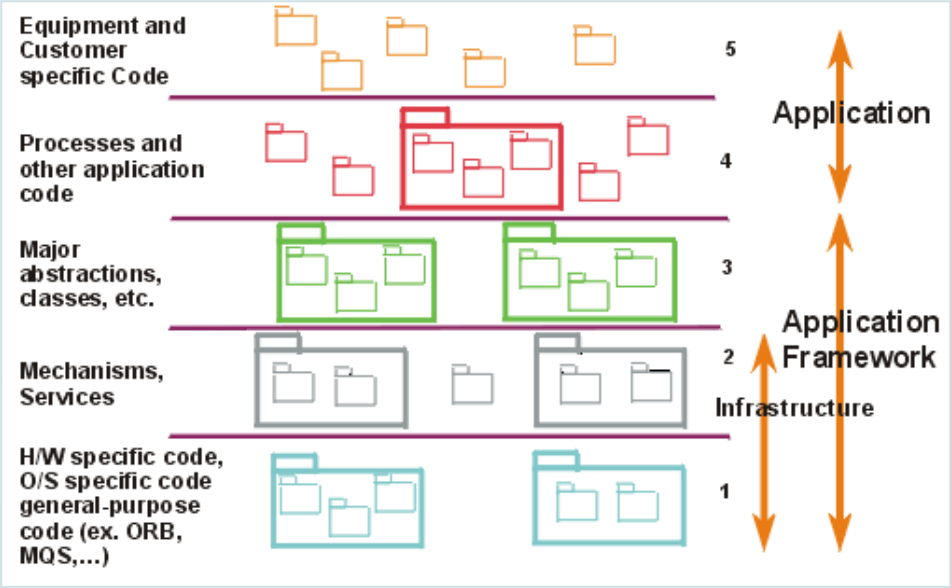


Image from IBM RUP 2005 Concept: Architecture

- Klient-server
  - tlustý klient
- 3-vrstvé a vícevrstvé
  - oddělení prezentace, business logiky a datové části
  - dnes standard

Servisně orientované styly

- SOA
  - varianta: Enterprise Service Bus
- Microservices
- Broker

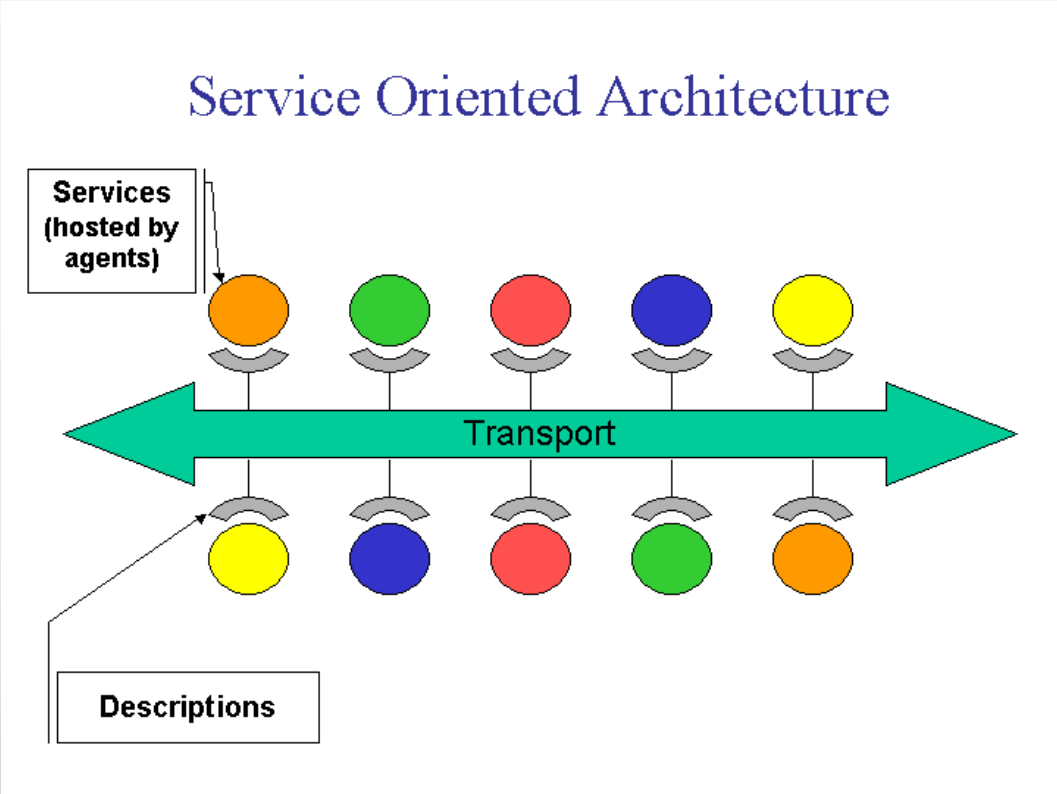


Image from http://lists.w3.org/Archives/Public/www-ws-arch/2003Feb/0030.html

Datově orientované styly

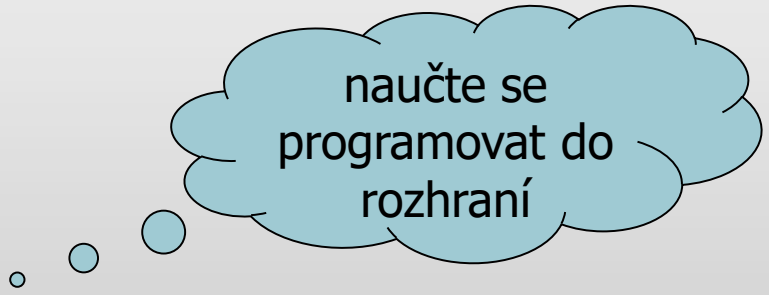
- Pipes and filters („kolona“)
- Blackboard
- Map-Reduce
  - e.g. http://architects.dzone.com/articles/how-hadoop-mapreduce-works

Rekapitulace

Architektura a návrh: rekapitulace

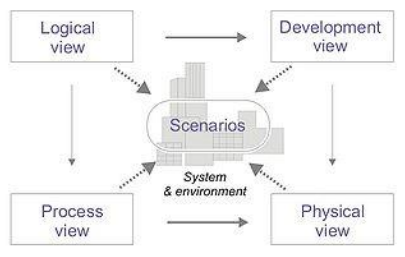
- Popis architektury: jak je produkt udělán, důvody, ověření
- Definuje (skoro) všechny podstatné věci implementace
  - vazba na konkrétní technologie
  - vyřešený způsob komunikace na okolní prostředí
  - struktura aplikace a rozhraní mezi částmi
  - používané konvence a návrhové vzory
- Rozpracovaná funkčnost validuje architekturu

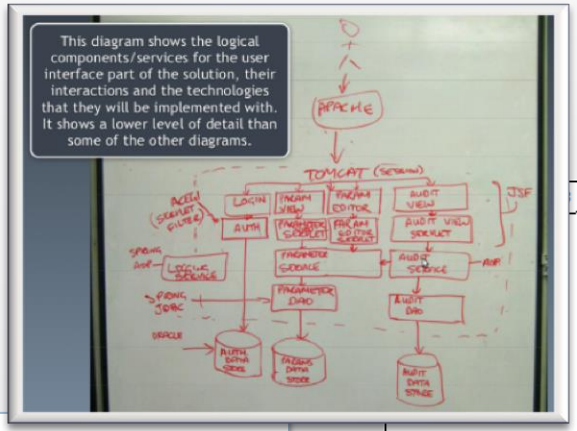
- Podrobnosti: KIV/SAR

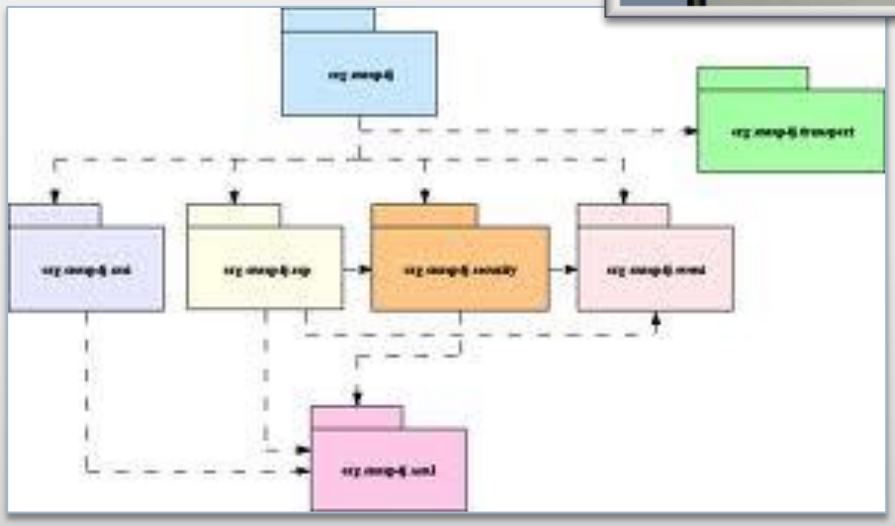


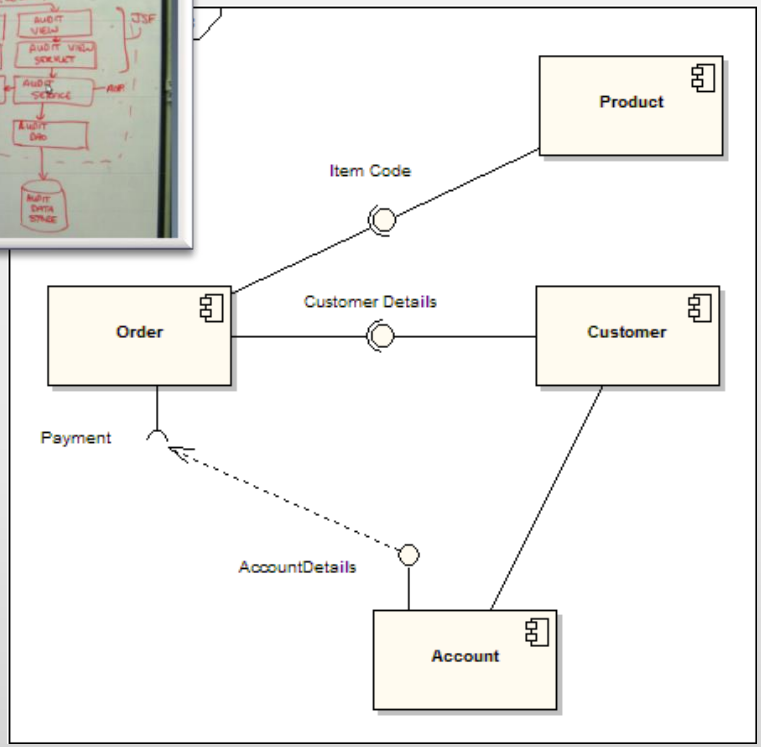
Struktura produktu

- Části, ze kterých je složen, a jejich vazby
  - logicky
  - fyzicky
- Moduly / balíky
- Komponenty









Or / Also you can read …

http://www.codingthearchitecture.com/presentations/sa2009-documenting-your-software-architecture-why-and-how