Vývoj Aplikácií s Viacvrstvovou Architektúrou

03. Vývojové technológie

IT ACADEMY

STU FIIT

Lektor: PhDr. Ing. Mgr. Miroslav Reiter, DiS., MBA, MPA
Kontakt: riaditel@it-academy.sk

Čo nás čaká a neminie...

1. časť

- Úvod do Javy
- Štruktúra platformy
- Vývojové technológie
- Kolekcie
- Logovanie
- Lokalizácia

2. časť

- XML, IO
- Regulárne výrazy
- Modularita
- JDBC
- Bezpečnosť
- Prehľad JEE a .NET

Best practices | Faily | Fuckupy

Čo nás čaká a neminie...

1. časť

Architektúra

2. časť

Java

Best practices | Faily | Fuckupy

Čo je architektúra v kontexte TOGAF?

ISO/IEC/IEEE 42010:20118

- The fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.

The TOGAF standard

- The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time.

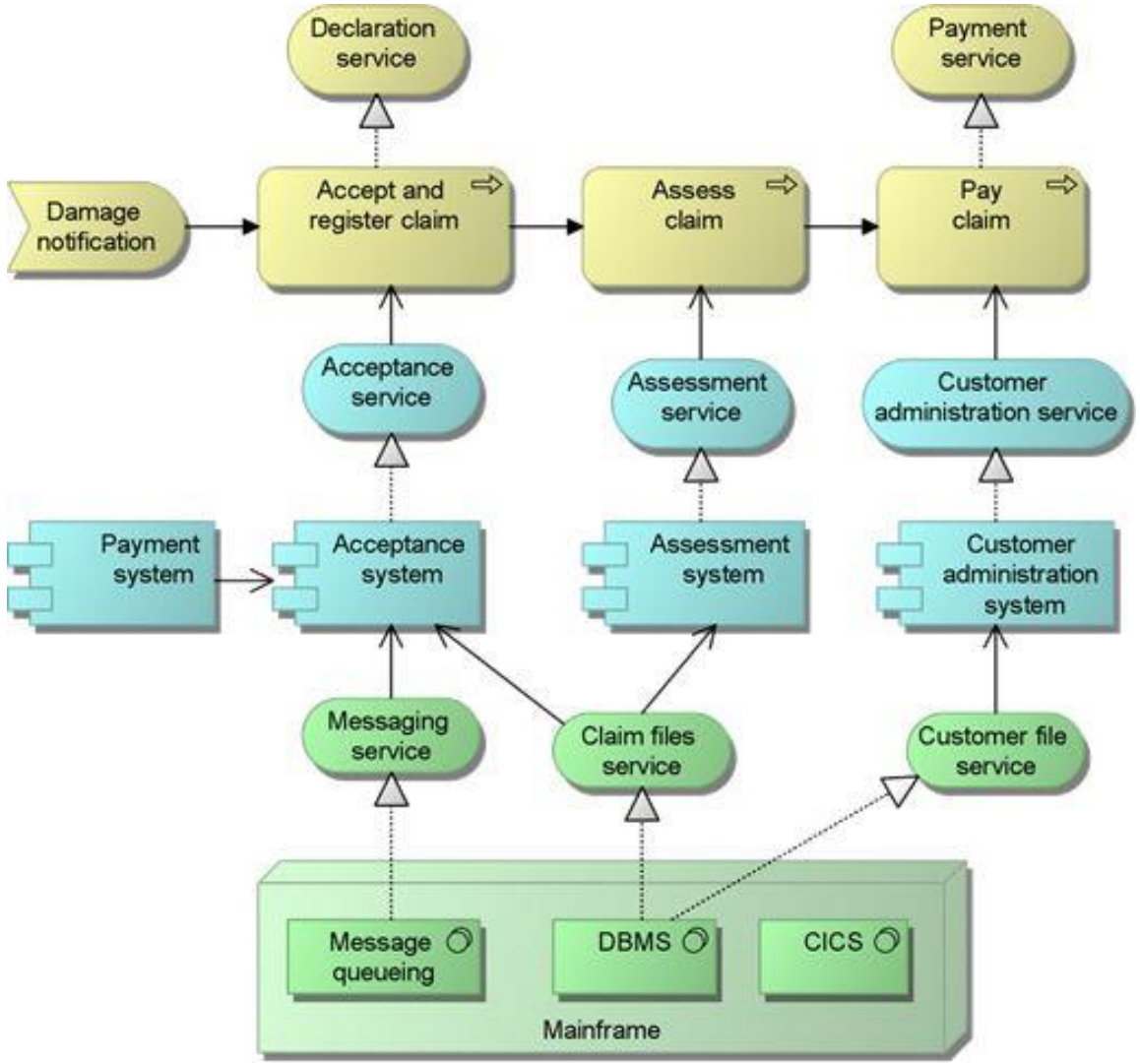
Čo je to podniková architektúra?

- Podniková architektúra zahŕňa **opis cieľov organizácie**, spôsoby ako sú tieto ciele dosahované prostredníctvom **obchodných postupov** a **spôsobov**, ako môžu tieto **procesy** byť **podporené technológiami**.

-- Roger Sessions

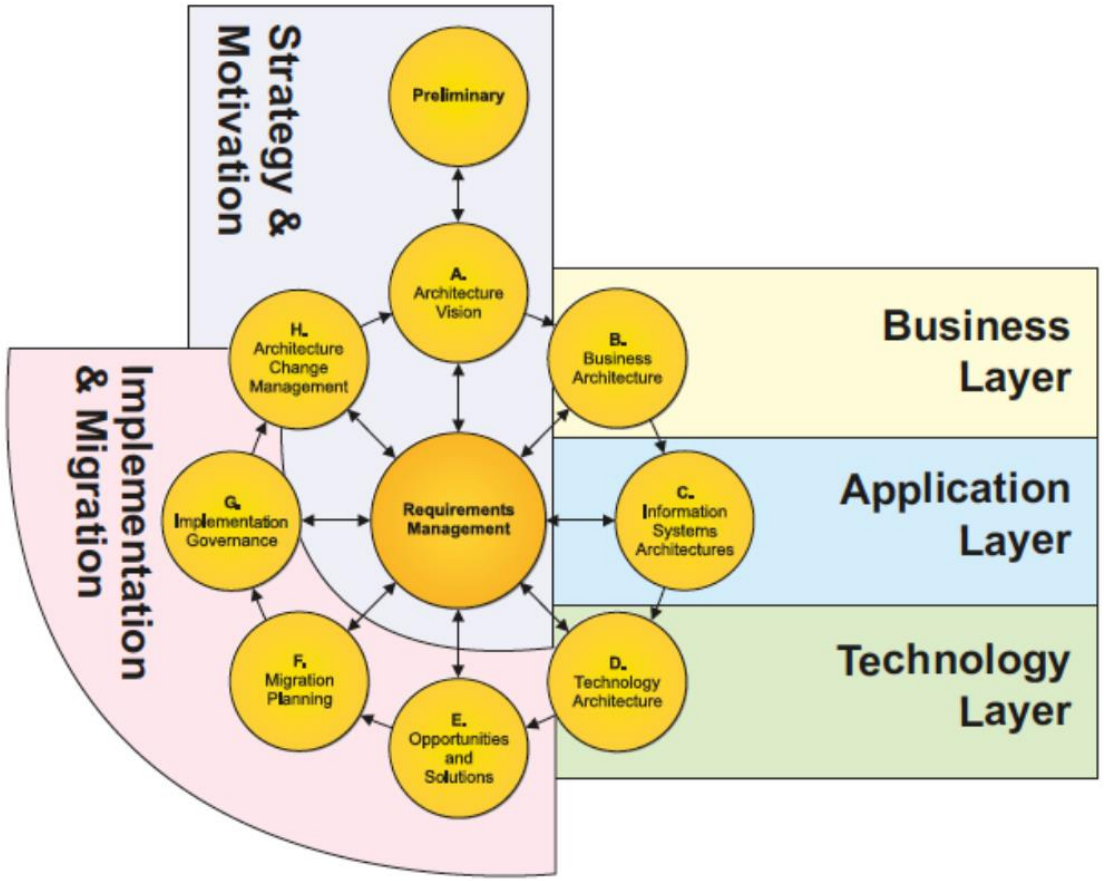
Čo je ArchiMate®?

- **Modelovací** (**vizuálny**) jazyk so množinou **predvolených** ikon na **opis**, **analýzu** a komunikáciu mnohých **záujmov** (**problémov**) **podniku**
- Poskytuje **množinu entít a vzťahov** s ich zodpovedajúcou ikonografiou pre reprezentáciu **opisov architektúry**
- Štandard Open Group pre architektúru

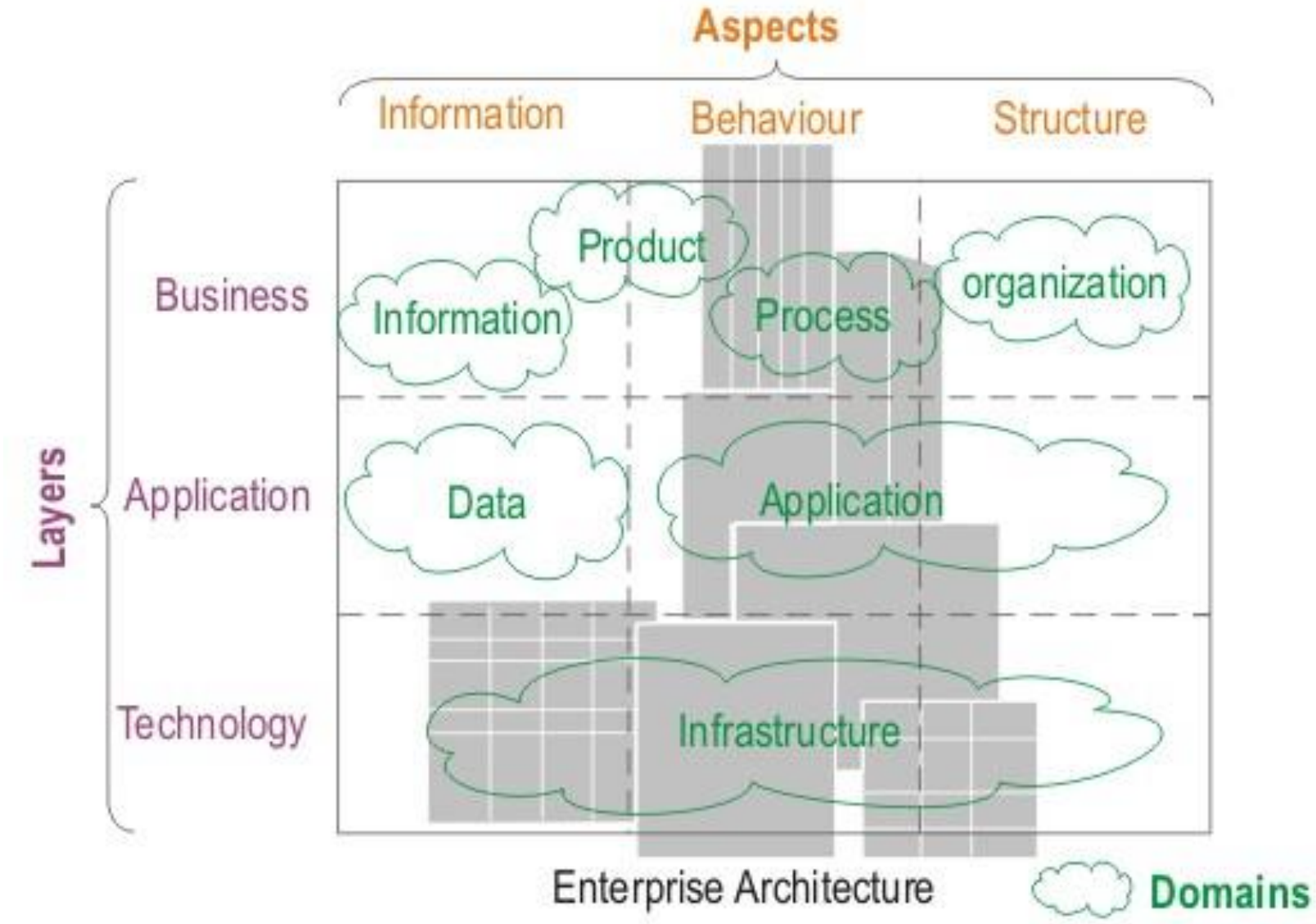


ArchiMate®, an Open Group Standard

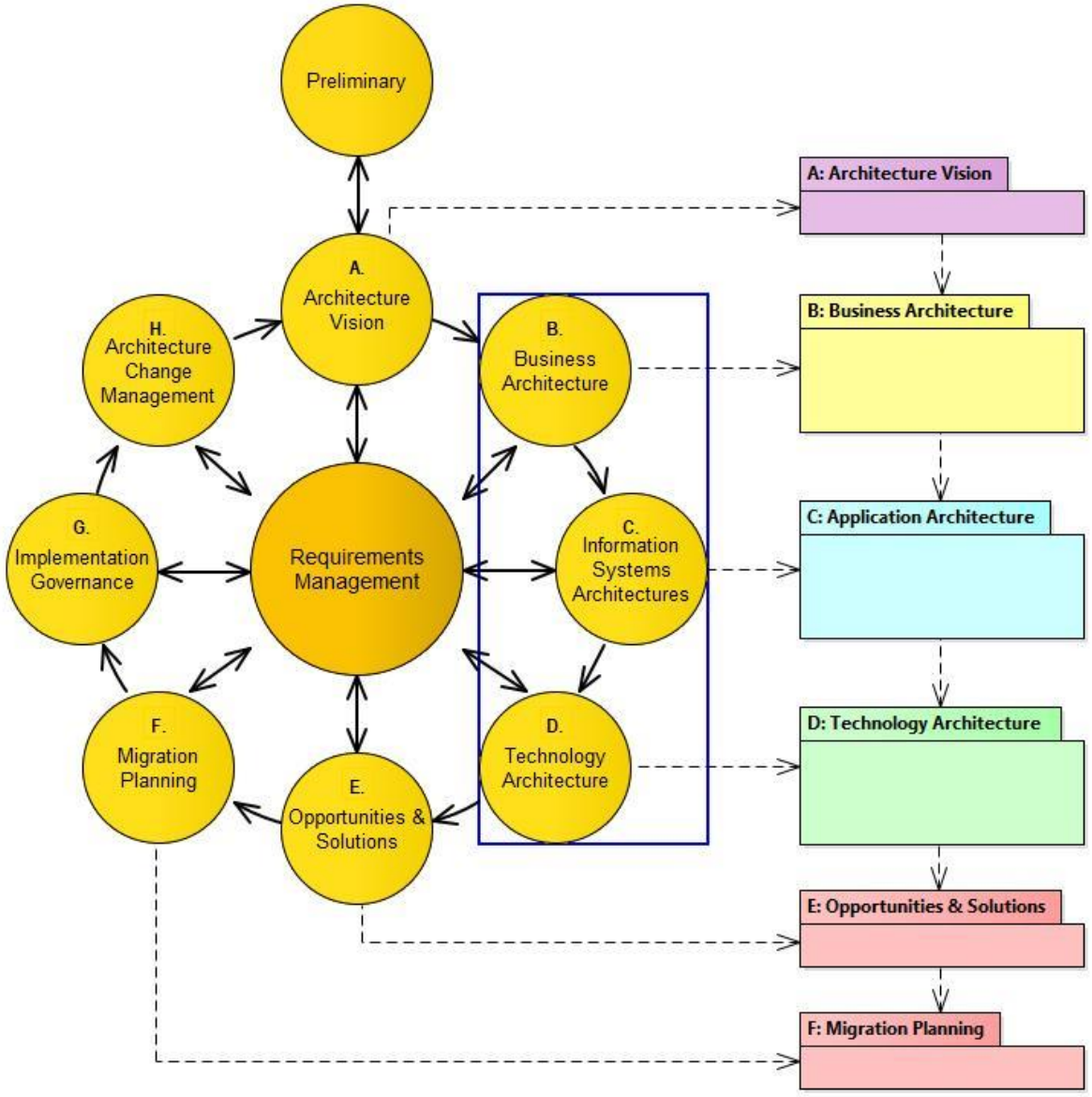
- Pomerne jednoduchý jazyk (napr. voči UML)
- Umožňuje modelovanie všetkých architektúr TOGAFu:
  - Biznis architektúry
  - Architektúry informačných systémov
  - Technologickej architektúry
- Okrem jazyka aj odporúčanie ako modelovať:
  - 18 architektonických pohľadov



Pohľad TOGAF a ArchiMate



Vzťah TOGAF a ArchiMate®



Prečo používať ArchiMate?

- Spoločný jazyk pre architektov
- 1 obrázok za 1000 slov
- Lepšia komunikácia vo vnútri spoločnosti
- Odstránenie nejednoznačností
- Umožňuje analýzy dopadu
- Kratšie zaškolenie
- Priama väzba na TOGAF

THE Open GROUP

DOWNLOAD

Welcome to the ArchiMate® 3.0.1 Specification, *an Open Group Standard*

Contents

Kto používa ArchiMate®?

Všetci veľkí hráči

Obmedzenia ArchiMate

Zakreslenie procesov

- Ponúka len základné prvky pre zakreslenie biznis procesov
- Pokiaľ je potrebné tieto procesy zakresliť do väčšieho detailu je vhodné použiť BPMN alebo diagram aktivít v UML

Popis aplikačných komponentov

- Ak je treba popísať vnútornú štruktúru aplikácií či informačných systémov, je vhodné použiť notáciu UML, ktorá k tomu poskytuje potrebné prostriedky

Biznis objekty

- ArchiMate nedovoľuje pridávať atribúty (podobne ako v diagramoch tried)

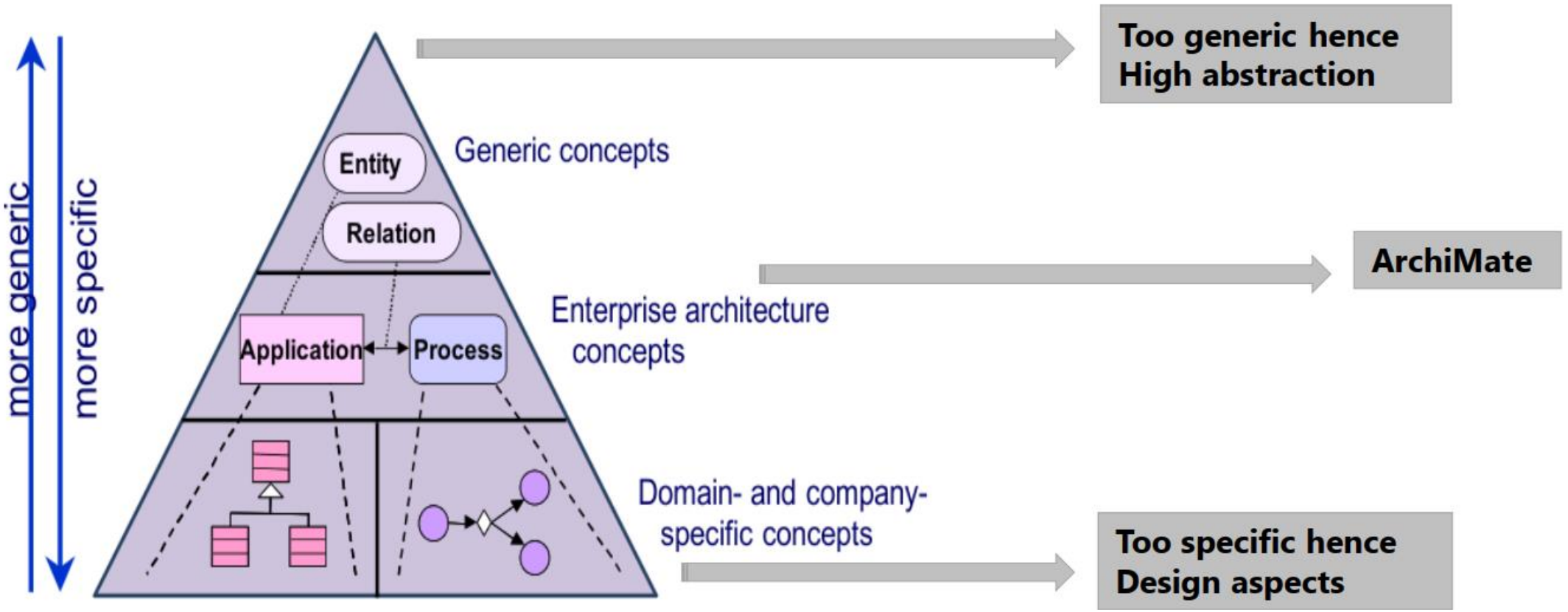
Pohľady v ArchiMate®

1. Úvodný pohľad
2. Organizačný pohľad
3. Pohľad kooperácie aktérov
4. Pohľad biznis funkcií
5. Pohľad biznis procesov
6. Pohľad kooperácie biznis procesov
7. Produktový pohľad
8. Pohľad správania aplikácie
9. Pohľad kooperácie aplikácií
10. Pohľad štruktúry aplikácie

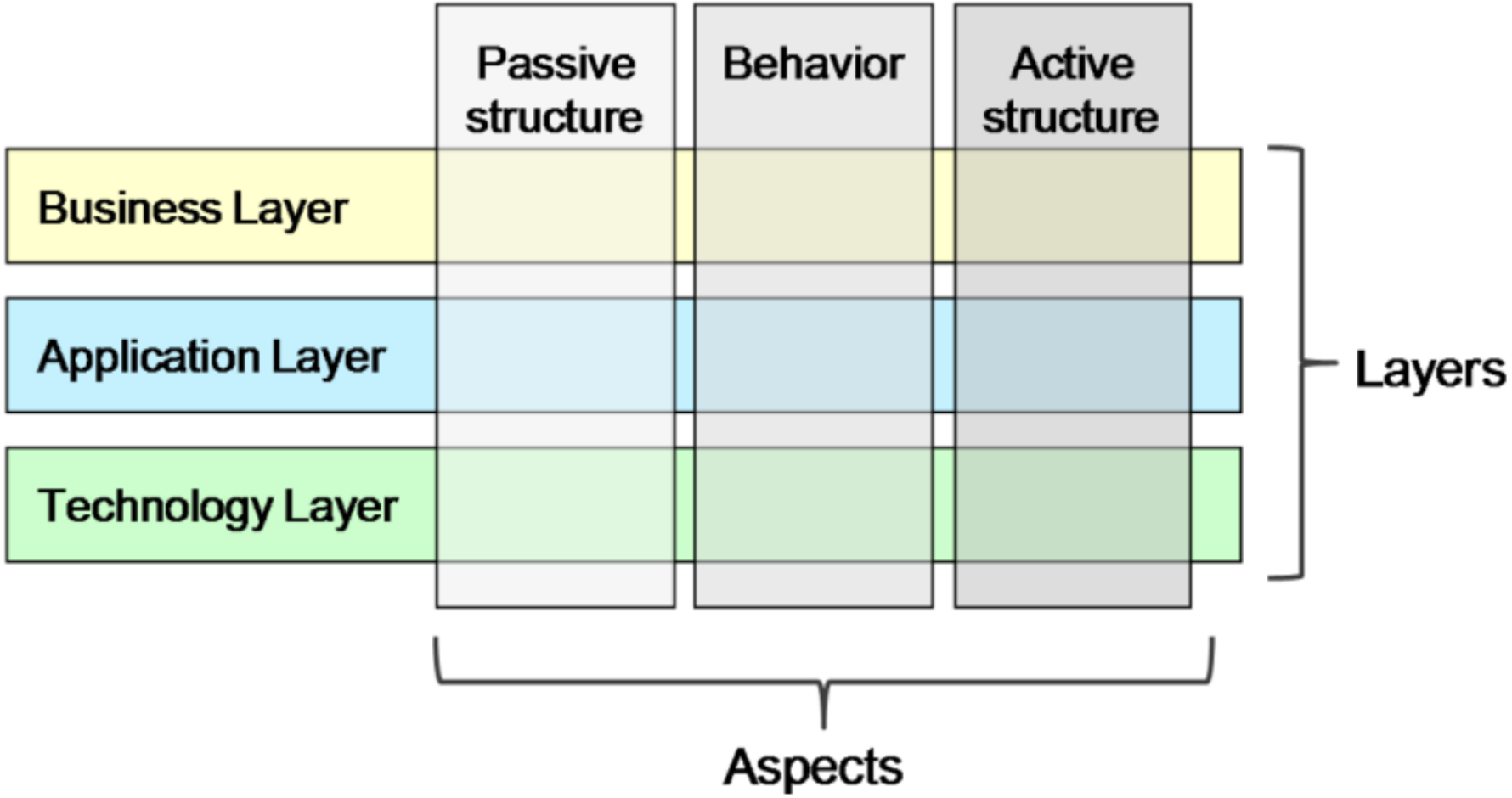
11. Pohľad použitia aplikácií
12. Infraštruktúrny pohľad
13. Pohľad použitia infraštruktúry
14. Pohľad implementácie a rozmiestnenia
15. Pohľad informačnej štruktúry
16. Pohľad realizácie služieb
17. Vrstvový pohľad
18. Mapový pohľad


Motivation Layer
Mission-Vision-Values View
Motivation View
Stakeholder View
Stakeholder Analysis View
Goals View
Principles View
Requirements View
Risk and Security View
Strategy Layer
Value Map - Strategy Map View
Goal to Strategy View
Strategy View
Strategy To Capability View
Business Model View
Capability Map View
Capability Planning View
Business Architecture Layer
Business Services View
Business Actors View
Business Roles View
Business Concepts View
Business Processes View
Business Functions View
Business Process Co-operation
Application Architecture Layer
Application Services View
Applications Map View
Application Functions View
Information View
Application Interfaces View
Application Integration View
Application Co-operation View
Layered View
Technology Architecture Layer
Technology Services View
Platform View
Technology Map View
Infrastructure View
Implementation and Migration Layer
Kanban Board (portfolio)
Programs & projects View
Roadmap View
Capability Realization View
Service Realization View

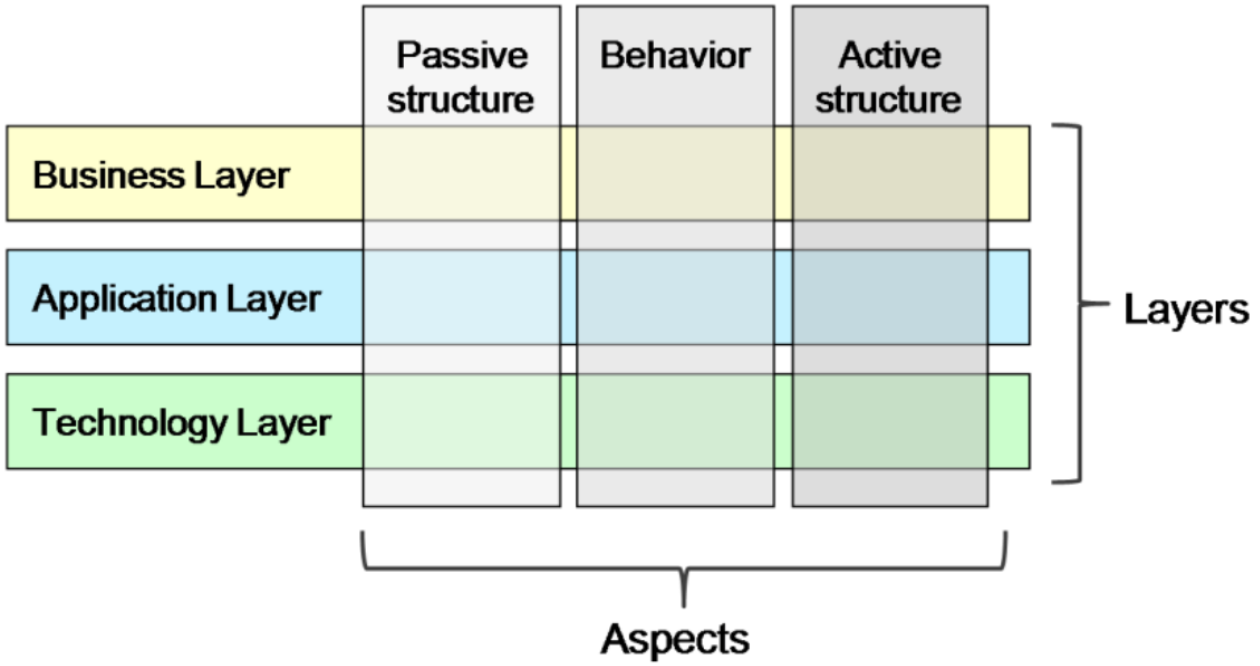
Popis architektúry na rôznych úrovniach



Core Framework – Vrstvy (Layers) a aspekty

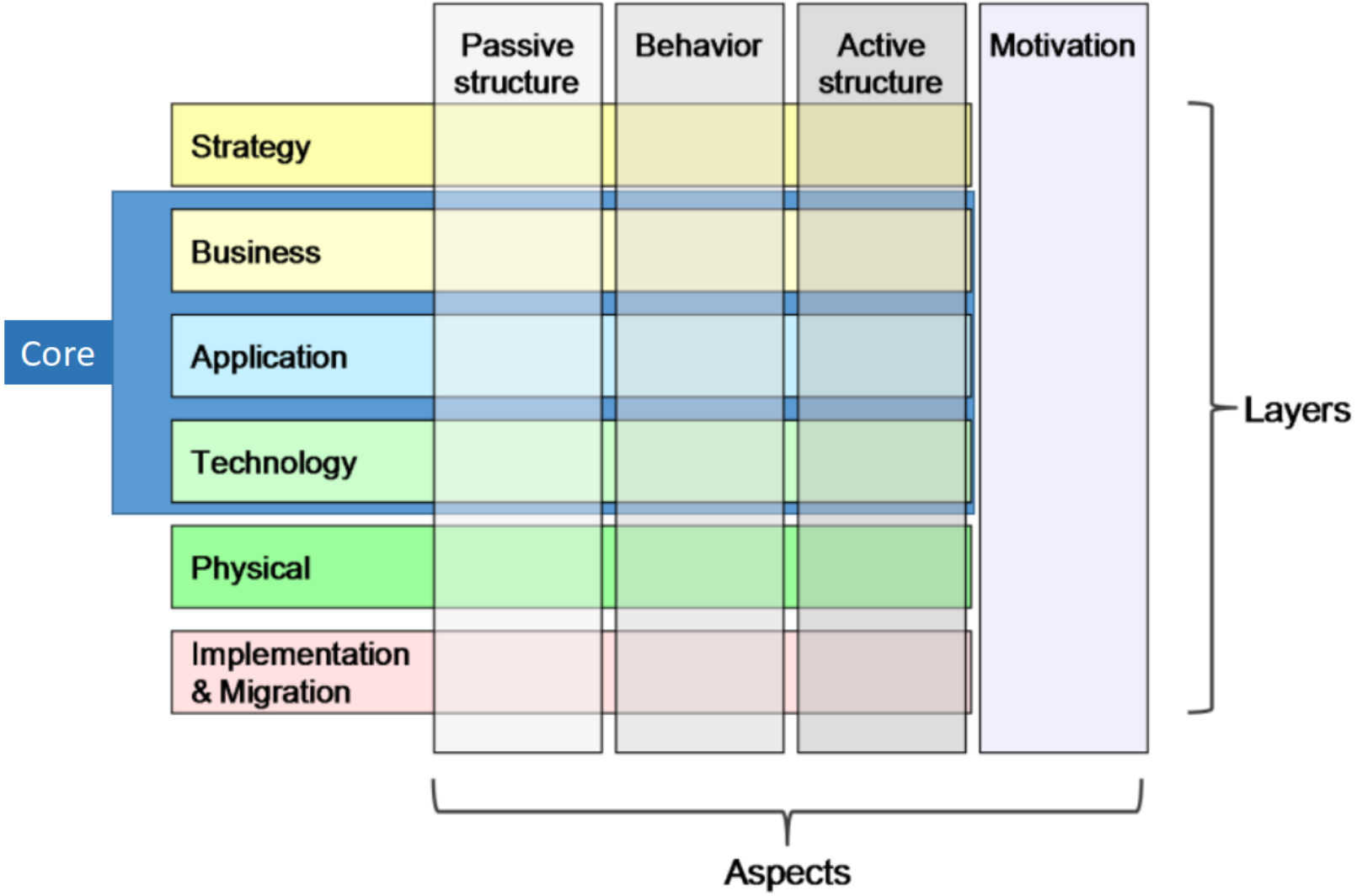


Core Framework – Vrstvy



1. **Business Layer** depicts business services offered to customers. These services are realized in the organization by business processes and are performed by business actors.
2. **Application Layer** depicts application services that support the business. These services are realized by application components.
3. **Technology Layer** depicts technology services serving the applications. These services (like processing, storage, and communication services) are realized by computer and communication hardware and system. Physical elements are added for modeling physical equipment, materials, and distribution networks to this layer.

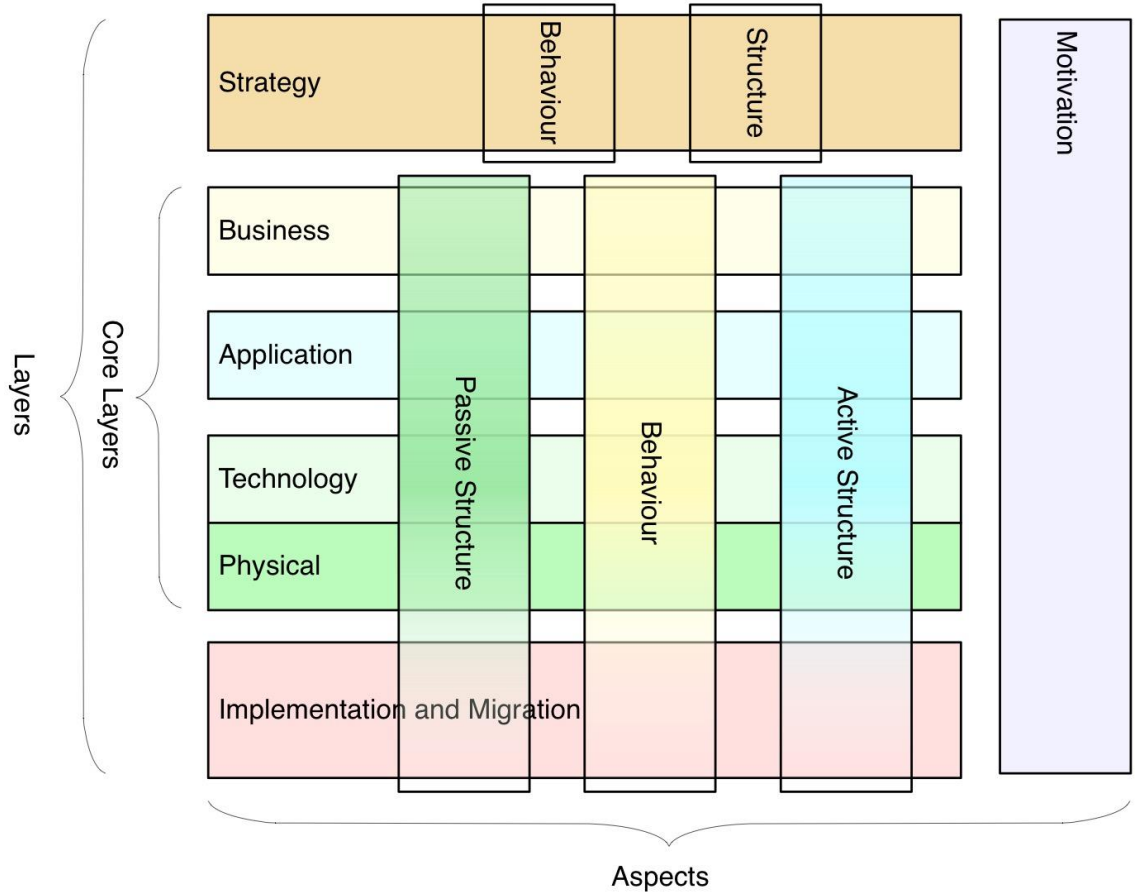
Úplný rámec - vrstvy a aspekty



Vrstvy (Layers) – ArchiMate Extensions

4. Strategy layer
5. Physical layer
6. Implementation & Migration layer

- Motivation extension

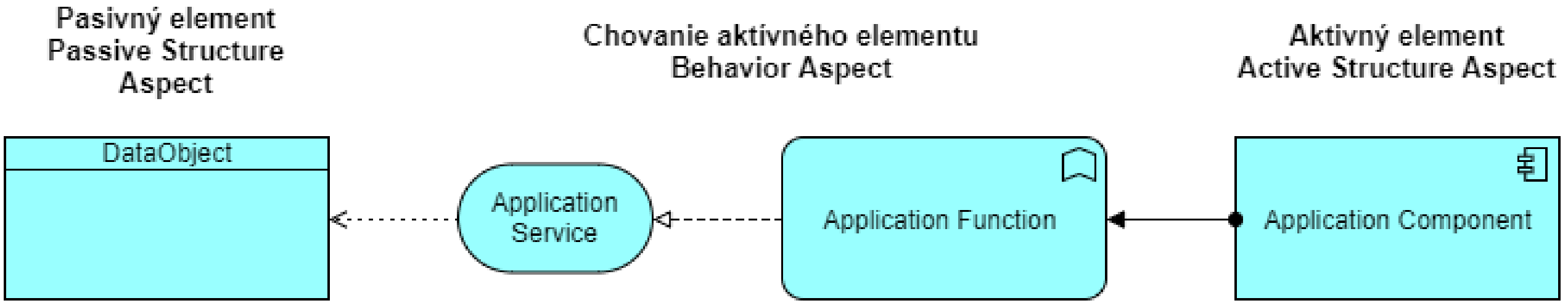


Aspekty (Aspects)

ArchiMate striktne oddeľuje 3 základné aspekty:

1. **Active Structure Aspect** – reprezentujú **štrukturálne elementy** napr. aktérov, role, komponenty, hardvér, ktoré sú schopné určitého chovania. Tiež sú tieto elementy označované ako aktívne.
2. **Behavior Aspect** – Reprezentuje **chovanie** (napr. procesy, funkcie, služby) **vykonávané aktívnymi** (štrukturálnymi) **elementami**. Chovanie je priradené (assigned) aktívnemu elementu.
3. **Passive Structure Aspect** (pasívne elementy) – **reprezentujú objekty** (napr. biznis objekty, dátové objekty ale aj fyzické objekty), na ktorých môže byť aplikované chovanie aktívneho elementu (či sú takýmto chovaním ovplyvnený)

Aspekty (Aspects)



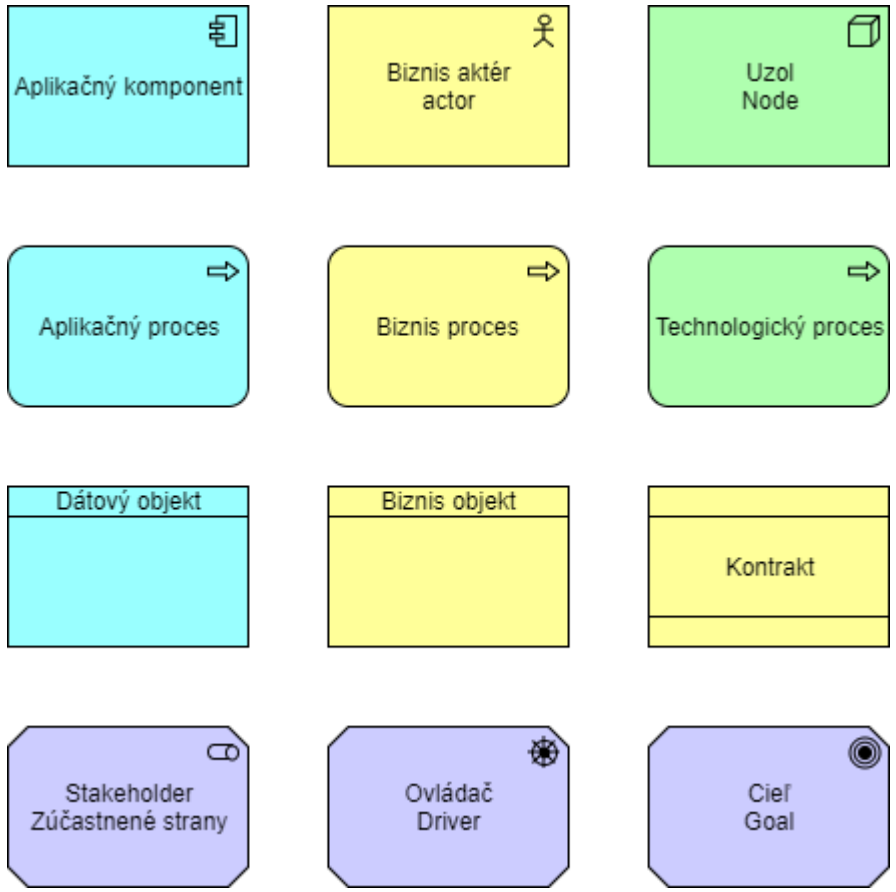
Pochopenie typov aspektov elementu

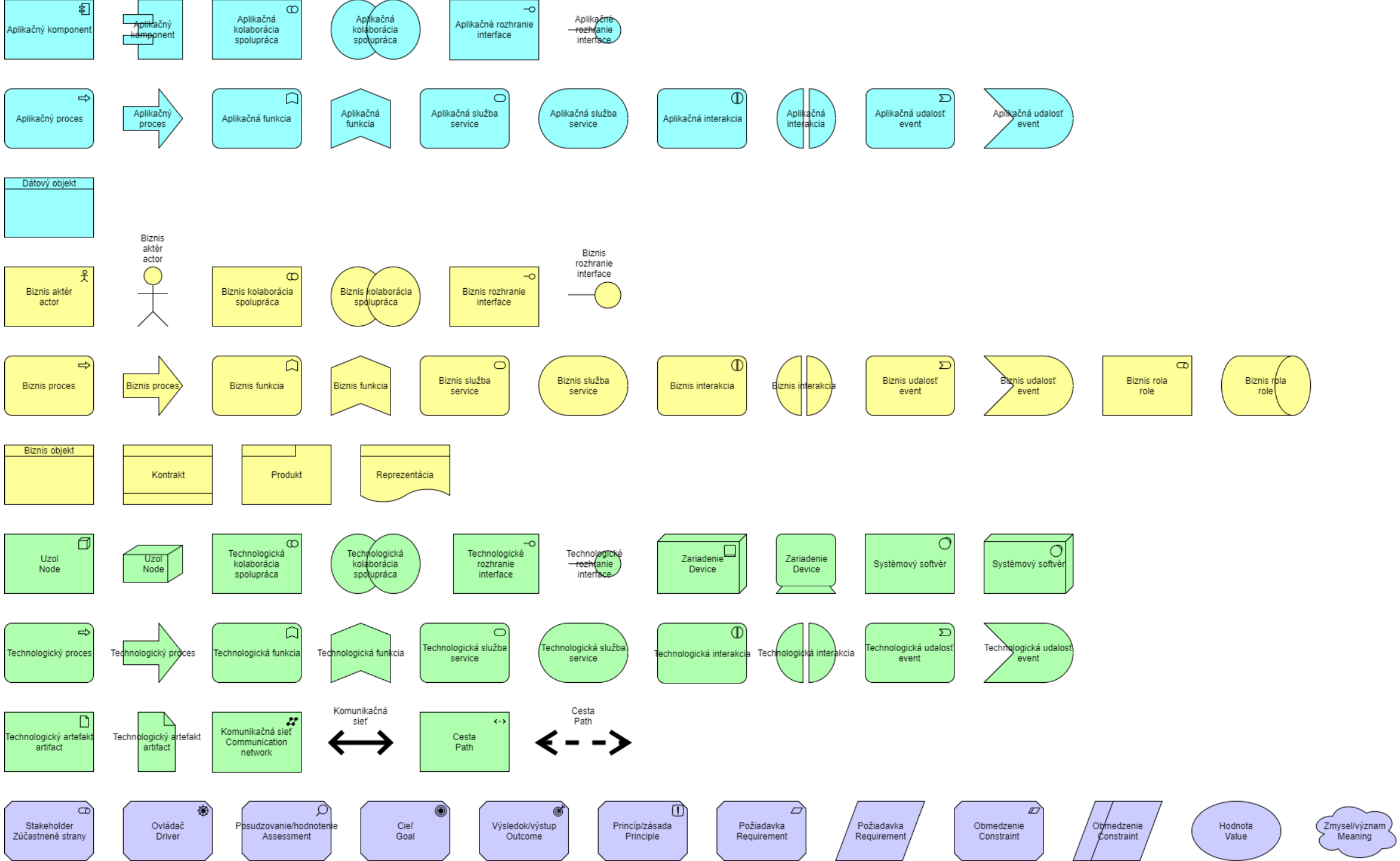
- Aká je aktívna štruktúra, behaviorálna a pasívna štruktúra?
  - Kuchár varí jedlo
  - Mechanik zvára auto
- **Kuchár/Mechanik**
  - **Aktívny prvok** štruktúry schopný **vykonávať správanie, zobrazuje** správanie – subject/**podmet** (**podstatné meno**)
- **Varenie/zváranie**
  - **Behaviorálny** prvok je jednotka **aktivity vykonávaná aktívnym elementom** (štruktúrami) - typicky verb/**prísudok** (**sloveso**)
- **Potraviny/Auto**
  - **Prvok pasívnej** štruktúry, **na ktorom sa vykonáva správanie** – objekt/**predmet** (**podstatné meno**) (informácie, údaje alebo fyzické)

Inšpirácia prirodzeným jazykom

Rozlíšenie elementov podľa rohov

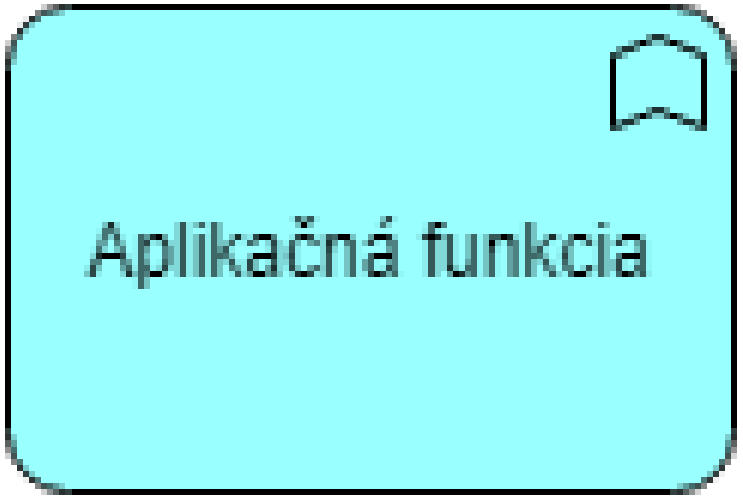
1. Ostré rohy – aktívne elementy
2. Zaoblené rohy – element chovania (behavior)
3. Ostrý roh s preškrtnutím – pasívne elementy
4. Skosené rohy – motivačný element




Aplikačný komponent
Aplikačný komponent
Aplikačná kolaborácia spolupráca
Aplikačná kolaborácia spolupráca
Aplikačné rozhranie interface
Aplikačné rozhranie interface
Aplikačný proces
Aplikačný proces
Aplikačná funkcia
Aplikačná funkcia
Aplikačná služba service
Aplikačná služba service
Aplikačná interakcia
Aplikačná interakcia
Aplikačná udalosť event
Aplikačná udalosť event
Dátový objekt
Biznis aktér actor
Biznis aktér actor
Biznis kolaborácia spolupráca
Biznis kolaborácia spolupráca
Biznis rozhranie interface
Biznis rozhranie interface
Biznis proces
Biznis proces
Biznis funkcia
Biznis funkcia
Biznis služba service
Biznis služba service
Biznis interakcia
Biznis interakcia
Biznis udalosť event
Biznis udalosť event
Biznis rola role
Biznis rola role
Biznis objekt
Kontrakt
Produkt
Reprezentácia
Uzol Node
Uzol Node
Technologická kolaborácia spolupráca
Technologická kolaborácia spolupráca
Technologické rozhranie interface
Technologické rozhranie interface
Zariadenie Device
Zariadenie Device
Systémový softvér
Systémový softvér
Technologický proces
Technologický proces
Technologická funkcia
Technologická funkcia
Technologická služba service
Technologická služba service
Technologická interakcia
Technologická interakcia
Technologická udalosť event
Technologická udalosť event
Technologický artefakt artifact
Technologický artefakt artifact
Komunikačná sieť Communication network
Komunikačná sieť
Cesta Path
Cesta Path
Stakeholder Zúčastnené strany
Ovládač Driver
Posudzovanie/hodnotenie Assessment
Cieľ Goal
Výsledok/výstup Outcome
Princíp/zásada Principle
Požiadavka Requirement
Požiadavka Requirement
Obmedzenie Constraint
Obmedzenie Constraint
Hodnota Value
Zmysel/význam Meaning

Spôsob zápisu elementov

Obdĺžnikový zápis



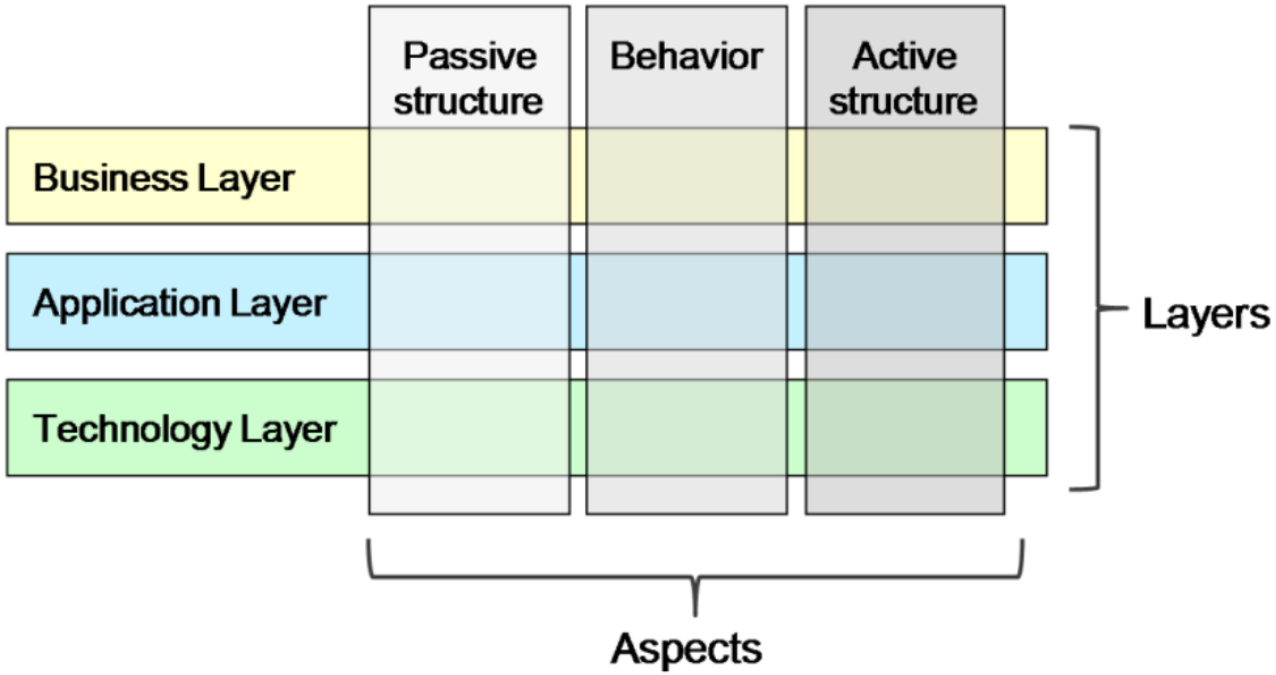
Zápis ikony

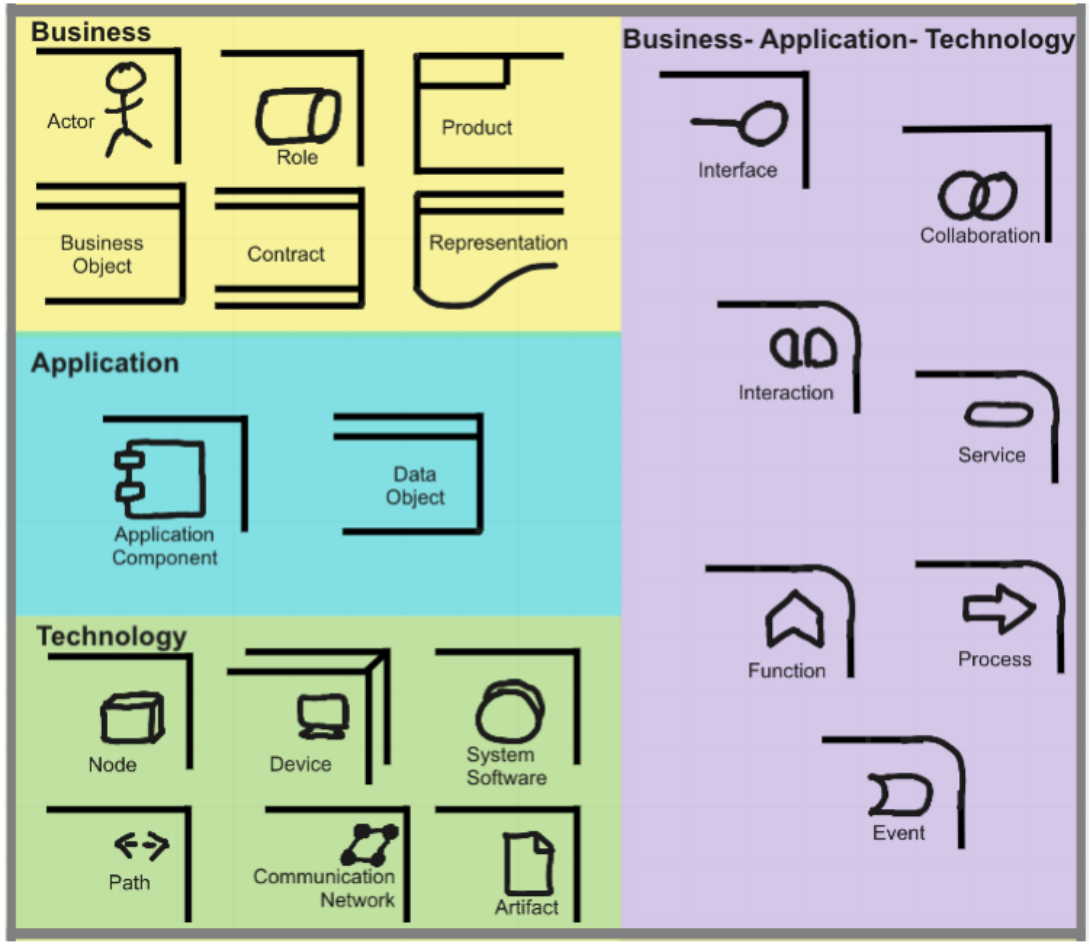
S	V	O	M	P	T
Subject/Podmet	Verb/Prísudok	Object/Predmet	Manner/Spôsob	Place/Miesto	Time/Čas
kto/čo?	čo robí?	koho/čo?	ako?	kde?	kedy?
Mike	rides	his bike	fast	in the park	every day.
Michal	*jazdí*	*na motorke*	*rýchlo*	*v parku*	*každý deň.*
Peter	writes	his homework		on Monday	morning.
Peter	*si píše*	*domácu úlohu*		*v pondelok*	*ráno.*
I	have	my dinner		at home	every evening.
Ja	*večeriam*			*doma*	*každý večer.*

Pochopenie typov aspektov elementu

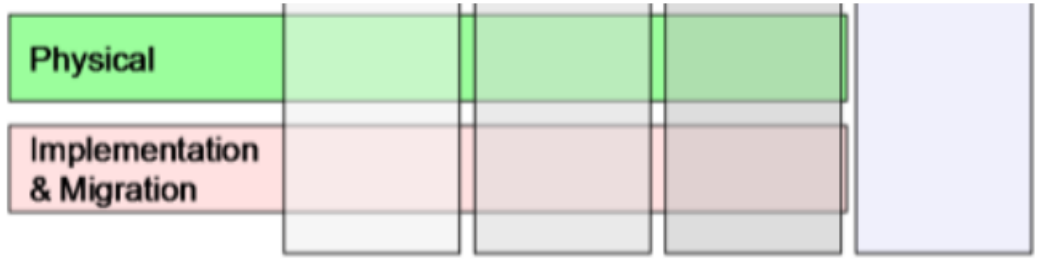
1. **Zástupca zákazníka**
   - Aktívny prvok štruktúry schopný vykonávať správanie, zobrazuje správanie - predmet
2. **Vybavovanie sťažností zákazníkom**
   - Behaviorálny prvok je jednotka aktivity vykonávaná aktívnou štruktúrou element (y) - (typicky sloveso)
3. **Správa o zákazníkovi**
   - Prvok pasívnej štruktúry, na ktorom sa vykonáva správanie – objekt (informácie, údaje alebo fyzické)

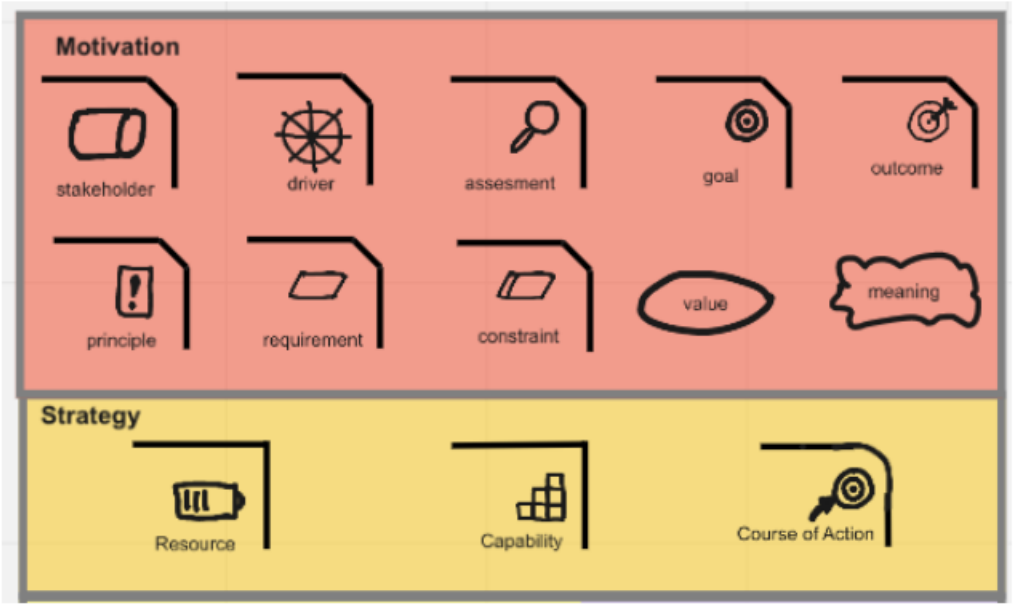
Všetky prvky v základnom rámci (Core Framework)

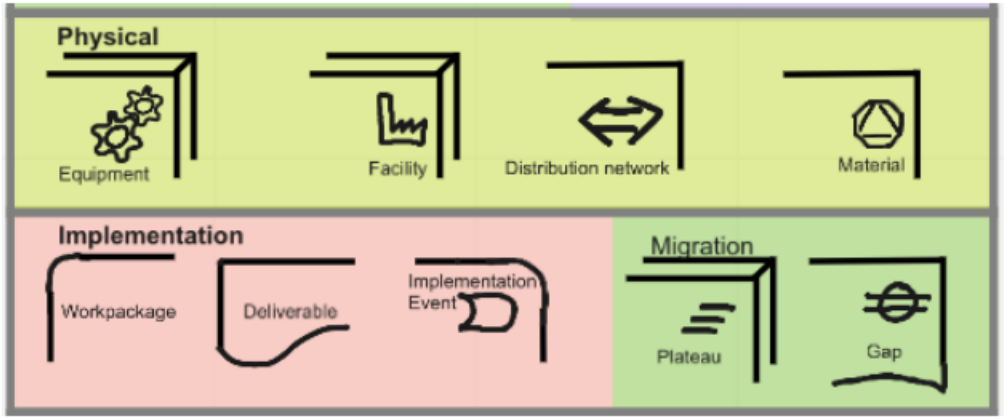




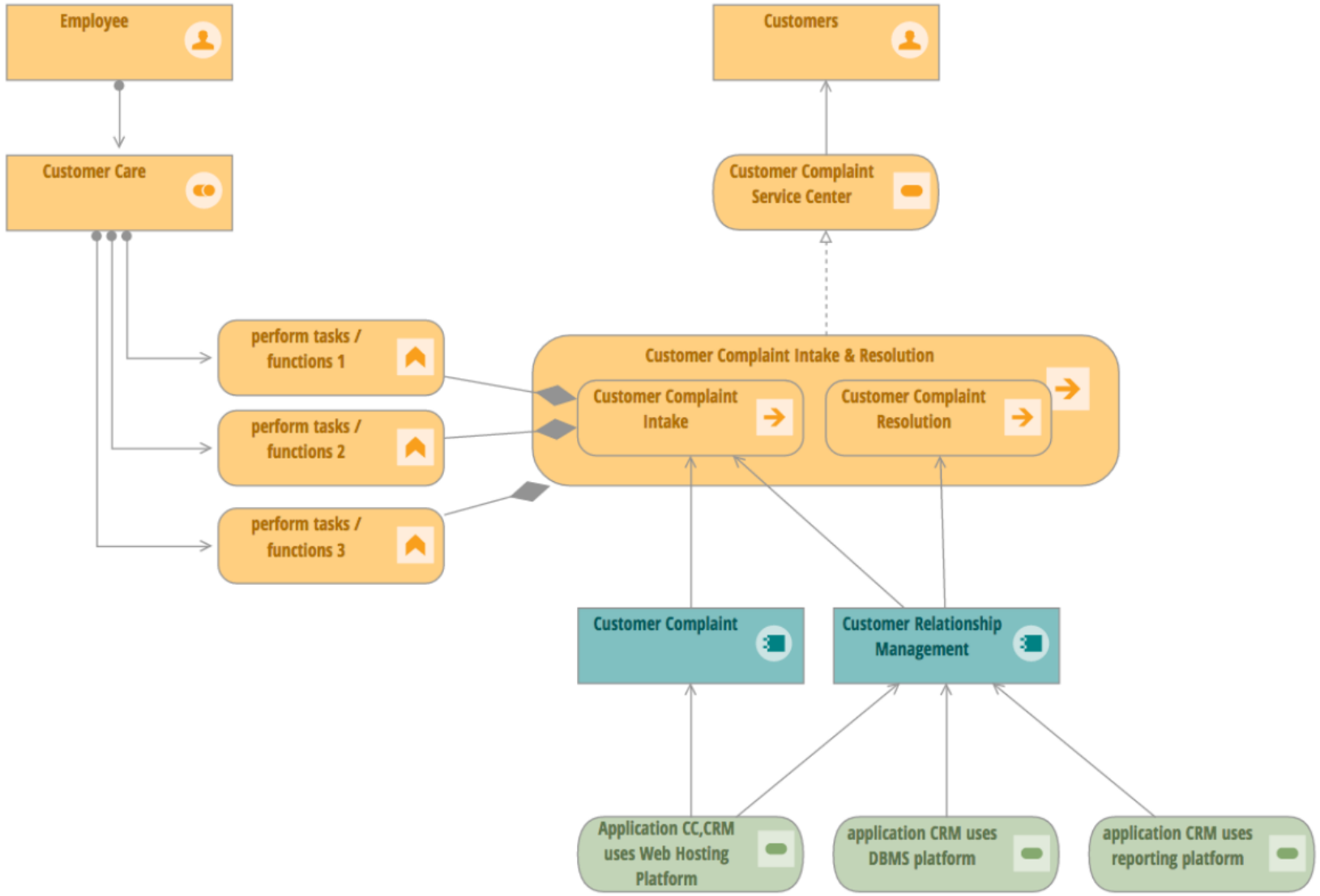
Core layers removed for simplicity







Príklad ArchiMate modelu - Core Layers



Toolkit Tuesday: A Best Practice for Developing a MOSA-based Reference Architecture

TO The Open Group <donotreply@opengroup.org>

THE Open GROUP

Dear Miroslav,

In the next episode of the **Architect's Toolkit Tuesday Broadcast Series,** we are talking about Modular Open Systems Approach (MOSA).

Join us on **Tuesday, February 22, 2022**, at 8:30 am PST, 4:30 pm GMT when Dr. Steven Davidson, Chief Scientist for Systems Architecture Electronic Systems Innovation Center, The MITRE Corporation will be talking about **A Best Practice for Developing a MOSA-based Reference Architecture**.

Čo je MVC architektúra a na čo je dobrá?

Je MVC návrhový vzor alebo architektúra?

Systém

Môže mať viacero úrovní a rozsah:

1. **Funkcia** - implementácia funkcie
2. **Trieda** - implementácia triedy
3. **Projekt** - vzťahy medzi triedami
4. **Riešenie** - vzťahy medzi projektmi
5. **Systém** - vzťahy medzi riešeniami

System software vs. application software

System software	Application software
General-purpose software that manages basic system resources and processes	Software that performs specific tasks to meet user needs
Written in low-level assembly language or machine code	Written in higher-level languages, such as Python and JavaScript
Must meet specific hardware needs; interacts closely with hardware	Does not take hardware into account and doesn't interact directly with hardware
Installed at the same time as the OS, usually by the manufacturer	User or admin installs software when needed
Runs any time the computer is on	User triggers and stops the program
Works in the background and users don't usually access it	Runs in the foreground and users work directly with the software to perform specific tasks
Runs independently	Needs system software to run
Is necessary for the system to function	Isn't needed for the system to function

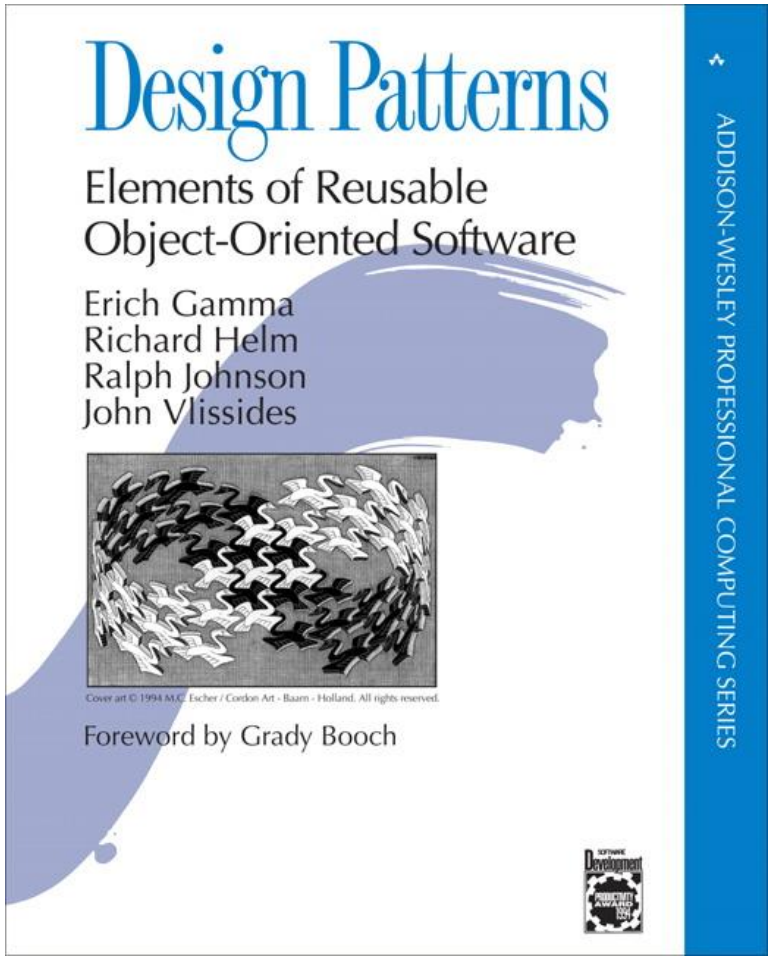
Návrhový vzor (Design pattern)

Je to všeobecne využiteľná predloha pre bežne sa vyskytujúci problém a kontexte dizajnu software

Je to predpis šablóny možného riešenia problému, ktorý je možné aplikovať vo viacerých situáciách

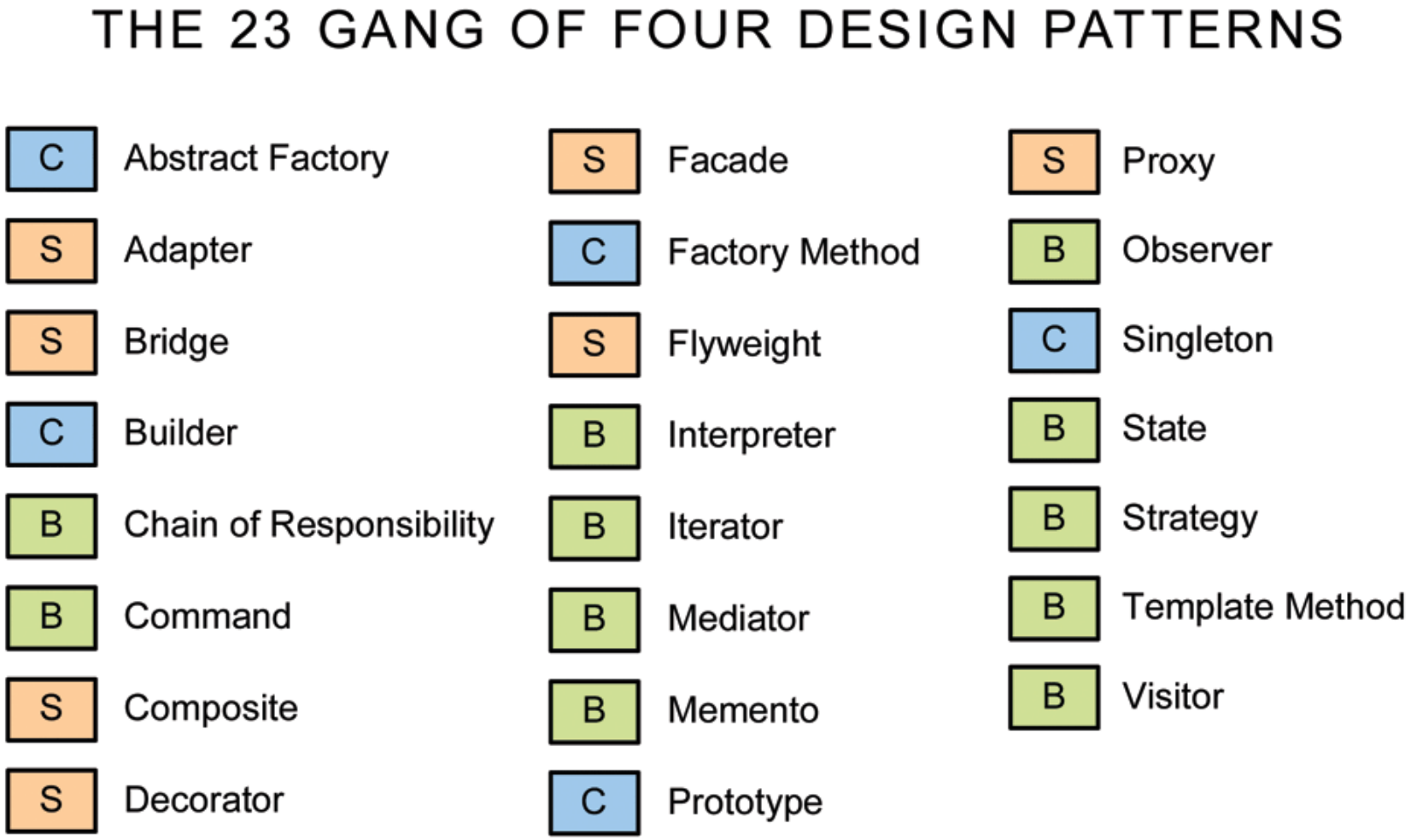
Vzory sú vypracované podľa best practises, aké môže programátor využiť pri riešení toho daného typu problému

Gang of four - GOF



Click here to next software adventure

Koľko a aké sú návrhové vzory?



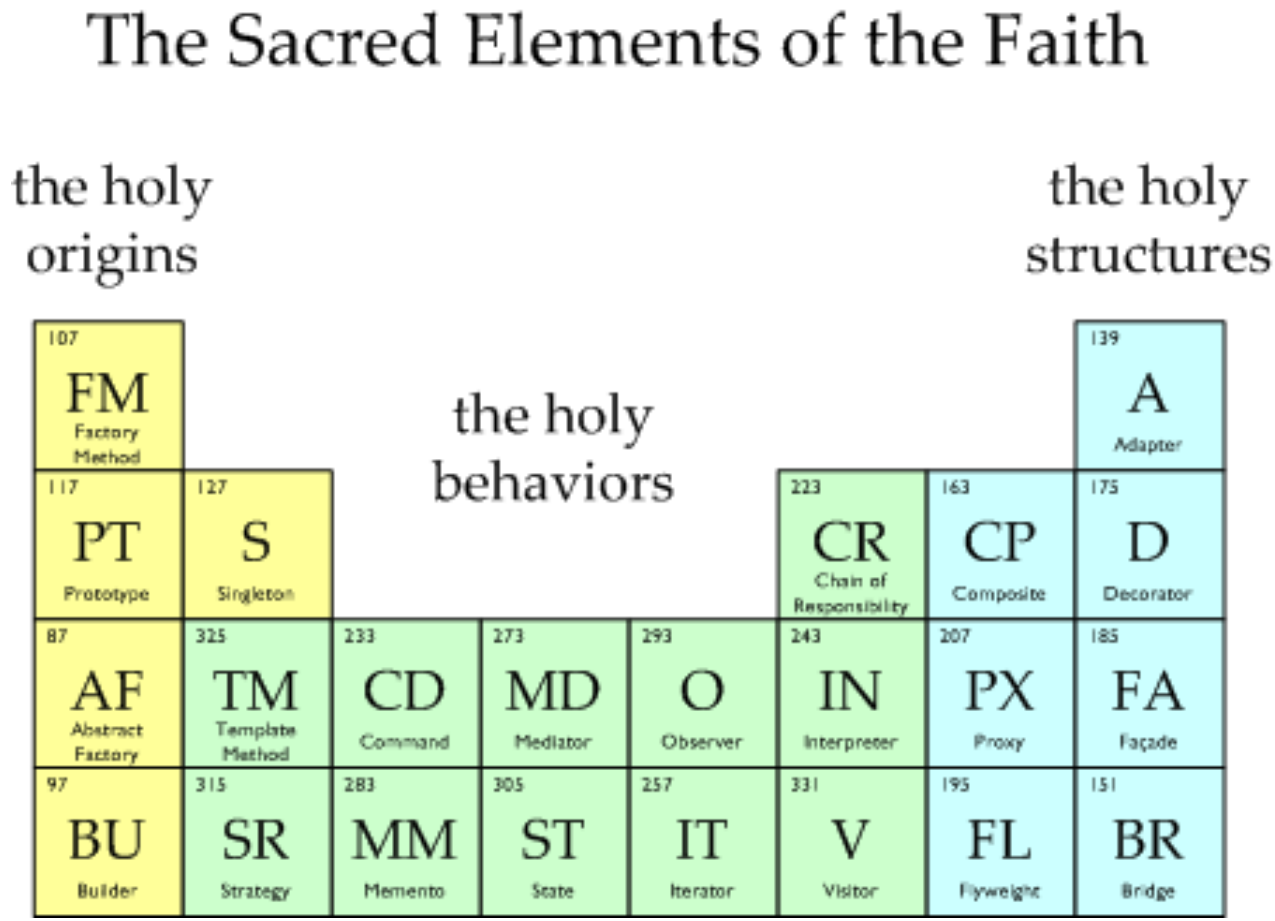
Podporné Vzory (Supporting Patterns)

Jednoduchá továrenská metóda - Simple Factory Method

Prepravka - Crate - Messenger

Prázdny objekt - Null Object

Knižničná trieda - Library class - Utility



C

Jednoduchá továrenská metóda
Simple Factory Method

Definuje statickú metódu nahradzujúcu konštruktor

Používa sa všade tam, kde potrebujeme získať odkaz na objekt

Pričom priame použitie konštruktora nie je moc ideálne

Rozdiel medzi vzorom a architektúrou

Návrhový vzor (Design pattern)

- Slúžia ako šablóny pre časť izolovaného problému
- Pomáha k lepšej štruktúre a udržiavateľnosti
- Zameriavajú na prvé **3 rozsahy**:

1. Funkcia - implementácia funkcie
2. Trieda - implementácia triedy
3. Projekt - vzťahy medzi triedami

Architektúra

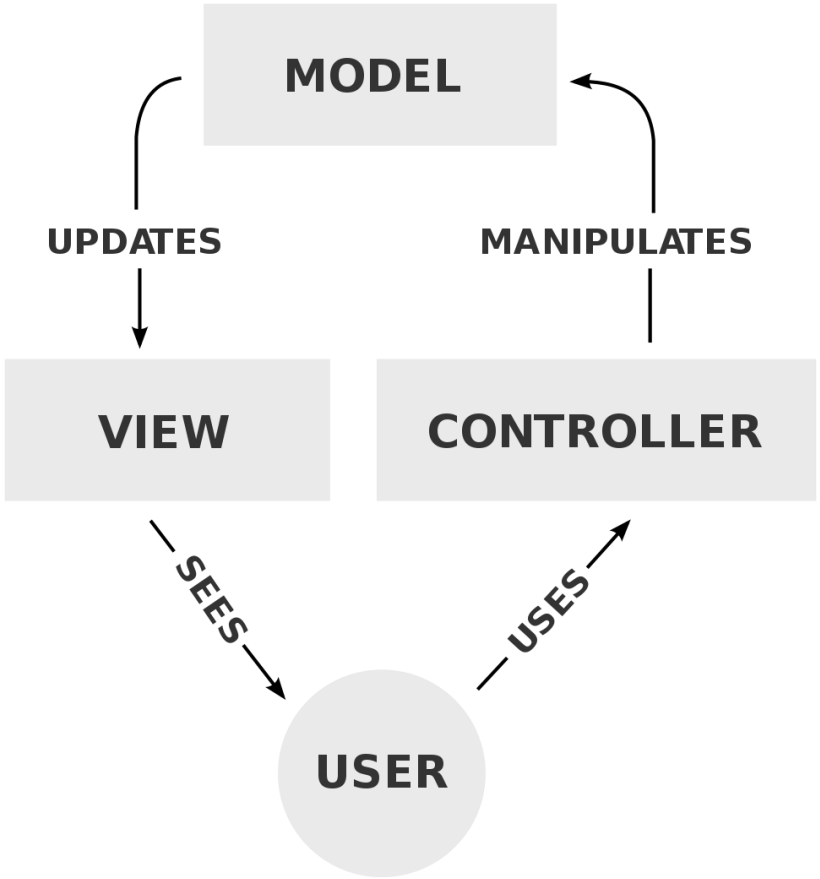
- Určená na spojenie všetkého v fungujúcom systéme
- Najvyššia úroveň návrhu riešenia ako celku
- Väčšinou premýšľate vo veľkom, keďže sa príliš nezachádzate do detailov rôznych komponentov
- Zaisťuje, že rôzne časti riešenia môžu spolu ľahko komunikovať buď priamo alebo cez sprostredkovateľa, týka sa to bezpečnosti, výkonu, nefunkčného nasadenia a systémových požiadaviek.
- Architektúra sa stará o **4 a 5 rozsahov** a niekedy aj 3

Model–view–controller (MVC)

- Je **softvérový návrhový vzor**
- Bežne používaný na **vývoj používateľských rozhraní**, ktoré **rozdeľujú** súvisiacu **programovú logiku** do **3 vzájomne prepojených prvkov**
- Cieľom je **oddeliť interné reprezentácie informácií** od spôsobov, akými sú informácie prezentované používateľovi a akceptované od používateľa
- Tradične používaný pre **desktopové grafické používateľské rozhrania** (GUI) a rovnako aj pri **navrhovaní webových aplikácií**
- Väčšina populárnych programovacích jazykov má dostupné MVC frameworky, ktoré uľahčujú implementáciu vzoru (**Spring** a **Liferay MVC**)

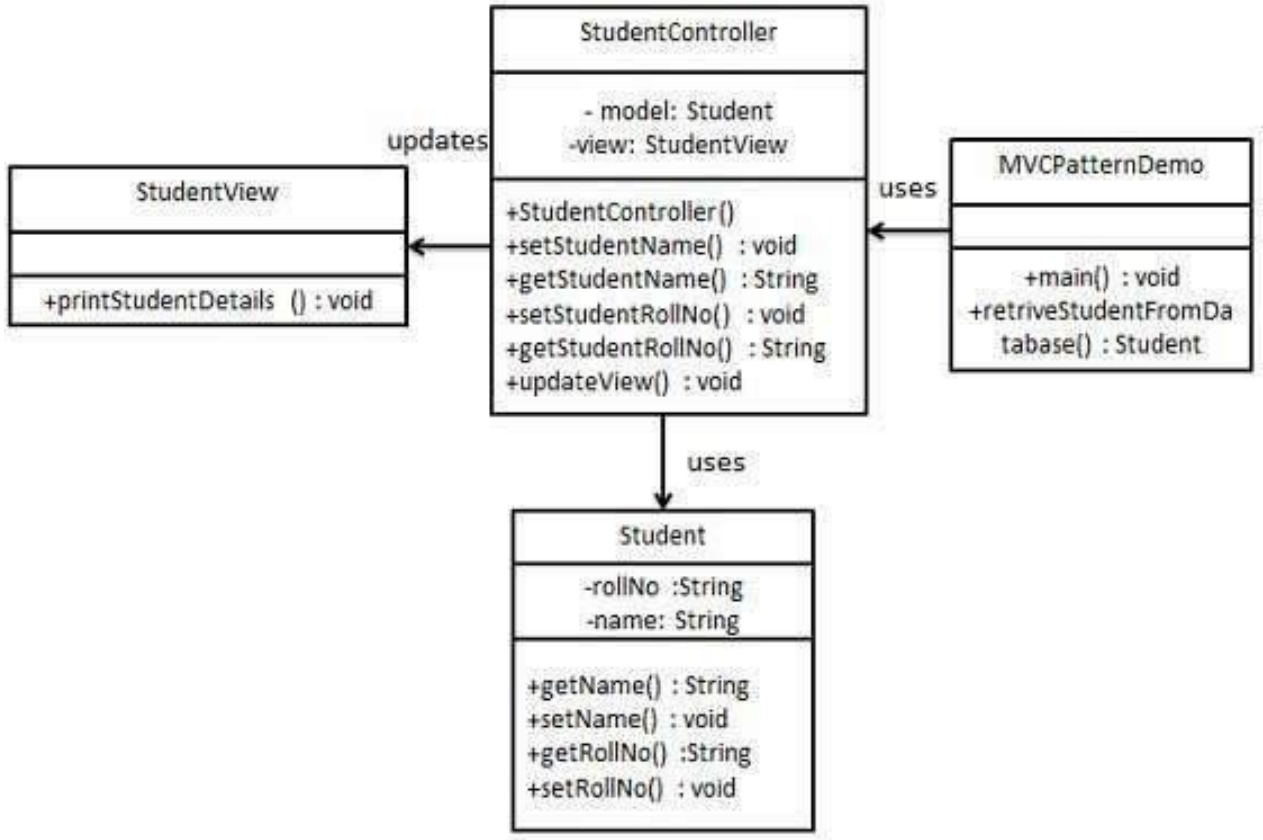
Model–view–controller (MVC)

1. **Model**: Hlavná/centrálna zložka vzoru. Je to dynamická dátová štruktúra aplikácie, **nezávislá od používateľského rozhrania**. Priamo **spravuje dáta, logiku a pravidlá aplikácie**.
2. **View**: **Akékoľvek reprezentácia informácií**, ako je graf, diagram alebo tabuľka. Sú možné viaceré zobrazenia rovnakých informácií, napríklad stĺpcový graf pre manažment a tabuľkové zobrazenie pre účtovníkov.
3. **Controller**: Prijíma **vstup** a **konvertuje** ho na **príkazy** pre **model** alebo **view**.

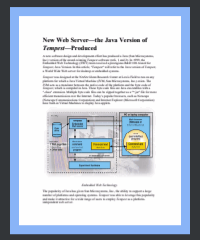


Model–view–controller (MVC)

- **Model** - predstavuje **objekt** alebo **JAVA POJO nesúci dáta**. Môže mať tiež **logiku** na aktualizáciu Controllera, ak sa zmenia jeho údaje.
- **View** – predstavuje **vizualizáciu údajov**, ktoré model obsahuje.
- **Controller** – pôsobí na model aj pohľad. **Riadi tok údajov** do objektu modelu a **aktualizuje zobrazenie** pri každej **zmene údajov**. Udržuje View a model oddelené.



POJO (Plain old Java object)


1

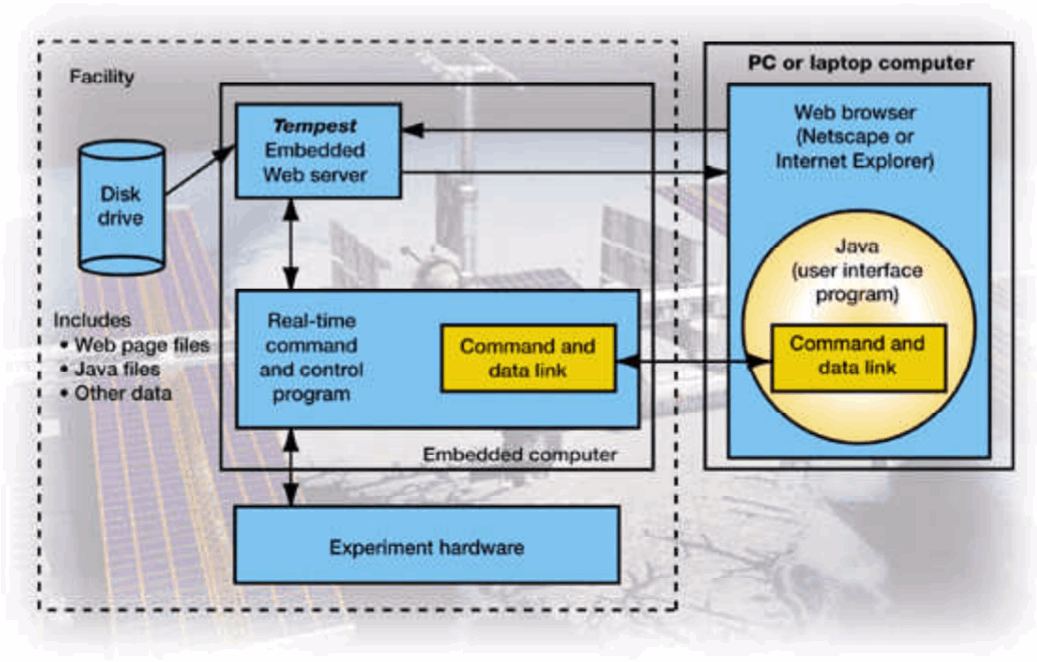

2


3

Tempest—Produced

A new software design and development effort has produced a Java (Sun Microsystems, Inc.) version of the award-winning *Tempest* software (refs. 1 and 2). In 1999, the Embedded Web Technology (EWT) team received a prestigious R&D 100 Award for *Tempest*, Java Version. In this article, "*Tempest*" will refer to the Java version of *Tempest*, a World Wide Web server for desktop or embedded systems.

Tempest was designed at the NASA Glenn Research Center at Lewis Field to run on any platform for which a Java Virtual Machine (JVM, Sun Microsystems, Inc.) exists. The JVM acts as a translator between the native code of the platform and the byte code of *Tempest*, which is compiled in Java. These byte code files are Java executables with a ".class" extension. Multiple byte code files can be zipped together as a "*.jar" file for more efficient transmission over the Internet. Today's popular browsers, such as Netscape (Netscape Communications Corporation) and Internet Explorer (Microsoft Corporation) have built-in Virtual Machines to display Java applets.



Embedded Web Technology.

JavaServer Faces Technology

Developed through the Java Community Process under JSR - 314, JavaServer Faces technology establishes the standard for building server-side user interfaces. With the contributions of the expert group, the JavaServer Faces APIs are being designed so that they can be leveraged by tools that will make web application development even easier. Several respected tools vendors were members of the JSR-314 expert group, which developed the JavaServer Faces 1.0 specification. These vendors are committed to supporting the JavaServer Faces technology in their tools, thus promoting the adoption of the JavaServer Faces technology standard.

Overview

JavaServer Faces technology includes:

- A set of APIs for representing UI components and managing their state, handling events and input validation, defining page navigation, and supporting internationalization and accessibility.
- A JavaServer Pages (JSP) custom tag library for expressing a JavaServer Faces interface within a JSP page.

Designed to be flexible, JavaServer Faces technology leverages existing, standard UI and web-tier concepts without limiting developers to a particular mark-up language, protocol, or client device. The UI component classes included with JavaServer Faces technology encapsulate the component functionality, not the client-specific presentation, thus enabling JavaServer Faces UI components to be rendered to various client devices. By combining the UI component functionality with custom renderers, which define rendering attributes for a specific UI component, developers can construct custom tags to a particular client device. As a convenience, JavaServer Faces technology provides a custom renderer and a JSP custom tag library for rendering to an HTML client, allowing developers of Java Platform, Enterprise Edition (Java EE) applications to use JavaServer Faces technology in their applications.

Ease-of-use being the primary goal, the JavaServer Faces architecture clearly defines a separation between application logic and presentation while making it easy to connect the presentation layer to the application code. This design enables each member of a web application development team to focus on his or her piece of the development process, and it also provides a simple programming model to link the pieces together. For example, web page developers with no programming expertise can use JavaServer Faces UI component tags to link to application code from within a web page without writing any scripts.

Developed through the Java Community Process under JSR - 314, JavaServer Faces technology establishes the standard for building server-side user interfaces. With the contributions of the expert group, the

Welcome to the Apache MyFaces Project

Apache MyFaces is a project of the Apache Software Foundation, and hosts several sub-projects relating to the JavaServer™ Faces (JSF) technology.

Projects

- MyFaces Core | Implementation of the JSF specification
- Apache Tobago | A component library

Inactive Projects (Maintenance Mode)

- MyFaces Commons | Utilities like components, converters, validators
- MyFaces Tomahawk | A component library
- MyFaces Trinidad | A component library (former Oracle ADF-Faces)
- MyFaces Orchestra | Utility library based on Spring
- MyFaces Extensions Validator | Validation framework based on annotations
- MyFaces Extensions CDI | Utility library based on CDI
- MyFaces Extensions Scripting | Adds scripting and rapid prototyping (hot deployment) to JSF
- MyFaces Portlet Bridge | Bridge between Portlets and JSF

Frameworks, JVM Internals, Tools

Java's evolution into 2022: The state of the four big initiatives

Nicolai Parlog | February 19, 2022
Developer Advocate, Oracle

This year will continue the evolution of projects Valhalla, Panama, Loom, and Amber in JDK 18 and JDK 19.

Download a PDF of this article

This will be a significant year for Java. As is well known, Java 18 (as JDK 18) will be generally available on March 22, 2022, and Java 19 will be generally available in September. Looking at the bigger picture of the specific JSRs, this year in Java will continue the evolution of four main initiatives: projects Valhalla, Panama, Loom, and Amber.

In this article, you will see the goals of each of these projects, their current state (as of January 2022), and their short-term plans.

Project Valhalla

Project Valhalla has two goals: an obvious one and a subtle one. The obvious goal is to introduce a new kind of type that "codes like a class, works like an int," as the mission statement says. The subtle goal is to heal the rift in Java's type system that separates reference types from primitive types; this will allow generification over *all* types.

Yes, you'll be able to create an `ArrayList<int>` that's backed by an `int[]`, with no boxing needed.

Valhalla launched in 2014 right after Java 8 was released and has spent much of that time in an exploratory phase during which the people behind the project, led by Brian Goetz, experimented with various proof-of-concept implementations.