# Softwarové architektury

# Lukas Grolig, váš učitel

# Organizační informace

# Přednášky

**2h týdně**, vždy v pondělí v 18–20 hod.

# Cvičení

**C#** od 15.10. Ostatní od 23.9.

# Poděkování cvičícím

Dominik Lašo, Erik Bača, Filip Kaštovský, Josef Krušina, Petr Šlézar

# Co od nás můžete očekávat?

# Discord

Organizační informace, diskuze, pomoc s projekty, hledání týmu, ...

https://discord.gg/JxJCYpFupJ

# Záznamy

Přednášky záznamy přednášek nejsou standardně dostupné.

# Stream

Stream přednášek pouze po předchozím oznámení na YT nebo Twitchi.

# Hodnocení

# Hodnocení předmětu

Maximum bodů: **110** Minimum pro úspěšné zakončení předmětu: **75**

# Týmový projekt

Rozdělený na 3 milestony. 25 bodů za každý milestone. 3-6 členné týmy. Členové mohou být z různých skupin, ale pracují pouze v 1 jazyce.

# Znalostní test

- 12b každý
- dva testy (začátkem listopadu a začátkem prosince)
- open book i výběr odpovědi

# Body na cvičení a za aktivitu

- za aktivitu na přednášce nebo cvičení
- max. 11 bodů
- vyplnění do formuláře

# Očekávané znalosti?

- návrh relační databáze, práce s SQL DDL a DML
- vývoj backendu ve vašem jazyce
- vývoj frotnendu
- znalost REST API

# Co za semestr probereme?

# Probíraná látka

- Úvod - role architekta a architektury
- Úvod do domain driven design
- Event storming
- Strategický design v DDD
- Taktický design v DDD
- Monolitické architektury I a II
- Servisně orientované architekury
- Mikroslužby
- Event-driven architektury
- Event sourcing
- Infrastructure as a Code
- Mikrofrontend

# Studijní materiály

## Literatura

- Fundamentals of Software Architecture by Mark Richards and Neal Ford
- Software Architecture: The Hard Parts by Neal Ford
- Software Architecture in Practice, 4th Edition by Len Bass, Paul Clements and Rick Kazman
- Building Microservices: Designing Fine-Grained Systems by Sam Newman
- Domain-Driven Design: Tackling Complexity in the Heart of Software by Eric Evans
- Learning Domain-Driven Design by Vlad Khononov

# Jak si z kurzu odnést nejvíce

- vytvořte si vlastní zadání projektů pro procvičení
- vyzkoušejte si všechny role v procesu vývoje bez přeskočení
- udělejte business analýzu a definici produktu
- navrhněte vhodný high level design a proveďte rozpad na storky a úkoly
- oceňte implementační čas/čas pro dodání
- vyberte vhodné technologie a procesy
- implementujte software

# Jak si z kurzu odnést nejvíce

- vyberte vhodné testovací strategie a nastavte testování
- nastavte si CI/CD pipeline
- vyberte si jednoho cloud providera kam nasadíte aplikaci (doporučuji si souběžně udělat certifikační zkoušku)
- proveďte nasazení pomocí IaaC
- pomocí testů simulujte provoz
- nastavte logování a trasování

# Course topics

- What is software architecture
- Role of an architect
- TOGAF standard
- Architecture Development Method
- Architecture Repository
- Architecture Decision Record

# Úvod do softwarových architektur

# Softwarová architektura

Some architects refer to software architecture as the blueprint of the system, while others define it as the roadmap for developing a system.

Někteří architekti označují softwarovou architekturu jako plán/projekt systému, zatímco jiní ji definují jako roadmapu pro vývoj systému.

# ISO Definition

ISO/IEC/IEEE 42010:2011 definuje architekturu:

"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."

# TOGAF definice architektury

Kromě ISO/IEC/IEEE 42010:2011, TOGAF do definice přidává druhý význam:

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

# Hierarchické rozdělení architektury

- aplikační architektura
- solution architektura
- enterprise architktura

# Enterprise architektura

# Business Architecture

defines the business strategy, governance, organization, and key business processes

# Data Architecture

describes the structure of an organization's logical and physical data assets and data management resources

# Application Architecture

provides a blueprint for the individual applications to be deployed, their interactions, and their relationships to the core business processes of the organization

# Technology Architecture

describes the logical software and hardware capabilities that are required to support the deployment of business, data, and application services; this includes IT infrastructure, middleware, networks, communications, processing, standards, etc.

# Kdo je softwarový architekt?

# Softwarový architekt

Člověk zodpovědný za architekturu.

# Ja se stát architektem

Vývojář, který se nezajímá jen o svůj malý kousek systému, ale o celý vývojový cyklus proces.

Přemýšlí na business hodnotou, cenou featur a jakou hodnotu přinesou společnosti.

# Očekávání od architekta

- Dělá architektonická rozhodnutí
- Kontinuálně analyzuje architekturu
- Udržuje své znalosti podle nejnovějších trendů
- Hlídá dodržování rozhodnutí
- Má znalost business domény
- Pracuje na svých softskills a na práci s lidmi
- Umí se pohybovat ve firemní politice

ozhodování o architektuře

Od architekta se očekává, že bude definovat rozhodnutí o architektuře a zásady návrhu, kterými se budou řídit technologická rozhodnutí v rámci týmu, oddělení nebo celého podniku.

# Architektonická rozhodnutí

Architektonická rozhodnutí jsou soubor pravidel, jak by měl být systém implementován, výběr technologií, doporučení a stanovení nefunkcionálních požadavků.

# Architektonická rozhodnutí

Rozhodnutí dělá jeden architekt, tým architektů nebo board archiktrů. Ve velkých společnostech existuje formální schvalovací proces pro zásadní rozhodnutí.

# Architecture Repository

Centrální úložiště důležitých dokumentů kolem architektury

# Architecture Repository

Měl by to být dobře organizovaný souborový server, verzovací systém, jako je Git, nebo knowledgebase tool jako je Confluence.

Vyberte si svůj vlastní jed.

# Architecture decision record

Každé rozhodnutí musí být zdokumentováno. To se obvykle provádí formou Architecture decision record (ADR).

Každý ADR popisuje architektonické rozhodnutí, jeho kontext a důsledky. ADR mají stavy a životní cyklus.

# The decision log

Výstupem procesu ADR je kolekce záznamů o architektonickém rozhodnutí. Této kolekci se říká decision log. Decision log by měl poskytnout kontext projektu a podrobné informace o implementaci a návrhu. Členové projektového týmu procházejí jednotlivé ADR, aby získali přehled o projektu a důvodech, které k rozhodnutí vedly.

# Scope ADR

Když tým zavádí proces vytváření ADR, tak jednotlivé ADR by měly vytvořeny na základe společné šablony.

Šablona zjednodušuje tvorbu ADR a zajišťuje, že ADR zachycuje všechny relevantní informace. Každé ADR by mělo definovat minimálně kontext rozhodnutí (proč to potřebujeme), rozhodnutí samotné (+ zdůvodnění volby), důsledky rozhodnutí pro projekt a výstupy.

# Úložiště ADR

- ADR ukládejte na centrálním místě
- Uchovávejte historii ADR
- Určete vlastníky jednotlivých ADR

Běžně používané systémy pro ukládání ADR jsou Git, Confluence, Google Docs.

# Vzorové ADR

Title: The software development lifecycle approach for ABC application development.
Status: Accepted Date: 2022-03-11

# Vzorové ADR

Context

ABC application is a packaged solution, which will be deployed to the customer's environment by using a deployment package. We need to have a development process that will enable us to have a controllable feature, hotfix, and release pipeline.

# Vzorové ADR

Decision

We use an adapted version of the GitFlow workflow to develop ABC application.

For simplicity, we will not be using the hotfix/* and release/* branches, because ABC application will be packaged instead of being deployed to a specific environment. For this reason, there is no need for additional complexity that might prevent us from reacting quickly to fix bugs in production releases, or testing releases in a separate environment.

The following is the agreed branching strategy:

Each repository must have a protected main branch that will be used to tag releases.

Each repository must have a protected develop branch for all ongoing development work.

# Vzorové ADR

Consequences

Positive:

Adapted GitFlow process will enable us to control release versioning of the ABC application.

Negative:

GitFlow is more complicated than trunk-based development or GitHub flow and has more overhead.

Compliance

The main and develop branches in each repository must be marked as Protected.

Changes to the main and develop branches must be propagated by using merge requests.

At least one approval is required for every merge request.

# Průběžná analýza architektury

Od architekta se očekává, že bude průběžně analyzovat architekturu a aktuálně používané technologie a prostředí, kde jsou nasazené. Na základě analýz bude navrhovat doporučení pro zlepšení.

# Držet krok s nejnovějšími trendy

Od architekta se čeká, že se bude držet v obraze s aktuálními trendy a technologiemi.

# Jak držet krok se nejnovějšími trendy

TechRadar od Thoughtworks

Newslettery:

- SWLW
- C# Digest, JVM Weekly, Javascript Weekly, React Status, High Scalability

# Zajištění dodržování rozhodnutí

Od architekta se očekává, že zajistí doržování architektonických rozhodnutí a zásad správného návrhu a vývoje.

# Zajištění dodržování rozhodnutí

- komunikace s týmem
- stále připomínání principů a rozhodnutí
- technický dozor a review práce
- audit dříve provedené práce

# Různorodost znalostí a zkušeností

Od achitekta se očekává, že se vyzkoušel různé technologie, frameworky, platformy, a prostředí.

# Nutnost získat business domain knowledge

Další z očekávání od architekta je získání dostatečné úrovně business expertízy.

Nejúspěšnější architekti jsou ti, kteří mají široké praktické technické znalosti kombinované s dobrou znalostí dané domény.

# Meziliská komunikace

Od architekta se očekává velmi dobrá schopnost mezilidské komunikace, týmové práce, exekutivy a vedení.

# Pochopení a proplouvání firemní politiky

Od architekta se očekává, že bude rozumět politickému prostředcí společnosti a bude schopen se v něm pohybovat.

Téměř každé rozhodnutí architekta bude zpochybňováno. Např. rozhodnutí architekta nebudou zpochybňovat jen vývojáři, kteří mají pocit, že jejich přístup je lepší.

# Diagramovaní a prezentace architektury

- Efektivní komunikace je kritický dovednost architekta a podporuje jeho úspěch.
- Diagramování a prezentace výsledků jsou dvě životně (nebo spíše kariérně) nejdůležitější dovednosti
- Není důležité ukázat a říct vše. Jde o to podstatné.

# Diagramming

- UML
- C4
- ArchiMate

# C4 Context

Reprezentuje celý kontext systému, včetně rolí uživatelů a externích závislostí.

# C4 Container

Fyzické (a i logické) hranice, nasazení a kontejnery v rámci architektury. Tento pohled je vhodnou úrovní detailu pro architekty, ops a vývojový tým.

# Component

Komponentní pohled na systém; ten by měl nejvíce odpovídat pohledu architekta na systém.

# Class

C4 používá stejný styl diagramů tříd jako UML, které jsou efektivní, takže není třeba je nějak nahrazovat.

# Diagramming guidelines

- Make sure all the elements of the diagram have titles or are well known to the audience
- Use rotation and other effects to make titles “sticky” to the thing they associate with and to make efficient use of space.

# Diagramming guidelines

- Čáry by měly být dostatečně silné, aby byly dobře vidět.
- Pokud čáry označují tok informací, použijte šipky pro označení směru nebo obousměrného toku.

# Diagramming guidelines

- vytvořte si vlastní databázi tvarů a log
- používejte trojrozměrné boxy k označení nasaditelných artefaktů
- používejte obdélníky k označení kontejnerů

# Diagramming guidelines

- olabelujte každou věc v diagramu
- architekti často upřednostňují jednobarevné provedení
- používejte barvy, pokud pomáhají odlišit jeden artefakt od druhého

# Prezentace

# Prezentace

Presentation Zen by Garr Reynolds Slide:ology: The Art And Science Of Creating by Nancy Duarte

# Infodeck

- Typ prezentace, který není určený k promítání.
- Obsahuje mnohem více textu.
- Jsou samostatné a jsou určeny k zaslání e-mailem a přečtení.
- Nevyžaduje animace.

# TOGAF

# What is TOGAF

The TOGAF® Standard, a standard of The Open Group, is a proven Enterprise Architecture methodology and framework used by the world's leading organizations to improve business efficiency.

# TOGAF

The TOGAF Standard is used by small, medium, and large commercial businesses, as well as government departments, non-government public organizations, and defense agencies

# Division of the Enterprise architecture

# Business Architecture

defines the business strategy, governance, organization, and key business processes

# Data Architecture

describes the structure of an organization's logical and physical data assets and data management resources

# Application Architecture

provides a blueprint for the individual applications to be deployed, their interactions, and their relationships to the core business processes of the organization

# Technology Architecture

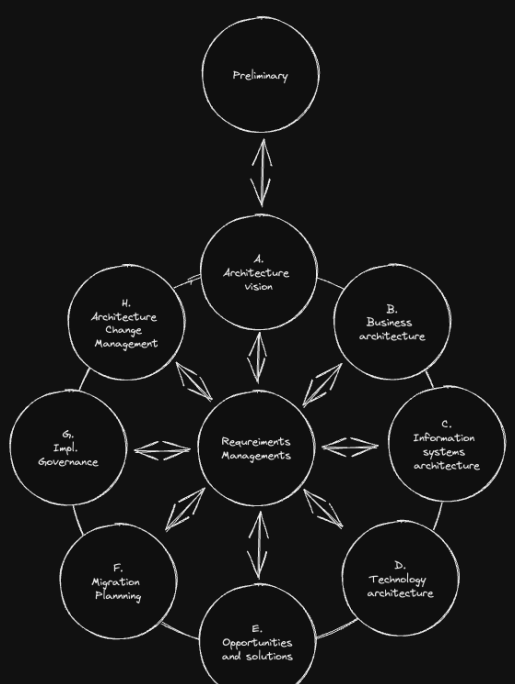
describes the logical software and hardware capabilities that are required to support the deployment of business, data, and application services; this includes IT infrastructure, middleware, networks, communications, processing, standards, etc.

# Architecture Development Method

- ADM provides a tested and repeatable process for developing architectures
- The ADM includes establishing an architecture framework, developing architecture content, transitioning, and governing the realization of architectures.

Each phase start after previous is finished. Phase can be paused to go back and revise a past phase ADM can be adapted to your specific needs

# ADM cycle


Preliminary
A.
Architecture
vision
B.
Business
architecture
C.
Information
systems
architecture
D.
Technology
architecture
E.
Opportunities
and solutions
F.
Migration
Planning
G.
Impl.
Governance
H.
Architecture
Change
Management
Requreiments
Managements

# That's it.

# See yout next time.