Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

DIPLOMOVÁ PRÁCE

2015 Petr ŽIŽKA

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Studijní obor: Podniková informatika

Integrace SaaS do on premise prostředí IS podniku

Autor diplomové práce : Bc. Petr Žižka

Vedoucí diplomové práce : Ing. Libor Gála, Ph.D.

Oponent diplomové práce : doc. Ing. Alena Buchalcevová, Ph.D.

Rok odevzdání: 2015

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze kterých jsem čerpal.

V Praze dne 8. prosince 2015

…….
Petr Žižka

Poděkování

Tímto bych rád poděkoval Ing. Liboru Gálovi za podnětné připomínky, odborné vedení mé diplomové práce a cenné rady. Děkuji za ochotu, trpělivost a čas, jenž mi věnoval.

Dále bych rád poděkoval své manželce a dceři za podporu během celého studia.

Věnování

Tátovi.

Abstrakt

Tato diplomová práce se zabývá integrací aplikací poskytovaných formou služeb (angl. Software as a Service, dále SaaS) do stávajícího on-premise podnikového informačního systému. Služby typu SaaS jsou v rámci podnikové informatiky (angl. Enterprise Computing) začleněny do většího celku tzv. cloudových služeb (angl. Cloud Services), které zažívají v posledních letech velice dynamický růst a pronikají v různých formách do podniků všech velikostí a oblastí podnikání.

Avšak nasazení SaaS v podnikové praxi naráží na specifické problémy s jejich integrací do podnikových informačních systémů. Dodavatelé SaaS se v posledních letech soustředili výhradně na vylepšování vlastní funkcionality a samotná integrace jejich služeb zůstala na okraji jejich zájmu.
Z tohoto důvodu je integrace SaaS stále pro mnoho společností komplikovaná a prakticky nedostupná. Obzvláště skupina malých a středních společností je v ohledu využití informačních technologií specifická a pro tuto skupinu jsou tradiční integrační prostředky nedostupné.

Cílem práce je proto *návrh metodického vzoru MeFIS pro integraci SaaS do on-premise prostředí podnikového informačního systému malých a středních společností*.

Nejprve provedu vymezení terminologie používané v oblasti integrace IT a cloudových služeb a vymezím základní technologické prostředky používané k integraci v IT. Na základě tohoto vymezení provedu rešerši dostupných informačních zdrojů s cílem nalézt vhodnou metodiku pro integraci SaaS, která by zohledovala specifické potřeby malých a středních společností. Na základě dostupných metodik provedu návrh metodického vzoru integrace SaaS, který bude vycházet z metodického rámce MeFIS.

V závěrečné fázi provedu aplikaci navrženého metodického vzoru na konkrétní integrační projekt ve skutečné české společnosti. Tato část si klade za cíl popsat možná řešení integračních problémů za použití navrženého metodického vzoru a také může sloužit jako nástin konkrétní integrace SaaS do podnikových informačních systémů malých a středních firem.

Klíčová slova

Software as a Service, Podnikový informační systém, iPaaS, Malé a střední společnosti, MeFIS, Integrace SaaS

Abstract

This Master's Thesis focuses on integration of software that are provided as services (SaaS) into current on-premise information systems of enterprises. In enterprise computing, services such as SaaS are usually integrated into cloud services. In recent years, cloud services have been experiencing a dynamic growth, and, in various forms, they have been entering enterprises of different sizes and fields.

At the same time, the experience show that implementation of SaaS into enterprise computing faces many challenges due to the fact that the SaaS vendors have exclusively focused on improvement of software functionality and the integration itself has completely remained out of their attention. Given that, the integration of SaaS is rather complicated or even inaccessible for many enterprises. In particular, small and medium-sized businesses are a very specific target groups and the traditional integration platforms are unaffordable for them.

The goal of this thesis is to develop a methodology based on MeFIS that would cover integration of SaaS into on-premise-based information systems in small and medium enterprises.

In this thesis, first, I will define the terms used in integration of IT and cloud services and describe the essential principles, tools, and technologies used. Second, I will conduct a literature review of available sources to find the best methodology for integration of SaaS, which would take into account the specific needs of small and medium enterprises. Third, I will develop a methodology based on the methodology framework MeFIS (Methodology Framework for IS/ICT Systems).

At the final stage, the developed methodology will be applied in one company in the Czech Republic. The emerged integration challenges will be described and their solutions will be developed; thus, this thesis might also serve as a specific example of SaaS integration into enterprise information systems within small and medium-sized businesses.

Key Words

Software as a Service, Enterprise Information System, iPaaS, Small and Medium Enterprises, MeFIS, SaaS Integration

Obsah

1 Úvod ... 1

2 Terminologické a tematické vymezení ... 5

2.1 Informační systém podniku ... 5

2.2 Specifika IS v malých a středních podnicích ... 6

2.3 Integrace v IT ... 9

2.4 Metoda versus metodika ... 11

2.5 Služba ... 12

2.6 Cloudové služby ... 14

2.6.1 Technologické vlastnosti cloudových služeb ... 15

2.6.2 Bezpečnost SaaS služeb ... 16

2.6.3 Integrace aplikací SaaS ... 17

3 Východiska integrace a její stavební prvky ... 18

3.1 Architektura IT ... 18

3.1.1 Vícevrstvá architektura ... 18

3.1.2 Shrnutí ... 22

3.2 Middleware ... 23

3.2.1 Základní middleware ... 25

3.2.2 Integrační middleware ... 26

3.2.3 Integrační platforma jako služba (iPaaS) ... 27

3.3 Metody integrace a způsoby komunikace ... 27

3.3.1 Způsoby komunikace ... 28

3.3.2 Metody integrace ... 32

3.3.3 Shrnutí ... 37

3.4 Integrace podnikových aplikací ... 38

4 Metodika integrace služeb SaaS ... 43

4.1 Rešerše metodických zdrojů ... 44

4.2 Metodický rámec MeFIS ... 47

4.3 Metodický vzor pro integraci služeb SaaS ... 52
4.3.1 Fáze ... 54
4.3.2 Role ... 62
4.3.3 Procesy ... 63
4.3.4 Principy metodického vzoru ... 81
4.3.5 Návrhové vzory ... 82
5 Praktická aplikace navrženého metodického vzoru ... 84
5.1 Stručná charakteristika vybraného podniku a výchozí situace ... 85
5.2 Informační systém iTravel ... 87
5.3 Aplikace navrženého metodického vzoru ... 88
5.3.1 Aktualizace podnikové politiky správy a řízení IT ... 88
5.3.2 Specifikace požadavků a rozsahu integračního projektu ... 89
5.3.3 Výběr služby SaaS ... 89
5.3.4 Volba metody integrace ... 89
5.3.5 Volba integračního scénáře ... 91
5.3.6 Analýza možných omezení využití služby SaaS ... 94
5.3.7 Detailní návrh jednotlivých komponent ... 95
5.3.8 Získání funkčního prototypu ... 95
5.3.9 Implementace řešení, testování řešení ... 96
5.3.10 Nasazení řešení ... 99
5.3.11 Administrace řešení ... 99
5.4 Shrnutí navrženého integračního řešení ... 100
6 Závěr ... 102
7 Terminologický slovník ... 103
8 Použité zdroje ... 106
9 Seznam obrázků a tabulek ... 112
9.1 Seznam obrázků ... 112
9.2 Seznam tabulek ... 113

10 Přílohy 115

10.1 Ukázka technické dokumentace ke službě iTravel 115

10.2 Ukázka vyexportovaných ceníků systémem iTravel 120

10.3 Ukázka základního prostředí navržené aplikace 123

10.4 Ukázka ceníku připraveného navrženou aplikací 124

1 Úvod

Tato diplomová práce se zabývá začleněním (integrací) různých softwarových komponent poskytovaných formou služeb (angl. Software as a Service, dále pouze SaaS) do stávajícího on-premise prostředí podnikového informačního systému. Software typu SaaS je zpravidla poskytován a spravován třetí stranou (tzv. dodavatelem cloudových služeb) a je poskytován konzumující straně (klientům) na základě SLA (Service Level Agreement, tj. smlouva o poskytování IT služeb).

Služby typu SaaS jsou v rámci podnikové informatiky (angl. Enterprise Computing) začleněny do většího celku tzv. cloudových služeb (angl. Cloud Services), které zažívají v posledních letech velice dynamický růst a pronikají v různých formách do podniků všech velikostí a oblastí podnikání. Dle studie IDC (2015) došlo k relativnímu nárůstu využití cloudových služeb v rámci celkové podnikové IT infrastruktury v roce 2014 o 27 procent oproti roku 2013 (viz Tabulka 1). To znamená, že na konci roku 2014 dosáhl celkový podíl cloudových služeb 17% podílu na celém koláči nákupu nových komponent (HW/SW) IT infrastruktury. Z tabulky je zřejmé, že zatímco tradiční IT komponenty v absolutní hodnotě stagnují, u cloudových služeb sledujeme výrazný růst.

Tabulka 1: EMEA Cloud IT Infrastructure Value, Calendar Year 2014 (Value in $ billions) (IDC, 2015)

Deployment Model ($B)	2014	2014 Segment Shares	2013	2013 Segment Shares	2014 YoY Growth
Private Cloud (on- and off-premises)	$ 2.4	10%	$ 1.9	8%	27.0%
Public Cloud	$ 1.7	7%	$ 1.3	6%	29.1%
Traditional IT	$ 19.8	83%	$ 19.8	86%	0.0%
Grand Total	**$ 23.9**	**100%**	**$ 23.0**	**100%**	**3.8%**

Na druhou stranu, i přes znatelnou dynamiku rozvoje cloudových služeb, tvoří tyto služby stále pouze zlomek tradičních IT řešení. Jinými slovy, růstový potenciál nebyl ještě zdaleka dosažen. Dynamický růst cloudových služeb indikuje také výzkum provedený společností Northstar na vzorku 700 kanadských organizací (Lovering, 2015), z nichž přes 52% deklaruje ochotu používat v nějaké formě cloudové služby v následujícím roce 2016.

Dalšímu rozvoji brání především vyšší složitost začlenění podobných služeb do tradičních on-premise řešení podnikových informačních systémů. Jak ostatně podtrhuje zástupce společnosti Salesforce.com[1] (Falotti, 2007, s. 283; volně přeloženo autorem): *„Dalším náročným úkolem v oblasti podnikových*

[1] Společnost Salesforce.com patří mezi nejznámější poskytovatele CRM řešení poskytovaného formou SaaS. V roce 2014 činily její tržby 4mld USD (v roce 2013, 3mld USD). (zdroj: en.wikipedia.org/wiki/Salesforce.com)

aplikací (SaaS) je integrace s existujícími databázemi či aplikacemi. Každá organizace potřebuje zajistit, aby software, který kupuje, spolupracoval s ostatními komponentami informačních systémů.“. Podobné závěry potvrzují zaměstnanci společnosti Progress Software (Štumpf, 2007, s. 288) na základě výzkumu provedeného společností Forrester v roce 2006: *„…nástroj umožňující takovou datovou integraci (je) nejpreferovanějším nástrojem pro sdílení dat a jejich integraci s podnikovými službami.“*

Jak ostatně podotýká Benoit Lheureux, viceprezident výzkumné společnosti Gartner (Stafford, 2015; volně přeloženo autorem), situace se za posledních osm let příliš nezměnila: *„Integrace aplikací se ztratila někde v honbě za cloudovými službami…“*, *„Mnoho společností překotně nasazovalo cloudové služby avšak zcela podcenily jejich integraci do stávajících informačních systémů…“*, *„Skutečnost, že poskytovatelé SaaS nezahrnují do svých služeb také integrační řešení, způsobila překvapení pro mnoho časných uživatelů (angl. Early Adopters)“*. Lheureux si také všímá skutečnosti, že dodavatelé SaaS se v posledních letech soustředili výhradně na vylepšování své vlastní funkcionality přístupné skrze API a předpokládali, že tím mají v otázce integrace splněno[2]. Tato skutečnost umožnila vznik celé řadě integračních platforem třetích stran (MuleSoft, Informatica, CloudElements atd.).

Na druhou stranu Lheureux věří, že v budoucnu významní poskytovatelé SaaS zahrnou integrační služby do svých produktů, ale nestane se tak v dohledné době. Proto Lheureux (Stafford, 2015; volně přeloženo autorem) v neposlední řadě radí, aby společnosti před pořízením SaaS provedly informované rozhodnutí z hlediska jejich možné integrace do stávajících informačních systémů.

A právě ze zmíněných důvodů jsem se rozhodl pro téma své diplomové práce. Využití SaaS nabízí v současné podnikové informatice značný potenciál, ale zároveň představuje také integrační výzvy, které bych rád v rámci této práce pojmenoval a navrhnul jejich řešení.

Cílem práce je proto *návrh metodického vzoru MeFIS pro integraci SaaS do on-premise prostředí podnikového informačního systému malých a středních společností*.

Prvním krokem k tomuto cíli bude nejprve vymezení terminologie používané v oblasti integrace IT a cloudových služeb (SaaS). Jak poznamenává Singletary (2003) a Feuerlicht (2008), integrace v IT prošla za posledních 30 let bouřlivým vývojem ruku v ruce s vývojem výpočetního modelu obecně. Od 70. let minulého století, kdy převažoval centralizovaný výpočetní model, přes decentralizovaný výpočetní model, až po zcela distribuovaný výpočetní model. Z těchto důvodů je *oblast integrace IT* značně široká a výzkum, který provedl Singletary (2003), prokázal, že jeho vymezení je velice různorodé a problematické i pro zkušené IT profesionály v oboru.

[2] Lheureux (Stafford, 2015) používá spojení, že integrace se provede (angl. Automagically) což dokresluje míru podcenění situace ze strany dodavatelů SaaS

Z provedeného výzkumu (Singletary, 2003) vyplývá ještě jedna podstatná skutečnost. Navržená metodika by měla respektovat různé zainteresované skupiny (angl. Stakeholders) uvnitř podniku, které akcentují jiný pohled na integraci IT. Každá z těchto skupin (uživatelé, IT profesionálové, manažeři) vnímají integraci jiným způsobem, ale každá ze skupin hodnotí ve výsledku její přínosy pro dosažení cílů organizace. Proto i navržená metodika toto musí zohledňovat a nemůže cílit pouze na jednu konkrétní skupinu.

Podobně je na tom také *oblast SaaS*. Služby SaaS nemají za sebou tak dlouhou historii jako oblast integrace IT, ale jejich překotný vývoj posledních let zanesl do vnímání SaaS mnoho různých pohledů. Abych různým pohledům předešel, nejprve terminologicky vymezím všechny uvažované oblasti a technologie. V rámci tohoto vymezení již pojmenuji některé klíčové vlastnosti SaaS modelu, které mohou následně významně ovlivnit hledání možného přístupu k integraci.

V rámci tohoto vymezení nastíním také základní technologické prostředky používané k integraci v IT, které budou sloužit jako vodítko pro následné pochopení různých integračních metod. Věřím, že tyto teoretické základy a pochopení celkové podstaty integrace v IT pomohou primární cílové skupině této práce (viz dále) k pochopení výhod a osvětlení některých principů navržené metodiky.

Na základě terminologického vymezení (vím, co hledám) provedu *v následném kroku* rešerši dostupných informačních zdrojů. Popíši dostupné metodiky zabývající se integrací SaaS do informačních systémů (vím, co hodnotím). V případě, že bude existovat více možných přístupů k integraci SaaS, provedu jejich srovnání a na základě stanovené metriky vyberu nejvhodnější metodu pro splnění cílů této práce. Výstupem tohoto kroku bude *navržená metodika integrace SaaS do informačního systému podniku (dále IS).*

V závěrečném kroku uskutečním reálnou aplikaci navržené metodiky prostřednictvím skutečné integrační úlohy: *integrace služby iTravel společnosti Lemax do informačního systému české společnosti.* V rámci tohoto kroku provedu aplikaci zvolené metodiky a tím ověřím její životaschopnost. Na základě této praktické úlohy metodiku kriticky zhodnotím a případně upravím.

Primárním výstupem mé práce bude *popsaná metodika integrace SaaS* snadno uchopitelná pro podniky libovolné velikosti, avšak mým primárním cílem jsou *malé a střední podniky (dále MSP)*. Existuje několik důvodů, které mě vedou k tomuto zaměření:

- Jak naznačuje analýza Hospodářské komory ČR a Českého statistického úřadu (HK ČR, 2005), malé a střední podniky jsou páteří české ekonomiky a mají v ní svou nezastupitelnou úlohu: *„Český statistický úřad podrobil analýze desetiletý vývoj malých a středních podniků v České republice. Sledoval jejich význam pro tuzemskou ekonomiku v letech 1995 až 2005. Podle zjištění statistiků tvoří nejpočetnější skupiny podniky do deseti zaměstnanců, nejméně*

početná je pak skupina středních firem s více než padesáti zaměstnanci. Drobné podniky (do 10 zaměstnanců) zaměstnávají zhruba tolik lidí, jako podniky malé (od 10 do 49 zaměstnanců) a střední (od 50 do 249 zaměstnanců) dohromady.“ Více informací o této cílové skupině rozvádím v kapitole 2.2;

- Paradoxně nejsou primárně malé a střední podniky v hledáčku dodavatelů tradičních IT řešení, kteří se soustřeďují spíše na velké korporace. Velké korporace jsou totiž obvykle schopny lépe alokovat zdroje na vyšší vstupní náklady spojené s nákupem tradičních on-premise IT řešení. Z toho vyplývá, že MSP mají k tradičním IT technologiím přístup mnohem později. A právě proto MSP mohou ze zapojení SaaS do on-premise IT infrastruktury těžit nejvíce. Vstupní náklady SaaS jsou zpravidla nižší a nekladou takové nároky na jejich správu a údržbu;
- V neposlední řadě, velké korporace mají obvykle značně rozsáhlé monolitické podnikové informační systémy, jejichž integrace zpravidla vyžaduje zcela unikátní přístup profesionálního systémového integrátora a byl by zcela jistě mimo rozsah této práce.

Tuto práci považuji také za příhodné pokračování mé bakalářské práce (Žižka, 2009) zaměřené na praktickou aplikaci webových služeb jako prostředek datové integrace. Zmíněná práce akcentovala jednu konkrétní implementaci a použitou technologii (SOAP) bez širšího teoretického přístupu k integraci v IT a tento deficit bych rád „v tomto pokračování“ vykompenzoval.

2 Terminologické a tematické vymezení

Tato kapitola vymezuje použitou terminologii a tematické oblasti, které jsou zohledněny v této práci.

2.1 Informační systém podniku

> *„Digitální nervový systém zahrnuje digitální procesy, jež těsně propojují všechny aspekty činností firmy.“ (Gates, 1999)*

Informační systém podniku definuji za pomocí pojmu, který použil již Gates (1999), a který dle mého názoru přesně vystihuje důvody existence *informačního systému podniku*.

Obrázek 1: Digitální nervový systém (Gates, 1999; upraveno autorem)

Gates (1999) přirovnává řízení korporací k nervovému systému člověka. Nervový systém je v podstatě zodpovědný za kompletní zpracování našich vjemů (slyšíme a vidíme) a umožnuje nám myslet, vyhodnocovat a konat. I každá společnost potřebuje ke svému životu vstupy, na jejichž základě přijímá rozhodnutí a koná (viz Obrázek 1).

Jinými slovy, pod pojmem *digitální nervový systém* rozumíme neustálý informační tok uvnitř společnosti zajištující dostatečný rozsah informací pro zajištění zdravého chodu organizace. Tento *informační systém* umožnuje zrychlit základní pracovní operace, zajistit inovaci na základě obchodně-tržních reflexů, zefektivnit intra-komunikaci i extra-komunikaci a v neposlední řadě poskytnout validní data pro úspěšně a perspektivní strategické rozhodování.

Hlavní atributy informačního systému podniku (dále IS) lze na základě (Gála, 2009) shrnout:

- Účelem IS je poskytnout adekvátní podporu podnikových procesů prostřednictvím informačních a komunikačních technologií (dále ICT);
- IS propojuje tři základní prvky: lidé, data a ICT;
- IS často bývá nositelem nových obchodních příležitostí nebo zvyšování celkové efektivity podniku.

Na základě těchto poznatků lze pojem *informační systém podniku* zjednodušeně popsat jako *systém, který používají lidé (zaměstnanci, správci), aby se dostali k datům prostřednictvím ICT prostředků.* Nejdůležitější skutečností je, že informační systém má sloužit jako podpora podnikových procesů a nikdy ne naopak. Jinými slovy, u každé části informačního systému by měl být zřejmý přínos vůči podnikovým procesům společnosti.

Jak si ostatně také všímá Krafzig (2004), informační systém podniku je zpravidla úzce svázán s podnikovými procesy a měl by být schopen pružně reagovat na jejich změny. V současném vysoce konkurenčním prostředí se podnikové procesy a obchodní model obecně vyznačují vysokou dynamikou změn, a proto i informační systém by měl být pružný natolik, aby byl schopen držet krok s procesními změnami uvnitř podniku. Toto je kritický faktor úspěchu každého informačního systému avšak jak podotýká (Krafzig, 2004), informační systémy podniku se zpravidla vyznačují nízkou schopností změn a z toho důvodu se časem do jisté míry stávají informační systémy podniků značně omezujícími.

2.2 Specifika IS v malých a středních podnicích

Jak již bylo zmíněno v úvodu, tato práce je primárně zaměřena na malé a střední podniky (angl. Small and Medium Enterprises, dále SME). Na základě publikace vydané Eurostatem (2015) je zřejmé, že i v této oblasti panují ve světě značné terminologické rozdíly, a proto je určitě žádoucí nejprve tuto skupinu organizací definovat.

Eurostat (2015) používá pro definici velikosti společností kombinací následujících ukazatelů:

- počet zaměstnanců;
- roční obrat;
- a bilanční sumu rozvahy.

Jak je zřejmé (viz Tabulka 2), pro zařazení společnosti do dané velikostní skupiny je zapotřebí, aby byla splněna podmínka počtu zaměstnanců a zároveň velikosti obratu či bilanční sumy.

Tabulka 2: Definice velikosti společnosti (Eurostat, 2015)

Company category	Employees	Turnover or	Balance sheet total
Medium-sized	< 250	≤ € 50 m	≤ € 43 m
Small	< 50	≤ € 10 m	≤ € 10 m
Micro	< 10	≤ € 2 m	≤ € 2 m

Na okraj je vhodné poznamenat, že také Eurostat (2015) akcentuje celkovou důležitost malých a středních firem a jejich podílu na ekonomice celé Evropské unie - tyto podniky totiž reprezentují více než 90% všech podniků v EU.

Avšak na druhou stranu, i přes celkovou důležitost malých a středních firem, jejich tempo adopce informačních technologií (dále ICT) je relativně nízké oproti velkým podnikům (Taylor, 2015). Jak si také všímá Voříšek (2011), s velikostí podniku roste počet řídících složek a komunikace se tak stává více obtížnou. Z tohoto důvodu, pokud velké firmy chtějí zajistit rychlejší reakci na situaci na trhu, pociťují vyšší tlak na použití informačních technologií než menší podniky. S rostoucím počtem zaměstnanců také roste tlak na standardizaci podnikových procesů, při které informační technologie hrají klíčovou úlohu. Sečteno podtrženo, velké firmy dosahují relativně vyšší míry využití informačních technologií než menší podniky.

Ostatně teorie Everetta Rogerse o postupném pronikání nových technologií (angl. The Diffusion of Innovation (DOI) Theory) pojmenovává následující faktory (volně zpracováno dle (Taylor, 2015)) tohoto stavu:

- **Faktor vedení**;
- **Interní faktory**;
- **Externí faktory.**

Faktor vedení identifikuje vliv vedení společnosti a jeho dopad na inovace uvnitř podniku. Vliv ředitele/majitele podniku roste s klesající velikostí podniku, a proto u menších firem je osoba ředitele či majitele kritická pro tempo využití (angl. Adoption Rate) informačních technologií. Vedení zpravidla iniciuje inovace a má konečné slovo při alokování zdrojů na náklady vyvolané inovacemi. Pokud je člen vedení dostatečně technicky erudovaný, lze očekávat vyšší míru afinity směrem k informačním technologiím na úrovni celého podniku.

Mezi *interní faktory* teorie řadí:

- **Kulturu podniku a jeho otevřenost vůči změnám**. U menších podniků se očekává vyšší otevřenost vůči změnám, protože jejich zpravidla jednodušší řídící struktura je flexibilní a

dokáže se agilněji změnám přizpůsobit. Tato otevřenost může být měřena např. spokojeností uživatelů/zaměstnanců;

- **Organizační změny.** Celkový dopad nových informačních technologií (dále ICT) bývá u menší společnosti více signifikantní než u větších společností. Zavedení nových ICT projektů, zpravidla vyvolá relativně vysoké náklady, které se musí promítnout výrazněji do vnitřního fungování menších společností;
- **Vnitřní zdroje.** Obecně platí, že menší společnosti dokáží hůře alokovat zdroje na nové ICT projekty a jsou více obezřetní při analýze jejich přínosů. Přínosy ICT projektů se zpravidla projevují v delším časovém horizontu, a proto menší podniky dávají zpravidla přednost jiným projektům, které přinesou zisk v kratším časovém horizontu;
- **Velikost firmy a její celkové schopnosti.** S velikostí firmy také úzce souvisí celkové schopnosti firmy. Větší firmy jsou zpravidla schopny alokovat více specifických odborníků (lidské zdroje, řízení financí, ICT profesionálové atd.), a proto vyhodnocení přínosů ICT projektů je snazší u větších společností;
- **Oblast podnikání.** Adopce ICT technologií také souvisí s oblastí podnikání. Oblasti s celkově intenzivnější závislostí na ICT mají samozřejmě větší sklon k zavádění nových ICT.

Mezi *externí faktory* teorie řadí:

- **Konkurenční tlak.** Větší společnosti jsou zpravidla více resistentní vůči tlaku svých zákazníků, konkurentů či dodavatelů než menší společnosti. Menší společnosti se zpravidla snaží vyhovět externímu tlaku a velice dynamicky se přizpůsobovat, avšak to může znamenat hektické zavádění ICT bez detailních podpůrných analýz možných dopadů;
- **Externí ICT podporu.** Jak již bylo řečeno, menší společnosti zpravidla nemají dostatek ICT odborníků, a proto jsou více odkázány na spolupráci s externími společnostmi, které se následně stávají hlavním faktorem dalších inovací na poli ICT v dané společnosti. Avšak těmto externím společnostem chybí důkladné porozumění trhu, na kterém daná společnost podniká, a proto reakce na tržní podněty může být značně pomalá.

I přesto, že představená teorie DOI je velice často stále používaná, někteří autoři (Taylor, 2015) ji vytýkají, že již nezohleduje všechny současné faktory ovlivňující adopci ICT. Proto vznikl v roce 1990 rámec TOE (angl. The Technology, Organization and Environment), který přidává k výše popsaným faktorům následující hlediska (volně zpracováno dle (Taylor, 2015)):

- **Technologické**. Připravenost a otevřenost firem k využití ICT technologií přímo souvisí s vyspělostí jejich ICT infrastruktury a personálním zázemím v IT oblasti. Menší společnosti vnímají zpravidla deficit v obou těchto oblastech;

- **Vlastnosti ICT.** Rámec TOE akcentuje samotné vlastnosti ICT, jako jsou: uživatelská přívětivost, popularita, kompatibilita se současnými procesy apod. jako klíčové faktory využití nových informačních technologií. U menších společností se proto očekává větší důraz na tyto charakteristiky při jejich rozhodování o budoucím nákupu;
- **Organizační faktory.** Integrace nových technologií zpravidla vyžaduje spolupráci více organizačních jednotek uvnitř společností, a proto menší společností mají v této oblasti značnou výhodu. Výsledky ICT projektů jsou proto vnímány intenzivněji v případě menší společností (dotýká se relativně většího počtu zaměstnanců).

Za pomocí obou zmíněných teorií lze *specifika využití informačních technologií u malých a středních firem* shrnout:

- Horší schopnost alokovat zdroje (finanční i personální) na nové ICT projekty;
- Horší schopnost předem vyhodnotit dopad ICT projektů z hlediska nákladů a výnosů;
- Dopad ICT projektů (pozitivní i negativní) je zpravidla velice intenzivní a menší společnosti mohou z úspěšných ICT projektů čerpat značné a široce dosahující výhody;
- Vyšší závislost na dodavatelích externích ICT služeb, avšak z toho vyplívající snížení přizpůsobivosti pružně reagovat na tržní změny;
- Silná vůdčí role managementu při využití ICT technologií, avšak nemůžeme zároveň předpokládat, že management je technicky erudovaný, protože erudice v oblasti ICT zpravidla nebývá důležitá pro primární oblast podnikání dané společnosti;
- Vyšší flexibilita z hlediska změn nutných pro integraci nových ICT projektů do vnitřních procesů, avšak zároveň nižší tlak na formalizaci těchto procesů, u kterých proto musíme raději předpokládat, že vnitřní procesy jsou neformalizované.

K těmto hlediskům musíme přihlédnout v procesu *výběru vhodné metodiky integrace SaaS od on-premise prostředí podniku*, protože je zřejmé, že malé a střední podniky jsou z hlediska využívání informačních technologií specifické v mnoha ohledech.

2.3 Integrace v IT

Existuje celá řada definic tohoto pojmu, ale pro účely této práce jsem vybral jednu, která přesně vystihuje *integraci* ve vztahu k zaměření této práce:

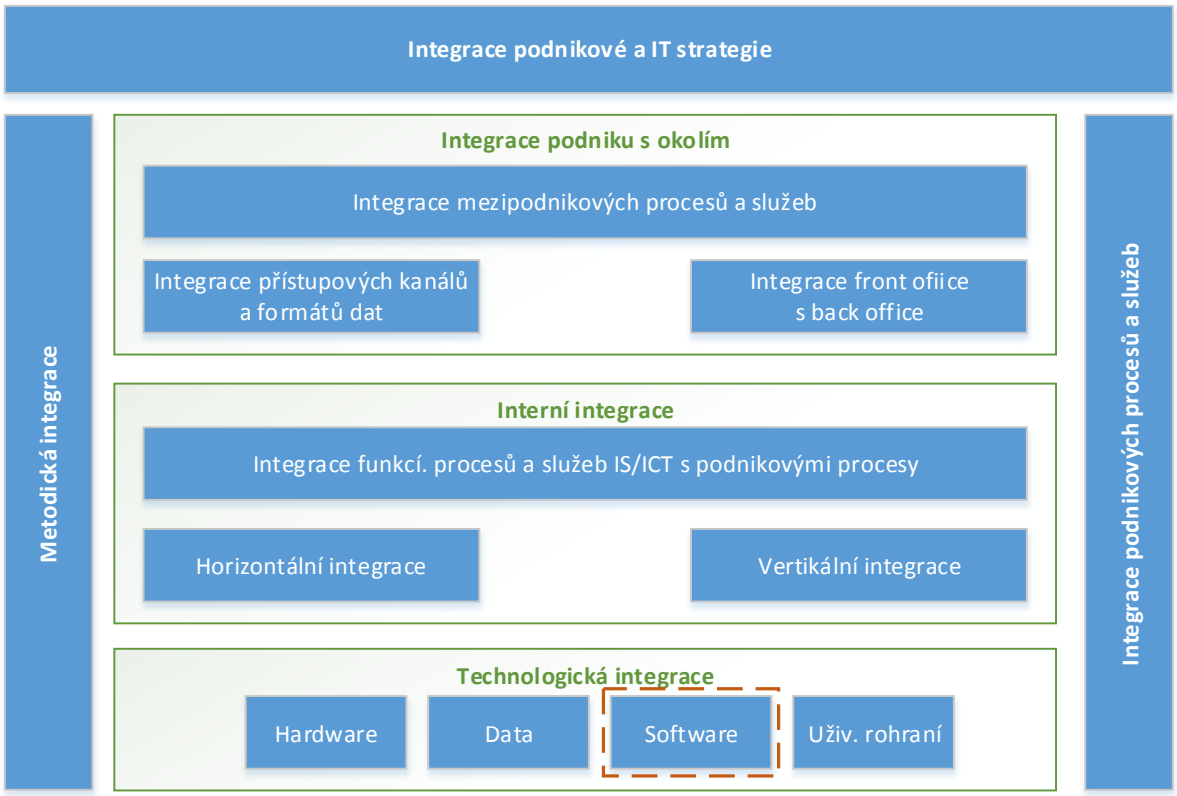
„V podnikové informatice se vyskytuje celá řada technologií, aplikačních systémů, různorodých dat, různorodých zájmů a cílů. Už od počátku nasazování počítačů se zabýváme otázkami, jak tuto různorodou množinu zdrojů vhodně propojit tak, aby podporovala cíle organizace a aby nově vzniklý celek přinesl organizaci novou přidanou hodnotu, reprezentovanou např. sníženými provozními náklady nebo vytvořením nových příležitostí pro organizaci.“ (Gála, 2009)

V nejširším možném smyslu, slovo „integrace“ znamená „začlenění“. Singletary (2003) tento pojem vymezuje v užším smyslu jako „spolupracovat“ (angl. Working together).

Pro účely této práce, a s přihlédnutím k definici výše, lze pojem „integrace“ popsat jako *„vzájemnou spolupráci (distribuovaných) ICT komponent takovým způsobem, aby plnily cíle organizace“*.

Vzhledem k zaměření práce, jež je *integrace SaaS*, proto nebudu dále uvažovat o integračních úlohách probíhající v rámci jednoho lokálního aplikačního procesu, např. technologie typu COM (angl. Component Object Model), dynamicky linkované knihovny (DLL) atd. a budu dále předpokládat, že integrace probíhá na úrovni *distribuovaných* komponent.

Práce také nezohledňuje některé specifické případy integrace, jako např. integraci podnikových procesů, business-to-business integraci (kooperaci B2B partnerů), integrace podniku s okolím atd. V rámci informačního systému podniku je možné různé integrační úlohy shrnout (viz Obrázek 2) a znázorněním výsledné oblasti vymezit primární zaměření této práce.



Obrázek 2: Model integrace IS/ICT podniku a vymezení oblasti zájmu této práce (Voříšek, 2011; upraveno autorem)

Na tomto místě je vhodné také upozornit na možné rozdílné vnímání pojmů *integrace aplikací* a komunikace v rámci *distribuovaných aplikací* (více vrstvých aplikací). Hohpe (2003) tyto dva způsoby rozlišuje a v podstatě komunikaci v rámci distribuovaných aplikací nepovažuje za skutečnou integrační úlohu. Avšak v rámci této práce nebudu toto rozlišení brát v potaz, protože se domnívám, že v případě cloudových služeb je možné formou SaaS pořizovat již libovolnou vrstvu (komponentu)

konkrétní aplikace (obchodní logika, prezentační logika nebo datová logika) a v tom případě se vývojář musí zpětně zabývat integrací tímto způsobem rozdělených aplikací.

2.4 Metoda versus metodika

V praxi se často setkáváme se zaměňováním těchto dvou pojmů, a proto se domnívám, že jejich vymezení má také své místo v této práci. Dle Buchalcevové (2005) vybírám dvě definice, které zohledňuji v této práci:

- **Metoda:** *„Metoda určuje, co je třeba udělat v určité fázi nebo činnosti vývoje systému. Zpravidla se týká jen některého úhlu pohledu na systém.“;*
- **Metodika:** *„Metodika představuje v obecném smyslu souhrn metod a postupů pro realizaci určitého úkolu.“.*

Jinými slovy lze vztah obou pojmů popsat: *metodika* sdružuje jednotlivé *metody*.

Metodiky můžeme dělit na dvě základní skupiny (volně zpracováno dle Buchalcevová, 2005):

- **Rigorózní metodiky:** velice detailní vycházející z přesvědčení, že všechno lze měřit, řídit, popsat a předem zcela definovat pomocí procesů;
- **Agilní metodiky:** prosazující myšlenku, že správnou cestou tvorby produktu je výhradní soustředění se na produkt samotný a vynechat cokoli co přímo nesouvisí s daným produktem. Tyto metodiky preferují rychlé vytvoření produktu (nebo jeho části), předložení zákazníkovi a na základě jeho požadavků produkt upravit a ve formě tzv. iterací pokračovat v dalším vývoji.

Rigorózní metodiky jsou zpravidla těžké, tzn. pokrývající celý životní cyklus vývoje informačního systému, včetně detailního návrhu systému, vývoje, implementace, nasazení a údržby. Rigorózní metodiky jsou také zpravidla globální, pokrývají celkový pohled na organizaci a její podnikové procesy a předmětem jejich zájmu je informační systém jako celek. Tyto metodiky zdůrazňují úlohu procesů při vývoji SW a věří, že kvalitně navržené procesy vedou ke kvalitnímu SW. V neposlední řadě je také akcentována potřeba detailní dokumentace. Rigorózní metodiky se zpravidla hodí pro velké celopodnikové informační systémy. Mezi rigorózní metodiky patří například metodika MMDIS (angl. Multidimensional Management and Development of Information System), EUP (angl. Enterprise Unified Process) nebo RUP (angl. Rational Unified Process).

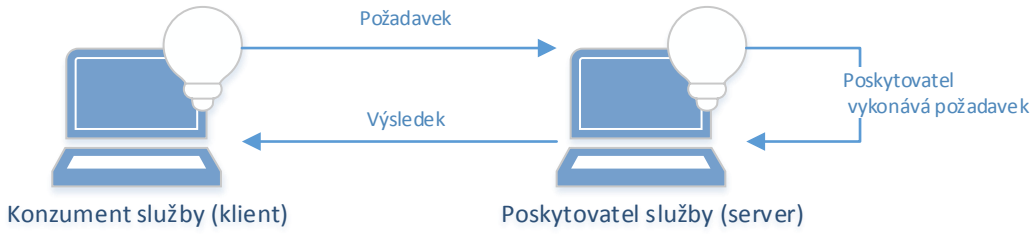
Oproti tomu *agilní metodiky* vycházejí z předpokladu, že SW procesy nelze detailně popsat, protože před vývojem informační systému má zákazník pouze mlhavou představu o jeho funkčnosti a je schopen uvažovat spíše v hrubých rysech. Agilní metodiky jsou lehké a obvykle zahrnují pouze obecná pravidla a praktiky, které lze při vývoji využít. Jejich rozsah je dán pouze jednotlivými

projekty, a proto patří mezi metodiky projektové. Tyto metodiky se snaží maximalizovat hodnotu pro zákazníka tím, že se plně soustřeďují na produkt jako takový a nízkou váhu přikládají dokumentaci a pomocným modelům. Při aplikaci těchto metodik má vývojář velmi volnou ruku a není příliš svazován procesy. Mezi agilní metodiky řadíme, např. Lean Development, Scrum nebo XP (eXtrémní programování).

2.5 Služba

V nejobecnějším vyjádření pojem služba můžeme definovat dle Kotlera (1988) jako *"...jednání nebo konání, které jedna strana může nabídnout jiné, a které je v zásadě nehmotné povahy."* (Voříšek, 2011).

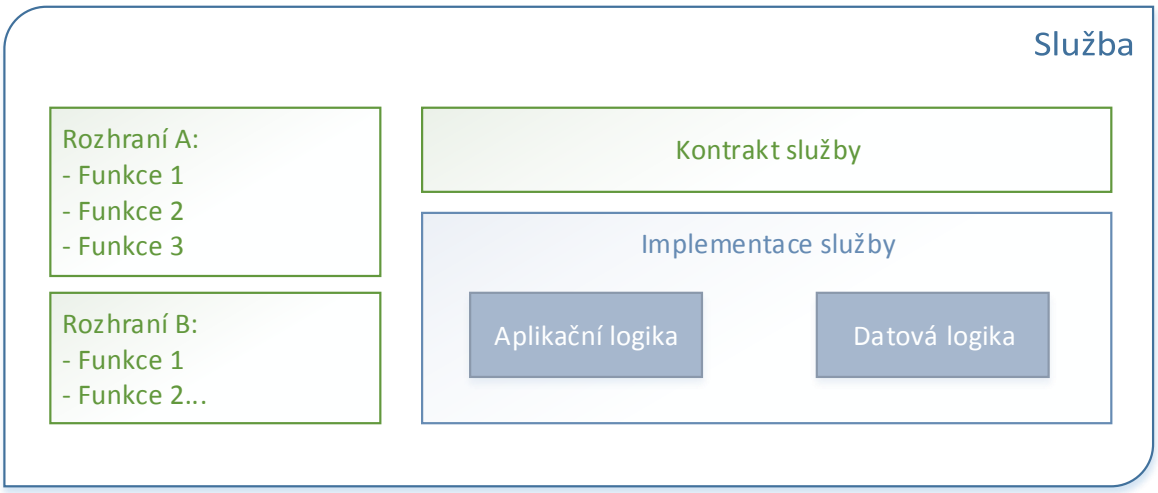
Avšak pro účely této práce tuto obecnou definici více zkonkretizuji a aplikuji na oblast IT resp. oblast komunikace mezi aplikacemi dle Krafziga (2004, volně přeloženo autorem): *„Služba je smysluplná aktivita, kterou vykonává počítačový program na základě požadavku jiného počítačového programu."* Nebo také jinými slovy: *„Služba je samostatná programová komponenta přístupná vzdáleně."* (viz Obrázek 3)



Obrázek 3: Pojem „služba" (Krafzig, 2004; upraveno autorem)

Z hlediska zaměření této práce rozeznáváme následující klíčové komponenty služeb (Krafzig, 2004; volně přeloženo autorem):

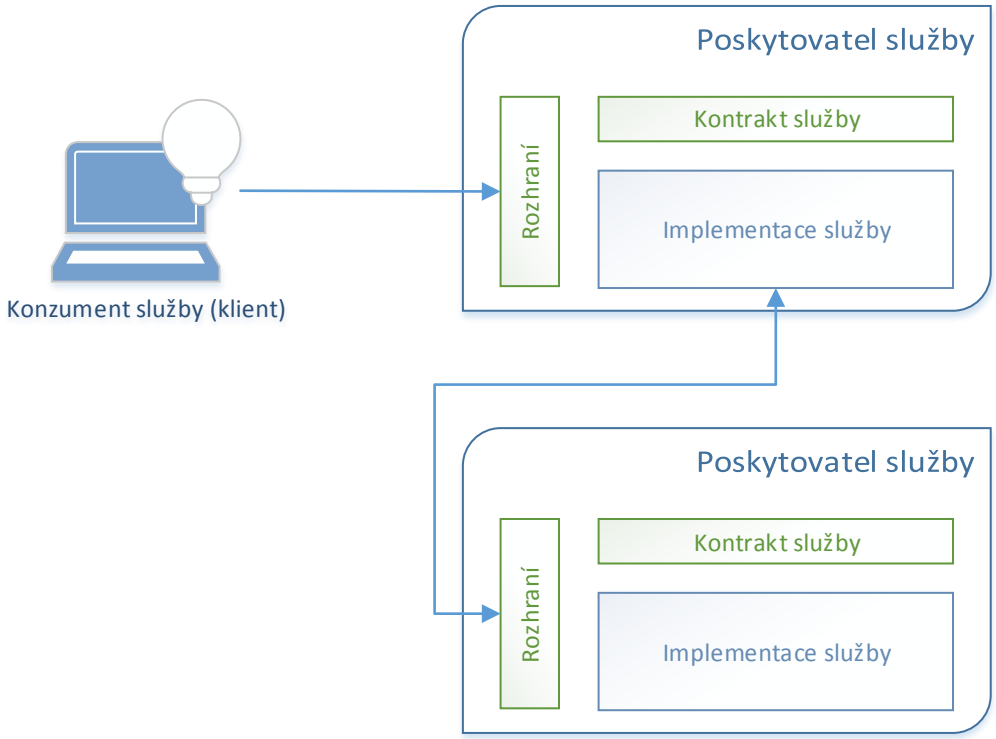
- **Kontrakt***:* neformální popis služby z hlediska účelu, funkcionality, omezení a užití;
- **Rozhraní***:* konkrétní funkcionalita služby je navenek vyjádřena formou takzvaného rozhraní, které přesně popisuje, jaké funkce jsou k dispozici a jakým způsobem je můžeme volat (použít);
- **Implementace***:* implementace služby je konkrétní technická realizace služby, která obvykle zapouzdřuje aplikační logiku a datovou logiku a reaguje na zveřejněné rozhraní služby.



Obrázek 4: Služba a její základní komponenty (Krafzig 2004; upraveno autorem)

Jak přibližuje obrázek (viz Obrázek 4), služba navenek zveřejnuje pouze *kontrakt* a *rozhraní*. Jinými slovy služba navenek komunikuje pouze skrze rozhraní a kontrakt. Konkrétní *implementace* je před okolím skryta.

Tato *zapouzdřenost* je velice důležitá vlastnost služeb, protože umožnuje měnit implementaci služeb, aniž by došlo ke změnám v rozhraní. To mimo jiné znamená, že můžeme různé služby různě řetězit a toto zřetězení libovolně upravovat, aniž by došlo k jakýmkoli změnám na straně konzumenta (viz Obrázek 5).



Obrázek 5: Detailní pohled na zapouzdřenost služeb

2.6 Cloudové služby

Mishra (2013; volně přeloženo autorem) popisuje cloudové služby jako: *„…distribuovaný výpočetní model poskytující se formou služeb (angl. service-oriented), který nabízí vysoce škálovatelné IT prostředky po síti“.*

Principy a myšlenky, na kterých stojí cloudové služby jsou známé již desítky let, avšak až společnost Google (od roku 2006) tento pojem začala aktivně používat pro zcela nový obchodní model inspirovaný modelem internetových služeb pro spotřebitele (Mishra, 2013). Od tohoto okamžiku se tento koncept začal uplatňovat a používat v širším měřítku.

Cloudové služby nabízejí následující výhody pro koncové zákazníky:

- Správa cloudových služeb je kompletně v rukou poskytovatele, který ručí za funkčnost. Příjemce služeb může zcela vypustit náklady za správu a platí pouze za provoz služby. Navíc správa a údržba je prováděna na straně poskytovatele, a proto také aktualizace (nové verze, opravy chyb atd.) jsou prováděny bez zásahu do vlastní infrastruktury příjemce;
- Příjemce cloudových služeb může služby začít používat v podstatě okamžitě bez vysokých vstupních nákladů či technologických omezení. Jinou otázkou je integrace takových služeb, která již obvykle leží na bedrech příjemce;
- Úhrada za použití cloudových služeb je zpravidla nabízena formou průběžných plateb pouze za skutečně spotřebované množství (angl. Pay as you go). Z toho důvodu příjemce nemusí zbytečně utrácet za předimenzované zdroje pro případ náhlých výkyvů spotřeby. Tato forma úhrady s sebou přináší pro organizace ještě jednu důležitou změnu. Náklady těchto služeb se účtují jako *provozní* a již ne jako *investiční*, což může být obzvláště pro menší společnosti výhodnější scénář;
- Díky sdílené formě nabízených prostředků je dosaženo vyšších úspor z hlediska nákladů, než v případě, pokud bychom chtěli stejný rozsah prostředků dedikovat vlastními silami;
- Cloudové služby jsou poskytovány na základě smlouvy o poskytování úrovně služeb (angl. Service Level Agreement, SLA).

Cloudové služby se člení do následujících skupin dle typu poskytovaných služeb (Mell, 2011; volně přeloženo autorem):

- **Software jako služba (angl. Software as a service – SaaS):** služby spojené s poskytováním podnikových aplikací formou cloudu. Tato skupina je stěžejní pro splnění cílů této práce;

- **Platforma jako služba (angl. Platform as a service – PaaS):** služby spojené s poskytováním kompletních aplikačních platforem určených pro vývoj vlastních aplikací (kombinace HW i SW);
- **Infrastruktura jako služba (angl. Infrastructure as a service – IaaS):** služby spojené s poskytováním prostředků IT infrastruktury (virtuální servery, datová uložiště atd.).

V praxi se mnohdy rozdíly mezi jednotlivými druhy stírají, a je tak spíše na té konkrétní situaci, do které konkrétní kategorie daná služba spadá. Například Salesforce je typicky uváděn jako SaaS, avšak může být zařazen také jako PaaS, protože poskytuje v neposlední řadě běhové prostředí pro provoz vlastních aplikací. Také asi nepřekvapí, že formou cloudu se již také nabízejí například kompletní *integrační platformy* (iPaaS) (viz Kapitola 3.2.3).

Cloudové služby lze dále členit např. podle druhu nasazení (Privátní, Veřejný atd.), ale další členění již nejsou podstatná pro splnění cílů této práce, a proto se jimi nebudu detailněji zabývat.

Kompletní oblast cloudových služeb tvoří velice obsáhlý systém se spoustou prvků a aktérů, avšak jejich popis stojí mimo cíle této práce. Detailní popis cloudové architektury je možné nalézt např. v (Liu, 2011).

V následujících kapitolách představím *technologické vlastnosti cloudových služeb* a princip *bezpečnosti cloudových služeb*. Obě tyto oblasti budou hrát stěžejní úlohu při posuzování různých metodik integrace služeb SaaS.

2.6.1 Technologické vlastnosti cloudových služeb

V předchozí kapitole byly představeny cloudové služby na obecné úrovni, avšak z hlediska zaměření této práce jsou podstatné následující technologické vlastnosti cloudových služeb, které musíme brát v úvahu při jejich integraci do informačního systému podniku (volně zpracováno dle (Voříšek, 2011)):

- Cloudové služby jsou zpravidla provozované na infrastruktuře poskytovatele služeb a ten se stará o jejich správu. Cloudové služby jsou tedy provozovány v *off-premise* prostředí (tj. fyzicky zcela mimo infrastrukturu daného podniku)[3] a komunikace se službou probíhá prostřednictvím internetu;
- Architektura cloudových služeb je navržena takovým způsobem, aby tutéž aplikaci mohlo ve stejném okamžiku využívat více uživatelů. Z toho také vyplývá nutnost jednoznačné identifikace uživatele při komunikaci se službou (viz Kapitola 2.6.2);

[3] Toto nemusí zcela platit v případě privátního cloudu

- Cloudové služby poskytují svou funkcionalitu prostřednictvím internetového prohlížeče nebo zveřejnují svá programovatelná *rozhraní služeb* (angl. Application Programming Interface) neformálně popsaná formou *kontraktu služby*;
- Cloudové služby jsou zpravidla platformě nezávislé, a proto je zcela nepodstatné na jaké platformě je daná služba vyvíjena na straně poskytovatele či používána na straně konzumenta;
- V rámci integrace cloudových služeb je tedy nutné vyřešit otázku komunikace v rámci zcela neheterogenních prostředích.

2.6.2 Bezpečnost SaaS služeb

V případě, že SaaS služby nejsou určené pro širokou veřejnost (přístupné pro každého uživatele), musí zpravidla jejich poskytovatel vyřešit důvěryhodný a spolehlivý způsob zajištění oprávněného přístupu k datům. Nejčastěji se tak děje prostřednictvím dvou základních metod (Mainka, 2014; volně přeloženo autorem):

- **Kombinace jména a hesla:** poskytovatel služby poskytne unikátní dvojici jména a hesla a tuto kombinaci očekává při užívání služby. Na základě těchto poskytnutých údajů poskytovatel služeb vyhledá konkrétní účet a provede autorizaci;
- **Jednotné přihlášení (angl. Single Sign-On):** V tomto případě se společně s užíváním služby posílá unikátní „token“, který byl spolehlivě autorizován od důvěryhodného poskytovatele identit (angl. Identity Provider).

Volba konkrétní metody zabezpečení závisí na rozhodnutí poskytovatele služeb a konzument služby (implementátor) by s danou metodou měl počítat při návrhu vlastní integrace SaaS.

Dalším stupněm ochrany bývá zabezpečení dat při komunikaci. Erl (2005) rozlišuje:

- **Zabezpečení vlastní komunikace se službou (angl. Transport-level Security):** zabezpečuje vlastní komunikaci (šifruje) se službou a chrání ji tak proti možnému odposlechu;
- **Zabezpečení posílané zprávy (angl. Message-Level Security):** zabezpečuje posílaná data (šifruje) a daná komunikace může být nešifrovaná;

Obě zmíněné možnosti se mohou vzájemně kombinovat. Rozdíl spočívá v situaci, kdy mezi konzumentem a vlastní službou vyžadujeme zapojení ještě nějakého prostředníka. V tom případě je komunikace zabezpečena pouze mezi konzumentem a prostředníkem, a na straně prostředníka mohou data být již nezabezpečená v případě, že zvolíme pouze zabezpečení na úrovni komunikace.

2.6.3 Integrace aplikací SaaS

Randová (2010) ve své diplomové práci shrnuje problémy, které pojmenoval Feuerlicht (2008), a které jsou specifické pro integraci aplikací typu SaaS: *„Aplikace implementované modelem on premise jsou často založeny na architekturách výrobce a jsou tak obtížné integrovatelné s ostatními podnikovými aplikacemi. Společnosti obvykle mají nasazeno více aplikací od různých prodejců – např. CRM, HRM, databáze a další. Je důležité, aby aplikace spolupracovaly, byly propojeny, dokázaly spolu komunikovat, a to nejen v rámci podniku, ale i mezi různými společnostmi. Integraci je možno provádět mnoha různými způsoby.“*

Jak ve své práci (Randová, 2010) analyzuje, obecně je možné rozeznat následující modely integrace aplikací SaaS:

- **Integrace stávající aplikace s aplikací SaaS – bez integrační vrstvy**: jedná se o situaci, kdy společnost vlastní aplikaci (např. ERP systém) a nově ji potřebuje integrovat s aplikací SaaS;
- **Integrace stávajícího IS s aplikací SaaS (včetně integrační vrstvy)**: v této variantě má společnost zaveden informační systém a potřebuje jej doplnit o aplikaci SaaS (včetně integrační vrstvy);
- **Integrace dvou a více SaaS aplikací:** tento model reprezentuje situaci, kdy společnost hodlá přenechat celé řešení podnikové informatiky na externím dodavateli;
- **Integrace skupiny SaaS aplikací s existujícím IS:** reprezentuje kombinaci výše uvedených modelů. Společnost již vlastní nějaké aplikace, které provozuje on-premise, a vlastní i aplikace poskytované jako SaaS. Nově dochází k zavedení další SaaS aplikace, která musí být propojena s on-premise aplikacemi i se SaaS aplikacemi.

Cílem této práce je návrh metodiky pro první dva zmíněné případy a částečně čtvrtý případ, tj. *integrace stávajícího on-premise IS s aplikací SaaS.*

Podle Randové (2010), můžeme dále rozlišovat, kdo konkrétně provádí integraci:

- Konzument;
- Poskytovatel (Integrace jako služba);
- Poskytovatel SaaS;
- Integrátor.

Metodika integrace, jež je stěžejním cílem této práce, zohledňuje případ, kdy integraci provádí sama společnost (*konzument*) a také případ, kdy technickou stránku integrace svěří konzument poskytovateli integrační platformy (*Integrace jako služba*). Různé metody integrace představuji v následující kapitole 3.

3 Východiska integrace a její stavební prvky

Tato kapitola si klade za cíl představit základní technologické prostředky integrace a pojmenovat teoretická východiska, která jsou podstatná pro následný popis a hodnocení navržené metodiky integrace služeb SaaS.

Z větší části se jedná o popis současných poznatků, myšlenek, modelů a ověřených postupů, které výrazným způsobem ovlivňují různé přístupy k integraci v IT. Součástí popisu těchto stavebních prvků je pojmenování základních vlastností, kterými lze různé integrační přístupy srovnávat a hodnotit.

3.1 Architektura IT

Informační a komunikační technologie (dále ICT) lze rozdělit do jednotlivých skupin (prvků), které lze vzájemně uspořádat do tzv. *podnikové IT architektury* (angl. Enterprise IT Architecture (Feuerlicht, 2008)). *Architektura IT* je nástroj, kterým lze jednotlivé ICT prvky uspořádat do uceleného systému a naznačit vzájemné vztahy mezi nimi.

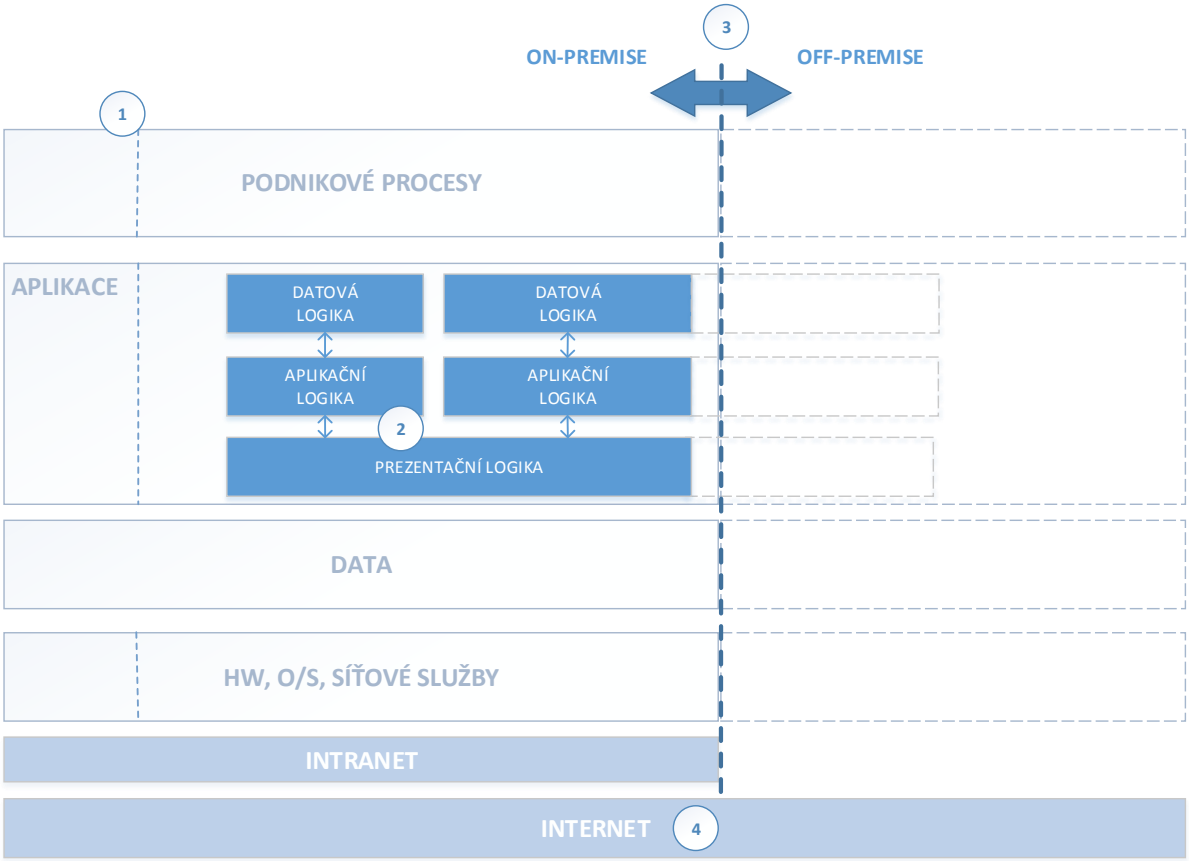
„Důležitou funkcí IT architektury je možnost strukturovaným způsobem popsat návrh, vývoj a údržbu informačních systémů“
(Feuerlicht, 2008; volně přeloženo autorem).

Právě pro popis základních východisek a technologických prostředků integrace jsem využil schématického znázornění různých ICT komponent v jednotlivých logických vrstvách, které společnou kombinací tvoří páteř podnikové informatiky (viz následující kapitola 3.1.1).

3.1.1 Vícevrstvá architektura

Jednotlivé vrstvy, které představují specifickou funkcionalitu informačních systémů a jsou charakterizovány různými ICT technologiemi, znázorňuje schéma níže (viz Obrázek 6).

Výhodou tohoto *vrstveného modelu* je, že zjednodušuje celkový pohled na celý systém a odděluje od sebe jednotlivé oblasti zájmu, např. databázový architekt se může soustředit pouze na datovou vrstvu a neuvažovat o ostatních vrstvách (Feuerlicht, 2008; volně přeloženo autorem).



Obrázek 6: Schéma vrstvené IT architektury (Feuerlicht, 2008; upraveno autorem)

Základní princip fungování zobrazeného výpočetního systému lze popsat následujícím způsobem:

- Na úrovni podnikových procesů vzniká požadavek (potřeba), který vyžaduje zapojení podnikové informatiky do podnikového procesu;
- Prostřednictvím spuštěné aplikace se uživatel dostává do interakce s informačním (výpočetním) systémem;
- Aplikace je spuštěna na dané ICT infrastruktuře (HW/OS) a získává data (informace), která jsou uložena v datovém uložišti (databáze, soubor atd.) reprezentovaném datovou vrstvou (DATA). V případě potřeby může daná aplikace také komunikovat s okolním světem mimo danou organizaci prostřednictvím internetu.

V nejzákladnějším možném scénáři „běží“ aplikace na dedikovaném HW, využívá dedikované uložiště a poskytuje funkcionalitu jednomu uživateli. Tento výpočetní model se obvykle označuje jako model *decentralizovaného zpracování dat*, nebo také *„model izolovaných PC“* (Gála, 2009). Vyznačuje se velice nízkou (skoro žádnou) schopností vzájemné výměny dat s okolním světem a nulovou interoperabilitou (vzájemnou spoluprací).

Při tomto scénáři také vzniká situace, kdy pro „každý“ podnikový proces jsou dedikovány vlastní ICT prostředky a tím vznikají *izolované aplikační ostrůvky* (angl. Application Silos (Feuerlicht, 2008)). To

je samozřejmě z hlediska ekonomických nákladů pro organizaci nežádoucí stav. Z tohoto důvodu se tento výpočetní model opouští a je nahrazován pokročilejším.

Představme si novou situaci, kdy některá ze znázorněných vrstev bude fyzicky rozdělena (např. různé aplikace, různé podnikové procesy atd., podobně jako na popisovaném modelu (viz Obrázek 6, bod 1), a jiná vrstva zůstane celistvá a její prostředky začnou být sdílené (v našem příkladu vrstva dat). Tento model je již efektivnější, protože různé aplikace a různé procesy mohou navzájem sdílet data, jelikož celý systém sdílí stejné prostředky (v zobrazeném příkladu stejné uložiště). Tento výpočetní model se nazývá „*distribuované zpracování aplikací*“ (Gála, 2009) a někdy bývá také označován pojmem *klient-server*, protože *sdílené prostředky* jsou obvykle poskytovány prostřednictvím specializovaného serveru. Uvedený scénář se nemusí týkat pouze datové vrstvy jako v tomto ilustračním příkladu, ale prakticky každou ze znázorněných vrstev můžeme sdílet a jiné můžeme použít odděleně. Rozdíl je samozřejmě v tom, jakými prostředky toho dosáhneme, tj. každá vrstva pro tento účel používá jiné technologické prostředky.

Tento inovovaný výpočetní model není výhodný pouze v rámci sdílení dat či informací, ale jednu z klíčových rolí hraje také efektivnější využití ICT prostředků obecně. Nedostatečně využité dedikované ICT prostředky způsobují organizacím nepřímé ztráty, a proto je žádoucí ICT prostředky sdílet, abychom maximalizovali jejich využití.[4]

Stále zůstaneme u popisovaného modelu a soustředíme se nyní na aplikační vrstvu (aplikace). Každá aplikace může být dále vnitřně členěna na logické celky, které se starají o specifickou funkcionalitu dané aplikace (viz Obrázek 6, bod 2). Jedná se o koncept, který se používá při vývoji aplikací a umožnuje vývojářům od sebe logicky oddělit jednotlivé části jedné aplikace takovým způsobem, že mohou být spravovány nezávisle (*vícevrstvá architektura aplikace*).

Výhodou tohoto modelu je, že daná aplikace může být případně rozdělena na jednotlivé komponenty a ty následně mohou být i fyzicky oddělené. Jinými slovy se z dříve monolitické aplikace stane modulární aplikace. Mezi základní komponenty patří (volně zpracováno dle (Gála, 2009)):

- **Prezentační logika (angl. Presentation Logic):** je to ta část aplikace, která se stará o interakci s koncovým uživatelem (formulář, webová stránka atd.) nebo jiným způsobem připravuje informace pro zobrazení (tisk atd.);
- **Aplikační logika (business logika, angl. Business Logic):** ta část aplikace, která koordinuje fungování celé aplikace a může také obsahovat *obchodní pravidla a omezení (angl. Business*

[4] Pokud například organizace zakoupí dedikovaný server a ten je využíván v průměru pouze na 12 procent své výpočetní kapacity, 88 procent kapacity zůstává v průměru nevyužito i presto, že bylo zaplaceno celých 100 procent kapacity

Rules and Constrains) dané daným podnikovým procesem, např. účtuj slevu 10%, pokud je objednávka vyšší než 1000 Kč;

- **Datová logika (angl. Data Logic):** ta část aplikace, která řídí přístup k datům a operaci s daty – ukládání, čtení, transformace atd.

Nabízí se otázka, proč je takový modulární model aplikace výhodný, protože je zřejmé, že celý systém se tímto krokem značně komplikuje:

- **Specializace kódu:** vývojáři se mohou soustředit na tu konkrétní část aplikace, která odpovídá jejich odbornosti (databázový specialista na datovou logiku, grafik na prezentační logiku atd.). Dokonce se může stát, že danou část aplikace vytvořila zcela jiná společnost;
- **Možnost rozdělit zpracování:** Jak podotýká (Peterka, 2007): *„Výpočetní model klient/server je založen na myšlence, že data by se měla zpracovávat především tam, kde jsou k dispozici. Nejsou-li k dispozici místně, tj. přímo u uživatele, ale jsou umístěna centrálně, pak nechť se také zpracovávají na onom centrálním místě, kde se skutečně nachází.“* Jinými slovy, pokud oddělíme datovou logiku mimo hlavní aplikaci, ta může běžet na zcela jiném počítači mnohem blíže k datům a zjednodušit tak komunikaci;
- **Škálování výkonu** (angl. Application Scaling): pokud bude nutné posílit jednu konkrétní část aplikace, je to možné udělat bez nutnosti zasahovat do ostatních částí aplikace;
- **Snadnější údržba:** Při údržbě se můžete soustředit na danou část bez ohledu na části jiné, např. aplikační logiku můžete aktualizovat dle aktuálních podmínek na trhu;
- **Flexibilní návrh aplikace odolný proti změnám:** danou část aplikace je možné zcela vyměnit, bez toho aniž by ostatní části aplikace tuto změnu zaregistrovaly. Například se můžete rozhodnout, že uživatelé nebudou aplikaci využívat pomocí desktopového klienta, ale například pomocí webového. V takovém případě proběhne změna pouze na úrovni prezentační vrstvy a ostatní vrstvy nemusíme vůbec měnit;
- **Sdílení kódu:** tato vlastnost je již naznačena v popisovaném obrázku (viz Obrázek 6, číslo 2). V modulární aplikaci můžete sdílet určité části kódu napříč celým systémem aplikací (např. kontrola pravopisu, přístup k datům atd.).

Celý model vícevrstvých aplikací lze shrnout: *„Logika aplikace je realizována jako množina programových komponent (programů)“* (Gála, 2009).

Dosud jsem popisoval situaci pouze uvnitř daného podniku. Jednotlivé vrstvy se rozdělily na různě oddělené komponenty, které ale zůstávaly sdílené pouze v rámci daného podniku (LAN, VPN atd.).

Nyní si do celého systému přidejme novou dimenzi: jednotlivé oddělené komponenty můžete také provozovat zcela mimo danou organizaci (angl. Off-premise), viz Obrázek 6, bod 3:

- Mimo organizaci můžeme provozovat jednotlivé vrstvy ICT prostředků (data, HW, aplikace, procesy atd.);
- Ale také jednotlivé části aplikací (jednotlivé komponenty).

Celá tato situace je naznačena čárkovanými vrstvami, které přesahují část mimo danou organizaci. Z obrázku je dále zřejmé, že daná organizace je propojena s okolním světem prostřednictvím internetu a jen díky internetu je možné oddělovat dané komponenty mimo danou organizaci (viz Obrázek 6, bod 4) a provozovat je například prostřednictvím cloudových služeb (viz Kapitola 2.6).

Z tohoto faktu mimochodem vyplývá jedna často opomíjená skutečnost, že jedním z výrazných hnacích motorů a podmínek rozvoje cloudových služeb byl rapidní rozvoj internetu a zdokonalování internetové infrastruktury obecně. I přes banalitu této myšlenky je třeba mít tuto skutečnost stále na paměti, protože úspěch integrace služeb SaaS stojí a zároveň padá na dokonalé internetové infrastruktuře.

3.1.2 Shrnutí

Předchozí kapitola 3.1.1 popisovala za pomocí vrstveného modelu IT jednotlivé komponenty ICT a logické vrstvy aplikace a jejich vzájemnou interakci. Byly předestřeny důvody, proč je výhodné a žádoucí jednotlivé komponenty od sebe odlišovat, oddělovat a zpětně integrovat.

Z dané kapitoly také vyplývá, že všechny prvky, které můžeme od sebe oddělit, můžeme navíc vydělit a provozovat zcela mimo organizaci. V takovém případě se komunikace mezi jednotlivými komponentami odehrává prostřednictvím internetu. Mezi hlavní výhody této modulární architektury řadíme:

- **Provoz a správa**: provoz a správa jednotlivých komponent může být zcela svěřena třetí straně;
- **Aktualizace/výměna/změna**: komponenty mohou být změněny, aktualizovány či vyměněny bez nutnosti fyzicky navštívit danou organizaci;
- **Sdílení:** Stejně jako můžeme sdílet programové komponenty v rámci organizace, s tímto inovovaným prostředím můžeme programové komponenty sdílet v rámci celého světa.

Právě tento přesah hranic jednoho podniku je principem celé škály poskytovaných cloudových služeb (angl. Software as a Service, Infrastructure as a Service, Platform as a Service). V dnešní době je možné prakticky kteroukoli ICT komponentu pořizovat formou cloudových služeb. Na druhou stranu, jakmile ICT komponenty jednou fyzicky oddělíme, musíme vyřešit jejich vzájemnou komunikaci takovým způsobem, aby nebyla narušena jejich primární funkce včetně bezpečnosti.

Gála (2009) tuto skutečnost shrnuje: *„Integrací v podnikové informatice rozumíme akt nebo proces, kterým kombinujeme, propojujeme a spojujeme různorodé zdroje podnikové informatiky do vyšších celků, přičemž komponenty dohromady spolupracují a sdílejí data bez znatelného zdržení a koordinují svou funkcionalitu tak, že se taková kombinace komponent jeví jako jednotný systém.“*

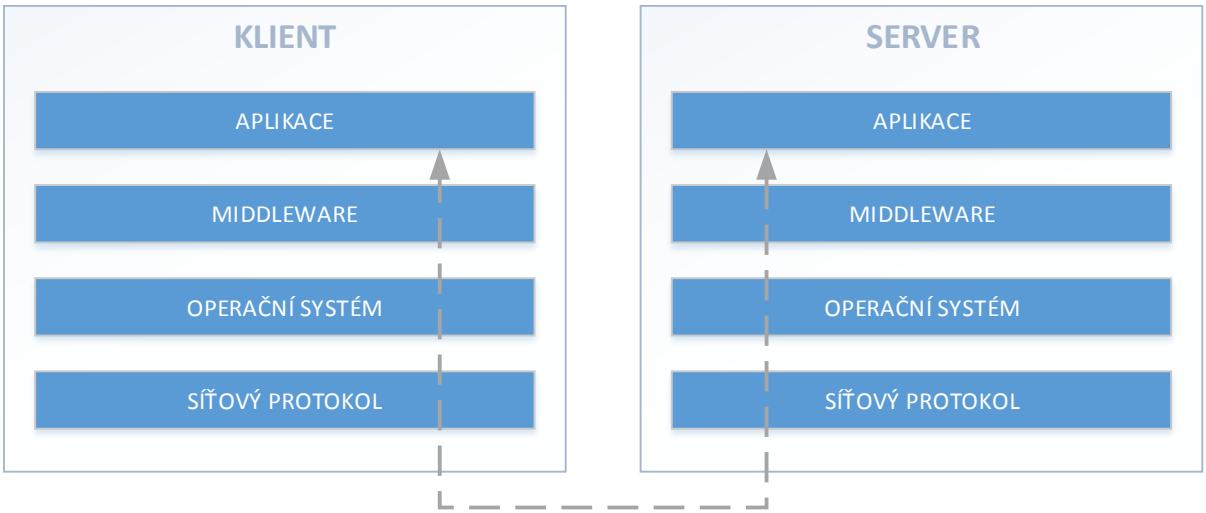
K dosažení tohoto stavu používáme právě *prostředky IT integrace*, při jejichž podrobnějším studiu nalezneme řešení mj. pro všechny zmíněné vrstvy ICT komponent:

- Vrstva aplikací (můžeme využít koncept Integrace podnikových aplikací (angl. Enterprise Application Integration, EAI));
- Vrstva dat (můžeme využít prostředky pro Integraci dat a informací);
- Vrstva ICT infrastruktury (můžeme využít služeb middleware).

V následujících kapitolách představím právě ony základní stavební prvky systémové integrace, které mají rozhodující vliv na různé návrhové vzory důležité pro dosažení cíle této práce.

3.2 Middleware

„Middleware je programové vybavení, které je umístěno mezi operačním systémem a aplikacemi a které nabízí aplikacím všestranně použitelnou množinu služeb. Služby vytvářejí provozní systémové prostředí, které umožnuje procesům, programům a aplikacím vzájemnou interakci v distribuovaném systému prostřednictvím sítě.“ (Gála, 2009)



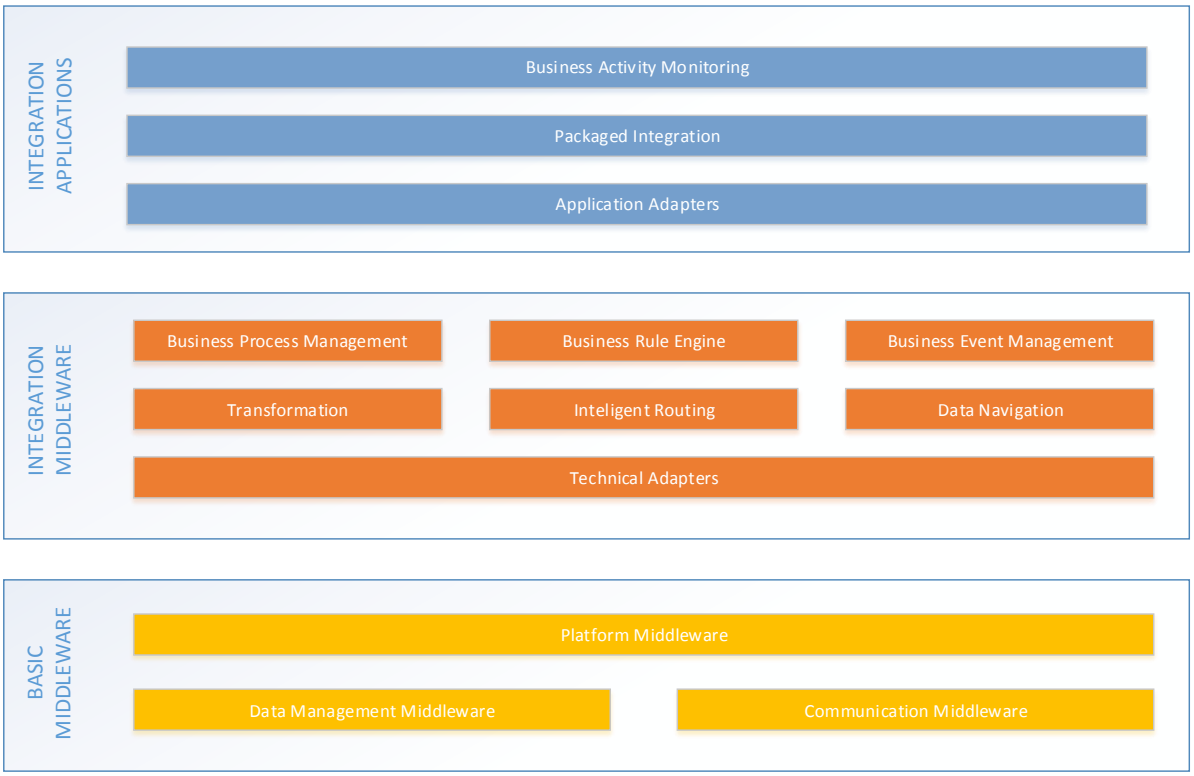
Obrázek 7: Úloha middleware (Krafzig, 2004)

Schéma výše (viz Obrázek 7) zobrazuje jednu z úloh middleware při komunikaci v distribuovaném prostředí více aplikací. V prostředí bez middleware by musela aplikace zařídit veškerou komunikaci sama včetně správy síťových paketů, odesílání dat, přijímání dat, kódování/dekódování dat a ošetření

chyb. S pomocí middleware je aplikace částečně od těchto nízko úrovňových úkonů odstíněna a vývojář se tak může plně soustředit na implementaci aplikační logiky.

Společnost Gartner (Lheureux, 2004) rozděluje celou oblast middleware do tří základních skupin dle možného užití (viz Obrázek 8):

- **Základní middleware (angl. Basic Middleware):** slouží pro potřeby integrace v rámci komponent jedné aplikace nebo více různých zpravidla sourodých aplikací;
- **Integrační middleware (angl. Integration Middleware):** slouží pro potřeby integrace naprosto nesourodých aplikací (od různých dodavatelů, od různých firem atd.). Zpravidla v sobě obsahují prvky základního middleware, ale navíc umožnují vyřešit technologické problémy pramenící z heterogenních aplikačních prostředí;
- **Integrace aplikací (angl. Integration Applications):** slouží pro potřeby integrace aplikací (viz Kapitola 3.2). Tato oblast je obvykle velice úzce navázaná na podnikové procesy.



Obrázek 8: Rozdělení middleware (Lheureux, 2004; upraveno autorem)

Z uvedeného je zřejmé, že middleware prostředky jsou klíčovou součástí integrace služeb a jsou proto klíčovou komponentou většiny současných přístupů a řešení integrace v IT.

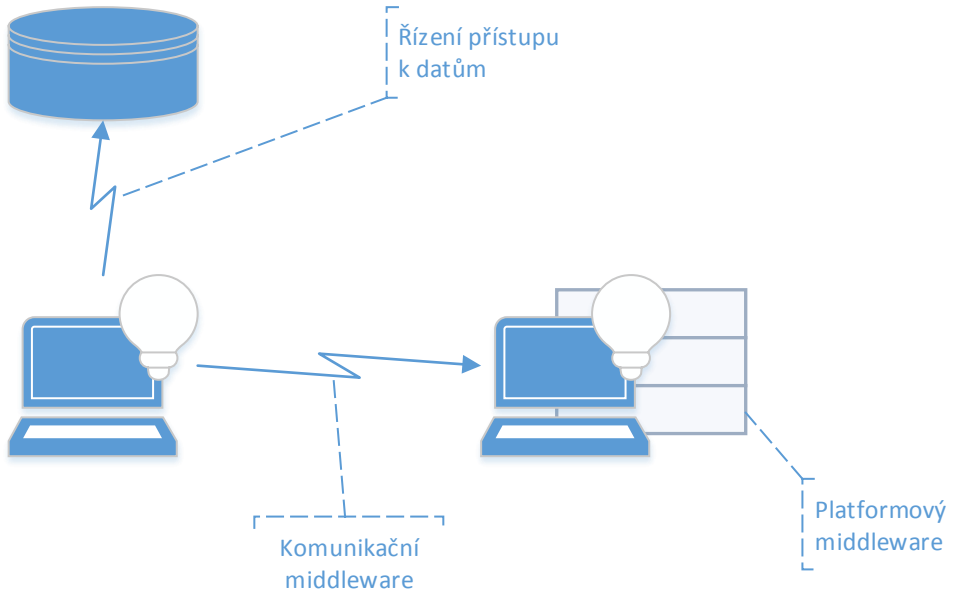
Základní middleware je nejvýznamnější skupinou middleware řešení a *integrační middleware* je velice úzce svázán s tématem této práce. Z těchto důvodů popíši obě oblasti v následujících kapitolách detailně.

3.2.1 Základní middleware

Základní middleware lze rozdělit do následujících skupin (Gála, 2009):

- **Komunikační middleware (angl. Communication Middleware)** umožňující vzájemnou komunikaci programů, ať již součástí jedné aplikace nebo aplikací různých;
- **Prostředky řízení přístupu k datům (angl. Data Management Middleware),** které zajištují přístup programů k datům uloženým v různých datových zdrojích;
- **Platformový middleware (angl. Platform Middleware),** který vytváří vhodné provozní prostředí s množinou obecně použitelných služeb.

Obrázek 9 znázorňuje, v jakých momentech životního cyklu aplikace můžeme nasadit konkrétní typ základního middleware.



Obrázek 9: Základní middleware a jeho použití (Lheureux, 2004; upraveno autorem)

Řízení přístupu k datům můžeme například použít v případech, kdy potřebujeme unifikovat přístup k datovým zdrojům skrze jednotné rozhraní (např. OLE DB, .NET DB Provider atd.). *Komunikační middleware* nám zase může pomoci vyřešit některé problémy komunikace v rámci heterogenních prostředí. A v neposlední řadě *platformový middleware* nám poskytuje provozní prostředí pro aplikace s množinou obecně použitelných služeb (Gála, 2009), např. .NET, Java atd.

3.2.2 Integrační middleware

Společnost Gartner (Lheureux, 2004; volně přeloženo autorem) dělí *integrační middleware* do následujících skupin dle jejich funkce:

- **Adaptéry (angl. Adapters)**: komponenty, které slouží pro vzájemné spojení různých aplikací nebo jiných middleware komponent;
- **Transformace (angl. Transformation):** komponenty, které provádějí syntaktické i sémantické konverze zpráv, dokumentů, souborů či jiných dat mezi různými aplikacemi;
- **Inteligentní směřování (angl. Intelligent Routing):** na rozdíl od *základního komunikačního middleware*, který umožnuje pouze základní komunikaci mezi daným odesilatelem a daným příjemcem, inteligentní směrovače přenášenou zprávu dekódují (angl. Parse) a na základě jejího obsahu a logických pravidel zvolí příjemce dané zprávy;
- **Datová navigace (angl. Data Navigation)**: komponenty, které se starají o sjednocení různých datových zdrojů;
- **Správa podnikových procesů (angl. Business Process Management)**: tato oblast middleware využívá nástroje podnikových procesů pro zajištění jejich správy a zpracování;
- **Vyhodnocení obchodních pravidel (angl. Business Rule Engines)**: zahrnuje různé způsoby pro zaznamenání a vyhodnocení různých obchodních pravidel v jednotné a srozumitelné formě;
- **Správa podnikových událostí (angl. Business Event Management)**: nástroje pro zpracování a záznam podnikových událostí.

Hranice mezi *integračním middleware* a *základním middleware* nejsou zcela ostré a není výjimkou, že komponenty z oblasti *integračního middleware* v sobě zahrnují některé funkce patřící spíše do oblasti *základního middleware* a naopak.

Pezzini (2011) ze společnosti Gartner připouští vysokou roztříštěnost tradičních řešení z oblasti middleware a identifikuje tržní poptávku po řešeních, které nabízí integrované funkce z různých middleware komponent do formy uceleného řešení. Tyto tradiční on-premise integrační platformy jsou pro mnoho malých a středních společností nedostupné z hlediska vysokých pořizovacích nákladů a požadovaných znalostí na implementaci.

Avšak s dynamickým rozvojem cloudových služeb souvisí také rozvoj ucelených integračních platforem provozovaných formou cloudu, tzv. iPaaS (angl. Integration Platform as a Service), které popíši v následující kapitole.

3.2.3 Integrační platforma jako služba (iPaaS)

Společnost Gartner (Pezzini, 2011; volně přeloženo autorem) popisuje *integrační platformu* jako: *„...způsob, poskytnout uživatelům ucelené integrační řešení pro integraci a správu (angl. Governance) různých aplikací takovým způsobem, aby společně spolupracovaly.“* Podobné řešení nenabízí pouze jednu z konkrétních middleware funkcí, ale nabízí ucelenou sadu funkcí tradičního middleware v rámci jednoho řešení.

Ruku v ruce s rozvojem cloudových služeb si Pezzini (2011) všímá rostoucí nabídky ucelených integračních platforem nabízených formou služeb (angl. iPaaS; Integration Platform as a Service) a předpokládá dynamický rozvoj zejména v této oblasti. Podobná řešení zpravidla nabízí následující funkce „pod jednou střechou“:

- Mapování protokolů (angl. Protocol Bridging);
- Přenos zpráv (angl. Messaging Transports);
- Transformace (angl. Transformation);
- Směřování (angl. Routing);
- Virtualizace služeb (angl. Service Virtualization);
- Adaptéry (angl. Adapters) apod.

Právě velmi dynamický rozvoj SaaS a zároveň absence nabídky řešení integrace do tradičních podnikových systémů považuje Pezzini (2011) za hnací motor rozvoje iPaaS. Paradoxně v této oblasti očekává adopci nejprve v rámci malých a středních společností, protože podobné společnosti obvykle nejsou zatížené nákladně budovanou IT on-premise infrastrukturou a mohou tak okamžitě čerpat výhody iPaaS.

Z nastíněných důvodů bude navržená metodika integrace SaaS zahrnovat částečně také iPaaS, ale nebude se omezovat pouze tímto směrem, protože pro mnoho společností bude iPaaS stále příliš nákladné a nedostupné řešení.

3.3 Metody integrace a způsoby komunikace

Tato kapitola je primárně zaměřená na popis různých způsobů komunikace mezi různými aplikacemi a představení různých metod integrace. Oblast integrace je postavena na propojování různých ICT komponent mezi sebou, a to se realizuje primárně prostřednictvím vzájemné digitální komunikace.

Z tohoto důvodu, abychom mohli hodnotit jakékoli integrační řešení, je nutné porozumět různým způsobům komunikace a omezením, která s sebou každý ze způsobů přináší.

3.3.1 Způsoby komunikace

Na úvod této kapitoly shrňme možné komplikace, se kterými se integrační úlohy musí vypořádat (volně zpracováno dle (Hohpe, 2004)):

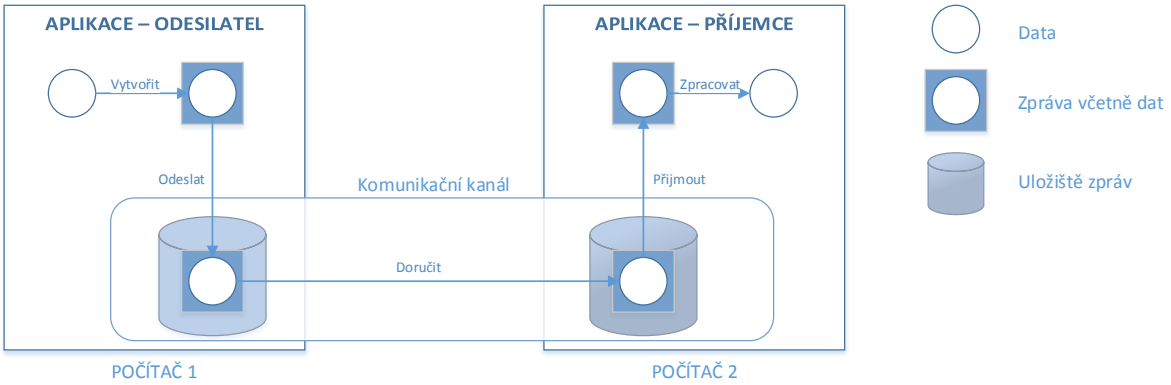
- **Komunikační sítě nejsou spolehlivé:** integrace se odehrává prostřednictvím komunikačních sítí, které ovšem nejsou stejně spolehlivé jako by se integrace odehrávala pouze v rámci jednoho počítače;
- **Komunikační sítě jsou pomalé;**
- **Jakékoli dvě různé aplikace jsou odlišné:** integrační úloha obvykle spojuje různorodé systémy a musí být schopna vypořádat se s různými formáty, různými prostředími, různými platformami atd.;
- **Změny jsou nevyhnutelné:** Aplikace se v průběhu času mění. Integrační řešení by mělo být navrženo dostatečně flexibilně, aby se přizpůsobilo změnám v aplikacích (komponentách), které spojuje.

Samotná komunikace se realizuje prostřednictvím *zprávy* (*volání*).

„Zpráva je sama o sobě určitý druh datové struktury (řetězec, číslo, objekt atd.). Může být interpretována jako data, nebo jako popis metody, která má být spuštěna u příjemce, nebo jako informace o události, která nastala na straně odesilatele.“ (Hohpe, 2004; volně přeloženo autorem)

Poznámka: jak je v textu výše naznačeno, považuji pojem *zpráva* i *volání* za stejný typ. Někteří autoři, např. (Hohpe, 2004), důsledně rozlišují oba pojmy. Pojem *zpráva* používají výhradně ve spojení asynchronní komunikace a *integračního middleware*, zatímco pojem *volání* používají ve spojení synchronního volání vzdálených procedur. Pro účely této práce nebudu tyto pojmy rozlišovat, protože rozdíly nejsou vzhledem k cílům práce podstatné.

Schéma procesu přenosu zprávy je zobrazeno níže (viz Obrázek 10).



Obrázek 10: Přenos zprávy (Hohpe, 2004)

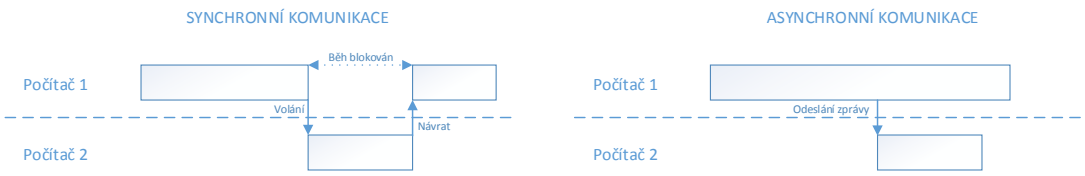
Principiálně rozeznáváme následující stavy v životním cyklu přenosu zprávy (volně zpracováno dle (Hohpe, 2004)):

- Odesilatel vytvoří zprávu a vloží do ní data. V tomto kroku je také důležité, že zpráva i data jsou zakódována ve formátu, ve kterém máme jistotu, že zprávu příjemce bude schopen úspěšně dekódovat;
- Odesilatel odesílá data prostřednictvím *komunikačního kanálu*. Komunikačním kanálem rozumíme počítačovou síť nebo také mezičlánek ve formě middleware (angl. Message-Oriented Middleware, dále MOM);
- V případě použití MOM může být v rámci komunikačního kanálu k dispozici navíc uložiště zpráv a fronta zpráv (angl. Queue);
- Následuje doručení zprávy do cílového počítače a cílové aplikace;
- Přijetí zprávy probíhá od operačního systému nebo od MOM a jeho uložišti zpráv;
- Samotnou zprávu je nutné interpretovat a získaná data následně zpracovat, ale ještě předtím je musí cílová aplikace dekódovat;
- Případně může cílová aplikace odpovědět a zpět poslat odpovědní zprávu stejným způsobem.

Ne nutně všechny kroky musí vlastnoručně řešit vývojář, ale v každém z kroků mu mohou výrazně usnadnit práci middleware komponenty (viz Kapitola 3.2). Avšak je nutné si uvědomit, jaká režie s odesláním jedné zprávy vzniká a tomu přizpůsobit návrh integrace.

Obecně rozeznáváme dva základní typy komunikace (viz Obrázek 11):

- **Synchronní:** u této komunikace odesílající aplikace po odeslání zprávy čeká do doby, než dostane odpověď od příjemce (s daty nebo informací o úspěšném doručení);
- **Asynchronní:** naopak u tohoto typu komunikace aplikace předá zprávu komunikačnímu kanálu (zpravidla MOM) a okamžitě pokračuje ve své další činnosti.



Obrázek 11: Synchronní a asynchronní komunikace (Hohpe, 2004)

Mezi hlavní přednosti *synchronní komunikace* patří mnohem jednodušší implementace na straně odesilatele i příjemce, avšak na druhou stranu můžeme vnímat výraznou latenci na straně odesilatele způsobenou čekáním na zpracováním zprávy příjemcem. *Asynchronní komunikace* poskytuje nespočet výhod, avšak jsou vykoupeny mnohonásobně náročnější implementací vyžadující zpravidla zapojení middleware, který se stará o správu doručovacího cyklu. Pro úplnost shrnuji výhody i nevýhody obou typů komunikace v následující tabulce (viz Tabulka 3).

Tabulka 3: Výhody a nevýhody synchronní a asynchronní komunikace (částečně zpracováno dle (Hohpe, 2004); upraveno autorem)

SYNCHRONNÍ KOMUNIKACE	
Výhody	Nevýhody
• Jednoduchost implementace • Je vždy zcela zřejmý okamžik, kdy druhá strana obdržela zprávu • Spolehlivost doručení je ihned ověřitelná • Synchronní model zpravidla funguje na všechny typy aplikací (na některé hůře, na některé lépe, ale funguje vždy)	• Latence v běhu aplikace • Riziko celkové nedostupnosti příjemce a tím omezení funkčnosti celé aplikace • Riziko přetížení příjemce v případě použití vysokého počtu odesilatelů
ASYNCHRONNÍ KOMUNIKACE	
Výhody	Nevýhody
• Žádná latence na straně odesilatele • Nižší riziko přetížení cílového stroje (MOM může škálovat intenzitu doručení) • Vyšší spolehlivost, ochrana proti výpadkům sítě (MOM doručí zprávy po obnovení činnosti i se zpožděním; mezitím zůstanou uloženy) • MOM může sloužit jako spolehlivý mediátor (možnosti transformace z/do různých platforem) • Díky použití MOM, jsou příjemce i odesilatel zcela nezávislé komponenty a mohou tak pracovat ve vlastním rytmu	• Výrazně vyšší složitost implementace • Zpravidla vyžaduje zapojení middleware (MOM) • Ne všechny cílové platformy mohou podporovat „Messaging“ • Není zřejmý okamžik, kdy došlo k doručení zprávy • Asynchronní model není vhodný pro všechny typy aplikací (na některé typy aplikací je vyloženě nefunkční) • Riziko uvěznění (angl. Lock-in) v použité komunikační platformě konkrétního dodavatele

Aby si komunikující komponenty mohly navzájem smysluplně vyměňovat data, je nutné, aby se nejen dokázaly správně „domluvit“, ale by informace navzájem správně interpretovaly. V tomto bodě se mohou obecně použít dva principy (volně zpracováno dle (Hohpe, 2004)):

- **Spojení těsnou vazbou (angl. Tightly coupled):** těsnou vazbou se nazývá situace, kdy komunikující komponenty o sobě musí znát spoustu dodatečných informací, aby spolu mohly úspěšně komunikovat (formát dat, adresa, počet parametrů atd.);
- **Spojení volnou vazbou (angl. Loosely coupled):** opakem těsné vazby je vazba volná, která se snaží minimalizovat nutnost dodatečných informací nutných pro úspěšnou komunikaci.

Těsnou vazbu můžeme demonstrovat pomocí následujícího příkladu. Představme si situaci, kdy si dvě komunikující komponenty vyměňují krátké zprávy v datové struktuře: odesilatel (řetězec) a obsah (řetězec). Komponenty ví, na kterém TCPIP portu mají očekávat data a v jakém formátu. Takže jedna z komponent zkontaktuje daný port a pomocí TCPIP pošle data jako kolekci bajtů převedenou z řetězce ve formátu: „ODESILATEL####ZPRAVA”. Druhá z komponent tuto kolekci bajtů přijme a řetězec dekóduje. Avšak pro úspěšné dekódování musí znát spoustu dodatečných informací, například: na jakém portu očekávat data, jakým způsobem pozná, že právě daná aplikace má oprávnění posílat data, v jaké znakové sadě bude text kódován (kolik bajtů připadá na jeden znak), jaký oddělovač je použit pro oddělení odesilatele a obsahu, jakým způsobem pozná, že řetězec byl obdržen celý a nic nechybí atd. Bez těchto informací není možné zprávu obdržet a smysluplně zpracovat. Navíc tyto informace musí být známé oběma stranám. Tuto situaci můžeme označit za *těsnou vazbu*.

Shrnuji několik možných vlastností, které určují *typ vazby* (Hohpe, 2004; volně přeloženo autorem):

- **Použitá platforma:** musíme znát použitou platformu, abychom správně interpretovali bajtovou reprezentaci čísel, řetězců atd. (v komunikaci závislé na bajtech není lhostejné, pokud je např. *integer* reprezentován 16bity nebo 32bity;
- **Lokace:** musíme znát adresu, port atd.;
- **Dostupnost:** v době komunikace musí být obě komponenty „na příjmu“;
- **Datový formát:** musíme znát pořadí parametrů atd.

Většinou jsou všechny tyto parametry nadefinované „natvrdo“ v kódu komponent, a proto z hlediska udržitelnosti a znovu použitelnosti se jedná o velmi nešťastné řešení. Z těchto důvodu se v případě integrace preferuje právě *volná vazba*. Ale jak to tak bývá, „něco za něco“. Volná vazba řeší všechny zmíněné nevýhody, ale za cenu mnohem složitější implementace a zpravidla také nutností použít *middleware* komponenty.

3.3.2 Metody integrace

V rámci této kapitoly zmíním možné metody samotné integrace (volně zpracováno dle (Hohpe, 2004)) a popíši jejich základní vlastnosti vzhledem k výše uvedeným způsobům komunikace:

- **Přenos souborů:** Jedna aplikace zapíše data do souboru na „domluveném“ místě a druhá aplikace data přečte. Typická ukázka těsné vazby: aplikace se musí předem domluvit na adrese souboru, formátu dat a časování čtení/zápis. Tuto metodu je možné použít pro asynchronní typ komunikace. U asynchronní komunikace musí navíc být precizně zajištěno správné časování, protože se soubory nelze současně provádět čtení a zápis;
- **Sdílené databáze:** více aplikací sdílí společnou databázi (např. SQL server). Opět příklad těsné vazby a asynchronní komunikace. Například SQL databáze poskytují účinný prostředek pro uchování dvou dimenzionálních dat (řádek, sloupec), ale složitější datové struktury (např. strom hierarchie objektů apod.) se musí speciálně formátovat a toto formátování musí být známé oběma stranám;
- **Použití vzdálených procedur či metod (angl. Remote Procedure Invocation nebo Remoting) / použití vzdáleného API:** aplikace poskytuje svou funkcionalitu prostřednictvím vzdálených metod či API a jiné aplikace je mohou konzumovat. Tento model bývá obvykle poskytován formou rozhraní služeb. V této oblasti můžeme sledovat velkou snahu o dosažení ideální volné vazby pomocí různých standardů např. webové služby (angl. Web Services) atd. Tento typ komunikace je typicky synchronní;
- **Sběrnice zpráv (angl. Messaging) / Zprostředkující služba**: jedná se o komunikaci s použitím middleware (MOM), který poskytuje *sběrnici zpráv* a správu celého doručovacího cyklu zpráv. Jde o příklad volné vazby a typický zástupce asynchronní komunikace.

Výše uvedené metody integrace je možné shrnout společně s odpovídajícím způsobem komunikace (viz Tabulka 4).

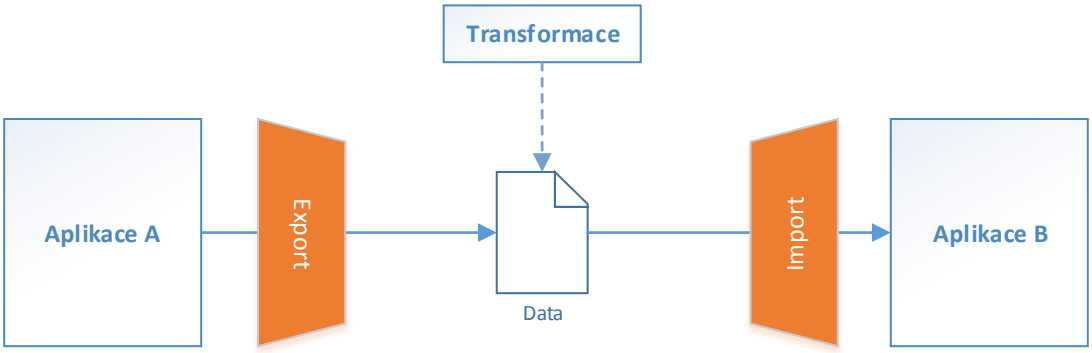
Tabulka 4: Metody integrace a odpovídající způsoby komunikace

	Asynchronní komunikace	Synchronní komunikace	Spojení těsnou vazbou	Spojení volnou vazbou
Přenos souborů	ANO	NE	ANO	NE
Sdílené databáze	ANO	NE	ANO	NE
Remoting / API	NE	ANO	ANO	ANO

	Asynchronní komunikace	Synchronní komunikace	Spojení těsnou vazbou	Spojení volnou vazbou
Sběrnice zpráv / zprostředkující služba	ANO	ANO	ANO	ANO

Všechny uvedené způsoby integrace sledují stejný cíl, ale nabízí rozdílné cesty. V praxi se můžeme setkat s různou kombinací těchto přístupů, protože není smyslem preferovat pouze jeden konkrétní způsob, ale skutečným mistrovstvím je zvolit takový způsob, který nejlépe vyhovuje dané situaci a požadavkům. V následujících kapitolách představím jednotlivé metody integrace ve větším detailu.

3.3.2.1 Přenos souborů



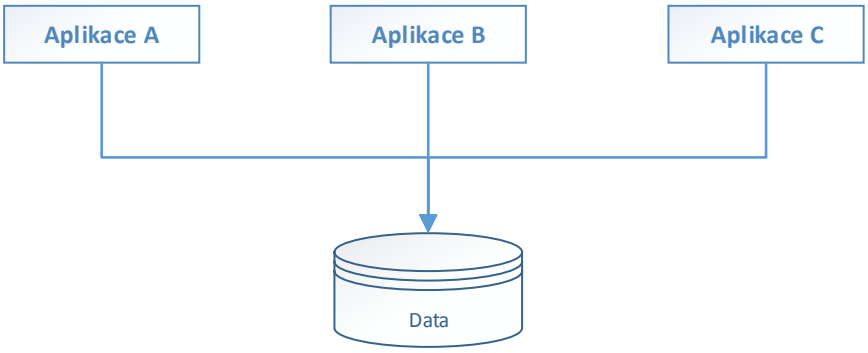
Obrázek 12: Princip přenosu souborů (Hohpe, 2004)

Systém přenosu souborů je nejstarší a nejzákladnější integrační metoda používaná už desítky let. I přes svou jednoduchost má své nesporné výhody. Jednou z výhod je, že integrátor nemusí vůbec vědět, jak která aplikace vnitřně funguje. Jinými slovy, můžete bez problému vyměnit jednu aplikaci za jinou, dokud poskytují konzistentní výstup. Výhodou také je, že můžete se souborem „na cestě“ provádět různé transformace v případě potřeby (viz Obrázek 12).

Tento typ komunikace je vhodný pro časově méně kritické aplikace, protože prodleva mezi výměnou informací je zpravidla delší, a proto není vhodná pro časté změny dat. Navíc programátor musí vyřešit časování přístupu k souborům a správu souborů (mazání starších dat atd.). U této metody integrace není formát dat definován, a musí se navrhnout ad-hoc (např. XML atd.).

Tato metoda patří mezi asynchronní a slouží zpravidla pro výměnu dat (ne metod).

3.3.2.2 Sdílené databáze



Obrázek 13: Princip sdílené databáze (Hohpe, 2004)

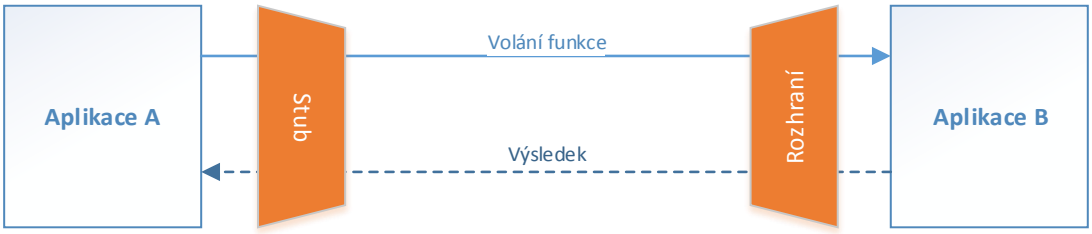
V případě, že integrované aplikace využívají databázový stroj a máme k němu přístup, můžeme využít tuto integrační metodu, která spočívá v integraci dat v rámci použitého databázového stroje (viz Obrázek 13). Tato metoda je velice populární a často využívaná pro *integraci dat*.

Výhodou je vysoká rychlost přesunu dat a pestré možnosti transformací. Formát dat je unifikovaný jazykem SQL, avšak zpravidla nebývá unifikovaná databázová struktura jednotlivých databází. Každá aplikace zpravidla vyžaduje vlastní strukturu tabulek a není příliš možné tuto strukturu měnit. V tom případě můžeme využít systém „triggerů“, který můžeme využít pro synchronizaci dat do jednotné centrální databáze (např. prostřednictvím Kanonického schéma).

Existují různé metody datové integrace: konsolidace, replikace a virtualizace (Gála, 2009).

V případě, že jednotlivé aplikace využívají stejnou databázi i tabulky, stále ještě nemáme vyhráno a musíme vyřešit časování zámku řádků. Jedná se o typ asynchronního přenosu.

3.3.2.3 Použití vzdálených procedur/metod



Obrázek 14: Princip vzdálených procedur/metod (Hohpe, 2008)

Integrace metodou vzdálených procedur je koncept, který vychází z dobře známého konceptu volání lokálních procedur (DLL), avšak aplikovaný do distribuovaného prostředí (angl. Remoting).

Princip spočívá v tom, že jedna aplikace zavolá vzdálenou proceduru/metodu implementovanou v druhé aplikaci a zveřejněnou prostřednictvím rozhraní (angl. Interface). Aplikace danou metodu

provede a výsledek vrátí původní aplikaci (viz Obrázek 14). Výhodou této metody je zapouzdřenost cílové aplikace (viz Kapitola 2.5). Tato metoda prošla v posledních letech bouřlivým vývojem, přes technologie jako jsou DCOM či CORBA až po současný standard webových služeb (angl. Web Services). Díky tomu je k dispozici pestrá škála standardů a podpora v různých integračních platformách (.NET, Java atd.). Proto má integrátor usnadněnou situaci z hlediska volby formátu dat.

Tato metoda je synchronní a lze ji použít pro integraci metod i dat. Tuto vlastnost lze částečně obejít pomocí více vláknovému návrhu aplikace (pro odeslání/získání dat se stará extra dedikované vlákno aplikace). Avšak musíte zajistit, aby obě aplikace byly v daném okamžiku funkční, jinak integrace selže. Metodu lze řadit mezi metody s těsnější vazbou, protože rozhraní cílové aplikace musí být známé, ale *zapouzdřenost* poskytuje flexibilitu pro případné změny.

Tato metoda a její konkrétní implementace prostřednictvím webových služeb byla stěžejní při zpracování mé bakalářské práce, ve které celý koncept popisuji ve velkém detailu (Žižka, 2009).

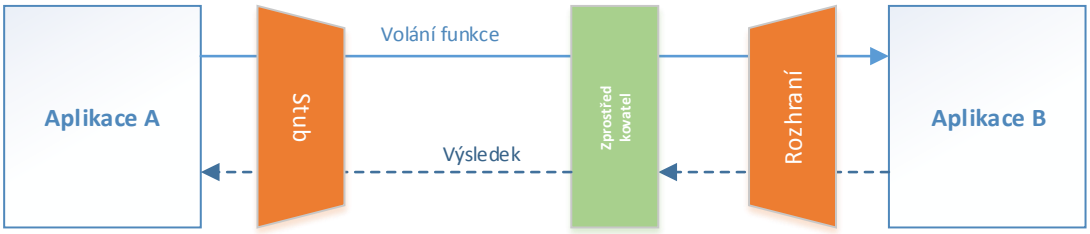
3.3.2.4 Sběrnice zpráv / Zprostředkující služba

Lze říci, že nevýhody předchozí metody daly vzniknout této metodě, která je úspěšně odstraňuje či minimalizuje. Princip této metody spočívá v tom, že přenos zprávy je svěřen mezičlánku (middleware / zprostředkující služba), který se stará o doručení zpráv. Tato metoda má spoustu výhod.

Obecně rozeznáváme dva základní koncepty výměny informací mezi komponentami (angl. Message Exchange Patterns (Erl, 2005)):

- **Koncept (angl. Request-Response):** způsob *synchronní komunikace*, při kterém zdrojová komponenta zastaví svou činnost a čeká na odpověď od cílové komponenty;
- **Koncept (angl. Fire and Forget):** způsob *asynchronní komunikace*, při které se o doručení stará zprostředkovatel a zdrojová komponenta může dál vykonávat svou činnost.

Koncept Request-Response je nejjednodušší a jedná se o standardní *point-to-point* typ komunikace, kdy každá komunikující dvojice vytváří komunikující kanál. Při této komunikaci se může používat tzv. *zprostředkovatel služeb* (angl. Intermediary services (Erl, 2005)). Jedná se o předěl mezi *použitím vzdálených procedur bez zprostředkovatele* (viz Kapitola 3.3.2.3) a *použitím sběrnice zpráv* (viz Obrázek 15).



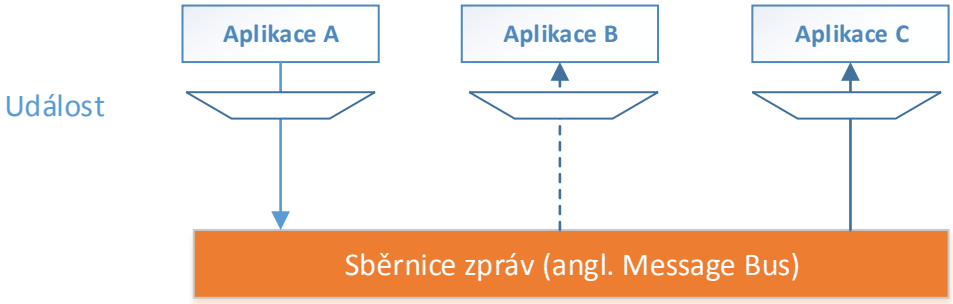
Obrázek 15: Využití zprostředkující služby při komunikaci (Erl, 2005; upraveno autorem)

Zprostředkovatel služeb může mít několik základních funkcí (volně zpracováno dle (Erl, 2005)):

- **Modifikace zprávy:** v případě potřeby modifikuje zprávu a modifikovanou zprávu posílá dále;
- **Předání zprávy:** zprostředkovatel obsahuje logiku, která zajistí doručení zprávy cílové aplikaci nebo více aplikací.

Obě zmíněné funkce se mohou vzájemně kombinovat. V tomto základním případě se stále ještě může jednat o formu *synchronní* i *asynchronní* komunikace. Záleží, jakým způsobem je zprostředkovatel navržen. Může se jednat o zprostředkovatele *aktivního*, který modifikuje posílané zprávy nebo *pasivního*, který zprávy pouze doručuje.

Koncept *Fire and Forget* je z podstaty asynchronní a stojí základem celé řady dalších složitějších variací jako například multi-cast, broadcast, publish and subscribe atd. Obecně lze říci, že koncept *Fire and forget* úspěšně mění komunikaci *point-to-point* na *point-to-many*. Při tomto konceptu může mít zprostředkovatel celou řadu dalších funkcí včetně například *vyvažování zátěže (angl. Load Balancing)*.



Obrázek 16: Princip sběrnice zpráv (Hohpe, 2004)

Nejvyšším stupněm asynchronní komunikace je použití tzv. *sběrnice zpráv (angl. Message Bus) ve formě middleware (MOM)*, který se stará kompletně o správu doručovacího cyklu zpráv. Díky tomu, je metoda imunní vůči výpadkům spojení, nefunkční druhé aplikaci atd. V případě potřeby se middleware stará o opakované doručení nebo uchování zpráv ve frontě až do bez problémového

doručení. Tato metoda navíc umožnuje využít konceptu „hub and spoke“, který minimalizuje tzv. „špagetovou integraci“ (Gála, 2009), která vzniká při použití *point-to-point* způsobu komunikace.

3.3.3 Shrnutí

V této kapitole jsem detailně představil různé metody integrace a pojmenoval hlavní vlastnosti a faktory, které ovlivňují volbu dané metody v dané situaci. Volbu konkrétní metody ovlivňují zejména následující faktory, které bychom měli zvážit při návrhu řešení konkrétní integrační úlohy (volně zpracováno dle Hohpe, 2004):

- **Stupeň vazby:** volba těsné vazby může výrazně zkrátit délku vývoje, avšak může velice negativně ovlivnit její následnou údržbu či možnost implementace případných změn. V této otázce musíme mít na paměti, že v případě těsné vazby je nutné zafixovat informace, které o sobě integrované aplikace předpokládají. V opačném případě komunikace selže;
- **Nutnost změn:** měli bychom předpokládat, že časem bude nutné provádět různé změny jak do procesu integrace, tak do obou komunikujících komponent. Proto by navržené řešení mělo být jednoduché natolik, aby aplikace změn nebyla složitá nebo zcela vyloučena;
- **Volba technologie:** různé způsoby integrace vyžadují variabilní sadu použitých ICT technologií. Některé technologie mohou být drahé, některé vedou k závislosti na konkrétním dodavateli. Na druhou stranu, integrace vyvíjena na zelené louce může být zdlouhavá;
- **Datový formát:** komunikující komponenty se musí předem domluvit na formátu výměny dat. Avšak měli bychom opět zvolit takový formát, který nám do budoucna umožní aplikaci změn či rozšiřitelnosti;
- **Datová prodleva:** jakmile uvažujeme v integrační rovině, vždy musíme předpokládat, že integrace s sebou přináší určitou prodlevu mezi bodem, kdy se jedna komponenta rozhodne data sdílet a bodem, kdy druhá komponenta data skutečně získá. Zvolená integrační metoda tento faktor výrazně ovlivňuje. Volba také závisí na kontextu konkrétní úlohy, avšak je nutné mít na paměti, že čím delší je prodleva mezi synchronizací dat, tím se zvyšuje riziko konfliktů (nesouladu) mezi daty;
- **Body integrace:** volbu metody v neposlední řadě určuje předmět integrace, zda integrujeme funkcionalitu nebo pouze data;
- **Vzdálená komunikace:** při volbě je nutné zvážit také negativní vlastnosti vzdálené komunikace a jejich minimalizaci (rychlost, synchronní/asynchronní komunikace, náchylnost k chybám, identifikace chyb apod.);
- **Spolehlivost**: integrační úloha vyžaduje zpravidla fungující obě komponenty, jedna bez druhé obvykle nefunguje, proto bychom měli s tímto stavem počítat a navrhnout takové řešení, které bude pokud možno imunní proti výpadkům komunikace.

3.4 Integrace podnikových aplikací

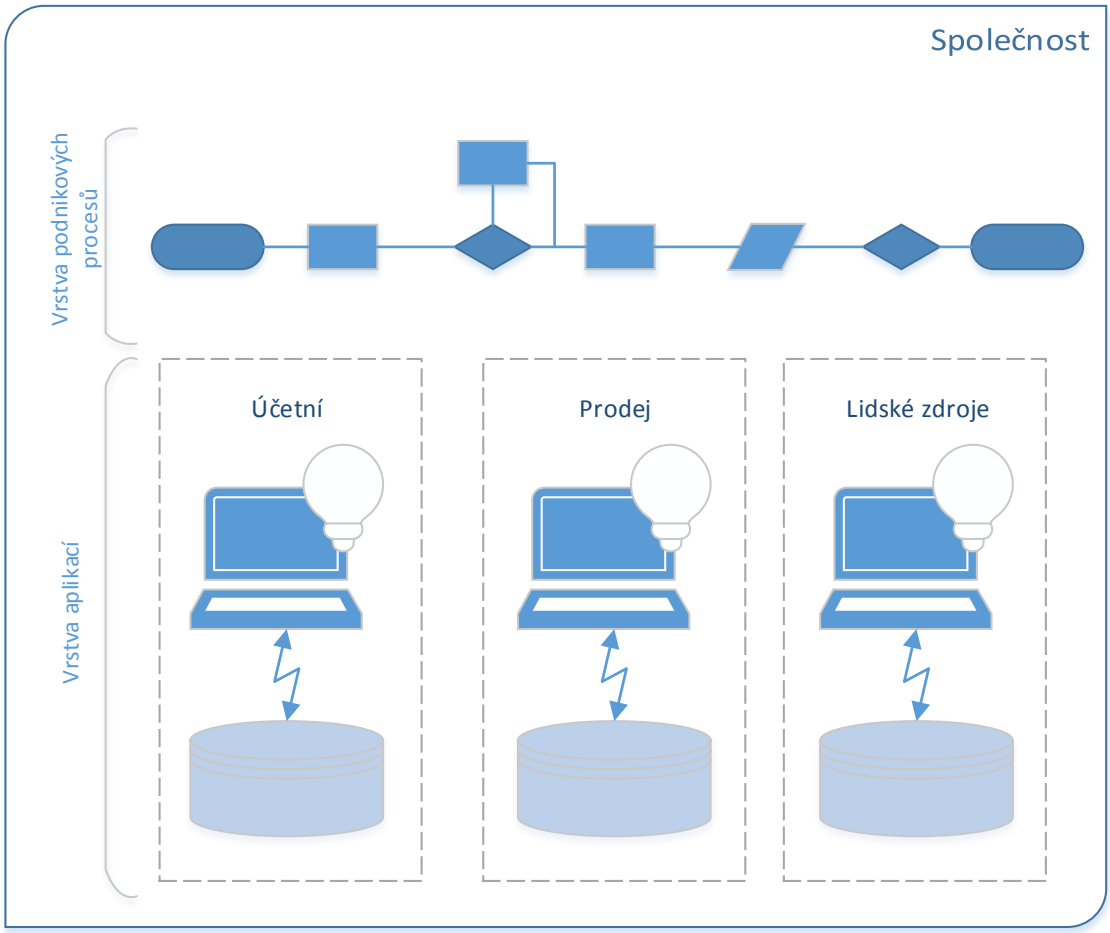
Nyní se podíváme na firmu více z globální perspektivy. Navážeme na kapitolu 3.1 a rozvedeme pojem *izolované aplikační ostrůvky* (angl. Application Silos (Feuerlicht, 2008, kap. 7.1) nebo take angl. Stovepipe Applications (Linthicum, 2000)).

Ve většině společnostech vznikají nebo se nakupují různé aplikace a nástroje na podporu podnikových procesů, které ovšem zůstávají úzce navázány na daný podnikový proces s minimální možnosti sdílení dat s okolím mimo daný proces. Dané aplikace obvykle řeší úzký okruh úkolů související výhradně s daným procesem, např. účetnictví, řízení lidských zdrojů atd. (Linthicum, 2000).

Schéma níže (viz Obrázek 17) ilustruje, že dané aplikace se vyznačují vysokou izolovaností, avšak to jde proti smyslu informačního systému podniku. Ve vlastním zájmu společností je sdílet data nebo celé procesy všude tam, kde to je žádoucí a pokud to je nutné, tak i mezi různými podnikovými odděleními.

---

*Cílem **integrace podnikových aplikací** (angl. Enterprise Application Integration, EAI) je provázání různých podnikových aplikací uvnitř společnosti takovým způsobem, že se tváří jako jeden monolitický systém.*
(Linthicum, 2000; volně přeloženo autorem)



Obrázek 17: Izolované aplikační ostrůvky (Feuerlicht, 2008), (Linthicum, 2000; upraveno autorem), částečně (Erl, 2005)

Existují rozličné důvody tohoto „izolovaného“ stavu (volně zpracováno dle (Linthicum, 2000)):

- **Zastaralé aplikace (angl. Legacy Applications):** aplikace se nakupují postupně v různých časových okamžicích a postupem času se v každé společnosti oddělí skupina technologií, které již nedosahují současné úrovně, avšak jsou stále používané z důvodu kompatibility dat atd. Ale z důvodu jejich zastaralosti je velice komplikovaná propojenost s ostatními aplikacemi podniku;
- **Nákup hotových jednoúčelových aplikací:** v rámci různých oddělení se časem mohou vyskytnout specifické jedno úlohové aplikace, jejichž primárním cílem je vyřešit velice specifické problémy daných procesů, avšak při prvotním nákupu se zájmy společnosti jako celku nezohledňovaly. Ale tento bod nemusí znamenat pouze negativní konotaci. Nákupem podobných aplikací získá každý vlastník procesu možnost koupit vždy tu nejlepší možnou variantu v dané oblasti (nejlepší účetnictví, nejlepší CRM atd.)(angl. Best of Breed);
- **Hranice procesů nemusí být úplně ostré:** vlastníci některých procesů časem potřebovali vyřešit některé okrajové úkoly, a tak řešení svěřili svým doménovým aplikací. Ale pro jiné

procesy mohou být podobné činností primární funkcí, ale výsledkem je, že v podstatě podobná data jsou zaznamenávána odlišně na různých místech v organizaci;

- **Nákup a implementace velkých integrovaných SW řešení je drahé:** společnosti typu SAP nebo PeopleSoft tyto problémy řeší nabídkou komplexních a všeobjímajících ERP řešení, avšak náklady jejich pořízení a implementace jsou natolik vysoké, že jsou tyto systémy pro mnoho společností nedosažitelné;
- **Množství protichůdných požadavků různých oddělení:** zájmy konkrétního oddělení se mohou částečně míjet s vizí společnosti jako celku a tento fakt může v budoucnu způsobit nemalé ekonomické ztráty;
- **Omezený rozpočet:** fixní náklady IT projektů mohou být nižší v případě malého měřítka (oddělení), avšak dané IT projekty se mohou značně prodražit v budoucnu při nutnosti dané řešení upravit pro potřeby celé společnosti;
- **Absence centrální vize IT:** může se jednat obecně o důsledek absence centrální vize řízení společnosti ve vztahu k IT.

Řešení naznačeného problému má dva úhly pohledu: *koncepční (strategický)* a *technologický*.

Koncepční řešení spočívá v uvědomění si tohoto stavu na nejvyšší podnikové úrovni a ustanovení funkce *podnikového architekta* (angl. Enterprise Architect). Podnikový architekt má za úkol sledovat podnikové procesy a jejich informační potřeby a v závislosti na těchto vstupech provést analýzu a vytvořit tzv. *centrální podnikovou architekturu*. Cílem této architektury je poskytnout externí zpětnou vazbu na podnikové procesy a jejich implementaci, a doporučení konkrétních ICT technologií, které umožní sdílení dat všude tam, kde to je potřeba a kde si to daný proces vyžaduje (Linthicum, 2000; volně přeloženo autorem).

Krafzig (2004) dodává, že podniková architektura se zpravidla musí vyrovnat s celou řadou protichůdných požadavků od celé řady zainteresovaných subjektů (angl. Stakeholders). Proto podnikový architekt na sebe přebírá roli externího „hybatele“ a kontrolora v jedné osobě. Neustále musí hodnotit individuální IT projekty a jejich dopad na globální strategický plán podniku a kontinuálně vyvažovat pestrou škálu různých IT požadavků. Navíc cíle podnikové architektury se mění v čase podle toho, jakým způsobem se transformuje trh, na kterém daná společnost působí. Z tohoto důvodu je žádoucí, aby podniková architektura byla odolná vůči změnám a dostatečně flexibilní.

Podniková architektura či role podnikového architekta není primárním cílem této práce, pouze jsem chtěl akcentovat skutečnost, že žádné ICT řešení bychom neměli hodnotit pouze z hlediska technologického, ale měli bychom zohledňovat všechny souvislosti v rámci celkového fungování organizace.

Technologické řešení se opírá o využití různých integračních metod (viz Kapitola 3.3.2) a jejich nasazení na různých úrovních (Linthicum, 2000), tzv. *bodů integrace*:

- **Integrace na úrovni dat:** soustřeďuje se na výměnu dat v rámci datových uložišť různých aplikací. Velice rozšířená podoba integrace z důvodu značného využití oddělených SQL uložišť a možnosti nezávislého přístupu k nim. Tato oblast se někdy také označuje jako *replikace dat* (angl. Data Replication).;
- **Integrace na úrovni aplikačních rozhraní:** aplikace mohou poskytovat programovatelná rozhraní (API), která mohou sloužit k automatizovanému přístupu k dané aplikaci. Toto rozhraní může zpřístupnit data, ale také celý podnikový proces;
- **Integrace na úrovni metod/objektů:** různé podnikové procesy mohou sdílet rozličné vlastní metody a jejich použitím umožnit sdílení prostředků v rámci společnosti. Tato forma bývá označována jako znovu použitelnost programového kódu. Tato oblast je někdy označována jako *sdílení podnikových funkcí* (angl. Shared business functions) a může být poskytována formou služeb, např. (Hohpe, 2004);
- **Integrace na úrovni uživatelského rozhraní (dále GUI):** tato forma bývá označována jako velice primitivní, avšak mnohdy stále potřebná v případech, kdy integrace na jiných úrovních není možná. Spočívá v konsolidaci různých aplikací prostřednictvím jejich uživatelských rozhraní.

Z hlediska konkrétního hodnocení podnikové architektury můžeme použít následující dimenze (Krafzig, 2004):

- **Jednoduchost:** podniková architektura musí být dostatečně jednoduchá, aby se její komponenty daly popsat všem zainteresovaným osobám. Skupina těchto lidí může být značně rozdílná včetně rozdílné profesionální zkušeností či zcela bez technických znalostí. Avšak celá tato skupina hodnotí dopady podnikové architektury jako celku;
- **Flexibilita a udržitelnost:** každá organizace prochází v čase změnami, kterým se musí přizpůsobit. Proto také podniková architektura musí být flexibilní a měla by umožnovat změny v budoucnu. Architektura by měla počítat s flexibilním rozmístěním jednotlivých komponent;
- **Znovu použitelnost:** mezi vývojáři se o znovu použitelnosti hotových bloků kódů hovoří již dlouhá desetiletí a podobný princip bychom měli zohledňovat také v podnikové architektuře. Avšak v podnikové architektuře se tento pojem spíše spojuje se znovu použitelnosti dat uvnitř podniku. Na druhou stranou musíme vždy zvážit, zda znovu použitelnost má v tom kterém případě smysl a náklady jejího zavedení nepřevýší přínosy;

- **Oddělení funkcionality a technologie:** podniková architektura by neměla být spjatá s žádnou konkrétní IT technologií. Životní cyklus podnikové architektury by měl být delší než inovační cyklus IT technologií, které podnikovou architekturu podporují. Jinými slovy, podniková architektura by měla být dostatečně odolná vůči změnám v otázce podpůrných technologií.

Uvedené dimenze mohou v neposlední řadě sloužit také jako vodítko při hodnocení každé jednotlivé integrační úlohy a jejího řešení. Proto můžeme uvedené dimenze použít i při hodnocení různých metodik integrace.

4 Metodika integrace služeb SaaS

Tato kapitola představuje existující metodiky pro budování informačního systému podniku, které mohou být použity pro popsání procesu integrace služeb SaaS nebo případně mohou sloužit jako základ pro vytvoření metodiky nové.

Při hodnocení daných metodik použijeme kritéria a vlastnosti integrace diskutované v kapitole 2 a obzvláště v kapitole 3. Jinými slovy, budeme hodnotit, zda dané metodiky dostatečně zohledňují různé *metody integrace* (viz Kapitola 3.3), *specifika integrace aplikací* (viz Kapitola 3.4) a také respektují technologické vlastnosti služeb typu SaaS (viz Kapitola 2.6.1).

Pro přehlednost shrnuji klíčové vlastností různých *způsobů integrace,* které mají klíčový vliv na jejich volbu:

- **Rozlišení způsobů komunikace:** synchronní a asynchronní;
- **Rozlišení stupně vazby:** těsná a volná;
- **Rozlišení různých způsobů integrace:** přenos souborů, sdílené databáze, použití vzdálených procedur či metod (API) nebo *sběrnice zpráv* (angl. Messaging);
- **Rozlišení alespoň základních způsobů výměny informací** (angl. Message Patterns);
- **Volba technologie, datový formát, datová prodleva**.

A také dodávám některé důležité principy *integrace aplikací*:

- Jednoduchost;
- Udržitelnost;
- Znovupoužitelnost;
- Oddělení funkcionality a technologie.

Shrnuji technologické vlastnosti SaaS:

- SaaS jsou provozované převážně v externím datovém centru[5] (off-premise) a komunikace se službami probíhá prostřednictvím internetu. V rámci integrace SaaS je tedy zpravidla nutné vyřešit otázku komunikace v rámci zcela neheterogenních prostředích;
- SaaS může ve stejném okamžiku využívat více uživatelů;
- SaaS jsou platformě nezávislé;
- Při integraci SaaS je nutné vyřešit otázku bezpečnosti.

[5] Toto neplatí v případě SaaS provozovaných z privátního cloudu

V neposlední řadě by měla metodika zohledňovat specifika využití ICT v malých a středních firmách (viz Kapitola 2.2):

- Horší schopnost alokovat zdroje (finanční i personální) na nové ICT projekty a předem vyhodnotit dopad ICT projektů z hlediska nákladů a výnosů;
- Vyšší závislost na dodavatelích externích ICT služeb;
- Silná vůdčí role managementu při využití ICT technologií, avšak nemůžeme zároveň předpokládat, že management je technicky erudovaný, protože tato vlastnost zpravidla nebývá důležitá pro primární oblast podnikání společnosti;
- Vyšší flexibilita z hlediska změn nutných pro integraci nových ICT projektů do vnitřních procesů, avšak zároveň nižší tlak na formalizaci těchto procesů, u kterých proto musíme raději předpokládat, že vnitřní procesy jsou neformalizované.

Osobně považuji za klíčové faktory: *vliv managementu* a spíše *neformální vnitřní procesy*. Vliv managementu se totiž prolíná do všech ostatních faktorů. Management může být hlavním hybatelem, ale také bude nejspíše hlavním rozhodovatelem. Proto musí být metodika integrace jednoduchá natolik, aby byla uchopitelná také pro netechnicky orientované osoby s rozhodovacími pravomocemi.

Výstupem této kapitoly bude návrh metodiky integrace SaaS do on-premise informačního systému podniku.

4.1 Rešerše metodických zdrojů

V této kapitole popíši provedenou rešerši dostupných metodik integrace služeb SaaS do on-premise informačního systému podniku. Rešerše byla provedena kombinací metody hledání na základě klíčových slov a metody sněhové koule (Nováková, 2012). Metoda hledání na základě klíčových slov spočívala v prohledávání dostupných informačních pramenů pomocí klíčových slov souvisejících se zaměřením práce. Metoda sněhové koule spočívala ve využití informačních zdrojů použitých v nalezených materiálech, které souvisely s tématem.

Rešerše se opírala o následující zdroje, které lze rozdělit do následujících skupin:

- Odborné elektronické zdroje dostupné především přes centrum knihovnických služeb VŠE (CIKS), zejména: ACM Digital Library (ACM-DL) či Safari Tech Books Online;
- Dostupné citační rejstříky, konkrétně: Web of Knowledge (Thomson Reuters) a SCOPUS;
- Databáze kvalifikačních prací a databáze publikační činnosti VŠE;
- Odborné elektronické zdroje dostupné pomocí fulltextových vyhledávačů, zejména: Google Scholar;
- Odborné publikace a knihy z různých vydavatelství související s tématem;

- Okruh zdrojů poskytovaných analytickými společnostmi typu IDC, Gartner či Forrester;
- Materiály dodavatelů integračních platforem typu MuleSoft, Informatica apod.
- Materiály významných poskytovatelů SaaS typu Salesforce apod.
- Dodatečně bylo použito také všeobecné full-textové hledání na internetu, které svou kvalitou sice neodpovídá požadavkům kladeným na úplnost, avšak takto získané informace poskytují zajímavé dokreslující aspekty k dané problematice.

Materiály dodavatelů integračních platforem a poskytovatelů SaaS v sobě zahrnují riziko marketingového zabarvení poskytovaných informací, a proto se musí výsledné informace podrobit srovnání s informacemi dostupnými z jiných zdrojů pro minimalizování marketingového vlivu.

Materiály, které vzešly z provedené rešerše, lze shrnout do následujících skupin dle jejich sémantického zařazení:

- Rigorózní či agilní metodiky vývoje a údržby informačních systémů, např. (Buchalcevová, 2005) či (Voříšek, 2011);
- Metodiky od poskytovatelů SaaS a poskytovatelů integračních platforem, např. (Kao, 2015), (MuleSoft, 2014) či (Schmidt, 2010);
- Specifické metodiky budování SW architektur a přístupů typu SOA (angl. Service Oriented Architecture) či EAI (angl. Enteprise Application Integration), např. (Mackenzie, 2006), (Erl, 2004), (Erl, 2005), (Yu, 2009), (Costa, 2008) či (Linthicum, 2000);
- Obecná doporučení vzhledem k integraci SaaS od různých autorů, např. (Fehling, 2011), (Christiansson, 2012), (Lytra, 2012), (Hentrich, 2006), (Joha, 2012), (Izza, 2006), (Narasimha, 2012) či (Hai, 2009);
- Kvalifikační práce zabývající se problematikou integrace SaaS, např. (Karas, 2006), (Hauptvogel, 2006), (Sedláček, 2013), (Randová, 2013) či (Paluch, 2010).

Dostupné *rigorózní metodiky* (např. MMDIS, EUP či RUP) pokrývají kompletní pohled na podnikové procesy a akcentují vývoj informačních systémů na globální úrovni, což je příliš vzdálené od cílů této práce. Zástupci rigorózních typů metodik reprezentují zpravidla těžké metodiky, a proto se příliš nehodí pro firmy menší velikosti. Navíc příliš akcentují procesní pohled na organizaci jako celek.

Dostupné *agilní metodiky* (např. SCRUM či Kanban) zaujímají projektový pohled na vývoj informačních systémů a jsou uchopitelné pro firmy všech velikostí, což je v souladu s cílem této práce, avšak na druhou stranu postrádají dostatečný doménový detail (služby SaaS) a proto nenabízejí dostatečnou oporu pro volbu konkrétní metody integrace v daných situacích.

Metodiky nebo spíše doporučení od *poskytovatelů SaaS* či *poskytovatelů integračních platforem* příliš akcentují mateřský produkt a jsou úzce spjaté s konkrétní integrační platformou či produktem, což v důsledku vede k hrozbě zvýšené závislosti na dodavateli (angl. Lock-in). Navíc tato doporučení je obtížné získat, protože zpravidla vyžadují interakci s obchodníkem, který daný materiál poskytne až po úvodní schůzce či interview. Například materiál (Kao, 2015) jsem získal až po úvodním interview s obchodním zástupcem společnosti Informatica. Tento typ materiálů se proto jako obecná metodika integrace SaaS příliš nehodí.

Metodické postupy budování SW architektur typu SOA či EAI jsou velice detailní, avšak jejich záběr je příliš široký pro malé a střední společnosti. Tyto metodiky akcentují procesní model řízení organizací, což může být diskvalifikačním faktorem pro mnoho malých a středních společností. Detailní analýzu tohoto typu metodik popisuje (Karas, 2006).

Dále bylo možné identifikovat řadu různých *doporučení s ohledem na integraci SaaS* do podnikových systémů, avšak tato doporučení nebyla úplná a neměla formu metodiky. Tyto materiály zpravidla akcentovaly jednu specifickou oblast, ale nepostihovaly kompletní proces implementace. Na druhou stranu sloužily jako vodítko pro tvorbu nové metodiky v daných etapách životního cyklu integračního projektu.

Z různých kvalifikačních prací zabývající se problematikou integrace jsem vybral následující zástupce a podrobil je důkladné analýze.

Randová (2013) se zabývá integrací SaaS do informačního systému podniku, avšak nenabízí ucelenou metodiku postupu a její analýza různých přístupů zůstává v obecnější rovině a nejde příliš do hloubky. Sedláček (2013) se soustředil na Integraci poskytovanou jako služba (IaaS), což Pezzini (2011) považuje za předstupeň popsanému řešení na bázi iPaaS. Práce poskytuje detailní popis základů integrace a dále se zabývá identifikací poskytovatelů IaaS a analýzou jejich schopností a nabídek. Ale nezahrnuje doporučení z hlediska různých integračních metod. Podobně práce kolegy Palucha (2010) se soustředí na analýzu nabídek IaaS a popisuje obecné přístupy k integraci, avšak tato práce také nenabízí konkrétní kroky v rámci integračního projektu.

Specifickými problémy integrace konkrétně v bankovním sektoru se zabývá Koten (2010) a popisuje možnosti aplikace přístupu SOA v bankovním sektoru. Také Hauptvogel (2013) se zabývá SOA přístupem, i když již v obecnější rovině. Tato práce popisuje obecnou metodiku budování SOA architektury ve formě metodického vzoru založeného na metodickém rámci MeFIS.

Detailní metodický přístup k integraci zpracoval Karas (2006) ve své doktorské disertační práci. Práce se zaměřuje na vytvoření metodiky pro komplexní podnikovou integraci aplikací (EAI) založenou na

metodice MMDIS. Metodika je velice detailně zpracovaná, ale příliš těžká pro potřeby integrace SaaS v prostředí malých a středních firem.

Na základě provedené rešerše jsem dospěl k názoru, že žádná z dostupných metodik není zcela vhodná pro splnění cílů této práce.

Z těchto důvodů jsem se rozhodl vytvořit vlastní metodiku integrace SaaS. Metodika čerpá z principů agilních metodik, které jsou z mého pohledu vhodnější pro vývoj systémů v dynamicky měnících se podmínkách, a jsou více než vhodné pro malé vývojové týmy. Navíc integrace SaaS sama vybízí k aplikaci agilních přístupů, protože služby v rámci SaaS jsou okamžitě k dispozici a vývojáři tak mohou připravit velice rychle první verzi integrovaného řešení a sbírat zpětnou vazbu od uživatelů ve velice krátkém časovém intervalu.

Při studiu dostupných metodik, jsem došel k podobnému poznání jako Buchalcevová (2005), že dostupné metodiky nejsou popsány ani kategorizovány v jednotné struktuře. Tato skutečnost je nepraktická nejen s ohledem na složitost jejich vyhledávání dle jednotných parametrů, ale brání ve vzájemném doplňování a rozšiřování. Proto jsem nově vznikající metodiku založil na *metodickém rámci MeFIS* (angl. Methodology Framework for IS/ICT Systems), který je vyvíjen na Katedře informačních technologií VŠE v Praze a vychází z jednotné klasifikace různých metodických vzorů. Další výhodou je jeho objektová orientace, která umožnuje kombinovat různé metodické vzory (od obecnějšího po speciálnější) a jejich vzájemné prolínání dle potřeb daného projektu.

4.2 Metodický rámec MeFIS

Metodický rámec MeFIS je detailně popsán v (Buchalcevová, 2005) a příklad jeho využití nalezneme např. v (Buchalcevová, 2006) nebo (Hauptvogel, 2013). Na tomto místě uvedu několik klíčových charakteristik, které jsou stěžejní konkrétně pro vytvářenou metodiku (volně zpracováno dle (Buchalcevová, 2005)):

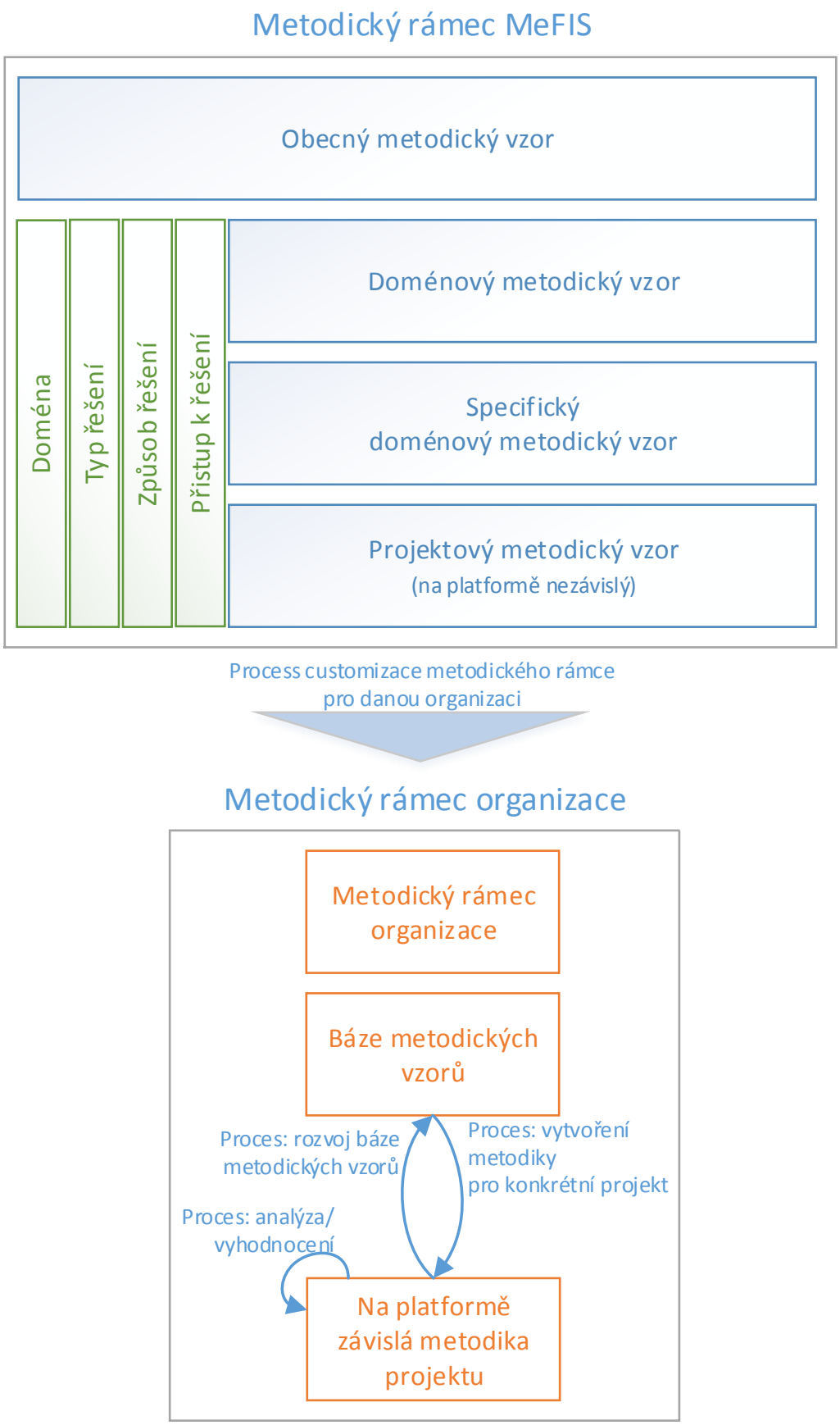
- Rámec nezohledňuje pouze perspektivu řízení, ale také perspektivu SW inženýrskou;
- Rámec podporuje pestrou škálu různých typů řešení včetně integrace;
- Rámec rozlišuje různou úroveň abstrakce (princip obecného rámce a z něj vycházející specifické rámce, tzn. specializace);
- Rámec využívá principů agilních technik;
- Rámec rozlišuje mezi typem prvku a instancí prvku;
- Rámec využívá multidimenzionální princip a neomezuje se výhradně na konkrétní role.

Struktura metodického rámce MeFIS je znázorněna na obrázku níže (viz Obrázek 18).

*„V rámci **metodického rámce MeFIS** jsou definovány metodické vzory různých typů. Základním vzorem je Obecný metodický vzor, od kterého jsou odvozeny vzory pro jednotlivé domény a typy projektů.“ (Buchalcevová, 2005)*

Tento popis poukazuje na stěžejní koncept *různé úrovně abstrakce* rámce MeFIS, který prostřednictvím *specializace* umožnuje, aby vzory nižších úrovní (např. specifický doménový vzor, projektový vzor atd.) dědily vlastnosti vzorů vyšší úrovně, a tím případně měnily stávající vlastnosti či přidávaly další.

Tato myšlenka je klíčová z toho důvodu, že případně můžeme kombinovat různé doménové vzory a různé projektové vzory za předpokladu, že oba vzory využívají pouze vlastnosti, které jsou definované na úrovni obecného vzoru – pouze je konkrétněji specifikují pro řešený problém. Například pokud bychom vytvořili diskutovanou metodiku *integrace služeb SaaS* jako metodický vzor na úrovni *specifického doménového vzoru*, což se ostatně nabízí, nic nebrání organizaci aplikovat tuto naší metodiku souběžně s nějakou obecnější metodikou definovanou na vyšší úrovni abstrakce, tj. *doménové úrovni*. Tímto způsobem je možné kombinovat obecnější metodické vzory akcentující globálnější pohled na organizaci a její procesy s metodickými vzory akcentující detailnější pohled na určité řešení.



Obrázek 18: Metodický rámec MeFIS (volně zpracováno dle (Buchalcevová, 2005))

Jednotlivé vzory v naznačené hierarchii jsou klasifikovány prostřednictvím následujících kritérií:

- **Domény:** Business Inteligence, Obecný SW, ERP, CRM atd.;
- **Typu řešení:** vývoj nového řešení, integrace řešení, užití řešení atd.;
- **Způsobu řešení:** vlastní, outsourcing;
- **Přístupu k řešení:** strukturovaný vývoj, objektový vývoj atd.

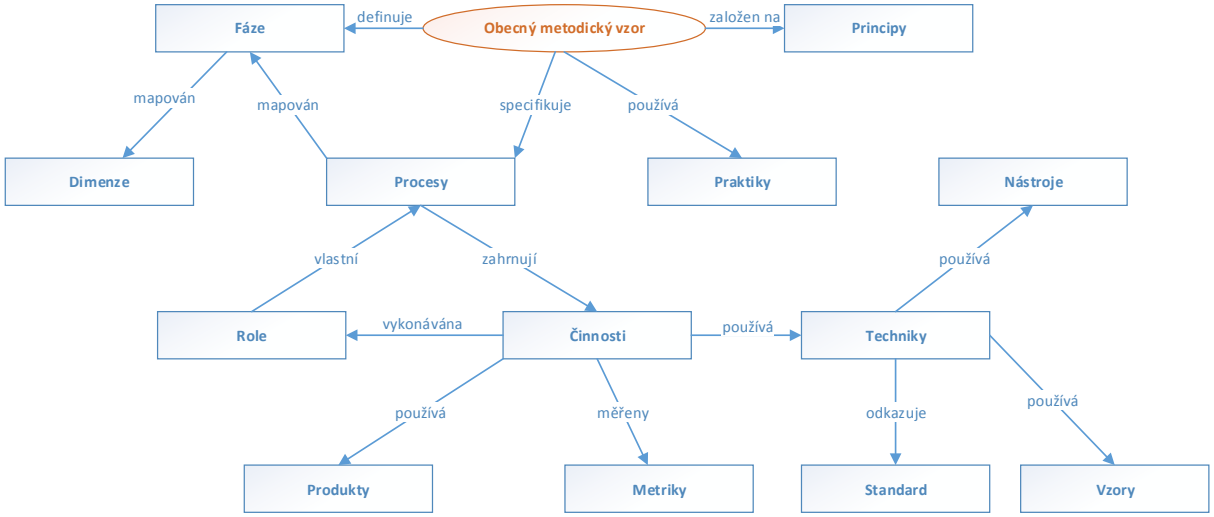
Buchalcevová (2005) shrnuje smysl vyjmenovaných kritérií: *„… je určen kontext daného vzoru, který umožnuje vyhledat odpovídající vzor a aplikovat jej pro konkrétní projekt.“*

Zmíněná kritéria jsou definována na úrovni *Obecného metodického vzoru*, a díky použitému konceptu *specializace* jsou zároveň zděděny do ostatních metodických vzorů. Kromě klasifikačních kritérií jsou v rámci obecného vzoru nadefinovány také jednotlivé *prvky metodiky*, které jsou upravovány (specializovány) na úrovni doménových či projektových vzorů (volně zpracováno dle (Buchalcevová, 2005)):

- **Fáze:** *definují* jednotlivé fáze životního cyklu;
- **Dimenze:** *definují* dimenze týkající se vyvíjeného systému, např. hardwarová, softwarová, datová atd.;
- **Role:** *definují* role pracovníků s definovanými odpovědnostmi, které zahrnují provádění určitých činností;
- **Principy:** *definují* základní principy (zásady), které je třeba uplatňovat;
- **Praktiky (angl. Best Practices):** *odkazují* na zkušenosti, znalosti a osvědčené přístupy k řešení. Mohou se týkat spolupráce, řízení nebo vývoje;
- **Procesy:** *definují* sadu činností, techniky a praktiky, které lidé používají při vývoji;
- **Činnosti:** *definují* jednotky práce, ze kterých se skládá proces. Činnosti popisují, co je vytvářeno (produkty); kým je vytvářeno (role) a jakým způsobem (praktiky);
- **Techniky:** *odkazují* na specializované techniky (metody), které jsou pro řešení vyžadovány činnostmi (např. UML, UI prototypování atd.);
- **Nástroje:** *odkazují* na automatizované nástroje, které podporují jednotlivé techniky, činnosti a procesy;
- **Vzory (angl. Patterns):** *odkazují* na vzory popisující obecná řešení známých problémů;
- **Metriky:** *definují* ukazatele nebo hodnotící kritéria;
- **Standardy:** *odkazují* na existující standardy, které musí být respektovány;
- **Produkty:** *definují* artefakty, které jsou vytvářeny, upravovány nebo užívány v rámci procesů a činností (plán projektu, obrázek atd.)

Je zřejmé, že metodický rámec MeFIS je navržen s ohledem na *těžké metodiky* popsané do nejmenšího detailu, ale také respektuje *metodiky lehké*, u kterých postačí definovat *principy, procesy a praktiky* (Buchalcevová, 2005).

Všechny prvky rámce a jejich vzájemné vztahy je možné shrnout pomocí následujícího grafu (viz Obrázek 19).



Obrázek 19: Prvky rámce MeFIS (volně zpracováno dle (Buchalcevová, 2005); upraveno autorem)

Fáze životního cyklu projektu jsou základním a z mého pohledu klíčovým integrujícím prvkem. Buchalcevová (2005) ve své práci akcentuje: *„Koncept fází a bran platí pro celou organizaci, je rámcem pro všechny projekty. Díky jednotnému pohledu na fáze může vedení organizace sledovat postup prací na jednotlivých projektech, vytvářet portfolio projektů a synchronizovat projekty, jejichž komponenty musí být integrovány“*. Z uvedeného popisu je zřejmé, že zmíněná synchronizace v rámci různých projektů může být využita i v případech, pokud například aplikujeme dva zcela odlišné metodické vzory z portfolia rámce MeFIS.

V předchozím odstavci jsem použil slovo „aplikovat“, což je ukázka dalšího velice důležitého konceptu rámce MeFIS a jeho *rozlišování mezi typem prvku a instancí prvku*. Rámec MeFIS totiž plně respektuje specifičnost každé organizace a unikátnost každého projektu. Z tohoto důvodu obsahuje rámec MeFIS tzv. rovinu meta-metodiky, která popisuje jakým způsobem vybrat daný metodický vzor, přizpůsobit jej pro potřeby dané organizace a posléze jej aplikovat na metodiku konkrétního projektu. V neposlední řadě také definuje, jakým způsobem mohou organizace udržovat vlastní bázi metodik a tím uchovávat znalosti a zkušenosti z realizace různých projektů. Díky tomuto konceptu je respektována jedinečnost každého projektu, protože metodický vzor je vždy aplikován (a přizpůsoben) na konkrétní projekt.

4.3 Metodický vzor pro integraci služeb SaaS

V této kapitole provedu popis navrženého metodického vzoru integrace služeb SaaS do on-premise prostředí malého či středního podniku ve struktuře vycházející z metodického rámce MeFIS (struktura tabulek převzata z dokumentace MeFIS (Buchalcevová, 2005)).

Vzhledem k relativně úzkému zaměření navrhované metodiky (SaaS) se bude jednat o *lehký doménově specifický metodický vzor s multidimenzionálním přístupem, který zahrnuje stránku technickou (vývoj), ale také uživatelskou (uživatele).* Metodika MeFIS doporučuje v případě lehkých projektových vzorů definovat minimálně: *fáze*, *principy*, *procesy* a *praktiky*. Proto i navržená metodika toto doporučení respektuje, avšak místo *praktik* definuje *vzory (angl. Patterns)*, které jsou vhodnější pro splnění cílů práce.

Tabulka 5: Základní popis navrženého metodického vzoru

Název	Integrace SaaS do on-premise prostředí informačního systému podniku
Klasifikace	Doména: EAI (Enterprise Application Intergration) Typ řešení: INT (Integrace řešení) Způsob řešení: IN (Vlastní) Přístup k řešení: ST (Strukturovaný vývoj), OO (objektově orientovaný vývoj)
Účel	Metodický vzor specifikuje fáze, principy, procesy a vzory pro integraci SaaS do IS podniku
Rodičovské vzory	M0 Obecný metodický vzor
Vztah k jiným vzorům	
Celopodnikové SW inženýrské role	**Vedoucí IT**
SW inženýrské role v projektu	**Vývojář** **Koordinátor služeb třetí strany**
Role zájmových skupin	**Sponzor**
Doménové role	**Uživatel** **Koordinátor služeb SaaS**

Principy	Iterativní přístup k vývoji Princip budování dlouhodobě udržitelné podnikové architektury Princip zachování bezpečného přístupu a komunikace
Procesy	Aktualizace podnikové politiky správy a řízení IT (angl. IT Governance), PA01 Specifikace požadavků a rozsahu integračního projektu, PB01 Výběr služby SaaS a vlastnosti služby SaaS, PB02 Volba metody integrace, PB03 Volba integračního scénáře, PB04 Analýza možných omezení využití služby (SaaS), PB05 Detailní návrh jednotlivých komponent integračního řešení, PC01 Získání funkčního prototypu, PC02 Implementace řešení, PD01 Testování řešení, PD02 Nasazení řešení, PE01 Administrace řešení, PF01
Vzory (angl. Patterns)	Kanonický datový model Doplnění obsahu zprávy Balíček zprávy Překladač zpráv

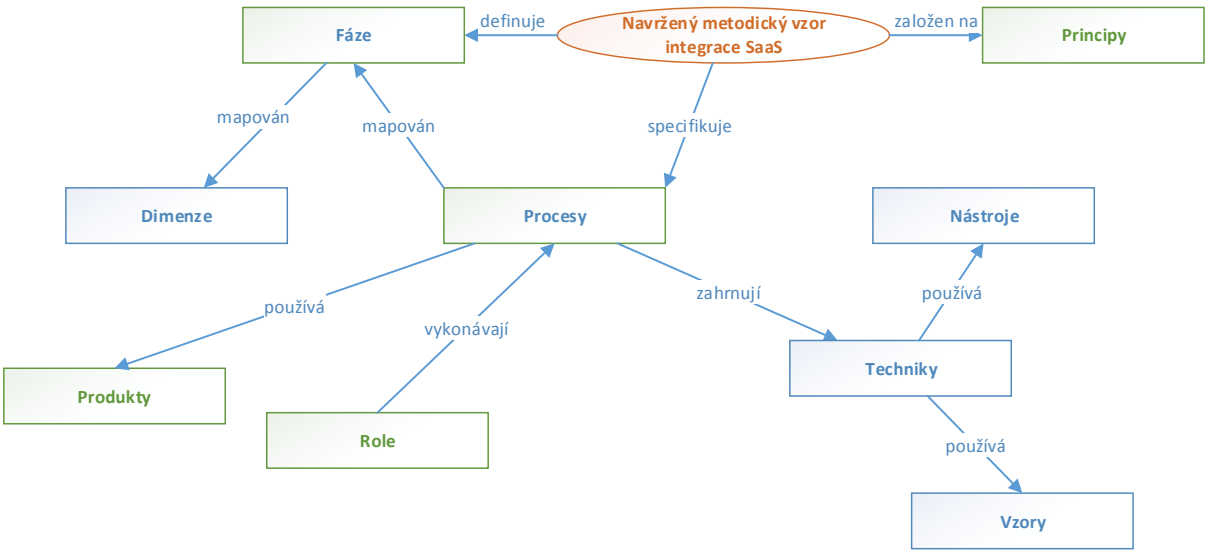
Tabulka 5 zobrazuje základní řídící struktura navrženého metodického vzoru. Struktura je převzata z dokumentace MeFIS (Buchalcevová, 2005; upraveno autorem) a obsahuje *klasifikaci* vzoru v rámci hierarchie metodických vzorů MeFIS. Součástí návrhu je také kompletní definice následujících prvků:

- Fáze;
- Procesy;
- Role;
- Principy;
- Produkty (artefakty, které jsou využívány či tvořeny jednotlivými procesy);
- Odkazy na návrhové vzory.

U každého procesu je vždy uveden také jednoznačný identifikátor ve formátu:

- Písmeno „P“ – konstanta, která označuje „Proces“;
- Písmena „A“ až „F“ – proměnná, která označuje příslušnost k dané fázi životního cyklu projektu;
- Číslo XX – číslo unikátně označující proces v rámci dané fáze.

Navržené prvky a jejich vzájemné vztahy je možné zobrazit pomocí upraveného schéma (viz Obrázek 20) vycházející z kompletního schéma MeFIS (viz Obrázek 19).



Obrázek 20: Prvky navrženého metodického vzoru

Jednotlivé prvky detailně popisuji v následujících podkapitolách.

4.3.1 Fáze

Integrační projekt je během své realizace rozdělen do jednotlivých fází životního cyklu a navržený metodický vzor rozeznává následující fáze:

- Fáze Informační strategie (viz Tabulka 7);
- Fáze Globální analýza a návrh (viz Tabulka 8);
- Fáze Detailní analýza a návrh (viz Tabulka 9);
- Fáze Implementace (viz Tabulka 10);
- Fáze Zavedení (viz Tabulka 11);
- Fáze Provoz a údržba (viz Tabulka 12).

Informační strategie je fáze, která je prováděna na úrovni celého podniku. Ostatní fáze jsou realizovány na úrovni daného projektu. Při implementaci projektu jsou jednotlivým fázím přiřazeny procesy, které zajištují technickou realizaci projektu. Mezi jednotlivými fázemi jsou definovány ještě tzv. *Brány*, které akcentují nutnost projektového řízení a definují hodnotící kritéria pro úspěšné ukončení jednotlivých fází.

Jednotlivé fáze a popis jejich atributů vychází ze struktury specifikované v *obecném metodickém rámci MeFIS* (Buchalcevová, 2005); upraveno autorem) a jednotlivým fázím jsou přiřazené procesy (viz Tabulka 6).

Tabulka 6: Fáze metodického vzoru

Fáze dle rámce MeFIS	*Navržené procesy v rámci metodického vzoru*
Celopodniková vrstva	
Globální strategie	*Tento metodický vzor tuto fázi nespecifikuje. Při aplikaci metodického vzoru je možné využít jiný obecnější vzor.*
Informační strategie	Aktualizace podnikové politiky správy a řízení IT (angl. IT Governance), PA01
Řízení a koordinace projektů	*Tento metodický vzor tuto fázi nespecifikuje. Při aplikaci metodického vzoru je možné využít jiný obecnější vzor.*
Projektová vrstva	
Úvodní studie	*Tento metodický vzor tuto fázi nespecifikuje. Při aplikaci metodického vzoru je možné využít jiný obecnější vzor.*
Globální analýza a návrh	Specifikace požadavků a rozsahu integračního projektu, PB01 Výběr služby SaaS a vlastnosti služby SaaS, PB02 Volba metody integrace, PB03 Volba integračního scénáře, PB04 Analýza možných omezení využití služby (SaaS), PB05
Detailní analýza a návrh	Detailní návrh jednotlivých komponent integračního řešení, PC01 Získání funkčního prototypu, PC02
Implementace	Implementace řešení, PD01 Testování řešení, PD02
Zavedení	Nasazení řešení, PE01
Provoz a údržba	Administrace řešení, PF01

Metodický rámec MeFIS je navržen s ohledem na princip multidimenzionality, který spočívá v uvědomění si různých úhlů pohledu na vyvíjené řešení v různých fázích, a proto i navržený metodický vzor tento princip podporuje a zohledňuje různé pohledy na vyvíjené řešení v rámci jednotlivých fází životního cyklu projektu.

Detailní popis jednotlivých fází je uveden v následujících tabulkách ve struktuře:

- Název fáze;
- Účel fáze;
- Náplň fáze v rámci jednotlivých dimenzí;
- Vstupy fáze;
- Výstupy fáze;
- Kritické faktory úspěchu;
- Brány, tj. specifikace hodnotících kritérií pro úspěšné ukončení fáze.

Každý integrační projekt by měl začínat od fáze *informační strategie* (viz Tabulka 7), protože zapojení SaaS do informačního systému podniku významně ovlivňuje celkovou podstatu podnikové IT strategie (více kapitola 4.3.3.1).

Tabulka 7: Popis fáze Informační strategie

Název fáze:	Informační strategie
Účel fáze:	Aktualizace podnikové informační politiky a řízení IT (angl. IT Governance) a předvýběr schválených poskytovatelů SaaS
Náplň fáze v dimenzích:	
Funkce/Procesy:	Definování korporátních politik Specifikace kontrolních mechanismů a záložních variant
Data	Parametry výběru poskytovatelů SaaS Požadavky nařízení a zákonů
HW	Návrh na zabezpečení HW
Technologie	Dopady na bezpečnost IS Požadavky na zabezpečení technologií
UI	
Lidé	Rámcová definice rolí a rámcová definice požadavků na dané role
Organizace	Využití SaaS v rámci organizační struktury
Ekonomika	Finanční dopad změn informační politiky
Vstupy:	Podniková informační strategie Seznam poskytovatelů SaaS připadajících v úvahu
Výstupy:	Aktualizovaná podniková informační strategie Aktualizovaný (zredukovaný) seznam poskytovatelů SaaS

Kritické faktory úspěchu:	Nepodceňování bezpečnostních rizik spolupráce s poskytovateli SaaS
Hodnotící kritéria brány:	Schválení informační strategie a schválení (zredukovaného) seznamu poskytovatelů SaaS

Procesy v rámci fáze *globální analýza a návrh* (viz Tabulka 8) jsou navržené takovým způsobem, aby před vstupem do fáze *detailní analýza a návrh* (viz Tabulka 9) byl celý integrační projekt smluvně ošetřen a bylo zřejmé, kteří aktéři budou vlastní řešení realizovat.

Tabulka 8: Popis fáze Globální analýza a návrh

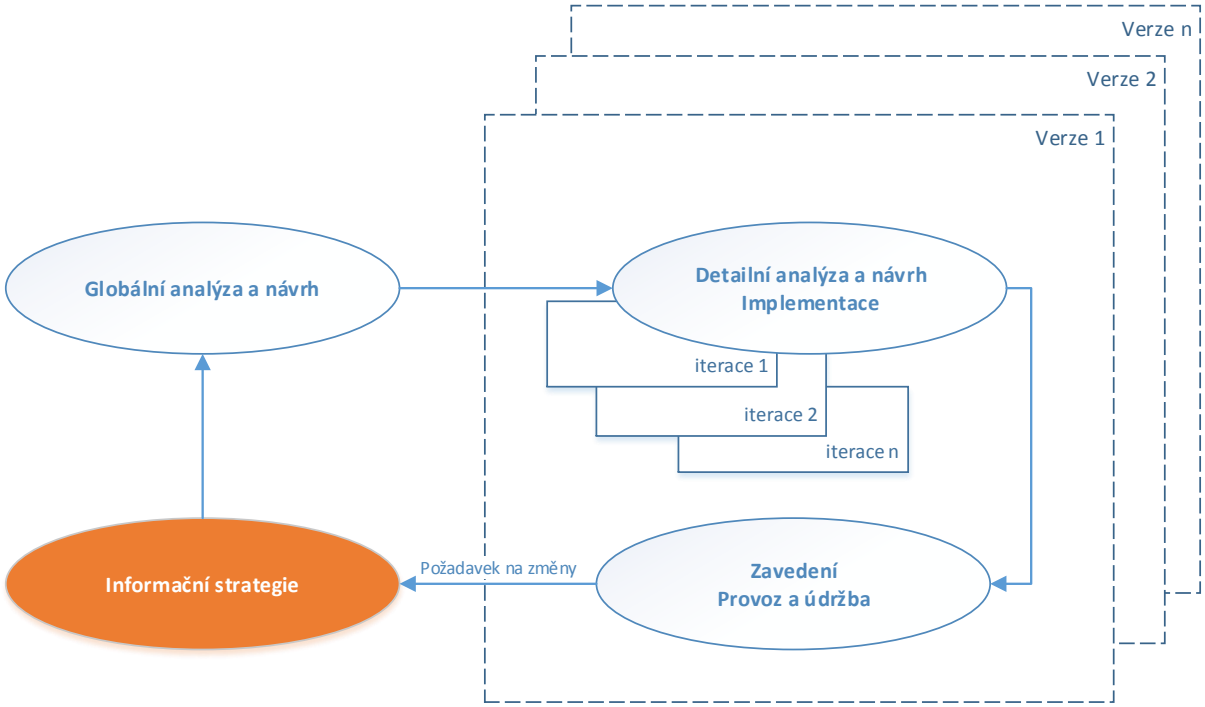
Název fáze:	Globální analýza a návrh
Účel fáze:	Cílem je detailní platformě nezávislý návrh řešení integračního projektu a architektury řešení sestávající z následujících komponent: • Definice požadavků • Volba jednoho konkrétního poskytovatele SaaS • Podpis SLA • Volba integrační metody a integračního scénáře • Analýzy možných omezení
Náplň fáze v dimenzích:	
Funkce/Procesy:	Podrobná specifikace uživatelských požadavků Detailní specifikace rozhraní, metody integrace a integračního scénáře
Data	Parametry integračního řešení
HW	Návrh HW architektury
Technologie	Návrh SW architektury (middleware)
UI	Požadavky na UI
Lidé	Uživatelské požadavky
Organizace	Organizační požadavky
Ekonomika	Finanční dopad navrženého integračního projektu
Vstupy:	Seznam poskytovatelů SaaS
Výstupy:	Detailní platformě nezávislý návrh integračního řešení Přehled všech požadovaných parametrů řešení Podepsané SLA s konkrétním poskytovatelem SaaS

Kritické faktory úspěchu:	Aktivní přístup uživatelů a vývojáře Respektování principů metodického vzoru Používání návrhových vzorů
Hodnotící kritéria brány:	Odsouhlasení návrhu řešení

Cílem fáze *detailní analýza a návrh* (viz Tabulka 9) je získání platformě *závislého* návrhu integračního řešení.

Tabulka 9: Popis fáze Detailní analýza a návrh

Název fáze:	Detailní analýza a návrh
Účel fáze:	Cílem je detailní platformě závislý návrh integračního řešení včetně funkčního prototypu
Náplň fáze v dimenzích:	
Funkce/Procesy:	Návrh vývojové platformy a z toho vyplývajících požadavků
Data	Technická dokumentace služby (SaaS) Parametry integračního řešení
HW	Požadavky na HW
Technologie	Požadavky na SW
UI	Požadavky vzhledem k UI
Lidé	Přístupová práva Rozsah cílové skupiny uživatelů
Organizace	Plán vývoje komponent řešení
Ekonomika	Aktualizované náklady projekt
Vstupy:	Detailní platformě nezávislý návrh integračního řešení Přehled všech požadovaných parametrů řešení
Výstupy:	Detailní platformě závislý návrh integračního řešení a plán implementace komponent řešení Funkční prototyp integračního řešení
Kritické faktory úspěchu:	Používání návrhových vzorů Jednoduchost a účelnost řešení Flexibilita řešení
Hodnotící kritéria brány:	Odsouhlasení funkčního prototypu řešení a plánu implementace jednotlivých komponent řešení



Obrázek 21: Schématické znázorněné fází metodického vzoru (volně zpracováno dle (HAI, 2009); upraveno autorem)

Navržený metodický vzor se řídí *principem iterativního přístupu k vývoji*, a tato iterativnost je znázorněna na obrázku výše (viz Obrázek 21). Základem tohoto principu je myšlenka, že celý samotný vývoj řešení je rozdělen na relativně krátké úseky, kdy na závěr každé iterace existuje funkční produkt nebo jeho část, který mohou hodnotit koncoví uživatelé. Tímto způsobem získá vývojář zpětnou vazbu již v raných fázích vývoje, a k této zpětné vazbě může v průběhu dalších iterací přihlédnout a tím ji zohlednit v průběhu fáze *implementace* (viz Tabulka 10). Jednotlivé iterace zahrnují také fázi *Detailní analýza a návrh*, protože v rámci zpětné vazby od uživatelů je možné, že bude nutné provést změny v celkovém návrhu integračního řešení.

Tabulka 10: Popis fáze Implementace

Název fáze:	Implementace
Účel fáze:	Tvorba funkčního integračního řešení do takového stupně vyspělosti umožňující jeho nasazení
Náplň fáze v dimenzích:	
Funkce/Procesy:	Realizace řešení Testování řešení
Data	Přesun dat, persistence dat
HW	Detailní požadavky HW

Technologie	Detailní požadavky HW
UI	Realizace UI
Lidé	Návrh distribuce řešení Zpětná vazba – testování řešení
Organizace	Příprava změn organizačního workflow s ohledem na integrační řešení
Ekonomika	Finální finanční dopad nákladů integračního řešení
Vstupy:	Detailní platformě závislý návrh integračního řešení a plán implementace komponent řešení Funkční prototyp integračního řešení
Výstupy:	Funkční integrační řešení v takovém stupni vyspělosti umožňující jeho nasazení
Kritické faktory úspěchu:	Používání návrhových vzorů Výběr vhodných metod pro testování
Hodnotící kritéria brány:	Odsouhlasení verze integračního řešení určené k distribuci

Jakmile dosáhne řešení určitého stupně vyspělosti, může být nasazeno obvykle v delších časových cyklech (jednotlivých verzích). Nasazení probíhá v rámci fáze *zavedení* (viz Tabulka 11).

Tabulka 11: Popis fáze Zavedení

Název fáze:	Zavedení
Účel fáze:	Zavedení integračního řešení do provozu
Náplň fáze v dimenzích:	
Funkce/Procesy:	Paralelní zpracování stávajícím systémem (Buchalcevová, 2005) Komplexní provozní otestování implementovaného řešení (Buchalcevová, 2005) Parametrické nastavení systému
Data	Příprava dat, přesun dat
HW	Plošná instalace HW
Technologie	Plošná instalace SW
UI	Plošné testování UI
Lidé	Zajištění plošného školení uživatelů (Buchalcevová, 2005) Počáteční nastavení přístupových práv
Organizace	Aplikace změn organizačního workflow

Ekonomika	Konečný účet a jeho kontrola
Vstupy:	Funkční integrační řešení v takovém stupni vyspělosti umožňující jeho nasazení
Výstupy:	Plošně zavedené integrační řešení – protokol
Kritické faktory úspěchu:	Zajištění stability řešení i při dodání nových verzí některých částí (Buchalcevová, 2005)
Hodnotící kritéria brány:	Potvrzení úspěšné distribuce řešení

Závěrečná fáze *provoz a údržba* (viz Tabulka 12) zajištuje provoz integračního řešení včetně nastavování parametrů nutných k jeho provozu. V rámci této fáze je také nutné sledovat a vyhodnocovat dodržování parametrů SLA ze strany poskytovatele SaaS.

Tabulka 12: Popis fáze Provoz a údržba

Název fáze:	Provoz a údržba
Účel fáze:	Zajištění provozu integračního řešení a nastavování parametrů nutných k jeho provozu při jejich změně.
Náplň fáze v dimenzích:	
Funkce/Procesy:	Poskytování funkcionality daného řešení
Data	Užití, zálohy, zabezpečení dat
HW	Monitoring využitých kapacit HW
Technologie	Monitoring SW Monitoring dodržování SLA
UI	Monitoring užití UI pro poskytnutí zpětné vazby pro případné úpravy
Lidé	Školení nově příchozích uživatelů (Buchalcevová, 2005)
Organizace	Monitoring efektivity změněného workflow
Ekonomika	Výpočet ekonomického přínosu a vyhodnocení
Vstupy:	Zavedené integrační řešení
Výstupy:	Požadavky na změny (Buchalcevová, 2005)
Kritické faktory úspěchu:	Pravidelný monitoring funkčnosti řešení
Hodnotící kritéria brány:	

Procesy přiřazené jednotlivým fázím jsou detailně popsány v kapitole 4.3.3.

4.3.2 Role

Navržený metodický vzor definuje následující role:

- **Vedoucí IT:** zodpovědný za aktualizaci podnikových politik řízení a správy IT;
- **Vývojář:** zodpovědný za implementaci navrženého řešení;
- **Koordinátor služeb třetí strany:** pracovník, který komunikuje s dodavatelem integračního řešení či iPaaS platformy;
- **Sponzor:** rozhoduje o využití nabízených službách, řeší konflikty a financuje projekt;
- **Uživatel:** pracovník využívající služby informatiky, zajištuje vstupní informace potřebné pro vývoj, specifikuje požadavky a testuje funkcionalitu;
- **Koordinátor služeb SaaS:** pracovník, který komunikuje s poskytovatelem SaaS a zajišťuje součinnost s poskytovatelem SaaS.

Jednotlivé role jsou klíčovou součástí procesů, jež se účastní. Při aplikaci navrženého vzoru na konkrétní projekt musí být provedeno rozdělení rolí s tím, že jedna konkrétní osoba může vykonávat více rolí současně.

Na tomto místě také vysvětlím, proč jsem zvolil dvě různé role pro dvě podobné činnosti, konkrétně:

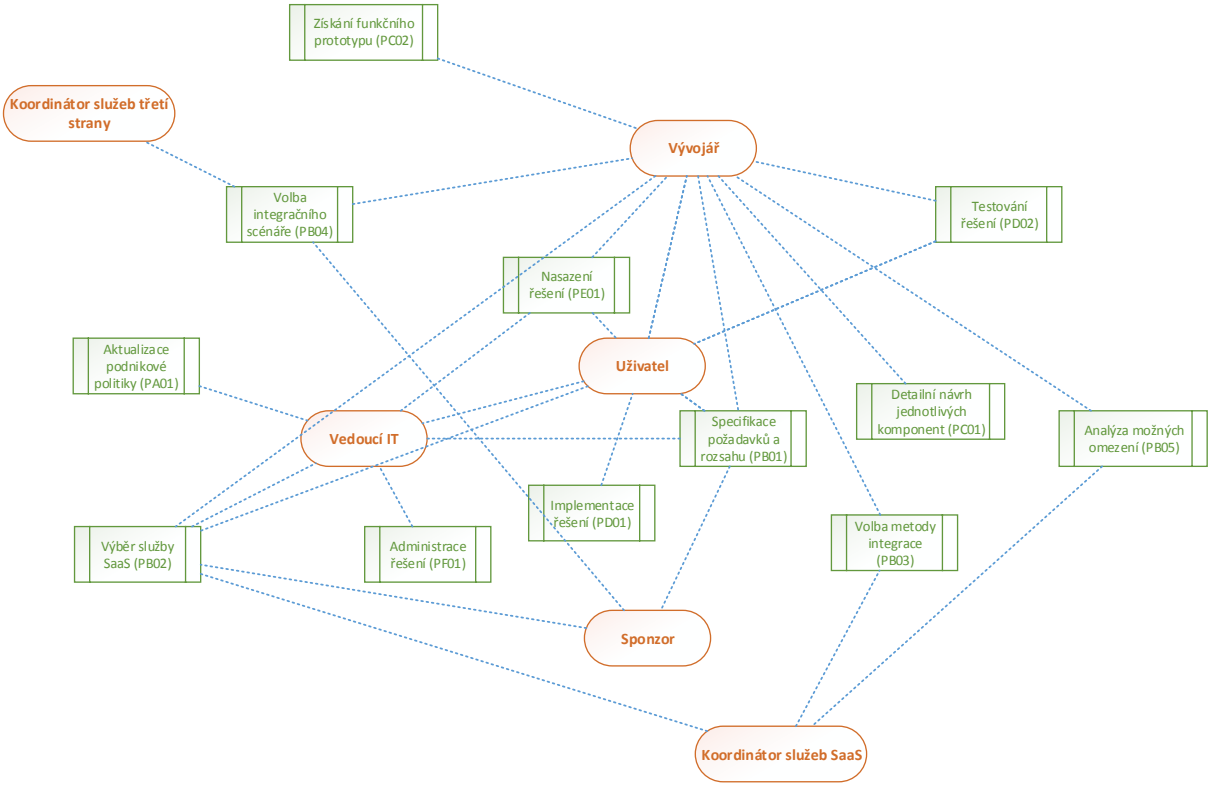
- Koordinátor služeb SaaS;
- Koordinátor služeb třetí strany.

I když to na první pohled nemusí být zřejmé, vnímám tyto dvě role zásadně odlišně. *Koordinátor SaaS* je z mého pohledu doménově specifická role, ve které je důležité, aby daný člověk rozuměl oblasti podnikání společnosti a dokázal určit, která funkcionalita SaaS je důležitá pro vykonávání podnikových procesů. *Koordinátor integračních služeb třetí strany* je role SW-technického zaměření, která je zodpovědná za realizaci integrační komponenty v případě, že tato komponenta bude „outsourcována“.

Pro každý proces je přiřazena role ze dvou různých hledisek:

- **Zodpovědnosti:** za konkrétní proces je zodpovědná vždy pouze jedna role;
- **Účasti:** procesu se může účastnit více rolí současně.

Všechny přiřazené role k daným procesům bez ohledu na toto hledisko jsou znázorněny na obrázku níže (viz Obrázek 22).



Obrázek 22: Procesy a přiřazené role

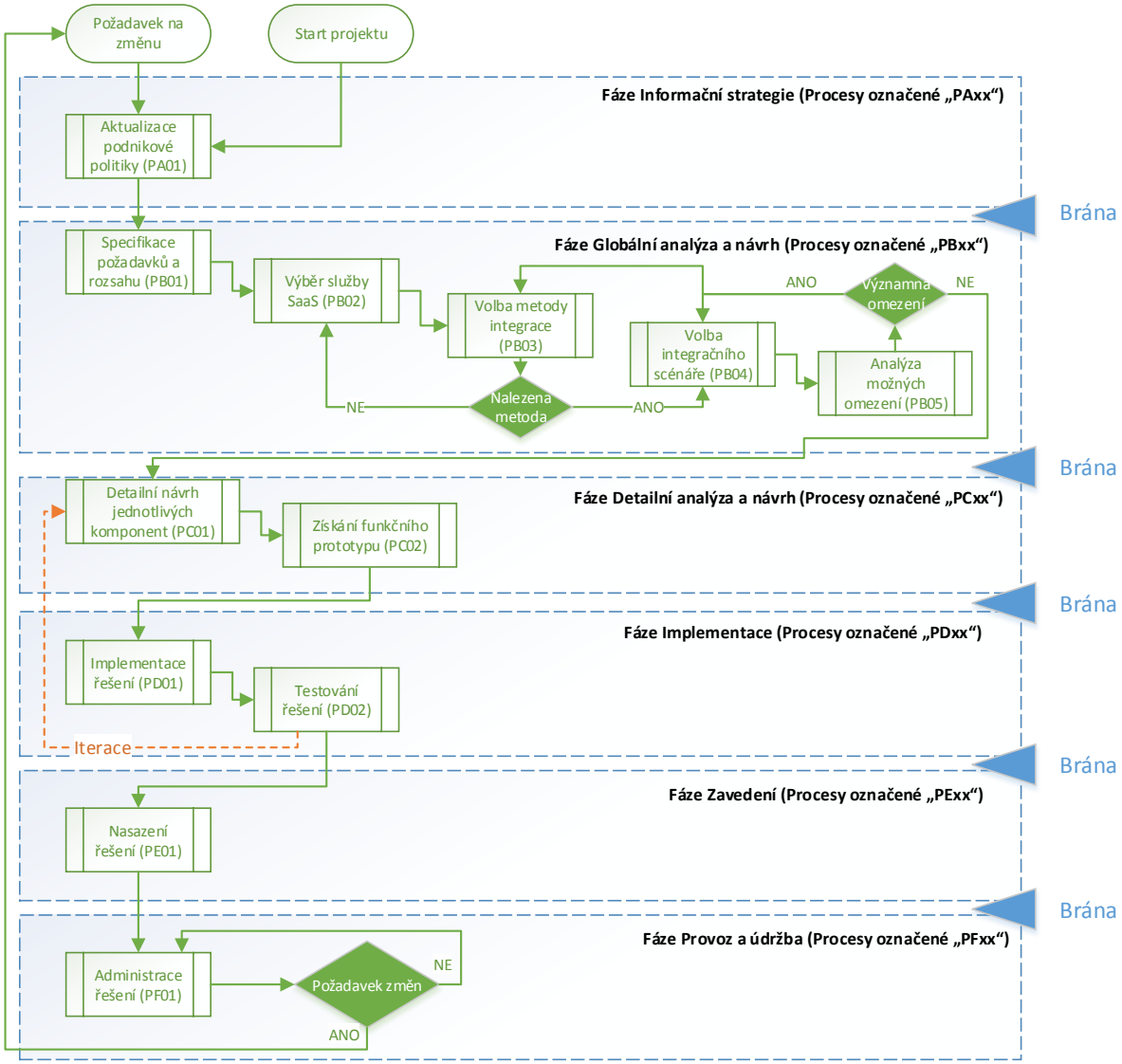
Z obrázku je zřejmé, že mezi velice frekventované role patří: *Vedoucí IT*, *Uživatel* a *Vývojář*. Proto je nutné alokovat dostatek prostoru před realizací integračního projektu právě těmto rolím.

Detailní popis jednotlivých procesů je součástí následující kapitoly 4.3.3.

4.3.3 Procesy

Tato kapitola obsahuje výčet navržených procesů a stručné shrnutí náplně a důvodu jejich zařazení do metodického vzoru. Jak již bylo zmíněno, vytvářený metodický vzor je lehký, a proto nebudu rozvádět jednotlivé procesy do detailních činností a procesních diagramů. Některé procesy obsahují navíc návrh metodických nástrojů a technik, které mohou usnadnit provádění činností související s daným procesem.

Vyjdeme-li z přiřazených procesů daným fázím (viz Tabulka 6) a znázorníme pohled na všechny definované procesy, získáme tak celkový průběh realizace projektu (viz Obrázek 23).



Obrázek 23: Průběh realizace projektu v různých fázích

Integrační projekt může být v podstatě zahájen dvěma způsoby:

- Jako zcela nový integrační projekt;
- Nebo jako požadavek na změnu stávajícího integračního řešení.

Avšak jednotlivé procesy jsou totožné v případě obou možností včetně aktualizace informační politiky, protože s každou novou verzí integračního řešení je nutné provést kontrolu souladu daného řešení s informační politikou společnosti.

Mezi jednotlivými fázemi jsou znázorněné *Brány*, které do celého projektu zavádí složku řízení. Každá fáze obsahuje specifikaci hodnotících kritérií, jejichž splnění indikuje úspěšné ukončení dané fáze a možnost pokračovat do další fáze. Popis hodnotících kritérii jednotlivých bran je uveden v kapitole 4.3.1.

Naopak projekt je považován za ukončený, pokud existuje nasazené funkční řešení, u kterého je již pouze prováděna jeho administrace. Avšak na druhou stranu je zřejmé, že nasazením verze 1 obvykle vývoj nekončí, ale jsou postupně sbírány požadavky pro verzi 2 atd. Celý proces realizace projektu, resp. jeho vylepšování, se proto může dále opakovat.

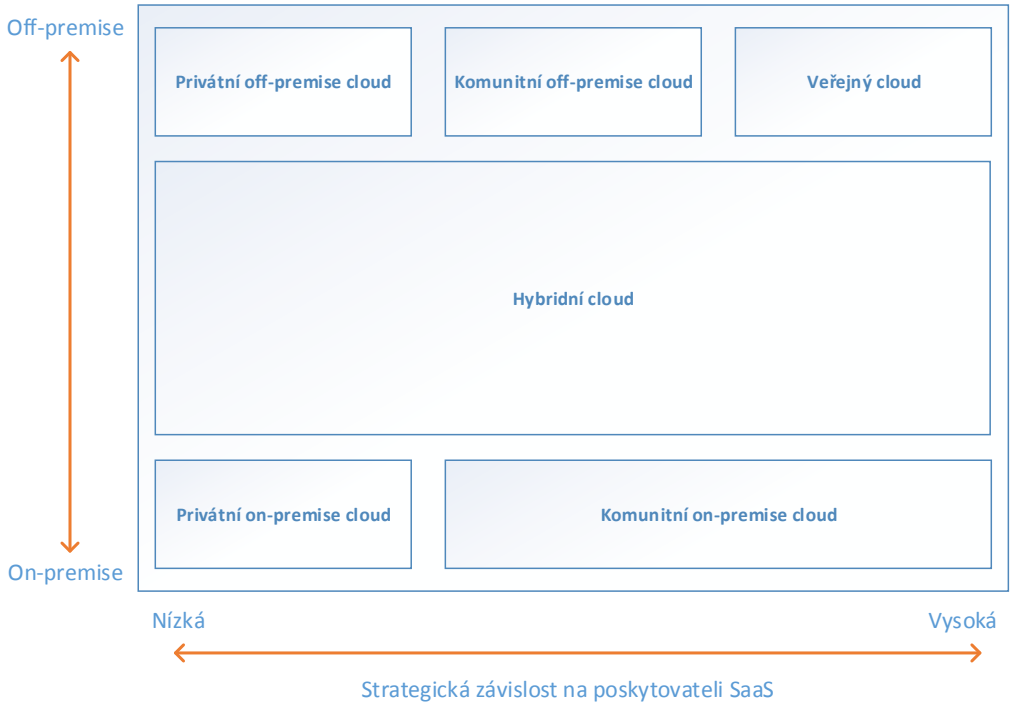
Detailní popis jednotlivých procesů je uveden v následujících podkapitolách. Každý proces je vždy uveden ve struktuře (struktura převzata z (Buchalcevová, 2005; upraveno autorem)):

- **Cíle procesu:** popis cílů, které daný proces splňuje;
- **ID procesu:** unikátní identifikátor;
- **Zodpovědnost:** role, která primárně zodpovídá za daný proces;
- **Účastníci:** role, které se případně zúčastní řešení daného procesu;
- **Vstupy procesu:** produkty, které jsou na vstupu procesu;
- **Výstupy procesu:** produkty, které jsou výstupem procesu;
- **Techniky:** odkazy na techniky, které lze využít při realizaci daného procesu;
- **Kritické faktory úspěchu:** faktory, které z velké části ovlivňují pozitivní výsledek procesu.

4.3.3.1 Proces: aktualizace podnikové politiky správy a řízení IT, PA01

Integrací SaaS do podnikového informačního systému se razantně mění princip fungování podnikového informačního systému a poskytování informatických služeb podnikovým procesům obecně. Po úspěšné integraci je poskytovatel SaaS nepřímou součástí podnikového informačního systému, a proto je nutné tento fakt zohlednit v celkové strategii řízení a správy informatických služeb podniku (angl. IT governance).

Oproti variantě, že celý systém je provozován on-premise, je zejména nutné zvážit zavedení nových kontrolních mechanismů a záložních variant. Tuto změnu a intenzitu závislosti je možné ilustrovat pomocí grafu níže (viz Obrázek 24).



Obrázek 24: Strategická závislost na poskytovateli SaaS (volně zpracováno dle (Joha, 2012); upraveno autorem)

Z grafu je zřejmé, že závislost na poskytovateli SaaS služeb je různá dle rozdílných typů cloudových služeb a společnost by měla vždy vyhodnotit rizika z daného typu pramenící. Lze zjednodušeně konstatovat, že *privátní* cloudové služby jsou méně rizikové, než *veřejné* cloudové služby a budou proto vyžadovat jinak intenzivní změny celkové informační politiky organizace. Cílem tohoto procesu je proto *vyhodnocení rizik závislosti na poskytovateli a aktualizace podnikové informační strategie na základě těchto rizik.*

Obecně platí, že čím vyšší závislost na poskytovateli SaaS, tím obezřetnější bychom měli být. Detailní popis jednotlivých variant cloudu je možné nalézt v (Liu, 2011).

Ze studie (Joha, 2012, strana 1515) vyplývá, že bychom se v tomto procesu měli soustředit zejména na následující oblasti:

- Návrh smluv o poskytování IT služeb (SLA, angl. Service Level Agreement);
- Definovaní obecných korporátních politik pro výběr poskytovatelů SaaS;
- Soulad nařízení (zákonů) s IT strategií (angl. Compliance);
- Řízení vztahu s poskytovateli SaaS (stanovení kritérií výběru atd.);
- Dopady na bezpečnost IS;
- Kontrola a monitoring IS.

Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 13).

Tabulka 13: Popis procesu Aktualizace podnikové politiky správy a řízení IT

Název procesu:	Aktualizace podnikové politiky správy a řízení IT (angl. IT Governance)
ID procesu	PA01
Cíle:	Vyhodnocení rizik závislosti na poskytovateli SaaS Zajištění/kontrola souladu podnikové informační politiky vzhledem k vyšší strategické závislosti na poskytovateli SaaS Výběr poskytovatelů SaaS, kteří vyhovují stanoveným kritériím podnikové informační strategie
Primární zodpovědnost (role):	Vedoucí IT
Účastníci (role):	Sponzor, Koordinátor SaaS
Vstupy procesu:	Stávající podniková informační strategie Seznam možných poskytovatelů SaaS
Výstupy procesu:	Aktualizovaná podniková informační strategie Aktualizovaný seznam poskytovatelů SaaS, kteří vyhovují aktualizované podnikové informační strategii Návrh smlouvy (smluv) s poskytovateli SaaS
Techniky:	Vyhodnocení míry strategické závislosti na poskytovateli SaaS
Kritické faktory úspěchu:	Nepodceňování bezpečnostních rizik spolupráce s poskytovateli SaaS

4.3.3.2 Proces: specifikace požadavků a rozsahu integračního projektu, PB01

V rámci tohoto procesu je nutné alespoň rámcově specifikovat SW požadavky od uživatelů vzhledem k danému řešení. Agilní metodiky nepředpokládají detailní znalost všech aspektů celého řešení v úvodu projektu, avšak uživatelé by měli být schopni nastínit alespoň rámcově požadavky na základě současné situace postupu prací (angl. Workflow). Součástí tohoto procesu by měla být schůzka většiny členů projektu (rolí).

Z vlastních zkušeností a také studie (Joha, 2012) vyplývá, že vývojář by se měl snažit získat odpovědi zejména na následující otázky:

- Popis funkčnosti vlastními slovy od uživatelů. Vývojář musí předpokládat, že uživatelé nebudou schopni uvažovat v modelech, grafech a podobných pomůckách;
- Kdo bude výsledné řešení používat. Bude se jednat pouze o interní uživatele nebo také externí uživatelé mimo danou organizaci;
- Z jakých zařízení budou uživatelé danou část informačního systému používat;
- Budou se uživatelé rekrutovat z jednoho regionu (ČR či EU) nebo více regionů (MEA, US atd.);

- V jakých časových úsecích se bude řešení používat. Bude vyžadován přístup 24/7 nebo pouze ve všední dny?;
- Rámcový odhad objemu dat, který si budeme vyměňovat mezi aplikací informačního systému a SaaS.

V rámci tohoto procesu je také vhodné ověřit užitnou hodnotu, kterou uživatelům použití dané služby (SaaS) přináší oproti čistě on-premise řešení. Tato částečně „povrchní“ informace může vývojáři poskytnout cenné vstupy při řešení následných otázek při samotné implementaci. Navíc je žádoucí v tomto kroku ověřit, zda jsou přínosy použití SaaS opravdu pro organizaci výhodné z důvodů naznačených v předchozí kapitole, resp. zda výhody převýší vyšší míru rizika (viz Kapitola 4.3.3.1).

Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 14).

Tabulka 14: Popis procesu Specifikace požadavků a rozsahu integračního projektu

Název procesu:	Specifikace požadavků a rozsahu integračního projektu
ID procesu	PB01
Cíle:	Shromáždění uživatelských požadavků od řešeného projektu Získání odpovědí na otázky mapující specifické požadavky směrem k poskytovateli SaaS
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	Sponzor, Uživatel, Vedoucí IT
Vstupy procesu:	Stávající (angl. Workflow) uživatelů
Výstupy procesu:	Popis uživatelských požadavků Odpovědi na klíčové otázky mapující specifické požadavky směrem k poskytovateli SaaS
Techniky:	Dotazování
Kritické faktory úspěchu:	Aktivní přístup uživatelů a vývojáře

4.3.3.3 Proces: výběr služby SaaS a vlastnosti služby SaaS, PB02

Při zahájení tohoto procesu je již obvykle jedna konkrétní služba SaaS vybrána a je zajištěn její soulad s informační strategií podniku, ale v případě, že existuje více odpovídajících variant SaaS, musíme provést výběr nejvhodnějšího kandidáta a toho postoupit do dalších fází projektu.

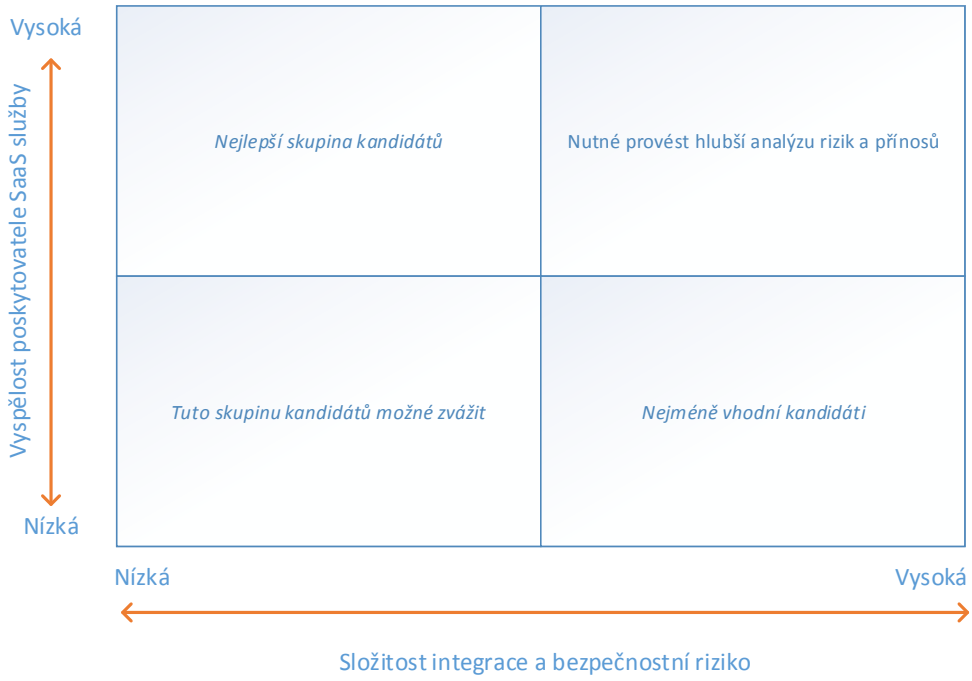
K výběru jednoho kandidáta nám může pomoci analýza a vzájemné srovnání dvou základních dimenzí u každého poskytovatele SaaS (volně zpracováno dle studie (Joha, 20012)):

- **Vyspělost (angl. Maturity):** kombinovaný indikátor vyspělosti poskytovatele SaaS;

- **Složitost integrace a bezpečnostní riziko:** kombinovaný indikátor relativní složitosti integrace a bezpečnostního rizika.

Při aplikaci metodického vzoru pro konkrétní společnost a projekt je nutné, aby si společnost dosadila svou metodiku výpočtu obou zmíněných indikátorů. Každá společnost je jiná a každý projekt je jiný. U méně kritických projektů, postačí měkčí kritéria výpočtu, u více kritických projektů je nutné použít přísnější měřítka. Možný námět hodnocení kritičnosti projektu nabízí metodický rámec MeFIS a metoda popsána v (Buchalcevová, 2005, strana 77).

Pro výpočet samotných dimenzí může společnost použít například následující vstupní indikátory: dostupnost, škálovatelnost, výkon, bezpečnost, zálohování či přenositelnost. Zkřížením obou dimenzí získáme graf (viz Obrázek 25), jehož kvadranty seskupují podobné poskytovatele SaaS dle hodnocených kritérií.



Obrázek 25: Dimenze pro výběr nejvhodnějšího poskytovatele SaaS (volně zpracováno dle (Joha, 2012); upraveno autorem)

Mimo zmíněných dvou dimenzí (viz Obrázek 25), můžeme použít i další dodatečná kritéria hodnocení různých poskytovatelů SaaS (volně zpracováno dle studie (Joha, 2012)):

- **Výsledná hodnota pro uživatele:** uživatelé by měli mít možnost ovlivnit výběr kandidáta z hlediska uživatelských vlastností;
- **Cílová skupina uživatelů:** měli bychom zohlednit otázku, zda cílovou skupinou uživatelů jsou interní pracovníci nebo také externí subjekty;

- **Architektura na straně poskytovatele:** poskytování prostředků pro více klientů (angl. Multi-tenant), který je základem cloudových služeb, je možné technicky docílit několika různými cestami (sdílené databáze, separátní databáze, virtualizace atd.). Každá má jiná bezpečnostní rizika a jiné možnosti správy;
- **Dodatečný rozsah služeb z hlediska integrace:** někteří poskytovatelé nabízejí rozšiřující služby ve formě již hotových integračních řešení s dostupnými platformami atd.;
- **Cenová struktura nákladů**;
- **Soulad s požadavky kladenými na řešený integrační projekt** (viz Kapitola 4.3.3.2)**.**

Výstupem tohoto procesu musí být volba pouze jednoho kandidáta, který postoupí do dalších fází projektu. Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 15).

Tabulka 15: Popis procesu Výběr služby SaaS a vlastnosti služby SaaS

Název procesu:	Výběr služby SaaS a vlastnosti služby SaaS
ID procesu	PB02
Cíle:	Výběr jednoho poskytovatele SaaS (v případě více kandidátů) Zjištění dodatečných informací od vybraného poskytovatele SaaS a zjištění souladu s požadavky kladenými na dané řešení (včetně specifických požadavků)
Primární zodpovědnost (role):	Vedoucí IT
Účastníci (role):	Sponzor, Uživatel, Koordinátor služeb SaaS, Vývojář
Vstupy procesu:	Seznam potenciálních poskytovatelů SaaS Popis uživatelských požadavků Odpovědi na klíčové otázky mapující specifické požadavky směrem k poskytovateli SaaS
Výstupy procesu:	Jeden vybraný poskytovatel SaaS
Techniky:	Dvoudimenzionální vyhodnocení vyspělosti poskytovatele a složitosti integrace jeho služeb Srovnání dodatečných parametrů a jejich vyhodnocení Dotazování
Kritické faktory úspěchu:	Správná volba a metodika výpočtu kombinovaných ukazatelů

4.3.3.4 Proces: volba metody integrace, PB03

Po výběru jednoho kandidáta je nutné opatřit si detailní technickou dokumentaci, aby bylo možné zodpovědně přistoupit k procesu, ve kterém zvolíme *integrační metodu*. Lze zjednodušeně říci, že tento proces musí spolehlivě identifikovat možné metody integrace, které pro splnění projektu přicházejí v úvahu.

Různé metody integrace byly detailně vysvětleny v kapitole 3.3 a kapitole 3.4, shrnuji:

- Integrace na úrovni dat (sdílené databáze);
- Integrace za pomocí přenosu souborů;
- Integrace na úrovni aplikačních rozhraní (API);
- Integrace na úrovni metod/objektů (angl. Remoting);
- Integrace na úrovni uživatelského rozhraní (UI).

Stojí za zmínku, proč od sebe odděluji *programovatelná rozhraní* a *remoting*. U běžného API předpokládám, že formát výměny dat je sice dán (např. JSON atd.), ale nemusí být zřejmá sémantika posílaných dat a cílová aplikace musí provádět transformaci těchto dat do struktur v daném vývojovém prostředí (zpravidla objektů). Při použití *metody remoting* předpokládám použití takového formátu dat, u kterého je zřejmá sémantika (např. SOAP) a můžeme velice rychle odlišit metodu od vlastnosti atd.

Jakmile je zřejmé, jaké všechny integrační metody daná služba (SaaS) podporuje, je nutné zvolit takovou variantu, která nabízí nejlepší možnou shodu s naším záměrem.

Jako vodítko můžeme použít míru obtížnosti integrace zobrazenou v následující tabulce (viz Tabulka 16). V řádcích jsou uvedené metody integrace, které požadujeme pro náš záměr (na základě uživatelských požadavků). Ve sloupcích jsou uvedené metody integrace, které nabízí daná služba (SaaS). V průsečíku jsou uvedené hodnoty 0-1, které autor teoreticky odhadl dle své zkušenosti[6], a vyjadřují teoretickou složitost integrace v dané kombinaci metod. Čím je hodnota vyšší, tím je integrace snazší. Hodnota 1 značí naprostou shodu. To je ideální varianta (diagonála v tabulce).

Tabulka 16: Míra složitosti integrace v různých kombinacích požadovaných metod a nabízených

Nabízí / *Požaduje*	**Sdílené databáze**	**Přenos souborů**	**API**	**Remoting**	**UI**
Sdílené databáze	**1**	0,6	0,6	0,6	0,1
Přenos souborů	0,6	**1**	0,7	0,7	
API	0,9	0,6	**1**	0,9 - 1	0,1
Remoting	0,7	0,6	0,9 - 1	**1**	0,1
UI	N/A				**1**

[6] Tyto hodnoty by bylo jistě vhodně empiricky ověřit v rámci průzkumu na vzorku společností

Uvedenou tabulku je také možné použít takovým způsobem, že ve sloupci nalezneme nabízený způsob integrace a najdeme takový řádek, který dosahuje nejvyšší hodnotu. To bude zároveň metoda integrace, kterou bychom měli na straně informačního systému preferovat.

Pro dokreslení důvodů zvolení konkrétní míry složitosti integrace v dané kombinaci uvádím popis vybraných vzájemných variant:

- **Požadujeme *Přenos souborů*, avšak služba nabízí *sdílenou databázi*:** v tomto případě budeme muset navrhnout převodník, který daný soubor přečte a data v něm uložená zapíše do vzdálené databáze služby SaaS;
- **Požadujeme *Remoting*, ale služba nabízí *sdílenou databázi*:** relativně vysoká míra shody, avšak bude nutné vyřešit konverzi dat – zavést mapování objektů do formy relačních tabulek což nemusí být úplně triviální úloha;
- **Požadujeme *API*, nicméně služba nabízí *pouze UI*:** velice složitá úloha s nejistým výsledkem. Bude zapotřebí „parsovat“ webové rozhraní a to převádět do formy datových struktur.

Při realizaci tohoto procesu si může každá společnost dosadit vlastní hodnoty míry obtížnosti při aplikaci této metodiky. Míra obtížnosti může vycházet z jejich současného programového vybavení. Pokud již například používají některou integrační platformu třetí strany, ta již může standardně nabízet některé typy transformací.

V případě, že výsledná míra obtížnosti není pro danou situaci akceptovatelná, je nutné se vrátit zpět k procesu výběru služby SaaS (Proces PB02).

V další fázi tohoto procesu je nutné zvolit vhodný formát výměny dat. V tomto ohledu bychom měli respektovat *principy dlouhodobě udržitelné podnikové architektury* (jednoduchost, udržitelnost, znovupoužitelnost, oddělení funkcionality od technologie a platformní nezávislost) a také preferovat *volnou vazbu* před *těsnou*.

Primárním výsledkem tohoto procesu je zvolená *integrační metoda* a *formát výměny dat*. Obě tyto vlastnosti determinují volbu integračního scénáře, kterou provedeme v následujícím procesu. Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 17).

Tabulka 17: Popis procesu: Volba metody integrace

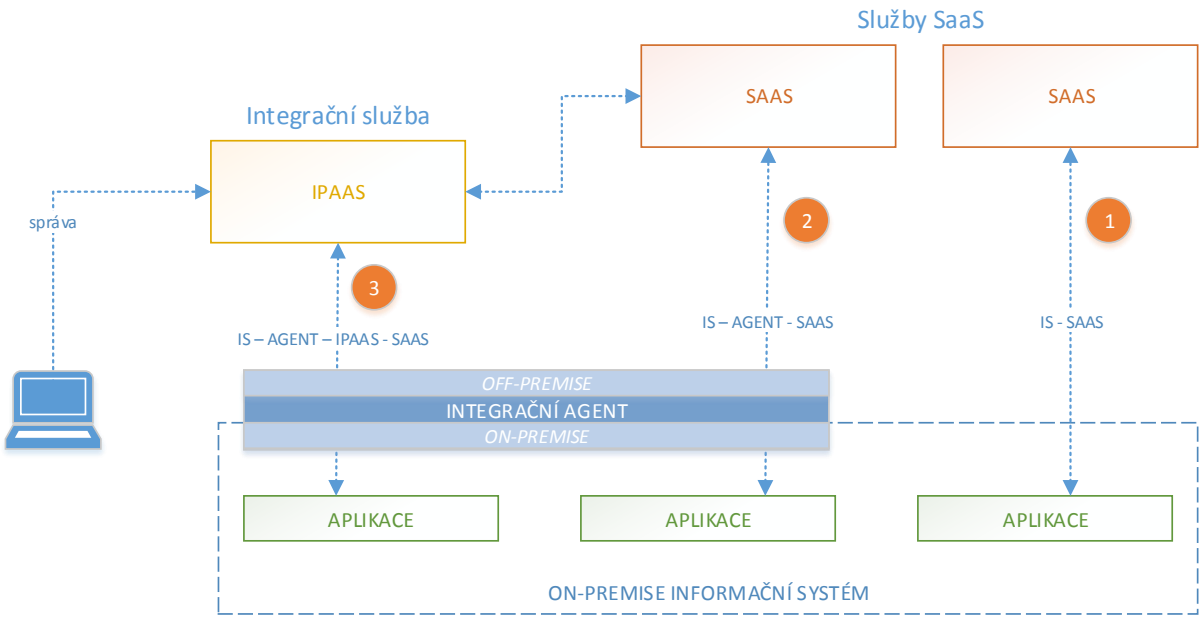
Název procesu:	Získání technické dokumentace SaaS a volba metody integrace
ID procesu	PB03

Cíle:	Získání detailní technické dokumentace (v případě, že již není k dispozici) Volba metody integrace Volba formátu výměny dat
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	Koordinátor služeb SaaS
Vstupy procesu:	Technická dokumentace služby (SaaS) v případě, že je již k dispozici Popis uživatelských požadavků Odpovědi na klíčové otázky mapující specifické požadavky směrem k poskytovateli SaaS
Výstupy procesu:	Konkrétní metoda integrace Konkrétní formát výměny dat Technická dokumentace služby (SaaS)
Techniky:	Dvoudimenzionální vyhodnocení míry složitosti integrace Analýza uživatelských požadavků, specifických požadavků a volba odpovídajícího formátu výměny dat
Kritické faktory úspěchu:	Dosažení maximální míry shody mezi nabízenou metodou integrace a požadovanou Respektování principů metodického vzoru

4.3.3.5 Proces: volba integračního scénáře, PB04

V rámci tohoto procesu je nutné provést volbu integračního scénáře. Dostupné zdroje, např. (Narasimha, 2012), (Hai, 2009) nebo (Joha, 2012) se vesměs shodují v tom, že se nabízí tři základní integrační scénáře (viz Obrázek 26):

- **Varianta 1:** Informační systém komunikuje přímo se službou SaaS. Tuto variantu označuji zkratkou *IS-SAAS*;
- **Varianta 2:** Informační systém komunikuje se službou SaaS skrze integračního agenta. Tuto variantu označuji zkratkou *IS-AGENT-SAAS*;
- **Varianta 3:** Informační systém komunikuje se službou SaaS skrze integračního agenta a integrační služby. Tuto variantu označuji zkratkou *IS-AGENT-ISAAS-SAAS*;



Obrázek 26: Základní integrační scénáře (volně zpracováno dle (Hai, 2009); upraveno autorem)

Varianta *IS-SAAS* je z hlediska návrhu nejjednodušší, avšak může vytvářet špagetovou integraci v případě, že informační systém využívá více různých služeb SaaS. Tato varianta není také vhodná, pokud je nutné provést nějakou transformaci dat „po cestě“. Další charakteristikou tohoto scénáře je obvykle výhradně *synchronní* způsob komunikace.

Pokud proto nevyhovuje základní varianta, můžeme použít variantu *IS-AGENT-SAAS*, která při komunikaci se službou SaaS používá mezičlánek ve formě integračního agenta. Integrační agent je *middleware* a může mít celou řadu funkcí, které se liší dle povahy řešeného projektu. Na tento druh komunikace můžeme aplikovat různé *návrhové vzory* (viz Kapitola 4.3.5). Tato varianta řeší také případnou špagetovou integraci v případě zapojení více služeb SaaS. V takovém případě integrační agent slouží jako integrační hub (či sběrnice zpráv). Agent může být vyvíjen vlastními silami[7] nebo můžeme nasadit existující hotová řešení dodavatelů integračních platforem. I samotný integrační agent může být provozován jako on-premise nebo off-premise řešení. Avšak toto rozlišení není pro další řešení projektu podstatné, a proto jej dále nebudu rozvíjet. Z hlediska způsobu komunikace může tato varianta být *synchronní* i *asynchronní*.

Třetím scénářem je varianta *IS-AGENT-IPAAS-SAAS*, kdy integrační agent není dodáván jako hotové on-premise řešení, ale je poskytován také formou služby prostřednictvím off-premise řešení (angl. Integration Platform as a Service, iPaaS). Na tuto variantu se vztahují všechny obecné vlastnosti modelu SaaS (nulové fixní náklady, řešení okamžitě k dispozici, pay-as-you-go model atd.) a toto řešení navíc umožnuje správu odkudkoli. Stojí za zmínku, proč se při tomto scénáři stále předpokládá agent, protože teoreticky by měla postačit komunikace přímo s iPaaS. Ano, agent není nutný vždy, ale

[7] Zajímavé koncepční řešení agenta v roli integračního hubu vlastními silami popisuje Wonil (2012)

dle (Hai, 2009), se jedná o standardní praxi. Použití agenta totiž výrazně rozšiřuje integrační schopnosti platformy (notifikace, push model atd.). Jinak o tomto scénáři platí vše, co bylo řečeno u varianty *IS-AGENT-SAAS*.

Zkombinováním možných scénářů integrace a možných variant zajištění daného řešení získáme následující tabulku možných variant (viz Tabulka 18).

Tabulka 18: Shrnutí integračních scénářů

Scénář	Integrační agent vyvíjen interně	Integrační agent vyvíjen externě	Integrační agent jako hotové řešení/iPaaS
IS-SAAS	*Bez nutnosti integračního agenta*		
IS-AGENT-SAAS	ANO/NE	ANO/NE	ANO/NE
IS-AGENT-IPAAS-SAAS	ANO/NE	ANO/NE	ANO/NE

V případě, že charakter projektu bude vyžadovat scénář IS-AGENT-SAAS nebo IS-AGENT-IPAAS-SAAS je nutné specifikovat, kdo bude zodpovědný za tvorbu komponenty integračního agenta a podle toho najmout další subjekt nebo rozšířit stávající kontrakt s vývojářem o tuto komponentu.

Jak již bylo naznačeno, volba určitého scénáře částečně předurčuje způsob navržené komunikace (synchronní, asynchronní). V případě použití integračního agenta musíme navíc upřesnit, jaký typ komunikace požadujeme (viz Kapitola 3.3.1).

Obecně platí, že implementace *asynchronní komunikace* je složitější a tím pádem může být i dražší a nemusí se hodit pro všechny případy. Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 19).

Tabulka 19: Popis procesu: Volba integračního scénáře

Název procesu:	Volba integračního scénáře
ID procesu	PB04
Cíle:	Volba integračního scénáře Volba typu komunikace (synchronní/asynchronní) Rozšíření stávající smlouvy s vývojářem nebo najmutí nového subjektu
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	Koordinátor služeb třetí strany, Sponzor

Vstupy procesu:	Technická dokumentace služby (SaaS) Popis uživatelských požadavků Odpovědi na klíčové otázky mapující specifické požadavky směrem k poskytovateli SaaS
Výstupy procesu:	Zvolený integrační scénář Zvolený typ komunikace (synchronní/asynchronní) Případně upravená smlouva s vývojářem nebo nová smlouva s novým subjektem
Techniky:	Analýza uživatelských požadavků, specifických požadavků
Kritické faktory úspěchu:	Respektování principů Princip budování dlouhodobě udržitelné podnikové architektury a Principu zachování bezpečného přístupu a komunikace

4.3.3.6 Proces: analýza možných omezení využití služby (SaaS), PB05

Cílem tohoto procesu je ověření, zda *zvolená služba (SaaS)*, *zvolená integrační metoda* a *zvolený integrační scénář* jsou společně kompatibilní s celkovými požadavky kladenými na dané integrační řešení.

Můžeme se například soustředit na následující oblasti:

- **Rámcový odhad objemu dat, který si budeme vyměňovat mezi aplikací informačního systému a SaaS:** některé integrační scénáře nejsou vhodné pro přenos extrémního množství dat či mohou existovat omezení množství procesovaných dat. Také mohou existovat omezení co do počtu vzdálených volání z jedné IP adresy na straně služby atd.;
- **Zpoždění:** některé integrační scénáře znamenají prodlevu mezi synchronizací dat (data se mohou v cílovém sytému projevit se zpožděním) a mohou znamenat prodlevy na straně uživatelského rozhraní;
- **Archivace dat na straně služby SaaS:** některé služby SaaS mohou mít nastavené archivační politiky, které jsou v rozporu s potřebami informačního systému (např. aktivní zpřístupnění dat pouze za poslední rok atd.);
- **Časové prodlevy:** některé služby mohou určité časové úseky používat pro vlastní administraci atd. Jinými slovy je dobré ověřit, zda daná služba je navržena s ohledem na provoz 24/7, pokud jej daný projekt vyžaduje.

V případě, že zjistíme významné omezení a nesoulad s požadavky, je nutné se vrátit do procesu volby metody či integračního scénáře a revidovat zvolené řešení pro zajištění vyšší míry kompatibility.
V případě, že zvolená služba (SaaS) je kompatibilní s naším projektovým záměrem, je možné přistoupit k podepsání SLA s daným poskytovatelem SaaS.

Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 20).

Tabulka 20: Popis procesu: Analýza možných omezení využití služby (SaaS)

Název procesu:	Analýza možných omezení využití služby (SaaS)
ID procesu	PB05
Cíle:	Analýza omezení dané služby, integračního scénáře a metody Podepsání SLA s vybraným kandidátem s požadovanými parametry služby
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	Koordinátor služeb SaaS
Vstupy procesu:	Technická dokumentace služby (SaaS) Popis uživatelských požadavků Odpovědi na klíčové otázky mapující specifické požadavky směrem k poskytovateli SaaS Zvolený integrační scénář Zvolený typ komunikace (synchronní/asynchronní) Konkrétní metoda integrace Konkrétní formát výměny dat
Výstupy procesu:	Rozhodnutí zda pokračovat v projektu s danými parametry Podepsaná smlouva SLA s vybraným poskytovatelem
Techniky:	Komplexní analýza uživatelských požadavků a specifických požadavků vzhledem ke zvoleným parametrům integračního řešení
Kritické faktory úspěchu:	Respektování principů Princip budování dlouhodobě udržitelné podnikové architektury a Principu zachování bezpečného přístupu a komunikace

4.3.3.7 Proces: detailní návrh jednotlivých komponent integračního řešení, PC01

V této fázi a tomto procesu má vývojář již všechny dostupné informace pro zahájení plánování samotného vývoje všech komponent. Dle zvolené metody integrace, formátu výměny dat a zvoleného integračního scénáře je nutné provést volbu vývojových platforem pro dané integrační řešení a sestavit platformě závislý model fungování integračního řešení.

Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 21).

Tabulka 21: Popis procesu: Detailní návrh jednotlivých komponent integračního řešení

Název procesu:	Detailní návrh jednotlivých komponent integračního řešení
ID procesu	PC01
Cíle:	Návrh na platformě závislého modelu integračního řešení
Primární zodpovědnost (role):	Vývojář

Účastníci (role):	
Vstupy procesu:	Technická dokumentace služby (SaaS) Popis uživatelských požadavků Odpovědi na klíčové otázky mapující specifické požadavky směrem k poskytovateli SaaS Zvolený integrační scénář Zvolený typ komunikace (synchronní/asynchronní) Konkrétní metoda integrace Konkrétní formát výměny dat
Výstupy procesu:	Platformě závislý model Plán vývoje jednotlivých komponent
Techniky:	Analýza uživatelských požadavků, specifických požadavků a aktivní využití návrhových vzorů
Kritické faktory úspěchu:	Jednoduchost a účelnost řešení Používání návrhových vzorů

4.3.3.8 Proces: získání funkčního prototypu, PC02

V rámci tohoto procesu je žádoucí prokázat základní životaschopnost navrženého platformě závislého modelu nějakým jednoduchým, ale funkčním prototypem. Není nutné tento prototyp konzultovat s cílovou skupinou uživatelů, ale měl by umožnit provést měření základních výkonnostních parametrů (rychlost odezvy, množství dat atd.).

Jelikož se zpravidla bude jednat o komunikaci v rámci heterogenních prostředí, míra možného neúspěchu je relativně vysoká a případné nedostatky by měl být schopen pomoci tento prototyp identifikovat. Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 22).

Tabulka 22: Popis procesu: Získání funkčního prototypu

Název procesu:	Získání funkčního prototypu
ID procesu	PC02
Cíle:	Vytvoření funkčního prototypu
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	
Vstupy procesu:	Technická dokumentace služby (SaaS) Platformě závislý model Plán vývoje jednotlivých komponent

Výstupy procesu:	Funkční prototyp integračního řešení
Techniky:	Objektově orientované programování Strukturované programování
Kritické faktory úspěchu:	Používání návrhových vzorů Výběr vhodných metod pro testování

4.3.3.9 Proces: implementace řešení, PD01

V tomto procesu začne vývojář implementovat vlastní řešení za použití iterativního přístupu (viz kapitola 4.3.1). Výsledkem tohoto procesu je fungující řešení nebo jeho část, která je předána do procesu testování skupinou uživatelů (a sbírání zpětné vazby od uživatelů). Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 23).

Tabulka 23: Popis procesu: Implementace řešení

Název procesu:	Implementace řešení
ID procesu	PD01
Cíle:	Vytvoření fungujícího řešení nebo jeho části
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	
Vstupy procesu:	Technická dokumentace služby (SaaS) Popis uživatelských požadavků Platformě závislý model Plán vývoje jednotlivých komponent Funkční prototyp integračního řešení
Výstupy procesu:	Fungující řešení nebo jeho funkční část, kterou je možné testovat skupinou uživatelů
Techniky:	Objektově orientované programování Strukturované programování Iterativní vývoj
Kritické faktory úspěchu:	Používání návrhových vzorů Výběr vhodných metod pro testování

4.3.3.10 Proces: testování řešení, PD02

Tento proces slouží pro otestování fungujícího řešení skupinou uživatelů. Vývojář předloží fungující řešení nebo jeho část k posouzení cílové skupině uživatelů a „sbírá“ zpětnou vazbu, kterou může v dalších iteracích zvážit a zapracovat. Relativně krátké iterace (např. cca 14 dnů) se opakují do doby, než dané řešení dosáhne určitého stupně vyspělosti, aby mohlo být provedeno nasazení.

Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 24).

Tabulka 24: Popis procesu: Testování řešení

Název procesu:	Testování řešení
ID procesu	PD02
Cíle:	Otestování řešení nebo jeho části skupinou uživatelů a sběr zpětné vazby
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	Uživatelé
Vstupy procesu:	Fungující řešení nebo jeho funkční část
Výstupy procesu:	Zpětná vazba od uživatelů
Techniky:	Testování
Kritické faktory úspěchu:	Aktivní a konstruktivní přístup uživatelů

4.3.3.11 Proces: nasazení řešení, PE01

V rámci tohoto procesu dochází k nasazení fungujícího a vyzkoušeného řešení a jeho ostrého nasazení k užívání mezi uživateli. Výsledkem tohoto procesu je fungující řešení nasazené do ostrého provozu.

Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 25).

Tabulka 25: Popis procesu: Nasazení řešení

Název procesu:	Nasazení řešení
ID procesu	PE01
Cíle:	Distribuce řešení a jeho nasazení
Primární zodpovědnost (role):	Vývojář
Účastníci (role):	Uživatelé, Vedoucí IT
Vstupy procesu:	Fungující řešení nebo jeho funkční část
Výstupy procesu:	Distribuce řešení (angl. Release)
Techniky:	
Kritické faktory úspěchu:	Zajištění stability řešení i při dodání nových verzí některých částí (Buchalcevová, 2005)

4.3.3.12 Proces: administrace řešení, PF01

Cílem tohoto procesu je administrace integračního řešení. Výstupem tohoto procesu je fungující integrační systém, který pracuje se všemi aktuálními parametry. Součástí tohoto procesu je také monitoring dodržování parametrů sjednaných v SLA a zajištění případných korekci ze strany poskytovatele SaaS.

V případě požadavků na změnu daného řešení je iniciován celý vývojový proces od začátku. Shrnutí všech důležitých parametrů procesu je možné nalézt v následující tabulce (viz Tabulka 26).

Tabulka 26: Popis procesu: Administrace řešení

Název procesu:	Administrace řešení
ID procesu	PF01
Cíle:	Distribuce řešení a jeho nasazení Průběžné vyhodnocení dodržování sjednaného SLA
Primární zodpovědnost (role):	Vedoucí IT
Účastníci (role):	Uživatelé
Vstupy procesu:	Nasazené řešení
Výstupy procesu:	Nasazené řešení pracující se všemi aktuálními parametry Případně seznam pořadavků na úpravu daného řešení
Techniky:	Kontrola a monitoring SW i HW
Kritické faktory úspěchu:	Pravidelný monitoring funkčnosti řešení

4.3.4 Principy metodického vzoru

Tato kapitola popisuje základní principy, na kterých je postaven navržený metodický vzor.

4.3.4.1 Iterativní princip vývoje aplikací

Iterativní princip vývoje aplikací je využíván v agilních metodikách vývoje SW a uplatnění tohoto principu je blíže popsáno v kapitole popisující jednotlivé fáze (kapitola 4.3.1).

4.3.4.2 Princip budování dlouhodobě udržitelné podnikové architektury

Na základě zkušeností z budování podnikové architektury (kapitola 3.4), lze shrnout základní vlastnosti dlouhodobě udržitelné podnikové architektury:

- **Jednoduchost**;

- **Flexibilita a udržitelnost**;
- **Znovupoužitelnost**;
- **Oddělení funkcionality a technologie.**

Při realizaci integračního projektu bychom se měli snažit zajistit maximální možnou míru respektování uvedených vlastností.

4.3.4.3 Princip zachování bezpečného přístupu a komunikace

Tento princip klade důraz na bezpečnost používání služeb SaaS. Bezpečnost můžeme rozdělit na dvě dimenze:

- **Bezpečná a důvěrná autorizace:** měli bychom zajistit, aby autorizace k přístupu služeb SaaS byla důvěrná a design integračního řešení neohrozil odhalení autorizačních údajů jeho uživatelů;
- **Bezpečná a důvěrná komunikace:** při komunikaci se službou SaaS bychom měli preferovat šifrované komunikační cesty.

Častým prohřeškem proti tomuto principu bývá umístění přístupových údajů přímo v útrobách informačního systému (do útrob zkompilovaných aplikací), kde existuje vysoké riziko vyzrazení. Je třeba mít stále na paměti, že cokoli umístíme na přístupových terminálech svých klientů, může být zneužito.

4.3.5 Návrhové vzory

Tato kapitola identifikuje možné návrhové vzory (viz Tabulka 27) vhodné pro řešení některých úloh v oblasti integrace služeb SaaS.

Z větší části se jedná o návrhové vzory výměny dat, které mohou pomoci s aplikací různých způsobů komunikace v rámci heterogenních systémů. Součástí každého vzoru je také referenční zdroj a popis situace, ve které je možné daný vzor zvážit a případně aplikovat.

Návrhové vzory, které jsou součástí tohoto metodického vzoru, vychází zejména ze struktury uvedené v (Hentrich, 2006).

Tabulka 27: Návrhové vzory

Vzor	Popis problému a řešení
Kanonický datový model (angl. Canonical Data Model) (Hohpe, 2004)	V případě, že systémy používají různé datové formáty, můžeme navrhnout vlastní datový formát, který není závislý ani na jednom z uvedených systémů a zajistit patřičné konverze.
Doplnění obsahu zprávy (angl. Content Enricher) (Hohpe, 2004)	V případě, že při komunikaci nemá zdrojový systém všechny dostupné informace, můžeme použít komunikační middleware, který požadované informace zajistí a danou zprávu obohatí.
Balíček zprávy (angl. Envelope Wrapper) (Hohpe, 2004)	V případě, že náš formát dat není kompatibilní s přenosovou cestou, je možné vytvořit speciální zprávu, která kompatibilní je a původní zprávu do ní zakódovat. Na konci přenosu je možné původní data extrahovat.
Překladač zpráv (angl. Message Translator) (Hohpe, 2004)	V případě nekompatibilního formátu dat mezi zdrojovým systémem a cílovým systémem je možné vytvořit middleware, který zajistí správný překlad zpráv.

5 Praktická aplikace navrženého metodického vzoru

Tato kapitola popisuje aplikaci navrženého metodického vzoru na skutečnou situaci v českém podniku. Jedná se o reálnou situaci v jedné malé české cestovní kanceláři do 25 zaměstnanců, která se rozhodla část svého podnikového informačního systému začít provozovat prostřednictvím aplikace třetí strany (SaaS). Oproti výchozí situaci, ve které byly všechny interní procesy řízeny a spravovány programovým vybavením provozovaným výhradně „uvnitř“ společnosti, se daná společnost dostala do zcela nové situace.

Daný informační systém standardně nabízí svým uživatelům veškerou funkcionalitu skrze webové rozhraní a společnost má proto na výběr dva možné scénáře:

- Převést veškerou agendu do daného informačního systému poskytovaného formou SaaS, což by znamenalo přesun většiny funkcí a dat ze stávajícího on-premise informačního systému do off-premise řešení;
- Integrovat data spravovaná daným systémem do stávajícího podnikového informačního systému a využít pouze část funkcionality daného systému.

Společnost se rozhodla pro druhou variantu z několika následujících důvodů:

- Nechtěla být zcela závislá na daném poskytovateli služby SaaS;
- Některé funkce jsou v daném informačním systému velice propracované a komplexní, avšak jiné jsou pouze standardnějšího charakteru bez možností dodatečných změn.

Z těchto důvodů bylo nutné provést integraci služby SaaS do stávajícího informačního systému. Cílovým stavem, tj. po úplném zavedení služby SaaS, se uvnitř dané společnosti bude používat pouze *softwarový klient*, který zabezpečeně komunikuje se službou (SaaS), která je provozována mimo danou společnost (angl. Off-premise). V dlouhodobém horizontu by měl tento softwarový klient zabezpečit řadu funkcionality pro většinu procesů uvnitř společnosti. V první fázi bude sloužit primárně pro přípravu ceníků, protože možnosti prezentace ceníků jsou v dané službě SaaS příliš základní a nejsou vhodné pro obchodní prezentaci a využití. Avšak na druhou stranu služba poskytuje velice propracovaný systém správy a administrace ceníků, a proto se společnost rozhodla tuto skutečnost využít a v první fázi integrovat správu a prezentaci ceníků z dané služby SaaS.

Klientská aplikace používaná pro přístup ke službě SaaS bude distribuována všem interním zaměstnancům společnosti, ale také několika klíčovým partnerům dané společnosti. Okruh příjemců této aplikace je tedy značně široký, a proto také riziko vyzrazení sady autorizačních údajů je značně vysoké. Je také nutné počítat s možnou fluktuací zaměstnanců či partnerů, a proto je nutné v

neposlední řadě vymyslet způsob, který umožní přístup v průběhu používání aplikace konkrétním jedincům odebrat.

Tato kapitola si klade za cíl navrhnout a popsat možné řešení těchto problémů za použití navrženého metodického vzoru, a také může sloužit jako nástin konkrétní integrace SaaS do podnikových informačních systémů malých a středních firem.

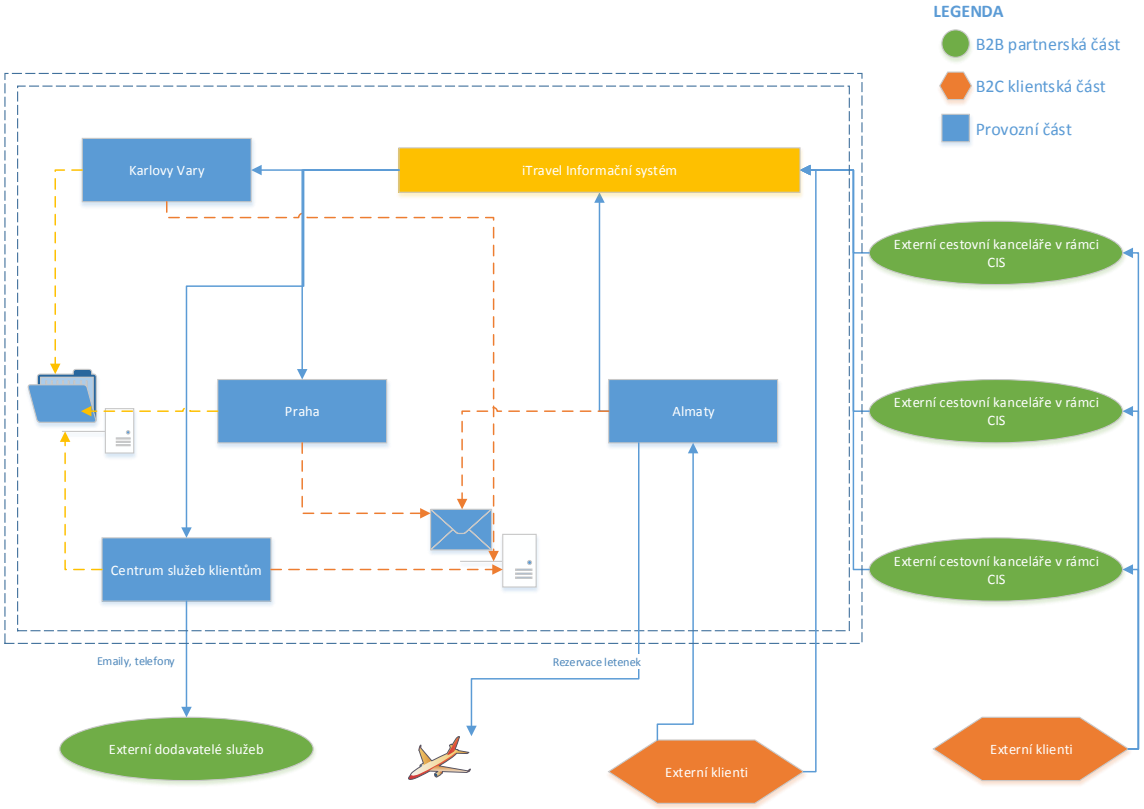
5.1 Stručná charakteristika vybraného podniku a výchozí situace

Pro účely této práce není popis vybraného podniku klíčový, avšak pro bližší seznámení s celým kontextem popisuji důležité aspekty fungovaní podniku a jeho informačních potřeb.

Jedná se o malou českou společnost do 25 zaměstnanců podnikající v oblasti cestovního ruchu, konkrétně v oblasti „incomingové“ turistiky, a zajištující doprovodné služby s touto oblastí související. Společnost se opírá o tři základní pilíře svého podnikání:

- Ubytování
- Transfery (individuální silniční přeprava osob)
- Exkurze

Základní fungování celé společnosti je ilustrováno přiloženým schématem (viz Obrázek 27) včetně znázornění primárních informačních toků/požadavků. Plné modré šipky znázorňují tok základních požadavků zpravidla od klienta směrem ke společnosti. Oranžové přerušované šipky znázorňují datový tok ke sdíleným IT prostředkům společnosti.



Obrázek 27: Základní princip fungování vybrané společnosti

V uvedeném obrázku jsou barevně odlišené komponenty informačního systému iTravel, které jsou řešené službou SaaS. Jedná se o následující části:

- Provozní část (angl. Back-office), která poskytuje vnitropodnikový informační systém pro potřeby cestovních kanceláří s možností evidence celého obchodního případu od jeho objednávky po jeho realizaci;
- B2C klientská část s možností online rezervací. Tato část primárně nabízí nadefinované služby cestovní kanceláře, např. ubytovací kapacity (popis, foto dokumentace, ceny atd.), široké veřejnosti a zprostředkovává jejich rezervaci;
- B2B partnerská část. Tato část produktu zprostředkovává rozhraní také pro partnery, kteří tímto způsobem mohou zadávat objednávky služeb dané cestovní kanceláře.

Jak je zřejmé z předchozích odstavců, uvedený informační systém se stará o evidenci všech nabízených ubytovacích kapacit včetně jednotkových cen. Tento katalog ubytování (včetně cen) je spravován pobočkou z Prahy a nabízen klientům z CIS prostřednictvím webové prezentace a také oficiálním zastoupením v Almaty.

Historicky byly tyto funkce informačního systému zajištovány kombinací různých technologií z oblasti kancelářského software, avšak postupem času se společnost rozhodla převést tuto agendu pod

aplikaci třetí strany nabízenou formou SaaS. Informační systém se jmenuje iTravel[8] a je vyvíjen a spravován společností Lemax, kterou z jejich webové prezentace cituji: *„iTravel je komplexní informační systém pro efektní správu a řízení cestovních kanceláří formou SaaS.“*

5.2 Informační systém iTravel

Informační systém iTravel je nabízen formou SaaS a svou funkcionalitu nabízí prostřednictvím webového uživatelského rozhraní (viz Obrázek 28).

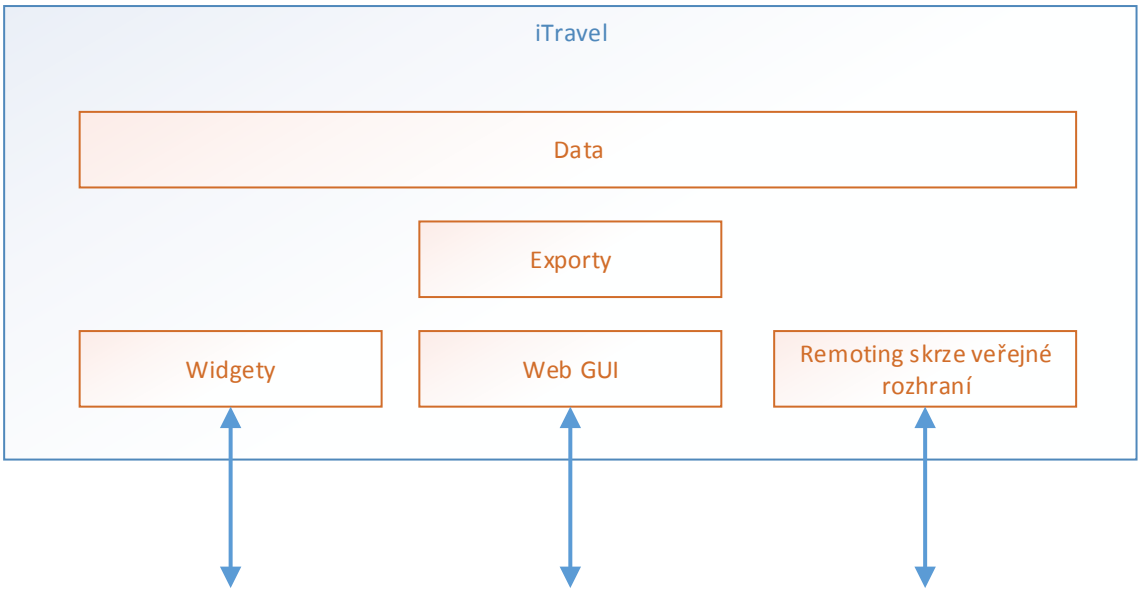
V rámci tohoto primárně uživatelsky orientovaného přístupu nabízí různé předdefinované exporty do HTML souborů a tím je možné získat data v uživatelsky přívětivé podobě a případně je také vytisknout. Tento způsob je primárně určen pro uživatele a není zcela připraven pro automatizované počítačové zpracování, protože exportované soubory jsou již zformátované bez jakékoli sémantické diferenciace (lze přečíst jednotlivé tabulky, ale už není jasné, jaká informace je v daných rádcích a sloupcích).

Jako možnou alternativou přístupu do systému jsou tzv. webové widgety. Tedy části připravených webových stránek, které se vygenerují na základě poskytnutých parametrů. Tento způsob primárně slouží pro integraci na úrovni webových prezentací (pomocí iFrame). Tento způsob také není zcela vhodný pro automatizované zpracování, protože informace generuje v primárně uživatelsky přívětivé podobě.

Třetí možností je volání vzdálených metod a výměny objektů (angl. Remoting), které systém podporuje prostřednictvím technologie webových služeb (angl. Web Services). Toto je jediný způsob, který je určen pro automatizované počítačové zpracování, a naopak zcela nevhodný pro běžné uživatele.

Všechny tři popsané způsoby lze shrnout prostřednictvím následujícího obrázku (viz Obrázek 28).

[8] http://www.itravelsoftware.com/



Obrázek 28: Možnosti komunikace s informačním systémem iTravel

5.3 Aplikace navrženého metodického vzoru

V této kapitole popíši integrační řešení, které jsem navrhl a realizoval za použití navrženého metodického vzoru. U každé fáze uvedu časovou náročnost daného kroku, který daný krok skutečně zabral, aby si uživatelé metodiky udělali rámcovou představu o relativní náročnosti jednotlivých procesů.

Při řešení tohoto projektu byli k dispozici pouze dva lidé:

- Zástupce společnosti, který byl zodpovědný za komunikaci s poskytovatelem SaaS řešení a také za komunikaci dovnitř společnosti (s uživateli). V neposlední řadě zastupoval také zájmy sponzora a vedoucího IT;
- Externí vývojář, který zodpovídal za realizaci integračního řešení.

5.3.1 Aktualizace podnikové politiky správy a řízení IT

V době řešení tohoto integračního projektu byl již daný informační systém používán a společnost jej do svého původního informačního systému integrovala prostřednictvím různých exportních sestav a externích souborů. Jinými slovy, podniková politika byla již upravena takovým způsobem, že připouštěla existenci SaaS jako součást řešení informačních potřeb společnosti.

Pokud vezmeme v úvahu hodnocení poskytovatele dle strategické závislosti, která byla popsána dříve (viz Kapitola 4.3.3.1), vychází zmíněná služba (SaaS) jako riziková (v kvadrantu off-premise, public cloud). Tato skutečnost může také odpovědět na otázku, proč se společnosti rozhodla jít raději cestou integrace než svěření všech procesů a dat do rukou jednoho poskytovatele SaaS.

5.3.2 Specifikace požadavků a rozsahu integračního projektu

Na základě schůzek vývojáře se zástupcem společnosti byly sestaveny následující základní požadavky integračního projektu:

- Zajistit automatizovaný přístup k ceníkům hotelů uložených v systému iTravel a umožnit vygenerování sestav dle různých kritérií;
- Výsledné integrační řešení budou v první řadě používat zaměstnanci společnosti, ale v delším horizontu také externí partneři;
- Primární funkcí bude tvorba tiskových sestav ve formátu Excel (minimálně ve verzi 2007 a vyšší), který může být případně dále distribuován externím partnerům;
- Systém bude využíván při každé aktualizaci ceníků, cca jednou týdně. Avšak na začátku sezóny i třeba jednou denně;
- K systému budou přistupovat zaměstnanci primárně z ČR, ale je možné vyžadovat přístup také během noci a víkendů;
- Do systému se předpokládá přístup výhradně prostřednictvím osobních počítačů na platformě Windows (minimálně verze 7 a vyšší). Přístup přes mobilní telefony není vyžadován;
- Počet hotelů v databázi se očekává na cca 250.

Časová náročnost zmíněných schůzek byla cca 16 člověkohodin. Do této časové dotace jsem započítal oba zaměstnance, tj. vývojáře, ale také zástupce společnosti.

5.3.3 Výběr služby SaaS

Do této fáze projektu byla vybrána již pouze jedna služba SaaS a proto tato fáze nebyla zapotřebí. Daná služba byla v tu dobu již společností používána cca 2 měsíce, a proto bylo zřejmé, že funkcionalita nabízené služby odpovídá požadavkům na ni kladených.

5.3.4 Volba metody integrace

V této fázi projektu bylo jasné, že pro integraci se teoreticky mohou použít dvě metody:

- Přenos souborů;
- Volání vzdáleného API.

Obě zmíněné metody iTravel nabízí (viz Kapitola 5.2), avšak v tuto chvíli ještě nebylo zřejmé, jaká metoda bude vhodnější.

Proto byla vyžádána dokumentace k nabízenému rozhraní (API) a také požadována ukázka vygenerovaného výstupu ceníků ve formátu Excel, který nabízí samotný systém iTravel. Ukázka

technické dokumentace je uvedena v příloze (viz Kapitola 10.1). Z důvodu rozsáhlosti celého materiálu je přiloženo pouze několik stran pro získání základní představy. V přílohách je také k dispozici ukázka exportních sestav poskytovaných systémem iTravel (viz Kapitola 10.2).

Celá exportní sestava trpí základní vadou, která ji diskvalifikuje pro další použití v podnikové praxi. Všechny ceníky hotelů jsou uvedeny na jednom listu, a proto tisk či vyhledávání jsou značně nepraktické. Analýzou výsledného souboru bylo navíc zjištěno, že iTravel negeneruje nativní formát Excel, ale generuje ceníky ve formátu HTML, který uloží s koncovkou „.xls“. Excel správně rozezná formát HTML při otevírání, a proto otevření ve výsledku provede, ale z tohoto důvodu ceníky nejsou a ani nemohou být na jednotlivých listech. I když přistoupíme k souboru ve správném formátu (HTML), jeho automatizované zpracování je velice nepraktické:

- HTML soubor je tvořen pouze standardními HTML značkami a neobsahuje žádné sémantické značky, pomocí kterých by bylo možné získat konkrétnější informaci o významu obsažený dat, např. v jakém řádku jsou časové periody, v jakém typy pokojů a v jakém jsou slevy atd.
- Zpracování daného souboru by bylo proto odkázáno na heuristiku, která by byla značně riziková a nepřispěla by k dlouhodobě udržitelnému řešení;
- Navíc systém založený na přenosu souborů by postrádal automatizovanou složku, která byla požadována. V takovém případě by soubor musel vždy nechat vygenerovat uživatel a výsledný soubor nechat zpracovat v jiné aplikaci.

S ohledem na princip „Princip budování dlouhodobě udržitelné podnikové architektury“ (viz Kapitola 4.3.4.2) a výše zmíněných důvodů bylo rozhodnuto, že přenos souborů nebude pro integraci použit.

Pro integraci proto zůstala „ve hře“ ještě metoda volání vzdáleného API, avšak bylo nutné ověřit, zda rozsah zveřejněných funkcí a metod bude dostatečný pro získání všech informací potřebných k sestavení ceníků. Jak bylo zdůrazněno v kapitole pojednávající o službách (viz Kapitola 2.5), služby komunikují s okolním světem skrze rozhraní, a proto i já jsem byl v této otázce odkázán na rozsah zveřejněného rozhraní služby.

Znamenalo to detailní analýzu obdržené dokumentace, po jejímž prostudování jsem došel k závěru, že rozsah zveřejněných API funkcí je dostatečný a sestavení ceníku je možné. Takže v této fázi bylo možné konstatovat, že víme, kterou metodu integrace v systému iTravel využijeme: metoda Remoting.

Jinými slovy jsme zafixovali sloupec ve zmíněné tabulce (viz Tabulka 16). Dle navrženého metodického vzoru je nutné zafixovat ještě řádek a zvolit metodu integrace na straně stávajícího informačního systému. Dle zmíněné tabulky by bylo ideální variantou použití stejné metody integrace, tj. Remoting, při kterém by dosažená shoda byla maximální možná.

Při opětovném prostudování požadavků na integrační řešení jsem došel k závěru, že v této otázce nejsou specifikovány žádné konkrétní požadavky, a proto bylo možné předpokládat, že budu moci použít Remoting i na straně stávajícího informačního systému pro dosažení maximální možné shody.

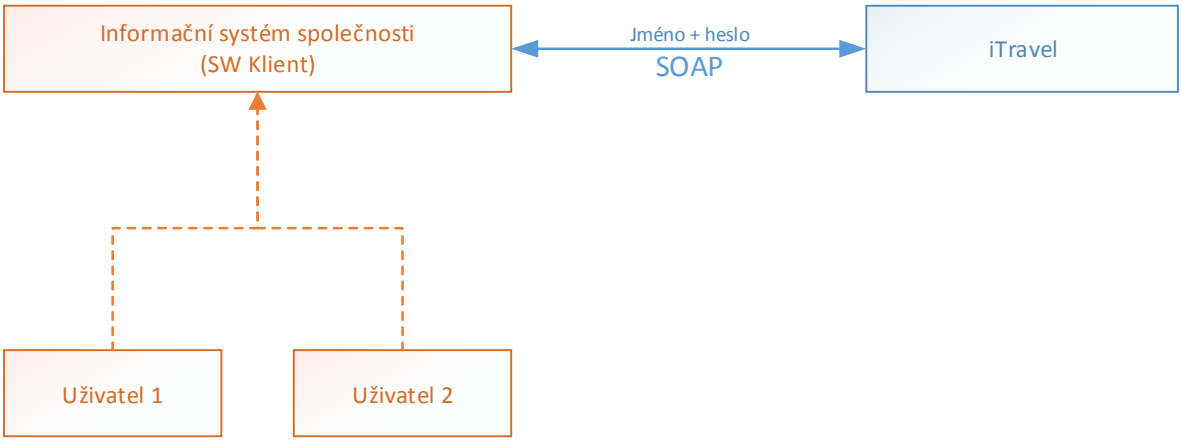
Výsledkem tohoto kroku je proto rozhodnutí, použít *Remoting* jako výslednou metodu integrace. Jelikož iTravel používá jako formát výměny dat SOAP, i my použijeme stejný formát pro zajištění maximální shody.

Celková časová náročnost tohoto procesu byla cca 40 člověkohodin.

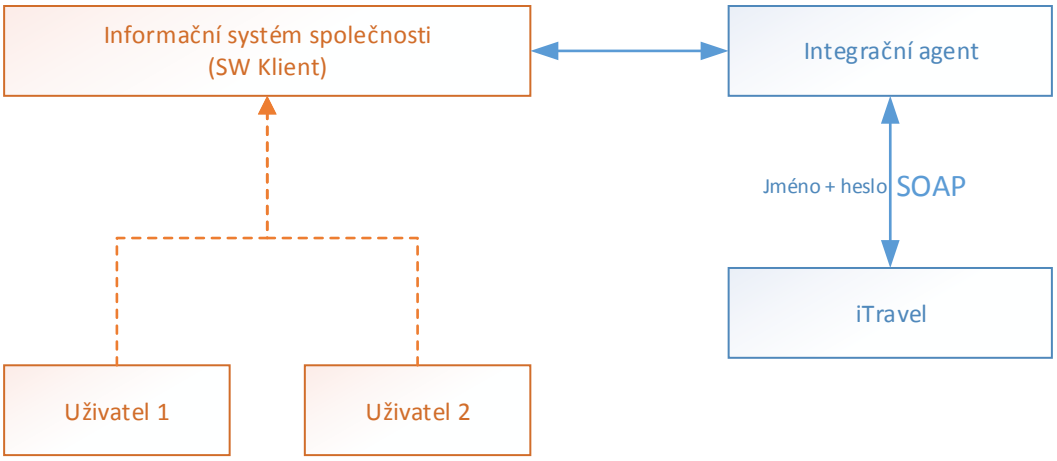
5.3.5 Volba integračního scénáře

V rámci tohoto procesu bylo nutné zvolit jeden ze tří možných integračních scénářů. Z požadavků na dané integrační řešení nevyplývala nutnost použití integrační platformy poskytované formou služby (iPaaS), a proto se dále srovnávaly pouze scénáře integrace s integračním agentem nebo bez integračního agenta.

Obě situace ilustruji formou následujících obrázků (viz Obrázek 29, Obrázek 30).



Obrázek 29: Možný integrační scénář bez integračního agenta



Obrázek 30: Možný integrační scénář s integračním agentem

Jak je naznačeno na obou schématech, služba iTravel vyžaduje pro autorizaci svých klientů jednu sadu autorizačních údajů (jméno a heslo), a tato sada je pouze jedna pro celou společnost. Je proto velice klíčové tuto sadu autorizačních údajů ochránit před „okolním světem“ a umožnit její použití pouze v autorizovaných případech.

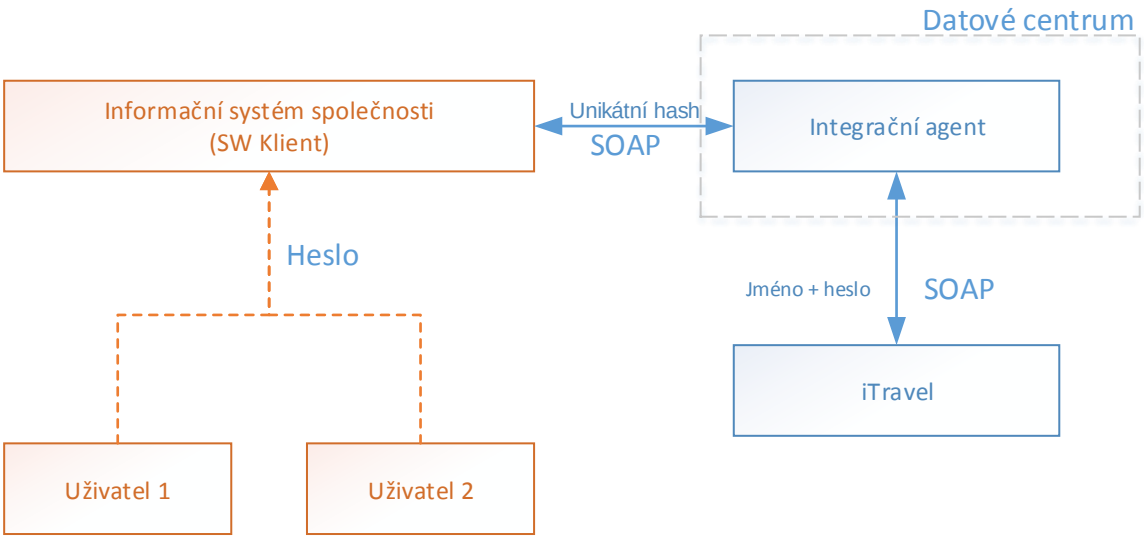
Obecné principy asymetrického šifrování jsou dostatečné známé a dobře popsané, avšak v podnikové praxi bývá hlavním problémem jakým způsobem bezpečně uchovat privátní klíč, aby se minimalizovalo jeho neoprávněné zneužití.

Na tomto místě je vhodné připomenout „*Princip zachování bezpečného přístupu a komunikace*“, který je součástí navrženého metodického vzoru. Dle tohoto principu bylo nutné vyřešit otázku, kam uložit autorizační údaje, aby se zabránilo jejich zneužití. Zároveň bychom se měli vyhnout pokušení uložit tyto údaje přímo do aplikace, která bude distribuována jednotlivým uživatelům. V takovém případě bychom celý systém vystavovali příliš vysokému riziku, protože jakkoli zkompilovaná aplikace není imunní proti různým formám reversního inženýrství a autorizační údaje vložené do těla komponenty by bylo možné získat.

Možným řešením naznačeného problému by byla nutnost vložení těchto údajů samotným uživatelem při každém použití systému. Toto by také nebyla vhodná varianta z důvodu možné fluktuace zaměstnanců. Při každém odchodu by bylo nutné z bezpečnostních důvodů změnit centrální autorizaci, a bylo by nutné tuto novou autorizaci opět sdílet se všemi ostatními uživateli.

Z výše naznačených důvodů bylo nutné integrační scénář bez integračního agenta zcela vyloučit, protože nebylo možné zaručit požadovanou úroveň zabezpečení. Při dalším návrhu integračního projektu jsem proto zvolil scénář s integračním agentem se záměrem vložit bezpečnostní údaje do něj a umístit jej někam mimo informační systém. Pro realizaci jsem se inspiroval návrhovým vzorem „Doplnění obsahu zprávy“, jehož identifikace je součástí metodického vzoru (viz Kapitola 4.3.5).

Zvolený integrační scénář je přiložen níže (viz Obrázek 31).



Obrázek 31: Výsledný integrační scénář

Navržené řešení lze shrnout postupem komunikace následujícím způsobem:

- Uživatel zadá přístupové heslo do samotné klientské aplikace. Pozn.: toto heslo nemá žádnou přímou spojitost s přístupovými údaji do systému iTravel;
- Zadané heslo bude použito k vygenerování bezpečného 128bitového „kódu“, který vznikne hashovací funkcí;
- Tento hash bude použit v SOAP požadavku pro systém iTravel místo platných autorizačních údajů;
- Avšak požadavek nebude odeslán přímo do systému iTravel, ale do integračního agenta (middleware), který je umístěn v zabezpečeném datovém centru jako serverová komponenta a je výhradně spravována danou cestovní kanceláří. Jinými slovy, tato komponenta není součástí systému iTravel;
- Middleware komponenta převezme SOAP požadavek a použije zadaný hash kód pro vyhledání daného uživatele;
- Pokud hash kód souhlasí (uživatel existuje), middleware vymění autorizační údaje v SOAP požadavku za správné iTravel autorizační údaje a spolu se SOAP požadavkem jsou odeslány do systému iTravel;
- Jakmile systém iTravel odpoví, middleware stejnou odpověď odešle zpět do klientské aplikace.

Důležitým aspektem celé komunikace je, že autorizační údaje systému iTravel jsou použité pouze mezi middleware a systémem iTravel a nikdy se nedostanou k samotné klientské aplikaci. Pro zvýšení celkového zabezpečení můžeme autorizační údaje iTravel zašifrovat také na straně middlware a jako

klíč použít zmíněný 128bitový hash „zadaný“ uživatelem. To znamená, že pokud by se útočník dostal k serverové části middleware (toto je vysoce nepravděpodobné), nalezl by iTravel autorizační údaje pouze v zašifrované podobě a ani analýza zdrojových kódů by nikam nevedla.

V tomto případě by pro získání iTravel autorizačních údajů musel útočník zcizit alespoň jedno uživatelské heslo a navíc prolomit zabezpečení serveru, což je již velice nepravděpodobná kombinace.

Navržený systém obsahuje ještě jednu velice užitečnou vlastnost – možnost centrálního řízení přístupu k systému iTravel. Jelikož je middleware zcela zodpovědný za přidělování přístupu, je možné centrálně přístup odejmout libovolnému uživateli.

Celá komunikace je navržená jako synchronní, což není proti žádnému z definovaných požadavků kladených na integrační řešení.

Dle navrženého metodického vzoru bylo nutné ještě vyřešit otázku, kdo bude zodpovědný za tvorbu integračního agenta. V tomto konkrétním případě se stal zodpovědným vývojář a byl smluvně rozšířen rozsah zakázky.

Celková náročnost tohoto kroku byla cca 42 člověkohodin.

5.3.6 Analýza možných omezení využití služby SaaS

V tomto kroku bylo nutné ověřit, zda navržená integrační metoda a scénář netrpí některými omezeními. Při analýze nebylo zjištěno žádné výkonové omezení ze strany poskytovatele služby, ale pouze bezpečnostní.

Poskytovatel služby neměl v době realizace projektu zabezpečené HTTPs připojení, což ve výsledku znamenalo, že data mezi integračním agentem a službou nebyla šifrovaná. Vliv na celkovou bezpečnost řešení byl nízký, a proto se pokračovalo v realizaci projektu. Riziko odposlechu mezi zabezpečeným datovým centrem je mnohem nižší, než riziko odposlechu mezi klientskou aplikací a integračním agentem.

Výstupem tohoto procesu bylo identifikováno následující omezení:

- Komunikace je zcela šifrována mezi informačním systémem a integračním agentem;
- Integrační agent komunikaci dešifruje a dále posílá požadavky na službu nešifrovaně.

Průběh tohoto procesu poukázal na možnou nedokonalost navrženého metodického vzoru. V tomto případě nebyly škody signifikantní, ale bylo by jistě lepší na podobnou bezpečnostní slabinu přijít již v raných fázích a zavést tento kontrolní bod již v průběhu procesu „Volba metody integrace, PB03“.

Celková náročnost tohoto kroku byla cca 10 člověkohodin.

5.3.7 Detailní návrh jednotlivých komponent

V tomto kroku jsem měl již všechny potřebné informace pro přistoupení k detailnímu návrhu všech komponent. Jelikož v dané společnosti byl celkově Excel velice často využívaným nástrojem, rozhodl jsem se pro jeho využití také v tomto případě a zvolil jsem Excel jako běhové prostředí v kombinaci s platformou .NET (konkrétně komponenta angl. Visual Studio Tools for Office).

Aplikační platforma .NET mi poskytla výkonné aplikační běhové prostředí, které nativně podporuje konzumaci vzdálených služeb založených na protokolu SOAP (angl. Web Services) a Excel mi poskytl nepřeberné prezentační schopnosti jejichž implementace byla v tomto módu velice rychlá a efektivní. Vše je navíc integrováno do platformy Microsoft Office, a proto se výsledný systém bude pro koncové uživatele tvářit kompaktně.

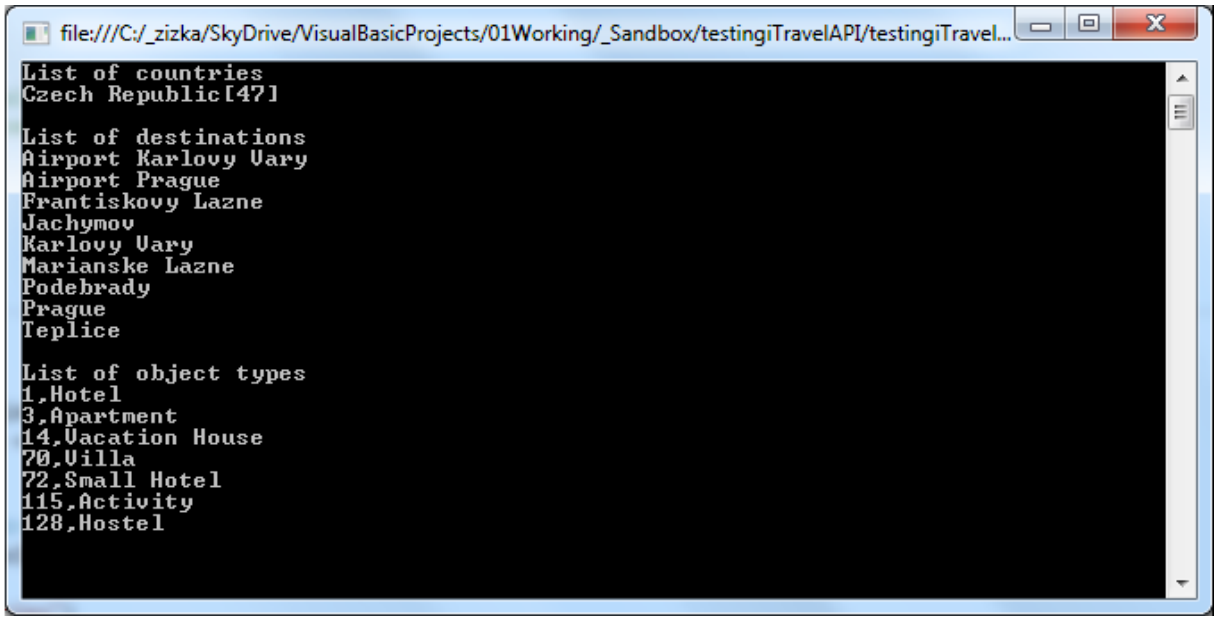
Pro realizaci integračního agenta jsem se rozhodl použít jazyk PHP běžící na webovém serveru Apache v zabezpečeném off-premise datovém centru. Daná cestovní kancelář již vlastnila virtuální server a tak jsem jej využil pro hostování integračního agenta.

Časová náročnost detailního návrhu byla cca 8 člověkohodin.

5.3.8 Získání funkčního prototypu

Pro ověření funkčnosti navrženého řešení jsem připravil jednoduchou konzolovou aplikaci, která dokázala komunikovat s danou službou napřímo.

Výstupy poskytovala formou konzolových výpisů a všechny výsledky navíc ukládala na disk, a proto jsem mohl provádět analýzu vrácených objektových modelů ve formátu XML. Příklad podobného testovacího výpisu je zobrazen níže (viz Obrázek 32).



Obrázek 32: Ukázka výstupu funkčního prototypu

Tento funkční prototyp se ukázal jako velice výhodný i v pozdějších fázích projektu, protože umožnoval velice rychle testovat různé scénáře, které bylo jinak obtížné simulovat v reálné aplikaci zatížené již nabobtnalou aplikační logikou.

Časová náročnost tohoto procesu byla cca 18 člověkohodin.

5.3.9 Implementace řešení, testování řešení

Proces implementace měl dvě části. V první části jsem vytvořil funkční řešení jako doplněk do MS Excel. Excel obecně nabízí dva režimy svých doplňků:

- Aplikační;
- Doplněk svázaný s dokumentem.

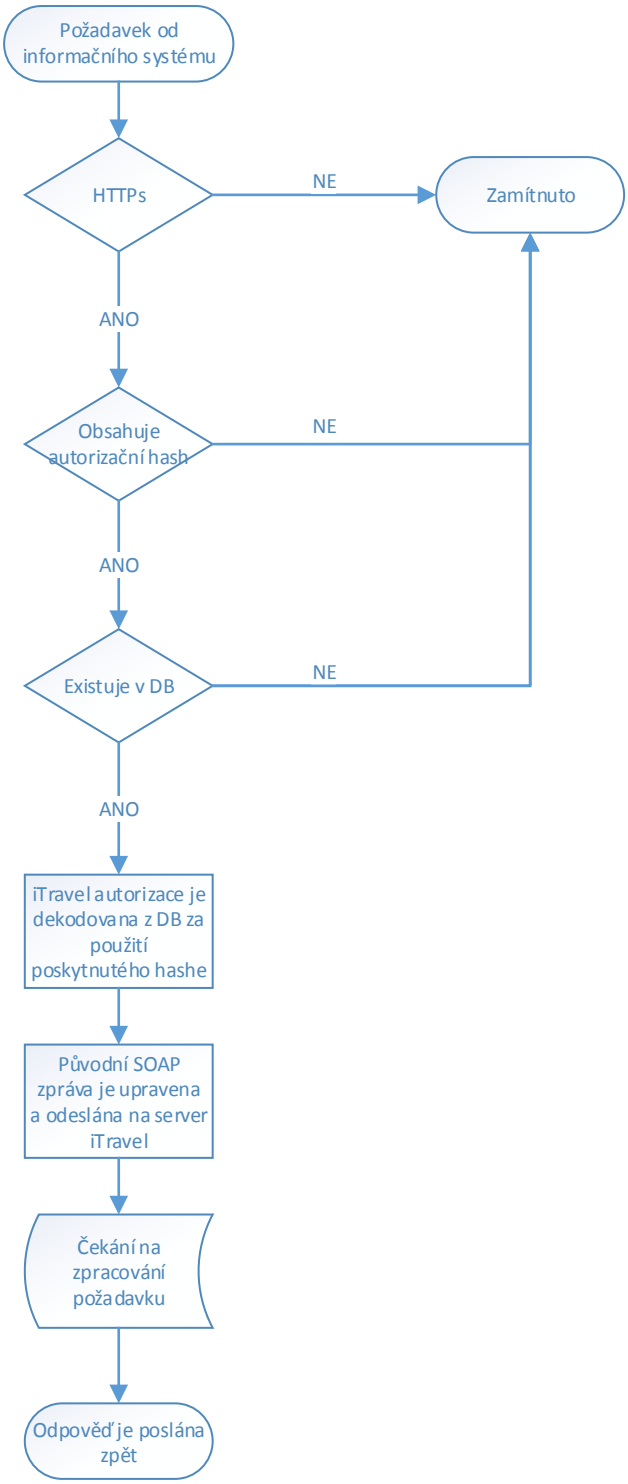
Aplikační doplněk předpokládá, že se bude používat bez ohledu na právě otevřený sešit. Proto jsem zvolil doplněk svázaný s konkrétním dokumentem.

Spolu s otevřením konkrétního sešitu se vždy zavede také daný doplněk a uživatel tak může plně využít jeho funkcionalitu. Navíc doplněk před zavedením vyžaduje, aby byl na daném počítači nainstalován důvěryhodný certifikát, kterým byl doplněk podepsán. Tímto způsobem lze kontrolovat, na kterých počítačích je možné doplněk používat. Pokud není nainstalován příslušný certifikát, doplněk se nezinicializuje.

Uživatelské rozhraní aplikace je zobrazeno v příloze (viz Kapitola 10.3). Je zřejmé, že uživatel může nadefinovat řadu parametrů před samotným generováním a aplikace navíc připraví obsah a každý ceník vygeneruje na vlastní sešit (viz Kapitola 10.4). Tímto způsobem je možné flexibilně sdílet

jednotlivé ceníky bez náročné editace. Parametrické generování ceníků a možnost zobrazit obsah nebylo se samotným systémem iTravel možné.

V druhé části jsem vytvořil integračního agenta. Jeho průběh je ilustrován běhovým diagramem postupu (viz Obrázek 33). Byl navržen pro *synchronní* způsob komunikace a v první verzi je spojen s informačním systémem *těsnou* vazbou, protože informační systém musí vědět jakým způsobem poslat zakódovaný hash jako součást SOAP zprávy. V dalších verzích aplikace je plánováno vytvořit spojení vazbou volnou.



Obrázek 33: Postup integračním agentem

Součástí implementace bylo neustálé testování vlastního řešení. Díky použití technologie ClickOnce[9] společnosti Microsoft, bylo možné zveřejnovat nové sestavení v libovolně krátkých intervalech a zástupce společnosti se tak mohl aktivně podílet na testování a poskytovat zpětnou vazbu již během vývoje integračního řešení. V tomto ohledu se ukázal iterativní princip vývoje jako dobrá volba.

[9] https://msdn.microsoft.com/en-us/library/142dbbz4(v=vs.90).aspx

Implementace řešení trvala cca 3 týdny (120 člověkohodin).

5.3.10 Nasazení řešení

Jakmile řešení dosáhlo určitého stupně vyspělosti, proběhlo jeho nasazení mezi zaměstnance společnosti. Opět byla použita technologie ClickOnce, která zajistila, že zaměstnanci stačilo otevřít dotyčný sešit, technologie ověřila nainstalované certifikáty a pokud se shodovaly, provedlo se stažení doplňku a jeho instalace.

Tato technologie je také zodpovědná za aktualizaci doplňku v případě existence novější verze.

Koncová pracovní stanice pro provoz vyžaduje minimálně následující komponenty:

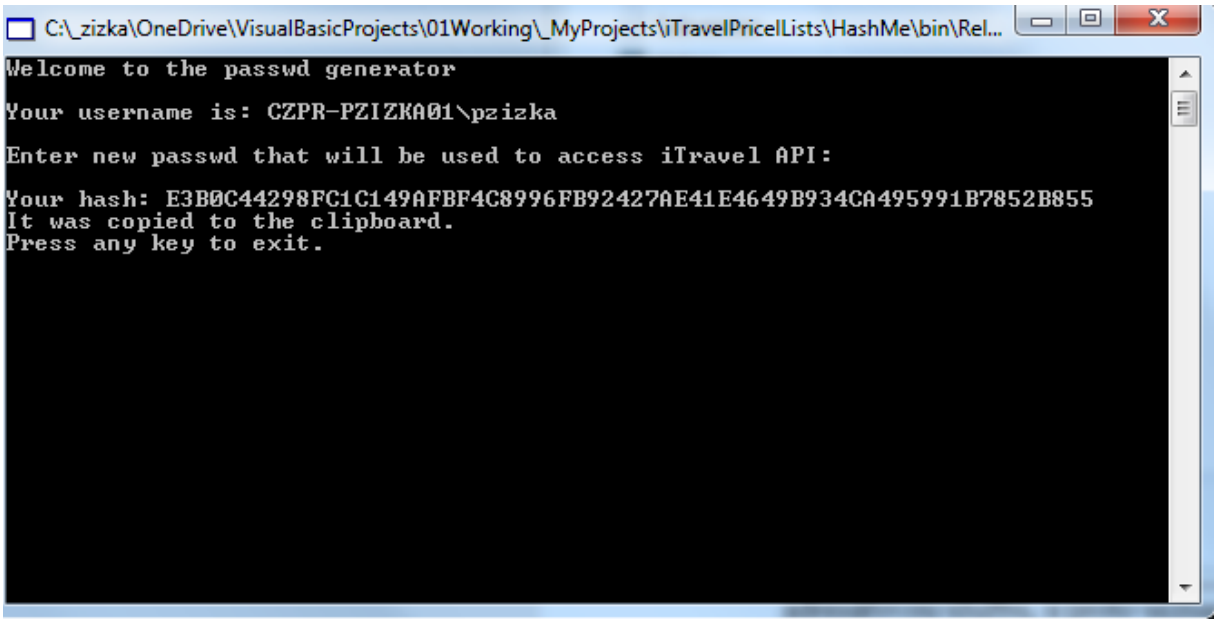
- Microsoft Office 2007 nebo vyšší
- .NET Framework verze 4.5
- Visual Studio Studio Tools For Office Runtime

Všechny zmíněné komponenty jsou standardní součástí výbavy v dané společnosti, a proto se celý systém pro koncové uživatele tváří jako součástí jednoho XLS dokumentu.

5.3.11 Administrace řešení

Dané řešení vyžaduje administraci integračního agenta, u kterého je nutné „pravidelně“ aktualizovat uživatele a jejich autorizační údaje. V úvodní verzi nebyl integrační agent vázán na žádnou adresářovou službu, a proto seznam oprávněných uživatelů byl zcela nezávislý od seznamu podnikových uživatelů.

Z tohoto důvodu byla vytvořena pomocná aplikace pro tvorbu unikátního klíče pro každého uživatele. Tímto způsobem ani vývojář neznal přístupová hesla jednotlivých uživatelů. Uživatelé spustili aplikaci a vývojář získal již pouze unikátní klíč a doménovou identifikaci daného uživatele (viz Obrázek 34).



Obrázek 34: Ukázka pomocné aplikace pro generování přístupových údajů

Avšak na tomto místě je nutné přiznat, že systém oddělených uživatelů od seznamu podnikových uživatelů jde proti principu „Princip budování dlouhodobě udržitelné podnikové architektury“ a v dalších verzích by bylo určitě záhodno seznam uživatelů navázat na adresářovou službu dané společnosti.

5.4 Shrnutí navrženého integračního řešení

V této kapitole jsem popsal implementaci konkrétní integrační úlohy z reálného prostředí českého podniku. Postupoval jsem podle navrženého metodického vzoru a prokázalo se, že integrátora vede úspěšně k cíli.

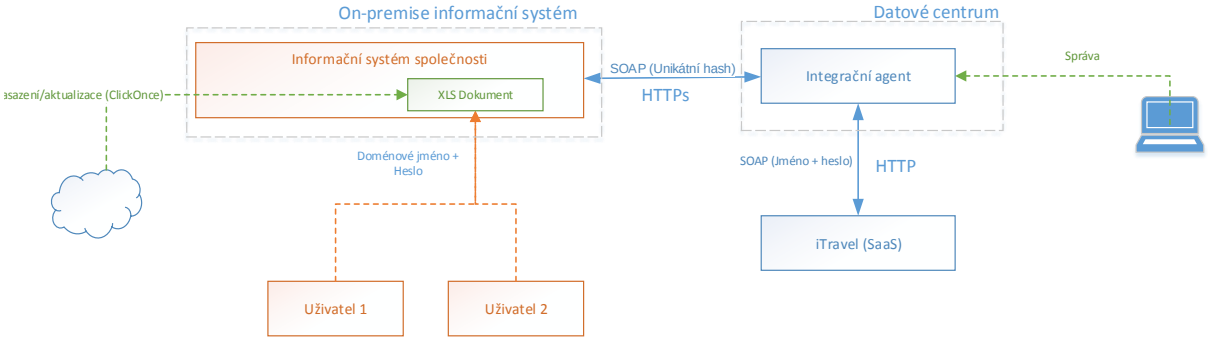
Během implementace konkrétního řešení se podařilo také nastínit, že dodržování „Principu zachování bezpečného přístupu a komunikace“ je nutné řešit již v rané fázi projektu, protože může významným způsobem ovlivnit volbu integračního scénáře.

Proces „administrace řešení“ poukázal na slabiny vývoje integračního agenta „na koleni“. Při tomto scénáři je nutné vyřešit integraci podnikových uživatelů takovým způsobem, aby byl dodržen „Princip budování dlouhodobě udržitelné podnikové architektury“. Z tohoto důvodu by bylo záhodné, pokusit se ověřit parametry a dostupnost již hotových integračních platforem (např. Informatica, MuleSoft atd.), které by mohly ve výsledku být výhodnější a kompletnější volbou. Každopádně navržený metodický vzor podobnou variantu zahrnuje, takže není nutná revize v tomto směru.

Obecně se použití scénáře s integračním agentem vyplatilo, protože celý systém měl ještě jednu zatím nepojmenovanou vlastnost. Odstínění služby SaaS od samotného informačního systému společnosti. To znamená, že pokud by se změnila lokace (URL) služby SaaS je možné tuto změnu aplikovat pouze

na straně integračního agenta (serveru) a samotná klientská aplikace zůstane beze změn. Tímto způsobem se podařilo zvolnit závislost na momentální konfiguraci služby SaaS.

Výsledné řešení integrace lze shrnout přiloženým obrázkem (viz Obrázek 35).



Obrázek 35: Výsledné řešení integrace

6 Závěr

V této práci jsem si kladl za cíl navrhnout metodiku integrace služeb SaaS do on-premise informačního systému malého a středního podniku. V kapitole 2 a kapitole 3 jsem terminologicky vymezil oblast práce a také pojmenoval základní stavební kameny integrace v IT. Z mého pohledu bylo velice důležité pochopit základní principy a vlastnosti různých elementárních přístupů k integraci a nesoustředit se pouze na komplexní modely integrace celé podnikové architektury, typu SOA, EAI, apod. Tato východiska se stala základem kapitoly 4, ve které jsem provedl návrh samotné metodiky.

Po rešerši dostupných metodik a metodických vzorů jsem dospěl k názoru, že integrace služeb SaaS je oblast specifická a zatím nebyla zcela metodicky popsána. Existují některé prameny, které pojmenovávají některé „best practices“ v této oblasti, ale ucelenou metodiku nebylo možné pro cíle této práce použít. Z tohoto důvodu jsem navrhnul vlastní metodiku integrace služeb SaaS a založil ji na metodickém rámci MeFIS. Použitím metodického rámce MeFIS byla aplikována již prověřená logická struktura metodiky a zároveň je dále možné metodiku objektově orientovaným způsobem rozvíjet a rozšiřovat. Dle terminologie metodického rámce MeFIS jsem vlastně provedl *návrh metodického vzoru*, který může být dále aplikován a upravován dle potřeb konkrétních organizací či projektů.

Navržený metodický vzor jsem ověřil aplikací na konkrétním projektu integrace služby iTravel do informačního systému malé české cestovní kanceláře. Projekt prokázal funkčnost navrženého metodického vzoru, avšak také poukázal na nutnost vidět projekty v širších souvislostech. Tyto „širší souvislosti“ jsou do metodického vzoru začleněny prostřednictvím tzv. *principů*, jejichž dodržováním zajistíme (alespoň částečný) soulad s celkovou architekturou podniku.

Realizace konkrétního projektu také ilustrovala možné komplikace a problémy, které musíme řešit při použití tzv. *integračních agentů*. Tyto otázky vybízí k úvaze, zda by nebylo účelnější při integraci použit již hotové ucelené integrační platformy, případně sáhnout po integračních službách (iPaaS), než se snažit integrovat „na zelené louce“ i pro projekty menšího rozsahu. Zodpovězení této otázky by jistě stálo za hlubší analýzu v možném pokračování této práce.

Dalším možným námětem pro pokračování by byla realizace výzkumu mezi českými společnostmi, které již mají zkušenosti s vlastní integrací služeb SaaS. V kapitole 4.3.3.4 jsem se zabýval volbou metody integrace a představil jsem teoretickou míru složitosti integrace (viz Tabulka 16) v rámci různých kombinací metod. Tuto míru složitosti jsem odhadl teoreticky bez nějaké ucelenější datové podpory. Z toho důvodu se domnívám, že empirický výzkum, jehož cílem by bylo změření reálných měr složitosti integrace na základě zkušeností českých společností, by byl celkově přínosem. Míra uvedené složitosti totiž může výrazně ovlivnit výslednou volbu metody integrace.

7 Terminologický slovník

Termín	Zkratka	Význam (zdroj)
Cloudové služby		Distribuovaný výpočetní model poskytující se formou služeb (angl. service-oriented), který nabízí vysoce škálovatelné IT prostředky po síti. (Mishra, 2013; volně přeloženo autorem)
Informační systém podniku	IS	Systém, který používají lidé (zaměstnanci, správci), aby se dostali k datům prostřednictvím ICT prostředků. (Autor)
Integer		Číselný datový typ používaný v programovacích jazycích pro uchování celých čísel. Může být reprezentován různým počtem bitů (16, 32, 64 atd.) (Autor)
Integrace v IT		V podnikové informatice se vyskytuje celá řada technologií, aplikačních systémů, různorodých dat, různorodých zájmů a cílů. Už od počátku nasazování počítačů se zabýváme otázkami, jak tuto různorodou množinu zdrojů vhodně propojit tak, aby podporovala cíle organizace a aby nově vzniklý celek přinesl organizaci novou přidanou hodnotu, reprezentovanou např. sníženými provozními náklady nebo vytvořením nových příležitostí pro organizaci. (Gála, 2009)
Integration platform as a service	iPaaS	Sada cloudových služeb, které slouží k integraci a správě různých aplikací takovým způsobem, aby společně spolupracovaly. Tato řešení zpravidla nabízí podporu v oblasti: přenosu zpráv, transformací, směřování, překladu protokolů apod. (Pezzini, 2011; volně přeloženo autorem)

Termín	Zkratka	Význam (zdroj)
Message-Oriented Middleware	MOM	Middleware komponenta, která se stará o kompletní správu doručovacího cyklu zpráv mezi aplikacemi a zpravidla umožnuje implementaci asynchronní komunikace. (Hohpe, 2004; volně přeloženo autorem)
Metoda		Metoda určuje, co je třeba udělat v určité fázi nebo činnosti vývoje systému. Zpravidla se týká jen některého úhlu pohledu na systém. (Buchalcevová, 2005)
Metodika		Metodika představuje v obecném smyslu souhrn metod a postupů pro realizaci určitého úkolu. (Buchalcevová, 2005)
Middleware		Middleware je programové vybavení, které je umístěno mezi operačním systémem a aplikacemi a které nabízí aplikacím všestranně použitelnou množinu služeb. Služby vytvářejí provozní systémové prostředí, které umožnuje procesům, programům a aplikacím vzájemnou interakci v distribuovaném systému prostřednictvím sítě. (Gála, 2009)
Service Level Agreement	SLA	Smlouva o poskytování IT služeb. (Autor)
Služba		Služba je smysluplná aktivita, kterou vykonává počítačový program na základě požadavku jiného počítačového programu. (Krafzig, 2004; volně přeloženo autorem)
Software as a Service	SaaS	Služby spojené s poskytováním podnikových aplikací formou cloudu. (Mell, 2011; volně přeloženo autorem)

Termín	Zkratka	Význam (zdroj)
Zpráva		Určitý druh datové struktury (řetězec, číslo, objekt atd.). Může být interpretována jako data, nebo jako popis metody, která má být spuštěna u příjemce, nebo jako informace o události, která nastala na straně odesilatele. (Hohpe, 2004; volně přeloženo autorem)

8 Použité zdroje

BUCHALCEVOVÁ Alena. Metodický rámec budování IS/ICT. *Systémová integrace*, 3/2004, s. 81-88. Praha: ČSSI, 2004. ISSN 1210-9479

BUCHALCEVOVÁ Alena. Metodický vzor pro servisně orientovaný vývoj. In: *Objekty 2006*, Praha. Praha: PEF ČZU, 2006, strana 153-161. ISBN 80-213-1568-7.

BUCHALCEVOVÁ Alena. *Metodiky vývoje a údržby informačních systémů*. Praha: Grada, 2005. ISBN: 8024710757

COSTA Mateus et al. Software Frameworks for Information Systems Integration based on Web Services. Proceedings of the 2008 ACM symposium on Applied computing. In: *SAC 2008*. New York: ACM, 2008. s. 777-782. ISBN: 978-1-59593-753-7

ERL Thomas. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall, 2004. ISBN: 978-0131428980

ERL Thomas. *Service-oriented architecture: concepts, technology and design*. Prentice Hall, 2005. ISBN: 978-0133858587

EUROSTAT. *User Guide to the SME Definition [ONLINE]*. Eurostat, 2015, cit. Červen 2015. Dostupné z: http://ec.europa.eu/growth/smes/business-friendly-environment/sme-definition/index_en.htm

FALLOTI Piero. Software as a Service – case study CRM. In: *Systems Integration 2008 – Proceedings.* Prague: University of Economics, 2007. ISBN: 978-80-245-1373-7

FEHLING Christoph et al. *A Collection of Patterns for Cloud Types, Cloud Service Models, and Cloud-based Application Architectures [ONLINE]*. Stuttgart: University of Stuttgart, 2011. Dostupné z: http://www.iaas.uni-stuttgart.de/RUS-data/TR-2011-05%20-Patterns_for_Cloud_Computing.pdf

FEUERLICHT, Jiří. *Enterprise computing: concepts, standards and architectures*. Praha: Oeconomia, 2008. ISBN: 987-80-245-1367-6.

GÁLA Libor et al. *Podniková informatika (2. Vydání).* Praha: Grada, 2009. ISBN: 978-80-247-2615-1

GATES Bill. *Byznys rychlostí myšlenky*. Praha: Management Press, 1999. ISBN: 80-85943-97-2

HAI Henry et al. SaaS and Integration Best Practices. *FUJITSU Sci Tech. J.*, Vol 45, No. 3, July 2009. Dostupné z: http://www.fujitsu.com/downloads/MAG/vol45-3/paper03.pdf

HAUPTVOGEL Roman. *Metodika integračních projektů a jejich charakteristika*. Praha, 2013. Doktorská disertační práce. Vysoká škola ekonomická v Práce, Fakulta informatiky a statistiky. Školitel: doc. Ing. Alena Buchalcevová, Ph. D. Dostupné z: https://isis.vse.cz/auth/zp/index.pl?podrobnosti_zp=14925

HENTRICH Carsten et al. Patterns for business object model integration in process-driven and service-oriented architectures. In: *PLoP '06 (Proceedings of the 2006 conference on Pattern languages of programs)*. Article No. 23, 2006. ISBN: 978-1-60558-372-3. Dostupné z: http://dl.acm.org/citation.cfm?id=1415499

HK ČR. *Počty malých a středních firem rostou hlavně v tržních službách [ONLINE]*. Praha: HK ČR, 2005, cit. Červen 2015. Dostupné z: http://www.komora.cz/hk-cr/hlavni-zpravy/art_23005/pocty-malych-a-strednich-firem-rostou-hlavne-v-trznich-sluzbach.aspx

HOHPE Gregor. *Enterprise integration patterns: designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004. ISBN: 978-0321200686

CHRISTIANSSON Benneth, Lean Integration Using Open Source Based Infrastructure and an Integration Competency Center Approach. In: *Communications of the IBIMA*, Volume 2012 (2012), Article ID 532248. IBIMA Publishing, 2012.

IDC, *Press Release: Cloud Investments Drove Growth in the IT Infrastructure Market in EMEA in 2014 [ONLINE]*, Framingham: IDC. 27 Apr 2015, cit. Červen 2015. Dostupné z: http://www.idc.com/getdoc.jsp?containerId=prUK25585915

IZZA S. Integration of industrial information systems: from syntactic to semantic integration approaches. *Enterprise Information Systems*, Volume: 3, Issue: 1. TAYLOR & FRANCIS, 2009. Dostpuné z: http://www.tandfonline.com/doi/abs/10.1080/17517570802521163

JOHA Anton et al. Design Choices Underlying the Software as a Service (SaaS) Business Model from the User Perspective: Exploring the Fourth Wave of Outsourcing. *Journal of Universal Computer Science*, vol. 18, no. 11, 2012, cit. Červen 2015. Dostupné z: http://www.jucs.org/jucs_18_11/design_choices_underlying_the

KAO Liz. *Salesforce Integration for Dummies (Informatica Special Edition)*. John Wiley & Sons, 2015. ISBN: 978-1-119-03763-7.

KARAS Jan. *Metodika integračních projektů a jejich charakteristika*. Praha, 2006. Doktorská disertační práce. Vysoká škola ekonomická v Práce, Fakulta informatiky a statistiky. Školitel: Prof. Ing. Jaroslav Jandoš, CSc. Dostupná z: https://isis.vse.cz/auth/zp/index.pl?podrobnosti_zp=14542

KOTEN Jiří. *Problémy on-line integrace systémů v bankovním prostředí při aplikaci SOA principů*. Praha, 2006. Diplomová práce. Vysoká škola ekonomická v Práce, Fakulta informatiky a statistiky. Vedoucí práce: Doc. Ing. Jan Pour, CSc. Dostupná z: https://isis.vse.cz/auth/zp/portal_zp.pl?podrobnosti_zp=12676

KOTLER Phillip. *Marketing management: analysis, planning, implementation, and control*. Prentice-Hall, 1988. ISBN: 9780135562673

KRAFZIG Dirk et al. *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall, 2004. ISBN: 978-0131465756

LHEUREUX B. *Who's Who in Middleware, 1Q04 [ONLINE]*. Gartner, 2004. Dostupné z: http://www-01.ibm.com/software/info/websphere/partners4/articles/gartner/garwho.html

LINTHICUM David S. *Enterprise Application Integration*. Addison-Wesley Professional, 2000. ISBN: 9780201615838

LIU Fang et al. *The NIST Cloud Computing Reference Architecture [ONLINE]*. National Insitute of Standards and Technology (NIST), 2011. NIST Special Publication 500-292. Dostupné z: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

LOVERING Joanne. *Canadian executives have their heads in the cloud [ONLINE]*. BramptonGuardian.com: November, 2015. Dostupné z: http://www.bramptonguardian.com/news-story/6131134-canadian-executives-have-their-heads-in-the-cloud/

LYTRA Ioanna et al. A Pattern Language for Service-based Platform Integration and Adaptation. In: *EuroPLoP '12 Proceedings of the 17th European Conference on Pattern Languages of Programs*, Article No. 4. New York: ACM, 2012. ISBN: 978-1-4503-2943-9. Dostupné z: http://dl.acm.org/citation.cfm?id=2603080

MACKENZIE Matthew et al. *Reference Model for Service Oriented Architecture [ONLINE]*. OASIS, 2006. Dostupné z: http://docs.oasis-open.org/soa-rm/v1.0/

MAINKA Christian et al. Your Software at my Service - Security Analysis of SaaS Single Sign-On Solutions in the Cloud. In: *CCSW '14 Proceedings of the 6th edition of the ACM Workshop on*

Cloud Computing Security, Pages 93-104. New York: ACM, 2014. ISBN: 978-1-4503-3239-2. Dostupné z: http://dl.acm.org/citation.cfm?id=2664172

MELL Peter et al. *The NIST Definition of Cloud Computing [ONLINE]*. National Insitute of Standards and Technology (NIST), 2011. NIST Special Publication 800-145. Dostupné z: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505

MISHRA Nandini et al. Technologies of Cloud Computing - Architecture Concepts based on Security and its Challenges. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, Volume 2, Issue 3, March 2013, ISSN: 2278 – 1323

MULESOFT. *White Paper: Secrets of a Great API*. MuleSoft, 2014.

NARASIMHA Bolloju, Dr. at al. Cloud-based B2B Systems Integration for Small-and-Medium-sized Enterprises. In: *ICACCI (International Conference on Advances in Computing, Communications and Informatics)*, 2012. Dostupné z: http://researchdirect.uws.edu.au/islandora/object/uws:14412

NOVÁKOVÁ Julie. *Jak udělat kvalitní rešerši? [ONLINE]*. Studentskefinance.cz, 2012. Dostupné z: http://student.finance.cz/zpravy/finance/349272-jak-udelat-kvalitni-resersi-/

PALUCH Juraj, *Integration as a Service*. Praha, 2010. Bakalářská práce. Vysoká škola ekonomická v Práce, Fakulta informatiky a statistiky. Vedoucí práce: Ing. Lukáš Burkoň. Dostupné z: http://isis.vse.cz/zp/index.pl?podrobnosti_zp=24889

PETERKA Jiří. *Model klient/server [ONLINE]*. eArchiv. cit. Červen 2015. Dostupné z: http://www.earchiv.cz/a96/a609k150.php3

PEZZINI Massimo et al. *Integration Platform as a Service: Moving Integration to the Cloud [ONLINE]*. Gartner, 2011. Dostupné z: https://www-304.ibm.com/industries/publicsector/fileserve?contentid=250736

PEZZINI Massimo et al. *Magic Quadrant for Enterprise Integration Platform as a Service, Worldwide [ONLINE]*. Gartner, 2015. Dostupné z: https://www.informatica.com/ipaas-magic-quadrant.html#fbid=gNZaWLofgcj

RANDOVÁ Libuše, *Integrace aplikací SaaS do podnikového informačního systému*. Praha, 2010. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky. Vedoucí práce: Ing. Libor Gála. Dostupná z: http://isis.vse.cz/zp/index.pl?podrobnosti_zp=23808

ROSSI, Ben. SaaS deployments are now mission critical – Gartner: Survey reveals enterprise cloud adoption plans through 2017 [ONLINE]. November 2014, *Information Age*, cit. Červen 2015. Dostupné na: http://www.information-age.com/technology/cloud-and-virtualisation/123458675/saas-deployments-are-now-mission-critical-gartner

SEDLÁČEK Ondřej. *Integrace jako služba*. Praha, 2013. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky. Vedoucí práce: Ing. Libor Gála. Dostupná z: http://isis.vse.cz/zp/index.pl?podrobnosti_zp=40016

SCHMIDT John. *Lean Integration – Translating an Inovative Agile Approach into Business Value [ONLINE]*. Informatica, 2010. Dostupné z: https://www.eiseverywhere.com/file_uploads/020ea1dd8dff63398ae412b4b48c43ee_PDF_John_Schmidt_-.pdf

SINGLETARY, L.A. *Empirical Study Of Attributes And Perceived Benefits Of Applications Integration For Enterprise Systems*. Louisiana State University, 2003. Dostupné z: http://etd.lsu.edu/docs/available/etd-0618103-160439/unrestricted/Singletary_dis.pdf

STAFFORD Jan. SaaS applications integration: Strategies and best practices for success - Why SaaS application integration requires new strategies, tools. *TechTarget (SearchCloudApplications.com) [ONLINE]*, 2015. Dostupné z: http://searchcloudapplications.techtarget.com/feature/Why-SaaS-application-integration-requires-new-strategies-tools

ŠTUMPF et al.Datová integrace prostřednictvím společného datového modelu. In: *Systems Integration 2008 – Proceedings.* Prague: University of Economics, 2007. ISBN: 978-80-245-1373-7

TAYLOR Paul. The importance of information and communication technologies (ICTs): An integration of the extant literature on ict adoption in small and medium enterprises. *International Journal of Economics, Commerce and Management (IJECM)*, Vol. III, Issue 5, May 2015. Dostupné z: http://ijecm.co.uk/wp-content/uploads/2015/05/3517.pdf

VOŘÍŠEK Jiří et al. *Principy a modely řízení podnikové informatiky*. Praha: Oeconomia, 2011. ISBN: 978-80-245-1440-6

WONIL Kim et al., Computers & Mathematics with Applications - An innovative method for data and software integration in SaaS. *Advanced Technologies in Computer, Consumer and Control*. Volume 64, Issue 5, September 2012. Dostupné z: http://www.sciencedirect.com/science/article/pii/S089812211200274X

YU Dongjin et al., Service Oriented Enterprise Application Integration and its Implementation Based on Open Source Software. In: *Proceedings of the Second Symposium International Computer Science and Computational Technology (ISCSCT '09)*. Academy Publisher, 2009.

ŽIŽKA Petr. *Webová služba pro syndikaci webových zdrojů*. Praha, 2009. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta managementu. Vedoucí práce: Ing. Jelínek Jiří, CSc. Dostupné z: https://isis.vse.cz/zp/portal_zp.pl?podrobnosti=67081

9 Seznam obrázků a tabulek

9.1 Seznam obrázků

Obrázek 1: Digitální nervový systém (Gates, 1999; upraveno autorem) 5

Obrázek 2: Model integrace IS/ICT podniku a vymezení oblasti zájmu této práce (Voříšek, 2011; upraveno autorem) 10

Obrázek 3: Pojem „služba“ (Krafzig, 2004; upraveno autorem) 12

Obrázek 4: Služba a její základní komponenty (Krafzig 2004; upraveno autorem) 13

Obrázek 5: Detailní pohled na zapouzdřenost služeb 13

Obrázek 6: Schéma vrstvené IT architektury (Feuerlicht, 2008; upraveno autorem) 19

Obrázek 7: Úloha middleware (Krafzig, 2004) 23

Obrázek 8: Rozdělení middleware (Lheureux, 2004; upraveno autorem) 24

Obrázek 9: Základní middleware a jeho použití (Lheureux, 2004; upraveno autorem) 25

Obrázek 10: Přenos zprávy (Hohpe, 2004) 29

Obrázek 11: Synchronní a asynchronní komunikace (Hohpe, 2004) 30

Obrázek 12: Princip přenosu souborů (Hohpe, 2004) 33

Obrázek 13: Princip sdílené databáze (Hohpe, 2004) 34

Obrázek 14: Princip vzdálených procedur/metod (Hohpe, 2008) 34

Obrázek 15: Využití zprostředkující služby při komunikaci (Erl, 2005; upraveno autorem) 36

Obrázek 16: Princip sběrnice zpráv (Hohpe, 2004) 36

Obrázek 17: Izolované aplikační ostrůvky (Feuerlicht, 2008), (Linthicum, 2000; upraveno autorem), částečně (Erl, 2005) 39

Obrázek 18: Metodický rámec MeFIS (volně zpracováno dle (Buchalcevová, 2005)) 49

Obrázek 19: Prvky rámce MeFIS (volně zpracováno dle (Buchalcevová, 2005); upraveno autorem) . 51

Obrázek 20: Prvky navrženého metodického vzoru 54

Obrázek 21: Schématické znázorněné fází metodického vzoru (volně zpracováno dle (HAI, 2009); upraveno autorem) 59

Obrázek 22: Procesy a přiřazené role 63

Obrázek 23: Průběh realizace projektu v různých fázích 64

Obrázek 24: Strategická závislost na poskytovateli SaaS (volně zpracováno dle (Joha, 2012); upraveno autorem) 66

Obrázek 25: Dimenze pro výběr nejvhodnějšího poskytovatele SaaS (volně zpracováno dle (Joha, 2012); upraveno autorem) 69

Obrázek 26: Základní integrační scénáře (volně zpracováno dle (Hai, 2009); upraveno autorem) 74

Obrázek 27: Základní princip fungování vybrané společnosti 86

Obrázek 28: Možnosti komunikace s informačním systémem iTravel 88

Obrázek 29: Možný integrační scénář bez integračního agenta 91

Obrázek 30: Možný integrační scénář s integračním agentem 92

Obrázek 31: Výsledný integrační scénář 93

Obrázek 32: Ukázka výstupu funkčního prototypu 96

Obrázek 33: Postup integračním agentem 98

Obrázek 34: Ukázka pomocné aplikace pro generování přístupových údajů 100

Obrázek 35: Výsledné řešení integrace 101

9.2 Seznam tabulek

Tabulka 1: EMEA Cloud IT Infrastructure Value, Calendar Year 2014 (Value in $ billions) (IDC, 2015) 1

Tabulka 2: Definice velikosti společnosti (Eurostat, 2015) 7

Tabulka 3: Výhody a nevýhody synchronní a asynchronní komunikace (částečně zpracováno dle (Hohpe, 2004); upraveno autorem) 30

Tabulka 4: Metody integrace a odpovídající způsoby komunikace 32

Tabulka 5: Základní popis navrženého metodického vzoru 52

Tabulka 6: Fáze metodického vzoru 55

Tabulka 7: Popis fáze Informační strategie 56

Tabulka 8: Popis fáze Globální analýza a návrh 57

Tabulka 9: Popis fáze Detailní analýza a návrh 58

Tabulka 10: Popis fáze Implementace 59

Tabulka 11: Popis fáze Zavedení 60

Tabulka 12: Popis fáze Provoz a údržba 61

Tabulka 13: Popis procesu Aktualizace podnikové politiky správy a řízení IT 67

Tabulka 14: Popis procesu Specifikace požadavků a rozsahu integračního projektu 68

Tabulka 15: Popis procesu Výběr služby SaaS a vlastnosti služby SaaS..... 70

Tabulka 16: Míra složitosti integrace v různých kombinacích požadovaných metod a nabízených..... 71

Tabulka 17: Popis procesu: Volba metody integrace 72

Tabulka 18: Shrnutí integračních scénářů 75

Tabulka 19: Popis procesu: Volba integračního scénáře..... 75

Tabulka 20: Popis procesu: Analýza možných omezení využití služby (SaaS)..... 77

Tabulka 21: Popis procesu: Detailní návrh jednotlivých komponent integračního řešení 77

Tabulka 22: Popis procesu: Získání funkčního prototypu..... 78

Tabulka 23: Popis procesu: Implementace řešení 79

Tabulka 24: Popis procesu: Testování řešení 80

Tabulka 25: Popis procesu: Nasazení řešení 80

Tabulka 26: Popis procesu: Administrace řešení 81

Tabulka 27: Návrhové vzory..... 83

10 Přílohy

10.1 Ukázka technické dokumentace ke službě iTravel

iTravel API

V3.0

LEMAX

1. CONTENTS

2. Functions 23
GetCountries 23
Parameters 23
Response 23
Example 23
GetRegions 23
Parameters 23
Response 24
Example 24
GetDestinations 24
Parameters 25
Response 25
Example 25
GetAllDestinations 26
PARAMETERS 26
RESPONSE 26
EXAMPLE 26
GetReservation 27
Parameters 27
Response 27
GetAllReservations 27
Parameters 28
Response 28
Example 28
GetAttributeGroups 28
parameters 28

2. FUNCTIONS

GetCountries

Returns all countries for given parameters.

PARAMETERS

Name	Type	Default value	Description
GetCountriesParameters	GetCountriesParameters		Object that contains parameters needed to get a list of countries

RESPONSE

List of objects of type Country.

EXAMPLE

In this example request is made to get all countries which contains object of type "Hotel" (objectTypeID = 1). Object type group and categoryID are irrelevant in this request and are set to null (objectTypeGroupID = null, categoryID = null). Results should be on English (languageID = "en").

REQUEST

```
iTravelAPI 3 0 api = iTravelWebsite.GetITravelApi();
var getCountriesParameter = new GetCountriesParameters();
getCountriesParameter.ObjectTypeID = 1;
getCountriesParameter.LanguageID = "en";

api.GetCountries(getCountriesParameter);
```

RESPONSE

```
<?xml version="1.0"?>
<ArrayOfCountry xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Country>
    <CountryID>27</CountryID>
    <CountryName>Austria</CountryName>
  </Country>
  <Country>
    <CountryID>110</CountryID>
    <CountryName>France</CountryName>
  </Country>
</ArrayOfCountry>
```

GetRegions

Returns all regions for given parameters

PARAMETERS

Name	Type	Default value	Description
GetRegionsParameters	GetRegionsParameters		Object that contains parameters used

for getting a list of regions

RESPONSE

List of object of type Region.

EXAMPLE

In this example request is made to get all regions in country "Croatia" (countryID = 2). Object type, object type group and category are irrelevant in this request and are set to null. Results should be in English (languageID = "en").

REQUEST

```
iTravelAPI 3 0 api = iTravelWebsite.GetITravelApi();

var getRegionsParameter = new GetRegionsParameters();
getRegionsParameter.CountryID = 2;
getRegionsParameter.LanguageID = "en";

api.GetRegions(getRegionsParameter);
```

RESPONSE

```
<?xml version="1.0"?>
<ArrayOfRegion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Region>
    <CountryID>2</CountryID>
    <RegionID>3</RegionID>
    <RegionName>Istria</ null RegionName>
  </Region>
  <Region>
    <CountryID>2</CountryID>
    <RegionID>4</RegionID>
    <RegionName>Kvarner</RegionName>
  </Region>
  <Region>
    <CountryID>2</CountryID>
    <RegionID>6</RegionID>
    <RegionName>Middle Dalmatia</RegionName>
  </Region>
  <Region>
    <CountryID>2</CountryID>
    <RegionID>5</RegionID>
    <RegionName>North Dalmatia</RegionName>
  </Region>
  <Region>
    <CountryID>2</CountryID>
    <RegionID>7</RegionID>
    <RegionName>South Dalmatia</RegionName>
  </Region>
</ArrayOfRegion>
```

GetDestinations

Returns all destinations for given parameters

PARAMETERS

Name	Type	Default value	Description
GetDestinationsParameters	GetDestinationsParameters		Object that contains parameters needed to get a list of destinations

RESPONSE

List of object of type Destination.

EXAMPLE

In this example request is made to get all destination in country "Croatia" (countryID = 2), in region "Kvarer" (regionID = 4), object type, object type group and category are irrelevant in this request and are set to null. Results should be on English (languageID = "en").

REQUEST

```
iTravelAPI 3 0 api = iTravelWebsite.GetITravelApi();

var getDestinationsParameter = new GetDestinationsParameters();
getDestinationsParameter.CountryID = 2;
getDestinationsParameter.RegionID = 4;
getDestinationsParameter.LanguageID = "en";

api.GetDestinations(getDestinationsParameter);
```

RESPONSE

```
<?xml version="1.0"?>
<ArrayOfDestination xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Destination>
    <RegionID>4</RegionID>
    <DestinationID>61</DestinationID>
    <DestinationName>Baška</DestinationName>
  </Destination>
  <Destination>
    <RegionID>4</RegionID>
    <DestinationID>62</DestinationID>
    <DestinationName>Cres</DestinationName>
  </Destination>
  <Destination>
    <RegionID>4</RegionID>
    <DestinationID>34</DestinationID>
    <DestinationName>Crikvenica</DestinationName>
  </Destination>
  ...
</ArrayOfDestination>
```

10.2 Ukázka vyexportovaných ceníků systémem iTravel

BRISTOL (4*) - Karlovy Vary

Price list (CZK)

per person/night	Passenger	Minimal nights	01.04. - 28.04.	11.05. - 01.11.	22.12. - 01.01.	01.01. - 11.01.	28.04. - 11.05.	01.11. - 22.12.	11.01. - 01.04.			
1. Single room, Fullboard and Treatment	1	7	4.243,20	4.243,20	4.243,20	4.243,20	4.773,60	3.182,40	3.182,40			
2. Double room, Fullboard and Treatment	1	7	5.383,56	5.383,56	5.383,56	5.383,56	6.311,76	4.084,08	4.084,08			
2. Double room, Fullboard and Treatment	2	7	3.845,40	3.845,40	3.845,40	3.845,40	4.508,40	2.917,20	2.917,20			
3. Suite, Fullboard and Treatment	1	7	4.773,60	4.773,60	4.773,60	4.773,60	5.569,20	3.580,20	3.580,20			
3. Suite, Fullboard and Treatment	2	7	4.773,60	4.773,60	4.773,60	4.773,60	5.569,20	3.580,20	3.580,20			
4. Single room, Bed and breakfast	1	1	3.580,20	3.580,20	3.580,20	3.580,20	4.110,60	2.519,40	2.519,40			
5. Double room, Bed and breakfast	1	1	4.455,36	4.455,36	4.455,36	4.455,36	5.383,56	3.155,88	3.155,88			
5. Double room, Bed and breakfast	2	1	3.182,40	3.182,40	3.182,40	3.182,40	3.845,40	2.254,20	2.254,20			
6. Suite, Bed and breakfast	1	1	5.754,84	5.754,84	5.754,84	5.754,84	6.868,68	4.084,08	4.084,08			
6. Suite, Bed and breakfast	2	1	4.110,60	4.110,60	4.110,60	4.110,60	4.906,20	2.917,20	2.917,20			

per person/night	Passenger	Minimal nights	01.04. - 28.04.	11.05. - 01.11.	22.12. - 01.01.	01.01. - 11.01.	28.04. - 11.05.	01.11. - 22.12.	11.01. - 01.04.			
Discount for child (treatment not possible) (2 - 15) (is valid for: 1, 2, 3)	1	1	-265.2	-265.2	-265.2	-265.2	-265.2	-265.2	-265.2			
Discount for accompanying person (no treatment) (15+) (is valid for: 1, 2, 3)	1	1	-397.8	-397.8	-397.8	-397.8	-397.8	-397.8	-397.8			
Supplement for full board (2+) (is valid for: 4, 5, 6)	1	1	265.2	265.2	265.2	265.2	265.2	265.2	265.2			
Supplement for half board (2+) (is valid for: 4, 5, 6)	1	1	132.6	132.6	132.6	132.6	132.6	132.6	132.6			
Discount for BBT from HBT (15+) (is valid for: 1, 2, 3)	1	1	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6			
Discount for child (2 - 15)	1	1	-30.00%	-30.00%	-30.00%	-30.00%	-30.00%	-30.00%	-30.00%			
Discount for adult on extra bed (15+, 15+, 15+)	1	1	-10.00%	-10.00%	-10.00%	-10.00%	-10.00%	-10.00%	-10.00%			
Discount for HBT from FBT (15+) (is valid for: 1, 2, 3)	1	1	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6			
Discount for infant - room only (0 - 2, 12+)	1	1	-100.00%	-100.00%	-100.00%	-100.00%	-100.00%	-100.00%	-100.00%			

per item per day	Passenger	Minimal nights	01.04. - 28.04.	11.05. - 01.11.	22.12. - 01.01.	01.01. - 11.01.	28.04. - 11.05.	01.11. - 22.12.	11.01. - 01.04.			
Pets	1	1	397.8	397.8	397.8	397.8	397.8	397.8	397.8			
Baby cot	1	1	265.2	265.2	265.2	265.2	265.2	265.2	265.2			

BRISTOL PALACE (4*) - Karlovy Vary

Price list (CZK)

per person/night	Passenger	Minimal nights	01.04. - 28.04.	11.05. - 01.11.	22.12. - 01.01.	01.01. - 11.01.	28.04. - 11.05.	01.11. - 22.12.	11.01. - 01.04.			
1. Single room, Fullboard and Treatment	1	7	4.508,40	4.508,40	4.508,40	4.508,40	5.038,80	3.447,60	3.447,60			
2. Double room, Fullboard and Treatment	1	7	6.126,12	6.126,12	6.126,12	6.126,12	7.054,32	4.641,00	4.641,00			
2. Double room, Fullboard and Treatment	2	7	4.375,80	4.375,80	4.375,80	4.375,80	5.038,80	3.315,00	3.315,00			
3. Suite, Fullboard and Treatment	1	7	9.282,00	9.282,00	9.282,00	9.282,00	10.395,84	6.311,76	6.311,76			
3. Suite, Fullboard and Treatment	2	7	6.630,00	6.630,00	6.630,00	6.630,00	7.425,60	4.508,40	4.508,40			
4. Suite, PRESIDENT, Fullboard and Treatment	1	7	12.809,16	12.809,16	12.809,16	12.809,16	14.479,92	8.353,80	8.353,80			
4. Suite, PRESIDENT, Fullboard and Treatment	2	7	9.149,40	9.149,40	9.149,40	9.149,40	10.342,80	5.967,00	5.967,00			
5. Single room, Bed and breakfast	1	1	3.845,40	3.845,40	3.845,40	3.845,40	4.375,80	2.784,60	2.784,60			
6. Double room, Bed and breakfast	1	1	5.197,92	5.197,92	5.197,92	5.197,92	6.126,12	3.712,80	3.712,80			
6. Double room, Bed and breakfast	2	1	3.712,80	3.712,80	3.712,80	3.712,80	4.375,80	2.652,00	2.652,00			
7. Suite, Bed and breakfast	1	1	8.353,80	8.353,80	8.353,80	8.353,80	9.467,64	5.383,56	5.383,56			
7. Suite, Bed and breakfast	2	1	5.967,00	5.967,00	5.967,00	5.967,00	6.762,60	3.845,40	3.845,40			
8. Suite, President, Bed and breakfast	1	1	11.880,96	11.880,96	11.880,96	11.880,96	13.551,72	7.425,60	7.425,60			
8. Suite, President, Bed and breakfast	2	1	8.486,40	8.486,40	8.486,40	8.486,40	9.679,80	5.304,00	5.304,00			

per person/night	Passenger	Minimal nights	01.04. - 28.04.	11.05. - 01.11.	22.12. - 01.01.	01.01. - 11.01.	28.04. - 11.05.	01.11. - 22.12.	11.01. - 01.04.			
Discount for child (treatment not possible) (2 - 15) (is valid for: 1, 2, 3, 4)	1	1	-265.2	-265.2	-265.2	-265.2	-265.2	-265.2	-265.2			
Discount for accompanying person (no treatment) (15+) (is valid for: 1, 2, 3, 4)	1	1	-397.8	-397.8	-397.8	-397.8	-397.8	-397.8	-397.8			
Supplement for full board (2+) (is valid for: 5, 6, 7, 8)	1	1	265.2	265.2	265.2	265.2	265.2	265.2	265.2			
Supplement for half board (2+) (is valid for: 5, 6, 7, 8)	1	1	132.6	132.6	132.6	132.6	132.6	132.6	132.6			
Discount for BBT from HBT (15+) (is valid for: 1, 2, 3, 4)	1	1	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6			
Discount for child (2 - 15)	1	1	-30.00%	-30.00%	-30.00%	-30.00%	-30.00%	-30.00%	-30.00%			
Discount for adult on extra bed (15+, 15+, 15+)	1	1	-10.00%	-10.00%	-10.00%	-10.00%	-10.00%	-10.00%	-10.00%			
Discount for HBT from FBT (15+) (is valid for: 1, 2, 3, 4)	1	1	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6	-132.6			
Discount for infant - room only (0 - 2, 12+)	1	1	-100.00%	-100.00%	-100.00%	-100.00%	-100.00%	-100.00%	-100.00%			

per item per day	Passenger	Minimal nights	01.04. - 28.04.	11.05. - 01.11.	22.12. - 01.01.	01.01. - 11.01.	28.04. - 11.05.	01.11. - 22.12.	11.01. - 01.04.			
Pets	1	1	397.8	397.8	397.8	397.8	397.8	397.8	397.8			
Baby cot	1	1	265.2	265.2	265.2	265.2	265.2	265.2	265.2			

10.3 Ukázka základního prostředí navržené aplikace

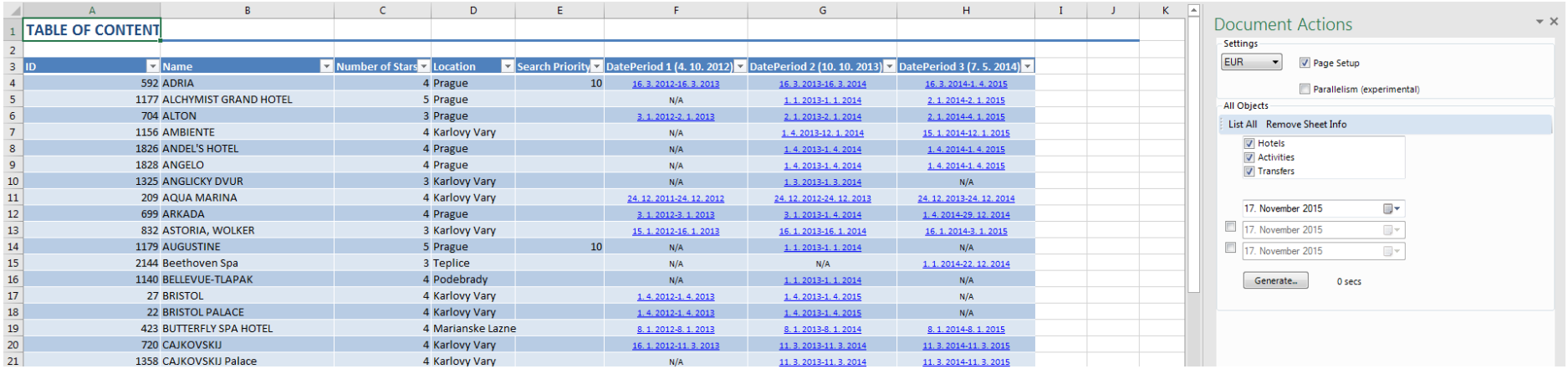


TABLE OF CONTENT

ID	Name	Number of Stars	Location	Search Priority	DatePeriod 1 (4. 10. 2012)	DatePeriod 2 (10. 10. 2013)	DatePeriod 3 (7. 5. 2014)
592	ADRIA	4	Prague	10	16. 3. 2012-16. 3. 2013	16. 3. 2013-16. 3. 2014	16. 3. 2014-1. 4. 2015
1177	ALCHYMIST GRAND HOTEL	5	Prague		N/A	1. 1. 2013-1. 1. 2014	2. 1. 2014-2. 1. 2015
704	ALTON	3	Prague		3. 1. 2012-2. 1. 2013	2. 1. 2013-2. 1. 2014	2. 1. 2014-4. 1. 2015
1156	AMBIENTE	4	Karlovy Vary		N/A	1. 4. 2013-12. 1. 2014	15. 1. 2014-12. 1. 2015
1826	ANDEL'S HOTEL	4	Prague		N/A	1. 4. 2013-1. 4. 2014	1. 4. 2014-1. 4. 2015
1828	ANGELO	4	Prague		N/A	1. 4. 2013-1. 4. 2014	1. 4. 2014-1. 4. 2015
1325	ANGLICKY DVUR	3	Karlovy Vary		N/A	1. 3. 2013-1. 3. 2014	N/A
209	AQUA MARINA	4	Karlovy Vary		24. 12. 2011-24. 12. 2012	24. 12. 2012-24. 12. 2013	24. 12. 2013-24. 12. 2014
699	ARKADA	4	Prague		3. 1. 2012-3. 1. 2013	3. 1. 2013-1. 4. 2014	1. 4. 2014-29. 12. 2014
832	ASTORIA, WOLKER	3	Karlovy Vary		15. 1. 2012-16. 1. 2013	16. 1. 2013-16. 1. 2014	16. 1. 2014-3. 1. 2015
1179	AUGUSTINE	5	Prague	10	N/A	1. 1. 2013-1. 1. 2014	N/A
2144	Beethoven Spa	3	Teplice		N/A	N/A	1. 1. 2014-22. 12. 2014
1140	BELLEVUE-TLAPAK	4	Podebrady		N/A	1. 1. 2013-1. 1. 2014	N/A
27	BRISTOL	4	Karlovy Vary		1. 4. 2012-1. 4. 2013	1. 4. 2013-1. 4. 2015	N/A
22	BRISTOL PALACE	4	Karlovy Vary		1. 4. 2012-1. 4. 2013	1. 4. 2013-1. 4. 2015	N/A
423	BUTTERFLY SPA HOTEL	4	Marianske Lazne		8. 1. 2012-8. 1. 2013	8. 1. 2013-8. 1. 2014	8. 1. 2014-8. 1. 2015
720	CAJKOVSKIJ	4	Karlovy Vary		16. 1. 2012-11. 3. 2013	11. 3. 2013-11. 3. 2014	11. 3. 2014-11. 3. 2015
1358	CAJKOVSKIJ Palace	4	Karlovy Vary		N/A	11. 3. 2013-11. 3. 2014	11. 3. 2014-11. 3. 2015

10.4 Ukázka ceníku připraveného navrženou aplikací

	A	B	C	D	E	F	G	H
1	**ADRIA (4*) - Prague**							
2	*Price List (EUR), 16. 3. 2012 - 16. 3. 2013*							
3	Back							
4	***Per night***	**PAX**	**Min. Nights**	**31.12–31.12**	**05.04–17.06 06.09–04.11 27.12–03.01**	**16.03–05.04 17.06–06.09 03.01–08.01**	**04.11–27.12 08.01–16.03**	
5	1. Single room, Bed and breakfast	1	1		177	123	88	
6	2. Double room, Bed and breakfast	1	1		184	123	88	
7	2. Double room, Bed and breakfast	2	1		191	137	102	
8	3. Suite (Junior), Bed and breakfast	2	1		220	167	127	
9	4. Suite, Bed and breakfast	2	1		247	193	140	
10								
11								
12	***Per person***	**PAX**	**Min. Nights**	**31.12–31.12**	**05.04–17.06 06.09–04.11 27.12–03.01**	**16.03–05.04 17.06–06.09 03.01–08.01**	**04.11–27.12 08.01–16.03**	
13	Additional charge for placement in Triton restaurant for NYE	1	1	40				
14	Obligatory extra charge for NYE	1	1	140				
15	Obligatory extra charge for NYE (child)	1	1	105				
16								
17	***Per person per night***	**PAX**	**Min. Nights**	**31.12–31.12**	**05.04–17.06 06.09–04.11 27.12–03.01**	**16.03–05.04 17.06–06.09 03.01–08.01**	**04.11–27.12 08.01–16.03**	
18	Additional - enlarged Halfboard (4 course menu)	1	1		22	22	22	
19	Additional halfboard	1	1		20	20	20	
20	Extra bed charge (2+)	1	1		15	15	15	
21	Infant (<2) - room only	1	1		-100%	-100%	-100%	
22								
23								