Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

CRM v prostředí SAP

Petr Toman

! ! !

Místo této strany vložíte zadání diplomové práce.

(Do jedné vazby originál a do druhé kopii)

! ! !

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "CRM v prostředí SAP" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2013 _____

Rád bych touto cestou poděkoval doc. Ing. Vojtěchu Merunkovi Ph.D. za jeho ochotu, vstřícný přístup, rady a inspiraci při vedení mé diplomové práce. Také bych rád poděkoval svému zaměstnavateli, firmě ERGO pojišťovna a jejím zaměstnancům, za velkorysost, která mi umožnila studovat při zaměstnání a poskytnutí přístupu k technologiím o kterých moje práce pojednává. Nemenší dík patří mé rodině za její trpělivost a podporu během celého mého studia.

CRM v prostředí SAP

CRM in SAP environment

Souhrn

Diplomová práce je zaměřená na problematiku řízení zákaznických vztahů v prostředí informačních systémů SAP v odvětví pojišťovnictví. Popisuje vzájemné souvislosti mezi CRM a vývojem informačních technologií v kontextu podnikových informačních systémů. Analytická část je členěna do kapitol, popisujících jednotlivé oblastí, ze kterých vychází teoretická fundace práce. Jsou to oblasti CRM, prostředí informačních systémů SAP, jazyk ABAP, technologie tvorby webových sídel a datové modelování. Projektová část práce demonstruje realizaci zákaznického portálu v praxi při užití metod softwarového inženýrství a grafické interpretace diagramy UML. Implementace využívá komunikačních prostředků SAP ve spojení s webovými technologiemi.

Summary

The diploma thesis is focused on the customer relationship management in SAP information systems environment in insurance branch. It describes mutual relationship between CRM and development of information technologies, in context of enterprise information systems. Analytical part is divided into chapters, describing particular topics, which the theoretical foundation is based on. These are CRM, SAP environment, ABAP language, technologies of web sites design and data modeling. Project part demonstrates realization of customer portal in praxis by using methods of the software engineering and graphical interpretation by UML diagrams. Implementation uses communication means of SAP in connection with web technologies.

Klíčová slova: CRM, SAP, ABAP, SAP Business Connector, UML, funkční prototypování, případ užití, pojišťovnictví, web portál.

Keywords: CRM, SAP, ABAP, SAP Business Connector, UML, Functional Prototyping, Use Case, insurance, web portal.

1. Úvod

Informační technologie pronikly do mnoha oblastí lidského života. S aplikacemi, založenými na více než 60 let staré myšlence strojového zpracovaní dat, se lze setkat jak v osobním životě, tak ve veřejné nebo podnikové sféře. Současně roste objem dat, která mají uživatelé k dispozici, a zvyšuje se výkonnost zařízení, která tato data zpracovávají. Ovšem s tím nabývá na intenzitě problém jak data správně interpretovat a vytvářet z nich relevantní informaci, dostupnou ve správném čase, místě a v přijatelné formě.

V podnicích si své pevné místo našly systémy ERP (Enterprise Resource Planning) pro správu a řízení podnikových zdrojů, které sjednotily původně roztříštěné agendy jednotlivých podnikových procesů. Data jsou v těchto systémech spravována konzistentně a v jednotném datovém úložišti. Díky propracované obchodní logice, postavené nad těmito daty, je možné podnikové procesy řídit nejen efektivněji, ale i zpracovávat procesy, které by bez použití ERP systémů nebyly možné. Pro ERP systémy bývá typická složitost softwarového řešení. Ta vyplývá z vysoké úrovně abstrakce při převodu podnikových procesů do softwaru tak, aby byla zachována podmínka znuvupoužitelnosti kódů aplikace v různých podnicích. Složitost těchto řešení bývá často přijímána uživateli s jistou nedůvěrou, autor této práce se ve své dosavadní praxi často setkal s poměrně vysokou mírou kritického odstupu budoucích uživatelů při zavádění nových řešení v podnicích.

Celosvětově narůstá penetrace internetových technologií do firem i domácností. V červenci 2012 mělo přístup k internetu 34,3% procenta světové populace, v Evropě pak 63,2%, v České republice 73% (De Argaez, 2013). Díky této skutečnosti se otevřely komunikační kanály mezi podniky a zákazníky. Dnes je naprosto samozřejmou praxí internetové obchodování, pro kontakt mezi zákazníky a firmou je internet běžným médiem. Tato forma komunikace je navíc technologicky podpořena celou řadou nových komunikačních standardů a protokolů, jako jsou elektronické objednávky a faktury ve formátu EDI, XML formát pro přenos strukturovaných dat, webovými službami SOAP a dalšími.

Globální ekonomické prostředí, pro které je typická vysoká míra konkurence, přinutilo firmy začít uvažovat o svých činnostech v nových souvislostech. Upouští se od

pohledu, jehož středem je nabízený produkt nebo služba. Nově se středem zájmu podniků stávají potřeby zákazníka a produkt se poměřuje mírou užitku, kterou může zákazníkovi poskytnout. Jak uvádí Lehtinen *„Zákazníci jsou ústředním bodem, ve kterém se nakonec propojí všechny zdroje, které jsou podniku dostupné*.“ (Lehtinen, 2007). Firmy, které chtějí být na trhu úspěšné, to nutí se stále více zabývat zákazníky jako jednotlivci, zjišťovat jejich individuální potřeby, a to nejen aktuální, ale i předvídat jak se zákaznické potřeby mohou měnit do budoucna. Prostředkem jak tyto potřeby zjišťovat je navázat se zákazníky vztah na úrovni, která je pro obě strany obchodních transakcí přiměřená a výhodná. Existují firmy, které mají jen několik málo zákazníků, pro ty je možné zákaznické vztahy vést na osobní úrovni. Zároveň však existují odvětví podnikání, příkladně bankovnictví nebo pojišťovnictví, pro které je typické velké množství zákazníků. Snahou je nejen získávat nové zákazníky, ale i udržet si zákazníky stávající. Protože se míra užitku z dodávaných produktů u zákazníků během jejich života mění, musí podniky reflektovat změny potřeb a správně na ně reagovat.

Východiskem propojujícím výše zmíněná fakta je CRM – „Customer Relationship Management“ - informačními technologiemi podpořená správa vztahů mezi podniky a zákazníky. Za zákazníky nemusí být považováni nutně jen konečí spotřebitelé, ale i obchodní partneři, nebo obecně kterýkoliv článek z odběratelskou-dodavatelského řetězce.

2. Cíl práce a metodika

Cílem práce je formulace funkčního prototypu systému CRM v prostředí SAP orientovaného na zákazníky v oblasti pojišťovnictví a rozhodnutí o optimálním využití doplňkové webové technologie. Dílčím cílem je přiblížit realizaci technického řešení demonstrací na příkladu z praxe, jimž je zákaznický portál pojišťovny.

Metodou vedoucí k dosažení cíle je rešerše základních teoretických východisek pojednávané problematiky a praktická aplikace analyzované teorie za využití následujících technik:

- Programovací jazyk ABAP,
- technologie tvorby webových sídel,
- softwarové inženýrství a modelování v jazyce UML.

Analytická část práce je členěna do oddílů podle oblastí, ze kterých vychází teoretická fundace práce. Jsou to oblasti CRM v kontextu strategického marketingu, vývoj a programování v jazyku ABAP a popis prostředí informačních systémů SAP, technologie používané při tvorbě webových sídel a datové modelování.

Praktická část se zaobírá tvorbou funkčního prototypu za použití metod softwarového inženýrství a pomocí modelování v jazyku UML. Identifikuje základní případy užití a vypracovává jejich scénáře. Vybrané scénáře poté rozpracovává jak z hlediska jejich statické skladby, tak z hlediska jejich dynamického chování. Implementace a realizace technického řešení a spolupráce systémů SAP s webovými technologiemi je demonstrována realizací případů užití a jejich naprogramováním v jazyku ABAP, s využitím skriptovacích jazyků webu.

3. Definice CRM a jeho vývoj v kontextu marketingu

CRM je zkratkou anglického termínu „Customer relationship management“. Český ekvivalent pro tento pojem je „Řízení vztahů se zákazníky“. CRM zahrnuje rozsáhlou kolekci analytických a obchodních postupů za podpory informačních technologií. Termín se objevuje od konce devadesátých let dvacátého století. Z pramenů lze dohledat následující definice:

„CRM je integrace technologií a obchodních procesů, sloužící k uspokojení potřeb zákazníka v průběhu dané interakce. CRM zahrnuje získávání, analýzu a využití znalostí o zákazníkovi, aby se prodalo více zboží nebo služeb a činilo se to efektivněji.“ (Bose, 2002)

„Řízení vztahů se zákazníky (CRM) je software pokrývající řídící obchodní procesy zákaznického životního cyklu a zajišťující podnikové funkce v prodeji, marketingu a zákaznických službách (včetně kontaktních a call center) prostřednictvím kolaborativní, operativní a analytické složky.“ (Gartner Group, Inc., 2012)

„CRM se snaží poskytnout strategické přemostění mezi informačními technologiemi a marketingovými strategiemi zaměřené na budování dlouhodobých vztahů a ziskovost. To vyžaduje ‚informačně intenzivní strategie‘.“ (Glazer, 1997)

„CRM je marketing řízený daty.“ (Kutner, a další, 1997)

„CRM představuje manažerský přístup, který umožňuje organizaci identifikovat, přilákat a udržet výnosných zákazníků pomocí řízení vztahů s nimi.“ (Hoby, 1997)

„CRM je tvořen sadou procesů a systémů podporujících obchodní strategii, k vybudování dlouhodobě výnosných vztahů se specifickými zákazníky.“ (Ling, a další, 2001)

„CRM je komplexní strategie a proces pro získávání, udržování a vytváření partnerství s vybranými zákazníky za účelem vytvářet vyšší hodnotu pro firmu i pro zákazníka.“ (Parvatiyar, a další, 2001)

„CRM je strategické využití informací, procesů, technologií a lidí, k řízení vztahu se zákazníkem a firmou (marketing, prodej, služby a podpora) po celou dobu zákaznického životního cyklu.“ (Kincaid, 2003)

„Customer Relationship Management (CRM) je obchodní přístup, který integruje lidi, procesy a technologie s cílem maximalizovat vztahy se zákazníky. CRM se stoupající mírou využívá Internetu k zajištění hladké koordinace mezi všemi funkcemi, se kterými se zákazník setkává.“ (Goldenberg, 2008)

Z uvedených popisů je patrné, že se prolíná pohled marketingový s pohledem informačně technologickým a že přesná definice není zatím ustálená. CRM je svým přístupem k zákazníkovi zcela v souladu se změnou paradigmatu v oblasti marketingového mixu. Klasické instrumentárium marketingu je představováno marketingovým mixem „4P“ – produkt, cena, distribuce, propagace (McCarthy, 1995). Zesílenou orientaci směrem od produkce k zákazníkovi a k tržním vztahům popisuje Philip Kotler jako změnu konceptu a přechod od „4P“ k „4C“.

Původní definice 4P:

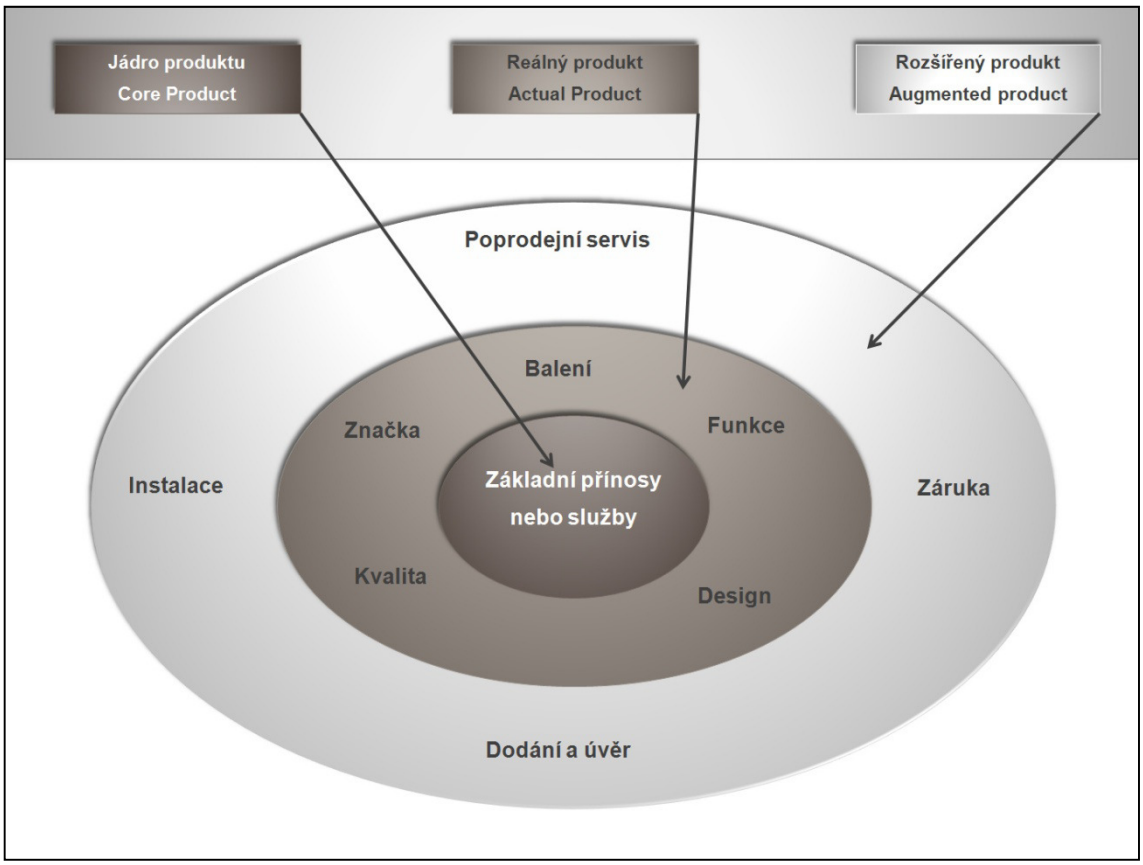
- Product (produkt),
- Price (cena),
- Placement (distribuce),
- Promotion (propagace),

vyjadřující pohled prodejců koresponduje podle Kotlera s 4C spotřebitele:

- Customer (zákazník),
- Costs (výdaje zákazníka),
- Convenience (dostupnost řešení),
- Communications (komunikace),

a dále uvádí, že zvítězí takové společnosti, jež z ekonomického hlediska naplní promptně a s efektivní komunikací potřeby zákazníků (Kotler, 2007). Přístup „4C“ více konvenuje s požadavkem na vytváření dlouhodobých vztahů se zákazníky.

Změna orientace marketingového mixu a zákazníka ovšem neznamená potlačení významu produktu nebo služby jako takové. Naopak rozšiřuje jeho vnímání a poukazuje na to, že užitek zákazníka není dán pouze produktem samotným. Existují i další faktory, s produktem pevně spojené, které mají vliv na jeho hodnotu. Produkt se téměř vždy sestává ze tří částí: fyzického výrobku, informace a služby, která je zákazníkovi poskytnuta (Lehtinen, 2007) . P. Kotler vytvořil model vrstev produktu, který je někdy označován jako cibulový model (Kotler, 2007). Model popisuje rozšiřující se vrstvy, každá vnější obsahuje vnitřní a dohromady představují celkovou užitnou hodnotu, kterou produkt přináší spotřebiteli. Samotné jádro produktu vyjadřuje úroveň



Obr. 1 Kotlerův model produktu, zdroj autor s využitím (Kotler, 2007).

3.1.Typy a součásti CRM

Řízení vztahů se zákazníky je komplexní a různorodá činnost. Popis jednotlivých aktivit CRM je velmi rozsáhlý a přesahuje do různých oborů lidského poznání. Pro lepší uchopení CRM je vhodné klasifikovat jej do typů tak, aby bylo možné lépe vystihnout

jeho problematiku. Francis Buttle rozděluje CRM do čtyř základních typů: strategické, operativní, analytické a kolaborativní CRM (Buttle, 2009).

3.1.1. Strategické CRM

Strategické CRM je obchodní strategie, v jejímž středu je zákazník a jeho cílem je získání a udržení si zákazníků. Je projevem konceptu zákaznické obchodní orientace podniku. Phillip Kotler udává tři další alternativy obchodní orientace podniků: výrobkovou, výrobní a prodejní (Kotler, 2007).

Výrobkově orientovaná přepokládá zákazníka, který si žádá produkt s nejlepší kvalitou, výkonem, designem a vlastnostmi. Je typická pro vysoce inovativní firmy, v počátku jejich existence na trhu, které nabízí jedinečný výrobek. Podnik vyvíjí značné úsilí ke zdokonalování svého produktu, často i za cenu vysoké ceny, nebo s vlastnostmi, které přesahují to, co zákazník vyžaduje.

Výrobně orientovaná přepokládá zákazníka, který se rozhoduje podle ceny. Podnik se snaží udržovat nízké produkční náklady, zisku dosahuje růstem objemů prodeje. Tato orientace je vhodná na trzích, kde se vyskytuje homogenní produkt, nebo existují substituty. Typickým příkladem výrobně orientovaného přístupu je produkce automobilky Ford v první polovině dvacátého století.

Prodejně orientovaná je založená na domněnce, že dostatečně silná podpora prodeje, reklamy, public relations přimějí zákazníky ke koupi. Často tato orientace navazuje na výrobní orientaci, kdy podniky masově vyrábí levné zboží a poté jej důrazně propagují na trhu, aby vyprodaly sklady.

Zákaznicky (též marketingově) orientovaná – je vyjádřením myšlenky, že na uspokojení potřeb zákazníka je založen profit firmy. Firma shromažďuje informace o zákaznících a konkurenci a trzích, aby mohla vyvinout produkt s vyšší užitnou hodnotou. Buttle uvádí, že zákaznická orientace silně koreluje s obchodním výkonem. *„There is evidence that customer-centricity correlates strongly with business performance“* (Buttle, 2009).

3.1.2. Operativní CRM

Operativní CRM zahrnuje množinu interakcí mezi podnikem a zákazníkem. Jedná se zejména o asistovanou podporu obchodních transakcí. Průběhy těchto transakcí jsou zaznamenávány do datového úložiště podniků pro jejich pozdější analýzu a využití. Tento typ CRM se zaměřuje na automatizování prodeje, marketingu a zákaznických služeb. Využívají se komunikační kanály, jako jsou interaktivní webové aplikace, call centra i další, neelektronické způsoby, vedoucí k získávání a uchování informací.

Součástí operativního CRM je automatizovaná podpora prodeje (SFA – Sales Force Automation). Aplikovanou SFA je například online prodej cestovního pojištění firmou ERGO Pojišťovna a.s., (ERGO Pojišťovna, a.s, 2009). Další součástí operativního CRM je vytváření marketingových kampaní a jejich sledování a vyhodnocení. Přínosem operativního CRM pro firmy je zejména generování individuálních zákaznických dat pro každou transakci, což vytváří předpoklad pro budoucí agregaci, segmentaci a vyhodnocování v analytické části CRM. Přínosem pro zákazníka je možné spatřovat v pohodlí a „nízkoprahovém“ přístupu, které mu můžou aplikace SFA nabídnout.

3.1.3. Analytické CRM

Účelem analytického CRM je analýza a vyhodnocování zákaznických dat pro strategické nebo taktické účely. Nad souborem dat lze provádět statistickou analýzu pro základní přehledy, vytvářet extrapolace pro odhady budoucího vývoje a modelovat vzorové situace, které je pak možné použít pro určení směru budoucího vývoje produkce a podniku. Informace získané analýzou CRM dat tak pomáhají k vyhodnocování dopadů marketingových kampaní a jejich optimalizaci. Dále můžou pomoci identifikovat nové prodejní možnosti, taktiku pro získání nových a udržení stávajících zákazníků. V neposlední řadě se spolupodílí na tvorbě cen a vývoje nových produktů. Obvyklým zdrojem dat je operativní CRM, často zasazené do kontextu externích analýz trhu a analýz zákaznického chování.

3.1.4. Kolaborativní CRM

Kolaborativní CRM aplikuje technologie napříč organizacemi hledíc na optimální užitek firmy, obchodních partnerů a zákazníků. Propojuje jednotlivé součásti dodavatelsko-odběratelského řetězce a pomáhá distribuovat informace, umožňující

výrobcům, distributorům a spotřebitelům spolupracovat a dosahovat tak optimálního užitku. Technologie podporující kolaborativní CRM jsou například elektronická výměna dat ve formátu EDI (Electronic Data Interchange), v prostředí SAP je to formát IDoc. Cílem je sdílení relevantních informaci vedoucích ke zvýšení kvality produktů a služeb poskytovaných zákazníkům.

3.2. Ekonomické aspekty CRM

Měření v oblasti zákaznických vztahů je složité, protože jej nelze spolehlivě oddělit od celkových výsledků podnikání, a identifikovat příčinné souvislosti, (Lehtinen, 2007). Za předpokladu, že účelem CRM je přitáhnout dlouhodobé zdroje zákazníků, pak vyjádřením úspěšnosti jeho nasazení v podniku je hodnota označovaná jako návratnost vztahu (*ROR - Return On Relationship*). Jedná se o paralelu s hodnotou návratnosti investice (*ROI – Return On Investment*). Hodnota celoživotní hodnoty zákazníka „*Customer Lifetime Value, CLV*" představuje pravděpodobnost celkového výnosu transakcí, které byly realizovány s konkrétním zákazníkem, zmenšená o náklady, spojené s jeho získáním a udržením. Provozování CRM, kdy operativní složka zachytí zákaznické interakce a analytická je vyhodnotí, může firmě pomoci stanovit referenční bod pro posuzování a v rozhodování, na které zákazníky se zaměřit, jak regulovat sílu vztahu a jakou zvolit taktiku při jednání.

3.3. Etika v interakcích podnik zákazník

Z hlediska etiky je sběr informací o zákaznících a obchodních partnerech citlivé téma. Firmy, v oblasti pojišťovnictví obzvláště, mají k dispozici značné množství osobních údajů zákazníků, nejen jejich personálie, ale i dílčí data o zdravotním stavu, majetkových nebo rodinných poměrech. Je nutné mít na paměti nejen bezpečnost dat, ale i to jak citlivě mohou být z hlediska zákazníků vnímány jejich sběr, uchování a analýza. Společnost, shromažďující data svých zákazníků, na sebe přijímá závazek, vyplývající z důvěry, kterou projeví zákazník poskytnutím informací o sobě. Tento závazek je definován jak legislativními požadavky, tak i obecnou morálkou. Snahu o efektivní vedení podnikatelské činnosti nelze nadřadit těmto hodnotám.

4. Prostředí informačního systému SAP

SAP AG je německá společnost vyvíjející a nabízející celou sadu podnikových softwarových řešení. SAP je zkratka pro „Systems, Applications and Products in data processing“ Sídlo společnosti je ve Walldorfu v Bádensku. Tato společnost je dodavatelem systémů pro řízení podnikových procesů. Původně se jednalo o ERP systém označený SAP/R3, který se postupem času přeměnil v sadu aplikací, pokrývající komplexní podnikovou agendu a zahrnující mnoho odvětví podnikání. Původní myšlenka byla vytvořit systém, který bude reflektovat všechny známé a osvědčené postupy využívané v různých podnikových odvětvích. Jak uvádí Maasen, tyto postupy se od sebe v různých podnicích neliší natolik, aby bylo nutné a smysluplné, aby každý podnik vyvíjel vlastní softwarová řešení. Z tohoto důvodu systém SAP obsahuje standardizované funkce a programy aplikační logiky, které jsou softwarovým vyjádřením podnikových procesů, (Maasen, a další, 2007). Tento přístup se ukázal jako velmi životaschopný, důkazem budiž, že systémy firmy SAP využívá v současné době zhruba 100 000 zákazníků ve více než 120 zemích světa a tento systém existuje v 40 jazykových verzích (Anderson, 2012).

4.1. Architektura SAP z hlediska informačních technologií

Obecným technologickým základem sytému SAP je klasický výpočetní model typu klient-server, založený na třívrstvé architektuře. Systém lze provozovat na různých hardwarových platformách a lze jej nainstalovat na různé operační systémy. Třívrstvou architekturu v systémech SAP tvoří:

- Databáze s datovým slovníkem *DDIC (Data Dictionary).* Představuje základní datové úložiště, které je možné provozovat v různých databázích jako je Oracle MSSQL, IBM DB2 a dalších. Datový slovník je uložen jako metadata v databázi, a uživatelé do něj mohou nahlédnout pomocí transakce *SE11 – ABAP Dictionary*.
- Aplikační (business) logika, která je naprogramována ve vlastním jazyce ABAP. Zdrojové kódy jsou čitelné a přehledně uspořádány v databázi. Protože provozovatelé mají pomocí transakce *SE80 – Object Navigator* přístup ke

zdrojovým kódům aplikační logiky SAP, je zde vysoká transparentnost podnikových operací.

- Prezentační vrstva, zastoupená vlastním tlustým klientem *SAP GUI (Graphical User Interface)* nebo webovým klientem. Tlustý klient je výhodný zejména pro použití v rozlehlých sítích typu WAN, protože značně redukuje datový tok potřebný k prezentaci zpracovávaných dat. Další výhodou tlustého klienta je jednotná interpretace ovládacích prvků a design na různých platformách. GUI je dostupný pro všechny nejrozšířenější operační systémy. Kromě tohoto klienta je k dispozici i tenký klient *SAP GUI for Java*, naprogramovaný v jazyce Java a přístupný pomocí běžného webového prohlížeče.

Bylo zmíněno, že uživatelé mají přístup k objektům datovému slovníku a zdrojovým kódům aplikační logiky. Pokud sami chtějí tyto objekty měnit nebo přidávat, musí mít přidělena patřičná uživatelská oprávnění a od společnosti SAP vygenerovaný vývojářský kód, kterým odemknou systém pro změny. Tento kód je vázáný na konkrétní licenci a instalaci systému a lze o něj požádat na internetových stránkách společnosti SAP.

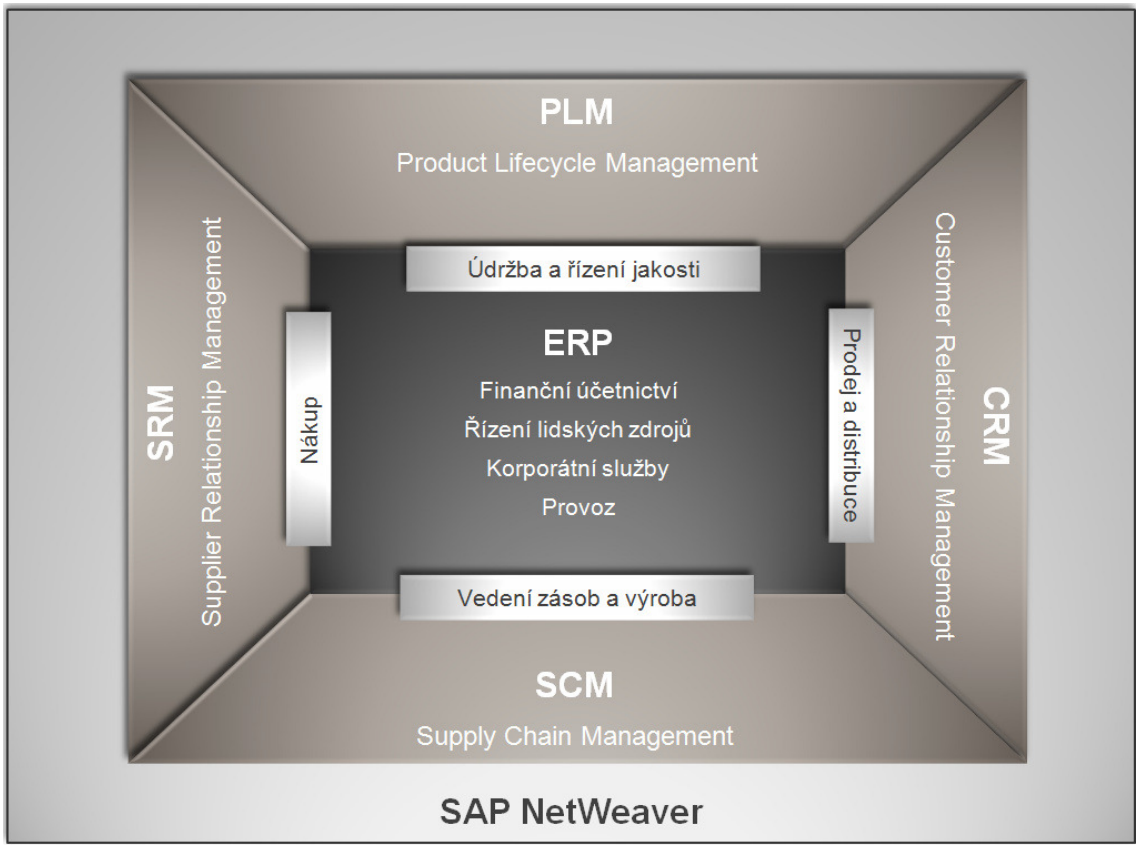
4.2. Architektura SAP z hlediska podniku

Z podnikového hlediska nabízí SAP univerzální řešení pro různá podniková odvětví s konceptem specializace a integrace podle konkrétních potřeb podniku. Sada aplikací je nabízena v konfigurovatelném balíku pod názvem *SAP Business Suite* pro velké firmy, popřípadě v balících *SAP Business One* nebo *SAP Business All-in-one* pro menší a střední firmy.

Jak udává firma SAP na svých internetových stránkách (SAP Česká republika, 2012), základem pro provoz těchto aplikací je technologická platforma SAP NetWeaver a na této platformě postavené následující řešení:

- **ERP** – Enterprise Resource Planning pro správu a řízení podnikových zdrojů,
- **SCM** – Supply Chain Management pro výrobní a logistické procesy,

- **SRM** – Supplier Relationship Management pro procesy pořizování dodávek zboží a služeb,
- **PLM** – Product Lifecycle Management pro vývoj a plánování produkce a životního cyklu produktů,
- **CRM** – Customer Relationship Management pro řízení vztahů se zákazníky.



Obr. 2 SAP Business Suite, komponenty, zdroj autor, podle (SAP Česká republika, 2012)

4.3. Komponenty, moduly a transakce SAP

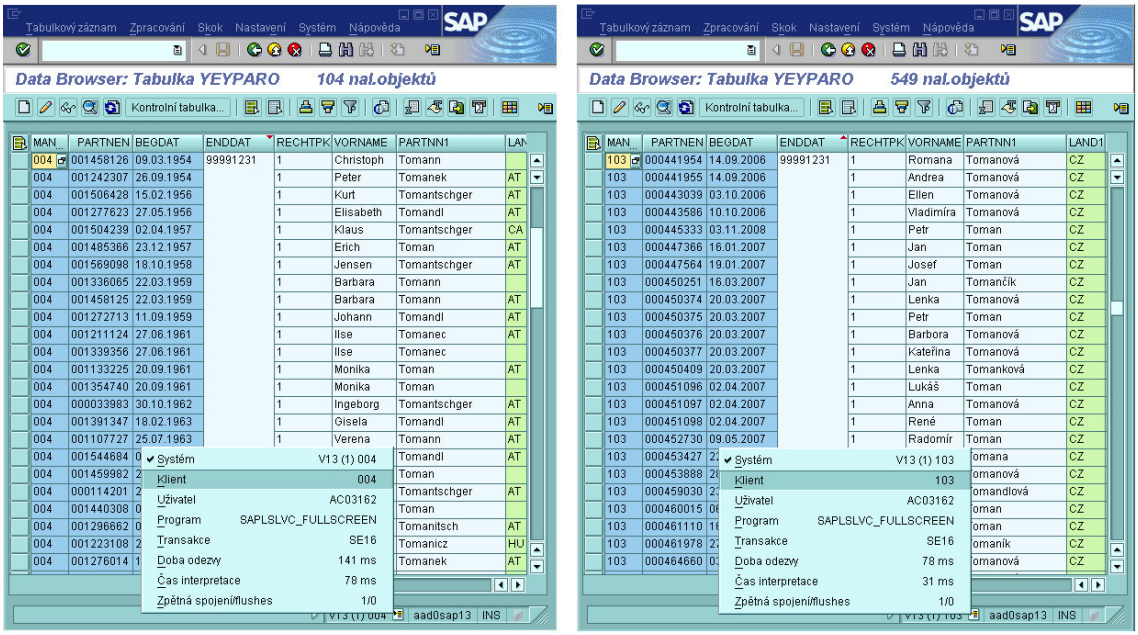
Společnost SAP dodává své systémy v různých modifikacích podle potřeb svých zákazníků. Názvosloví společnosti SAP člení součásti svého systému do hierarchické struktury.

Na nejvyšší úrovni jsou jednotlivé *komponenty* neboli *podnikové aplikace*. Komponenty se sestávají z více modulů. *Moduly* nabízejí funkcionalitu v rámci komponent. Například moduly finanční účetnictví, plánování výroby a materiálové hospodářství společně tvoří komponentu ERP.

Podnikové procesy (scénáře) tvoří posloupnost jednotlivých transakcí. Podnikovým procesem je například prodej. Skládá se z transakcí zadání objednávky, nákupu zásob, odběru zásob ze skladu, expedice dodávky a vytvoření faktury. Některé podnikové procesy vyžadují transakce z více modulů nebo komponent. Při implementaci SAP je třeba mít na paměti, že analýza podnikových procesů a jejich případný reengineering je intelektuálně náročná činnost, která musí zůstat v souladu s cílem podniku. Z tohoto důvodu společnost SAP vyvinula implementační metodiku ASAP (Accelerated SAP), pomáhajícím činnosti spojené s implementací systému SAP projít s co nejmenšími úskalími, v nejkratším čase a s kvalitním výsledkem.

4.4. Koncept klienta SAP

Klient (the client, der Mandant) v prostředí SAP neznamená zákazník. Klientem je v prostředí SAP nazýván samostatný podnik nebo samostatná účetní jednotka. V jednou nainstalovaném systému SAP může být a 999 různých podniků, které využívají stejnou technologickou infrastrukturu a funkce systému, ale mají navzájem oddělená data. Uživatel při přihlášení musí zadat číslo klienta, ke kterému se chce přihlásit, transakce pak probíhají nad daty klienta. Jednotlivé podnikové procesy se řídí pomocí tzv. customizingu, neboli parametrů uložených v databázi, které v každém klientu ovlivňují způsob provádění aplikační logiky. Databázové tabulky SAP jsou společné pro všechny klienty, ale na úrovni datové vrstvy jsou oddělena data jednotlivých klientů. Implementačně je oddělení klientů řešeno v primárním klíči tabulkových záznamů atributem MANDT, obsahujícím kód klienta, kterému záznam patří. Při selekcích a projekcích se specifikace klienta provede automaticky. V aplikační vrstvě při programování v jazyku ABAP už není nutné řešit, jak získat data klienta, systém sám vyhodnotí, kde byl program spuštěn a má k dispozici jen odpovídající datové záznamy. Existují ovšem i speciální, klientově nezávislé tabulky, zejména tabulky obsahující základní systémové parametry a tabulky obsahující metadata jako je datový slovník SAP.



Obr. 3 Rozdílná data stejné databázové tabulky na jednom systému SAP v různých klientech, snímek obrazovky GUI SAP, zdroj autor.

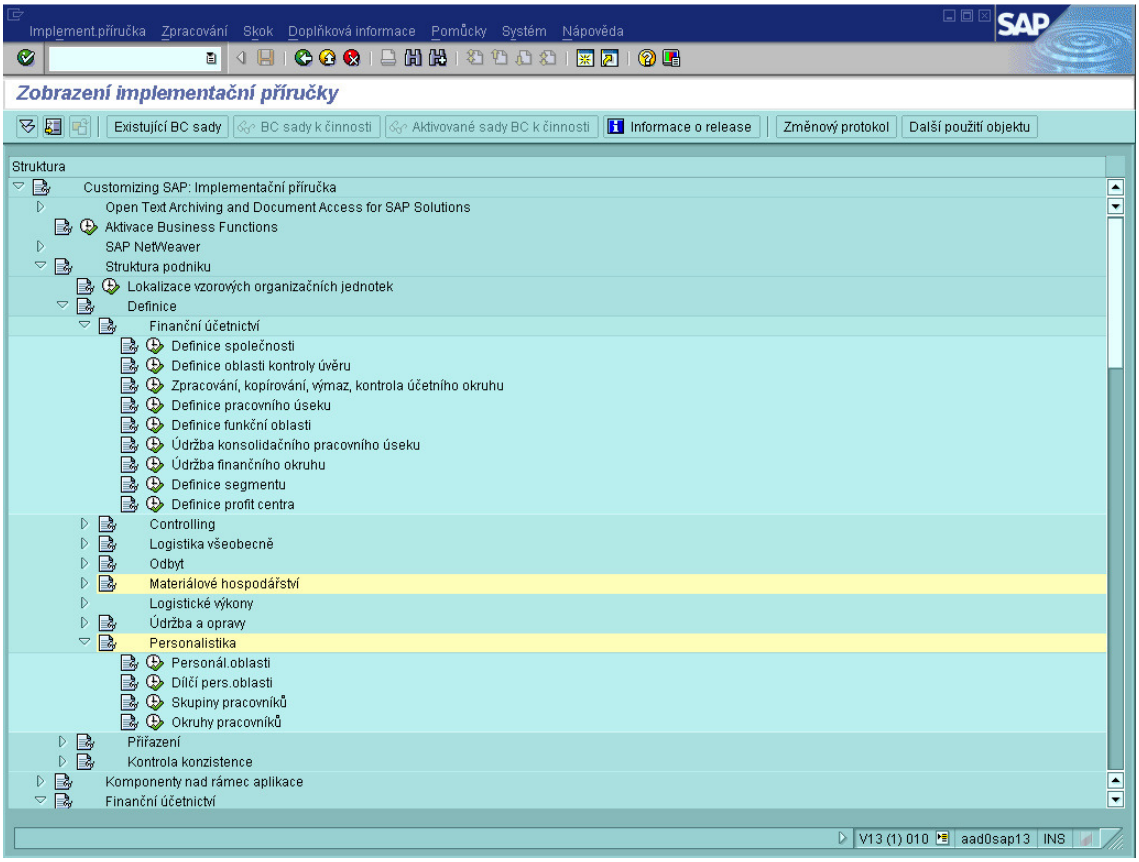
4.5. Možnosti přizpůsobení systému SAP a řízení změn

Většina procesů využívaných při řízení podnikové ekonomiky se mezi jednotlivými podniky neliší natolik, aby bylo nutné psát speciální software pro každý podnik zvlášť (Maasen, a další, 2007). Aplikační logika je v prostředí SAP naprogramována tak, aby postihovala obecné principy obchodních procesů využitelné ve více podnicích. Při implementaci systému se pak provádí přizpůsobení jednotlivým firemním požadavkům. Obecně existují čtyři možnosti, jak systém SAP přizpůsobit zákazníkům:

- Customizing,
- rozšíření standardu SAP,
- změna standardu SAP,
- vlastní vývoj.

Customizing je metoda, kterou doporučuje SAP jako základní mechanismus pro změny v nastavení systému. Pomocí customizingu lze upravit standardně dodávané funkce systému tak, aby odpovídaly tomu, co podnik požaduje. Z pohledu implementace se jedná o data, uložená v tabulkách databáze, která fungují jako parametry pro řízení programů aplikační logiky. Tabulky obsahující takováto data se

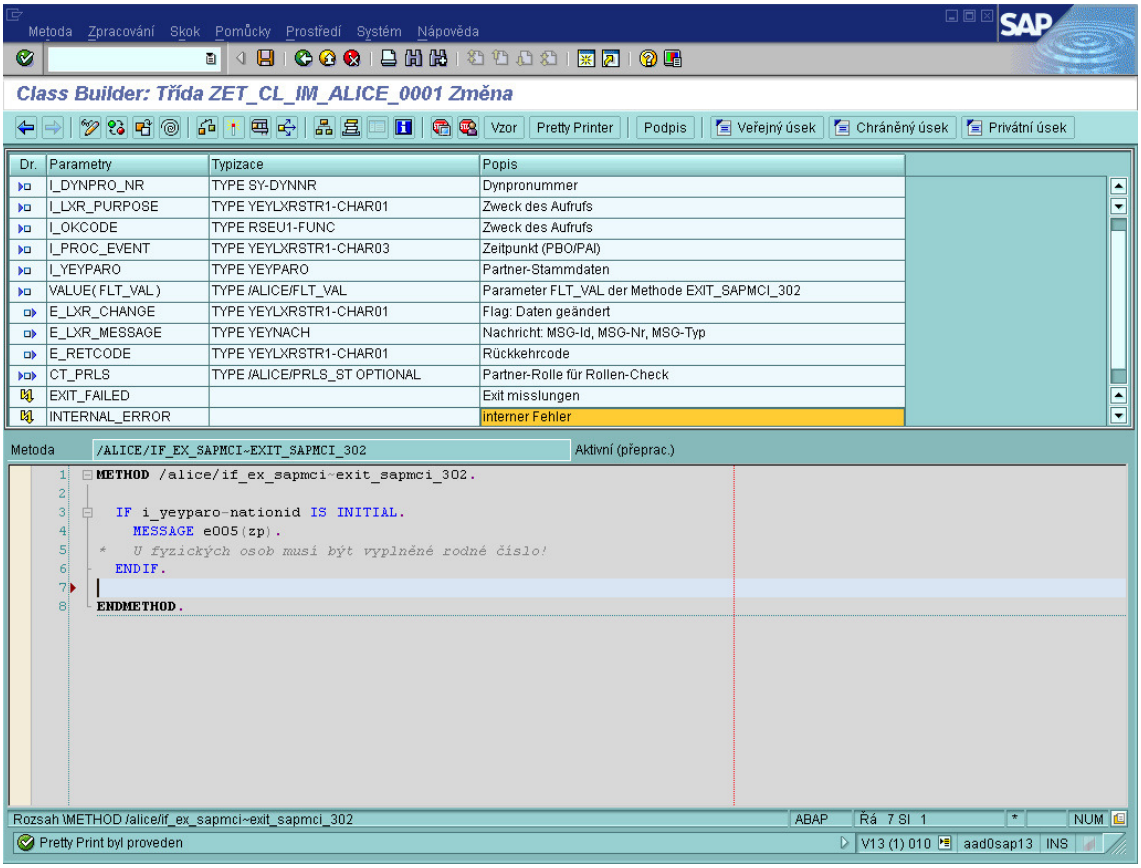
nazývají kustomizační tabulky, mají svůj příznak v datovém slovníku a při změně dat v těchto tabulkách se automaticky vytváří změnový požadavek a ukládá se seznam změn, který je pak možné odeslat i s daty na další servery SAP. Díky tomuto mechanismu lze pak na jednom serveru parametry nastavovat, na jiném testovat a dalším provozovat. Transakce *SPRO - Implementační příručka* (*Implementation Guide, IMG*) je základním vstupním bodem, kudy může zákazník k nastavením customizingu přistupovat. Implementační příručka obsahuje stromovou strukturu, či strukturovaný plán činností, které je zapotřebí vykonat, aby podnik dosáhl požadované funkcionality.



Obr. 4 Implementační příručka customizingu, snímek obrazovky, zdroj autor.

Rozšíření standardu SAP je umožněno díky user-exitům, případně díky jejich následníkům nazývaných Business Add-Ins (BADIs). Standardní programy aplikační logiky SAP obsahují volání statických metod předem připravených tříd s přesně definovanými vstupními a výstupními parametry. Tyto metody jsou otevřené pro změny a zákazník do nich může umístit části svého kódu v jazyku ABAP, který modifikuje logiku aktuálně běžícího programu. Společnost SAP označuje tato místa jako „prázdné

modifikační dutiny“ v programech (SAP AG, 2009). Protože rozhraní tříd BADI je definováno společností SAP, může zákazník upravovat logiku jen v omezené míře. Na druhé straně je tento fakt vyvážen tím, že při změnách a upgrade systému zůstává funkčnost zákaznických rozšíření zachována a zákazníkovi není poskytnuta možnost, aby umisťoval svůj kód na nevhodná místa a narušoval tak integritu aplikační logiky.



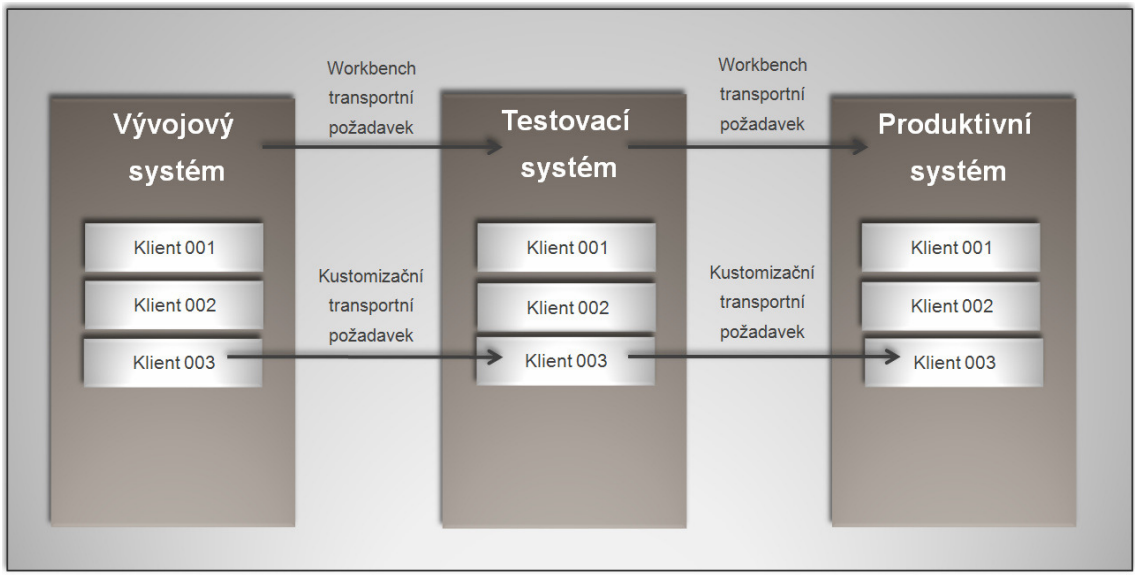
Obr. 5 Rozšíření logiky SAP při kontrole dat, snímek obrazovky, zdroj autor.

Změna standardu SAP znamená odemčení zdrojových kódů dodaných společností SAP a jejich úpravu vlastním kódem. Představuje hluboký zásah do logiky systému a je-li to možné, měl by se zákazník úvahám o změně standardu SAP vyhnout. Takovéto změny totiž vyžadují hlubokou znalost problematiky. Dalším důvodem je, že při upgrade systému tyto změny mohou být přepsány novější verzí programu, což vede k nepředpokládaným změnám v chování systému.

Vlastní vývoj představuje doplnění systému o vlastní programy, transakce, dynpra, databázové tabulky a ostatní prvky datového slovníku SAP. Objekty vlastního vývoje

mohou vhodně doplňovat stávající funkce SAP, nebo mohou vytvářet alternativu ke standardu SAP, pokrývající část řešené problematiky. Pro tyto účely je v systému definován jmenný prostor, obecně platí, že názvy všech objektů, které si zákazník vytváří sám, se označují začátečním písmenem Z nebo Y. Objekty vlastního vývoje se uspořádávají do paketů neboli balíčků, které obsahují všechny tematicky spolu související objekty. Při upgrade systémů na novější verze je někdy třeba vlastní programy upravit, aby reflektovaly změny ve struktuře datového slovníku.

V praxi se obvykle změny systému neprovádějí přímo na produktivním systému, kde pracují uživatelé. Obvyklé a společností SAP doporučené je oddělení vývojové, testovací a produktivní platformy (Maasen, a další, 2007). Veškeré aktivity týkající se vývoje a customizingu se provádějí na vývojovém systému, odkud se přenášejí do testovacího systému a teprve po úspěšném odzkoušení se změny transportují do produktivního systému.



Obr. 6 Organizace vývojového prostředí, zdroj autor s použitím (Maasen, a další, 2007)

Při požadavku na změnu nebo při vytváření nového programu nebo kustomizačních data se automaticky generuje změnový požadavek tzv. transport, který se po dokončení vývoje uvolní a odesílá na testovací nebo produktivní server. Pomocí transportu lze řídit proces vývoje, v jednom transportu může být více objektů, které jsou součástí jednoho projektu. Transport musí být uvolněn všemi vývojáři, kteří se podíleli na změnách. Na

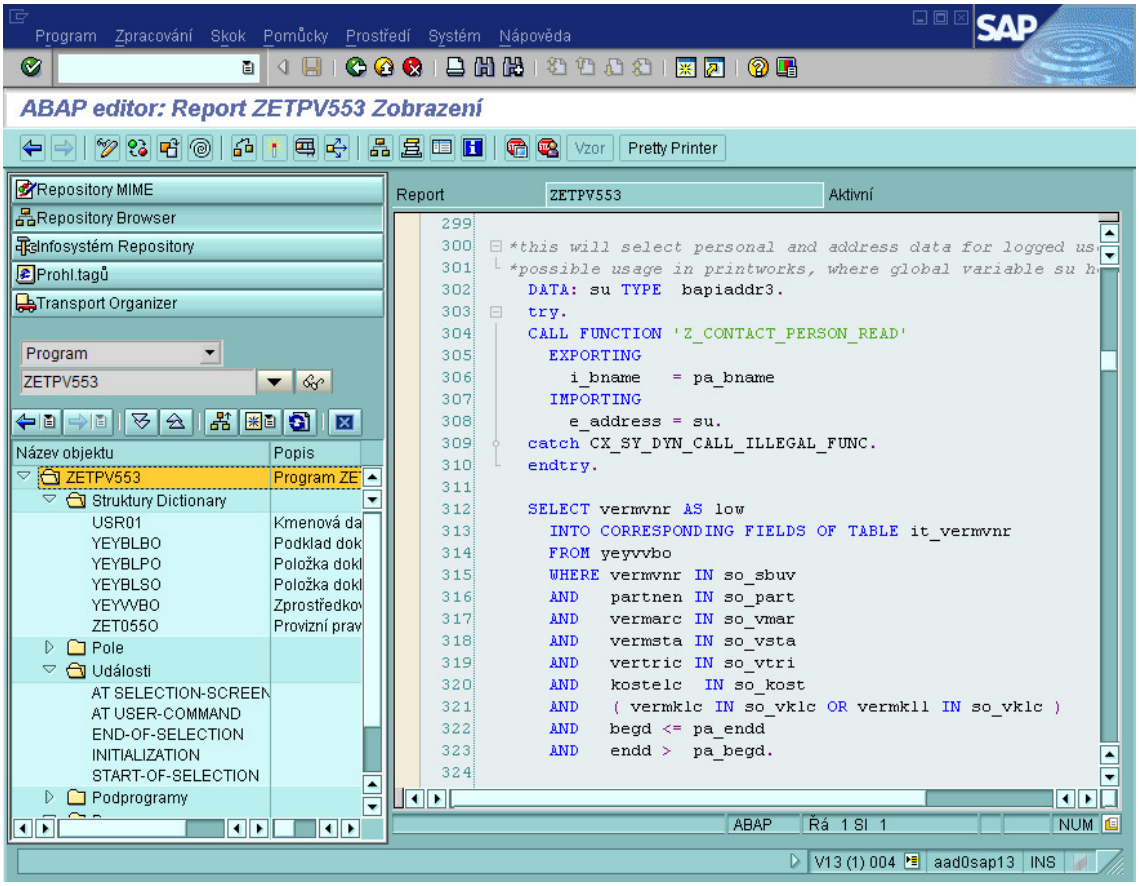
transporty je možné připojit workflow neboli automatické generování zpráv všem zúčastněným osobám a vyžadujícím různé akce jako je potvrzení správnosti. Transportní požadavky se dělí na *kustomizační,* závislé na klientu a obsahující databázové záznamy a *workbench* transporty, které obsahují objekty na klientu nezávislé, tedy většinou programy v jazyce ABAP.

4.6. SAP očima vývojáře – programovací jazyk ABAP

Programovací jazyk systému SAP je *ABAP (Advanced Business Application Programming).* Jedná se o vlastní návrh jazyka, se kterým se nelze setkat mimo prostředí SAP. Tento jazyk je multiparadigmatický, obsahuje prvky deklarativních, procedurálních (imperativních) i objektově orientovaných jazyků - příkazy select/insert/update, funkce, větvení kódu if / case, cykly, procedury i definice tříd, objektů a jejich atributů a metod. SAP umožňuje zákazníkům vyvíjet vlastní aplikace napsané v tomto jazyce. Součástí systému je komplexní vývojové prostředí Object Navigator, přístupné v transakci s kódem *SE80*. Další dílčí činnosti spojené s vývojem mají své vlastní transakce, které jsou ovšem dostupné z prostředí Object Navigator jako jednotlivé položky v menu. Součástí vývojového prostředí je rozsáhlá nápověda, systém správy verzí a debugger pro krokování a odlaďování programů.

Jazyk ABAP pracuje se značným množstvím různých datových typů a struktur. Jsou to jednoduché datové typy (char, string, integer, float…), složené datové typy a pole známé i z ostatních programovacích jazyků. Dále definuje odvozené typy, vzniklé na základě původních typů, majících ovšem doplňkové vlastnosti, jako je výčet rozsahu přípustných hodnot nebo délka typu. ABAP umí pracovat se třídami a objekty. Neobvyklým, ale velmi silným nástrojem pro sekvenční i hromadné zpracování databázových záznamů je datový typ interní tabulky. Jedná se o datový typ, u kterého lze definovat strukturu analogickou k databázovým tabulkám, a může obsahovat násobné záznamy. Představuje tak kolekce vybraných entit, které lze v programech zpracovávat, například v cyklech. Deklarací takzvaných „field-symbols" jazyk ABAP umožňuje i práci s pointery a referencovanými proměnnými. Postup práce s pointery je analogický jako například v jazyce C. Nejedná se zde ovšem o přímý přístup do paměti počítače, ale o přístup do virtuální paměťové oblasti serveru. Kompletní reference

jazyka ABAP je k dispozici v knižní podobě (Keller, 2005), nebo na webu společnosti SAP (SAP AG, 2013)



Obr. 7 Vývojové prostředí Object Navigator v SAP, snímek obrazovky GUI SAP, zdroj autor

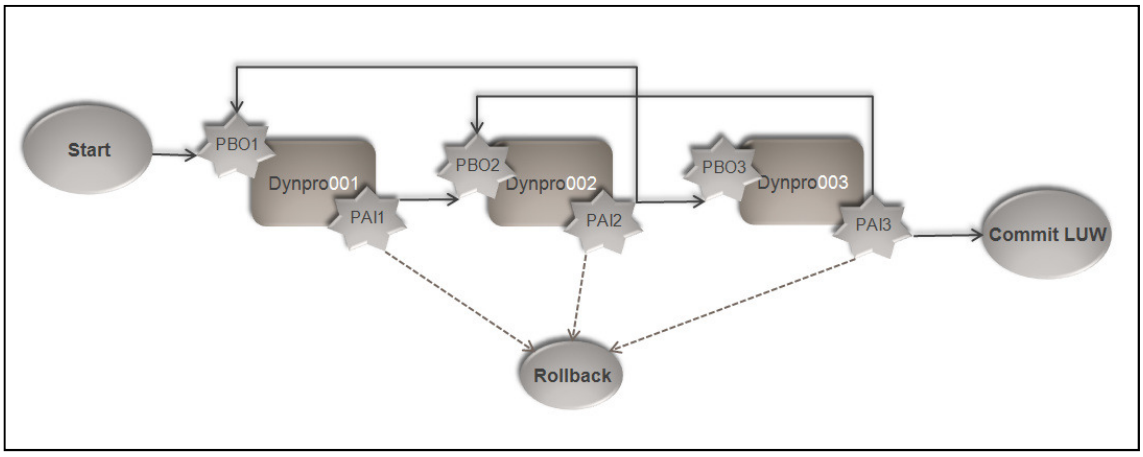
Celá aplikační logika SAP je napsána v ABAP. Zdrojové kódy jsou uloženy jako záznamy v databázi a jsou pro uživatele čitelné, kompilují se do bytecode při prvním spuštění nebo po změně. Systém automaticky udržuje historické verze úprav programů, při změně se vytváří nová verze, stará se archivuje a lze jí obnovit nebo porovnávat jednotlivé verze programů, a to i mezi jednotlivými servery. Tím, že všechny zdrojové kódy aplikační logiky systému jsou pro provozovatele systému čitelné, je dosaženo vysoké transparentnosti systémových operací. Zákazník, či uživatel systému má kdykoliv možnost zjistit, jakým způsobem jsou jeho data zpracovávána.

4.7. Objekty SAP vývoje, workbench a repository

Transakce je logická jednotka zpracování dat, která dávají smysl, jen pokud jsou zadána kompletně a najednou. Používáním transakcí se předchází vzniku nekonzistentních stavů databáze, kdy uživatel vkládá data, která se odkazují na neexistující entity, nebo data nabývají hodnot z nepovolených rozsahů. Transakce také umožňují vrátit stav databáze do předchozího stavu, pokud uživatel nedokončí svoji činnost. Data transakce se do databáze zapisují pokud možno najednou nebo v dílčích logicky uzavřených jednotkách. Konzistence dat patřících do jedné transakce je posuzována jak vnitřně (data musí společně tvořit nerozporný a kompletní celek) i z vnějšku (data transakce musí splňovat všechny předepsané podmínky entitní, referenční a doménové integrity vůči již existujícím datům). V systému SAP, podobně jako v ostatních systémech, se tato kontrola odehrává jak deklarativními tak i procedurálními mechanismy. V případě úspěšného ukončení transakce se potvrdí zápis data do databáze příkazem `COMMIT`. V opačném případě se zápis záznamů ruší příkazem `ROLLBACK`.

Z hlediska uživatele systému SAP se transakcí rozumí posloupnost dialogových kroků zajišťující jeden nebo několik vzájemně souvisejících obchodních případů (Maasen, a další, 2007). Název transakce je tvořen alfanumerickém kódem. Transakce se spouští jejím vyhledáním v nabídce klientské aplikace nebo přímo zadáním jejího názvu.

Dynpro je zkratka pro dynamický program a označuje jednotlivé dialogové kroky (obrazovky, formuláře), umožňující interakci uživatele se systémem. Posloupnost dynper tvoří transakci. Grafické uživatelské rozhraní pomáhá navrhovat WYSIWYG nástroj Screen Painter. Při přechodu z jednoho dynpra do druhého nastávají události *PAI (Process After Input)* a *PBO (Process Before Output)*, které jsou zodpovědné za inicializaci, kontrolu a předání dat. Data se předávají v paměti, k potvrzení aktualizace databáze příkazem `COMMIT` dochází v okamžiku, kdy uživatel ukončí transakci a pořízená data úspěšně projdou formální kontrolou a kontrolou konzistence.



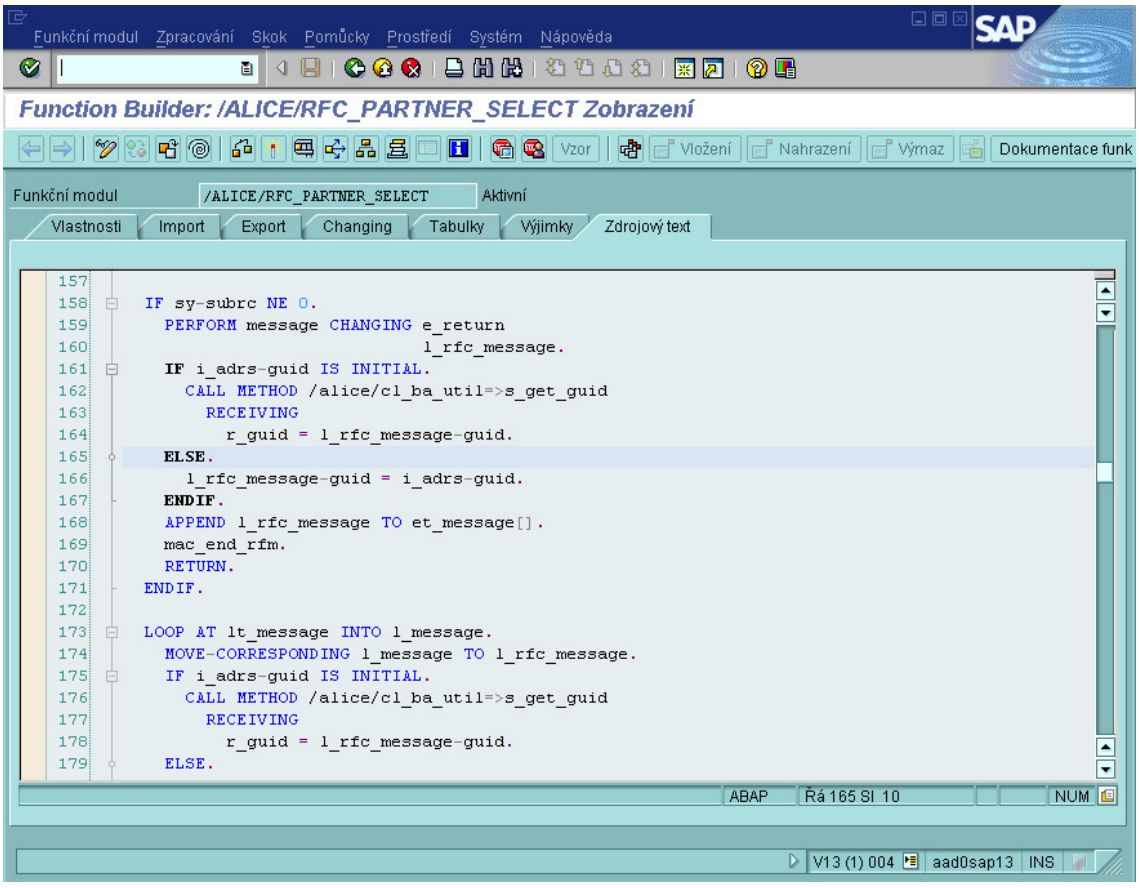
Obr. 8 SAP transakce a průběh dynper jako variace na konečný automat, zdroj autor, s využitím (Keller, 2005)

Report je program procedurálního typu start-cíl. Průběh reportu začíná typicky u výběrové obrazovky, kde uživatel zadává podmínky výběru dat. Poté report provede selekci z databáze a vybraná data podle účelu reportu buď zobrazí, změní, zruší nebo vytvoří nová. Typické využití reportů je pro statistická vyhodnocení a přehledy, dále pro hromadné aktualizace dat (obzvláště v dávkových úlohách, spouštěných automaticky bez asistence uživatelů). Příkladem může být hromadné účtování plateb přijatých na účet v bance, nebo výpis účetního výkazu zisků a ztrát.

User Exits a *Business Add-Ins (BADI) jsou* předdefinovaná místa v originálních SAP programech, ve kterých se volá odskok do uživatelem modifikovaných funkcí. BADI je vylepšeným následníkem původních User Exitů. Jejich volání je založeno na objektovém přístupu a díky polymorfismu jednotlivých metod lze volat rozdílné implementace BADI podle klienta (tedy zákazníka, jak bylo uvedeno v kapitole „Koncept klienta v SAP“). Původní User Exits byly klientově nezávislé a změna se tedy projevila ve všech firmách používající stejný server. Obvykle se volají na začátku zpracování dat pro inicializace, poté během procesu zpracování jednotlivých záznamů pro změny logiky procesu a na konci běhu pro aktualizace a kontrolu nově vytvořených dat.

Funkční moduly jsou samostatně naprogramované procedury v jazyku ABAP, které jsou uspořádány do funkčních skupin. Umožňují zapouzdřit a opakovaně využívat kód v ostatních programech, ať už reportech, user exitech nebo v obrazovkách jednotlivých

transakcí. Programují se jako samostatné jednotky v transakci *SE37-Function Builder* a existují jako samostatné objekty datového slovníku DDIC. U funkčních modulů se definuje rozhraní (interface) vstupních a výstupních parametrů a jejich předepsaný datový typ. Parametrem může být i interní tabulka. Funkční moduly podporují vyvolání výjimek s návratovým kódem (exceptions handling). Vyvoláním výjimky (raise exception) funkční modul oznamuje, že došlo k nedefinovanému nebo nežádoucímu stavu, většinou díky žádosti o operaci s daty, která k takové operaci nejsou způsobilá. Typickým příkladem je pokus o dělení nulou. Ošetření výjimek je pak odpovědností programu, který funkce volá.

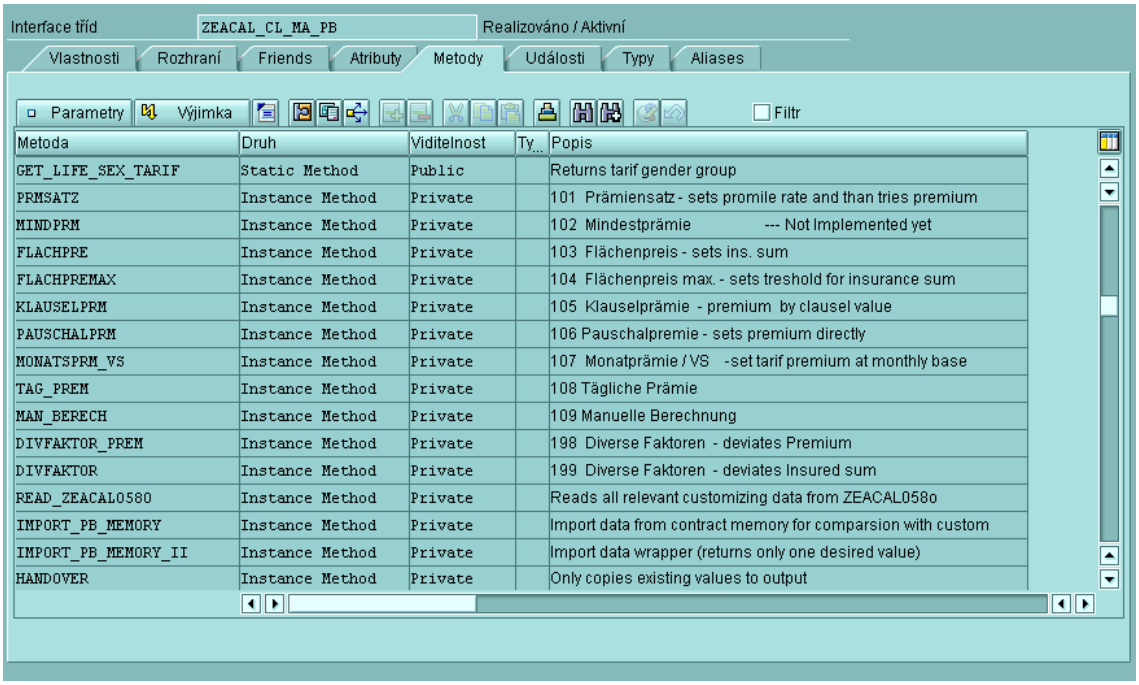


Obr. 9. Funktion Builder, vývojové prostředí, snímek obrazovky GUI SAP, zdroj autor

Funkční moduly hrají důležitou roli v komunikaci a interakci mezi různými SAP systémy nebo SAP a aplikacemi třetích dodavatelů. Pokud se ve vlastnostech funkčního modulu aktivuje volba „Modul schopný remote“ s podporou XML, je možné takové moduly vzdáleně volat i z prostředí mimo systému SAP. Parametry lze pak předávat

pomocí XML rozhraní, nebo pomocí RFC přístupu, které bude popsán v kapitole „SAP - komunikační technologie“.

Třídy, objekty rozhraní: v transakci SE24 (Class Builder) lze naprogramovat definice tříd, jejich atributy a metody a interface tříd. V jakémkoli programu v jazyku ABAP pak lze vytvářet objekty (instance) těchto tříd a dále s nimi pracovat. U tříd je možné naprogramovat i statické metody, tedy takové, kterým je možné předávat zprávy i bez vytvoření instance této třídy.



Metoda	Druh	Viditelnost	Ty...	Popis
GET_LIFE_SEX_TARIF	Static Method	Public		Returns tarif gender group
PRMSATZ	Instance Method	Private		101 Prämiensatz - sets promile rate and than tries premium
MINDPRM	Instance Method	Private		102 Mindestprämie --- Not Implemented yet
FLACHPRE	Instance Method	Private		103 Flächenpreis - sets ins. sum
FLACHPREMAX	Instance Method	Private		104 Flächenpreis max. - sets treshold for insurance sum
KLAUSELPRM	Instance Method	Private		105 Klauselprämie - premium by clausel value
PAUSCHALPRM	Instance Method	Private		106 Pauschalpremie - sets premium directly
MONATSPRM_VS	Instance Method	Private		107 Monatprämie / VS -set tarif premium at monthly base
TAG_PREM	Instance Method	Private		108 Tägliche Prämie
MAN_BERECH	Instance Method	Private		109 Manuelle Berechnung
DIVFAKTOR_PREM	Instance Method	Private		198 Diverse Faktoren - deviates Premium
DIVFAKTOR	Instance Method	Private		199 Diverse Faktoren - deviates Insured sum
READ_ZEACAL058O	Instance Method	Private		Reads all relevant customizing data from ZEACAL058o
IMPORT_PB_MEMORY	Instance Method	Private		Import data from contract memory for comparsion with custom
IMPORT_PB_MEMORY_II	Instance Method	Private		Import data wrapper (returns only one desired value)
HANDOVER	Instance Method	Private		Only copies existing values to output

Obr. 10 Class Builder – vývoj třídy, snímek obrazovky GUI SAP, zdroj autor

Přístup k datům datového úložiště. Objekty datové vrstvy jsou definovány jako metamodel uložený databázi. SAP systémy obsahují několik desítek tisíc databázových tabulek. Datové struktury, typy, domény, tabulky se definují pomocí transakce SE11 (Data Dictionary). Pro přístup k datům v syrové formě lze využít transakce SE16 (Data Browser). Pro použití ad hoc je možné datové typy také definovat přímo ve zdrojovém kódu programů, pak jsou ovšem k dispozici jen po dobu vykonávání programů.

Pro přístup k objektům datové vrstvy v aplikační logice se využívá téměř výhradně prostředků jazyku ABAP. Pro tyto účely ABAP obsahuje skupinu příkazů, která je dialektem jazyku SQL a nazývá se *Open SQL* (SAP AG, 2010). Tento dialekt se

omezuje na část pro manipulaci s daty DML (Data Manipulation Language). Funkce pro definici dat DDL (Data Definition Language) a kontrolu přístupu DCL (Data Control Language) jsou v systémech SAP zajištěny jiným způsobem. Příkazy OpenSQL jsou pomocí rozhraní konvertovány do jazyku specifického pro databázový stroj, na kterém je provozována datová vrstva systému SAP (Keller, 2005). Nejvýraznější rozdíl oproti standardnímu ANSI SQL je, že příkaz select lze zapsat i jako cyklus.

```
SELECT result
       FROM source
       INTO|APPENDING target
       [[FOR ALL ENTRIES IN itab] WHERE sql_cond]
       [GROUP BY group] [HAVING group_cond]
       [ORDER BY sort_key].
  ...
[ENDSELECT].
```

Zdrojový kód 1 Příkaz select – endselect jako cyklus v ABAP, zdroj (SAP AG, 2010)

Při každém průběhu je pak k dispozici právě jeden vybraný záznam. OpenSQL automaticky pracuje s konceptem klientů v SAP. Při přístupu do tabulek s klientově specifickými daty není třeba explicitně specifikovat identifikaci klienta, pro kterého se přístup realizuje. Server sám rozezná, ve kterém klientu byl příkaz spuštěn, a zpřístupní mu pouze relevantní data. Pokud potřebuje programátor získat data i z dalších klientů provozovaných ve stejném systému, může automatické přiřazování klienta potlačit doplněním OpenSQL příkazu o dodatek `CLIENT SPECIFIED`.

4.8. Komunikační technologie systému SAP

Součástí technologické platformy SAP NetWeaver (dříve modul BC – Báze) je sada technologií, umožňující komunikaci systému SAP s okolím. Ne všechny podniky využívají SAP systémy pro všechny své procesy, používají i software jiných výrobců, nebo potřebují automatizovanou komunikaci se svými obchodními partnery, kteří SAP nemají. Proto SAP nabízí infrastrukturu založenou na různých technologiích, které propojí aplikační logiku napsanou v jazyku ABAP s dalšími aplikacemi, napsaných

v jiných programovacích jazycích. Nativním protokolem pro komunikaci serveru SAP je protokol SAP-NI (Network Interface), přenášený standardním TCP/IP protokolem.

Komunikace mezi dvěma systémy může v zásadě probíhat dvěma způsoby: synchronně a asynchronně. Synchronní komunikace využívá jednoho volání funkce na straně odesílatele požadavku a předpokládá, že příjemce zprávy je aktivní a naslouchá. Po přijetí požadavku příjemce potvrzuje přijetí, respektive odešle odpověď na požadavek. Asynchronní volání řadí požadavky do odchozí fronty a příjemce požadavek zpracuje ve vhodnou chvíli jako dávku. Výhodou synchronní komunikace je rychlost a možnost interaktivního chování na straně odesílatele. Výhodou asynchronní komunikace je, že šetří systémové zdroje příjemce, a umožňuje příjemci koordinovat pořadí zpracování více požadavků od různých odesílatelů tak, aby byla zachována logická návaznost. Například příjemce nezpracuje požadavek na zaúčtování plateb z bankovního rozhraní dříve, než založí smlouvy, ke kterým platby náleží.

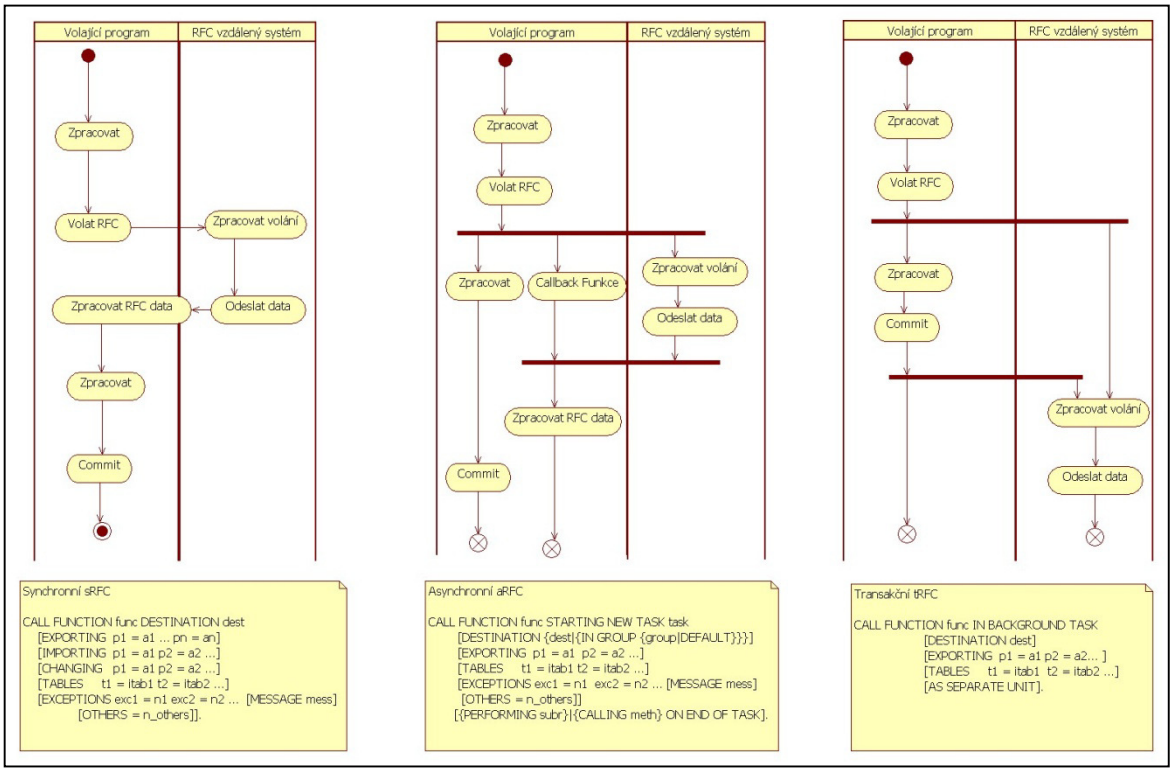
Komunikace mezi SAP a dalšími systémy probíhá zásadně na úrovni aplikační logiky. SAP brání nezprostředkovanému přístupu do datové vrstvy. Společnost SAP dělí sadu svých komunikačních prostředků do dvou skupin, na klasické SAP technologie založené na ABAP a na technologie pro komunikaci mezi ABAP a „Non-ABAP“ technologiemi (SAP AG, 2012). Mezi klasické technologie patří:

4.8.1. RFC – Remote Function Calls

RFC je standardní SAP rozhraní pro komunikaci mezi různými systémy SAP. Funkční moduly, vytvořené pomocí Function Builderu a s aktivovanou vlastností „Modul schopný remote“, mohou být volány ve vnitřním prostředí, a současně nabízí své rozhraní pro volání ze vzdálených systémů v distribuovaném prostředí. Volání se uskutečňuje pomocí síťového protokolu TCP/IP. Při komunikaci mezi dvěma systémy SAP stačí v programu ABAP zavolat funkční modul s parametrem `DESTINATION`. Tento parametr se odvolává na definici konfigurace připojení vzdáleného systému, jeho IP adresu a bezpečnostní nastavení. Definice pro připojení ke vzdáleným systémům se spravují v transakci SM59 - Konfigurace spojení RFC. Pokud požadovaná funkce na vzdáleném systému existuje, vykoná se její kód a vrátí se výsledek. Pro připojení aplikací mimo prostředí SAP je k dispozici knihovna RFC API a vývojářská sada RFC

SDK, obsahující funkce pro připojení k aplikačnímu rozhraní v jazyce C. Programátorům se tak nabízí možnost vyvíjet vlastní aplikace, které využívají funkcí a dat z prostředí SAP.

Volání lze uskutečnit jako synchronní i jako asynchronní. Při synchronním volání volající program počká na dokončení vzdáleného volání a po obdržení výsledku pokračuje ve svém běhu. Pokud se při volání funkce doplní parametr `STARTING NEW TASK`, dojde k asynchronnímu volání, kdy volající program ihned pokračuje ve svém běhu a výsledek vzdáleného volání může být získán pomocí „callback“ rutiny. Princip callback rutiny spočívá ve vytvoření nového vlákna programu, které naslouchá a čeká na odpověď ze vzdáleného systému. Pokud odpověď dorazí, zpracuje se kódem této rutiny. Při volání RFC s dodatkem `IN BACKGROUND TASK` je volaná funkce označená ke spuštění a provede se při nejbližším commitu databáze volajícího systému, tedy buď explicitně příkazem, nebo automaticky při ukončení běhu programu.



Obr. 11 Způsoby volání RFC funkcí v diagramu aktivit UML, zdroj autor.

4.8.2. IDoc Interface/ALE

Rozhraní IDoc – (Intermediate Document) je určeno k výměně obchodních dat s externími systémy. Obsahuje současně definici datové struktury a procesní logiku pro zpracování této struktury. Vnitřní formát rozhraní IDoc je podobný formátu XML, rozšířený o metadata, která popisují procesy a postup, jak s ním má být nakládáno v cílovém systému. IDoc také přenáší stavovou informaci o sobě samém.

Technologie *Application Link Enabling (ALE)* plní funkci middleware mezi jednotlivými instalacemi distribuovaného SAP systému. Zajišťuje řízenou výměnu zpráv mezi oddělenými systémy SAP. Ve sdíleném modelu je definováno, která data budou předávána, z kterého zdrojového systému, na který cílový systém a v jakém typu zprávy to bude. Časté využití je při synchronizaci kmenových dat (Master Data, Stammdaten), například mezi mateřskou společností a filiálkou. Účelem je, aby v obou databázích byl uložen stejný seznam artiklů se stejnými atributy, i když může mít mírně odlišnou formu a mohou s ním být posléze prováděny odlišné operace. Dle doporučení SAP (SAP AG, 2012), jsou IDoc formáty vhodnější pro asynchronní komunikaci, pro synchronní komunikaci je vhodnější používat volání RFC funkcí.

4.8.3. BAPI a Business Objects

Součástí prostředí SAP je sada metod označovaná pojmem Business Application Programming Interface (BAPI). Jedná se o vyšší vrstvu abstrakce aplikační logiky, kde jednotlivá, vzájemně související data a funkce jsou uspořádány do *Business Objects(BO).* Business Objects jsou uloženy v *Business Object Repository (BOR)* a jsou používány pro zajištění funkcionalit jednotlivých obchodních úloh. Jsou založeny na myšlence zapouzdřit data a procesy do jednoho objektu, přičemž samotná implementace může zůstat skrytá.

Každý Business Objekt je instancí své specifické třídy. Vycházíme-li z pojetí objektového paradigmatu (Merunka, 2008), pak je logické, že jednotlivé Business Objekty stejné třídy tedy mají i stejný protokol, na který lze posílat zprávy z aplikačních programů. Nejedná se o čistě objektový, ale spíše o smíšený přístup k OO. Identita BO jako instancí tříd je v SAP definována pomocí:

- *Typu objektu*, tedy třídy, ke které BO náleží,

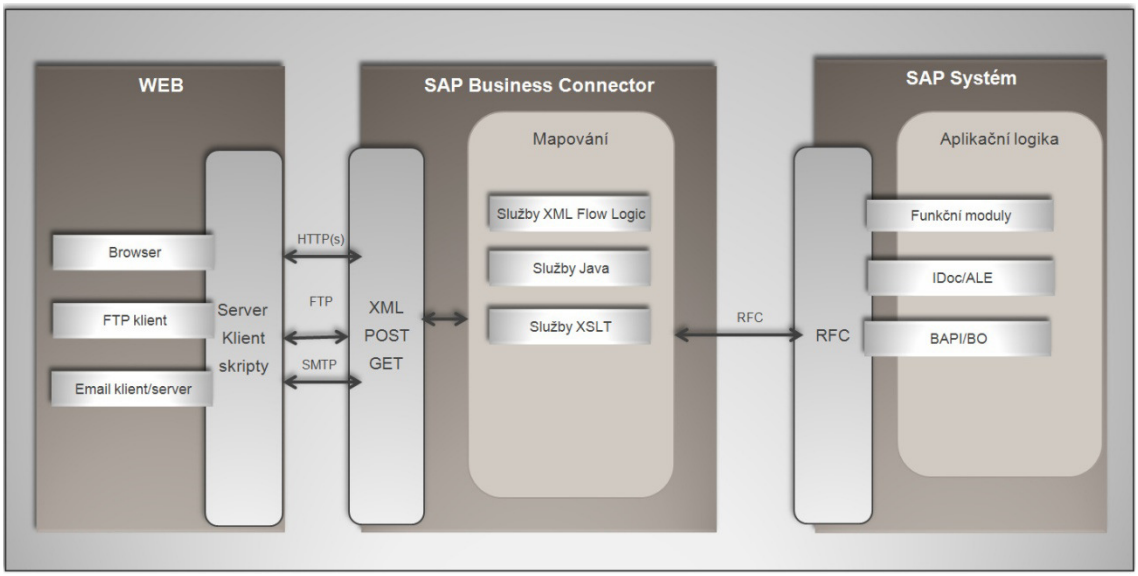
- *klíčových polí*, tedy jednoznačné identifikace, odkazující na související entity v databázi SAP,
- *metod*, tedy množiny zpráv, které lze objektu zasílat a na které reaguje aktivitou,
- *atributů*, tedy datových proměnných, specifikující vlastnosti BO a
- *událostí,* tedy spouštěčů, indikujících změnu stavu BO.

Business Objekty umožňují dědění a polymorfismus. Implementací je XML schéma, popisující rozhraní BO a sada RFC funkcí, které vykonávají předepsané úkony aplikační logiky. Tím je dosaženo, že BO mohou být přenášena technologií BAPI mezi jednotlivými systémy SAP. Při práci s klientem SAP GUI se tento obrací velmi často právě na Business Objekty, které mu poskytují aplikační rozhraní pro požadované operace.

4.8.4. SAP Business Connector

Pro zajištění výměny dat mezi informačními systémy se stále více využívá internetových technologií. Proto vyvstala nutnost vybavit SAP schopností komunikovat pomocí protokolů HTTP, HTTPS, XML a SOAP. K dosažení tohoto cíle je nutná transformace proprietárních struktur systémů do internetových standardů. Výše popsané technologie (RFC, IDoc, BAPI) bylo nutné zpřístupnit internetovým aplikacím.

SAP ve spolupráci s firmou webMethods vyvinul *Business Connector (SAP BC)*. Jedná se o samostatný software, který se instaluje a administruje nezávisle na SAP systému. Instalační soubory jsou dostupné na webu společnosti SAP (SAP AG, 2012). SAP BC obsahuje plně funkční RFC server a klienta. Umožňuje obousměrnou komunikaci v reálném čase se SAP systémy a to synchronně i asynchronně. Z pohledu SAP je komunikace stejná jako s kterýmkoliv RFC serverem. Interní datový formát SAP je poté konvertován do XML nebo HTML, aplikace připojené k SAP BC ze strany internetu mohou zprostředkovaně komunikovat se SAP systémy pomocí otevřených internetových standardů. Z pohledu vnějších aplikací je SAP BO webovým serverem a plní tak úlohu middleware.



Obr. 12 Architektura SAP BO, zdroj autor s použitím (Wittebrock, 2006) a (Bellof, 2006).

SAP BC může být využíván k volání BAPI metod, vykonávat RFC funkce, předávat dokumenty IDoc a využívat ALE služeb. Součástí BO je i *SAP Business Developer*, což je IDE – integrované vývojové prostředí pro správu a konfiguraci služeb serveru. V *SAP Business Developer* je k dispozici knihovna služeb a funkcí pro vytváření dynamického obsahu webu. Jsou to například služba pro práci se soubory, XSL transformace, funkce pro práci s textem a další, naprogramované v jazyku JAVA. Tyto služby lze pak skládat do jednoduchých sekvencí, s možností větvení, cyklů a předáváním parametrů a tímto skládáním pak vytvářet vlastní nové služby.

4.8.5. SAP .NET Connector, Java Connector and Resource Adapter

Představují middleware umožňující vývoj SAP kompatibilních komponent v jazyku JAVA nebo .NET. Java Resource Adapter je adaptér kompatibilní s J2EE. Umožňuje integraci systémů SAP s externím SAP J2EE aplikačními servery. Těchto komponent lze využít při vývoji aplikací v .NET a JAVA, využívajících obousměrné komunikace se servery SAP. Dodávají se ve formě programových knihoven

4.8.6. Internet Communication Framework a Web Service

Je integrovaná komponenta SAP, zajišťující přímé připojení k aplikačnímu serveru SAP pomocí HTTP protokolu. SAP může plnit roli klienta i serveru. S rozšířením SOAP

framework umožňuje vytvářet a používat webové služby, založené na protokolu SOAP. Toto rozšíření je podle společnosti SAP zastaveno ve svém vývoji, a je nahrazeno komponentou Web Service Technologies in SAP Web AS. Tato komponenta centralizuje komunikační technologie a poskytuje jednotné prostředí pro využívání technologií BAPI, RFC, IDoc, popsaných v předešlých kapitolách a přidává SOAP (Simple Object Access Protocol) a WDSL (Web Service Description Language) služby. Je to reakce společnosti SAP na nástup a rozšíření těchto technologií v internetu.

Detailní popis technologií, využívaných ke komunikaci s okolními systémy je možné dohledat na webových stránkách SAP Library. (SAP AG, 2012)

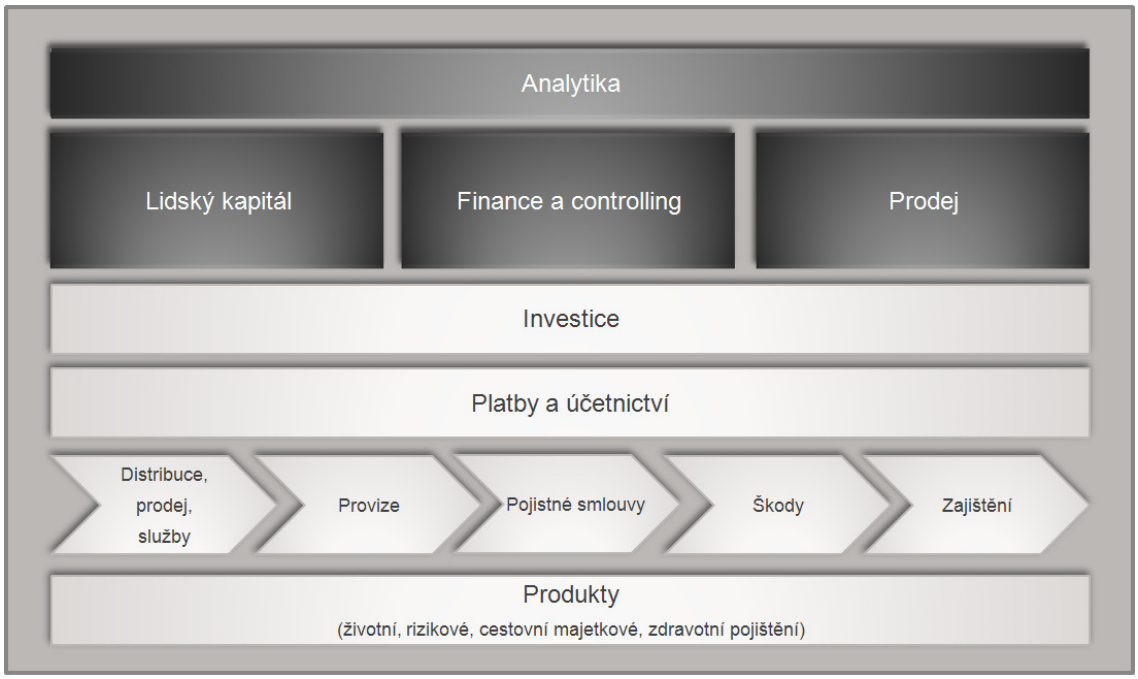
4.9.Pojišťovnictví v prostředí SAP

Od roku 2005 je součástí odvětvových řešení SAP/R3 i řešení pro pojišťovnictví. Schéma na obrázku naznačuje uspořádání komponenty pro pojišťovací odvětví v systému SAP.

4.9.1. Úvod do problematiky obchodního modelu pojišťoven:

Nad skupinou pojistných produktů se odehrává scénář, typický pro pojišťovací obchod. Pojišťovací agenti, makléři a obchodní zástupci sjednávají se zákazníky pojistné smlouvy. Za jejich námahu se při úspěšně sjednaném obchodu vyplácí odměna, většinou ve formě provize. Sjednané pojistné smlouvy je třeba evidovat a spravovat. V případě nehody, kryté pojistnou smlouvou, pojišťovna uhradí škodu. Pojišťovny samozřejmě nejsou ochotny nést veškeré riziko sami, a tak část rizika přenesou na zajišťovny, což jsou velké, většinou nadnárodní společnosti s velkým kapitálem, schopným pokrýt neočekávané události. Zajišťovny samozřejmě za tyto služby vybírají poplatky. Mezi pojišťovnou jejími zákazníky a obchodními partnery se každý den odehrává velké množství finančních transakcí. Jsou to například výplaty provizí, výběry pojistného, výplaty škod a platby zajišťovně. Peníze, které pojišťovna vybírá na pojistném, neleží pouze na jejích bankovních účtech. Část těchto peněz pojišťovny zhodnocují investováním do finančních fondů, akcií a dalších investičních nástrojů. Nad správou financí bdí finanční účetnictví a controlling. Tak jako jiné podniky i v pojišťovnách je nutné vykonávat běžné administrativní úkony, jako je nákup nebo

řízení lidského kapitálu. Pro management společnosti je pak k dispozici analytika, tedy statistické výkazy o kondici firmy ve formě reportů, ulehčujících a urychlujících rozhodování a kontrolu pro vedení společnosti.



Obr. 13 SAP pro pojišťovny, zdroj autor, s použitím (Cummings, 2011)

Podniková aplikace pro pojišťovnictví kopíruje výše zmíněný model a sestává se z následujících komponent:

- Správa inkas a výplat - FS-CD - Collections and Disbursements,
- správa škod - FS-CM - Claims Management,
- správy odměn a provizí - FS-CS - Incentives and Commission Management,
- správy zajištění FS-RI - Reinsurance Management,
- správy pojistných smluv - FS-PM - Policy Management (Tonade, 2010).

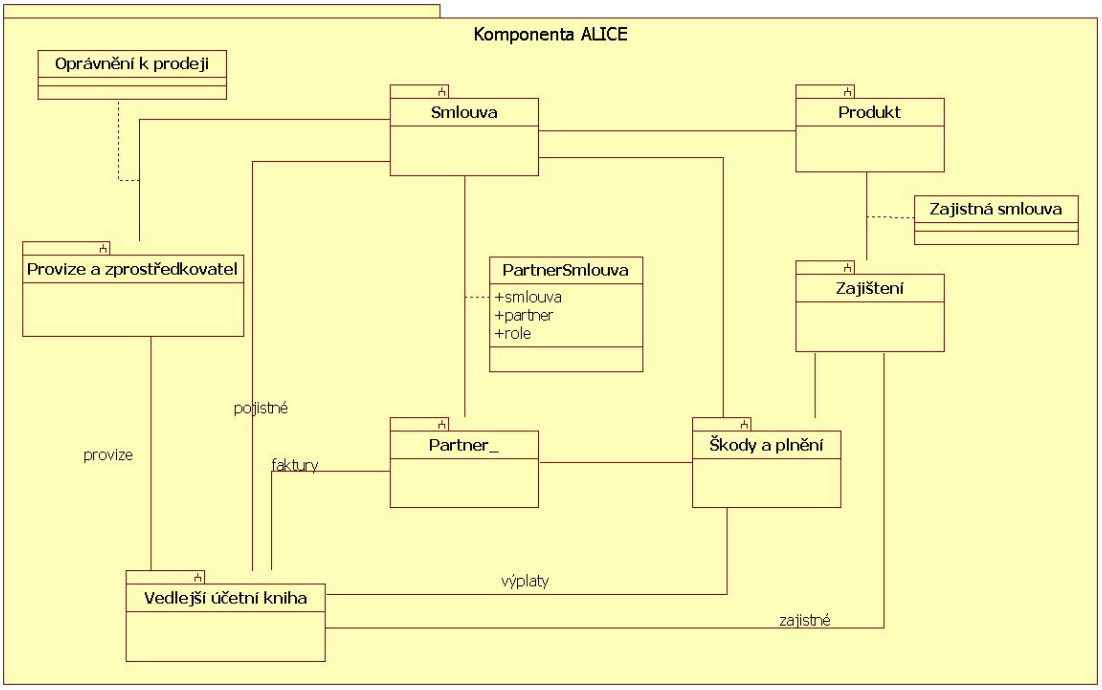
4.9.2. ALICE – externí komponenta pro pojišťovnictví:

Kromě aplikace pro pojišťovnictví v prostředí SAP existuje další pozoruhodné externí řešení, vyvíjené v rámci německého pojišťovacího koncernu ERGO a nazvané ALICE. Toto řešení rozšiřuje funkcionalitu SAP o vlastní komponenty, pokrývající podobnou oblast jako řešení SAP a přidává další funkce jako je implementace pojistné a finanční matematiky, zejména výpočtů majetkového, životního, důchodového a investičního

pojištění. Aplikaci ALICE využívá v současné době většina pojišťoven patřících do koncernu ERGO, zejména v Německu a v Rakousku, ale i v České republice, kde má koncern ERGO svoji pobočku - společnost ERGO pojišťovna, a.s. V praktické části bude návrh rozšíření o CRM funkce, vycházející právě z této komponenty. Podle dokumentace dodavatele (Weinrich, 2007) je komponenta ALICE členěná na následující subsystémy:

- Partner, obsahující správu obchodních partnerů, zákazníků a jejich rolí,
- smlouva, obhospodařující správu pojistných smluv,
- vedlejší účetní kniha, obsahující všechny finanční transakce týkající se pojištění,
- provize pro stanovení odměn za sjednání smluv a péči o zákazníky,
- zajištění, řídící problematiku rozdělení rizik mezi zajišťovnou a pojišťovnou,
- škody, obsahující agendu hlášení a správy škod a výplat pojistného plnění,
- produkt – pro definice pojistných produktů, a výpočty pojistné matematiky.

Obrázek znázorňuje zjednodušené uspořádání jednotlivých subsystémů komponenty ALICE pro pojišťovnictví. Tato komponenta je včleněna do systému SAP zejména spojením modulů SAP finanční účetnictví – ALICE vedlejší účetní kniha.



Obr. 14 SAP – komponenta ALICE pro pojišťovny, zdroj autor.

5. Výpočetní model klient – server, webové technologie

Původní myšlenka výpočetního modelu, kdy se klientská zařízení připojují k serveru a žádají ho o služby, zažívá s rozvojem internetu a zejména dynamickým generováním obsahu webu svojí renesanci. Nejčastěji se využívá třívrstvá architektura, která odděluje

- prezentační služby,
- aplikační logiku a
- datové služby

do tří oddělených a vzájemně komunikujících vrstev (Williams, a další, 2002). Rozložení těchto vrstev je obvykle realizováno tak, že klientu jsou přiděleny úkoly prezentační, server obstarává úkony aplikační logiky a na serveru je také nainstalován databázový stroj (Systém řízení báze dat - SŘBD a báze dat - DB). Takto vymezené hranice však nejsou nepřekročitelné, v praxi je možné se setkat s případy, kdy část úkolů aplikační logiky přebírá klient, nebo je aplikační logika řízena procedurami databázového stroje. Relativně novým trendem zaznamenaným na poli výpočetních modelů je fenomén *Cloud Computing*. Je založen na myšlence distribuovat výpočetní model s důrazem na serverovou část. Klienti se redukují na odběratele služeb, server nabízí služby softwarových aplikací (SaaS), infrastruktury (IaaS) nebo platformy (PaaS).

5.1.1. Dynamický obsah webu

Pojem označuje široké spektrum technologií, které umožňují interaktivní chování uživatelů. Původní koncept obsahu webu předpokládal, že na požadavek klienta server zašle odpověď, která se uživateli zobrazí na jeho počítači. V dnešní době je obvyklé, že webové servery sestavují obsah stránek dynamicky pomocí skriptování, neboli programů, reagujících na požadavky klienta.

Skriptování na straně klienta je zřejmě nejčastěji využíván jazyk *JavaScript*, využívajícího objektového modelu dokumentu *DOM* webové stránky. JavaScript umožňuje dynamicky měnit obsah zobrazené stránky, zejména na základě událostí generovaných uživatelem. V posledních letech s rozvojem HTML5 doznaly některé funkce JavaScriptu zásadních změn. Je to zejména možnost asynchronní výměny dat se

serverem ve formátu XML nebo HTML. Toto nové využití existujících standardů nese označení *AJAX* (W3Schools, 2010). Umožňuje měnit obsah načtené webové stránky nebo její části bez nutnosti aktualizovat ji celou. Uvědomíme-li si, že tak lze využívat k řízení reakcí na uživatelské události kontextu nezměněné části obsahu stránky, docházíme snadno k závěru, že se programátorům dostal do ruky silný nástroj k vývoji interaktivních webů.

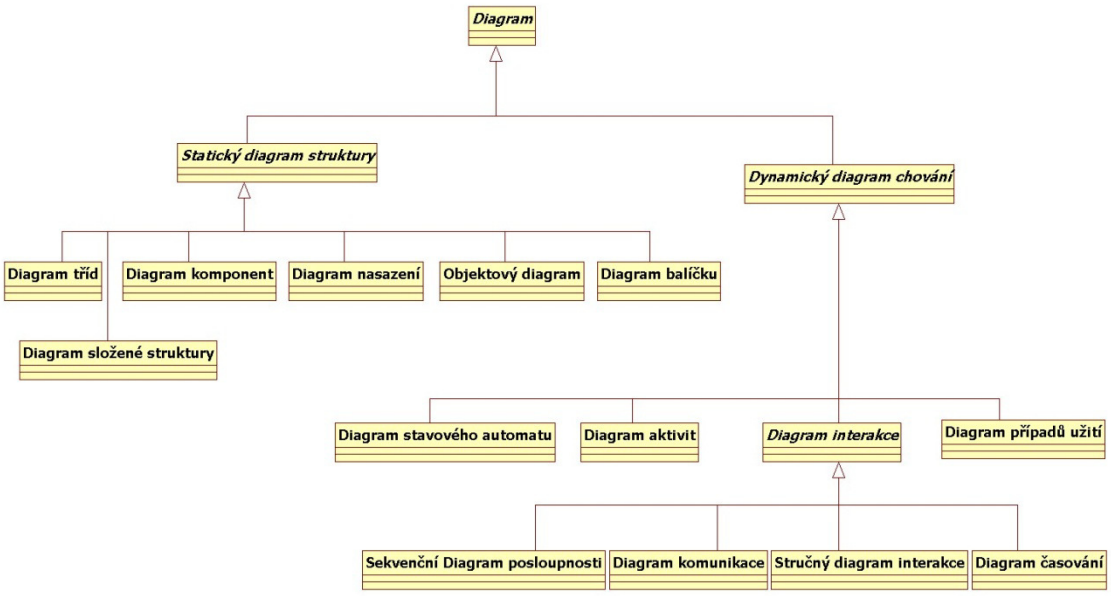
Ke skriptování na straně serveru se nabízí široké možnosti, omezené jen schopnostmi a konfigurací hostitelského zařízení, na kterém je webový server instalován. Častým zástupcem skriptování je jazyk *PHP*, (PHP Group, 2013), který obsahuje bohatou kolekci knihoven funkcí pro práci s textem, soubory, grafickým i multimediálním obsahem, umožňující posílat emaily nebo generovat PDF soubory a mnoho dalších. Často se také využívá jazyku PHP pro komunikaci s databází MySQL. Programy napsané v tomto jazyku jsou pak často nositelem aplikační logiky různých internetových obchodů, diskusních fór, redakčních systémů a dalších.

Komunikace mezi klientem a webovým sídlem je zajištěna pomocí protokolu *HTTP*. Jeho specifikaci lze dohledat v takzvaných *Request for Comments* pod číslem *RFC 2616* (Fielding, a další, 1999). Tento protokol je ve své specifikaci bezstavový, tj. nenavazuje persistentní připojení k serveru, ale pouze asynchronně přenáší požadavky (*requests)* a odpovědi (*responses*). Přenášení parametrů se pak děje pomocí metod *POST* nebo *GET*. Existují i další metody protokolu, které ovšem nejsou tak často využívané. Pro operace, kdy je vyžadována jednoznačná identifikace uživatele a jeho požadavků je nutné suplovat bezstavovost protokolu pomocí cookies, nebo bezpečněji pomocí sessions. Z důvodů zajištění vyšší bezpečnosti byl protokol HTTP modifikován přidáním šifrování *SSL*. Tato verze protokolu nese označení *HTTPS*, a je stále častěji využívanou alternativou k původnímu protokolu.

6. Datové a objektové modelování, UML

UML (Unified Modeling Language) je jazyk pro vizuální modelování systémů (Arlow, a další, 2011). Vznikl sloučením a unifikací více metod, používaných k modelování a vývoji software. Dnes se tento jazyk považuje za průmyslový standard. UML je použitelný nejen pro modelování objektově orientovaných softwarových systémů, ale má i širší využití. Není jen sadou diagramů, za kterou bývá někdy mylně považován, ale je to jazyk, zachycující modely systémů. *Diagramy* nabízejí jen partikulární pohledy na model a ulehčují komunikaci mezi zadavatelem, analytikem a vývojářem ve fázi návrhu. Dalšími stavebními prvky jazyka UML jsou *předměty,* a *relace* mezi nimi. Předměty UML jsou strukturní abstrakce, chování a seskupení. Relace je množina vazeb, umožňující zachytit sémantický vztah mezi předměty (Arlow, a další, 2011). Relací může být závislost, asociace, agregace, kompozice, zobecnění a realizace.

Jak bylo zmíněno diagramy UML hrají roli zprostředkovatele mezi abstraktním modelem a jeho vizuální reprezentací. Na obrázku je seznam diagramů, které se v UML používají. Diagramy se dělí na dvě hlavní skupiny a to podle toho, jaký pohled na realitu nabízí. Jsou to diagramy, jejichž účelem je zachycení statické struktury modelovaného problému a druhou skupinou jsou diagramy, zachycující dynamiku procesů systému, jeho chování a změny v čase.



Obr. 15 Diagramy v UML – hierarchie, zdroj autor s využitím (Arlow, a další, 2011)

Mezi nejčastěji používanými diagramy UML patří diagram *případů užití*, který zachycuje funkčnost, která bude systémem pokryta a typy uživatelů, kteří budou funkce používat (Kanisová, a další, 2006). Jeho předměty jsou aktéři a případy užití, relacemi pak vazby mezi nimi. Diagram se využívá pro uvedení navrhovaného systému do kontextu modelované reality.

Statickým diagramem, zachycující vlastní vnitřní strukturu systému je *diagram tříd*. Tento diagram vychází z objektového přístupu, ale lze jej použít i pro modelování neobjektových modelů, například může nahradit tradiční entitně-relační diagram pro návrh databázové struktury (Entity-Relationship Diagram, ERD).

Z diagramů popisující dynamiku systému jsou to pak *diagramy aktivit*, znázorňující akce a přechody mezi akcemi, větvení akcí a rozhodovací bloky. Lze je vzdáleně přirovnat ke klasickým vývojovým diagramům známým z úloh algoritmizace procedurálního programování. Kromě popisu samotného toku akcí umožňují i označit pomocí vertikálních zón takzvaných *plaveckých drah* i odpovědnou třídu nebo osobu, která činnosti vykonává.

Sekvenční diagramy a *diagramy objektové komunikace* patří mezi diagramy, vyjadřující spolupráci jednotlivých objektů definovaných v systému, a lze je do určité míry mezi sebou nahrazovat. Sekvenční diagramy znázorňují jednotlivé objekty (instance tříd) se svislou čárou života. Vodorovné šipky pak znázorňují zprávy, které si mezi sebou objekty zasílají. Uspořádání sekvenčních diagramů vyjadřuje přehledně časovou posloupnost (sekvenci) jednotlivých zpráv. Diagramy interakce více kladou důraz na schematické rozložení objektů a zprávy je nutné číslovat, aby bylo možné zachytit pořadí, v jakém se vyskytují. Tyto diagramy jsou více zaměřené na výčet možnosti interakcí než na jejich časový průběh.

7. Návrh a realizace prototypu zákaznického portálu

Projektová část práce se zaměřuje na vytvoření funkčního prototypu zákaznického portálu pojišťovny, s využitím znalostí popisovaných v předchozím analytickém oddílu.

Případová studie vychází z reálné situace ve společnosti ERGO pojišťovna a.s., která disponuje vlastním oddělením vývoje IT. Ve společnosti se ozývá volání po zvýšení přidané hodnoty produktů dodávaných zákazníkům ve formě rozšířených informačních služeb s těmito produkty spojenými. Také obchodní partneři a zprostředkovatelé deklarují zájem o lepší přístup k informacím o své činnosti, přehledech jejich pojistného kmene a s tím souvisejících provizí. Firma provozuje svůj informační ERP systém v prostředí SAP, rozšířený o komponentu ALICE pro pojišťovnictví. Přístup do systému mají pracovníci ústředí firmy a management. Velká většina informací, statistik, přehledů a výpisů účtů se dodává zákazníkům a obchodním partnerům zprostředkovaně, verbálně nebo pomocí přenosu datových souborů kancelářských aplikací. K elektronické komunikaci se využívají zejména exporty dat do souborů MS Office, PDF souborů nebo do textových souborů rozhraní, která si pak klienti importují do svých informačních systémů. Tyto dokumenty se přenáší pomocí elektronické pošty nebo sdílením na společném diskovém úložišti.

Jako řešení vedoucí ke zlepšení stávající situace se nabízí zřízení nového, webového komunikačního kanálu v podobě portálu pro zákazníky a obchodní zástupce. Portál by plnil úlohu prostředku automatizované podpory prodeje, tak jak bylo popsáno v kapitole „Operativní CRM“. Zvýšil by úroveň služeb a tím i hodnotu produktů nabízených pojišťovnou, a v neposlední řadě by se mohl stát zdrojem informací o zákaznících.

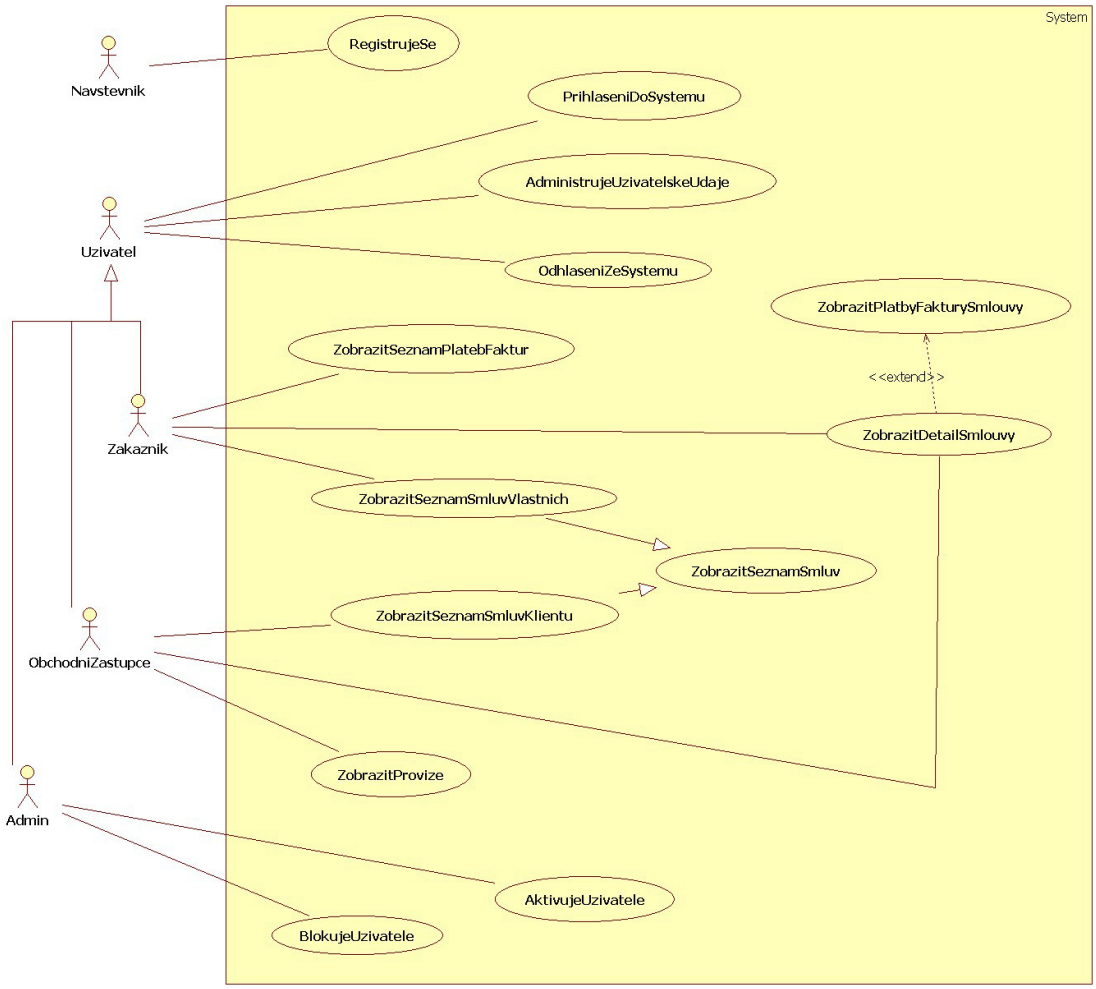
Technické řešení spočívá v aplikaci doplňkových webových technologií ke stávajícímu ERP systému SAP, konfiguraci komunikace mezi SAP a webem, a přizpůsobení a vývoji funkcí aplikační logiky SAP s cílem rozvinout technologické zázemí pro budoucí rozvoj CRM ve firmě.

7.1.Analýza požadavků, případy užití a scénáře

Při definici a specifikaci požadavků na tvorbu informačních systémů je třeba věnovat pozornost tomu, aby požadavky deklarované budoucími uživateli byly relevantní a týkaly se problémové oblasti, a aby výklad jejich formulace byl jednoznačný. Taxonomie požadavků udává rozdělení na *funkční* a *nefunkční požadavky.* Funkční požadavky popisují, co by měl systém dělat, jaké funkce by měl nabízet. Příkladem funkčního požadavku může být „Uživatel může změnit heslo.“ Nefunkční požadavky, odhlédneme-li od lehce pejorativní konotace, představují omezující podmínky, které musí systém splňovat, neboli jak by to měl systém dělat. Příkladem nefunkčního požadavku může být „Hesla a jména uživatelů bude uloženo v databázi MySQL.“

Při návrhu řešení informačních systémů je třeba tyto dva typy požadavků rozlišovat. Při počáteční analýze a návrhu by nefunkční požadavky neměly ovlivňovat modelování budoucího systému. Jejich čas přichází teprve ve chvíli, kdy se dostane na řadu implementace. Autor ze své praxe zná řadu případů, kdy bazírování na nefunkčních požadavcích podstatně zkomplikovalo realizace IT projektů.

Při rozhovorech se zaměstnanci ERGO pojišťovny, vykrystalizovala postupně hrubá představa budoucí funkcionality zákaznického portálu. Jako funkční se objevují požadavky, aby systém nabízel uživatelům přehledy vlastních smluv, plateb a faktur, dále přehledy provizí a možnost spravovat vlastní uživatelská data. Mezi nefunkční požadavky lze zařadit požadavek na bezpečnou komunikaci protokolem HTTPS, design v souladu s firemní vizuální identitou. Po utřídění všech požadavků do seznamu lze identifikovat čtyři aktéry, tedy role uživatelů portálu. Jsou *to Administrátor, Zákazník, Obchodní zástupce, a Návštěvník.* Návštěvník je náhodný uživatel systému, jehož jediná možnost interakce se systémem je požádat o registraci a změnit tím svoji roli. Zákazník představuje klienta pojišťovny, který má s pojišťovnou sjednánu jednu nebo více pojistných smluv a je zaregistrován v systému. Obchodní zástupce má s pojišťovnou sjednánu dohodu o obchodní spolupráci a pojišťuje zákazníky, Administrátor je role, jíž je přiřazena odpovědnost za správu zaregistrovaných zákazníků a obchodních zástupců. UML diagram případů užití (Obr. 15) udává vazby mezi aktéry a jejich činnostmi v systému



Obr. 16 UML Use Case diagram zákaznický portál pro pojišťovny, zdroj autor.

Pro bližší specifikaci funkcí budoucího systému je zapotřebí vypracovat scénáře pro jednotlivé případy užití. Scénáře je vhodné zpracovávat v jednotné formě, Arlow a Neustadt (Arlow, a další, 2011) doporučují použít jednotnou šablonu, obsahující následující informace:

- ID, pro jednoznačnou identifikaci případu užití,
- stručný popis,
- hlavní aktéry,
- vedlejší aktéry,
- vstupní podmínky,
- hlavní scénář,
- výstupní podmínky,

- odkaz na alternativní scénáře.

Protože výpis všech scénářů by znepřehlednil text, je zde uveden příklad scénáře „ZobrazitDetailSmlouvy“, který je rozšířen scénářem „ZobrazitPlatbyFakturySmlouvy“. Kompletní seznam scénářů pro všechny definované případy užití lze dohledat v příloze.

Případ užití	**ZobrazitDetailSmlouvy**
ID	4
Stručný popis	Uživatel čte detail zvolené smlouvy
Hlavní aktéři	Zákazník, Obchodní Zástupce
Vedlejší aktéři	žádný
Vstupní podmínky	Aktér je přihlášen. Požadovaná smlouva patří do jeho seznamu. Je zobrazen seznam smluv.
Hlavní scénář	1. Aktér: vybere číslo smlouvy ze seznamu 2. Systém: načte smlouvu 3. Systém: zobrazí data smlouvy extend:ZobrazitPlatbyFakturySmlouvy
Výstupní podmínky	Žádné
Alternativní scénáře	Žádné

Tabulka 1 Scénář případu užití „ZobrazitDetailSmlouvy“

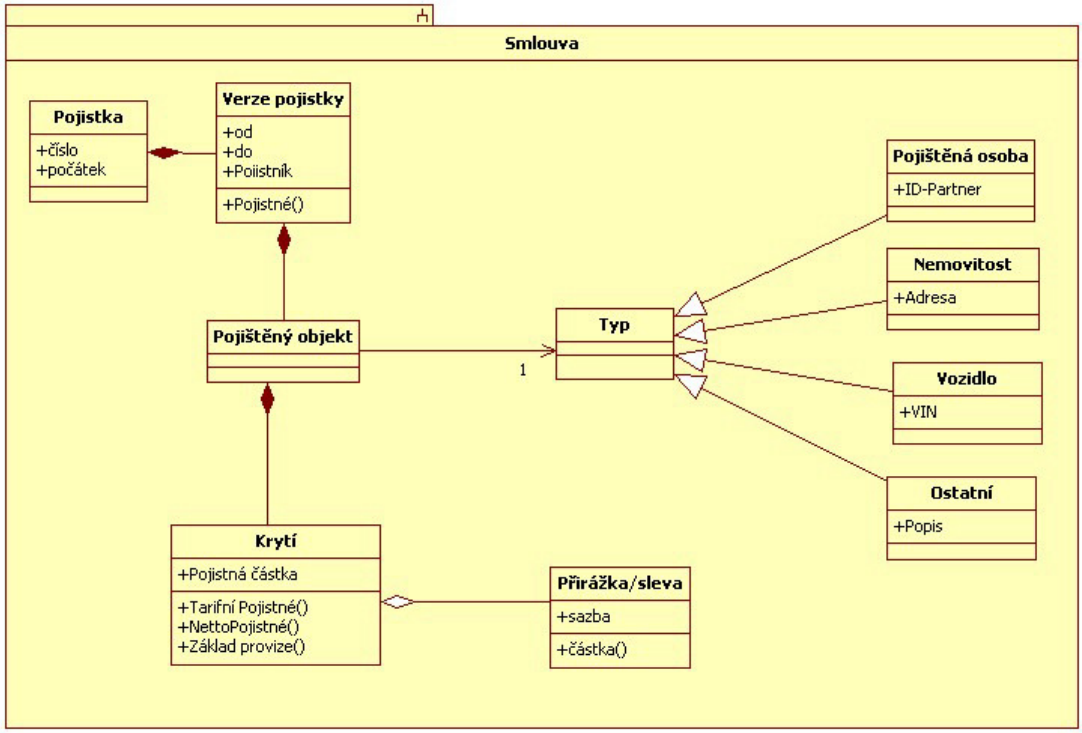
Případ užití/Extend	**ZobrazitPlatbyFakturySmlouvy**
ID	4.I
Stručný popis	Uživatel čte platby zvolené smlouvy
Hlavní aktéři	Zákazník, Obchodní Zástupce
Vedlejší aktéři	žádný
Vstupní podmínky	Aktér je přihlášen. Smlouva patří do jeho seznamu. Systém zobrazuje detail smlouvy.
Hlavní scénář	1. Aktér: vybere příkaz Zobrazit Platby 2. Systém: načte Platby smlouvy 3. Systém: zobrazí data plateb smlouvy
Výstupní podmínky	Žádné
Alternativní scénáře	Žádné

Tabulka 2 Scénář případu užití „ZobrazitPlatbyFakturySmlouvy“

7.2. Statický model a struktura dat

Model je do značné míry determinován existujícím řešením komponenty ALICE v systému SAP. Dlouhodobě vyvíjená a prověřená implementace představuje dobrý zdroj inspirace pro obdobná řešení. Přestože se v případě ALICE jedná o entitně-relační návrh databáze, lze použít diagram tříd jazyka UML pro jeho popis. Jednotlivé třídy pak v tomto zápisu představují databázové tabulky, objekty těchto tříd jsou pak jednotlivými entitami těchto tabulek. Samozřejmě dědění, použité v následujících diagramech, není možné v entitně relačních modelech realizovat. Tento úkol musí suplovat aplikační logika, většinou pomocí větvení kódu v podmínkách. Při vytváření scénářů jednotlivých případů užití se objevily následující entity nebo atributy entit, se kterými jednotlivý aktéři operují: smlouva, platba a faktura, provize, uživatelské jméno a heslo, adresa, email, telefon.

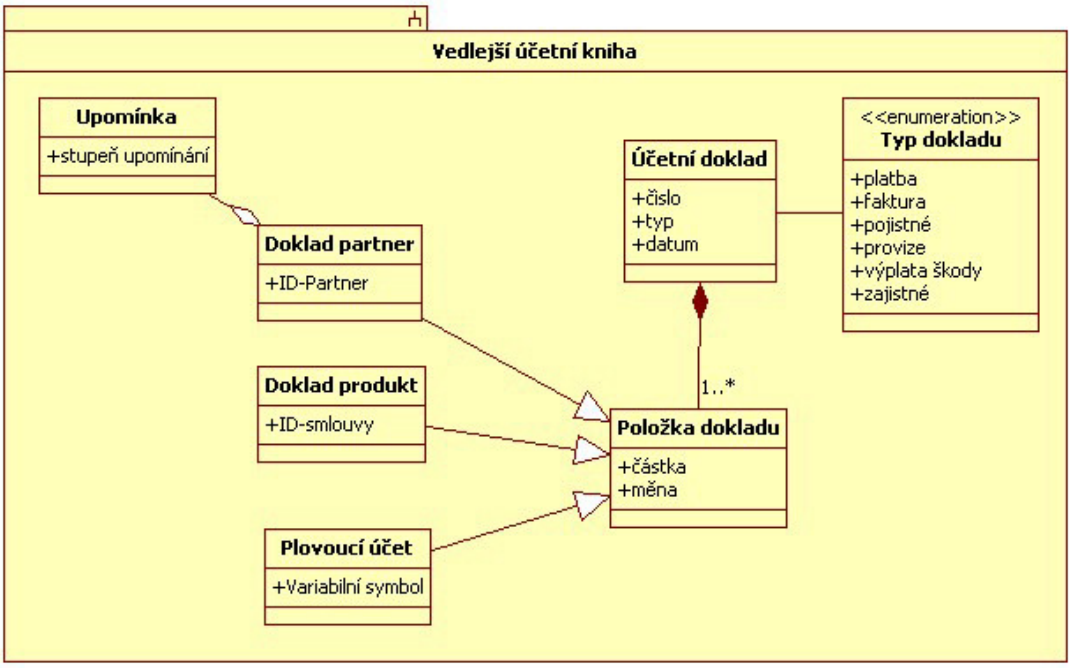
Návrh smlouvy je tvořen skládáním entit. Jedná se o kompozici „*pojistka - pojištěný objekt – krytí*“. UML Diagram na obrázku popisuje vzájemné relace mezi jednotlivými třídami entit.



Obr. 17 Subsystém smlouva, zdroj autor s využitím (Weinrich, 2007).

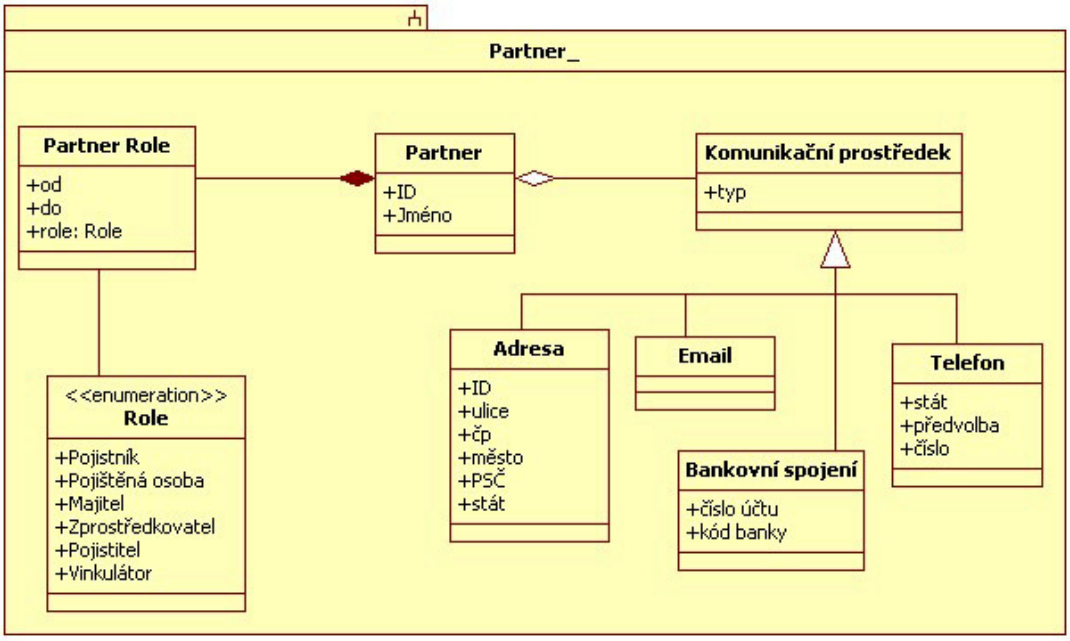
Termín krytí označuje druh pojištěného rizika. Pojištěný objekt, uváděný v kontextu pojišťovnického názvosloví, je nutné rozlišovat od objektu jako instance třídy. Bohužel názvosloví obou oborů se zde dostávají do konfliktu. Mocnost vazby je obecně 1:N, u pojištěných objektů je užito generalizace a odvozené třídy pak představují různé typy pojištěných objektů. Takové uspořádání pojištěných objektů dle jejich typů odkazuje na návrhový vzor „Stav“, popsaný například zde (Merunka, 2008).

Návrh vedlejší účetní knihy plateb a faktur je tvořen účetním dokladem (hlavičkou dokladu, obsahující správní informace) a jednotlivými položkami dokladu. Tyto položky se opět vyskytují v různých variantách (stavech) v závislosti na jejich určení.



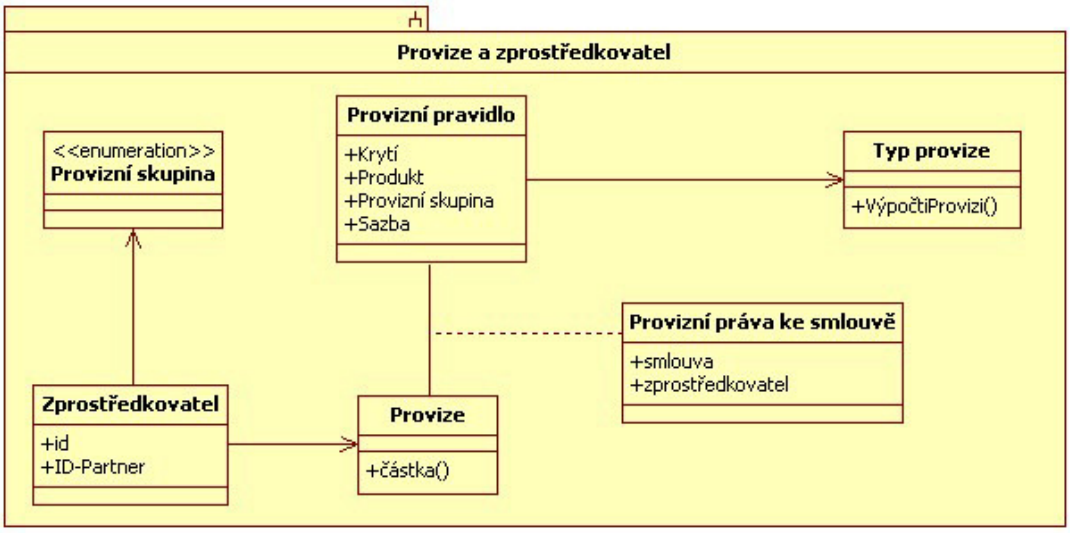
Obr. 18 Subsystém pro doklady vedlejší knihy, zdroj autor s využitím (Weinrich, 2007).

Subsystém partner zachycuje obecně zákazníka pojišťovny, nikoliv uživatele systému. Model bude muset být dodatečně doplněn tak, aby bylo možné nakládat s uživatelskými účty. Jedná se zejména o vazbu mezi zákazníkem a uživatelem systému a o administraci uživatelských účtů a jejich použití při přihlašování.



Obr. 19 Subsystém partner, zdroj autor s využitím (Weinrich, 2007).

Provize a zprostředkovatel stojí mezi smlouvou a vedlejší účetní knihou. Pomocí asociační třídy „Provizní práva ke smlouvě“ se stanoví, jestli se zprostředkovateli vypočítá provize podle provizního pravidla a posléze vyúčtuje. Typ provize určuje způsob výpočtu provize a je součástí provizního pravidla. Provizní pravidlo se uplatní, pokud jeho atributy odpovídají atributům zprostředkovatele, produktu smlouvy a krytí.



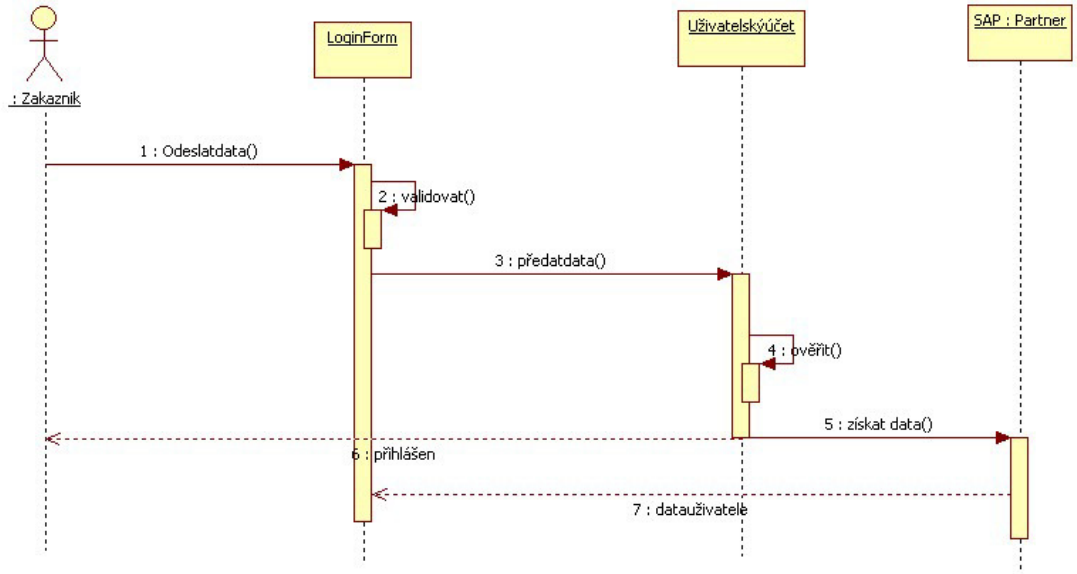
Obr. 20 Subsystém pro provize, zdroj autor s využitím (Weinrich, 2007).

7.3. Dynamický model případů užití

Portál je ve většině svých funkcí poměrně statický, používá transformace již zpracovaných dat. Pouze několik scénářů případů užití vyžaduje spolupráci více stran. Jedním z nich je přihlášení do systému, kde se validuje platnost zadaných údajů po formální stránce, poté se ověřuje jméno a heslo proti otisku uloženému v databázi a po úspěšném ověření se doplní data z kmenového záznamu uživatele v SAP.

Případ užití	PřihlášeníDoSystému
ID	1
Stručný popis	Přihlášení uživatele do systému
Hlavní aktéři	Uživatel
Vedlejší aktéři	žádný
Vstupní podmínky	Uživatel není přihlášen Uživatel není blokován
Hlavní scénář	1. Uživatel: zadá jméno a heslo. 2. Systém: validuje jméno a heslo 3. Systém: ověří jméno a heslo 4. Uživatel: je přihlášen 5. Systém zobrazí uživatelská data
Výstupní podmínky	Uživatel je přihlášen
Alternativní scénáře	NeplatnéJménoHeslo

Tabulka 3 Scénář případu užití „ZobrazitPlatbyFakturySmlouvy“

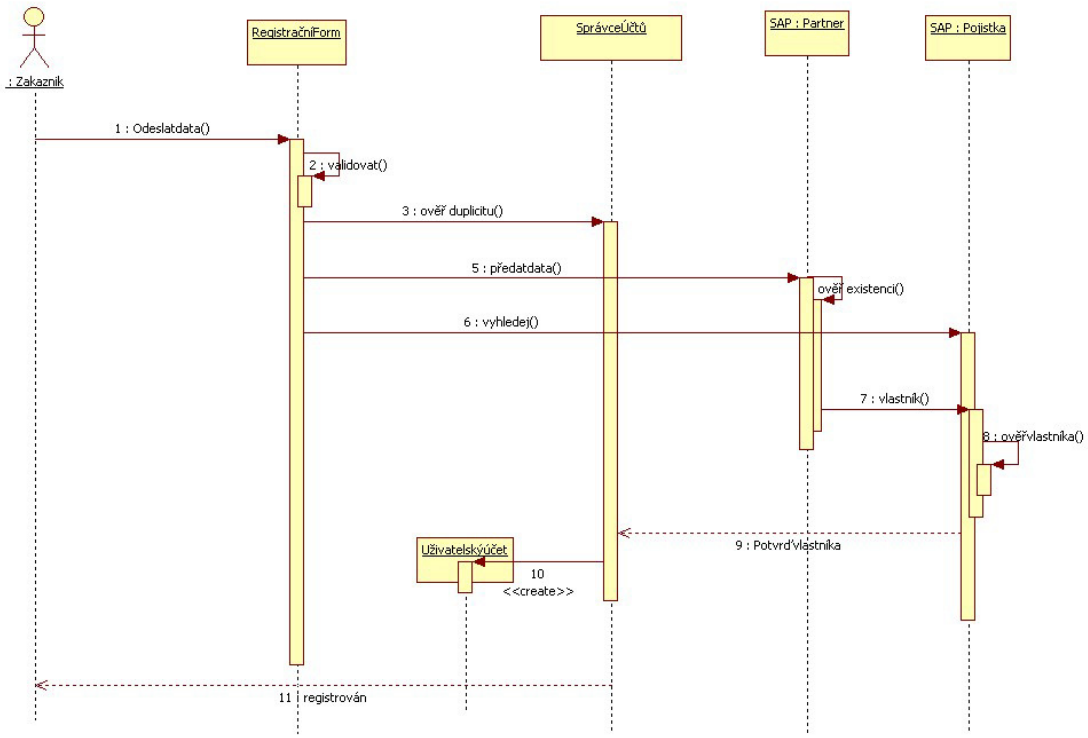


Obr. 21 Sekvenční diagram přihlášení do systému, zdroj autor.

Dalším scénářem s dynamickým chováním je registrace nového uživatele. Podle scénáře je nutná kooperace více objektů. Návštěvník se obrací na systém s žádosti o registraci, které je vyhověno v případě, že prokáže že má alespoň jednu sjednanou pojistku a uvede její číslo, a dále uvede své správné osobní údaje tj. jméno, příjmení a rodné číslo. Realizace scénáře pak může probíhat tak, že systém zkontroluje formální správnost (validace) zadaných údajů, poté věcnou správnost, a konfrontuje databázi SAP s dotazem, jestli existuje osoba a patří jí smlouva. Pokud ano, systém založí nový záznam uživatele.

Případ užití	**RegistujeSe**
ID	3
Stručný popis	Registrace nového uživatele
Hlavní aktéři	Návštěvník
Vedlejší aktéři	Uživatel
Vstupní podmínky	Uživatel není přihlášen. Návštěvník není registrován.
Hlavní scénář	1. Návštěvník zvolí registraci 2. Systém: zobrazí registrační formulář 3. Návštěvník: zadá jméno, příjmení a budoucí heslo, rodné číslo a číslo některé ze svých smluv 4. Systém: validuje úplnost a formální náležitosti zadaných údajů 5. Systém: ověří jméno a rodné číslo v db zákazníků 6. Systém: ověří, jestli existuje smlouva a patří zákazníkovi 8. Systém: zkontroluje, jestli účet už není založen 7. Systém: vytvoří nový účet 8. Systém: zobrazí výsledek registrace 9. Návštěvník: je zaregistrován
Výstupní podmínky	Návštěvník je zaregistrován
Alternativní scénáře	NevalidniRegistracniUdaje NeplatnéRegistračníUdaje UcetJizExistuje

Tabulka 4 Scénář případu užití „ZobrazitPlatbyFakturySmlouvy“

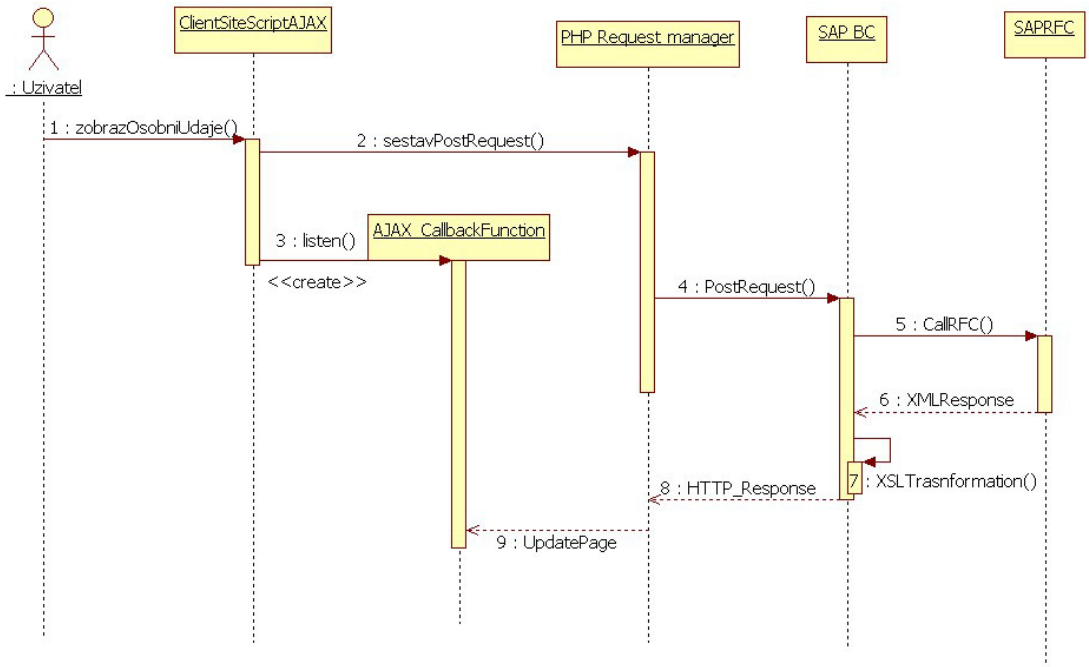


Obr. 22 Sekvenční diagram registrace do systému, zdroj autor.

7.4.Implementace a realizace vybraného případu užití

Technická realizace využívá aplikační logiky komponenty ALICE, napsané v jazyce ABAP. Jednotlivé případy užití předpokládají vytvoření funkčních modulů RFC a jejich mapování pomocí programu SAP Business Connector. Ten je také zodpovědný za XSL Transformace vrácených hodnot do HTML. Generování požadavků a jejich výsledky se do stránky umísťují pomocí technologie AJAX – asynchronního volání JavaScriptu. Předání volání jednotlivých funkcí od klienta zajistí skript na straně serveru napsaný v jazyku PHP. Důvodem, proč předává volání skript na straně serveru je, že technologie AJAX se z bezpečnostních důvodů může pracovat s požadavky pouze uvnitř vlastní domény. Toto je mimochodem dobrá ukázka tzv. nefunkčního požadavku, Design grafické prezentace zabezpečí kaskádový styl.

Představíme-li si jednotlivé komponenty, účastnící se komunikace od uživatelského požadavky až po jeho zobrazení, jako instance tříd, lze volání zachytit sekvenčním diagramem UML.



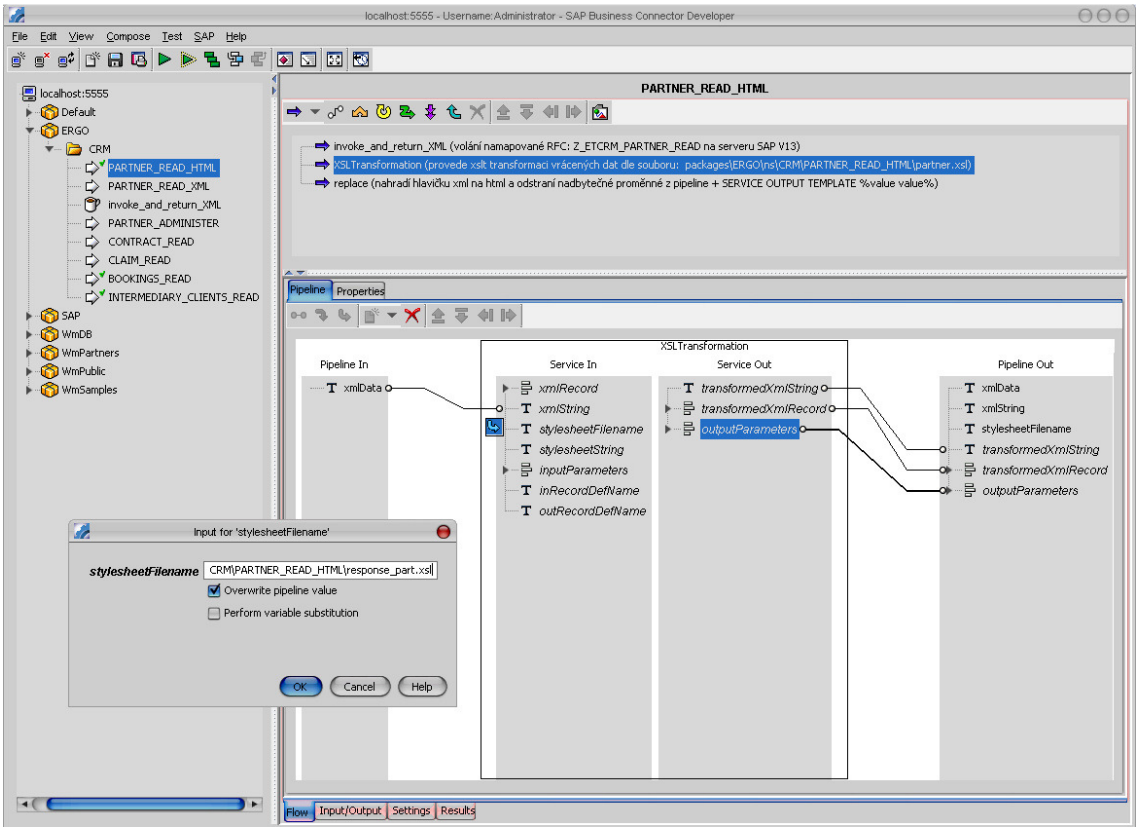
Obr. 23 Sekvence zpracování požadavku webové stránky na SAP RFC funkci, zdroj autor.

Příkladem implementace je případ užití „AdministrujeUzivatelskeUdaje“. V databázi SAP jsou uložena kmenová data v tabulce partnerů YEYPARO. Procedury aplikační logiky se budou odehrávat v prostředí SAP a na doplňkové webové technologii zůstanou jen úkoly prezentační vrstvy. Výběr zákaznických dat obstará funkční modul naprogramovaný v jazyku ABAP se zapnutou možností komunikace RFC (modul schopný remote). Zdrojový kód funkce v jazyce ABAP je k nahlédnutí v příloze II. Zde je uvedena část, popisující rozhraní volání.

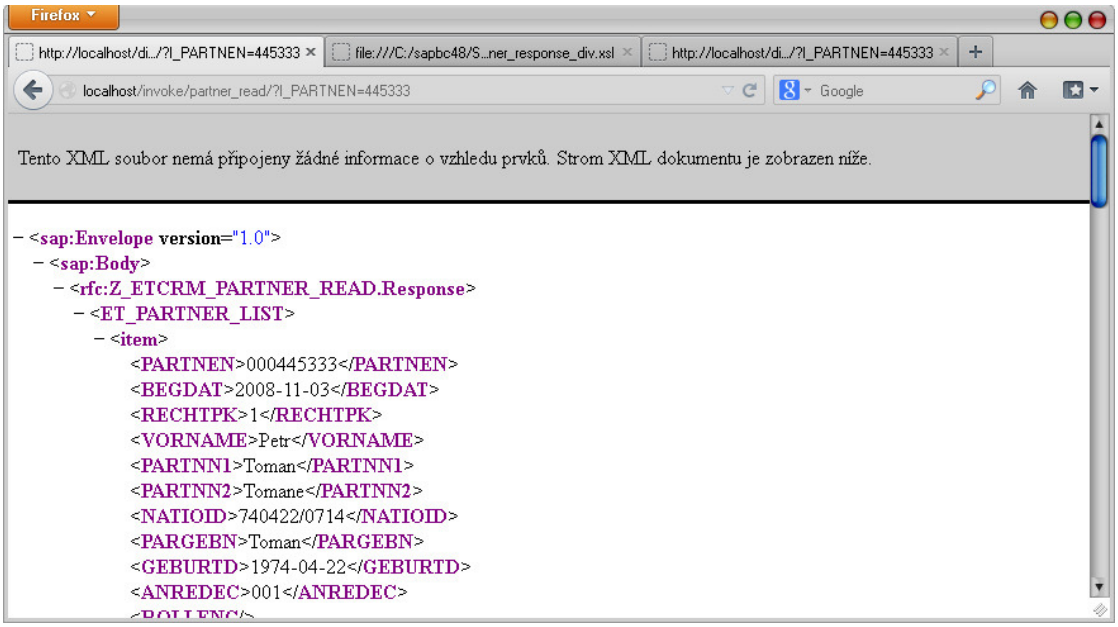
```
FUNCTION z_etcrm_partner_read.
*"----------------------------------------------------------------------
*"*"Lokální rozhraní:
*"  IMPORTING
*"     VALUE(I_PARTNEN) TYPE  /ALICE/PARTNEN
*"     VALUE(I_DATUM) TYPE  /ALICE/STICHDAT DEFAULT SY-DATUM
*"  EXPORTING
*"     VALUE(ET_ROLLE) TYPE  /ALICE/PADB_ST
*"     VALUE(ET_PARTNER_LIST) TYPE  /ALICE/RFC_PARTNER_RES_ST
*"     VALUE(E_ADRESS) TYPE  YEY011GRAF-ADRES70
*"  EXCEPTIONS
*"      NOTFOUND
*"----------------------------------------------------------------------
```

Zdrojový kód 2 Rozhraní funkce pro výběr dat zákazníka v jazyku ABAP, zdroj autor.

Dalším krokem je nastavení serveru SAP Business Connector tak, aby zprostředkoval předání vstupních a výstupních parametrů funkce. Realizovat to lze vytvořením služby, organizované v balíčku ERGO/CRM. Služba je pojmenována „PARTNER_READ_HTML“. Po přijetí požadavku HTTP s předáním parametrů metodou POST služba předá volání RFC SAP, přijme odpověď ve formátu XML a provede XSL transformaci. Výsledkem je úryvek HTML kódu, který může callback funkce AJAX včlenit do určeného oddílu zobrazené webové stránky. Ukázka zdrojového kódu volání AJAX je uvedena v příloze III.

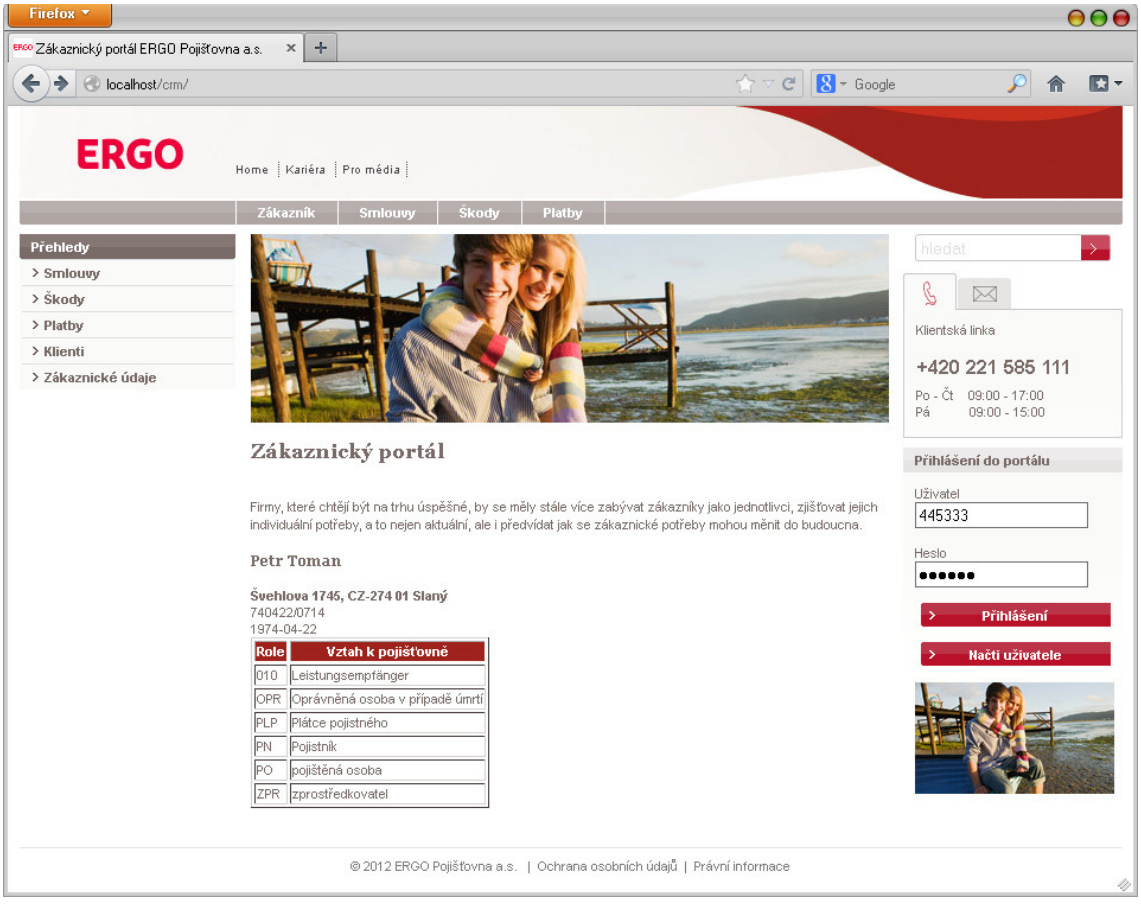


Obr. 24 Nastavení služby v uživatelském rozhraní SAP Business Connector Developer,snímek obrazovky, zdroj autor.



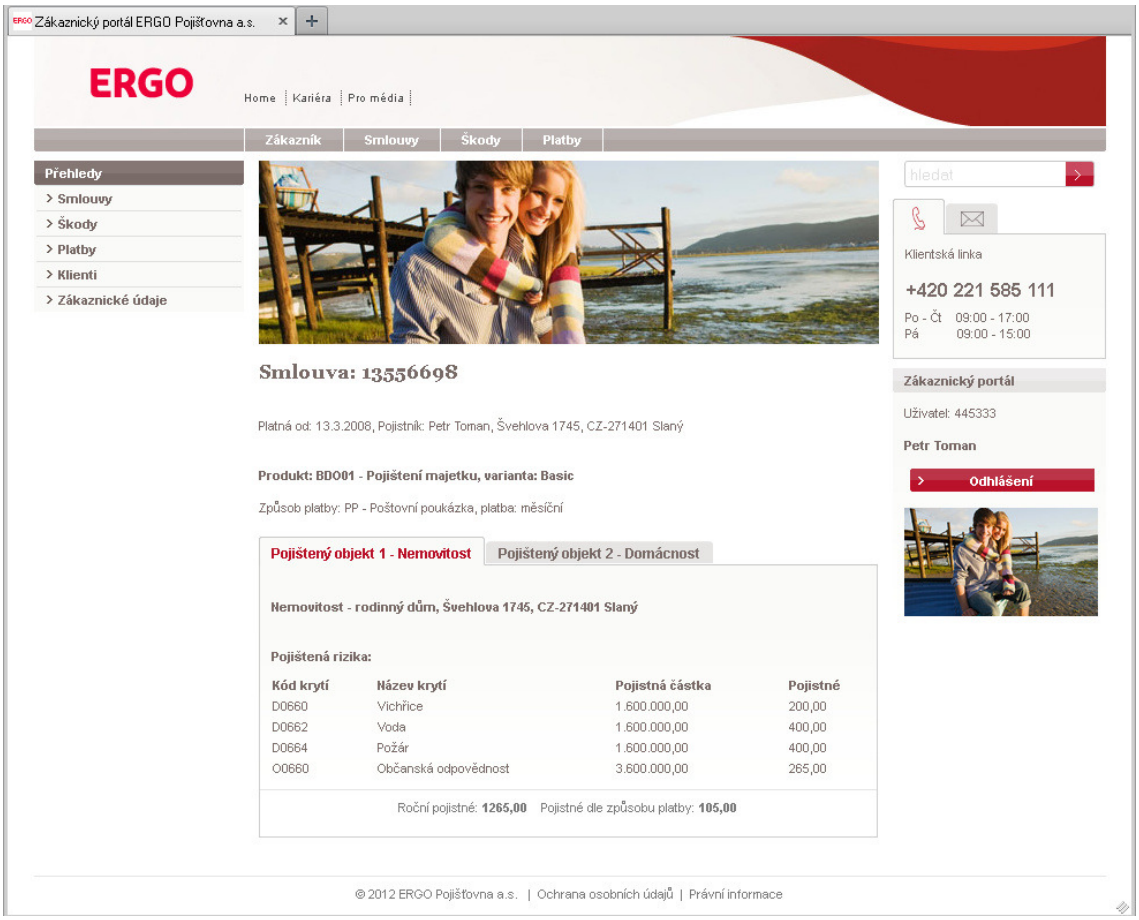
Obr. 25 Návratové hodnoty RFC funkce v XML, snímek obrazovky, zdroj autor.

.



Obr. 26 Odpověď vnořená do HTML kódu webupřipravovaného portálu pomocí technologie AJAX, zdroj autor.

Na obrázku 27 je ilustrace výstupu a grafického rozvržení jako výsledku případu užití „ZobrazitDetailSmlouvy“. Design stránky vychází z předepsaného nefunkčního požadavku na dodržení visuální identity korporace ERGO.



Obr. 27 Implementace a design, případ užití „ZobrazitDetailSmlouvy“, snímek obrazovky,, zdroj autor.

8. Závěr

CRM stojí rozkročen na pomezí strategického marketingu a informačních technologií. Netvoří ovšem hranici, ale oba tyto obory spojuje. Řízení vztahů se zákazníky ovšem nelze řešit pouze nasazením software. CRM vyžaduje celkovou změnu přístupu a firemní kultury. Funkční prototyp zákaznického portálu je příspěvkem, který poskytuje technologickou platformu jedné ze součástí řízení vztahů se zákazníky, totiž automatizované podpoře prodeje.

Portál otevírá nový komunikační kanál mezi firmou, zákazníky a obchodními partnery. Technické řešení prototypu je spíše než komplexním řešením studií proveditelnosti. Ukazuje směr. Zákaznický portál, založený na řešení popsaném v této práci, bude s velkou pravděpodobností realizován ve firmě ERGO pojišťovna, a.s. Předpokladem pro úspěšnou realizaci bude rozšířená analýza případů užití a propracování jednotlivých scénářů do větších detailů.

Přínos portálu lze spatřovat také v tom, že doplňuje služby, spojené s pojistnými produkty. Přidává tím hodnotu na 3. rozšířené vrstvě Kotlerova modelu produktu. Protože ve společnosti se 80% konkurence odehrává právě na této úrovni, může portál pomoci nejenom k řízení vztahů se zákazníky, ale i zlepšit konkurenceschopnost firmy.

V řešení, založeném na doplňující webové technologii a využívající dat a aplikační logiky systému SAP, lze spatřovat velký potenciál. Systémy SAP nabízejí vyzrálou a robustní technologii pro firmy, jsou dostatečně široce rozšířeny a jejich řešení obchodních a podnikových procesů se stalo de facto standardem. Internetové technologie, které zprostředkovávají komunikaci mezi prostředím SAP a uživateli portálu jsou všeobecně dostupné a nepředstavují žádnou technickou bariéru pro uživatele, kteří chtějí portál využívat.

Použití jazyku UML pro modelování portálu je velmi účinné ve fázi správy požadavků a ve vytváření případů užití. Při modelování procesů je ovšem poznat, že tento jazyk předjímá objektově orientované paradigma. Je proto nutné se vypořádat s impedančním nesouladem mezi objektovým přístupem a procedurami nad entitně relačními strukturami aplikační logiky SAP. Chování portálu, jehož hlavní činnost je

transformace dat, nemá při modelování interakcí jednotlivých objektů natolik dostatečnou dynamiku, aby bylo nutné využívat celého potenciálu tohoto jazyku.

Pořízení, údržba a správa systémů SAP není levnou záležitostí, a pokud se firma rozhodne provozovat své informační systémy právě na této platformě, musí věnovat zvýšenou pozornost tomu, aby se její investice zužitkovala. Implementovat software tak, aby byl v souladu s podnikovými procesy, je intelektuálně náročná činnost. Jedním z vedlejších přínosů této práce je propojení teoretických znalostí, získaných studiem a praktických zkušeností z vývoje v prostředí SAP.

Pro podniky není většinou problém, že by neměly dostatek dat, ale informace z nich složené bývají často nekvalitní, nebo není zájem o jejich využití. Zcela zásadní se tedy jeví schopnost data správně interpretovat a také záměr takto získané informace využívat. Činnost IT oddělení by se neměla soustředit pouze na poskytování technologické základny pro správu těchto dat, ale i na vytváření infrastruktury, která umožní smysluplný přístup k těmto datům a ulehčí uživatelům IT získávat z těchto dat relevantní informace. Jak s nadsázkou uvádí Lehtinen *„S tabulkou v Excelu nemůžeme vést, musíme s ní řídit.“* (Lehtinen, 2007)

9. Citovaná literatura

ANDERSON, G. W. 2012. *Naučte se SAP za 24 hodin.* Brno : Computer Press, a.s., 2012. ISBN 978-80-251-3685-0.

ARLOW, J. a NEUSTADT, I. 2011. *UML 2 a unifikovaný proces vývoje aplikací.* Brno : Computer Press, a.s., 2011. ISBN 978-80-251-1503-9.

BELLOF, R. 2006. Der schwere Weg vom BC zu SAP XI. *Computer Woche.* [Online] 13. 07 2006. [Citace: 03. 03 2013.] http://www.computerwoche.de/a/der-schwere-weg-vom-bc-zu-sap-xi,1215323.

BOSE, R. 2002. Customer relationship management: Key components for IT success. *Industrial Management & Data Systems.* 2002, Sv. 102, 2, stránky 89 - 97.

BUTTLE, F. 2009. *Customer relationship management: concepts and technologies.* 2nd ed. Burlington : Butterworth-Heinemann, 2009. ISBN 978-1-85617-522-7.

CUMMINGS, ROBERT. 2011. SAP for insurance in 7 minutes. *Insurance Industry Experts.* [Online] 10. 10 2011. [Citace: 13. 03 2013.] http://www54.sap.com/industries/insurance/experts.html.

DE ARGAEZ, E. 2013. World Internet Users Statistics Usage and World Population Stats. *Internet World Stats.* [Online] 17. 02 2013. [Citace: 15. 03 2013.] http://www.internetworldstats.com/stats.htm.

ERGO POJIŠŤOVNA, A.S. 2009. *Cestovní pojištění online.* [Online] ERGO Pojišťovna, a.s, 20. 07 2009. [Citace: 10. 03 2013.] https://online.ergo.cz.

FIELDING, R., a další. 1999. Hypertext Transfer Protocol -- HTTP/1.1. [Online] 03. 03 1999. [Citace: 03. 12 2013.] http://www.ietf.org/rfc/rfc2616.txt.

GARTNER GROUP, INC. 2012. Customer Relationship Management (CRM) Definition. *Technology Resarch.* [Online] 2012. [Citace: 12. 02 2013.] http://www.gartner.com/it-glossary/customer-relationship-management-crm/.

GLAZER, R. 1997. Strategy and structure in information-intensive markets: the relationship between marketing and IT. *Journal of Market Focused Management.* Září 1997, Sv. 2, 1, stránky 65-81.

GOLDENBERG, B. J. 2008. *CRM in Real TIme: Empowering Customer Relationship.* Medford : Information Today, Inc., 2008. ISSN 978-0-910965-80-4.

HOBY, J. 1997. Looking after the one who matters. *Accountancy Age.* 1997, Sv. October, 28, stránky 28-30.

CHLEBOVSKÝ, V. 2005. *CRM Řízení vztahů se zákazníky.* Brno : Computer Press, a.s., 2005. ISBN 80-251-0798-1.

JQUERY FOUNDATION . 2012. jQuery: The Write Less, Do More, JavaScript Library. [Online] jQuery Foundation , 16. 06 2012. [Citace: 06. 02 2013.] http://jquery.com/.

KANISOVÁ, H. a MÜLLER, M. 2006. *UML srozumitelně.* Brno : Computer Press, a.s., 2006. ISBN 80-251-1038-4.

KELLER, H. 2005. *Official ABAP Refference.* Bonn : Galileo Press, 2005. ISBN 1-59229-039-6.

KINCAID, J. W. 2003. *Custumer Relationship Management: Get It Right.* Upper Saddle River : Hewlett Packard Books, 2003. ISBN 0-13-035211-X.

KOTLER, P. 2007. *Marketing management.* Praha : Grada, 2007. ISBN 978-80-247-1359-5.

KOTLER, P. 2007. *Moderní marketing.* Praha : Grada, 2007. ISBN 978-80-247-1545-2.

KUTNER, S. a CRIPPS, J. 1997. Managing the customer portfolio of healthcare enterprises. *The Healthcare Forum Journal.* 1997, Sv. 4, Sept-Oct, stránky 52-54.

LEHTINEN, J. 2007. *Aktivní CRM, Řizení vztahů se zákazníky.* Praha : Grada Publishing, 2007. ISBN 978-80-247-1814-9.

LING, R. a YEN, D. C. 2001. Customer relationship management: An analysis framework and implementation strategies. *Journal of Computer Information Systems.* 2001, stránky 82-97.

MAASEN, A. 2007. *SAP R/3, Kompletní průvodce.* Brno : Computer Press, a. s., 2007. ISBN 978-80-251-1750-7.

MCCARTHY, E. J. 1995. *Základy marketingu.* Praha : Victoria Publishing, 1995. ISBN 80-856-0529-5.

MERUNKA, V. 2008. *Objektové modelování.* Praha : Alfa Nakladatelství, s.r.o., 2008. ISBN 978-80-871974-04-02.

PARVATIYAR, A., SHETH, J. N. a SHAINESH, G. 2001. *Customer relationship management: emerging concepts, tools, and applications.* New Delhi : Tata McGraw Hill Publishing Company, 2001. stránky 3-25. ISBN: 9780070435049.

PHP GROUP. 2013. PHP: Hypertext Preprocessor. [Online] The PHP Group, 13. 03 2013. [Citace: 13. 03 2013.] http://php.net/.

SAP AG. 2012. Components of SAP Communication Technology. *SAP Library.* [Online] SAP AG., 14. 12 2012. [Citace: 15. 03 2013.] http://help.sap.com/saphelp_nw04/helpdata/en/36/020d3a0154b909e10000000a114084/frameset.htm.

SAP AG. 2009. Changing the SAP Standards. *SAP Library.* [Online] 17. 12 2009. [Citace: 03. 03 2013.] http://help.sap.com/saphelp_nw04/helpdata/en/bf/ec07a25db911d295ae0000e82de14a/frameset.htm.

SAP AG. 2010. Open SQL. *SAP Library.* [Online] SAP AG., 17. 12 2010. [Citace: 10. 03 2013.] http://help.sap.com/saphelp_nw04/helpdata/EN/fc/eb3969358411d1829f0000e829fbfe/content.htm.

SAP AG. 2013. SAP help portal. *Central place fo SAP documentation.* [Online] SAP AG, 03. 03 2013. [Citace: 13. 03 2013.] http://help.sap.com/.

SAP ČESKÁ REPUBLIKA. 2012. SAP Business Suite: řešení pro jakokoliv podnik a jakýkoliv proces. *Sap Česká republika.* [Online] 10. 10 2012. [Citace: 11. 03 2013.] http://www.sap.com/cz/solutions/business-suite/index.epx.

TONADE, D. 2010. SAP Community Network Wiki - FS-CD. *SAP Community Network.* [Online] 18. 10 2010. [Citace: 11. 03 2013.] http://wiki.sdn.sap.com/wiki/display/HOME/FS-CD.

W3SCHOOLS. 2010. AJAX Tutorial. *W3Schools Online Web Tutorials.* [Online] W3Schools, 12. 12 2010. [Citace: 15. 03 2013.] http://www.w3schools.com/ajax/.

WEINRICH, U. 2007. *ALICE Anwenderhandbuch.* [CD ROM] Düsseldorf : ALICE Software Service GmbH, 2007.

WILLIAMS, E. H. a LANE, D. 2002. *Programujeme webové aplikace pomocí PHP a MySQL.* Praha : Computer Press, a.s., 2002. ISBN 80-7226-760-4.

WITTEBROCK, T. 2006. SAPs Business Connector migrieren: Wie sich der XML-Datenaustausch auf Netweaver XI umstellen lässt. *Computerwoche.* [Online] 23. 03 2006. [Citace: 10. 03 2013.] http://www.computerwoche.de/a/saps-business-connector-migrieren,1214015.

10. Seznam Obrázků

Příloha I: scénáře případů užití zákaznického portálu.

Případ užití	PřihlášeníDoSystému
ID	1
Stručný popis	Přihlášení uživatele do systému
Hlavní aktéři	Uživatel
Vedlejší aktéři	žádný
Vstupní podmínky	Uživatel není přihlášen Uživatel není blokován
Hlavní scénář	1. Uživatel: zadá jméno a heslo. 2. Systém: validuje jméno a heslo 3. Systém: ověří jméno a heslo 4. Uživatel: je přihlášen 5. Systém zobrazí uživatelská data
Výstupní podmínky	Uživatel je přihlášen
Alternativní scénáře	NeplatnéJménoHeslo

Alternativní scénář	PřihlášeníDoSystému:NeplatnéJménoHeslo
ID	1.I
Stručný popis	Systém informuje zákazníka, že zadal neplatné uživatelské údaje
Hlavní aktéři	Uživatel
Vedlejší aktéři	žádný
Vstupní podmínky	Uživatel není přihlášen Uživatel zadal neplatné jméno/heslo
Hlavní scénář	1. Začíná krokem 3. hlavního scénáře 2. Systém: informuje uživatele, že zadal neplatné jméno nebo heslo
Výstupní podmínky	Uživatel není přihlášen
Alternativní scénáře	Žádné

Případ užití	OdhlášeníZeSystému
ID	2
Stručný popis	Odhlášení uživatele ze systému
Hlavní aktéři	Uživatel
Vedlejší aktéři	žádný
Vstupní podmínky	Uživatel je přihlášen
Hlavní scénář	1. Uživatel: zvolí odhlášení 2. Systém: zneplatní přihlašovací sezení 3. Systém: zobrazí přihlašovací stránku 4. Uživatel: je odhlášen
Výstupní podmínky	Uživatel je odhlášen
Alternativní scénáře	žádné

Případ užití	RegistujeSe
ID	3
Stručný popis	Registrace nového uživatele
Hlavní aktéři	Návštěvník
Vedlejší aktéři	Uživatel
Vstupní podmínky	Uživatel není přihlášen Návštěvník není registrován
Hlavní scénář	1. Návštěvník zvolí registraci 2. Systém: zobrazí registrační formulář 3. Návštěvník: zadá jméno, příjmení a budoucí heslo, rodné číslo a číslo některé ze svých smluv 4. Systém: validuje úplnost a formální náležitosti zadaných udajů 5. Systém: ověří jméno a rodné číslo v db zákazníků 6. Systém: ověří, jestli existuje smlouva a patří zákazníkovi 8. Systém: zkontroluje, jestli účet už není založen 7. Systém: vytvoří záznam o registraci 8. Systém: zobrazí výsledek registrace 9. Návštěvník: je zaregistrován
Výstupní podmínky	Návštěvník je zaregistrován
Alternativní scénáře	NevalidniRegistracniUdaje NeplatnéRegistračníUdaje UcetJizExistuje

Alternativní scénář	RegistujeSe:NevalidniRegistracniUdaje
ID	3.I
Stručný popis	Systém informuje Návštěvníka, že zadal nevalidní údaje k registraci
Hlavní aktéři	Návštěvník
Vedlejší aktéři	Uživatel
Vstupní podmínky	Uživatel není přihlášen Návštěvník není registrován Návštěvník zadal nevyhovující registrační údaje
Hlavní scénář	1. Začíná krokem 4. v hlavním scénáři 2. Systém: informuje uživatele, že zadal nevalidní registrační údaje
Výstupní podmínky	Návštěvník není zaregistrován
Alternativní scénáře	Žádné

Alternativní scénář	RegistujeSe:NeplatnéRegistračníUdaje
ID	3.II
Stručný popis	Systém informuje Návštěvníka, že zadal neplatné údaje k registraci
Hlavní aktéři	Návštěvník
Vedlejší aktéři	Uživatel
Vstupní podmínky	Uživatel není přihlášen Návštěvník není registrován Návštěvník zadal nevyhovující registrační údaje
Hlavní scénář	1. Začíná krokem 6. v hlavním scénáři 2. Systém: informuje uživatele, že zadal neplatné registrační údaje
Výstupní podmínky	Návštěvník není zaregistrován
Alternativní scénáře	Žádné

Alternativní scénář	RegistujeSe:UcetJizExistuje
ID	3.III
Stručný popis	Systém informuje Návštěvníka, že zadal neplatné údaje k registraci
Hlavní aktéři	Návštěvník
Vedlejší aktéři	Uživatel
Vstupní podmínky	Uživatel není přihlášen Návštěvník není registrován Návštěvník zadal registrační údaje k účtu, který již existuje
Hlavní scénář	1. Začíná krokem 7. v hlavním scénáři 2. Systém: informuje uživatele, že registrace již proběhla
Výstupní podmínky	Návštěvník není zaregistrován
Alternativní scénáře	Žádné

Případ užití	ZobrazitDetailSmlouvy
ID	4
Stručný popis	Uživatel čte detail zvolené smlouvy
Hlavní aktéři	Zákazník, Obchodní Zástupce
Vedlejší aktéři	žádný
Vstupní podmínky	Aktér je přihlášen Požadovaná smlouva patří do jeho seznamu Je zobrazen seznam smluv
Hlavní scénář	1. Aktér: vybere číslo smlouvy ze seznamu 2. Systém: načte smlouvu 3. Systém: zobrazí data smlouvy extend:ZobrazitPlatbyFakturySmlouvy
Výstupní podmínky	Žádné
Alternativní scénáře	Žádné

Případ užití/Extend	ZobrazitPlatbyFakturySmlouvy
ID	4.I
Stručný popis	Uživatel čte platby zvolené smlouvy
Hlavní aktéři	Zákazník, Obchodní Zástupce
Vedlejší aktéři	žádný
Vstupní podmínky	Aktér je přihlášen Smlouva patří do jeho seznamu Systém zobrazuje detail smlouvy
Hlavní scénář	1. Aktér: vybere příkaz Zobrazit Platby 2. Systém: načte Platby smlouvy 3. Systém: zobrazí data plateb smlouvy
Výstupní podmínky	Žádné
Alternativní scénáře	Žádné

Případ užití	ZobrazitSeznamPlatebFaktur
ID	5
Stručný popis	Zobrazí přehled faktur a přijatých plateb, patřících konkrétnímu zákazníkovi
Hlavní aktéři	Zákazník
Vedlejší aktéři	žádný
Vstupní podmínky	Zákazník je přihlášen
Hlavní scénář	1. Uživatel: zvolí časové období 2. Systém: vybere seznam odpovídajících záznamů faktur a plateb zákazníka 3. Systém: zobrazí faktury a platby
Výstupní podmínky	žádné
Alternativní scénáře	žádné

Případ užití	ZobrazitSeznamSmluv
ID	6
Stručný popis	Zobrazí přehled smluv
Hlavní aktéři	Zákazník, Obchodní zástupce
Vedlejší aktéři	žádný
Vstupní podmínky	Hlavní aktér je přihlášen
Hlavní scénář	1. Hlavní aktér: zvolí příkaz seznam smluv 2. Systém: vybere seznam odpovídajících záznamů smluv 3. Systém: zobrazí seznam smluv
Výstupní podmínky	žádné
Alternativní scénáře	žádné

Případ užití	ZobrazitSeznamSmluvVlastních
ID	6.I, ID předka 6
Stručný popis	Zobrazí přehled smluv, patřících konkrétnímu zákazníkovi
Hlavní aktéři	Zákazník
Vedlejší aktéři	žádný
Vstupní podmínky	Zákazník je přihlášen
Hlavní scénář	1. Zákazník: zvolí příkaz seznam smluv 2.(o.2)Systém: vybere seznam odpovídajících záznamů smluv, které patří zákazníkovi 3. Systém: zobrazí seznam smluv
Výstupní podmínky	žádné
Alternativní scénáře	žádné

Případ užití	ZobrazitSeznamSmluvKlientu
ID	6.II, ID předka 6
Stručný popis	Zobrazí přehled smluv, které pojistil Obchodní zástupce
Hlavní aktéři	Obchodní zástupce
Vedlejší aktéři	žádný
Vstupní podmínky	Obchodní zástupce je přihlášen
Hlavní scénář	1. Obchodní zástupce: zvolí příkaz seznam smluv 2.(o.2)Systém: vybere seznam záznamů smluv, které patří klientům obchodního zástupce 3. Systém: zobrazí seznam smluv
Výstupní podmínky	žádné
Alternativní scénáře	žádné

Případ užití	ZobrazitProvize
ID	7
Stručný popis	Zobrazí seznam účetních dokladů provizí
Hlavní aktéři	Obchodní zástupce
Vedlejší aktéři	žádný
Vstupní podmínky	Obchodní zástupce je přihlášen
Hlavní scénář	1. Obchodní zástupce: zvolí příkaz provize 2. Obchodní zástupce: zadá období od - do 3. Systém: vybere seznam odpovídajících záznamů účetních dokladů provizí 4. Systém: zobrazí seznam provizí
Výstupní podmínky	žádné
Alternativní scénáře	žádné

Případ užití	AdministrujeUzivatelskeUdaje
ID	8
Stručný popis	Uživatel zobrazuje a upravuje svoje adresu, email a telefon a heslo
Hlavní aktéři	Uživatel
Vedlejší aktéři	žádný
Vstupní podmínky	Uživatel je přihlášen
Hlavní scénář	1. Uživatel: zvolí příkaz uživatelské údaje 2. Systém: zobrazí uživatelské údaje přihlášeného uživatele 3. Uživatel: změní uživatelské údaje 4. KDYŽ Uživatel: potvrdí změnu uživatelských údajů 4.1. Systém: validuje úplnost a formální náležitosti zadaných údajů 4.2. Systém: aktualizuje záznamy uživatelských údajů 4.3 Systém: zobrazí uživatelské údaje
Výstupní podmínky	žádné
Alternativní scénáře	žádné

Případ užití	BlokujeUzivatele
ID	9.I
Stručný popis	Administrátor blokuje uživatelský účet
Hlavní aktéři	Administrátor
Vedlejší aktéři	Uživatel
Vstupní podmínky	Administrátor je přihlášen
Hlavní scénář	1. Administrátor: zvolí příkaz správa účtů 2. Systém: zobrazí seznam uživatelů 3. Administrátor: vybere uživatelský účet ze seznamu 4. Systém: zobrazí záznam vybraného uživatele 5. Administrátor: vybere příkaz blokovat uživatele 6. Systém: aktualizuje záznamy uživatelských údajů 7. Systém: zašle zprávu uživateli 8. Systém: zobrazí záznam vybraného uživatele
Výstupní podmínky	Uživatel je zablokován
Alternativní scénáře	žádné

Případ užití	AktivujeUzivatele
ID	9.II
Stručný popis	Administrátor aktivuje blokovaný uživatelský účet
Hlavní aktéři	Administrátor
Vedlejší aktéři	Uživatel
Vstupní podmínky	Administrátor je přihlášen
Hlavní scénář	1. Administrátor: zvolí příkaz správa účtů 2. Systém: zobrazí seznam uživatelů 3. Administrátor: vybere uživatelský účet ze seznamu 4. Systém: zobrazí záznam vybraného uživatele 5. Administrátor: vybere příkaz aktivovat uživatele 6. Systém: aktualizuje záznamy uživatelských údajů 7. Systém: zašle zprávu uživateli 8. Systém: zobrazí záznam vybraného uživatele
Výstupní podmínky	Uživatel je aktivován
Alternativní scénáře	žádné

Příloha II: zdrojový kód v jazyce ABAP pro výběr dat zákazníka.

```
FUNCTION z_etcrm_partner_read.
*"----------------------------------------------------------------------
----
*"*"Lokální rozhraní:
*"  IMPORTING
*"     VALUE(I_PARTNEN) TYPE  /ALICE/PARTNEN
*"     VALUE(I_DATUM) TYPE  /ALICE/STICHDAT DEFAULT SY-DATUM
*"  EXPORTING
*"     VALUE(ET_ROLLE) TYPE  /ALICE/PADB_ST
*"     VALUE(ET_PARTNER_LIST) TYPE  /ALICE/RFC_PARTNER_RES_ST
*"     VALUE(E_ADRESS) TYPE  YEY011GRAF-ADRES70
*"  EXCEPTIONS
*"      NOTFOUND
*"----------------------------------------------------------------------
----
  DATA: wa_pars          TYPE          /alice/rfc_pars,
        wa_partner_list  LIKE LINE OF  et_partner_list,
        wa_rolle         LIKE LINE OF  et_rolle,
        lt_padb          LIKE          et_rolle,
        wa_padb          LIKE LINE OF  et_rolle,
        lt_prlo          TYPE          /alice/prlo_st,
        wa_prlo          LIKE LINE OF  lt_prlo.

*inicializace prázdného vstupního parametru datum na aktuální datum
  IF i_datum IS INITIAL.
     i_datum = sy-datum.
  ENDIF.

  wa_pars-partnen = i_partnen.

*volání funkce aplikační logiky pro selekci partnera
  CALL FUNCTION '/ALICE/RFC_PARTNER_SELECT'
    EXPORTING
      i_pars          = wa_pars
    TABLES
      et_partner_list = et_partner_list.


*výstup je typu interní tabulka, obsahuje 1 rádek
*načtením si ověříme, jestli se podařilo nalézt záznam
*pokud ne funkce se ukončí vyvoláním vyjímky
  READ TABLE et_partner_list INTO wa_partner_list INDEX 1.
  IF sy-subrc <> 0.                                    "máme výstup?
    RAISE notfound.
  ENDIF.
```

```
*** vyhledáme a sestavíme adresu partnera, zformátovanou do řetězce
  CALL FUNCTION '/ALICE/PR_GET_ADRESSENSTRING'
    EXPORTING
      adressnummer = wa_partner_list-adressn
    IMPORTING
      address_long = e_adress
    EXCEPTIONS
      not_found    = 1
      OTHERS       = 2.


***načtení rolí partnera (pojistník, pojištěná osoba, zprostředkovatel..
  SELECT * FROM yeyprlo
           INTO TABLE lt_prlo
           WHERE partnen = i_partnen.

***načtení textů rolí
  IF lt_prlo[] IS NOT INITIAL.
    SELECT * FROM  yeypadb
              INTO TABLE lt_padb
              FOR ALL ENTRIES IN lt_prlo
              WHERE rollenc = lt_prlo-rollenc.
  ENDIF.

*doplnění textů rolí partnera v odpovídajícím jazyku, nebo defaultním
  LOOP AT lt_prlo INTO wa_prlo.
    MOVE-CORRESPONDING wa_prlo TO wa_padb.
    READ TABLE lt_padb         INTO wa_padb
         WITH KEY rollenc =  wa_prlo-rollenc
                  spras   =  wa_partner_list-sprachc.
    IF sy-subrc <> 0.
      READ TABLE lt_padb  INTO wa_padb
      WITH KEY rollenc =  wa_prlo-rollenc.
    ENDIF.
    APPEND wa_padb TO et_rolle.
  ENDLOOP.


ENDFUNCTION.
```

Příloha III: zdrojový kód asynchronní volání AJAX pro aktualizaci stránky

```
function AjaxRetrieve(src_element, target_element, rfc_request) {
//do elementu target_element vloží výsledek požadavku rfc_request

var newdiv = document.createElement("div");
var response_container = document.getElementById(target_element);
response_container.appendChild(newdiv);
var xmlhttp;

if (window.XMLHttpRequest)   {  xmlhttp=new XMLHttpRequest();  }
else      { xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  }

xmlhttp.onreadystatechange=function()   //asynchronně, definujeme
callback
  {

  if (xmlhttp.readyState=='4' && xmlhttp.status=='200') {
           response_container.innerHTML = xmlhttp.responseText;
     }
  else if (xmlhttp.readyState=='0' ) {
          response_container.innerHTML = "Nepřipravený
požadavek...";   }
  else if (xmlhttp.readyState=='1' ) {
          response_container.innerHTML = "Připojeno...";   }
  else if (xmlhttp.readyState=='2' ) {
          response_container.innerHTML = "Požadavek přijat..." ;}
  else if (xmlhttp.readyState=='3' ) {
          response_container.innerHTML = "Zpracování požadavku...";
}
  else if (xmlhttp.readyState=='4' && xmlhttp.status!='200') {
          response_container.innerHTML = "Neplatný požadavek...";
}
  }

xmlhttp.open("POST", rfc_request, true);
xmlhttp.send();
}
```