Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Jiří Kulhánek

Integrace heterogenních informačních systémů

Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Filip Zavoral, Ph.D.
Studijní program: Informatika
Studijní plán: Softwarové inženýrství
Studijní program: Softwarové systémy

Poděkování

Rád bych na tomto místě poděkoval především vedoucímu své diplomové práce RNDr. Filipu Zavoralovi, Ph.D. za cenné rady, návrhy a připomínky, jež významně přispěly k vylepšení konečné podoby textu.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 21. července 2006 Jiří Kulhánek

Obsah

Název práce: Integrace heterogenních informačních systémů
Autor: Jiří Kulhánek
Katedra: Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Filip Zavoral, Ph.D.
E-mail vedoucího: filip.zavoral@mff.cuni.cz
Abstrakt: Informační systémy se stávají nedílnou součástí dnešního světa. S jejich významem vyvstává zároveň potřeba integrace, která by umožnila jejich vzájemnou spolupráci a rozšířila tak možnosti a kvalitu jejich použití. Takováto integrace s sebou ovšem nese řadu obtíží a překážek, kterým musí architekti těchto nových a mnohdy neprobádaných řešení čelit. V této práci je daná problematika shrnuta, jsou uvedeny základní koncepty možných přístupů a jsou adresovány konkrétní problémy spojené s integrací spolu se způsoby jejich řešení. Díky tomu pak může být podrobně analyzován předně projekt Integrace informačních systémů Univerzity Karlovy se svojí architekturou Datového Stohu a dále pak projekt Jednotné sítě italské administrativy jako příklad kooperačního informačního systému. Tato dvě integrační řešení z praxe jsou názornou ukázkou aktuálních problémů, se kterým se v současnosti na tomto poli potýkáme.
Klíčová slova: Integrace systémů, Integrace Dat, Interoperabilita, Heterogenní informační systémy, Případová studie.

Title: Integration of Heterogeneous Information Systems
Author: Jiří Kulhánek
Department: Department of Software Engineering
Supervisor: RNDr. Filip Zavoral, Ph.D.
Supervisor's e-mail: filip.zavoral@mff.cuni.cz
Abstract: Information systems are becoming an inseparable part of today's world. Together with their significance arise an emergency to integrate them to enable their mutual cooperation and so extend the facility and quality of their use. Nevertheless, this integration is carrying a number of difficulties and obstacles, which are needed to be dealt with by architects of these new and often unexplored solutions. In this work the given problematic is summarized, the basic concepts of possible approaches are introduced and the concrete challenge connected to the integration are aimed together with the ways how to solve them. Thanks to this, the project Integration of Information Systems of Charles University with its architecture of Data-Pile structure and the project Unitary Network of Italian Government as an example of Cooperating Information System can be analyzed in detail. These two integrating solutions from practice are illustrative examples of actual problems in this field, which we face in this day and age.
Keywords: System integration, Data integration, Interoperability, Heterogeneous information systems, Case study.

1 Úvod

V praxi nezřídka nastává situace, kdy se v rozsáhlé organizaci postupem času vyvine více informačních systémů (IS), které se různě profilují a navzájem jsou značně heterogenní. Vyvstane-li pak nutnost tyto heterogenní systémy integrovat, je architektura použitého řešení netriviálním problémem s celou řadou obtíží a překážek.

Oblast integrace systémů je již po dlouhou dobu předmětem výzkumu z mnoha různých směrů a úhlů pohledu a objevilo se množství metod a rozdílných přístupů, jak čelit některým problémům, jež jsou pro tyto integrační řešení typické; zároveň s tím ale zůstávají další aspekty stále otevřeny a především víceméně neexistuje obecný ucelený přehled problematiky. Jsme tak svědky situace, kdy vzniklá potřeba integrovaných spolupracujících IS vede ke vzniku množství ad-hoc řešení. Cílem této práce je podat nástin této oblasti a uvést a zhodnotit některé z takto vzniklých řešení.

Nyní nejprve podrobněji charakterizujeme problém integrace heterogenních IS a následně v kapitole 2 podáme přehled teoretického základu problematiky a provedeme základní rozdělení jednotlivých integračních metod a přístupů. V další kapitole pak navazujeme rozvedením některých konkrétních integračních problémů se současným nástinem jejich možných řešení. V kapitolách 4 a 5 je pak provedena podrobná analýza dvou konkrétních integračních řešení z praxe a v návaznosti na to jsou v kapitole 7 analyzovaná řešení krátce shrnuta.

1.1 Charakteristika heterogenních IS a jejich integrace

Díváme-li se na problém integrace informačních systémů, můžeme odlišit jeho rozdílné úrovně, které jsou dány strukturou a funkcí IS. Ten představuje především jakousi platformu zajišťující správu dat a přístup k nim, bez ohledu na to, jak a kde jsou tato data uložena. Integrace IS pak tedy může být chápána s ohledem na integraci samotných dat nebo s ohledem na zpřístupnění služeb operujících nad těmito daty nebo s důrazem na zajištění celkové spolupráce integrovaných IS.

1.1.1 Vývoj pohledu na integraci

Přestože tedy IS představuje především služby nad daty a ne data samotná, důležitým a v minulosti zřejmě nejvíce studovaným tématem je zde právě integrace dat. To je dáno faktem, že IS je v této problematice brán jako zobecnění databázových systémů, na které byla v minulosti pozornost zaměřena. Nejprve v 80. letech bylo hlavním předmětem zájmu shromažďování dat ze samostatných aplikací do centrálního úložiště, v 90. letech pak vývoj směřoval k možnosti kombinovat data z různých DBMS [Bus99]. Původní tzv. *vícenásobné databázové systémy* se pak rozvinuly do tzv. *sjednocených databázových systémů*, kde se hlavně práce [She90] stala východiskem pro další vývoj a zavedla názvosloví, přičemž definovala klasickou 5-ti vrstvou architekturu takových systémů. S mohutným rozšířením IS pak vznikla potřeba zavedené postupy rozšířit o prostředky umožňující spolupráci na úrovni služeb, čehož je názornou ukázkou práce Wiederholda [Wie92], který prezentoval svůj pohled na prostředí s řadou malých nezávislých komponent, tzv. „*mediátorů*“, které mají oddělovat data od uživatelských aplikací, poskytovat rozmanitou paletu služeb a vzájemně se doplňovat. Řada idejí tohoto modelu pak ovlivnila i další řešení v této oblasti. V poslední době pak s nástupem Internetu vyvstává potřeba integrace nekontrolovatelných webových zdrojů. Pro svou specifičnost ovšem tato oblast není předmětem zájmu této práce. K historickému vývoji nakonec uveďme, že v minulosti byla často podstatnou stránkou otázka fyzického umístění integrovaných zdrojů a podobné aspekty technického rázu. Jejich důležitost ovšem značně poklesla vzhledem k pokroku učiněnému na tomto poli, kdy jsou s výhodou používány distribuované technologie typu CORBA a přenosové protokoly jako rozšířené HTTP.

Jak vývoj ovlivňoval pohled na architekturu IS i softwarových systémů obecně, postupně bylo upouštěno od velkých monolitických centralizovaných aplikací a místo toho jsou systémy budovány jako soustavy spolupracujících komponent. Architektura klient-server je nahrazována více-vrstevnatými systémy používajícími řadu technologií a různých protokolů. Typický IS tak má často podobnou strukturu, jaká je načrtnuta v diagramu 1. Zde je přístup k datům uložených v databázi umožněn nativními rozhraními a často řízen aplikačním serverem. Rozhraní pro klientské aplikace je pak vytvářeno například CORBA nebo WWW serverem a značné množství klientských aplikací je provozováno na úrovni webového

prohlížeče. Právě IS s podobnou charakteristikou pak mnohdy představují systémy určené ke vzájemné integraci.

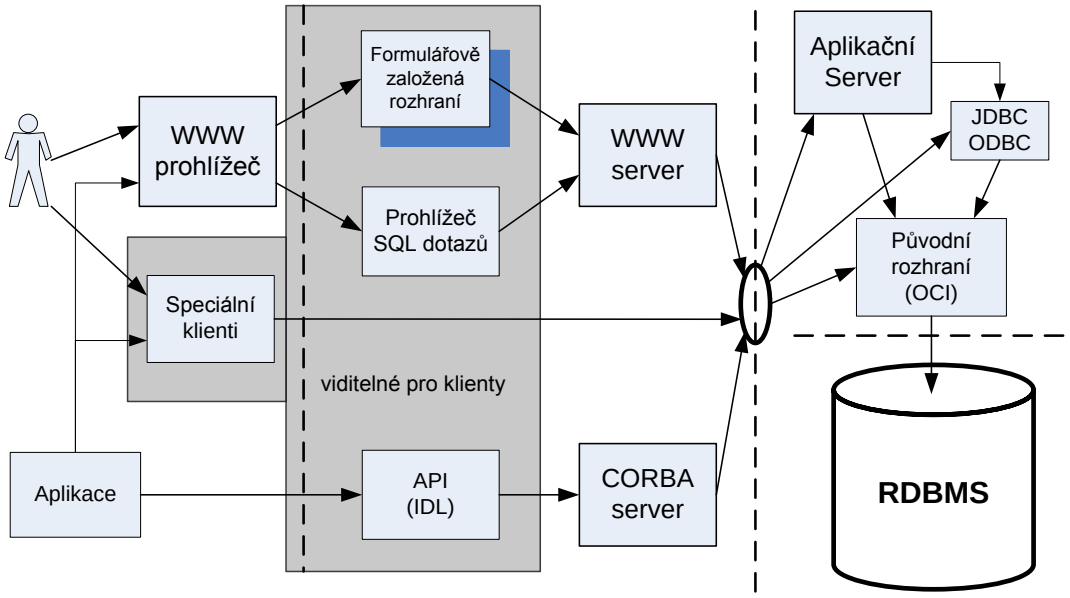


Diagram 1: Typický stav přístupu k datům v současných distribuovaných aplikačních prostředích [Bus99].
Čárkovanou čarou jsou naznačeny obvyklé přechody mezi fyzickým umístěním.

Výsledná podoba integračního řešení pak může být charakterizována jako tzv. *federativní* neboli *sjednocený informační systém* (federated IS; FIS) definovaný v [Bus99], jehož architektura je naznačena v diagramu 2.

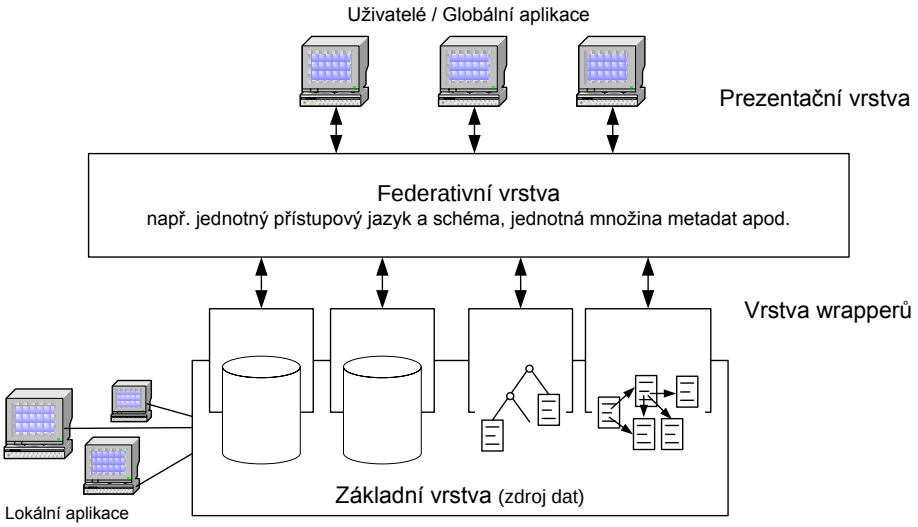


Diagram 2: Federativní informační systém dle [Bus99].

Zde je FIS definován jako soustava odlišných autonomních komponent informačních systémů, které se účastní spolupráce. Poznamenejme, že heterogenita zde není přímo požadována, ale je charakteristickým znakem. V diagramu vidíme typickou trojvrstvou architekturu FIS, kde uživatelské aplikace přistupují k heterogenním zdrojům prostřednictvím tzv. *federativní vrstvy*. Ta je komponentou systému poskytující jednotný způsob přístupu k datům, k čemuž může používat různých prostředků jako sjednocené schéma, jednotný dotazovací jazyk apod. Technické rozdíly v přístupu ke zdrojům pak jsou obvykle řešeny pomocí tzv. „*wrapperů*".

1.1.2 Zavedení integračního řešení

Integrace IS s sebou nese ovlivnění stávajících systémů. Tzv. „*legacy systems*" jsou systémy původního řešení, které je třeba začlenit do nové integrující architektury v jejich stávající podobě, protože rozsáhlejší zásah do nich již není možný a jejich nahrazení je taktéž nežádoucí.

Způsob, jakým může být proveden přechod původních aplikací na cílový model integračního řešení je znázorněn v diagramu 3. Zatímco při *migraci* (a) je vliv původního systému postupně potlačován a v určitou chvíli dojde k prohození rolí, ve druhém případě *začlenění* (b) je provedena skutečná integrace a původní systém je upraven tak, aby mohl být zapojen do nového modelu. Začlenění systémů se zdá být variantou, která by měla být preferována pro svůj menší dopad na uživatele a vynucené změny v organizaci, ovšem v některých případech, které jsou dány především nevyhovujícími a zastaralými původními systémy, může být nutno zvolit i migrační řešení.

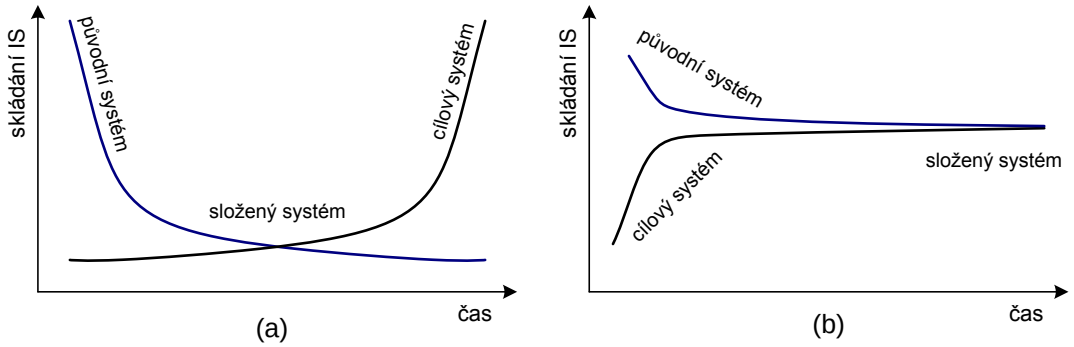


Diagram 3: Rozdíl mezi postupnou migrací (a) a začleněním (b) [Mec01b].

1.1.3 Heterogenita

Důležitým bodem je heterogenita IS, která je přirozenou součástí autonomního vývoje integrovaných systémů. Její překlenutí je v mnoha ohledech hlavním úkolem integrace. Existuje řada možných klasifikací heterogenity, například dle [Bus99] rozlišujeme 3 základní typy heterogenity a jejich další podtypy:

- *Syntaktická heterogenita* zahrnuje *technickou heterogenitu* (např. protokoly nebo bezpečnostní procedury) a *heterogenitu rozhraní*, která se týká rozdílů v přístupu k jednotlivým komponentám (např. omezená vyjadřovací schopnost použitého dotazovacího jazyku).
- *Heterogenita datového* modelu vyjadřuje rozdíly v sémantice konceptů, které zachycují (např. chybějící dědičnost u relačního modelu versus její přítomnost u modelu objektově-orientovaného). Tento druh heterogenity je vhodné oddělit od dalších sémantických konfliktů vzhledem k úrovni jeho výskytu, protože typicky bývá tento problém řešen samostatně (např. konceptem wrapperů, které zakrývají konkrétní charakter modelu [Wie92], nebo speciální transformační vrstvou v systému [She90]).
- *Logická heterogenita* je pak nejkomplexnějším druhem přinášejícím zřejmě největší problémy do procesu integrace systémů. Dále ji můžeme rozlišovat na *sémantickou heterogenitu*, která zahrnuje vyjadřovací (např. synonyma a homonyma) a reprezentační (např. rozdílné jednotky) konflikty, *heterogenitu schématu* a *strukturální heterogenitu*.

2 Teoretický pohled na integraci

2.1 Integrace dat

Integrace samotných dat, která je stále důležitou součástí většiny integračních řešení, je do jisté míry klíčová i pro tuto práci. Lze ji definovat například jako „problém kombinace dat pocházejících a udržovaných v rozdílných zdrojích, přičemž je uživateli na tyto data poskytován unifikovaný pohled“ [Len02]. Uvedená definice velmi dobře popisuje situaci, která je v této práci postihnuta.

Při popisu teoretického pozadí stojícího za integrací dat vycházíme z článku Lenzeriniho [Len02]. Zde jsou uvedeny dva základní principy, které rozdělují možná řešení podle způsobu, jakým je modelován vztah mezi reálnými integrovanými daty (zdroj dat) a mezi integrovaným virtuálním pohledem na tato data (globální uspořádání), a sice:

- globální pohled (global-as-view; GaV)
- lokální pohled (local-as-view; LaV)

Kromě těchto dvou základních pak mohou existovat i alternativní přístupy, které jsou výsledkem kombinace zmíněných dvou.

Globální pohled je postaven na principu, že globální uspořádání je zachyceno na základě zdroje dat, zatímco *lokální pohled* vyžaduje, aby globální uspořádání bylo určeno nezávisle na zdrojových datech, a vztahy mezi globálním uspořádáním a zdroji jsou stanoveny určením každého zdroje dat jako pohledu nad globálním uspořádáním.

Pokusíme-li se tyto dva odlišné přístupy srovnat, docházíme dle Lenzeriniho k závěru [Len02], že práce s integrovanými daty je problematická v LaV, zatímco v GaV bývá jednodušší, jelikož zde můžeme s výhodou použít fakt, že existuje mapování určující přímo, která zdrojová data odpovídají globálnímu uspořádání. Oproti tomu v LaV víme o datech v globálním uspořádání pouze prostřednictvím pohledů představujících zdroje a ty poskytují pouze částečnou informaci o datech. Potom nelze přímo odvodit jak použít zdrojová data, dotazujeme-li se na data v globálním uspořádání, protože pro každý zdroj je přiřazeno mapování nad globálním uspořádáním. Na druhou stranu výhoda LaV spočívá ve snadné rozšiři-

telnosti, protože pro přidání nového zdroje dat stačí definovat nové mapování tohoto zdroje na globální schéma, zatímco v GaV to znamená nutnost zásahu do globálního schématu. Použití GaV má tedy význam pouze pokud integrujeme statickou množinu zdrojů; v opačném případě je třeba sáhnout spíše po LaV modelu za cenu horších podmínek pro vyhodnocování dotazů nad zdroji. Současné systémy jsou ovšem přesto založeny více na GaV modelu [Len02], případně na nejrůznějších variacích obou modelů.

2.2 Interoperabilita a klasifikace integračních systémů

Interoperabilita neboli „vzájemná operační součinnost“ je pojem, kterým bývá označován problém propojení heterogenních zdrojů a vzájemného přístupu mezi nimi. Dříve byla interoperabilita limitována pouze na strukturované datové zdroje jako databáze, postupem času se ale přesunula na široké spektrum obecných, nestrukturovaných zdrojů. Může být definována jako schopnost dvou nebo více systémů (nebo systémových komponent) vyměňovat si informace a takto získané informace používat [Nach02].

Interoperabilitu lze klasifikovat rozdělením na dva základní typy: sémantickou a syntaktickou. Zatímco *syntaktická interoperabilita* spočívající v aplikační spolupráci bez ohledu na implementační rozdíly (jazyk, platforma apod.) není zvláště s rozvojem middlewaru a webových technologií velkým problémem, *sémantická interoperabilita*, která umožňuje spolupráci na úrovni znalostí i přes sémantické rozdíly, je skutečnou výzvou. Zde je třeba vyrovnat se s odlišnou reprezentací informací a rozdílným chápáním jejich významu a vytvořit sémanticky kompatibilní prostředí založené na konceptech uznávaných autonomními jednotkami. [Park04]

2.2.1 Sémantická interoperabilita

Budeme-li se tedy dále zabývat sémantickou interoperabilitou, můžeme rozpoznat několik odlišných strategií, jak ji dosáhnout. Dvě základní tradiční řešení vycházející z integrace databází uvádí Sheth a Larson [She90], kteří rozlišují těsně a volně spojené řešení. V *těsně spojeném* neboli statickém řešení je heterogenita řešena již při zapojení nové komponenty vytvořením wrapperů, které převádí schéma lokálního datového modelu do kanonického datového modelu, a vrstvou mediátorů, která kombinuje násobná schémata komponent do integrovaného schématu, kde jsou

vyřešeny nekonzistence. Připouštěna zde je jak možnost vytvoření globálního schématu, do kterého jsou integrovány všechny komponenty se svými schématy, tak možnost zachování vícenásobného integrovaného schématu. Ve *volně spojeném* neboli dynamickém řešení je naopak heterogenita řešena až koncovým uživatelem ve chvíli položení dotazu, což je umožněno existencí metadat popisujících zdroje a nutné konverze, a předdefinované pohledy nejsou poskytnuty. [Nach02]

2.2.2 Systémy založené na mediátorech

Uvedené rozdělení lze dále zobecnit, aby odpovídalo definovaným federativním IS (viz 1.1.1), a rozšířit, aby zahrnovalo samostatně uvedené informační systémy založené na mediátorech [Bus99]. Celkový náhled na klasifikaci FIS je znázorněn v diagramu 4; je ovšem nutno poznamenat, že uvedené rozdělení je jen přibližné a uvádí pouze 3 nejzákladnější druhy systémů.

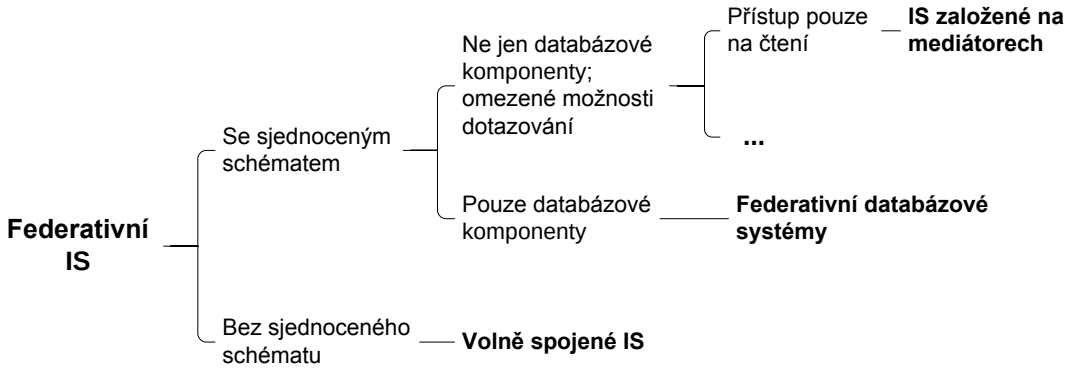


Diagram 4: Klasifikace federativních IS [Bus99].

IS založené na mediátorech jsou v současnosti důležitou kategorií integračních modelů. Spolu s federativními DB systémy spadají mezi těsně spojené řešení se sdíleným datovým schématem, ale oproti nim se liší pouze na čtení orientovaným přístupem k datům.

Princip mediátorů, zavedený Wiederholdem v práci [Wie92] (viz 1.1.1), spočívá v zajišťování spolupráce mezi uživatelem a fyzickým datovým zdrojem. Má jít o odlehčené, flexibilní, autonomní a opakovaně použitelné SW komponenty, které mohou být vzájemně propojovány. Jedná se tedy o velmi moderní, svým způsobem servisně orientovaný přístup, kdy je možné pružně reagovat na vznikající potřeby systému přidáváním mediačních komponent a jejich úpravami. Mediátory jsou tedy

cestou, jak integrovat rozdílné informační zdroje, které pracují se stejnými entitami reálného světa, díky jejich schopnosti odpovídat na dotazy týkající se těchto entit [Gar97]. Zároveň jsou takto založené systémy často budovány metodou „top-down“ vytvořením sjednoceného schématu dle aktuálních informačních potřeb. [Bus99]

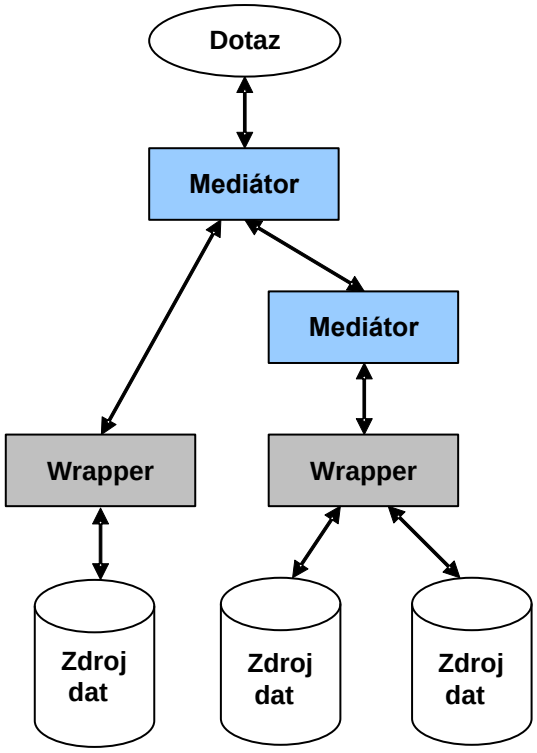


Diagram 5: Integrační schéma s použitím mediátorů [Ull97].

Ukázka principu architektury integrace informačních zdrojů založené na mediátorech je znázorněna v diagramu 5. Zde je mediační vrstva se vzájemně se využívajícími mediátory, které získávají informace z nižších úrovní, překládají je podle jim příslušejícího federativního schématu, případně tyto informace dále zpracovávají (např. s ohledem na kvalitu dat), a následně je zprostředkovávají jim nadřazeným komponentám systému. Mediátor tak představuje pohled na data nalezená v jednom nebo více zdrojích [Ull97, Gar97]. Konečně wrappovací komponenty znázorněné v diagramu zakrývají technické rozdíly a heterogenitu datového modelu integrovaných datových zdrojů. Tyto wrappery poskytují převod uživatelských dotazů na dotazy mající formu specifickou pro zdroje, které zakrývají [Gar97]. Typicky tak je použit (a případně i definován) globální dotazovací jazyk,

který implementují wrappery pro jednotlivé zdroje, a jenž je poté využíván mediátory pro jednotný přístup k těmto zdrojům. Pro wrappery je tak oproti mediátorům typické, že existují právě v jedné vrstvě a poskytují sjednocený pohled na zdroje, což je dále využíváno mediátory. Ty pak obvykle zajišťují provedení specifických akcí nad informacemi získanými ze zdrojů a charakteristické pro ně je vrstvení a vzájemné využívání se.

Jak dále uvádí Ullman, komponenty označované jako mediátory mají v realizovaných řešeních často charakter úložišť podobných materializovaným pohledům. Nejedná se tak o skutečné mediátory a implementace takovýchto úložišť s sebou nese značné problémy. [Ull97]

2.2.3 Alternativní klasifikace interoperatibilních řešení

Uvedené rozdělení vycházející z těsně a volně spojených systémů je sice stále platné a často aplikované i na současná řešení (např. [Nach02]), Park a Ram [Park04] ovšem v závislosti na jejich analýze výzkumných projektů poslední doby uvádějí revidovanou kategorizaci možných přístupů. Jedná se o tyto 3 široké oblasti:

- *Řešení založené na mapování* (mapping-based). Zde je snahou vytvořit mapování mezi sémanticky souvisejícími informačními zdroji, většinou metodou vytvoření globálního, sjednoceného schématu, na které se lokální schémata mapují. Nevýhodou řešení pak je jeho závislost na konkrétních schématech a aplikacích.
- *Řešení založené na zprostředkovatelích* (intermediary-based) je postaveno na principu zprostředkujícího mechanismu zajišťujícího interoperabilitu. Ten může být například realizován mediátory, agenty nebo ontologiemi. Často se používají standardizované protokoly pro vzájemnou komunikaci mezi komponentami a ontologie definující společnou terminologii. V praxi se ovšem zdá být obvykle neproveditelné vytvořit a spravovat takovéto ontologie kvůli komplexnosti jednotlivých domén, což značně omezuje obecné použití. Pokud se takové řešení nakonec přeci jen podaří uskutečnit i v rozsáhlé aplikaci, bývá charakterizováno značnou složitostí a závislostí na aplikační doméně, kdy není možno jednoduše začlenit další domény.

- *Dotazově orientované řešení* (query-oriented) se vyznačuje použitím interoperabilitního jazyku, kterým může být například rozšíření SQL. Pomocí něj pak lze provádět dotazy nad více zdroji, kdy sémantické rozdíly jsou překlenovány existencí metadat, které jsou součástí výrazů. Zřejmou nevýhodou řešení je ovšem zátěž kladená na uživatele, po nichž je vyžadováno porozumění struktuře dotazovaných zdrojů a také řešení vzniklých sémantických konfliktů.

Při bližším pohledu na tuto kategorizaci ovšem vidíme také její blízkou návaznost na tradiční rozdělení prezentované dříve. Dále pak Park a Ram poznamenávají, že uvedené přístupy se navzájem nevylučují a bývají různě kombinovány. Také je podle nich zřejmé, že žádné z řešení neposkytuje komplexní řešení pro dosažení sémantické interoperability a každé z nich je vhodné pro specifický druh problémů. [Park04]

2.3 Aplikace teoretických principů

Výzkumné projekty snažící se aplikovat uvedené teoretické principy vycházejí obvykle z architektury řešení založených na mediátorech a díky jejich provedení je lze také charakterizovat jako integraci systémů pomocí pohledů [Ull97]. Zde se zaměříme především na systémy TSIMMIS a Information Manifold, na něž je často odkazováno a z nichž vychází i řada současných řešení.

2.3.1 Systém TSIMMIS

Systém TSIMMIS[1] [Gar97] vyvíjený na Standfordské univerzitě ve spolupráci s IBM je jedním z příkladů řešení realizujících architekturu mediátorů. Jedná se o systém pro integraci informací z řady rozdílných zdrojů. Pro vyrovnání se s tímto problémem poskytuje systém několik logických komponent a složením jejich funkcí je vytvořeno výsledné řešení. Jedná se o [Gar97, Ull97]:

[1] TSIMMIS představuje akronym názvu „*The Stanford-IBM Manager of Multiple Information Sources*“, tedy „Standford-IBM manažer vícenásobných informačních zdrojů“.

- odlehčený objektový model OEM (Object-Exchange Model), který je objektový z hlediska jeho základního konceptu skládání objektů, ale nevyžaduje přísnou kontrolu typů a je značně flexibilní. Každý takový objekt je složen ze 4 atributů, kterými jsou identifikátor (nepovinný), název, typ (může být primitivní jako číslo, řetězec nebo dokonce java-script a nebo složený představující množinu dalších OEM typů) a hodnota. Pomocí těchto atributů pak lze tyto objekty skládat tak, že simulují prakticky všechny struktury, které lze nalézt v běžných objektově orientovaných systémech.
- logicky založený objektově orientovaný jazyk specifikující mediátory (Mediator Specification Language; MSL), který může být chápán jako jazyk definující pohledy nad heterogenními zdroji a pracující nad OEM datovým modelem.
- jazyk specifikující wrappery (Wrapper Specification Language), který rozšiřuje MSL a dovoluje popis obsahu jednotlivých zdrojů a jejich dotazovací možnosti.
- společný dotazovací jazyk spojující komponenty. MSL je jazyk používaný k dotazování a specifikaci mediátorů a k dotazování wrapperů, jazyk LOREL („Lightweight Object REpository Language“) cílený na částečně strukturovaná data je pak orientovaný na koncové uživatele.
- generátor wrapperů a mediátorů, který pracuje na základě jednoduchého popisu jejich funkce.

V konečném důsledku pak systém neposkytuje globální schéma mediátorů, ale naopak propaguje všechna schémata wrappovacích komponent až k uživatelům [Bus99] a jedná se tak o řešení založené na konceptu GaV. V důsledku toho je pak značně ztíženo rozšiřování tohoto systému, jelikož pro každý nový zdroj musí být nejen vytvořena wrappovací komponenta, ale zároveň je třeba předefinovat mediátory, které jej mají využívat, a administrátor systému musí určit, zdali a jak má být nový zdroj použit [Ull97]. Naopak výhodou řešení je jeho práce s pouze částečně strukturovanými daty, kdy díky MSL podporuje práci se záznamy, u kterých je dostupná rozdílná množina informací a navíc se jedna informace může vyskytnout v odlišných strukturách různých objektů [Ull97].

V podání systému TSIMMIS je mírně deformován princip mediátorů, které zde představují pouze jednotlivé integrované pohledy nad daty, a naopak některé úkoly běžně určené mediátorům jsou zde přenechána wrapperům, jež obstarávají například dekompozici dotazů a kompenzování chybějící schopnosti dotazů [Bus99].

2.3.2 Information Manifold

Systém Information Manifold[2] (IM) [Lev96] byl vyvíjen jako prototyp v rámci výzkumného projektu AT&T s cílem poskytnout sjednocené rozhraní pro dotazy nad distribuovanými zdroji informací. Jeho primárním zaměřením byla integrace strukturovaných datových zdrojů WWW, ale jak sami autoři uvádějí, jeho použití je možné přenést i do prostředí rozsáhlejší organizace. Princip IM spočívá v zakrytí heterogenity integrovaných komponent pomocí schématu mediátorů, které je navrženo na základě potřeb na vrcholu systému. Systém je konstruován jako dvouúrovňový, složený pouze z vlastních zdrojů a z tzv. „globálních mediátorů"; v principu by bylo ale možné aplikovat vrstvení komponent i zde a použít jednu aplikaci IM jako zdroj pro jinou, kde by byla obalena pevnou množinou pohledů, které by ji zpřístupňovaly [Ull97].

Zřejmou výhodou systému IM je jeho snadná rozšiřitelnost a flexibilita daná autonomií jednotlivých integrovaných komponent a nezávislostí jejich schémat na schématu mediátorů, což je umožněno použitím konceptu LaV. IM tak poskytuje mediátory nezávislé na zdrojích i na dotazech. Dále pak toto řešení nabízí explicitní mechanismy pro popis speciálních vlastností informací, které poskytuje daný zdroj, a použití těchto informací v algoritmech zpracovávajících dotazy [Ull97].

2.4 Trendy v oblasti integrace systémů

Kromě uvedeného přehledu dlouhodobě zavedených a široce akceptovaných přístupů k integraci heterogenních informačních zdrojů existuje řada alternativních cest, které zmíněné metody dále rozvíjejí a otvírají nové možnosti. Jestliže původně se integrace týkala převážně samotných dat uložených v databázích a datových

[2] Název „*Information Manifold*" znamená jednak „rozvod informací" a dále také „informační různorodost" a jedná se tak o slovní hříčku, jelikož systém vyhovuje oběma charakteristikám.

skladech [She90], postupně se její chápání zobecňovalo, aby byla zahrnuta i integrace obecných informačních zdrojů jako jsou IS [Bus99] a především je znatelná orientace na komponentní systémy a služby [Wie92], což je jednoznačným trendem moderních řešení.

2.4.1 SOA – Servisně orientovaná architektura

Model architektury založené na službách by mohl být považován za jeden z hlavních principů, od kterého si slibují architekti systémů velké možnosti, a který můžeme naleznout v řadě moderních řešení týkající se integrace systémů.

Není možné podat jedinečnou definici termínu SOA, ale často bývá charakterizován jako styl architektury, jejímž cílem je dosáhnout volné vazby mezi interaktivními SW komponentami a jejich vzájemné spolupráce pomocí poskytování a konzumování služeb. Například jedna z čistě architektonických definic popisuje SOA jako „aplikační architekturu v níž jsou všechny funkce definovány jako nezávislé služby s dobře definovanými vyvolatelnými rozhraními, které mohou být volány v definovaných sekvencích, aby formovaly business procesy“ [Cha03].

Principy SOA můžeme vidět v podstatě všude; navíc pak není ani nutné omezovat se pouze na oblast SW. Servisně orientovaná architektura umožňuje tvorbu systémů, které poskytují služby dalším aplikacím prostřednictvím publikovaných a objevitelných rozhraní, a kde služby mohou být vyvolávány vzdáleně po síti. Klad SOA lze spatřovat v tom, že poskytuje pohled na architekturu nezatížený technologickými omezeními. Dále jsou také redukovány vývojové náklady a implementační riziko. [Cha03]

Důležitý princip, který je třeba aplikovat také při tvorbě integračních systémů je pravidlo SOA říkající, že je třeba hledat řešení, které produkuje co nejmenší počet rozhraní tak, aby přidání nové komponenty systému vedlo pouze k přidání jednoho rozhraní (tj. aby nebyly různé rozhraní pro různé komponenty). Důsledkem je, že toto není možno provést přímými propojeními komponent. [Cha03]

Často zmiňované webové služby (Web Services) neznamenají přímo SOA, ale jsou to pouze technologické nástroje, které usnadňují tvorbu servisně orientovaných řešení a jsou důkazem, že SOA může být implementována. Zároveň pak webové služby, které v sobě zahrnují technologie jako XML, SOAP, WSDL

a UDDI, jsou důležité technologie pro řešení integračních problémů. Na tomto místě je třeba vyzdvihnout právě XML technologie, které se stávají standardem pro přenos informací a pomáhají tak řešit heterogenní konflikty.

2.4.2 Kooperativní informační systémy

Kooperativní informační systém (The Cooperative IS; CIS) je definován jako velké množství dílčích systémů rozmístěných ve velké a komplexní počítačové a komunikační síti, které spolu navzájem spolupracují, mohou si mezi sebou efektivně vyměňovat informace a sdílet omezení a cíle. Informace a služby jsou pak dostupné v mnoha možných formách z původních i nových informačních úložišť, které podporují protokoly informačních služeb. [Dem97] Jak je uvedeno v [Bus99], nelze přesně rozlišit hranici mezi CIS a systémy založenými na mediátorech (viz 2.2.2). CIS jsou ovšem více zaměřeny na interakci mezi jednotlivými komponentami, což je široká oblast výzkumu autonomních agentů.

Kooperativní informační systémy bývají často označovány jako „systémy třetího věku“. Pro dosažení požadavků kladených na CIS musí takový systém naplnit dva hlavní úkoly. Prvním z nich je vyvinout nástroje a produkty, které umožní spolupráci nekompatibilním informačním zdrojům a aplikacím. Druhým cílem je pak vyvinout technologie, které dovolí průběžné vylepšování a evoluci systému, aby zachytil a mohl reagovat na současný enormní vývoj v oblasti informačních zdrojů a systémů. Taková infrastruktura musí podporovat konverzi velkého počtu nezávislých databází a aplikačního software od různých dodavatelů do dynamických a propojených spolupracujících komponent. [Myl97]

Spolupráce mezi informačními systémy bývá chápána jako schopnost výměny informací a zpřístupnění interní funkcionality systému z jeho vnějšku. Tato vlastnost bývá často charakterizována jako interoperabilita (viz 2.2), ale ve skutečnosti jsou procesy probíhající v kooperativních systémech více komplexní, zahrnují více než jen běžné sdílení dat a základní spolupráci a interoperabilita tak může být považována za prerekvizitu pro kooperativitu. [Dem97]

Některé další vlastnosti a charakteristiky CIS jsou:

- CIS neadresují pouze technologické problémy, ale berou v úvahu zároveň i sociální otázky.

- CIS se neorientují pouze na zpracovávání informací, ale zároveň poskytují podporu pro změnu kontextu, ve kterém jsou tyto informace zpracovávány.
- CIS jsou založeny na architektuře, kde uživatelsky orientované procesy a distribuovaný systém zpracování informací jsou vzájemně oddělené, autonomní a kooperující moduly jednoho systému.
- Návrh a tvorba CIS není dostatečně podporována současnými postupy a metodami softwarového inženýrství.

Podrobnější popis teorie vztahující se k CIS je nad rámec tohoto textu, ale další informace lze najít například v [Dem97] a [Myl97].

3 Obecné aspekty integrace heterogenních IS

V této kapitole budou rozebrány některé z hlavních problémů, které jsou spojeny s integrací heterogenních IS. Tyto budou poté adresovány při popisu konkrétních řešení v kapitolách následujících a bude provedeno srovnání úspěšnosti jejich řešení.

Kromě zde zmiňovaných konkrétních problémů, které se týkají především otázek spojených s kvalitou dat, jsou pak jako na každý nový systém kladeny značné nároky na služby, které bude poskytovat, přičemž je ale požadováno, aby byl systém dostatečně rychlý a výkonný. Jelikož jsou ale tyto dva požadavky v přímém protikladu (více poskytovaných služeb znamená větší výpočetní nároky), je třeba řešení obou úkolů vhodně vyvážit, protože naplnění v dostatečné míře obou je pro úspěch integračního systému rozhodující.

3.1 Kvalita integrovaných dat

Kvalita dat (KD) by měla být důležitým bodem při návrhu každého systému. Můžeme ji vyjádřit například tak, že má-li systém KD 100%, jeho data zcela odpovídají stavu reálného světa, který reprezentují, což platí vždy pro aktuální okamžik [Orr98].

Pojem zachování kvality dat zahrnuje celou řadu požadavků na uchování plnohodnotné informace v těchto datech, především pak požadavky na jejich konzistentnost, kompletnost, přesnost a zachování časových závislostí [Tay98, Nau99]. Tyto 4 potřeby identifikovali Ballou a Pazer (1985) a později je různí autoři rozšiřovali o další vlastnosti v závislosti na úhlu pohledu na řešený problém, vždy se ale typicky drželi těchto 4 základních charakteristik. Tímto způsobem tak získáváme základní dimenze, v rámci nichž řešíme jednotlivé otázky kolem KD a jejichž oddělením lze dosáhnout nejlepších výsledků [Tay98].

Uvedené 4 charakteristiky se zároveň shodují s těmi, které jsou nejvíce relevantní v našem případě spolupracujících systémů [Ber01]. Zde jsou za hlavní otázky považovány [Ber01, Sca04] tyto:

- Ohodnocení kvality dat spravovaných jednotlivými integrovanými jednotkami.
- Určení metod a technik pro výměnu informací o kvalitě dat.
- Zlepšení kvality dat v rámci integrovaných jednotek.
- Heterogenita daná odlišnou sémantikou dat v integrovaných jednotkách.

Obecně lze říci, že konkrétní problémy vztahující se ke KD byly v teorii i praxi již široce řešeny, vždy ovšem pouze v rámci jednotlivých systémů a ne v kontextu více provázaných heterogenních IS, kde jsou snahy o hlubší analýzu takto zobecněného problému patrny až v poslední době [Sca04].

Nejvíce již získaných zkušeností tak lze v kontextu integrace IS uplatnit v oblastech ohodnocení kvality dat a jejich zlepšení, kde je možno používat dříve navržené techniky v podstatě beze změny. Mezi ně patří především celá řada metod čištění dat, které řeší jak zlepšení, tak ohodnocení dat. Za všechny metody uveďme řešení prezentované v [Her98]. Obecně ale pro čistící postupy platí a je široce akceptováno, že provádět čištění dat bez analýzy procesů, kterými je těmito daty manipulováno, je pouze dočasné řešení a v konečném důsledku plýtvání zdroji, pokud nejsou analyzované procesy upraveny tak, aby díky nim nedocházelo k opětovnému znečištění dat [Mis03].

Otázka výměny kvality dat byla zatím adresována pouze částečně, příspěvkem do této oblasti je řešení DaQuinCIS prezentované v [Sca04]. Zde je navržena architektura pro správu kvality dat v integračním prostředí kooperačních IS, která ohodnocuje data jejich kvalitou, umožňuje dotazování se na KD jednotlivých záznamů a dovoluje výběr dat v závislosti na jejich kvalitě. Tím, že je poskytována věrohodná a akceptovaná certifikace kvality dat, je rozšířena možnost jejich použití v rámci integrovaných autonomních systémů, kde může být tato vlastnost požadována. Řešení toho, jak pracovat zároveň s daty i s daty o jejich kvalitě je zde zajištěno speciální ad-hoc službou, která zprostředkovává žádost o data a odpovídá v závislosti na kvalitě dat. Další řešení počítají například s vyhodnocením předdefinovaných parametrů a jejich zveřejněním v metadatech každým ze zdrojů nebo s rozšířením tradičního ER-schématu a jsou podrobněji uvedeny právě ve [Sca04].

Konečně sémantické problémy dané heterogenitou dat se týkají především rozpoznání společných entit integrovaných systémů a tento rozsáhlý problém je rozebírán samostatně v kapitole 3.2.

Uvedené potřeby na dosažení uspokojivé KD rozšiřují požadavky kladené na integrující systém a můžeme říci, že otevírají nový, limitující rozměr pro jeho řešení, jehož naplnění je klíčové pro úspěch celého systému. Pro ilustraci tohoto tvrzení uveďme například údaj z praxe, kdy bylo až 60% informací integrovaných v rámci velké firmy nepoužitelných právě kvůli nízké kvalitě dat [Orr98, Nau99].

Pro dosažení uspokojivé kvality dat v IS zavádí Orr obecná pravidla [Orr98], na která je třeba myslet a brát ohled při návrhu systému, tedy že:

- nepoužívaná data nemohou zůstat správná po dlouhou dobu;
- KD v IS je funkcí jejich použití, ne jejich kolekcí;
- KD nebude lepší než nejpřísnější použití dat;
- problémy týkající se KD se zhoršují se stárnutím systému;
- čím méně se některá z datových položek mění, tím horší je situace, kdy se tak stane;
- pravidla KD platí rovnocenně jak pro data, tak pro metadata.

Tato pravidla jsou odvozena z pohledu na IS jako na systém se zpětnou kontrolní vazbou, což odpovídá pohledu na schéma integrace IS. Za hlavní potřebu k dosažení dobré KD však Orr považuje především docílení stavu, kdy se pracuje pouze s daty, která jsou skutečně používána. Zdaleka ne vždy tomu totiž tak je a jako příklad uvádí konkrétní systém, který obsahoval 3x více datových elementů, než skutečně potřeboval. Tento příklad pak jistě není zdaleka ojedinělý. Toto pravidlo pak platí velmi přesně i pro integrační systémy, u kterých lze docílit minimalizováním množiny integrovaných dat lepších výsledků práce s nimi a zároveň výrazně zjednodušit implementaci i administraci takového systému.

3.2 Identifikace společných entit

Integrujeme-li heterogenní zdroje dat, klíčovým bodem je identifikace sémanticky si odpovídajících záznamů, tedy záznamů, které v daných systémech reprezentují totožnou entitu reálného světa. To se týká jak integrace fyzické

(např. při tvorbě warehouse řešení), tak i při integraci pouze na logické úrovni (např. s použitím uvedené architektury mediátorů a wrapperů). Tento problém byl popsán řadou autorů, kteří jej adresovali pod názvy jako „identifikace entit“, „přibližná unifikace záznamů“, „spojování/čištění“ [Her98] a nebo „spojování záznamů“ [Zha05]. Zde bude pro tuto činnost, při které jsou pro vstupní položku hledány položky v cílových datech jí odpovídající, použit termín „unifikace“ jako pojem charakterizující anglický výraz „matching“. Ačkoli jde o problematiku, která přímo souvisí s kvalitou dat (viz 3.1), pro specifičnost problému a jeho důležitost pro integraci heterogenních systémů je zde charakterizována samostatně.

Bylo ukázáno, že daný problém představuje velmi komplexní a časově náročný proces, a to především kvůli znečištěným datům, které typicky obsahují řadu nepřesností a chyb, a kvůli sémantickým heterogenitám mezi různými datovými zdroji. Pro ilustraci tohoto faktu uvádí například Winkler situaci, kdy projekt integrace několika seznamů adresátů pro US Census of Agriculture v roce 1992 spotřeboval tisíce pracovních hodin a to přesto, že byl použit automatizovaný unifikační nástroj. [Zha05]

Dle [Zha05] existují dvě základní metody řešení problému unifikace: řešení založené na pravidlech (rule-based) a řešení založené na samo-učících se technikách (learning-based). Ty se liší podle toho, jakým způsobem jsou odvozena pravidla, na jejichž základě se určuje shoda dvou záznamů.

Pokud je použito *řešení s definováním rozhodovacích pravidel*, jsou tato odvozena od znalostí konkrétních domén a pravidla se týkají především srovnání celkového podobnostního koeficientu při určité hranici citlivosti (tzv. „threshold“). Tento koeficient bývá většinou vyjádřen lineární kombinací úrovní podobnosti atributů srovnávaných záznamů. Takto zadávaná pravidla bývají většinou nastavena manuálně na základě důležitosti atributů, případně s doplněním o použití statistických metod. Nadstavbu tohoto řešení pak tvoří tzv. „equational theory“, kde se pracuje s množinou pravidel, které zachycují sémantiku procesu identifikace entit. Hernández a Stolfo prezentovali možnost jejího použití vytvořením vysokoúrovňového deklarativního jazyku, pomocí kterého mohou uživatelé snadněji definovat komplexní pravidla. Přes dobré výsledky, kterých dosáhli, má ovšem tato

metoda svá omezení (úzkým místem je proces získávání informací) a není příliš použitelná pro praktické aplikace [Mis03, Bed05].

Takovýto model založený na definování pravidel má výhodu především v tom, že v něm lze vyjádřit i složité vztahy specifické pro danou oblast, které nelze zapsat jako jednoduchou ekvivalenci klíčů. Na druhou stranu definování takových pravidel je časově náročné, vyžaduje hlubokou znalost aplikační domény a také je nutné experimentální ověřování chování definovaných pravidel. [Zha05]

Z toho důvodu se výzkum v této oblasti zaměřuje především na *samo-učící se techniky*. Jejich princip spočívá v automatickém generování rozhodovacích pravidel na základě poskytnutých vzorových dat. Doménoví experti jsou tedy zodpovědní pouze za poskytnutí klasifikovaných příkladů místo konkrétních pravidel, což značně usnadňuje jejich práci. Přední výhodou této metody tedy je hlavně eliminace procesu získávání znalostí o datech, podle kterých jsou jinak pravidla určována [Zha05].

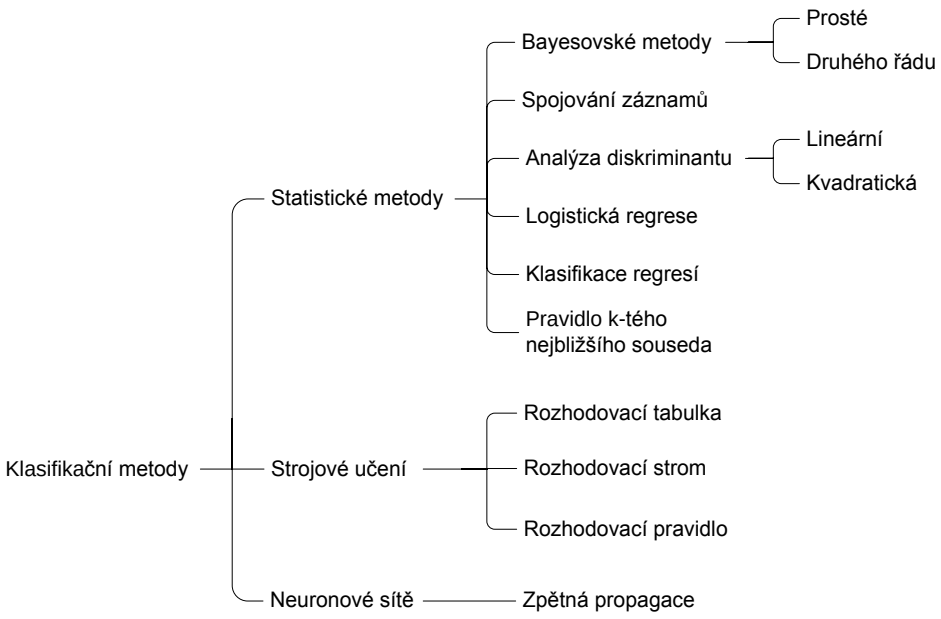


Diagram 6: Přehled klasifikačních technik pro generování rozhodovacích pravidel [Zha05].

Způsoby odvozovaní těchto rozhodovacích pravidel (klasifikátorů) se nazývají *klasifikační techniky* a bývají založeny zejména na statistickém rozpoznávání vzorů, na strojovém učení a na neuronových sítích. Podrobnější přehled některých nejrozšířenějších klasifikačních technik s jejich vnitřním dělením je znázorněn v diagramu 6. Tyto techniky jde pak s výhodou kombinovat a vylepšit tak jejich

přesnost. V takovém případě pak mluvíme o vícenásobných klasifikačních technikách. Příkladem takového řešení může být práce prezentovaná v [Zha05], kde jsou jednotlivé kombinované techniky vybrány na základě experimentů tak, aby se co nejvíce hodily pro identifikaci entit.

Při unifikaci záznamů narážíme také na výkonnostní problém, který je dán především faktem, že každý záznam musíme porovnat (a tedy pokaždé vykonat unifikační algoritmus) se všemi spravovanými záznamy, kterých může být mnoho. Pokud máme dvě datové množiny o velikosti m a n, pak v nejhorším případě je provedeno porovnání $m \times n$ párů, ale pouze u zlomku z nich je předpoklad, že budou unifikovány a celková cena na jednu nalezenou unifikující dvojici je tak neúměrně vysoká. [Mis03]

K řešení tohoto problému byla publikována řada technik. Z nich nejstarší je zřejmě tzv. „blocking“ navržený Newcombem, kde hlavní idea spočívá v definování množiny blokových proměnných $\{A_i\}$, které bývají obvykle unifikovány, a podle nich je datová množina rozdělena do oddílů tak, že všechny záznamy v každém oddílu sdílejí stejné hodnoty pro množinu $\{A_i\}$. Algoritmus pak porovnává pouze záznamy v rámci příslušných oddílů. Tím je snížena komplexita algoritmu, ale nemusí být nalezeny některé unifikující záznamy, pokud by se dostaly do rozdílných bloků. Určení blokových proměnných je tedy klíčové pro funkčnost algoritmu a je závislé na konkrétní doméně. [Mis03]

Podobné řešení pak je založeno na řazení dat, které je prováděno opět vzhledem k definované množině atributů $\{A_i\}$. Pokud jsou tyto řadící proměnné vybrány vhodně, stačí porovnávat pouze nejbližší sousedící záznamy určené jakýmsi okénkem s danou velikostí. Pokud má toto okénko velikost w, složitost algoritmu může být snížena na $O(wN)$. na druhou stranu pokud jsou v seřazené množině dat chyby jako přepisy, které mohly ovlivnit řazení, mohou být opět některé unifikační dvojice vynechány. Proto lze zavést vícenásobné řazení dle několika proměnných, pro každé řazení provést průchod algoritmu zvlášť, a konečný výsledek získat tranzitivním uzávěrem získaných hodnot. Kromě toho je možná ještě celá řada dalších optimalizací uvedených postupů a studovány byly i další techniky postavené na odlišných principech. [Mis03]

3.2.1 Problém neexistence společných domén

Podproblémem identifikaci společných entit je problém mající svůj základ v neexistenci společných domén, tedy skutečnosti, že při snaze o kombinování informací ze dvou různých heterogenních zdrojů se nelze spolehnout na to, že data z obou těchto zdrojů budou náležet do stejného jmenného prostoru. Typicky totiž integrujeme data, která můžeme podle Cohena nazvat „*jmenné konstanty*“ [Coh98], což jsou jména a názvy entit reálného světa. U nich dochází ke značné variabilitě v jejich zápisu, jako například neúplné názvy, použití zkratek apod. Pro ilustraci uveďme příklad, který v této souvislosti Cohen zmiňuje, a sice jak rozhodnout, které jmenné páry ze jmen „AT&T Bell Labs“, „AT&T Labs“, „AT&T Labs-Research“, „AT&T Research“, „Bell Labs“ a „Bell Telephone Labs“ odpovídají totožným výzkumným institucím. Jestliže pak na takto podobné entity narazíme při integraci, je legitimní požadavek na schopnost rozhodnout, zda jde či nejde o entity totožné. Jak k tomu Cohen uvádí, pouze v ideálním světě by byly totožné entity odkazovány společnými konstantami, v reálném světě je ovšem nejčastější situace, kdy v rozdílných zdrojích jsou používány rozdílné jmenné konstanty, a je třeba se s tímto problémem vypořádat při návrhu integrujícího systému.

Tato problematika je pak blízce spjata s dalším problémem, kterým je výskyt přepisů a podobných chyb v zápisech hodnot. To sice není dáno neexistencí společných domén, ale řešení problému je v mnoha aspektech totožné a oba problémy bývají pokrývány společně.

Řešení problému typicky spočívá v normalizaci zmíněných jmenných konstant pomocí definic vytvořených na základě specifických podmínek integrovaných IS tak, jak jsou identifikovány při analýze systému. Toto s sebou může nést problémy dané nízkou flexibilitou a značnou náročností na lidské zdroje.

Poměrně zjednodušená situace nastává v případě, kdy lze pro daný typ záznamů nalézt pevnou množinu možných hodnot, kterých může nabývat. Takových případů je v typických IS celá řada, například se jedná o tituly osob, o množinu rozšířených číselných kódů (kódy států, PSČ apod.) nebo i dalších umělých kódů tak, jak jsou používány danou organizací. Tím je usnadněn identifikační proces, protože problém je omezen na nalezení mapování mezi vstupní hodnotou a množinou přípustných hodnot a jestliže v obecném případě je odhadována shoda mezi dvěmi rovnocennými

položkami záznamů, zde je vytvořen společný převodník, vůči kterému mohou být obě položky porovnávány nezávisle. Takovéto usnadnění pak lze svým způsobem aplikovat i u obecnějších jmen jako jsou například jména obcí nebo vlastní jména osob, kde můžeme hledat pomoc v databázích používaných pojmenování. Výhodné také je, pokud má záznam splňovat určitý předpis (např. n-místné číslo). Tím je dána možnost pro další optimalizaci a eliminaci chyb při identifikaci entit a jejich unifikaci.

Obdobným pomocným faktorem pro rozhodnutí sémantické identity hodnot jsou schémata fonetického kódu, jako například „Soundex code“. Tato metoda sice nemůže být aplikována samostatně, ale s výhodou ji lze použít jako doplněk hlavního algoritmu. [Mis03]

Pro obecné řešení problému *přibližné unifikace řetězců* („approximate string matching“) pak existuje řada algoritmů. Ty považují každý záznam za řetězec, který je rozdělen do fragmentů dle doménově specifického rozboru (nebo v nejjednodušším případě podle mezer) a algoritmus je vykonán jednotlivě pro každou dvojici fragmentů. Příkladem takových algoritmů jsou „edit distance“, „Jaro algorithm“ a „n-grams“. [Mis03]

Další přístupy k řešení problému vycházejí z podobných postupů jako unifikace celých entit probíraná v předchozí kapitole a uplatnění zde získávají opět statistické metody. Příkladem může být Cohenem navržený princip „WHIRL“ [Coh98], kde je totožnost podobných jmen určována s pomocí statistické analýzy entitám příslušejících dokumentů.

Takto popsané techniky mají svou důležitost ovšem především v oblasti integrace obecných, nekontrolovatelných zdrojů, tedy v situaci, na kterou narážíme především v souvislosti s daty přicházejícími z Internetu. Jejich negativy jsou větší časová náročnost a implementační složitost a při integraci IS v rámci určité organizace nenachází tedy takové uplatnění. V této oblasti, na kterou jsme zde primárně zaměřeni, je při tvorbě systému poměrně silná kontrola nad množinou integrovaných dat a často proto není výhodné zavádět obecné mechanismy, které mohou být nahrazeny specifickými metodami vycházejícími z potřeb daného řešení. Takovéto řešení pak mnohdy vykazuje lepší výsledky a zároveň je rychlejší.

Na závěr je také nutno podotknout, že sebelepší algoritmus nemůže ve všech případech 100% rozhodnout problém identity záznamů. Určitá chybovost nemusí v některých případech vadit, ale naopak v řadě integračních systémů je správnost důležitým aspektem vycházejícím z povahy integrovaných dat a v takových případech je třeba, aby byly algoritmy nastaveny tak, aby upozornily na sporné případy při zvýšeném nebezpečí vzniku chyby. Definitivní rozhodnutí v takových situacích pak zůstává na lidské obsluze, která potvrdí algoritmickou heuristiku.

3.3 Problém relevance dat a hodnověrnosti zdrojů

V heterogenním prostředí, kde dochází k promíchávání dat pocházejících z různých informačních systémů, narážíme na další problém, kterým je kolize hodnot z různých zdrojů. Tedy jde o situaci, kdy dva integrované systémy pracují se stejnou entitou E a postupně oba systémy předají integrujícímu systému stejný atribut entity E, ovšem s různou hodnotou. Je třeba definovat pravidla pro transparentní řešení tohoto stavu, aby bylo vždy možno určit, který ze zdrojů je pro entitu E a daný atribut důvěryhodnější, což lze charakterizovat právě jako zavedení relevance dat. Konkrétním příkladem [Ber01], který je třeba řešit, je například situace, kdy dva zdroje, kterými jsou finanční úřad a obecní úřad, poskytují adresu osoby i její číslo sociálního pojištění, přičemž v tomto případě má adresa získaná od obecního úřadu vyšší relevanci než adresa od finančního úřadu, ale naopak relevanci čísla sociálního pojištění má nejvyšší mezi všemi zdroji zase finanční úřad.

Řešení problému, který nebyl zatím v teorii široce adresován, vychází ze zavedení funkce $<Zdroj, Data> \rightarrow R$, kde R je číslo udávající relevanci vstupní dvojice [Sce01]. Otázkou, kterou je pak nutno vyřešit v závislosti na konkrétním případu, zůstává granularita dvojic, kdy je třeba určit, na jaké úrovni se bude relevance měřit (například zda pro celé entity reálného světa nebo i pro jejich jednotlivé atributy apod.). Problémem pak zůstává způsob definování takovéto funkce a její aktualizace, má-li se dynamicky přizpůsobovat reálnému stavu. Ten se může měnit podle zkušeností získaných při používání nebo také se zvýšením kvality dat konkrétních zdrojů. Je zřejmé, že úvodní inicializaci je třeba provést na základě integrační analýzy propojovaných systému, další úpravy pak lze provádět

manuálně či poloautomatizovaně dle hlášených nedostatků, které by mohly být kompenzovány změnou v některých přiřazeních původní funkce.

3.4 Variabilita, flexibilita a rozšiřitelnost integrujícího systému

Navržení systému integrujícího data tak, aby byl schopen vypořádávat se efektivně s požadavky na změny a nejrůznější úpravy, je další velmi důležitou oblastí, na kterou je obvykle nutno se zaměřit. Jelikož informační systém za dobu své existence typicky prochází neustálým vývojem a chová se jako živý organizmus přizpůsobující se aktuálním požadavkům, je nutno, aby i integrující systém byl schopen všechny změny zachytit. Nesplnění této potřeby pak evidentně musí vést k neustálým problémům a dodatečným nákladům, které jsou spojeny s údržbou a úpravami takového systému, a případně až k selhání celého systému. To umocňuje zkušenost ze současné situace, kdy v praxi jsou nebo donedávna byly stále používány systémy IS staré 20 i 25 let a to přesto, že jejich životnost v době vzniku byla očekávána na 5 let [Orr98].

Hlavní problematickou oblastí změn, kterým je takový systém vystaven, pak zřejmě je množina integrovaných dat spolu s množinou závislostí mezi těmito daty. Zde jsme při návrhu systému postaveni před nelehký úkol, a sice, jak zajistit, aby, začne-li se v jednom z integrovaných systémů pro entitu uchovávat nová položka, která je relevantní i pro další systémy, bylo možno tuto položku zapojit do integračních mechanismů. To vše při zachování konzistence mezi novými a starými daty a zároveň bez nutnosti zásadnějších úprav v tomto systému.

Spolu s tím se pak rozšiřitelnost týká také možnosti přidávání nových informačních zdrojů do systému. Zajištění takovéto rozšiřitelnosti na úrovni zdrojů dat napomáhá volba vhodné metody pohledu na tyto zdroje z globální úrovně prezentovaná v kapitole 2.1. Zde byly uvedeny řešení LaV a GaV, přičemž LaV je považováno za vhodné pro rozšiřitelnost, která je zajištěna pouhým definováním nového mapování.

Správa dat v klasických relačních schématech je pro řešení popisovaného problému velmi nevhodná a víceméně nelze jednoduše a pružně reagovat na požadavky na změny [Obd06]. Je tedy třeba hledat metody, jak tomuto problému čelit.

Jednou z cest může být použití servisně orientované architektury (viz 2.4.1), kdy je integrační systém charakterizován značným množstvím komponent a interakce jsou řízeny dobře definovanými rozhraními. Extensibilita je v takovém případě základní charakteristikou řešení a je dosažena pouhým rozšířením příslušného rozhraní, čímž nejsou ovlivněny další komponenty systému a je na jejich vlastním rozhodnutí, zda budou nově nabízenou službu využívat. Za architektury využívající některých principů SOA, které jsou zde popisovány, lze svým způsobem považovat i v dnešní době oblíbené integrační řešení založené na mediátorech (viz 2.2.2) a CIS (2.4.2) a z toho pohledu se zdají být vhodné pro řešení tohoto problému. Kromě toho jsou předmětem výzkumu i možnosti, jak upravit nebo nahradit zmíněné rozšířené relační schéma. Příkladem je třeba vertikalizace dat a jejich řízení meta-daty, které mohou být snadněji spravovány [Bed05]. Řešení založené právě na tomto principu bude popsáno v následující kapitole.

4 Analýza konkrétního řešení 1: Integrace informačních systémů Univerzity Karlovy

Informační systém Univerzity Karlovy (ISUK), tak jak byl navržen a jeho projekt zahájen v roce 2003, bude v této práci podrobně analyzován, jelikož ve všech směrech odpovídá zde popisovanému problému a zároveň je postaven na netradiční datové architektuře [Bed05]. Tento projekt si klade za cíl integrovat systémy jednotlivých samostatných částí Univerzity Karlovy (UK), ze kterých se tato rozsáhlá instituce skládá. UK jako nejstarší a největší vysoká škola v ČR je složena z velkého množství celkem 17 fakult a dále jsou její součástí další výzkumné a jiné akademické instituce. To vše je zastřešeno rektorátem a právě jím by měl být výsledný integrující systém spravován, což ovšem neznamená, že by se s tím měla přenést zároveň i zodpovědnost za konkrétní integrované systémy. Ty si zachovávají svou autonomitu drženou jednotlivými institucemi, které je používají.

Vznik projektu byl motivován především potřebou vedení UK získat větší kontrolu a možnosti řízení součástí univerzity, což bylo vynuceno také novou legislativou, která přenášela určité administrativní požadavky z jednotlivých institucí na celou UK. To nebylo za tehdejšího stavu možné a například neexistovala centrální evidence osob, přestože řada zaměstnanců má pracovní vztah s více univerzitními institucemi současně. V druhé řadě pak měla plánovaná změna přinést výhody pro všechny osoby mající nějaký vztah k UK v podobě zjednodušení potřebných administrativních úkonů a celkově zprůhlednit a zefektivnit činnost informačních systémů na univerzitě.

Rozsah projektu ilustruje počet integrovaných IS, kterých je 30 a kromě nich se integrace týká ještě 20 dalších aplikací. Tyto všechny systémy dohromady spravují data 60 000 osob a ročně tak produkují 30 mil. záznamů. Tyto systémy jsou například mzdové a účetní systémy pro evidenci zaměstnanců fakulty i studijní systémy, které evidují informace o studentech včetně jejich známek, studentských prací a dalších informací.

4.1 Požadavky na projekt

Cíl projektu, tedy integrace lokálních aplikací samostatných částí UK a jejich stávajících IS, měl být dosažen vytvořením centrálního systému, ke kterému budou připojeny ostatní integrované systémy, přičemž takto vytvořená soustava bude splňovat následujících tři základní požadavky [Bed05]:

- Uchovávání dat – vytvoření centrálního úložiště dat, které bude obsahovat všechna data (z množiny těch, co jsou relevantní) ze všech integrovaných systémů. Při iniciálním připojení daného systému k centrálnímu tedy dojde k přenosu všech stávajících dat do centra a v průběhu dalšího přenosu pak bude každá změna v datech replikována do tohoto centrálního systému.
- Propagace dat – data budou rozšiřována mezi lokálními integrovanými systémy a to tak, že každá položka, která je propagována do centrálního úložiště je poslána zároveň i do všech lokálních systémů, pro něž je relevantní.
- Zaznamenávání historie dat – pro každou datovou položku propagovanou do centrálního systému je uchovávána její historie tak, aby byl zpětně dohledatelný vývoj hodnoty této položky v čase. Toto je vyžadováno především z důvodu zavedení interoperability systémů nad společnou množinou dat, přičemž takto je zajištěna kontrola nad proběhlými změnami v těchto datech.

Důležitým bodem pak je požadavek, aby k integraci lokálních systémů došlo bez jejich vnitřních úprav nebo dokonce nahrazení (pokud to ovšem není zároveň přání instituce, která má takový systém na starosti). To v důsledku znamená, že je třeba provést podrobné analýzy těchto systémů a na jejich základě určit několik málo možných způsobů spolupráce lokálních aplikací s centrálním systémem. Vždy je ale tato spolupráce možná jedině dopsáním speciální komponenty, která tvoří nadstavbu nad lokální aplikací a zajišťuje pro ni export a import dat. Tyto speciální komponenty jsou v projektu nazývány „*exportně/importní filtry lokálních aplikací*“ nebo jen „*lokální filtry*“.

Dalším podstatným požadavkem, který zásadním způsobem formoval podobu řešení, je potřeba flexibility a rozšiřitelnosti systému. Musí být snadné v budoucnu integrovat do navržené soustavy další lokální systémy a měnit ty stávající

bez nutnosti zásahu do jádra integrujícího systému. To má význam například pokud se některá instituce rozhodne nahradit svou lokální aplikaci za modernější. Kromě toho se požadavek na flexibilitu týká také množiny dat, která jsou globálně zpracovávána a propagována. Kromě možnosti přidávat a odebírat některé typy položek to znamená také možnost přeskupit logické schéma této množiny dat, tedy vazby mezi nimi. Tento požadavek pak má své opodstatnění ne jenom kvůli tomu, že systém byl budován s dlouhodobým výhledem na svou funkčnost a tedy byla potřeba, aby mohl pružně reagovat na budoucí provozní nároky, ale také jednoduše z toho důvodu, že u takto rozsáhlého systému není dost dobře možné, jak ukazují zkušenosti z jiných projektů, modelovat globální (integrační) schéma dat „na zelené louce“ bez nároku na jeho úpravy po uvedení systému do reálného provozu.

Kromě těchto základních požadavků je jako u každého podobného systému vyžadováno zabezpečení přenosu dat; zde pak se zvláštním důrazem vzhledem k tomu, že komunikace lokálních systémů s centrální aplikací neprobíhá v rámci uzavřeného okruhu. Je tedy třeba především zajistit, že nedojde k úniku dat nebo dokonce k jejich podvržení a dále pak zaručit odolnost vůči všem standardním síťovým útokům. Naopak podpora transakčního zpracování není vyžadována vzhledem k asynchronní spolupráci systémů.

4.2 Popis a charakteristika řešení

Vzhledem k tomu, že podpora pro dynamickou změnu interpretace dat musí být v jádru systému přímo vestavěna, jsou prakticky vyloučena řešení založená na triviálním konceptu statického modelování globálního schématu dat pomocí relačních databází. Takovéto tzv. tradiční řešení by sice mělo vynikající podporu v současných technologiích a metodologiích, ale nesplňovalo by požadavek flexibility a rozšiřitelnosti. Kromě toho toto tradiční řešení s sebou nese i další problémy jako například, že typicky nezachycuje původ ani historii dat a neurčuje, která data jsou relevantní pro který lokální systém.

4.2.1 Datový stoh

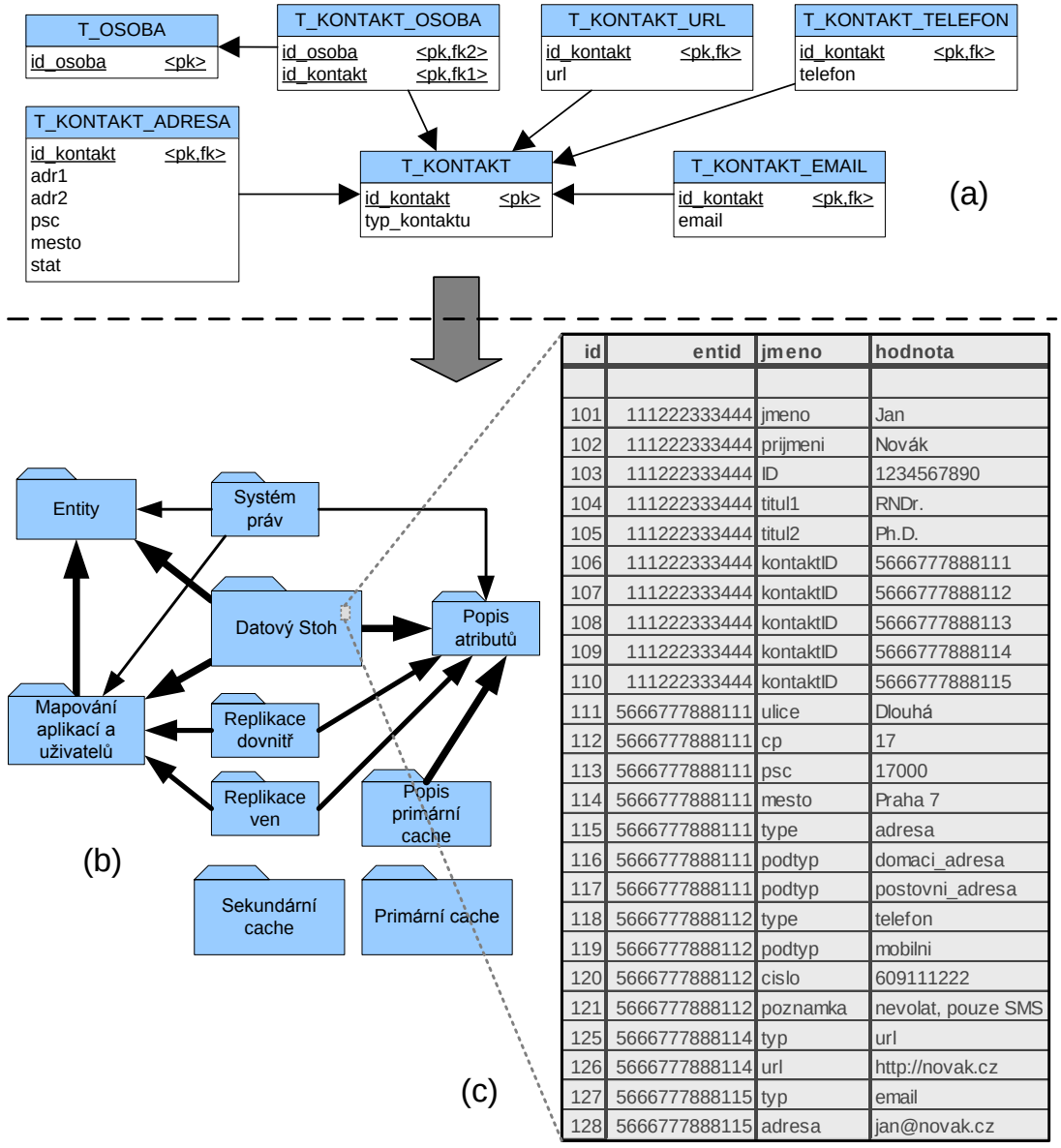
Ze zvýše zmíněných důvodů tedy autoři potřebovali nalézt alternativní cestu, kterou se může architektura integračního řešení ubírat. Zde je třeba vrátit se ke stan-

dardním integračním modelům GaV a LaV (viz 2.1), jejichž charakteristiky napovídají o vhodnosti těchto řešení a jejich negativech. Vzhledem k rozšiřitelnosti, která není možná v modelu GaV, a některým implementačním potížím, které jsou charakteristické pro model LaV, bylo třeba najít řešení kombinující oba zmíněné modely. Výsledkem této snahy pak bylo řešení založené na principu struktury *Datového Stohu* (DS; Data-Pile Structure). Toto řešení, stejně jako to klasické, je postaveno na technologiích relačních databází, které v současnosti dominují v oblasti uchování a zpracování dat a poskytují nezanedbatelnou podporu v podobě vysokého výkonu, bezpečnostních mechanismů, transakčního zpracování i existence řady vývojových nástrojů. Oproti klasickému řešení je zde ale rozdílná architektura datové struktury, kdy relační funkce databáze je použita pouze pro reprezentaci relační podoby dat a pomocných datových struktur.

Datový Stoh je postaven na principu vertikalizace dat (viz diagram 7). To znamená, že položka (v rámci projektu označovaná jako „entita“) s *N* atributy, která je v konvenčním datovém modelu reprezentována jedním záznamem, je v DS reprezentována množinou *N* záznamů hodnot odpovídajícím jednotlivým atributům a dále je doplněna sadou podpůrných metadat. Tato metadata určují vztahy mezi jednotlivými vertikalizovanými záznamy, umožňují z nich tak zpětně zrekonstruovat celý záznam entity se všemi jejími atributy v původní podobě a dále pak rekurzivně určují vztahy mezi takovýmito entitami. Tímto modelují celé schéma tak, jako by bylo reprezentováno klasickými prostředky relačních databází (a).

Tato představa je pak realizována schématem (b), kterému vévodí hlavní stohová tabulka (c) obsahující všechna data (kromě velkých datových objektů, které jsou z kapacitních důvodů vyčleněny do samostatné tabulky). Zde jsou uchovávána nejen aktuální, ale také historická data. Každá položka stohové tabulky je tvořena především identifikátory entity a atributu, ke kterým přísluší, vlastní datovou hodnotou (uložena v příslušné položce dle datového typu), indikátorem lokálního systému, odkud hodnota pochází, a časovými razítky, pomocí kterých je zachycen vývoj položky v čase. Důležité je podotknout, že z této tabulky nejsou během provozu systému mazána žádná data, ale místo toho je zastarávání hodnot realizováno právě nastavováním zmíněných časových razítek. Další datové tabulky, které tuto hlavní doplňují, definují množinu všech entit, se kterými systém pracuje,

mapují je na jim odpovídající položky v integrovaných lokálních systémem a zaznamenávají vývoj těchto entit v čase, jelikož může docházet k jejich slučování, rozdělování apod. Podotkněme, že každá entita má jednoznačný identifikátor v rámci celého systému.



id	entid	jmeno	hodnota
101	111222333444	jmeno	Jan
102	111222333444	prijmeni	Novák
103	111222333444	ID	1234567890
104	111222333444	titul1	RNDr.
105	111222333444	titul2	Ph.D.
106	111222333444	kontaktID	5666777888111
107	111222333444	kontaktID	5666777888112
108	111222333444	kontaktID	5666777888113
109	111222333444	kontaktID	5666777888114
110	111222333444	kontaktID	5666777888115
111	5666777888111	ulice	Dlouhá
112	5666777888111	cp	17
113	5666777888111	psc	17000
114	5666777888111	mesto	Praha 7
115	5666777888111	type	adresa
116	5666777888111	podtyp	domaci_adresa
117	5666777888111	podtyp	postovni_adresa
118	5666777888112	type	telefon
119	5666777888112	podtyp	mobilni
120	5666777888112	cislo	609111222
121	5666777888112	poznamka	nevolat, pouze SMS
125	5666777888114	typ	url
126	5666777888114	url	http://novak.cz
127	5666777888115	typ	email
128	5666777888115	adresa	jan@novak.cz

Diagram 7: Klasické relační schéma (a) a jeho vertikalizovaná podoba reprezentovaná soustavou řídících a datových tabulek (b) a vlastním Datovým Stohem (c).

Dále je pak toto schéma tvořeno řadou řídících a pomocných tabulek, z nichž nejdůležitější jsou:

- *metadata entit* určující druhy entit a atributy, ze kterých se skládají, a v podstatě definují globální schéma integrujícího systému. Takto definované schéma dat lze libovolně rozšiřovat, ovšem pouze inkrementálně kvůli udržení jeho zpětné kompatibility s daty v systému.
- *metadata aplikací*, která charakterizují aplikace integrovaných lokálních systémů. Upřesněme, že tyto aplikace jsou sdružovány do nadřazených skupin podle jejich druhu, protože typicky nastává situace, že více lokálních systémů používá stejný druh aplikace a pro ně je pak žádoucí mít možnost určit stejnou množinu metadat.
- *metadata aplikačně-entitní* reprezentovaná XML schématy, která v podstatě mapují schéma lokální aplikace na globální schéma. Reprezentace těchto mapování v podobě XML byla zvolena z důvodů dobré manipulace s takovou reprezentací, podpořenou validačními a transformačními nástroji. V datovém schématu systému je pak každé toto XML schéma zachyceno v soustavě společných databázových tabulek.

V praxi je pak definováno základní globální schéma a pomocí XML jsou modelována mapování schémat aplikací lokálních systémů na toto schéma, vždy jednou pro každý druh aplikace bez ohledu na příslušnost k systému. V případě změny v lokální aplikaci vedoucí ke změně datového schématu, nad kterým tato aplikace pracuje, je v první řadě provedeno inkrementální rozšíření globálního schématu (je-li to nutno), v druhé řadě je pak vytvořena nová verze mapování lokálního schématu na globální. Propagace dat do aplikací, které případně stále pracují s původním mapováním, poté probíhá na podmnožině dat vzniklé průnikem starého a nového lokálního schématu.

4.2.1.1 Výhody a nevýhody Datového Stohu

Hlavní výhody architektury DS vyplývají z požadavků na projekt ISUK, právě kvůli jejichž naplnění bylo toto řešení zvoleno. Připomeňme tedy, že je to především relativně snadná rozšiřitelnost a dobrá udržovatelnost datové struktury,

dále pak možnost uchovávání historie dat, zachycení příslušnosti dat ke zdroji a konečně podpora pro snadné definování řídících struktur určujících vlastnosti a interpretaci dat. K dalším výhodám pak také patří například existence globálně jednoznačného identifikátoru entit.

Tyto cenné výhody jsou ovšem vykoupeny řadou nevýhod. První zřejmou nevýhodou je nárůst objemu dat plynoucí ze struktury uložení dat. Tento nárůst závisí do značné míry na vlastní podobě integrovaných dat, ale měření na reálných datech personálního systému ukázala nárůst o cca. 130 % při 200 tis. položkách v DS. [Bed03] Avšak vzhledem k současnému hardwarovému vývoji a nárůstu diskových kapacit by se nemělo jednat o problém zásadní.

Daleko podstatnější nevýhodu lze ovšem spatřovat v problematickém přístupu a zpracování dat uložených v DS. Schéma zde uložených dat není podporováno vyjadřovacími prostředky databází (například nelze v dostatečné míře používat integritní omezení) ani technickými nástroji pro jejich tvorbu a správu. Také DBMS optimalizátor není dobře vybaven na dotazy nad tímto schématem, které lze navíc formulovat jen velmi komplikovaně a toto prostředí tak není uživatelsky ani programátorsky přívětivé, což znesnadňuje vývoj administrátorských i jiných aplikací. Kvůli tomuto problému byl tedy navržen koncept *cachí*. Jde vlastně o modelování dat v DS pomocí klasického relačního schématu, kdy jsou tyto cache udržovány a průběžně dynamicky aktualizovány, aby prezentovaly aktuální data v relační podobě, jak jsou zvyklí uživatelé i programátoři. S těmito cachemi jsou pak dále spojeny ještě další specifické pohledy (views) a data jsou zpřístupněna z vně systému až s jejich pomocí. Toto řešení pomocí cachí je vykoupeno dalším zhoršením výkonu a také znamená uložení značného množství redundantních informací. Navíc, a to považujeme za nejhorší, se tímto zeslabují některé hlavní výhody DS jako rozšiřitelnost a flexibilita. Proto je nutno používat tento koncept co nejméně a s rozvahou; jeho použití v omezené míře je ovšem v současnosti nutností. Například v projektu ISUK se necachují historické záznamy, ale takto zpřístupněny jsou pouze aktuální hodnoty pro základní přehled o datech a složitější rešeršní aplikace pracují přímo s daty v Datovém Stohu.

4.2.1.2 Modely podobné architektuře Datového Stohu

Myšlenka vertikalizace dat, na které je model DS založen, není zcela nová a již dříve byla předmětem výzkumu v oblasti správy dat. Za všechny uveďme projekt *C-Store* [Sto05], který efektivně využívá vertikalizaci dat v architektuře databáze optimalizované na čtení (zatímco většina současných DBMS může být považována za optimalizované na zápis). Toto řešení je založeno na ukládání dat ne po řadě po jednotlivých záznamech, ale po sloupcích, kdy je stejný druh hodnot jednoho typu záznamu uložen za sebou. Takto uložených dat pak lze s výhodou použít k optimalizaci sekvenčního přístupu k datům při operacích jako „join“ nebo seřazení. Díky tomu pak tento systém vykazuje vynikající výsledky v podobě několikanásobného zrychlení oproti současným komerčním produktům.

Porovnáme-li ovšem řešení C-Store s modelem použitým v projektu ISUK, vidíme základní rozdíl v úrovni pohledu na problém. Zatímco C-Store modeluje vertikalizaci dat nízkoúrovňově, kdy každý sloupec je uložen v odděleném prostoru, a až nad tím je stavěno schéma relační databáze, Datový Stoh je modelován právě jen prostředky samotných relačních databází a fyzicky je celý uložen v jedné DBMS tabulce. [Obd06] Tím logicky ztrácí výkonnost (která zde ovšem není nejvyšší prioritou), ale nahrazuje ji vysokou flexibilitou vzniklého modelu reprezentace dat. Zdá se tedy, že obě řešení by se dala s výhodou kombinovat pro podpoření výkonu dotazů kladených nad DS.

Druhou podobnost modelu Datového Stohu lze pak spatřovat v technologiích a principech rodiny XML [Bed03]. Organizace uložení dat v DS a způsob manipulace s nimi tak lze chápat jako určité překlenutí technik relačních databází a technologií XML, čemuž odpovídá kromě toho, že je jimi popsáno schéma dat, i způsob propagace dat mezi systémy, který, jak dále uvidíme, může být realizován právě pomocí XML souborů.

4.2.2 Architektura systému

Architektura řešení ISUK není dána pouze Datovým Stohem. Kromě něj je podstatný také způsob začlenění DS do celkové struktury systému. Základ této struktury, tak jak je znázorněna v diagramu 8, je v *Aplikačním Serveru* (AS), který tvoří jakousi bránu k DS, přes kterou jej zpřístupňuje ostatním částem systému,

především ale lokálním aplikacím integrovaným systémů. AS tak svým výlučným postavením zajišťuje udržení DS v konzistentním stavu, řeší otázku přístupových práv pomocí zabudovaného systému právových mechanismů, řeší konfliktní stavy systému a v neposlední řadě umožňuje v rámci sítě fyzické oddělení jádra systému od okolí a posiluje tím bezpečnost komunikačního modelu.

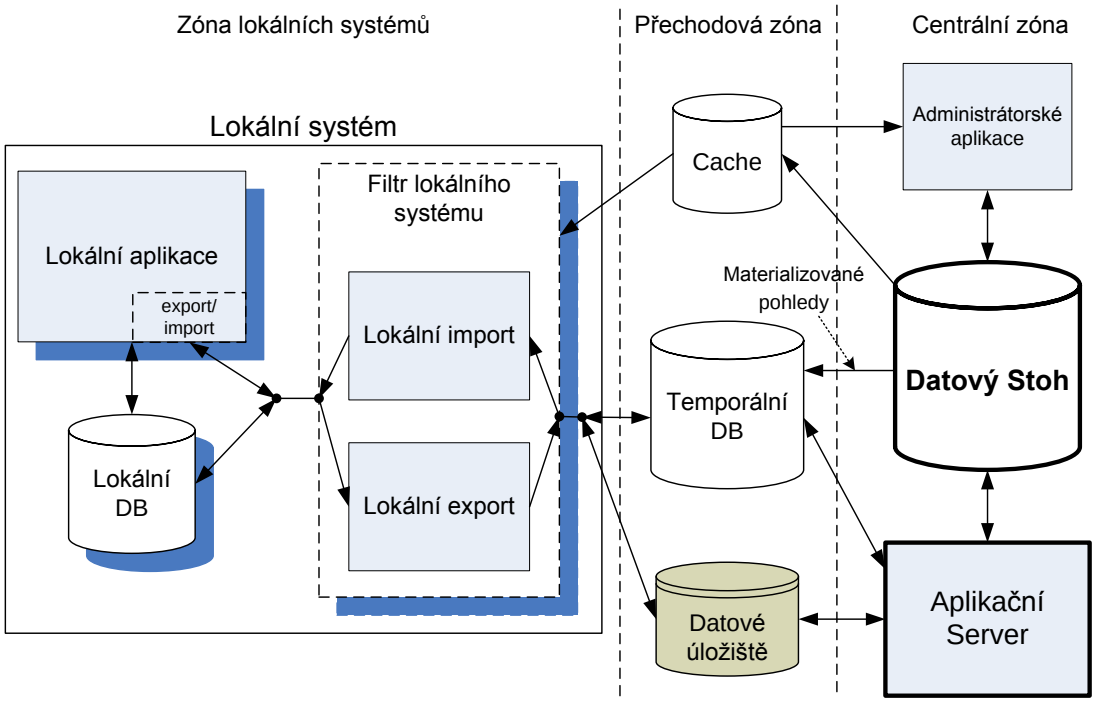


Diagram 8: Náčrt celkové architektury systému projektu ISUK.

Tento komunikační model je založen na tzv. „*Temporální Databázi*“, která je umístěna v přechodové zóně, a slouží jak pro výměnu dat mezi lokálním systémem a Aplikačním Serverem, tak pro zprostředkování komunikace mezi nimi. To se děje zápisem žádostí do speciálních tabulek, odkud jsou v pravidelných intervalech čteny AS. Ten pak do stejných tabulek zapisuje odpovědi. Dále pak Temporální Databáze poskytuje tabulky pro výměnu dat ve stohovém formátu, které jsou zde uloženy podle datového schématu příslušejícího dané lokální aplikaci. Toto schéma je pak zpřístupněno z Datového Stohu, kde je uloženo, pomocí materializovaných pohledů nad vybranou podmnožinou metadat.

4.2.2.1 Bezpečnost

Bezpečnost je založena, jak bylo naznačeno, na rozdělení integračního schématu do tří zón, které mají charakter vrstev a komunikace je možná vždy pouze mezi sousedícími dvěmi vrstvami. Tímto je účinně bráněno proniknutí útočníka k Datovému Stohu a narušení jádra systému. Dále je ale třeba zajistit ochranu proti úniku a především podvržení dat, která jsou do centra předávána. To je zajištěno definováním přístupových práv pro jednotlivé lokální aplikace a určením oprávněnosti pro podávání jednotlivých žádostí a dále pak výměnou šifrovacích klíčů při předávání dat přes Datové Úložiště.

4.2.3 Replikační procesy

Proces integrace v projektu ISUK je založen na datové výměně pomocí replikace dat přes DS replikačními procesy. Ty rozdělujeme na dva druhy, „import“ a „export“, podle směru přenosu dat z pohledu lokálního systému. Představa kooperace v rámci integračního systému tedy při bližším pohledu vypadá tak, že lokální aplikace se s centrálním systémem v pravidelných intervalech synchronizují prováděním exportních a importních operací. Při vyvolání *exportu* se příslušná data nachystají s pomocí exportního filtru do přechodové zóny pro AS, který dále propaguje informace o změnách dat do DS. Při žádosti o *import* do lokální aplikace poté AS vyhledá všechna data z množiny dat známých dané aplikaci, která se změnila od jejího posledního úspěšného importu, opět tato data připraví k přenosu a importní filtr dané aplikace je do ní vloží.

Konkrétní provedení přenosu dat a způsobu práce filtru lokálního systému závisí na druhu aplikace a její přizpůsobivosti pro integraci. Existují dva základní způsoby přenosu dat a to buď databázovou cestou přes Temporální Databázi nebo souborovou cestou pomocí XML souborů s daty, které jsou předávány přes Datové Úložiště. Prvně jmenovaná metoda je pak primární a použitelná u všech lokálních aplikací, které samy pracují s vlastní databází a umožňují tak aplikačnímu filtru relativně dobrou možnost správy dat (například sledování změn) a jejich snadný přenos pouhým překopírováním mezi databázemi. Druhý způsob byl určen pro aplikace hůře integrovatelné, které kupříkladu poskytovaly export svých dat pouze ve specifickém textovém formátu. Zkušenosti s projektem ovšem ukazují, že množství komplikací

spojených s touto metodou je příliš velké a integraci nepřizpůsobivých aplikací spíše zavrhují.

Specifický problém pak představuje rozdíl ve způsobu reprezentace datových entit v lokálním systému a centrální aplikaci. Jelikož řada starých aplikací pracuje například se všemi daty jedné osoby jako s jednou entitou (tedy právě jedna položka pro adresu, telefon atp.), zatímco v centru jsou tyto informace uchovávány odděleně, je na konkrétním lokálním filtru aplikace, aby data transformoval tak, aby mohla být v dané aplikaci uložena. Tento filtr tedy stále pracuje s lokálními identifikátory entit a v Datovém Stohu je pak implementován převod těchto lokálních identifikátorů na centrální, na jehož základě AS seskupuje a rozděluje data lokálních entit.

4.2.3.1 Export

Při exportu dat z lokálních aplikací do DS jsme postaveni před typické problémy spjaté s integrací dat, které byly adresovány v kapitole 3. Především jde o rozpoznání totožných entit přicházejících z různých zdrojů a o určení relevancí pro konkrétní data dle jejich příslušnosti ke zdroji.

První problém je řešen unifikační komponentou exportu, kdy je pro každou entitu (mající definovanou množinu určujících a relevantních atributů) spuštěn unifikační algoritmus vyhledávající ve stohu ji totožnou entitu. Tento algoritmus, který je pro celý systém klíčový, prošel celou řadou úprav zaměřených na zvýšení jeho kvality a rychlosti. Ve své výsledné podobě pak algoritmus pracuje nad cachemi DS, jelikož jeho vyhodnocení je značně prostorově i časově náročné a provádění výpočtu nad celým DS bylo neúnosné. [Dok05]

Druhý hlavní problém, kromě výpočetní náročnosti, pak představuje nastavení algoritmu tak, aby bylo minimalizováno množství nesprávně unifikovaných entit (nejhorší možný případ mající vážný dopad na kvalitu integrovaných dat) při současné dostačující úspěšnosti nalezení shody. To lze docílit především komplexním vyhodnocením všech určujících atributů daných entit, což může ovšem vést zpětně k nevyhovující časové náročnosti, jelikož by to znamenalo rekurzivní prohledávání všech dat. Finální verze algoritmu, která je výsledkem kompromisu těchto dvou aspektů dosahuje dostatečné kvality i rychlosti.

Sémantická analýza hodnot na shodu (viz 3.2.1), která je podmnožinou popsaného problému a vychází z heterogenní podstaty integrovaných systémů,

a především její obecné pokrytí, je pak v tomto řešení ignorováno. To například znamená, že pokud je stejná adresa reprezentována syntakticky různými hodnotami („ul.“ versus „ulice“), není tato shoda detekována. Pokud se ovšem nejedná o určující atribut, na základě kterého se provádí unifikace, není to ani problémem a hodnota se ustálí díky standardním replikačním procesům Datového Stohu. U specifických typů entit (např. titul osoby) pak jsou zavedeny číselníky pokrývající veškerou množinu přípustných hodnot a položky z lokálních aplikací jsou na ně mapovány, čímž je v těchto případech problém vyřešen.

Podrobný pseudokód unifikačního algoritmu spolu s implementačními detaily a vyhodnocením lze nalézt v [Dok05].

Dalším podstatným problémem při příchodu dat do DS z různých systémů je otázka relevance (viz 3.3). Je třeba zavést rozhodovací mechanismy, na základě kterých by bylo možné automaticky rozhodovat, jakým způsobem řešit situace, kdy je hodnota měněna jinou aplikací, než která byla jejím původcem. To zvláště proto, že integračními mechanismy by se tato změna následně přenesla i do této aplikace, jejíž původní hodnota mohla být správná. K takovémuto nežádoucímu stavu může dojít především v případě, kdy je do celého integračního systému připojována nová aplikace, jejíž data nemusí být v dostatečné míře vyčištěna. Na druhou stranu i aplikace s menší kredibilitou může být potencionálně zdrojem kvalitních dat specifického typu a nejenom proto nelze paušálně některým integrovaným systémům znemožnit přepisování dat již známých DS.

K řešení tohoto problému byl navržen systém, kdy je přiřazeno kladné číslo představující jakousi známku důvěryhodnosti (označováno jako „*kredibilita*“) typům aplikace, konkrétním aplikacím daného typu a konečně i konkrétním atributům pro jednotlivé typy entit a konečná hodnota pro danou položku se spočítá jako součin těchto tří kredibilit. Na základě porovnání těchto hodnot pro položku existující v DS a pro položku nově přicházející pak lze již snadno rozhodnout, zda může k jejímu nahrazení příchozí hodnotou dojít.

4.2.3.2 Import

U importu dat do lokální aplikace máme sice přesně danou množinu dat, která se změnila a má být aplikaci poslána, ale narážíme na množství problémů s nepřizpůsobivostí těchto lokálních aplikací, zvlášť za situace, kdy jde o starou, přetrvávající

aplikaci nepřipravenou na jakýkoli druh kooperace. To v mnohých případech znamená potřebu přenosu redundantních dat, bez kterých importní filtr nedokáže provést import do aplikace. Dále pak u mnoha aplikací je požadavek na to, aby každá datová položka přicházející z centra mohla být potvrzena lidskou obsluhou tohoto systému a mohla tak být odsouhlasena nebo zamítnuta.

Podrobnější detaily k importu dat z aplikace do centra a způsobu řešení řady kooperačních problémů lze dále nalézt v [Obd06].

4.3 Implementační detaily, výkonnost a efektivita systému

Naprostou nezbytností pro systém je podpora ve výkonné databázi, která dokáže především poskytnout zázemí pro souběh řady transakcí nad značnými objemy dat. Kromě toho je ale využíváno i množství dalších pokročilých prvků, jako jsou například materializované pohledy, „triggery“, SQL procedury apod. Důležitý je také výkonný optimalizátor, který je schopen přizpůsobit se specifickému DB schématu, které je dáno Datovým Stohem. To vše s ohledem na skutečnost, že systém je konstruován tak, aby co nejvíce výpočetní náročnosti bylo přesunuto právě na databázi.

V projektu ISUK bylo zvoleno databázové řešení Oracle, konkrétně DB stroj postupně ve verzích 9 a 10. Dá se říci, že toto řešení poskytlo uspokojivý výpočetní výkon i dostatečné množství funkcí, služeb a nástrojů pro vývoj a ukázalo se tak jako vyhovující pro tento systém.

K další konfiguraci pak uveďme, že jádro systému reprezentované Aplikačním Serverem bylo implementováno v C++ s použitím knihoven zajišťujících relativní nezávislost jak na platformě tak na databázi. Databázová přenositelnost je pak mírně problematická vzhledem k použití specifických vlastností použitého databázového řešení, zatímco platformní nezávislost byla v praxi ověřena a ač AS v provozu běžel na OS RHAS3, vyvíjen byl částečně i pod OS Windows. Na aplikační filtry zajišťující komunikaci s centrem pak nejsou kladeny specifické požadavky.

4.3.1 Replikační mechanismy

4.3.1.1 Export

U exportu je rozhodující kvalita unifikačního algoritmu entit. U něj můžeme kvalitu hodnotit podle různých aspektů, v daném případě byla stanovena kvalita algoritmu Q_u vzorcem $Q_u = 1 - P_{Ch} / (OK_u + OK_{neu})$, kde P_{Ch} je počet přípustných chyb (entity, které se měly unifikovat, ale nestalo se tak), OK_u značí počet správně unifikovaných entit a OK_{neu} značí počet nesprávně unifikovaných entit. Cílem je vyhnout se nepřípustným chybám, tedy situaci, kdy se unifikovaly entity, u kterých k tomu nemělo dojít. [Dok05]

Rozhodující je, že kvalita algoritmu je závislá na vstupních datech lokálních systémů, které se mají unifikovat s daty v centrálním DS. Problém, který zde nastává, spočívá ve skutečnosti, že totožnost dvou entit měla být dle teoretického obecného návrhu algoritmu porovnávána na základě vytipované množiny jejích atributů, kde jsou některé označeny jako *určující* a další, méně významné, jako *relevantní*. Tento obecný algoritmus pak předpokládá porovnání všech takových atributů (rekurzivně i pro entity vztažené k porovnávané entitě) a pouze při jejich shodě jsou entity prohlášeny za totožné. Má-li ovšem entita na straně lokální aplikace definováno pouze omezené množství určujících atributů, je problematické její porovnávání. Tento stav nastává hlavně v případě identifikace osob, který se ukázal být zároveň nejdůležitějším a nejproblematičtějším případem. Proto byl pro něj dodatečně navržen speciální algoritmus, který omezil unifikaci pouze na množinu identifikací (například rodné číslo, číslo pasu apod.) takové entity a zároveň dovolil pouze shodu v určujících atributech přítomných na obou stranách (což je ovšem poměrně implementačně náročné vzhledem k tomu, že tato množina se musí hledat zvlášť pro každou dvojici entit). Ostatní entity se pak dále unifikují podle původního návrhu s mírnými úpravami – především bez vyhodnocování relevantních atributů kvůli výkonnosti a se zamezením cyklů v rekurzi, což se týká obou verzí algoritmu. [Dok05]

V závislosti na těchto úpravách pak bylo dosaženo kvality 0.9987 Q_u při rychlosti 100 tis. entit za 31 min. To vše na vzorku dat o velikosti 600 tis. entit s 11 mil. atributy [Dok05]. Dosažený stupeň kvality je ovšem jistým způsobem

zkreslen a nevypovídá dostatečně o dosažené kvalitě unifikování, jelikož v celkovém vzorku dat je řada entit, u kterých se unifikace vůbec neprovádí, případně méně problematických entit než jsou entity osob. Pokud bychom se zaměřili pouze na unifikaci osob, dojdeme k daleko horším číslům, jelikož právě ony jsou i přes provedené úpravy hlavním zdrojem chyb. To je ovšem dáno především kvalitou dat, nad kterými algoritmus pracoval.

4.3.1.2 Import

Druhým replikačním procesem je import, kde je situace o poznání jednodušší. Především tu totiž pracujeme pouze s daty, které jsou přímo v DS. Je tedy dána množina dat, která se do lokálního systému posílá a tím je také usnadněna propagace těchto dat, jelikož odpadá problém s jejich identifikací. Hlavním problémem tak zůstává konverze dat mezi schématem používaným v DS a schématem lokálního systému, což je ovšem závislé na konkrétní situaci v daném systému a musí být vyřešeno v jeho importním filtru. Od centrálního systému je pak pro řešení tohoto problému nutná pouze podpora pro definování speciálních množin dat, které jsou nějakým způsobem relevantní pro specifické položky určené k importu. Ty jsou pak posílány spolu se změněnými daty. To může být potřeba pro zápis dat do lokálního systému, právě kvůli rozdílnému schématu uložení jeho dat. Navíc podobná funkčnost je potřeba i pro zobrazení importovaných dat k potvrzení – aby bylo možno uvést data v jejich kontextu, jsou pro ně definovány množiny závislostí, které určují data spolu s nimi zobrazované.

I přes tato rozšíření pak dosahuje import relativně dobré rychlosti a data jsou i v případě rozsáhlého DS generována v řádech 10 až 100 za vteřinu v závislosti na rozsahu importu, kdy i import všech dat relevantních pro konkrétní lokální systém proběhl do několika málo minut. Jako obzvlášť důležitá se zde ukázala optimalizace SQL dotazů, kterými jsou data z DS vybírána. Především pak bylo nutné časté opětovné analyzovaní dat pro správnou funkčnost optimalizátoru, který začínal vykazovat špatné chování už po krátkých intervalech práce nad DS, což by mohlo ukazovat na jeho horší vyrovnávání se s netradičním schématem uložení dat.

4.4 Naplnění požadavků kladených na systém

Z uživatelského hlediska je změna zaběhnutých pracovních mechanismů vždy problematická a zavedení nového informačního systému do organizace je s tím neodvratně spojeno. Proto by měl být příchod takového IS spojen s výhodami s ním spjatými, které jsou buď zúčastněnými jednotkami organizace akceptovány nebo existují prostředky danou změnu vynutit. V případě projektu ISUK, kdy jsou integrované jednotky značně autonomní, je ovšem možnost nařízení omezená, což je u projektu tohoto typu problematické. Vlastní přínos integrovaným systémům je totiž rozporuplný a jejich administrátoři se mohou obávat skutečnosti, že do jejich aplikací budou přicházet data z jiných systémů bez možnosti kontroly. Tento aspekt byl částečně vyřešen zavedením mezikroku s potvrzováním dat, přicházejících do lokální aplikace. To s sebou ovšem nese nárůst objemu práce pro obsluhu těchto lokálních aplikací a v důsledku další nespokojenost s existencí integračního systému.

Projekt byl zahajován s vizí, že zmíněný problém se spoluprácí lokálních systémů bude vyřešen časem postupným nárůstem jejich důvěry v integrační mechanismy. Pro správnou funkčnost těchto mechanismů je ovšem třeba spolupráce integrovaných aplikací, čímž se dostáváme do začarovaného kruhu. V projektu ISUK bylo příliš velké množství specifických nároků kladených na centrální integrační systém ze strany integrovaných aplikací neochotných přizpůsobit se a dá se říci, že vzhledem ke svému nevyzrálému stavu nedokázal tento systém dostát všem nárokům spojených s požadavky na integraci obecných aplikací.

4.4.1 Rozšiřitelnost

Na druhou stranu, požadavky na snadnou rozšiřitelnost a flexibilitu byly uspokojeny v plné míře a navržená architektura systému se ukázala jako velmi vhodná pro přidávání a změny integrovaných systémů. Celý koncept je tak ukázkou integrační metody začleňování původních aplikací a pozvolného přechodu systémů na novou architekturu (viz 1.1.2).

Podíváme-li se na to, co je třeba pro zapojení systému do integračního systému, vidíme, že nejdůležitější je zřejmě analýza tohoto přidávaného systému, která identifikuje množinu dat, které mají být předávány do DS a sdíleny tak s ostatními systémy. V návaznosti na to je třeba definovat mapování lokálního schématu těchto

dat na schéma používané DS, které může být případně inkrementálně rozšířeno, a na základě tohoto mapování vytvořit filtr pro tento systém, který by zajišťoval replikaci dat mezi ní a DS.

Pro vývoj tohoto filtru je pak nutná ještě analýza integrovaného systému z technického hlediska, která identifikuje možnosti zapojení filtru do systému a přístup k jeho datům. Například pokud má takový systém data uložena v interním nezveřejněném formátu a zároveň neposkytuje žádné rozhraní pro přístup k nim, je jeho zapojení do integračního modelu nemožné. Takové systémy pak byly v projektu ISUK z integrace vyloučeny. Naopak výhodné je, pokud integrovaný systém spravuje data v databázi, ke které je možný přímý přístup filtru. Pak celá práce filtru spočívá v replikaci dat mezi dvěmi databázemi. Jako druhá výhodná alternativa se navíc jeví možnost, že by integrované systémy poskytovaly množinu specifických služeb a rozhraní a pro takové systémy by pak bylo možno použít obecný filtr jako samostatný modul zajišťující replikaci.

V neposlední řadě je pak třeba, aby data lokálního systému splňovala určité kvalitativní nároky a bylo tak možné jejich zapojení do replikačních procesů. Pokud žádné takové nároky nejsou určeny, velmi pravděpodobně to povede k znehodnocení celého integračního mechanismu. K zajištění vyhovujícího stavu je potřeba provést vyčištění dat některou ze známých metod. Pokud totiž budou data integrovaného systému nekvalitní, budou tato data zanesena i do ostatních integrovaných systémů. Navíc, pokud už při zavádění integračního systému není věnována pozornost kvalitě dat, fungující replikační prostředí nemůže být vůbec vytvořeno zejména kvůli zmíněné unifikaci dat. To je pak případ projektu ISUK, kde právě tento aspekt lze považovat za hlavní příčinu problémů při jeho zavádění.

4.4.2 Kvalita dat a replikační mechanismy

Podstatnou oblastí dotčenou zavedením nového integračního systému je celkové zlepšení kvality dat v organizaci a především v jejich samostatných jednotkách díky procesům, které s nimi manipulují a udržují je tak aktuální. To byl jeden z podstatných cílů projektu, jelikož situace, kdy různé heterogenní systémy pracují nad společnou podmnožinou dat, vede k řadě problémů s jejich aktuálností a zastaráváním.

V daném případě je to vyřešeno vznikem centrální jednotky, která změny v datech distribuuje. Ovšem míra úspěšnosti dosažení tohoto cíle navrženou architekturou systému přímo úměrně závisí na kvalitě unifikačního algoritmu (viz 4.2.3.1) a na jeho úspěšnosti v rozpoznávání totožných entit. To je zřejmé z faktu, že pokud by data pocházející z různých heterogenních zdrojů nemohla být identifikována jako totožná, centrální DS by spravoval pro každý integrovaný systém disjunktní podmnožinu dat a tyto systémy by tak zůstaly navzájem neovlivněny. Původní teoretický návrh tohoto unifikačního algoritmu se ukázal při testovacím provozu jako nepoužitelný právě z důvodu nízké úspěšnosti a především toho, že neřešil některé důležité stavy, ke kterým docházelo. Průběžně byl ale vylepšován a postupně začal dosahovat o poznání lepších výsledků. [Dok05] Přes tuto skutečnost je ale znát citelný deficit daný podceněním této komponenty integračního systému, ke kterému zřejmě došlo, čímž bylo už na počátku způsobeno znehodnocení návrhu projektu a postupný vznik řady dalších problémů. Především je ovšem problematický stav integrovaných systémů a jejich dat, které nevyhověly potřebám na funkčnost unifikačního algoritmu.

Zmiňovaná unifikace je založena na principu manuálního nastavení unifikačních parametrů – rozhodovacích pravidel (viz 3.2). Toto nastavení bylo zamýšleno tak, aby bylo co nejjednodušší pro běžné uživatele a bylo snadno pochopitelné [Bed05]. Proto také vyšlo z pouhé možnosti nastavení parametru u atributů s hodnotami „relevantní“ a „určující“, jejichž názvy jsou samovysvětlující. V praxi se ovšem ukázalo, že u většiny entit není potřeba relevantní atributy uvádět (nebo spíše pro ně ani neexistují atributy, které by takovéto označení potřebovaly) a naopak jejich vyhodnocování je značně nežádoucí kvůli časové a výpočetní náročnosti. Na druhou stranu u specifických dat, kterými se konkrétně ukázaly být entity osob, navržený algoritmus nevyhovoval a jak bylo uvedeno dříve, musel být upraven, přičemž ani tak nebyl ideální. Z toho důvodu se ukázal navržený model rozhodovacích pravidel jako v praxi nevyhovující a to i přes to, že struktura takto řízených dat není v daném případě příliš složitá.

Kromě distribuce aktuálních dat mezi integrovanými systémy je pak dalším přínosem v oblasti kvality dat jejich čištění. Data integrovaných IS jsou typicky zadávána lidskou obsluhou a jsou tak zdrojem častých chyb; integrovaný Datový

Stoh s sebou ovšem přináší mechanismy, kterými jsou postupně tyto chyby nalezeny a opraveny. Předpokladem je ovšem existence správní jednotky řešící vzniklé konflikty, které jsou hlášeny z lokálních systémů nebo jsou zaznamenány sledovacími nástroji přímo nad DS. Takový detekovaný stav DS může být například tzv. „kmitání“, kdy je hodnota položky opakovaně měněna dvěmi soupeřícími lokálními systémy. Hlášení lokálními systémy pak probíhá především automatizovaně, kdy je založeno na existenci potvrzovací komponenty importu, která umožňuje zamítnutí položky přicházející do systému z centra. V takovém případě je vytvořen záznam o této skutečnosti, který je opět podnětem pro obsluhu administrátorské aplikace k vyřešení situace. Zřejmý problém, který tato metoda s sebou nese, je nárůst nároků na administrační obsluhu, bude-li počet odmítnutí velký. Zároveň s tím je mírně znehodnocen navržený koncept relevancí, který by měl být přehodnocen. Celkově se dá ale říci, že teoretický návrh způsobů zvyšování kvality dat by mohl fungovat za předpokladu správně pracujících podpůrných procesů a spolupracujících aplikací, což se zatím nepodařilo dostatečně ověřit v praxi.

4.4.3 Univerzalita v praxi: propojení stohových systémů

O tom, jak je systém založený na architektuře Datového Stohu univerzální svědčí také skutečnost, že systémy postavené na této architektuře lze navzájem velmi snadno kombinovat a integrovat. Tedy bez větších obtíží je možné jeden z takových systémů, který má mít status podřízeného, zapojit do celkového schématu centralizované integrační architektury jako další lokální aplikaci a to bez nutnosti existence komplexních filtrů, jaké jsou nezbytné pro zapojení běžných systémů. V tomto případě stačí tedy pouze u obou systémů předstírat, že druhý systém je lokální aplikace.

V reálném provozu je pak třeba vyřešit některé konkrétní detaily týkající se především vztahu takto propojených systémů a speciálních požadavků na replikaci mezi nimi vycházejících ze specifického postavení těchto systémů. Spolu s tím je třeba řešit způsob manipulace s identifikátory entit, jelikož tato funkčnost není zahrnuta v generickém algoritmu replikace, ale je zajišťována filtry lokálních aplikací, a v tomto specifickém případě musí být implementována zvlášť. Uvedené problémy pak řeší speciální aplikace mající podobu konektoru dvou takto

propojených systému, která oběma simuluje druhou stranu jako k němu připojený lokální systém a zajišťuje přenos dat mezi nimi. Takto je podporována jednosměrná jednorázová i obousměrná pravidelná synchronizace.

Uvedené řešení bylo použito spíše jen experimentálně a stále nejsou vyřešeny všechny aspekty takovéto synchronizace DS, aby byl zachován konzistentní stav obou systémů, ale v zásadě se ukázalo, že takové řešení je použitelné a jde o ukázku flexibility systémů založených na DS.

4.5 Závěrečné zhodnocení

Předně uveďme, že projekt ISUK v podobě, jak zde byl prezentován, byl nasazen pouze v pilotním provozu, který nesplnil očekávání zadavatele projektu (Univerzity Karlovy) kvůli výskytu velkého množství chyb a problémů s čistotou dat. Z toho důvodu byl i po snahách o nápravu na konci roku 2005 projekt pozastaven a nyní dochází k revizi projektu, která zřejmě vyústí v jednodušší řešení založené na standardních metodách a tím pádem také s menšími nároky kladenými na takovýto nový systém a naopak s většími nároky na integrované systémy, zejména pak na kvalitu dat, která poskytují.

Přesto ovšem nelze říci, že by uvedené skutečnosti snižovaly kredibilitu řešení založeného na architektuře Datového Stohu. Ty ukazují, že hlavní problémy byly dány stavem aplikací, jež měly být integrovány, a především pak daty, která poskytovaly replikačním procesům. Jako druhý aspekt vzniku problémů pak je nezkušenost a neprobádanost použitých technik.

Toto řešení pak ukázalo pravdivost některých teoretických předpokladů a dále je upřesnilo a zavedlo nové omezující parametry pro vznik takovéhoto systému; rozhodně tak nelze říci, že myšlenka integračního systému s architekturou DS měla být zavrhnuta. Naopak, jsme přesvědčeni, že uvedené řešení má své místo mezi použitelnými koncepty na integraci heterogenních IS. To ovšem platí pouze ve specifických případech, kdy výhody tohoto řešení převáží nad problémy s ním spjatými. Jedná se tedy zvláště o situace, kdy mají být integrována data z většího množství aplikací, které s nimi provádějí spíše jednoduché manipulace. Centralizované jádro s architekturou DS pak slouží jako místo, kde jsou data udržována, spravována a analyzována. [Bed03]

Informace získané díky implementaci tohoto řešení by pak měly být použity pro zrevidování původního teoretického konceptu, na základě čehož by mohl být vytvořen jakýsi základní, snadno konfigurovatelný framework obsahující jádro systému a balík základních služeb. Tím by tak mohlo být ustaveno prostředí, nad nímž by mohly být vyvíjeny již s daleko menším úsilím konkrétní integrační řešení.

5 Analýza konkrétního řešení 2: The Unitary Network of the Italian Government

V této kapitole bude podrobně analyzován systém vznikající v Itálii, jehož cílem je propojení informačních systémů italské administrativy. Tento projekt je další ukázkou integrace heterogenních IS; v řadě ohledů jde o systém s podobnými požadavky jako předchozí řešení, od nějž se ale odlišuje jak technickým provedením, tak i rozsahem.

„The Unitary Network (UN) *of Italian Government"* (RUPA[3]) neboli *„Jednotná síť italské administrativy"* si klade za cíl propojit a umožnit spolupráci a výměnu dat mezi systémy provozovanými v rámci jednotlivých samostatných administrativních jednotek jako jsou například sociální, finanční a jiné úřady, celá ministerstva a také lokální správní instituce. Díky tomu pak chce zjednodušit administrativní procesy, které jsou nákladné a zdlouhavé, a v konečném důsledku tak usnadnit občanům komunikaci s úřady. Tyto požadavky vycházejí z provedené analýzy, jejíž výsledky ukázaly zajímavá fakta, která validují potřebu vzniku systému [Bag98]. Z nich například vyplývá, že v celých 38% případů interakce občana se státním úřadem nemá úředník dostatek informací a musí je získat odjinud (často je tím pověřen sám žadatel). Dále se pak ukázalo, že informační systémy vykazují víceméně nulovou schopnost horizontální komunikace s dalšími IS, přestože v 35% případů je třeba o provedené operaci uvědomit další úřad. To vše je pak dovršeno faktem, že podstatná část komunikace probíhá stále papírovou poštou a dokument je uznán platným pouze pokud je na papíře a je podepsán. [Mec01][Bat02]

Celý projekt je značně rozsáhlý a ambiciózní a jak jeho autoři uvádějí [Mec01b], systém je navrhován s ohledem na úspěchy a selhání projektů s obdobnými cíli, které se již uskutečnily v minulosti. Také z toho důvodu je projekt rozdělen do řady

[3] Akronym RUPA pochází z původního názvu v italštině: "*Rete Unitaria della Pubblica Amministrazione*".

nezávislých fází a budován metodou „bottom-up“, tedy s minimalistickým jádrem systému a postupným přidáváním funkčnosti a rozšiřováním působnosti zapojováním dalších systémů. Dále pak bylo na počátku zahájeno několik projektů, jejichž cílem bylo otestovat a potvrdit správnost původního návrhu systému a to mnohdy ve více variantách technického provedení, aby mohla být určena ta z nich, která se ukáže jako více vyhovující daným podmínkám. Výsledky této fáze budou podrobněji adresovány později v jedné z podkapitol.

Rozsah projektu charakterizují údaje uvedené v [Bag98]. Podle nich se projekt týká integrace systémů, které využívá 0,5 mil. zaměstnanců zpracovávajících 12 tis. procesů. S nimi je spojeno 500 databází, které spravují 12 tera bytů dat. Integrovat je pak nutno 120 původních systémů. Udávaný roční rozpočet projektu je 2 mld. $.

Vznik projektu RUPA byl iniciován v roce 1993 vládní agenturou „Authority for IT in the Public Administration“. Poté následoval návrh projektu, po jehož dokončení a schválení se na začátku roku 1999 začaly implementovat první základní komponenty a také se rozběhly některé sub-projekty, jejichž cílem bylo provést již zmíněné ověření navržených konceptů. Úspěšně dokončit se podařilo jen některé z těchto dílčích projektů, ale postupně se uvedly do provozu komponenty, díky nimž začala UN poskytovat základní interoperabilní služby, které začaly být široce využívané, a také byl nastartován přechod některých administrativ na novou architekturu. V roce 2000 pak byl vládou ustanoven nový akční plán dotující projekt značnými prostředky s cílem dosáhnout spolupráce mezi administrativami do konce roku 2002. [Bat02]

5.1 Požadavky a původní představy o projektu RUPA

Cílem projektu RUPA bylo navrhnout a poskytnout potřebnou infrastrukturu propojující informační systémy italské administrativy a to přímo nebo i nepřímo. Takto integrované mohou být jak původní systémy (bez jejich přepsání), tak nové systémy, které mohou vzniknout v budoucnosti, proto je UN modelována jako svazek vysoce autonomních a nehomogenních spolupracujících systémů.

Představa způsobu integrace původních systémů, které jsou spravovány jednotlivými administrativními jednotkami, vychází z požadavku na zachování autonomity těchto systémů a na možnost v budoucnu dovolit jejich přechod na novou

architekturu nezávisle na ostatních, ovšem bez dopadu na globální architekturu. Spolupráce těchto jednotek pak má umožnit skrytí jejich heterogenní povahy poskytováním sémanticky bohaté sady rozhraní, které by zpřístupňovaly data a služby zainteresovaných stran. [Mec00]

Jedním z mála technických požadavků stanoveným na počátku bylo, aby byl model založený na vyspělých technologiích a komerčně dostupných produktech, čímž byly myšleny například ověřené middlewarové technologie jako CORBA, a dále aby byl otevřen řešením více dodavatelům na trhu, tedy aby nezvýhodňoval jedno marketingové řešení na úkor dalších. Samozřejmostí pak jsou, vzhledem k povaze dat, se kterými je manipulováno, požadavky na bezpečnost a na spolehlivost každé operace v rámci UN. Zároveň s tím bylo požadováno garantování vlastností ACID pro transakce, které budou v UN probíhat. [Bug98]

Jak bude patrno z následujícího podrobnějšího popisu systému, zdaleka ne všechny tyto původní představy o projektu RUPA byly naplněny a praktické zkušenosti získané vývojem prvních systémů spolu s organizačními důvody vedly k jejich postupnému přehodnocování.

5.2 Charakteristika a popis systému

UN je příkladem kooperativního informačního systému [Mec01] (viz 2.4.2). Jak autoři projektu uvádějí, rozhodně se nejedná o jeden velký monolitický systém, jehož cílem by bylo získávat informace o všem, ale má být cestou, jak efektivně reagovat na situace, kdy se v některém přidruženém systému stane něco, co se zároveň týká i dalších systémů, které tak mohou být informovány. RUPA dokonce není ani samostatným projektem. Naopak jde především o soustavu celé řady nezávislých (ale souvisejících) projektů, které mohou být jenom velice vzdáleně koordinovány ve vztahu ke společné architektuře a managementu. [Bug98]

Pokusíme-li se klasifikovat UN z technického úhlu pohledu, vidíme, že z topologického hlediska vykazuje UN prvky hvězdicového uspořádání – je organizována jako páteřní síť, k níž je připojována řada dalších sítí (každá pro jednotlivou administrativu zapojenou do projektu RUPA) a ty se připojují vždy přes centrální síť, kudy probíhá komunikace i v případě interakce mezi dvěmi lokálními administrativami. Jedná se tedy o jakousi „síť sítí“. Dále pak může být UN charakterizována

jako organizační síť („intranet“) italské vlády, ale s přihlédnutím na organizační strukturu a reálnou autonomii každé z administrativ se jedná spíše o „extranet“.

Integrované systémy lze podle jejich typu a organizační působnosti rozdělit do dvou základních kategorií: na centrální a lokální systémy administrativy. Vždy existuje právě jeden centrální systém administrativy, zatímco lokálních systémů administrativ existuje pro každou celá řada, přičemž je pro ně společná množina služeb, které poskytují, ale liší se vnitřní strukturou.

5.2.1 Kooperativní architektura

Základní princip, na kterém je UN postavena, je tzv. *kooperativní architektura* (Cooperative Architecture). Tímto termínem nazývají autoři model distribuovaného prostředí, pro který budou vyvíjeny a kde budou rozmisťovány nově vznikající *spolupracující informační systémy* (SIS) kooperativní architektury, které jsou navrženy podle zásad CIS [Bat02]. Takto vyvíjené SIS jsou pak postaveny obvykle nad původními IS jednotlivých organizací jejich rozšířením o kooperativní funkce a pracují s aplikacemi, které zůstávají zachovány.

Pro kooperativní architekturu je pak nejdůležitější koncept domén. Za doménu je považována každá konkrétní administrativa a k ní se vážící aplikace, data a sítě s dalšími počítačovými zdroji. Každá doména pak může být chápána jako samostatná entita, bez ohledu na svou vnitřní strukturu, ačkoli bylo očekáváno, že typicky půjde o nějaké původní aplikace založené na mainframech.

Co se týká způsobu přechodu původních systémů na tuto novou architekturu (viz 1.1.2), mají zde své uplatnění obě strategie. Připojení jednotlivých domén a inicializace jejich kooperace může být založena pouze na plynulém začleňování stávajících systémů, ovšem pro systémy v rámci jedné domény je možná, pokud je to výhodné, i jejich postupná migrace na novou architekturu. [Mec00]

5.2.2 Vrstevnatá architektura

Pro splnění dříve zmíněných požadavků byla UN navržena jako soustava spolupracujících funkčních vrstev (viz diagram 9). Nejníže položená z nich je *transportní vrstva* (transport service layer), poskytující transportní služby založené na protokolech TCP/IP. Důležité pro tuto vrstvu, vzhledem k povaze dat, je zajištění jejich bezpečného přenosu.

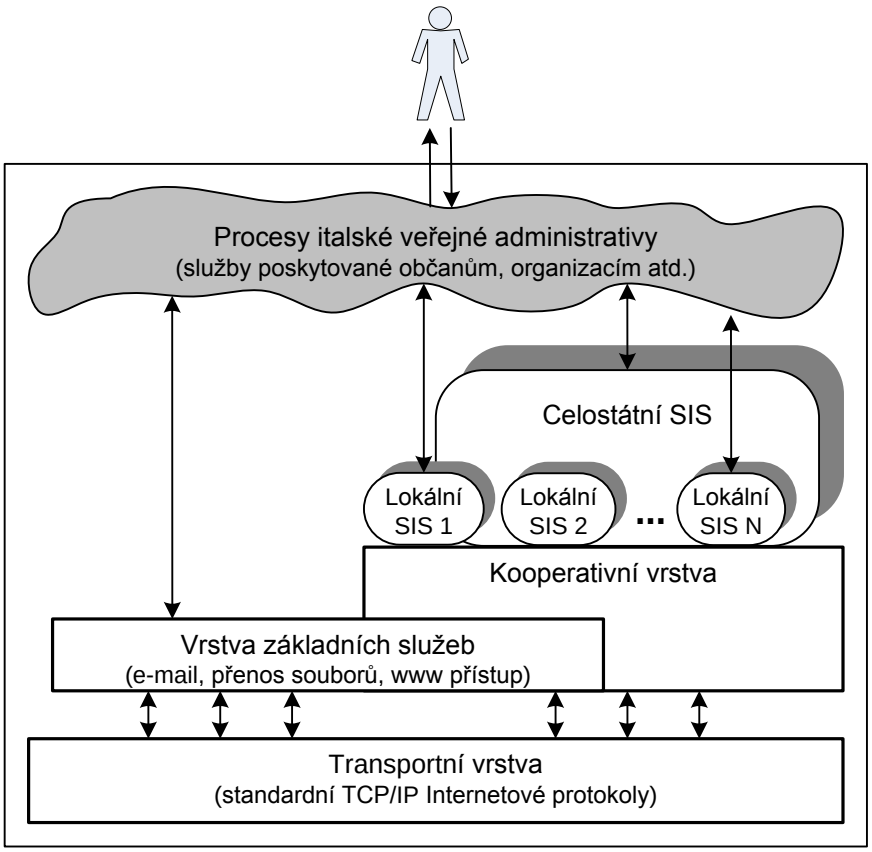


Diagram 9: vrstvy UN [Bat02].

Další vrstvou je *vrstva základních služeb* (basic service layer) jejíž funkcí, jak název napovídá, je poskytování základních (především komunikačních) služeb (například e-mail, přístup k Internetu a přenos souborů) uživatelům, resp. uživatelským aplikacím.

Nejdůležitější vrstvou, klíčovou pro systém, je pak *kooperativní vrstva* (cooperative service layer), na níž je vlastně založen model kooperativní architektury. Jejím cílem je umožnit aplikacím prostřednictvím SIS přístup v distribuovaném prostředí jak ke stávajícím systémům, tak i k systémům, jež budou vznikat v budoucnu. Proto je tato vrstva navržena tak, aby poskytovala množinu technologií, aplikačních protokolů a služeb pro efektivní spolupráci mezi kooperujícími jednotkami představovanými SIS. [Bat02]

Kooperativní vrstva byla zároveň zřejmě nejvíce diskutovanou částí UN a docházelo k neustálým posunům v podobě jejího návrhu s tím, jak se získávaly další a další zkušenosti při vývoji prvních SIS, které ji využívaly a ukazovaly se nové, specifické požadavky kladené na tuto vrstvu. Jedním z důsledků je také

postupné překrývání této vrstvy s vrstvou základních služeb. Tyto dvě vrstvy byly v počátečním návrhu zcela oddělené [Bag98], ale postupně se sblížily, jak je také patrno z diagramu 9.

5.2.3 Kooperační brány a rozhraní

Kooperační brány (Cooperative Gateways) (viz diagram 10) jsou na předělech mezi jednotlivými vrstvami a umožňují tak komunikaci mezi doménami. Domény jsou připojeny do UN pomocí doménových kooperačních bran a přes ně poskytují množinu dat a aplikačních služeb, kterou definují *kooperativním rozhraním* (KR). Tímto exportovaným rozhraním tedy určují, jaké služby poskytují dalším systémům připojeným k UN. Aplikační komponenta, která exportuje kooperativní rozhraní napříč rozdílnými branami různých administrací jsou základní stavební bloky pro vývoj SIS. Nové aplikace jsou tak budovány sestavováním takových komponent. [Bat02]

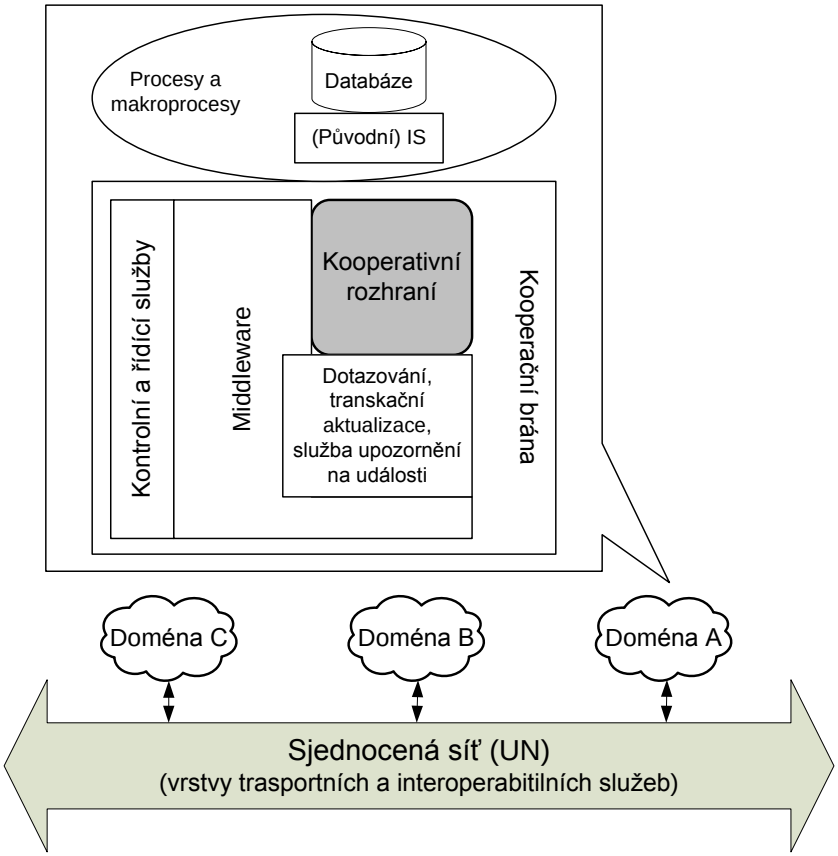


Diagram 10: Kooperativní brána [Mec01a].

Konkrétní podoba KR není pevně daná. Rozhraní mohou být navržena na základě různých principů a implementována odlišnými technologiemi. Několik pilotních projektů s nimi postupně experimentovalo a hledalo možné řešení, přičemž zvažované varianty lze rozdělit do tří skupin: *řešení založené na middleware technologiích*, kde by byla kooperační brána modelována jako objektově/komponentový server (např. jako Corba, COM+ nebo EJBs); *webově orientované řešení*, kde by byla data popsána a exportována pomocí XML a přenos založen na standardních internetových technologiích (převážně na HTTP protokolu); v poslední řadě pak také *„tradiční" řešení* založené na vzdáleném volání procedur (RPC) při použití rozšířených protokolů a transakčních monitorů. [Mec01b]

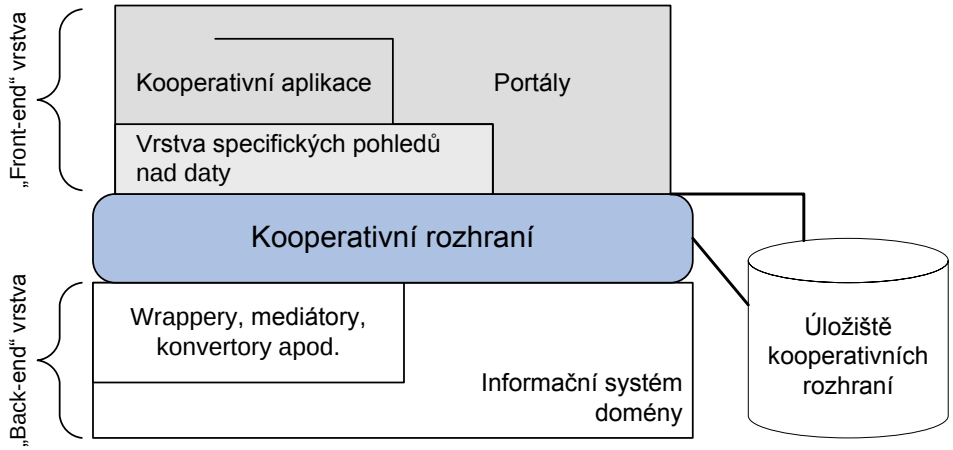


Diagram 11: Architektura SIS [Mec01a].

Cílová podoba SIS zahrnující kooperativní rozhraní je pak znázorněna v diagramu 11. Zde je znázorněna logická vrstva KR distribuovaná napříč rozdílnými kooperačními branami (každá administrativa rozmístí své vlastní rozhraní na svých branách). Tato vrstva je na vrcholu každého jednotlivého systému administrativ. Dále „back-end" vrstva je tvořena rozdílnými, většinou původními, systémy zpracovávající informace v konkrétních doménách, které jsou doplněny o soustavu wrapperů, mediátorů a dalších konverzních mechanismů. Konečně „klientská" vrstva se skládá z portálů, které zpřístupňují data a služby uživatelům, a z kooperativních aplikací specifických pro jednotlivé administrativní jednotky, které integrují data a služby exportované různými kooperativními branami. Tato vrstva může být případně rozšířena o mezivrstvu specializující KR zvláštním požadavkům aplikací.

Vedle toho přítomné úložiště kooperativních rozhraní exportované různými doménami je podmíněno „bottom-up“ vývojem těchto rozhraní. Zde jsou uchovávány informace o nich a jejich sémantice. Příkladem použití jsou definice událostí k upozorňování, na něž se lze přihlásit, které jsou v tomto úložišti zveřejněny. Odtud jsou pak používány aplikacemi, které se přihlašují k odebírání události nebo událost publikují, i komponentou, která informaci o proběhlé události odesílá. [Mec01a]

5.2.4 Způsoby kooperace

Kooperační architektura je navržena tak, aby podporovala více různých způsobu spolupráce mezi jednotlivými SIS. Autoři uvádějí 3 základní kooperační strategie: [Bat02]

- *Spolupráce na průběhu operace* (Workflow-based approach) vychází ze situace, kdy různé komponenty musí mezi sebou spolupracovat, aby došly k výsledku. Takovým případem je například ověření platnosti nějakého stavu o osobě, které si musí získat vykonávající úřad dotazem na jiný úřad. To je vyřešeno tím, že jednotlivé komponenty (představující systémy popisovaných úřadů), zveřejní pomocí kooperativního rozhraní služby, jejichž voláním lze získat příslušnou informaci. U takového druhu spolupráce je ovšem nutno zajistit striktní shodu na použitých technologiích, schématech apod. u všech zúčastněných stran. Také díky tomu je pak typicky třeba podobnou spolupráci vynutit pomocí nařízení (např. v podobě zákonů), případně potřebou v dané organizaci po podobné službě, aby se docílilo akceptace omezení a úprav potřebných pro kooperaci. V opačném případě, při chybějící motivaci na všech zúčastněných stranách, dochází k selhání tohoto konceptu.
- *Zpožděná aktualizace* (Lagging consistency-based approach) je založena na asynchronním poskytnutí dat jiné komponentě po té, co u ní došlo k interní změně v datech. Tím je se zpožděním zajištěno udržení konzistence v datech, která přesahují do více různých systémů a vyřešení jejich nesoudržnosti je jeden z důvodů, proč se k integraci heterogenních IS uchylujeme. Oproti předchozímu konceptu zde není zajištěna jakákoli koordinace interních procesů s procesy odehrávajícími se na vzdálené straně. Příkladem, kde byl

zároveň tento princip v projektu RUPA použit, je ministerstvo financí, které spravuje velké množství katastrálních dat, se kterými zároveň pracují lokální úřady. Zde je toto řešení s výhodou použito a umožňuje tak lokálním úřadům pracovat a měnit tyto katastrální data přímo, bez nutnosti přímé spolupráce s centrálním systémem, při současném zachování konzistence v datech.

- *Koncept zveřejnění informace* (Publish-based approach) je pak třetím možným přístupem, kde jedna konkrétní entita definuje množinu dat a služeb, které poskytuje a ostatní strany postupně mohou vyvinout aplikace, postavené právě na využití takto poskytovaných služeb. Toto řešení je specifické tím, že strana využívající tato data a služby je jednostranně závislá na straně, která je poskytuje. Z provedených experimentů vyplývá, že toto řešení má význam právě tehdy, kdy jedna administrativa má dominantní postavení v dané oblasti, které ji zajišťuje exkluzivní přístup k určitým informacím a existují další jednotky, které mají zájem o přístup k nim, ať už přímo nebo nepřímo pomocí služeb nad nimi, a tedy dobrovolně vytvoří komponenty, využívající poskytnuté možnosti.

5.3 Bližší pohled na řešení

Původní představy při návrhu projektu RUPA předpokládaly [Bug98], že vznikající kooperující systémy propojující domény budou založeny na podobných technologiích, aby byla snadná jejich integrace. Takovou sjednocující technologií měl být objektově orientovaný middleware, založený na ORB (object request broker), který měl představovat společnou infrastrukturu pro nasazení a integraci SIS. Tento plán vzal ovšem za své, jakmile začal vlastní vývoj prvních systémů, nebo spíše prototypů, a narazilo se na implementační i koordinační problémy. Hned první z těchto prototypů, kterým byl projekt „Arconet“, skončil neúspěchem a nebyl uveden do provozu, přestože byly postupně vyvíjeny dvě jeho verze – po selhání řešení založeného na middleware technologiích byla neúspěšná i verze postavená na XML a softwarových komponentách (EJB, COM+). Zároveň s tím se vývojové týmy dalších systémů také přeorientovaly na alternativní řešení a v konečném důsledku bylo rozhodnuto, že jednotlivé projekty budou testovat různá možná řešení, založená na jednom ze tří konceptů (middleware / webové technologie / tradiční

řešení), tak jak byly popsány v kapitole 5.2.3. Zároveň pak došlo k rozdělení dílčích projektů v rámci RUPA do dvou kategorií – na stěžejní (core-business) a testovací (proof-of-concept) projekty, kde projekty ověřující koncepty nebyly tolik komplexní a klíčové pro celý projekt a naopak stěžejní projekty byly více konzervativní a zároveň byly založeny právě na tradičních řešeních.

Popsaná diverzita sice značně zkomplikovala následnou integraci a spolupráci systémů, ovšem zároveň přinesla i klady v podobě cenných srovnání použitých technologií a užitečných poznatků pro vývoj dalších systémů. Hlavní poznatky a ponaučení, které autoři systému uvádějí [Mec01a] [Mec01b] [Bat02], jsou tyto:

- Při rozhodování mezi middleware řešením a mezi webově orientovaným řešením se ukazuje, že použití XML a standardních webových technologií je výhodnější, pokud se kooperace sestává pouze z jednoduché výměny dat. Na druhou stranu použití middlewaru se stává efektivnější ve chvíli, kdy je kooperace založena na poskytování služeb. Je ovšem třeba podotknout, že prudký rozmach webových služeb (jako je například SOAP) v posledních letech zřejmě ještě více umocní možnost použití webových technologií i v případech komplexních kooperačních konceptů. Další otázkou middleware řešení je problém zajištění bezpečnosti, který zde vzniká. Konkrétně IIOP protokol, jehož použití bylo plánováno, je obtížně přenášen skrz firewally, přičemž toto je vyřešeno ve webově orientovaném řešení. V případě projektu RUPA byla v první fázi implementována právě pouze podpora jednoduché datové výměny s vizí dalšího rozšiřování služeb a použití middleware řešení bylo považováno za příliš riskantní (také vzhledem k selhání prvotních prototypů jako Arconet). Zároveň bylo také počítáno s alternativou postupného skloubení obou řešení.
- Pro klientské aplikace se podle očekávání osvědčil „tenký klient“, typicky založený na HTML a reprezentovaný pouze internetovým prohlížečem, což je v souladu se současným trendem. „Tlustý klient“ (fat client) pak sice může nalézt uplatnění v případě aplikací určených pro větší samostatné instituce, ale i zde je patrný příklon k tenkému klientu.
- S tím, jak jednotlivé kooperativní projekty vyvinuly nezávisle na sobě rozdílné řídící komponenty kooperativních bran, vyvstal zájem o to, aby se

z nich vyextrahovaly služby společné pro všechny takové projekty a standardizovaly se tak některé z těchto komponent, aby mohly být jednotně poskytovány kooperativní vrstvou. Takovým krokem bylo například vyčlenění funkcionality společné síťové bezpečnosti kooperativních bran. Poněkud kontroverzním je ovšem tvrzení autorů, že slabá koordinace v první fázi projektu nevadí, pokud je časem vyřešena.

- Proces vývoje centrálního SIS propojujícího lokální systémy nemůže být zcela centralizovaný a homogenní, protože je složité získat všeobecný souhlas lokálních stran, které si drží svoji autonomii. Proto bylo prakticky nemožné při prvotním návrhu systému vytvořit takový model, k němuž by se v nezměněné podobě přidávaly všechny lokální systémy, ale naopak bylo nutné splnit jejich specifické požadavky. Východiskem z této situace se tedy zdá být iterativní a cyklický vývoj, při němž jsou jednotlivé cykly ve vývoji centrálního SIS v přímém vztahu k vývojovým cyklům lokálních SIS (viz diagram 12 a diagram 13). K tomu se pak ještě přidává „bottom-up“ vývoj, tedy vývoj s minimální množinou základních technologií převzatých všemi lokálními SIS a následným rozšiřováním této množiny v dalších vývojových iteracích.

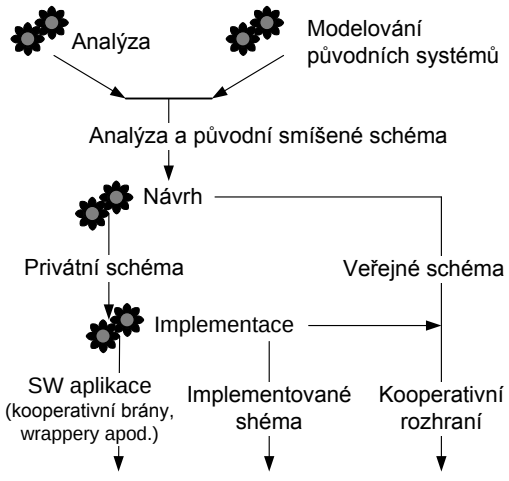


Diagram 12:

Celkový iterativní proces [Mec01a].

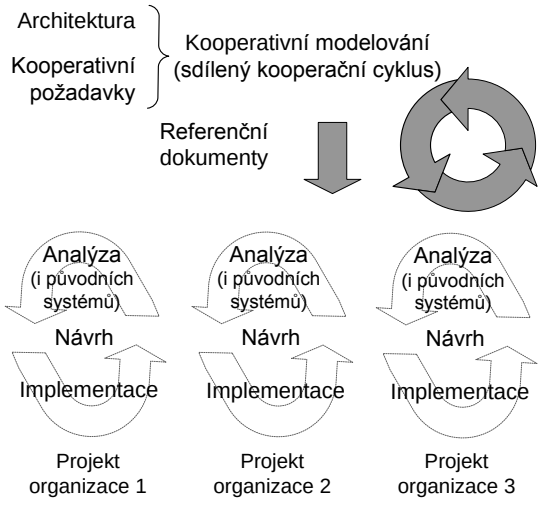


Diagram 13:

Proces iterace konkrétního systému [Mec01a].

5.3.1 Heterogenita, datová výměna a kvalita dat

Jelikož jsou jednotlivé kooperující systémy v rámci RUPA autonomní, je otázka řešení heterogenity jednou z důležitých při řešení projektu. To se týká nejen syntaktické heterogenity, ale také heterogenity technologické, protože je nutno překlenout rozdíly v technologiích, které SIS používají a umožnit jejich spolupráci. Sémantické problémy se zde objevují přesto, že všechny systémy pracují nad stejnými daty z reálného světa, protože interpretace těchto dat je různá podle toho, jak se odlišoval vývoj jednotlivých systémů. Zároveň ale platí, že tato sémantická heterogenita je možná pouze u řešení, kde kooperace není založena na principu „spolupráce na průběhu operace“, kde je nutná shoda kooperujících stran. U dalších dvou konceptů (viz 5.2.4) je heterogenita řešena později, jakmile je přistoupeno k integraci, pomocí „slepujících“ aplikací, tedy jakýmsi překlenovacím modulem. [Bat02]

V závislosti na studiích týkajících se kvality dat bylo vyhodnoceno, že pro každou spravovanou entitu, která je definována vstupními a výstupními procesy, může být (a je třeba, aby byl) definován jednoznačný „*data stevard*“, což je entita zodpovědná za kvalitu dané entity a zároveň stojí u jejího zrodu, kdy ji zpřístupňuje kooperačním procesům [Mec00, Mis03]. Existence tohoto datového stevarda je pak využita při návrhu některých kooperačních vzorů; například s výhodou jej lze použít v konceptu upozorňování na události návrhového vzoru Observer s architekturou „publish & subscribe“. V kontextu toho pak v reálné situaci hlášení změny adresy občanem u lokálního úřadu, který zde má roli datového stevarda, jsou notifikovány některé centrální instituce, pro něž je tato změna relevantní a jsou registrovány k jejímu odebírání. Potřeba tohoto typu kooperace pak vychází ze stavu, kdy jsou informace duplicitně udržovány ve více systémech. [Mec00]

Jako problematické se jeví, že některé otázky kvality dat (především jejich kvalita v jednotlivých původních systémech) nebyly brány v úvahu už v prvních fázích projektu, ale až ve chvíli, kdy začala působit potíže prvním dílčím projektům, jak je patrno například z [Mec01b], což vychází zřejmě z iterativní představy vývoje

projektu. V důsledku toho pak byly vytvořeny metodologie [Mis03], které adresovaly specifické problémy kvality dat. Příkladem je projekt RAE[4], kde notifikace událostí s nedostatečnou identifikací subjektů vedla k nízkému procentu unifikace událostí s daty odběratelů. Dále se pak ukázalo potřebné řešit duplicitu, neshodu a zastaralost záznamů (převážně záznamů adres), což vedlo od úvah o důkladném vyčištění dat až k definování frameworku řešícího kvalitu dat v projektu komplexně. [Mis03]

Součástí tohoto frameworku je pak postupně [Mis03]:

- předběžné ohodnocení dat na základě kontroly jejich vzorku vzhledem k dimenzím kvality dat (viz 3.1).
- definování cílů identifikováním možných vylepšení KD na základě výsledků předběžné analýzy a obchodních priorit.
- zkoumání příčin nízké kvality dat a návrhy na úpravu procesů, které manipulují s daty. Zde se vychází ze stavu, že důvody nízké KD je nutno hledat analýzou procesů, které s těmito daty manipulují. Tím se dostáváme k procesnímu modelování, které v tomto případě zahrnuje tři stavové domény, ve kterých se odehrávají události nad daty: reálný svět; abstraktní pohled na reálný svět, jak je viděna některou administrativní jednotkou; a abstraktní pohled na reálný svět, jak je viděný IS.
- monitorování KD pomocí instalovaných procedur, které periodicky hodnotí stav dat a určují, která data a jak degradují svou kvalitou.

V rámci projektu RAE byly řešeny dva specifické problémy týkající se kvality dat: identifikace chybných a zastaralých záznamů adres a ohodnocení jejich přesnosti a aktuálnosti. Podstatnou součástí řešení obou těchto problémů pak je unifikace a rozpoznání shodných záznamů. Pro tuto unifikaci se ukázal být dostatečně přesný algoritmus přibližného srovnávání řetězců s nastavitelnými parametry nezávisle aplikovaný na různé páry z pole záznamů, což je jeden z nejzákladnějších možných přístupů. [Mis03]

[4] Jeden z projektů vedených v rámci projektu RUPA zabývající se řízením upozorňování na události.

Tento algoritmus je zde chápan v kontextu klasifikačních technik jako klasifikátor přiřazující každé položce z množiny záznamů identifikátor určující míru s jakou vyhovuje měřeným podmínkám. Klasifikátor je implementován souborem zpracovávaných bloků, které jsou organizovány do datových toků. Rozhodovací logika je pak zapouzdřena do jednotlivých bloků. Tak je například umožněno v jednom z bloků provádět srovnání dvou záznamů podle definice vzdálenosti mezi některými jejich položkami (tzv. „edit distance“ algoritmus) a v dalším bloku pak provádět klasifikační rozhodnutí na základě prahové hodnoty pro spočítanou vzdálenost. [Mis03]

Přesnost tohoto algoritmu pak kriticky závisí na volbě tréninkového vzorku dat, který je použit k nastavení parametrů procesních bloků. Tento tréninkový vzorek dat je výsekem z celkové množiny dat s vlastností, že identifikátory přiřazené jednotlivým datovým položkám jsou známy. Takový vzorek pak je použit k určení parametrů u všech hodnot, kdy je experimentováno s nastavením algoritmu tak, aby podával co nejlepší výsledky, tedy aby byla minimalizována množina chybně unifikovaných záznamů. Jak již bylo uvedeno dříve, zde můžeme rozlišit dva druhy chyb, přičemž v rámci tohoto projektu bylo určeno, že prioritou je minimalizace množiny nerozpoznaných záznamů, které měly být unifikovány, na úkor chybně unifikovaných záznamů a to z toho důvodu, aby byly zachyceny potencionální problémy. [Mis03]

5.4 Závěrečné zhodnocení

Prezentované řešení sjednocené sítě s kooperativní architekturou je ukázkou servisně orientovaného přístupu k integraci heterogenních informačních systémů. Na uvedeném příkladu jsou názorně vidět některé problémy spojené s takovýmto řešením a také lze pozorovat složitost vývoje projektu a jeho postupné adaptování se novým požadavkům tak, jak se objevují s obsáhnutím dalších problémů a integrovaných systémů. Zde je třeba navrhnout dostatečně přizpůsobivou a zároveň jednoduchou architekturu, která vytvoří základní prostor pro integraci heterogenních systémů, a při tom umožní pružně reagovat na nově vznikající nároky těchto systémů. Takovéto řešení by tak mělo být maximálně modulární skládající se z množství nahraditelných komponent propojených jednotnou logikou.

Kooperativní architektura UN je příkladem architektury vycházející z těchto principů. Integrační logika je zde dána kooperativní vrstvou, principem domén a systémem bran a kooperativních rozhraní. Problémem jejího návrhu se pak může zdát její přílišná složitost daná velkou komplexitou a nejasnostmi v nárocích na ni, které vedly k neustálým změnám její podoby. To v konečném důsledku způsobilo existenci řady odlišných kooperativních strategií použitých v rámci integrace různých systémů a také přílišnou obecnost v možnostech definic kooperativních rozhraní, což dále vedlo k potřebě dalších konverzních mechanismů. Celkově tak bylo způsobeno zvýšení složitosti integračních mechanismů.

Důkazem komplexity celého návrhu je pak také přítomnost dolních dvou vrstev UN, zvláště pak vrstvy základních služeb. Jejím účelem je mimo jiné poskytnout internetovou konektivitu jednotným způsobem v rámci celé UN, v tomto rozsahu je to ovšem problematické a řada zainteresovaných lokálních organizací preferovala vlastní zajištění těchto služeb. To byla také jedna z příčin rozhodnutí, kterým byla UN nahrazena tzv. „Public Connectivity System“, který má vyřešit tento a podobné problémy. V rámci tohoto projektu tak mají být nahrazeny dolní dvě vrstvy původní architektury UN, kooperativní vrstva ovšem zůstává zachována a s tím i všechny její integrační mechanismy.

Integrace původních systémů pomocí jejich vzájemného propojení byla zvolena na místo provedení klasických procesů re-inženýrství stávajících aplikací a business-procesů kvůli rozsahu takového projektu. Ten by nebylo možno uskutečnit v krátkém nebo střednědobém horizontu a dlouhodobý projekt by byl omezen nejen z hlediska času, ale také by byl pro enormní technologický vývoj v podstatě neproveditelný při současném dosažení dostatečně moderního systému. Proto se zdá automatizace kooperativních procesů pomocí spolupracujících aplikací jako efektivní řešení. [Bat02]

Kooperativní architektura a s ní spojené projekty představují cenné prostředí pro výzkum v oblasti kooperativních informačních systému (viz 2.4.2). Tento fakt ovšem značně znesnadnil vývoj projektu. Zde hraje roli nedostatek zkušeností s tvorbou podobně rozsáhlého systému postaveného na principech CIS a tak na rozdíl od technologií, které se prudce rozvíjí a lze je s výhodou pro budování CIS použít, je jiná situace u metodologií, jak takové systémy vyvíjet. [Mec01a]

Za hlavní nedostatek projektu RUPA lze zřejmě považovat problémy s vedením projektu. Především pak návrh lokálních SIS byl proveden s mizivou koordinací jejich vzájemné spolupráce, což vedlo k rozdílným strategiím a řešením této spolupráce a k použití odlišných technologií. Na druhou stranu, jak i autoři uvádějí, tento nedostatek koordinace byl zároveň částečně užitečný, jelikož umožnil v praxi vyzkoušet a experimentovat s různými možnými řešeními, ověřit možnost jejich kombinace a díky tomu získat řadu zkušeností. [Bat02]

Právě organizační problémy bývají typické pro projekty tohoto rozsahu. Je otázkou zřejmě pro samostatnou práci, jaký vliv na to má stát, jeho byrokracie a další problémy vyskytující se mimo komerční sektor, což jsou typické vlastnosti znesnadňující vývoj podobně rozsáhlých systémů. Tento text je zaměřen na technickou stránku řešení a jeho cílem není analyzovat marketingové strategie a podobné aspekty projektů, ale ke zhodnocení projektu RUPA je třeba poukázat právě na fakta z těchto oblastí, protože právě zde lze spatřovat důsledky částečného selhání projektu, může-li to být tak pojmenováno.

Kritika v tomto bodě se týká především ekonomického modelu projektu RUPA, který byl nastaven tak, že značně znesnadňoval rozšiřovaní systémů v dalších letech. To bylo způsobeno některými administrativními omezeními, které například stanovily nutnost vypsání veřejné soutěže zvlášť na řešení projektu služeb v rámci jedné domény a zvlášť na řešení pro spolupráci mezi doménami. Další problém pak představovala výkonnost centrální sítě, jejíž model architektury nebyl dostatečně efektivní a připojováním dalších domén rostl součet nákladů na přístup k centrální části systému exponenciálně. V důsledku těchto problémů pak vznikly v některých regionech menší regionální sítě s cílem propojení některých systémů na lokální úrovni a tyto projekty byly budovány nezávisle na projektu RUPA. [Ros06]

Závěrem můžeme říci, že uvedené řešení prezentuje množství pozitivních i negativních poznatků, ze kterých lze čerpat při návrhu dalších řešení. Také je zde vidět důležitost zajištění širokého konsensu na vytvářeném integračním systému mezi všemi dotčenými stranami a potřebu existence centralizované instituce mající schopnost určit omezující podmínky pro integrované systémy. Přítomnost ekonomických tlaků je pak hlavním nepřítelem podobných projektů.

6 Srovnání analyzovaných řešení ISUK a RUPA

Cílem obou analyzovaných projektů ISUK a RUPA je propojení původních informačních systémů a umožnění jejich spolupráce nad množinou dat, která je pro jednotlivé skupiny obsažených IS společná. Oba projekty probíhají v organizacích, pro něž je charakteristická přítomnost řady oddělených a separátně pracujících systémů příslušejících do jisté míry autonomním jednotkám. Tato situace evidentně vede k existenci heterogenity na různých úrovních (viz 1.1.3), kterou musí integrační řešení překonávat. Prioritou pak je zprůhlednit nebo vůbec umožnit procesy, které by měly probíhat mezi původními systémy daných organizací, a poskytnout tak uživatelům kvalitnější práci s těmito systémy.

Základní rozdíl mezi řešeními spočívá v úrovni pohledu na integraci informačních systémů. Zatímco v projektu ISUK je tento problém nahlížen pouze z hlediska integrace dat rozšířených v původních IS, v projektu RUPA je integrace nahlížena mnohem obecněji. Zde je snahou vytvořit prostředí umožňující různorodou spolupráci systémů, které sice spadají pod odlišné organizační jednotky, ale jejich činnosti se vzájemně ovlivňují. Jedná se tak o integraci služeb a integrace dat může být brána pouze jako její specifická součást.

Z této charakteristiky vychází také odlišné provedení architektury obou řešení. V projektu ISUK je architektura postavena na centralizované komponentě, kterou lze v daném případě výhodně použít k dosažení tří hlavních cílů projektu (propagace dat, jejich uchovávání a zaznamenávání historie – viz 4.1). Potřeba tohoto centrálního prvku je dána požadavkem na centralizované uchovávání dat, který je dále do značné míry podmíněn nutností uchovávat jejich historii. K hlavnímu cíli, kterým je propagace dat do relevantních systémů, pak tato centralizace není zdaleka tak potřebná, čehož je projekt RUPA ukázkou. Zde je v rámci integračního systému pouze definován systém kooperačních prostředků, podle kterého integrované systémy reprezentované jednotlivými SIS mohou zveřejňovat rozhraní svých služeb a využívat takto nabízené služby dalších systémů. Celostátní centrální SIS zde nemá charakteristiku centrální řídící jednotky a jeho postavení je spíše na úrovni ostatních

lokálních SIS a komunikace může probíhat mezi lokálními SIS odděleně bez nutnosti jeho zapojení do tohoto procesu. Specifikum centrálního SIS pak spočívá v tom, že zahrnuje především hlavní (řídící) aplikace nadřazené aplikacím lokálních systémů a spravující největší množství dat a z velké části tak kooperace probíhá právě mezi ním a lokálními (podřízenými) systémy.

Vrátíme-li se k centralizovanému řešení projektu ISUK, nabízí se zde otázka potřeby jeho centrální komponenty. Ta by mohla být nahrazena de-centralizovanou architekturou pouze za předpokladu, že by nebylo požadováno fyzické uložení všech dat v centrálním systému. Pak by propagace dat mohla probíhat mezi dotčenými systémy přímo na základě některého z návrhových vzorů, který tuto problematiku řeší. Celkový (centralizovaný) pohled na data by pak mohl být realizován prostřednictvím pohledů nad daty v lokálních systémech (řešení GaV). V tomto případě však zůstává otevřena otázka uchovávání historie dat, která by musela být řešena interně v rámci jednotlivých integrovaných systémů nebo návrhem samostatné komponenty zapojené do replikačních procesů a logující prováděné operace. Takové řešení by se pak vyhnulo některým typickým problémům centralizovaných řešení (např. závislost na jediné komponentě, která je zranitelným bodem), ovšem na druhou stranu by bylo spojeno s jinými problémy jako je zmíněné uchovávání historie dat nebo způsob distribuce dat do všech zainteresovaných systémů. Celkově se tak zdá být centrální komponenta v projektu ISUK vhodně zvoleným prvkem využívajícím charakteristik systému.

Vedle toho se pak spíše otevírá možnost použití více servisně orientované architektury v projektu ISUK. Použité řešení vykazuje také některé její prvky, ale zdá se, že zvláště komunikační rozhraní mezi centrem a lokálními systémy by mohla být důsledněji a přesněji definována, aby byly jasné závislosti mezi jednotlivými komponentami integračního systému. Jeho architektura byla totiž navržena tak, že je příliš závislá na situaci dané konkrétními integrovanými systémy a na komponentách zajišťujících replikaci jejich dat, jejichž požadavkům bylo celkové řešení neustále upravováno. To je ze značné míry dáno experimentováním s novými technikami, ovšem přesto by tomuto stavu mohlo napomoci rozdělení integračních procesů probíhajících mezi centrem a lokálními systémy do více samostatných, nezávislých komponent, které by pomohly lépe oddělit specifika

konkrétních problémů od integračních mechanismů společných pro všechny takové systémy.

Svým způsobem lze nahlížet na problém řešený v projektu ISUK pouze jako podproblém (s určitými specifiky) komplexního řešení, tak, jak je adresováno v projektu RUPA. Zde se také řeší integrita dat, ale pouze jako jeden z případů problémů, které se v daném prostředí integrovaných aplikací vyskytují. V praxi jsou pak průběžně vytipovávány oblasti, kde je žádoucí zavést mechanismy řešící integritu dat v rámci konkrétních systémů, a na jejich základě jsou pak využity prostředky kooperativního prostředí k definování, zveřejnění a používání rozhraní pro řešení daného problému. Kromě replikace specifické množiny dat, se kterými pracuje více různých systémů, je pak dalším obvyklým integračním procesem v rámci UN například prosté zpřístupnění dat z cizího systému. Především je ale důležitá možnost spolupráce na provedení vícekrokové operace dotýkající se několika nezávislých systémů. To jsou integrační mechanismy, které v řešení projektu ISUK nejsou postihnuty.

Zaměříme-li se na problémy obou projektů, můžeme vidět poměrně zajímavou podobnost a shodu v celé řadě bodů. Co se týká technické stránky, zde je klíčové řešit kvalitu integrovaných dat a tato otázka nebyla ani u jednoho z projektů brána jako prioritní už od počátečního návrhu, ale její řešení bylo z velké části odloženo až na situaci, kdy bude nutno postavit se konkrétním problémům, které se objeví, jak tomu bylo v projektu RUPA, nebo se původní teoretické návrhy konceptů řešící otázky kvality dat (unifikace, relevance) ukázaly jako neadekvátní a nevyhovující v praxi. S kvalitou dat související unifikace záznamů je v obou případech adresována pouze s pomocí nejzákladnějších technik a použití sofistikovaných teoretických postupů zde nebylo nutné.

Zajímavý rozdíl pak vidíme v této oblasti při pohledu na priority těchto unifikačních algoritmů, kdy v případě projektu ISUK byl jako nepřípustná chyba definován stav, kdy jsou dva neodpovídající záznamy unifikovány, zatímco v projektu RUPA je naopak tímto nepřípustným stavem situace neunifikování záznamů, u kterých k tomu mělo dojít.

Dále se pak v projektu RUPA postupně měnilo zaměření na použité technologie a jejich původní výběr musel být záhy přehodnocen a s ním i představa o jejich

uniformním použití, což ovšem nebylo překážkou vzhledem k zakrytí technologických rozdílů samostatnými komponentami. Na druhou stranu v projektu ISUK podobné změny nebyly vynuceny a technologická realizace řešení odpovídající původním představám se ukázala jako vyhovující.

Rozhodující problém obou systémů lze ovšem spatřovat v oblasti jejich řízení. Při analýze obou zkoumaných řešení je patrné, že doplatily především na problémy s vedením dané jejich velikostí a především tím, že zasáhly do existence původních systémů. Díky tomu byly nuceny řešit řadu dalších otázek, které neúměrně komplikovaly jejich původní návrh. V důsledku toho tak nenaplnily všechna očekávání na ně kladená a to i přesto, že po technické stránce jde o pokrokové návrhy vytvořené odborníky se zázemím ve vědecké sféře.

Závěrem pak řekněme, že pro oba projekty je charakteristické jejich experimentování s novými technologiemi a hlavně technikami a koncepty tvorby SW systémů, což u obou systémů vede k potřebě formulování sjednoceného rámce pro architekturu integrující heterogenní aplikace, který by bral v úvahu komplexitu organizace, nevyhnutelnou přítomnost původních systémů a dostupné technologie [Mec99, Obd06]. Formulování takového rámce pak bylo neoddělitelnou součástí při vývoji projektů a obě řešení ve výsledku prezentují příklady architektury pokrývající jim odpovídající oblasti integračních problémů s řadou cenných poznatků.

7 Závěr

Analyzovaná řešení názorně ukázala problematičnost integračních systémů operujících v heterogenním prostředí IS, u nichž dochází k neustálému vývoji a není jasná doba jejich existence. To vše je znásobeno enormním technologickým vývojem a často hrozícími změnami v managementu, které vedou k rozsáhlému přehodnocování požadavků. Právě proto, a zkušenosti z minulosti to potvrzují, je nerealizovatelné budovat kolosální monolitické systémy. Místo toho je třeba mít na paměti základní pravidla SW inženýrství, například skutečnost, že nákladnost změny v systému stoupá s její hloubkou výskytu v procesu vývoje.

Proto je důležité, a u experimentálního řešení založeném na nových metodách to platí dvojnásob, aby integrační řešení bylo navrženo s co nejkompaktnějším jádrem a konkrétní požadavky byly realizovány v co největší míře pomocí samostatných komponent s jasně definovanou funkčností, které mohou být snadněji testovány. Vývoj celého systému by pak měl postupovat v iteracích, mezi nimiž je prováděno testování ověřující funkčnost jednotlivých komponent a tím i správnost navrženého konceptu. Díky tomu tak lze minimalizovat problémy spojené se změnovými požadavky, včas lze zachytit problematická místa, kde původní předpoklady nesplnily své očekávání, a je usnadněno přidávání nové funkčnosti. Kromě toho je také nutné co nejvíce oddělit implementační specifika jednotlivých integrovaných systémů od vlastního integračního řešení (nejlépe definováním samostatné vrstvy). Tato pravidla jsou široce známa a také oba analyzované projekty se jim snaží vyhovět, přesto i zde jsou vidět rezervy v jejich naplnění.

V rámci této práce byly zmíněny některé z možných přístupů k řešení komplexního problému integrace heterogenních informačních systémů; oblasti, která získává stále více na své důležitosti. Cílem bylo především podat shrnující úvod do problematiky, poukázat na hlavní problémy, kterým je nutno čelit, a prezentovat možná řešení a to zejména na konkrétních studiích z reálného světa. Zvláštní důraz byl při tom kladen na řešení Informačního systému Univerzity Karlovy a jeho architekturu se strukturou Datového Stohu. O tomto řešení si myslíme, že přichází s moderním a inovujícím pohledem na daný problém. Zatím ovšem nebylo uvedeno v kontextu dalších řešení a nemohlo tak být provedeno jeho srovnání s konku-

renčními přístupy. Snahou tedy bylo toto řešení tímto lépe charakterizovat. Kromě toho pak je součástí práce analýza řešení Sjednocené sítě italské administrativy jako doplněk k prvnímu řešení, jelikož ukazuje další širší možnosti integrace systémů.

Nutno podotknout, že prezentované řešení problému integrace IS zdaleka nepředstavují kompletní seznam možných přístupů a v poslední době se objevuje řada projektů adresující tuto problematiku z různých úhlů pohledu. Pro ně je ovšem charakteristická podobná množina přístupů, které jsou přítomny ve zmíněných dvou řešeních. Dále je pak typický značný příklon k technologiím XML (např. [Nach02]). Především se ale jedná o velmi širokou oblast, kde se jednotlivé výzkumy zaměřují na konkrétní specifická řešení a v podstatě neexistuje sjednocující teorie (přestože za snahu o to lze považovat například práci [Bus99]). Věříme, že tímto může být práce vodítkem při návrhu nových systémů a rozcestníkem pro další zkoumání oblasti.

Reference

[Bag98] Bagatin F., Batini C., Massari A., Osnaghi A. (1998): The Cooperative Architecture of the Italian Government. Dostupné on-line na adrese http://www.omg.org/~soley/italian-govt.doc

[Bat02] Batini C., Cappadozzi E., Mecella M., Talamo M. (2002): Cooperative Architectures: The Italian Way Along e-Government. Elmagarmid A.K., McIver Jr W.J. (eds.): *Advances in Digital Government: Technology, Human Factors and Policy*. Kluwer Accademic Publishers, 2002.

[Bed03] Bednárek D., Obdržálek D., Yaghob J., Zavoral F. (2003): Synchronisation of Large Heterogenous Data Using DataPile Structure. *3rd Workshop on Theory and Practice of Information Technologies (ITAT 2003)*, 113-121, Sliezsky dom, Slovakia.

[Bed05] Bednárek D., Obdržálek D., Yaghob J., Zavoral F. (2005): Data Integration Using DataPile Structure. *9th East-European Conference on Advances in Databases and Information Systems (ADBIS 2005)*, 178-188, Tallin, Estonia.

[Ber01] Bertolazzi P., Scannapieco M. (2001): Introducing Data Quality in a Cooperative Context. *6th International Conference on Information Quality (ICIQ)*, 431-444, Boston, MA.

[Bus99] Busse S., Kutsche R.-D., Leser U., Weber H. (1999): Federated Information Systems: Concepts, Terminology and Architectures. Technical Report, Nr. 99-9, TU Berlin.

[Cha03] Channabasavaiah K., Holley K., Tuggle E. Jr. (2003): Migrating to a service-oriented architecture, Part 1. Dostupné on-line na adrese http://www-128.ibm.com/developerworks/webservices/library/ws-migratesoa/

[Coh98] Cohen W. W. (1998): Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. *1998 ACM SIGMOD international Conference on Management of Data*, Seattle, WA. Tiwary A., Franklin M. (eds.): *SIGMOD 98*, 201-212. ACM Press, New York, NY.

[Dem97] De Michelis G., Dubois E., Jarke M., Matthes F., Mylopoulos J., Papazoglou M. et al. (1997): Cooperative Information Systems: a Manifesto. Papazoglou M., Schlageter G. (eds.): *Cooperative Information Systems: Trends & Directions*, 315-363. Accademic-Press, 1998.

[Dok05] Dokulil J., Yaghob J., Zavoral F. (2005): Evoluce replikačních algoritmů v stohově orientovaných systémech. 5th *Workshop on Theory and Practice of Information Technologies (ITAT 2005)*, 393-401, Račkova dolina, Slovakia.

[Gar97] Garcia-Molina H., Papakonstantinou Y., Quass D., Rajaraman D., Sagiv Y., Ullman J., et al. (1997): The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, vol. **8**, no. **2**, 117–32.

[Her98] Hernadez M. A., Stolfo S. J. (1998): Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Journal of Data Mining and Knowledge Discovery*, vol. **1**, no. **2**, 9-37.

[Len02] Lenzerini M. (2002): Data integration: a theoretical perspective. *21st SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002). PODS 02*, 233-246. ACM Press, New York, NY.

[Lev96] Levy A. Y., Rajaraman A., Ordille J. J. (1996): Querying Heterogeneous Information Sources Using Source Descriptions. *Proceedings of the 22th international Conference on Very Large Data Bases*. Vijayaraman T. M., Buchmann A. P., Mohan C., Sarda N. L. (eds.): *Very Large Data Bases*, 251-262. Morgan Kaufmann Publishers, San Francisco, CA.

[Mec00] Mecella M., Batini C. (2000): Cooperation of Heterogeneous Legacy Information Systems: a Methodological Framework. *4th International Enterprise Distributed Object Computing Conference (EDOC 2000)*, Makuhari, Japan. *IEEE Computer Society*, 216-225.

[Mec01a] Mecella M., Batini C. (2001): The Unitary Network and the Cooperative Architecture: a Review of the First Projects. Technical Report, Department of Systems and Computer Science, University of Rome, La Sapienza.

[Mec01b] Mecella M., Batini C. (2001): Enabling Italian E-Government through a Cooperative Architecture. *Computer*, vol. **34**, no. **2**, 40-45.

[Mis03] Missier P., Lalk G., Verykios V., Grillo F., Lorusso T., Angeletti P. (2003): Improving Data Quality in Practice: a Case Study in the Italian Public Administration. *Distributed and Parallel Databases*, vol. **13**, no. **2**, 135-160.

[Myl97] Mylopoulos J., Papazoglou M. (1997): Guest Editor's Introduction: Cooperative Information Systems. *IEEE Expert: Intelligent Systems and Their Applications*, vol. **12**, no. **5**, 28-31.

[Nach02] Nachouki G., Quafafou M.: Extracting, Interconnecting, and Accessing Heterogeneous Data Sources (2002): An XML Query Based Approach. *First Eurasian Conference on information and Communication Technology*. *Lecture Notes in Computer Science*, vol. **2510**, 442–449. Springer-Verlag, London, 2002.

[Nau99] Naumann F., Leser U., Freytag J. Ch. (1999): Quality-driven Integration of Heterogeneous Information Systems. *25th VLDB Conference*, 447-458, Edinburgh, Scotland.

[Obd06] Obdržálek D., Kulhánek J. (2006): Generating and handling of differential data in DataPile-oriented systems, *24th IASTED International Multi-Conference on Databases and Applications*, 7-12, Innsbruck, Austria.

[Orr98] Orr K. (1998): Data quality and systems theory. *Communications of the ACM*, vol. **41**, no. **2**, 66-71.

[Park04] Park, J., Ram S. (2004): Information systems interoperability: What lies beneath?. *ACM Transactions on Information Systems (TOIS)*, vol. **22**, no. **4**, 595-632.

[Ros06] Rossi V., Marino J. (2006): Report on the role of IXPs and e-government in Italy. Dostupné on-line na adrese http://www.cocombine.org/pdf/D29.pdf.

[Sca04] Scannapieco M., Virgillito A., Marchetti C., Mecella M., Baldoni R. (2004): The DaQuinCIS architecture: a platform for exchanging and improving data quality in cooperative information systems. *Information Systems*, no. **29**, vol. **7**, 551-582.

[She90] Sheth A. P., Larson J. A. (1990): Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, vol. **22**, no. **3**, 183-236.

[Sto05] Stonebraker M., Abadi D. J., Batkin A., Chen X., Cherniack M., Ferreira M., et al. (2005): C-store: a column-oriented DBMS. *31st international Conference on Very Large Data Bases*, Trondheim, Norway. *Very Large Data Bases*, 553-564. VLDB Endowment.

[Tay98] Tayi G. K., Ballou D. P. (1998): Examining Data Quality. *Communications of the ACM*, vol. **41**, no. **2**, 54-57.

[Ull97] Ullman J. D. (1997): Information Integration Using Logical Views. *6th international Conference on Database theory*. Afrati F. N., Kolaitis P. G. (eds.): *Lecture Notes In Computer Science*, vol. **1186**, 19-40. Springer-Verlag, London.

[Wie92] Wiederhold G. (1992): Mediators in the Architecture of Future Information Systems. *Computer*, vol. **25**, no. **3**, 38-49.

[Zha05] Zhao H., Ram, S. (2005): Entity identification for heterogeneous database integration: a multiple classifier system approach and empirical evaluation. *Information Systems*, vol. **30**, no. **2**, 119-132.