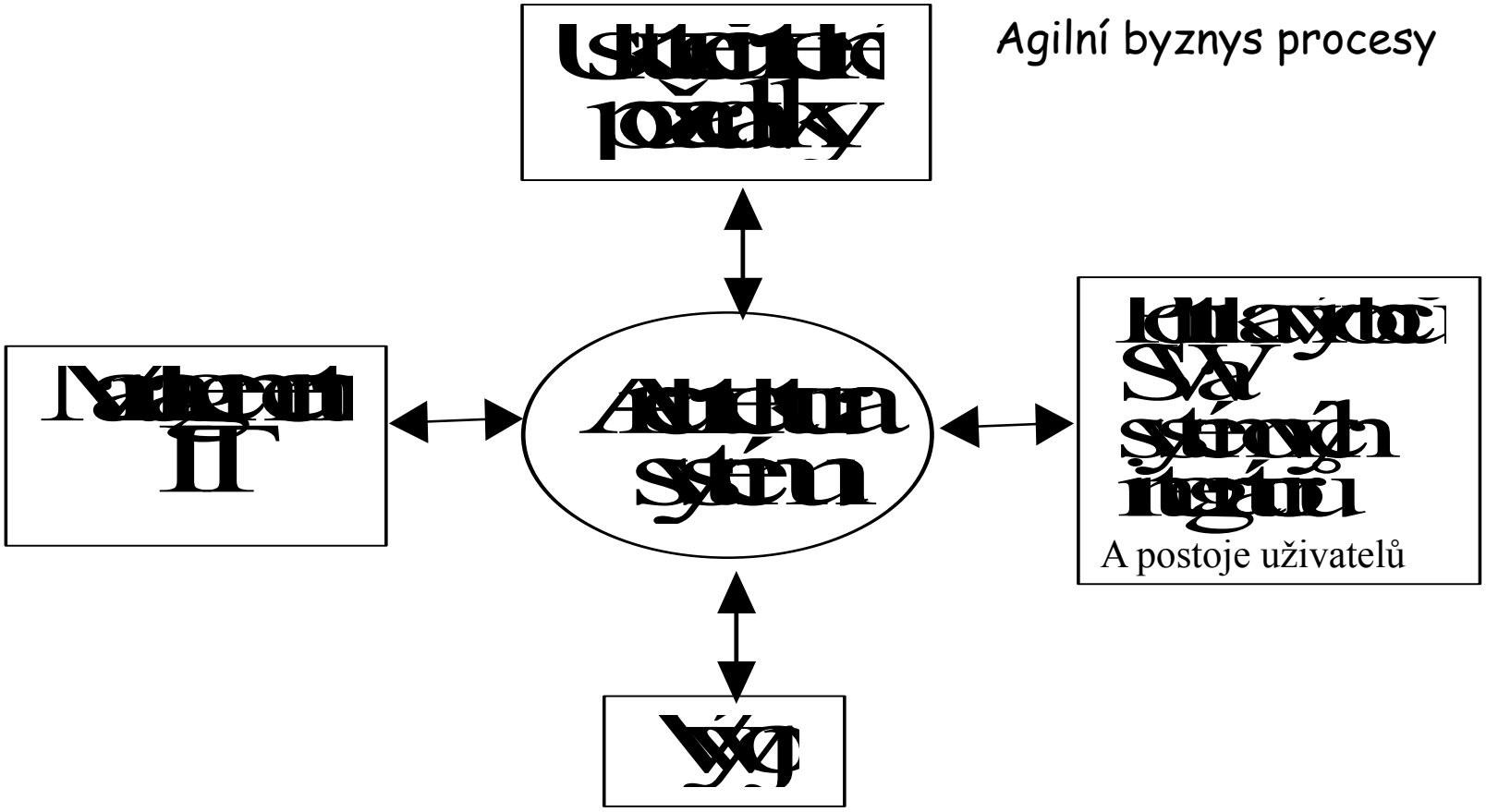
Softwarové architektury

SW struktury ve velkém

Logické a technologické struktury SW systémů

Klíčová role architektury



Co umožňují SW architektury

- Umožňuje agilitu i při vývoji velmi velkých systémů (vývoj pomocí změn konektorů a pravidel jejich využívání) a zapojení SME do vývoje
- Snížení pracnosti vývoje
  - Vyvinout N malých autonomních SW artefaktů délky *x* méně pracné než vývoj jednoho délky N*x, zrychlení je* $N^{-1/8}$, i když vše píši znovu,
    - Jsou-li komponenty autonomní lze je snadno koupit či znovu použít koupit nebo převzít, **to je hlavní přínos!!!!!**
  - *Je možný inkrementální vývoj (je dříve vidět, jak to bude fungovat, snadné prototypování,*
  - Snazší údržba a inkrementální výměna komponent
  - Umožňuje velmi velké systémy

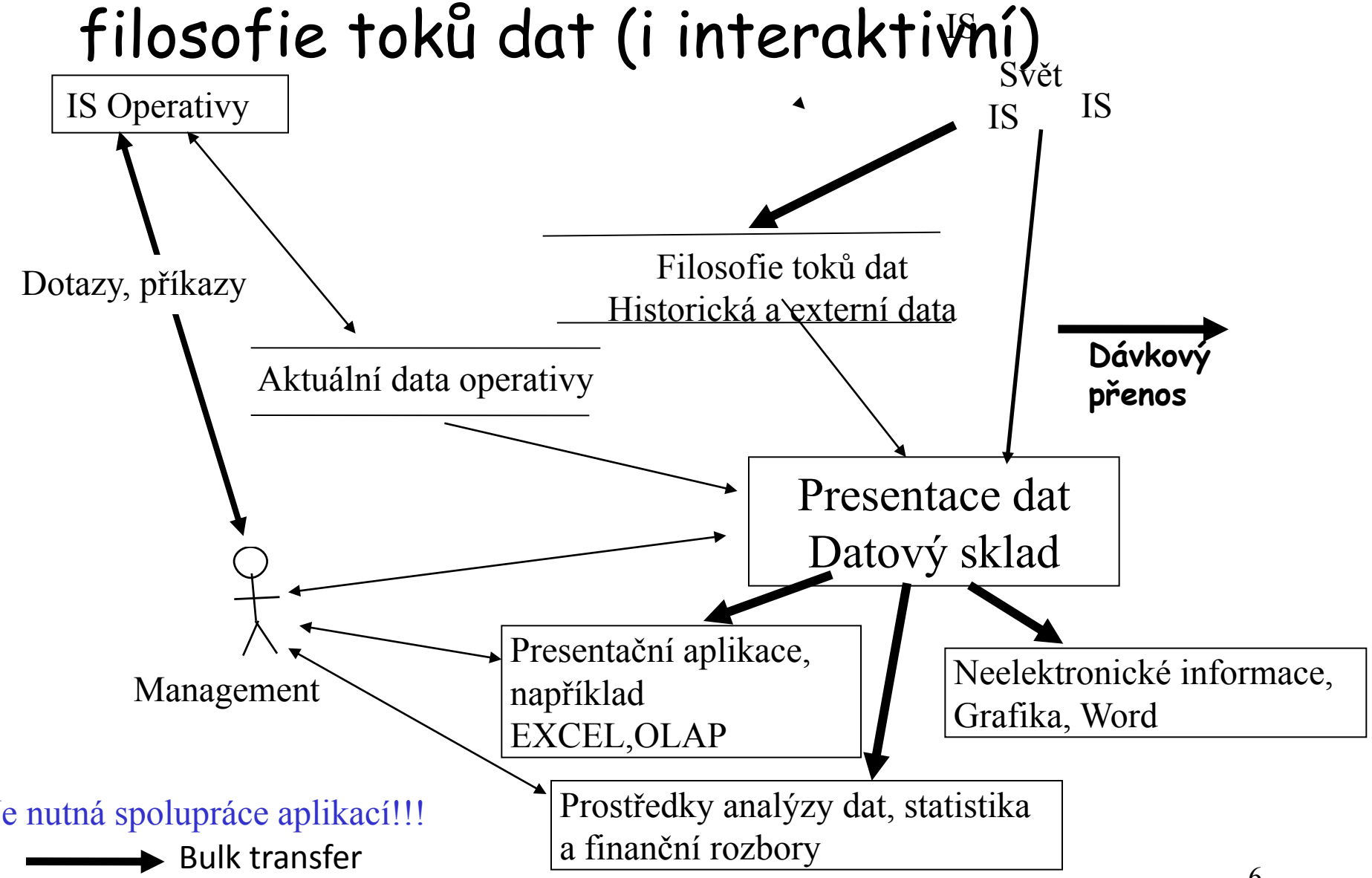
Pohled na architekturu SW (mírně zjednodušeno)

1. Podprogramy,
2. Objekty: Skupiny podprogramů se společnými daty
3. Komponenty: propojené skupiny objektů či komponent
4. Aplikace: Funkčně propojené skupiny komponent
5. Systémy: Spolupracující soubory (sítě)aplikací případně i systémů s různou mírou formalizace (varianty:katedrála, město nebo bazar či favela) a s různou mírou autonomie

Dávkové systémy

- Systém je tvořen souborem dávkových aplikací, které transformují kolekce vstupních dat (soubory) do kolekcí výstupních dat (výstupních souborů)
- Modelem jsou diagramy toků dat
- Formální model je založen na přístupu
  - Resource
  - Execution
  - Output
  - Agent

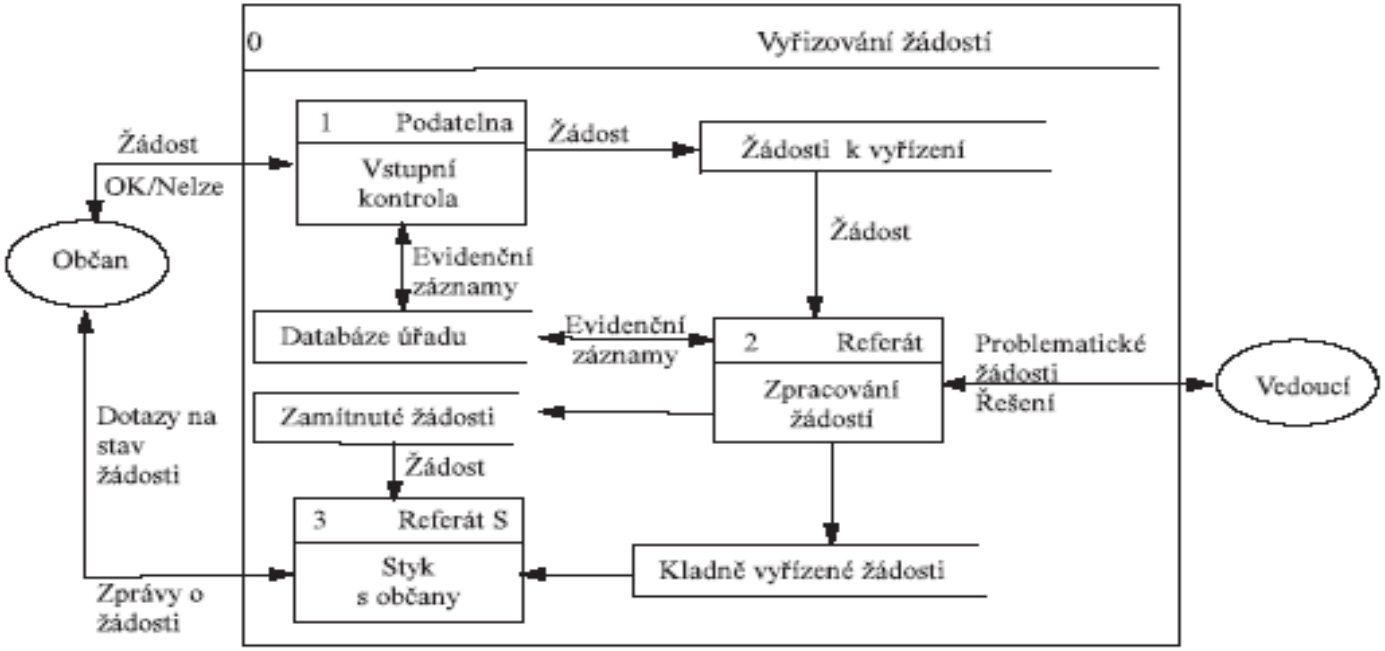
Manažerský IS, příklad aplikace filosofie toků dat (i interaktivní)



Pozorování

- Spolupráce SW komponent může být dávková
  - exportem a importem dat ob (bulk transfer)
  - pomocí společného (virtuálního) datového úložiště plněného dávkově, je-li třeba
- Spolupráce může být interaktivní, primárně asynchronní
  - Přímá výměna zpráv
  - Aktivní databáze, případně s uloženými zprávami
- Některé služby mohou být dávkové (Excel)
- Často lze dávkový výpočet zapouzdřit jako službu, rozhraní je pak v podstatě zobecněný příkazový řádek

Příklad diagramu toků dat



Obr. 12.9: Diagram toků dat systému Vyřizování žádostí.

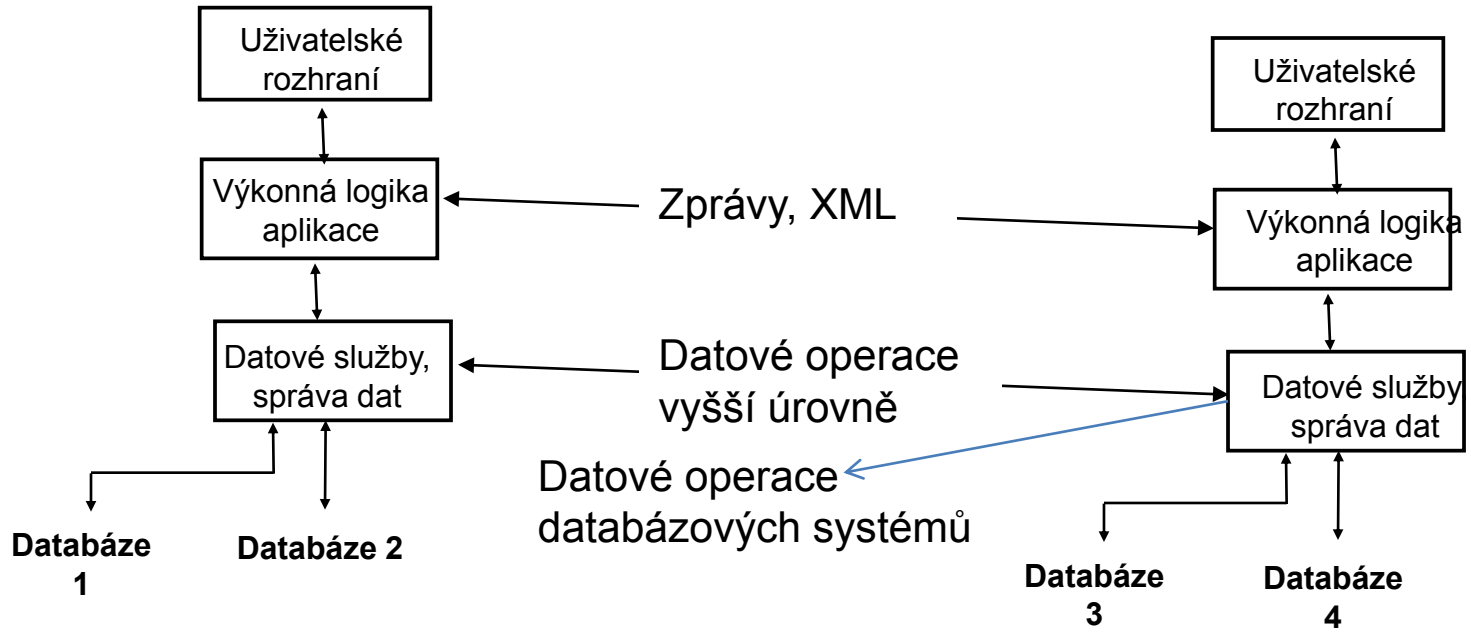
V klasickém přístupu přenos souborů, zobecnění bulk transfer

Role částí v on-line aplikacích postupně i v systémech

- Klientské funkce (hlavní úkol zajištění uživatelského rozhraní)
- Výkonná logika (server)
- Datová část (databáze)

Spolupráce třívrstvých komponent

- Spolupráce podle vrstev,
  - Logika je na aplikačním serveru, datová na datovém

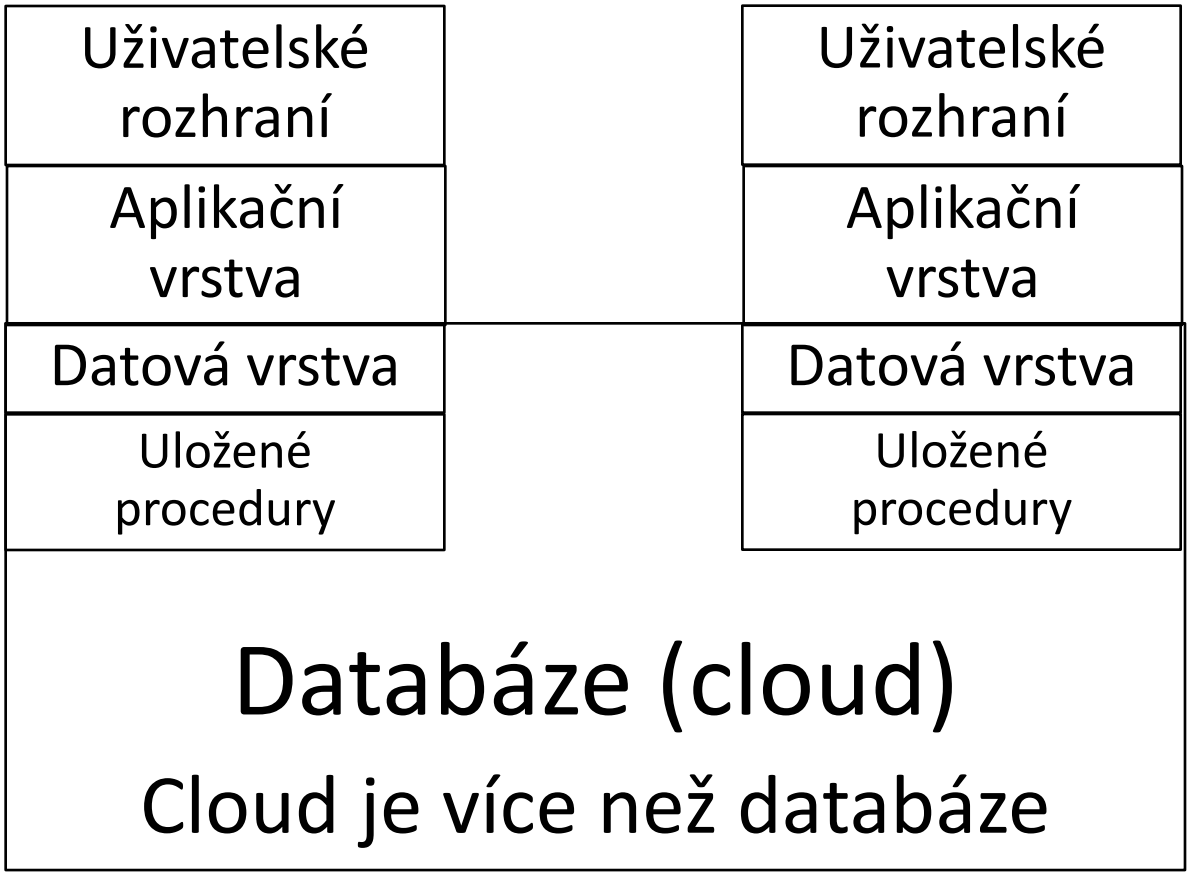


V SOA mohou být vrstvy tvořeny podsítěmi (pod SOA)
Komunikace mže být asnchronní

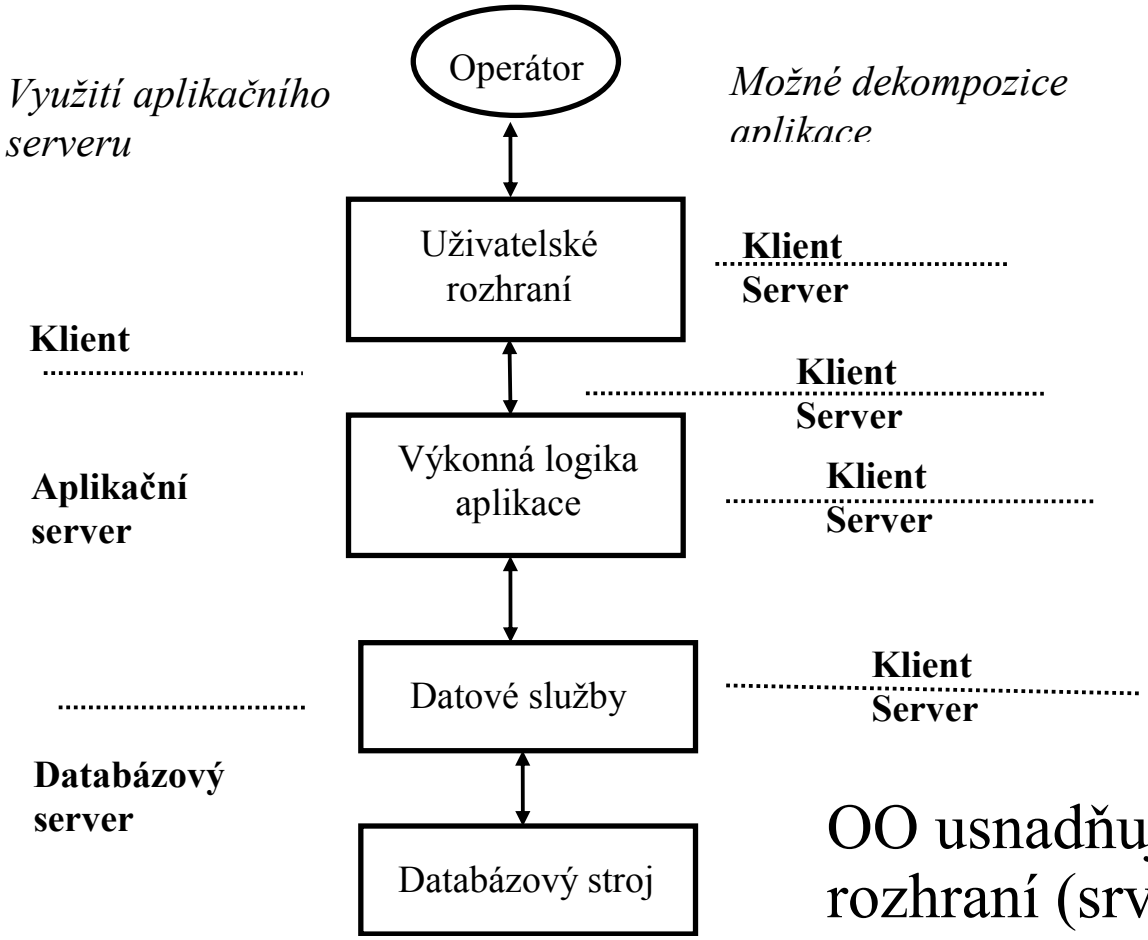
Databázově orientované systémy

- Aplikace pracující nad stejnou DB
- Aplikační vrstva zčásti pomocí uložených procedur
- Nutná disciplina při vývoji, lze pak vytvořit systém, který se do značné míry obejde bez middlewaru (ten je nahrazen službami databáze)

Systémy propojené přes databázi

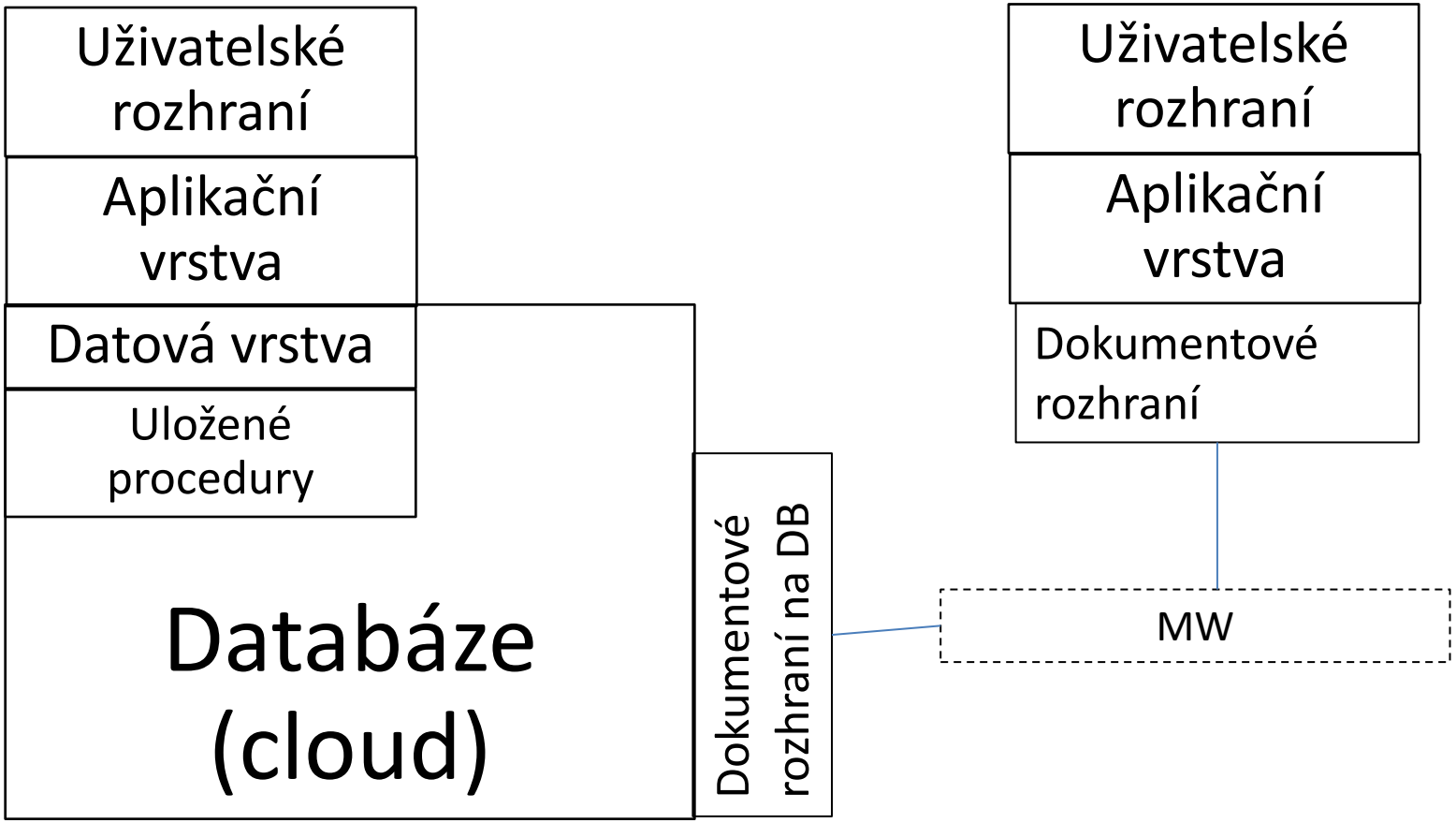


Tři vrstvy a server

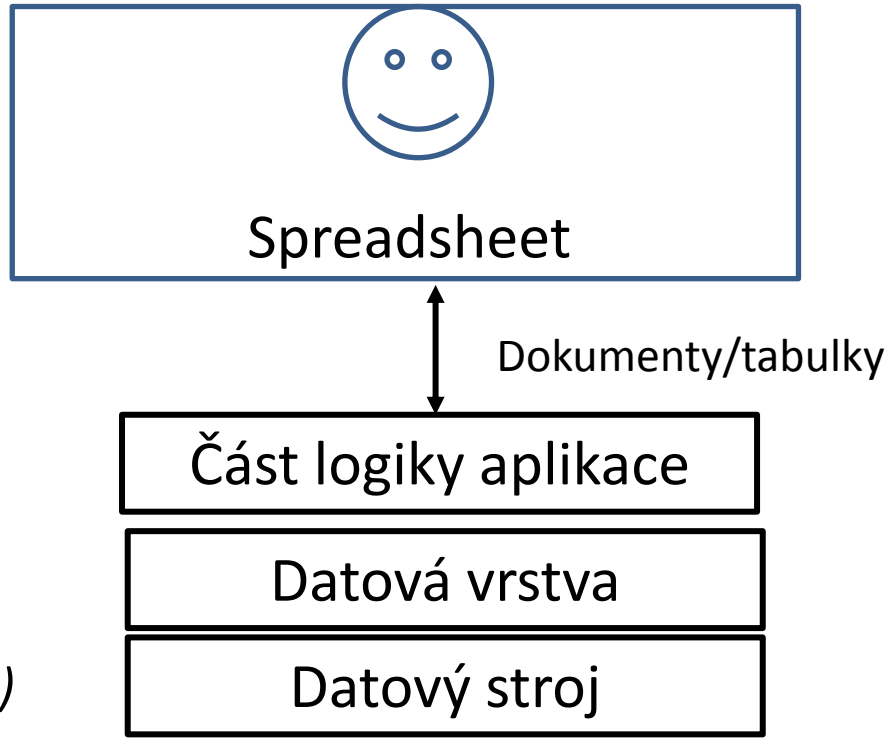


OO usnadňuje posun rozhraní (srv. connectors)

Systémy propojené přes dokumentové rozhraní databáze



Spreatsheet jako tlustý klient

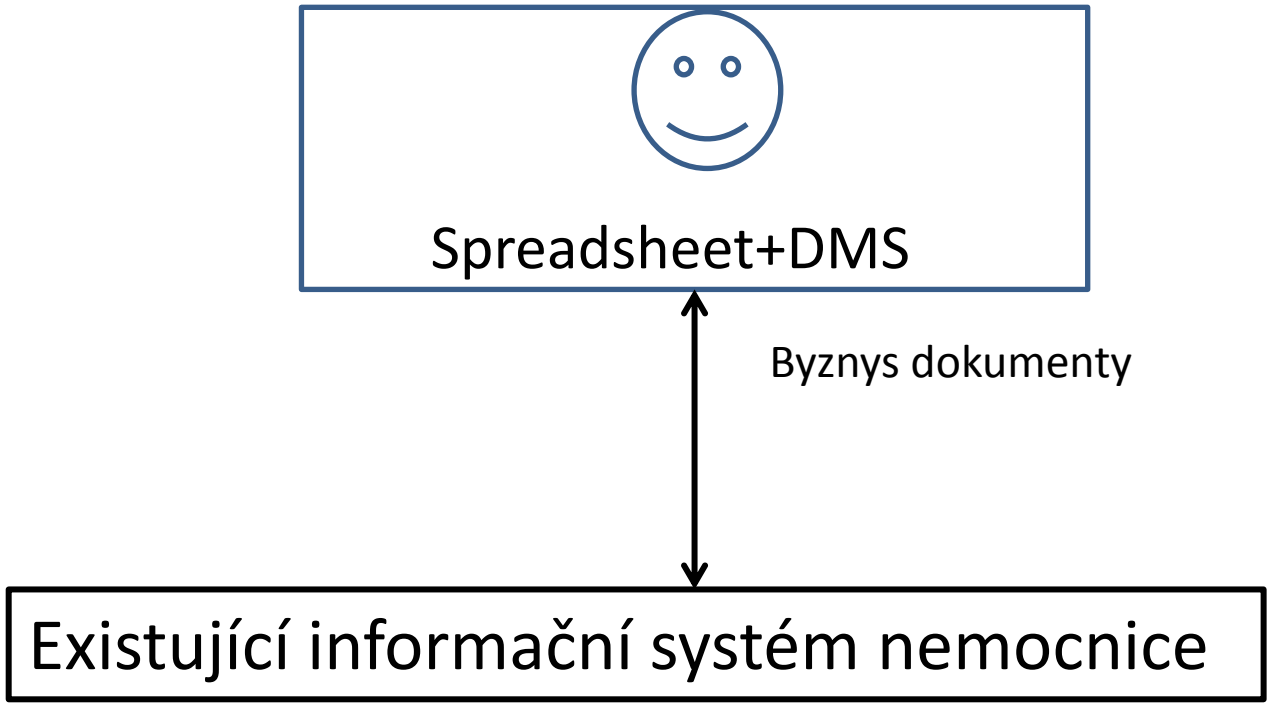


Může být správa dokumentů (DMS)

Dokumenty lze používat v různých optimalizacích u výkonových pracovníků i pro manažery
Slabé místo: nespolehlivost Excelu

Jak to dopadlo

- Dva subjekty komunikující pomocí byznys dokumentů v podstatě p2p protokolem



Je běžné, že informační systémy komunikují s jinými informačními systémy.

Informační systémy mohou fungovat jako systémy automatického řízení (avionika letadla), dávat rozkazy lidem, být (dočasně) nahrazeny lidmi, řídit procesy reálného světa (tam je nutno řešit problém dostupnosti dat a problém včasnosti odezvy) a také nová formulace toho, co nazýváme transakce

Je běžné, že informační systémy komunikují s jinými informačními systémy.
Ty je možné zapouzdřit tak, aby se chovaly jako služby ve smyslu SOA (peers v p2p systému)
To je výhodné pro byznys procesy prováděné více subjekty.
Vyžaduje to komunikaci pomocí standardních byznys dokumentů (smluv, faktur, dodacích listů, atd).
Technicky je to velmi podobné řešením z eGovernmentu
Je to výhodné použít i pro komunikaci komponent uvnitř podniku (back office, autonomní oddělení podle Bati)

Problém

Problémem je, že servisní orientace je zdánlivě samozřejmá.
Jde ale o nové paradigma, **zvládnutí je obtížné**
Paradoxně ji právě proto nevěnujeme dostatek pozornosti a dokonce se domníváme, že to není nic nového, a nejsme ji proto schopni-ochotni používat, ačkoliv se stává vedoucí filosofií současného softwaru. Ten se stává síťový z logického hlediska a reálnému světu podobný z uživatelského hlediska
Cloud computing a gridovské systémy zvyšují použitelnost a výhodnost SOA
SOA berou velké SW firmy jako značku svých výrobků, používají často normy OASIS skupiny
Protestují, když se SOA používá v poněkud jiném smyslu

Mnohé je známo po dlouhou dobu, celek a principy je však obtížné uplatnit, Je pro to třeba:

Změnit marketing (jak od předsudků osvobodit, jak změnit pravidla spolupráce), změnit management
Použít existující komponenty (legacy sys.)
Změnit metody specifikace požadavků
Zajistit dostupnost operací , jako je prototypování, decentralizace, sourcing,…
Umožnit nové postupy na straně uživatelů i výrobců SW (SaaS - SW jako služba, decentralizace, otevřenost), cloudy
Překročit hranice OO

Servisně orientovaná architektura (SOA) v našem smyslu

- Virtuální síť p2p síť (velkých) volně vázaných komponent, spolupráce komponent je podobná spolupráci služeb v reálném světě,
  - Komponenty jsou zpravidla permanentně aktivní procesy, asynchronnost komunikace
  - Komponenty mohou být i dávkové systémy, např. Excel či OLAP, excel může tvořit jádro klentské části
  - Existují i jiné komponentové přístupy, servisní je asi v informačních systémech nejužitečnější

Hlavní oblasti aplikace servisně orientované architektury

Techniky využívané v SOA se značně liší podle oblastí nasazení a způsobu dodávky, typické příklady:

1. Řízení technologií, hlavně varianta soft real-time
2. Systémy, kdy je nutná agilita při vývoji i používání, sourcing, a znovupoužívání a integrování systémů
   - malé a střední podniky, e-government, použití ve výzkumných projektech, obvykle velké komponenty a uživatelsky orientovaná komunikace
   - **pravděpodobně hlavní směr příštího rozvoje SOA, datové schránky**
   - **Systém se buduje hlavně integrací, tedy v podstatě zdola**

Hlavní oblasti aplikace servisně orientované architektury

3. Systémy pro velké podniky na klíč
   - Silně standardizovaná řešení, OASIS norma SOA reference architecture, v menších IS Open Group SOA reference architecture
   - Spousty (malých) služeb, komunikace pomocí vzdáleného volání procedur, jemnozrnná komunikce
   - Budování podle celkového plánu (shora, podobné klasickému vývoji customizací)
   - Pracné! Systémy mají tendence být velké a propojené s jinými dynamicky měnitelnými systémy, proto tendence k přístupů nutným po maslé a střední podniky (zdola)

Náš přístup – varianty implementace

- P2p síť autonomních komponent s rozhraním podobným „rozhraní“ služeb reálného světa
  - Základní je asynchronní komunikace a předávání dokumentů (srv. datové schránky)
  - Podobnost se zvyky reálného světa může být vzdálenější nebo užší, žádoucí je dokumentová orientace, přenos byznys dokumentů
- Ne vždy se SOA takto chápe, požadují se další vlastnosti resp. dodržování norem

SOA se na považuje za klíčovou a již vyzrálou technologii

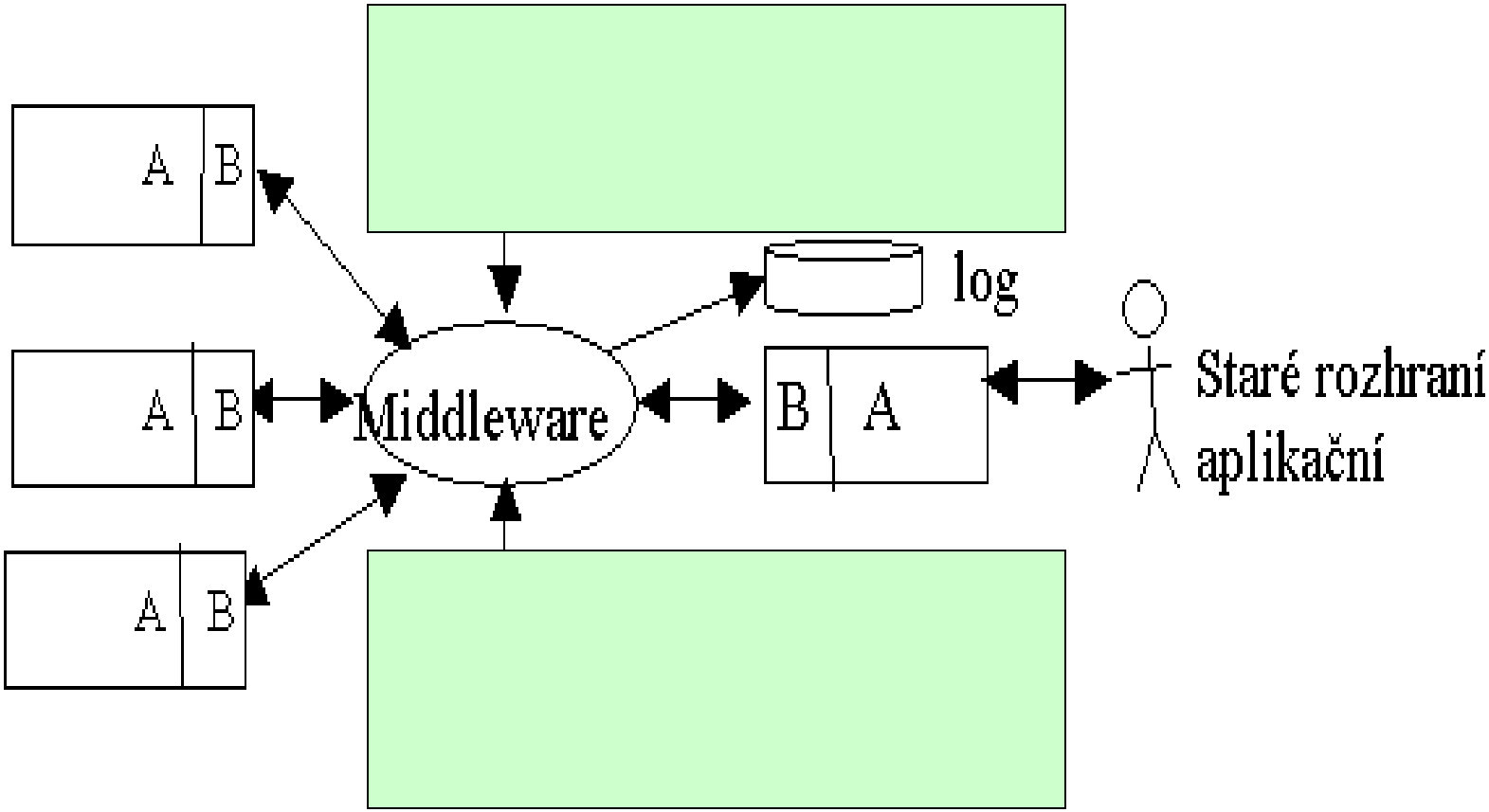
Viz polohu na SOA hype křivce v r. 2009 podle analytiků Gartner Group

Mnoho úspěšných použití (viz např. Progress Software, IBM, dokonce SAP)

Značné investice do SOA výrobců SOA i uživatelů

Ale: Stále nové normy a doporučení (W3C, OASIS, Open Group), servisně orientované ekkosystémy

Struktura jednoduché SOA



A – aplikační služba, B – její rozhraní (primární brána) UR je uživatelské rozhraní, např. portál

Staré aplikační rozhraní je u komplexnějších služeb nutností (viz IS úřadů), usnadňuje to podstatně i vývoj.

Servisně orientovaná architektura v intuitivním smyslu

- Virtuální p2p síť (velkých) volně spolupracujících komponent,
  - Z hlediska uživatelů je výhodné, aby se komponenty v informačních systémech chovaly podobně jako služby reálného světa
    - Mnohdy je žádoucí uživatelsky (byznys) orientované rozhraní nejlépe obchodních služeb (rozhraní jako u reálných služeb, - výměna obchdních), je to obvykle výhodné i pro cloudy, musí se to ještě prozkoumat
    - Takové rozhraní je založeno na jazycích používaných ve znalostních doménách uživatele.
    - Mohou to být byznys dokumenty jako faktury, objednávky atd.

Servisně orientovaná architektura
Varianta vhodná pro intuitivní přístup

- Komponenty je výhodné implementovat jako permanentně aktivní procesy, chovají se jako služby v reálném životě, jsou to SW služby
- Služby komunikují způsobem, který je možné chápat jako výměnu zpráv-dokumentů
- Midleware musí být schopen tento požadavek splnit

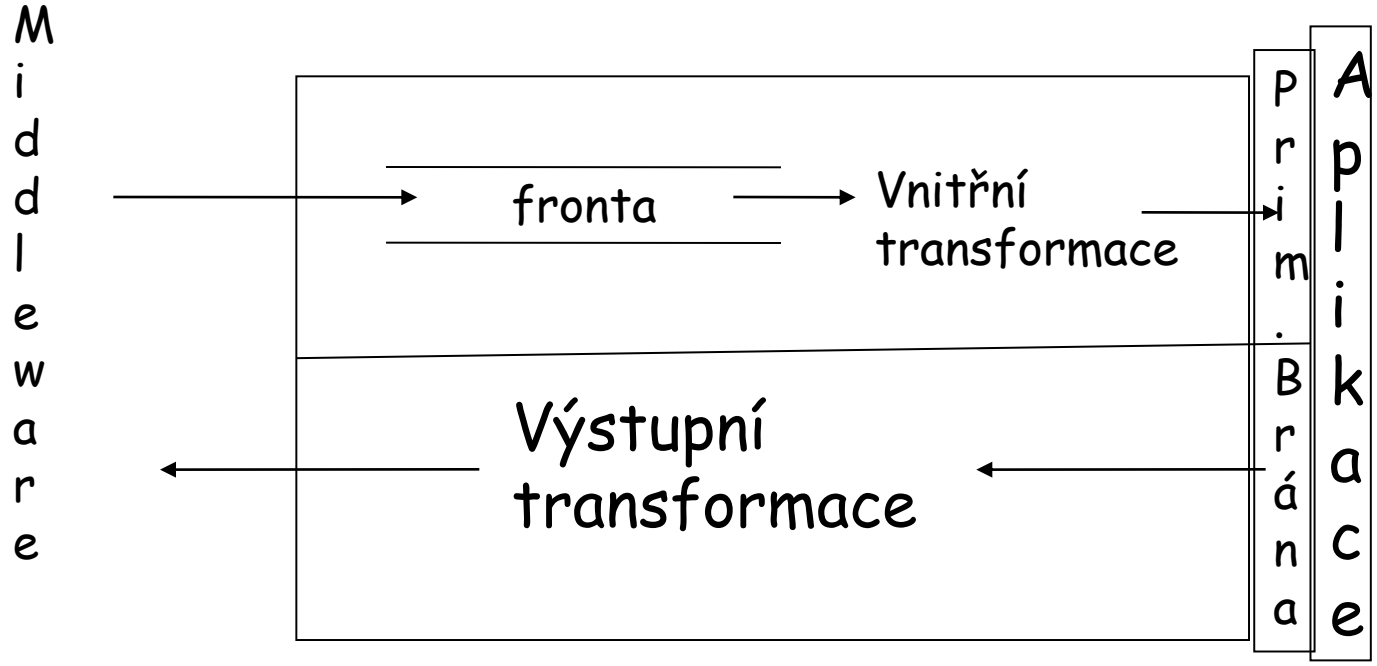
Servisně orientovaná architektura v intuitivním smyslu

- V tom není úplná shoda a to způsobuje problémy
  - Uživatelská orientace rozhraní služeb omezuje standardizaci a aplikaci obratů typických pro objektový vývoj
  - Přehnaná standardizace může být pro takový přístup pastí (antivzor standardizační paralýza)
  - Zesiluje chápání SOA jako procesu integrace (Open Group)

Techniky middleware

- Mohou být různé a mohou se kombinovat v jednom systému
  - Prostředky (distribuovaného) OS, maibox, messagequeue
  - Prostředky komunikace mezi systémy, různé úrovně komunikace, dnes hlavně webové služby se SOAP dokument encoding,architekturní služby – viz níže
  - Cloud, společné tendence k využívání document management systems a EDA - even driven systems, lze v nich implementovat virtuální přenos zpráv
- Zaměříme se na výměnu zpráv – je to transparentní po nás i uživatele,lze implementovat i v cloudech

Struktura brány pro zapouzdření aplikace jako služby



Pokud middleware nabízí jen point-to-point mezi aplikacemi je třeba zavést nový typ služeb. Řešení je možné přes specializované (architekturní) služby. V českém e-govermentu je možné jako bránu použít datovou schránku

Struktura brány

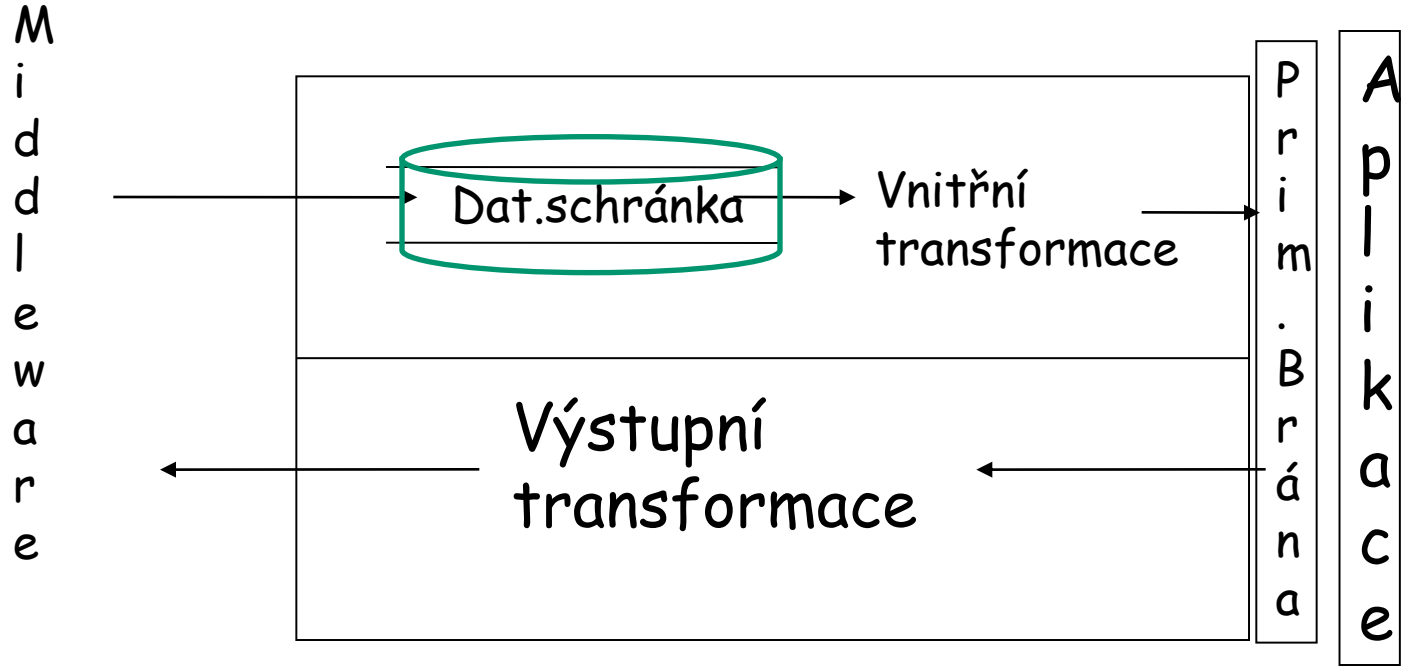
Při implementaci lze použít dotNet, primitiva UNIXu, MQ firmy IBM, nebo použít nějaký ESB….

Lze využít konektor ve smyslu komponentových architektur.

Moderní systémy nabízejí modernější řešení (publish-subscribe, …). To lze implemetovat pomocí služby data store ukládající zprávy nebo pomocí Document management system

Jiná možnost je použít datové schránky ve státní správě jako architekturní služby

Brána s datovou schránkou



Brána může být značně komplikovaná, schránka je vlastně SW službou, výstupní transformaci a adresaci může provádět aplikace

Sítě dávkových služeb jako předchůdce SOA

- Sítě dávkových služeb (např. strukturovaná analýza a návrh) tak, jak je zobrazovaly diagramy toků dat, byly v lecčems předobrazem SOA.
  - Autonomie komponent, komunikace přes datová úložiště
- Známé výhody dávkových systémů
  - Modifikovatelnost, levný vývoj
  - Stabilita (viz zkušenosti s Y2K a systémy psanými v Cobolu)
  - Škálovatelnost
  - Znovupoužitelnost
- Je často nutné dávkové aplikace integrovat do SOA
  - K tomu stačí, aby dávkové aplikace se zbytkem systému komunikovaly přes datové úložiště a obvykle pomocí dávkového (bulk) přenosu dat

Některé ideje DFD použity v konceptu REA, pracuje se na vylepšení

Řízení technologií

- Nejstarší oblast používání principů SOA, především ve variantě soft real-time; od sedmdesátých let
  - Soft real time (odpovědět včas, vyjímečné opoždění není v zásadě chyba, jen nepříjemnost, př. Počítač s terminály), časti hard hard real time
  - OS a inteligentní periferie, to je hlavně soft
  - U hard real-time (odpověď vždy v termínu) jsou dodatečné nároky a použití SOA v nich nebývá vždy možné, pokud kritickou část neřeším v autonomních částech HW a SW
- Rozhraní služeb nemusí být v technologiích uživatelsky orientováno
- Proprietální řešení komunikace komponent není výjimkou
- Obvykle celkem malý počet služeb
- Komponenty nejsou zpravidla rozsáhlé
- Agilita použití je omezena
  - Běžný uživatel nemá skoro žádnou možnost principy práce technologie měnit a ani tomu nerozumí
  - I ti co mají povolenu agilitu musí být odborníci a i ti pak mají zpravidla jen omezené možnosti měnit chování systému
  - Nelze aplikovat postup pokus-omyl, např. tak jak je obvyklé u agilních form vývoje

Systémy, kdy je nutná agilita, sourcing, a znovupoužití - konfederace

- Typické pro malé a střední podniky a některá řešení e-government, např. propojování úřadů
- Je nutné použít legacy a produkty třetích stran
  - bývají velké, jejich změny jsou drahé a nejsou na ně peníze,
  - existují legislativní omezení,
  - rozhraní má být srozumitelné pro uživatele (neškolené v IT),
  - požaduje se snadnost outsourcingu, …
- Rozhraní musí umožňovat agilní změny byznys procesů používajících služby v SOA (rychlá reakce na změny)
- Tlak na outsourcing resp. využívání cloudů
- Využívání existujícího know how je nutností
  - Realistické nároky na zaškolování koncových uživatelů
  - Snazší záběh a hladší provoz
  - Nízké náklady a možnost postupné i agilní implementace

Systémy, kdy je nutná agilita, sourcing, a znovupoužívání: vlastnosti řešení

- Služby, které jsou partnery v komunikaci, se nemusí obvykle vyhledávat, protože jsou známi a nemusí se měnit, nebo se vyhledávají prostředky mimo systém (ručně)
  - Rozhraní služeb lze dohadovat ad hoc, partneři v komunikaci jsou „prověření“ , normy výhodou – ne nutností
  - Služby nemusí být (webové služby) ve smyslu norem
    - Pokud se používají webové služby, nemusí nutně vyhovovat všem normám, jak je prosazuje OASIS w3c a jiní. Webová služba je pak intuitivně spíše libovolná funkcionalita dosažitelná přes web.
    - Složitost norem může implikovat složitost používání a také nežádoucí tlak na změnu osvědčených postupů a použití jemnozrnných služeb což zhoršuje vlastnosti systému
- Používání norem je věcí dohod, případně legislativy.

Systémy, kdy je nutná agilita, sourcing, a znovupoužití: vlastnosti řešení 2

- Služby poskytující uživatelské funkce jsou většinou velké, často legacy, často je nutné omezené využívání norem, do chodu služeb mohou zasahovat uživatelé on-line,
  - Nejen klasická výměna zpráv a web,
    - Využívání datových úložišť a úložišť zpráv, probereme později
  - Zprávy a obvykle i služby mají být uživatelsky orientované a sémanticky bohaté a tedy hrubozrnné (coarse grained), výměna dokumentů,
  - Někdy je nutné v komunikaci používat datová úložiště (pro integraci dávkových systémů, a pro implementaci sofistikovaných variant komunikace), dokonce i datové proudy
  - Lze u databázově orientovaných systémů použít i uložené procedury

Systémy, kdy je nutná agilita, sourcing, a znovupoužívání (malé a střední podniky, e-government …)

- Je nutné použít legacy (jsou velké, změna je drahá a nejsou na ni peníze, je nutné vyhovět legislativě, rozhraní služeb má být srozumitelné pro neškolené uživatele, snadnost outsourcingu, použití toho, co funguje, …)
- Rozhraní musí umožňovat-usnadňovat agilní změny procesů
- Je žádoucí využívat existující know-how

Systémy pro velké podniky na klíč, byznys partner se zpravidla vyhledává

Hlavní výhoda pro uživatele

– Malá pravděpodobnost krachu projektu v důsledku volby osvědčeného dodavatele, je to i alibi pro management uživatele

Princip řešení a jeho výhody pro dodavatele

- **Mnoho malých služeb a použití standardů, aby se dosáhlo univerzality**

Nevýhoda

- Pro malé firmy (uživatelé a většinou i dodavatelé) drahé, obtížně zvládnutelné, potřeba měnit byznys procesy

Výhody malých služeb z hlediska velkého dodavatele, nevýhody pro malého uživatele

- Zákazník se obvykle stane závislý na dodavateli
  - antivzor *vendor lock in* a do jisté míry i *fine-grained services*
- Dají se využít (zlo)zvyky a dovednosti objektového vývoje
- V principu velmi silné, ale pracné na vývoj i údržbu (viz asembler kontra Java)
  - Další příčina závislosti na dodavateli
- Velmi často založené na webových službách ve smyslu OASIS a w3c (antipattern *SOA=Webservices*) a standardech (*Standardization paralysis)*

Systémy pro velké podniky na klíč (partner se zpravidla vyhledává automaticky),

Hlavní nevýhody pro uživatele

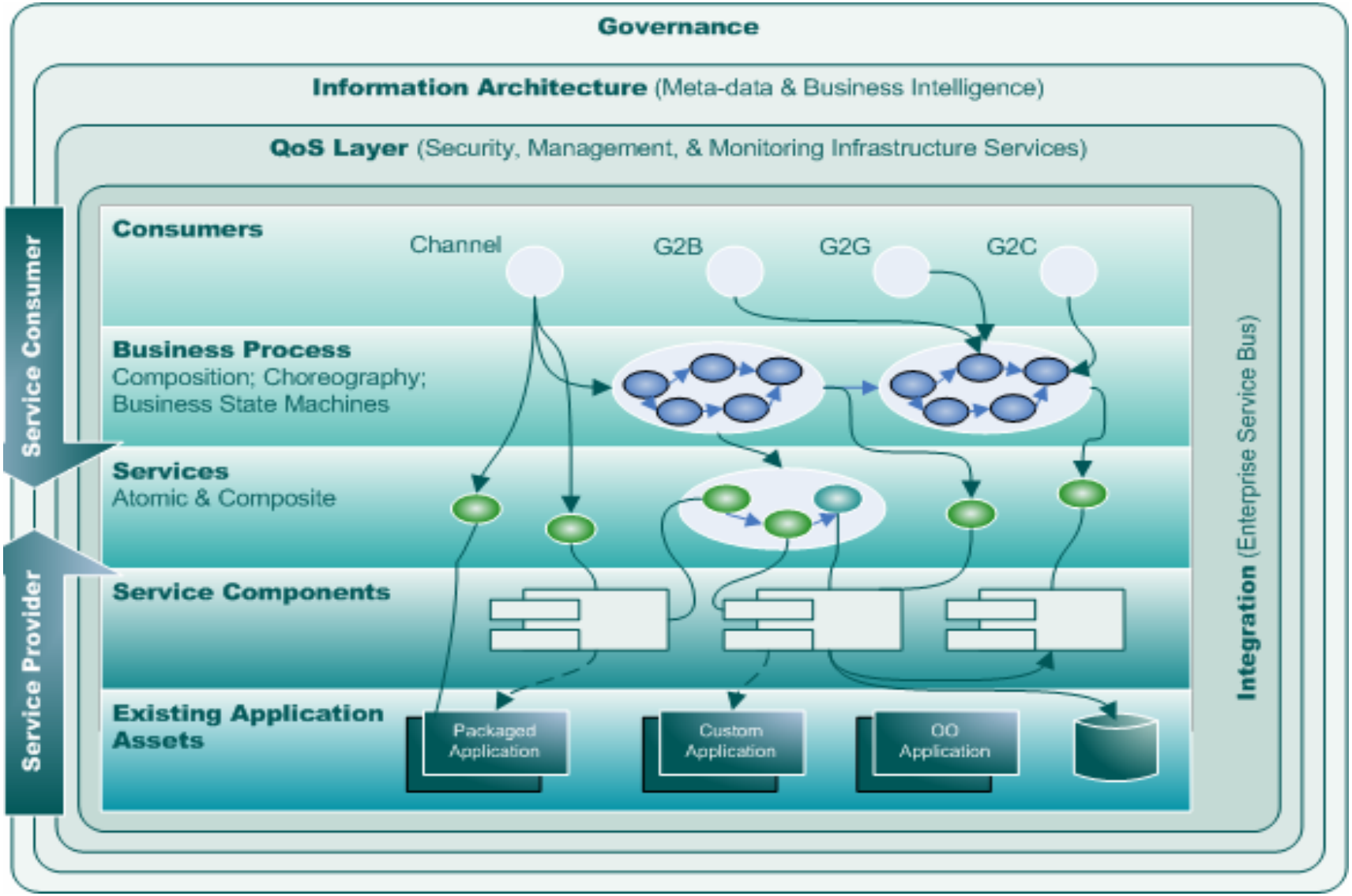
- Rozhraní služeb lze modifikovat jen za drahé peníze a za příliš dlouho a i pak bývá nepříliš uživatelsky příjemné
  - Problémy s agilitou byznys procesů
- Od velkých pro velké:
  - Pro malé podniky nevhodné nejen pro nákladnost a závislost na dodavateli, ale také proto, že neodpovídá jejich kultuře
  - V e-governmentu ten problém, že se zpravidla musí propojovat velké celky (IS celých úřadů)
    - Technicky lze provést pomocí datových schránek
  - Prostor pro různé standardy, které jsou ale natolik komplikované, že je menší podniky nemohou samy použít, a drahé, rychle zastarávají
    - Musí se spolehnout na dodavatele a jejich knihovny a jiná jejich řešení
- Tento typ SOA se z marketingových důvodů ztotožňuje se SOA vůbec a jsou na něj časté stížnosti, hlavně od malých firem

SOA pro velké podniky podle standardů OASIS

- According to the OASIS SOA-RM specification, SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. The SOA-RM specification bases its definition of SOA around the concept of "needs and capabilities", where SOA provides a mechanism for matching needs of service consumers with capabilities provided by service providers.
- Pozor, SOA umožňuje vývoj velkých sytémů

Návrh podle OASIS

http://dts.utah.gov/techresearch/techarchitecture/resources/soara081407.pdf



Vlastnosti

- Služby jemnozrnné
- Převažuje volání služeb jako procedury
- Vznik služeb jako reakce na událost
- Obtíže s budováním hierarchie
- Spíše pro velké

Odkazy

Arsanjani, Ali, and Jorge Diaz, *SOA Reference Architecture: Concepts and Usage*, IBM: SOA Center of Excellence, July 25, 2007.

Bennett, Stephen, *Best Practices and Patterns from the field for SOA Governance*, Open Group Architecture Practitioners Conference, July 23-25, 2007, Austin Texas.

Bringing SOA Value Patterns to Life, Oracle White Paper, June 2006.

Dico, Awel, *Addressing Enterprise Business Needs through SOA and TOGAF*, Open Group Architecture Practitioners Conference, July 23-25, 2007, Austin Texas.

Gejnevall, Matt, *How EA and SOA Connect*, Open Group Architecture Practitioners Conference, July 23-25, 2007, Austin Texas.

High, Rob, Jr. et al, *IBM's SOA Foundation: An Architectural Introduction and Overview.* Version 1.0, November, 2005. http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-soa-whitepaper.pdf

Keen, Martin et al, *Patterns: SOA Foundation -Business Process Management Scenario,* Redbook, http://www.redbooks.ibm.com/redbooks/pdfs/sg247234.pdf

Odkazy

———————————, *Patterns: SOA with an Enterprise Service Bus,* Redbook, http://www.redbooks.ibm.com/redbooks/pdfs/sg246494.pdhttp://www-128.ibm.com/developerworks/library/ws-model7

Reference Model for Service Oriented Architecture 1.0, Committee Specification 1, 2 August 2006, Copyright © OASIS Open 2005-2006. All Rights Reserved.

Sides, Jim, *Applying Patterns, Platform Intelligence and Products to ESB Deployments*, IBM, Open Group Architecture Practitioners Conference, July 23-25, 2007, Austin Texas.

Sudarsan, Sridhar, *Adopting SOA - Aligning the business and IT*, IBM, Open Group Architecture Practitioners Conference, July 23-25, 2007, Austin Texas.

The Actionable SOA: Patterns in SOA, Applied Technology Solutions, Open Group Architecture Practitioners Conference, July 23-25, 2007, Austin Texas.

TOGAF 2006 Edition (Incorporating 8.1.1), The Open Group, August 2006.

Utah GovPay: The Official Payment Solution for Utah Government (Terchnical Manual), Utah Interactive: Salt Lake City, 2006.

Příklad uplatnění

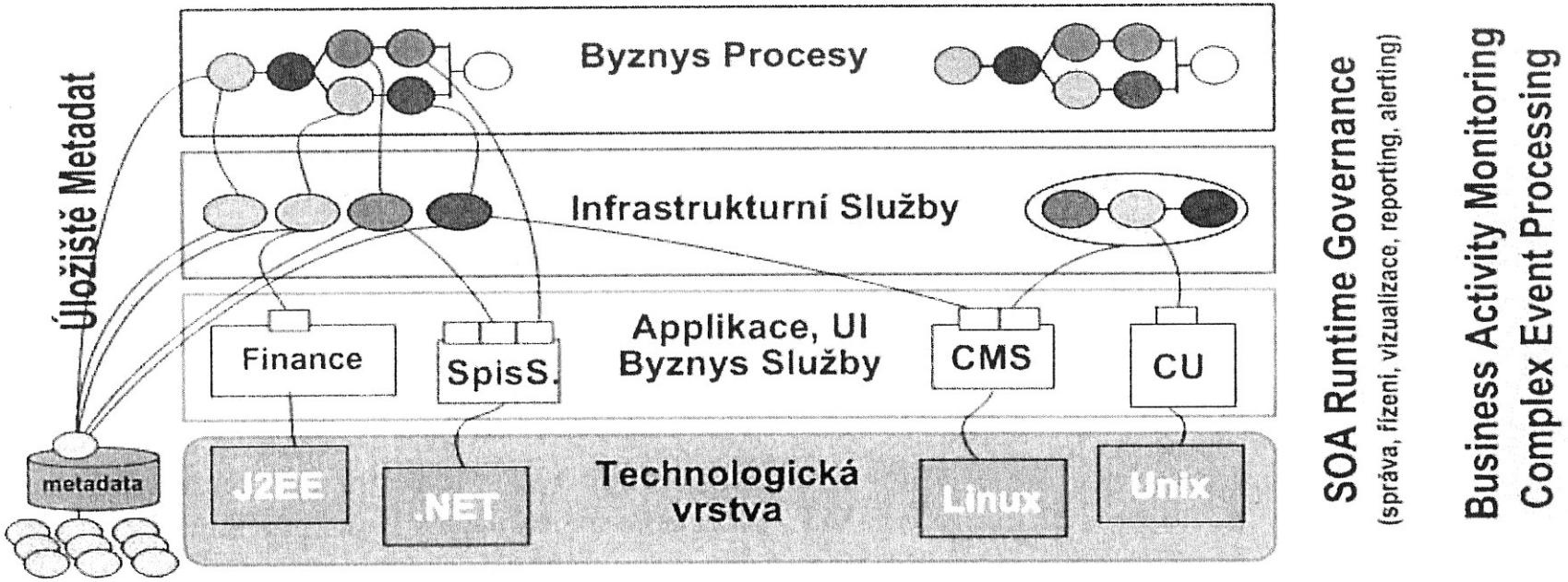
Systém sběru recepisů a kontroly výdeje léků, primárně těch zneužitelných pro výrobu pervitinu

Předpokládaná úspora miliarda až dvě ročně a zabránění škodám na zdraví, později i zefektivnění léčby

ÚOOÚ fungující systém zakázal používat

Velké ztráty a ohrožení základních lidských práv

Logický návrh



Obr. 2 Použitý referenční model SOA společnosti Progress Software.

Dodavatelé


O2 Business Solutions
PROGRESS
SOFTWARE
Aquasoft
PUREdata
PUREbusiness
netprosys
O2 Business Solutions

Deset vrstev - asi nic pro malé

- I relativné malý projekt dělalo několik velkých hráčů
- Nedostatečné prostředky pro využívání hotového a spojování IS po celém světě
- Využívání obecně používaných prostředfků
- Malé zkušenosti s údržbou

Příklad SOA – IS globálního podniku

- Relativně autonomní komponenty proměnné velikosti
- Často systém dodaný na klíč
- Střední úroveň otevřenosti
- Systém tvořen relativně velkým počtem komponent,
  - Tendence k jemnozrnosti
- Často napojení na e-komerci
- Uživatelské rozhraní komponent je možné, není nutně a nebývá proto používáno
- Podpora strojové byrokracie
- ESB se používá často

Příklad SOA – IS menšího až středního podniku

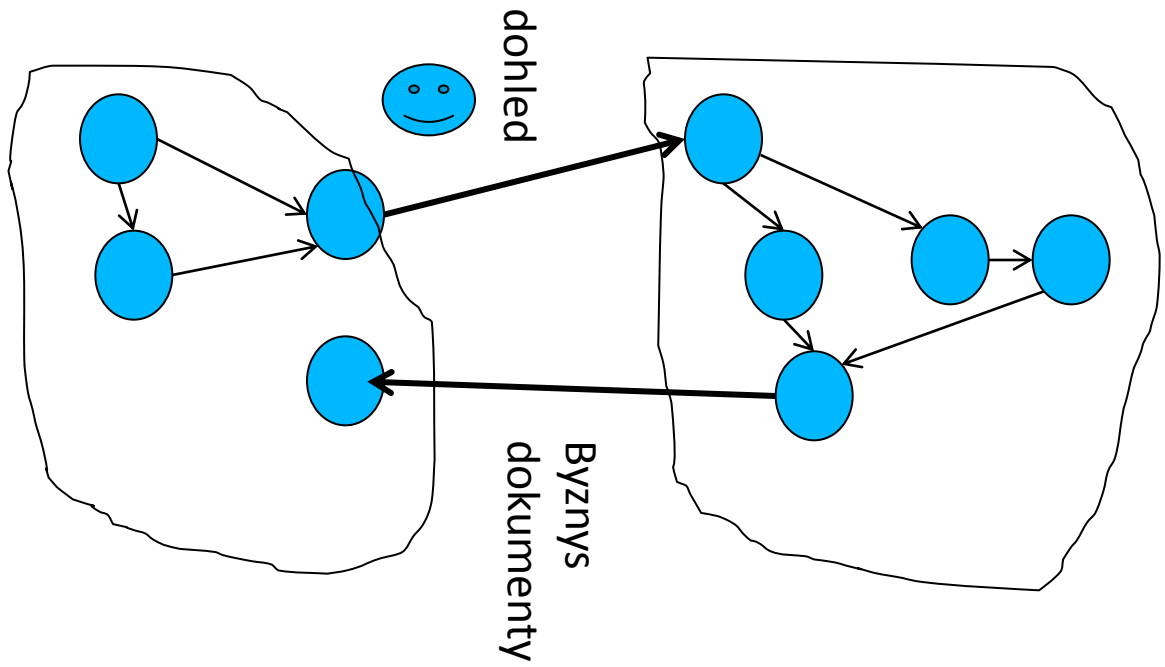
- Relativně autonomní komponenty proměnné velikosti
- Často integrace komponent z různých zdrojů, nakoupené i vyvíjené, nutná integrace legacies
- Střední úroveň otevřenosti
- Jádro tvořeno relativně menším počtem větších komponent, často zapouzdřené legacy
- Uživatelské rozhraní komponent prakticky nutné
- Různorodí uživatelé, není jich mnoho, střední úroveň znalostí IT
- Podpora strojové byrokracie s prvky demokracie
- ESB spíše ne (cena)

Příklad SOA – IS menšího až středního podniku

- Zvláštnosti byznys procesů
  - Nutné časté změny, agilita
  - Změnám musí rozumět uživatelé, často je musí i provádět
  - Procesy zpravidla zahrnují i více subjektů
    - To implikuje používání byznys dokumentů (smlouvy, dodací listy, faktury,…)
    - Navázání spolupráce zpravidla domlouvají uživatelé

IS menšího až středního podniku

- Používání byznys dokumentů (smlouvy, dodací listy, faktury,…)
- Navázání spolupráce zpravidla domlouvají uživatelé



Příklad SOA – soft real-time

- Relativně autonomní komponenty
- Často integrace komponent z různých zdrojů
- Nižší úroveň otevřenosti,
- Jádro tvořeno relativně menším počtem komponent, napojení na e-komerci obvykle chybí
- Uživatelské rozhraní komponent není prakticky nutné
- Různorodí uživatelé, není jich mnoho, nemají zpravidla přístup k rozhraní komponent
- Fine grained rozhraní služeb založené na konceptu volání (vzdálených) procedur/metod
- ESB spíše ne
- Praktické použití – řízení procesů

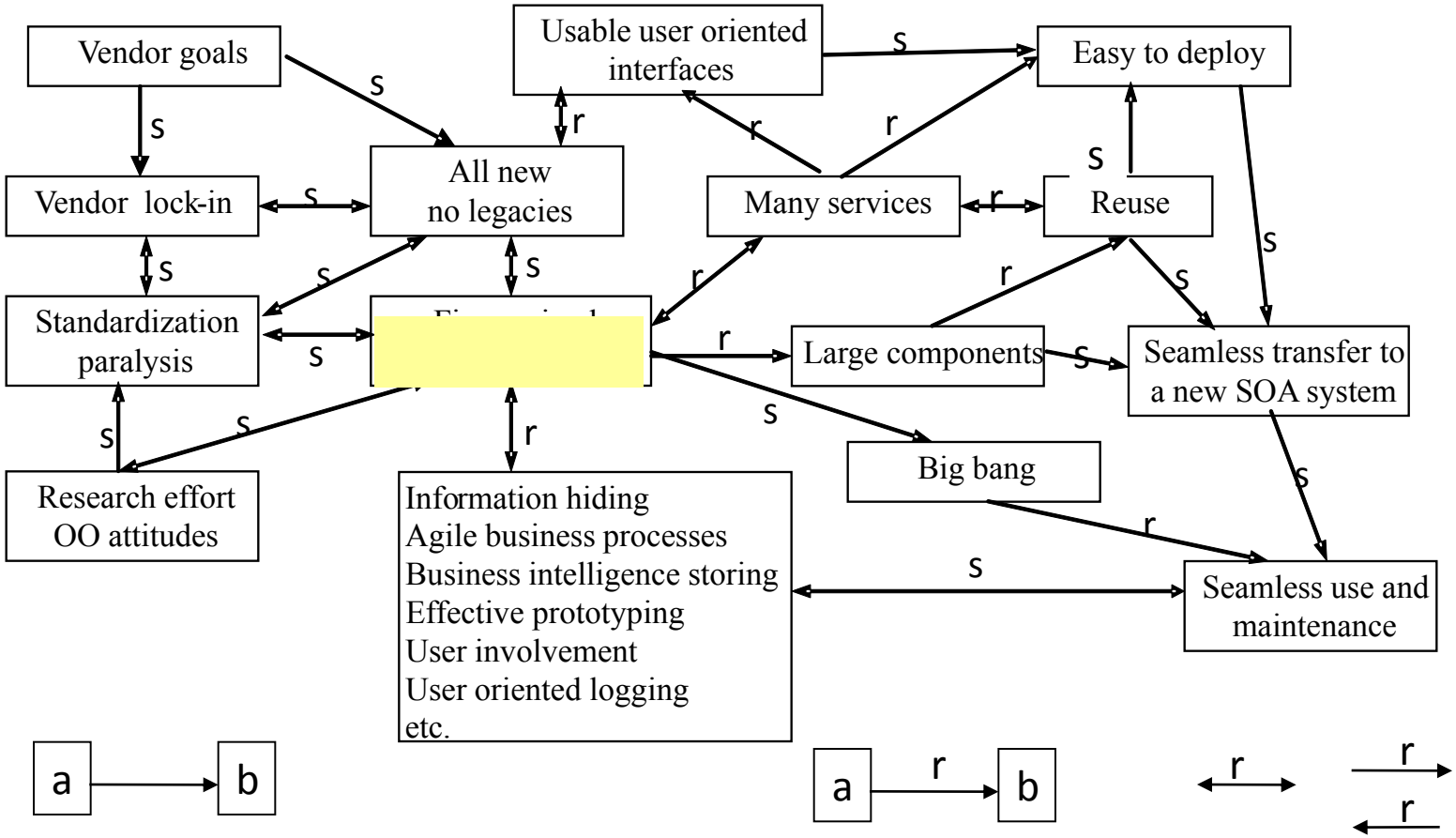
Problém fine grained services v SOA

- Jsou obvyklé u velkých dodavatelů a u webových služeb
  - Je jich mnoho, jsou spíše malé
- Do značné míry se vzájemně podmiňují s normami (normy je do jisté míry vyžadují, samy fine grained services jsou vhodné pro standardizaci)
- Z pohledu SME indukují mnoho nevhodných vlastností a praktik
- Mají tendenci používat RPC (i když to z norem neplyne, je to spíše díky nevhodnému používání praxe z objektově orientovaného vývoje)
  - To je pro uživatele nepříjemné
    - Zhoršuje to možnosti agility a spoluúčasti uživatelů při specifikaci požadavků a nadměrně zvyšuje počet služeb

Potvrzeno některými stížnostmi na SOA

- Kdo se má v spoustě služeb vyznat, cena je vysoká
- Velké náklady správu služeb (service government)
- Menší úspěch SOA u malých firem a někdy ve státní správě.
- Nevyhovuje daným potřebám
- Změny metodou velkého třesku

Undesirable consequences of object thinking (RPC + fine grained interfaces)



If a is getting stronger, b also does if a is getting weaker, b also does

If a is getting weaker, b is getting stronger if a is getting stronger, b is getting weaker

N^2 formátů a service bus

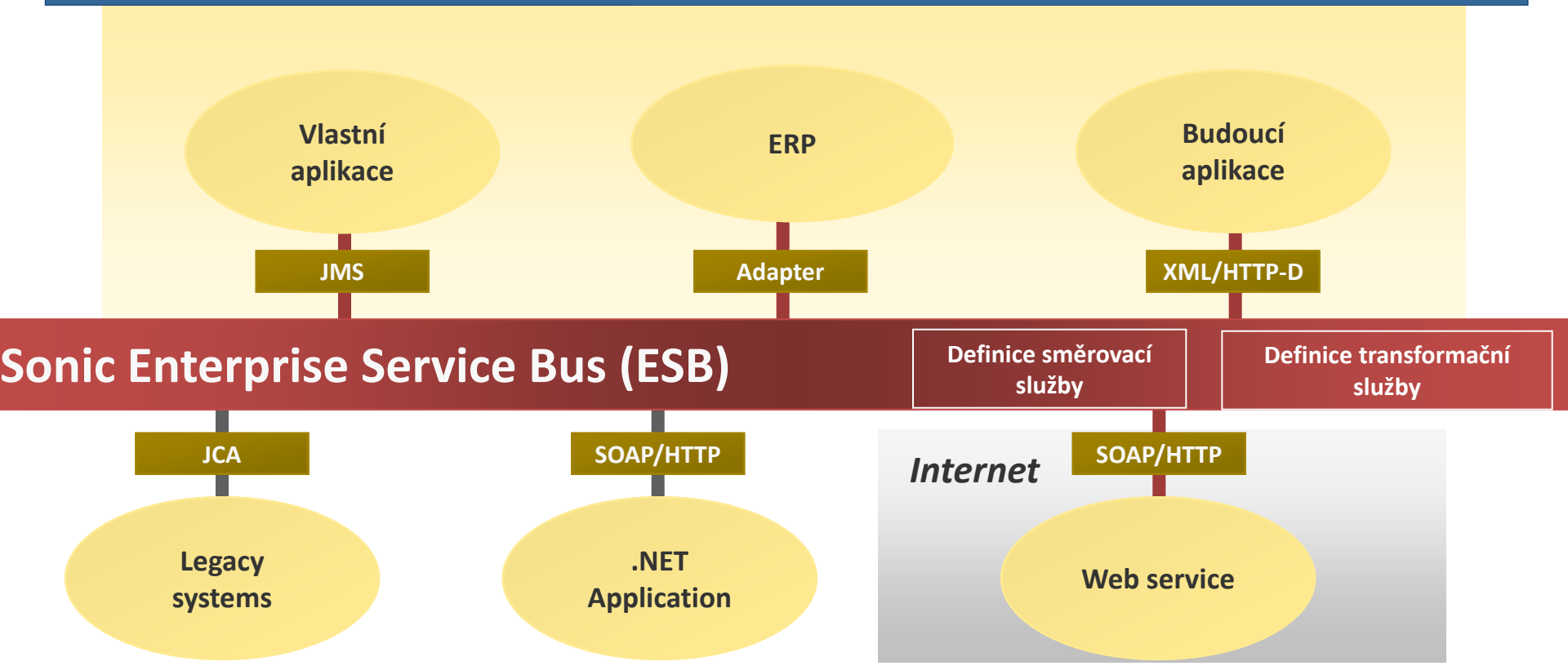
Komunikace může obecně pro N služeb vyžadovat až N*(N-1) transformací zpráv. Řešením může být jediný formát pro zprávy mezi všemi předřazenými branami. To je základní idea řešení Enterprise Service Bus od Sonic Software, nyní Progress Software.

Zprávy mohou být přenášeny mnoha způsoby (signály, mail, telefon, www, …)

Stumpf Jindřich, Progress Software, Systémová integrace 2004, příklad

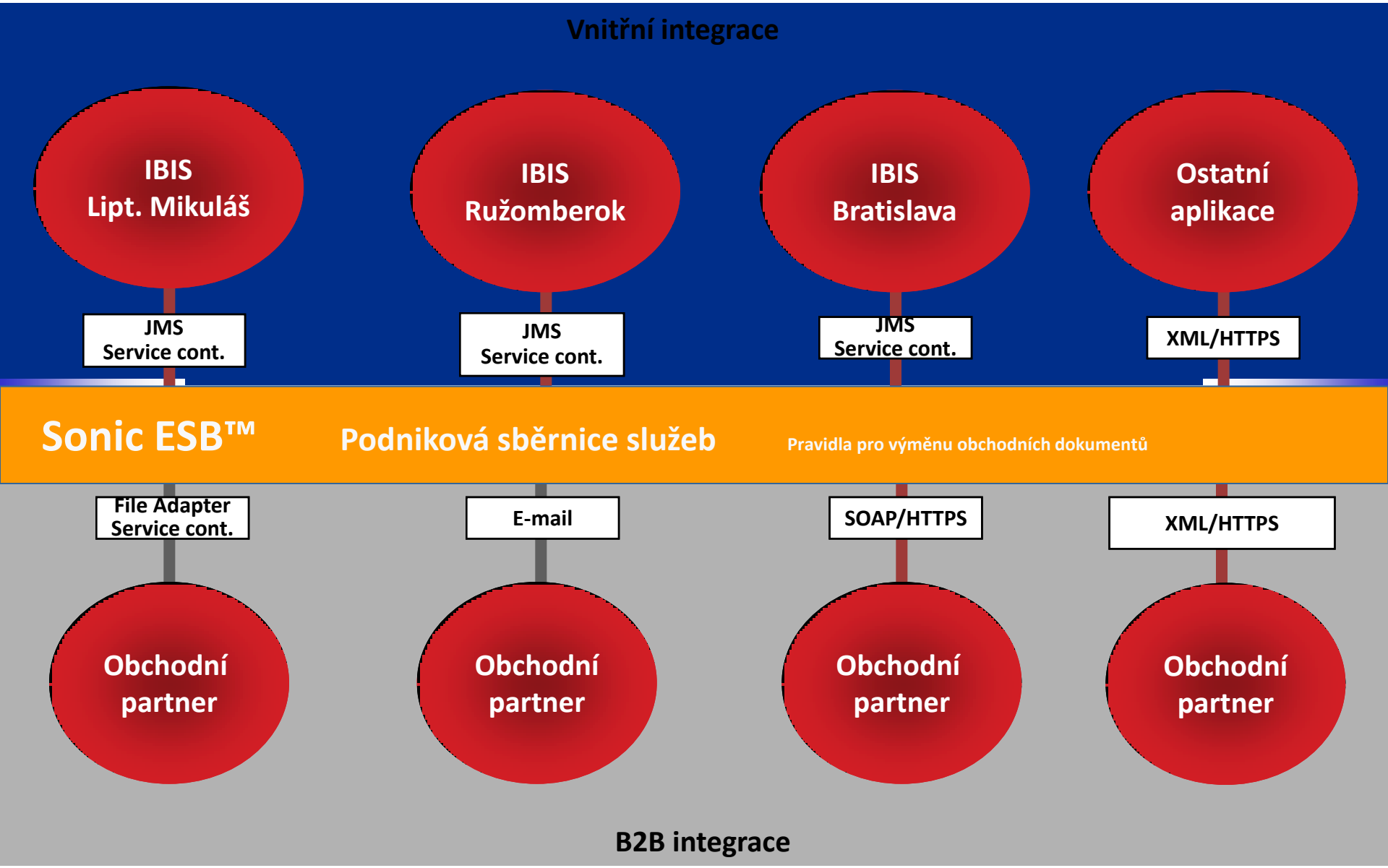
"Do roku 2005 bude *Podnikovou sběrnici služeb* (ESB) kombinující messaging, webové nebo i jiné služby, inteligentní směrování a transformaci dat, modelování a řízení business procesů, používat většina podniků."

Roy Schulte, VP and Research Fellow, Gartner Inc.



Stumpf Jindřich, Progress Software, SI 2004,

příklad konkrétního projektu

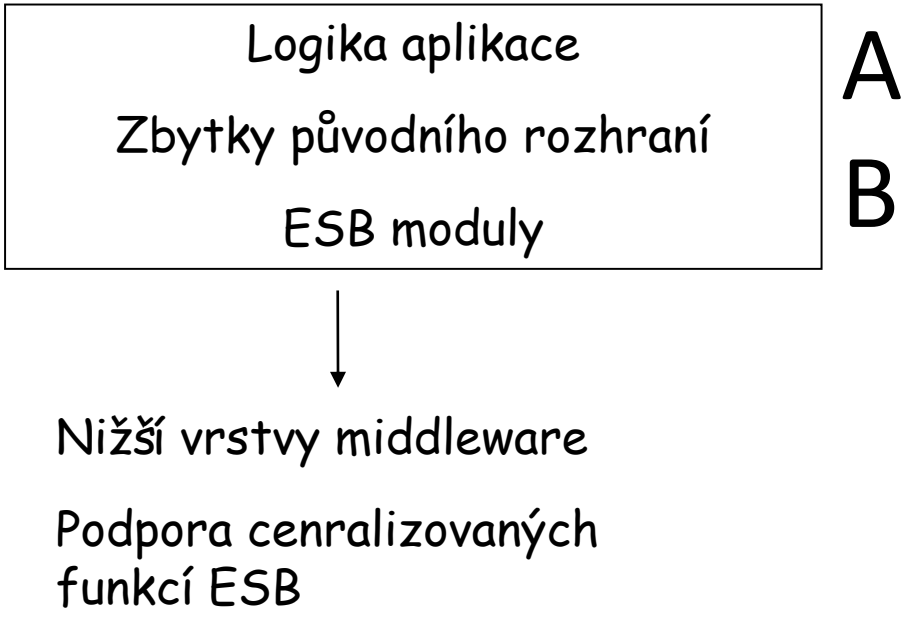


Co v daném případě přinesl ESB podle Progress SW

- Skutečně se téměř nemuselo do existujících aplikací výrazně zasahovat
- Bylo to velmi rychle hotovo
- Plná spokojenost uživatele
- Dodavatel ale vlastně realizoval dodávky spíše menší velikosti. Dříve by taková dodávka byla podstatně větší
- Bylo méně práce i pro vývojáře (pro ty dramaticky) a tedy i menší výnos pro dodavatele

Jak se použije ESB

- Nakoupí se knihovna zásuvných modulů
- Aplikace dostane s ESB tvar



Omezení ESB

- Mimo podnik hůře použitelné
- Závislost na dodavateli
- Malé možnosti agilního zapojení uživatelů do vývoje systémů
- Problém s řešením byznys problémů
  - Logování pro uživatele
  - Náhradní opatření při výpadcích není dobře řešitelné

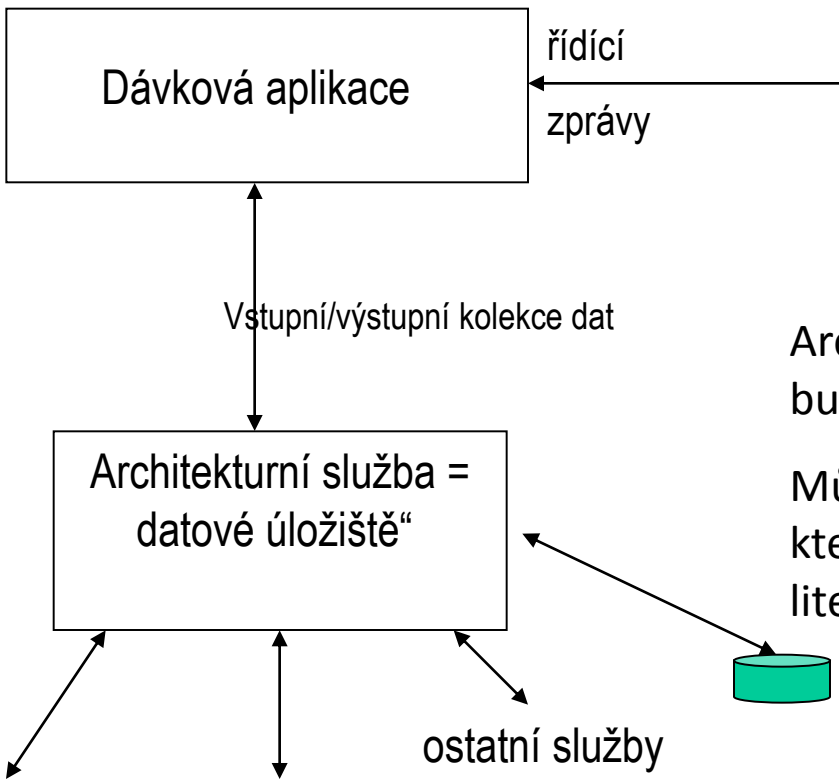
Omezení na podnik

- Nevede to k plnohodnotnému SOA
- Vychází to z módy
- Problém jednotného formátu vyřešeilo přijetí byznys dokumentů jako formátu zpráv

Otevřené systémy nemohou plně použít principy ESB

- Rychle se mění partneři
- Není příliš výhodná pro uživatelsky (obchodně) orientovanou komunikaci
- Řešení
  - Hrubozrnná dekompozice (IS organizačních jednotek)
  - Komunikace pomocí přenosu byznys dokumentů, ty jsou dosti standardizované
  - Adaptéry/wrappery resp. konektory jako služby

Integrace dávkového programu do SOA



Architekturní služba, nástroj budování architektury

Může se chovat jako webová , ale která webová (SOAP document literal?

Změna protokolu komunikace

- Datové úložiště pro zprávy
- Politika jejich vybírání, zpracování a odesílání

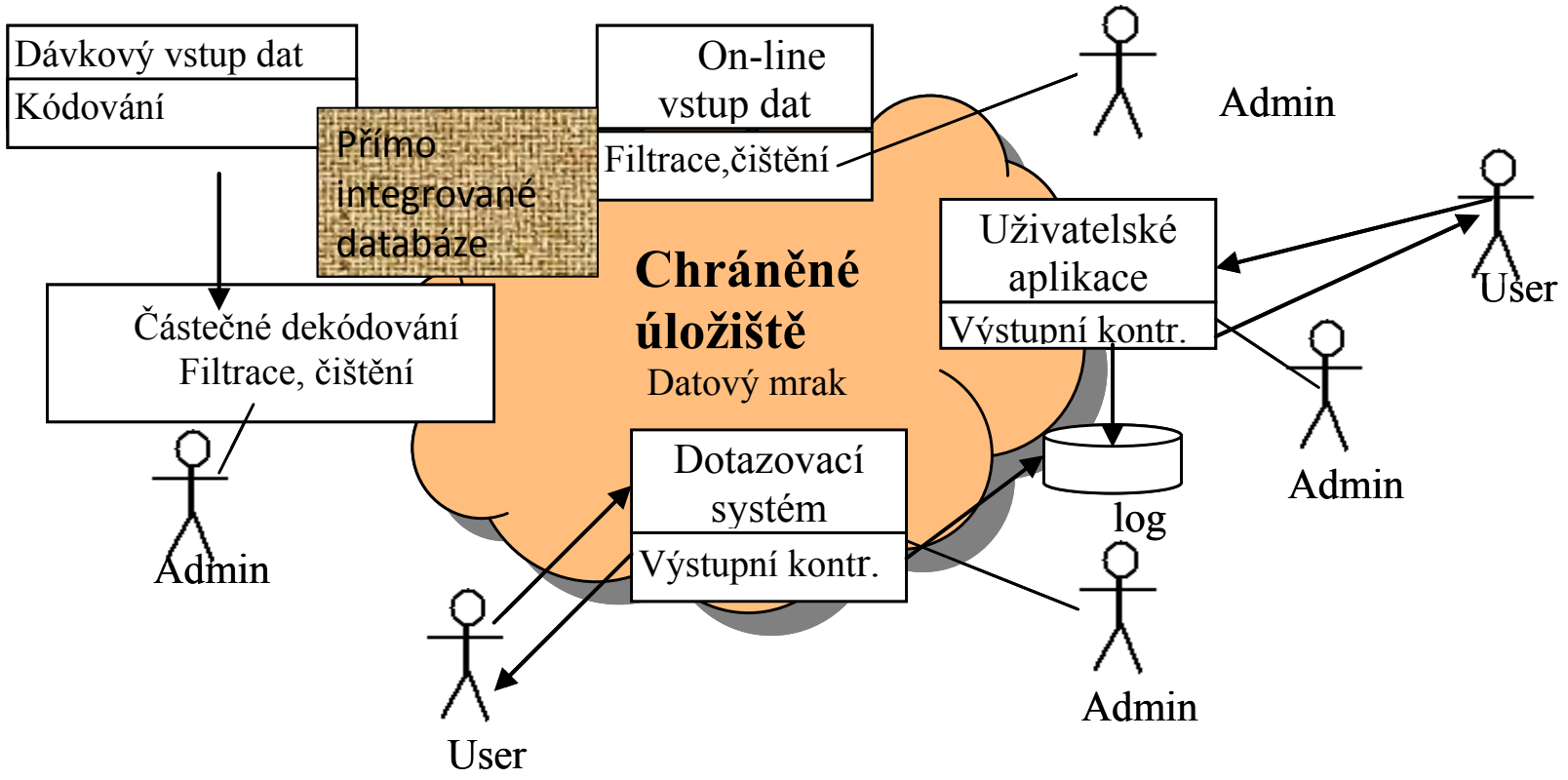
Kdy jsou potřeba komunikační služby s datovými úložišti

1. Služba běží příliš dlouho nebo produkuje velké množství dat (př. plánování a rozvrh výrobních operací), nebo je prostě dávková
2. Je žádoucí použít existující dávkovou službu
3. Implementace složitých variant komunikace (obsluha částečně záměných pracovišť)
4. Snazší a stabilnější implementace nebo snazší sourcing
5. Decentralizace repositářů (globálních služeb)
6. Úložiště může být použito pro určitý typ uživatelů, kteří mají specifické principy na hodnocení souhrnné kvality souborů dat, obsahujících podmnožiny dat různé kvality

Nedostupnost dat a řešení pomocí dávkové služby v SOA

- V e-gov ÚOOÚ de facto znepřístupňuje všechna (citlivá) data
- Zveřejnitelná informace je pro zjišťování ze strany občanů nedostupná, jeli k jejímu výpočtu potřebný citlivý údaj
- Přeceňují se případné škody z prozrazení dat a nedoceňuje ztráty ze ztráty informací
  - Je mnoho kanálů, kterými data unikají
    - Mobily, banky, sociální sítě, registrace na internetu, …
  - Současná praxe je cosi jako zákaz v podniku používat operativní data pro budování byznys inteligence

Zlepšení dostupnosti dat SOA a jednoduchý cloud



Časté zákazy pořizování dat a omezení na jejich využívání, neví se jak dělat výstupní filtrace, nehodnotí se ztráty vyvolané znepřístupněním dat

Technické důvody pro SOA

- SW inženýrské přednosti
  - inkrementálnost, stabilita, láce, …
- Lze se vyhnout tzv. reorg. Cycle (než systém stihnu přeprogramovat, je nová verze již zastaralá)

Potřeba metadefinic

Konfederovaný systém může obsahovat velmi mnoho komponent komplikované funkcionality.

Komponenty je výhodné propojovat deklarativně (co je třeba udělat, nikoliv jak to udělat).

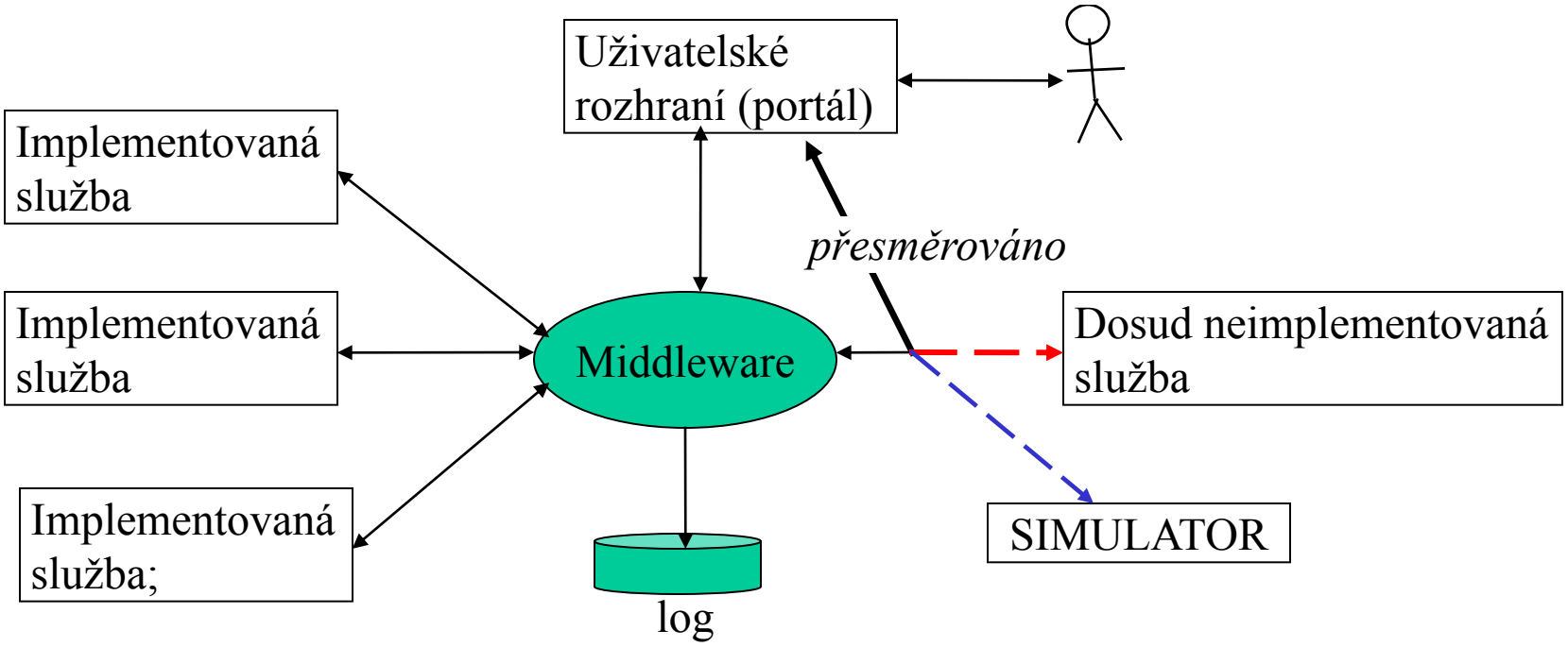
Pro různé skupiny komponent jsou třeba různé formáty.

Je nereálné je definovat jednou provždy nebo centralizovaně => formáty dohadovat lokálně. **Jsou nutné dialekty!!!**

Řešení: *Poskytnout nástroj pro rozšiřování syntaxe z jistého jádra*

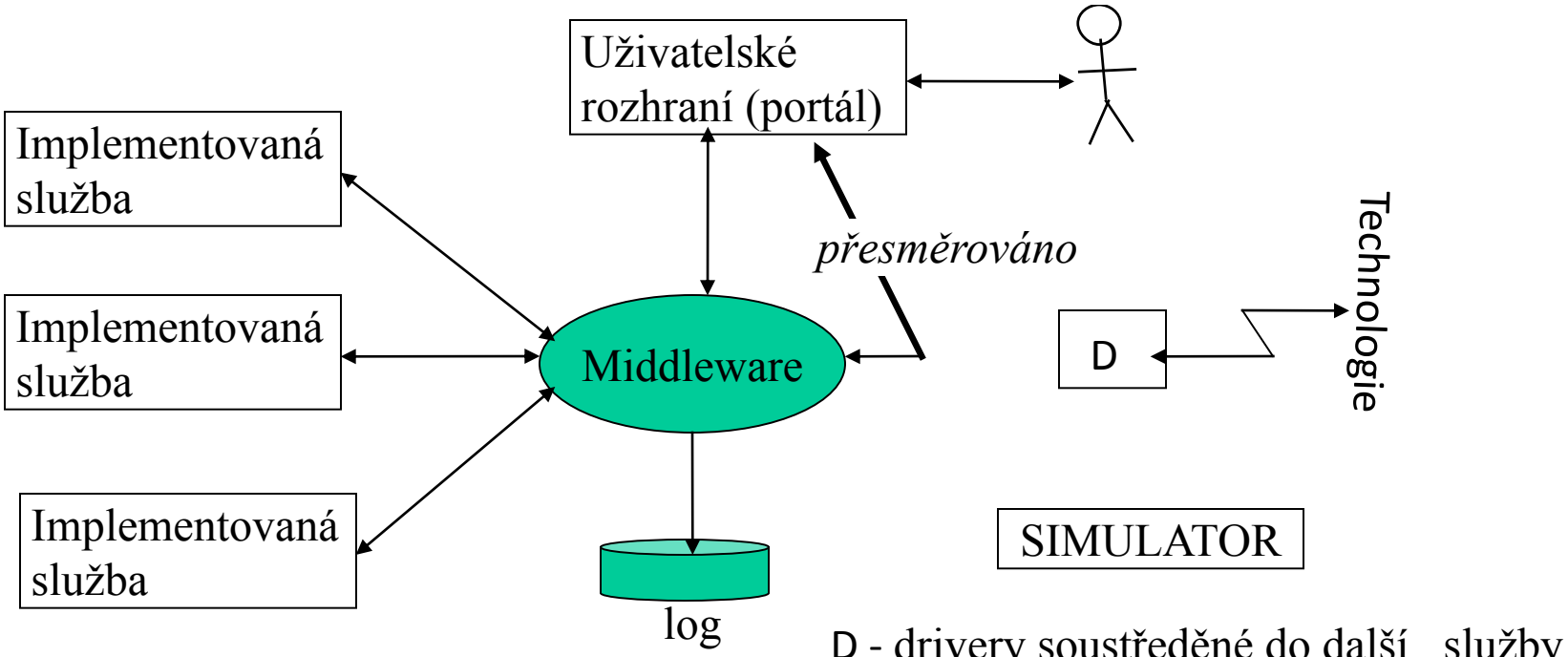
Řešení: XML,nebo koupě systému

Techniky 2
Prototypování, ladění RT systémů



Zprávy lze přesměrovat **beze změny implementovaných** služeb buď na uživatelské rozhraní (efektivní prototyp, použitelné i za běhu systému), nebo dokonce na simulátor (ladění RT systémů)

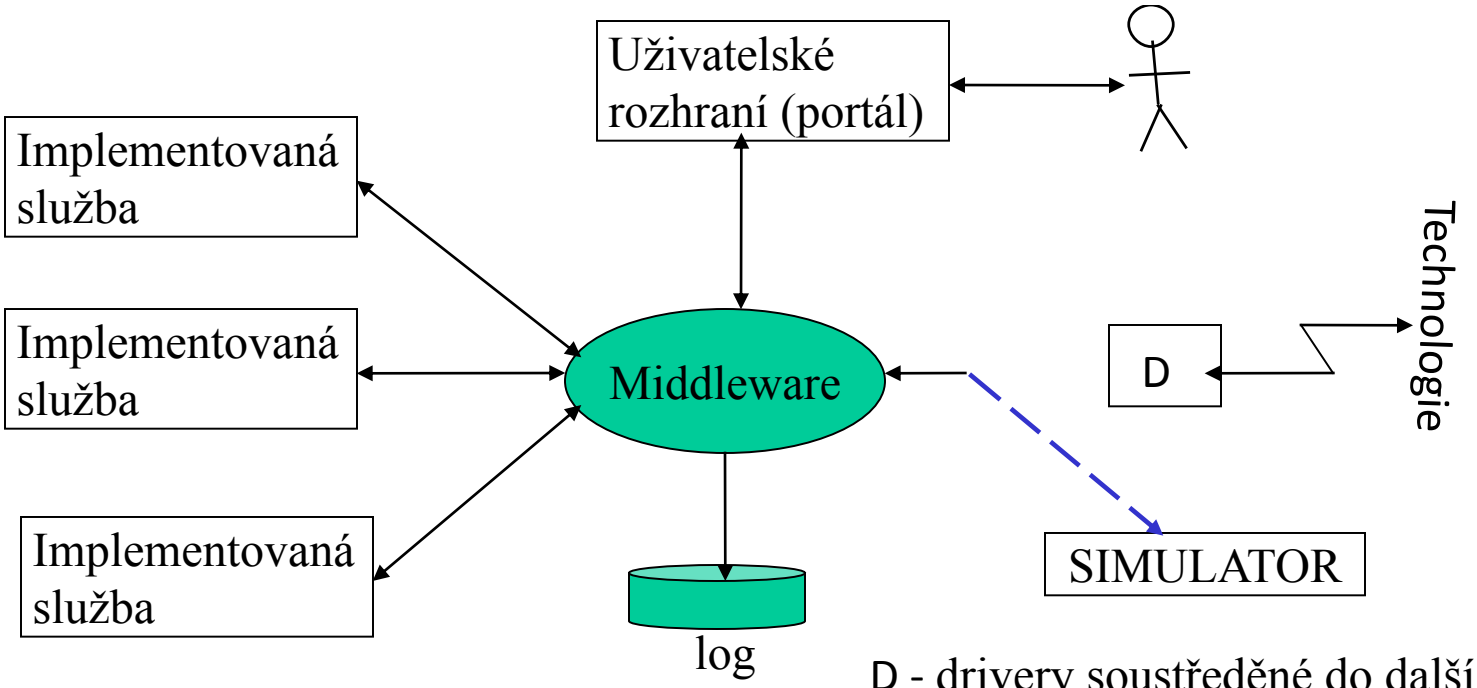
Techniky 2
Prototypování, ladění RT systémů



Zajímavý obrat – jak zjistit při dobu odezvy za nepřítomnosti řízené soustavy. Řešení

- Kalendář událostí v simulátoru,
- simulátor má nejvyšší prioritu

Techniky 2
Prototypování, ladění RT systémů

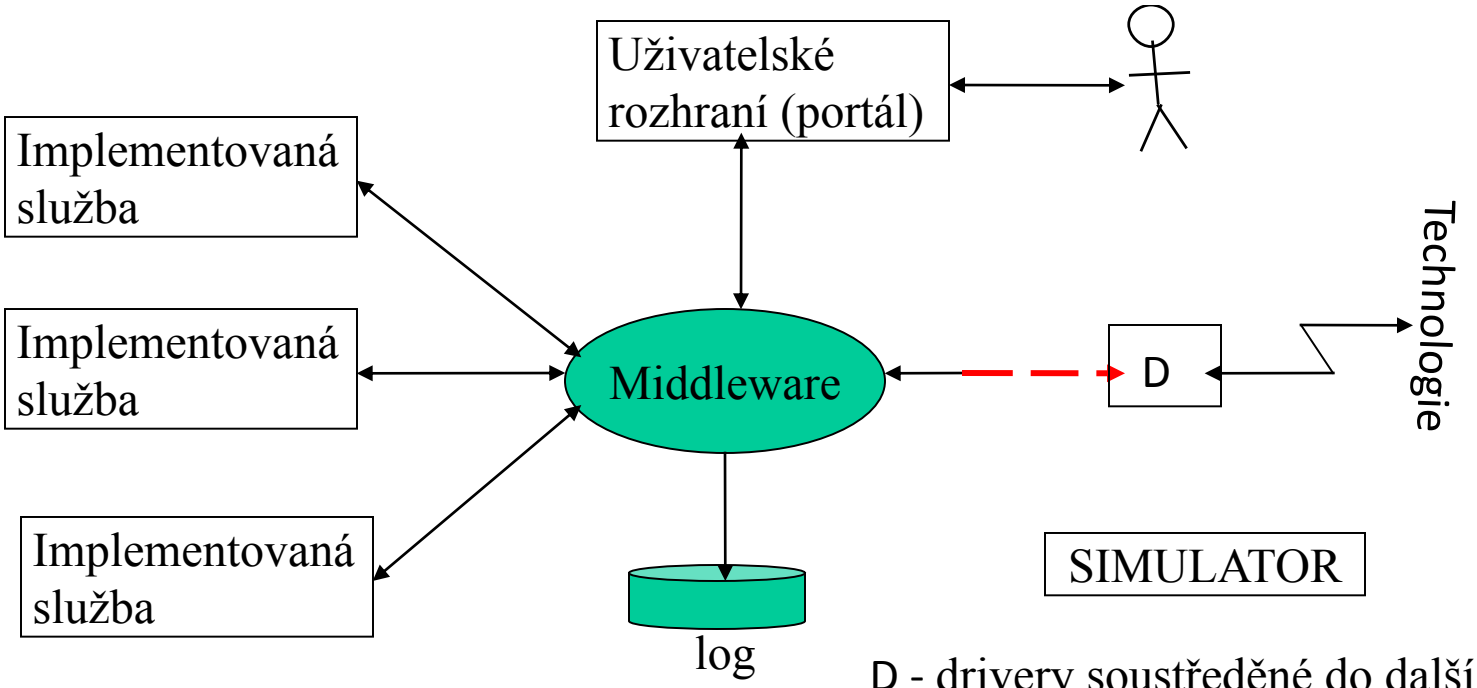


D - drivery soustředěné do další služby

Zajímavý obrat – jak zjistit při dobu odezvy za nepřítomnosti řízené soustavy. Řešení

- Kalendář událostí v simulátoru,
- simulátor má nejvyšší prioritu

Techniky 2
Prototypování, ladění RT systémů

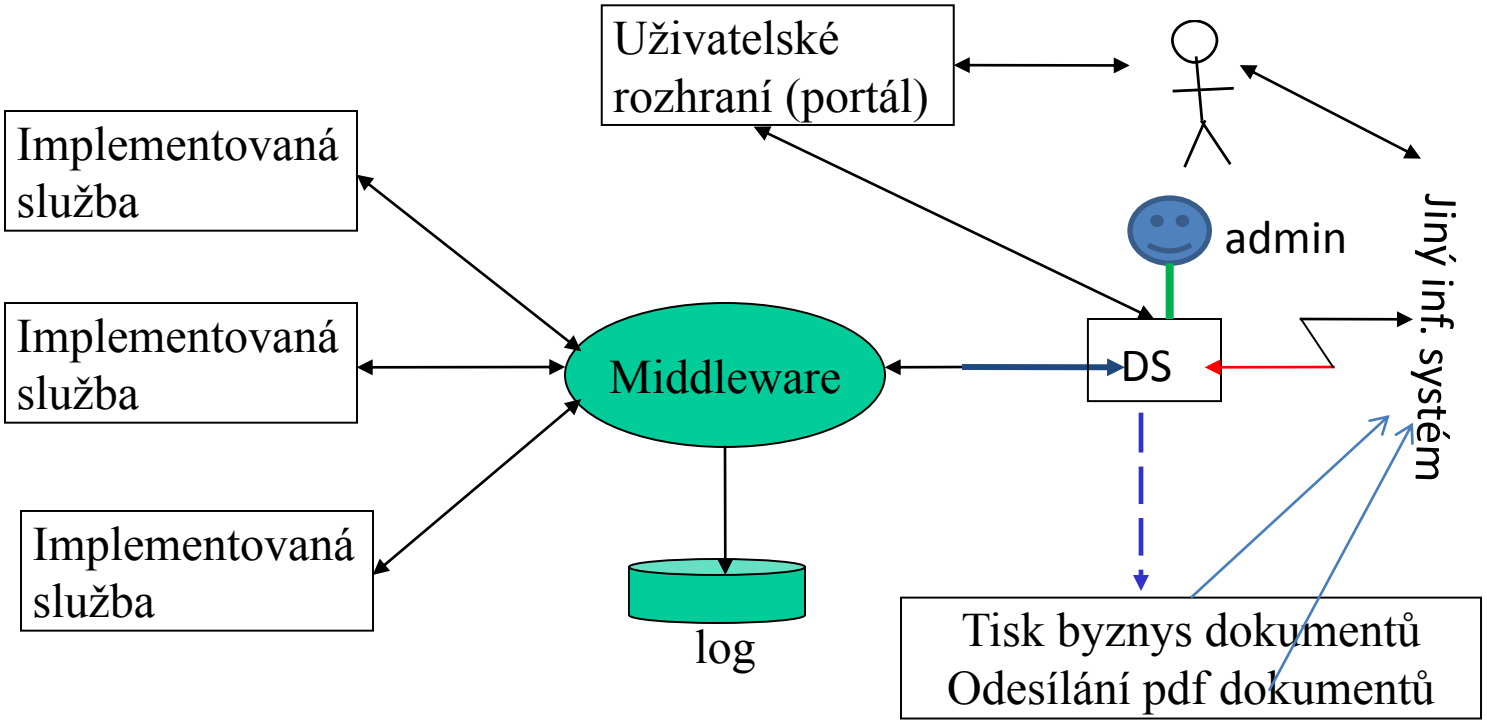


D - drivery soustředěné do další služby

Zajímavý obrat – jak zjistit při dobu odezvy za nepřítomnosti řízené soustavy. Řešení

- Kalendář událostí v simulátoru,
- simulátor má nejvyšší prioritu

Postupné budování a údržba SO systému



DS – Brána jako služba

Implementace inkrementální změny formátu zpráv a způsobu jejich přenosu a jejich směrování

Spolupráce služeb, call nebo request?

Systémy mezi sebou spolupracují
Aby mohly být používány rozumně, musí spolupracovat podobně jako služby reálného světa, být většinou stále k dispozici, (asynchronně) reagovat na požadavky z různých zdrojů a být používány jako černé skříňky. To reálně lze dosáhnout jen, tvoří-li virtuální p2p síť a používáme service request nejen service call.

Jak zajistit, aby jedna služba čekala na dokončení jiné služby (synchronnost)

- Service request: volající nečeká na provedení požadavku,
- Service call: Volaná služba o skončení požadavku vrací odpověď obsahující identifikaci volání a informace, které umožní pokračovat v daném vláknu volající komponenty, viz web 2 jako při RPC
- Je možné systémově a obecněji, proto byl vyvinut standard *enterprise service bus*. Jde o řešení typu middlewaru vhodné spíše pro podnikové systémy, ještě s k tomu vrátíme

Servisně orientovaná architektura, normy

Podrobný rozbor lze nalézt v *OASIS Reference Model for Service Oriented Architecture 1.0*

- Zahrnuje i méně používané modely
- Zdá se nehotový (málo pragmatický)

Jiné normy: W3C , workflow, byznys (podnikové) procesy atd., Open Group SOA Reference Model, Open System Integration Maturity Model

Desetitisíce stránek dokumentace norem

Open Group SOA Reference Architecture

Zaměřeno na integraci

Koncept ABB – Architecture Building Block

Také devět vrstev

Silný důraz na synchronní volání služeb

Nezdá se vhodné pro agilitu

Nové faktory v SW průmyslu

- Změna potřeb
  - Globalizace => nutnost a výhodnost konfederovaných distribuovaných systémů
  - Informační služby státu, dtto
  - Spojování produktů různých výrobců
  - Další SW inženýrské potřeby, např. inkrementální vývoj

Nové faktory v SW průmyslu

- Změna možností technologie, konfederace jsou dostatečně efektivní
  - Dostupnost a kapacita komunikační infrastruktury
  - Existence programovatelného uživatelského rozhraní, existence vyhovujících norem
- Potřeba otevřených byznys sytémů

Paradigma

- Základní filosofie, techniky, výsledky a postupy daného oboru
  - Newtonova mechanika * kvantová mechanika
- Nutné pro řešení nových problémů, nějak integruje starší paradigmata
- Obtížné přijmout lidmi zvyklými na starší paradigmata
  - Musí je často nahradit až nová generace
  - Přijetí a vycizelování nových postupů trvá roky
  - Spojeno se změnami obchodních a manažerských strategií a postupů
- V programování změna vůdčího paradigmatu jednou za deset let
- Strukturovaný vývoj, objekty, sítě a metanástroje,….

Problém přijetí paradigmatu Registr účtů a dluhů

Jak udělat registr účtů pro kontrolu oprávněnosti žádosti o sociální dávky a podpořit zjišťování okolností pro poskytování úvěrů, žádoucí je rozšíření na dluhy a perspektivně na majetky

Registr účtů a dluhů

Návrh v klasickém stylu:

Vytvořit centrální databázi účtů a úřad, který by ji spravoval. To chtějí úřady.

Pozorování:

Jde o centrální službu v konfederaci, p2p nemají „rády“ centrální služby (služby jsou IS bank, úřadů, …). Lze proto očekávat potíže, např.

- Náročnost replikace dat a aktualizace
- Malá flexibilita (a co jiný majetek, co dluhy)
- Problémy se zneužíváním (kdo použil, nebezpečí zcizení dat)
- Vysoké náklady na provoz

Registr účtů a dluhů

Alternativní řešení:

- Vytvořit relativně jednoduchou aplikaci, která se chová jako portál mající omezený přístup k ke všem IS bank pro dotazy tvaru: „Kolik má u vás pan X zrovna teď peněz/dluhů.“ Pro banky je to také jednoduché, většinou už mají téměř vše implementováno a mohou si ohlídat, kdo se to vlastně ptá.
- Problémy s efektivností jsou nepravděpodobné, dotaz musí odpovědný úředník naťukat, takže dotazy nebudou si příliš časté, a nepředpokládají se komplikovanější způsoby zpracování dat. Je nutná ochrana dat.
- Je třeba dokumentové rozhraní a tedy ne zcela tenký klient!!!
- Skrytý problém: Ochrana osobních dat

Registr vozidel

- Řešení propojuje nezávislé subsystémy používané jako služby, ty mohou mít různou architekturu, i SOA
- Výkon závisí na různých skutečnostech
  - Síť
  - Databáze
  - Logika
  - Pravidla pro programování klientů
- Je třeba orchestrace a hrubozrnné protokoly

Nutnost SOA, IS globálního podniku

- Globální podnik je tvořen sítí autonomních divizí, které lze prodat nebo které byly nakoupeny
- Nutnost spolupráce s IS proměnné množiny obchodních partnerů pro B2B
- Potřeba neustálé modernizace IS
- Potřeba používat nové nástroje analýzy dat vyvíjené různými výrobci
- Velmi rozsáhlý soubor koncových uživatelů s různorodými a proměnnými potřebami a právy
- Sourcing

Výhody SOA při prodeji a nákupu org. jednotek

- Nakupované divize bývají kupovány se svými IS, které se musí nějaký čas používat
- Divize je při prodeji cennější, je-li prodávána se svým IS. Ten proto musí mít v rámci podniku jistou autonomii (lze ho oddělit, neprodává se s ním příliš mnoho znalostí o celofiremním IS)
- Je žádoucí/nutné, aby se při B2B (business to business) používaly IS partnerů pokud možno bez úprav

Nutnost SOA (konfederace), státní správa

- Jednotlivé úřady mají své IS
  - Vzniklo historicky
  - Je politický zájem zachovat samostatnost IS jednotlivých úřadů
  - Je to i zájem věcný
    - Ochrana dat
    - Jasné odpovědnosti
    - IS potřebují k práci –> budou se o něj starat
  - Spolupráce s IS podniků a daňových poplatníků
- Přístup občanů k IS úřadů => Různorodé skupiny koncových uživatelů, mnoho uživatelů

Kdy je nutné použít SO, příklad státní administrativy

1. Jednotlivé složky státní správy mají své systémy, které si chtějí pokud možno zachovat
   - Vložily do nich svoje znalosti, mnoho funkcí se nesmí měnit a musí se provádět na daném úřadě
   - Musí ručit za jejich práci a proto jim musí věřit, leccos je tajné
   - Lidé se brání se změnám, které jim bezprostředně nic nepřináší a mohou je i ohrozit, změny se ale musí dělat ve spolupráci s nimi

Kdy je nutné použít SO, příklad státní administrativy

1. Jednotlivé složky státní správy mají své systémy, které si chtějí pokud možno zachovat
   - Vložily do nich svoje znalosti, mnoho funkcí se nesmí měnit a musí se provádět na daném úřadě
   - Musí ručit za jejich práci a proto jim musí věřit, leccos je tajné
   - Lidé se brání se změnám, které jim bezprostředně nic nepřináší a mohou je i ohrozit, změny se ale musí dělat ve spolupráci s nimi

Kdy je výhodné použít SO, příklad státní administrativy

1. Jednotlivé složky státní správy mají své systémy, které si chtějí pokud možno zachovat
   - Mocensky se lidé i úřady snaží zachovat svoji autonomii a své úkoly a zaměstnance
   - Autonomie je výhodná pro získání výhod (známosti mimo úřad, různé výhody, provize až korupce)
   - Je nereálné všechny systémy přepisovat znovu (cena, termíny, spolehlivost, rizika při přechodu) a i rekonstrukce nemůže mít za cíl monolitické řešení
   - Nelze připustit přílišnou závislost na jednom dodavateli

Kdy je vhodné použít SO, příklad státní administrativy, výhody

2. Modifikace systému jsou snazší, mnohé SW inženýrské přednosti
3. Systém musí být otevřený a spolupracovat s obdobnými systémy (obce a města, armáda, IS podniků, zdravotní systémy, IS mezinárodních organizací). Rozsah spolupráce je pro různé úřady různý. Je proto žádoucí použít takovou architekturu, která umožní stejný způsob spolupráce mezi subsystémy uvnitř státní správy i mimo ni. *Možná řešení* ***Active MQ, Datové schránky***
4. *Problém s politickými omezeními*

Požadavky

- Je nutno zachovat autonomii IS úřadů
  - Nelze jinak z politických důvodů
  - Je to výhodné věcně (alespoň potenciálně)
  - Je to výhodné softwarově inženýrsky
- Je nutné umožnit spolupráci s autonomními IS podniků
- Je nutné umožnit přístup občanům i IS podniků.

To jsou prakticky identické požadavky jako u SOA pro malé a střední podniky !!!

Proč to nejde

- Zájmy firem nepřijít o zakázky (SOA může snížit rozsah zakázky), zájmy politiků
- Nutnost změnit pravidla spolupráce
- Past standardů
- Hrozba účinné kontroly korupce a hospodaření
- Technické problémy

Příklad nákupního centra

Americké automobilky se před jistou dohodly, že zhromadní nákupy součástek do automobilů tím, že vytvoří společnou nákupní organizaci NO. Proces zhromadňování musí být založen na spolupráci IS v NO s IS klientských automobilek, které jsou i vzájemnými konkurenty. Odtud plyne, že IS v NO a IS klientů musí být nezávislé a komunikovat vhodným způsobem. IS klientů musí být integrovány jako černé skříňky. Je nejvýše vhodné, aby takový systém měl servisně orientovanou architekturu

Společné vlastnosti všech servisně orientovaných systémů, opakování

- Virtuální peer-to-peer spolupráce autonomních komponent (autonomie využití a také vývoje) spolupracujících jako služby masové obsluhy, p2p je nutnou podmínkou, aby se SW komponenty chovaly obdobně jako služby v reálném světě(v tom, co je služba není ještě mezi experty úplná shoda, pro nás autonomní komponenta, peer ve virtuální síti pracující asynchronně)

Společné vlastnosti všech servisně orientovaných systémů II

Dostupnost některých operací jinak obtížně realizovatelných (částečný outsourcing, decentralizace, podpora manažerských operací uživatelů obecně)

- Výhodné SW-inženýrské vlastnosti (otevřenost, modifikovatelnost, metody oživování, znovupoužitelnost, úspory při vývoji a údržbě,…)
- Použitelnost agilních forem vývoje, implemetace agilních byznys procesů

Kdy je nutné použít SO

- Spolupráce existujících systémů, které musí být použity tak, jak jsou, z následujících důvodů:
  - nelze je dost rychle přepsat (viz reorg cycle, vývoj trvá tak dlouho, že než systém dokončím, je zastaralý)
  - dosavadní uživatelé musí „svým“ službám věřit a musí mít příležitost specifikovat, co potřebují (feeling of ownership), neradi se učí něco, co jim nezlepšuje práci, jsou ochotni nést odpovědnost jen za to, čemu věří
  - uživatelé jsou sami velmi autonomní a většina funkčnosti jejich systémů zůstává lokální (srv. CRM, zdravotnictví, *e*-komerce), totéž chtějí od systému
  - politické a mocenské důvody
  - levoty (provize, úplatky, snahy o to, aby nebyl dohled)

Kdy je nutné použít SO případně komponentovou orientaci

- Velký systém musí být budován a modifikován po částech ze softwarově inženýrských důvodů
- Některé operace nelze rozumně impementovat jinak, než v SO (selektivní insourcing a outsourcing, decentralizace, integrace lidí do systému, agilní business procesy)
- Systém sám má charakter spolupráce služeb nebo aplikací (typické pro řízení procesů majících charakter služeb)
  - zčásti přítomno v systémech psaných v COBOLU
  - Soft RT, operační systémy – obsluha periferií, symetrický multiprocessing, komunikační sítě

Kdy použít SO

- Jako prostředek podpory distribuovanosti
- Jako prostředek dekompozice na daném místě, v daném počítači
- Jako prostředek sdružování kapacit (gridovské systémy, viz projekt CETI,cloud)
- Podpora inkrementálního vývoje a jiné SW inženýrské výhody, např. znovupoužitelnost, kombinace produktů různého původu
- Podpora otevřenosti, CRM a SCM
- **Aby to vůbec šlo vyvinout v rozumných termínech** (Neexistence SOA v sedmdesátých létech byla jedním z důvodů krachu projektu protiraketové obrany SALT)

Výhody servisně orientované architektury

- Chyby zůstanou lokalizované, modernizaci lze provádět po částech
- Stávající systémy mohou sloužit nadále bez podstatnějších změn, lze integrovat produkty třetích stran
- Flexibilita: Výše uvedený registr se dá snadno upravit na kontrolu dluhů a případně majetků
- Škálovatelnost: Lze snadno doplňovat nové služby a také je klonovat

Výhody komponetově (servisně) orientované architektury

- Autonomie částí zvětšuje odolnost systému proti výpadkům
- Dosavadní uživatelé existujících služeb jim mohou věřit, protože se v podstatě neměnní,. Nemusí se učit příliš nového a mohou důvěřovat i datům.
- Úspory nákladů z důvodů znovupoužitelnosti, použití produktů třetích stran a snadnějšího vývoje všeobecně

$$Prac = c * Delka^{1+a},\ a \cong 1/8$$

Dekompozice do n autonomních částí sníží pracnost přibližně n^{-a} krát

Business procesy

Řešení s využitím standardů popsaných v www.bpmi.org není jediné možné. Další varianty:

- Popisy pracovních toků (workflow, BPMN)
- Použití activity diagrams z UML nebo diagramů s použitím IDEF 3.0
- Jednoduchý fulltextový popis se zaznamenáváním provedených akcí/kroků

! Často je nutné používat všechny varianty!!!!

- Obtížně se implementuje v centrální databázi u velmi rozsáhlých systémů(obecný problém centrálních DB v rozsáhlých p2p systémech)

Uživatelsky orientované rozhraní služeb

- Výše uvedené požadavky na podnikové procesy je možné splnit, jsou-li rozhraní služeb uživatelsky orientované - srozumitelné pro vlastníky procesů a pokud možné blízké zavedeným způsobům zadávání prací v reálném světě.
  - To je důležité i pro případné soudní spory a je to i podmínkou, aby bylo možné požadovat odpovědnost vlastníků procesů za případné škody
- Uživatelsky orientované rozhraní má tendenci být deklarativní (pak skrývá i implementační detaily a to je významná výhoda)
- Zvláště výhodné je využívání byznys dokumentů ve zprávách

Inženýrské výhody uživatelsky orientovaných rozhraní

- Uživatelsky orientované rozhraní má tendenci být
  - Deklarativní (pak skrývá implementační detaily služeb což je známo jako významná výhoda pro modifikace)
  - Stabilní v čase (bývá založeno na prověřených znalostech a postupech)
  - Sémanticky bohaté (snižuje nároky na komunikační kanály)
  - Snadno použitelné při definici obrazovkových prototypů
- Nevýhodou je, že nebývá standardizováno
  - Může být i výhodou. Předčasná standardizace bývá nedokonalá a může proto představovat značný problém (norma se nehodí na daný úkol, často se mění, náročná na použití)
- Lze využít DMS, document management systems

Inženýrské výhody uživatelsky orientovaných rozhraní

- Uživatelsky orientované rozhraní a autonomie komponent zvětšuje možnost aplikace principů agilního vývoje
  - Stačí dokumentovat jen rozhraní
  - Malé úkoly
  - Nezávislost úkolů
  - Spolupráce s uživateli
    - Viz pravidla extrémního programování a obecně agilních metod Scrum, atp
- Otevřená otázka: Pro jaké úkoly je nutná jemná granularita systému

Uživatelsky orientované rozhraní (UOR) a agilní formy vývoje

- UOR se musí vyvíjet ve spolupráci s uživateli
- Dobré UOR silně omezuje potřebu dokumentace, kromě prostředků potřebných pro používání systému. Práce služby se totiž dá často odvodit z jeho rozhraní.
- Použití prototypů umožňuje rychlou odezvu uživatelů.
- Dekompozice do služeb usnadňuje inkrementální vývoj
- ALE: UOR je možné jen za podmínky správné dekompozice (přibližně hranice organizačních jednotek)

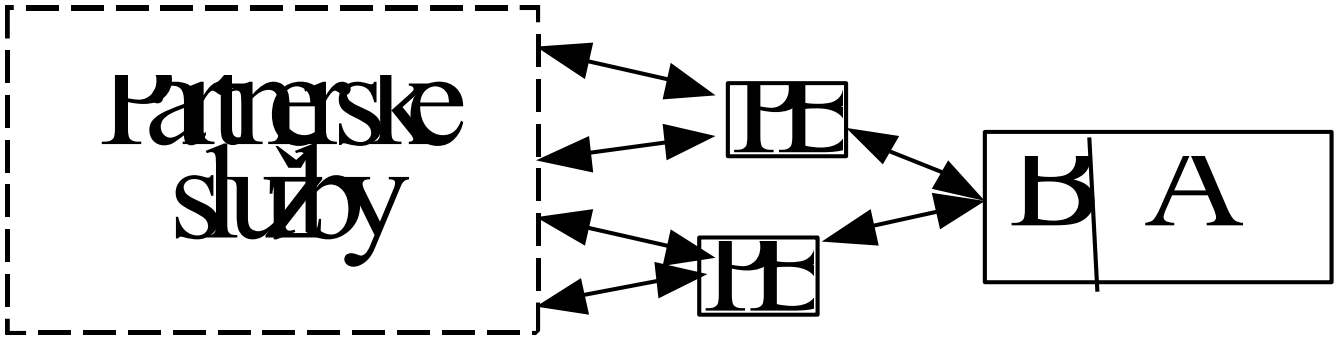
Optimální řešení: výměna byznys dokumentů, předřazené brány

- S účetním subsystémem spolupracuji pomocí výměny dokumentů, jako je faktura, pohledávka atd.
- Je to srozumitelné a jako tlustý klient se může použít služba, jejíž centrální část je tabulkový kalkulátor.
- Lze použít pro globální spolupráci informačních systémů.
- Takové rozhraní je dobře použitelné koncovými uživateli nejen u malých firem, je nutností v e-governmentu

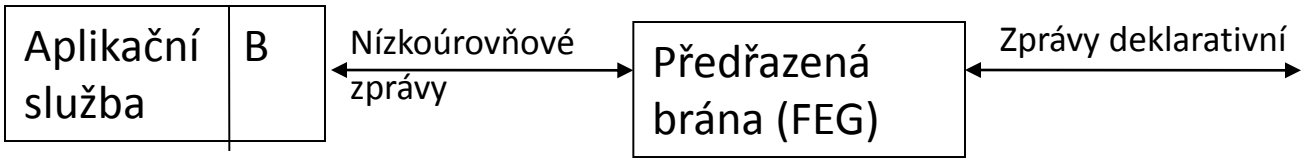
Nástroje pro implementaci uživatelsky orientovaného rozhraní

- Existující aplikace nemusí mít vhodné rozhraní
  - Nutný odposlech uživatelského rozhraní aplikace, aplikace lepší bránu nemá
    - Snazší má-li třívrstvou architekturu
  - Rozhraní není uživatelsky orientované (je např. OO, RPC)
- Různí partneři požadují různé rozhraní -

Řešení: Infrastrukturní služba jako přeřazená brána

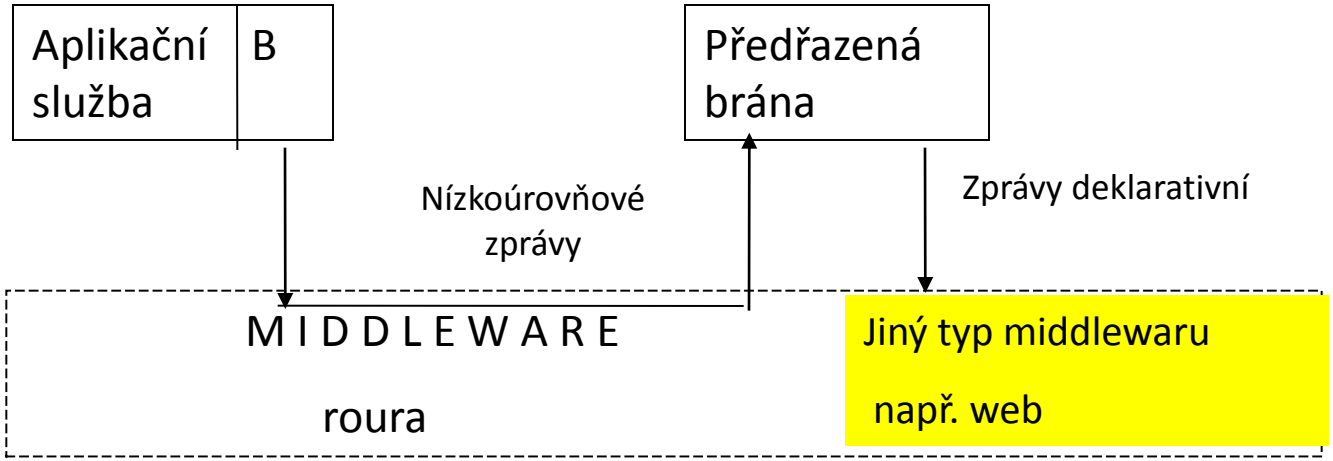


Logický a implementační pohled na předřazenou (proxy) bránu (front end gate, FEG)

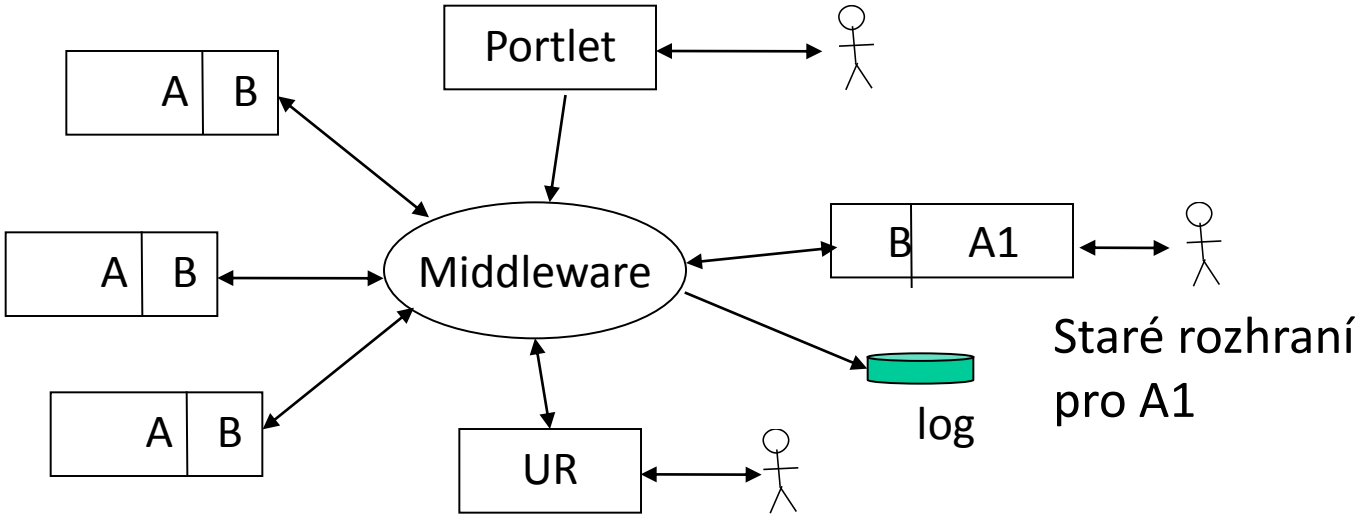


a) Logický pohled

B se často implementuje jako API komponenta, zvaná service component

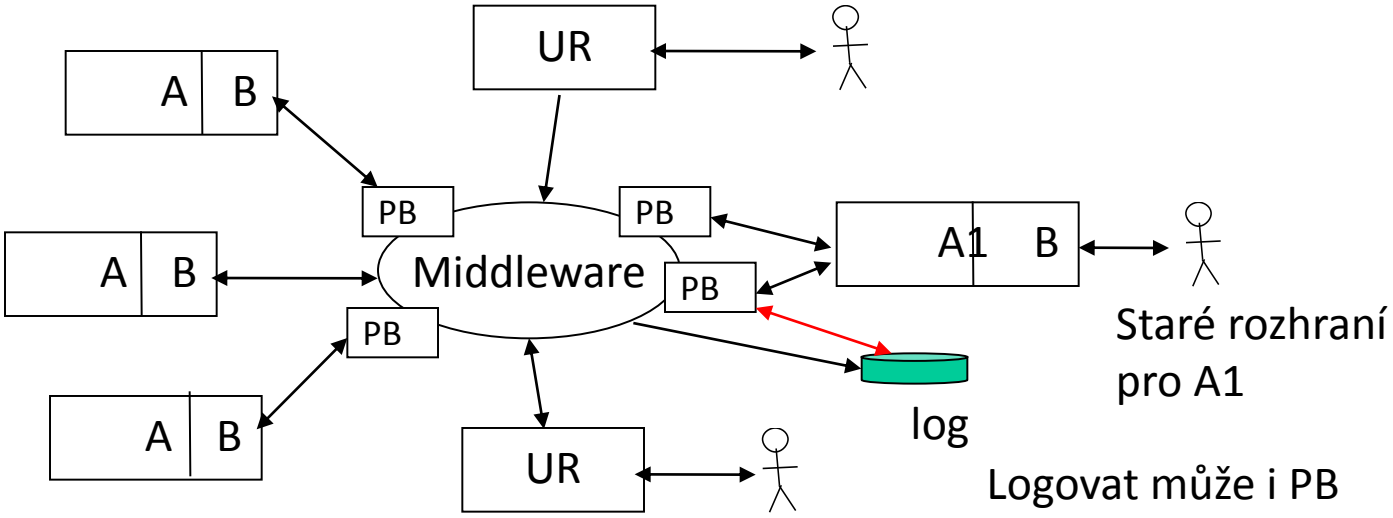


SOA bez konektorů jako služeb



Nástroje na programování přeřazených bran jsou stejné jako u portálů/portletů

SOA s přeřazenými branami, konkrétní propojení

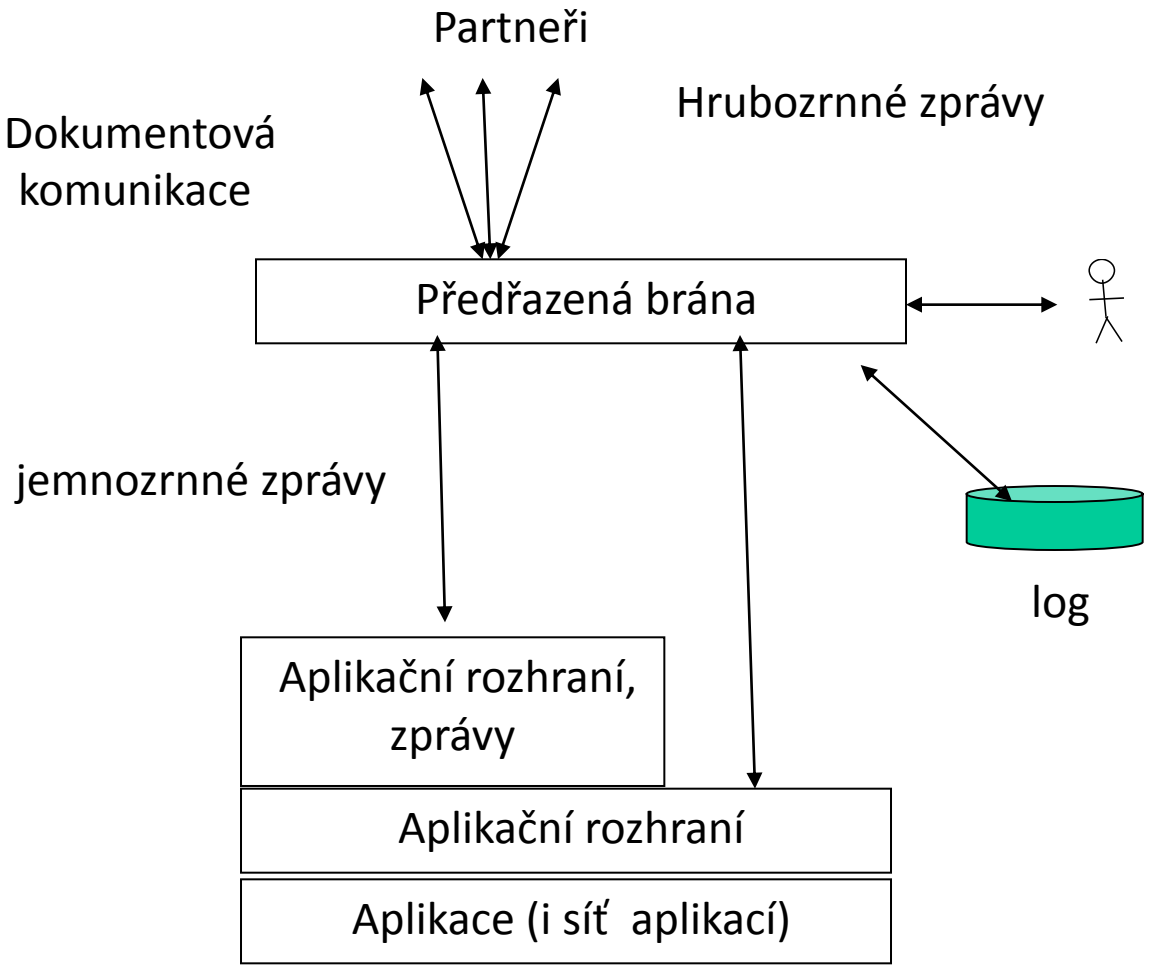


PB — Předřazená brána, může jich být pro danou službu více, mohou i chybět. Pozor! B a PB mohou případně komunikovat přes internet, PB je příklad konektoru jako **služby.PB mohou zajistit, aby midleware komunikoval zprávy v jednotném formátu (viz ESB)**

Jak implementovat brány

1. Integrovat do komponent
2. Jako zcela plnoprávné služby
3. Integrovat do middlewaru
4. Kombinovat přístupy (dvoustupňové rozhraní: vnitřní brána - wrapper a předřazená brána PB), to preferujeme
5. !! I midleware může obsahovat infrastrukturní (architekturní) služby, takové SOA nazveme konfederací

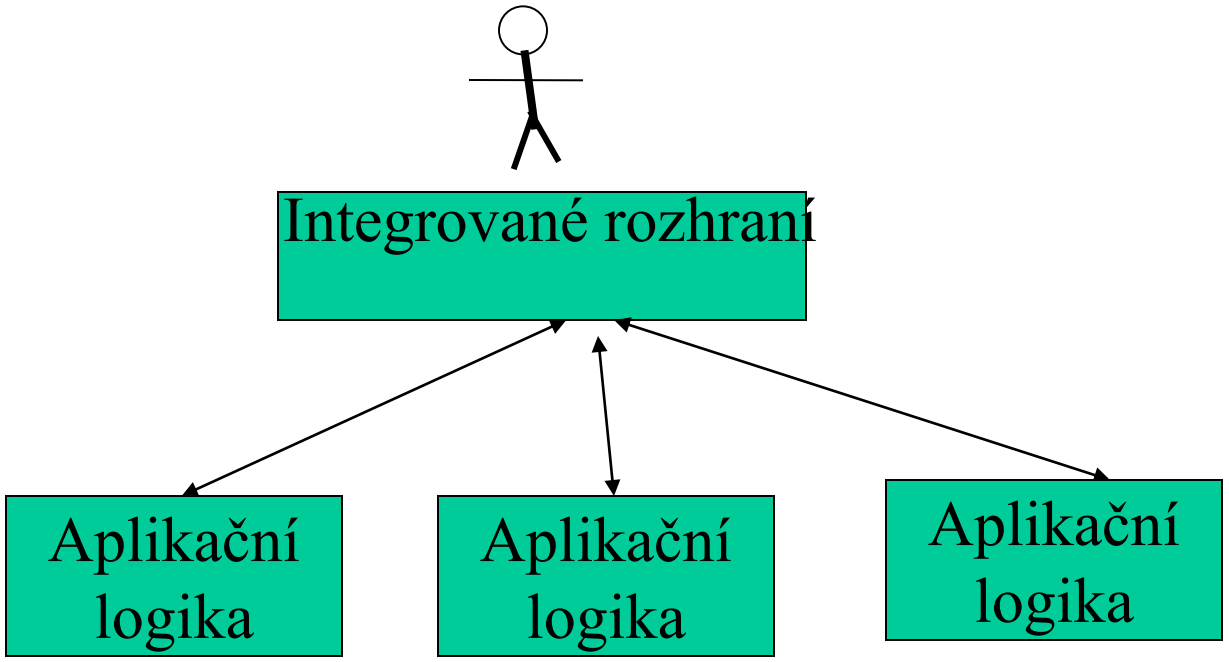
Vícevrstvé rozhraní



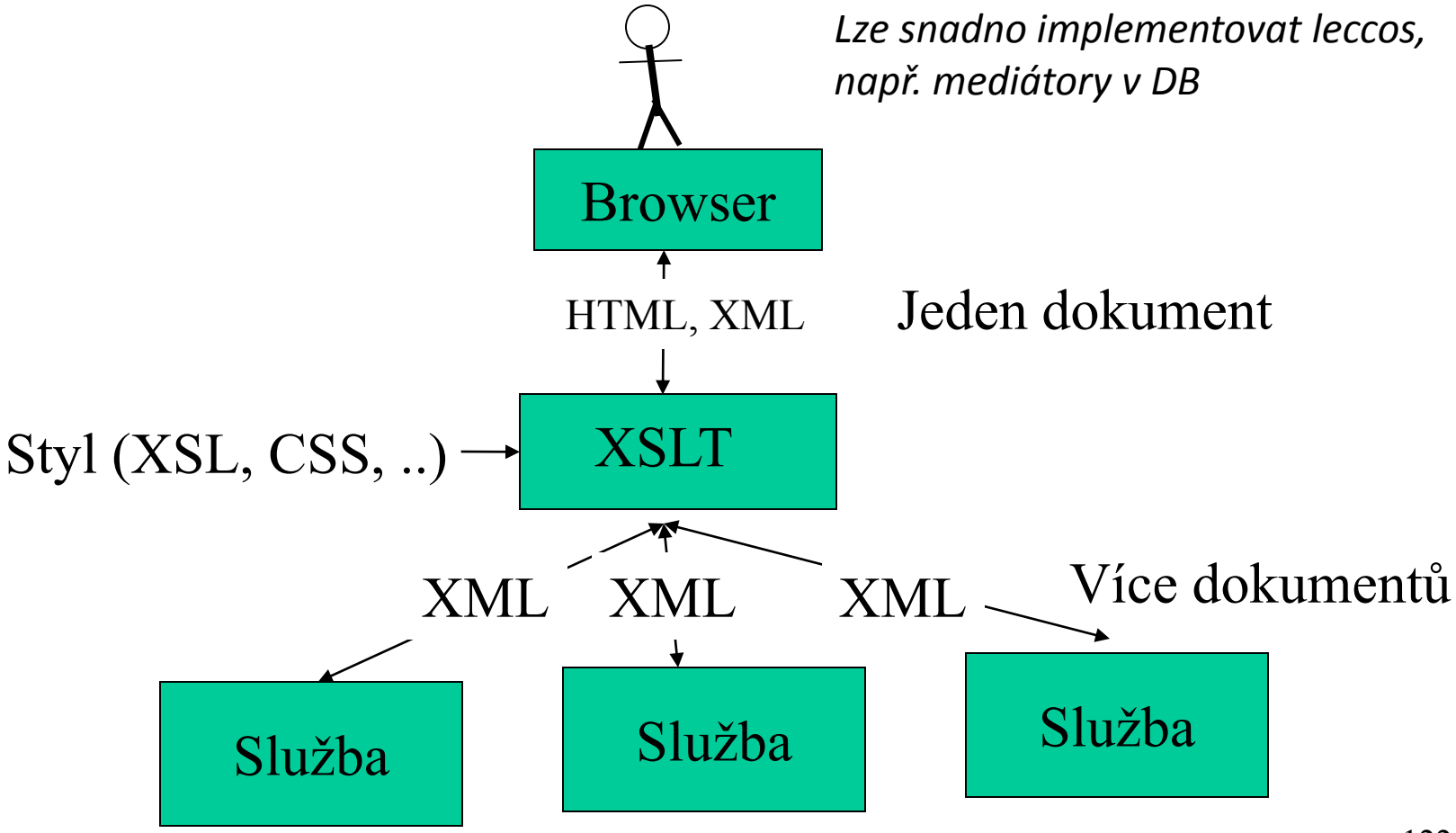
Výhody

- Použitelnost komponent s uzavřeným kódem
- a legacy systémů
- Vyhovění potřebám vývojářů (přístup ke všem možnostem, nízkoúrovňové změny kódu) a potřebám uživatelů
- Agilní změny protokolů
- Žurnály (logy) podle potřeby
- Flexibilita, co nepotřebuji nedělám.
- Dokonalé skrývání informací

Hlavní problém uživatelského rozhraní: Potřeba skládat dokumenty



XML a uživatelské rozhraní srv též SAP



XSLT lze použít tak, že může transformovat v jednom kroku m vstupních zpráv na n výstupních zpráv

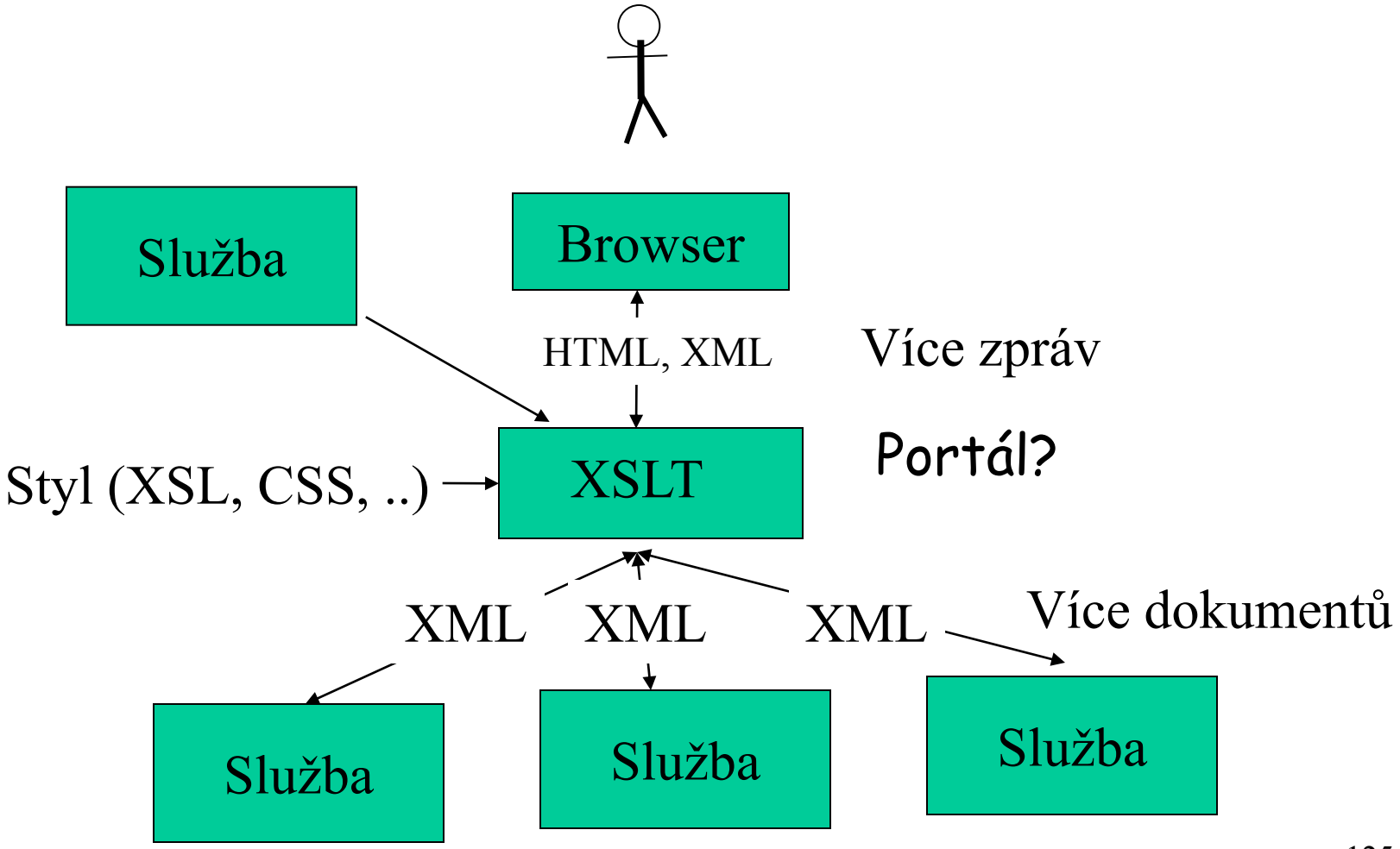
Takový aparát můžeme chápat jako zobecnění míst (places) v Petriho sítích s barevnými značkami

Problém: XSLT není zatím moc úspěšný pokud se nepohybujeme v objektovém světě

Doplnit

- Webové služby a doplněk o SOAP
- Překrmené, zkušenosti s kamerami
- Možnosti ontologií a pravidel pravidla a prototypy, problém výpadky a údržby
- Problém cloudu a zkušenost bsystems

XML a uživatelské rozhraní



V čem je XML s podpůrnými nástroji důležitý (nutný)

1. Rozšiřitelnost, přijatý standard
2. Vhodný pro deklarativní rozhraní mezi aplikacemi (pravděpodobně nejdůležitější)
3. Schopnost transformovat a skládat dokumenty a schopnost definovat a využívat styly pro UI (nutné i pro práci s heterogenními databázemi)
4. Schopnost definovat metadata a schopnost tvořit dotazy a specifikovat semistrukturovaná data, schopné podporovat i *datový pohled*. Velmi slibné, nejméně významné (zatím)

METANÁSTROJE IMPLEMENTACE PŘEDŘAZENÝCH BRAN

Výzvy

- Nová technologie,
  - nejsou známy hlavní omezení
  - Neřeší implementaci byznys procesů
  - Spojování do sítí a hierarchizace
- Je potřeba cosi jako router

Důsledky

- Obchodní procesy mohou „programovat“ uživatelé a mohou za ně ručit
- Kromě aplikačních služeb jsme diskutovali služby, které bychom mohli nazvat službami architekturními
  - Předřazené brány (konektory služeb implementované jako služby)
  - Portály
  - Manažery procesů
  - Datová úložiště
    - Databáze
    - Zprávy
- Architekturní služby se většinou pro danou aplikaci vyvíjejí od počátku. Mohou existovat další typy infra služeb.
- Existuje pokus o obdobu manažerů na www

Pozorování

- Chování systému (business procesy) ovlivňuje do jisté míry sám koncový uživatel zadáváním parametrů instanciace procesu a úpravami jeho chování (to lze do jisté míry považovat za vývoj, který provádí koncový uživatel - enduser development)
  - Je snadné v konfederacích
- To je projev nových trendů ve vývoji a vlastnostech softwarových systémů, především pro malé a střední podniky a e-government

Kde jsou hlavní přínosy pro uživatele

- Horizontální spolupráce přes hranice oddělení (příklad Otavan, obchodní cestující si reservuje výrobní kapacity), ale to ohrožuje některé manažery
- Dostupnost informací (PC+internet)
- Sociální kontakty, zlepšení ovzduší ve firmě
- Zefektivnění procesů
- Zpřesnění a vyšší dostupnost dat, zkvalitnění rozhodování
- Restrukturalizace obchodních procesů, větší agilita
- Zlepšení spolupráce s externími partnery

Business proces v SOA

- Síť kroků
- Krok je provedení příkazu nějakou službou (i sítí služeb, kompositní službou)
- Je vhodné si pamatovat průběh provedených procesů, resp. modelovat proces
  - Neměl by být centrální repositář modelů procesů využívaný v průběhu celého procesu (důsledek p2p architektury), raději distribuovaná databáze
  - Potřeba využívání různých prostředků popisu procesů (volný text, BPEL, Aris, Petriho sítě,….)

Pragmatika

Propojování může být založeno na různých technologiích, některé může být určeno k propojování na daném stroji, jiné je možné použít globálně mezi různými stroji či sítěmi

Business procesy

- Mají stav (vlastní data, stav řešení).
- Jejich průběh musí být vlastníkem procesu měnitelný i „za běhu“, musí být agilita
  - Nereálné provádět uvnitř aplikací, které jsou dávno odladěné a používané jako černé skříňky
- Je nevhodné řídit proces v rámci centralizované služby (důsledek p2p architektury)

Kde mít řídící data

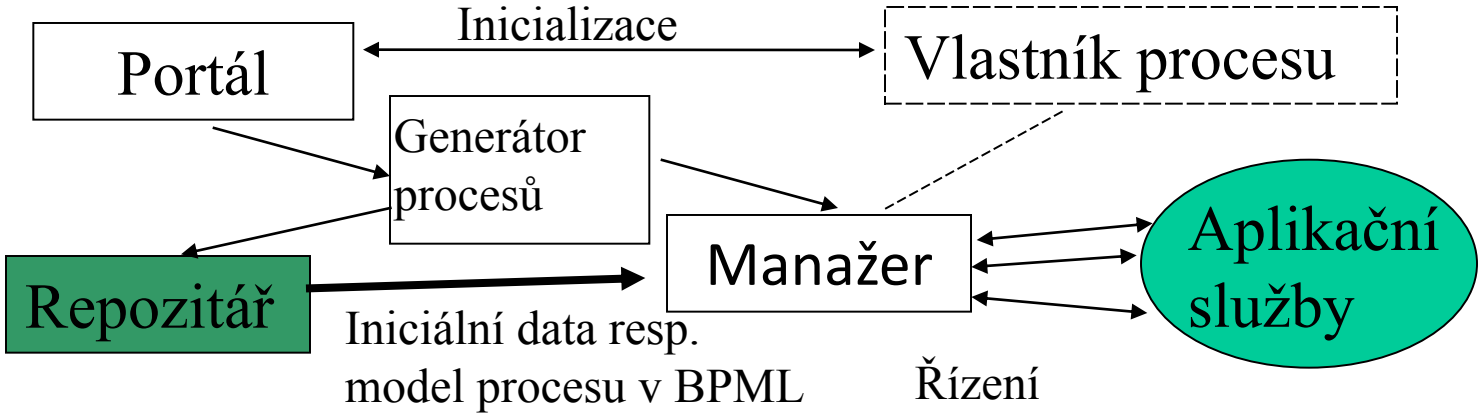
1. Nikde – služba ví s kým kdy co
   - Není možno sledovat průběh
   - Změna procesu $\Rightarrow$ zásah do komponent, ale to jsou obvykle černé skříňky
2. V předávaných zprávách (rozdělovníky)
   - Pružnější, explicitní řídící data
   - Obtíže při sledování
   - Jednotný formát, větší změny $\Rightarrow$ zásah do komponent

Kde mít řídící data pro BP

3. V centrální databázi
   1. Implementační potíže chceme-li různorodá data a snadnou přístupnost
   2. Neefektivní, obtížné sledování, centrální entita v p2p systému
4. Ve službě vytvořené pro každý proces znovu.
   - Lze snadno sledovat a implementovat různé modely. Není to levné

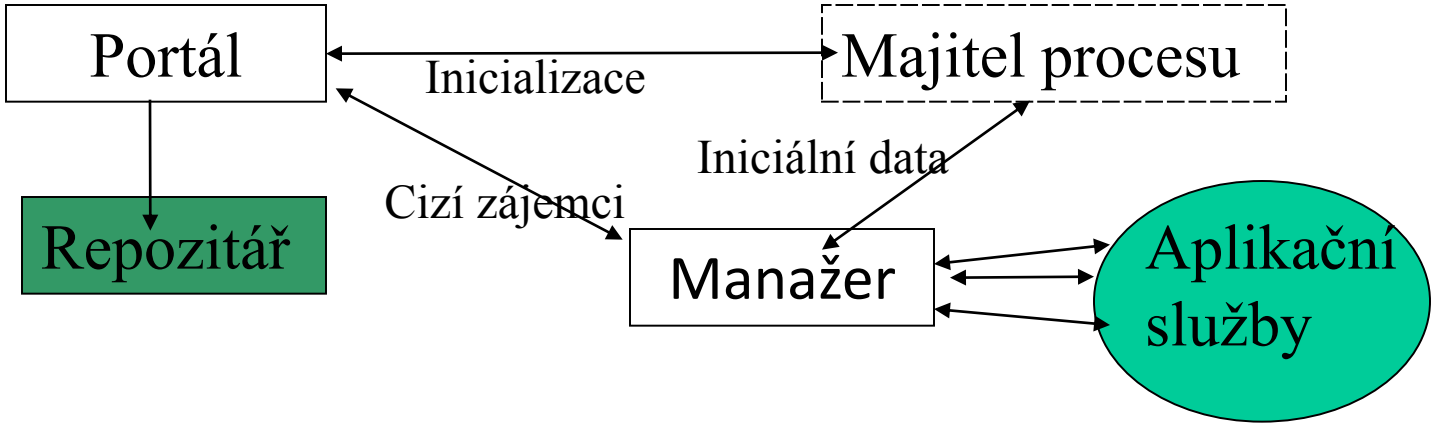
Business procesy

- Řešení: *Specializovaná služba **manažer** pro každý proces, simuluje prvky portálu, iniciální data z repozitáře, jedno z možných řešení, optimumální se hledá*



Manažer se vytváří pomocí generátoru procesů s případným využitím modelů procesů uložených v repozitáři pro každý proces znovu, manažer je služba která se chová jako instance procesu

Business procesy



Řešení s využitím standardů popsaných v www.bpmi.org, jiné řešení je v DMS (document management systems)

- Modely procesů v BPML (dialekt XML)
- Data manažeru v BPML (XML), upravují se podle parametrů inicializace zadané vlastníkem procesu, tím vzniknou řídící data procesu, např. v BPEL

Manažer odpovídá instanci procesu podle BPML. BPML ale neuvažuje o implementaci jako o službě

Business procesy

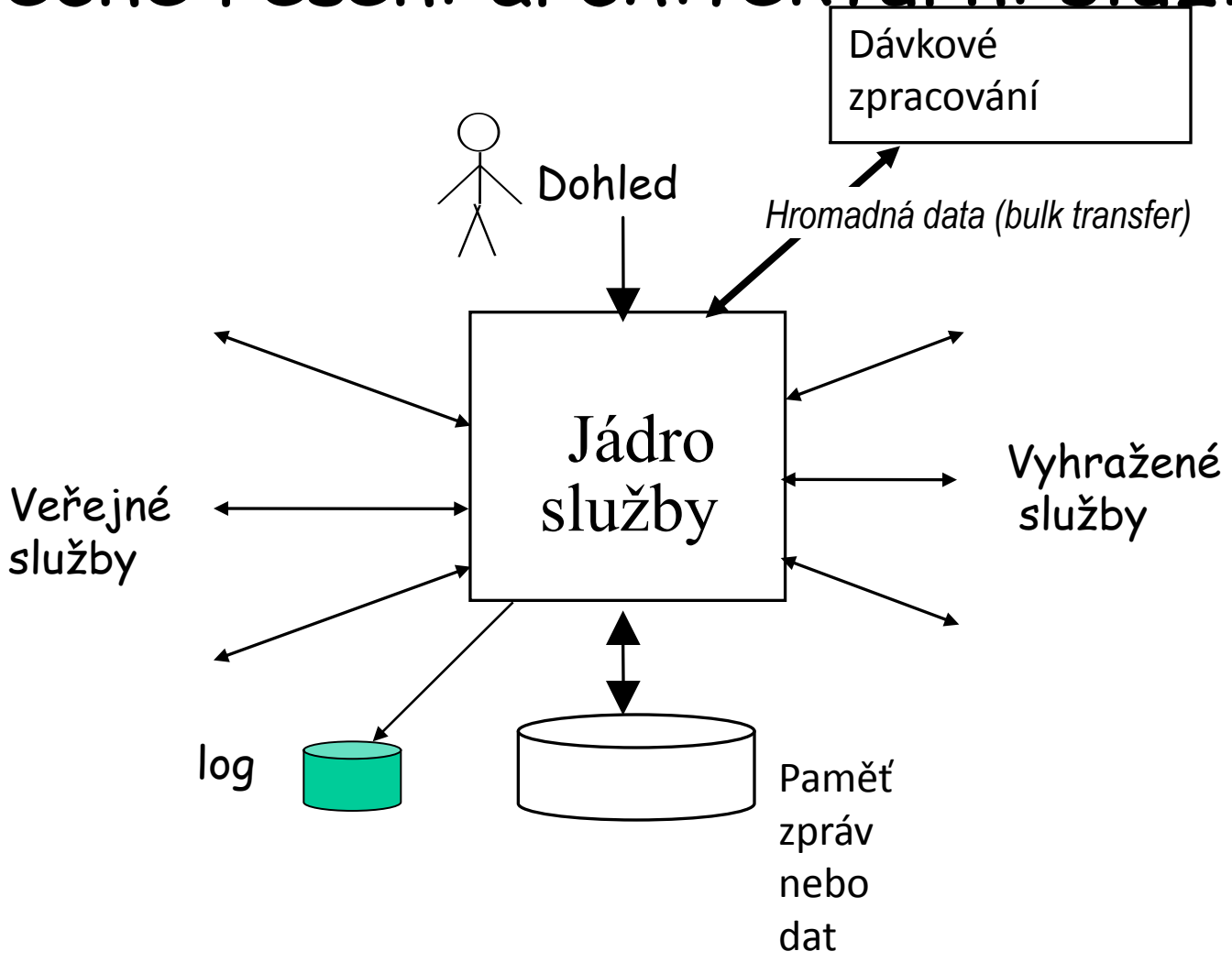
Řešení s využitím standardů popsaných v www.bpmi.org není jediné možné. Další varianty:

- Popisy pracovních toků (workflow, BPMN)
- Použití activity diagrams z UML nebo diagramů s použitím IDEF 3.0
- Jednoduchý fulltextový popis se zaznamenáváním provedených akcí/kroků

! Často je nutné používat všechny čtyři varianty!!!!

- Obtížně se implementuje v centrální databázi u velmi rozsáhlých systémů(obecný problém centrálních DB v rozsáhlých p2p systémech) ověřují se možnosti cloudů

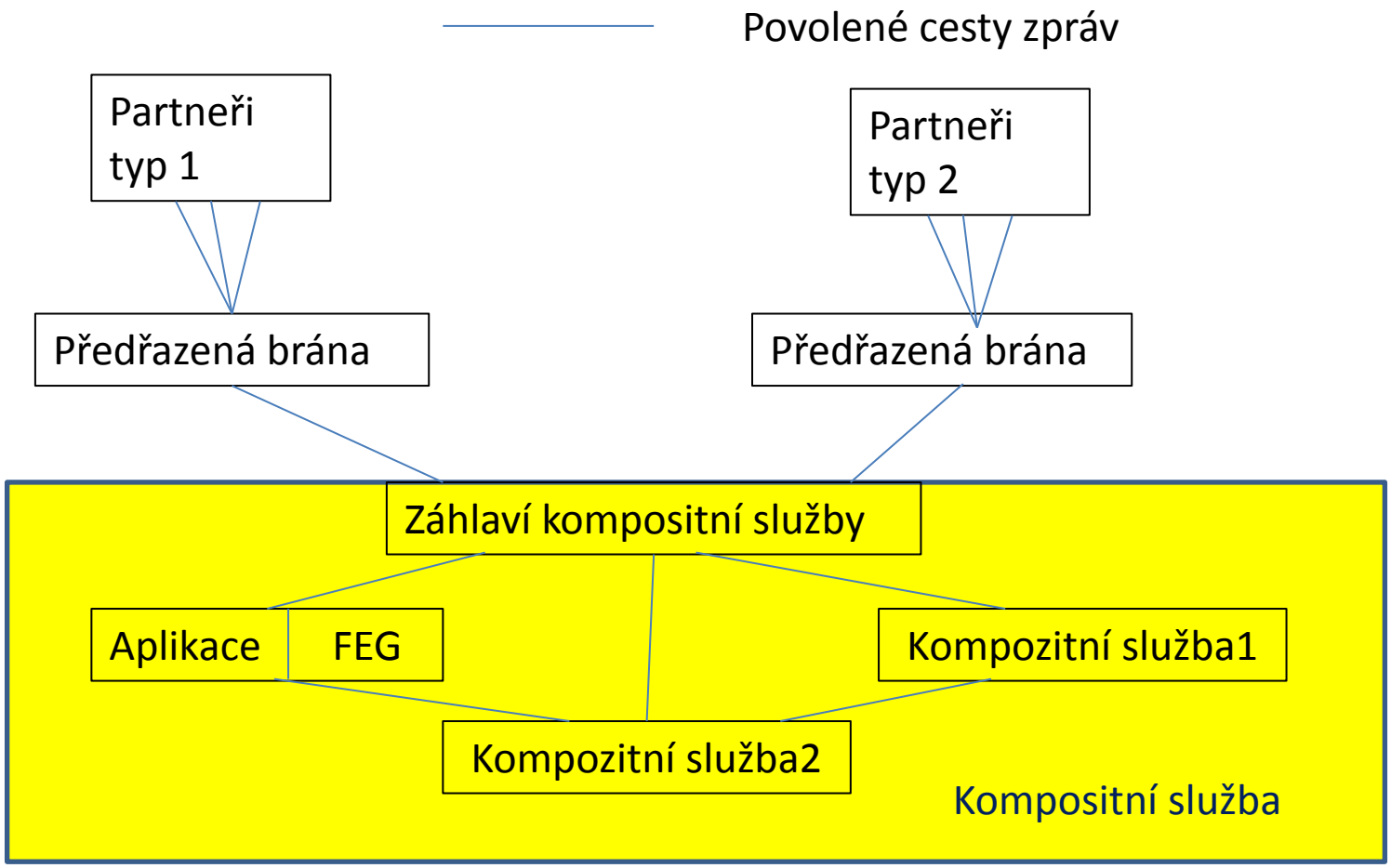
Obecné řešení architekturní služby



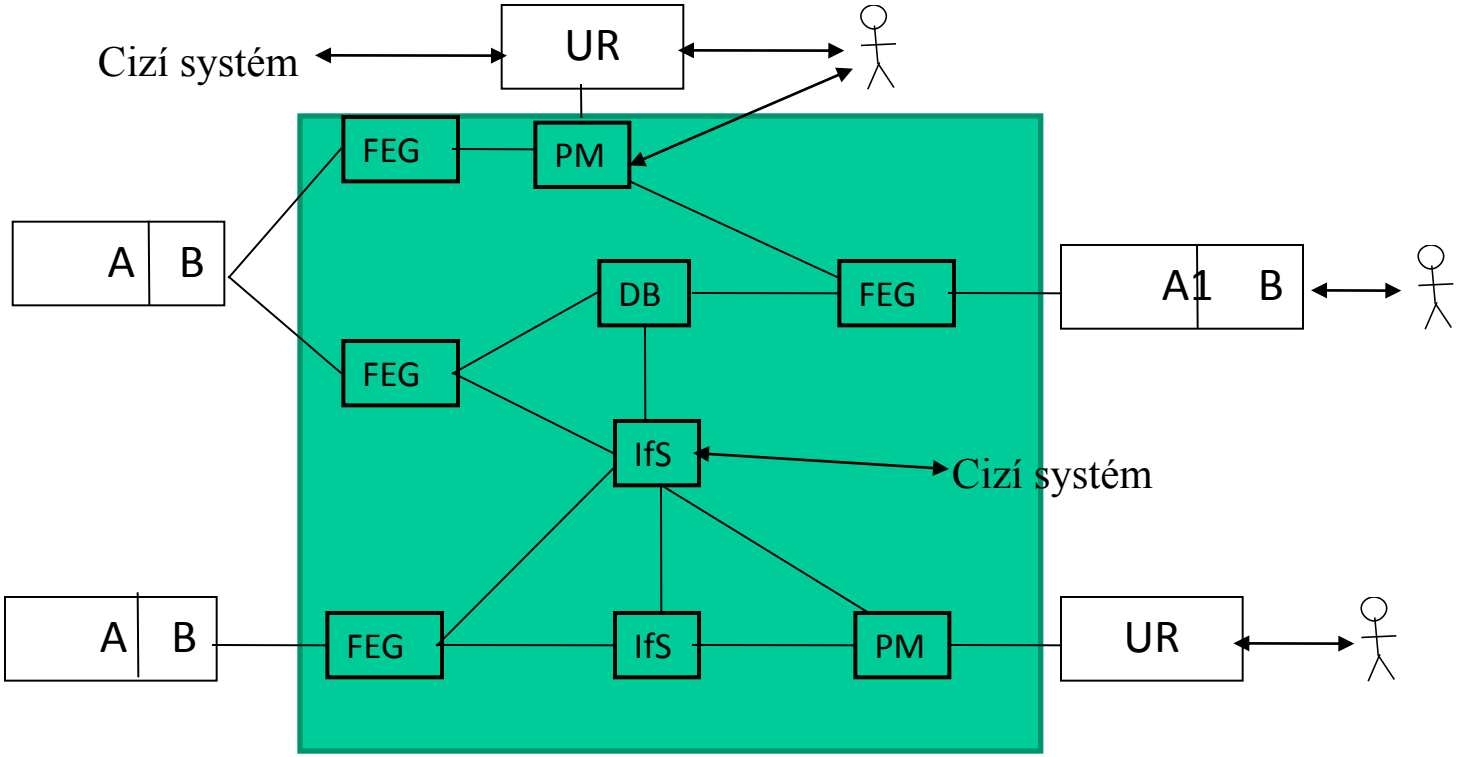
Záhlaví kompositní služby

Omezení (kontrakt resp. politika): libovolný požadavek na službu S_i od služeb z ostatních částí systému musí jít přes A.

Hierarchie kompozitních služeb

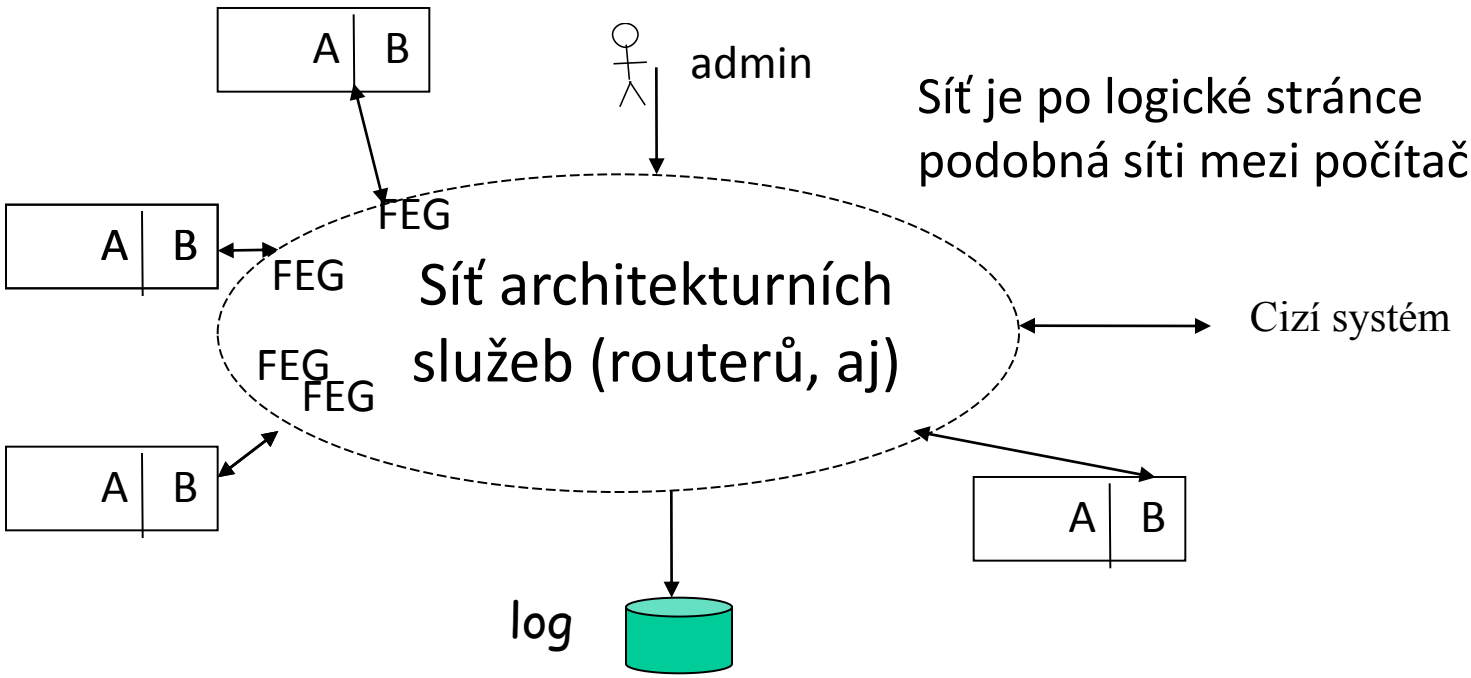


Logický pohled na SOA s architekturními službami



FEG – předřazená brána, UR – portál, PM - manažer procesu,
DB – datové úložiště
IfS - infrastrukturní služba, obvykle záhlaví kompositní služby nebo router

Jiný pohled



Dá se využít na gridech a v cloudech

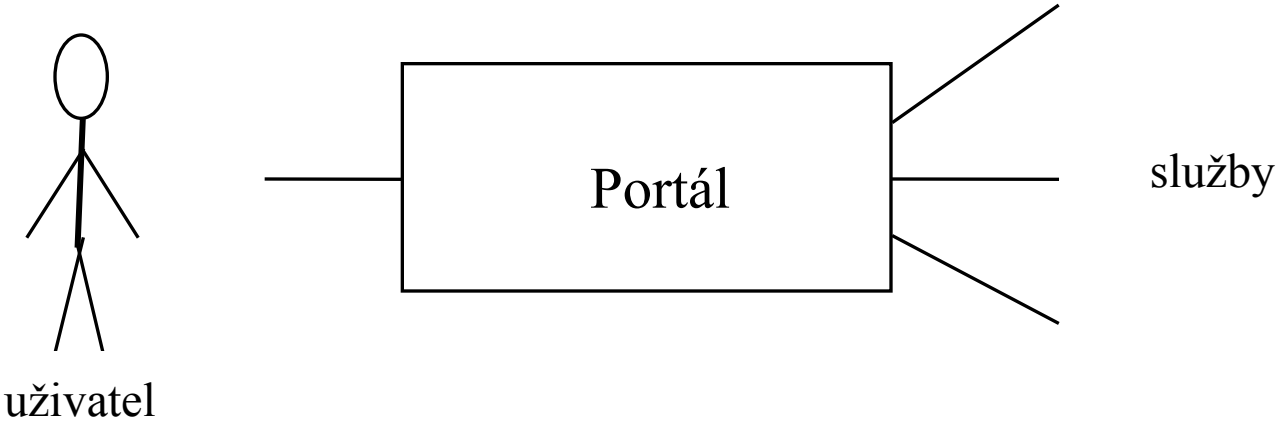
Síť se chová jako služba poskytující architekturu, funkcemi silně připomíná HW poskitující konektivitu u počítačových sítí

Architekturní služba funguje jako konektor implementovaný jako služba

Architekturní služba

- Předřazenou bránu lze snadno modifikovat tak, aby
  - Přijímala zprávy od různých zdrojů
  - *m*-tice zpráv transformovala na *n*-tice zpráv
  - Výsledné zprávy směrovala na dynamicky volitelné adresáty
- Výsledek nazveme architekturní službou ArS
  - Jsou to konektory, routery a transformatory jako služby

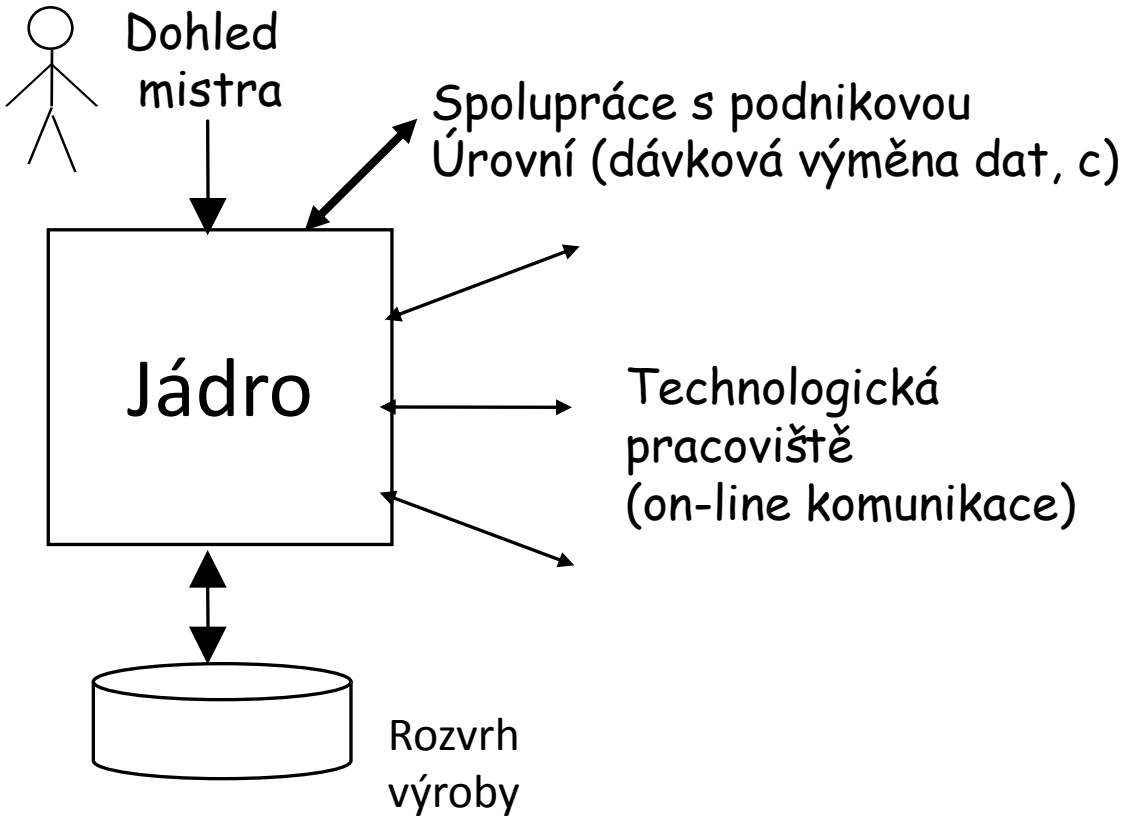
Varianta GPP (specializace), portál, vnější „služby“ jsou uživatelé



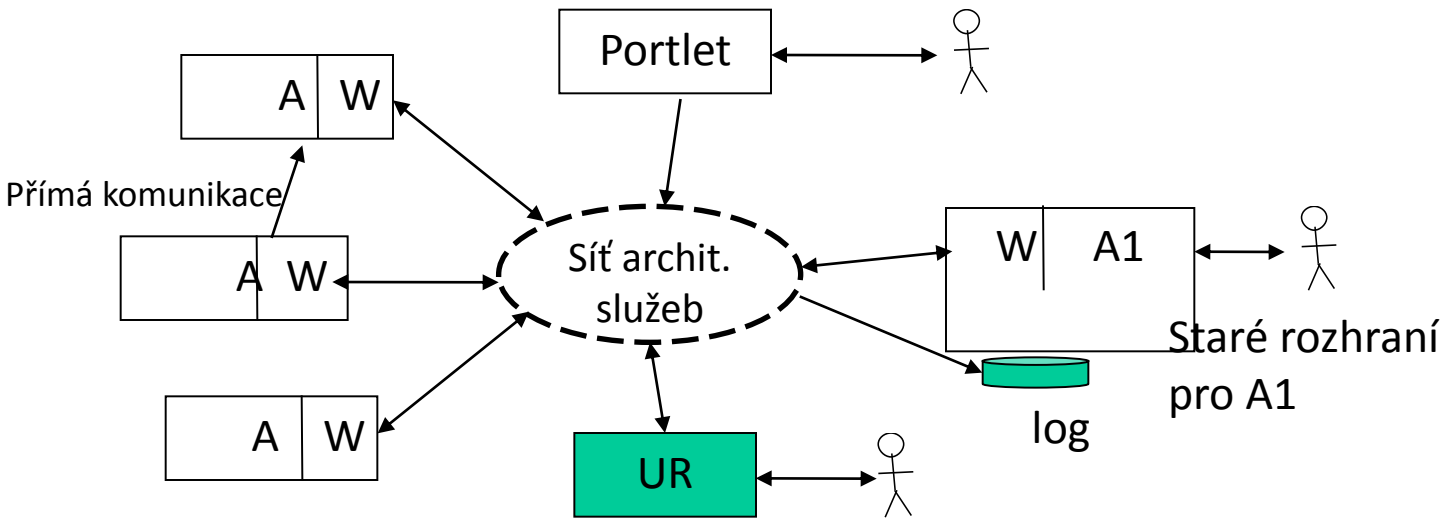
Omezení: zprávy mezi službami prochází prakticky vždy Portálem
Pro takové služby existuje skupina norem, výsledek se jmenuje

Portlet Service

Příklad prakticky použité varianty architekturní služby, řízení výroby

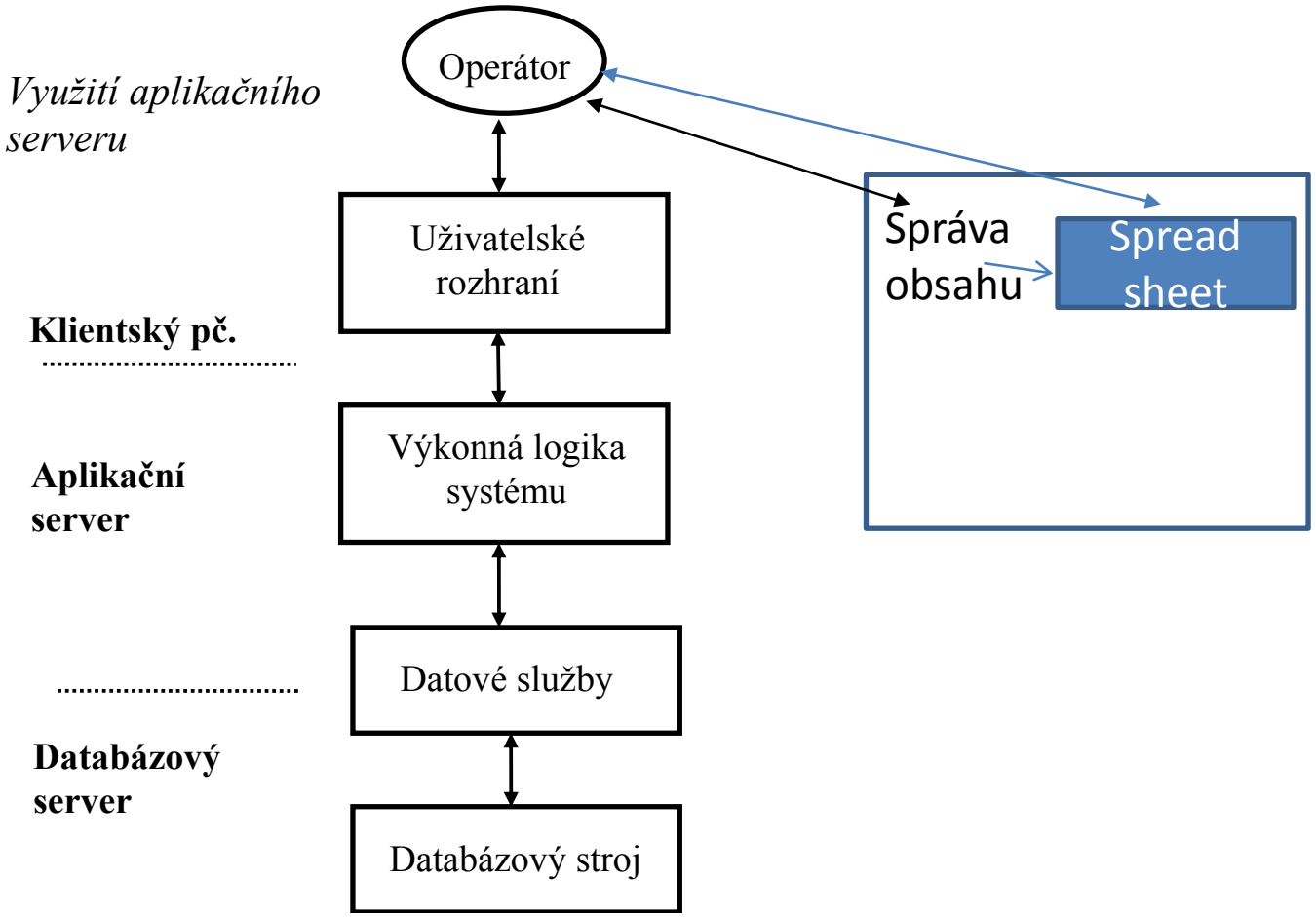


SOA s architekturními službami



Zprávy mezi službami jdou zpravidla přes několik architekturních služeb. V tomto smyslu je architektura dvojvrstvá (nearchitekturní a architekturní služby podobně jako v cloudech). Lze vytvářet logické vrstvy podle variant funkcí arch. služeb). Síť architekturních služeb je vlastně také služba = jde tedy o implementaci architektury jako služby

Tři vrstvy s podporou tabulkového kalkulátoru



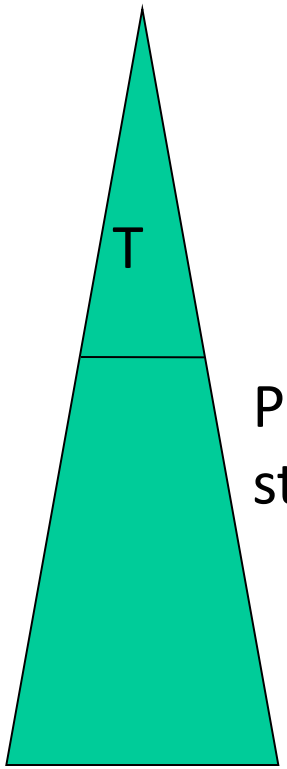
Pozorování

- Chování systému (business procesy) ovlivňuje do jisté míry sám koncový uživatel zadáváním parametrů instanciace procesu a úpravami jeho chování (to lze do jisté míry považovat za vývoj, který provádí koncový uživatel - enduser development)
  - Je snadné v konfederacích
- To je projev nových trendů ve vývoji a vlastnostech softwarových systémů, především pro malé a střední podniky a e-government

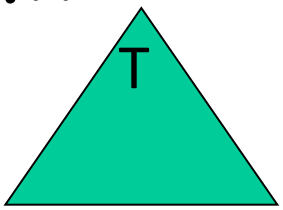
Kde hlavní přínosy

- Horizontální spolupráce přes hranice oddělení (příklad Otavan, prodejce si reservuje výrobní kapacity), ale to ohrožují některé manažery
- Dostupnost informací (PC+internet)
- Sociální kontakty, zlepšení ovzduší ve firmě
- Zefektivnění procesů
- Zpřesnění a vyšší dostupnost dat, zkvalitnění rozhodování
- Restrukturalizace obchodních procesů, větší agilita
- Zlepšení spolupráce s externími partnery

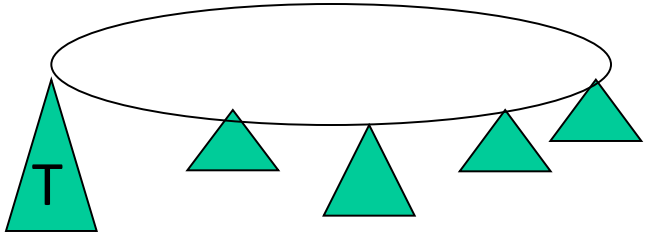
Dopad SOA na organizační hierarchii



Původní struktura



Čekalo se (šéf uřídí více lidí)



Obvykle se stalo (decentralizace)

Vždy méně středního managementu i u něho změna profesní struktury (větší samostatnost šéfů divizí) při decentralizaci, příčina odporu a neúspěchů!!! Méně středního managementu ubylo při decentralizaci (další důvod výhiodnosti decentralizace)

Co se kupodivu v některých případech zjistilo

- Při pokusu o nízkou hierarchii se neušetřili úředníci
- Jsou náznaky, že se někdy zhoršila kvalita celkového vedení podniku (Jo Owen: Flat Organizations, Flat Results)
  - Zapomělo na to, že mnohé z inteligence, dovedností, zkušeností a znalostí, které jsou konkurenční výhodou firmy, jsou pouze v hlavách středníhomanagementu
  - Srv. špatné zkušenosti s business process reingeneering (BPR)

Co se kupodivu zjistilo

- Nastala mohutná centralizace a nárůst byrokracie. Místo odstraněných článků formálního hierarchického řízení zaujaly rozšiřující se štábní útvary, jejichž úkolem bylo koordinovat trénink, personální politiku, standardy, kvalitu, komunikaci, finanční kontrolu a celofiremní iniciativy.
- Rozhodnutí, která dřív vyžadovala 2 - 3 lidi náhle vyžadovala 10 - 20 lidí.
  - Snaha o alibi??

Co se kupodivu zjistilo

- Narůstalo politikaření a zmenšovala se zodpovědnost.
- Manažeři přestávali mluvit jeden s druhým. Tam, kde dřív jeden manažer zavolal druhému, aby s ním něco probral, nebo si dojednal schůzku, nyní nastupovaly sekretářky, které musely koordinovat velké množství diářů.
- Rozmnožily se porady a interní konference.
  - Často se nevědělo jak na věc když spousta znalostí a dovedností zmizela
  - Domněnka: Použity jemnozrnné služby a komunikace

Asi se chybně upravila pravidla řízení

- Administrativní režie se zavedením automatizace kancelářských prací a dalších informačních technologií zprvu nejen nezmenšila, ale naopak se v širokém spektru odvětví vytrvale zvětšuje. (P. Attewell: Information Technology and the Productivity Paradox).
- Špatné zkušenosti s razantní změnou procesů (business process restructuring, BPR), někdy za vším jen snaha vyhodit staré znalosti se strany nového managementu
- Větší tlak na změnu pravidel je v případě fine grained services (malé sužby, krátké zprávy)

Obrana: Service governance

- Sledování služeb a jejich vývoje
- Řízení projektů
- Stanovování priorit
- Bezpečnost
- …..

Musí být za účasti CEO a uživatelů

Přednosti a nevýhody SOA

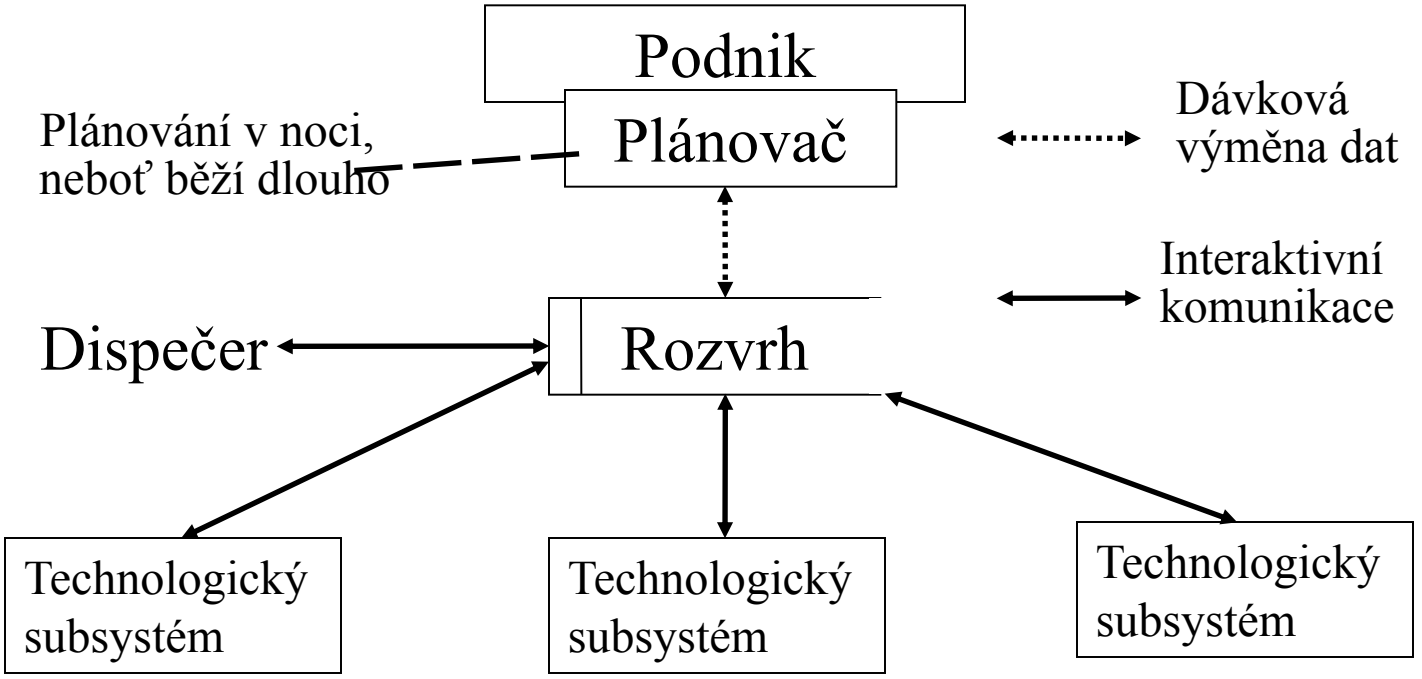
- Neví se, je-li vůbec možná model driven architecture, chybějí příklady řešení a CASE nástroje pro SOA
- Problémy s proprietálními řešeními, poněvadž pro ně nelze vždy použít standardy
  - lze zmírnit správnou strategií (užití XML)
  - může urychlit vznik uživatelských XML jazyků a tím nevýhodu změnit na výhodu
- +- Nutná úzká spolupráce mezi skoro všemi vývojáři a mnoha uživateli i z nejvyšších postů (CEO se musí účastnit specifikací toho, co bude používat)
- ++ Velmi dobré zkušenosti s provozem (dvacet let bez údržby, viz též zkušenosti s Y2K)

Pozorování

- Některé akce musí být dávkové (trvají příliš dlouho, je to nutné z podstaty věci), to platí pro některé algoritmy, pro lidi a pro procesy reálného světa
- Jsou třeba zásahy dispečera, tj. je třeba agilní provádění procesů
  - Nečekané nebo vzácné události –
    - nevyplatí se je zahrnovat do rozvrhování (Vonásek je lempl, Pepa se včera ztřískal)
    - Turbulence na trhu nebo u dodavatelů
  - Kvalita dat
    - Nedostupná, neznámá, nepřesná (mají velký rozptyl)
    - Zřídka potřebná (nevyplatí se sbírat)
  - Potřeba využít inteligenci lidí jako součásti procesů

Neinteraktivní komunikace

Spolupráce plánovacích algoritmů s provozem vyžaduje inteligenci při přenosu požadavků

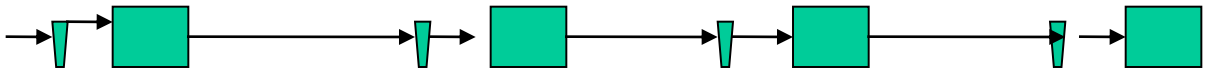


Další důvody, proč je třeba dispečer

- Data jsou nevčasná, neboť plánovací algoritmy jsou pomalé, změny se musí přesto udělat rychle, tj. musí je udělat člověk, aby výroba či obchodní proces nestály, když se objeví nečekaná překážka
- Data jsou nepřesná $\Rightarrow$plán je nedokonalý

Řízení na buffery

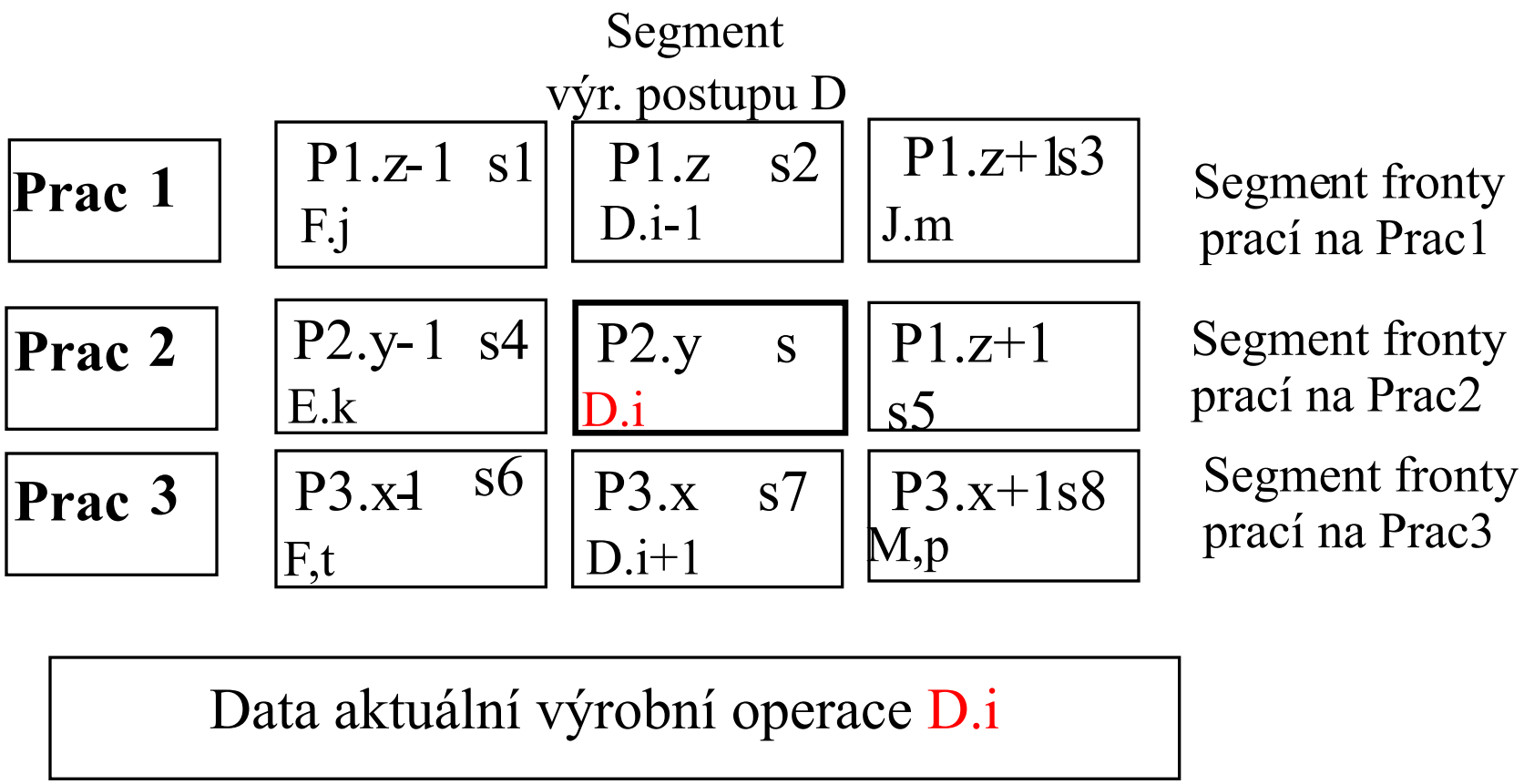
- Před každým pracovištěm fronta požadavků (buffer požadavků)



Mistr sleduje buffery a hledá pro dané pracoviště práci, když se jeho bufer nepříjemně zkracuje (řízení na průšvih, ale na průšvih, který nastane v budoucnosti). Mistr nemusí zasahovat, když je plán OK

Nutné pro výroby s krátkými sériemi a velkou variabilitou

Prostředek sledování postupu prací



Pozorování 2

- **V daném příkladu neběží jen o poskytování informací, předávají se i příkazy, služby mohou být služby reálného světa (raketové základny)**
- **Datově bohaté rozhraní je typické pro manažerskou úroveň řízení, ta se stává hlavním tématem současnosti**
- Snadné kombinování ručních a automatizovaných postupů je nejvýše žádoucí a je v konfedeaci snadná
- Vysoká autonomie služeb
- Částečný návrat DFD a datových úložišť (používání dříve zavrhované funkční dekompozice), důležité vstupní data se nemění, výpočet je vždy možné spustit znovu
- Podle současných znalostí je implementace rychlého datově bohatého rozhraní v aliancích obtížná, i když možná

Pozorování 3

- Datové úložiště může být virtuální (bez replikace dat), data poskytovaná službami mohou být čtena z jejich databáze, vypočítávána, měřena, zadávána uživateli – *klasická DB nemusí být adekvátní koncepcí (problém indexace), viz sémantický web*
- Důležité je deklarativní hrubozrnné uživatelsky orientované rozhraní
- Úložiště může být plněno z více zdrojů současně (lze použít předřazené brány diskutované níže)
- Výhodné dávkové plnění dat
  - ušetří to mnoho práce při vývoji a údržbě, stabilita řešení

Další výhody SOA

- Paradigma typické pro služby v lidské společnosti, je proto srozumitelné uživatelům
- Jedná se o jiné paradigma než objektová orientace nebo vzdálené volání procedur (kombinace s těmito paradigmaty je možná, pokud se koncepce používají jako ortogonální nástroje, často ale vede k nadměrnému počtu malých služeb – antipattern fine grained services)
- Lze poměrně snadno vylaďovat spolehlivost (doručitelnost zpráv), zabezpečení, kontrolu průběhu, efektivnost, rozhraní a umožnit ruční zásahy do průběhu procesů
- Poměrně snadné je využívání aplikací, které se nakupují, existují nebo jsou téměř nezávisle vyvíjeny

Shora nebo zdola v případě IS

- **Shora**:Od celkového návrhu k relativně autonomním převážně vlastním komponentám. Převažuje o ERP. Časté u scrum
  - Praxe ukazuje, že to lze u velkých dodavatelů a opakovaných resp. podobných řešení (customizace je častý případ), kbnfekční řešení: nějak to funguje, dá se používat, sláva to není
  - Malý na to nemají zdroje a pro malé uživatele to klade příliš velké nároky a provozní podporu a znalosti koncových uživatelů
  - Rostoucí složitost způsobuje, že se to špatně uržuje i u velkých dodavatelů

Shora nebo zdola

- **Zdola:** Integrací autonomních až nezávislých částí k celku:
  - Potřebné pro malé SW výrobce a pro SME uživatele
  - Velmi vhodné pro globální byznys procesy
  - Umožněno existencí globálních sítí a zčásti i globálními cloudy
  - Rostoucí složitost IT a rostoucí počet existujících systémů a otevřeného SW vyšuje výhodnost integrace

Horor s web services.

- Pokus vyřešit správu vlaku přísně podle norem
- Připojení kamer pomocí SOAP (tj. jako webových služeb) se s použitím SOAP rozhraní výrobců podařilo jen vyjímečně bez potíží
- Normy pro vlakovou multimediální síť a podporu řízení vlaku mají tisíce řádek a stále se mění. Nemohou tedy být bez chyb a nejasností
  - Vzorové řešení nemůže být v plné shodě s normami

Nové trendy SOA a jinde, orchestrace přístupů.

- Komunikace výměnou byznys dokumentů, SOAP document literal
- Cloud, zapojení prostředků document management systems DMS a event. Driven architecture
- Požití metod REA (modernizace procesů v diagramech toků dat)
- Někdy i přehnaná snaha standardizovat jako webové služby

10 vůdčích principů služeb v SOA

1. Explicit Boundaries
2. Shared Contract and Schema, not Class
3. Policy-driven
4. Autonomous
5. Wire formats, not Programming Language APIs
6. Document-oriented
7. Loosely coupled (stimulated by previous point)
8. Standards-compliant (if possible)
9. Vendor independent
10. Metadata-driven (XML)

Vzory a antivzory

Dobrá a špatná řešení často se vyskytujících problémů

Vzor

- Osvědčené řešení nějakého často se vyskytujícího problému
  - Fasáda
  - Broker
  - Proxy
  - Mikrokernel
- Typy
  - Architekturní
  - Návrhové
  - Zkratky

Vzor

- Osvědčené řešení nějakého často se vyskytujícího problému,
- Popis
  - ID
  - Podstata (popis podstaty úkolu)
  - Popis řešení
  - Známé případy použití
  - Rizika, příklady neúspěchu

Antivzor

- Často používané, ale velmi neefektivní až průšvihové řešení.
- Popis
  - ID
  - Podstata (popis podstaty)
  - Symptomy a důsledky
  - Hlavní příčiny
  - Známé případy použití, kdy se dá úspěšně použít
  - (Náprava refaktorizace)

Antivzory (OO), hlavní případy

- Zlaté kladívko (vše podle jednoho mustru, pro jeden typ postupu)
- Blob (univerzální třída)
- Špagetový kód
- Stálé zastarávání (přejdu na nové postupy hned, jak se objeví)
- Ostrov automatizace **Vzor v SOA**
- Používání legacy systémů **Vzor v SOA**
- Vendor lock-in

Antivzory v SOA

- Problémy s přijetím SOA
  - Nu, co je na tom nového
  - Velký třesk, přechod na SOA velkým skokem
  - Přeceňování technologických, nikoliv obchodních aspektů
  - Fine grained SOA, malinké služby na úrovni objektu nebo malé komponenty
  - Služba = Třída
- Design
  - Web service=SOA, služby být nemusí nutně webovské, problém s byznys dokumenty, normy k nim nejsou dosti vstřícné
  - Nepřijetí dokumentového (především byznys) rozhraní
  - Fine grained messages
  - Nadměrná standardizace, jsou i protichůdná doporučení

Antivzory v SOA

- Design
  - Ne *legacy!!!!!* …..Základní antivzor, vzor v OO
  - **Ne dávkovým subsystémům** (nejdůležitější)
  - Standardizační paralýza (použití nedokonalých, nevhodných, či příliš komplikovaných standardů)
  - Všechno znova (nepoužívat hotové),
  - Velký třesk (všechno naráz)
  - Web service=SOA, služby nutně webovské, nepoužití dokumentově orientovaných rozhraní
  - Ne datovým úložištím, jsou zastaralá

  Fine grained messages (často důsledek použití SOAP-rpc)
  - Problematická centralizace
    - UDDI

Antivzory v SOA

- Implementace
  - Fine grained interfaces (Chatty services)
  - Point to point services (důsledek používání SOAP-RPC, omezené používání p2p přístupu, tj. asynchronosti služeb)
  - Obří komponenty, nevhodně chápané vrstvy (proti obvyklému chápání datové úložiště může zajišťovat transportní služby ale také orchestraci služeb)
  - Strojová byrokracie v SOA (centrální služby)
    - Částečná výjimka u architekturních služeb a byznys procesů

ITIL and SOA: Can they Co-Exist?

Vijay Raghavan
TowerStrides Inc.

scaling new Frontiers …

Why ITIL

- Organize large, heterogeneous Chaotic IT Systems
- Instal best practices to avoid costly errors
- Gain Credibility, Cut Costs
- Improve Service
- Better Asset Utilization
- Helps you to focus on critical business processes

- Lze využít v SOA

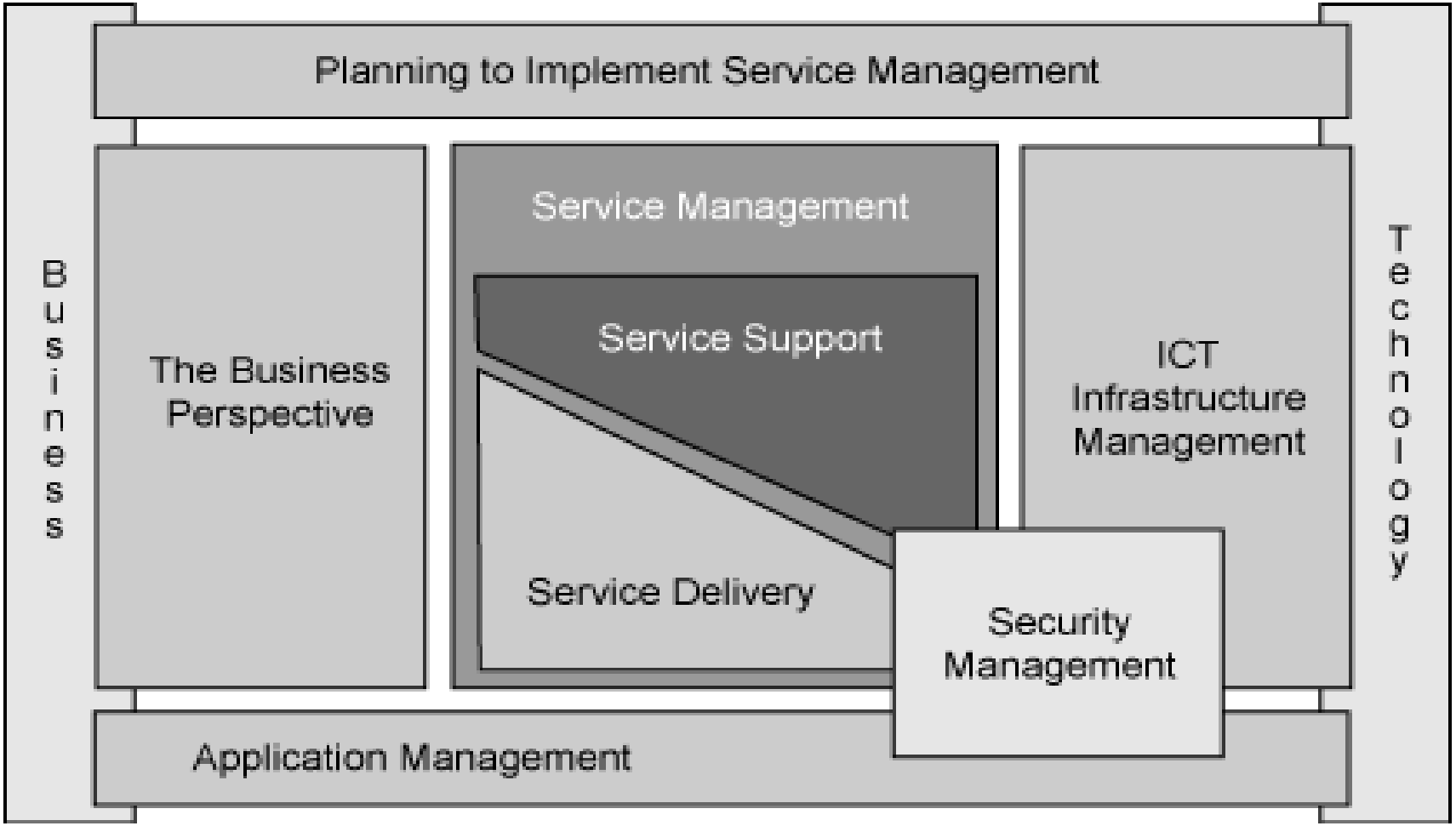
The common Ground

- SOA is not about Web services, it is about an approach that creates agility and responsiveness in both IT and business.
- That responsiveness to environmental conditions requires monitoring, reporting, and responding
- ITIL/ITSM focuses on just that.
- ITIL is all about governance, and SOA needs a good healthy dose of governance to move to the next level.

The ITIL v2 is defined in a collection of seven books,

- *The Business Perspective*
- *Planning to Implement Service Management*
- *Service Delivery*
- *Service Support*
- *Security Management*
- *Application Management*
- *ICT Infrastructure Management*

The seven ITIL books



http://www-128.ibm.com/developerworks/ibm/library/ar-soaitil/

ITIL v3, core 5 volumes proti v2.0 silně upraveno

Přiblížení k SOA

Standardy ISO 20000, BS 15000

People and Process

- ITIL is all about people and process
- SOA cannot be successful if People and process are not together
- IT and Operations come together as part of SOA.

Book 1: Service Strategy

- Identification of market opportunities for which services could be developed in order to meet a requirement on the part of internal or external customers.
- The output is a strategy for the design, implementation, maintenance and continual improvement of the service as an organizational capability and a strategic asset.
- Key areas
  - Service Portfolio Management
  - Financial Management.

Book 2: Service Design

- Activities that take place in order to develop the strategy into a design document which addresses all aspects of the proposed service, as well as the processes intended to support it.
- Key areas
  - Availability Management,
  - Capacity Management,
  - Continuity Management and
  - Security Management.

.

Book 3: Service Transition

- Implementation of the output of the service design activities and the creation of a production service or modification of an existing service. There is an area of overlap between Service Transition and Service Operation.
- Key areas
  - Change Management,
  - Release Management,
  - Configuration Management and
  - Service Knowledge Management.

Book 4: Service Operation

- focuses on the activities required to operate the services and maintain their functionality as defined in the Service Level Agreements with the customers. Key areas of this volume are
  - Incident Management,
  - Problem Management and
  - Request Fulfillment.

Book 5:
Continual Service Improvement

- Ability to deliver continual improvement to the quality of the services that the IT organization delivers to the business. Key areas of this volume are
  - Service Reporting,
  - Service Measurement and
  - Service Level Management.

ITIL v 2.0

1. *The Business Perspective*
2. *Planning to Implement Service Management*
3. *Service Delivery*
4. *Service Support*
5. *Security Management*
6. *Application Management*
7. *ICT Infrastructure Management*

ITIL v 3.0

1. **Service Strategy**
2. **Service Design**
3. **Service Transition**
4. **Service Operation**
5. **Continual Service Improvement**

Nezohledňuje vliv architektury!!!!, nezná pojem konektoru

ITIL - možná past

- Úplná změna struktury ITIL svědčí o nedomyšlenostech
- Snaha o návrh výhradně shora nemusí být to pravé
- Výhoda jednotné terminologie a pojmů
- Není jasné, zda ITIL bude úspěšné v ČR, otázky jsou i nad výsledky v jiných zemích