Orientace na služby – klíčové paradigma současného softwaru

Dávno známé, přesto nové

Varování

Budu říkat zřejmé věci, kterých si však ke své škodě nevšímáme. Opomíjení samozřejmostí je hlavní problém IT všeobecně a softwaru zvláště.

Varování 2

Problémem je, že servisní orientace je zdánlivě samozřejmá. Paradoxně ji právě proto nevěnujeme dostatek pozornosti a nejsme ji proto schopni používat, ačkoliv se stává vedoucí filosofií současného softwaru. Ten se stává síťový z logického hlediska

Nová móda?

Počet výsledků vyhledávání

- Service oriented architecture +software 250000
- Dtto +journal 25000
- Dtto +ISBN 10000

Mnoho komerčních produktů, různé definice a různé varianty podobného přístupu

- Enterprise service bus, NetViewer
- Web services, semantic web, …
- Composite applications, assembled applications

Příklady SO ze života

- *e*-komerce
- Informační systém státní správy
- Spolupráce zdravotnických zařízení
- Možná implementace CRM, SCM
- Informační systém globálního podniku
- Řízení procesů (soft real-time)
- Atd.

Termíny:***Aliance*** (partner se musí vyhledat) – ***konfederace*** (partneři jsou v podstatě známi)

Budeme se zabývat zpravidla konfederacemi. Mnohé lze použít i pro aliance

Konfederace a aliance

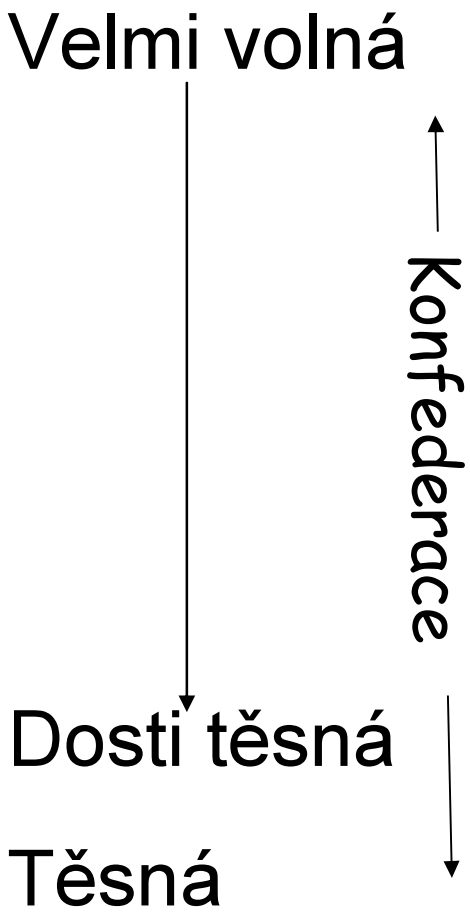
- ALIANCE
  - Na začátku komunikace se partner musí vyhledat. Typické pro e-komerci, reservační systémy atp.
- KONFEDERACE
  - partneři jsou zpravidla známi. Široká paleta aplikací, historicky nejstarší SOA systémy (e-government, řízení gloálních společností, výrobní systémy, koalice podniků a zdrav. zařízení)
- Existují i mezistupně mezi konfederacemi a aliancemi, nejsou ale časté. Konfederace může využívat služby vyvinuté pro aliance

Konfederace volné a úzce vázané

- Konfederace mohou být různého typu
  - IS podniku, části musí poslouchat centrum a systém není extrémně velký
  - IS státní správy nebo globálního podniku, části jsou téměř nezávislé

Těsnost vazeb

- *e*-komerce
- Informační systém státní správy
- Spolupráce zdravotnických zařízení
- Možná implementace CRM, SCM
- Informační systém globálního podniku
- Řízení procesů (soft real-time)
- OO aplikace



Příklady SO ze života

- *e*-komerce
- Informační systém státní správy
- Řízení procesů (soft real-time)
- Spolupráce zdravotnických zařízení
- Možná implementace CRM, SCM
- Informační systém globálního podniku
- Lokální divize globálního podniku, menší podnik.
- Atd.

Málo vázné

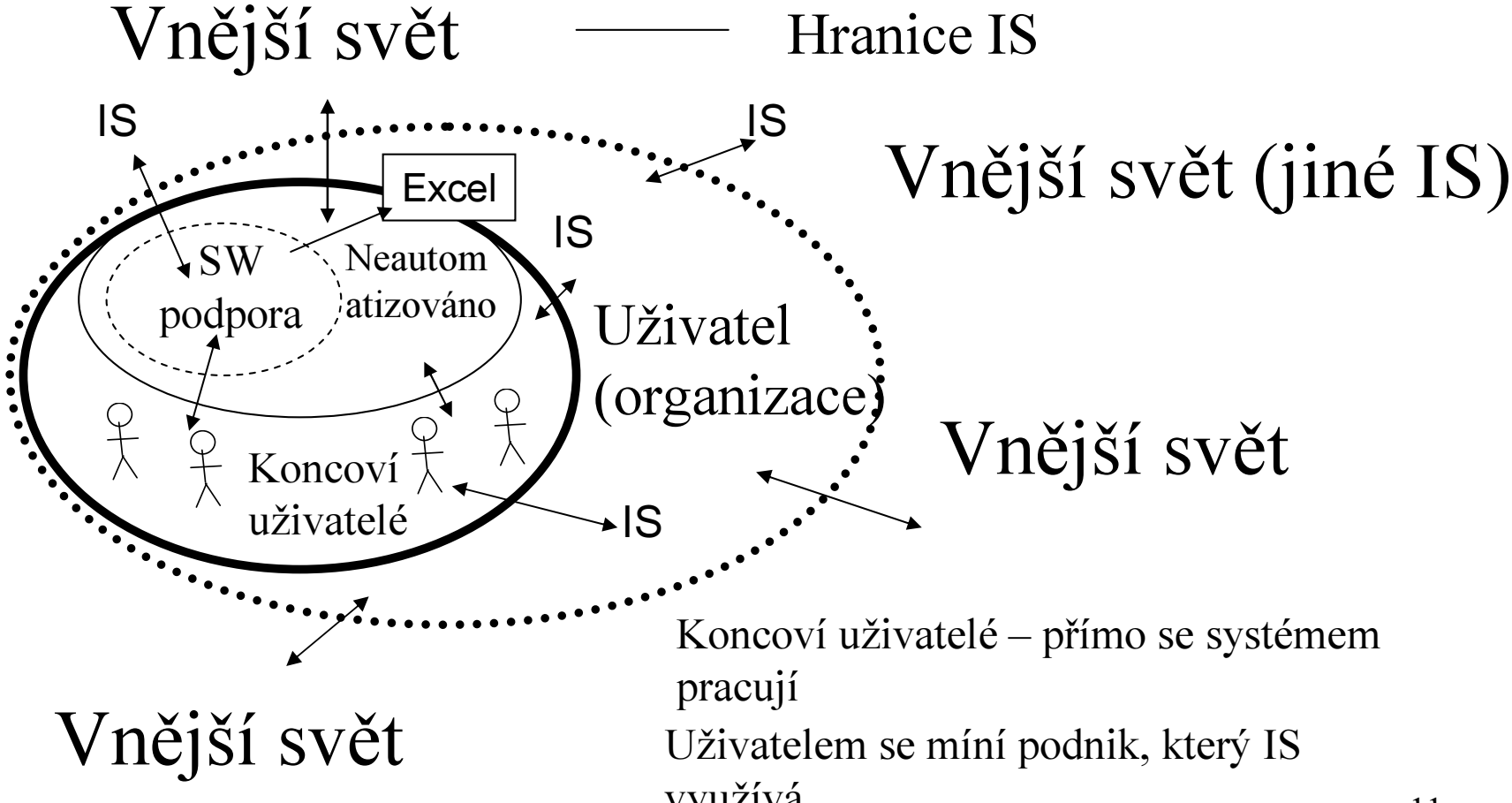
} Enterprise Service Bus

Silně vázané

Prvky SO se dosti běžně používají

- *Propojování aplikací interaktivní i dávkové*
- *IS a jiné programy spolupracují*

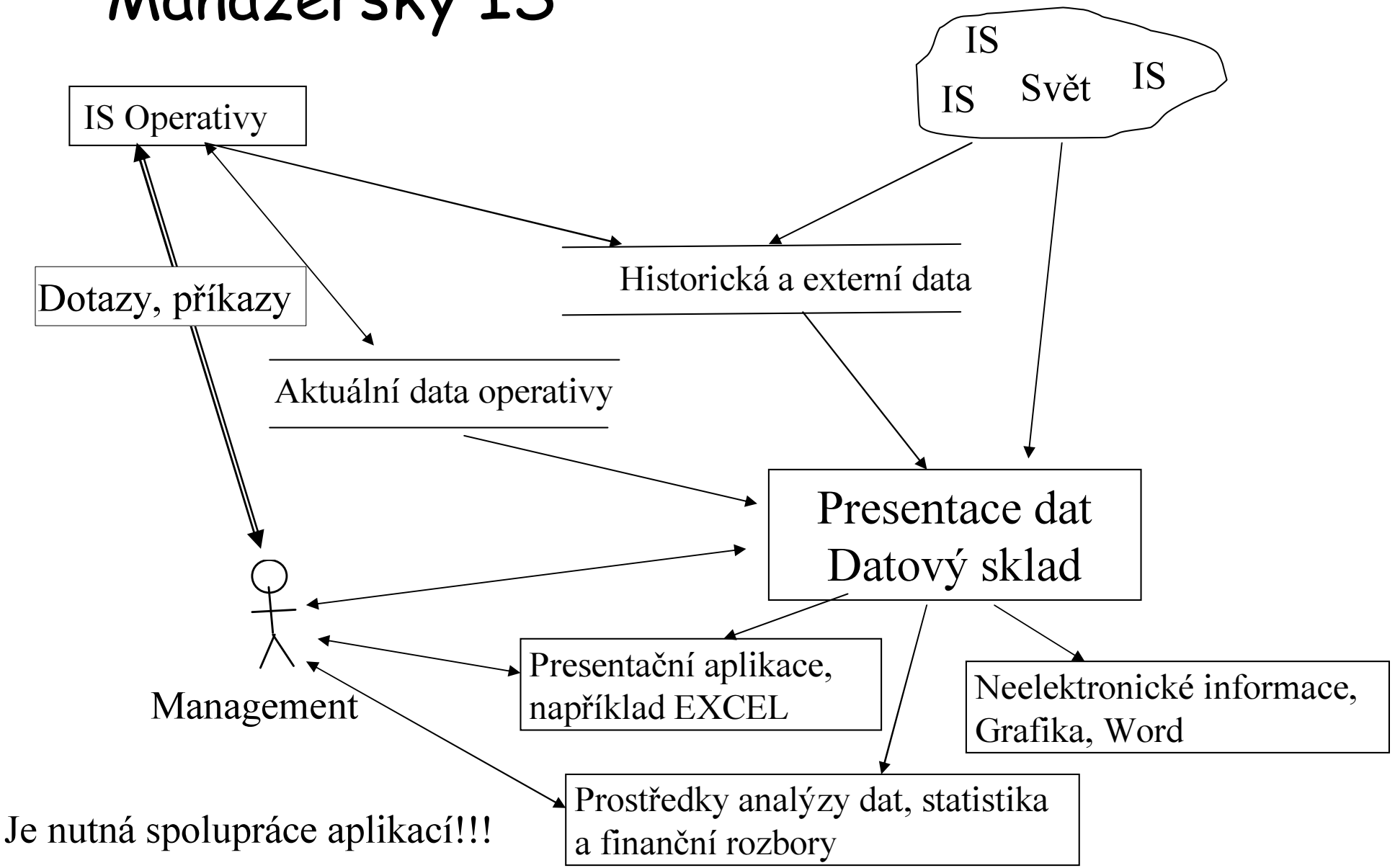
Informační systém je vždy součástí většího systému obvykle zahrnujícího i lidi a neautomatizované činnosti



Koncoví uživatelé – přímo se systémem pracují

Uživatelem se míní podnik, který IS využívá

Manažerský IS



Je nutná spolupráce aplikací!!!

Systémy mezi sebou spolupracují
Aby mohly být používány
rozumně, musí spolupracovat
podobně jako služby reálného
světa, asynchronně reagovat na
požadavky z různých zdrojů a být
používány jako černé skříňky. To
reálně lze dosáhnout jen, tvoří-li
virtuální p2p síť.

Pozorování

- Spolupráce může být dávková
  - exportem a importem dat
  - pomocí společného (virtuálního) datového úložiště plněného dávkově
- Spolupráce může být interaktivní
  - Přímá výměna zpráv
  - Aktivní databáze
- Některé služby mohou být dávkové (Excel)

Pozorování

- Spolupráce může být dávková
  - exportem a importem dat
  - pomocí společného (virtuálního) datového úložiště plněného dávkově
- Spolupráce může být interaktivní
  - Přímá výměna zpráv
  - Aktivní databáze
- Některé služby mohou být dávkové (Excel)

Jak zajistit, aby jedna služba čekala na dokončení jiné služby

- Odpověď obsahuje identifikaci volání, tj. odesilatele požadavku, informace, které umožní pokračovat v daném vláknu
- Je možné systémově a obecněji, proto standard service bus

Praha 1.11.2005

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)

Servisně orientovaná architektura

Soubor softwarových komponent spolupracujících obdobně jako služby reálného světa

- Všechny služby jsou stále aktivní
- Vyřizují požadavky asynchronně

Služby a infrastruktura, zajišťující jejich spolupráci (middleware), včetně možnosti čekání na odpověď

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 a modernizace, znovupoužitelnost, …..
- Změna možností technologie, konfederace je dostatečně efektivní
  - Dostupnost a kapacita komunikační infrastruktury
  - Existence programovatelného uživatelského rozhraní, existence vyhovujících norem

Aliance a konfederace a normy

Aliance:

- Partner se na začátku komunikace vyhledává, nelze předpokládat, že je na začátku komunikace znám. Je proto nutné použít celosvětové standardy

Konfederace:

- Partnerské komponenty se znají, lze dohodnout proprietální pravidla a protokoly, je-li to výhodné
- Komunikace může být i jiného typu, než výměna zpráv založená na internetu

Ukážeme, že servisní orientace je specifické paradigma.

Proto ji není snadné (zprvu) používat

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 a změnami struktury podniků

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

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

Registr účtů a dluhů

Lepší 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.

Nutnost konfederace, 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

Důsledky I.

- 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 používaly IS partnerů pokud možno bez úprav

Nutnost 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
   - Vložili 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, změny se 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
   - Mocensky se lidé i úřady snaží zachovat svoji autonomii a své joby
   - Autonomie je výhodná pro získání výhod (známosti mimo úřad, různé výhody, provize až korupce)
   - Je nereálné všechny systémy přepisovat znovu (cena, termíny, spolehlivost) a i rekonstrukce nemůže mít za cíl monolitické řešení
   - Nelze připustit přílišnou závislost na jednom dodavateli

Kdy je nutné 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

Důsledky II

- 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 globálních podniků !!!

Kdy je nutné použít SO, příklad nákupního kartelu amerických automobilek

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

Jako černé skříňky musí také komunikovat IS jednotlivých složek zdravotního systému.

Společné vlastnosti všech servisně orientovaných systémů

- 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 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 ….

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, hned jak dokončím, musím začít znovu)
  - 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)
  - politické a mocenské důvody
  - levoty (provize, úplatky, snahy o to, aby nebyl dohled)

Kdy je nutné použít SO

- 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)
- 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

Výhody servisně orientované architektury

- Chyby zůstanou lokalizované
- 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

Výhody servisně orientované architektury

- Systém je možné zprovoznit v rozumném čase, s rozumnými náklady
- Chyby zůstanou lokalizované
- 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

Výhody 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

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

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

Proč až nyní takový poprask

- Informační systémy musí být nyní *dekomponovány* do služeb pro zachování investic,
  - dříve to nebylo ultimativním požadavkem, systémy bylo možné budovat znovu od počátku, některé komponenty *musí* být integrovány jako černé skříňky (IS externích organizací a třetích stran)
- Nebyla *vhodná infrastruktura* pro všeobecné použití (internet) a výkon hardwaru včetně komunikací

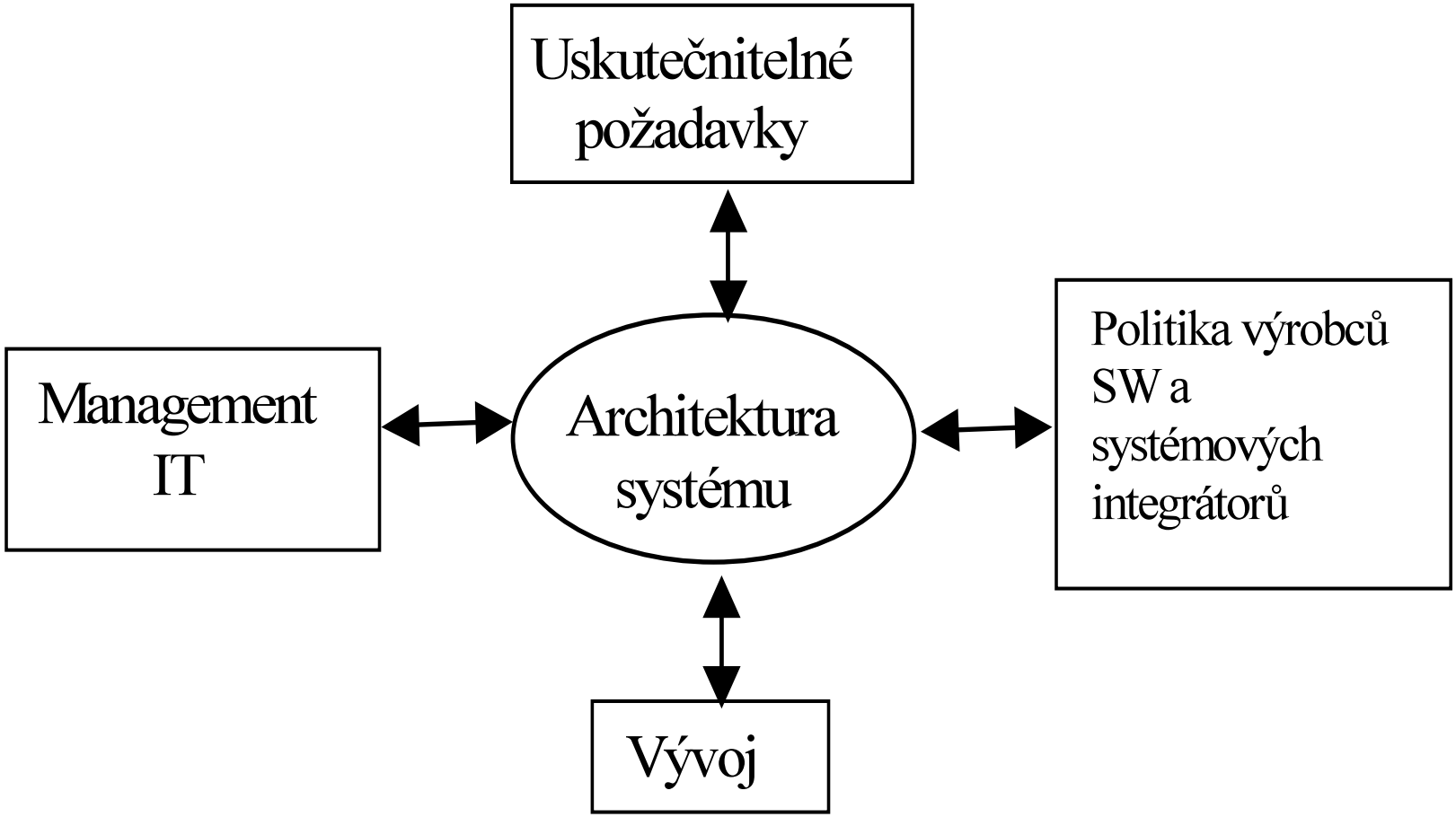
Proč až nyní takový poprask

- *Marketingové důvody*, neochota nabídnout zákazníkům větší samostatnost, využití stávajících customizovaných systémů, nutnost změny obchodní strategie výrobců softwaru
- Utajování metodologie SOA jako *konkurenční výhody nebylo možné do nekonečna, dnes se stává kontraproduktivní*

Proč až nyní takový poprask

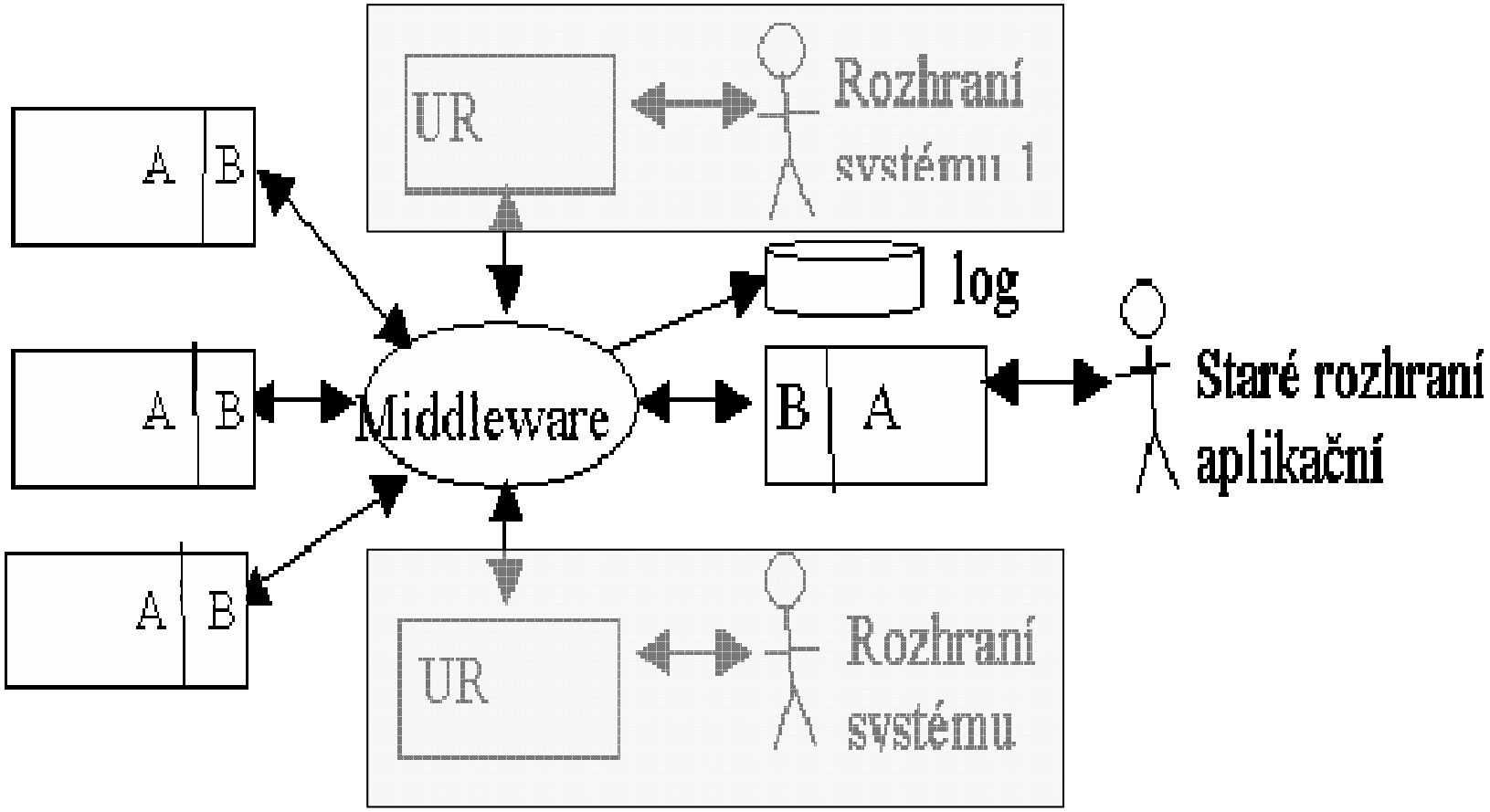
- Pro mnohé *nové paradigma* – změna filosofie, technik, dovedností, nepříjemná nutnost používat existující zdánlivě zastaralé aplikace a kombinovat dávkové a interaktivní způsoby práce a navíc i datově a příkazově orientovaná rozhraní, *nové paradigma i pro managementy dodavatelů i zákazníků*!
- Potřeba uživatelsky orientovaného rozhraní => *nutnost hlubší spolupráce většiny vývojářů s uživateli z různých stupňů organizační hierarchie*, uplatňování principů agilního vývoje, viz níže

Klíčová role architektury



Praha 9.11.2005

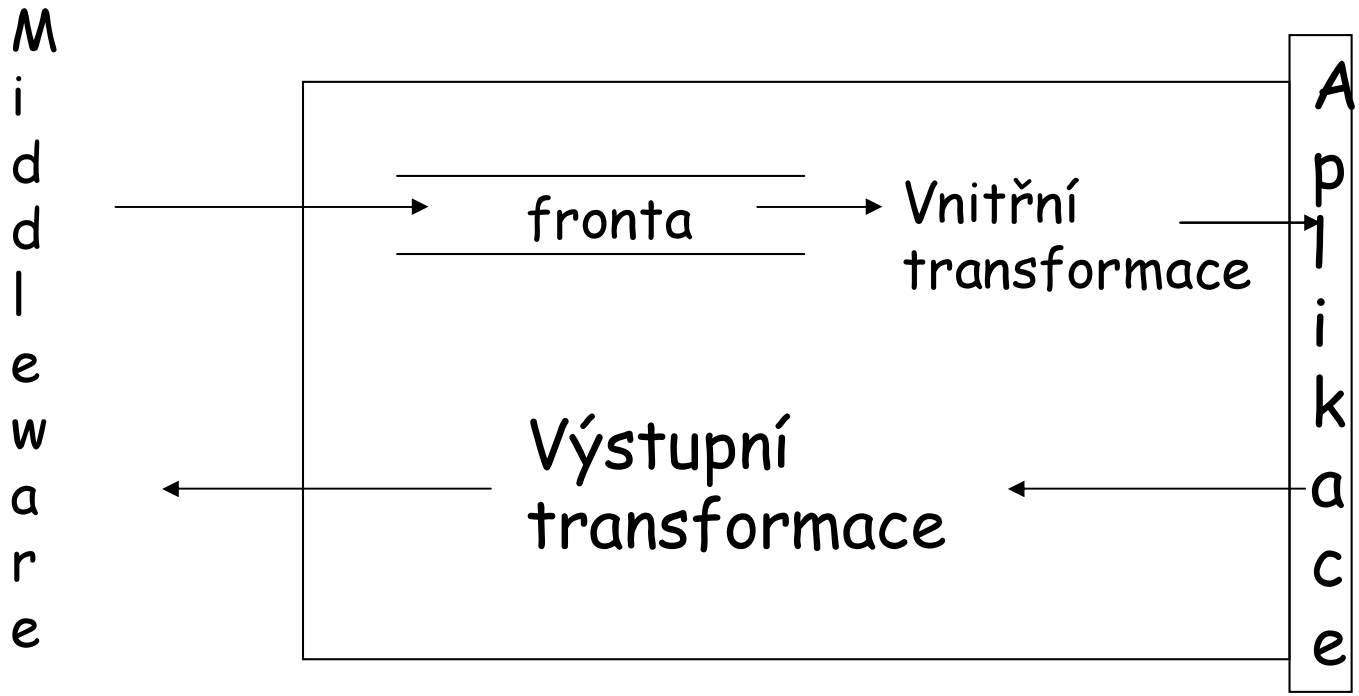
Architektura aliance a konfederace



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 podstatně vývoj.

Struktura brány



Brána má interní skrytou datovou strukturu

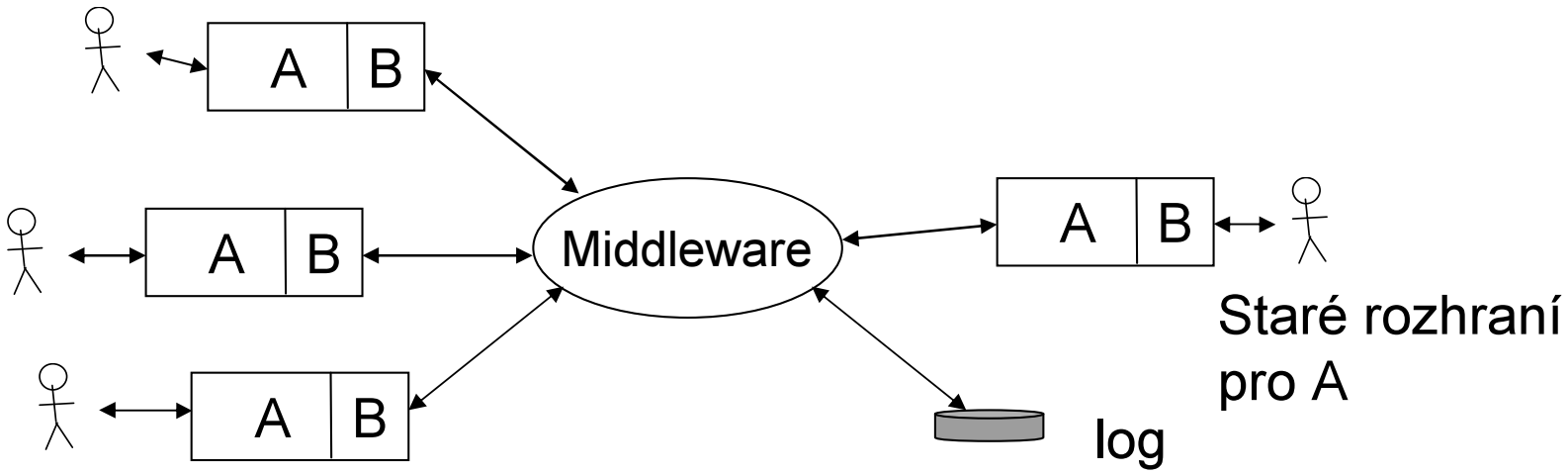
Důsledky p2p architektury

- Stabilita systému zásadním způsobem závisí na stabilitě (neměnnosti) rozhraní
- Rozhraní se bude málo měnit, pokud nebude záviset na implementačních detailech a změnách úrovně technologie (je-li např. deklarativní a uživatelsky orientované, uživatelé v operativě používají léty ověřené postupy, viz účetnictví)

Důsledky p2p architektury

- P2p systémy se málo kamarádí s centralizovanými službami, i jen funkčně centralizovanými Takové služby lze sice někdy používat, lze však očekávat problémy
- Přidání/ubrání služeb je relativně snadné, pokud mají vhodné rozhraní
- XML a www výhodou, ne však podmínkou

Obvyklá architektura aliance, *e*-komerce



Podmínky spolupráce v alianci

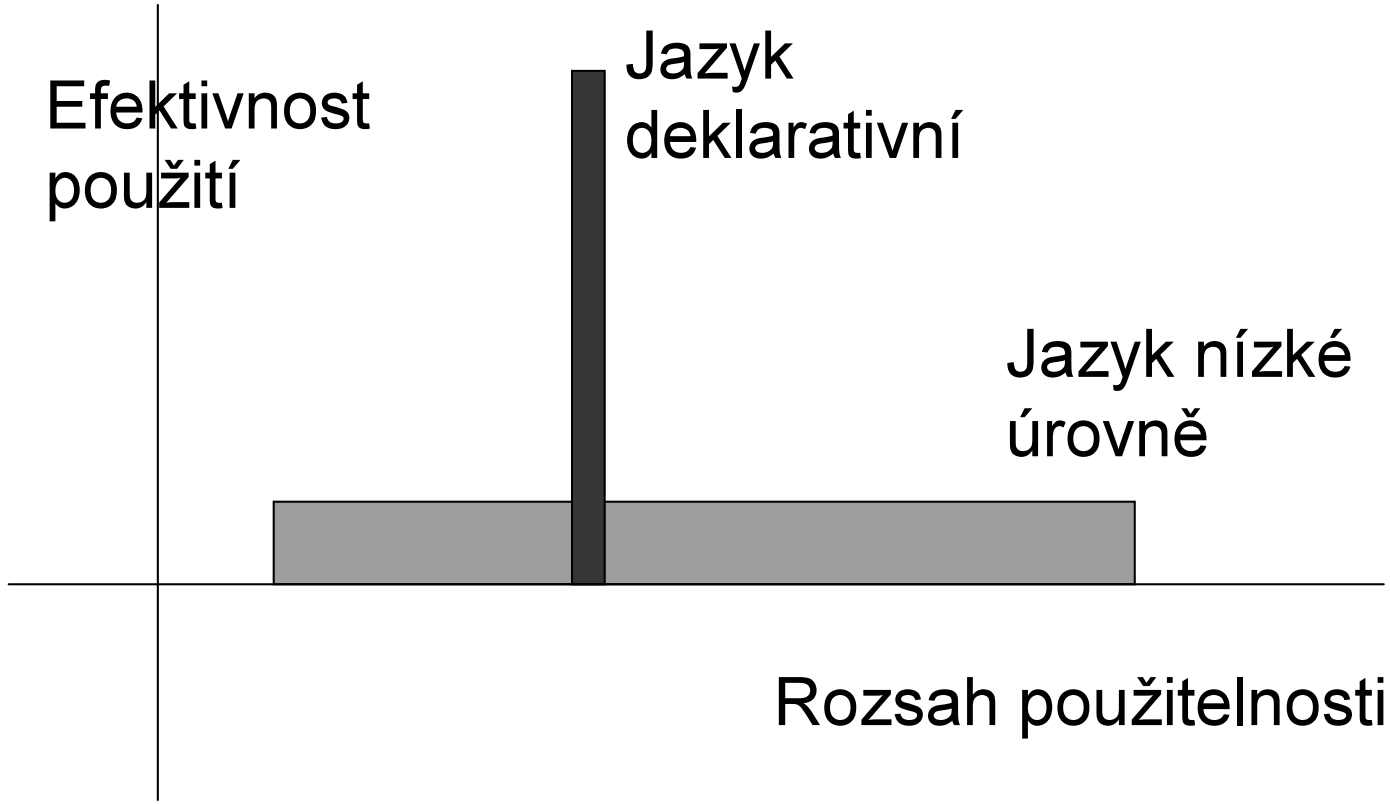
Partner se musí nejdříve vyhledat ⇨

- Partneři dostupní přes veřejný celosvětový middleware
- Rozhraní služeb na obecných standardech
  - Jazyk zpráv standardizován, v jistém smyslu univerzální
  - Přístup přes standardní nástroje (prohlížeče)
  - Poskytované služby popsány standardním způsobem, popis dostupný i pro programy (WSDL)
  - Výhodný je telefonní seznam (UDDI), ale to je centrální služba a mohou s tím být potíže v p2p prostředí, to se již potvrzuje (údržba seznamu)

Komunikace formou zpráv

- žádná data v middlewaru, na internetu se datové úložiště obtížně implementuje → bezstavová komunikace!!
- Pokusy to změnit - sessiony

Deklarativnost kontra procedurálnost



Formáty zpráv v alianci

- Univerzální řešení: univerzalita $\Rightarrow$ nízká úroveň $\Rightarrow$ procedurálnost $\Rightarrow$ SOAP
  - Programátorsky orientováno $\Rightarrow$ omezení role uživatelů při definování a řízení business procesů a zásazích do jejich chodu, není výhodné pro uživatelsky orientovaná rozhraní
  - Implikuje tvar specifikace služeb (WSDL) a telefonního seznamu (UDDI)
  - Vhodnější pro operativu a pro rozhraní bez vlastní inteligence

Nevýhody aliancí

- **Služby nemají uživatelsky orientované rozhraní – obtíže při použití (např. upodnikových procesů) i při vývoji (specifikace)**
- Potíže při řešení sporů, odpovědností a ad hoc zásazích do procesů (nečekané události, nové informace), zprávám koncoví uživatelé a experti nerozumí
- Změny obchodních procesů je nutné řešit uvnitř služeb
  - Neflexibilní, u služeb s vlastnostmi černých skříněk (produkty třetích stran) téměř nereálné

Nevýhody aliancí

- Existence centrálních služeb je proti duchu p2p
- Rozhraní je pracné, není uživatelsky orientované a má tendenci zviditelňovat implementační detaily
- Potřebné normy se rychle mění a rychle rostou => nestabilita rozhraní

Výhody aliancí

- V *e*-komerci v podstatě jinak nelze
- Služby lze programovat autonomně, podobně jako kdysi programy používající hloupé terminály
- Využívají se webovské služby a výsledky výzkumu sémantického webu
- V lidské společnosti fungují služby podobně, lze použít analogické postupy

Proč konfederace

- Chceme-li použít uživatelsky orientované nebo existující rozhraní a middleware jiný než webovský
- Propojování systémů různých zdrojů
  - Existující aplikace, vývoj, třetí strany
- a různých vlastníků,
  - Organizační subsystémy decentralizace
  - Spolupráce samostatných podniků
- a různých technologií a úkolů
- Příliš velký systém na monolit
- Chceme inteligentní rozhraní, dávkové techniky
- SW inženýrské přednosti
  - inkrementálnost, stabilita, láce, …

Souhrn hlavních rysů konfederace

- Síť autonomních komponent, které mají rysy nebo přímo jsou aplikacemi
- Při navazování komunikace se partner nemusí zpravidla vyhledávat (je znám předem)
- Komponenty jsou používány jako černé skříňky
  - Komplexní funkcionalita, komplexní rozhraní
  - Různé technologie, různí výrobci
  - Na různých místech
  - Peer to peer
- Výkonný middleware s podporou flexibilního uživatelského rozhraní a s dalšími funkcemi o které se může dělit s komponentami (kryptografie, směrování, optimalizace polohy komponent,…)

Vlastnosti konfederací

+ Lze dohodnout pravidla šitá na míru
  - formát zpráv může být deklarativní, uživatelsky orientovaný – *klíčová přednost*, snazší specifikace a snazší integrace uživatelů do procesů
  - lze použít i jiný než webovský middleware, lze kombinovat různé technologie middlewaru i komunikace
  - služby mohou zahrnovat i služby reálného světa (řízení procesů, uživatele), ?doba odezvy?, na webu složitější

Vlastnosti konfederací

+ Lze dohodnout pravidla šitá na míru
  - lze snadno kombinovat dávkové a interaktivní zpracování
  - lze kombinovat funkcionální a objektové techniky
  - snadná integrace produktů třetích stran a existujících aplikací
  - snadné modifikace a výhodné SW inženýrské vlastnosti
  - k dokumentaci často stačí popis rozhraní služeb
    - výhodné pro agilní formy vývoje

Vlastnosti konfederací II

++ Služby mohou být i informační systémy jednotlivých organizací a organizačních jednotek nebo nakupované/vyvíjené aplikace a dokonce přímo technologie či jednotliví pracovníci (za vhodným rozhraním), výhody

- + snazší decentralizace, prodej/akvizice částí podniku
- + nástroje kontroly spolupráce částí
- + snazší spolupráce s obchodními partnery (CRM, SCM, nákupní koalice..) a se státní správou
- + podpora strategických záměrů managementu

Implementace podnikových systémů ve formě aliance je riskantní, neefektivní a pracná (proto raději např. enterprise service bus)

Vlastnosti konfederací III

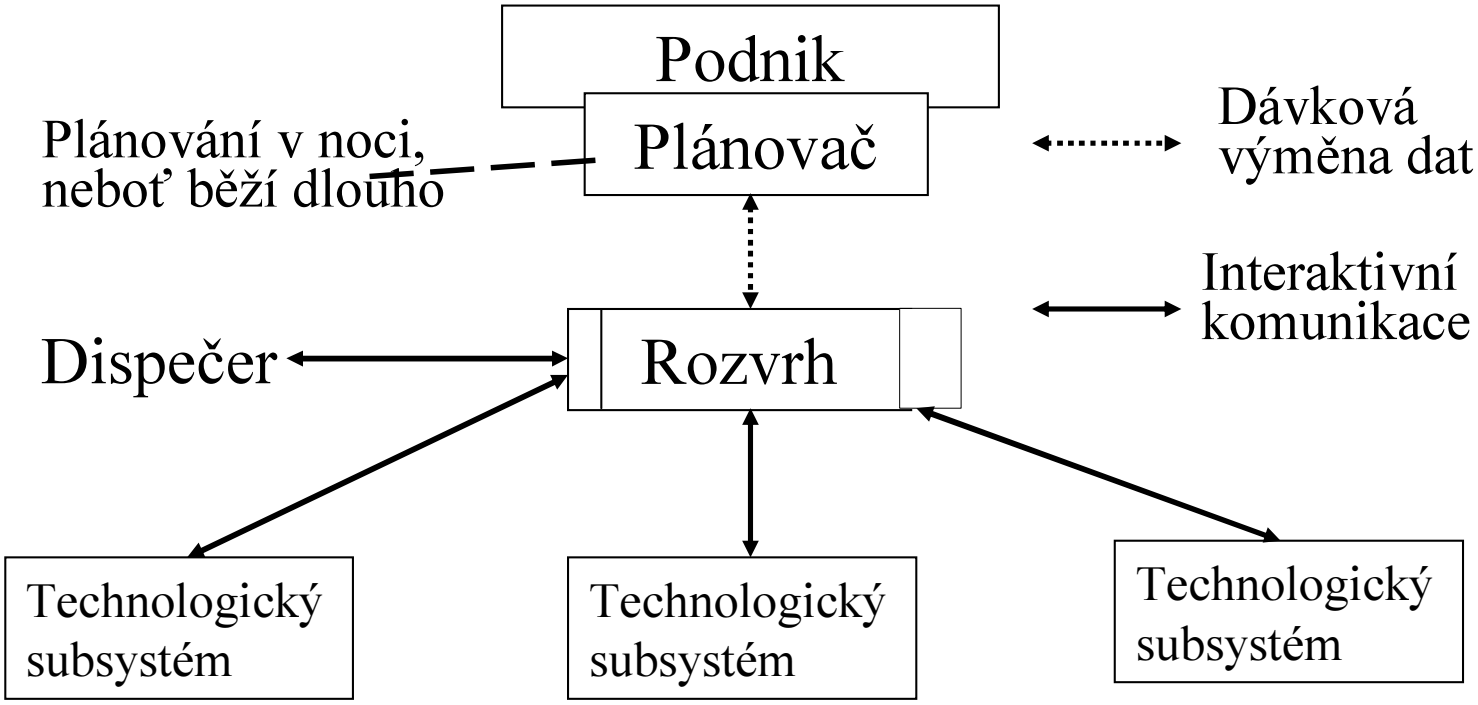
- Neví se, je-li vůbec možná model driven architecture, platí zčásti i pro aliance, chybějící příklady řešení a CASE nástroje
- Problémy s proprietálními řešeními takže nelze vždy použít standardy
  - lze zmírnit správnou strategií (XML)
  - může urychlit vznik uživatelských XML jazyků a tím nevýhodu změnit na výhodu

+- Nutná úzká spolupráce mezi skoro všemi vývojáři a mnoha uživateli i z nejvyšších postů (CEO se musí účastnit specifikací)

++ Velmi dobré zkušenosti s provozem (dvacet let bez údržby, viz též zkušenosti s Y2K)

Neinteraktivní komunikace

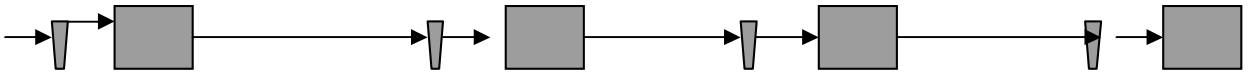
Spolupráce plánovacích algoritmů s provozem vyžaduje inteligenci při přenosu požadavků



Důvody

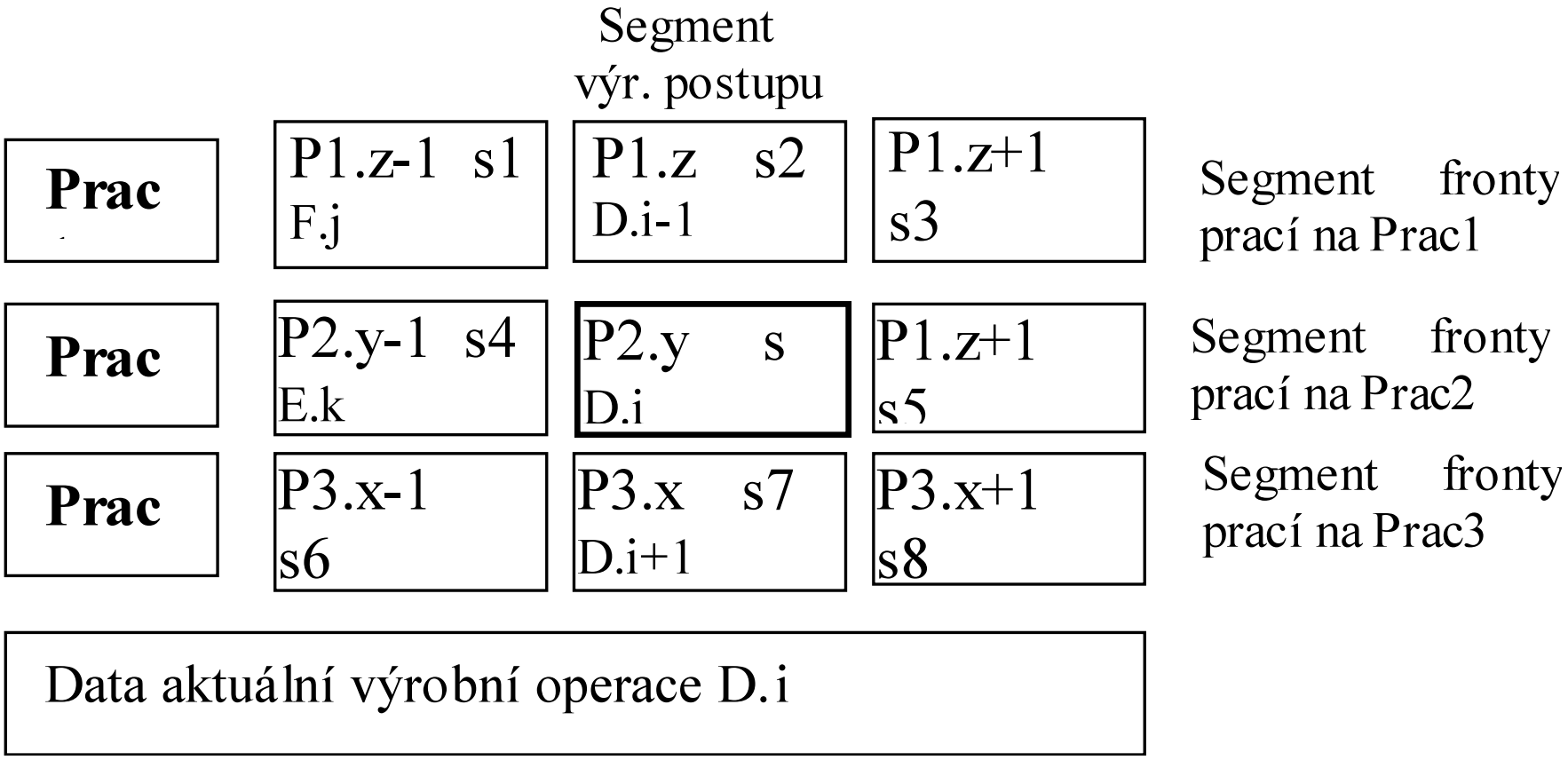
- Data jsou nevčasná, neboť plán algoritmy jsou pomalé, změny musí proto udělat rychle člověk aby výroba nestála když se objeví nečekaná překážka
- Data jsou nepřesná $\Rightarrow$plán je nedokonalý

Řízení na buffery



Mistr sleduje bufery a hledá pro práci, když se bufer nepříjemně zkracuje (řízení na průšvih)

Nutné pro výroby s krátkými seriemy a velkou variabilitou


Segment
výr. postupu
Prac
P1.z-1 s1
F.j
P1.z s2
D.i-1
P1.z+1
s3
Segment fronty prací na Prac1
Prac
P2.y-1 s4
E.k
P2.y s
D.i
P1.z+1
s5
Segment fronty prací na Prac2
Prac
P3.x-1
s6
P3.x s7
D.i+1
P3.x+1
s8
Segment fronty prací na Prac3
Data aktuální výrobní operace D.i

Pozorování

- Některé akce musí být dávkové (trvají příliš dlouho), to platí pro některé algoritmy, pro lidi a pro procesy reálného světa
- Jsou třeba zásahy dispečera
  - Nečekané/vzácné události - nevyplatí se je zahrnovat do rozvrhování (Vonásek je lempl, Pepa se včera ztřískal)
  - Kvalita dat
    - Nedostupnost, neznámá, nepřesná (mají velký rozptyl)
    - Zřídka potřebná (nevyplatí se sbírat)
  - Potřeba využít inteligenci lidí jako součásti procesů

Pozorování

- **Neběží jen o poskytování informací, předávají se i příkazy, služby mohou být služby reálného světa (raketové základny)**
- **Datové rozhraní je typické pro manažerskou úroveň řízení, ta se stává hlavním tématem současnosti**
- Snadné kombinování ručních a automatizovaných postupů
- Vysoká autonomie služeb
- Částečný návrat DFD (zavrhované funkční dekompozice)
- Podle současných znalostí je implementace datového rozhraní v aliancích obtížná, i když možná

Pozorování

- 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 semantic web*
- Důležité je deklarativní 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, stabilita řešení

Další výhody konfederací

- Typické pro služby v lidské společnosti 
  srozumitelné uživatelům
- Jiné paradigma než objektová orientace nebo vzdálené volání procedur (kombinace s těmito paradigmaty je možná, pokud se koncepce používají jako ortogonální nástroje)
- 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

Doporučení, uplatnit, pokud lze

- Použít SOA a XML, možná SOAP
- Raději komunikace přes datové úložiště a to raději neinteraktivní (bez trigerů)
- Používat i dávkové zpracování, šetří to náklady, vyšší stabilita
  - Jde použít častěji, než se má za to

Doporučení, uplatnit, pokud lze

- Uživatelsky orientované rozhraní, hranice služeb obdobná hranicím služeb v reálném světě
- Uživatelé by měli být schopni snadno měnit obchodní procesy během provádění
- Umožnit simulaci služeb uživateli
- Uživatelé by měli být schopni analyzovat žurnál komunikace pro kontrolu a pro případné soudní řízení
- Maximálně využívat znalosti obsluhy

OLAP a snad i datové sklady používat jako služby

Klíčová role rozhraní

- Stabilita konfederace závisí především na stabilitě (neměnnosti) rozhraní
- Jazyk rozhraní by měl být vysoké úrovně (deklarativní)
- Rozhraní by se mělo chovat podobně jako rozhraní služeb v reálu (být uživatelsky orientované)
  - Být srozumitelné uživatelům, mělo by být podobné tomu, nač jsou zvyklí; je to nutné pro návrh a kontrolu průběhu business procesů
  - Takové má šanci nebýt měněno, je vlastně ověřeno dlouhodobým používáním

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.

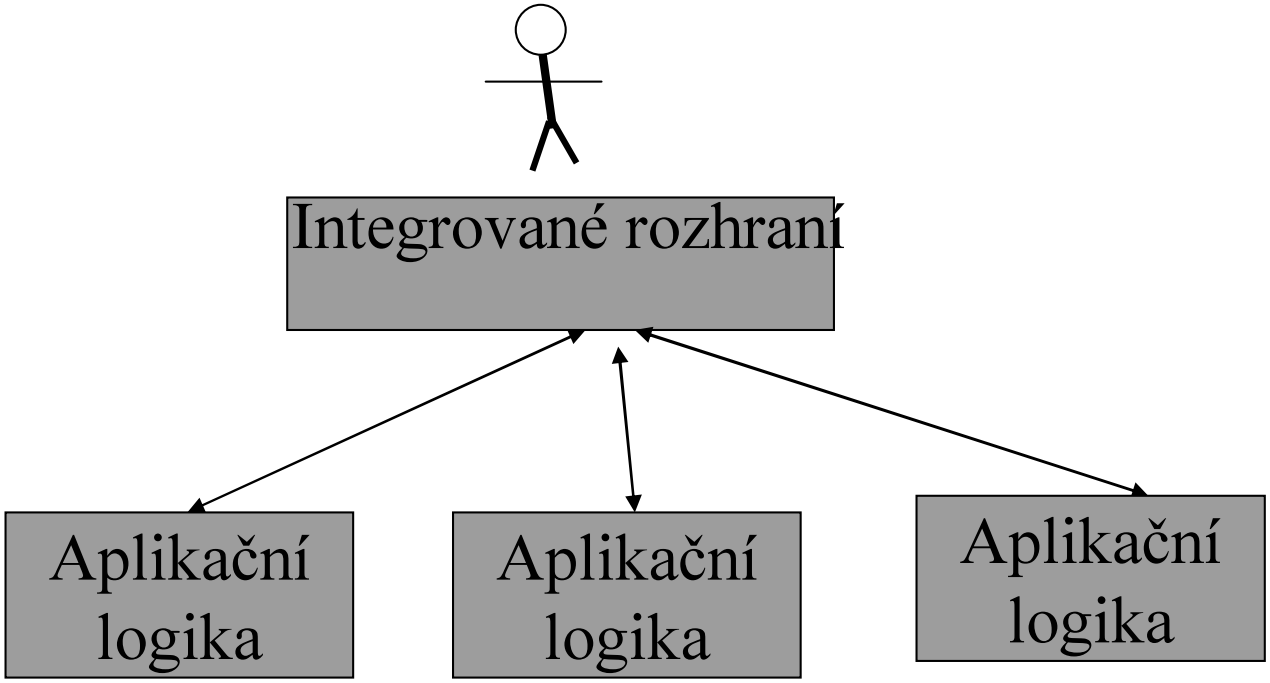
Je nereálné je definovat jednou provždy nebo centralizovaně => formáty dohadovat lokálně.

Jsou nutné dialekty!!!

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

Řešení: XML

Hlavní problém uživatelského rozhraní: Potřeba skládat dokumenty



Řešení bylo nalezeno také v jazyce XML a nástroji XSLT

Textový

Dovoluje rozšiřování

Jsou dostupné prostředky pro skládání a transformaci dokumentů (XSLT)

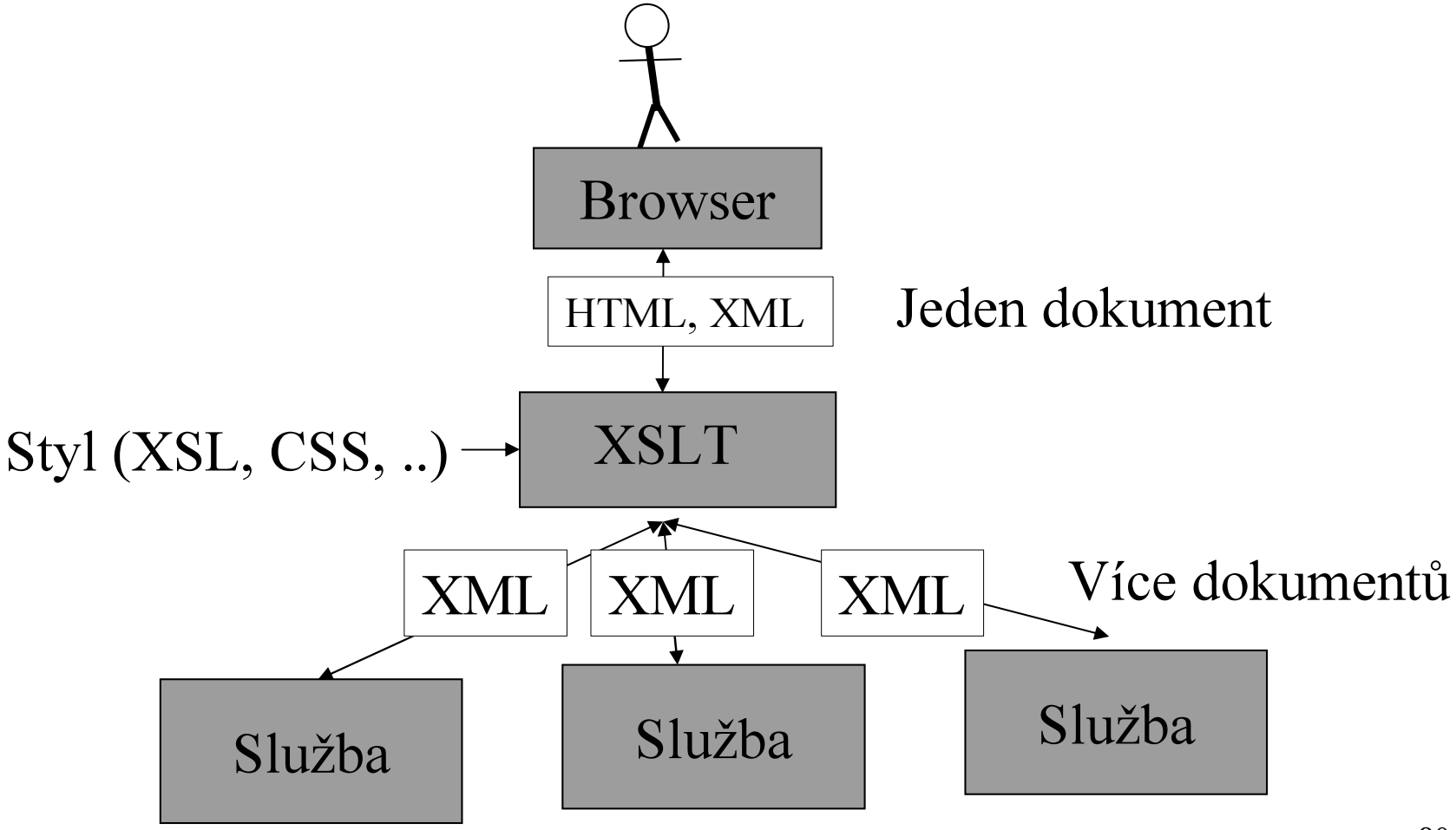
Trochu problémy s efektivností a spolehlivostí

Dá se použít i při programování uživatelského rozhraní

Funguje to !!!!

Nebezpečí Babylonské věže!!

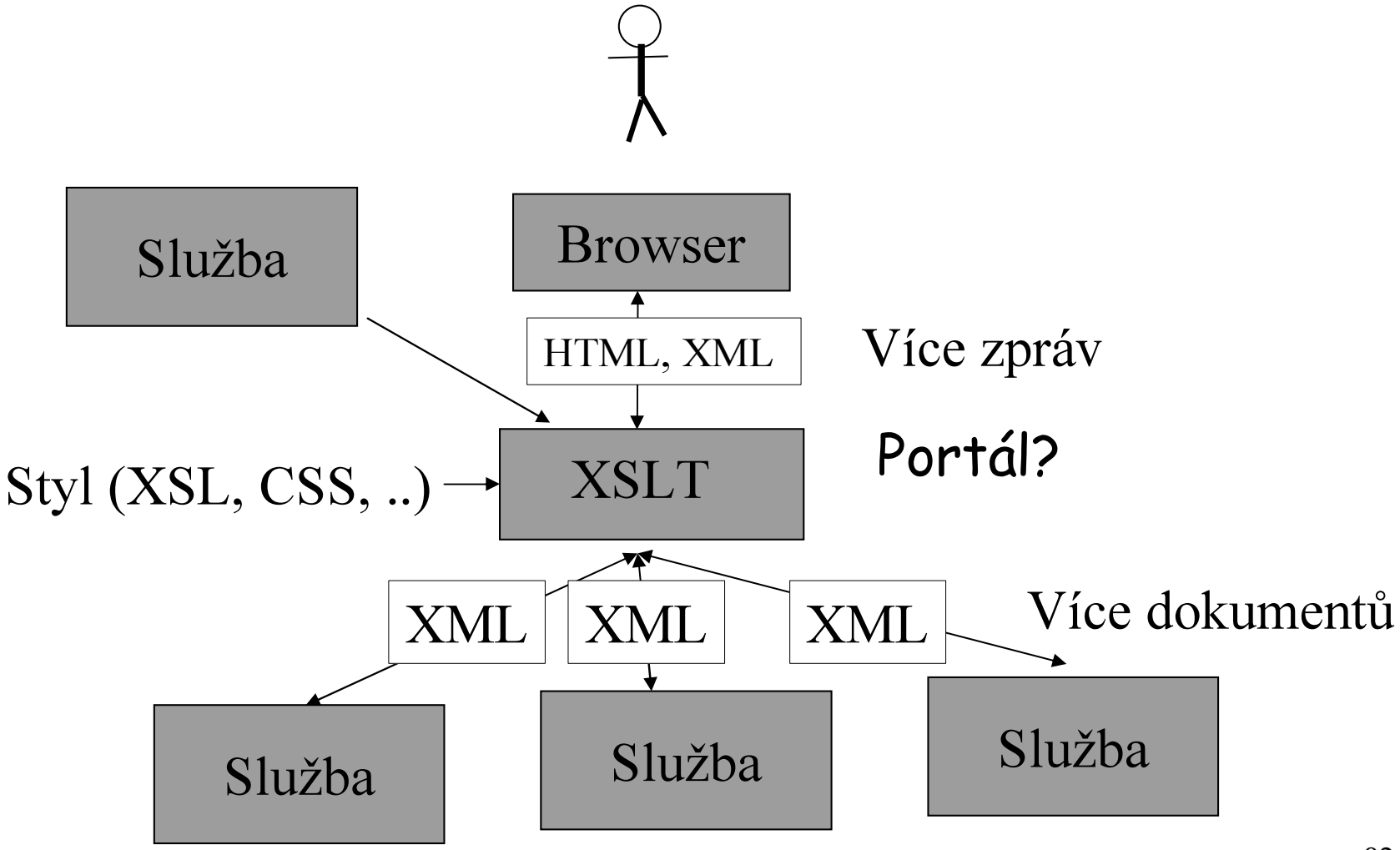
XML a uživatelské rozhraní



XSLT 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

XML a uživatelské rozhraní



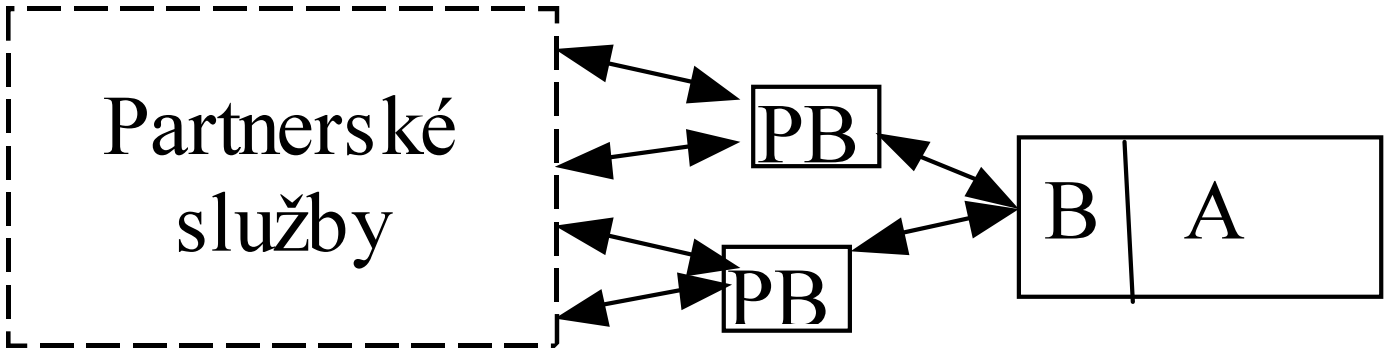
V čem je XML s podpůrnými nástroji důležitý (nutný)

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)

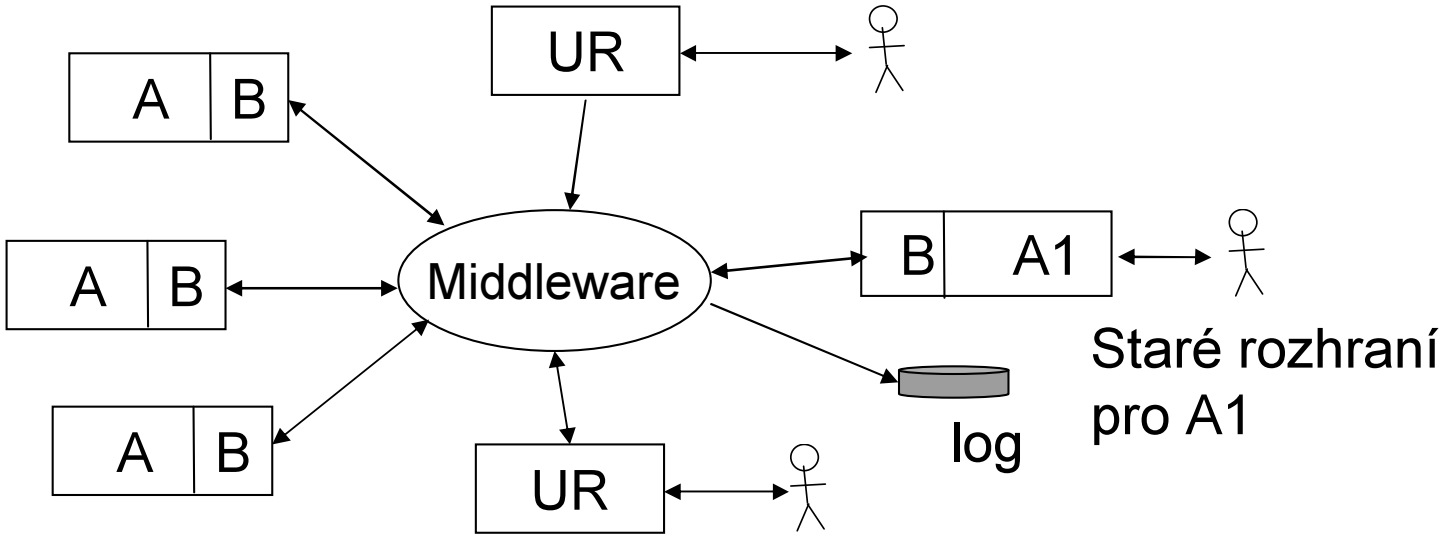
Techniky 1
Zlepšení stability rozhraní

- 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)
- Různí partneři požadují různé rozhraní -

Řešení: Infra služba jako přeřazená brána

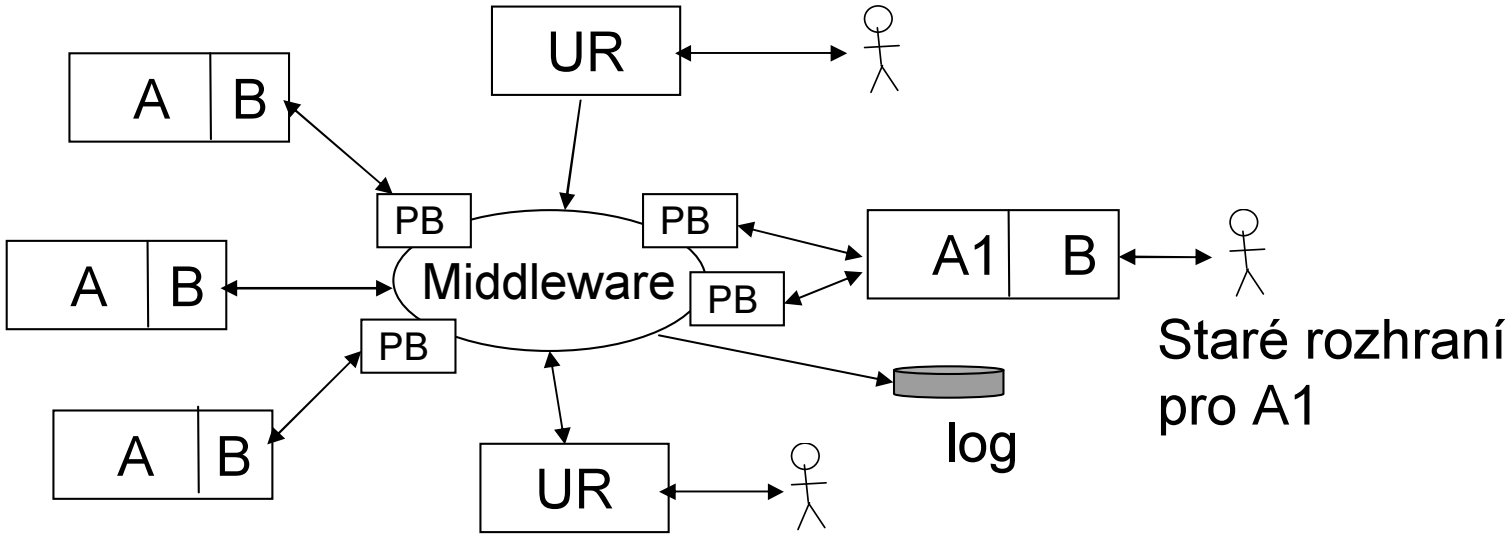


Konfederace a SOA



Nástroje na programování přeřazených bran jsou stejné jako u portálů

Konfederace s přeřazenými branami, konkrétní propojení



PB Předřazená brána, může jich být více, mohou i chybět

Jak implementovat brány

- Jako zcela plnoprávné uzly
- Integrovat do middlewaru
- Integrovat do akomponent
- Kombinovat přístupy

Konfederace s přeřazenými branami

- Podobná řešení pro web services
  - Mediator (www.wsmo.org), není zamýšlen jako obecná služba mimo rámec web services a objektové technologie
  - Sessioners
  - Fasády v design patterns
- Nástroje XSLT
  - Ale má zatím chyby, neefektivita

Předřazené brány (PB) jako místa v zobecněných Petriho sítích a barvenými značkami

Techniky vývoje PB jsou prakticky stejné jako u UR. Lze použít XSLT a transformovat *m* vstupních a *n* výstupních zpráv. PB (a UR) lze proto chápat jako zobecněná místa v barvených Petriho sítích

Jiný pohled na konfederaci

Cooperative commerce (c-commerce)

Assembled applications

Composite applications

Konfederace pro strojovou byrokracii (podniky) jsou úžeji vázány, lze požadovat větší úpravy komponent a použít pak standardy. Pro takové systémy nemusí být přeřazené brány službami a mohou se implementovat jako rozšíření služeb, tj. splynou s branami služeb spolu

s doplňky do Middlewaru

Pro takové případy je výhodné použít Prostředí Enterprise Service Bus od IBM, Bea Systems nebo Progress Software

SAP používá konkurenční koncept NetVeawer

N^2 formátů a service bus

Komunikace služeb může obecně vyžadovat N*(N-1)/2 formátů zpráv. Řešením může být jediný formát pro zprávy mezi všemi předřazenými branami. To je základní idea řešení Enterprise Service Bus od Sonic Software.

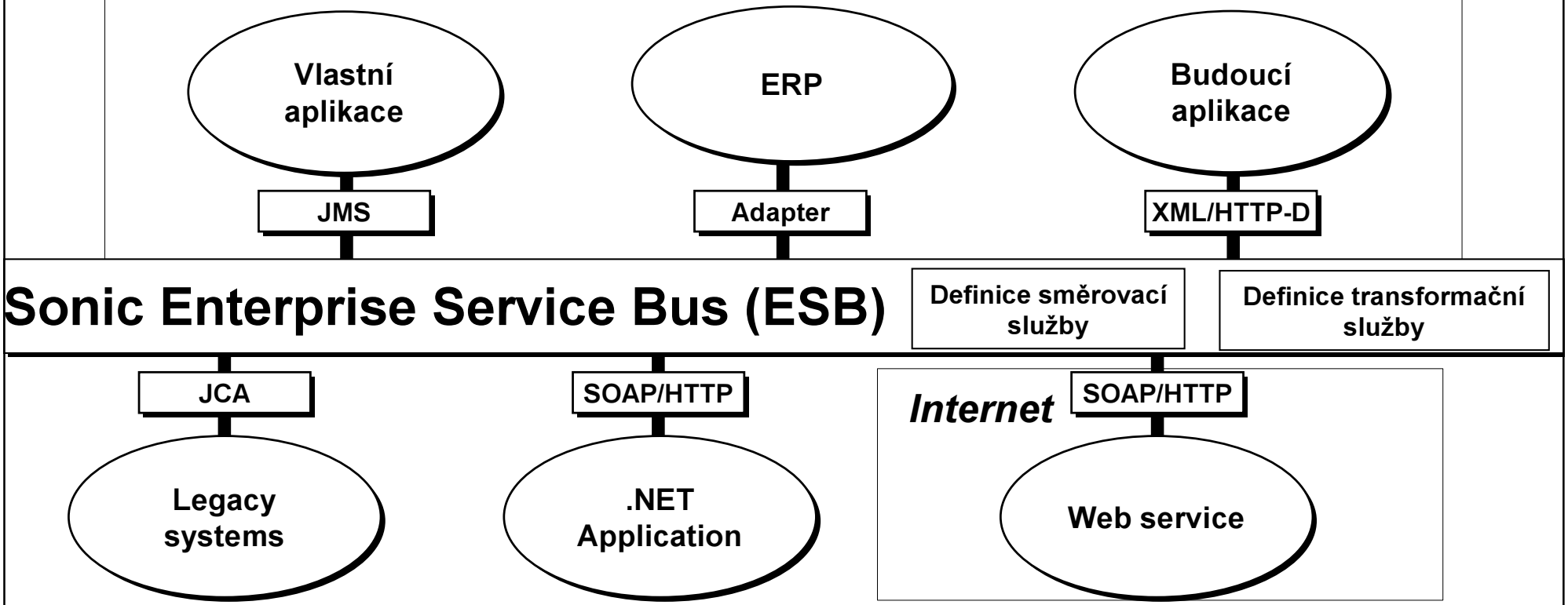
Na rozdíl od aliancí si partneři při spolupráci formáty zpráv nedomlouvají

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

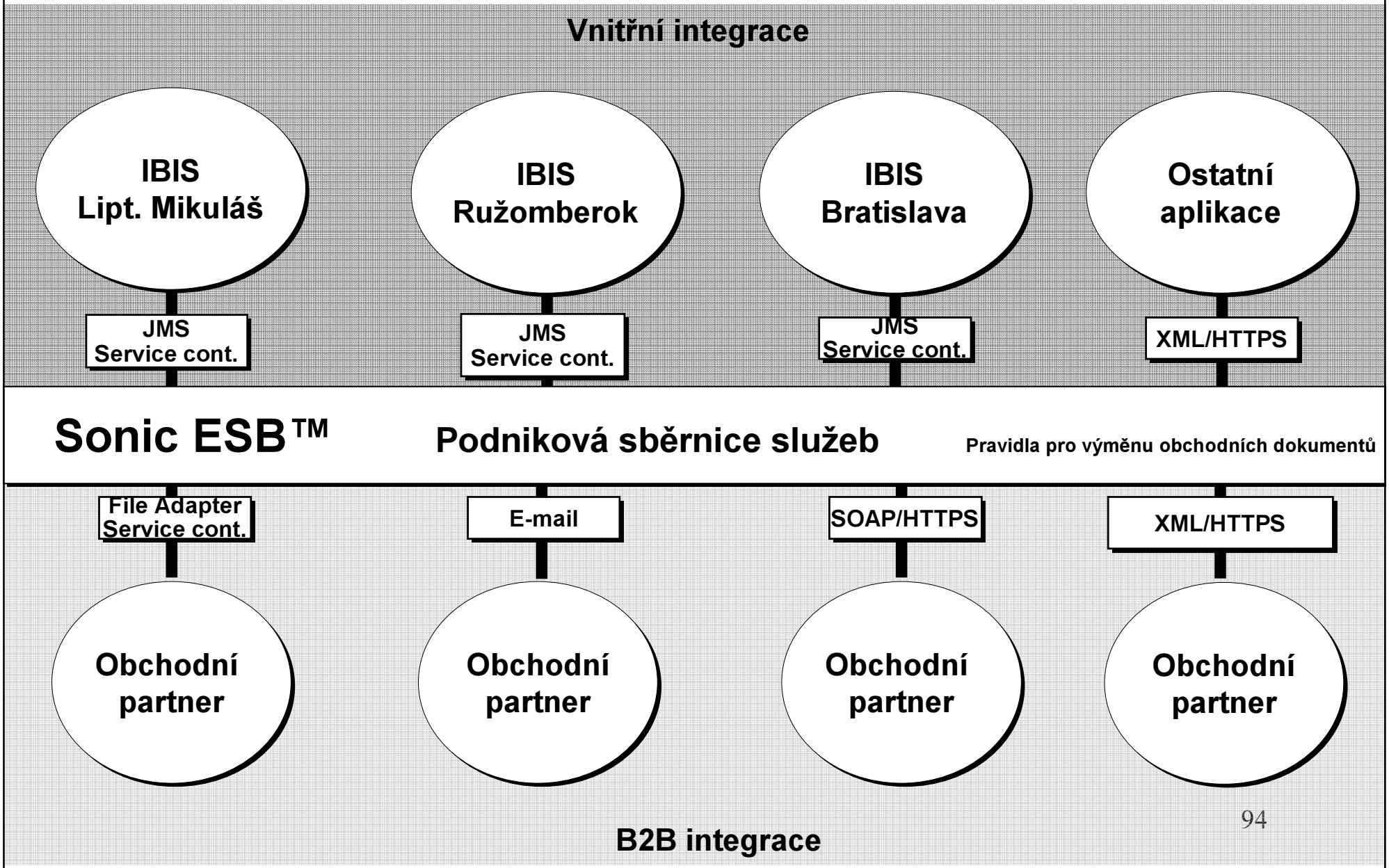
Stumpf Jindřich, Progress Software, Systémová integrace 2004

“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 to přineslo

- Skutečně se téměř nemuselo do existujících aplikací sahat
- 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)

Hrozby a příležitosti

- Snížení pracnosti desetkrát proti dřívějším zvykům → desetkrát menší příjem, desetkrát méně programátorů
- Je ale možné navrhnout nové služby a ty vyvinout → více programátorů
- *Není jasné, který trend převládne*

Co to ale stojí

V daném okamžiku není jasné, zda u ESB nehrozí podobné problémy, jako je tomu u aliancí, tj. převodníky mohou vyvíjet pouze programátoři a nikoliv koncoví uživatelé. Hrozí i problémy s porozuměním komunikaci a pro ruční zásahy.

Net Weaver od SAP

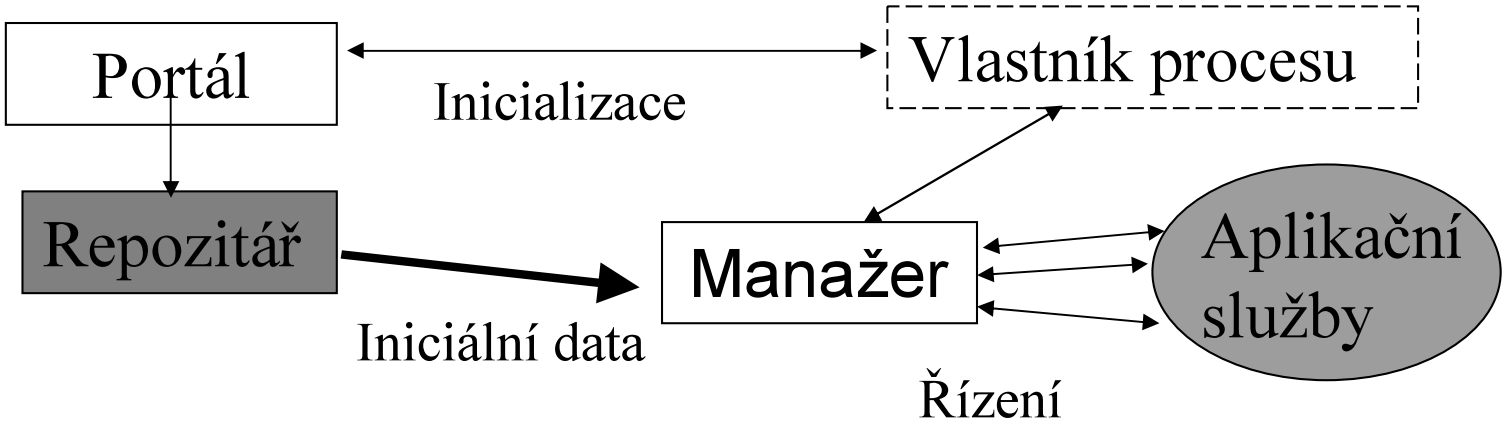
- Technologie konkurenční k ESB
- Je méně otevřená
- Není jasné, jak dopadne
- Nevíme zda nejsou komerční systémy pro ESB i NetViewer příliš omezující.

Business procesy

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

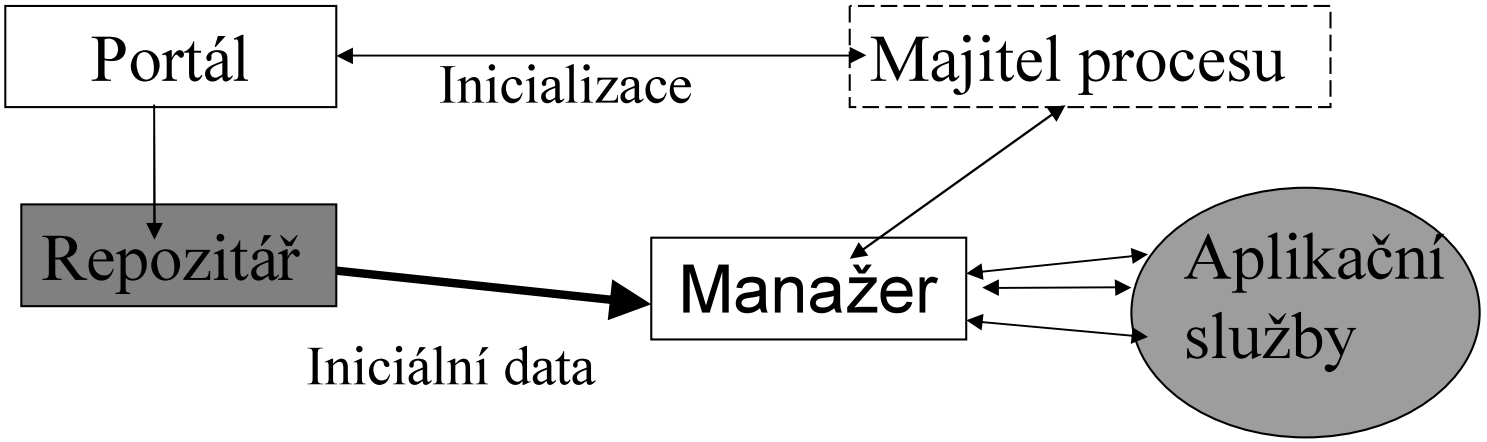
Business procesy

- Řešení:*Specializovaná služba **manažer** pro každý proces, simuluje prvky portálu, iniciální data z repozitáře*



Manažer se vytváří pro každý proces znovu, je to služba která se chová jako instance procesu

Business procesy



Řešení s využitím standardů popsaných v www.bpmi.org

- Modely procesů v BPML (dialekt XML)
- Data manažeru v BPEL (XML), upravují se podle parametrů inicializace zadané vlastníkem procesu

Manažer odpovídá instanci procesu podle BPMI. BPMI 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)
- Použití activity diagrams z UML
- Jednoduchý fulltextový popis se zaznamenáváním provedených akcí/kroků

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

- Obtížně se implementuje v centrální databázi (obecný problém centrálních DB v 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. 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

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
  - Předčasná standardizace bývá nedokonalá

Principy agilního vývoje 1

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.

Principy agilního vývoje 2

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Principy agilního vývoje 3

8. Continuous attention to technical excellence and good design enhances agility.
9. Simplicity--the art of maximizing the amount of work not done--is essential.
10. The best architectures, requirements, and designs emerge from self-organizing teams.
11. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile development prefers:

A. Individuals and interactions over processes and tools

B. Working software over comprehensive documentation

C. Customer collaboration over contract negotiation

D. Responding to change over following a plan

Uživatelsky orientované rozhraní (UOR) a agilní formy vývoje

- UOR se musí vyvíjet ve spolupráci s uživateli
- Dobré UOR silně omezuje potřeby dokumentace. Práce služby se totiž dá často odvodit z jeho rozhraní.
- Použití prototypů umožňuje rychlou odezvu uživatelů.
- Dekompozice do služeb usnadňuje inkrementální vývoj

Důsledky

- Obchodní procesy mohou „programovat“ uživatelé a mohou za ně ručit
- Kromě aplikačních služeb jsme diskutovali služby, které bychom mohli nazvat službami infrastrukturními
  - Předřazené brány
  - Portály
  - Manažery procesů
- Infrastrukturní služby se většinou pro danou aplikaci vyvíjejí od počátku. Mohou existovat další typy infra služeb.
- Existuje pokus o obdobu manažerů na www

Pozorování

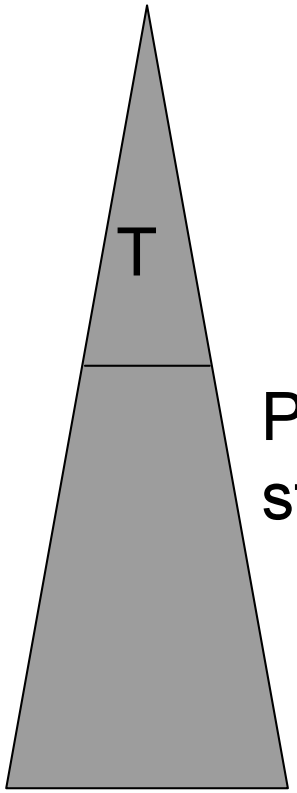
- Chování systému (business procesy) ovlivňuje do jisté míry sám koncový uživatel zadáváním parametrů instanciace procesu a úpravami jeho chování (to lze do jisté míry považovat za vývoj, který provádí koncový uživatel - end user development)
- To je projev nových trendů ve vývoji a vlastnostech softwarových systémů

Typy infrastrukturních služeb

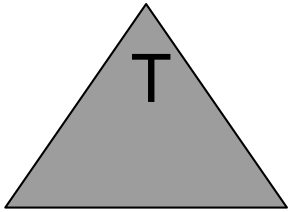
- Transformátory zpráv a směrovače (předřazené brány)
  - Dají se zobecnit na více vstupů a výstupů a chápat jako zobecněná místa ve smyslu Petriho sítí
- Konvertory dat
- Portály
  - Chápat je jako služby
- Řízení procesů
- Brány k virtuálním databázím

Použitelnost XSLT ?!

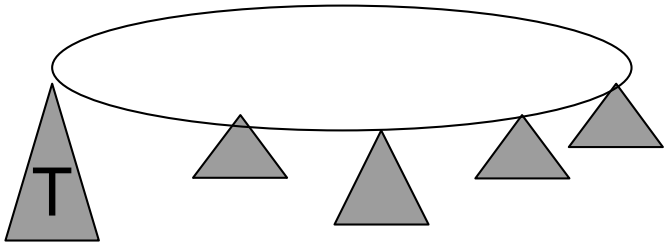
Dopad na organizační hierarchii



Původní
struktura



Čekalo se
(šéf uřídí více lidí)



Obvykle se stalo (decentralizace)

Vždy méně hlavně středního managementu i u něho změna profesní struktury (větší samostatnost šéfů divizí) při decentralizaci

Kde hlavní přínosy

- Horizontální spolupráce přes hranice oddělení (příklad Otavan)
- Dostupnost informací (PC od IBM)
- Sociální kontakty, zlepšení ovduší ve firmě
- Zefektivnění procesů
- Zpřesnění a vyšší dostupnost dat, zkvalitnění rozhodování
- Restrukturalizace procesů
- Zlepšení spolupráce s externími partnery

Možný model SOA

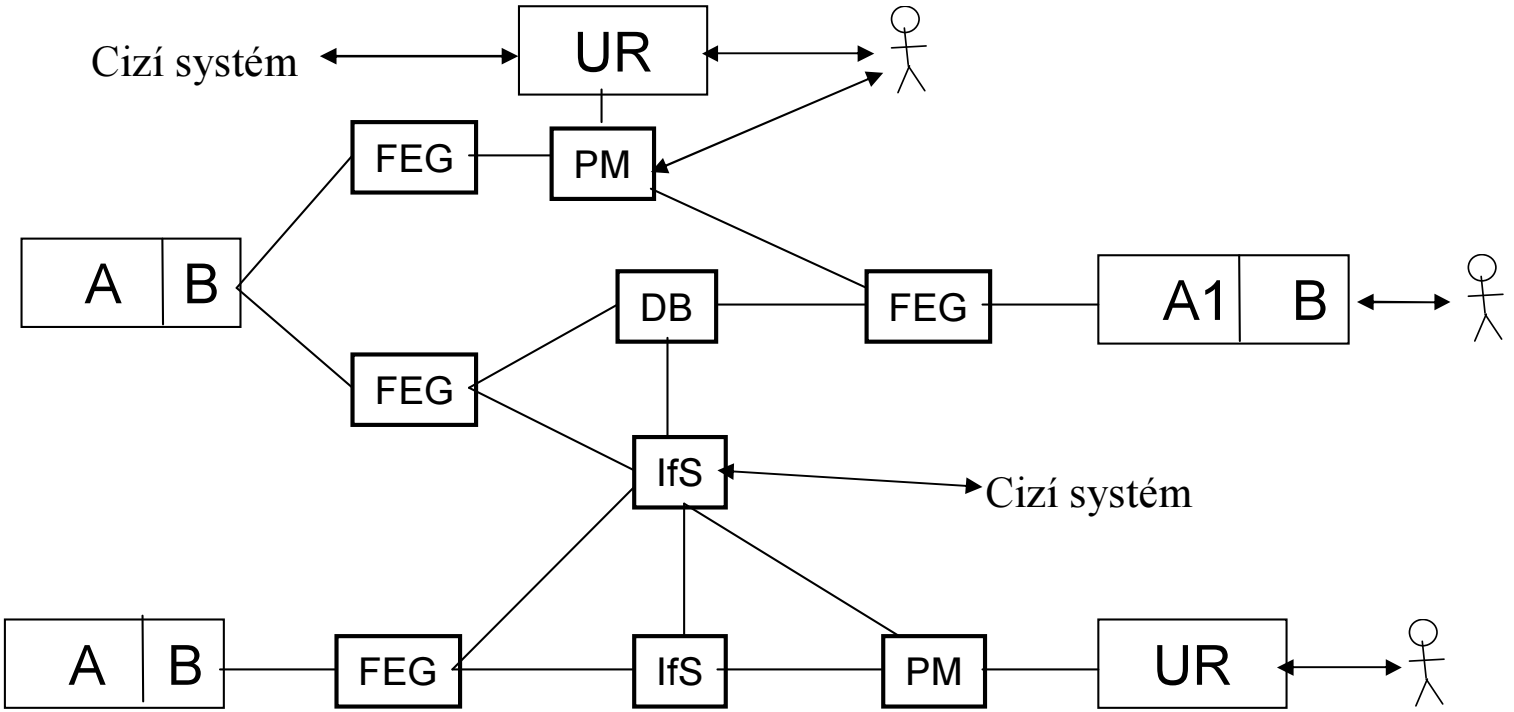
Infrastrukturní služby je možno chápat jako místa v zobecněné Petriho síti s barvenými značkami. Zobecněná místa mohou mít vlastní inteligenci.

Není jasné, zda je toto pozorování dostatečně nosné

Infrastrukturní služba

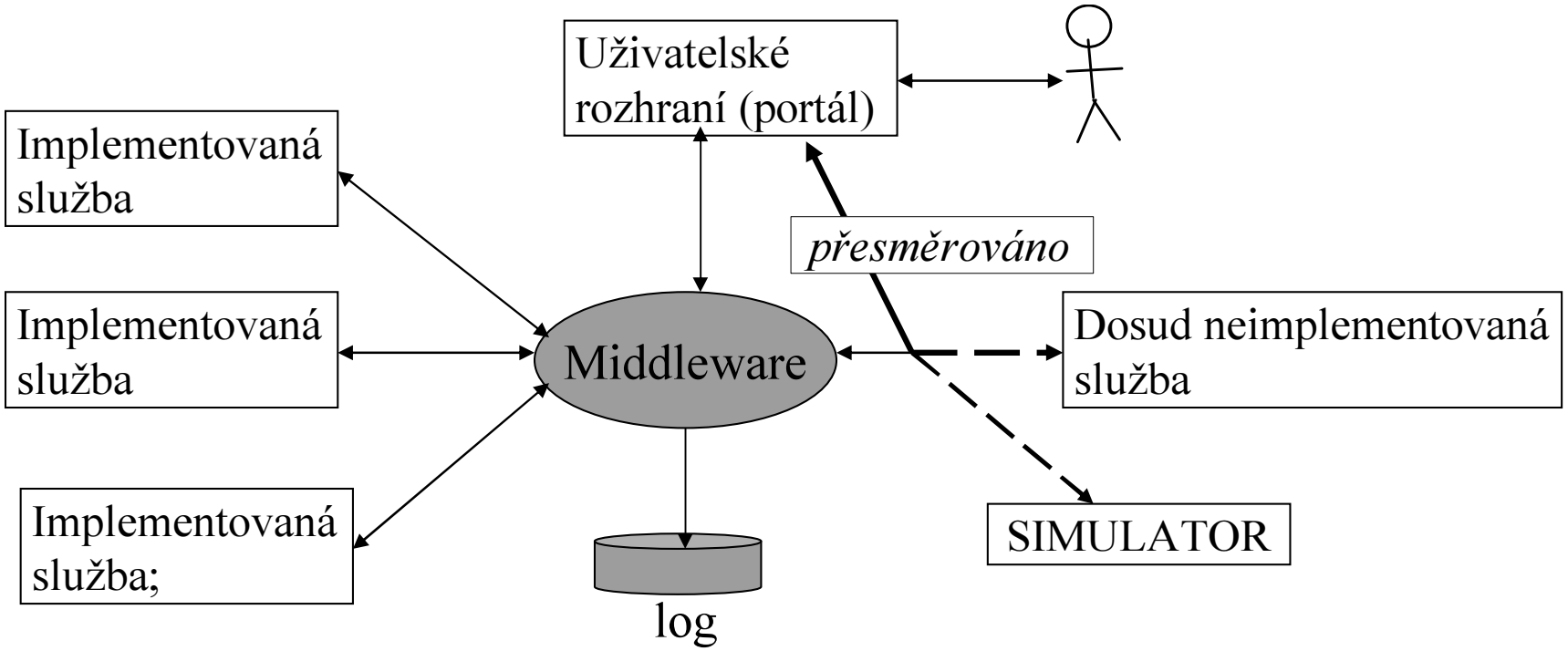
- FEG lze snadno modifikovat tak, by
  - Přijímala zprávy od různých zdrojů
  - *m*-tice zpráv transformovala na *n*-tice zpráv
  - Výsledné zprávy směrovala na dynamicky volitelné adresáty
- Výsledek nazveme infrastrukturní službou IfS

Logický pohled na konfederaci



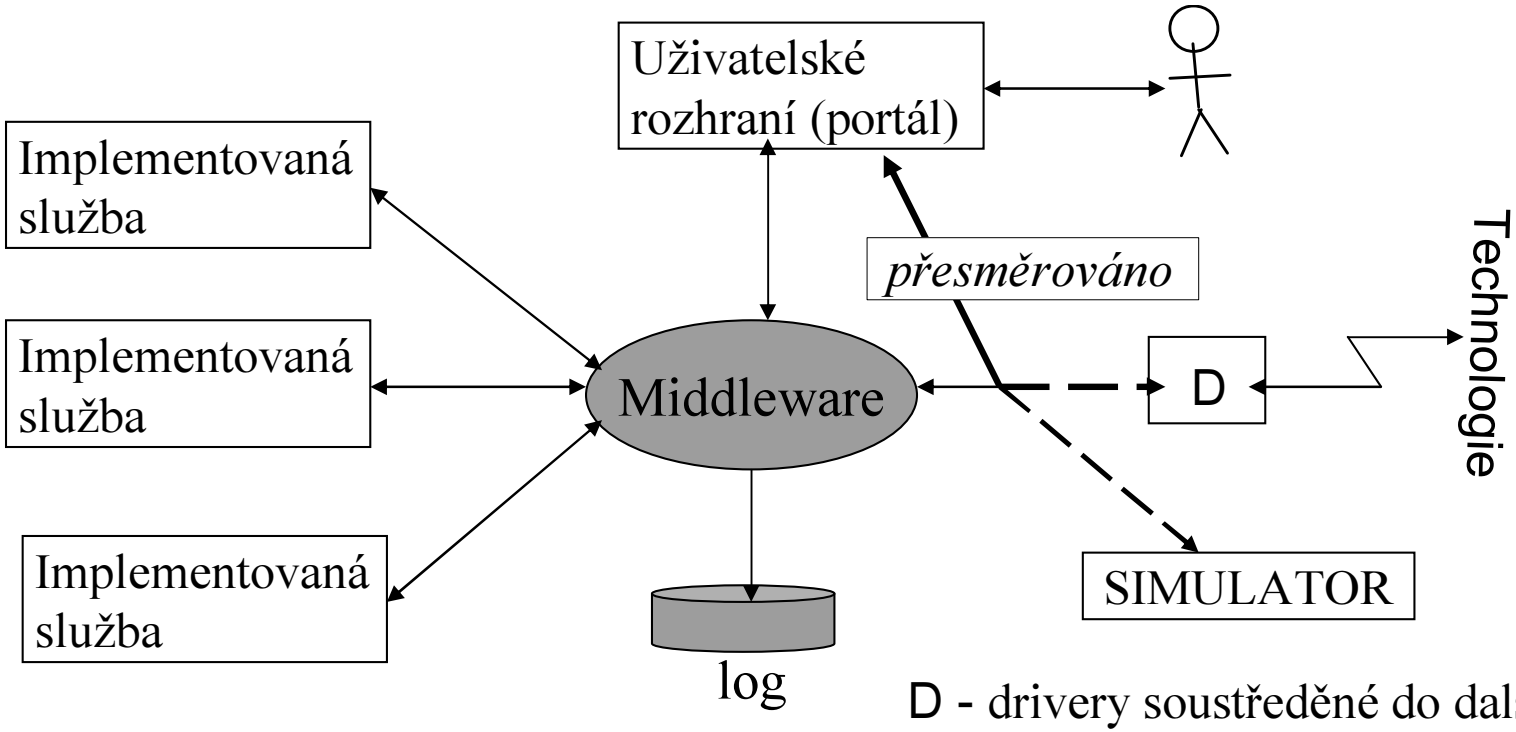
FEG – předřazená brána, UR – portál, PM - manažer procesu,
DB – datové úložiště IfS - infrastrukturní služba

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



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 3

Nouzové základní brány

Aplikace nemusí být vybavena branou, jak doplnit?

- Mám zdroje – upravím programy
- Odposlech na rozhraní UR * aplikační vrstva
- Odposlech terminálu

Techniky 4
Patterns and Antipatterns

Brown, W.J., et al: *Antipatterns. Refactoring Software, Architectures, and Projects in Crisis*, Wiley&Sons, 1998

Antipatterns jsou takové prohřešky, které při vývoji OO systémů vedou s velkou pravděpodobností k selhání projektu

- Některé jsou fatální vždy (chybné specifikace),
- jiné platí spíše pro monolitické systémy,
- jiné jsou v prostředí SOA předností.

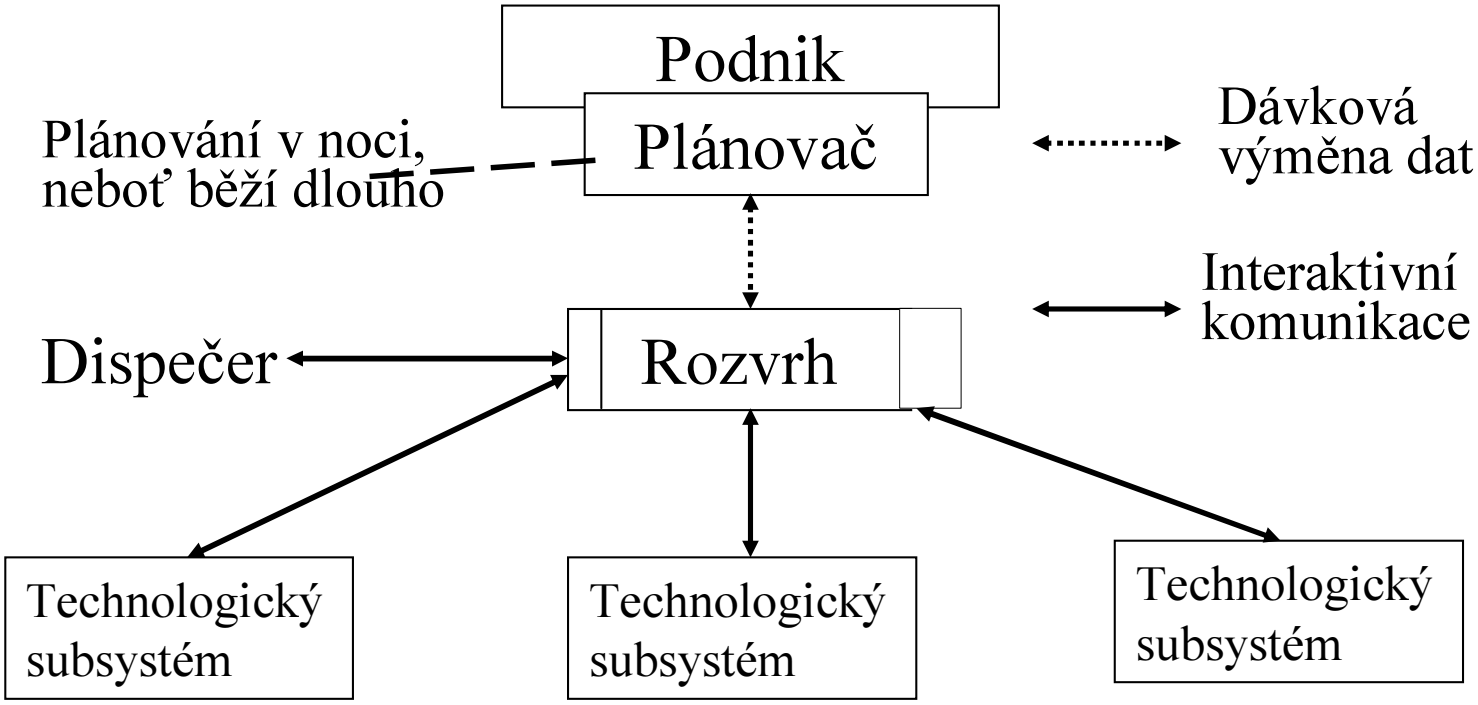
Antiapatterns v OO, které jsou patterns v SO

Brown, W.J., et al: *Antipatterns. Refactoring Software, Architectures, and Projects in Crisis*, Wiley&Sons, 1998

- Použití existujících aplikací (legacy systems),
- Ostrovy automatizace (viz PVS),
- Funkční dekompozice, používání datových úložišť (DFD)

Neinteraktivní komunikace

Spolupráce plánovacích algoritmů s provozem vyžaduje inteligenci při přenosu požadavků



Patterns a antipatterns

Brown, W.J., et al: *Antipatterns. Refactoring Software, Architectures, and Projects in Crisis*, Wiley&Sons, 1998

Antipatterns, které mají v SO jen omezený vliv, nejsou tedy v pravém smyslu antipatterns:

- Změna metodik vývoje během řešení
  - nedotýkají-li se změny infrastruktury systému nebo vývoje dané komponenty a jsou skryty za rozhraním,
- vystřihovánky (volné kombinování) na úrovni služeb.

Patterns a antipatterns

Brown, W.J., et al: *Antipatterns. Refactoring Software, Architectures, and Projects in Crisis*, Wiley&Sons, 1998

Snadná prevence následujících antipatterns

- Plánování bez konce,
- Obtížný člen týmu (dáme mu vyvinout autonomní službu)
- Dlouhé termíny než je systém funkční,
- Návrh výborem který fakticky za nic neručí,
- Znovu vynalézat kolo (snadno používáme existující aplikace),
- Vendor lock-in (závislost na jednom dodavateli)

Specifikace požadavků

- Volba SOA může ovlivnit i vize (co je reálně možné požadovat), takže rozhodnutí použít SOA musí být učiněno většinou velmi časně
- Specifikace požadavků je pro SOA odlišná od klasických metod, neexistuje vhodné CASE, MDA nelze použít, snad jen zatím
- Volba SOA musí být rozhodnutím i CEO i CIO uživatele, ti musí opustit své předsudky (vše naráz, vše od jednoho výrobce, najít cesty jako využít SOA)
- Mnohé se musí naučit i vývojáři

Objektovost a konfederace

- Deklarativní rozhraní v XML je výhodně koncipovat neprocedurálně
  - Není tedy voláním metod
  - Je spíše typu peer to peer
  - Stavební prvky jsou velké a uzavřené
- Používané technologie a praxe ukazuje, že i přístup k věci je v SO odlišný od objektové metodologie

Kdy je potřeba objektovost

- Od jisté úrovně hierarchie níže již nemohou mít komponenty konfederativní strukturu a nemohou být používány jako černé skříňky $\Rightarrow$ musí být spolehlivé $\Rightarrow$ měly by být OO
- Př. Ve velmi dobře navržené a implementované konfederaci z USA nebyly kurzovní operace staženy do jednoho objektu. Výsledek – nedalo se to použít

Úvahy a otevřené problémy

- Není inženýrsky zvládnuto
  - Chybí zkušenost (success/failure stories)
    - Chybí metriky
    - Chybí osvědčené SW procesy pro SOA
    - Problémy s profesní skladbou týmů (méně klasických programátorů)
    - Nevíme, co je obtížné a co optimální
  - Co vše má dělat middleware (kódování)
  - Jak na mobilní klienty a klonování služeb

Úvahy a otevřené problémy

- Nevíme jak optimálně využívat značkovací jazyky, především jejich metaúroveň
- Obchodní politika uživatelů a výrobců
- Souvislost s autonomními agenty, gridy a metapočítáním
- Optimalizace výkonu konfederace (mobilita, klonování)
- Kdy je konfederace výhodná a kdy ne

Úvahy a otevřené problémy

- Jak upravit studium počítačových expertů
  - Nutnost výuky ekonomie, induktivního uvažování a některých humanitních oborů. To je obecný problém inženýrského studia, pro SW zvláště urgentní
  - Nutnost otevřenosti a adaptibility
  - Syndrom o všem něco nic pořádně
  - Syndrom hackera. S lidmi nerad jednám

Co chybí

- Obecná shoda o typech SOSS
- Povědomí, jako optimálně kombinovat OO a SO,
- Příklady řešení (case studies, best practices)
- Dělba práce mezi middleware, centrální služby, infrastrukturní služby a aplikační služby
- Vhodné modelovací nástroje (model driven architecture) a CASE systémy

Co lze očekávat v budoucnosti

- Zlepšení HW podpory
  - Dostupnost a kapacita linek
  - Poslední míle (WiMax, WiFi, Mobily, telefony)
- Komerční nástroje
  - .NET?, Service Bus, Net Weaver, nové CASE
- Vyřešení věcných problémů
  - Hranice služeb, uživatelsky orientované standardy, komerční řešení, zvládnutí paradigmatu, best practices, Petriho sítě

Co lze čekat v budoucnosti

- Obchodně manažerské problémy
  - Změny obchodních praktik výrobců i odběratelů, změny SW procesů, změny organizace uživatelů
- Subjektivní omezení
  - Změna postojů, přijetí paradigmatu, specifikace požadavků ve spolupráci „všech“, změny ve výchově informatiků
- Vedle podpory obchodních procesů průnik SOA do světa zábavy což vynese gigadolary

Příbuzné problémy

- Systémy hromadné obsluhy, kdy klonovat služby?
- Systémy mající prvky p2p
  - Gridovské systémy
  - Softwaroví agenti
- Není jasné, zda shody nejsou spíše povrchní a zda se mohou SOA systémy poučit od agentů a gridů

Důsledky pro SW profese

- Nejasný vliv SO na potřebu psaní nových komponent od začátku
  - Lze snáze uplatnit v existujícím balíku a snáze doplnit funkce do svého produktu
  - Komponentu lze získat kdekoliv na světě
- Větší důraz na nepočítačové znalosti a otevřenost myšlení
  - Nutnost rozumět funkcím komponent, tedy jiným oborům (např. statistika)
  - Potřeba umět doporučit správné komponenty
- Umět vidět a analyzovat souvislosti (výhodné i pro změnu profese)
- Výhodné pro agilní formy vývoje i velkých systémů