STU FIIT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Príprava na zápočet z predmetu

Architektúra softvérových systémov

2013 / 2014

Okruhy pochádzajú z emailu pána Poláška.
Ku každej otázke vždy vyžaduje:

- ***obrázok***
- ***definicu***
- ***vyhody/nevyhody***.

Spomínal, že mu najviac záleží na tých obrázkoch (na ich počte)

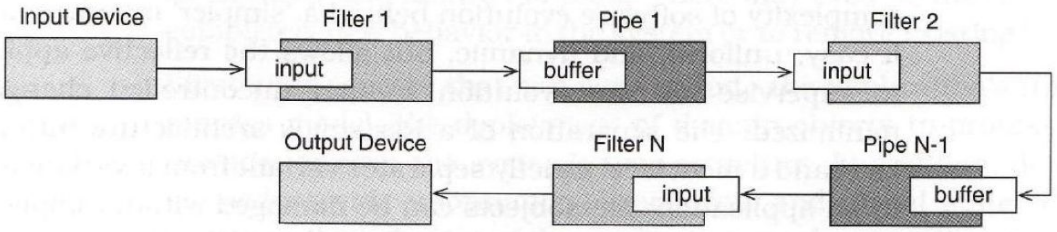
Ak nebudete s niečim súhlasiť, alebo budete mať pripomienky, prípadne otázky, prosím píšte komentáre (nie priamo v texte)

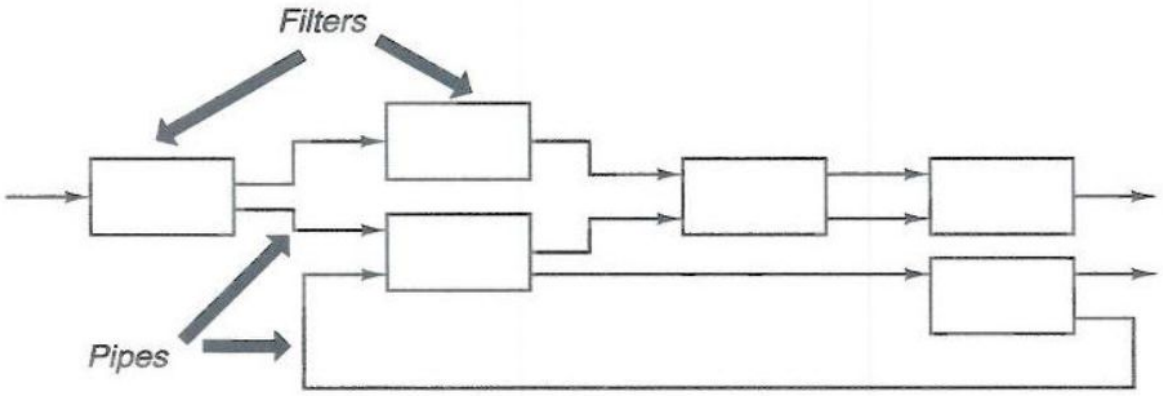
Stránka predmetu
http://www2.fiit.stuba.sk/~polasek/courses/ass-sk/index.html#prednasky

---

Obsah dokumentu

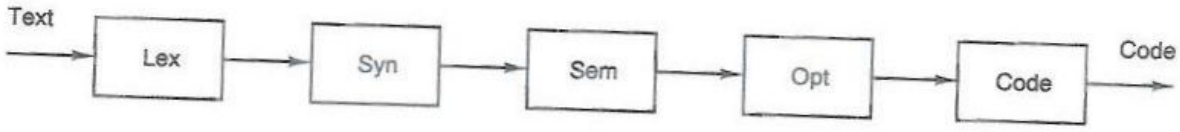
Pipes and Filters





Garlan and Shaw

Batch Sequential



Definícia:

- dáva štruktúru systémom, ktoré spracúvajú postupnosti údajov
- každý krok spracovania je skrytý v jednej filtrovacej súčiastke
- údaje sa pohybujú dátovodmi medzi susediacimi filtrami
- spracovanie údajov postupností
- nazýva sa aj model dátovodov a filtrov (pipes and filters - ako v UNIX shells)

Výhody:

- Podporuje transformačné znovupoužitie
- Intuitívny spôsob organizovania častí pre komunikáciu s podielnikmi
- Jednoduché pridávanie nových transformácií (filtrov)
- Relatívne jednoduché implementovať buď ako súbežný alebo sekvenčný systém

Nevýhody:

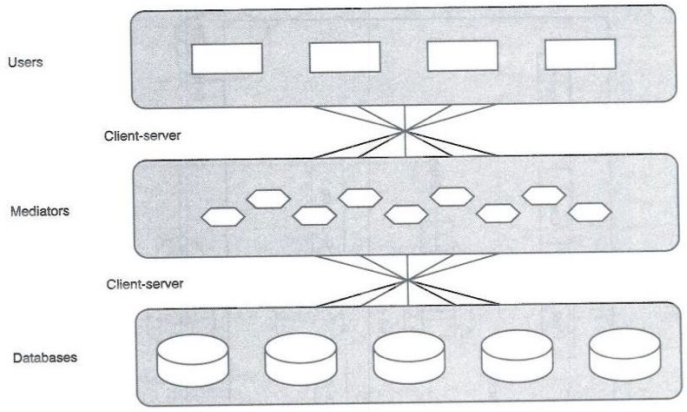
- Vyžaduje spoločný formát pre prúdenie dát v dátovode a je zložité podporovať interakciu založenú na udalostiach
- Nie veľmi vhodný model pre interaktívne systémy

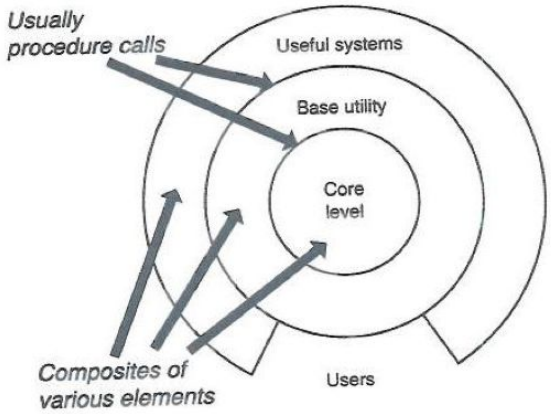
Zdroj:

- ASS02ArchStyly.pdf / slajd 24

//Ak by este niekomu nebol jasny rozdiel medzi pipes and filters a Batch Sequential:
http://se.inf.ethz.ch/old/teaching/2010-S/0050/slides/15_softarch_styles.pdf - 15-19

Layered Systems





Definícia:

- štruktúruje aplikácie, ktoré sa dajú rozložiť na skupiny pod-úloh
- každá skupina pod-úloh je na jednej úrovni abstrakcie
- využívajú vrstvy na oddelenie rôznych častí funkcionality
- každá vrstva komunikuje s vrstvami nad a pod ňou (žiadateľ a poskytovatel služby)
- komunikácia prebieha cez definované rozhrania
- použité napríklad pri sieťových protokoloch

Výhody:

- znovupoužiteľnosť

podporuje normalizáciu

- lokalizácia závislostí (zmena vo vrstve ovplyvní len susedné vrstvy)
- zameniteľnosť

Nevýhody:

- reťazenie zmien
- nižšia efektívnosť (réžia komunikácie medzi vrstvami)
- nadbytočná práca
- ťažkosti pri stanovovaní vrstiev

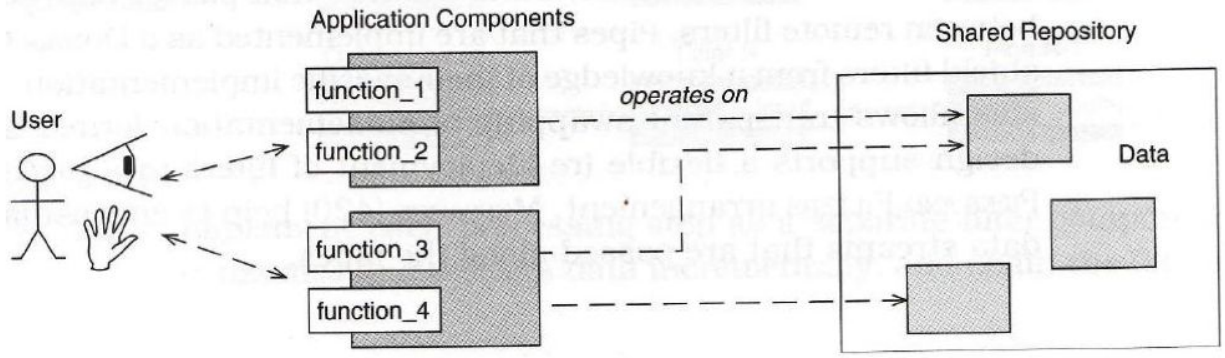
Zdroj:

- ASS02ArchStyly.pdf / slajd 17
- http://www.dossier-andreas.net/software_architecture/layers.html

Repository/Blackboard

---

fsRepository



Blackboardh

Zbierka relatívne nezávislých programov, ktoré spolupracujú nad spoločnou štruktúrou údajov. Každý program sa špecializuje na vyriešenie jednej čiastkovej podúlohy z celého problému. Programy sa navzájom nevolajú. Poradie ich vykonávania nie je predurčené.

Rozdiel medzi Repository a Blackboard

Pri repository výber programu riadi človek (User) , pri Blackboard program (Control).

Definícia Repository:

centrálnym prvkom je štruktúrovaný sklad údajov, s ktorým pracujú jednotliví (nezávislí) klienti:

- **komponenty:**
  - sklad údajov – je pasívny
  - klienti – majú navzájom nezávislé toky riadenia
- **konektory:**
  - dotazy a aktualizácie – ak sa odovzdáva riadenie skladu údajov, tak len za účelom vykonania príslušnej operácie (čítanie/zápis)
- **príklady:**
  - „klasický“ informačný systém
  - nástroje CASE založené na spoločnej databáze

Výhody:

- efektívny spôsob zdieľania veľkého množstva dát
- ak ide o perzistentné údaje, dá sa centrálne riešiť zálohovanie, bezpečnosť, prístupové

práva, zotavenie (repository manager)
- ľahká integrácia nových klientov, ktorí sú schopní pracovať s údajmi v danej štruktúre
- podpora zmien a udržovateľnosti
- odolnosť voči poruchám a robustnosť

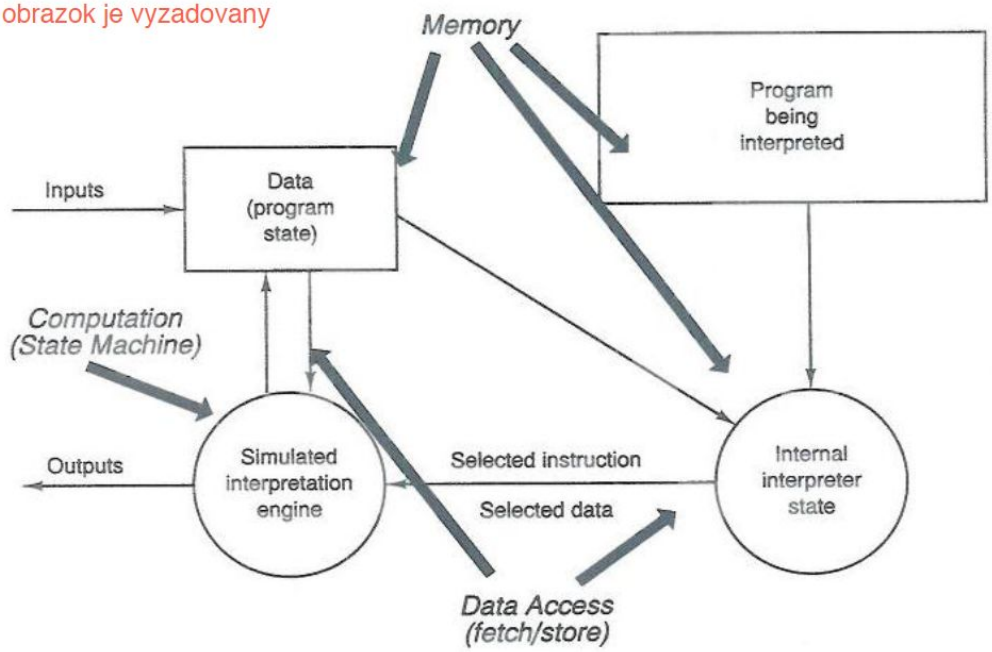
Nevýhody:
- spoločný dátový model môže byť príliš zložitý - treba spraviť kompromis na ktorom sa dohodnú klienti
- zmeny v modeli sú ťažko realizovateľné
- jednotliví klienti môžu mať rôzne požiadavky na zálohovanie, bezpečnosť, zotavenie ..
- nie je jednoduché distribuovať repository na viacero počítačov
- ťažké riešenie
- vysoké nároky na vývoj
- nízka efektívnosť

Zdroj:
- http://dai.fmph.uniba.sk/courses/tvorbaIS/sl/tis5.pdf
- ASS02ArchStyly.pdf / slide 50

Interpreters

---

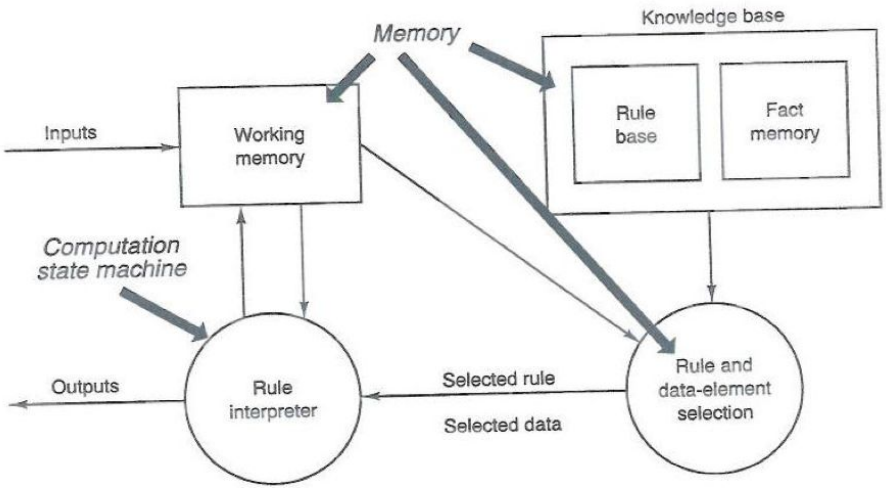
tento obrazok je vyzadovany



- Virtual machines that close the gap between the program and HW

2.30 - 14.

Rule Based



Definícia:

- štýl založený na virtuálnom stroji vytvorenom v softvéri
- špeciálny prípad vrstvovej architektúry, kde vrstva je implementovaná ako interpreter

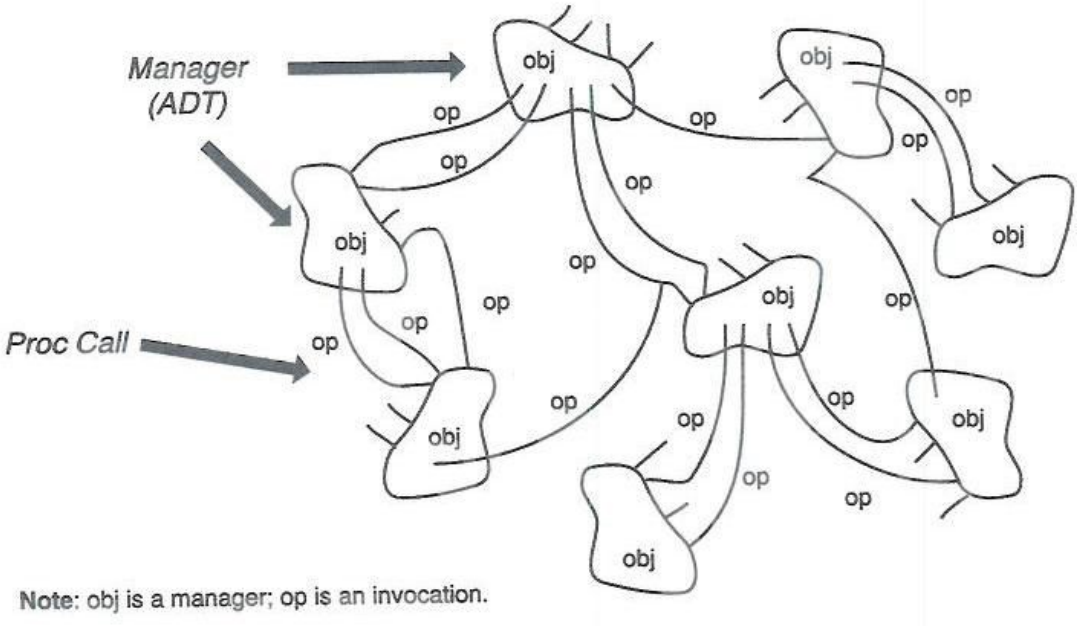
skutočného jazyka
- parsuje príkazy, ktoré následne vykonáva a mení pri tom svoj stav
- pod obrázkom z prednášky: Virtual machines that close the gap between the program and HW

Výhody:
- prekladové pravidlá sa dajú ľahko implementovať
- prekladové pravidlá sa dajú ľahko rozširovať

Nevýhody:
- pri zložitých prekladových pravidlách zaberá veľa pamäte a potrebuje veľa procesorového času

Zdroje:
- http://se.inf.ethz.ch/old/teaching/2010-S/0050/slides/15_softarch_styles.pdf
- Architectural_Styles.ppt
- interpreter + architecture + advantages

Data Abstraction: Object-Oriented Organization



Definícia:

- *Štruktúra systému do skupiny voľne viazaných objektov s dobre definovanými rozhraniami*
- *Objektovo-orientované dekomponovanie sa zaoberá identifikovaním objektových tried, ich atribútov a operácií.*
- *Po implementovaní tried sa vytvárajú z týchto tried objekty a existuje nejaký model riadenia na koordinovanie objektových operácií.*

Výhody:

- Objekty sú voľne viazané, čiže implementácia jedného sa dá zmeniť bez toho, aby boli ovplyvnené iné objekty
- Objekty môžu odrážať entity z reálneho života
- OO implementačné jazyky sa bežne používajú

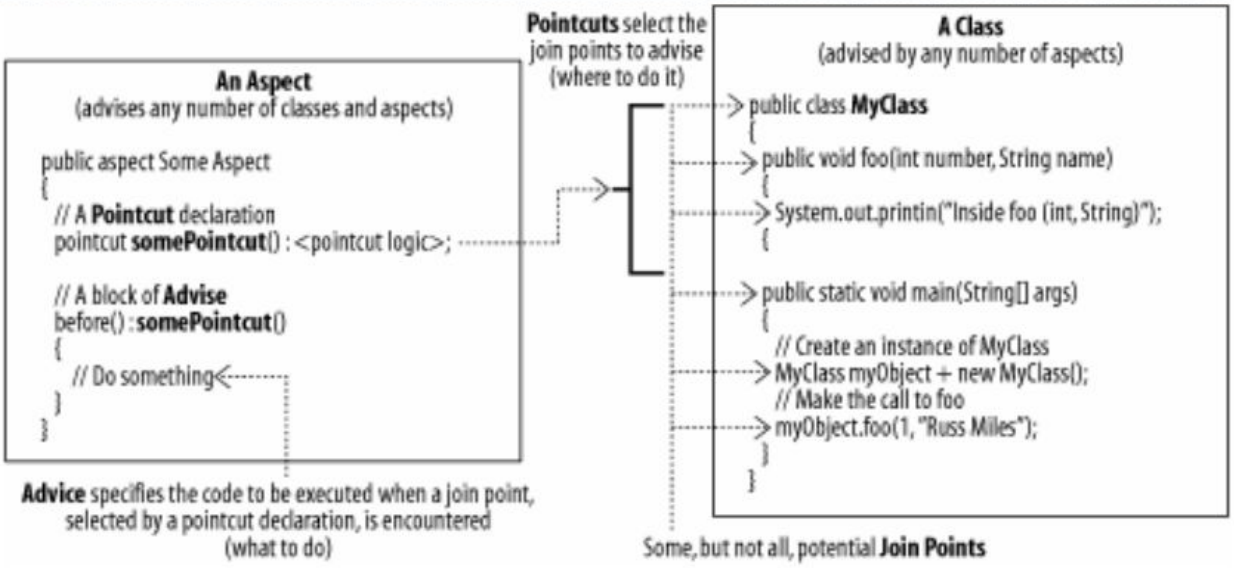
Nevýhody:

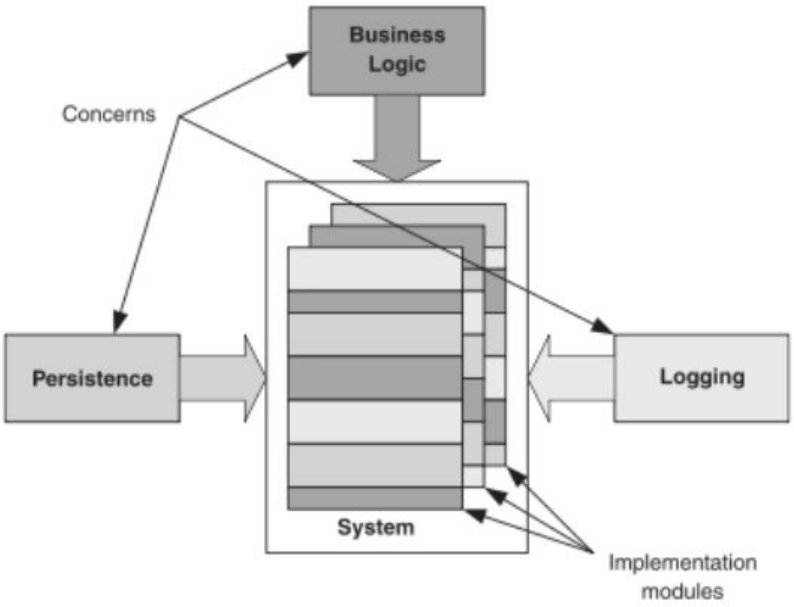
- Avšak, zmeny rozhrania objektu môžu spôsobiť problémy
- zložité entity sa môžu dať ťažko reprezentovať objektami

Zdroj: ASS03ArchStyly2.pdf / str. 42

Data Abstraction: Aspect-Oriented Org.

Aspects & Classes





Definícia:
Poukazuje na veľký softvérový problém - oddelenie záležitostí (separation of concerns)

- Záležitosti sú implementované ako aspekty
- aspekty modifikujú vykonávanie v bodoch spájania (join points)
- body spájania sú určené bodovymi prierezmi (point cuts)
- modifikácie sú vyjadrené pomocou videní (advices)
- Videnie sa vykonáva pred (before), po (after), alebo namiesto (around) bodu spájania

Výhody:

- ľahké odčlenenie hlboko prerezaných(crosscuted) záležitostí (napr. logy, profiling a iné monitorovacie podsystémy)
- funkcie zostávajú zamerané na konkrétnu funkcionalitu a prerezané záležitosti sú oddelené
- rôzne prístupy pre vtkanie (preprocessing, post-processing, AOP-prekladač, v čase zavádzania,,zabehu)

Nevýhody:

- ťažká identifikácia pointcutov v rozsiahlých systémoch
- problém s odhalovaním chýb vo vtkanom kóde(debugging)
- preplietanie sa môže stať nepredvídateľné
- problematické nasadenie na existujúce systémy, ktoré nebrali ohľad na prípadne vplietanie aspektov

Zdroj:

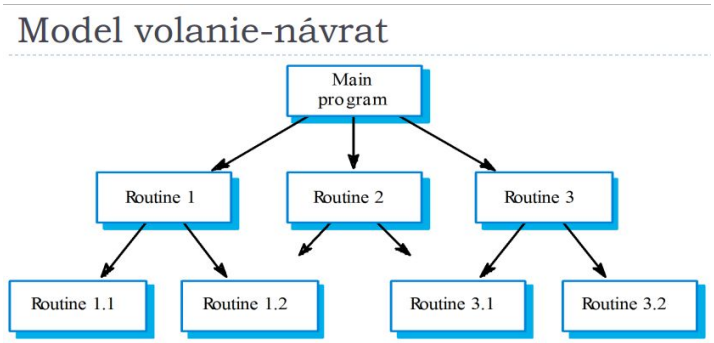
- http://www2.fiit.stuba.sk/~polasek/courses/ass-sk/files/ASS03ArchStyly2.pdf
- http://cs.wikipedia.org/wiki/Aspektov%C4%9B_orientovan%C3%A9_programov%C3%A1n%C3%AD

Process Control (Centralizované riadenie a jeho druhy)

Definícia:
Jeden podsystém má všeobecnú zodpovednosť za riadenie a štartuje a zastavuje ostatné podsystémy.

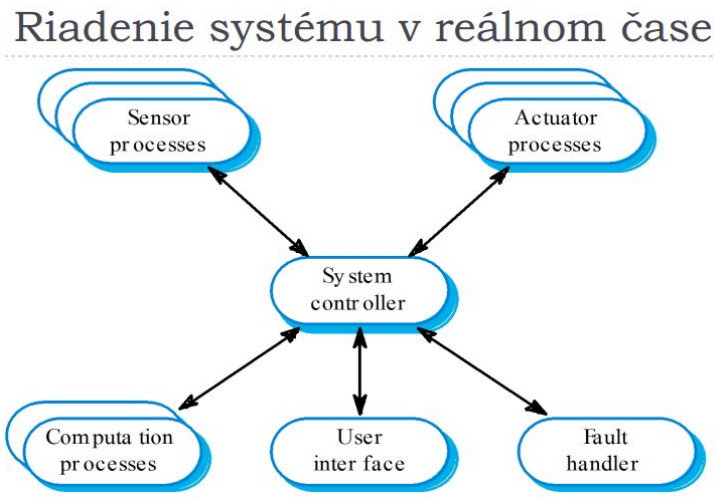
Model volanie-návrat

model pod-programov "zhora-nadol", kde riadenie začína na vrchu hierarchie pod-programov a pohybuje sa volaním akoby smerom dole, návratmi sa neskôr vracia späť nahor. Dá sa použiť na sekvenčné systémy.



Model s manažérom (riadenie systému v reálnom čase)

Dá sa použiť na súbežné systémy. Jeden systémový komponent ovláda zastavovanie, štartovanie a koordináciu ostatných systémových procesov. V sekvenčných systémoch sa dá implementovať pomocou príkazu ***case****.*



Výhody:
Nevýhody:
Zdroj:

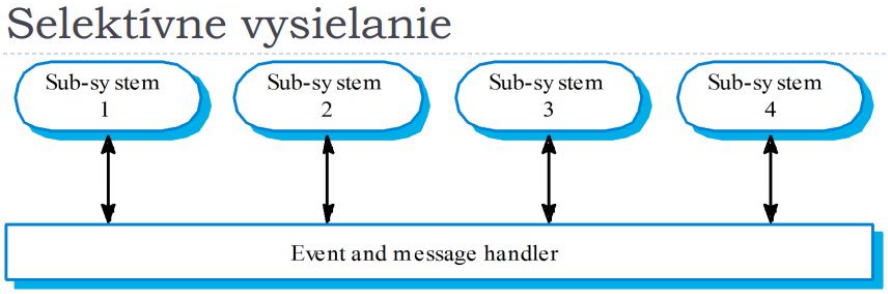
- ASS04StylyRiadeniaECMA.pdf

Event-Based (riadenie, založené na udalostiach a jeho druhy)

Definícia:
Systém sa ovláda externe generovanými udalosťami, kde dobu vzniku udalosti nemajú podsystémy, ktoré spracúvajú udalosť, pod kontrolou. Existujú dva hlavné modely.

Modely s vysielaním

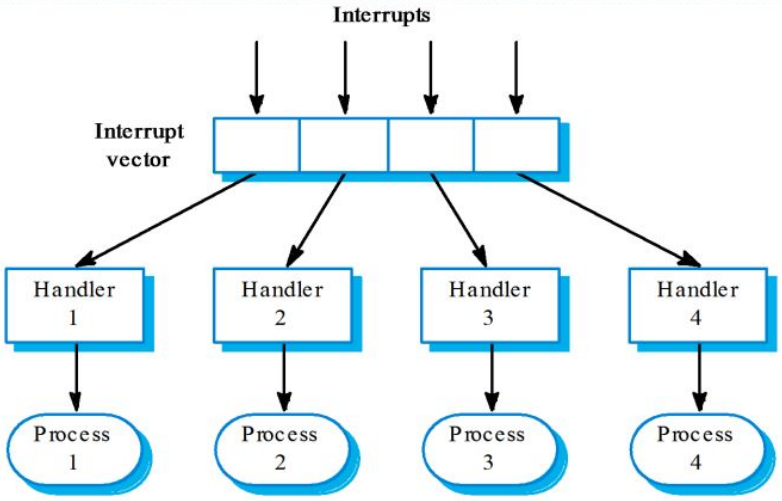
- Udalosť (informácia o nej) sa vysiela všetkým podsystémom. Ktorýkoľvek podsystém, ktorý dokáže vybaviť/ošetriť udalosť, tak môže učiniť.
- účinný, ak treba integrovať podsystémy na rozličných počítačoch v sieti
- podsystémy sa zaregistrujú k udalostiam (ohlásia záujem spracovať udalosť). Keď udalosť nastane, riadenie sa prenesie na podsystém, ktorý ju dokáže spracovať.
- politika riadenia nie je súčasťou správcu udalostí a správ. Podsystémy sa rozhodujú, ktoré udalosti budú chcieť spracúvať (o ktoré majú záujem).
- Avšak podsystémy nevedia, či a kedy budú príslušnú udalosť spracúvať.



Modely s ovládaním prerušeniami

- Používa sa v systémoch reálneho času (kde je nutná rýchla odpoveď na udalosť), kde sa prerušenia rozpoznajú správcom prerušení (interrupt handler) a odovzdajú sa určenému komponentu na spracovanie.
- existujú známe typy prerušení a pre každý typ je určený ovládač prerušení
- s každým typom je združené miesto v pamäti, v ktorom sa nastavením definovanej hodnoty (interrupt bit) indikuje, že nastalo prerušenie a hardvérový prepínač spôsobí prenos na príslušný ovládač.
- Model vedie k systémom s rýchlou odozvou. Ťažko sa programuje a validuje.

Riadenie ovládaním prerušeniami

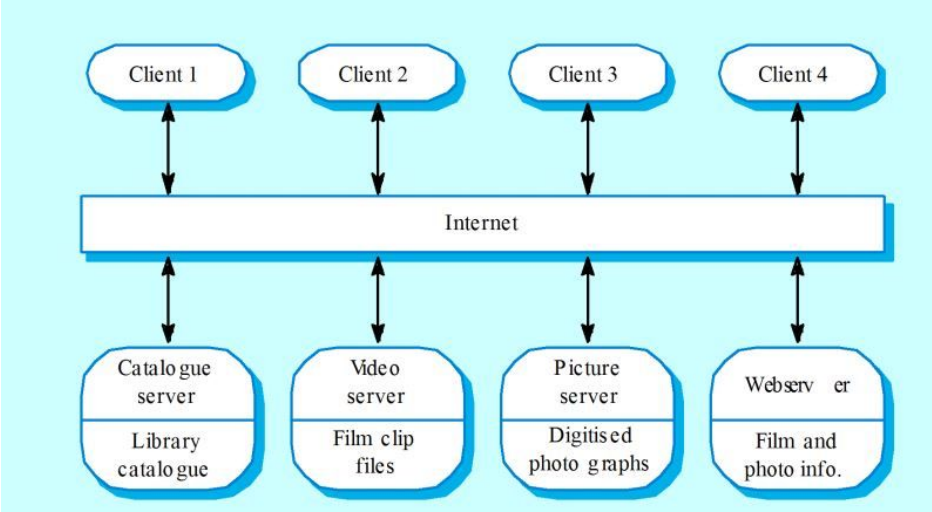


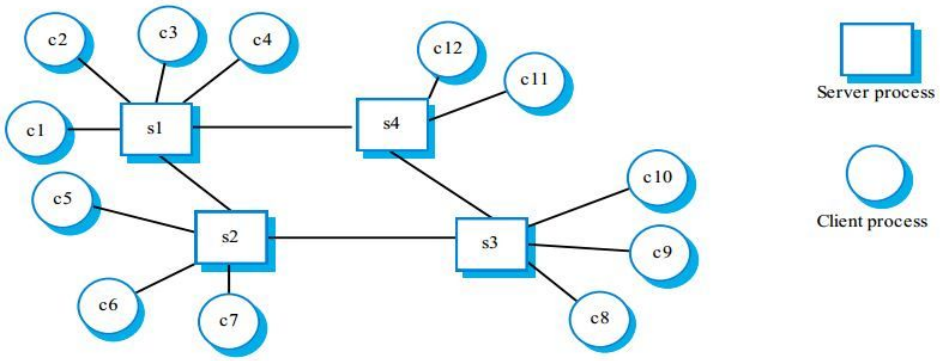
Používa sa keď je potrebná rýchla odozva. Pre rôzne typy prerušení rôzne ovládače. Ťažko sa programuje a validuje, je to veľmi rýchle.

Zdroj:

- ASS04StylyRiadeniaECMA.pdf

Client/Server





Definícia:

prednáška ASS02ArchStyly:
Model distribuovaného systému ktorý opisuje ako dáta a procesy sú distribuované medzi komponentami. Skupina samostatne stojacich serverov ktoré ponúkajú služby, skupina klientov ktoré služby využívajú a sieť ktorá umožňuje klientom pristupovať na servery.

ASS05distrib:
Aplikácia sa modeluje ako skupina služieb, ktoré sú poskytované servermi a skupinou klientov, ktorí používajú tieto služby
Klienti poznajú servery ale servery nepotrebujú vedieť o klientoch
Klienti a servery sú logické procesy
Mapovanie procesorov na procesy nie je nutne 1:1

Výhody:

- priamočiara distrubúcia dát
- robí vyžívanie sieťových systémov efektívnym.
- Môže vyžadovať lacnejší HW
- jednoduché pridať nové servery alebo upgrade

Nevýhody:

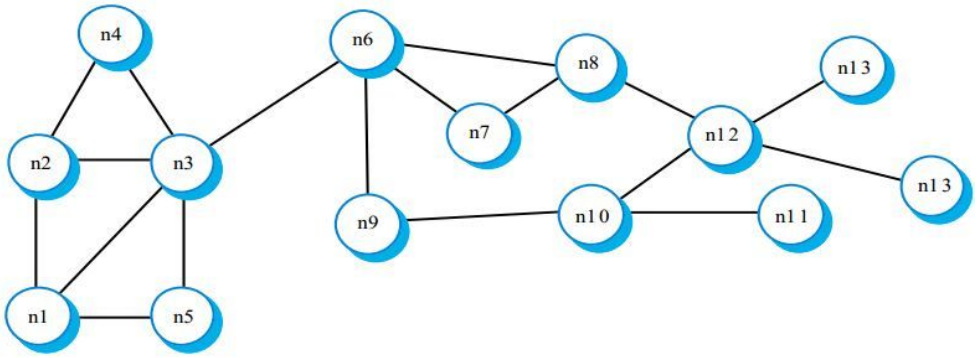
- neposkytuje model pre spoločné používanie dát, čiže podsystémy používajú rozličné spôsoby organizovania dát. Vzájomná výmena dát môže byť neefektívna
- nadbytočné manažovanie v každom serveri

Zdroj:

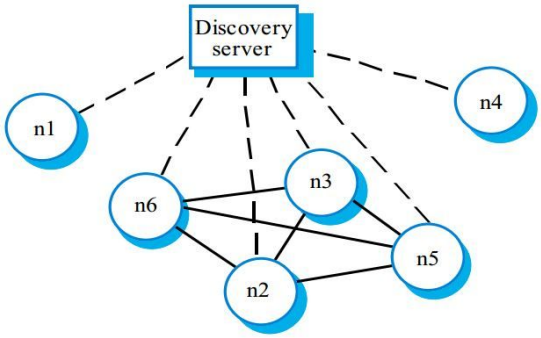
- ASS02ArchStyly.pdf
- aj: http://www2.fiit.stuba.sk/~polasek/courses/ass-sk/files/ASS05distrib.pdf

P2P

Decentralizovaná



Semi-centralizovaná



Definícia:
*Decentralizované architektúry kde **nie je** rozdiel medzi klientom a serverom. Počítania môžu byť vybavené rôznym uzlom v sieti. Celkový systém je navrhnutý tak aby bral výhody z výpočtového výkonu a úložiska.*

Výhody:

- *Výhodami* tejto architektúry je jednoznačne jej **robustnosť** (sieť nie je limitovaná počtom užívateľov). **Škálovateľnosť** a **spoľahlivosť**, ktorú P2P siete ponúkajú, je vyžadovaná v každom sieťovom prostredí.
- Tento systém je navrhnutý, aby dokázal zvládať ako **replikáciu**, tak aj **samoorganizovanie**. Máme možnosť využiť cetralizovaný i decentralizovaný prístup. Vybudovanie takejto siete je menej finančne náročné. Veľkou výhodou je i priama spolupráca klientov.

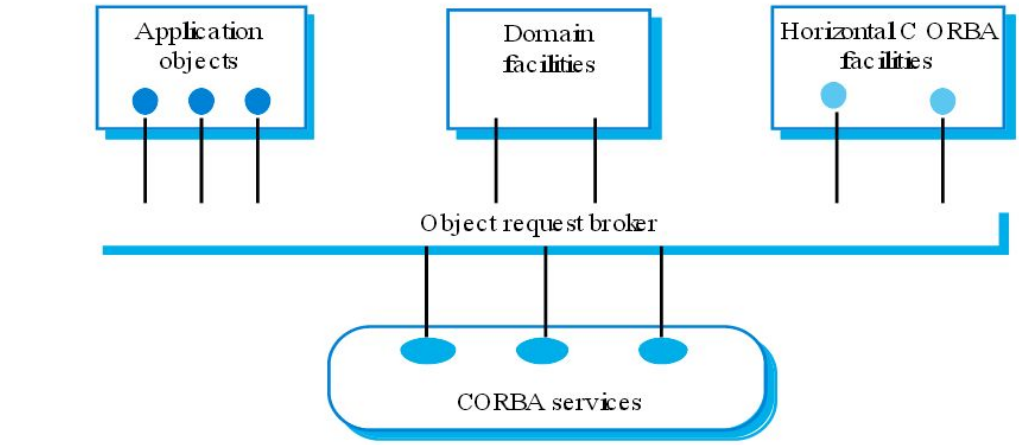
Nevýhody:

- Vďaka decentralizácii máme horšiu možnosť kontroly nad tokom dát. Distribuovanosť zdrojov občas spôsobuje ich nedostupnosť, čo nie je žiadúce. V tomto systéme sú kladené na klientov (uzly) omnoho väčšie požiadavky ako v prípade modelu klient-server.
- Nasadzované technológie nie sú často vyvinuté dostatočne, prípadne nie sú správne pochopené a implementované. Na záver, jednou z najväčších nevýhod je náročná správa bezpečnosti tejto architektúry.

Zdroj:

- ASS05distrib.pdf
- vyhody, nevyhody - **http://tinyurl.com/l6nrgy6 - str 6**

Corba



Definícia:

- **C**ommon **O**bject **R**equest **B**roker **A**rchitecture
- CORBA umožňuje písanie programov v rôznych programovacích jazykoch a ich spúšťanie na rôznych zariadeniach, pričom navonok môžu fungovať ako jedna aplikácia(celok) alebo set služieb. Presnejšie prináša **mechanizmus normalizácie volania metód**, či už vrámci jedného adresného priestoru (aplikácie) alebo zdieľaného adresného priestoru (rovnaký host, vzdialený host alebo prostredníctvom siete) [1]
- CORBA je medzinárodný štandard pre **Object Request Broker – middleware na manažovanie komunikácie** medzi distribuovanými objektmi
- CORBA štandardy middleware štandardov, ktoré podporujú distribuované objektové architektúrysú skupinou
- Middleware pre distribuované počítanie sa vyžaduje na 2 úrovniach:
    - Na úrovni logickej komunikácie, middleware dovoľuje objektom na rôznych počítačoch vymieňať dáta a riadiť informácie
    - Na komponentovej úrovni, middleware poskytuje základ na vývoj kompatibilných komponentov. Zadefinovali CORBA komponentové štandardy.
- je to štandard definovaný Object Management Group (OMG)

Výhody:

- Môžu medzi sebou komunikovať aj objekty napísané **v rôznych programovacích**

jazykoch

- **IDL** striktne definuje rozhrania (vysoká udržateľnosť a stabilita) [2]
- ľahká cesta ako prepojiť systémy [2]
- Vyspelosť - CORBA bola definovaná v 1991 a odvtedy sa pracuje na vývoji. Prístup "slow but sure" sa osvedčil a dnes CORBA obsahuje širokú škálu features, podporuje mnoho platforiem a progr. jazykov, op. systémov, tranzakcie, bezpečnosť, …
- Je to otvorený štandard - používateľ si môže vybrať zo širokej škály implementácii,
- Široká podpora platforiem a OS (spomínané vyššie), bárz aké Linuxy, Unixy, Winy, OS X atď.
- Podpora pr. jazykov (spomínané vyššie),
- Efektívnosť prenosu dát - napr. použitím tzv. marshallovania, napr. pomocou binárnej reprezentácie (ak poznáte JAXB, viete, je to niečo podobné...),
- Škálovateľnosť

Nevýhody:

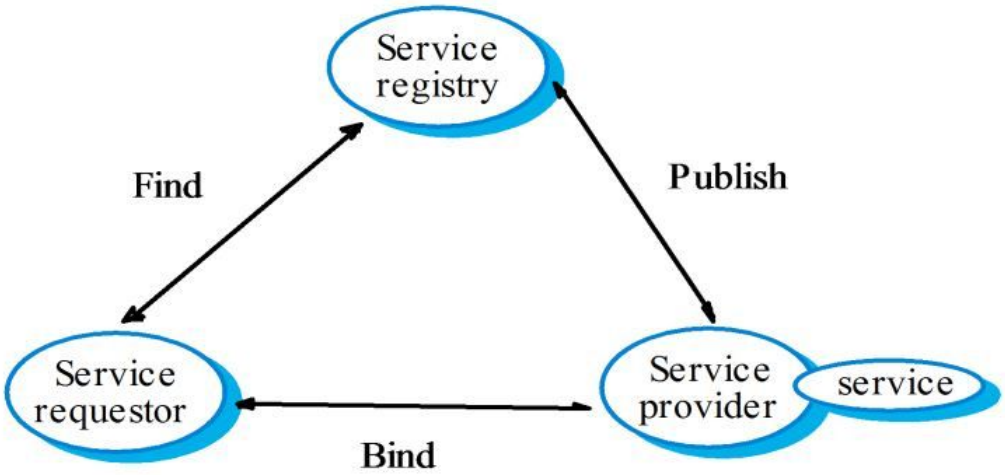
- **transparentnosť umiestnenia**: Ťažké(nemožné) určiť adresný priestor volanej metódy. Z toho plynie problém určenia stratégie výberu(1ms lokálny prístup vs 1s vzdialený prístup s potenciálnym rizikom znehodnotenia informácie prenosom a následným 30s timeouto-m) [1]
- **defekty návrhu a procesu:** problém s prístupom k definovaniu štandardu a la návrh podľa prispievateľa (design by comitee); z toho plynúce časté revízie s nutnosťou tvorby ďalších a ďalších rozšírení [1]
- náročnosť implementácie[2] (samotnej architektúry)
- nutné poznať IDL - problematická impl ala Copy&Paste [2]
- nový jazyk = nové IDL = veeeľa práce [2]
- "Firewall unfriendly" - neexistuje oficiálny štandard na viazanie ORB a ich klientov na porty, resp. rozsah portov, existujú iba proprietárne implementácie,
- Neexistujú oficiálne mapovania na niektoré jazyky, napr. Perl

//Nejaké ďalšie výhody a nevýhody:
http://wiki.answers.com/Q/What_are_the_advantages_and_disadvantage_OF_CORBA - výcuc pridaný do výhod/nevýhod.

Zdroj:

- [1] http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture
- [2] http://www.javacoffeebreak.com/articles/rmi_corba/

Web services



Definícia:
Service provider poskytuje služby. Tieto služby sú prístupné cez sieť pričom sú zadefinované v danom formáte. Service requestor využíva služby, pričom tieto služby sú nezávislé na implementačnom prostredí a jazyku. Existuje viacero typov služieb (napr SOAP, REST). Tieto služby sú bezstavové, pričom o šifrovanie a zabezpečenie sa stará priliehajúci trasportačný protokol.

Výhody:

- Jednoduchosť a podpora pre širokú škálu platforiem.
- Webová služba môže pridávať nové metódy bez toho, aby to ovplyvnilo činnosť klienta. (Webová služba musí ale poskytovať staré metódy a parametre.)
- Interoperabilita

Nevýhody:

- neobsahuje vlastný formát pre zabezpečenie a šifrovanie
- obtiažnosť vyhľadania služieb
- veľká réžia
- bezstavovosť

Zdroj:

- vyhody, nevyhody - **http://tinyurl.com/pz7lt4s - 1.5**