Architektura GIS z pohledu toků dat

Mgr. Tomáš Skopal
Katedra informatiky, FEI
VŠB – Technická univerzita Ostrava
tř. 17. Listopadu
708 33 Ostrava – Poruba
***E – mail**:* Tomas.Skopal@vsb.cz

Abstract

This article introduces original model of an open software architecture for GIS, which should hit the intent – accelerate and improve GIS applications design. First part deals with the solution motivation, second part is focused on formal mechanisms of choosen "data flow" paradigm. In this part, there is also established the "Pipe" idea – something like prefabricated component – on which depends the whole architecture design. In further text, there is developed the issue of Pipes interconnection leading to data-flow network creation. This network is the final design product and allows to provide analysis and synthesis of geographical data. Third part of the text describes the implementation of GIS Pipes catalogue. As the conclusion, an example is shown, demonstrating the real use possibilities and providing the reader complex overview. The article is an excerpt from author's diploma work published on KMI UP Olomouc. Software and detailed specifications are available on URL: http://www.inf.upol.cz/

Abstrakt

Tento článek představuje původní model otevřené softwarové architektury GIS, který má za úkol zrychlit a zkvalitnit návrh specifických aplikací GIS. V úvodu se zabývá motivací řešení, druhá část je věnována formalnímu aparátu zvoleného paradigmatu „toků dat". V této části je rovněž zavedem pojem „Pipe" – jakási stavebnicová součástka – která je klíčovou entitou celého návrhu. V dalším textu je rozpracováno propojení objektů Pipes do sítě, která, jakožto výsledný produkt konkrétního návrhu, poskytuje prostředky pro analýzu a syntézu geografických dat. Třetí část se věnuje implementaci katalogu Pipes (konkrétních součástek pro oblast GIS) a přinese praktickou představu použití. Závěr tvoří podrobný příklad, který demonstruje diskutované vlastnosti a poskytne čtenáři kompaktní přehled o problematice. Článek čerpá z diplomové práce, kterou autor řešil na KMI UP v Olomouci. Zmíněný software i podrobná specifikace celého návrhu je k dispozici na URL: http://www.inf.upol.cz/

1. Úvod a motivace

Vývoj databázových systémů přinesl v průběhu desítek let obecně použitelné technologie v oblasti manipulace s perzistentními daty. Naproti tomu oblast „informačních systémů" (tj. systémů, které zpracovávají hromadná data jako netriviální informace) potažmo systémů GIS, díky své rozmanitosti obecné zastřešující technologie postrádá. Koncové systémy jsou stále realizovány se značným dílem vynalézavosti, kdy systémoví analytici stále znovu vymýšlejí řešení „na míru" konkrétnímu zákazníkovi.

Problematika systémů GIS, specifických informačních systémů pracujících s geografickými objekty, s sebou nese velké množství rozmanitých aplikací. Tato rozmanitost

ovšem přináší jako neblahý vedlejší efekt značnou roztříštěnost konkrétních technologií pro návrh a implementaci různých systémů GIS. Velmi často je využíváno proprietárních řešení „lepících“ dohromady původně pro jiné účely navržený software (učebnicovým příkladem může posloužit produkt FRAMME od společnosti Intergraph). Téměř vždy poslouží jako nejnižší (datová) vrstva návrhu architektury GIS relační databázový model.

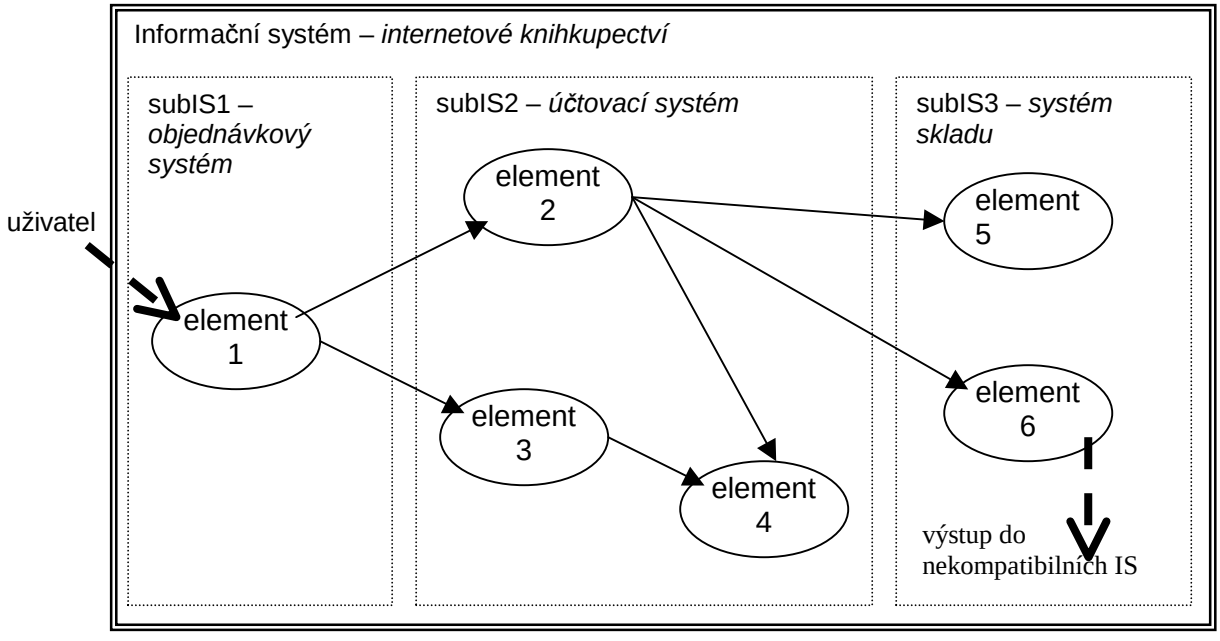
Motivací mého původního řešení bylo navrhnout a realizovat formální technologický aparát a následně otevřenou softwarovou architekturu, která by automatizovala a zrychlovala velkou část návrhu homogenních systémů hromadného zpracování dat. Homogenním systémem mám na mysli prostředí, ve kterém jsou zpracovávána data stejné nebo podobné povahy. Pro aplikace GIS se lze tomto případě soustředit na data geografické povahy.

2. Toky dat

Pokud vycházíme z podstaty funkce libovolného informačního systému, pochopitelně dojdeme k potřebě nějakým způsobem pracovat s daty. Problémem je, jak zadefinovat nějaký hmatatelný princip manipulace s daty tak, aby byl široce použitelný?

Moje představa je založena na zdůraznění **toků dat** v informačním systému (částečně vycházím z konceptu DFD – data flow diagramů). Data „protékají“ propojenými částmi systému, aniž by samotné tyto části „věděly“, jestli a s kterými ostatními částmi jsou propojeny – jinými slovy jak je celý systém **konfigurován**.

Situaci si můžeme ukázat na následujícím příkladě z prostředí elektronického obchodu:



Vycházíme z příkladu internetového knihkupectví. Uživatel objednává knihu prostřednictvím systému subIS1. Tento požadavek se zaeviduje a vyúčtuje přes systém subIS2 a nakonec se objedná ze skladu nebo od jiného dodavatele kniha prostřednictvím systému subIS3.

Pro náš případ knihkupectví by jistě stačilo zadefinovat komunikační rozhraní mezi třemi podsystémy, které dohromady tvoří internetový obchod. Použít „hrubou granularitu“ a

rozdělit celý systém na tři elementy. Tím by ovšem vzniklo řešení příliš specifické pro tento konkrétní případ – čím větší element, tím méně obecný – a tudíž by nebylo systémové.

Naproti tomu „rozumně jemná granularita“ způsobí rozdělení systému na více menších nezávislých jednotek – elementů systému – které jsou dohromady propojeny do **homogenní sítě**. Tyto elementy se propojují přes standardizované komunikační rozhraní (protokol) a takto tvoří funkční síť. Každý element je kombinovatelný s každým a pracuje lokálně – ve svém funkčním a datovém kontextu – přijímá data ze vstupů a transformuje je na výstupy. Stav (výstup) každého elementu tím **přímo závisí** na stavu vstupujících elementů.

Více „součástek“ sítě, resp. informačního systému, může vytvořit katalog a jednotlivé specifické systémy je možné „poskládat“ ze součástek z katalogu. Jedinou prací navíc je konfigurace součástek a jejich vzájemné propojení. Fungování takového stavebnicového systému se podobá klasickému IS, přes jednotlivé elementy lze klást lokální požadavky na datové prezentace či manipulace. Výhodou oproti klasickému návrhu IS je navíc splnění důležitých kritérií moderního otevřeného systému.

Chtěl bych zdůraznit, že předkládaný model je vhodný především pro homogenní informační systémy, ve kterých „proudí“ data stejného nebo podobného typu. U našeho uvodního příkladu internetového knihkupectví putují objednávky a faktury mezi systémy subIS1 a subIS2, expediční objednávky a dodací listy mezi subIS2 a subIS3. Všechny tyto dokumenty můžeme považovat za podobné a celý systém tím do jisté míry za homogenní. Oblast internetového obchodu podobným způsobem interpretuje například produkt BizTalk Server firmy Microsoft a související specifikace.

2.1 Formální koncept Pipes

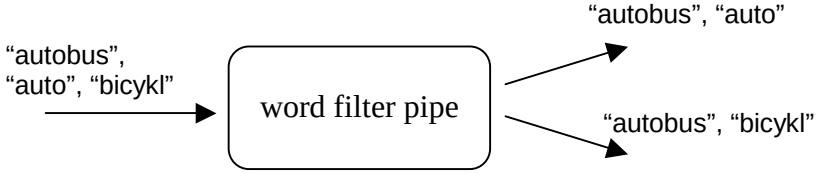
Nyní je potřeba dát síti elementů jasnější podobu. Pro reprezentaci elementu systému jsem zvolil pojem Pipe [pajp] - roura (tento termín zde má přes své dosavadní aplikace v IT nový význam).

Definice Pipe:

Pipe je tvořena posloupností vstupů I, posloupností výstupů O a posloupností sémantických funkcí S.

I určuje, které výstupy z jiných Pipes a v jakém pořadí vstupují dovnitř do Pipe. O určuje, které výstupy a v jakém pořadí poskytuje Pipe ven. Každému výstupu je přiřazeno právě jedno sémantické pravidlo z S. Sémantické pravidlo určuje, jak se trasformují data ze všech vstupů na konkrétní výstup.

Příklad:



pravidlo S_1: propusť z I_1 do O_1 taková slova, která začínají na “a“
pravidlo S_2: propusť z I_1 do O_2 taková slova, která obsahují “b”

Podle počtu vstupů a výstupů můžeme Pipes dále členit na:

a) **originator pipe** (žádný vstup) – původce dat. Protože nemá vstup, vytváří/poskytuje data svou logikou. V praxi to nejčastěji znamená poskytování dat z externích zdrojů (databáze, dokumenty, …) Jsou to *vstupní brány* do sítě Pipes.

b) **client pipe** (žádný výstup) – konzument dat. Je určena pro konečnou prezentaci dat uživateli nebo jako *výstupní brána* ze sítě Pipes. Jako brána může ukládat výsledky v různých formách – databáze, spreadsheety, dokumenty, HTML, DirectX, atd…

c) **multi – multi pipe**
Umožňuje složité vícevstupové analýzy s heterogenními (vícevýstupovými) výsledky. Zatím nemá elementární způsoby využití.

d) **multi – single pipe**
Slučovací pipe, která pracuje nad více zdroji, které zpracovává do jediného výstupu. Typickým využitím může být integrace datových skladů do jediného virtuálního, sofistikovanější typy Pipes přiřazují konkrétním vstupům role, podle kterých se na výstup transformuje výsledek funkce více parametrů.

e) **single – multi pipe**
Analyzuje vstupní tok dat a podle požadovaných vlastností třídí data do různých výstupů. Jako příklad aplikace mohou posloužit nejrůznější filtry.

f) **single – single pipe**
Asi nejpoužitelnější typ pipe. Transformuje data z jednoho vstupu na jeden výstup.
Nejčastější transformace jsou 1:1, tj. jedna datová entita se transformuje na jinou entitu.

Celý výše popsaný formální aparát ukazuje hrubé obrysy pro konkrétní softwarové řešení. Návrh jádra systému založený na předchozích konstrukcích je třeba navíc doplnit o platformově závislé a jiné specifikace. V další části se zaměřím na jádro knihovny Pipes pro GIS.

3. Piped GIS

PG.NET (Piped GIS .NET) je objektová knihovna navržená pro použití na běžných komponentových platformách jako je COM/DCOM/COM+, CORBA a zde v tomto případě nová platforma Microsoft **.NET Framework**. Vzhledem k zaměření na komponentové platformy byl model Pipes konkretizován podle možností komponentových technologií. Veškerá komunikace je synchronní (RPC), datová komunikace je zajištěna přes rozhraní obsahující metody a události.

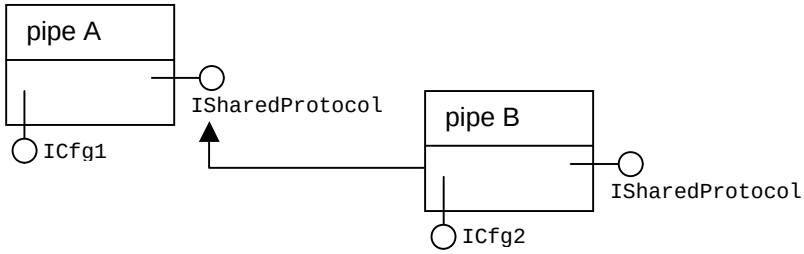
Celý životní (run-time) cyklus sítě Pipes můžeme rozdělit na dvě činnosti:

a) konfigurace sítě
b) vlastní chování sítě

3.1 Konfigurace sítě Pipes

Všechny třídy Pipes poskytují pomocí prostředků OOP sdílené rozhraní pro komunikaci mezi propojenými Pipes – přes toto rozhraní je dosaženo polymorfismu Pipes. Ostatní rozhraní, která Pipes implementují jsou nesdílená (tzv. konfigurační – specifická pro každou Pipe) a ta jsou využívána klientskou aplikací, která jednotlivé Pipes konfiguruje do sítě. Konfigurace sítě Pipes spočívá jednak v propojení Pipes přes sdílené rozhraní `ISharedProtocol` a jednak v nastavení specifických stavů Pipes přes konfigurační rozhraní `ICfgXXX`.

Schéma (kus sítě):



Pipe B je napojena svými vstupy na výstupy A přes sdílený protokol (znázornění je symbolické). B je ve vztahu k A klient a A ve vztahu k B server. B klade požadavky na A právě přes sdílený protokol. Prakticky je konfigurace sítě realizována klientskou aplikací, která může celou síť vytvořit – tj. instanciovat všechny Pipes – v rámci svého vlastního procesu. Tento způsob zajišťuje vytvoření sítě Pipes jako in-process serverů. Typickým příkladem je klientská aplikace ve Visual Basicu, která referencuje a používá knihovnu obsahující katalog Pipes jako kteroukoliv jinou komponentovou knihovnu. Tato aplikace zároveň celou síť používá a je tedy jejím jediným klientem.

Sofistikovanější konfigurace může probíhat v distribuovaném prostředí, kdy jeden klient celou síť vytvoří a nakonfiguruje do permanentně běžící služby na serveru a tím jeho funkce skončí. Nakonfigurovanou síť pak může v rámci dané služby používat/rekonfigurovat řada dalších klientů na různě distribuovaných platformách. K Pipes v této síti se potom přistupuje jako k out-of-process (resp. remote) serverům.

Společnou vlastností všech klientů sítě Pipes je znalost (alespoň částečná) topologie a sémantiky sítě – na rozdíl od jednotlivých Pipes, které (ačkoliv jsou si v síti navzájem také klienty a servery) jsou nezávislé.

3.2 Chování sítě Pipes

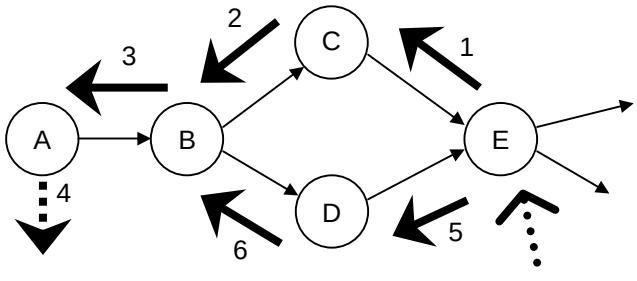
Nakonfigurovaná síť Pipes může plnit očekávanou funkcionalitu. Té je dosaženo prostředky sdíleného komunikačního protokolu mezi Pipes.

Sdílený protokol se skládá ze dvou složek:

a) metody – dotazy na informační obsah pipe
b) události – události změny obsahu pipe

3.2.1 Dotazy na obsah Pipe

V nakonfigurované síti Pipes lze pokládat dotazy na datový obsah jednotlivých Pipes (resp. jejích výstupů). Každá Pipe je buď ve stavu platném nebo neplatném. Na začátku jsou všechny Pipes ve stavu neplatném. Po obdržení požadavku na data na výstupu Pipe zjistí svůj stav, pokud je neplatný, vyžádá si data od všech Pipes, která do ní vstupují. Jakmile obdrží data ze svých vstupů, poskytne zpracovaná data na výstup a označí svůj stav za platný. Takto se vyvolá série požadavků směrem nahoru po zřetězených Pipes, na jejímž konci klient obdrží požadovaný obsah výstupu Pipe. Pokud je Pipe v platném stavu, předají se její data okamžitě, bez potřeby volání série dotazů na vstupech.



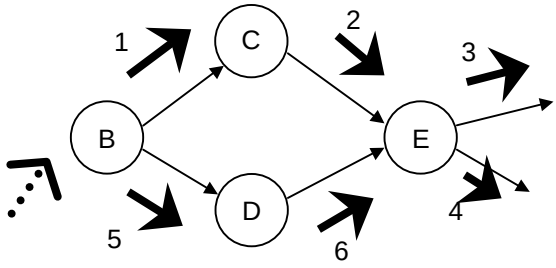
Příklad:
Klient požaduje data z E (resp. data jednoho ze dvou výstupů). Vyvolá se série dotazů zjištění datového stavu Pipes. Sekvence volání dotazů je serializována (vyplývá z povahy volání metody komponenty, resp. RPC).

Dotaz na první vstup E:
C→B→A – zde je datový stav A zjištěn (z externích zdrojů) a proběhne série návratů až do E. A,B,C jsou již označeny jako platné.
Dotaz na druhý vstup E:
D→B – stav B už je platný, tudíž nemusí proběhnout dotaz na A. B vrátí svůj stav.

3.2.2 Události změny obsahu Pipe

Pokud se datový obsah Pipe P z nějakého důvodu změní (objeví se nová data, vymažou se některá stávající data, modifikují se data), vyvolá se událost a ostatní Pipes, do kterých ústí výstupy z P, aktualizují na základě této události svůj obsah.

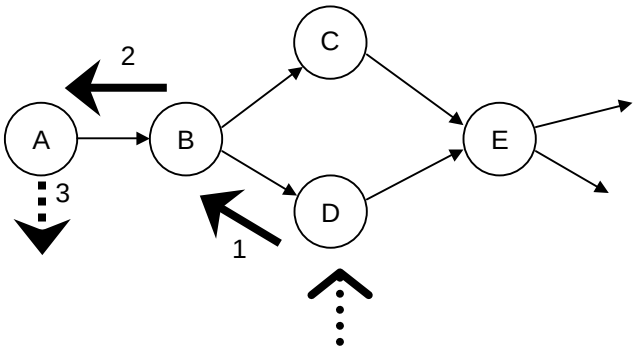
Příklad:
V B nastane změna, která se událostmi šíří všemi výstupy. Podobně jako u dotazů je vyvolání události serializováno.



Událost na první výstup B:
C→E→další… – událost způsobí aktualizaci obsahu všech dotčených Pipes v **platném** stavu
Událost na druhý výstup B:
D→E – událost v E je nyní ignorována, protože stav již je aktuální
Dotazy a události společně zajišťují **iluzi toku** dat. Za zmínku stojí fakt, že dotazy a události nejsou symetrické operace vzhledem k objemu přenesených dat. Zatímco většina dat „teče“ pomocí dotazů (směr klient→server), události pouze korigují stav sítě tak, aby zůstala konzistentní (směr server→klienti). Na této úrovni návrhu již také odpadají různé spekulace o „rychlosti toku“, synchronnosti, atd…

3.2.3 Propagace modifikací

Až dosud se prostřednictvím sítě Pipes dala analyzovat a číst data. Nyní potřebujeme zavést mechanismus, kterým by data šla zvenčí přidávat, upravovat nebo mazat. Tento mechanismus jsem nazval *propagace modifikací*. Modifikovat data v celé síti lze klientským požadavkem, který se pošle do některé z Pipes (resp. do některého jejího výstupu) v síti. Pipe vyhodnotí legitimitu požadavku pro daný výstup a deleguje požadavek po svých vstupech výše ke kořenu (kořenům) sítě. Kořen (kořeny) se delegovanou změnou pokusí aktualizovat svůj obsah a v případě, že se tak stane, vyšlou sérii událostí změny obsahu.



Příklad:
Klient chce do sítě vložit nová data přes pipe D. Požadovaná operace se propaguje ke kořenu sítě. V A se změna zaznamená do externího zdroje (např. databáze) a následně je vyvolána séria událostí změny obsahu A. Tato série je známým způsobem zpracována (viz. předchozí sekce).

4. Demonstrační katalog Pipes pro GIS

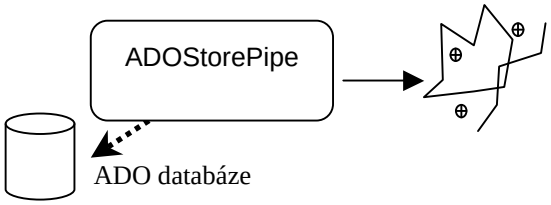
Data, která proudí mezi elementy v systému GIS se nazývají **Features** (prvky) a představují geografické jednotky – např. parcely, potrubí, silnice, atd…. Tyto jednotky zpravidla obsahují geometrii, identifikátor a nějaké popisné atributy. Identifikátor je určen třídou (**class**) prvku – např. parcela, budova – a identifikátorem v rámci třídy (**Id**).

Všechny PG.NET Pipes jsou autonomní objekty, takže celá síť může být distribuována po internetu.

Pro představu uvádím některé koncepty implementovaných Pipes:

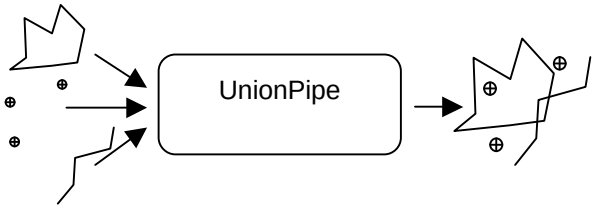
4.1 ADOStorePipe (originator pipe)

Poskytuje Features z databáze přes ADO.NET rozhraní.



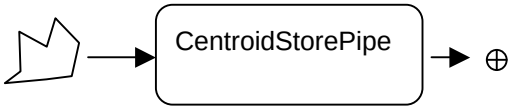
4.2 UnionPipe (multi-single pipe)

Spojuje data z více zdrojů do jediného.



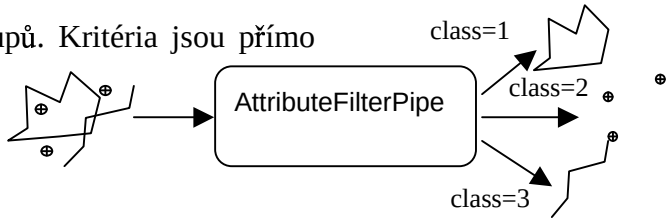
4.3 CentroidStorePipe(single-single pipe)

Vytváří nové prvky s geometrií centroidu, počítané z geometrie původního prvku.



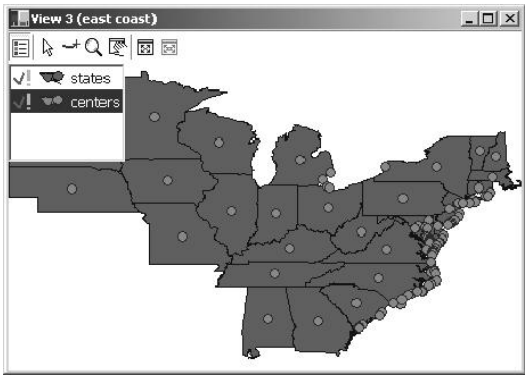
4.4 AttributeFilterPipe (single-multi pipe)

Třídí zdrojová data podle kritérií do více výstupů. Kritéria jsou přímo sémantická pravidla výstupů.
Např. ze zdroje městských sítí vyseparuje ulice, kanalizace, elektrické kabely, apod.



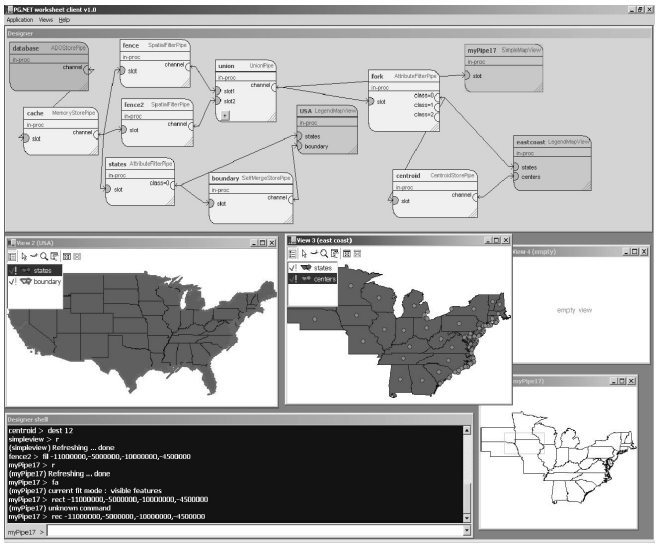
4.5 LegendMapView (client pipe)

GUI třída, která graficky prezentuje svůj obsah. Má více vstupů, z nichž každý je prezentován v samostatné vrstvě mapového okna. Umožňuje základní práci se zobrazenými prvky jako je zoom, posunování výřezu okna, práce s vrstvami. Dále umožňuje pokládat nové prvky do vrstev a následně jejich vytvoření propagovat výše.



Na obrázku vidíme GUI rozhraní třídy **LegendMapView**, které zobrazuje svoje dva vstupy (dvě vrstvy) – východní státy USA a jejich geometrické středy (centroidy – těžiště).

Pro potřeby demonstrace a testování bylo vytvořeno testovací prostředí – tzv. *PG.NET worksheet client* – které umožňuje implementované Pipes zapojovat a simulovat reálné případy použití. Na dalším obrázku vidíme prostředí PG.NET. V horním okně jsou schématicky zobrazeny Pipes a jejich propojení. Do ostatních oken (tzv. *Views*) lze umístit Pipes, které podporují zobrazování svého obsahu – v tomto případě klienty **SimpleMapView** a **LegendMapView**.
V dolním (černém) okně se přes příkazovou řádku konfigurují Pipes specifickými příkazy. Příkazy lze rovněž žádat Pipes o uvedení se do platného stavu.

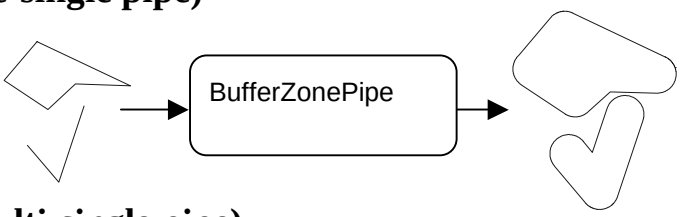


5. Příklad pro GIS

Pro lepší ilustraci uvádím fiktivní příklad z prostředí GIS, který naznačí praktické použití architektury Pipes. V příkladu jsou použity (vedle již definovaných) některé další Pipes, které nejsou součástí implementace balíku PG.NET, ale které by jistě nebylo problém realizovat.

5.1 BufferZonePipe (single-single pipe)

Vytváří ke geometriím prvků obálky, které obsahují geometrie zdrojových prvků a jejich okolí v zadané konstantní vzdálenosti.



5.2 IntersectFilterPipe (multi-single pipe)

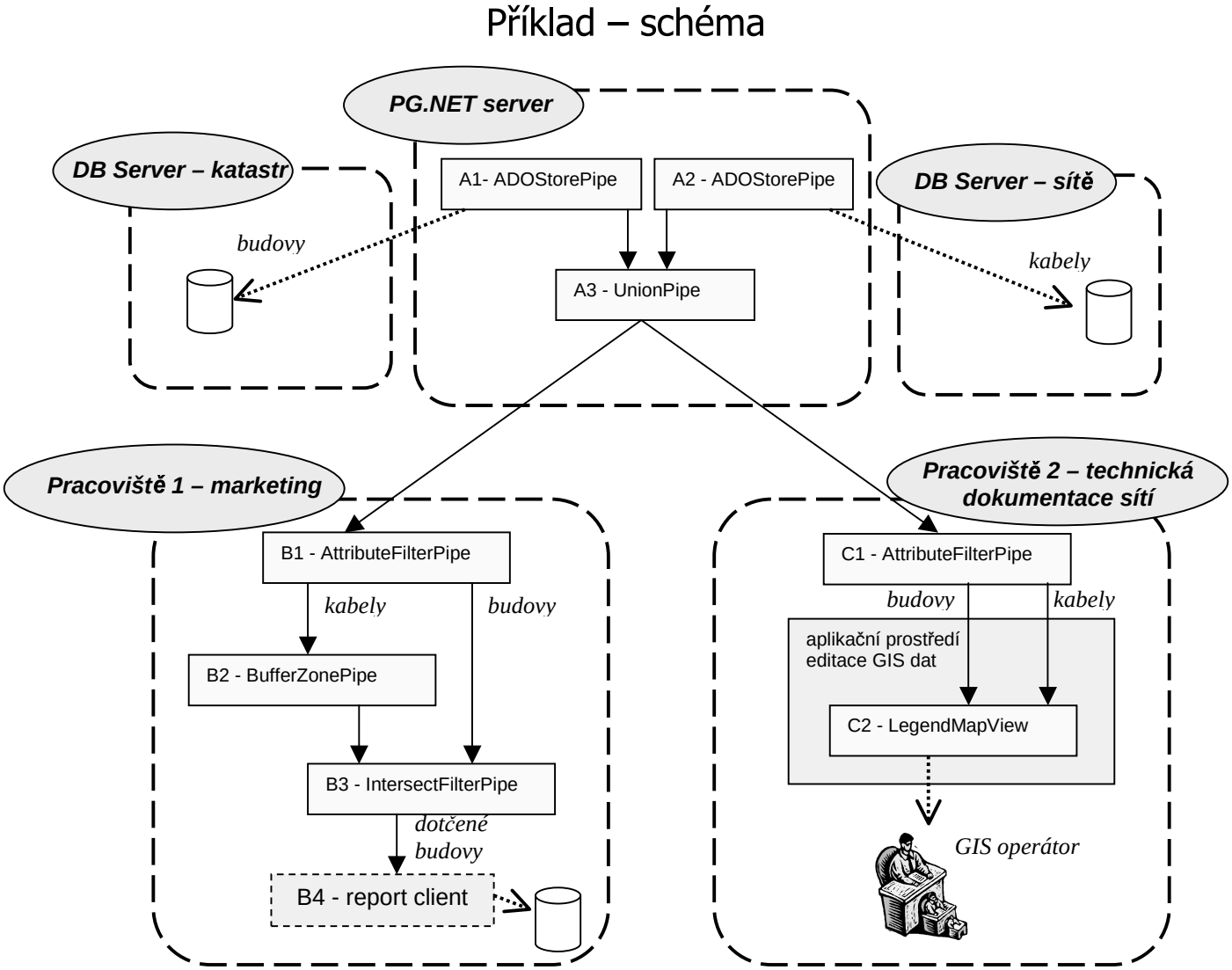
Filtr který propouští ty prvky z druhého vstupu, jež mají neprázdný geometrický průnik alespoň s jedním prvkem z prvního vstupu.

5.3 Popis schématu

Schéma na další straně popisuje tři části GIS systému telekomunikační firmy – např. kabelové televize.

Část *PG.NET server* zajišťuje integraci dvou zdrojů GIS dat a to: firemní databáze kabelových sítí a data budov z katastrálního úřadu. Tato integrace je realizována jednoduše přes dvě **ADOStorePipe** a slučovací **UnionPipe**.

Pracoviště 1 – marketing se zabývá získáváním nových zákazníků. Využívá k tomu informace získané ze stavu firemní kabelové sítě a polohy budov vůči této síti. Podle toho, zda leží budovy potenciálních klientů v těsné blízkosti stávající kabelové sítě se aktualizuje seznam těch potenciálních klientů, jimž by zavedení kabelové přípojky znamenalo minimální finanční náklady. Realizace této funkcionality je dosaženo zřetězením čtyř Pipes. První pipe – **B1 AttributeFilterPipe** – rozdělí zdroj dat na *kabely* a *budovy*. Do **BufferZonePipe** s nastavenou tolerancí – např. 50 metrů – vstupují data *kabely*. Do **IntersectFilterPipe** vstupují prvním vstupem vytvořené obálky kabelové sítě a druhým vstupem budovy. Výstupem jsou ty budovy, které mají neprázdný průnik s obálkami. Takto získaná množina prvků vstupuje do klienta **Report Client**, který získaná data transformuje ven ze systému Pipes na požadovaný výstup. Tímto výstupem mohou být tabulky MS-Excel, databáze nebo jen textový soubor.



Pracoviště 2 – technická dokumentace sítí se stará o aktualizaci digitální formy dokumentace kabelové sítě. GIS operátoři (lidé u počítače) digitalizují nové geodetické zákresy kabelů nebo upravují stávající data. Realizace je dosaženo opět oddělením budov a

kabelů pomocí **C1 AttributeFilterPipe** a následným napojením na koncového klienta **LegendMapView** v rámci desktopové GIS aplikace. Operátor může digitalizuje data přímo přes tohoto klienta, který propaguje veškeré změny postupně až **A2 ADOStorePipe** na *PG.NET serveru*. Operátor může pochopitelně aktualizovat pouze vrstvu/vstup kabelových sítí, data budov jsou mu k dispozici pouze pro čtení (už **A1** je poskytuje jen pro čtení). Pokus o propagaci přidání budovy přes **C2** tedy projde až do **A1**, tam ale skončí neúspěchem.

Poznámky:

Vlastností sítě Pipes je konzistence a integrita dat v síti. To znamená, pokud GIS operátor potvrdí položení nového kabelu, tato změna se okamžitě promítne až do **Report Clienta** a to prostřednictvím série propagací/událostí. Podobně, pokud **A1 ADOStorePipe** zaznamená změnu v databázi katastru, vyšle událost, která se promítne jak do **Report Clienta**, tak do příslušné vrstvy **LegendMapView**.

Příslušná pracoviště i *PG.NET server* jsou znázorněna čárkovanými rámečky. Tyto rámečky mohou představovat distribuované platformy v kontextu více počítačů na internetu, v kontextu LAN nebo v kontextu více procesů jednoho systému. Tato hlediska jsou zajímavá spíše pro analytiky, systémové integrátory, administrátory. Sítě Pipes mohou fungovat i skrytě, v kontextu jediné desktopové aplikace. Takový přístup už je předmětem zájmu vývojářů, kteří Pipes využívají jako kteroukoliv objektovou knihovnu.

Tuto transparentnost umožňuje implementace pomocí standardních komponentových technologií, v tomto případě .NET Framework. Koncová aplikace, která Pipes konfiguruje (vytváří a spojuje), sama určí, kde a jak Pipes vytvoří. Komunikace Pipes zajišťují nižší systémové služby – marshalling, resp. RPC/RMI.

6. Závěr

Sítě Pipes najdou hlavní uplatnění v homogenních technologických IS. Z předchozího příkladu vidíme, že celým systémem *protékají* data jednoho typu – geografické objekty.

Zobecnění celé architektury by znamenalo zadefinovat pro vstupy/výstupy Pipes další sémantická pravidla, například typ transportovaných prvků, jejich uspořádání, transakční pravidla, bezpečnostní politika, atd…

Detailní analýza a design architektury Pipes, stejně jako implementace balíku PG.NET, je k dispozici ve výše zmíněné diplomové práci.

Literatura

Skopal T.: *Technologie hromadného zpracování dat*. KMI UP Olomouc, 2001.

Gamma E. et al: *Design Patterns*. Addison-Wesley Pub Co, 1995. ISBN 020-163-36-12.

Fowler M.: *Analysis Patterns*. Addison-Wesley Pub Co, 1996. ISBN 020-189-54-20.

Korte G.B.: *The GIS Book*. Delmar Publishing, 1998. ISBN 076-682-82-04.

Rogerson D.: *Inside COM*. Microsoft Press, 1997. ISBN 157-231-34-98.

Brockschmidt K.: *Inside OLE*. Microsoft Press, 1995. ASIN 155-615-84-32.

Fowler M. et al: *UML Distilled*. Addison-Wesley Pub Co, 1999. ISBN 020-165-78-3X.

Intergraph Corp.: *Working with GeoMedia*. Intergraph, 2000.

Microsoft: *.NET Framework SDK beta*. Microsoft MSDN, 2000.