UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Evidenčné číslo: 23471651-daf4-432f-bbda-ebcd892bf652

SÚBOROVÝ MANAŽÉR

Rok predloženia: 2011

Peter Eliáš

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Súborový manažér

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	9.2.1 Informatika
Školiace pracovisko:	Katedra Informatiky FMFI
Školitel:	RNDr. Richard Ostertág, PhD.

Bratislava 2011

Peter Eliáš

Abstrakt

Práca sa zaoberá návrhom programu typu "Súborový manažér" na platforme Microsoft .NET Primárnou motiváciou je popísať, ako má vyzerať moderné grafické užívateľské rozhranie (GUI) a oboznámiť čitateľa s grafickou knižnicou WPF.

Sekundárnou motiváciou je dodanie aplikácie orientovanej na potreby užívateľa, keďže riešenia prítomné na súčasnom trhu nekladú na tieto potreby dôraz.

Abstract

This work is treatise concerning design of “File manager”-type program on Microsoft .NET platform.

Primary motivation is to describe how modern graphical user interface (GUI) should look like and acquaint reader with WPF graphic library.

Secondary motivation is to deliver application dedicated to user’s needs, since solutions which are present on contemporary market do not put emphasis on these needs.

Čestné prehlásenie

Čestne prehlasujem, že som túto prácu vypracoval sám s použitím uvedených zdrojov.

Obsah

Úvod

Táto práca pojednáva o programe typu súborový manažér. Čo však je tento “súborový manažér”? Na začiatok je potrebné zaviesť základné pojmy, s ktorými narába táto práca.

Súborový manažér je program, ktorý umožňuje

1. kopírovať,
2. premiestňovať,
3. premenovávať,
4. vymazávať

objekty nachádzajúce sa v súborovej štruktúre. Tieto operácie nad súborovu štruktúrou budeme ďalej označovať skratkou KPPV.

Súborová štruktúra je prehľadne organizovaná množina súborov, pomocných objektov a operácií, ktoré je nad nimi možné vykonávať (nepliesť si so súborovým systémom). Táto množina je vytváraná operačným systémom (ďalej len OS). Operačný systém umožňuje používateľovi pracovať so súbormi z takejto abstraktnej množiny, avšak v skutočnosti pracuje so súbormi uloženými na pamäťových médiách počítača, ako sú Hard disky, Solid state disky, optické disky, USB kľúče a iné.
Pomocné objekty adresárovej štruktúry sú adresáre a pamäťové médiá. Tieto pomocné objekty slúžia na prehľadnú organizáciu súborovej štruktúry, v ktorej je možné súbory ľahko nájsť.

Teraz uvedieme zjednodušené popisy dvoch konkrétnych súborových štruktúr.

Súborová štruktúra v OS Windows (ďalej len WinFS): Súbory sú organizované ako les stromov. Každý strom predstavuje jedno pamäťové médium počítača a obsahuje len súbory z tohto média. Uzly stromu sú priradené reálnym objektom nasledovným spôsobom:

1. koreň je fyzické médium, na ktorom sú uložené súbory
2. vnútorné uzly sú adresáre
3. listy sú súbory

Súborová štruktúra v OS Linux: Súbory sú organizované ako jeden strom súborov. Uzly stromu predstavujú nasledovné objekty:

1. koreň stromu je abstraktný objekt operačného systému, ktorý slúži len k tomu, aby tvoril strop hierarchie.
2. vnútorné uzly sú adresáre a fyzické médiá. Fyzické médiá OS Linux vystavuje ako súčasť súborovej štruktúry, pričom ich obsahom nie je súborová štruktúra, ktorá by vznikla ich interpretáciou operačným systémom, ale konkrétne bity, ktoré je možné čítať vo forme v akej sú uložené na médiu.
3. listy sú súbory

Uvedené príklady súborových štruktúr sú len zjednodušenými popismi, nie všeobecnými vodítkami pre tvorbu súborových štruktúr. Pri tvorbe OS sa tvorcovia dohodnú na vytvorení takej súborovej štruktúry, ktorá odpovedá cieľu ktorý chcú dosiahnuť. Cieľom pri návrhu súborovej štruktúry je spravidla prezentovať pohľad na uložené súbory, v ktorom sa užívateľ ľahko zorientuje a ľahko nájde súbor, s ktorým chce pracovať medzi desaťtisícmi iných súborov.

Napríklad, keby sme chceli kompletne popísať súborovú štruktúru reálnych OS, tak treba povedať, že v mnohých OS je možné včleňovať do súborovej štruktúry súbory nachádzajúce sa na inom počítači (prístupnom cez sieť), prípadne priradiť niektorým typom súborov dodatočné vlastnosti ktorým rozumie aj OS, ako je napríklad vlastnosť “byť odkazom na iné objekty súborovej štruktúry”. Treba si uvedomiť, že v reálnom prostredí čitateľ narazí na mnohé špeciality jednotlivých OS. Avšak podstata súborovej štruktúry zostáva vždy rovnaká a pre potreby tejto práce postačuje jednoduchá definícia zo začiatku.

V tejto práci budeme pracovať s WinFS. Je na to niekoľko dôvodov. Za prvé, WinFS je intuitívny vďaka prezentácii pamäťových médií ako najvyšších uzlov súborovej štruktúry, teda užívateľ sa v ňom dokáže ľahko zorientovať. Za druhé, súborový manažér, ktorý je predmetom tejto práce, je písaný pre OS Windows.

Na začiatku bolo povedané, že existuje pojem súborový systém, ktorý treba odlišovať od súborovej štruktúry. Systém a štruktúra sú síce významovo veľmi príbuzné termíny, pojednávame tu však o odlišných objektoch.

Súborová štruktúra je termín zavedený pre potreby tejto práce. Koncept, ktorý predstavuje, nie je rozšírený v odbornej literatúre. Používateľ OS ho vníma ako programové rozhranie (ďalej API), ktorým OS prezentuje súbory programom, ktoré pod OS bežia. OS tiež umožňuje programom vykonávať pomocou tohto API aj KPPV operácie.
Súborový systém je oproti tomu dobre zavedený a často používaný termín. Je to spôsob, akým sú súbory kódované do série bitov, ktoré sú potom nižšou vrstvou (ovládačom disku) zapísané na fyzické médium. OS sa snaží uchrániť programy od nutnosti preklápať medzi súborovým systémom a súborovou štruktúrou, aby mohli tieto programy priamo pracovať so súbormi.

Táto práca je rozdelená na niekoľko kapitol, ktoré zodpovedajú fázam kreatívneho procesu tvorby aplikácie. Prvá kapitola (Analýza) sa zaoberá analýzou situácie na trhu a identifikovaním nedostatkov existujúcich produktov. Druhá kapitola (Špecifikácia) popisuje návrh aplikácie, ktorá rieši rovnaký doménový problém ako konkurenčné produkty, avšak sa vyhýba spomínaným nedostatkom. Tretia kapitola (Implementácia) obsahuje technický popis navrhovanej aplikácie v zmysle jej vnútorného členenia a použitých technológií.

1. Analýza

Vynoruje sa jedna otázka, ktorú je potrebné zodpovedať: Prečo písať ešte jeden súborový manažér? Na pochopenie tejto nutnosti je potrebné spraviť analýzu funkcionality existujúcich súborových manažérov. Touto analýzou sa bude zaoberať nasledujúca kapitola. Kapitola nemá za cieľ používať formalizované analytické postupy na vyšetrovanie reálneho sveta. Cieľom je vyvolať rozhorčenie v čitateľovi nad dizajnovými nedostatkami súčasných programov a prinútiť ho všímať si veci, ktoré mu samotnému vadia a chcel by ich zmeniť. Bez aktivity na strane používateľov sa totiž používateľske rozhranie súčasných programov zlepšovať nebude.

Na výber programov do vzorky, ktorú budem analyzovať, som si zvolil niekoľko kritérií.

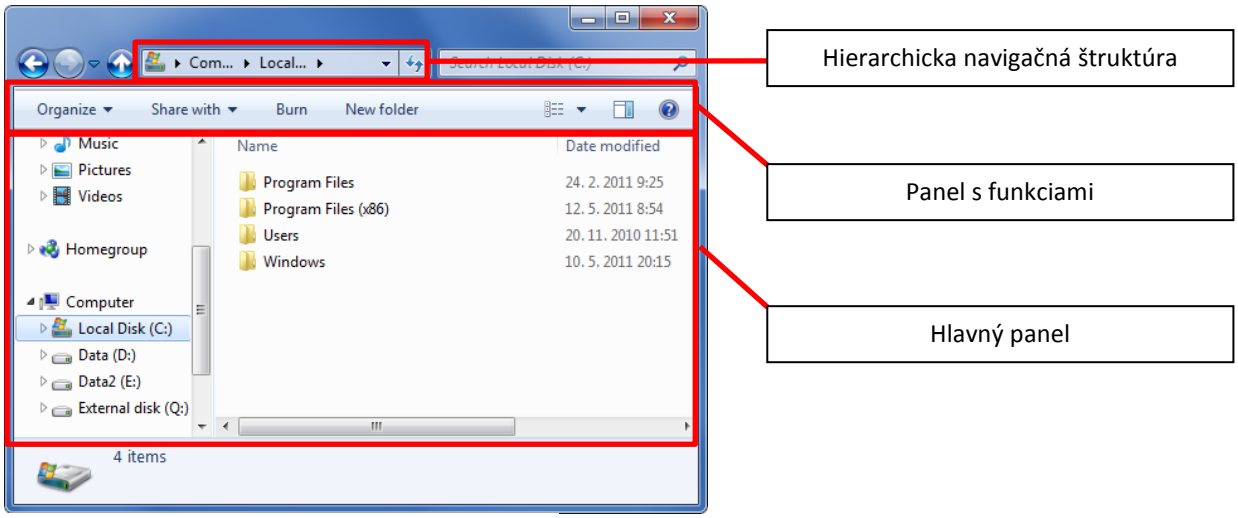
1. Porovnávané programy sú písané pre Windows. Zamerali sme sa na tento operačný systém, keďže navrhovaná aplikácia, ktorá odstraňuje chyby týchto programov, bude musieť byť z technických dôvodov písaná tiež pre tento operačný systém. Detaily sú preberané v kapitole Implementácia.
2. Porovnávané programy sú dostupné zadarmo vo svojej plnej verzii. To znamená, že sa nejedná o platené programy, alebo programy dostupné vo viacerých verziách, z ktorých tá najmenej vybavená je dostupná zadarmo.
   Je to dobré eliminačné kritérium, keďže aj súborový manažér APE (ďalej len APE), ktorý je popisovaný v tejto práci, má byť dostupný zadarmo pod Open Source licenciou. To znamená, že program sa snaží konkurovať programom, ktoré sú široko prístupné a používané, programom, ktoré ponúkajú všeobecnú funkcionalitu na prácu so súbormi, nie úzko špecializovaným produktom pre malú klientelu. APE sa snaží byť náhradou každodenne používaného súborového manažéra, programu po ktorom každý intuitívne siahne, keď dennodenne vykonáva desiatky bežných operácií nad súborovou štruktúrou. Podozrievavý čitateľ sa môže domnievať, že toto kritérium je použité zo strachu pred porovnaním s programami, ktoré sú dostupné za peniaze. Ich funkcionalita by totiž mala byť aspoň papierovo bohatšia než výtvory nadšencov. Ako však uvidíme, zvýšený počet funkcií sa nerovná väčšej použiteľnosti softvéru, čo je aj jedna z dizajnových smerníc, ktoré boli použité pri tvorbe tohto programu. Porovnávanie s komerčnými programami je preto zbytočné.

Do testovacej vzorky sa dostali nasledujúce programy. Zoradené sú podľa poradia, v ktorom budú analyzované.

- Windows Explorer (ďalej len WinExplorer)
- CubicExplorer
- FreeCommander
- Double Commander
- unreal commander
- q-dir
- Explorer++
- STDU Explorer
- xplorer^2

Ich jednotlivým plusom a nedostatkom sa budeme podrobne venovať na ďalších stránkach. Výstupom z tejto analýzy budú poznatky určujúce požiadavky na funkcionalitu a užívateľské rozhranie pre môj súborový manažér. Podľa týchto poznatkov bude ľahké zostrojiť špecifikáciu náhrady WinExplorera.

Najprv by som však vymenoval spoločné grafické komponenty, ktoré všetky aplikácie typu súborový manažér používajú na poskytnutie viac-menej navzájom podobnej funkcionality. Na tieto komponenty sa často budeme odvolávať pri popise jednotlivých súborových manažérov. Súborový manažér pozostáva minimálne z:



Obrázok 1: Časti súborového manažéra

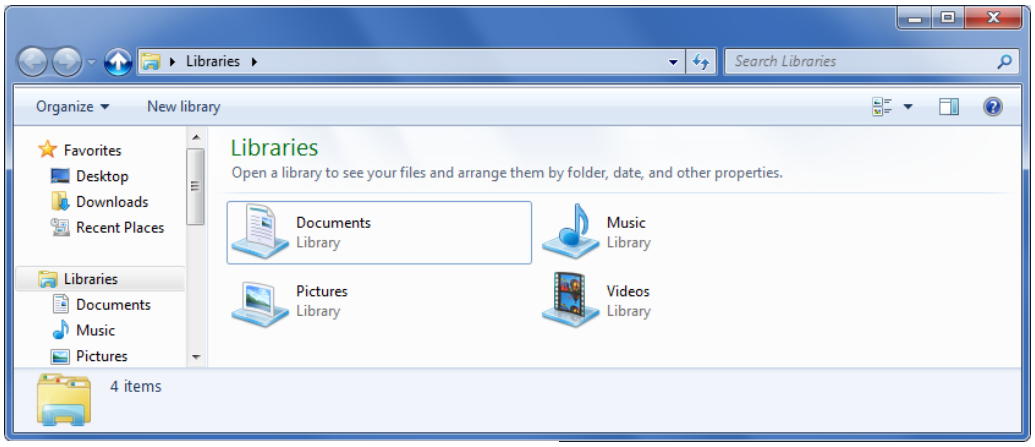
- hlavný panel zobrazuje obsah zvoleného adresárového uzla
- hierarchicka navigačná štruktúru umožňuje pohybovať sa medzi jednotlivými uzlami
- panel s funkciami umožňuje vydávať povely aplikácii

1.1. Windows Explorer

Jedná sa o súborový manažér dodávaný s OS Windows. Pre väčšinu ľudí je jediným súborovým manažérom, ktorý poznajú. To znamená, že keď sa povie súborový manažér, myslí sa tým práve tento program. Častokrát pri popise pojmu súborový manažér je rovno vhodné vymenovať tento program ako definiciu pojmu manažér a uzavrieť tému. Väčšina laikov totiž ťažko porozumie tomuto pojmu z výčtu činností (kopírovanie, vymazávanie, …) ktoré manažér vykonáva, ale skôr z konkrétneho príkladu programu, s ktorým sa stretli počas používania počítača.

Existuje viacero verzií WinExplorera, každá verzia OS Windows je dodávaná s novšou verziou. Ja sa však sústredím na verziu dodávanú s Windows 7, ktorá je momentálne najaktuálnejšiou a v nadchádzajúcom čase sa nebude radikálne meniť, ako tomu bolo pri prechode z Windows XP na Windows Vista.

WinExploreru budeme venovať väčšiu časť analýzy, jedná sa o najpoužívanejší program zo vzorky pretože sa dodáva s najpoužívanejším OS, ktorým je Windows. Tým pádom je benchmarkom, oproti ktorému sa porovnávajú konkurenčné aplikácie.



Obrázok 2: Štartovacia obrazovka Windows Explorera

Začal by som pozitívnymi vlastnosťami WinExplorera, ktoré sa je nutné snažiť zachovať pri tvorbe jeho náhrady.

1. Užívatelia sú dobre oboznámený s programom, pretože sú s ním nútený pracovať každodenne. Chtiac-nechtiac sa naučili používať väčšiu časť jeho funkcionality a sú navyknutý na istú používateľskú schému. Túto bude potrebné aspoň okrajovo dodržať pri tvorbe nového programu.
2. Užívateľské rozhranie (ďalej len UI) pôsobí ako súčasť OS, pretože WinExplorer bol tak implementovaný. Je to integrálna súčasť Windows. Používa rovnaký grafický štýl, ovládacie prvky, integruje sa do panela úloh. Súčasťou WinExplorera je tiež dialógové okno na výber/otvorenie/uloženie súborov prístupné z iných aplikácií (File Open/Save dialog), integrované do OS v zmysle predchádzajúcej vety. Jednoducho, práca so súbormi je štandardizovaná v celom OS, a ako štandard je použitý WinExplorer.
3. UI aspoň čiastočne nasleduje moderné trendy. Rozhranie sa škáluje spolu s rozlíšením monitora (vďaka grafickej knižnici WPF). Sú tu vkusne použité farebné prechody a čistý grafický dizajn (veľké homogénne farebné plochy a malý počet farieb). Klikateľné plochy a väčšina ikon je primerane veľkých a užívateľ teda nemusí loviť myšou klikateľné miesta. Prínos takého dizajnu je že si užívateľ osvojí UI oveľa rýchlejšie, pretože veľmi rýchlo získa predstavu o funkciách aplikácie a o tom, kde sa zhruba nachádzajú. Väčšina funkcií je prístupných na 2 kliknutia, buď cez kontextové menu alebo cez ponuky v paneloch nástrojov, čo znamená že sa dajú ľahko použiť. Užívateľ tiež dostáva vizuálnu spätnú väzbu v podobe zvýraznenia klikateľných plôch keď sa nad nimi nachádza myš. Má to tri výhody:

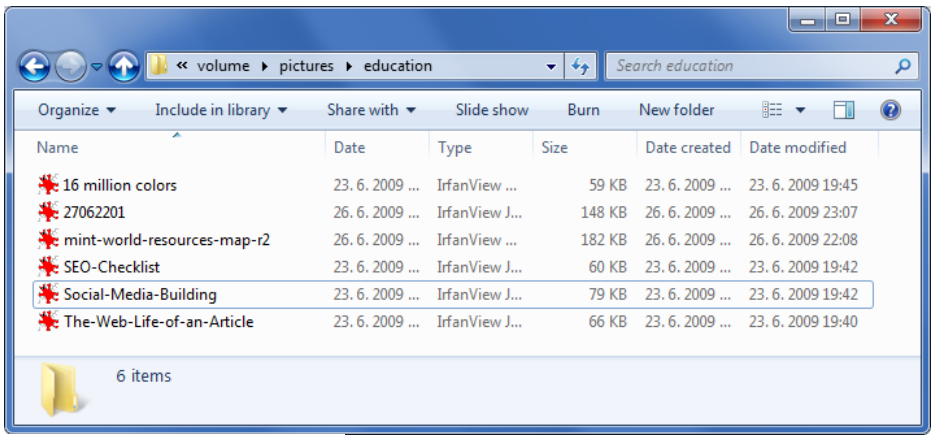
I. rozhranie nie je plné čiar a grafiky, ktorá nenesie žiadnu funkčnú informáciu
II. užívateľ priebežne spoznáva rozhranie aplikácie a nadobúda predstavu o tom, čo aplikácia ponúka
III. užívateľ vie, kedy sa nachádza kurzor myši nad klikateľnou plochou

4. Ako jediný používa koncept štartovacej obrazovky. Je to stav aplikácie, do ktorého sa aplikácia dostáva automaticky po štarte. Hlavný panel nezobrazuje obsah niektorého existujúceho uzla súborovej štruktúry, ale obsah fiktívneho uzla. Fiktívny uzol pozostáva z odkazov na iné často prezerané uzly a je väčšinou prezentovaný ako súčasť súborovej štruktúry. V prípade WinExplorera je na štartovacej obrazovke zobrazený fiktívny uzol Libraries (viď Obrázok 2).

Väčšia časť analýzy WinExplorera bude však venovaná analýze nedostatkov a možných vylepšení.

1. Vizuálny šum (viď Obrázok 3: Vizuálny šum).
   Toto negatívum sa bude opakovať u každého programu. Vizuálny šum znamená vizuálnu informáciu, ktorá je pre užívateľa zbytočná, avšak nachádza sa na hlavnej obrazovke programu. Výsledkom je prostredie, ktoré je veľmi namáhavé na oči. Užívateľ je pri používaní programu bombardovaný vizuálnymi vnemami, ktoré nepotrebuje. Tieto vnemy ho odpútavajú a musí sa najprv zorientovať, aby našiel tú správnu funkciu ktorá spraví to, čo požaduje. Príkladom nezmyselne prezentovaných informácií sú napríklad nový Status bar alebo ponuka funkcií na nástrojovej lište. Status bar obsahuje ikonu práve vybraného objektu, dátum jeho vytvorenia a niekoľko ostatných atribútov, ktoré sú však závislé od formátu vybraného objektu a nie je ich možné určiť. To má za následok, že v Status bare sú zväčša len neželané informácie. Jediná zmysluplná informácia, ktorá môže ušetriť čas je počet objektov v aktuálnom priečinku súborovej štruktúry. Aj keď na to nie je potrebné zaberať trikrát viac obrazovkovej plochy, než postačuje na prezentáciu tejto informácie. Samozrejme, dá sa zapnúť aj starý Status bar, ktorý túto informáciu ponúka v úspornej podobe. Problémom však je, že nový Status bar sa nedá vypnúť a teda pri zapnutí oboch Status barov by informácia bola prezentovaná redundantne na miestach vzdialených od seba zopár pixelov. Ďalším problémom sú nezmyselne prezentované funkcie na nástrojovej lište, ako napríklad “Slide show”, “Share with” alebo

“Burn”. Okrem toho, že tieto funkcie skoro nesúvisia s managementom súborov, tak percento ľudí, ktorý ich používajú je skoro nulové.



Obrázok 3: Vizuálny šum

2. Aplikácia by nemala užívateľa obťažovať grafickými prvkami, ktoré si nevyžiadal. Jedná sa o tooltip, ktorý sa zobrazuje pri zastavení myši nad niektorým uzlom zobrazovanom v hlavnom paneli. Táto banalita je dosť závažná, pretože ukazuje, ako sa dá pokaziť UI neuváženou funkcionalitou. Tooltip je z definície malý textový rámček, ktorý sa vykreslí pri zastavení myši nad prvkom rozhrania. Jeho zmyslom bolo pôvodne napovedať užívateľovi, aký je účel konkrétnej funkcie programu, ktorú má záujem použiť. Vďaka jeho invazívnej povahe je vhodný na popis častí UI, ktoré sa nenachádzajú v hlavnej pracovnej oblasti, hodí sa teda presne na podrobnejší popis položiek v paneli funkcií. Tu je však použitý na zobrazovanie informácií o každom uzle súborovej štruktúry. Okrem toho, že tieto informácie sú opäť redundantné, pretože opakujú informácie uvedené v Status bare, tak je táto funkcionalita pri tak celoplošnom nasadení značne rušivá a zabraňuje používateľovi efektívne pracovať.
   Ďalším rušivým prvkom je rámček okolo naposledy vybraného uzla, ktorý bol medzitým odznačený a používateľ doteraz nevybral žiaden iný uzol. Táto rámčeková dekorácia signalizuje jeden z troch možných stavov uzla, ďalšie dva stavy sú výber uzla (signalizovaný plným rámčekom) a nevybratý uzol (žiaden rámček okolo uzla). Účel tohto stavu, ktorého samotný opis je komplikovaný, zostáva záhadou pre používateľov na celom svete. Každopádne, tento rámček sa vyskytuje v súborových manažéroch Microsoftu už od čias Windows 95. Jeho jedinou funkciou je, že upútava pozornosť

užívateľa na uzol, s ktorým nechce ďalej pracovať a tým pádom zvyšuje vizuálny šum rozhrania. To má za následok menšiu efektivitu pri používaní aplikácie.

3. Nadbytočnosť funkcií.

   Ako už bolo spomenuté, WinExplorer obsahuje veľké množstvo funkcií, z ktorých väčšina je však užívateľom neznáma alebo ich jednoducho nepoužívajú. Dovolili by sme si tvrdiť, že 90% užívateľov používa to len jedno a to isté percento funkcií. Jednou z nepoužívaných funkcií je napríklad funkcia Undo. Pri ankete prejavilo vedomosť o existencii tejto funkcionality menej ako 10% dotazovaných. Čo je však šokujúcejšie, dotazovaná vzorka pozostávala z ľudí so silným informatickým pozadím. To znamená, že povedomie ľudí o dostupnej funkcionalite, či už užitočnej alebo nie, je žalostne malé. Z toho vyplýva, že neinformovanosť je rovnako závažný problém ako nezáujem. V prípade užitočnej funkcionality sa oplatí užívateľov o nej informovať, v ostatnom prípade je vhodné danú funkciu, . čo je však už problém dokumentácie (viď bod 6).

   Hlavné funkcie sú síce ľahko dostupné keď užívateľ vie, kam sa má pozerať, ale zvyšok len vyvoláva v užívateľovi pocit neistoty pri používaní rozhrania (viď Odsek 1 – Vizuálny šum). Otázka je, má vôbec zmysel ponúkať užívateľovi všetky tieto funkcie? Filozofiou návrhu programu by mala byť úzka špecializácia. Keď užívateľ otvorí okno aplikácie, má totiž v úmysle vykonať nejakú konkrétnu aktivitu. Všetky ostatné možnosti aktivít ho v tej chvíli len odpúvatajú. Ako jeden príklad za všetky uvedieme funkciu Burn prístupnú z panela funkcií. Tá nie je súčasťou tradičnej funkcionality súborového manažéra a využíva sa len zanedbateľné percento času oproti tradičným funkciám (KPPV). Konceptovo sa síce jedná o kopírovanie/premiestňovanie dát na optické médium, proces je však administratívne oveľa náročnejší ako len zadanie zdrojových a cieľového uzla súborovej štruktúry.

   Ak sa teda aj užívateľ rozhodne niečo napáliť na CD, rozhodne to nebude z prostredia programu, kde nie je možné nastaviť parametre napaľovania.

   Rozhodne to nebude z prostredia programu, ktorý nedokáže napalovať prepisovateľné médiá.

   A určite to nebude z prostredia programu, ktorý neumožňuje vytvoriť si zoznam súborov, ktorý chce používateľ napáliť.

   Je to akoby aplikácia na rezerváciu leteniek ponúkala tiež upratovaciu službu. Možno to je dobre mienené, ale v konečnom dôsledku to nemá na danom mieste čo hľadať.

   Ako bolo jasné z predchádzajúceho príkladu, tak ďalšou nevýhodou je nedostatočná

prepracovanosť daných funkcií. Veľký počet funkcií znamená menší čas strávný implementovaním jednotlivých funkcií a aktualizovaním funkcionality. To má za následok menšiu použiteľnosť funkcií, keďže autori sa snažia dodať prvú fungujúcu verziu a presunúť sa na inú časť programu. Poučením je, že pri návrhu funkcionality programu si treba dobre premyslieť, ktoré funkcie má a nemá obsahovať.

Súvisiacou nevýhodou je, že keď sa chce človek naučiť používať celú funkcionalitu programu, musí začať hrať akúsi Point&Click adventúru. Dôvodom je, že pri veľkom počte funkcií sa jednoducho všetky nevojdú na hlavnú obrazovku a teda sú poschovávané všade možne. Toto prispieva k frustrácii užívateľa z používania programu. Výsledkom je, že táto funkcionalita zostáva tak či tak nevyužívaná, pretože nikoho nebaví hrať sa na Sherlocka Holmesa.

Dobrým riešením je zvoliť filozofiu návrhu programu, ktorý je dnes veľmi populárny – minimalizmus. Príkladom tejto filozofie je napríklad prehliadač Google Chrome. Minimalizmu bude venovaná Špecifikácia.

Hlavným pokušením, ktorému je potrebné nepodľahnúť, je pridávanie funkcionality, ktorej jediný účel je rozšíriť množstvo funkcionality prítomnej v programe. Častokrát programátori neuvážene pridávajú novú funkcionalitu za účelom zlepšenia svojej aplikácie. Výsledkom je spravidla nejasné zameranie aplikácie, roztiahnutie vývoja na široké spektrum priemernej funkcionality a znížená použiteľnosť. Možno najprirodzenejší je tento trend u komerčných produktov, kde je navyše prítomná finančná motivácia dodať na trh novú verziu ktorá prinesie nové zisky. Anglický výraz na označenie tohto fenoménu je feature creep. Výsledný stav, keď program obsahuje viac funkcionality než je potrebné sa nazýva feature bloat. Príkladom takejto nadbytočnosti je napríklad možnosť triediť súbory podľa 300 rôznych parametrov, z ktorých skoro všetky sú špecifické len pre súbory určitého formátu.

Na druhú stranu je potrebné dať si pozor, aby nové verzie softvéru boli badateľne odlišné. V opačnom prípade má užívateľ pocit, že bol pri obstaraní si novej verzie podvedený a nedostal to, čo očakával. V tomto smere je Chrome zlým príkladom, pretože nové verzie vylepšujú jadro aplikácie, avšak používateľský zážitok zostáva stále rovnaký.

4. Neprispôsobenosť pre single-click interface.

   V tejto časti budeme hovoriť o nedokonalej implementácii single-click interface vo WinExploreri, najprv však musíme zaviesť pojmy a oboznámiť čitateľa so súčasným

stavom na poli UI.

Jedným z problémov súčasného UI na platforme Windows je, že niektoré prvky UI reagujú na dvojité kliknutie (dvojklik) a zároveň aj na jednoduché kliknutie (klik). Každý typ kliknutia však predstavuje inú udalosť v systéme a vyvoláva inú funkciu. Táto interakčná schéma sa nazýva double-click interface a ukazuje sa, že spomaľuje používateľov pri osvojovaní si práce s PC. Začínajúci používateľ má bohužiaľ strach z používania nových technológií a pristupuje veľmi opatrne k interakcii s PC v strachu že niečo pokazí. Keď sa mu povie, že dvojklik a klik znamenajú iné funkcie, tak pri každom kliknutí bude váhať, aby nespravil niečo čo by pokazilo počítač. Než získa cit pre používanie, tak sa môže stať, že sa vzdá a prestane PC používať vôbec. Radšej si zapne TV ktorá vyžaduje minimálnu interakciu.

Ďalšou nevýhodou dvoch typov kliknutia je, že zdržujú skúsených používateľov. Podvedomie musí pracovať tiež viac, aby sa správne rozhodlo a človek občas aj tak spraví chybu v počte kliknutí.

Dvojklik tiež kladie zvýšený nárok na motorickú schopnosť používateľov. Z tohto dôvodu bol v OS Windows od verzie Vista zvýšený maximálny časový interval, ktorý môže uplynúť medzi dvoma stačeniami dvojklikun. Fundamentálny problém však naďalej pretrváva, a to je samotná existencia dvojkliku.

Táto situácia je smutná, pretože WinExplorer je najrozšírenejšia aplikácia, ktorá používa tento spôsob interakcie a tým pádom udáva trendy pre ostatné. Preto sa bohužiaľ stretávame s dvojklikom aj v ostatných programoch, ktoré vo svojom UI obsahujú prvky vzdialene pripomínajúce adresárové uzly z WinExplorera.

Riešením tejto situácie je zrušiť dvojklik, ktorý je menej časovo a ergonomicky efektívny a nahradiť ho niečím iným. Takéto rozhranie, ktoré využíva len jedno kliknutie, sa nazýva single-click interface. Kliku sa priradí tá z funkcií systému, ktorá sa používa častejšie. Funkcia reagujúca na dvojklik bude volaná inou interakciou. Akou presne, to závisí od konkrétnej nahrádzanej funkcionality. V prípade APE je kliku priradená funkcia dvojkliku (otvorenie uzla) a pôvodná funkcia kliku (výber objektu) je vyvolaná kliknutím so stlačeným modifikátorom CTRL.

Ako vidíme, single-click interface je lepšou voľbou. WinExplorer dokonca ponúka možnosť zapnúť single-click interface. Problémom je však implementácia, ktorá zostáva značne nedotiahnutá už dlhé obdobie. Hlavným problémom je, že ak väčšina komponent

reaguje na jednoduché kliknutie, tak je nevyhnutné prispôsobiť ostatné interakcie užívateľa s aplikáciou tak, aby nedochádzalo k neželanému vyvolaniu funkcionality klikom. Jednoducho povedané: Akcie, ktoré sa doteraz vyvolávali kliknutím na komponenty pozične blízke komponentám reprezentujúcom adresárové uzly, sa musia vyvolávať iným spôsobom. Napríklad je potrebné upraviť prepínanie focusu medzi jednotlivými kontajnermi tak, aby nebolo nutné focus nastavovať explicitne. Takéto prepínanie je vykonávané klikom v single-click interface a zároveň aj v double-click interface. Zatiaľ čo v druhom prípade tento spôsob prepínania zvlášť nenarúša prácu s programom, pretože žiadna akcia meniaca obrazovku nie je viazaná na klik, tak v prípade single-click interface dochádza pri prepínaní klikom častokrát k prechodu na iný adresárový uzol.

5. Prezentovaná funkcionalita závislá na aktuálne zvolenom uzle.
   Ďalším problémom súčasných UI je prezentovanie funkcionality užívateľovi v závislosti od kontextu. Najprv uvediem problém na príklade Office a potom sa budem venovať WinExploreru.
   V prípade Office je vinníkom Ribbon, čo je vlastne panel funkcií rozdeľujúci funkcie do tabov. Každý tab obsahuje sémanticky príbuzné funkcie. Ribbon sa zvykne používať v programoch s množstvom funkcionality, ktorú je potrebné nahustiť na obrazovku. Jednou z čŕt tohto rozhrania je takzvané skrývanie funkcionality. Keďže taby obsahujú funkcionalitu vzťahujúcu sa na konkrétnu časť aplikácie, tak je možné skryť tento tab pred očami užívateľa ak pracuje s inou časťou aplikácie. Myšlienka za skrývaním je, že užívateľ nie je zahltený funkciami, ktoré nepotrebuje, pričom neprijde o možnosť použiť žiadnu relevantnú funkcie, keďže v danom momente sa aj tak nenachádza v správnom kontexte. Zatiaľ to znie lákavo, problém tohto rozhrania je však v strate prehľadu nad aplikáciou, čo ústi v pocit straty kontroly, pretože UI používateľovi ponúka funkcie o ktorých existencii nemal doteraz poňatie. Strata kontroly ústi v pocit frustrácie a frustrácia ústi v neochote používať program. Rovnako osvojovanie si aplikácie je problematické, pretože funkcie aplikácie nie sú podané v celku, ale závisle od kontextu. To sťažuje vytvorenie si prehľadu o funkcionalite, čo má za následok, že sa nepoužíva. Jedno z možných riešení by bolo napríklad zaviesť interaktívny tutoriál do aplikácie objasňujúci danú funkcionalitu alebo jednoducho navrátiť sa k starému štýlu prezentovania funkcionality – všetko naraz.

Čo sa týka WinExplorera, tak jeho nedostatok v tomto smere je našťastie oveľa menšieho rozsahu, najmä vďaka tomu, že sa nejedná o tak rozsiahlu aplikáciu a tým pádom počet skrytých funkcií nie je tak veľký. Skryté funkcie sa nachádzajú na panely funkcií a prítomnosť je kontextovo závislá od dvoch uzlov – zobrazovaného a aktuálne zvoleného uzla súborovej štruktúry. Napríklad, ak sa nachádzame v koreňovom uzle, máme k dispozícii funkciu "Uninstall or change program" na manažovanie programov v počítači (čo je vlastne abstraktnejší pohľad na prácu so súbormi). Ak navyše označíme uzol predstavujúci optickú mechaniku, tak pribudne funkcia "Eject" na vysúvanie podnosu pre CD.

Situácia sa v blízkej dobe nebude v tomto smere zlepšovať, skôr naopak. Uniknuté informácie týkajúce sa vývoja nového OS Windows 8 naznačujú, že Ribbon prvok bude zavedený do novej verzie WinExplorera. Skrývanie funkcionality tým bude ešte viac zvýšené. To znamená, že motivácia na vytvorenie nového súborového manažéra riadiaceho sa inou dizajnovou filozofiou je tým silnejšia.

6. Nezdokumentovanosť rozhrania.

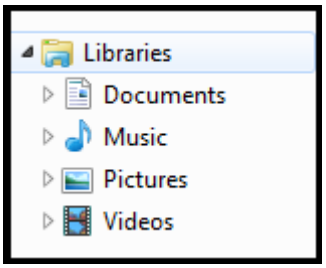
   Jedná sa o nasledovný problém: Užívateľ vsunie do PC s OS Windows USB kľúč so súbormi pochádzajúcimi z PC s OS Linux. Odrazu sa objaví dialóg vyzývajúci na opravu poškodených dát. Nič netušiaci užívateľ neuvedomujúci si podstatu problému vydá pokyn na "opravu" v dobrej viere, že tá skutočne "opraví" to čo má. Bohužiaľ, po tejto oprave sú súbory nečitateľné. Podstatou tohto problému je, že užívateľ bol konfrontovaný s funkcionalitou o ktorej nevedel, že vôbec existuje. Funkcionalita sa tvárila, že je nápomocná používateľovi a ten ju v dobrej viere použil.

   Problémom tu je, že popis tejto a mnohých ďalších funkcionalít nie je možné získať zo systému nápovedy. To má za následok, že užívateľ nemá prehľad o tom, čo aplikácia dokáže a čím ho môže prekvapiť, a takisto nevie ako použiť funkcie o ktorých existencii sa už dozvedel. U proprietárneho sofvéru (distribuovaného bez zdrojového kódu) je tento problém neodstrániteľný zásahom zvonka. V prípade Open Source softvéru sa dá tento problém v najhoršom prípade odstrániť štúdiom zdrojového kódu a napísaním užívateľskej príručky treťou stranou. Každopádne, malo by byť povinnosťou autorov prezentovať svoj program čo najviac otvorene, keďže oni sú jeho tvorcami a majú určitú víziu o jeho použití. Aj otvorenosť v tomto smere je súčasťou filozofie otvorenosti open-source sofvéru.

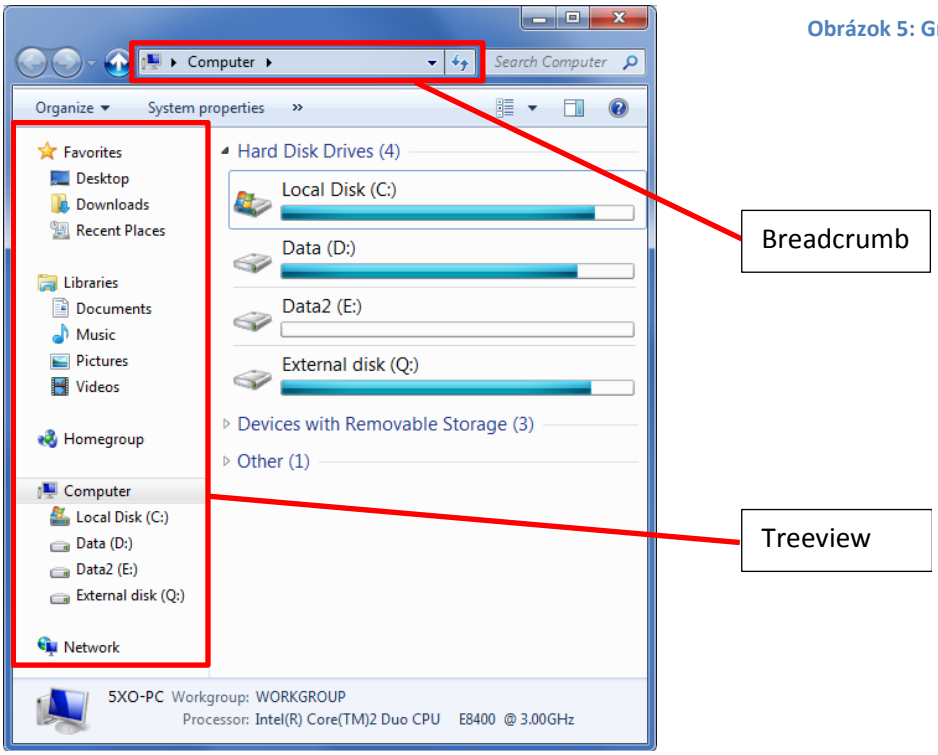
Náhrada WinExplorera bude teda vydaná pod open-source licenciou a navyše všetka funkcionalita bude stručne a prehľadne popísaná v referenčnom manuáli. Ten, na rozdiel od používateľského manuálu má menší rozsah a teda lepšiu čitateľnosť, pretože neoboznamuje užívateľov s funkcionalitou tutoriálovým spôsobom, len podáva prehľad o možnostiach programu.

7. Neprehľadný treeview.

Každý súborový manažér potrebuje mať vyriešené, ako podať užívateľovi prehľad o pozícii aktuálneho uzla v rámci súborovej štruktúry. Deje sa to väčšinou pomocou takzvaných hierarchických komponent, to znamená



Obrázok 5: Grafika stromu



Obrázok 4: Vyznačené hierarchické komponenty

komponent prezentujúcich hierarchické štruktúry, akou je napríklad aj súborová štruktúra. WinExplorer sa o to snaží dvoma spôsobmi, a to pomocou komponent treeview a breadcrumb. Samozrejme, táto redundancia je zbytočná, avšak o tomto dizajnovom nedostatku sa už pojednáva v rámci bodu 3 – Nadbytočnosť funkcií.

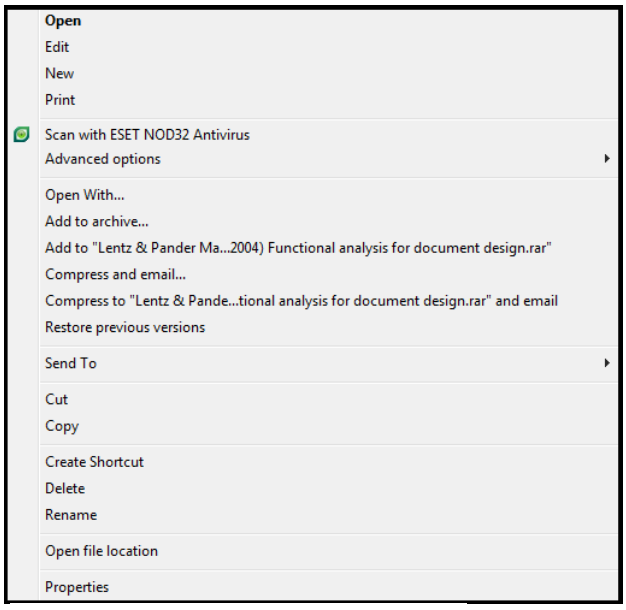
Teraz sa budeme venovať komponente, ktorú spomedzi týchto dvoch považujeme za menej vhodnú pre moderné grafické rozhranie a tou je treeview.

Zatiaľ čo v predchádzajúcej verzii WinExplorera bola táto komponenta pomerne

použiteľná, od verzie Vista došlo k zhoršeniu. Komponenta sa snaží prezentovať príliš veľa stromov, z ktorých ten najpodstatnejší strom Computer sa nachádza v spodnej časti obrazovky. Podruhé, orientovanie sa v hierarchii je v tomto strome znížené, keďže grafický dizajn prevážil nad funkcionalitou. Jedná sa o nevýraznú grafiku stromov (viď Obrázok 5: Grafika stromu). Okrem toho, že markery uzlov splývajú s farbou pozadia, tak celkové grafické prevedenie stromu nijako nespájaja synov so svojimi rodičmi okrem odsadenia. Navyše, markery sú zobrazované len onHover, teda len keď sa nad nimi nachádza myš. To znamená, že ani táto minimálna informácia ktorá bola naimplementovaná nie je prístupná užívateľovi letmým pohľadom, ale musí si ju explicitne vyžiadať tým, že prekryje časť stromu kurzorom myši, čo len ďalej znižuje jeho čitateľnosť. Poučením je, že funkcionalitu, ktorá má byť viditeľná kvôli uľahčeniu práce s aplikáciou je potrebné štandartne zobrazovať a nezahŕňať ju medzi dočasne zobrazované vizuálne efekty.

8. Zahltené kontextové ponuky.

   Ako bolo spomenuté, operácie nad vybraným uzlom systému sú zobrazované v paneli funkcií. Druhá možnosť, ktorá ponúka užívateľovi bezprostrednejšiu možnosť prístupu k operáciám nad uzlom súborovej štruktúry je vyvolať takzvané kontextové menu. Jeho výhodou je, že lepšie viaže operácie nad uzlom s uzlom samotným a to tým, že funkcie sú zobrazené po kliknutí pravým tlačidlom myši na uzol. Funkcia teda nie je separovaná od svojho operandu ako pri označení uzla a vyvolania odpovedajúcej funkcie v paneli funkcií. Ďalšou výhodou kontextového menu je, že na rozdiel od limitovaného miesta v paneli funkcií ponúka všetku relevantnú funkcionalitu. Nevýhodou je, že všetka funkcionalita sa nachádza na jednom mieste a pôsobí neprehľadne (viď Obrázok 6: Zahltené kontextové menu). Riešením je umožniť užívateľovi konfigurovať obsah tejto ponuky rovnako ako je mu umožnené konfigurovať obsah panela funkcií v ostatných programoch. Hlavnou



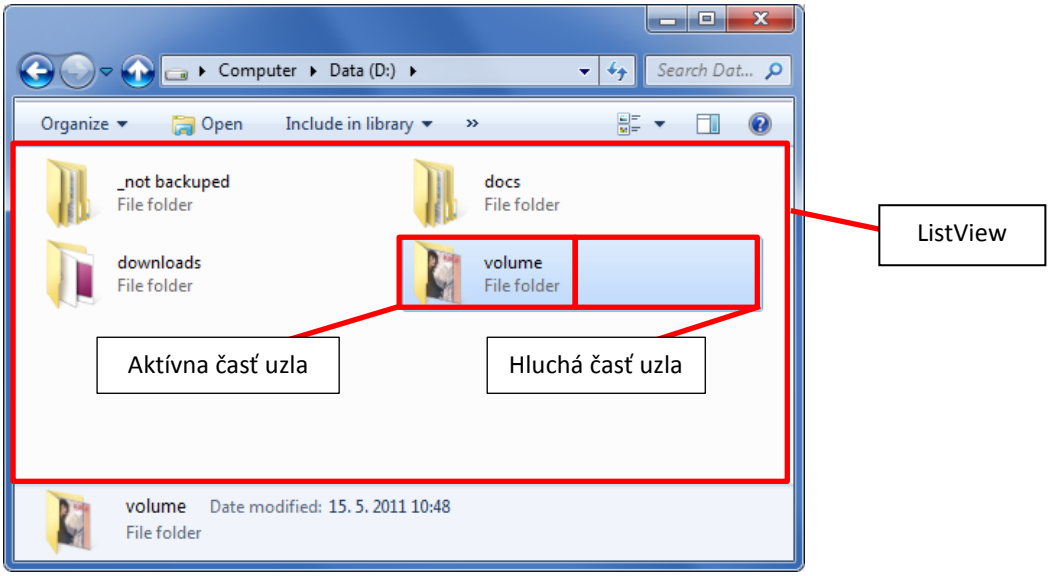
Obrázok 6: Zahltené kontextové menu

motiváciou tu je umožniť užívateľovi čo najviac optimalizovať svoju prácu s aplikáciou a využívať funkcie, ktoré sa v nej nachádzajú čo najefektívnejšie.

9. Kontextová závislosť štandartnej akcie počas drag&drop (ďalej len d&d).
   Štandartná akcia, ktorá sa vykoná pri d&d uzlov sa mení podľa vzťahu zdrojového a cieľového uzla. Ak sa oba uzly nachádzajú v rovnakom strome, tak je štandartnou akciou premiestňovanie. Ak sa nachádzajú v rozličných stromoch, tak je štandartnou akciou kopírovanie. Pripomíname, že koreňom stromu je uzol predstavujúci fyzické médium. O princípe za touto praktikou sa teda dá uvažovať aj nasledovne: Ak sú zdrojový a cieľový adresár na rovnakom disku, tak sa súbory medzi nimi premiestňujú. Ak sú na rôznych diskoch, tak sa kopírujú.
   Toto rozlišovanie síce znie logicky na papieri, pretože je za ním vidieť snahu programátorov odhadnúť najpravdepodobnejšu akciu, ktorú sa v danej chvíli snaží užívateľ vykonať a podľa toho nastaviť štandartné hodnoty. Problémom však je, že užívateľ pri práci s programom nerozmýšla v tak širokých súvislostiach ako je vzťah jednotlivých adresárov. Užívateľ má v úmysle buď niečo skopírovať alebo premiestniť. Keďže štandartná akcia sa mení podľa kontextu ktorý nie je od pohľadu zrejmý, výsledok štandartnej akcie je často iný než užívateľ chce. Toto ilustruje ďalší pozoruhodný jav vzťahujúci sa k užívateľským rozhraniam: Prílišná verzatilita programu a jeho prispôsobovanie sa rôznym situáciám spomaľujú viac používateľa ako jednoduchá zákonitosť, ktorú by sa naučil a bezmyšlienkovite sa ňou riadil ďalej. Ak by chcel použiť inú funkciu, naučil by sa inú zákonitosť a riadil by sa ňou. Skrátene: Dril je častokrát efektívnejší ako rozmýšlanie. Ak sa užívateľ cíti efektívne, znamená to že je šťastný. Šťastný užívateľ je cieľom programátora.
   Jedným zo spôsobov, ako zefektívniť užívateľa, je nastaviť štandartnú akciu vždy na kopírovanie. Kopírovanie je lepším kandidátom ako presúvanie pretože sa jedná o nedeštruktívnu činnosť. Pôvodný uzol zostane na svojom mieste, len sa vytvorí jeho kópia na inom mieste. Inou možnosťou by bolo nastaviť ako štandartnú akciu premiestňovanie. Bolo by to logické v tom prípade, ak by sa ukázalo že premiestňovanie je častejšie používaným príkazom. Takisto sa jedná o časovo jednoduchšiu činnosť zo systémového hľadiska. Premiestňovanie v rámci jedného súborového systému je otázkou prepisu administratívnych údajov, kdežto kopírovanie všeobecne a premiestňovanie medzi rôznymi súborovými systémami si vyžaduje naozajstné presunutie bajtov medzi

fyzickými médiami. Z týchto dôvodov by sa mohlo ukázať premiestňovanie ako lepšia voľba. Tretím spôsobom je spýtať sa užívateľa vždy po skončení tretej fázy d&d operácie, či si želá uzol skopírovať alebo premiestniť. Je tu ešte aj štvrtý spôsob: Ponúknuť užívateľovi lepšie vizuálne vodítka pre kontext v ktorom sa nachádza, napríklad pomocou farieb. Túto možnosť však rozvádzam v kapitole Špecifikácia programu.

10. Hluchá časť uzla počas zobrazovania v režimoch Tiles a Content.



Obrázok 7: Rozdelenie uzla

Pre celkovú analýzu sú samozrejme podstatné vyššie spomenuté dizajnové problémy a nie konkrétne chyby, avšak pre úplnosť analýzy spomenieme aspoň jeden problém z kategórie chýb. Podstatou tohto problému je, že pri vyvolaní kontextového menu nad prvkom súborovej štruktúry v režime zobrazovania Tiles alebo Content nie je vždy zobrazené relevantné kontextové menu. Namiesto toho sa niekedy zobrazuje všeobecné kontextové menu obsahujúce príkazy pre komponentu listview. Zrejmou príčinou je, že klik na uzol je registrovaný len aktívnou plochou uzla (tá ktorá obsahuje ďalšie komponenty). Klik na hluchú plochu uzla je odchytený až nadradenou komponentou v logickom strome komponent, a tou je komponenta listview (pre detaily o logických stromoch viď Knižnica WPF). Je tak zrejme z toho dôvodu, že samotný uzol pozostáva z viacerých komponent a len jedna z nich reaguje na klik. Následne, keď klik minie túto konkrétnu komponentu tak je poslaný vyššie v logickom strome a odchytí ho až listview.

11. Neprítomnosť tabov

Väčšina moderných programov narábajúcich s užívateľskými dokumentmi (súbory, stránky) má implementované takzvané tab rozhranie. Princípom tohto rozhrania je, že

umožňuje zobrazovať viacero dokumentov v jednom okne aplikácie. Dokumenty však nie sú zobrazované všetky naraz, ale užívateľovi je umožnené sa medzi nimi prepínať. Takéto rozhranie sa začalo masívne presadzovať vo webových prehliadačoch a doteraz je pre ne špecifické. Výhodou tohto rozhrania je, že sprehľadňuje prácu pri veľkom množstve otvorených dokumentov. Jednak sa sprehľadní zoznam otvorených okien, ktorý poskytuje OS a za druhé sa umožní väčšia flexibilita užívateľovi, ktorý môže sústreďovať súvisiace dokumenty do rôznych okien tej istej aplikácie a tak oddeliť jednotlivé úlohy, ktoré rieši pri práci s PC.

12. Úvodnú obrazovku nie je možné konfigurovať.

    Nie je možné zvoliť uzol súborovej štruktúry, ktorého obsah sa zobrazí pri štarte aplikácie. Táto črta je dôležitá kvôli tomu, že zvyšuje použiteľnosť rozhrania a pritom ho nijako nekomplikuje, podobne ako možnosť konfigurovať obsah kontextových ponúk. Nemožnosť nastaviť úvodný uzol je analogická problému, ktorý by nastal, keby webové prehliadače neumožňovali nastaviť domovskú stránku. Nepopierateľne, v dnešnej dobe je nutné zohladniť pri návrhu rozhrania aplikácie webové prehliadače a UI praktiky ktoré používajú, pretože sú to programy s ktorými dnešný typický užívateľ trávi väčšinu svojho času. Ako sa hovorí: “Užívateľ trávi väčšinu času inde než v programe, ktorý sa práve programátor snaží vytvoriť”. A sú to práve webové prehliadače, ktoré udávajú trendy v používateľskom rozhraní.

    Všeobecne sa dá povedať, že ak sa dá situácia rovnako dobre vyriešiť už zaužívanou praktikou, tak by sa malo použiť zaužívané riešenie. Ak program navodzuje analogickú situáciu, tak by mal užívateľovi poskytnúť rovnaké možnosti používania programu. Jedná sa o rovnaký problém nezdokumentovanosti rozhrania. V skutočnosti je možné úvodnú obrazovku nastaviť, avšak nastavenie tejto funkcionality je nutné vykonať pomocou textových parametrov ktoré sú dodávané WinExploreru pri spustení. Táto skutočnosť sama o sebe znemožňuje využiť túto funkcionalitu drvivej väčšine populácie. Veľká väčšina nepozná prácu s programom inak než cez grafické UI programu, takže užívatelia si nie sú vedomí iných možností. Zostáva v nich len pocit frustrácie, pretože sa cítia byť aplikáciou spomalovaný a nevedia ako si pomôcť. Naviac popis textových parametrov nie je dodávaný s aplikáciou, takže užívateľ je odkázaný si ho dohľadať inde. Ďalším argumentom proti je malá oboznámenosť používateľov s textovým rozhraním aplikácií, z čoho vyplýva neochota učiť sa novému spôsobu práce. To tvorí ďalšiu bariéru

zabraňujúcu využiť túto funkcionalitu. Posledným argumentom proti je neustálená konvencia textových argumentov, z ktorých niektoré sú kryptickej povahy, zrejme z toho dôvodu, že prílišne odhaľujú vnútro fungovania aplikácie. Napríklad argument hovoriaci aplikácii, aby sa spustila vo fiktívnom uzle obsahujúcom fyzické disky PC (My Computer), je nasledovný reťazec: “20D04FE0-3AEA-1069-A2D8-08002B30309D”. Tento reťazec je tu uvedený z toho dôvodu, že núti WinExplorer spustiť sa v uzle “My Computer”, ktorý bol zaužívaný dlhé roky vo verzii z OS Windows XP, čo je ako si vysvetlíme neskoršie rozumná voľba pre úvodnú obrazovku súborového manažéra. Našťastie, väčšina reťazcov nevystavuje vnútorný chod programu týmto spôsobom. Napríklad reťazec pre spustenie v uzle predstavujúcom disk C je “C:”.

Keď sme si predstavili limitovaný spôsob konfigurácie úvodnej obrazovky, mali by sme sa zamyslieť ako by sa dal vylepšiť. Úvodná obrazovka slúži na rýchly presun do uzlov, s ktorými chce pracovať. Chceli by sme poskytnúť užívateľovi spôsob, ktorým by sa dostal do uzlov súborovej štruktúry, ktoré často navštevuje. Prvým vylepšením je možnosť pridávať odkazy na adresáre do zoznamu. Užívateľ by si tak mohol vybrať, ktoré adresáre chce mať priamo dosiahnuteľné z úvodnej obrazovky. Druhým vylepšením je zobrazovať zoznam naposledy navštívených miest. Tento by slúžil na zobrazenie uzlov, ktoré užívateľ explicitne nepridal na úvodnú obrazovku, ale existuje veľká pravdepodobnosť, že by s nimi v blízkej dobe mohol pracovať.

13. Nemožnosť konfigurovať klávesové skratky.

    Na zrýchlenie práce s programom pre pokročilých užívateľov je potrebné pridať možnosť nastavovať kombinácie kláves, ktoré invokujú jednotlivé funkcie. Každý užívateľ má iný spôsob práce s klávesnicou a tiež je navyknutý na iné aplikácie a ich skratkové konvencie. Z tohto dôvodu je nielen potrebné pridať klávesovú skratku pre každú funkciu, ale aj umožniť jej zmenu.

14. Nemožnosť exportovať nastavenia rozhrania.

    Táto funkcionalita by umožnila zachovať prostredie programu v prípade preinštalovania OS alebo prechodu na iný PC. Výhodou je, že do UI stačí pridať dve tlačidlá na miesto druhej triedy (na ich stlačenie sú potrebné 2 kliky myšou) a teda nezahlcuje používateľské prostredie. Samotná práca so súbormi tak nie je sabotovaná. Nastavení, ktoré je možné exportovať, nie je veľa v aplikácii typu súborový manažér. Jedná sa najmä o klávesové skratky a režimy zobrazenia pre jednotlivé adresáre. Dobrou vlastnosťou

exportu je, že nestojí skoro žiadnu obrazovkovú plochu a podporuje užívateľov v efektívnej práci s programom. To robí export vhodnou črtou na implementáciu do programu, pretože má dobrý pomer medzi úžitkom a zvýšenou komplexnosťou používania. Pri implementovaní novej funkcionality sa chceme predovšetkým vyhnúť fenoménu feature creep, pri ktorom sa UI stáva neprehľadným pre veľké množstvo nadbytočných funkcií.

Snažili sme sa analyzovať problém zoširoka a z priestorových dôvodov sa vyhnúť rozoberaniu detailov každej funkcionality programu. Väčšina funkcionality je totiž redundantná alebo je v nej priestor na zlepšenie.

V zásade sme mali možnosť pozorovať tri kategórie problémov:

a. Dôsledok zlého dizajnu (body 1-9).
   Problémy z tejto kategórie vznikli implementáciou zlej používateľskej špecifikácie. Zlá špecifikácia pramení z neznalosti častých úskalí vývoja UI a jej dôsledkami sú zahltenosť rozhrania, neefektívnosť používania a nezdokumentovanosť rozhrania. Na tieto nedostatky sa však narazí až pri používateľskom testovaní produktu. Potom je potrebné zmeniť používateľskú špecifikáciu, čo je často problémom kvôli neochote meniť základný dizajnový dokument neskoro vo vývojom cykle produktu. Často však tieto nedostatky ostávajú roky nepovšimnuté v prípadoch, keď sa nevenuje pozornosť použiteľnosti programu.
b. Chyba (bod 10).
   Problémy z tejto kategórie vznikli zlou implementáciou používateľskej špecifikácie. Ich oprava je jednoduchšia, pretože z špecifikácie je jasné, ako by sa mala daná funkcionalita správať. Nie je tu potrebné opravovať zlý dizajn alebo navrhnúť nové riešenia.
c. Chýbajúca funkcionalita (body 11-14).
   Problém tohto typu vznikol neprítomnosťou kľúčovej funkcionality v používateľskej špecifikácii. Táto funkcionalita by podstatným spôsobom zlepšila použiteľnosť aplikácie bez zbytočného komplikovania jej rozhrania.

1.2. Ostatné programy

Teraz, keď sme popísali hlavné nedostatky dominantnej aplikácie na poli súborových manažérov, skúsme v krátkosti analyzovať nádejných konkurentov z radov programov, ktoré sú ponúkané zadarmo. Každý z nich má svoje dobré a zlé stránky, rovnako ako WinExplorer. Nebudeme už podrobne rozpisovať jednotlivé body analýzy. Za prvé, analyzované črty sú veľmi podobné ako pri WinExploreri, keďže väčšina aplikácií implementuje rovnakú funkcionatlitu rovnakým spôsobom. Za druhé, dizajnové princípy za jednotlivými funkcionalitami sú už známe z analýzy WinExplorera – ide vo všeobecnosti o použiteľnosť a efektívnosť.
Ak analyzované črty nevyhovujú týmto princípom, sú zaradené do kategórie nedostatkov, inak sú zaradené do kategórie predností daného programu.

CubicExplorer

Plusy: hmatateľné a prehľadné rozhranie, taby, prispôsobiteľnosť úvodnej obrazovky a panelov funkcií, breadcrumb navigácia.
Mínusy: ťažkopádnosť UI (pomalé a občas chybné prekreslovanie komponent, pomalý štart), veľa medzier v efektivite používania (práca s tabmi, vizuálny šum), chýba undo funkcionalita, single-click interface, onhover komponenty, nastavovanie klávesových skratiek, prispôsobovanie kontextových menu.

FreeCommander, Double Commander, Unreal Commander

Jedná sa o freeware klony obľúbeného spoplatneného softvéru Total Commander, zoradené sú v poradí ich kvality.
FreeCommander má obzvlášť zahltené rozhranie, prílišne miniaturizované ikony, žiaden breadcrumb, undo a ani taby. Double Commander okliešťuje funkcionalitu ešte väčšmi, pričom obsahuje experimentálnu funkcionalitu ako Show occupied space, ktorá spôsobuje zamrznutie PC. Unreal Commander obsahuje UI pozostávajúce z predimenzovaných rozpixelizovaných čiernobielych ikon, miniatúrnych nečitateľných tlačidiel s prečnievajúcim obsahom a nečitateľného textu. Ani jeden z týchto programov už našťastie nie je minimálne 2 roky vo vývoji.

Q-Dir

Aplikácia prichádza s novým prístupom k prezentácii hlavného panela aplikácie. Namiesto jedného panela ako je to v tradičných aplikáciách (WinExplorer) alebo dvoch panelov ako v aplikáciách typu X-Commander (FreeCommander, Total Commander), Q-Dir ponúka možnosť zobraziť 1-4 panely v rôznych konfiguráciách. Bohužiaľ, táto schopnosť prispôsobiteľnosti je jediná, ktorou aplikácia oplýva. Čo však nie je na škodu, keby umožňovala vykonávať základné operácie efektívne. Aplikácia však síce obsahuje veľa možností prispôsobiteľnosti panelov, komponent a spôsobov ako prísť k cieľu, bohužiaľ samotné jadro súborového manažéra (KPPV) neplní svoje poslanie efektívne. Rozhranie je miniaturizované, single-click interface reaguje s badateľným oneskorením, texty sú miestami zle preložené z nemčiny, nefunguje funkcia Find. Za dobrú zmienku však stojí iniciatíva autora ponúknuť užívateľovi zmysluplné prispôsobovanie komponent (šírka tabov), pokus o onhover prepínanie (v tomto prípade sa však focus prenáša posunutím kolečka myši nad komponentou, čo je však stále zlepšením oproti nutnosti kliknúť do oblasti komponenty) alebo netradičné funkcie, ako je napríklad export zoznamu uzlov súborovej štruktúry, ktoré obsahuje aktuálny priečinok, do textového súboru. Bohužiaľ, akokoľvek užitočná je táto bonusová funkcionalita, je pochovaná pod záplavou ostatnej nepotrebnej funkcionality. Program sa stáva obeťou fenoménu feature creep, pri ktorom sa do aplikácie bezhlavo pridáva funkcionalita, z ktorej je väčšina škodlivá pre použiteľnosť programu.

STDUExplorer

Klon WinExplorera, ktorý obsahuje namiesto panela funkcií pestrofarebný Ribbon známy z prostredia Office. Našťastie v tomto prípade neobsahuje žiadnu skrytú funkcionalitu. Ponuka funkcií je malá, neobsahuje väčšinu vecí, ktoré sme požadovali v predchádzajúcej analýze. Rozhranie je však vďaka tomu veľmi priateľské a použiteľné, k čomu prispieva aj jednoduchá pastelová grafika a separabilné komponenty rozhrania. Aplikácia je totiž rozdelená na tri časti: TreeView, ListView a Ribbon. Toto delenie znamená, že užívateľ je schopný si rýchlo vytvoriť myšlienkový model o aplikácii a cítiť sa v nej familiárne. Ak si dokáže používateľ takto rýchlo osvojiť aplikáciu, znamená to že vývojár dosiahol jeden z cieľov dizajnu – použiteľnosť.

Explorer++

Dá sa prirovnať k WinExploreru. Je v podstate jeho priamou kópiou až na to, že nepoužíva novú grafickú knižnicu WPF a prichádza o niektoré výhody, ktoré budeme preberať v nasledujúcich kapitolách. To znamená, že na neho platia všetky body analýzy WinExplorera. Explorer++ pridáva síce možnosť vytvárať Bookmarky a exportovať obsah uzlu do textového súboru, avšak to nedokáže odčiniť už spomínané výčitky.

xplorer^2

Ako vyplýva z názvu, jedná sa o ďalšiu kópiu WinExplorera. Obsahuje tiež rovnaký nedostatok ako Q-Dir, a to single-click interface s oneskorením reakcie na kliknutie. Táto črta je pravdepodobne pozostatkom z čias WinExplorera z OS Windows, ktorý mal podobný problém. To napovedá, že sa pravdepodobne jedná o technický problém, keď postihnuté programy používajú tú istú verziu grafickej komponenty ListView, ktorá obsahuje neželané správanie. Podarilo sa zistiť, že takýto ListView po kliknutí na svoju položku vyčkáva presne časový interval, ktorý je nastavený v OS, a jedná sa o dobu ktorá môže uplynúť medzi dvoma klikmi aby bola zaregistrovaná udalosť double-click. Jednoduché zníženie tejto systémovej hodnoty teda umožňuje zmierniť oneskorenie a priblížiť sa k ozajstnému single-click interface. Bohužiaľ, hodnota sa v systéme nedá nastaviť na 0, musí mať kladnú hodnotu väčšiu ako určitý prah ľudskej rýchlosti, pretože OS predpokladá, že niektoré kritické úkony (ako napríklad nastavovanie tejto systémovej hodnoty) sa dajú spraviť len použitím dvojkliku a preto sa snaží zabrániť stavu, kedy si užívateľ sám dá šach a bude musieť preinštalovať OS, aby sa vrátil do pôvodného stavu.
Čo sa týka pozitívnych stránok, tak xPlorer^2 obsahuje najväčšie množstvo funkcionality z tu spomínaných troch klonov WinExplorera. Navyše táto funkcionalita nestojí užívateľovi v ceste, ale nachádza sa prehľadne usporiadaná v hlavnom menu aplikácie. Bohužiaľ, keďže sa jedná o klon, obsahuje tiež rovnaké neduhy ako WinExplorer.

1.3. Zhrnutie

Celá táto kapitola by sa dala pre prehľadnosť zhrnúť do tohto odstavca:
Rozhranie súčasných produktov je neprehľadné so záplavou zbytočnej alebo málo používanej funkcionality. Dôležité funkcie sa stratia medzi ostatnými.

Tiež sme ukázali, že ani jedna aplikácia neimplementuje všetky nutné črty moderného súborového manažéra, aj keď každá črta sa nachádza aspoň v jednej aplikácii. Z tohto plynie potreba navrhnúť aplikáciu, ktorá implementuje všetky tieto črty. Než príde takáto aplikácia na trh, čitateľ by si mal vyskúšať spomínané programy a oboznámiť sa s dizajnovými problémami, ktorým sa venovala táto kapitola. Keďže sú všetky tieto programy freeware, tak ich čitateľ nájde pribalené k tejto práci. Môže si tak utvoriť svoj vlastný názor a vybrať zo spomínaných programov ten, ktorý najviac vyhovuje jeho potrebám a vkusu.

Posledným nedostatkom, ktorému sme sa ešte nevenovali, je bitmapová podstata UI väčšiny spomenutých aplikácií. Jedinú výnimku tvorí WinExplorer z dôvodu použitia grafickej knižnice WPF o ktorej bude ešte reč v neskorších kapitolách. Všetky ostatné aplikácie využívajú staré grafické knižnice, ktoré používajú absolútne zadávanie rozmerov UI v pixeloch a bitmapové obrázky. Prejavuje sa to tak, že pri škálovaní rozhrania sa rozostruje text a aj grafika aplikácie. Z tohto dôvodu budeme používať grafickú knižnicu WPF, ktorá je vybudovaná na vektorovom princípe, teda používa relatívne dĺžkové jednotky a vektorové obrázky. To umožňuje bezproblémové škálovanie UI bez straty ostrosti. Zvýši sa tak použiteľnosť pre používateľov majúcich vysoké DPI monitora, pre zrakovo postihnutých a pre ľudí, ktorý preferujú väčšie rozmery komponent UI.

Všetky tieto neduhy sa bude snažiť odstrániť program APE, ktorého popis UI sa nachádza v nasledujúcej kapitole.

2. Špecifikácia

Táto kapitola sa zaoberá špecifikáciou aplikácie APE – Another Penultimate Explorer. APE je súborový manažér navrhnutý tak, aby sa vyhýbal priepastiam zlého dizajnu, ktorého množstvo príkladov bolo uvedených v predchádzajúcej kapitole.

Obrázok 8: Posun k minimalizmu

Ako už bolo naznačené, návrh aplikácie sa bude riadiť filozofiou minimalizmu. Čo však tento minimalizmus znamená?
Napríklad by sa to dalo vystihnúť dizajnovými pravidlami:
“V jednoduchosti je sila.”
“Ponúknuť užívateľovi všetky funkcie, ktoré potrebuje, ale ani o jednu viac.”
“Program má vedieť plniť práve jednu úlohu a má ju vedieť plniť efektívne.”
Princípy minimalizmu – odstránenie prebytočných entít a zvýšenie prehľadnosti – ilustruje napríklad posledná zmena ikony prehliadača Google Chrome. Google prešiel od 3D modelu s farebnými prechodmi a odleskami k jednoduchej 2D kresbe s 5 farbami. Čitateľ by mal prestať na chvílu čítať a skúsiť vymenovať päť farieb, ktoré sa nachádzajú na obrázku. Piatou farbou je biela farba, ktorej účelom je oddeliť “jadro” od “obalu” ikony. Voľné miesto je totiž rovnako dôležité ako samotná obrazová informácia, pretože umoňuje užívateľovi zorientovať sa a zvyšuje prehľadnosť [Whi02].

Cieľom tejto kapitoly je vytvoriť systémovú špecifikáciu. Je to dizajnový dokument, ktorý je vytvorený pred začiatkom implementácie programu a voči ktorému sa aplikácia na konci vývojového cyklu testuje, či správne rieši zadanie. Predstavuje rozhrania aplikácie smerom k užívateľom. Toto rozhranie by malo riešiť problémy nastolené v užívateľskej špecifikácii, čo je dokument vytvorený počas analýzy problematiky, pre ktorej riešenie užívateľ potrebuje daný softvér. Niekedy sa užívateľská špecifikácia označuje ako analýza požiadaviek. Dôvod, prečo robiť systémovú špecifikáciu je ten, že je potrebné vytvoriť dokument, ktorý presne popisuje konkrétnu implementáciu riešenia problému, nie len potreby a ciele užívateľov.
Vývoj sofvéru všeobecne neobnáša len písanie programového kódu, ale aj špecifikáciu problému na viacerých úrovniach a testovanie softvéru, či vyhovuje všetkým týmto špecifikáciám. Táto metodika sa nazýva V-Model [Wit11] (viď Obrázok 9: V-Model).
Užívateľská špecifikácia sa nachádza na začiatku V-Modelu, teda k jej vytvoreniu dochádza

chronologicky na začiatku vývojového cyklu. V ideálnom prípade by jej testovanie mal vykonať užívateľ na konci vývojového cyklu, predtým, než sa softvér dostane do bežného používania. Typicky je však toto testovanie vykonávané programátormi, ktorých predstava o použiteľnosti programu sa nezlučuje s predstavou užívateľa. Bohužiaľ, aj keď užívateľ otestuje aplikáciu voči špecifikácii na konci vývojového cyklu, je už neskoro vykonať nápravu bez nutnosti investovať veľa časových zdrojov. Z tohto dôvodu je dobré, ak sa testovanie užívateľom vykonáva vždy, keď sa podarí naimplementovať niektorú samostojnú funkcionalitu, teda funkcionalitu, ktorá dokáže fungovať bez iných externých komponent. V prípade súborového manažéra môže ísť napríklad o implementáciu zobrazovania obsahov priečinkov. Takéto inkrementálne testovanie umožňuje meniť dizajnovú špecifikáciu podľa potreby, čo odstraňuje jeden z dôvodov zlej použiteľnosti výsledného programu – prílišnú nemennosť dizajnového dokumentu.

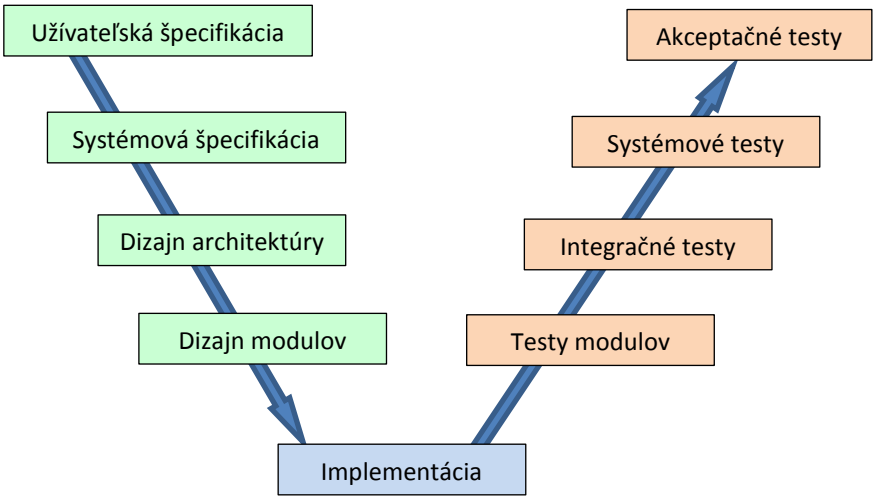
Čo sa týka ostatných dizajnových dokumentov, poskytujeme ich popis len v krátkosti, pretože sa jedná o technické dokumenty, ktoré sú tvorené až počas programovania aplikácie.

1. Dizajn architektúry

   Popisuje technickú stránku aplikácie z vysokoúrovňového hľadiska. Preberá použité technológie, popisuje nezávisle bežiace komponenty aplikácie (napríklad serverová a klientská časť) a častokrát aj menšie bloky aplikácie (triedy v prípade objektovo-orientovaného programovania) a interakciu medzi nimi.

2. Dizajn modulov (unitov)

   Popisuje metódy jednotlivých blokov a definuje testy, ktoré musia splniť.



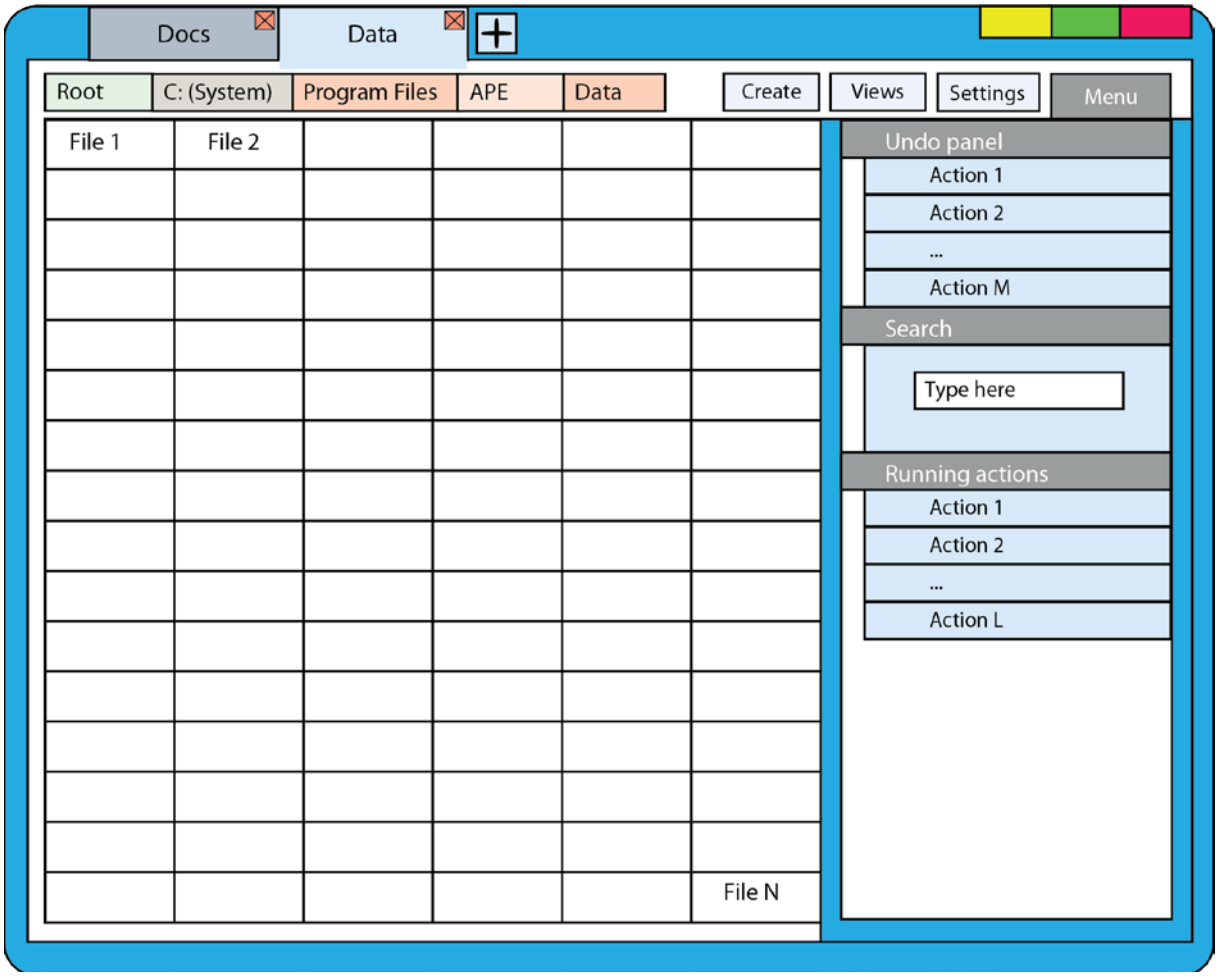
Obrázok 9: V-Model

2.1. Prehľad komponent programu

V tejto podkapitole konečne zúročíme analytické poznatky z predchádzajúcej kapitoly, čo nám umožní napísať podrobný prehľad funkcionality programu APE v súlade s princípmi dobrého dizajnu. Po sformulovaní funkcionality musíme vytvoriť náčrt rozhrania aplikácie. Poskytne nám to mentálny prehľad o rozvrhnutí jednotlivých funkcionalít a grafickom štýle aplikácie. Toto využijeme ako referenciu počas vývoja aplikácie, jednak ako vizuálne vodítko pri rozmiestňovaní funkcionality a za druhé ako kontrolu, aby sme nevybočili z dodržovania minimalistického grafického štýlu.

Náčrt však nie je nemenný, ale prispôsobuje sa novoobjaveným požiadavkám na funkcionalitu. Len počas samotnej špecifikácie prešiel niekoľkými obmenami, je len prirodzené, že k obmenám prijde aj počas implementácie programu.

Aj keď je náčrt vytvorený až po spísaní požiadaviek na funkcionalitu, uvádzame ho predbežne, aby si čitateľ spravil predstavu o výzore UI.



Obrázok 10: Náčrt rozhrania aplikácie

Panel s tabmi

Nachádza sa v najvrchnejšej časti okna, táto prominentná pozícia demonštruje, že použitie tabov je jedným z hlavných znakov moderných UI. Panel zaberá miesto, ktoré je tradične okupované záhlavím okna.

Prepínanie tabov funguje systémom onHover. Stačí myšou ukázať na tab a obsah okna sa prepne. Je to jediná funkčnosť v aplikácii, ktorá používa systém onHover. Takýto spôsob volania funkcionality je nový a je stále považovaný za experimentálny [Cli05]. Je ho nutné používať uvážene a existuje niekoľko odporúčaní, ktoré je dobré mať na pamäti.

1. Je vhodný na často používanú funkcionalitu.
   Medzi jeho pozitíva patrí zrýchlenie práce s UI. Vďaka jeho negatívam je ho však nutné používať len na určitú podmnožinu funkcionality, v tom prípade je často používaná funkcionalita vhodným kandidátom.
2. Dá sa použiť len na nedeštruktívnu funkcionalitu.
   Vďaka sklonu k neželanej aktivácii je vhodný len na volanie funkcií, ktoré nemenia stav údajov nad ktorými pracuje aplikácia. Napríklad, mazanie súborov pomocou systému onHover by mohla viesť k množstvu omylom zmazaných súborov.
3. Funkcionalita by sa mala nachádzať na hraniciach UI.
   Z dôvodu minimalizácie neželaných aktivácií je dobré, ak sa funkcionalita nachádza pri okraji okna aplikácie. Takto odbudnú tie neželané aktivácie, ku ktorým došlo keď užívateľ putoval myšou k inej funkcionalite.
4. Je potrebné umožniť vypnutie onHover systému kvôli nasadeniu aplikácie v blind-friendly prostredí. OnHover systém je založený čisto na pozícii myši vo vizuálnom priestore, preto bez vizuálnej spätnej väzby dochádza pri manipulácii s myšou k neželanej aktivácii funkcií.

Každý tab má zatvárací krížik a na konci panela je tlačidlo na vytvorenie nového tabu. V čase písania tejto špecifikácie však ešte nie je známe, či bude tieto ovládacie prvky možné vypnúť v Settings paneli, alebo nebudú vôbec prítomné v rozhraní. Závisí to od spätnej väzby používateľov, ak by si väčšina dokázala privyknúť na používanie klávesových skratiek na otváranie a zatváranie záložiek, mohlo by dôjsť k vynechaniu týchto rušivých prvkov. Cieľom

je upriamiť pozornosť používateľa na obsah - súbory a adresáre, čo sa dá dosiahnuť odstránením nepotrebných prvkov UI. Prednastavená klávesová skratka pre otvorenie je *Ctrl+T* a pre zatvorenie je *Ctrl+W*.

Pri veľkom počte otvorených záložiek hrozí situácia, že sa nebudú môcť všetky zobraziť v jednom riadku okna. Hovoríme, že nastane takzvané pretečenie. V takomto prípade pribudnú ovládacie prvky na horizontálny posun zoznamu tabov. Zoznam tabov je tiež možné posúvať kolečkom myši, myš však musí byť onHover na paneli. Posúvanie tabov prebieha plynule v krokoch po jednej záložke.

Taby podporujú drag&drop, takže súbory sa dajú kopírovať ťahaním myšou medzi adresármi otvorenými v rôznych záložkách.

Tab je možné pomocou d&d premiestniť v rámci panela tabov, premiestniť do panela tabov iného okna alebo pustiť ho mimo akéhokoľvek panela a vytvoriť tak nové okno.

Vďaka náhrade záhlavia okna panelom tabov by v aplikácii neexistovala plocha, pomocou ktorej by bolo možné presúvať okno aplikácie. Tento potencionálny nedostatok rieši samotný panel tým, že poskytuje túto funkcionalitu pomocou prázdneho miesta.

Breadcrumb navigátor

Je to jednoduchá komponenta určená na orientáciu užívateľa v súborovej štruktúre. Je implementovaný ako zoznam tlačidiel usporiadaných horizontálne za sebou, každé z nich reprezentuje jeden adresár nachádzajúci sa na ceste od koreňa k aktuálnemu adresáru. Tlačidlá sú zoradené podľa hĺbky adresárov, ktoré predstavujú. Kliknutie na tlačidlo spôsobí zmenu aktuálneho adresára hlavného panela na ten, ktorý je reprezentovaný daným tlačidlom.

Tlačidlá sú farebne označené podľa typu súborového uzla. Root adresár, disky a adresáre majú navzájom odlišné farby kvôli prehľadnejšej identifikácii.

Pri pretečení tlačidiel sa nepostupuje ako v prípade panela tabov, kde sa táto situácia riešila horizontálnymi posuvníkmi. Táto metóda nie je vhodná pre breadcrumb, pretože znižuje bezprostrednosť orientácie v súborovej štruktúre. Pri hľadaní riešenia je vhodný princíp Occamovej britvy. Preto sa budeme snažiť nájsť najjednoduchšie fungujúce riešenie, ktoré sa

dá vysvetliť pár vetami. Pri pretečení tlačidiel budú odstraňované tlačidlá od začiatku súborovej cesty až pokým sa zvyšok nezmestí na obrazovku. Odstránené tlačidlá budú prístupné cez roletové menu, ktoré sa rozprestrie onHover.

Breadcrumb spolupracuje s drag&drop funkciou. Pri pustení súborového uzla nad tlačidlom sa tento prenesie/skopíruje do zodpovedajúceho adresára.

Toolbar

Vo východzom stave obsahuje len zopár základných tlačidiel. Sú to Create, View, Settings a Menu. V prípade potreby je možné tlačidlá pridávať a odoberať.

Create je drop-down menu obsahujúce ponuku súborov, ktoré je možné vytvoriť. Tento zoznam je prispôsobovateľný v Setting dialógu.

View je drop-down menu obsahujúce zobrazovacie režimy, v ktorých je možné vykreslovať súborovú štruktúru. Aplikácia poskytuje režimy List, Details, Thumbnails a Tiles(prednastavená voľba). Ak by sa vyskytli argumenty pre iné zobrazovacie režimy, po diskusii ich je možné doimplementovať.
Zmyslom režimu List je ponúknuť užívateľovi spôsob na zobrazenie čo najväčšieho počtu uzlov na obrazovke.
Zmyslom režimu Details je zobrazenie čo najväčšieho počtu informácií o jednotlivých uzloch.
Zmyslom režimu Tiles je poskytnúť najpohodlnejší spôsob práce s uzlami vďaka veľkým aktívnym plochám.
Zmyslom režimu Thumbnails je poskytnúť náhľad na obsah obrazových typov súborov. Aktívnu plochu v tomto prípade tvorí z väčšej časti miniatúra obsahu súboru.
Všetky tieto režimy zobrazujú uzly ako horizontálne pretiahnuté obdĺžniky. Je tomu tak preto, že západné civilizácie používajú spôsob písania textu zľava doprava a teda je žiadúce poskytnúť v horizontálnom smere viac priestoru, aby dochádzalo k menej zalomeniam riadkov. Pre tento dôvod sú tiež ikonové režimy, na aké sme zvyknutí napríklad z windows desktop, nevhodné pre efektívnu prácu s textovými elementmi.

Menu je prepínacie tlačidlo, ktoré zapína bočný panel.

Settings dialóg

Okno, ktoré združuje všetku konfiguráciu aplikácie pod jednu strechu. Obsahuje niekoľko sekcií, ktoré sú stručne vysvetlené v nasledujúcom zozname.

- General settings – všeobecné nastavenia ktoré nepatria pod žiadny iný okruh.
    - Startup folder – nastavovanie uzla súborovej štruktúry, v ktorom sa program spustí.
    - Interface export – uloženie klávesových skratiek, nastavení režimov zobrazovania a obsahu kontextových menu do textového súboru.
- Create menu Editor – pridávanie a odoberanie položiek z Create menu.
- Right click menu Editor – pridávanie a odoberanie položiek z kontextového menu.
- Shortcut Editor – nastavovanie klávesových skratiek. Všetky *klávesové skratky*, ktoré uvádzame v tejto kapitole a sú označené kurzívou, je možné predefinovať.

Bočný panel

Panel pokrývajúci väčšinu doplnkovej funkcionality. Nachádza sa v pravej časti okna. Skladá sa z niekoľkých panelov, ktoré predstavujú rôzne okruhy funkcionality:

- Undo panel – obsahuje chronologicky usporiadaný zoznam akcií vykonaných nad súborovým systémom. Tieto akcie je možné vrátiť naspäť (undo funkcionalita), alebo ich znova vykonať ak už boli predtým vrátene naspäť (redo funkcionalita). V prípade, že používateľ vráti akcie a potom vykoná novú akciu, tak sa vrátené akcie stratia zo zoznamu a sú nahradené touto novou akciou. Tým pádom ich už nie je možné znovu vykonať. Toto riešenie je zvolené kvôli prehľadnosti undo/redo funkcionality, pretože v opačnom prípade by bolo nutné do zoznamu akcií pridávať aj Undo operácie, ktoré sú však komplexnejšie, pretože môžu združovať viacero operácií nad súborovým systémom.
- Search panel – panel, ktorý slúži na zadanie výrazu, ktorý slúži ako vstup pre full-textový vyhľadávací algoritmus. Výstupom sú uzly, ktoré majú v názve daný výraz. Tieto uzly sa dočasne zobrazia v hlavnom okne. Po vymazaní výrazu sa obsah hlavného okna vráti do predchádzajúceho stavu, teda zobrazovania aktuálneho adresára.
  Existujú však viaceré prístupy, ako ponúknuť užívateľovi výsledky vyhľadávania.
  Jeden spočíva v prezentácii výsledkov vyhľadávania v samostatnom okne oddelene od súborovej štruktúry, čo má však svoje nevýhody. V prípade CubicExplorera, ktorý používa

tento prístup, výsledky vyhľadávania neobsahujú adresáre, pretože by bolo nutné poskytnúť možnosť tieto adresáre otvoriť, čo je však možné len pri plnohodnotnej integrácii vyhľadávania so zobrazovacím jadrom aplikácie, ktorý je základom druhého prístupu.

Druhý lepší prístup je integrovať výsledky vyhľadávania do samotného procesu pohybu v súborovej štruktúre, čo umožní nestratiť mentálny fokus na prebiehajúcu úlohu a umožní zaradiť výsledky vyhľadávania do histórie navštívených uzlov. Tento prístup používa aj WinExplorer, avšak APE zlepšuje použiteľnosť jeho implementácie. V princípe ide o to, že sa na výsledky vyhľadávania pozerá ako na fiktívny uzol adresárovej štruktúry, čo prináša výhody ako možnosť zobrazovať jeho obsah v hlavnom okne aplikácie a zaraďovať jeho návštevu do histórie, čo umožní sa k výsledkom vyhľadávania ľahko vracať. Problémom WinExplorera je mätúce zaradenie tohto uzla priamo pod koreňový adresár. Oveľa logickejší prístup, ktorý používa APE, je vytvorenie tohto uzla priamo pod prehľadávaným adresárom a jeho pomenovanie reťazcom v tvare *#hľadaná-fráza*. Takto sa dosiahne intuitívnejšej pohybu v súborovej štruktúre a výsledky vyhľadávania budú navyše bookmarkovateľné.

- Actions panel – obsahuje zoznam prebiehajúcich činností nad súborovu štruktúrou, ako je kopírovanie alebo premiestňovanie. Panel obsahuje tlačidlo prepínajúce mód práce medzi sekvenčným a nárazovým režimom. V sekvenčnom režime sa operácia zaradí do zoznamu čakateľov a začne sa vykonávať, až keď prijde na ňu rad. V nárazovom režime sa každá nová operácie začne vykonávať okamžite. Z tohto panelu je tiež možné jednotlivé operácie pozastaviť, obnoviť alebo skončiť.
- Status Panel – obsahuje informácie o vybranom uzle, ako je napríklad meno, veľkosť. Slúži ako náhrada status baru.

Hlavné okno

Nachádza sa tu obsah aktuálneho priečinka zobrazený v jednom z režimov: List, Details, Thumbnails alebo Tiles.

Predtým, než s uzlami môžeme vykonávať operácie, je nutné ich označiť. Označenie prebieha jednoducho prítomnosťou myši nad aktívnou plochou uzla, odznačenie prebieha keď myš opustí aktívnu plochu uzla. V takomto prípade hovoríme o single-select výbere, pretože

vďaka spomenutému mechanizmu je možné vybrať naraz len jeden prvok. Ak chceme vybrať viacero uzlov, robí sa to tiež prítomnosťou myši nad uzlami, tentoraz však so stlačeným klávesovým modifikátorom *Ctrl* alebo *Shift*. Tieto uzly sa pridajú do takzvanej množiny multi-select výberu. Odznačenie takýchto uzlov prebieha stlačením klávesy *Esc*. Tento spôsob sa nazýva multi-select výber. Multi-select a single-select sa navzájom nevylučujú, preto uzly môžu byť vybrané multi-selectom aj single-selectom zároveň. V tejto situácii ale dochádza pri volaní operácií ku konfliktu. Tento konflikt sa rieši tak, že operácia sa vykoná nad uzlami vybranými multi-selectom.
Tento spôsob označovania uzlov je jednou z výnimočných vlastností APE-u, ktorá konečne dokazuje, že funkcionalitu výberu je možné implementovať efektívnym spôsobom.

Operácie nad označenými uzlami aktuálneho priečinka sa robia nasledovne (v zátvorke je vždy uvedená prednastavená skratka danej operácie).

Premenovanie (*F6*):
Vyberieme uzol (okrem disku), ktorý chceme premenovať a stlačíme príslušnú klávesovú skratku. Do textového poľa napíšeme nový názov uzlu a potvrdíme Enterom. Keď nový názov vyhovuje konvenciám súborového systému, tak sa uskutoční premenovanie. Práca s textom pri editovaní názvu má jedno vylepšenie pre expertných používateľov inšpirované MS Office. Pri označení textu sa totiž často stane, že sa užívateľ rozhodne ho nezmazať a namiesto toho posunie textový kurzor pred alebo zaň. V takomto prípade tradičné textové polia presunú kurzor o jednu pozíciu ďalej, než bola hranica označeného textu a užívateľ sa potom musí vrátiť naspäť. Inteligentné textové editory presunú kurzor presne na hranicu označeného bloku. Doteraz nebol implementovaný tento spôsob práce do textových polí operačného systému.

Kopírovanie (*F7*, *Ctrl+C*):
Vyberieme uzly (okrem disku), ktoré chceme skopírovať a stlačíme príslušnú klávesovú skratku. Systém si do schránky zapamätá adresu objektu a operáciu, ktorú chceme vykonať. Presunieme sa pod uzol, kde chceme nakopírovať obsah schránky. Stlačíme F4. Systém pod daný uzol skopíruje objekt.

Presúvanie (*F8*, *Ctrl+X*):
Vyberieme uzly (okrem disku), ktoré chceme presunúť a stlačíme F8. Systém si do schránky

zapamätá adresu objektu a operáciu, ktorú chceme vykonať.
Presunieme sa pod uzol, kde chceme presunúť obsah schránky. Stlačíme F4. Systém pod daný uzol presunie objekt.

Zmazanie (*F10*):
Označíme uzly, ktoré chceme zmazať a stlačíme F10. Objekty budú zmazané.

Navyše, operácie presúvanie a kopírovanie je možné vykonávať aj pomocu drag&drop systému.
Ako cieľový uzol pre d&d operáciu je možné použiť priečinok nachádzajúci sa v hlavnom okne alebo v breadcrumbe, prípadne prejsť na iný tab ľubovolného otvoreného okna APE-u a zvoliť si cielový uzol v ňom. APE je tiež v tomto smere interoperabilný s WinExplorer-om, takže d&d funguje v oboch smeroch medzi týmito aplikáciami.
Prednastavená akcia pre drag&drop je presúvanie. Pre uskutočnenie operácie kopírovania je nutné v tretej fáze drag&dropu (viď Slovník) pridržať modifikátor *Ctrl*.

Poslednou funkcionalitou hlavného okna je možnosť vyvolať kontextové menu pravým tlačidlom nad označenými uzlami.

Úvodná obrazovka

Prednastavene zobrazuje koreňový adresár. Koreňový adresár okrem diskov obsahuje históriu a zoznam odkazov na obľúbené priečinky. Do zoznamu odkazov je možné pomocou drag&drop pridať priečinky z histórie alebo ľubovolného otvoreného tabu.

Ostatné vlastnosti

- Vektorová škálovatel'ná grafika.
  Aplikácia je písaná v knižnici WPF, preto je táto črta inherentne podporovaná.
- Integrácia do OS Windows.
  Problémom plnohodnotného nahradenia WinExplorer je jeho previazanosť s OS. Aby teda nebol užívateľ schizofrenicky nútený v rámci OS jednať s rôznymi interakčnými schémami, je potrebné nahradiť aj ostatné skryté artefakty rozhrania WinExplorera, ktoré sú prítomné v OS Windows.
  - Kontextové menu.
    Viacero kontextových menu v OS Windows je vlastne kontextovými menu uzlov

súborovej štruktúry. Zmena kontextových menu v APE by sa mala prejaviť aj v korešpondujúcich systémových menu, ako je napríklad kontextová ponuka desktopu.

- Drag&drop na desktop a štart ponuku.
  Keďže desktop a štart ponuka predstavujú uzly súborovej štruktúry, môžu sa zúčastniť d&d operácií so súborovým manažérom. Je teda možné presúvať a kopírovať uzly medzi APE a týmito objektami.
- File Open/Save Dialog.
  Aby bolo možné poskytnúť skutočne konzistentné rozhranie naprieč celým OS, aspoň čo sa týka narábania so súbormi, je nutné nahradiť dialógy na otváranie/ukladanie súborov modernou verziou s dizajnom APE. Táto črta si však bude vyžadovať samostatnú implementáciu, komplikovanú nepredvídateľnými komplikáciami, keďže sa jedná o nahradenie systémovej komponenty inou. Napríklad je známe, že v OS Windows 7 momentálne existujú dve verzie dialógu File Open/Save – novší, ktorá pribudol vo verzii Vista a starší z verzie XP. Rozdiel je poznať napríklad v známom opozdení reakcie pri single-click rozhraní.
- Nahradenie Win+E skratky.
  Táto skratka OS vyvoláva program WinExplorer, a keďže APE má za cieľ kompletne nahradiť tento program, tak sa ponúka nápad volať touto skratkou APE. Z dizajnového hľadiska to nie je dobrý nápad, pretože Win+E nie je určená na volanie prednastaveného súborového manažéra, ale špecificky na volanie “WinExplorera”. Tento síce môže byť kedykoľvek nahradený rovnako schopným súborovým manažérom, problém pri nahradení tejto skratky však je, že sa jedná o súčasť identity WinExplorera (odhliadnuc od faktu, že technicky je súčasťou OS a technicky by sme nemali meniť používateľské štandardy OS). Z dizajnového hľadiska nemôže aplikácia narušovať identitu inej aplikácie v rovnakom systéme, či už kvôli kompatibilite s budúcimi bezpečnostnými opatreniami a zmenami v systéme alebo nenarušovaniu funkcionality iného programu. Podruhé, ak je možné zachovať spätnú kompatibilitu s WinExplorerom, tak je výhodné sa o ňu pokúsiť. Preto nechávame vytvorenie volacej skratky plne v réžii užívateľa a jeho operačného systému.

Farebné odlišovanie operácií

Problém kopírovania a presúvania uzlov pomocou schránky je ten, že užívateľ počas procesu zadávania cieľového uzla nemá dostatočnú spätnú odozvu ohľadne typu operácie. Naproti tomu, počas d&d je typ prevádzanej operácie intuitívny vďaka tomu, že je nezávislý od kontextu (len v APE. Neplatí v iných súborových manažéroch). Z toho vyplýva, že prí kopírovaní pomocou klávesových skratiek (*F7*, *F8*) je potrebné používateľa viac uistiť o operácii, ktorú chce uskutočniť. Otázkou zostáva, akým spôsobom toho docieliť. Malo by ísť o jednoduchú symboliku, ktorú si užívateľ spojí s danou akciou. Jedna možnosť je využiť už používaný systém zvýrazňovania vybraných uzlov modrou farbou a vložiť doň informáciu o prebiehajúcej operácii. Máme k dispozícii niekoľko druhov zvýraznenia - pohyb, tvar, farba alebo text. Vymenovali sme ich v poradí v akom človek spracováva jednotlivé druhy zvýraznenia [Fer06]. Rozoberme si ich výhody a nevýhody. Pohyb vďaka svojej pútavosti je príliš agresívny pre akékoľvek použitie, zmena tvaru by kazila dojem z pravidelného geometrického usporiadania mriežky uzlov a text dokáže síce reprezentovať veľa rôznych stavov, ale je príliš nevýrazný a zbytočne zahlcuje UI. Farebné odlišovanie je možné efektívne použiť, ak potrebujeme rozlíšiť od seba dostatočne malý počet stavov. V takomto prípade sa dá užívateľ vytrénovať na určitú farbu, ktorú si spojí s daným stavom. Najrýchlejšia práca s aplikáciou je totiž podvedomá a užívateľ sa pri malom počte farieb dá vytrénovať na podvedomé spojenie si farby so stavmi, čo mu umožní ich odlíšiť. Výsledok je, že používateľ dosiahne používateľský komfort, pretože sa bude cítiť, že má vždy kontrolu nad operáciou ktorú vykonáva. Pozorovanie však ukázalo, že v prípade veľkého počtu entít je lepšie použiť textové identifikátory, prípadne spojiť text a grafiku. Výzkumy hovoria, že človek si len ťažko udrží prehľad o viac ako 7 veciach naraz [Mil56]. Keď má pred sebou veľký počet možností, tak si nestíha vybaviť čo ktorá môže znamenať a preto je mu potrebné poskytnúť aj textovú opisnú informáciu.

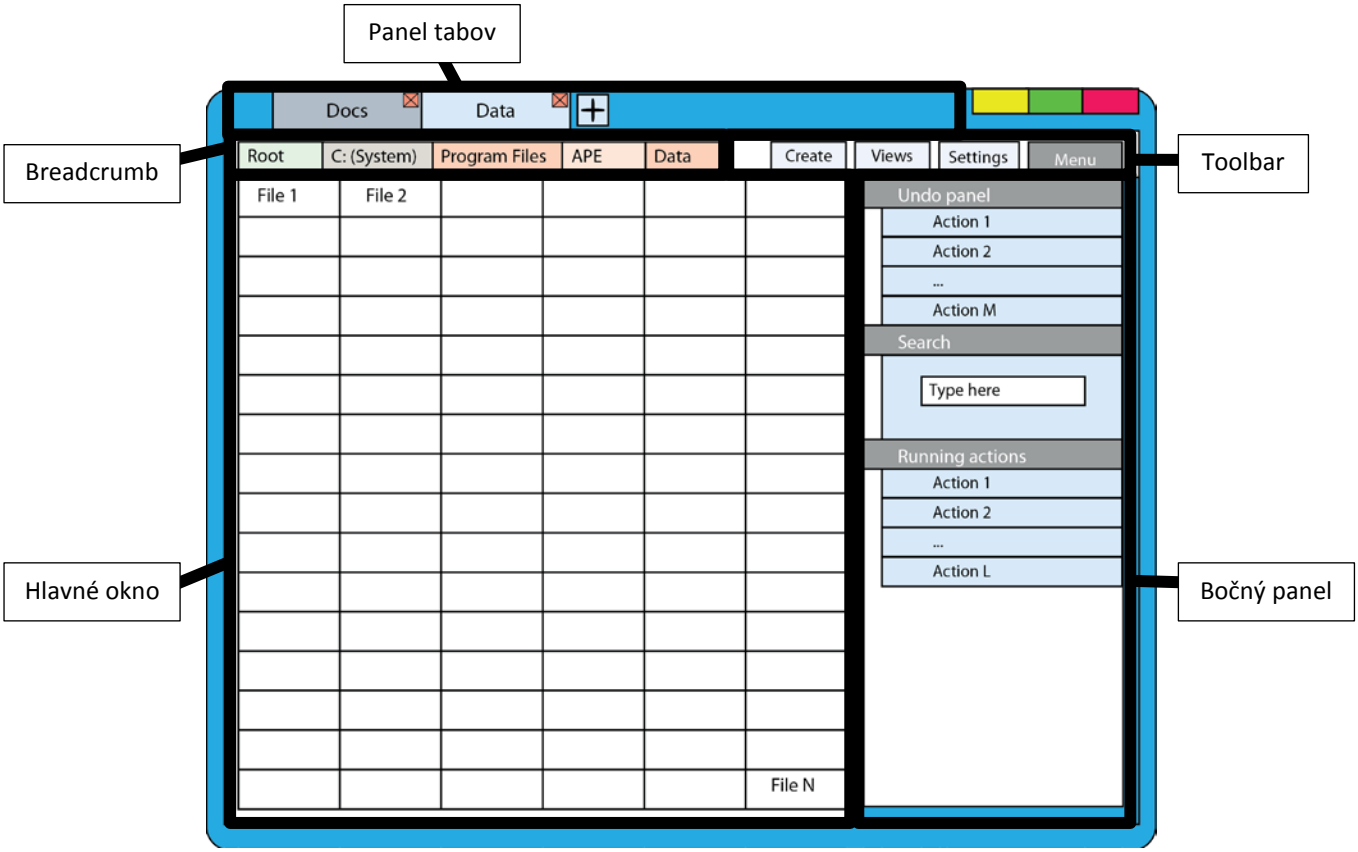
Keď človek dodrží magickú konštantu 7 pri rozhodovaní ktorú formu označovania použije, tak sa dosiahne optimálneho používateľského zážitku (ďalej len UX) pri veľkom aj malom počte volieb.

V tomto prípade použijeme farby, konkrétne pri označení uzlov určených na kopírovanie (*F7*) sa tieto zafarbia na zeleno, pri označení na premiestnenie sa zafarbia na červeno (*F8*). Modrá farba je použitá na označenie obyčajných vybraných uzlov. Pri stlačení *Esc* sa uzly odznačia a

odstránia zo schránky. Toto poskytuje vizuálne uistenie, že operácia nad súborovou štruktúrou bola skutočne prerušená.

2.2. Zhrnutie

Teraz, keď sme vymenovali celú funkcionalitu súborového manažéra APE, je čas spraviť náčrt funkcionality v grafickom editore, aby sme mali referenčný bod, ktorý budeme používať pri vývoji aplikácie. Tento náčrt nám poskytne predstavu o rozmiestnení funkcionality, rovnako ako aj ilustráciu minimalistického grafického štýlu, ktorým sa budeme riadiť. Podľa princípu minimalizmu prebehlo aj pridávanie funkcionality do aplikácie. Na začiatku bol len jednoduchý prototyp obsahujúci základnú KPPV funkcionalitu. Každá nasledujúca črta bola pridávaná k prototypu s cieľom zvýšiť efektivitu práce so súbormi a pritom byť prístupnou, intuitívnou na použitie a nespomaľovať a neodpútavať užívateľa pri používaní iných častí aplikácie.



Obrázok 11: Popis náčrtu rozhrania

3. Implementácia

Teraz, keď máme v rukách podrobné dizajnové smernice, môžme sa pustiť do ich realizácie. V dnešnej dobe možno rozdeliť vývoj aplikácií na dva základné typy – klient/server a desktopový. Klient/server sa vyznačuje rozdelením aplikácie na serverovú a klientskú časť, ktoré medzi sebou komunikujú posielaním textových správ, ktoré je možné šíriť Internetom. Serverová časť obsahuje výpočtovú časť aplikácie a klientská časť obsahuje prezentačnú časť aplikácie. Tento model vývoja nie je potrebný pre súborový manažér. "Výpočtová" časť súborového manažéra sa typicky nachádza na tom istom PC, ako "prezentačná" časť. Z tohto dôvodu sa zatiaľ neuvažuje o zovšeobecnení komunikácie medzi týmito dvoma časťami do textovej podoby, ktorú je možné šíriť kanálom, akým je napríklad sieť.
Desktopový model pozostáva z jednej časti, ktorá celá beží na jednom PC. Tento model postačuje pre aplikáciu typu súborový manažér, pokiaľ užívateľ nepotrebuje pracovať so súbormi na inom PC. Výhodou desktopového modelu sú spravidla nižšie náklady na vývoj aplikácie a eliminácia problémov, ktoré súvisia s prenosom informácie cez pomalý nespoľahlivý kanál, ako je napríklad badateľné zvýšenie času potrebného na vykonanie príkazov.

Po výbere modelu zostáva vybrať programovacie prostredie, v ktorom bude prebiehať vývoj. Vybraný jazyk musí mať dostatočný počet externých knižníc, aby sa zabezpečilo, že všetky triviálne úlohy, na ktoré je možné naraziť pri vývoji, už boli vyriešené. Pre vývoj pre desktop existuje niekoľko tradičných možností, ktoré fungujú na dnes najviac rozšírenom objektovo-orientovanom prístupe k programovaniu: Pascal, C++, Java alebo NET.
Pascal sa neteší veľkému rozšíreniu medzi programátorskou verejnosťou a preto zaostáva v množstve a kvalite doplnkových prostriedkov ako sú externé knižnice a integrované vývojové prostredie (ďalej len IDE). C++ má naproti tomu širokú mieru používania, avšak tiež prebujnelú syntax a nejednotný štandard kompilátorov, čo spôsobuje problémy pri používaní externých knižníc. Ako predposledný zostal jazyk Java. Bohužiaľ, berúc do úvahy nutnú integráciu súborového manažéra s OS, nie je implementácia v jazyku Java riešením, ktoré vyvoláva dôveru v dosažiteľnosť cieľov (táto obava sa samozrejme potvrdila pri riešení špecifických UI požadaviek). Využívanie všetkých prostriedkov OS nie je silnou stránkou Javy, keďže sa jedná o multiplatformový jazyk, ktorý musí vyhovieť spoločnej množine

prostriedkov všetkých OS. Ako posledná vhodná voľba teda zostal NET, čo je skratka pre .NET Framework. Od predchádzajúcich možností sa líši tým, že sa nejedná o žiaden špecifický jazyk, ale o iniciatívu Microsoftu poskytnúť programátorom pre OS Windows moderné prostriedky na vývoj aplikácií. Programátori tak majú k dispozícii IDE, Garbage Collector, veľké množstvo knižníc umožňujúcich riešiť väčšinu programátorských úloh a niekoľko jazykov, ktoré sú medzi sebou navzájom kompatibilné (ďalej len NET jazyky). To znamená, že jednu aplikáciu je možné rozdeliť do viacerých modulov písaných v rôznych NET jazykoch, pričom moduly môžu medzi sebou zdieľať kód. Hlavnými NET jazykmi sú v súčasnosti C# a Basic, oba z nich sú zameniteľné čo sa týka externého kódu, ktorý je z nich volateľný, oba z nich sú objektovo orientované a výrazová sila oboch je udržovaná na rovnakej úrovni. Jediný zostávajúci rozdiel je syntax. Tu Basic stráca body, pretože jeho syntax sa líši od tradičnej syntaxe väčšiny moderných jazykov, ktorá je odvodená z pôvodného C. Posledná zostávajúca voľba je programovací jazyk C#, ktorý je v súčasnosti najvhodnejšou voľbou pre vývoj pre platformu Windows. Nakoniec uvádzame rekapituláciu jeho výhod oproti vyššie spomínaným jazykom, aby bol badateľný rozdiel: familiárna syntax pripomínajúca C, Garbage Collector, nachádza sa v aktívnom vývoji, k dispozícii je subjektívne najlepšie IDE na svete, veľké množstvo knižníc, pohodlný prístup k prostriedkom OS Windows.

Náš zvolený jazyk C# má samozrejme aj dve nevýhody, tieto sú: nižší výkon oproti jazykom bez Garbage Collectora a fixovanie výslednej aplikácie na OS Windows.

Prvý nedostatok je našťastie irelevantný. Nízky výkon sa totiž môže prejavovať dvoma rôznymi spôsobmi a ani jeden z nich sa netýka APE.

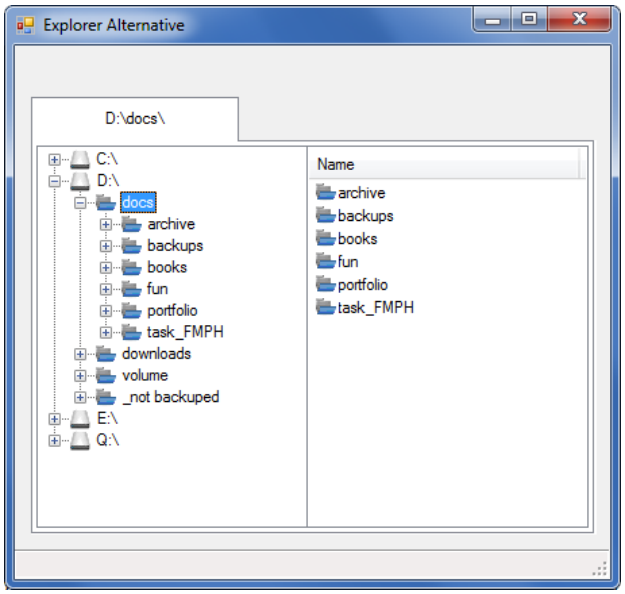
Môže sa napríklad stať, že GUI aplikácie začne odpovedať s oneskorením na užívateľove príkazy. Našťastie požiadavky samotného OS na ktorom beží APE sú dostatočne vysoké na to, aby hardvér mohol poskytnúť dostatok zdrojov na prevádzku GUI. Na modernom hardvéri mladšom ako 7 rokov je vylúčené, aby došlo k oneskoreniu odozvy GUI z dôvodu nízkeho výkonu.

Druhý spôsob, akým sa prejavuje nízky výkon, je pri riešení časovo náročných úloh, ako môže byť napríklad hľadanie prvočísel. APE však nerobí žiadne časovo náročné výpočty, jedná sa o aplikáciu, ktorá si udržuje zoznam súborov a tie prezentuje pomocou GUI.

Čo sa týka fixácie na OS Windows, je to prijateľná obeta. Pri vývoji tohto typu programu nie je možné robiť kompromisy v kvalite UI. Tú je možné zabezpečiť len pri nasadení na jeden OS. Je potrebné sa teda rozhodnúť, pre ktorý OS chceme vyvíjať program. Mal by to byť ten,

ktorý je medzi užívateľmi najrozšírenejší, aby sme mohli aspoň maximalizovať počet užívateľov, ktorý budú môcť aplikáciu využívať. Najrozšírenejším OS s podielom na trhu niekde medzi 80-90% je OS Windows [Usa]. Toto dôvodenie je príčinou možno šokujúceho zamerania sa na jeden operačný systém, ktoré bolo deklarované na začiatku práce.

Po vybratí programovacieho jazyka a cieľového OS sa dostávame k výberu vývojových nástrojov, ktoré budú použité pri vývoji aplikácie. Tu je výber jednoznačný – existujú síce zhruba 3 nezávislé IDE pre vývoj pre NET platformu, avšak všetky len dobiehajú úsilie Microsoftu v tejto oblasti. Oproti IDE Microsoft Visual Studio zaostávajú v podpore platformy NET a fukčnej sade. Visual Studio je poskytované zadarmo v najnižšej funkčnej verzii Express pre komerčné využitie, pre akademické účely je zadarmo poskytovaná druhá najnižšia verzia Professional, ktorá bude použitá pre vývoj APE.



Obrázok 12: Explorer Alternative

3.1. Prvý pokus

NET obsahuje 2 grafické knižnice, ktoré je možné využiť pri implementácii GUI. Sú to knižnice WinForms a WPF. WinForms je staršou knižnicou a medzi jej hlavné nevýhody patrí veľké úsilie nutné na napísanie vlastných komponentov UI. Táto vlastnosť sa ukázala byť kritickou pre vývoj softvéru vyžadujúceho si neštandartné UI, akým je napríklad APE. Každopádne, prvý pokus o implementáciu bol vykonaný práve pomocou knižnice WinForms najmä z toho dôvodu, že je overená vďaka dlhej dobe svojej existencie a jej použitie na jednoduché účely

je veľmi intuitívne a priamočiare. Pre osvojenie si tejto knižnice stačí prakticky poznať základy jazyka C#, ktoré je možné nadobudnúť z úvodných kurzov pre platformu NET [IluC#, Pro].
Program, ktorý vznikol v tejto prvej iterácii vývojového cyklu bol nazvaný *Explorer Alternative* (viď Obrázok 12) a je priložený k tejto práci. Podarilo sa v ňom implementovať KPPV funkcionalitu, bohužiaľ vďaka neflexibilite prispôsobovania UI bola táto vetva vývoja opustená.

3.2. Knižnica WPF

WPF je skratkou pre Windows Presentation Foundation, čo sa dá preložiť ako *Základ prezentácie na OS Windows*. Tento čudný názov vyjadruje, že sa jedná o grafickú knižnicu a že je to preferovaný spôsob tvorby GUI pre OS Windows. Pôvodný názov WPF bol Avalon, lenže z dôvodu optimalizácie času, ktorý je nutné vynaložiť na napísanie WPF oproti Avalonu vyhral nakoniec kryptický reťazec WPF, jednak vďaka svojej polovičnej dĺžke a za druhé vďaka striedaniu pravej a ľavej ruky pri písaní tohto názvu pri použití všetkých desiatich prstov. Výsledkom je, že vyhľadávanie na webe sa urýchlilo. Už z tohto úvodu je zrejmé, že do vývoja WPF bolo vložené nemálo úsilia, keďže jeho cieľom je stať sa hlavnou knižnicou pre vývoj desktopových a RIA aplikácií.

Medzi jej výhody patrí vysoká znovupoužiteľnosť častí UI, oddelenie stavu komponent UI od ich výzoru, deklaratívny prístup k tvorbe štruktúry programu a spôsob práce nútiaci programátora používať dizajnové vzory. Bohužiaľ, jedná sa o mladú technológiu, preto k nevýhodám zatiaľ stále patrí neúplná funkcionalita jednotlivých častí knižnice a nedospelosť vývojových nástrojov. K tomu sa pridávajú pomerne veľké požiadavky na znalosti programátora, ktorý musí zvládnuť deklaratívny prístup k tvorbe GUI, a v prípade, že sa snaží vytvoriť vlastné komponenty, tak je nutné aby na naučil orientovať v návrhových vzoroch, ktoré využíva WPF a naučil sa využívať množstvo podporných nástrojov. Mal by si osvojiť debuggovacie nástroje ako Mole a Snoop, ktoré umožňujú prechádzať štruktúru komponent UI a Expression Blend, ktorý umožňuje tvorbu vektorovej grafiky pre použitie vo WPF aplikácii. Avšak knižnica WPF má podstatnú výhodu, ktorá ju v súčasnosti radí nad ostatné riešenia. Tou je používanie vektorov na definíciu vizuálnych elementov rozhrania, vďaka čomu je možné škálovať UI pre rôzne veľkosti zobrazovacích zariadení, pričom sa nestráca

kvalita prekreslenia. Odstráni sa tak problém miniaturizovaného rozhrania pri použití veľkej hustoty obrazovej mriežky a tiež problém pixelizácie pri práci so zväčšeným UI, s ktorým typicky pracujú slabozraký ľudia.

Z týchto dôvodov a mnohých ďalších, ako sú garantovaná podpora knižnice pre dohľadnú dobu a jednoduchá migrácia WPF aplikácií do prostredia prehliadačov, je WPF doporučovaná voľba pre vývoj nových aplikácií pre platformu Windows. To spolu s faktom, že tvorba vlastných UI komponent je vďaka architektúre WPF veľmi prirodzenou, robí WPF vhodnou knižnicou pre vývoj súborového manažéra APE.

Záver

WinExplorer aj ostatné spomínané aplikácie rozhodne majú svoje miesto na dnešnom trhu so súborovými manažérmi. Poskytujú akýsi vstupný bod pre začínajúceho používateľa, ktorý sa chce oboznámiť s možnosťami tohto typu softvéru. Snažia sa robiť mnoho vecí naraz, aj keď žiadnu z nich nerobia efektívne. V prípade, že sa užívateľ zamyslí nad funkcionalitou, ktorú skutočne potrebuje a príde na to, že na reálnu prácu mu vo väčšine prípadov postačí jednoduchý softvér, tak je pripravený prejsť na APE.

Čo sa týka cieľov práce, tak sa ich podarilo naplniť z dvoch tretín. Podarilo sa spraviť analýzu doménovej problematiky a podarilo sa spraviť špecifikáciu rozhrania, ktoré sa vyhýba daným problémom a zjednoduší prácu s manažérom. Prvý pokus o implementáciu špecifikácie bol vykonaný len po implementáciu základnej KPPV funkcionality súborového manažéra, keďže použitá grafická knižnica WinForms neposkytuje efektívne prostriedky pre tvorbu neštandartného UI. Pre ďalšiu iteráciu vývoja bola zvolená knižnica WPF, ktorá už poskytuje široké možnosti prispôsobovania grafických komponentov UI, hoci táto pozitívna vlastnosť je vyvážená pomerne dlhou dobou návyku na spôsob práce s WPF.

Ako ďalšie pokračovanie tu začatej práce vidíme práve implementáciu špecifikácie z kapitoly 2 pomocou programovacieho jazyka C# a knižnice WPF.

Príloha A - Slovník

Táto príloha má za cieľ zaručiť, že význam prečítaného textu je rovnaký ako skutočne zamýšľaný význam. Je pravdepodobné, že niektoré pojmy už čitateľ pozná, ale aby sa predišlo občasným nedozorumeniam, tak táto príloha má za cieľ zjednotiť slovník autora a čitateľa, prípadne objasniť niektoré exotickejšie termíny. Jedná sa spravidla o pojmy z doménovej problematiky, ktorú riešia súborové manažéry. Touto doménou je narábanie so súbormi uloženými na lokálnych pamäťových médiách PC.

APE – Another Penultimate Explorer, čoho volný slovenský preklad je "Ďalší takmer najlepší prehliadač". Druhý význam slova APE je "Opica".
Aktívna plocha – časť UI, po ktorej kliknutí nastane reakcia programu v podobe vykonania niektorej funkcie alebo stavu na obrazovke. Dobre navrhnuté UI umožňuje užívateľovi ľahko rozpoznať aktívne plochy.
Benchmark – Anglický výraz pre štandard voči ktorému sa pomeriavajú objetky z danej oblasti.
Breadcrumb – Anglický výraz pre chlebovú omrvinu. Používa sa od OS Windows verzie Vista pre označenie špecifickej komponenty UI, ktorá zobrazuje sled odkazov, po kliknutí na ktoré sa obsah hlavného okna patrične zmení. V prípade súborových manažérov odkazy zo sledu smerujú na adresáre tvoriace cestu k aktuálnemu zobrazovanému adresáru súborovej štruktúry. Breadcrumb komponenta patrí do kategórie takzvaných hierarchických komponent, keďže poskytuje užívateľovi prehľad o určitej hierarchii objektov.
Drag&drop – Spôsob interakcie s UI pozostávajúci z niekoľkých fáz. V prvej fáze je stlačené tlačidlo myši nad zdrojovou komponentou. V druhej fáze je myš so stlačeným tlačidlom potiahnutá nad cieľovú komponentu. V tretej fáze je pustené tlačidlo myši nad cieľovou komponentou.
Skatka pre drag&drop je d&d.
Feature – Anglický výraz pre črtu/vlastnosť. V tejto práci sa tým spravidla myslí vlastnosť programu.
Feature bloat – Stav, kedy program obsahuje toľko funkcionality, že znižuje použiteľnosť programu.
Feature creep – Anglický výraz pre bezhlavé pridávanie funkcionality do programu.

Dôsledkom tejto činnosti je, že používanie programu sa stane pre používateľa zaťažujúce, keďže sa medzi ponúkanou funkcionalitou nevie zorientovať.

Focus – Vlastnosť UI komponenty, ktorá určuje či komponenta príjma užívateľove príkazy. Táto vlastnosť je reprezentovaná premennou, ktorá nadobúda jeden z dvoch stavov: Focused alebo Unfocused. V stave Focused komponenta príkazy príjma a v stave Unfocused ich ignoruje. Takáto stavovosť je nutná z toho dôvodu, že komponenty väčšinou predstavujú rovnaké entity (uzly súborovej štruktúry) alebo jednoducho reagujú na rovnaké príkazy (komponenty obsahujúce scrollbar reagujú na otočenie kolečka myši). Keďže počet príkazov, ktoré je užívateľ schopný zadať z klávesnice je limitovaný a aj z nich je len určitá podmnožina pamäťovo alebo motoricky ergonomická, nie je možné pre každú takúto komponentu používať odlišné príkazy. Preto sa príkazy posielajú vždy len komponente, ktorá je v stave Focused. Takáto komponenta môže byť v aplikácii vždy len jedna. Je to zabezpečené tým spôsobom, že pri nastavení komponenty do stavu Focused správca komponent nastaví všetky ostatné do stavu Unfocused.

Hmatateľné rozhranie – UI ktoré dáva užívateľovi vizuálnu odozvu, keď sa myš nachádza nad aktívnou plochou. V podstate dnes už každé UI používajúce myš je v tomto zmysle aspoň čiastočne hmatateľné.

Komponenta UI – Časť UI vizuálne a funkčne odlíšiteľná od zvyšku UI.

KPPV – Skratka znamenajúca Kopírovanie, Presun, Premenovávanie, Vymazávanie. Jedná sa o základné operácie, ktoré musí umožňovať súborový manažér.

Les – Množina stromov.

Listview – Komponenta UI patriaca do kategórie všeobecných skupinových komponent, pretože zobrazuje určitú skupinu objektov, medzi ktorými však nepanuje žiadna hierarchia.

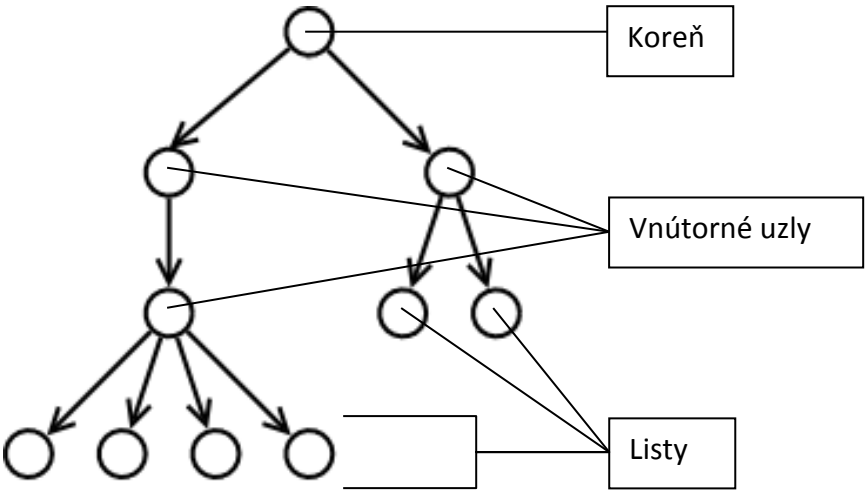
Point&Click adventúra – Žáner počítačových hier. Zlatý vek týchto hier bol v prvej polovici 90. rokov. Dnes sú už mimo módy práve z dôvodu frustrácie hráčov z ich používateľského rozhrania. Niekedy sa tento pojem používa ako metafora pre zle navrhnuté UI. Takéto UI spravidla pozostáva z miniatúrnych klikateľných plôch a malého vizuálneho vodítka k tomu, kde sa tieto klikateľné plochy nachádzajú.

RIA – Rich Internet Application. Aplikácia implementovaná v jednom z frameworkov Flash, JavaFX alebo Silverlight. Vyznačuje sa používaním prostriedkov PC, ktoré nie sú štandartne prístupné z API webového prehliadača, a tak poskytuje väčšie možnosti programátorom webových aplikácií.

Strom – Množina objektov, ktorú je možné jednoducho vizualizovať vďaka vzťahom, ktoré objekty tejto množiny majú medzi sebou. Každý objekt je vo vzťahu rodič-potomok buď s jedným alebo viacerými ďalšími objektami. Počet možností v akom môže byť objekt vo vzťahu k ostatným je však limitovaný na nasledovné tri prípady:

1. Objekt, ktorý sa nachádza v jednom alebo viacerých vzťahoch, kde zastáva iba úlohu rodiča, sa nazýva koreň. Takýto objekt je v strome len jeden.
2. Objekt, ktorý sa nachádza iba v jednom vzťahu, kde zastáva úlohu potomka, sa nazýva list. Takýchto objektov môže byť v strome viacero.
3. Objekt, ktorý sa nachádza vo viacerých vzťahoch, pričom práve v jednom zastáva úlohu potomka a vo všetkých ostatných zastáva úlohu rodiča, sa nazýva vnútorný uzol.

V prípade, že je pre čitateľa koncept stromu nejasný, je dobré, aby sa pokúsil nájsť analógiu medzi uvedenou definíciou a nasledujúcim obrázkom. Obrázok je totižto príkladom stromu z definície a definícia je všeobecným popisom objektu na obrázku.



Obrázok 13: Príklad stromu

Treeview – Komponenta UI patriaca do skupiny hierarchických komponent, čo je špeciálny prípad skupinových komponent, ktoré zobrazujú skupinu objektov tvoriacu hierarchiu. Konkrétne treeview zobrazuje strom objektov.

UI – User Interface. Anglický výraz pre používateľské rozhranie.

UX – User experience. Anglický výraz pre používateľský zážitok. Jedná sa o pocit používateľa

z používania aplikácie. Čím je lepší, tým lepšia je spokojnosť užívateľa s aplikáciou a v prípade komerčných produktov sa zvyšuje vernosť firme a ochota platiť za nové verzie produktu.

Príloha B – Priložený softvér

K práci sú priložené inštalačné súbory freeware súborových manažérov, ktoré boli analyzované v tejto práci. Tieto programy sú voľné šíriteľné za účelom nekomerčného používania. Čitateľ sa môže s nimi oboznámiť a porovnať si ich použiteľnosť s programom, ktorý tradične používa.
Priložené sú tieto programy: CubicExplorer, FreeCommander, Double Commander, unreal commander, q-dir, Explorer++, STDU Explorer, .xplorer^2.

Rovnako je priložená aplikácia Alternative Explorer s jej zdrojovými kódmi. Jedná sa o prvý pokus o implementáciu súborového manažéra, ktorý vyhovuje špecifikácii z tejto práce.

Posledným priloženým elektronickým obsahom sú zdrojové kódy súborového manažéra APE. Je v nich zatiaľ implementovaný kód, ktorý implementuje KPPV operácie a sú v ňom položené základy UI, na ktoré je možné nadviazať a dokončiť ich.

Literatúra

[Fer06] Ferko A., *Creating Web Graphics* (2006), *prednášky*

[Whi02] White A. W., The elements of graphic design: space, unity, page architecture, and type (2002), Allworth press

[Wit11] Withalm J., *IT Quality Management* (2011), *prednášky*

[Cli05] Frank A., *www.dontclick.it* (2005), (artwork for diploma in Communication Design)

[Mil56] Miller G. A., *The Magical Number Seven, Plus or Minus Two* (1956), Psychological Review journal

[Usa] *Usage share of operating systems*, http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

[Unsh] Nathan A., *WPF 4 Unleashed* (2010), Sams.

[Ctrl] Podila P., Hoffman K., *WPF Control Development* (2010), Sams.

[Act] Feldman A., Daymon M., *WPF in Action with Visual Studio 2008* (2009), Manning Publications.

[IluC#] Solis D., *Illustrated C#* (2008), Apress

[Pro] Troelsen A., *Pro C# 2010 and the .NET 4 Platform, Fifth Edition* (2010), Apress

Zoznam obrázkov