Tvorba aplikace v Linuxu

Bakalářská práce

Tomáš Václavík, DiS

Vedoucí bakalářské práce: Mgr. Jiří Pech, Ph.D.

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

2009

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 18. 4. 2009

Anotace

Tato práce podává přehled současných vývojových prostředí pro Linux. Jednotlivá prostředí jsou mezi sebou vzájemně porovnána dle stanovených kritérií. Práce dále popisuje rozdíl v tvorbě aplikací pro Linux a Windows. Nakonec je popsán vývoj konkretní aplikace ve vybraném prostředí.

Abstract

This work makes a list of actual integrated development environments available for Linux. Every development environment is compared among each other according to defined criterion. Furthermore the work describes a difference in development of applications under Windows and Linux. Finally is described a process on a development of concrete application in the selected development environment.

Poděkování

Rád bych poděkoval vedoucímu této práce

Mgr. Jiřímu Pechovi, Ph.D. za cenné rady a vedení během mé práce.

Obsah

1 ÚVOD....7

2 CÍLE PRÁCE....9

3 SOUČASNÝ STAV PROBLEMATIKY....10

3.1 Literární rešerše....11

4 METODIKA....15

4.1 Sestavení referenčního modelu....15

4.2 Způsob hodnocení....15

4.3 Výběr vývojových prostředí....17

4.4 Testovací příklady....18

5 TEORETICKÝ ÚVOD....20

5.1 Linux....20

5.2 Grafické uživatelské prostředí - GUI....21

5.3 Knihovna Qt....21

5.4 Knihovna GTK+....22

5.5 Překladač....23

5.6 Debugger....23

5.7 Vývojové prostředí - IDE....24

6 POPIS REFERENČNÍHO MODELU....25

6.1 Charakterová kategorie....25

6.2 Designová kategorie....26

6.3 Kódová kategorie....26

6.4 Správová kategorie....26

7 VÝBĚR VÝVOJOVÝCH PROSTŘEDÍ....27

8 SROVNÁNÍ VÝVOJOVÝCH PROSTŘEDÍ.....31

9 SROVNÁNÍ TVORBY APLIKACE VE WINDOWS A LINUXU.....45

9.1 Fundamentální rozdíl.....45

9.2 Obliba negrafických nástrojů pro vývoj.....46

9.3 Možnost předat parametry aplikaci.....46

9.4 Využití svobodného softwaru.....46

9.5 Rozvržení layoutu na formuláři.....47

9.6 Způsob publikování aplikace.....47

9.7 Volba grafických toolkitů.....48

10 VÝVOJ KONEČNÉ APLIKACE.....49

10.1 Volba vývojového prostředí.....49

10.2 Popis aplikace.....50

10.3 Postup vývoje a vzniklé problémy.....50

10.4 Vyhodnocení.....54

11 ZÁVĚR.....56

12 PŘEHLED LITERATURY.....58

13 SEZNAM PŘÍLOH.....60

14 PŘÍLOHY.....61

1 Úvod

Dnes již není nutné psát aplikace pro Linux pomocí příkazové řádky jako tomu bylo dříve. Vývojáři mají k dispozici mnohem silnější nástroj. Tento nástroj se nazývá vývojové prostředí neboli IDE. O Linuxu se často hovoří jako o vývojářské platformě, a proto mne zajímalo, jaké nástroje, o jaké kvalitě a v jakém množství má k dispozici. Zajímalo mne také, jak obtížné je vytvořit v Linuxu použitelnou aplikaci.

Náplní mého výzkumu je zmapování současné situace a sestavení seznamu nejrozšířenějších vývojových prostředí pro tuto platformu. Dále v práci tato vývojová prostředí mezi sebou porovnávám. Snažím se o objektivní a nezaujatý přístup. Smyslem tohoto srovnání je vystihnout charakter každého z nich a seřadit je dle jejich kvality. Výsledkem porovnání je tabulka s přehledem vlastností a hodnocením všech zkoumaných prostředí.

Práce také pojednává o možných rozdílech mezi operačními systémy Windows a Linux a to ve způsobu, jakým se v nich aplikace vytvářejí. Rozdíl je popsán spíše z hlediska způsobu tvorby aplikace, nežli z hlediska práce ve vývojových prostředích na různých platformách.

V další části této práce si vybírám vývojové prostředí, které mi nejvíce vyhovuje a v něm vytvářím konečnou aplikaci. Touto aplikací bych rád demonstroval, jak snadno lze ve vybraném vývojovém prostředí vytvořit použitelný přehrávač multimédií s grafickým uživatelským rozhraním.

V práci je dále popsán postup při jeho tvorbě a řešení vzniklých problémů.

V příloze je umístěno schéma struktury referenčního modelu, na kterém je patrné, podle jakých kritérií jsem prostředí hodnotil. Nejdůležitější přílohou je tabulka s hodnocením testovaných prostředí, kde jsou patrné vlastnosti každého z nich. Dále v příloze naleznete screenshot konečné aplikace a ukázku části kódu, na které je popsáno řešení problému předávání informací mezi objekty.

Přiložené médium obsahuje sadu zdrojových kódů, které jsem použil při testování jednotlivých prostředí a zdrojový projekt konečné aplikace. V adresáři */bez_gui_designeru/* se nalézají zdrojové kódy pro vývojová prostředí bez návrháře grafického rozhraní. V adresáři */s_gui_designerem/* se nacházejí zdrojové kódy pro ta, co jsou designerem vybavena. Projekt konečné aplikace se nachází v adresáři */konecna_aplikace/prehravac/*.

2 Cíle práce

Cíle této práce můžeme shrnout do třech částí.

Hlavním cílem je podat přehled současných vývojových prostředí, která jsou k dispozici pro Linux. Práce by měla zachytit rysy jednotlivých vývojových prostředí, vzájemně je porovnat na základě předem definovaného modelu a ohodnotit je podle stanovené hodnotící metody. Výsledek mého srovnání pak může programátorovi posloužit při rozhodování, které vývojové prostředí je pro něj nejvhodnější.

Dalším cílem je porovnat, v čem se liší tvorba aplikace v Linuxu a ve Windows. To může opět posloužit programátorovi, který přechází ze systému Windows na Linux a připravit ho na možné rozdíly.

Cílem posledním je pomocí odladěné aplikace demonstrovat, jak snadno lze v Linuxu vytvořit graficky i uživatelsky přívětivé aplikace srovnatelné s aplikacemi pro Windows. Práce by měla také dokázat, že pomocí kvalitního vývojového prostředí lze i s pouhou základní znalostí některého programovacího jazyka snadno a rychle vytvořit použitelnou aplikaci.

Výstupem této práce je tabulka s přehledem současných vývojových prostředí pro Linux, sada ukázkových aplikací a konečná odladěná aplikace spolu s popisem její tvorby.

3 Současný stav problematiky

V současné době je k dispozici celá řada integrovaných vývojových prostředí pro Linux. Značně rozsáhlý seznam, skýtající až šedesát prostředí, je k dispozici k nahlédnutí na internetové adrese viz [1]. Jedná se o pouhý seznam, ve kterém nejsou jednotlivá prostředí mezi sebou porovnávána. U každého vývojového prostředí je uvedena jeho krátká charakteristika a počet stažení. Bohužel, na údaj o počtu stažení nelze nahlížet jako na statistický ukazatel počtu používání daného prostředí, neboť se jedná pouze o počet stažení daného prostředí skrze tento server. Uvedený údaj tedy pouze ukazuje kolik uživatelů tohoto serveru si vývojové prostředí stáhlo, ovšem již nezahrnuje kolik uživatelů si prostředí stáhlo z jiného zdroje, např. z domovských stránek produktu. Tento údaj nelze také použít jako ukazatel oblíbenosti daného prostředí, neboť to, že si uživatel program stáhl neznamená, že jej používá. Z uvedeného seznamu jsem při výběru prostředí pro testování nevycházel, ale spíše jsem se zaměřil na ohlasy uživatelů na různých diskuzních serverech. Seznam mi posloužil pouze ke zjištění, jaká vývojová prostředí jsou k dispozici.

Při hledání některé vědecké práce na téma porovnání vývojových prostředí v Linuxu jsem žádnou použitelnou a spolehlivou nenalezl. Jediný článek, který se tímto tématem zabývá do hloubky je publikován na portálu Wikipedie. Bohužel autor tohoto srovnání není znám a podle

referencí nebylo dílo ani odnikud převzato. Článek se nachází na internetové adrese viz [2].

Autor zde porovnává jednotlivá vývojová prostředí podle základních charakteristik a výsledky zaznamenává do tabulky, ve které jsou uvedeny pouze hodnoty *ano* a *ne*. Toto řešení mi nevyhovuje, neboť nedokáže dostatečně popsat danou vlastnost. Jelikož se nejedná o fundovaný zdroj, nemohu data převzít do své práce. Tento článek mne však inspiroval a ve své práci jdu dále, kdy zavádím vícestupňové hodnocení.

3.1 Literární rešerše

Učebnice ABC/Linuxu [3]

Tato kniha je vydávána pod licencí GNU FDL a lze ji tudíž stáhnout a používat zdarma. Publikace je vhodná zejména pro začínajícího uživatele Linuxu. Uživatel je seznámen s historií Linuxu a se základními pojmy. Je mu vysvětlena práce s procesy, souborovým systém a s příkazovou řádkou. Dále uživatel v knize nalezne popis a princip práce s grafickým serverem a počítačovou sítí.

Advanced Linux Programming [4]

I tato publikace je vydávána pod svobodnou licencí Open Publication License a je zdarma ke stažení. V porovnání s předešlou knihou je určena již pro pokročilé programátory a uživatele Linuxu. Čtenáři pomůže s úplnými začátky, kdy popisuje, jak lze pod Linuxem napsat a přeložit

jednoduchou aplikaci. Seznámí čtenáře s kolekcí základních nástrojů pro vývoj aplikací v Linuxu jako jsou GCC a GDB a popíše postup, jak pomocí nich aplikaci přeložit. Postupně pak přechází ke složitějším věcem, jako je práce s procesy a vlákny a komunikace mezi nimi. Všechny popisované principy jsou předváděny přímo na ukázce zdrojového kódu a jednotlivé části jsou podrobně a srozumitelně vysvětleny.

Pro téma této práce je pak zajímavá především kapitola *3.2.2 Using fork and exec*. V této kapitole autor uvádí rozdíl při vytvoření a spuštění nového procesu v Linuxu a ve Windows. Další zajímavou kapitolou, ve které autor pojednává o rozdílu v meziprocesorové komunikaci ve Windows a Linuxu, je kapitola *5.4.5 FIFOs*.

Všechny uvedené ukázky v knize jsou napsány v jazyku C++, ale k pochopení vysvětlovaných principů není nutně znalost tohoto jazyka vyžadována. Kniha je v angličtině, ovšem používá jednoduché slovní obraty, kterým porozumí i člověk se středně pokročilou znalostí jazyka.

Tato publikace byla vhodnou pomůckou při dosažení cíle práce, kdy porovnávám rozdíl tvorby aplikace v Linuxu a ve Windows. Kniha se zabývá pouze samotnou tvorbou aplikace a nezmiňuje se o žádném vývojovém prostředí, ani o práci s využitím některého grafického toolkitu.

Beginning Linux Programming [5]

Podobně jako předchozí kniha, i tato uživatele seznámí se základy tvorby aplikací v Linuxu a systémem samotným. Na rozdíl ale od předchozí publikace se zaměřuje více na využití programových knihoven. Uživatel v knize nalezne návod jak pracovat například s databází MySQL nebo jak vytvořit grafické okno a ošetřovat vzniklé události. Poměrně velká část je věnována popisu tvorby graficky uživatelsky přívětivých aplikací.

Pro tuto práci jsou zajímavé hlavně kapitoly, ve kterých autor popisuje způsoby práce s grafickými knihovnami jako jsou GTK+ a Qt. Právě poznatků z těchto kapitol je využito při tvorbě aplikací pro otestování jednotlivých vývojových prostředí a při vývoji konečné aplikace. Zajímavá je pak kapitola *16. Programming GNOME using GTK+*, kde autor popisuje práci s okny a dalšími prvky z této grafické knihovny. Další zajímavou kapitolou je kapitola *17. Programming KDE using Qt*, kde se pro změnu autor věnuje popisu tvorby aplikací pro desktopové prostředí KDE. V *kapitole 9. Development Tools* autor také podává stručnou charakteristiku několika nejrozšířenějších prostředí.

www.root.cz [6]

Tento server se zabývá převážně operačním systémem Linux a obecně programováním. K dispozici je obrovské množství článků, diskusí, zpráv, knih a manuálů.

Právě na diskuze a názory k článků jsem se zaměřil nejvíce. Zajímaly mne reakce uživatelů na jednotlivé články a hlavně jejich zkušenosti s některými diskutovanými vývojovými prostředími. Na základě tohoto průzkumu jsem se pak rozhodl, která prostředí vyberu k otestování.

www.abclinuxu.cz [7]

Na uvedené adrese se nalézá server s podobným zaměřením jako výše zmiňovaný. Opět zde nalezneme různé články a diskuze.

Z hlediska této práce je pak zajímavý seznam vývojových prostředí umístěný na adrese viz [8]. U jednotlivých prostředí je uveden i údaj s hodnocením uživatelů. Tuto hodnotu již lze považovat za použitelný statistický údaj a přihlížím k němu při volbě, která prostředí porovnám. Na tomto serveru nalezneme také diskuze uživatelů na téma vývojová prostředí.

4 Metodika

V této části rozeberu postup, který jsem zvolil k dosažení cílů práce.

4.1 Sestavení referenčního modelu

Jako první bod bylo sestavení tzv. referenčního modelu, podle něhož budu schopen jednotlivá vývojová prostředí ohodnotit a posléze porovnat. Tento model odráží strukturu jakéhosi ideálního vývojového prostředí a ostatní vývojová prostředí s ním budou srovnávána.

Struktura modelu vychází ze zkušeností, které již mám s prací ve vývojových prostředí Delphi, Builder, NetBeans a Visual Studio ve Windows. Na základě těchto zkušeností jsem definoval kritéria, podle kterých budu jednotlivá prostředí porovnávat.

Při sestavování modelu bylo nutné abstrahovat od nedůležitých vlastností a zabývat se pouze základními charakteristikami, které by mělo každé vývojové prostředí mít. Struktura referenčního modelu bude rozebrána dále v práci.

4.2 Způsob hodnocení

Během hodnocení pak zjišťuji, na kolik se zkoumaná vlastnost vývojového prostředí přiblížila hodnotě referenčního modelu. Za účelem rovnocenného srovnání zavádím vlastní čtyřprvkovou klasifikační stupnici.

Stupnice nabývá hodnot od 0 do 3 a tyto hodnoty mají takovýto význam:

0 – nemá

1 – špatné

2 – průměrné

3 – výborné

Hodnocení *nemá* znamená, že dané vývojové prostředí nedisponuje testovanou vlastností. Hodnocení *špatné* znamená, že prostředí danou vlastnost sice má, ale přibližuje se vlastnosti referenčního modelu jen velice málo. Hodnocení *průměrné* znamená, že splňuje testovanou vlastnost tak na 50 % a *výborné,* že testovanou vlastnost splňuje zcela nebo téměř úplně.

Pokud například testuji prostředí na základě editoru zdrojového kódu, v referenčním modelu jsem si stanovil deset kritérií a dané prostředí jich splňuje šest, hodnocení za tuto vrstvu je tedy *průměrné*. Přiblížilo se více než na 50 % k ideální hodnotě referenčního modelu.

Číselné hodnoty klasifikační stupnice jsou pak použity při sumarizaci výsledků testů a k seřazení vývojových prostředí podle počtu dosažených bodů. Výsledky srovnávání vývojových prostředí jsou vyneseny do tabulky v příloze *B. Tabulka s hodnocením vývojových prostředí*.

4.3 Výběr vývojových prostředí

Vývojových prostředí v Linuxu je celá řada a jsou různého druhu. Já se rozhodl ve své práci zabývat pouze vývojovými prostředími, která podporují nejrozšířenější programovací jazyky a dokáží vytvořit kompilovanou binární aplikaci. Z tohoto hlediska netestuji vývojová prostředí pro tvorbu webových stránek či skriptů.

Zdrojem, který jsem použil pro rozhodnutí, která vývojová prostředí vyberu, byly převážně diskuzní portály zabývající se tématikou Linuxu a programování. Tyto servery byly již zmíněny v předešlé kapitole. Procházením různých článků na téma vývoj aplikace v Linuxu a čtením reakcí uživatelů jsem postupně získával informace o tom, která vývojová prostředí jsou mezi uživateli nejoblíbenější a nejrozšířenější. Z takto získaných informací lze jen těžko vytvořit statistiku, ovšem mně to pomohlo zaměřit se na vývojová prostředí nejvíce zmiňovaná a ta mezi sebou porovnat.

Jako další zdroj byly domovské stránky různých vývojových prostředí, na které jsem se dostal pomocí internetových vyhledávačů.

4.4 Testovací příklady

V této kapitole jsou popsány testovací aplikace, které jsem vytvořil pro každé z vybraných prostředí za účelem ohodnotit jeho jednotlivé části.

Všechna vývojová prostředí jsem nejprve otestoval na notoricky známé aplikaci typu *hello world*. Tato aplikace u vývojových prostředí bez GUI návrháře vypíše do výstupního bufferu text „Ahoj svete!“. Na přiloženém médiu se nachází v adresáři */bez_gui_designeru/HelloWorld/*. U vývojového prostředí vybaveného tímto nástrojem aplikace vytvoří okno s titulkem „Ahoj svete!“ a navíc také stejný text vypíše i do výstupního bufferu. Touto aplikací chci ověřit, zda je vývojové prostředí správně nainstalované, zda má k dispozici správnou cestu k hlavičkovým souborům knihoven v systému a případně i ke kompilátoru. Naleznete ji na přiloženém médiu v adresáři */s_gui_designerem/* v podadresáři se jménem vývojového prostředí.

Další testovací aplikace se skládá ze dvou tříd. Druhá třída je odvozená od třídy první a obsahuje několik atributů včetně dvou přetížených metod. Metody mají stejný návratový typ a název, ovšem rozdílný počet parametrů. Přetížené metody jsem se rozhodl použít, abych při testování nástroje pro automatické dokončování kódu ověřil, zda budou obě metody nabídnuty a odlišeny. Zdrojový kód se nachází na přiloženém médiu v adresáři */bez_gui_designeru/dedicnost/*.

Za účelem otestovat kvalitu návrháře grafického rozhraní vytvářím v každém prostředí, které je tímto nástrojem vybaveno, jednoduchou aplikaci. Tato aplikace se skládá z formuláře, na kterém jsou rozmístěny prvky o určitých rozměrech. Jedná se o dvě tlačítka a dvě textová návěstí. Při stisku některého z tlačítek se v odpovídajícím návěstí zobrazí zpráva o této akci. Touto jednoduchou aplikací zjišťuji, jak se s návrhářem pracuje, jakých dosahuje kvalit a jakým způsobem vývojové prostředí zajišťuje spojení prvku a metody, která ošetřuje vzniklou událost. Zdrojový kód je uveden na přiloženém médiu v adresáři */s_gui_designerem/* v pod adresáři se jménem prostředí.

5 Teoretický úvod

V této práci jsou používány některé pojmy, které je potřeba vysvětlit.

5.1 Linux

Linux, či přesněji GNU/Linux, je moderní operační systém. Skládá se z jádra zvaného Linux, jehož autorem je Linus Torvalds a základních knihoven a nástrojů pocházejících převážně z projektu GNU, který založil Richard Stallman. GNU/Linux je především svobodný, sofistikovaný a univerzální systém.

[3]

Linux ideově vychází z kořenu Unixu, jehož počátky sahají do sedmdesátých let. Přestože s původním Unixem nesdílí ani řádku zdrojového kódu, má stejné aplikační rozhraní i filozofii. Díky tomu je kompatibilní s ostatními Unixy na úrovní zdrojových kódů. Proto má i stejné aplikace a nástroje. Linux podporuje souběžnou práci více uživatelů, z nichž každý může spouštět libovolný počet programů. Linux byl upraven (portován) na spoustu jiných platforem, než je obvyklé PC.

Najdete jej od kapesních počítačů s procesory Arm, přes Macintoshe, pracovní stanice od Sunu až po mainframy od IBM.

[3]

Jestliže v této práci hovořím o Linuxu, myslím tím distribuci, nikoli samotné jádro Linux. Distribuce se skládá nejenom z vlastního jádra, ale také z celé řady dalších aplikací a knihoven.

5.2 Grafické uživatelské prostředí - GUI

Jedná se o uživatelské rozhraní, které zprostředkovává interakci uživatele s aplikací. Uživatel s rozhraním komunikuje pomocí klávesnice a myši. Grafické uživatelské rozhraní se skládá z grafických ovládacích prvků zvaných widgety. Pod pojmem widget si můžeme představit tlačítko, textové pole, posuvník, menu, okno a další. Sada takovýchto widgetů se nazývá grafický nebo formulářový toolkit. Mezi nejznámější patří Qt a GTK+.

5.3 Knihovna Qt

Qt je vyspělý, multiplatformní GUI toolkit napsaný v jazyku C++. Qt je duchovním dítětem Trolltechu, norské společnosti, která vyvíjí, prodává a podporuje Qt a s ním spojený software pro komerční trh. Trolltech široce propaguje snadnou přenositelnost Qt mezi různými platformami, která je nesporně působivá. Qt má nativní podporu pro Linux a systémech založených na Unixu, dále Windows, Mac OS X a dokonce i na přenosných zařízeních, což mu dává jasnou výhodu před jeho konkurenty.

[5]

Speciální verze Qt běží na mobilních telefonech. Jiná verze běží na Sharp Zaurus PDA a podobné platformě. Qt Jambi poskytuje javovskou verzi toolkitu.

[5]

Pro tuto knihovnu typický je způsob ošetření vzniklých událostí. Každý objekt odvozený od třídy QObject dokáže přijímat a vysílat informace v případě vzniklé události. Jestliže některá událost nastane, zdroj této události odešle tzv. SIGNAL všem objektům, které jsou na něj připojené. Vyhodnocení pak proběhne za pomoci tzv. SLOTU, který je umístěn právě v těchto objektech. Tímto způsobem lze snadno a efektivně předávat informace mezi objekty různých typů. Této metody se využívá například při kliknutí myši, stisku tlačítka či jiných událostech.

5.4 Knihovna GTK+

GTK+ přišlo na svět jako část populárního programu GNU Image Manipulation Program, GIMP, odkud získalo také svůj název (The Gimp ToolKit). Programátoři GIMPu byli prozíraví, když vytvořili tento projekt, neboť se rozrostl a rozvinul v jeden z nejschopnějších a nejpopulárnějších toolkitů vůbec.

[5]

GTK+ je celé napsáno v jazyku C, a tak je i většina softwaru založeném na GTK+ vytvořena v C. Naštěstí existuje spousta vazeb do jiných jazyků, což umožňuje používat GTK+ ve spojení s vaším preferovaným jazykem jako C++, Python, PHP, Ruby, Perl, C# nebo Java.

[5]

5.5 Překladač

Překladač převádí zdrojový kód čitelný pro člověka do strojového kódu, který lze následně spustit. Na systému Linux jsou překladače součástí kolekce nástrojů zvané GNU Compiler Collection označované zkratkou GCC. GCC zahrnuje překladače pro C, C++, Java, Objective-C, Fortran a Chill .

[4]

5.6 Debugger

Debugger je program, který se používá v případě, že chcete dohledat, proč se program nechová tak, jak si myslíte, že by měl. GNU Debugger, označován GDB, je debugger používán většinou programátorů v Linuxu. GDB můžete použít ke krokování vašeho zdrojového kódu, nastavení míst přerušení běhu programu, či ke sledování aktuálních hodnot lokálních proměnných.

[4]

5.7 Vývojové prostředí - IDE

IDE je grafické prostředí, které typicky spojuje dohromady pouze některé nebo všechny nástroje potřebné pro vytvoření, ladění a spuštění aplikace. Je vybaveno přinejmenším editorem, správcem souborů a nástrojem pro spuštění aplikace a odchycení jejího výstupu. Kvalitnější prostředí navíc nabízí možnost přidat již předpřipravený soubor se zdrojovým kódem pro různé typy projektů, integraci se systémem správy verzí, a automatické generování dokumentace.

[5]

6 Popis referenčního modelu

Sestavení referenčního modelu hraje klíčovou roli v porovnávání jednotlivých vývojových prostředí a stanovení následného hodnocení. Struktura modelu je znázorněna v příloze práce *A. Schéma referenčního modelu*.

Model se skládá ze čtyř kategorií a každá kategorie z několika vrstev.

6.1 Charakterová kategorie

Charakterová kategorie logicky spojuje vrstvy, týkající se popisu daného prostředí. Většina těchto vrstev není ohodnocena, neboť právě daná vlastnost může být tvůrcem vývojového prostředí žádána. Například si tvůrce může přát podporu pouze jednoho konkrétního programovacího jazyka nebo konkrétní typ použitého kompilátoru. Vrstvy, které nejsou hodnoceny jsou ve schématu označeny šedým písmem. Vrstvy, které jsou hodnoceny jsou hardwarová náročnost (značeno *hw*) a dokumentace. V testování hardwarové náročnosti měřím dobu, po kterou vývojové prostředí nabíhá a množství spotřebované operační paměti bez otevřeného projektu.

Měření jsem provedl na počítači s následující konfigurací:

Operační systém: Linux OpenSUSE 11.1,
GNOME 2.26.0

Hardware: procesor: Intel Pentium IV, 3 GHz
operační paměť: 1 GB
pevný disk: 80 GB, 7 200 ot./min

6.2 Designová kategorie

V Designové kategorii se zaměřuji na testování vizuálních nástrojů spjatých s návrhem aplikace. Hodnotím zde kvalitu návrháře grafického rozhraní a editoru UML.

6.3 Kódová kategorie

Kódová kategorie se zaměřuje na testování nástrojů spjatých s psaním kódu. Mezi ně patří nástroj na dokončování kódu, indikaci chyb vzniklých během psaní, refaktoring a editor kódu.

6.4 Správová kategorie

Kategorie správy se zaměřuje na testování kvality různých správců, jako jsou správci tříd a objektů, projektu nebo pluginů.

7 Výběr vývojových prostředí

Na základě procházení různých diskuzních serverů a výsledků uvedených na serveru viz [8] jsem se nakonec rozhodl pro následující prostředí, která budu mezi sebou porovnávat. Vývojových prostředí je celkem dvanáct. Jsou to tato:

Geany

Geany jsem se rozhodl zahrnout do svého hodnocení, neboť bylo mezi uživateli na diskuzních serverech poměrně dobře hodnoceno a chváleno. Dle seznamu vývojových prostředí publikovaným na serveru viz [8] dosahovalo ke dni 6. 1. 2009 hodnocení celkem 100 % od 15 uživatelů.

Lazarus

Toto vývojové prostředí již nebylo mezi uživateli tak často zmiňováno, což ovšem bylo způsobeno tím, že neumožňuje vývoj aplikací v jazyku C++, nýbrž pouze v Object Pascal. Ke dni 22. 3. 2007 dosahovalo hodnocení 71 % od 7 uživatelů.

CodeBlocks

Mezi uživateli poměrně oblíbené prostředí. Ke dni 18. 9. 2006 dosahovalo hodnocení 100 % od 6 uživatelů.

TheIde

I když je toto prostředí často uváděno v seznamech spolu s ostatními vývojovými prostředími, nenalezl jsem k němu žádné hodnocení ze strany uživatelů, kteří jej používají. Toto prostředí pro framework Ultimate++ dosahuje ke dni 14. 4. 2007 kladného jediného hodnocení pouze od jednoho uživatele.

NetBeans

Jedná se o široce používané a mezi uživateli velmi dobře hodnocené prostředí pro vývoj aplikací zejména v Javě a C++. Na serveru viz [8] je pak ohodnoceno dne 1. 5. 2007 celkem 92 % od 25 uživatelů.

Eclipse CDT

Mezi uživateli na diskuzních serverech bylo toto prostředí hodnoceno stejně jako prostředí NetBeans. Ke dni 6. 12. 2007 dosáhlo od 31 uživatelů hodnocení 97 %.

Gambas

Toto prostředí bylo komentováno na diskuzích jen zřídka, přesto jsem se ho rozhodl do svého testování zahrnout, protože výsledek může být zajímavý pro vývojáře v jazyku Gambas Basic. Ze tří uživatelů ho kladně ohodnotil pouze jeden.

Qt Creator

Přesto, že se jedná o nově vyvinuté prostředí, na diskuzních serverech se již objevili převážně kladné ohlasy. Dne 12. 1. 2009 bylo hodnoceno pouze dvěma uživateli, a to v obou případech kladně.

CodeLite

K tomuto vývojovému prostředí jsem nenalezl žádné diskuze. Ovšem podle toho, že je projekt stále aktivní, je jistě mezi programátory hojně používaný. Na serveru viz [8] jsem k tomuto prostředí nenašel žádný popis.

Anjuta

Anjuta je mezi uživateli velice oblíbená, zejména pak mezi těmi, co programují pod GTK+ a pro GNOME. Hodnocení, kterého dosáhlo na serveru viz [8] je pouze 58 % od 12 hlasujících.

MonoDevelop

Diskuzí k tomuto prostředí jsem příliš nenalezl, což je asi dáno podporou, v Linuxu spíše exotického, jazyku C#. Nalezené diskuze byly pak pozitivního charakteru. Toto prostředí dosáhlo hodnocení 80 % od 10 uživatelů.

KDevelop

Zkušenosti programátorů s používáním tohoto prostředí byly téměř ve všech případech pozitivní. Nejvíce bylo chváleno, jak snadno v něm lze vše nastavit. Na serveru, ze kterého přebírám statistiky, však dosáhlo hodnocení pouhých 80 % od 10 hlasujících. Do testování jsem zahrnul verzi 3.5, i když je k dispozici již verze 4.0. Nejedná se ovšem o stabilní verzi. KDevelop 4 jsem vyzkoušel, ale zatím se jedná pouze o holé prostředí bez dalších nástrojů, a proto jsem jej do svého testování nezahrnul.

8 Srovnání vývojových prostředí

V následující kapitole je provedeno porovnání jednotlivých testovaných vývojových prostředí mezi sebou. Do této chvíle byla porovnávána pouze s referenčním modelem. Srovnání provedu podle stejných kritérií, jako jsem je hodnotil a bude se týkat jednotlivých vrstev referenčního modelu. Vycházím z výsledků dosažených během mého testování prostředí uvedených v příloze *B. Tabulka s hodnocením vývojových prostředí*.

Srovnání dle Platformy

Všechna testovaná vývojová prostředí lze bez problémů nainstalovat na operační systém Linux. Ve velkém množství případů, jestliže bude možné nainstalovat prostředí na Linux, bude možné jej nainstalovat i na Unix. Není to ovšem vždy podmínkou a já vycházím z informací o podpoře na domovských stránkách produktu. Instalaci na Unixu netestuji.

Na operační systém Unix lze s jistotou nainstalovat Gambas, Codelite, KDevelop, Geany, Lazarus, CodeBlocks, NetBeans a TheIDE.

Na Windows lze nainstalovat prostředí Qt Creator, Codelite, Geany, Lazarus, CodeBlocks, TheIDE, NetBeans a Eclipse.

Instalaci na MacOS podporují vývojová prostředí Qt Creator, Codelite, Geany, Lazarus, CodeBlocks, NetBeans a Eclipse.

Srovnání dle licence

Všechna testovaná prostředí jsou šířena pod licencemi svobodného softwaru, což umožňuje uživateli si je stáhnout, nainstalovat a používat zcela zdarma.

Pod licencí GNU GPL jsou šířena vývojová prostředí Gambas, Codelite, Anjuta, MonoDevelop, KDevelop, Geany a CodeBlocks.

Vývojová prostředí šířená pod licencí GNU LGPL jsou Qt Creator a Lazarus.

Prostředí TheIDE je šířeno pod licencí BSD a Eclipse pod vlastní licencí EPL (Eclipse Public License). NetBeans využívá duální licence CDDL a GNU GPL. Uživatel si pak může zvolit licenci, která mu nejvíce vyhovuje.

Srovnání dle data poslední verze

Vývojové prostředí, které je k dispozici s nejstarším datem vydání je CodeBlocks. Nyní se nachází ve verzi 8.02 vydané 28. února 2008. Ze stejného roku pochází také verze vývojového prostředí TheIDE. Jedná se o verzi 2008.1 vydanou dne 6. srpna 2008. Ke konci roku 2008 vydalo KDevelop dne 18. prosince svou zatím poslední stabilní verzi 3.5.4.

Všechna ostatní testovaná vývojová prostředí mají verze vydané již tento rok, to jest 2009. Nejnovější verzi 2.0 uvolnilo vývojové prostředí MonoDevelop. Stalo se tak 30. března. Bohužel, tato verze vyšla až příliš pozdě na to, abych ji do svého výzkumu zahrnul. Vývojové prostředí s nejaktuálnější verzí, které jsem však do svého výzkumu ještě zahrnul, je Lazarus. To se stále nachází ve verzi 0.9.26.2 vydané dne 23.3. 2009.

Srovnání dle programovacího jazyka

Jazyk Pascal:

Podporu pro založení projektu v jazyku Pascal mají z testovaných vývojových prostředí pouze tři a to Lazarus, KDevelop a Geany.

Jazyk Basic:

Jediné prostředí podporující tvorbu v jazyku Basic je Gambas. Prostředí Gambas je určeno pro jazyk Gambas, což je ve své podstatě jazyk Basic s rozšířením o objektový přístup.

Jazyky C a C++:

Jazyky C a C++ jsou podporovány všemi zbývajícími testovanými prostředími.

Jazyk Java:

Vývojovými prostředími podporujícími tvorbu aplikací v jazyce Java jsou bezesporu NetBeans, Anjuta, KDevelop a Geany.

Jazyk Python:

Tvorba aplikace v jazyku Python je podporována v prostředí Geany, Anjuta a KDevelop.

Srovnání dle grafického toolkitu

Vývojové prostředí Lazarus nabízí svůj vlastní grafický toolkit pod názvem LCL. Stejně tak prostředí TheIDE poskytuje vlastní toolkit Ultimate++. NetBeans jako jediné prostředí podporuje vyžívání toolkitu Swing a AWT.

GTK+:

Vytvářet aplikace pod grafickým toolkitem GTK+ nabízejí vývojová prostředí Gambas, Anjuta a KDevelop. Prostředí MonoDevelop pak používá k tvorbě svých aplikací s grafickým uživatelským rozhraním toolkit GTK#. GTK# spojuje toolkit GTK+ s určitými Gnome knihovnami.

Qt:

Grafickou knihovnu Qt podporují prostředí Gambas, Qt Creator a KDevelop. Qt Creator vyžaduje verzi Qt 4. KDevelop 3.5.4 verzi Qt 3.2 a vyšší, avšak nižší než Qt 4.

WxWidgets:

Tento grafický toolkit podporují pouze dvě testovaná vývojová prostředí, a to Anjuta a KDevelop.

Srovnání dle kompilátoru

Všechna testovaná prostředí využívají pro překlad projektu v jazyku C a C++ překladač GCC. Prostředí Gambas je vybaveno vlastním překladačem Gbc. Stejně tak prostředí Lazarus používá vlastní překladač FPC. MonoDevelop nabízí pro překlad aplikací překladače Mcs a Gmcs a prostředí NetBeans překladač založeném na Javac. CodeBlocks a Codelite dovolují použít i překladače MSVC++ pro platformu Windows. CodeBlocks navíc podporuje celou řadu dalších překladačů.

Srovnání dle debuggeru

Jediné z testovaných vývojových prostředí bez nástroje pro ladění aplikace je Geany. Vlastním nástrojem pro ladění je vybaveno prostředí Gambas. Všechna ostatní prostředí podporují debugger GDB.

MonoDevelop podporuje navíc ještě MDB, ovšem typ debuggeru musí být zvolen již při instalaci. Pro ladění aplikací napsaných v jazyce Java, má k dispozici prostředí NetBeans ladící nástroj Jdb. CodeBlocks umožňuje ladit aplikace pod Windows za pomoci nástroje MS CDB a Eclipse pomocí Cygwin a MinGW.

Srovnání dle hardwarové náročnosti

Majitelé slabšího hardwaru jistě ocení vývojové prostředí s menšími nároky. Z výsledků plynoucích z mého testování můžu jednoznačně říci, že mezi taková bezesporu patří Geany a Gambas. Tyto zmiňované mají

velice nízké nároky na paměť a dokáží naběhnout prakticky okamžitě. Ovšem na rozsáhlejší projekty bych tato vývojová prostředí nevolil. Gambas podporuje pouze jediný programovací jazyk a to jazyk založený na Basicu – Gambas. Naproti tomu Geany má podporu pro mnoho jazyků, bohužel nedisponuje dalšími důležitými prvky, jako jsou GUI návrhář, debugger či podporou databází. Dalšími hardwarově nenáročnými vývojovými prostředími jsou Anjuta, a což mě kvůli své vysoké kvalitě překvapilo, i KDevelop 3.

Na druhou stranu velice hardwarově náročnými prostředími jsou pak ta, jež běží na javovském virtuálním stroji. Jedná se o NetBeans a Eclipse. Obě vyžadují až příliš mnoho systémových prostředků, zejména paměti. Doba jejich načítání je ze všech testovaných nejdelší. To je dáno samozřejmě spuštěním virtuálního stroje.

Srovnání dle kvality dokumentace

Co se obecně ohledně dokumentací a mých zjištění týče, valná většina testovaných prostředí má k dispozici na svých domovských stránkách velice kvalitní dokumentaci. Lze jenom těžko říci, která z nich je nejlepší.

Nejhoršího hodnocení dosáhlo MonoDevelop. Nikoliv co se kvality dokumentace týče, ale kvůli tomu, že není ještě zcela dokončena a nachází se stále ve výstavbě.

Nejrozsáhleji a nejkvalitněji zpracovanou dokumentací disponují bezesporu NetBeans, Eclipse a KDevelop. Ve všech zmíněných lze vyhledávat. U KDevelop a Eclipse je dokumentace dokonce přímo integrována do vývojového prostředí.

Srovnání dle kvality GUI návrháře

Ne všechna testovaná vývojová prostředí byla vybavena návrhářem grafického uživatelského rozhraní. Takovým příkladem jsou Geany, Eclipse CDT, CodeBlocks a CodeLite.

Testované vývojové prostředí s nejhorším GUI designérem bylo Anjuta. Největším nedostatkem tohoto designéru bylo to, že postrádá možnost měnit rozměry widgetů. Jejich rozměr můžeme změnit pouze zadáním číselné hodnoty do korespondujících políček správce vlastností.

Naproti tomu, asi po všech stránkách nejlepším návrhářem, disponuje vývojové prostředí Qt Creator. Navrhnout i složitější schéma uživatelského rozhraní s velkým množstvím widgetů různorodě na formuláři zarovnaných je otázkou chvíle. Zvláště pak oceňuji snadnou správu signálů a slotů. Neméně kvalitním designérem je také vybaveno KDevelop. To, stejně jako Qt Creator, návrh aplikace uživateli značně usnadňuje a v porovnání s ostatními má k dispozici velké množství widgetů. Vývojové prostředí s největším množstvím widgetů je Lazarus. Lazarus umožňuje uživateli nastavit jednotlivým widgetům nejširší škálu vlastností ze všech testovaných.

Za zmínku stojí také GUI návrhář vývojového prostředí pro platformu Ultimate++ TheIDE. Ohodnocené je jako průměrné a není esteticky příliš působivé, naproti tomu však vyniká svou jednoduchostí, rychlostí návrhu a schopností efektivně widgety zarovnávat.

Srovnání dle kvality UML editoru

Při testování vývojových prostředí pod Linuxem mě poněkud překvapil fakt, jak málo jich je vybaveno editorem UML modelu. Dokonce i prostředí dosahující vysokých kvalit jako Qt Creator, MonoDevelop a Lazarus nemají tento nástroj k dispozici. Jedinými vývojovými prostředími z testovaných, vybavených tímto nástrojem, jsou pouze KDevelop a Anjuta. Žádný z nich však bohužel ani z části nedosahuje kvalit a schopností UML editoru, který má v sobě obsaženo vývojové prostředí Visual Studio na konkurenční platformě Windows.

U KDevelop můžeme dokonce UML schéma pouze prohlížet a není možné do jeho struktury jakýmkoliv způsobem zasáhnout.

Anjuta nám nabízí o něco kvalitnější nástroj, avšak ani skrze něj není možné přidávat další atributy do tříd a ani není možné z tohoto UML schéma vygenerovat zdrojový kód.

Řešením absence nástroje pro editaci UML modelu může být následné doinstalování odpovídajících pluginů, které pro některá testovaná prostředí již existují.

Srovnání dle kvality správce databází

Situací obdobnou jako v předešlém srovnání je i to, že pouhý zlomek testovaných vývojových prostředí je vybaveno nástrojem pro správu databází. Jsou pouhá tři taková a to MonoDevelop, Gambas a NetBeans. Všechna zmiňovaná mají podporu pro databázové systémy MySQL a PostgreSQL. Gambas pak přidává podporu i pro Firebird a SQLite, a MonoDevelop pro MS SQL a, stejně jako Gambas, i SQLite. NetBeans podporují pouze Java DB, MySQL a PostgreSQL.

Nejlepšího správce databází z testovaných vývojových prostředí má NetBeans. Na rozdíl od ostatních je možné celou strukturu tabulky i dat uvnitř vyexportovat do binárního souboru a případně znovu načíst. To se uplatní zejména při zálohování nebo přenosu databází. Všichni testovaní správci databází dovolují základní operace jak se strukturou tabulek, tak s jejich daty. Co však u všech správců postrádám je například vyhledávání v připojených databázích.

Ostatní vývojová prostředí bez správců musejí k databázím přistupovat jiným způsobem. Buď pomocí zdrojového kódu a připojených knihoven nebo pomocí speciálních widgetů v designéru GUI. Dobrým příkladem takovéhoto přístupu je Qt Creator, který k databázovým systémům přistupuje pomocí GUI vrstvy modulu SQL v knihovně Qt 4.

Srovnání dle kvality editoru zdrojového kódu

Co se srovnání kvality editorů zdrojového kódu týče, všechna testovaná vývojová prostředí dosáhla alespoň průměrného hodnocení. Nejdůležitější požadavek referenčního modelu zvýrazňovat syntaxi jazyka a jeho klíčových slov splňují všechna a všechna také zvýrazňují párové ukončovací značky.

Mezi vývojová prostředí s nejhorším editorem se řadí Gambas a TheIDE. Editor v prostředí Gambas nedokáže skrývat bloky kódu a v TheIDE nedovede přecházet na deklaraci vybrané proměnné. Oba dva pak nezvýrazňují proměnné definované uživatelem a nejsou schopny vyznačit jejich další reference v kódu.

Naopak prostředí, která kvalitou svého editoru zdrojového kódu ostatní značně převyšují jsou NetBeans, KDEvelop, Eclipse, Qt Creator a MonoDevelop. Poslední tři zmiňované dokonce zobrazují i popis proměnné pod kurzorem.

Srovnání dle dokončování kódu

Všechna testovaná vývojová prostředí mají k dispozici nástroj pro automatické dokončování kódu. Téměř polovina z nich dosáhla špatného hodnocení. Jedním z nejhorších je Gambas, který jediné co dokáže je nabízet u funkcí argumenty a vyznačit z nich právě ten editovaný. Podobně je na tom i Geany, která nabízí jen názvy bez určení datového typu.

NetBeans, Eclipse a MonoDevelop pomáhají dokončit kód ze všech nejlépe a požadavky referenčního modelu splňují všechny. Velice kvalitním nástrojem pro dokončování kódu je vybaven i Qt Creator, který postrádá jen popis funkce nabízené metody.

Srovnání dle indikace chyb

Více než polovina testovaných vývojových prostředí nemá žádnou kontrolu chyb během editace zdrojového kódu. Mezi vývojová prostředí s průměrným hodnocením tohoto nástroje patří Qt Creator, MonoDevelop, KDevelop a Eclipse. Eclipse se po stránce úpravy kódu svým chováním velice podobá NetBeans, avšak jeho nástroj pro indikaci chyb nedosahuje stejné kvality. Všechna zmíněná postrádají detekci neznámých symbolů.

Jako jediné vývojové prostředí, které disponuje i detekcí neznámých symbolů a dokonce rozpozná i přiřazení hodnoty nesprávného datového typu, je NetBeans. To splňuje všechny nároky referenčního modelu na detekci chyb vzniklých již během editace kódu.

Srovnání dle refaktoringu

Nástroji pro refaktoring stávajícího kódu je vybaveno pouze šest z testovaných. Nejméně možností jak vylepšit kód nabízí vývojové prostředí Codelite. Celkem nabízí pouze přejmenování proměnné, zapouzdření atributů tříd a implementace metod.

Naproti tomu nejširším spektrem jak zastaralý kód upravit je vybaveno vývojové prostředí NetBeans. Nabízí až osmnáct různých možností od přejmenování proměnné, zapouzdření po extrakci rozhraní ze třídy nebo změnu atributů metody.

Srovnání dle testování tříd

Podporou pro testování tříd jsou vybavena pouze vývojová prostředí Lazarus, NetBeans, Codelite a MonoDevelop. Lazarus a Codelite dovolí pouze do projektu přidat speciální třídu pro testování, ovšem již nezajistí žádnou správu vytvořených testů ani nezobrazí jejich výslednou statistiku úspěšnosti. NetBeans dokáže testy spravovat a spouštět vybrané. K dispozici je pak přehled o úspěšnosti jednotlivých testů. MonoDevelop dokáže dokonce výsledky testů vizuálně zobrazit do grafu.

Srovnání dle pluginů

U všech testovaných vývojových prostředí je možné rozšířit jejich vlastnosti pomocí zásuvných modulů – pluginů. Jediná dvě prostředí nedisponovala žádným správcem pluginů, ani možností pluginy nastavit. Jsou to Qt Creator a TheIDE. Nové pluginy lze přidat pouze umístěním do určitého místa v systému souborů. V Qt Creatoru pak máme k dispozici alespoň náhled na nainstalované a zavedené pluginy.

Kvalitními správci pluginů jsou pak vybavena prostředí NetBeans, Eclipse, Anjuta, MonoDevelop a KDevelop. MonoDevelop sice nabízí pohodlné přidání pluginů, ovšem k dispozici jich má v repozitáři malé

množství. KDevelop nemá možnost stahovat pluginy přímo z repozitáře na webu, ovšem jsou vyvíjeni komunitou a lze je získat z různých míst na internetu. Anjuta má kvalitního správce rozšíření, samotných pluginů pro toto vývojové prostředí je málo. Nejlepšími správci rozšíření disponují NetBeans a Eclipse. Za pomoci jejich správců můžeme pluginy snadno spravovat, získávat další rozšíření nebo aktualizovat již nainstalovaná. Pluginů pro tato prostředí jsou desítky.

Srovnání dle správce projektu

Vývojové prostředí TheIDE a Lazarus tento nástroj zcela postrádají. Nejhoršími správci projektu jsou vybavena vývojová prostředí Geany a Gambas. Ani jeden z nich neumožňuje žádnou pokročilejší správu projektu. Projekt lze jen založit a přidávat do něj soubory, případně je odebírat.

Zbylá vývojová prostředí, až na Codelite, mají správce projektu na velmi dobré úrovni. Prostředí Codelite nemá průvodce založením nového projektu. Nejlepšími správci jsou pak vybaveny NetBeans, Eclipse a CodeBlocks. Všechna tato jmenovaná prostředí umožňují spravovat více projektů najednou a svými schopnostmi jsou si rovnocenná. Vlastnosti prostředí Anjuta, MonoDevelop a KDevelop se liší od předešlých tím, že dokáží spravovat projekt vždy jen jeden.

Srovnání dle správce tříd a objektů

Všechna testovaná vývojová prostředí jsou vybavena správcem objektů a tříd. Prostředí s hodnocením *špatné* spojuje jeden fakt, a to, že jejich správce dokáže pouze objekty a třídy v projektu zobrazit. Jakékoliv manipulace s nimi jsou nad rámec jeho schopností.

Hodnocení *výborné* dosáhla pouze dvě vývojová prostředí a to KDevelop a NetBeans. Obě dvě se od ostatních liší hlavně v tom, že umožňují určitou manipulaci s objekty, jako například přidání metody nebo atributů do třídy, přejmenování, extrahování rozhraní atd. KDevelop umožňuje pouze jmenované, kdežto u NetBeansu lze uplatnit na objekty ve správci všechny dostupné možnosti refaktoringu.

9 Srovnání tvorby aplikace ve Windows a Linuxu

Ještě než jsem započal práci na této bakalářské práci, měl jsem již jisté zkušenosti s prací ve vývojových prostředí v operačním systému Windows. Díky těmto zkušenostem a zkušenostem získaným při testování vývojových prostředí pod Linuxem, mohu porovnat rozdíly při tvorbě aplikací na těchto platformách.

Rozdíly, ke kterým jsem dospěl, nyní shrnu v několika bodech.

9.1 Fundamentální rozdíl

Fundamentálním rozdílem mezi tvorbou aplikace pod Windows a Linuxem je způsob, kterým operační systém nahlíží na procesy, vlákna, události a další záležitosti, týkající se systému. Přestože bychom tu samou aplikaci vyvíjeli ve stejném programovacím jazyku na Windows a Linuxu, její zdrojový kód se bude v určitých částech lišit. Budeme-li například chtít spustit v aplikaci nový proces, na systému Windows použijeme jeden příkaz *CreateProcess()*. Na systému Linux je postup o něco složitější. Nejprve se vytvoří kopie současného běžícího procesu pomocí příkazu *fork()* a následně se tento proces přepíše programem načteném z disku pomocí příkazu *exec()*. Dalších obdobných rozdílů je celá řada.

9.2 Obliba negrafických nástrojů pro vývoj

Ačkoliv je v dnešní době obrovské množství grafických nástrojů pro tvorbu aplikací v Linuxu, tato práce je toho dokladem, přesto stále široké množství programátorů využívá pro vývoj negrafických nástrojů v terminálu. Je to dáno kvalitou konzolových textových editorů jako Emacs a Vim, které lze s pomocí klávesových zkratek ovládat stejně rychle jako editory grafické.

Na systému Windows je takovýto efektivní způsob tvorby aplikace pomocí příkazového řádku nemyslitelný.

9.3 Možnost předat parametry aplikaci

U linuxových aplikací, i když jsou s grafickým rozhraním, se očekává možnost spuštění z okna terminálu a přidání argumentů. Programátor by měl při tvorbě aplikace pro systém Linux s touto možností počítat.

Aplikace vyvíjené pro systém Windows takovýto přístup ve většině případů nevyžadují.

9.4 Využití svobodného softwaru

Pro Linux je k dispozici velké množství svobodného softwaru. Díky tomu má programátor možnost studovat funkci nějakého programu či jej jakýmkoliv způsobem modifikovat. Vývoj aplikací se pak značně urychlí a usnadní. Takovýto počin má však také své restrikce.

Vývojář pro Windows nemá takové možnosti v přístupu ke zdrojovému kódu aplikací pro tento systém.

Co se rozdílů práce ve vývojových prostředí týče, jedná se o tyto:

9.5 Rozvržení layoutu na formuláři

Pod Linuxem se mnohem častěji při návrhu GUI aplikace užívají pro rozmístění widgetů na formuláři různé kontejnery. Tyto kontejnery pak sami kontrolují chování prvků při změnách rozměrů formuláře.

Naproti tomu ve Windows se používá způsob, kdy je každý widget přichycen k některému z okrajů formuláře. Změna rozměrů formuláře pak ovlivní i rozměr widgetů.

Dle mého názoru je první způsob výhodnější a rychleji aplikovatelný než způsob druhý, kdy se musí každému widgetu určovat takovéto vlastnosti.

9.6 Způsob publikování aplikace

Další zásadní rozdíl je ve způsobu publikování hotové aplikace. Ve Windows je nejčastější způsob předání hotové aplikace buď přímo ve formě binárního spustitelného souboru nebo ve formě instalačního balíčku.

Vývojová prostředí v Linuxu také nabízejí zmiňované možnosti, ovšem mnohem častější je způsob předání ve formě zdrojového kódu a souborů určené pro automatický překlad. Uživatel si pak aplikaci

na svém systému sám přeloží a nainstaluje. Tato možnost je zejména výhodná díky tomu, že se pro překlad využijí knihovny v systému a není proto nutné je šířit spolu s aplikací.

9.7 Volba grafických toolkitů

Jestliže chceme vytvořit aplikaci pro Linux s grafickým rozhraním, musíme se rozhodnout, jaký grafický toolkit pro vykreslování widgetů použijeme. Může to být například GTK+ nebo Qt. V Linuxu máme mnohdy v rámci jednoho vývojového prostředí podporu pro více toolkitů (ne všechna jsou podporována návrhářem rozhraní). Toolkity jsou pak spjaty s určitými nainstalovanými knihovnami v systému. Jestliže se tyto knihovny nalézají například i ve Windows, lze tuto aplikaci zde přeložit a spustit.

Ve Windows vývojová prostředí nejčastěji využívají platformně závislé grafické toolkity jako MFC, WinForms nebo WPF. Ovšem i zde je možnost vytvořit aplikaci s platformně nezávislými prvky, avšak musí být ve Windows nainstalovány potřebné knihovny a nástroje.

Ve většině distribucí Linuxu se již některá platformně nezávislá grafická knihovna nachází. Rozdíl tedy mezi tvorbou aplikací ve Windows a Linuxu je ten, že v Linuxu není grafický toolkit úzce spjat s vývojovým prostředím a máme větší možnost si ho zvolit než ve Windows.

10 Vývoj konečné aplikace

V následující kapitole jsou popsány důvody, které mne vedly při výběru vhodného vývojového prostředí pro tvorbu konečné aplikace. Dále pak popisuji jak jsem postupoval a jakým problémům jsem při tvorbě čelil.

10.1 Volba vývojového prostředí

Ze všech testovaných vývojových prostředí jsem si nakonec vybral Qt Creator. Během testování si ve srovnání s ostatními vedlo celkem dobře. Jedinou nevýhodou tohoto prostředí je, že není vybaveno žádným nástrojem pro refaktoring stávajícího kódu a ani editorem UML modelu. Díky tomu není vhodné na rozsáhlejší projekty.

Vybral jsem si jej zejména kvůli jeho kvalitnímu textovému editoru a nástroji na dokončování textu. Toto vývojové prostředí také disponuje nejlepším GUI návrhářem ze všech testovaných. Qt také umožňuje tvorbu multiplatformních aplikací, které lze přeložit pro více operačních systémů.

Práce v Qt Creatoru je velice pohodlná, rychlá a díky režimům zobrazení i přehledná. Uživatel může během práce mezi jednotlivými režimy přepínat a věnovat se tak naráz editaci zdrojového kódu, správě projektů nebo listovat v manuálu.

Dalším důvodem proč jsem si vybral právě toto prostředí je skutečnost, že mne zajímalo jak se uplatní při tvorbě o něco náročnější aplikace nežli byly testované příklady a zda se jeví slibně do budoucna.

10.2 Popis aplikace

Konečnou aplikací, kterou jsem se rozhodl ve vybraném prostředí vytvořit, byl jednoduchý přehrávač multimédií.

Důvodem proč jsem se rozhodl zrovna pro přehrávač je, abych demonstroval, že i na Linuxu je možné jednoduše vytvořit tak složitou aplikaci jako je přehrávač médií. Pro přehrávání médií využiji již hotovou multimediální knihovnu, což celý vývoj značně usnadní. Jelikož jsem se rozhodl použít Qt Creator, který má podporu pouze pro programovací jazyk C++, je celá aplikace napsána právě v něm.

Od tohoto přehrávače se očekává, že bude schopen přehrát jak hudební soubor, tak soubor videa. Běh přehrávání bude možné řídit pomocí ovládacích prvků jako jsou tlačítka a položky v menu, ale také pomocí posuvníku. Hlasitost výstupního zvuku a parametry obrazu bude možné nastavit. Přehrávač bude během reprodukce zobrazovat uplynulý čas a celkový čas trvání skladby či videa. Seznam vybraných skladeb bude moci upravovat, to jest mazat a přidávat soubory nové.

10.3 Postup vývoje a vzniklé problémy

Prvním krokem pro nového uživatele nejspíše bude instalace vývojového prostředí Qt Creator. Protože toto prostředí běží na Qt 4, je nutné ho nejprve do systému nainstalovat. Jestliže chceme vyvíjet aplikace, což v tomto případě chceme, je potřeba zvolit verzi Qt SDK. Prostředí Qt Creator je již v tomto vývojářském balíku obsaženo.

Instalaci je možno stáhnout z domovských stránek na následujícím odkazu viz [9].

Při vývoji aplikace jsem spolupracoval s referenční dokumentací programového rozhraní knihovny Qt, která je k dispozici na adrese viz [10].

10.3.1 Použití multimediální knihovny

Pro přehrávání multimédií jsem se rozhodl použít knihovnu Phonon. S využitím této knihovny lze přehrávat soubory za pomoci krátkého úseku zdrojového kódu. Do systému je nutné doinstalovat vývojářský balíček.

Důležitými částmi této knihovny jsou prvky MediaObject, VideoWidget a AudioOutput. VideoWidget představuje prvek, na který se promítá obraz. AudioOutput reprodukuje zvuk a MediaObject zajišťuje ovládání těchto prvků. Jediný VideoWidget je vizuálním prvkem.

10.3.2 Návrh vzhledu GUI

Po spuštění založíme nový projekt typu Qt4 GUI Application. První věcí, kterou jsem začal ihned po založení nového projektu, byl návrh vzhledu uživatelského rozhraní aplikace. Ukázka vzhledu se nachází v příloze *C. Ukázka konečné aplikace*. K rozmístění widgetů na formuláři používám několik druhů layoutu. Centrální widget zarovnává prvky horizontálně a ovládací tlačítka jsou zarovnány do mřížky. Tímto způsobem se layout nerozpadne ani při změně rozměrů okna.

10.3.3 Problém s podporou prvků Phononu

Bohužel návrhář GUI zvoleného vývojového prostředí není vybaven podporou pro přidávání widgetů z knihovny Phonon. Řešení jak přidat prvek VideoWidget na formulář aplikace je několik.

Prvek můžeme vytvořit dynamicky za běhu aplikace. Nevýhodou tohoto řešení je, že při návrhu rozhraní tento prvek nebude k dispozici. Vzniklé volné místo v layoutu vyplní ostatní widgety a vzhled aplikace po spuštění bude jiný než v návrháři.

Já jsem zvolil řešení takové, abych mohl nadále návrhář GUI používat a přitom rozmístění prvků zůstalo stejné. Namísto prvku VideoWidget jsem v návrháři použil základní komponentu QWidget. Od této vizuální komponenty jsou všechny ostatní odvozeny. Tuto komponentu mohu již na formulář přidat a měnit libovolně její rozměry nebo pozici. V hlavičkovém souboru s deklarací prvků formuláře, v našem případě v souboru *ui_mainwindow.h*, pak přepíši datový typ prvku z QWidget na VideoWidget. Nevýhodou tohoto řešení je, že při každé změně vzhledu formuláře se změněný datový typ přepíše zpět.

10.3.4 Problém objektově orientovaného přístupu

Při vývoji konečné aplikace se snažím zachovat objektově orientovaný přístup. Z tohoto důvodu vytvářím novou třídu v novém souboru, která má za úkol pouze samotné přehrávání multimédií. Třída se nazývá

CorePlayer. Ostatní třídy jí mohou ovládat pouze prostřednictvím volání veřejných metod.

Použití objektového přístupu však přináší problém, kdy třída *CorePlayer* nedokáže přistupovat k prvkům formuláře, které se nacházejí v třídě jiné. Přístup k nim mít musí, aby bylo možné například zapisovat odehraný čas skladby do stavové lišty hlavního okna.

Problém lze vyřešit předáním ukazatele na stavovou lištu při volání konstruktoru třídy *CorePlayer*. Já se ale rozhodl pro mnohem účinnější řešení. Využiji efektivní způsob mezi-objektové komunikace v Qt. Princip byl již popsán v teoretické části práce. Prvek MediaObject emituje při přehrávání v pravidelném intervalu aktuální hodnotu časové pozice. K této události připojím vlastní signál s názvem *timeChanged()*, který je pak připojen ke slotu ve třídě hlavního okna aplikace. Tímto způsobem byla předána informace o změně časové pozice přehrávaného souboru ze třídy *CorePlayer* do třídy hlavního okna, která již má přístup k textovému návěstí ve stavovém panelu okna. Ukázka zdrojového kódu popsaného způsobu je umístěna v příloze
D. *Mezi-objektová komunikace*.

Obdobným způsobem je řešeno i předávání hodnot posuvníků z dialogového okna pro nastavení obrazu videa do hlavního okna aplikace. V tomto případě je emitován signál při změně hodnoty posuvníku, který je následně vyhodnocen v okně hlavní aplikace.

10.4 Vyhodnocení

Jako název konečné aplikace jsem použil rekurzivní zkratku, což je v Linuxu častý způsob pojmenování, APE. Význam této zkratky je APE Plays Everything. V překladu to znamená APE přehraje všechno. Ape v angličtině také znamená opice.

Požadavky, které byly kladeny v kapitole *10.2 Popis aplikace* byly všechny splněny. I když se tvorba s využitím knihovny Qt v některých aspektech liší od tvorby ve Windows, není problémem nový přístup pochopit a naučit se jej používat. Rozdíly tvorby v obou operačních systémech byly již popsány v kapitole *9. Srovnání tvorby aplikace ve Windows a Linuxu*. Knihovna Qt je dobře zdokumentována a obsahuje i krátké ukázky kódu s popisem.

Přehrávač jsem otestoval na platformě Linux. Překlad na platformě Windows XP se mi nepodařil kvůli problémům s multimediální knihovnou Phonon, která není pro tuto platformu nativní a nachází se stále ještě ve vývoji. Postupoval jsem dle návodu viz [11] s použitím SDK Visual C++ 2008. Překlad skončí neúspěchem bez popisu konkrétní příčiny.

Na Linuxu bezproblémově přehrává jak video tak audio soubory. Pro přehrání některých typů souborů je potřeba mít nainstalovány v systému odpovídající kodeky, na což jsme ovšem upozorněni. Ovládání přehrávače také funguje správně.

Výběr vývojového prostředí Qt Creator byl správný. Dle mého názoru, volbou kteréhokoliv jiného z testovaných prostředí, bych nedosáhl požadovaných výsledků v tak rozumném čase s obecnou znalostí programování, jako tomu bylo v případě Qt Creatoru.

Všechny zdrojové kódy projektu jsou umístěny na přiloženém médiu v adresáři */konecna_aplikace/prehravac/*.

11 Závěr

Hlavním cílem této práce bylo podat přehled současných vývojových prostředí pro Linux a porovnat je. V práci jsem se zaměřil na vývojová prostředí, ve kterých je možné vytvořit binární spustitelnou aplikaci. Při výběru prostředí jsem dbal na jejich aktuálnost. Z tohoto důvodu jsem musel svůj výzkum vždy s každou vydanou novou verzí přehodnotit. Uživateli se tak dostává do rukou aktuální seznam prostředí. Nevýhodou může být fakt, že stále vychází nové verze programů s novými vlastnostmi a za pár měsíců bude tento výzkum již neaktuální.

Výhodou mnou zvoleného postupu při hodnocení prostředí je, že díky vícestupňové hodnotící metodě lze snadno zjistit charakteristiku každého z nich. Jelikož je tabulka s hodnocením seřazená podle dosažených výsledků, je patrné, která z testovaných prostředí jsou nejkvalitnější. Programátor si může snadno s pomocí tabulky vybrat prostředí, které mu nejvíce vyhovuje. Tato práce může také posloužit tomu, kdo by chtěl v mém výzkumu pokračovat a využít pro srovnávání použitý referenční model. Dle mého názoru byl cíl podat přehled prostředí a porovnat je splněn.

Dalším cílem práce bylo porovnat, v čem se liší tvorba aplikace v Linuxu a ve Windows. Během svého výzkumu jsem odhalil celkem sedm kategorií odlišnosti tvorby. Na odhalení dalších rozdílů by bylo potřeba daleko rozsáhlejšího a dlouhodobého výzkumu na složitějších

aplikacích. Přesto se domnívám, že cíl přednést alespoň ty nejzákladnější rozdíly, splněn byl.

Posledním cílem, který měla má práce splnit byla demonstrace, že i na Linuxu lze pomocí kvalitního vývojového prostředí velice rychle a snadno odladit graficky uživatelsky přívětivou aplikaci. Nestudoval jsem žádný manuál k vybranému prostředí, ani jsem hluboce nestudoval programovací jazyk C++, přesto se mi podařilo takovouto aplikaci vytvořit. Postup, který jsem zvolil je popsán a může uživateli pomoci s tvorbou jeho vlastní aplikace založené na knihovně Qt.

Přehrávač jsem odzkoušel a je zcela funkční. Jediné, co se mi nepodařilo, bylo portovat ho na platformu Windows. Zde byl problém s použitou multimediální knihovnou. Přenést ji na jinou platformu ovšem nebylo cílem práce. Byl jsem jen toho názoru, že by to tuto práci mohlo posunout výše.

12 Přehled literatury

[1] *Přehled software pro Linux v kategorii vývojové prostředí* [online]. c2003-2009 [cit. 2009-03-23]. Česky. Dostupný z WWW: <http://www.linuxsoft.cz/sw_list.php?id_kategory=112>. ISSN 1801-3805.

[2] Comparison of integrated development environments. *Wikipedia, The Free Encyclopedia* [online]. last revision 19 April 2009 [cit. 2009-03-22]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments>.

[3] *Učebnice ABC/Linuxu*. [online]. ze dne 19.10.2006 [cit. 2009-04-10]. Dostupný z WWW: <www.abclinuxu.cz/download/ucebnice_abc_linuxu-20061019.pdf>.

[4] MITCHELL, Mark, OLDHAM, Jeffrey, SAMUEL, Alex. *Advanced Linux Programming*. 1st edition. Indianapolis : New Riders Publishing, c2001. xxiii, 340 s. Dostupný z WWW: <http://www.advancedlinuxprogramming.com/alp-folder>. ISBN 0-7357-1043-0.

[5] MATTHEW, Neil, STONES, Richard. *Beginning Linux® Programming*. 4th edition. Indianapolis : Wiley Publishing, c2008. xxx, 759 s. ISBN 978-0-470-14762-7.

[6] Root.cz : informace nejen ze světa Linuxu [online]. [Praha] : 4WeB, c1998-2009 [cit. 2008-01-29]. Dostupný z WWW: <http://www.root.cz/>. ISSN 1212-8309.

[7] AbcLinuxu [online]. Stickfish, c1999-2009 [cit. 2008-03-29]. Dostupný z WWW: <http://www.abclinuxu.cz/>. ISSN 1214-1267.

[8] IDE. *AbcLinuxu* [online]. 2009 [cit. 2009-04-08]. Dostupný z WWW: <http://www.abclinuxu.cz/software/programovani/ide>. ISSN 1214-1267.

[9] *Downloads : Qt – A cross-platform application and UI framework* [online]. Finland : Nokia Corporation, c2008-2009 [cit. 2009-04-17]. Dostupný z WWW: <http://www.qtsoftware.com/downloads>.

[10] *Qt 4.4.3 : Qt's Classes* [online]. Nokia Corporation, c2008 [cit. 2009-04-18]. Dostupný z WWW: <http://doc.trolltech.com/4.4/classes.html>.

[11] *Qt 4.4.3 : Phonon Overview* [online]. Nokia Corporation, c2008 [cit. 2009-04-18]. Dostupný z WWW: <http://doc.trolltech.com/4.4/phonon-overview.html>.

13 Seznam příloh

A. Schéma referenčního modelu

B. Tabulka s hodnocením vývojových prostředí

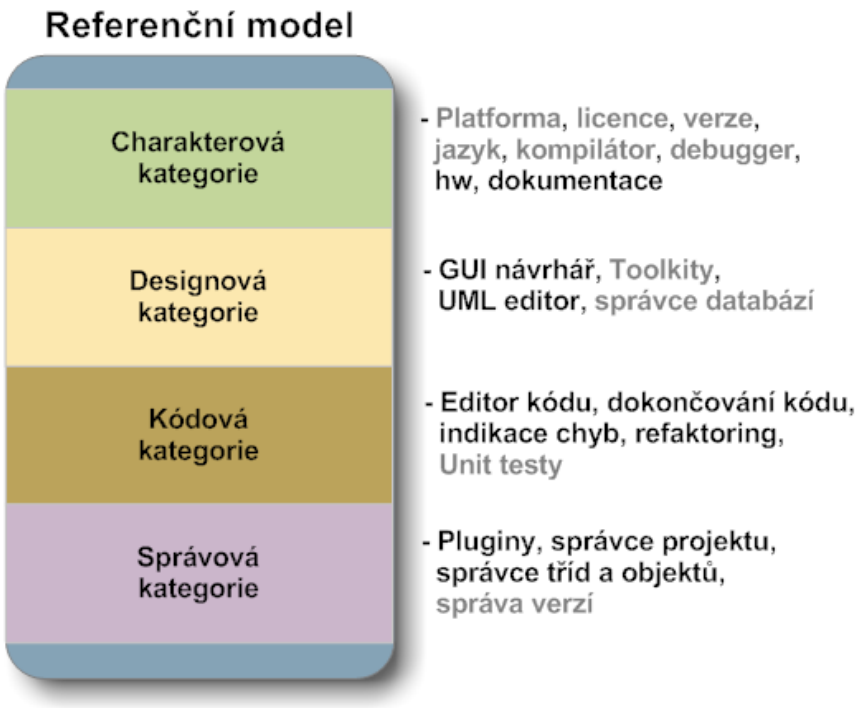
C. Ukázka konečné aplikace

D. Mezi-objektová komunikace

E. Adresářová struktura přiloženého média

14 Přílohy

A. Schéma referenčního modelu



B. Tabulka s hodnocením vývojových prostředí

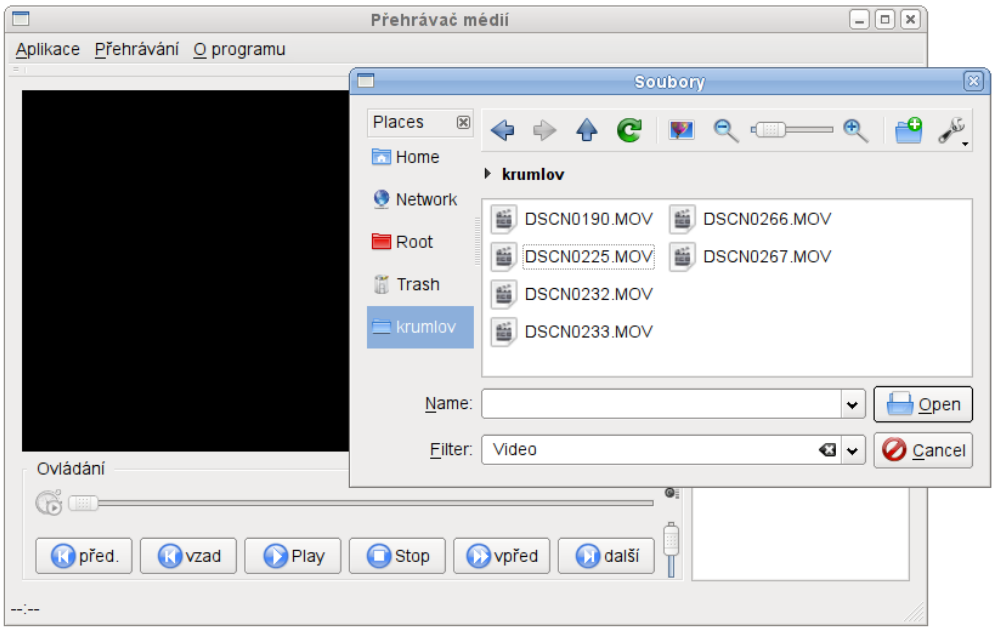
Název	NetBeans	KDevelop3	MonoDevelop	Eclipse CDT	Qt Creator	Anjuta	Lazarus	Code::Blocks
Platforma	Linux, MacOS, Windows, Unix	Linux, Unix	Linux	Linux, MacOS, Windows	Linux, MacOS, Windows	Linux	Linux, Unix, MacOS, Windows	Linux, Unix, MacOS, Windows
Licence	GNU GPL, CDDL	GNU GPL	GNU GPL	EPL	GNU LGPL	GNU GPL	GNU LGPL/GPL	GNU GPL
Poslední verze datum uvolnění	6.5 10.11. 2008	3.5.4 18. 12. 2008	2.0 Beta 2 16. 3. 2009	5.0.2 5.3. 2009	1.0 3. 3. 2009	2.26.0 16. 3. 2009	0.9.26.2 23. 3. 2009	8.02 28. 2. 2008
Prog. jazyk	C, C++, PHP, Java, Ruby, Groovy, ...	C, C++, Java, Pascal, Fortran, Python, Perl, PHP, Ruby	C, C++, C# a CIL	C, C++	C, C++	C, C++, Java, Python, Vala	Object Pascal	C, C++
Kompilátor	Javac, GCC	GCC, Fortran77	Mcs, Gmcs	GCC	GCC	GCC	FPC	GCC, MSVC++, ...
Debugger	Jdb, GDB,	GDB	MDB, GDB	GDB, Cygwin, MinGW	GDB	GDB	GDB	GDB, MS CDB
Paměť, načítání	90 MB, < 15 s	10,2 MB , <3 s	28,4 MB , < 8 s	190 MB, < 30 s	16 MB, < 3 s	10,1 MB, < 3 s	13,7 MB, < 3 s	22,8 MB, < 8 s
Dokumentace	výborné	výborné	špatné	výborné	průměrné	průměrné	výborné	průměrné
GUI návrhář	výborné	výborné	průměrné	ne	výborné	špatné	výborné	ne
Toolkity	Swing, AWT	Qt, GTK+, wxWidgets	GTK#	ne	Qt	GTK+, WxWidgets	LCL	ne
UML editor	ne	špatné	ne	ne	ne	průměrné	ne	ne
Správce databází	Java DB, MySQL, PostgreSQL	ne	SQLite, MySQL, PostgreSQL, SQL	ne	ne	ne	ne	ne
Editor kódu	výborné	výborné	výborné	výborné	výborné	průměrné	průměrné	výborné
Dokončování kódu	výborné	průměrné	výborné	výborné	výborné	špatné	špatné	průměrné
Indikace chyb	výborné	průměrné	průměrné	průměrné	průměrné	ne	ne	ne
Refaktoring	výborné	průměrné	průměrné	průměrné	ne	ne	průměrné	ne
Unit testy	JUnit	ne	NUnit	ne	ne	ne	FPCUnit	ne
Pluginy	výborné	výborné	výborné	výborné	špatné	výborné	průměrné	průměrné
Správce projektu	výborné	výborné	výborné	výborné	výborné	výborné	ne	výborné
Správce tříd a objektů	výborné	výborné	průměrné	průměrné	špatné	průměrné	průměrné	průměrné
Správa verzí	CVS, SVN, Mercurial	ClearCase, SVN, Perforce, CVS	SVN	CVS	SVN, Git, Perforce	SVN, Git	ne	ne
SUMA	**27**	**25**	**21**	**21**	**18**	**16**	**15**	**14**

B. Tabulka s hodnocením vývojových prostředí -pokračování

Název	Gambas	Codelite	TheIDE	Geany
Platforma	Linux, Unix	Linux, MacOS, Unix, Windows	Linux, Unix, Windows	Linux, Unix, MacOS, Windows
Licence	GNU GPL	GNU GPL	BSD	GNU GPL
Poslední verze datum uvolnění	2.12 21.3. 2009	1.0.2759 9. 2. 2009	2008.1 6.8. 2008	0.16 15. 2. 2009
Prog. jazyk	Gambas Basic	C, C++	C, C++	C, C++, Java, PHP, Python, Perl, Pascal, ...
Kompilátor	Gbc	GCC, MSVC++	GCC	GCC
Debugger	Gambas	GDB	GDB	ne
Paměť, načítání	7,5 MB, < 1 s	30 MB, < 6 s	15,1 MB, < 3 s	4,6 MB, < 1 s
Dokumentace	výborné	průměrné	výborné	průměrné
GUI návrhář	výborné	ne	průměrné	ne
Toolkity	GTK+, Qt	ne	Ultimate++	ne
UML editor	ne	ne	ne	ne
Správce databází	MySQL, SQLite, PostgreSQL, ...	ne	ne	ne
Editor kódu	průměrné	výborné	průměrné	průměrné
Dokončování kódu	špatné	průměrné	špatné	špatné
Indikace chyb	ne	ne	ne	ne
Refaktoring	ne	špatné	ne	ne
Unit testy	ne	UnitTest++	ne	ne
Pluginy	průměrné	průměrné	špatné	průměrné
Správce projektu	špatné	průměrné	ne	špatné
Správce tříd a objektů	špatné	špatné	špatné	špatné
Správa verzí	SVN	SVN	ne	ne
SUMA	**13**	**13**	**10**	**9**

C. Ukázka konečné aplikace

přidávání souborů



D. Mezi-objektová komunikace

Ukázka kódu

coreplayer.cpp:

```
      ...
// pripoji signal pri zmene doby trvani nahravky.
connect(this->mediaObject, SIGNAL(totalTimeChanged(qint64)), this, SLOT(getTime()));
// pripoji signal emitovany behem prehravani nahravky.
connect(this->mediaObject, SIGNAL(tick(qint64)), this, SLOT(getTime()));

      ...
void CorePlayer::getTime()
{
      ...

   // emituje signal s informaci o celkove a odehrane dobe.
   emit timeChanged(current, total);
}
      ...
```

mainwindow.cpp:

```
      ...

// pripoji se signal emitovany prehravacem pri zmene doby prehravani k metode
// timeChanged().
connect(this->myPlayer, SIGNAL(timeChanged(long,long)), this,
        SLOT(timeChanged(long, long)));

      ...

void MainWindow::timeChanged(long current, long total)
{
      ...

   // cas se vypise do navesti ve stavove liste okna.
   this->timeLabel.setText(timeString);
}
```

E. Adresářová struktura přiloženého média