Rozšíření aplikace wxFormBuilder

Extension of wxFormBuilder

Bc. Jiří Růčka

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jiří RŮČKA**
Osobní číslo:	**A08470**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Rozšíření aplikace wxFormBuilder**

Zásady pro vypracování:

1. **Vytvořte rešerši na téma RAD nástroje pro tvorbu GUI aplikací využívajících SW knihovnu wxWidgets.**
2. **S využitím programovacího jazyka C++ a knihovny wxWidgets rozšiřte nástroj pro vizuální tvorbu GUI aplikací wxFormBuilder o podporu pro nové GUI komponenty (wxMediaCtrl, wxBitmapComboBox, a dalších) a implementujte kompletní podporu pro wxAUI.**
3. **Pro nové komponenty implementujte generování kódu pro jazyky C++, Python a systém zdrojů XRC.**
4. **Ověřte funkcionalitu implementovaných rozšíření na platformách MS Windows, Linux a OS X.**
5. **Upravené a rozšířené zdrojové kódy aplikace publikujte na www stránkách SourceForge.net (http://sourceforge.net/projects/wxformbuilder/)**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BLIŽŇÁK, Michal. Systémové programování. 1. vyd. Zlín: UTB ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1.**
2. **SMART, Julian, HOCK, Kevin. Cross-Platform GUI Programming with wxWidgets, Prentice Hall, 2006, ISBN 0-13-147381-6**
3. **wxFormBuilder at SourceForge.net, ULR: http://sourceforge.net/projects/wxformbuilder/**
4. **wxFormBuilder website, URL: http://wxformbuilder.org/**
5. **wxWidgets website, URL: http://www.wxwidgets.org/**
6. **FEATHERS, C. Michael. Údržba kódu převzatých programů, Computer Press, 2009, ISBN 978-80-251-2127-6.**
7. **MLÝNKOVÁ, Irena. XML Technologie, Grada, 2008, ISBN 978-80-247-2725-7.**
8. **ALEXANDRESCU, Andrei. Moderní programování v C++, Computer Press, 2004, ISBN 80-251-0370-6.**
9. **HARMS, Daryl, MCDONALD, Kenneth. Začínáme programovat v jazyce Python, BEN, 2008, ISBN 978-80-2512-161-0.**
10. **YOUNG, Michael J. XML – Krok za krokem, Computer Press, 2006, ISBN 80-251-1070-2.**

Vedoucí diplomové práce: **Ing. Michal Bližňák, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce je rozšířit aplikaci wxFormBilder o komponenty wxMediaCtrl, wxBitmapComboBox a kompletní podporu pro wxAUI. V první části práce jsou nejprve popsány základní vlastnosti softwarové knihovny wxWidgets a je popsáno a porovnáno několik hlavních aplikací pro návrh grafického rozhraní aplikací za použití knihovny wxWidgets. V závěru teoretické části je popsána vnitřní struktura aplikace wxWidgets. Druhá část se zabývá vlastním přidáváním komponent. Nejprve je obecně popsán postup při přidávání nových komponent a potom konkrétní komponenty, které byly přidány.

Klíčová slova: wxFormBuilder, wxWidgets, C++, Python, XML, open-source

ABSTRACT

The aim of this thesis is to add wxMediaCtrl, wxBitmapComboBox components and add full support for wxAUI to wxFormBuilder. In the first part, there is a description of software library wxWidgets, its basic features and several wxWidgets GUI designers are introduced and compared. In the end of theoretical part inernal structure of wxFormBuilder is described. The second part deals with adding components itself. First, the general process of adding component is described and then contrete components, which were added.

Keywords: wxFormBuilder, wxWidgets, C++, Python, XML, open-source

Chtěl bych velmi poděkovat vedoucímu diplomové práce Ing. Michalu Bližňákovi, PhD. za cenné rady, čas a trpělivost při konzultacích.

„Tajemství úspěchu v životě není dělat, co se nám líbí, ale nalézt zalíbení v tom, co děláme.“

Edison Thomas Alva

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

S rozvojem výpočetní techniky souvisí i potřeba stále nových a nových aplikací. Zákazníci požadují zhotovení aplikace v co nejkratším čase s co nejnižšími náklady a tak musí programátoři znovu vytvářet celé aplikace a navrhovat jejich grafické prostředí. Aby se práce ulehčila a zrychlila, byly vytvořeny metody a prostředky pro zrychlení vývoje tzv. RAD (Rapid Application Development).

Toto ulehčení vývoje obsahuje mnoho aspektů od generování grafického rozhraní až po vytváření šablon pro generování prototypů tříd a podobně. Návrh grafického rozhraní je velmi stereotypní a při ručním programování i značně nepřehledná. GUI designery značně ulehčují práci a vizuální návrh uživatelského rozhraní je takto daleko přehlednější.

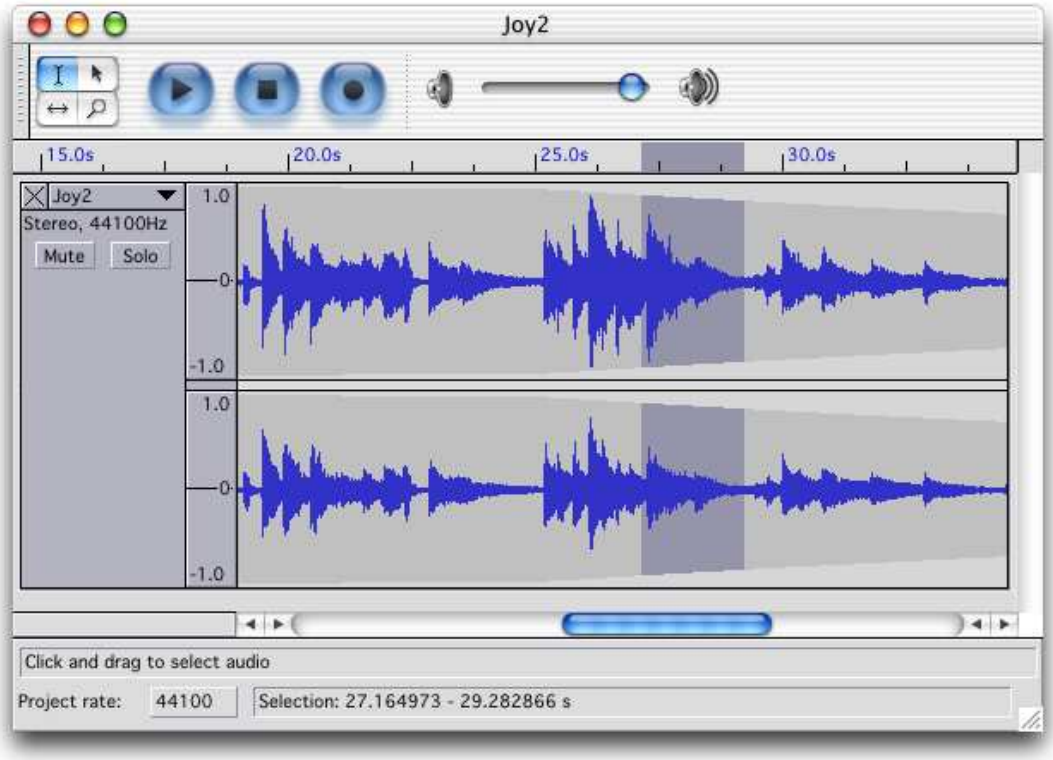
Designerů, vývojových prostředí a grafických knihoven je celá řada. V této práci se vyskytuje knihovna wxWidgets a nástroj wxFormBuilder. Oba tyto vývojové prostředky jsou open-source. Open source nebo také open-source software (OSS) je počítačový software s otevřeným zdrojovým kódem. Otevřenost zde znamená jak technickou dostupnost kódu, tak legální dostupnost - licenci software, která umožňuje, při dodržení jistých podmínek, uživatelům zdrojový kód využívat, například prohlížet a upravovat. V nepřesném ale poměrně běžném vyjadřování se označení *open source* používá i pro mnoho vlastností, které s otevřeností zdrojového kódu nesouvisí, ale vyskytují se u mnoha open source programů. Například může jít o bezplatnou dostupnost software, vývoj zajišťovaný úplně nebo z podstatné části dobrovolnickou komunitou nebo „nekomerčnost“.

I. TEORETICKÁ ČÁST

1 SOFTWAROVÁ KNIHOVNA WXWIDGETS

Aplikace wxFormBuilder je vytvořena pomocí několika programovacích jazyků. Jedním z nich je jazyk C++, konkrétně softwarová knihovna s názvem wxWidgets.

Knihovna byla určena primárně pro jazyk C++, ale v dnešní době je wxWidgets možné použít i pro jazyky Python (wxPython), Perl (wxPerl), Ruby, Lua a mnoho dalších. Použití softwarové knihovny wxWidgets má mnoho výhod. Mezi ty největší výhody patří multiplatformnost – zdrojový kód je s minimálními změnami možné zkompilovat na mnoha platformách (Windows, Linux, Mac, BSD atd.). Pro každou platformu existuje jedna verze knihovny, která využívá nativních knihoven systému pro tvorbu GUI. Další výhodou je, že knihovna je vyvíjena a šířena jako Open-Source, takže k ní existuje velké množství dokumentací, vývojových nástrojů, které lze volně získat a zdarma používat. [1]



Obr č. 1: Program Audacity vytvořený pomocí wxWidgets

1.1 Historie

Autorem první verze knihovny je Julian Smart, který jako student na Univerzitě v Edinburgu v roce 1992 potřeboval naprogramovat aplikaci, která bude fungovat na

platformách Windows a X-windows na unixových pracovních stanicích. Komerční multiplatformní knihovny byly v té době příliš drahé, tak student vytvořil knihovnu vlastní. První název knihovny byl wxWindows (w – Windows, x – X-windows). Institut aplikací umělé inteligence dovolil umístit zdrojové kódy knihovny na Internet a tak začal masivní vývoj. [1]

K vývoji wxWidgets přispělo i mnoho českých vývojářů. Mezi nejznámější patří Václav Slavík, který je autorem části wxHTML a wxXRC. Část wxHTML slouží k zobrazování HTML souborů v aplikaci a třída wxXRC umožňuje načítat strukturu dialogových oken ze souboru XML. [1]

1.2 Základní vlastnosti knihovny wxWidgets

Mezi základní prvky knihovny patří velké množství tříd pro vytváření dialogových oken, ovládacích prvků a třídy pro tvorbu uživatelsky konfigurovaného rozhraní GUI. Jakákoliv třída odvozená od třídy wxEvtHandler může zpracovávat zprávy vygenerované systémem nebo aplikací. Tyto zprávy mohou být namapovány na obslužné rutiny dynamicky pomocí funkce Connect, nebo staticky tabulkou událostí. Knihovna také nabízí podporu ladění programu prostřednictvím skupiny speciálních funkcí, které mohou být použity k zápisu ladících informací a chybových stavů na standardní výstup nebo do souboru. Navíc paměťové operace jsou vytvořeny tak, že umožňují detekovat případné úniky paměti v podobě absence uvolnění alokované paměti. V Knihově wxWidgets je implementována architektura Dokument/Pohled, kdy třída wxDocument se stará o data aplikace a třída wxView umožňuje jejich vizualizaci. Touto architekturou jsou zajištěny standardní funkce aplikace, jakými jsou otevíraní a ukládání dokumentů, vytváření nových dokumentů nebo poskytování funkcí Undo/Redo. [1] [10]

2 RAD APLIKACE

RAD (Rapid Aplication Development) je metoda vývoje software, která umožňuje rychlejší prototypování aplikace na úkor minimálního plánování. Při vývoji touto metodou, je proces plánování prokládán tvorbou samotného programu. Omezení plánování obecně vede k rychlejšímu vývoji aplikace a k jednodušším změnám požadavků na software. [8]

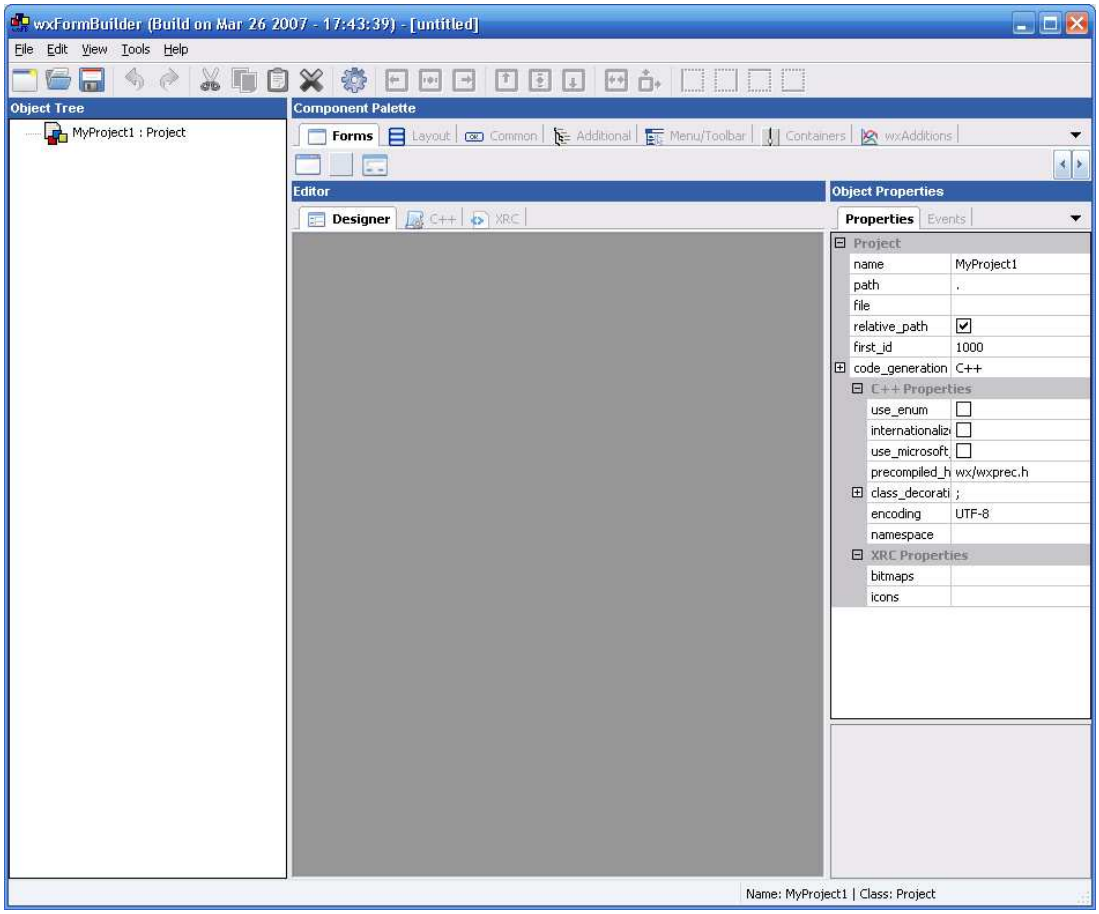
Jeden z hlavních problému při multiplatformním návrhu GUI je, že dialogové prvky (widgety) mívají rozdílnou velikost na různých platformách. Například grafická knihovna Motif (X-Window) má širší okraje, zatímco ovládací prvky operačního systému Windows se snaží zabrat co nejméně místa. Navíc uživatel si často může změnit velikost písma, nebo zvolit jiný skin či téma oken, což vede opět ke změně proporcí ovládacích prvků. V neposlední řadě se velikost ovládacích prvků může lišit i v jiných programovacích jazycích, jako například proměnná typu word má různou velikost v různých jazycích. [7]

2.1 wxFormBuilder

wxFormBuilder je multiplatformní open-source RAD aplikace pro tvorbu GUI s použitím softwarové knihovny wxWidgets. Cílem této aplikace je poskytnout uživateli možnost vizuálního vytváření GUI a vygenerování zdrojového kódu pro další použití v uživatelské aplikaci. Návrh uživatelského rozhraní je realizován pouze pomocí sizerů a mezi výhody tohoto programu patří možnost práce s ne-grafickými komponentami a poměrně jednoduché prostředí pluginů pro úpravu a přidávání nových tříd a komponent knihovny wxWidgets. [2]

Aplikaci lze získat mnoha způsoby. Na stránkách wxFormBuilderu se nachází už přeložená aplikace pro různé platformy. V případě linuxových distribucí obsahující balíčkovací systém APT je možné najít nejnovější stabilní verzi v repositáři. Pro ostatní se na webových stránkách nachází zdrojové kódy, které lze získat i přes SVN. [2]

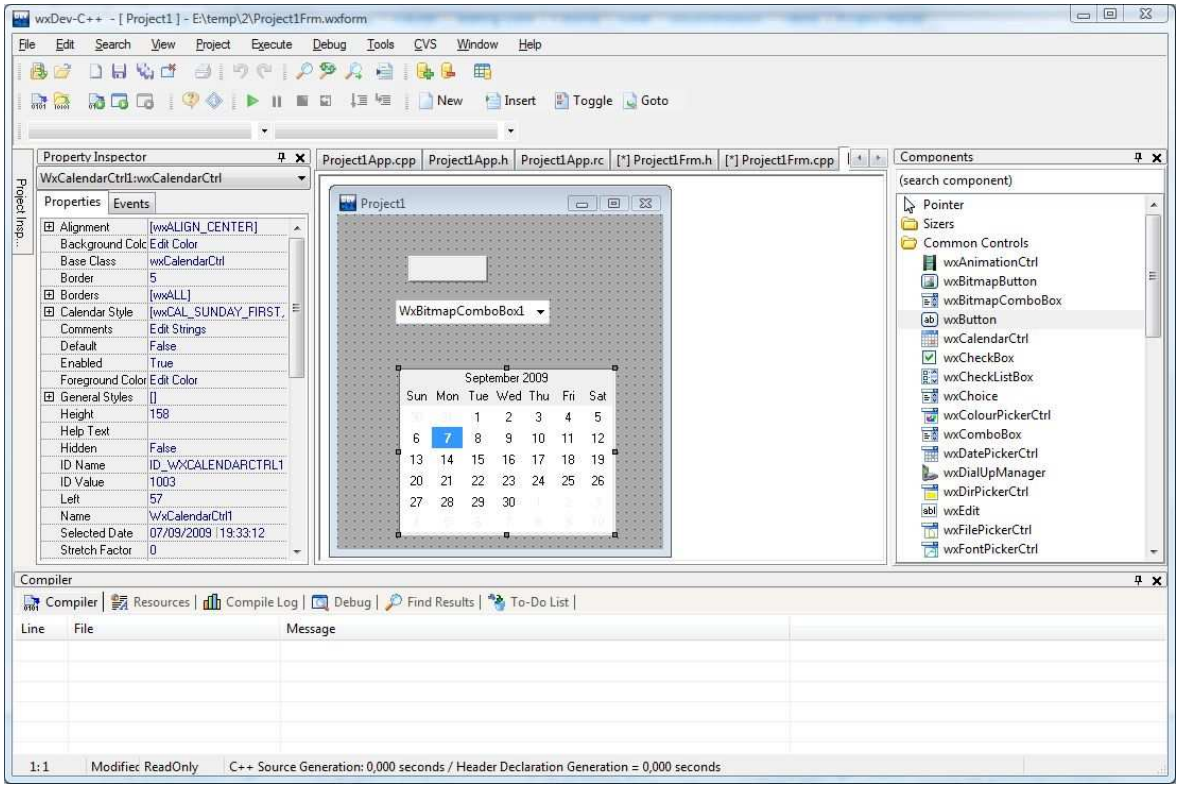
Nejedná se o plnohodnotné vývojové prostředí, ale pouze o GUI designer, avšak tento nedostatek ne nahrazen spoluprácí s několika jinými vývojovými prostředí. Tato spolupráce spočívá v tom, že při vytváření projektu, je možné vygenerovat soubor, který wxFB načte a může přímo editovat. Změny jsou potom vidět hned v prostředí. [2]



Obr č. 2: Prostředí aplikace wxFormBuilder

2.2 wxDev-C++

wxDev-C++ je rozšíření vývojového prostředí Dev-C++ vytvořeného Colinem Laplacem. Tento program také umožňuje vizuálně vytvářet dialogová okna a jejich ovládací prvky knihovny wxWidgets. wxDev-C++ stejně jako Dev-C++ jsou aktivně vyvíjené projekty a jejich hlavním cílem je nabídnout komunitě freewarový, open-source s komerční nástavbou, IDE/RAD nástroj, pro vývoj aplikací pomocí softwarové knihovny wxWidgets. [5]



Obr č. 3: Prostředí aplikace wxDev-C++

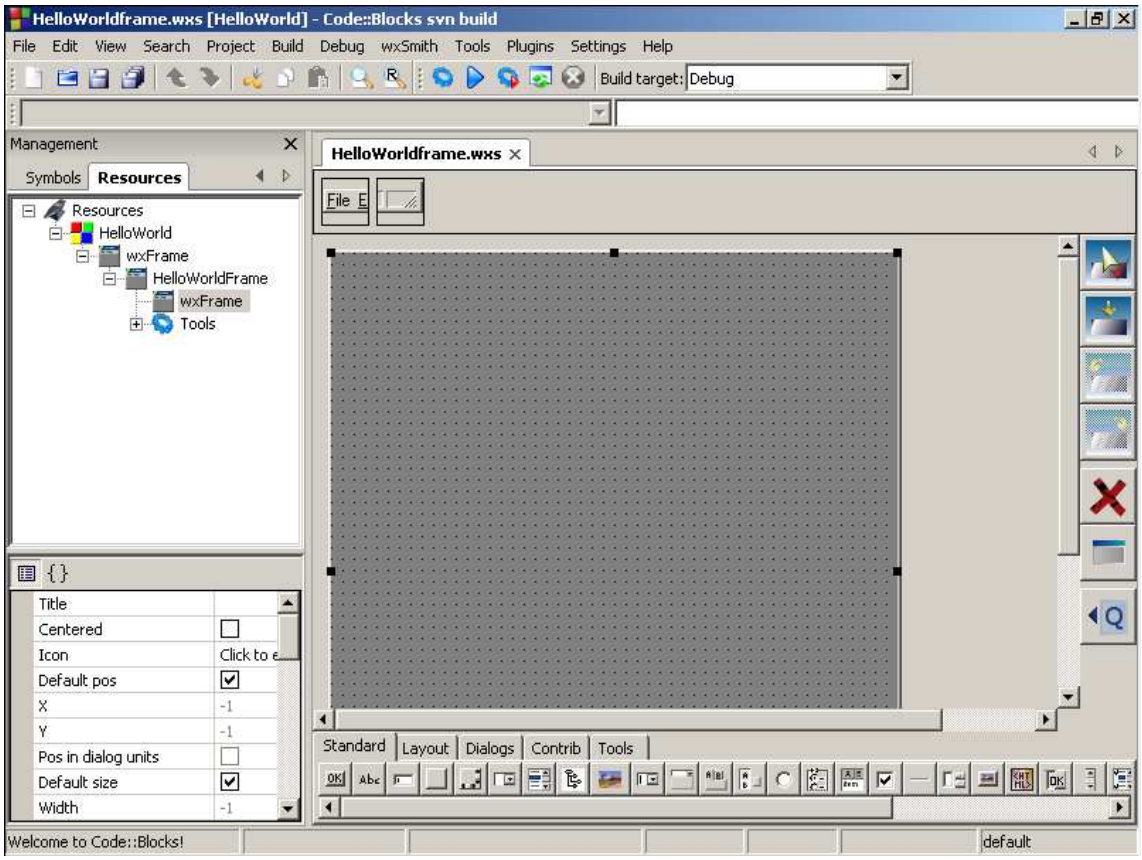
GUI designer vývojového prostředí pracuje na principu drag&drop a umožňuje generovat XRC XML zdroje. Při návrhu je možné využít sizery, ale jako jeden z mála designerů umí tvořit GUI i bez nich. Mapování událostí na obslužné rutiny probíhá pomocí funkce Connect přímo v editoru. [5]

Prostředí wxDev-C++ obsahuje i integrovaný debugger s podporou GDB (GNU Project Debbuger), který dokáže zobrazovat registry CPU a udržuje seznam lokálních proměnných aplikace. Kromě těchto vlastností je implementován automatický sledovač zásobníku instrukcí a mnoho dalších. [5]

V editoru jsou zakomponovány standardní funkce, jaké je možné najít v kterémkoliv vývojovém prostředí. Mezi tyto funkce patří prohlížeč tříd, doplňování kódu, správa projektu, různé profily projektu, nastavitelné zvýrazňování syntaxe atd. Aplikace má také zabudovanou podporu pro CVS. [5]

2.3 wxSmith

wxSmith je plugin vývojového prostředí IDE Code::Blocks pro návrh grafického uživatelského rozhraní. Uživatel může navrhovat GUI jak bez pomocí sizerů, tak i s nimi. Stejně jako každý GUI designer generuje wxSmith kód v jazyce C++, bohužel však už neumožňuje generování do jiných kódu, jako například Python. Je to dáno hlavně prostředím Code::Blocks, které je určeno primárně pro jazyk C++. [4]



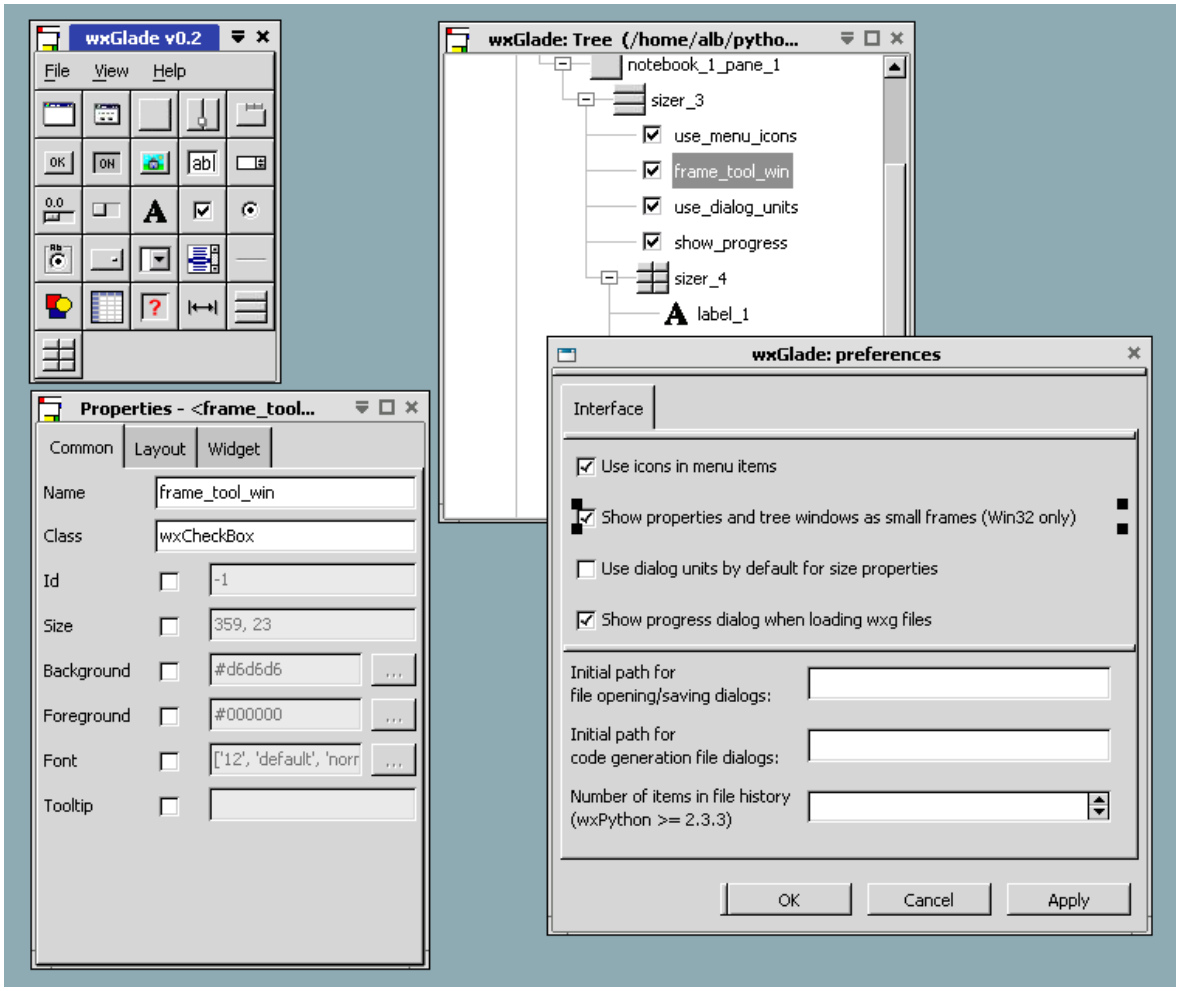
Obr č. 4: Prostředí aplikace Code::Blocks s pluginem wxSmith

Samozřejmostí je práce s souborem zdrojů XRC a to jak načítání, tak ukládání, avšak načítání je zatím omezeno pouze na jeden dialog. Plugin umí také generovat tabulku událostí a šablony pro obslužné rutiny událostí. Jedna z mála věcí, co wxSmith zatím nezvládá jsou validátory. Ty jsou vloženy do ovládacího prvku a slouží jako prostředník mezi daty programovacího jazyka a ovládacího prvku. Tyto data jsou mezi zpracováváním validátorem kontrolována. [4]

2.4 wxGlade

wxGlade je aplikace pro návrh GUI, napsaná v jazyce Python, za použití velmi oblíbené knihovny wxPython, díky které je možno lépe vytvářet uživatelská rozhraní. V současné verzi je možno vygenerovat zdrojový kód pro jazyky Python, C++, Perl, Lisp a soubor zdrojů XRC. [6]

Už podle názvu je zřejmé, že vzorovou aplikací, podle které je tato vytvořena, je program Glade, který je určen pro vytváření GUI za použití knihovny GTK+/GNOME, se kterou wxGlade sdílí základní filosofii a návrh, avšak nemají společný ani jeden řádek kódu. [6]



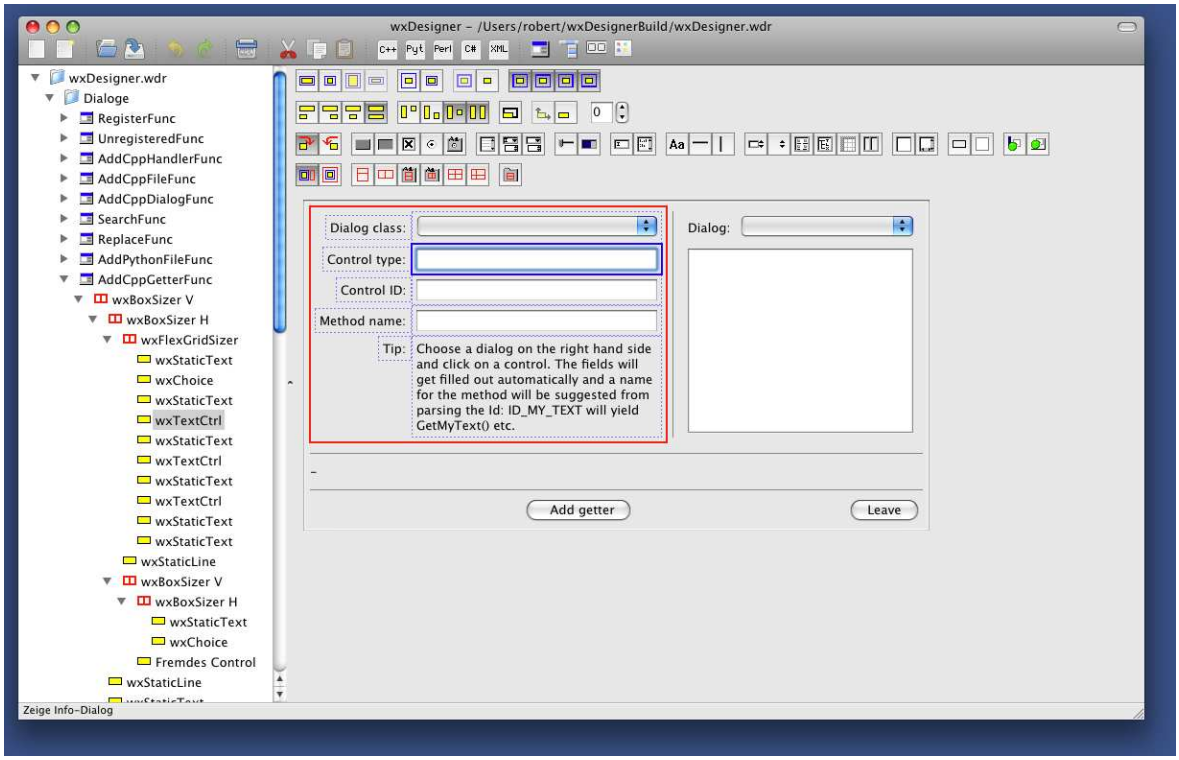
Obr č. 5: Vzorová session prostředí wxGlade

Tato aplikace není a nikdy nebude plnohodnotné vývojové prostředí, ale jednoduchý designer, takže vygenerovaný zdrojový kód umí využít pouze k zobrazení navrhnutého prostředí. Pokud by uživatel přesto požadoval plnohodnotné vývojové prostředí, je možno

použít prostředí Boa Constructor, PythonCard nebo Spe, která mají wxGlade zakomponovaný přímo. [6]

2.5 wxDesigner

wxDesigner je proprietární nástroj pro rychlou tvorbu dialogových oken pro aplikace založené na knihovně wxWidgets. Jeho rozhraní umožňuje i méně znalým uživatelům wxWidgets vytvořit esteticky přijatelné grafické rozhraní aplikace a obsahuje základní funkce jako Copy/Cut/Paste, Undo/Redo a možnost náhledu vytvořeného dialogového okna, což ulehčuje testování. wxDesigner generuje zdrojový kód pro jazyky: C++, Python, Perl a C#. [7]



Obr č. 6: Prostředí aplikace wxDesigner

Návrh GUI je ve wxDesigneru založen na sizerech, což jsou komponenty návrhového systému knihovny wxWidgets, které umožňují pracovat s ovládacími prvky v určitých hranicích, jako jsou řady, sloupce nebo daleko komplexnější mřížky. Díky tomu, že sizery kontrolují velikosti ovládacích prvků, změna písma operačního systému nezpůsobí vážnou deformaci okna, protože dialogové okno se bude patřičně zmenšovat a zvětšovat jako

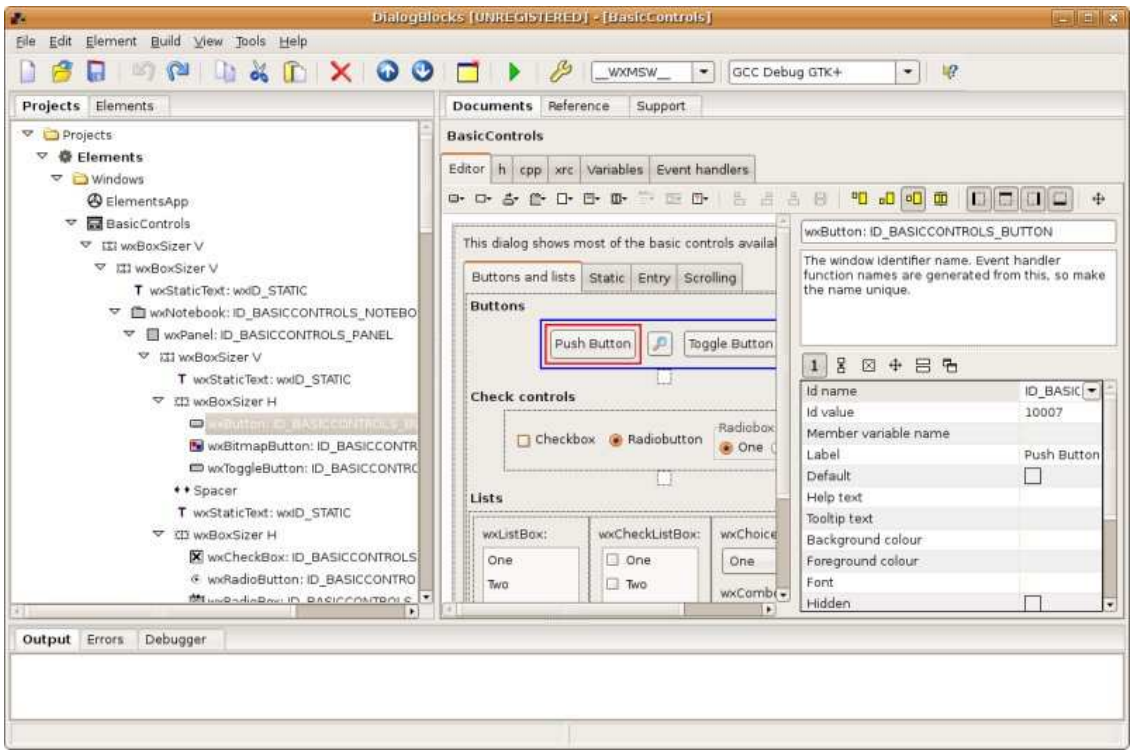
celek. Typy sizerů je navíc možné měnit „za běhu“ a vkládání sizerů do již vytvořeného návrhu je také velmi jednoduchou záležitostí. Kromě toho jsou podporovány standardní editační úpravy (copy/paste) a práce se schránkou. [7]

Dalším krokem při návrhu GUI je práce se zprávami (událostmi), která je možná pouze ze zdrojového kódu. wxDesigner podporuje práci s událostmi, vytváření nových tříd, nových souborů, nových event handlerů, nebo funkcí pro přístup k ovládacím prvkům dialogového okna. To vše je integrováno do editoru zdrojového kódu, který je součástí wxDesigneru. [7]

2.6 DialogBlocks

DialogBlocks je GUI designer, jejímž autorem je Julian Smart (autor knihovny wxWidgets) a firma Anthemion Software. Návrh je založen na použití sizerů a umožňuje rychlé vytvoření profesionálně vyhlížejících oken, wizardů a dialogů pro platformy Windows, Linux, Mac a všechny další, které jsou podporovány knihovnou wxWidgets. [9]

Hlavním cílem DialogBlocks je rychlost návrhu, proto má téměř pro každou operaci klávesovou zkratku a dvojklikem na komponentu je vytvořeno pop-up okno s editovatelnými vlastnosti, jako alternativa k editoru vlastností. [9]



Obr č. 7: Prostředí aplikace DialogBlocks

Mimo jiné má DialogBlocks také „výukovou“ funkci – ke každé komponentě existuje překlad nastavení sizeru do angličtiny ve formě popisného módu, stejně tak jako popis samotné komponenty. To umožní uživateli lépe pochopit nastavení a odhalit a opravit chybu daleko rychleji. [9]

Mezi základní funkce DialogBlocks patří:

- vytváření dialogů a standardních ovládacích prvků
- podpora wxAUI a dokování
- vytváření wizardů a property sheet dialogů
- výstup do C++ nebo XRC
- pohodlný property editor pro každý ovládací prvek či sizer

Aplikace DialogBlocks je dostupná na mnoha platformách: Windows, několik Linuxových distribucí, FreeBSD, Solaris x86 a Mac (PPC a Intel). [9]

3 POROVNÁNÍ RAD/GUI NÁSTROJŮ

3.1 Obecné informace

Přestože je knihovna wxWidgets zdarma a open-source, ne všechny vývojové nástroje sdílejí stejnou filosofii a licenci. Obecné vlastnosti, včetně autorů popisuje tabulka č.1. Všechny prostředí však mají zdarma alespoň trialovou verzi. I když jsou všechny freeware nástroje v tabulce open-source, nemusí to být pravidlo. [4] [6]

Tab.č. 1: Obecné informace o RAD aplikacích

RAD	**Autor**	**Cena**	**Open-Source**	**Licence**
wxSmith	Bartlomiej Swiecki	zdarma	Ano	GPL
Dialog Blocks	Julian Smart	zdarma(trial), €56(normal), €30(student)	Ne	Proprietární
wxDesigner	Robert Roebling	zdarma(trial), €129(normal), €29(student)	Ne	Proprietární
wxFormBuilder	José Antonio Hurtado, Juan Antonio Ortega, Ryan Mulder, Ryan Pusztai, Michal Bližňák	zdarma	Ano	GPL
wxDev-C++	Guru Kathiresan, Tony Reina, Malcolm Nealon, Joel Low	zdarma	Ano	GPL
wxForms for Delphi / C++ Builder	Guru Kathiresan	zdarma(trial), $29.99	Ne	Proprietární

3.2 Podpora programovacích jazyků

Všechny designery dokáží generovat zdrojový kód pro jazyk C++ a kromě programu wxSmith i pro jazyk Python (wxPython), jak je vyznačeno v tabulce č.2. Oproti tomu jazyky Lua a Ruby nejsou v uvedených aplikacích podporovány. Jazyky Perl a C# jsou podporovány pouze v proprietárním designeru wxDesigner. [4] [6]

Tab.č. 2: Tabulka podpory programovacích jazyku RAD aplikací

Jazyk	**wxSmith**	**DialogBlocks**	**wxDesigner**	**wxFB**	**wxDevC++**	**wxGlade**
C++	Ano	Ano	Ano	Ano	Ano	Ano
Python	Ne	Ano	Ano	Ano	Ano	Ano
Lua	Ne	Ne	Ne	Ne	Ne	Ne
Ruby	Ne	Ne	Ne	Ne	Ne	Ne
Perl	Ne	Ne	Ano	Ne	Ne	Ne
C#	Ne	Ne	Ano	Ne	Ne	Ne
XRC	Ano	Ano	Ano	Ano	Ano	Ano

Další vlastnost, která uživatele zajímá, je podpora pro generování různých tříd. Všechny designery jsou schopny pracovat a vygenerovat zdrojový pro základní třídy (wxFrame, wxButton,wxTextCtrl…). Podpora pro nestandardní ovládací prvky knihovny wxWidgets je zobrazena v tabulce č.3. [4] [6]

Tab.č. 3: Tabulka podpory vybraných speciálních widgetů

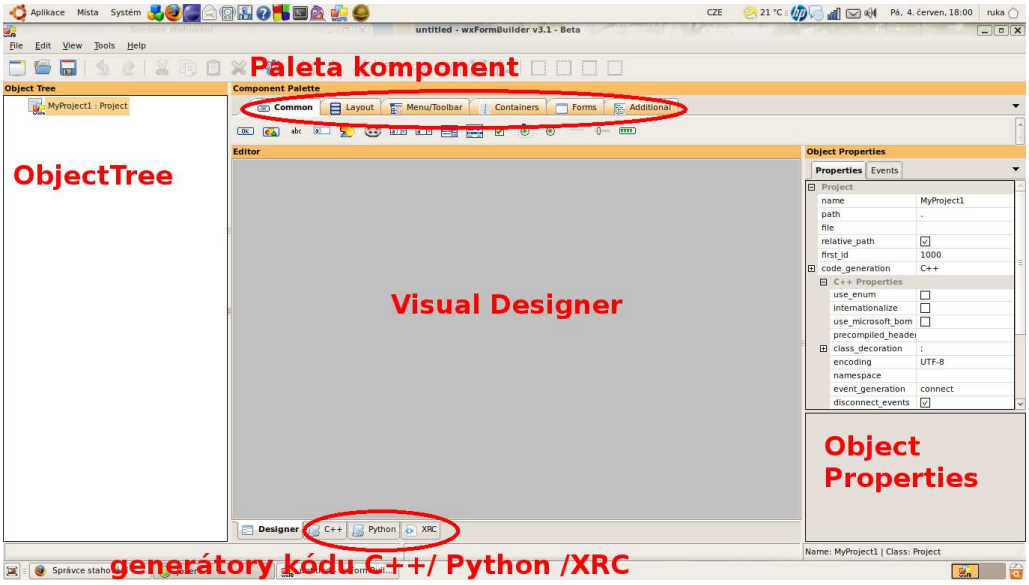
Widget (wx…)	**wxSmith**	**Dialog Blocks**	**wxDesigner**	**wxFB**	**wxDevC++**	**wxGlade**
AnimationCtrl	Ano	Ano	Ne	Ano	Ano	Ne
AuiNotebook	Ano	Ano	Ne	Ano	Ano	Ne
CalendarCtrl	Ano	Ano	Ne	Ano	Ano	Ne
ColourDialog	Ano	Ne	Ne	Ne	Ano	Ne

DialUpManager	Ne	Ne	Ne	Ne	Ano	Ne
DirDialog	Ano	Ne	Ne	Ne	Ano	Ne
FileDialog	Ano	Ne	Ne	Ne	Ano	Ne
FontDialog	Ano	Ne	Ne	Ne	Ano	Ne
GridBagSizer	Ne	Ano	Ne	Ano	Ne	Ne
HtmlListBox	Ne	Ne	Ne	Ne	Ne	Ne
HtmlWindow	Ano	Ano	Ne	Ano	Ano	Ne
MessageDialog	Ano	Ne	Ne	Ne	Ano	Ne
PrintDialog	Ano	Ne	Ne	Ne	Ano	Ne
ProgressDialog	Ano	Ne	Ne	Ne	Ano	Ne
SashWindow	Ano	Ano	Ne	Ne	Ne	Ne
ScrollBar	Ano	Ano	Ne	Ano	Ano	Ne
StaticBox	Ano	Ano	Ne	Ne	Ano	Ne
TextEntryDialog	Ano	Ne	Ne	Ne	Ano	Ne
Treebook	Ne	Ne	Ne	Ne	Ano	Ne

4 STRUKTURA WXFORMBUILDERU

Po spuštění aplikace se v hlavním okně nachází tyto základní části: [3]

- Paleta Komponent: umožňuje přidávat objekty do projektu.
- Object Tree: zobrazuje stromovou strukturu projektu
- Visual Designer: poskytuje vizuální reprezentaci právě navrhovaného GUI
- Object Properties: zobrazuje všechny vlastnosti vybraného objektu.
- C++ code editor: obsahuje vygenerovaný kód v C++ (pouze pro čtení)
- XRC code editor: obsahuje vygenerovaný kód pro systém zdrojů XRC (pouze pro čtení)
- Python code editor: obsahuje vygenerovaný kód v Pythonu (pouze pro čtení).



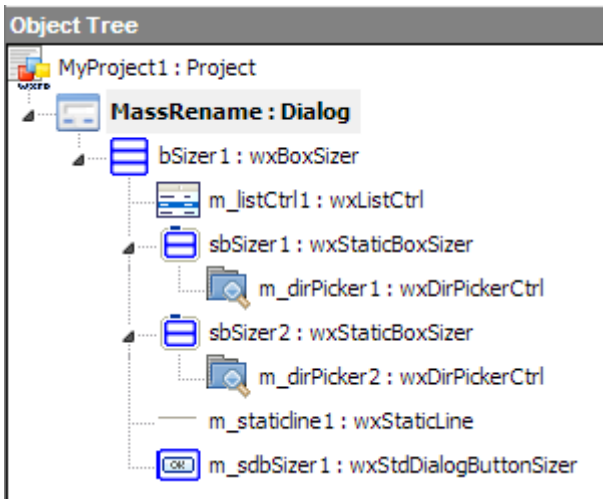
Obr č. 8: Umístění základních částí hlavního okna wxFB

Z tohoto rozdělení vyplývá, jak tyto jednotlivé části pracují dohromady. Například, pokud je editována vlastnost „label“ třídy wxButton, je možné vidět jak se změní popisek tlačítka v Design editoru a zároveň se informace o provedené změně zobrazí ve status baru. [3]

Aplikace je navržena tak, aby všechny tyto části byly nezávislé jak jen to bude možné, za použití tzv. pozorovatelského návrhu. Tento návrh se skládá ze dvou hlavních rozhraní: [3]

- Pozorující rozhraní: všem objektům, které využívají tohoto rozhraní, je možno zasílat zprávy, pokud dojde ke změně pozorujícího objektu. Jinými slovy, rozhraní obsahuje všechny metody, potřebné k zachycení událostí wxFB. Všechny výše popsané části jsou objekty, které toto rozhraní implementují. Pozorující rozhraní využívá wxEvtHandler.
- Pozorovatelné rozhraní: poskytuje všechny metody pro editaci dat a observer připojení. Všechny pozorující objekty mají přístup k tomuto objektu, za účelem změn v projektu. K tomu slouží objekt ApplicationData (appdata.h/.cpp), který je takovým zprostředkovatelem wxFB. Toto rozhraní zajišťuje všechny příkazy wxFB jako jsou: vytvoření objektu, modifikace vlastnosti, atd. Jeho další funkcí je, že po vykonání příkazu rozesílá zprávy wxEvent ostatním observerům.

Všechny pozorující objekty mají přístup k instanci třídy ApplicationData, protože pracují s jejími daty. Pokud jakýkoliv observer zavolá jakoukoliv metodu třídy ApplicationData, tato třída odešle zprávu všem ostatním. [3]



Obr č. 9: ObjectTree

Souhrn základních procesů, při vytváření nového objektu z palety komponent: [3]

1. Je zavolána obslužná rutina OnButtonClick. Protože každá položka v paletě komponent (wxToolBar) obsahuje název komponenty, je tento název připojen k události a následně je zavolána funkce ApplicationData::CreateObject(name), kde name je název vytvářené komponenty.

2. ApplicationData se pokusí vytvořit novou instanci požadované komponenty. Tato komponenta je potomek označeného objektu.
3. Objekt ApplicationData upozorní všechny dostupné pozorovatele událostí wxFBObjectEvent s identifikátorem wxEVT_FB_OBJECT_CREATED.
4. ObjectTree zpracuje tuto událost tím, že převezme project tree z instance třídy AplicationData a tento strom přetvoří na strom ve třídě wxTreeCtrl.
5. Object Inspektor také zpracuje tuto událost a z instance třídy ApplicatonData převezme označený objekt a zobrazí jeho vlastnosti.
6. VisualDesigner převezme project tree a překreslí GUI.
7. Zbytek Observers nijak na událost nereaguje.

Nejdůležitější skutečnost je, že každý pozorovatel není ovlivňován ostatními, takže může nezávisle vykonávat svou funkci. Tento fakt zaručuje čistější a více škálovatelný kód. Například objekty CppPanel a XrcPanel, které jsou zodpovědné za vygenerování C++ a XRC kódu, pracují zcela samostatně, takže není velký problém vytvořit PythonPanel, který bude generovat kód v jazyce Python. Jediné, co je potřeba udělat je vytvoření panelu v konstruktoru MainFrame a přiřazení event handleru instanci třídy ApplicationData. [3]

V třídě CommandProcessor je implementována funkce Undo/Redo, kdy každý příkaz v projektu je zapouzdřen do podtřídy Command. Konkrétní příkaz by měl poskytovat operace pro vykonání příkazu a vrácení zpět. [3]

Například, příkaz InsertObjectCmdFor vloží objekt do object tree a uchová dva objekty:

1. Vkládaný objekt.
2. Objekt, který bude po vložení tohoto objektu jeho rodičem.

Pokud je tedy příkazovým porocesorem zavolána metoda Execute() daného příkazu, vznikne vazba mezi těmito dvěma objekty, takže pokud příkazový procesor zavolá funkci Restore, bude tato vazba zrušena. Proto CommandProcessor udržuje dva příkazové zásobníky, příkazy pro Undo a příkazy pro Redo a tím je zajištěna tato funkce. [3]

Třída GlobalApplicationData byla vytvořena aby bylo možné přistupovat k některým proměnným kdekoliv v kódu aplikace. Například cesta k projektu je potřeba k sestavení celé cesty bitmapy, protože tyto bitmapy jsou uloženy jako relativní cesty k projektu. [3]

4.1 Datová struktura wxFormBuilderu

Jedna z nejužitečnějších vlastností wxFB je skutečnost, jak jsou jednotlivé komponenty implementovány. Komponenty zde nejsou začleněny do zdrojového kódu, ale jsou vytvořeny jako externí moduly, které jsou načítány a vytvářeny za běhu aplikace. [3]

Tyto externí moduly a balíčky jsou vytvářeny pomocí xml object descriptor (*.xml*), sdílené knihovny (*.dll/.so*) a šablony pro generování kódu (*.cppcode*). [3]

Při startu wxFB jsou prohledány adresáře s pluginy a jsou hledány soubory pluginů (*.xml, .cppcode, .so/.dll*). Pokud je nalezen plugin se všemi těmito soubory, wxFB načte všechny moduly a uloží je do instance třídy *ObjectDatabase* (*database.h/.cpp*). [3]

Datový model wxFB obsahuje několik základních objektů: [3]

- ObjectDatabase: uchovává reference na všechny objekty s reference counted *boost::shared_ptr*, takže žádné objekty nemohou být odstraněny dokud existuje instance třídy *ObjectDatabase*. Tato třída je také zodpovědná za vytvoření objektů (*ObjectBase*), jelikož má k dispozici všechny informace potřebné k jejich vytvoření.
- ObjectPackage: reprezentuje balíček objektů a ukládá seznam objektů v daném balíčku. V paletě komponent jsou jednotlivé komponenty roztříděny právě podle těchto balíčků.
- ObjectInfo: obsahuje popisy jednotlivých komponent, které jsou reprezentovány typem objektu, souborem vlastností, souborem rodičovských tříd (za účelem dědění vlastností) a unikátním jménem.
- ObjectType: Definuje typ objektu. Tento typ je použit k vytvoření hierarchické struktury objektů. Například všechny ovládací prvky jsou typu *widget*, sizery typu *sizer* a objekty, které obsahují ostatní widgety (jako wxPanel) jsou typu *container*. wxFB umožní vytvoření objektu typu *widget* uvnitř objektu typu *sizer*, ale ne uvnitř jiného objektu *widget*.

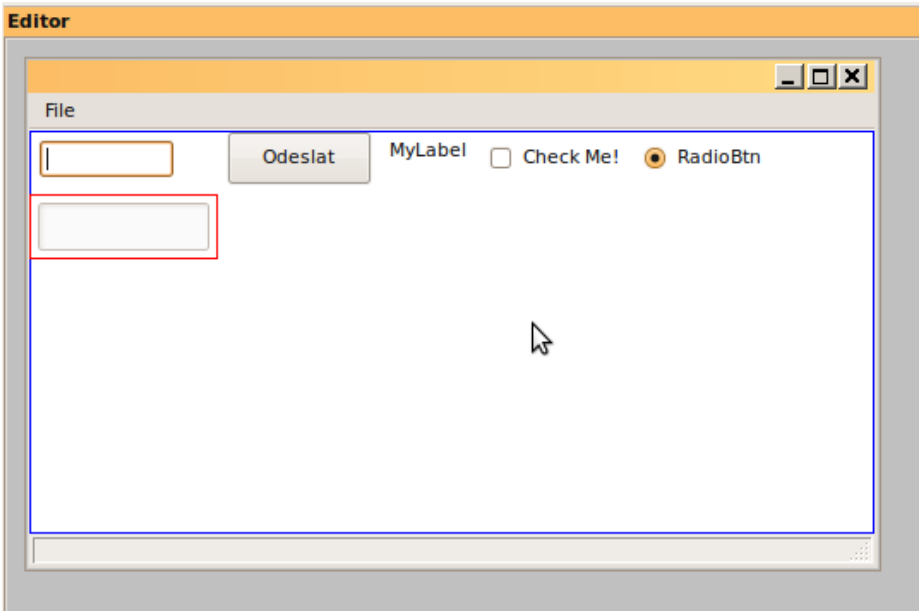
U třídy ObjecType je taky možné nastavit příznak *item*. To značí, že objekt bude považován za *item object*, což je virtuální objekt, který rozšiřuje soubor vlastností podřízeného objektu. Například pokud je vytvořen objekt typu *widget* uvnitř sizeru, je potřeba vytvořit virtuální objekt *sizeritem*, takže pokud je widget označen, jsou k dispozici i vlastnosti objektu *sizeritem*, který ale není vidět v *object tree*. [3]

- PropertyInfo: popisuje vlastnosti komponenty. Každá vlastnost má jméno, výchozí hodnotu a typ. Vlastnost může taky obsahovat seznam možností u typů *bitlist, option* nebo seznam potomků typu *parent*.
- PropertyType: je typ vlastnosti (*PT_TEXT, PT_BOOL, PT_INTEGER,* ...)
- ObjectBase: instance komponenty, která bude umístěna do uživatelského project tree. Tyto instance jsou vytvořen instancí třídy ObjectDatabase a metody pro přidávání vlastností nejsou soukromé, ale mohou být volány pouze ObjectDatabase.
- Property: reprezentuje vlastnost objektu a obsahuje hodnotu zadanou uživatelem. Všechny hodnoty jsou vnitřně uloženy jako textové řetězce, nezávisle na typu hodnoty. Proto je v projektu mnoho funkcí, které převádí typy proměnných na textové řetězce a naopak (typeconv.h/.cpp).

Všechny objekty, kromě ObjectBase a Property, jsou uloženy v xml souborech. Mezi tyto soubory patří především objtypes.xml, který obsahuje základní popis a pravidla pro vytváření OjectType. Kromě toho jsou uvnitř souboru popsány vztahy mezi rodiči a potomky. [3]

4.2 Visual Editor

Designer je část wxFormBuilderu, která zobrazuje uživateli rozložení ovládacích prvků. wxFB pracuje se dvěma druhy prvků wxWidgets a jsou to prvky sizer a window. Na všechny komponenty window se nahlíží jednotným pohledem a to samé platí u sizerů. Hlavní myšlenka je totiž ta, že pluginy zapouzdřují práci s konkrétními třídami a wxFB práci s komponenty v obecné rovině. [3]



Obr č. 10: Visual Editor aplikace wxFB

Designer obsahuje několik základních objektů:

- VisualEditor: hlavní třída, která zobrazuje náhled GUI a obsluhuje události wxFBEvents. Ukládá si také referenci na poslední vytvořené okno, aby se předešlo kompletnímu překreslení, pokud je označen jiný objekt tohoto okna.
- wxObjectMap a ObjectBaseMap: pokud je vybrán objekt a wxFB vyvolá událost wxFBObjectEvent s id wxEVENT_FB_OBJECT_SELECTED všem Observers, reference na ObjectBase události je odkaz na objekt, který byl označen. Toto mapování umožňuje vyhledání třídy wxObject podle příslušné instance třídy ObjectBase.
- VObjEvtHandler: event handler pro všechny objekty wxObjects v designeru. Po vytvoření třídy wxObject je VObjEvtHandler vložen do event handler zásobníku,
- IObjectView a IObject: tyto třídy tvoří rozhraní (čistě virtuální), které pochází z kódu pluginu. Pluginy nepoužívají ukazatele, proto tyto rozhraní poskytují další možnost přístupu.

4.3 Pluginy

Pluginy jsou při startu aplikace načítány z podadresáře plugins v kořenovém adresáři wxFB. Aplikace prohledá tento adresář a jeho podadresáře, ve kterých hledá adresáře xml, icons a sdílené knihovny (.so/.dll). Pokud je nalezen adresář xml a v něm soubor .xml, je tento soubor rozparsován na jednotlivé popisy objektů, které jsou v daném pluginu obsaženy. [3]

Příklad objektového popisu pro třídu wxTextCtrl:

```
1  <objectinfo class="wxTextCtrl" icon="text_ctrl.xpm" type="widget">
2      <inherits class="wxWindow" />
3      <property name="name" type="text">m_textCtrl</property>
4      <property name="style" type="bitlist">
5        <option name="wxTE_MULTILINE"     help="The text control allows multiple
   lines."/>
6        <option name="wxTE_PASSWORD"      help="The text will be echoed as
   asterisks."/>
7        <option name="wxTE_READONLY"      help="The text will not be user-editable."/>
8      </property>
9      <property name="value" type="wxString_i18n" />
10     <property name="maxlength" type="uint" help="The maximum length of user-entered
   text.">0</property>
11   </objectinfo>
```

wxFB načte ikonu text_ctrl.xpm z adresáře icons. Sdílená knihovna by měla obsahovat třídu komponenty wxTextCtrl a vrátit ji wxFB, pokud se ji uživatel pokusí vytvořit. Sdílená knihovna může mít libovolné jméno a je specifikována atributem lib v kořenovém prvku souboru .xml pro daný plugin. [3]

Příklad:

```
1  <package name="Common" lib="libcommon" icon="button16x16.xpm" desc="wxWidgets common
   controls">
```

4.4 Generování kódu

Celý systém generování kódu je založen na konceptu subtříd. Zdrojové soubory generované wxFB není možné měnit, protože všechny změny provedené uživatelem mimo wxFB by byly v dalším generování kódu ztraceny. Proto je v uživatelských aplikacích vytvářena odvozená třída od třídy generované wxFB a všechny události je možné zpracovávat touto třídou. Samozřejmě, že tato třída musí být v jiném souboru, který bude obsahovat odkaz na hlavičkový soubor generované třídy. [3]

Další velmi ceněná vlastnost wxFB je generování kódu pomocí šablon. Zde je typický příklad vytvoření tlačítka a text controlu v programu:

```
1  myButton = new wxButton(parent, ID_MYBUTTON, wxT(“Click me!”), wxDefaultPosition,
   wxDefaultSize, 0);
2  myTextCtrl = new wxTextCtrl(parent, ID_MYTEXTCTRL, wxT("My Text!"),
   wxDefaultPosition, wxDefaultSize, 0 );
```

Jak je možné vidět, vytváření tříd je velmi podobné a odehrává se pouze na jednom řádku. Proto je vhodné, aby generování kódu příslušné komponenty bylo taky pouze na jednom řádku, ale v případě typického objektově orientovaného přístupu toho není možné dosáhnout. Proto bylo vytvořeno rozhraní, které umožňuje vygenerování zdrojového kódu do sekcí jako jsou deklarace třídy, konstruktor, nastavení atd. Toto rozhraní je ve formě jednoduchého jazyku šablon, pomocí kterých je zdrojový kód generován. Tyto šablony jsou definovány v externím textovém souboru a nejsou obsaženy v kódu aplikace, v případě změny není potřeba kompilace programu. [3]

```
1 $name = new wxButton( #wxparent $name, $id, $label, $pos, $size, $style );
```

Ve wxFB je implementován rekurzivní-sestupný parser, který je schopen rozpoznat malý počet příkazů. Všechny hodnoty parametrů komponent jsou značeny znakem „$“. Například výraz $name bude v kódu nahrazen hodnotu parametru name příslušné komponenty. Tento parametr je typu text, takže parser vygeneruje hodnotu jako čistý text. Naproti tomu parametr label je typu wxString, takže parser vygeneruje hodnotu jako wxString (wxT(„MyLabel“)). Příkaz #wxparent $name znamená vygenerování hodnoty parametru name nejbližší rodičovské třídy typu wxWindow pro příslušný objekt. [3]

Všechny šablony jsou definovány v xml souboru s příponou .cppcode. Jazyk XML sice není vhodný formát pro šablony, protože neumožňuje plnou kontrolu nad syntaxí (mezery a odřádkování jsou odstraněny), ale je to takhle naimplementováno. [3]

Toto je šablona, která stačí pro vytvoření tlačítka wxButton:

```
1 <templates class="wxButton">
2   <template name="declaration">wxButton* $name;</template>
3   <template name="construction">$name = new wxButton( #wxparent $name, $id, $label,
   $pos, $size, $style );</template>
4 </templates>
```

Šablonový parser by měl být považován za nástroj ke generování kódu, nikoliv jako jediný způsob. Parser zjednodušuje práci s generováním C++ kódu, ale není vhodný na generování XRC souborů, takže generování neprobíhá v šablonách. [3]

Všechny šablony používají k identifikaci jméno a je možné definovat tolik šablon, kolik je potřeba ke generování zdrojového kódu. Pro správné generování zdrojového kódu je možno použít tyto šablony:

- Declaration: generuje deklaraci objektu v příslušné sekci
- Construction: generuje konstruktor třídy
- Settings: generuje volitelné nastavení objektu a nastavení související s dědičností
- Include: umožňuje specifikovat závislosti na hlavičkových souborech
- Base: šablona pro hlavní třídu okna
- Cons_decl: deklarace konstruktoru okna
- Cons_def: definice konstruktoru okna

Mnoho dalších šablon bylo podle potřeby vytvořeno speciálně pro nové komponenty. Ve visual editoru bylo mnoho problému vyřešeno vytvořením různých typů vizuálních komponent:

- COMPONENT_TYPE_WINDOW: pro objekty typu window.
- COMPONENT_TYPE_SIZER: pro objekty typu sizer.
- COMPONENT_TYPE_ABSTRACT: pro ostatní objekty.

V aplikaci je mnoho různých typů komponent, které dělají kód mnohem nepřehlednějším. Toto je taky důkazem, že stávající model nevyhovuje potřebám a je žádoucí nepokračovat v takto zavedeném trendu. [3]

TemplateParser je základní třídou, protože v ní nelze vytvořit hodnoty parametrů jednotlivých komponent. To je dáno konkrétním programovacím jazykem.

- CppTemplateParser konkretizuje podobu virtuálních funkcí třídy TemplateParser pro potřeby generování C++ kódu.

- CodeWriter obsahuje rozhraní pro výstup generovaného kódu. FileCodeWriter zapisuje kód do souboru a TCCodeWriter zapisuje do ovládacího prvku wxScintilla.
- CodeGenerator poskytuje běžné rozhraní pro generování kódu, avšak tato třída bůže být vynechána.
- CppPanel a XrcPanel jsou moduly, takže tyto třídy zobrazí zdrojový kód, jakmile přijmou od instance třídy ApplicationData zprávu o požadavku na zobrazení vygenerovaného kódu.

II. PRAKTICKÁ ČÁST

5 OBECNÝ POSTUP PŘIDÁVÁNÍ NOVÉ KOMPONENTY

Přidání nové komponenty do aplikace wxFormBuilder je možné rozdělit do tří částí. První část je úprava příslušného XML souboru (podle toho, na které paletě má být ikona komponenty umístěna), druhá část spočívá ve vytvoření komponenty designeru v C++ a vytvoření funkce pro generování XRC a třetí část obsahuje vygenerování kódu v jazyce C++ a Python.

5.1 Úprava XML souboru

XML soubory, které upravují vlastnosti komponent se nachází v adresáři *plugins/*prislusna_paleta*/xml/* a v adresáři xml/. V prvním adresáři se nachází jednotlivé komponenty, zatímco druhý adresář obsahuje obecné popisy (*wxWindow*), seznam ikon a typy objektů. Pro přidání např. tlačítka (*wxButton*) je potřeba editovat soubor *plugins/common/xml/common.xml*.

```
1  <objectinfo class="wxButton" icon="button.xpm" type="widget">
2      <inherits class="wxWindow"/>
3      <property name="name" type="text">m_button</property>
4      <property name="style" type="bitlist">
5        <option name="wxBU_LEFT"      help="Left-justifies the label. Windows and GTK+
   only."/>
6        <option name="wxBU_TOP"       help="Aligns the label to the top of the button.
   Windows and GTK+ only."/>
7        <option name="wxBU_RIGHT"     help="Right-justifies the bitmap label. Windows
   and GTK+ only."/>
8        <option name="wxBU_BOTTOM"    help="Aligns the label to the bottom of the
   button. Windows and GTK+ only."/>
9        <option name="wxBU_EXACTFIT"  help="Creates the button as small as possible
   instead of making it of the standard size (which is the default behaviour)."/>
10       <option name="wxNO_BORDER"    help="Creates a flat button. Windows and GTK+
   only."/>
11     </property>
12     <property name="label" type="wxString_i18n">MyButton</property>
13     <property name="default" type="bool">0</property>
14     <event name="OnButtonClick" class="wxCommandEvent" help="Process a
   wxEVT_COMMAND_BUTTON_CLICKED event, when the button is clicked" />
15   </objectinfo>
```

Každá komponenta je uvozena párovým tagem *objectinfo* s vlastnostmi *class*, *icon* a *type*. Vlastnost *clas*s obsahuje název komponenty a může být libovolná. *Icon* značí ikonu, kterou bude mít komponenta na paletě a soubor s touto ikonou se musí nacházet v adresáři ../icon/. Následuje typ komponenty, který je důležitý z hlediska určení potomků

komponenty a jejich počet. Například tlačítko (typ *widget*) musí být vloženo do sizeru (typ *sizer*) a ten musí být vložen do rámcového okna nebo dialogu (typ *frame*) atp. Základních typů objektu je několik a jsou popsány v souboru *objtypes.xml*.

```
1   <objtype name="form">
2     <childtype name="sizer" nmax="1" />
3     <childtype name="menubar" nmax="1" />
4     <childtype name="statusbar" nmax="1" />
5     <childtype name="toolbar" nmax="1" />
6   </objtype>
```

V tagu *childtype* je kromě jména potomka ještě maximální počet potomků na jednoho rodiče (parametr *nmax*). Pokud je tento parametr vynechán, není počet potomků daného typu nijak omezen.

Dalším důležitým tagem v souboru s tlačítkem je tag *inherit*s. Komponenta má kromě svých vlastností ještě vlastnosti obecné, které jsou zděděny od rodičovské třídy, jejíž název je uveden právě v tomto tagu. Ve většině případů je to *wxWindow* (kromě sizerů).

Následuje tag *property*, který určuje vlastnosti, které bude komponenta mít. Má tři parametry: *name* (jméno parametru, nesmí obsahovat mezery), *type* a *help*. Typů parametru je celá řada od základních (integer, bool, string, …) až je komplexnějším (wxPoint, wxSize, file, path, …). Celý seznam lze nalézt na konci souboru database.cpp. Poté už následuje jenom tag event, který zapouzdřuje události, které je možné provázat s touto komponentou.

5.2 Vlastní vytvoření komponenty v designeru

XML soubor je jenom popis komponenty, vlastní programování probíhá v cpp souboru. Každá paleta má svůj soubor, např. pro tlačítko je to soubor common.cpp.

Nejprve je potřeba zavolat makro WINDOW_COMPONENT se dvěma parametry. První je název komponenty (parametr class v objectinfo komponenty) a druhý je název třídy, která zapouzdřuje vytvoření tlačítka, konkrétně přetížení funkce Create rodičovské třídy ComponentBase. V případě vytváření abstaktní třídy je nutné zavolat jiné makro ABSTRACT_COMPONENT se stejnými parametry.

```
1 WINDOW_COMPONENT("wxButton",ButtonComponent)
```

Poté je potřeba vytvořit samotnou třídu, odvozenou od třídy ComponentBase, která obsahuje metodu Create. Parametry této funkce jsou instance třídy *IObject*, která zapouzdřuje vlastnosti komponenty parsované z XML souboru a instanci třídy *wxObject*, což je rodičovský prvek komponenty. Návratová hodnota funkce je objekt typu wxObject, což je v podstatě každý ovládací prvek GUI knihovny wxWidgets.

```
1  class ButtonComponent : public ComponentBase
2  {
3  public:
4    wxObject* Create(IObject *obj, wxObject *parent)
5    {
6            wxButton* button = new wxButton((wxWindow*)parent,-1,
7                    obj->GetPropertyAsString(_("label")),
8                    obj->GetPropertyAsPoint(_("pos")),
9                    obj->GetPropertyAsSize(_("size")),
10                   obj->GetPropertyAsInteger(_("style")) | obj-
   >GetPropertyAsInteger(_("window_style")));
11           if ( obj->GetPropertyAsInteger( _("default") ) != 0 )
12           {
13                   button->SetDefault();
14           }
15           return button;
16   }
17   ticpp::Element* ExportToXrc(IObject *obj)
18   {
19           ObjectToXrcFilter xrc(obj, _("wxButton"), obj-
   >GetPropertyAsString(_("name")));
20           xrc.AddWindowProperties();
21           xrc.AddProperty(_("label"),_("label"),XRC_TYPE_TEXT);
22           xrc.AddProperty(_("default"),_("default"),XRC_TYPE_BOOL);
23           return xrc.GetXrcObject();
24   }
25   ticpp::Element* ImportFromXrc( ticpp::Element* xrcObj )
26   {
27           XrcToXfbFilter filter(xrcObj, _("wxButton"));
28           filter.AddWindowProperties();
29           filter.AddProperty(_("label"),_("label"),XRC_TYPE_TEXT);
30           filter.AddProperty(_("default"),_("default"),XRC_TYPE_BOOL);
31           return filter.GetXfbObject();
32   }
33 };
```

Uvnitř funkce Create dochází k vytvoření komponenty standardní cestou. K získání vlastností komponenty slouží sada funkcí třídy IObject GetPropertyAs… s parametem typu

wxString, který obsahuje název vlastnosti. Další funkce ExportToXrc a ImportFromXrc slouží k uložení a načtení komponenty ze souboru XRC.

Pokud je potřeba s vytvořenou komponentou interaktivně pracovat, je potřeba zachytávat události generované komponentou. Příkladem může být ovládací prvek wxTextCtrl, kdy jeho vlastnost value je možné změnit jak v Properties tak přímo v designeru. K zajištění této funkcionality je potřeba nejprve ve funkci Create přidat do event handler zásobníku event handler pro text control.

```
1     wxTextCtrl* tc = new wxTextCtrl( ... );
2     tc->PushEventHandler( new ComponentEvtHandler( tc, GetManager() ) );
```

Poté je potřeba vytvořit obslužnou rutinu OnText pro zachycení události EVT_TEXT. Ke změně vlastnosti slouží funkce ModifyProperty, jejíž parametry jsou komponenta třídy wxWindow, název editovaného parametru a jeho nová hodnota.

```
1  void ComponentEvtHandler::OnText( wxCommandEvent& event )
2  {
3   wxTextCtrl* tc = wxDynamicCast( m_window, wxTextCtrl );
4   if ( tc != NULL )
5   {
6           m_manager->ModifyProperty( m_window, _("value"), tc->GetValue() );
7           tc->SetInsertionPointEnd();
8           tc->SetFocus();
9   }
10 }
```

5.3 Vygenerování kódu

Generování kódu je vytvořeno rozparsováním dvou souborů *.cppcode a *.pythoncode. Pro tyto účely byl vytvořen jednoduchý jazyk (template language), který umožňuje jednoduše zapsat generování kódu do tzv. šablon. Každá šablona zpracovává kód, který bude vložen do jiné části výsledného kódu. Pro tlačítko vypadá šablona generování takto:

```
1  <templates class="wxButton">
2      <template name="declaration">#class* $name;</template>
3      <template name="construction">$name = new #class( #wxparent $name, $id, $label,
   $pos, $size, $style #ifnotnull $window_style @{ |$window_style @} #ifnotnull
   $window_name @{, wxDefaultValidator, $window_name @} );</template>
4      <template name="settings">
5           #ifnotequal $default "0"
```

```
6           @{ $name->SetDefault(); @}
7       </template>
8       <template name="include">@#include &lt;wx/button.h&gt;</template>
9       <template name="evt_entry_OnButtonClick">EVT_BUTTON( $id, #handler )</template>
10      <template name="evt_connect_OnButtonClick">$name->Connect(
    wxEVT_COMMAND_BUTTON_CLICKED, #handler, NULL, this );</template>
11    </templates>
```

Šablona declaration obsahuje deklaraci třídy a bude vložena do hlavičkového souboru GUI, zatímco další šablony construction a settings budou v konstruktoru dialogového okna. Kromě těchto šablon jsou zde ještě šablony pro vložení hlavičkového souboru (include) a dvě šablony pro dynamické namapování událostí na obslužné rutiny.

Jazyk je velmi intuitivní. Znakem „#“ jsou uvozeny klíčová slova jazyka a znak „$“ značí název proměnné (parametru komponenty). Výraz (slovo), který neobsahuje tyto znaky je do kódu vloženo beze změny.

6 WXMEDIACTRL

Třída wxMediaCtrl slouží k přehrávání multimediálních souborů v aplikaci. Využívá nativní prostředí a nativní kodeky. V každém operačním systému je využíváno jiného prostředí tzv. backend.

Tab.č. 4: Tabulka identifikátorů backend a operačních systémů

Identifikátor prostředí	**Operační systém**
wxMEDIABACKEND_DIRECTSHOW	Windows
wxMEDIABACKEND_QUICKTIME	Mac
wxMEDIABACKEND_GSTREAMER	Unix
wxMEDIABACKEND_WMP10	Windows

wxMediaCtrl má dva konstruktory:

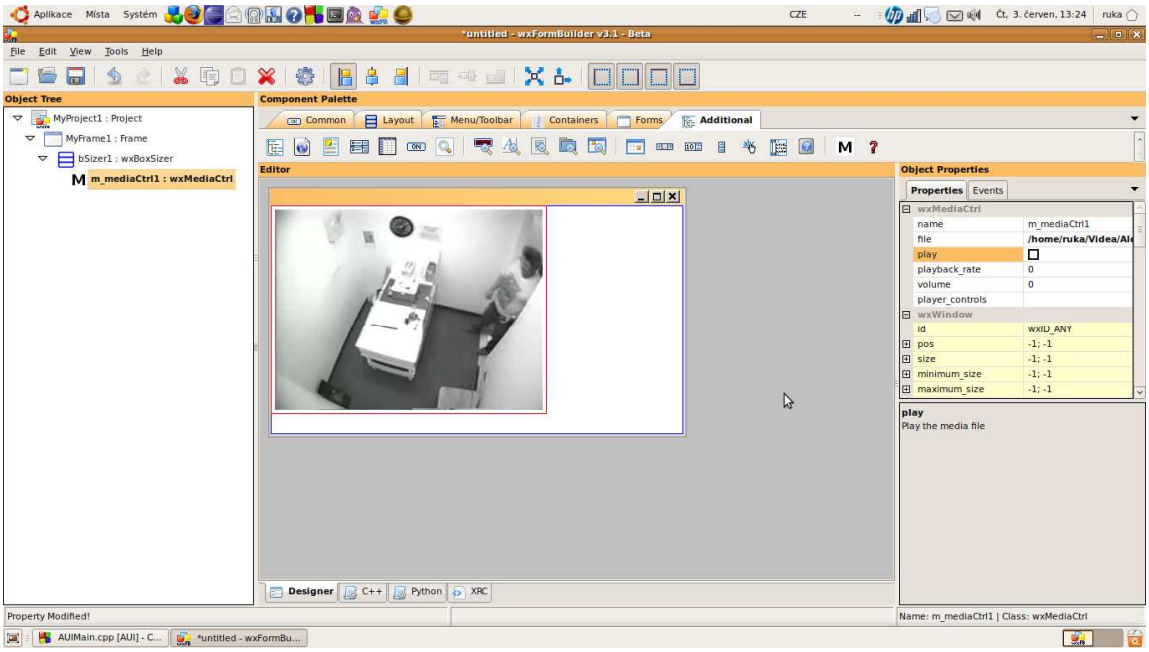
```
1 wxMediaCtrl()
2 wxMediaCtrl( wxWindow* parent, wxWindowID id, const wxString& fileName = wxT(""),
  const wxPoint& pos = wxDefaultPosition, const wxSize& size = wxDefaultSize, long
  style = 0, const wxString& szBackend = wxT(""), const wxValidatorvalidator =
  wxDefaultValidator, const wxString& name = wxPanelNameStr )
```

V případě prvního prázdného konstruktoru je potřeba zavolat funkci Create před tím, než bude volána jakákoliv další funkce. V případě druhého konstruktoru je funkce Create volána automaticky. V některých případech je doporučeno volat funkci ručně, protože na rozdíl od konstruktoru vrací Create hodnotu false, pokud nelze načíst soubor z parametru fileName, nebo není možné načíst jeden z nativních backendů pro používaný operační systém.

Do wxFB byly implementovány tyto funkce třídy wxMediaCtrl:

- *bool Load(const wxString& fileName):* načte soubor z cesty fileName. V případě, že se soubor nepodaří načíst, vrací false.
- *bool Play( ):* pokračuje v přehrávání souboru
- *bool SetPlaybackRate(double dRate):* nastaví rychlost přehrávání souboru. Není podporováno pro backend Gstreamer (Unix). Při úspěšném nastavení vrací true.

- *bool SetVolume(double dVolume):* nastaví hlasitost v rozmezí 0 – 1. Při úspěšném nastavení vrací true.
- *bool ShowPlayerControls(wxMediaCtrlPlayerControls flags):* zobrazí ovládací prvky přehrávače. Implementováno zatím pouze pro QuickTime a DirectShow backend.



Obr č. 11: Třída wxMediaCtrl ve wxFormBuilderu

6.1 Zobrazení komponenty v designeru

Zdrojový kód pro zobrazení v designeru je velmi podobný třídě wxAnimationCtrl. Navíc je potřeba do punkeru v projektu zahrnout i dvě knihovny a to libwx_gtk2u_media-2.8 a libwx_gtk2u_mmedia-2.8, které tato třída využívá. Nejprve je však potřeba přidat popis komponenty do XML souboru aditional.xml:

```
1 <objectinfo class="wxMediaCtrl" startgroup="1" icon="media_ctrl.xpm" type="widget">
2   <inherits class="wxControl" />
3   <inherits class="wxWindow" />
4           <property name="name" type="text" hlep="BlaBla1">m_mediaCtrl</property>
5           <property name="file" type="file" help="The path to media file."/>
6           <property name="play" type="bool" help="Play the media file"/>
7           <property name="playback_rate" type="float"/>
8           <property name="volume" type="float" help="Volume must be 0 - 1."/>
9           <property name="player_controls" type="option">
```

```
10                  <option name="STEP" help="wxMEDIACTRLPLAYERCONTROLS_STEP"/>
11                  <option name="VOLUME" help="wxMEDIACTRLPLAYERCONTROLS_VOLUME"/>
12                  <option name="DEFAULT" help="wxMEDIACTRLPLAYERCONTROLS_DEFAULT"/>
13                  <option name="NONE" help="wxMEDIACTRLPLAYERCONTROLS_NONE"/>
14          </property>
15   </objectinfo>
```

Komponenta wxMediaCtrl je typu widget dědí vlastnosti tříd wxControl a wxWindow. Kromě standardního názvu obsahuje taky vlastnosti file (cesta k multimediálnímu souboru), play (typu bool, určuje, jestli se má multimediální soubor po spuštění aplikace přehrávat), playback rate (rychlost přehrávání souboru) a volume (hlasitost přehrávání). Poslední důležitou vlastností je příznak player controls, kterým je možné nastavit ovládací prvky přehrávače. K dispozici jsou 4 možnosti:

- wxMEDIACTRLPLAYERCONTROLS_NONE: žádné ovládací prky, objekt wxMediaCtrl se vrátí do výchozího stavu.
- wxMEDIACTRLPLAYERCONTROLS_STEP: zobrazí ovládací prvky posunu v multimediálním souboru, například rychlé přehrávání, krokování atd.
- wxMEDIACTRLPLAYERCONTROLS_VOLUME: zobrazí ovládací prvky hlastiosti, například ikonu reproduktoru, posuvník hlasitosti atd.
- wxMEDIACTRLPLAYERCONTROLS_DEFAULT: zobrazí standardní ovládací prvky. Je to v podstatě kombinace dvou předchozích příznaků.

Po úpravě XML souboru je potřeba přidat C++ kód pro zobrazení komponenty v designeru, editací souboru aditional a vložením následujícího kódu:

```
1  class MediaCtrlComponent : public ComponentBase
2  {
3  public:
4      wxObject* Create(IObject *obj, wxObject *parent)
5   {
6          wxMediaCtrl* mc = new wxMediaCtrl((wxWindow *)parent, wxID_ANY, wxT(""),obj-
   >GetPropertyAsPoint(_("pos")),obj->GetPropertyAsSize(_("size")),    obj-
   >GetPropertyAsInteger(_("style")) | obj->GetPropertyAsInteger(_("window_style")));
7          if ( !obj->IsNull( _("file") ) )
8          {
9                  if( mc->Load( obj->GetPropertyAsString( _("file") ) ) )
10                 {
11                         if (!obj->IsNull( _("playback_rate"))) mc-
   >SetPlaybackRate(obj->GetPropertyAsFloat(_("playback_rate")));
12                         if (!obj->IsNull( _("volume")) && (obj-
   >GetPropertyAsFloat(_("volume"))>=0)&&(obj->GetPropertyAsFloat(_("volume"))<=1))
```

```
13                                      mc->SetPlaybackRate(obj->GetPropertyAsFloat(_("volume")));
14                          if (!obj->IsNull(_("player_controls")))
15                          {
16                                  if(  obj->GetPropertyAsString( _("player_controls") ) == wxT("STEP") )mc->ShowPlayerControls(wxMEDIACTRLPLAYERCONTROLS_STEP);
17                                  if(  obj->GetPropertyAsString( _("player_controls") ) == wxT("VOLUME") )      mc->ShowPlayerControls(wxMEDIACTRLPLAYERCONTROLS_VOLUME);
18                                  if(  obj->GetPropertyAsString( _("player_controls") ) == wxT("DEFAULT") )     mc->ShowPlayerControls(wxMEDIACTRLPLAYERCONTROLS_DEFAULT);
19                                  if(  obj->GetPropertyAsString( _("player_controls") ) == wxT("NONE") )mc->ShowPlayerControls(wxMEDIACTRLPLAYERCONTROLS_NONE);

20                          }
21                          if ( !obj->IsNull( _("play") ) && ( obj->GetPropertyAsInteger( _("play") ) == 1 ) ) mc->Play();
22                          else
23                                  mc->Stop();
24                  }
25          }
26          mc->PushEventHandler( new ComponentEvtHandler( mc, GetManager() ) );
27          return mc;
28   }
29   ticpp::Element* ExportToXrc(IObject *obj)
30   {
31          ObjectToXrcFilter xrc(obj, _("wxMediaCtrl"), obj->GetPropertyAsString(_("name")));
32          xrc.AddWindowProperties();
33          return xrc.GetXrcObject();
34   }
35   ticpp::Element* ImportFromXrc( ticpp::Element* xrcObj )
36   {
37          XrcToXfbFilter filter(xrcObj, _("wxMediaCtrl"));
38          filter.AddWindowProperties();
39          return filter.GetXfbObject();
40   }
41 };
42 WINDOW_COMPONENT("wxMediaCtrl",MediaCtrlComponent)
```

Po vytvoření instance třídy wxMediaCtrl program zkontroluje, jestli uživatel zadal cestu k multimediálnímu souboru. Pokud ano, pokusí se instance třídy funkcí Load tento soubor načíst. Pokud i tato část proběhne v pořádku, je možné nastavovat další parametry třídy. Nejprve je nastavena rychlost přehrávání a hlasitost, která může být pouze v intervalu 0-1. Poté následuje nastavení ovládacích prvků přehrávače, kdy aktivní může být pouze jeden. Následuje už jenom spuštění či zastavení přehrávání souboru, podle hodnoty vlastnosti

play. Následující funkce slouží k uložení (ExportToXrc) resp. načtení (ImportFromXrc) třídy do/ze souboru zdrojů XRC.

6.2 Vygenerování zdrojového kódu

Šablony pro generování zdrojového kódu se nachází v souborech aditional.cppcode a aditional.pythoncode, každý pro jiný programovací jazyk.

```
<templates class="wxMediaCtrl">
    <template name="declaration">#class* $name;</template>
    <template name="construction">
         $name = new #class( #wxparent $name, $id, wxT(""), $pos, $size);
         #ifnotnull $file
         @{ #nl $name->Load( $file );@}
         #ifnotnull $playback_rate
         @{#nl $name->SetPlaybackRate($playback_rate);@}
         #ifnotnull $volume
         @{#nl $name->SetVolume($volume);@}
         #ifnotnull $player_controls
         @{#nl $name-
>ShowPlayerControls(wxMEDIACTRLPLAYERCONTROLS_$player_controls);@}
    </template>
    <template name="settings">
         #ifequal $play "1"
         @{ #nl $name->Play(); @}
         #ifnotequal $play "1"
         @{ #nl $name->Stop(); @}
    </template>
    <template name="include">@#include &lt;wx/mediactrl.h&gt;</template>
  </templates>
```

Ve stejném pořadí jako byly nastavovány hodnoty parametrů v designeru, jsou nastaveny i při generování kódu. Příznak pro nastavení ovládacích prvků přehrávače je vytvořen tak, že první část tvoří společný řetězec pro všechny čtyři příznaky a zbytek, ve kterém se liší, má stejný tvar, jako hodnota parametru, takže nastavení parametru se vejde na jeden řádek kódu.

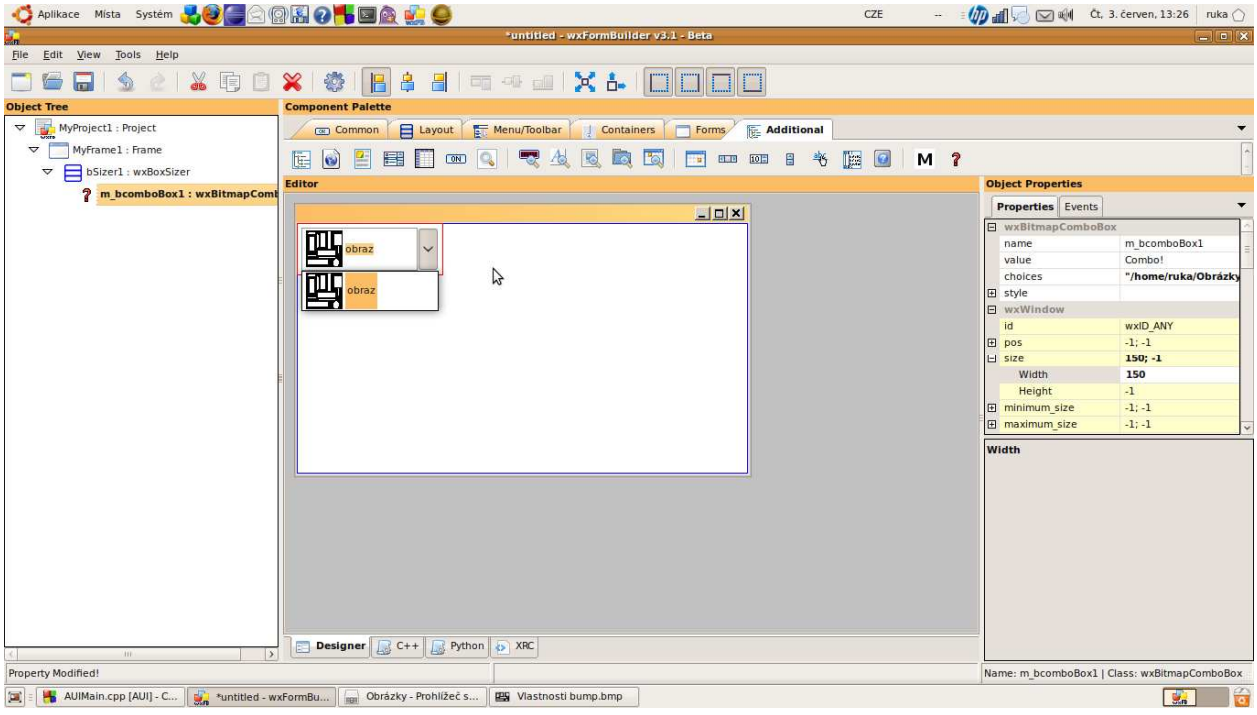
7 WXBITMAPCOMBOBOX

Třída wxBitmapComboBox je variantou třídy wxComboBox s tím rozdílem, že ke každé položce seznamu lze připojit bitmapový obrázek, který je zobrazen vedle příslušné položky.Přestože jsou s třídou wxComboBox velmi podobné, není tato třída od ní odvozena. Ve skutečnosti, pokud nemá platforma nativní implementaci, je třída wxBitmapCobmoBox odvozena od třídy wxOwnerDrawnComboBox. Je možné zjistit, jestli je implementace generická tím, že je definována konstanta wxGENERIC_BITMAPCOMBOBOX.

Konstruktor třídy wxBitmapComboBox má následující tvar:

```
1 wxBitmapComboBox(wxWindow* parent, wxWindowID id, const wxString& value = "", const
  wxPoint& pos = wxDefaultPosition, const wxSize& size = wxDefaultSize, int n = 0,
  const wxString choices[] = NULL, long style = 0, const wxValidator& validator =
  wxDefaultValidator, const wxString& name = "comboBox")
```

Je zřejmé, že stačí při volání zadat pouze dva parametry a to rodičovské okno (parent) a identifikátor (id). Kromě tohoto konstruktoru existuje ještě prázdný konstruktor, po kterém je však nutné zavolat funkce Create().



Obr č. 12: Třída wxBitmapComboBox ve wxFormBuilderu

Pro účely zobrazení komponenty v designeru byly implementovány tyto metody třídy wxBitmapComboBox:

- *int Append(const wxString& item, const wxBitmap& bitmap = wxNullBitmap):* přidává nový prvek na konec seznamu ComboBoxu. Prvek je určen názvem (item) a bitmapou (bitmap), která však nemusí být uvedena.

7.1 Zobrazení komponenty v designeru

Jako první je opět potřeba přidat popis komponenty do souboru aditional.xml:

```
<objectinfo class="wxBitmapComboBox" icon="combo_box.xpm" type="widget">
    <inherits class="wxWindow" />
    <property name="name" type="text">m_bcomboBox</property>
    <property name="value" type="wxString_i18n">Combo!</property>
    <property name="choices" type="stringlist" help="Contents of the Combo Box"></property>
    <property name="style" type="bitlist">
      <option name="wxCB_READONLY"      help="Same as wxCB_DROPDOWN but only the strings specified as the combobox choices can be selected, it is impossible to select (even from a program) a string which is not in the choices list."/>
      <option name="wxCB_SORT"          help="Sorts the entries in the list alphabetically."/>
      <option name="wxTE_PROCESS_ENTER" help="The control will generate the event wxEVT_COMMAND_TEXT_ENTER (otherwise pressing Enter key is either processed internally by the control or used for navigation between dialog controls)."/>
    </property>
  <event name="OnCombobox" class="wxCommandEvent" help="Process a wxEVT_COMMAND_COMBOBOX_SELECTED event, when an item on the list is selected. Note that calling GetValue returns the new value of selection." />
  <event name="OnText" class="wxCommandEvent" help="Process a wxEVT_COMMAND_TEXT_UPDATED event, when the combobox text changes." />
  <event name="OnTextEnter" class="wxCommandEvent" help="Process a wxEVT_COMMAND_TEXT_ENTER event, when <RETURN> is pressed in the combobox." />
   </objectinfo>
```

V tomto ohledu se od třídy wxComboBox neliší. Zadaná bitmapa nepotřebuje ke svému zpracování žádné další parametry.

Po této úpravě následuje opět vytvoření komponenty v souboru aditional.cpp:

```
class BitmapComboBoxComponent : public ComponentBase
{
public:
  wxObject* Create(IObject *obj, wxObject *parent)
  {
          wxBitmapComboBox *bcombo = new wxBitmapComboBox((wxWindow *)parent,-1,
                  obj->GetPropertyAsString(_("value")),
                  obj->GetPropertyAsPoint(_("pos")),
                  obj->GetPropertyAsSize(_("size")),
                  0,
                  NULL,
                  obj->GetPropertyAsInteger(_("style")) | obj->GetPropertyAsInteger(_("window_style")));
```

```
13          // choices
14          wxArrayString choices = obj->GetPropertyAsArrayString(_("choices"));
15          for (unsigned int i=0; i<choices.Count(); i++)
16          {
17                  wxImage img(choices[i].BeforeFirst(wxChar(58)));
18                  bcombo->Append(choices[i].AfterFirst(wxChar(58)), wxBitmap(img));
19          }
20          return bcombo;
21  }
22  ticpp::Element* ExportToXrc(IObject *obj)
23  {
24          ObjectToXrcFilter xrc(obj, _("wxBitmapComboBox"), obj-
    >GetPropertyAsString(_("name")));
25          xrc.AddWindowProperties();
26          xrc.AddProperty(_("value"),_("value"),XRC_TYPE_TEXT);
27          xrc.AddProperty(_("choices"),_("content"),XRC_TYPE_STRINGLIST);
28          return xrc.GetXrcObject();
29  }
30  ticpp::Element* ImportFromXrc( ticpp::Element* xrcObj )
31  {
32          XrcToXfbFilter filter(xrcObj, _("wxBitmapComboBox"));
33          filter.AddWindowProperties();
34          filter.AddProperty(_("value"),_("value"),XRC_TYPE_TEXT);
35          filter.AddProperty(_("content"),_("choices"), XRC_TYPE_STRINGLIST);
36          return filter.GetXfbObject();
37  }
38 };
```

Zde je opět jeden rozdíl oproti wxComboBox a to v tom, že kromě popisků položek je potřeba načítat také cesty k bitmapě. To je vyřešeno tak, že uživatel zadává parametr choice ve tvaru cesta:popis. Nejprve je funkcí třídy wxString BeforeFirst, kde je parametr znak, před nímž má být vybrán řetězec, načtena cesta k souboru s bitmapou a potom je funkcí AfterFirst se stejným parametrem vybrán popisek k položce. Nakonec je funkcemi ExportTo/ImportForm Xrc implementováno načítání do/ze souboru zdrojů XRC.

7.2 Vygenerování zdrojového kódu

Generování zdrojového kódu opět probíha pomocí šablon v souboru aditional.cppcode a aditional.pythoncode:

```
1 <templates class="wxBitmapComboBox">
2   <template name="declaration">#class* $name;</template>
3   <template name="construction">
```

```
4          $name = new #class( #wxparent $name, $id, $value, $pos, $size, 0, NULL,
   $style #ifnotnull $window_style @{ |$window_style @} #ifnotnull $window_name
           @{, wxDefaultValidator, $window_name @} );

5                  #foreach $choices

6                  @{ $name->Append( wxString(#pred).AfterFirst(wxChar(58)),
   wxBitmap(wxImage(wxString(#pred).BeforeFirst(wxChar(58))))); @}

7

8    </template>

9    <template name="evt_entry_OnCombobox">EVT_COMBOBOX( $id, #handler )</template>

10   <template name="evt_connect_OnCombobox">$name->Connect(
   wxEVT_COMMAND_COMBOBOX_SELECTED, #handler, NULL, this );</template>

11   <template name="evt_entry_OnText">EVT_TEXT( $id, #handler )</template>

12   <template name="evt_connect_OnText">$name->Connect( wxEVT_COMMAND_TEXT_UPDATED,
   #handler, NULL, this );</template>

13   <template name="evt_entry_OnTextEnter">EVT_TEXT_ENTER( $id, #handler )</template>

14   <template name="evt_connect_OnTextEnter">$name->Connect( wxEVT_COMMAND_TEXT_ENTER,
   #handler, NULL, this );</template>

15   <template name="include">@#include &lt;wx/bmpcbox.h&gt;</template>

16 </templates>
```

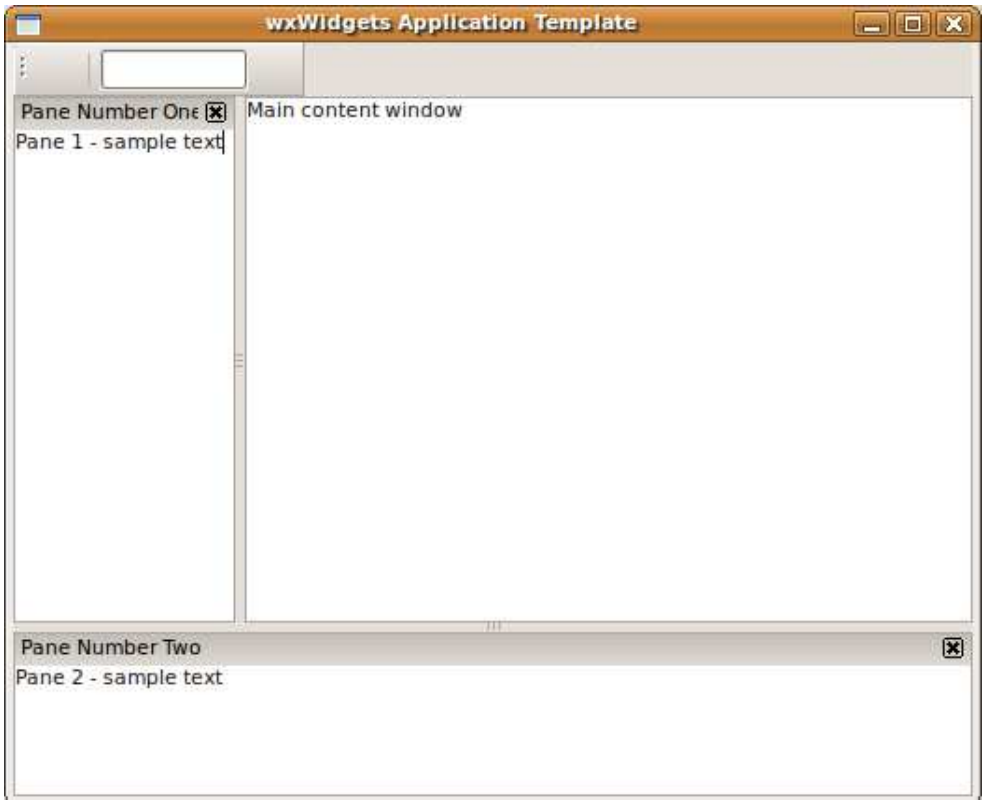
Velmi podobným způsobem, jako je vytvořeno parsování volby choice, je vytvořeno načítání cesty a popisku i v generovaném kódu.

8 WXAUI

wxAUI je vyvíjeno firmou Kirix Corporation a zapouzdřuje rozhraní pro správu tzv. dokovacích oken. Téměř každý ovládací prvek lze vložit do AuiManageru, který zajišťuje funkcionalitu těchto plovoucích či zadokovaných oken (pane).

Framework wxAUI obsahuje dvě základní třídy:

- wxAuiManager: slouží ke správě panelů asociovaných ke konkrétnímu frame-u.
- wxAuiPaneInfo: obsahuje vlastnosti konkrétního panelu, například pozice, směr dokování, velikost apod.



Obr č. 13: Vzorová aplikace za použití wxAUI

wxAuiManager pracuje následovně: programátor přidá do této třídy ovládací prvek (pane), nebo změní již existující vlastnosti panelu (dokovací pozice, plovoucí pozice, atd.). K zobrazení provedených změn je nutné zavolat funkci třídy wxAuiMnanager Update(), která tyto změny provede a zobrazí. Tato funkce by měla být volána co nejméně, aby se zabránilo problikávání. Proto je vhodné nejprve provést všechny změny najednou a potom zavolat tuto funkci.

8.1 Implementace třídy wxAuiManager do designeru

Možností jak implementovat podporu pro framework wxAUI je celá řada. Jako nejjednodušší se ukázala varianta s instancí třídy wxAuiManager pro třídy wxFrame, wxDialog a wxPanel. Vytvoření manageru probíhá ve funkci Create třídy VisualEditor:

```
1 // --- AUI
2 if(  m_form->GetObjectTypeName() == wxT("form") )
3 {
4   if(  m_form->GetPropertyAsInteger( wxT("aui_managed")) == 1)
5   {
6           m_mgr = new wxAuiManager(m_back->GetFrameContentPanel());
7   }
8 }
```

Pokud je vytvářený prvek typu frame a jeho parametr aui_managed má hodnotu true (1), pak je vytvořena dynamická instance třídy wxAuiManager a touto instancí spravovaným oknem je FrameContentPanel.

Program pokračuje vytvářením jednotlivých komponent, které jsou rozděleny na windows a sizers. Z hlediska AUI jsou důležité windows komponenty, které mohou být vloženy do AUI manageru. Pokud je vytvářená komponenta typu window, je volána funkce SetupWindow, která nastavuje obecné vlastnosti prvků. Proto je na konci této funkce vložen následující kód:

```
1 //AUI
2   if( m_mgr && ( obj->GetObjectInfo()->GetObjectType()->GetName() == wxT("widget")||
  obj->GetObjectInfo()->GetClassName()==wxT("wxAuiToolBar") ) )
3   {
4           SetupAui(obj, window);
5           m_managerUsed = true;
6   }
```

Pokud je požíván AuiManager, proměnná m_mgr obsahuje ukazatel na instanci této třídy (v opačném případě je v proměnné m_mgr hodnota NULL) a zároveň je právě vytvářený prvek typu widget nebo třídy wxAuiToolBar, je zavolána funkce SetupAui s parametry obj (třída PobjectBase, popisující vlastnosti komponenty z PropertyGridu) a window (vytvořená komponenta, odvozená od třídy wxWindow). Příznak m_managerUsed značí používání manageru a díky němu je omezeno volání funkce Update třídy wxAuiManager.

Funkce SetupAui slouží nastavení vlastností panelů, které jsou nastaveny v souboru default.xml. Tyto vlastnosti jsou přiřazeny k obecné třídě wxWindow, kde je pro ně speciální kategorie XML:

```
<category name="AUI">
        <property name="caption_visible" type="bool">1</property>
        <property name="caption" type="text" />
        <property name="close_button" type="bool">1</property>
        <property name="maximize_button" type="bool">0</property>
        <property name="minimize_button" type="bool">0</property>
        <property name="pin_button" type="bool">1</property>
        <property name="pane_border" type="bool">1</property>
        <property name="show" type="bool">1</property>
        <property name="gripper" type="bool">0</property>
        <property name="center_pane" type="bool">0</property>
        <property name="default_pane" type="bool">0</property>
        <property name="toolbar_pane" type="bool">0</property>
        <property name="moveable" type="bool">1</property>
        <property name="resize" type="option">
                <option name="Resizeable"/>
                <option name="Fixed"/>
        </property>
        <property name="pane_size" type="wxSize" />
        <property name="dock" type="option">
                <option name="Dock" />
                <option name="Float" />
        </property>
        <property name="dock_fixed" type="bool">0</property>
        <property name="floatable" type="bool">1</property>
        <property name="BottomDockable" type="bool">1</property>
        <property name="TopDockable" type="bool">1</property>
        <property name="LeftDockable" type="bool">1</property>
        <property name="RightDockable" type="bool">1</property>
        <property name="docking" type="option">
                <option name="Top"/>
                <option name="Bottom"/>
                <option name="Left"/>
                <option name="Center"/>
                <option name="Right"/>Left
        </property>
        <property name="position" type="wxSize" />
        <property name="layer" type="uint"/>
 </category>
```

Některé vlastnosti nemají nastavenou výchozí hodnotu, je to především z toho důvodu, že některé funkce, pokud jsou zavolány, se navzájem vylučují a tento „neurčitý stav“ dává uživateli lepší výchozí pozici. Většina hodnot (především typu bool) má však nastavenu výchozí hodnotu panelu, takže až v případě změny je vygenerován zdrojový kód a i když je

to otázka pouze několika písmen, je to úspora kódu. Tyto vlastnosti jsou zpracovány funkcí SetupAui:

```
1  void VisualEditor::SetupAui( PObjectBase obj, wxWindow* window )
2  {
3    m_mgr->AddPane( window );
4    if( obj->GetPropertyAsInteger( wxT("center_pane") )) m_mgr->GetPane( window
   ).CenterPane();
5    if( obj->GetPropertyAsInteger( wxT("default_pane") )) m_mgr->GetPane( window
   ).DefaultPane();
6    if( !obj->IsNull(wxT("caption"))) m_mgr->GetPane(window).Caption(obj-
   >GetPropertyAsString(wxT("caption")));
7    m_mgr->GetPane( window ).CaptionVisible( obj->GetPropertyAsInteger(
   wxT("caption_visible") ) );
8    m_mgr->GetPane( window ).CloseButton( obj->GetPropertyAsInteger( wxT("close_button")
   ) );
9    m_mgr->GetPane( window ).MaximizeButton( obj->GetPropertyAsInteger(
   wxT("maximize_button") ) );
10   m_mgr->GetPane( window ).MinimizeButton( obj->GetPropertyAsInteger(
   wxT("minimize_button") ) );
11   m_mgr->GetPane( window ).PinButton( obj->GetPropertyAsInteger( wxT("pin_button") )
   );
12   m_mgr->GetPane( window ).PaneBorder( obj->GetPropertyAsInteger( wxT("pane_border") )
   );
13   if( obj->GetPropertyAsInteger( wxT("gripper") ) ) m_mgr->GetPane( window
   ).Gripper();
14   m_mgr->GetPane( window ).BottomDockable( obj->GetPropertyAsInteger(
   wxT("BottomDockable") ) );
15   m_mgr->GetPane( window ).TopDockable( obj->GetPropertyAsInteger( wxT("TopDockable")
   ) );
16   m_mgr->GetPane( window ).LeftDockable( obj->GetPropertyAsInteger(
   wxT("LeftDockable") ) );
17   m_mgr->GetPane( window ).RightDockable( obj->GetPropertyAsInteger(
   wxT("RightDockable") ) );
18   if( !obj->IsNull(wxT("dock")) )
19   {
20           if( obj->GetPropertyAsString( wxT("dock") ) == wxT("Dock"))
21           {
22                   m_mgr->GetPane( window ).Dock();
23                   if( !obj->IsNull(wxT("docking")) )
24                   {
25                           if( obj->GetPropertyAsString(wxT("docking")) == wxT("Bottom")
   )  m_mgr->GetPane( window ).Bottom();
26                           if( obj->GetPropertyAsString(wxT("docking")) == wxT("Top") )
   m_mgr->GetPane( window ).Top();
27                           if( obj->GetPropertyAsString(wxT("docking")) == wxT("Center")
   )  m_mgr->GetPane( window ).Center();
28                           if( obj->GetPropertyAsString(wxT("docking")) == wxT("Right") )
   m_mgr->GetPane( window ).Right();
29                   }
30           }
31           else
32           {
33                   m_mgr->GetPane( window ).Float();
```

```
34                 m_mgr->GetPane( window ).FloatingPosition( obj->GetPropertyAsPoint( wxT("position") ) );
35          }
36   }
37   if( !obj->IsNull(wxT("resize")) )
38   {
39          if( obj->GetPropertyAsString( wxT("resize") ) == wxT("Resizeable")) m_mgr->GetPane( window ).Resizable();
40          else m_mgr->GetPane( window ).Fixed();
41   }
42   m_mgr->GetPane( window ).DockFixed( obj->GetPropertyAsInteger( wxT("dock_fixed") ) );
43   m_mgr->GetPane( window ).Movable( obj->GetPropertyAsInteger( wxT("movealbe") ));
44   m_mgr->GetPane( window ).Floatable(obj->GetPropertyAsInteger( wxT("floatable") ));
45   m_mgr->GetPane( window ).FloatingSize( obj->GetPropertyAsSize( wxT("pane_size") ));
46   if( obj->GetPropertyAsInteger( wxT("toolbar_pane") ) ) m_mgr->GetPane( window ).ToolbarPane();
47   if( !obj->IsNull( wxT("layer") ) )  m_mgr->GetPane( window ).Layer( obj->GetPropertyAsInteger( wxT("layer") ));
48   if( !obj->GetPropertyAsInteger( wxT("show") ) ) m_mgr->GetPane( window ).Hide();
49 }
```

Načítání parametrů panelu probíhá standardně přes zavolání funkce třídy wxAuiManager GetPane(window), která vrátí popis panelu v podobě třídy wxPaneInfo. Takto jsou načítány parametry jeden za druhým, kromě jednoho případů a to je nastavení dokování, jelikož pane může být buď zadokován na stranu (případně centrován), nebo plovoucí. Pokud je parametr dock nenulový, je otestován na řetězec Dock a v případě shody je pane zadokován příslušným směrem a v případě neshody je zavolána funkce Float a Floating position, která určuje pozici panelu.

8.2 Implementace pozice a velikosti plovoucích panelů

Jeden z prvních problémů, který se u implementace wxAUI vyskytl bylo uchování pozice a velikosti plovoucího okna. wxFB je navržen tak, že při každé změně prostředí (změna parametru, přidání komponenty, odebrání komponenty atd.) jsou všechny komponenty odstraněny a podle ObjectTree jsou s provedenými změnami opět znovu vytvořeny. Proto je potřeba změnu pozice a velikosti okna zachytit a uložit do připravených properties, které přetrvají do dalšího cyklu vytváření. Jako nejvhodnější se ukázalo místo na začátku funkce VisualEditor::Create(), těsně před počátečním smazání všech komponent:

```
1  SetPanePosition( m_back->GetFrameContentPanel() );
2  SetPaneSize( m_back->GetFrameContentPanel() );
```

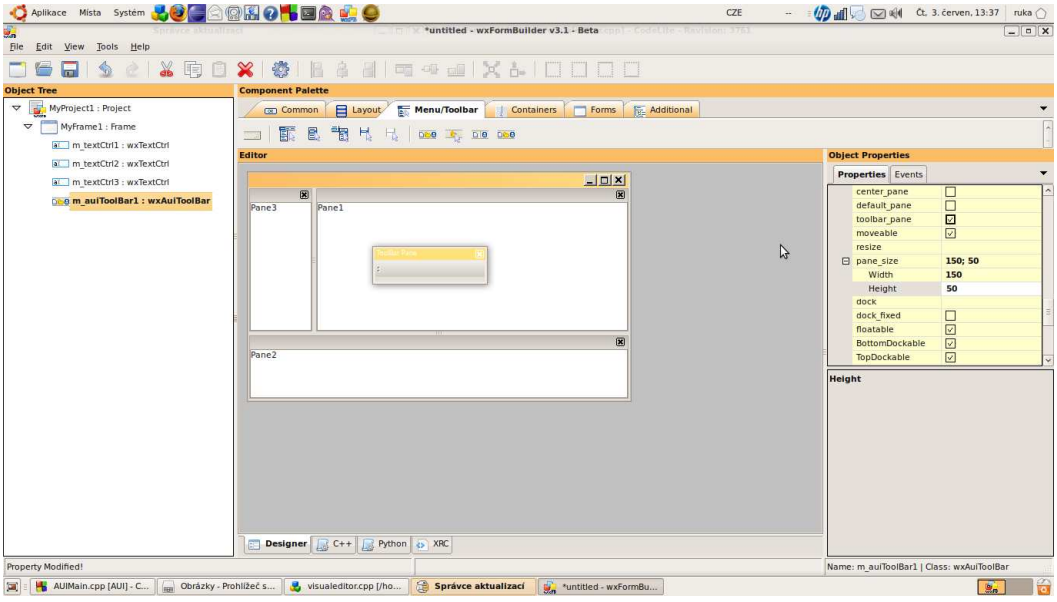
```
3   // Delete objects which had no parent
4   DeleteAbstractObjects();
5   m_managerUsed = false;
6   // Clear selections, delete objects
7   m_back->SetSelectedItem(NULL);
8   m_back->SetSelectedSizer(NULL);
9   m_back->SetSelectedObject(PObjectBase());
10  ClearComponents( m_back->GetFrameContentPanel() );
11  m_back->GetFrameContentPanel()->DestroyChildren();
12  ClearAui();
13  m_back->GetFrameContentPanel()->SetSizer( NULL );
```

K nastavení těchto vlastností panelu jsou volány dvě funkce SetPanePosition a SetPaneSize:

```
1  void VisualEditor::SetPanePosition( wxWindow* parent){
2   wxLogNull stopTheLogging;
3   const wxWindowList& children = parent->GetChildren();
4   for ( wxWindowList::const_reverse_iterator child = children.rbegin(); child !=
   children.rend(); ++child ){
5           SetPanePosition(*child);
6           PObjectBase obj = GetObjectBase( *child );
7           if ( obj ){
8           PObjectInfo obj_info = obj->GetObjectInfo();
9           wxString cname = obj_info->GetObjectType()->GetName();
10          if( cname == wxT("widget")){
11                  wxAuiPaneInfo inf = m_mgr->GetPane(*child);
12                  if(inf.IsOk()){
13                          wxString dock;
14                          if( inf.IsDocked()){
15                                  wxString dockDir;
16                                  switch(inf.dock_direction){
17                                  case 1:
18                                          dockDir = wxT("Top");
19                                          break;
20                                  case 2:
21                                          dockDir = wxT("Right");
22                                          break;
23                                  case 3:
24                                          dockDir = wxT("Bottom");
25                                          break;
26                                  case 4:
27                                          dockDir = wxT("Left");
28                                          break;
29                                  case 5:
30                                          dockDir = wxT("Center");
```

```
31                                      break; }
32                                  PProperty pdock = obj->GetProperty( wxT("docking") );
33                                  AppData()->ModifyProperty( pdock, dockDir );
34                                  dock = wxT("Dock");}
35                              else{
36                              wxPoint pos = inf.floating_pos;
37                              if ( pos.x != -1 && pos.y != -1 ){
38                              PProperty pposition = obj->GetProperty( wxT("position") );
39                              AppData()->ModifyProperty( pposition, TypeConv::PointToString(
    pos ) );}
40                              dock = wxT("Float");}
41                              PProperty pfloat = obj->GetProperty(wxT("dock") );
42                              AppData()->ModifyProperty( pfloat, dock);}}}}}
```

Parametrem této funkce je rodičovská třída, jejíž potomky je potřeba dále zpracovat. Pomocí reverzního iterátoru třídy wxWindow jsou jednotlivé prvky procházeny a rekurzivně nad nimi volána funkce SetPanePosition. Potom je k jednotlivý potomkům přiřazen odpovídající ObjectBase a ObjectInfo. Pokud je potomek typu widget nebo třídy wxAuiToolBar, je zjištěna a uložena jeho plovoucí pozice, nebo dokovací směr a ten je uložen do příslušné property. Dokovací směr je uložen v proměnné dock_direction, který je typu int, takže k jeho zpracování lze použít příkaz switch. Poté je zjištěna z proměnné floating_pos pozice plovoucího okna a pokud není definována (hodnoty x a y jsou -1) není tato pozice dál nijak zpracovávána.



Obr č. 14: Návrh GUI za použití frameworku wxAUI

Velmi podobná je i funkce pro nastavení velikosti panelu:

```
1  void VisualEditor::SetPaneSize( wxWindow* parent)
2  {
3   wxLogNull stopTheLogging;
4   const wxWindowList& children = parent->GetChildren();
5   for ( wxWindowList::const_reverse_iterator child = children.rbegin(); child != children.rend(); ++child )
6   {
7           SetPaneSize(*child);
8           PObjectBase obj = GetObjectBase( *child );
9           if ( obj )
10          {
11                  PObjectInfo obj_info = obj->GetObjectInfo();
12                  wxString cname = obj_info->GetObjectType()->GetName();
13                  if( cname == wxT("widget"))
14                  {
15                          wxAuiPaneInfo inf = m_mgr->GetPane(*child);
16                          if(inf.IsOk())
17                          {
18                                  wxSize paneSize = inf.floating_size;
19                                  if ( paneSize.x != -1 && paneSize.y != -1 )
20                                  {
21                                          PProperty pSize = obj->GetProperty( wxT("pane_size") );
22                                          AppData()->ModifyProperty( pSize, TypeConv::SizeToString( paneSize ) );
23                                          AppData()->ModifyProperty( obj->GetProperty( wxT("resize") ), wxT("Resizeable"));
24                                  }
25                          }
26                  }
27          }
28   }
29 }
```

Parametrem funkce je opět rodičovská třída, jejíž potomky je potřeba dál zpracovat. Stejně jako funkce SetPanePosition je i tato funkce volána rekurzivně. Po ověření typu a třídy komponenty je z proměnné floating_size získána velikost panelu. Pokud není výchozí (x i y jsou -1), je tato hodnota uložena do příslušné property pane_size.

8.3 Další nutné úpravy

Při návrhu okna aplikace za použití frameworku wxAUI se nedoporučuje používaní sizerů, ale vkládání objektů (widgetů) přímo do hlavního okna aplikace. To však wxFB neumožňuje – při pokusu o vložení widgetu přímo do okna je zobrazen dialog se zprávou,

že tato akce není možná a jestli uživatel nezapomněl vytvořit sizer. Toto je třeba obejít a umožnit tak uživateli při použití wxAUI vkládat widgety přímo.

Jak již bylo zmíněno výše, tyto vazby jsou uloženy v souboru objtypes.xml.

```
<objtype name="form">
  <childtype name="sizer" nmax="1" />
  <childtype name="menubar" nmax="1" />
  <childtype name="menu" nmax="1" />
  <childtype name="statusbar" nmax="1" />
  <childtype name="toolbar" nmax="1" />
  <childtype name="widget" />
</objtype>
```

Proto tedy bylo do popisu typu frame vložen řádek, který dovoluje vložit widget přímo do frame. To ovšem umožní vkládání i bez využití wxAUI, což však není v souladu s filosofií designu wxFB. Aby bylo možné vkládání widgetu do framu pouze ve wxAUI, je potřeba vložit jeden řádek do souboru database.cpp a funkci ObjectDatabase::CreateObject:

```
if( !parent->GetPropertyAsInteger(wxT("aui_managed")) && objType->GetName() ==
wxT("widget") ) max = 0;
```

Touto funkcí je ručně měněna proměnná nmax, podle které je možné určit, zda je povoleno vytvoření prvku v závislosti na jeho rodiči. Jestli je hodnota nulová, není dovoleno vytvořit prvek, v případě hodnoty větší než 0 je dovoleno vytvořit právě tolik prvků, jaká je hodnota a pokud je hodnota -1, není tento počet nijak omezen. Takže pokud má rodičovská třída parametr aui_managed nulový a zároveň je typu widget, je hodnota nmax nastavena na 0, čímž je zabráněno jeho vytvoření.

Další problém, který je potřeba odchytit, je změna parametru aui_managed v případě, že už existuje v projektu několik objektů. To lze vyřešit editací souboru objinspect.cpp a funkce ObjectInspector::OnPropertyGridChange:

```
if( prop->GetName() == wxT("aui_managed") )
 {
        PObjectBase propobj = prop->GetObject();
        if( propobj->GetChildCount() )
        {
               wxMessageBox(wxT("You have to remove all child widgets first."));
               AppData()->ModifyProperty( prop, event.GetPropertyValueAsBool() ?
wxT("0") : wxT("1") );
        }
        else AppData()->ModifyProperty( prop, event.GetPropertyValueAsBool() ?
wxT("1") : wxT("0") );
 }
```

Pokud má tedy změněná vlastnost název aui_managed, je přes tuto vlastnost získán objekt, který obsahuje tuto vlastnost. Objekt obsahuje funkci GetChildCount, která spočítá počet potomků objektu a pokud je nenulový, není dovoleno změnit tuto vlastnost.

8.4 Generování kódu frameworku wxAUI

Vložení kódu pro wxAUI je nutné provést v několika šablonách. V šabloně třídy frame (soubor forms.cppcode) je podle parametru aui_managed vygenerován kód pro třídu wxAuiManaged. Nejprve je potřeba třídu deklarovat v šabloně cons_decl:

```
<template name="cons_decl">
                ...
 #ifequal $aui_managed "1"
 @{ m_mgr wxAuiManager(); #nl @}
 </template>
```

Následuje konstrukce samotné třídy wxAuiManager v šabloně settings toho samého souboru:

```
<template name="settings">
                ...
                #ifequal $aui_managed "1"
                @{ m_mgr.SetManagedWindow(this); #nl @}
</template>
```

Přidání ovládacího prvku do manageru je potřeba vytvořit ve třídě wxWindow, konkrétně v souboru default.cppcode:

Možností jak přidat vlastnosti panelu je několik. Jedna z možností je zavolat funkci GetPane, která vrátí třídu wxPaneInfo, nad kterou lze zavolat funkce, které tyto vlastnosti nastaví. Díky jednoduchému jazyku šablon lze však nastavit už ve funkci AddPane, která přidává ovládací prvek do AuiManageru.

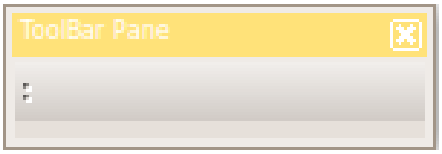
```
<templates class="wxWindow">
        <template name="settings">
                ...
                #ifequal #parent $aui_managed "1"
                @{
                        m_mgr.AddPane( $name, wxAuiPaneInfo().Name( "$aui_name"
).$docking()#ifnotnull $caption
                        @{.Caption( "$caption" )@}#ifequal $caption_visible "0"
                        @{.CaptionVisible( $caption_visible )@}#ifequal $close_button
"0"
                        @{.CloseButton( $close_button )@}#ifequal $maximize_button "0"
                        @{.MaximizeButton( $maximize_button )@}#ifequal
$minimize_button "0"
```

```
16                      @{.MinimizeButton( $minimize_button )@}#ifequal
    $minimize_button "0"
17                      @{.PinButton( $pin_button )@}#ifequal $pane_border "0"
18                      @{.PaneBorder( $pane_border )@}#ifequal $gripper "1"
19                      @{.Gripper()@}#ifequal $show "0"
20                      @{.Hide()@}#ifequal $moveable "0"
21                      @{.Movable( $moveable )@}#ifnotnull $dock
22                      @{.$dock()#ifequal $dock "Float"
23                              @{.FloatingPosition( $position )@}
24                      @}#ifnotnull $resize
25                      @{.$resize()#ifequal $resize "Resizeable"
26                              @{.FloatingSize( $pane_size )@}
27                      @}#ifequal $dock_fixed "0"
28                      @{.DockFixed( $dock_fixed )@}#ifequal $BottomDockable "0"
29                      @{.BottomDockable( $BottomDockable )@}#ifequal $TopDockable
    "0"
30                      @{.TopDockable( $TopDockable )@}#ifequal $LeftDockable "0"
31                      @{.LeftDockable( $LeftDockable )@}#ifequal $RightDockable "0"
32                      @{.RightDockable( $RightDockable )@}#ifequal $floatable "0"
33                      @{.Floatable( $floatable )@}#ifnotnull $layer
34                      @{.Layer( $layer )@}#ifequal $center_pane "1"
35                      @{.CentrePane()@}#ifequal $default_pane "1"
36                      @{.DefaultPane()@}#ifequal $toolbar_pane "1"
37                      @{.ToolbarPane()@}
38                      ) #nl
39                  @}
40          </template>
```

Generování kódu je napsáno tak, že je maximálně využíváno implicitních nastavení pro pane, takže zdrojový kód je vytvářen pouze tam, kde se hodnoty zadané uživatelem liší od těch implicitních. Tento způsob zápisu umožní vygenerování zdrojového kódu pouze na jeden řádek pro každou komponentu.

9 WXAUITOOLBAR

Třída wxAuiToolBar je skoro totožná, jako třída wxToolBar. Rozdíl je v několika málo funkcích, které jsou navíc přidány a ve způsobu vykreslení – ten je realizován přes framework wxAUI. Pro použití wxAuiToolBar je potřeba přidat hlavičkový soubor wx/aui/auibar.h.



Obr č. 15: wxAuiToolBar pane

Tato třída je stejně jako celé wxAUI vyvíjena třetí stranou (Kirix Coproration), takže zatím není v dokumentaci na oficiálních stránkách wxWidgets.

9.1 Zobrazení komponenty v designeru

Vlastnosti komponenty jsou popsány stejně jako pro třídu wxToolBar a jsou umístěny v souboru menutoolbar.xml:

```
1  <objectinfo class="wxAuiToolBar" icon="toolbar.xpm" type="toolbar">
2      <inherits class="wxWindow" />
3      <property name="name" type="text">m_auiToolBar</property>
4      <property name="bitmapsize" type="wxSize" help="Default size of each tool bitmap.
   The default bitmap size is 16 by 15 pixels." />
5      <property name="margins" type="wxSize"    help="Values to be used as margins for
   the toolbar." />
6      <property name="packing" type="uint"      help="Value used for spacing tools.
   Remarks: The packing is used for spacing in the vertical direction if the toolbar is
   horizontal, and for spacing in the horizontal direction if the toolbar is
   vertical.">1</property>
7      <property name="separation" type="uint"   help="The default separator size.
   Remarks: This is the size between each toolbar tool.">5</property>
8      <property name="style" type="bitlist">
9        <option name="wxTB_FLAT"               help="Gives the toolbar a flat look
   (Windows and GTK only)." />
10       <option name="wxTB_DOCKABLE"            help="Makes the toolbar floatable and
   dockable (GTK only)." />
11       <option name="wxTB_HORIZONTAL"          help="Specifies horizontal layout." />
12       <option name="wxTB_VERTICAL"            help="Specifies vertical layout." />
13       <option name="wxTB_TEXT"                help="Shows the text in the toolbar
   buttons; by default only icons are shown." />
14       <option name="wxTB_NOICONS"             help="Specifies no icons in the toolbar
   buttons; by default they are shown." />
15       <option name="wxTB_NODIVIDER"           help="Specifies no divider (border)
   above the toolbar (Windows only)." />
16       <option name="wxTB_NOALIGN"             help="Specifies no alignment with the
   parent window (Windows only, not very useful)." />
```

```
17       <option name="wxTB_HORZ_LAYOUT"         help="Shows the text and the icons
    alongside, not vertically stacked (Windows and GTK 2 only). This style must be used
    with wxTB_TEXT." />
18       <option name="wxTB_HORZ_TEXT"           help="Combination of wxTB_HORZ_LAYOUT
    and wxTB_TEXT." />wxTB_HORIZONTAL</property>
19   </objectinfo>
```

Mezi zahrnuté vlastnosti třídy patří parametr bitmapsize, který určuje velikost bitmapy přidaného prvku Tool. Mezi další parametry patří margin a packing, které určují velikost mezery mezi toolbary resp jednotlivými ikonami nástrojů. Jako poslední jsou určeny velikost separátoru (svislice mezi ikonami nástrojů) a styl ovládacího prvku. Nové funkce, které třída obsahuje na rozdíl od wxToolBar, nejsou při návrhu dost dobře použitelné, takže jejich volání a používání musí programátor obsloužit sám.

Samotné vytvoření je pak realizováno v souboru common.cpp, kde jsou načteny a nastaveny všechny vlastnosti popsané v XML:

```
1  wxObject* Create(IObject *obj, wxObject *parent)
2   {
3           wxAuiToolBar *tb = new wxAuiToolBar((wxWindow*)parent, -1,
4                   obj->GetPropertyAsPoint(_("pos")),
5                   obj->GetPropertyAsSize(_("size")),
6                   obj->GetPropertyAsInteger(_("style")) | obj-
    >GetPropertyAsInteger(_("window_style")) | wxTB_NOALIGN | wxTB_NODIVIDER |
    wxNO_BORDER);
7           if (!obj->IsNull(_("bitmapsize")))
8                   tb->SetToolBitmapSize(obj->GetPropertyAsSize(_("bitmapsize")));
9           if (!obj->IsNull(_("margins")))
10          {
11                  wxSize margins(obj->GetPropertyAsSize(_("margins")));
12                  tb->SetMargins(margins.GetWidth(), margins.GetHeight());
13          }
14          if (!obj->IsNull(_("packing")))
15                  tb->SetToolPacking(obj->GetPropertyAsInteger(_("packing")));
16          if (!obj->IsNull(_("separation")))
17                  tb->SetToolSeparation(obj->GetPropertyAsInteger(_("separation")));
18          tb->PushEventHandler( new ComponentEvtHandler( tb, GetManager() ) );
19          return tb;
20   }
```

9.2 Vygenerování zdrojového kódu

Šablony pro vygenerování zdrojového kódu pro třídu wxAuiToolBar se nachází v souboru menutoolbar.cppcode. a obsahují čtyři části declaration, include (třída je zapouzdřena v souboru wx/aui/auibar.h), construction a after_addchild (po vložení všech prvků je třeba zavolat funkci Realize).

```
<templates class="wxAuiToolBar">
    <template name="declaration">#class* $name;</template>
    <template name="include">@#include &lt;wx/aui/auibar.h&gt;</template>
    <template name="construction">
         #ifnotnull $bitmapsize @{ #nl $name->SetToolBitmapSize( $bitmapsize ); @}
         #ifnotnull $separation
         @{
           #ifnotequal $separation "5"
           @{ #nl $name->SetToolSeparation( $separation ); @}
         @}
         #ifnotnull $margins    @{ #nl $name->SetMargins( $margins ); @}
         #ifnotnull $packing
         @{
                #ifnotequal $packing "1"
                @{ #nl $name->SetToolPacking( $packing ); @}
         @}
    </template>
    <template name="after_addchild">
     #ifparenttypeequal "form"
      @{
         #ifequal #parent $aui_managed "1"
                @{
                        ...
                @}
      @}
    </template>
  </templates>
```

Přídání třídy wxAuiToolBar do AuiManageru je realizováno až na konci vytváření třídy. Ja to z toho důvodu, že před vložením do manageru je nutné nejprve realizovat přidání všech ovládacích prvků k toolbaru.

ZÁVĚR

Cílem této práce bylo vytvořit rešerši na téma RAD nástroje pro tvorbu GUI aplikací s využitím softwarové knihovny wxWidgets a přidání nových prvků wxMediaCtrl, wxBitmapComboBox a kompletní podporu pro wxAUI do multiplatformní open-source aplikace wxFormBuilder.

V teoretické části byla nejprve popsána softwarová knihovna wxWidgets a její základní vlastnosti. Další kapitola se zabývá základním popisem několika designerů a vývojových prostředí, které jsou k dispozici pro tvorbu aplikací s knihovnou wxWidgets. Mezi tyto aplikace patří wxFormBuilder, wxDev-C++, wxGlade, wxSmith, wxDesigner a DialogBlocks. Každá aplikace má své specifické vlastnosti, které jsou srovnány v další kapitole. DialogBlocks a wxDesigner jsou proprietární aplikace s volně stažitelnou trialovou verzí, zatímco ostatní jsou volně dostupné, open-source a vydané pod licencí GPL. Všechny tyto aplikace dokážou generovat zdrojový kód v jazyce C++ a kromě wxSmith i Python. Podpora pro ostatní jazyky je nulová, výjimku tvoří wxDesigner, který umí generovat zdrojové kódy pro jazyky Perl a C#. V poslední kapitole teoretické části je popsána vnitřní struktura aplikace wxFormBuilder.

V praktické části je popsán obecný postup pro přidávání komponent do aplikace wxFormBuilder. K tomu jsou zapotřebí tři kroky: nejprve popsat v jazyce XML vlastnosti prvku, které budou zobrazeny v tabulce vlastností, potom naprogramovat kód v jazyce C++ pro vytvoření komponenty v designeru a nakonec vytvoření tzv. šablony pro generování zdrojového kódu pro dané jazyky. Třída wxMediaCtrl slouží k přehrávání multimediálních souborů, za pomocí nativního backendu pro daný operační systém. Tato třída má několik nedostatků, zvláště v případě unixového backendu GStreamer.Komponenta vyžaduje externí multimediální knihovny wxWidgets. Třída wxBitmapComboBox je velmi podobná třídě wxComboBox s tím rozdílem, že vedle položky seznamu je zobrazena i bitmapa. wxFB obsahuje malé rozhraní pro zadávání položek v seznamu, které je použito i pro třídu wxBitmapComboBox. Bitmapa i popisek jsou zadávány v jednom řetězci a jsou odděleny znakem „:“. Tento řetězec je potom v programu rozparsován. Další kapitola je zaměřena na implementaci wxAUI. To je realizováno přidáním třídy wxAuiManager do wxFB. Tato třída je vytvořena po zaškrtnutí parametru aui_managed ve vlastnostech dialogového okna. Každý ovládací prvek je potom funkcí AddPane přiřazen manageru. V případě wxAUI bylo potřeba udělat výraznější zásahy do kódu aplikace. Nejprve bylo nutné vyřešit vkládání

komponent (widgetů) přímo do okna bez sizerů. To bylo dosaženo editováním souboru objtypes.xml, kde jsou tyto vztahy rodič-potomek popsány a úpravou database.cpp, kde jsou tyto vztahy zpracovávány. Nakonec bylo potřeba zajistit, aby nebylo možné zrušit parametr aui_managed, dokud má okno pod sebou widgety. Jako poslední byla do aplikace přidána třída wxAuiToolBar, která se moc neliší od třídy wxToolBar, pouze je vykreslena frameworkem wxAUI.

Upravené zdrojové kódy byly otestovány na zadaných platformách a pro komponenty bylo implementováno generování zdrojového kódu pro C++, Python a XRC (kromě wxAUI, kde není XRC zatím podporováno).

ZÁVĚR V ANGLIČTINĚ

The aim of this thesis was to create a background research about RAD tool for creation of GUI applications with usage of software library wxWidgets and to add new elements wxMediaCtrl, wxBitmapComboBox and full support for wxAUI to wxFormBuilder, which is a cross-platform, open-source GUI designer.

In theoretical part, there was described some basic features of software library wxWidgets. Next chapter deals with description of several designers and development enviroments, using wxWidgets library. These applicatons are: wxFormBuilder, wxDev-C++, wxGlade, wxSmith, wxDesigner and DialogBlocks. Each of these applications has its own specific features and functions, which are compared in the next chapter. DialogBlocks and wxDesigner are proprietal software with free trial version and others are free and open-source and released under GPL license. All of these tools can generate a C++ source code and Python code except wxSmith. There is a small support for other languages, one exception is wxDsigner, which can generate Perl and C# source code. In the last chapret of theoretical part is described internal structure of wxFormBuilder.

In practical part is described adding new components to wxFormBuildes as a general process. This can be divided into three parts: property description of a new component in XML language, creating C++ code for inserting a new component into a designer window and finally creating templates for C++ and Python source code generation. wxMediaCtrl class can play a media files, using native backend for each operating systém. This class has several disadvanages, one of them is support for GStreamer backend isn't complete. The component requires external wxWidgets media libraries. wxBitmapComboBox class is very similar to wxComboBox class, with one difference. A bitmap is displayed next to the label of choice. wxFB has a small interface for inserting choices to list, which is used also for wxBitmapComboBox. Bitmap and label are inserted in one string, sepereted by „:“ character. This string parsed in the program. Next chapter is aimed for wxAUI implementation. AUI is realized by wxAuiManager class, which is created after aui_managed property is checked in property grid. Each contol is added to AuiManager by AddPane function. wxAUI implementation required more complex changes of wxFormBuilder source code. First, adding widgets to forms without sizers had to be solved. This was achieved by editing objtypes.xml file, where are parent-children relations are described and database.cpp, where these relations are procesed. Finally, correct

cheching and unchecking of aui_managed property had to me ensured. It cannot be unchecked, if the frame has child widgets. As last wxAuiToolBar was added. This class has small difference between it and wxToolBar class. wxAuiToolBar is drawn by wxAUI framework.

Modified source codes were tested on required platforms and C++, Python and XRC source code generating for new componets was implemented, except wxAUI, where XRC isn’t supported yet.

SEZNAM POUŽITÉ LITERATURY

[1] BLIŽŇÁK, Michal, „Systémové programování“, UTB ve Zlíně, 2005, ISBN 80-7318-364-1

[2] Internetové stránky aplikace wxFormBuilder [online], 2007 [cit. 2010-6-2], Dostupný z WWW: <http://wiki.wxformbuilder.org/Main/HomePage>

[3] Vnitřní struktura aplikace wxFormBuilder [online], 2006 [cit. 2010-6-2], Dostupný z WWW: <http://wiki.wxformbuilder.org/Main/WxFormBuilderInternals>

[4] Internetové stránky pluginu wxSmith [online], 2010 [cit. 2010-6-2], Dostupný z WWW: <http://wiki.codeblocks.org/index.php?title=Comparison_of_wxSmith_features>

[5] Internetové stránky aplikace wxDev-C++ [online], 2008 [cit. 2010-6-2], Dostupný z WWW: <http://wxdsgn.sourceforge.net/>

[6] Internetové stránky aplikace wxGlade [online], 2009 [cit. 2010-6-2], Dostupný z WWW: <http://wxglade.sourceforge.net/>

[7] Internetové stránky aplikace wxDesigner [online], 2009 [cit. 2010-6-2], Dostupný z WWW: <http://www.wxdesigner-software.de/info.html>

[8] http://en.wikipedia.org/wiki/Rapid_application_development

[9] Internetové stránky aplikace DialogBlocks [online], 2008 [cit. 2010-6-2], Dostupný z WWW: <http://www.dialogblocks.com/index.htm>

[10] SMART, Julian, HOCK, Kevin, Cross-Platform GUI Programming with wxWidgets, Patience Hall, 2006, ISBN 0-13-147381-6

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

GUI Graphics User Interface – Grafické uživatelské rozhraní

RAD Rapid Application Development – Zrychlený vývoj aplikace

wxFB wxFormBuilder

AUI Advanced User Interface – Pokročilé uživatelské rozhraní

IDE Integrated Development Environment – Integrované vývojové prostředí

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD-ROM

Přiložené CD obsahuje tuto práci v elektronické podobě (formát PDF), zdrojové kódy aplikace wxFormBuilder a tutoriál na používání nových funkcí pomocí aplikace ScreenToaster.