

# Zápis uživatelského rozhraní v XML

(projekt do předmětu SCS)

# Obsah

Obsah .....	1
Zadání projektu .....	2
Společná část popisu .....	2
Popis varianty .....	2
Výchozí podmínky .....	3
Jednoduchost zpracování strojem .....	3
Jednoduchost ručních úprav .....	3
Rozšiřitelnost .....	3
Tvorba uživatelského rozhraní .....	4
Statické rozhraní .....	4
Dynamické rozhraní .....	5
Metoda tvorby .....	5
Praktická ukázka .....	6
Postup .....	6
Zápis v XML .....	7
Konverze do HTML a CSS .....	8
Detailnější popis formátu .....	9
Aplikace .....	9
Okno .....	9
Komponenty .....	10
Strom komponent .....	11
Závěr .....	13
Použitá literatura .....	14

## Zadání projektu

### Společná část popisu

Projekty mohou řešit jednotlivci nebo skupiny nejvýše o 3 osobách. V případě vypracování projektu skupinou je třeba při odevzdání projektu přesně popsat role řešitelů v projektu. Projekt se typicky hodnotí stejně pro členy skupiny, ale je vyhrazena i možnost individuální hodnocení uvnitř skupiny odlišit. Je třeba odevzdat zdrojové texty, spustitelný projekt (jsou-li předmětem řešení) a zprávu v rozsahu cca 3 stran A4. Programovací jazyk je libovolný.

Individuální vlastní zadání jsou vítána. V případě, že o individuální zadání máte zájem, přihlaste se na variantu "Vlastní zadání" a pošlete e-mail s návrhem zadání - další postup bude individuální.

### Popis varianty

Vytvořte společný formát pro zápis uživatelského rozhraní aplikací různých platforem (Linux, Web, Windows). Formát musí splňovat následující požadavky:

- jednoduchost zpracování strojem
- jednoduchost ručních úprav
- rozšiřitelnost

Demonstrujte použití navrhnutého zápisu na generování jednoduchého rozhraní pro Web (transformace XML na HTML a CSS).

## Výchozí podmínky

Důvod proč je potřeba společného formátu je ten, že v dnešní době jsou aplikace závislé na použitém operačním systému (Linux vs. Windows) nebo grafické nadstavbě (GNOME vs. KDE) či knihovnách (OpenGL vs. DirectX). Jenom malé procento aplikací je tvořeno tak, aby byly přenositelné a této vlastnosti nabyly někdy až po tom jak se staly úspěšnými (Mozilla využívající QT se objevila jen nedávno).

To, že aplikace nejsou přístupné pro všechny systémy, může mít několik důvodů. Jedním z nich je přílišná složitost přenositelné implementace, což není jen problém začínajících programátorů kteří píšou zcela nové programy, ale také ostatních, co se snaží udělat z nepřenositelných programů přenositelné. Dalšími důvody jsou třeba ty marketingové, související s obchodní politikou výrobce daného programového vybavení.

Tato práce se zabývá řešením problému přenositelnosti uživatelského rozhraní pomocí definice dalšího, společného, formátu na jeho zápis. Jako formát dat bylo nejlepší zvolit XML, které je univerzální a nezávislé na platformě a splňuje všechny požadavky ze zadání projektu. Dalšími formáty které připadaly do úvahy byly binární zápis a textový zápis v nějakém novém jazyku pro zápis uživatelského rozhraní. Hodnocení vlastností je v následující tabulce a detaily jsou uvedeny níže.

	XML	Textový formát	Binární formát
Strojové zpracování	1	3	2
Ruční úpravy	2	3	nemožné
Rozšiřitelnost	1	3	3

Hodnocení: menší číslo značí lepší hodnocení.

## Jednoduchost zpracování strojem

Jednoduchost XML spočívá v standardizaci formátu. Proto s daty v XML můžeme pracovat např. v jazyku PHP velice jednoduše takto:

```
<?php
    if ($app = simplexml_load_file('./gui.xml')) {
        echo "Název zpracovávané aplikace je ", $app->title, "\n";
    }
?>
```

Kdyby to byl binární formát, tak ho načteme pomocí několika cyklů do struktur, což není až tak složitá operace, ale musíme si naprogramovat funkce na načítání a ukládání. V případě textového formátu je situace nejhorší protože musíme sestavit gramatiku jazyka ve kterém se ten textový popis napsal a pak sestavit analyzátor. Nejrozumnější cesta tedy vede na použití XML.

## Jednoduchost ručních úprav

Pokud je požadavek na ruční úpravy, tak prakticky neexistuje žádný zázračný formát na zápis a teoreticky by pomohla snad jenom umělá inteligence. V případě binárního formátu můžeme rovnou zapomenout na nějaké ruční úpravy dat, takže zůstává XML a text. U obojího je sice potřeba znát problematiku, ale v konečném důsledku to bude horší u textového formátu. XML znova vede, protože kromě dat obsahuje v sobě taky meta-data (názvy elementů). Textový formát obsahuje většinou jenom data které jsou seskupeny nějakou (pro začátečníka neznámou) syntaxí.

## Rozšiřitelnost

Problém rozšiřitelnosti v XML prakticky nejstuvuje, protože XML bylo koncipováno tak od počátku a má to taky v názvu (eXtensible Mark-up Language). Rozšíření textového nebo binárního formátu je víceméně stejně složité a způsobí porušení zpětné kompatibility dokumentů. Je zřejmé, že by bylo možné navrhnout univerzální formáty, jak binární tak textové, ale to není cílem tohoto porovnání ani projektu jak celku.

## Tvorba uživatelského rozhraní

Aplikace, které mají nějaké uživatelské rozhraní (tzv. interaktivní aplikace) se dají reálně rozdělit do dvou skupin: s grafickým a textovým rozhraním. Tento projekt se zabývá pouze o grafické uživatelské rozhraní a i to jen ty, které jsou založené na principu oken a komponent.

### Statické rozhraní

U jednoduchých aplikací se většinou uživatelské rozhraní vytváří staticky, tj. v době návrhu nebo implementace aplikace. Příkladem může být třeba kalkulačka, která vypadá stejně po celou dobu svého běhu, mění se jen číslo na displeji, což je atribut komponenty, ne komponenta nebo struktura rozhraní aplikace.

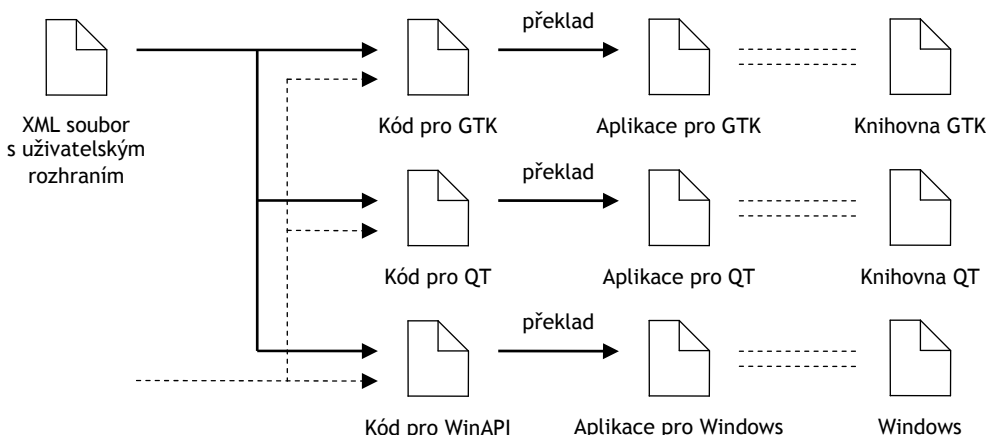
Pro zápis statického rozhraní použijeme výše vybraný formát XML. Každý XML soubor se dá chápat jako strom - struktura s jedním kořenem která se pak dále větví. První problém na který narazíme, je správné určení toho co budeme považovat za kořen - má to být projekt, aplikace nebo okno? Brát projekt (skupinu aplikací) za kořen našeho dokumentu je zbytečné, jelikož není účelem tohoto projektu vytvářet systém na správu projektů. Okno, jako volba z druhého konce je zas až příliš specifická a byla by nutnost definovat spojovací prvek protože aplikace můžou a i mývají více oken (minimálně je slušné implementovat okénko „O aplikaci“ informující o autorovi a verzi).

Správná volba teda je mít za kořen XML dokumentu naši aplikaci. Aplikace pak bude obsahovat okna, a okna budou obsahovat komponenty, třeba takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<application name="Kalkulačka">
  <window name="Hlavní okno">
    <edit id="display" readonly="1" label="0." />
    <button id="btn0" label="0" />
    <button id="btn1" label="1" />
    :
    :
  </window>
  <window name="O aplikaci">
    :
    :
  </window>
</application>
```

Přesný zápis, názvy komponent a jejich atributy budou uvedeny v dalším textu.

Použití uživatelského rozhraní napsaného v XML je pak následující. Jelikož každá grafická knihovna (WinAPI, GTK, QT) poskytuje jenom služby na tvorbu komponent (widgets) za běhu (nic takového jako načítání rozhraní ze souboru neexistuje) tak je nutno rozhraní přeložit na kód, který se provede při inicializaci aplikace a vytvoří příslušné okno. Graficky lze situaci naznačit následovně:

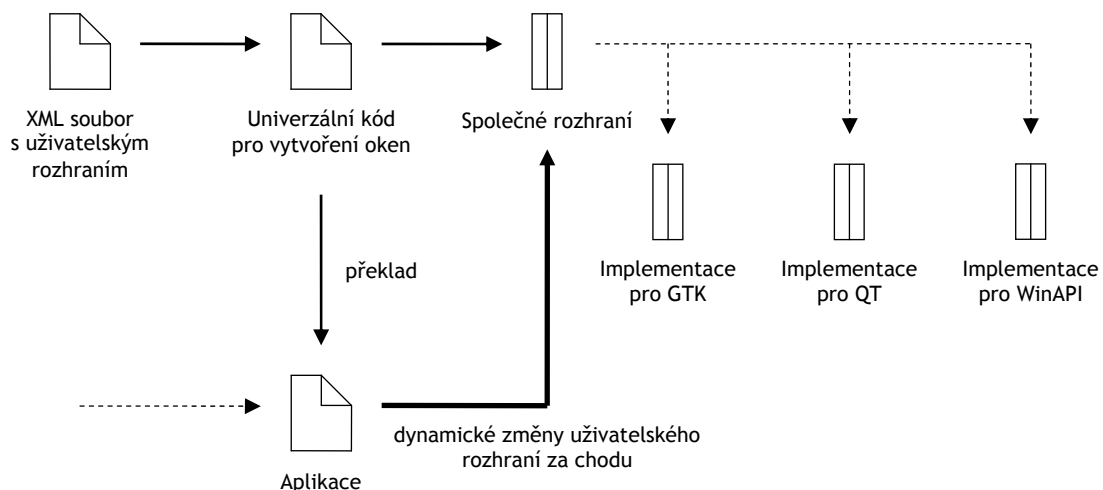


V obrázku není zakreslena cesta kódu aplikace, který je společný pro všechny varianty. Tato cesta bude předmětem diplomové práce, zjednodušeně lze říct, že ze sémantického stromu programu v XML se bude generovat zdrojový kód v požadovaném jazyce, překlad a spuštění je pak možno s použitím rozdílových knihoven.

## Dynamické rozhraní

Z praxe je znát, že ne každá aplikace se spokojí se statickým rozhraním a je nutné umožnit tvorbu uživatelského rozhraní taky v běžících programech. Příkladem za všechny může být nástroj pro návrh uživatelského rozhraní (editor oken) který vytváří komponenty za běhu, jak si uživatel přeje.

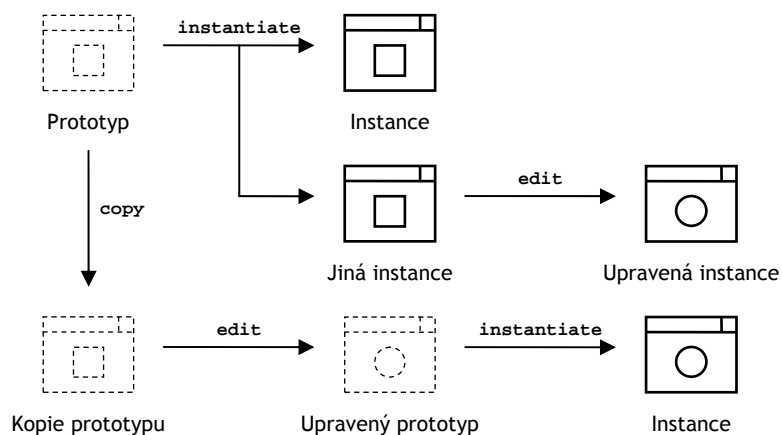
Sjednotit tuto část už není tak jednoduché jako v předchozím případě, protože tady se již jedná o konkrétní programy, využívající různé grafické knihovny. Problém to je ale určitě řešitelný, protože lze definovat (novou) sadu rozhraní, které bude jednotné, nezávislé na knihovnách. Toto rozhraní má ale výhody i pro předcházející situaci, kde se zjednoduší překlad statického rozhraní, které se překládá na univerzální kód pracující s rozhraním (toto slovo rozhraní je zde z objektového programování, ne z grafiky), takže je potřeba jenom jediné transformace z XML na textový formát - zdrojový kód nebo skript.



## Metoda tvorby

Uživatelské rozhraní které je předmětem tohoto projektu se vytváří dvoufázově. Značí to zejména to, že se vždy vytvoří prvně prototyp, který se následně instanciuje. Rozhraní pro editaci prototypu i instance je to samé, takže je možná i dynamická změna rozhraní a to jak v prototypu tak v instanci. Tímto teda jde o nástroj v podstatě silnější než dosavadní programovací nástroje jako Delphi nebo Visual C++, kde je možno měnit jenom instanci protože prototyp je vždy zkompileován do programu napevno.

Následující schéma ukazuje různé cesty pro dosažení stejného efektu:

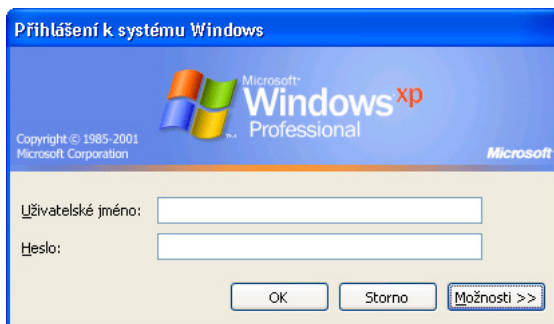


## Praktická ukázka

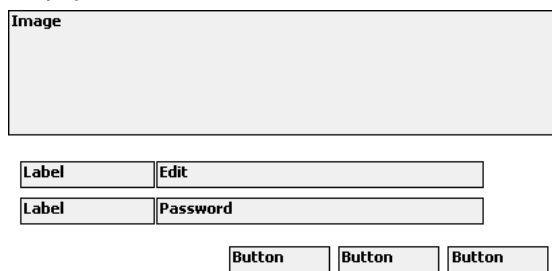
### Postup

Jelikož v budoucnu bude univerzální rozhraní generováno z modelu automaticky - strojově, je zbytečné ho implementovat teď pouze pro tuto ukázkou. Namísto univerzálního rozhraní jsem teda implementoval jen jednu specifickou transformaci, konkrétněji XML na HTML a na CSS pro dosažení požadovaného efektu (vzhledu). Ukázka je založena na přihlašovací okně, které je ve většině systémů první co uživatel vidí.

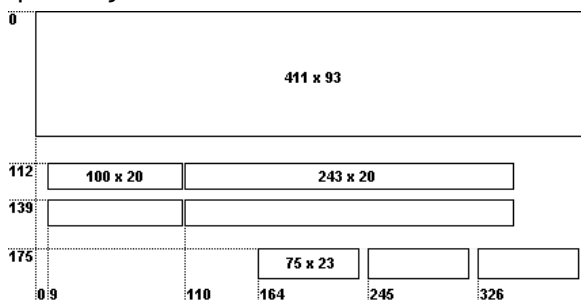
Původní obrázek včetně dekorace sejmutý z reálného systému (vzdálená plocha):



Po analýze komponent dospějeme k závěru, že okno se skládá z následujících objektů:



Poloha a velikost komponent je:



Pokud toto všechno víme, můžeme zapsat okno ve formátu XML ručně, a pak pomocí transformace převést zápis v XML do HTML a CSS které už umí zobrazit jakýkoliv prohlížeč (testováno na prohlížečích MS Internet Explorer 6 a Mozilla Firefox 1.0):



## Zápis v XML

```
<?xml version="1.0" encoding="Windows-1250"?>
<application name="login">
  <window id="main">
    <size><x>411</x><y>209</y><style>INNER</style></size>
    <caption>Pøihlášení k systému Windows</caption>
    <style>
      <background>
        <color>#EFEBDE</color>
      </background>
      <font>
        <family>Tahoma</family>
        <size unit="pt">8</size>
        <color>#000</color>
      </font>
    </style>
    <focus>btnOk</focus>
    <contents>
      <image>
        <source>banner.png</source>
        <position><x>0</x><y>0</y></position>
        <size><x>411</x><y>93</y><style>CROP</style></size>
      </image>
      <label>
        <caption>Uživatelské jméno:</caption>
        <position><x>9</x><y>112</y></position>
        <size><x>110</x><y>20</y></size>
        <alignment><x>LEFT</x><y>MIDDLE</y></alignment>
      </label>
      <edit id="username">
        <position><x>110</x><y>112</y></position>
        <size><x>243</x><y>20</y></size>
        <style>
          <border><color>#7B9EBD</color><style>solid</style><width>1</width></border>
          <background><color>#FFF</color></background>
        </style>
      </edit>
      <label>
        <caption>Heslo:</caption>
        <position><x>9</x><y>139</y></position>
        <size><x>110</x><y>20</y></size>
        <alignment><x>LEFT</x><y>MIDDLE</y></alignment>
      </label>
      <password id="password">
        <position><x>110</x><y>139</y></position>
        <size><x>243</x><y>20</y></size>
        <style>
          <border><color>#7B9EBD</color><style>solid</style><width>1</width></border>
          <background><color>#FFF</color></background>
        </style>
      </password>
      <button id="btnOk">
        <caption>OK</caption>
        <position><x>164</x><y>175</y></position>
        <size><x>75</x><y>23</y></size>
      </button>
      <button id="btnCancel">
        <caption>Storno</caption>
        <position><x>245</x><y>175</y></position>
        <size><x>75</x><y>23</y></size>
      </button>
      <button id="btnMore">
        <caption>Možnosti &gt;&gt;</caption>
        <position><x>326</x><y>175</y></position>
        <size><x>75</x><y>23</y></size>
      </button>
    </contents>
  </window>
</application>
```

## Konverze do HTML a CSS

Pro lepší čitelnost je výpis naformátován ručně. CSS pro WM (window manager, správce oken) je zadán staticky protože se předpokládá že je platný v celém systému (dané CSS reprezentuje hodně zjednodušený standardní skin pro Windows XP).

```
<html>
<head>
  <title>SCS projekt</title>
</head>
<style>

  .wm-window-decoration {
    border : 3px solid #0A52E4;
    position: absolute;
    left: 32px;
    top: 32px;
  }

  .wm-window-caption {
    background-image: url('xp/caption-bg.png');
    background-repeat: repeat-x;
    line-height: 26px;
    font-family: "trebuchet ms", sans-serif;
    font-size: 10pt;
    color: #FFF;
    font-weight: bold;
    padding-left: 4px;
  }

  .wm-window-contents {
  }

</style>
<body>
<div class="wm-window-decoration">
  <div class="wm-window-caption">Přihlášení k systému Windows</div>
  <div class="wm-window-contents"
    style="font-family:Tahoma;font-size:8pt;color:#000;
    width:411px;height:209px;background-color:#EFEBDE">
    <!-- image -->
    
    <!-- label -->
    <div style="width:110px;height:20px;position:absolute;left:9px;top:138px;
      line-height:20px">Uživatelské jméno:</div>
    <!-- edit -->
    <input type="edit" name="username" value=""
      style="width:243px;height:20px;position:absolute;left:110px;top:138px" />
    <!-- label -->
    <div style="width:110px;height:20px;position:absolute;left:9px;top:165px;
      line-height:20px">Heslo:</div>
    <!-- password -->
    <input type="password" name="password"
      style="width:243px;height:20px;position:absolute;left:110px;top:165px" />
    <!-- button -->
    <input type="button" name="btnOk" value="OK"
      style="width:75px;height:23px;position:absolute;left:164px;top:201px;
      font-family:Tahoma;font-size:8pt;color:#000" />
    <!-- button -->
    <input type="button" name="btnCancel" value="Storno"
      style="width:75px;height:23px;position:absolute;left:245px;top:201px;
      font-family:Tahoma;font-size:8pt;color:#000" />
    <!-- button -->
    <input type="button" name="btnMore" value="Možnosti &gt;&gt;"
      style="width:75px;height:23px;position:absolute;left:326px;top:201px;
      font-family:Tahoma;font-size:8pt;color:#000" />
  </div>
</div>
</body>
</html>
```

## Detailnější popis formátu

Jak již bylo uvedeno výše, formát dat je XML a tato kapitola se věnuje popisu obsahu souboru s uživatelským rozhraním.

### Aplikace

Každá aplikace je identifikována jménem, které je pak možno zobrazit při otevření souboru s popisem uživatelského rozhraní. Aplikace se pak skládá z jednotlivých oken, které mají stejné vlastnosti, tj. žádné okno není upřednostňováno před jiným, jako je tomu např. v Delphi (kde první okno je hlavní okno aplikace).

```
<?xml version="1.0" encoding="Windows-1250"?>
<application name="calc">
  <window id="main">
    :
    :
  </window>
  <window id="about">
    :
    :
  </window>
</application>
```

### Okno

Stejně jako aplikace, i každé okno jsou identifikovány identifikátorem, ale tady je to již nutnost na rozdíl od aplikace, protože uvedený identifikátor slouží k odkazování se na okno, přesněji prototyp okna (soubor s popisem uživatelského rozhraní obsahuje prototypy, ne instance). Okno má několik speciálních atributů a dále může obsahovat komponenty.

```
<window id="main">
  <size>
    <x>411</x>
    <y>209</y>
    <style>INNER</style>
  </size>
  <caption>Přihlášení k systému Windows</caption>
  <style>
    <background>
      <color>#EFEFDE</color>
    </background>
    <font>
      <family>Tahoma</family>
      <size unit="pt">8</size>
      <color>#000</color>
    </font>
  </style>
  <focus>btnOk</focus>
  <contents>
    :
    :
  </contents>
</window>
```

Velikost okna se určuje v pixelech a současně se specifikuje způsob jak se má chápat daný rozměr, tj. jestli jsou hodnoty velikostí klientské části (vnitřek okna, INNER) nebo na celého okno i s dekorací (vnějšek okna, OUTER). Titulek okna je zadán v elementu i když by mohl být v atributu, ale zápis přes element dovoluje dělat vícejazyčné rozhraní následovným způsobem:

```
<caption xml:lang="cz">Přihlášení k systému Windows</caption>
<caption xml:lang="en">Log-in to the Windows system</caption>
```

Další element reprezentuje upřesnění grafického vzhledu pomocí pravidel které mají hierarchii velice podobnou hierarchii atributů v CSS, takže jsou ruční úpravy o něco jednodušší pro ty co znají CSS. Předposlední element specifikuje identifikátor komponenty která dostane výchozí zaostření (kurzor) a v poslední části je seznam komponent v daném okně.

Atributy které ještě nebyly zahrnuty do specifikace obsahují data o výchozí modálnosti okna, informace jestli a jak lze zvětšovat okno a dále taky určení ikony a systémových tlačítek okna. Tyto vlastnosti jsou pro tento projekt nepodstatné protože se jedná jen o jednoduchou ukázkou principu.

## Komponenty

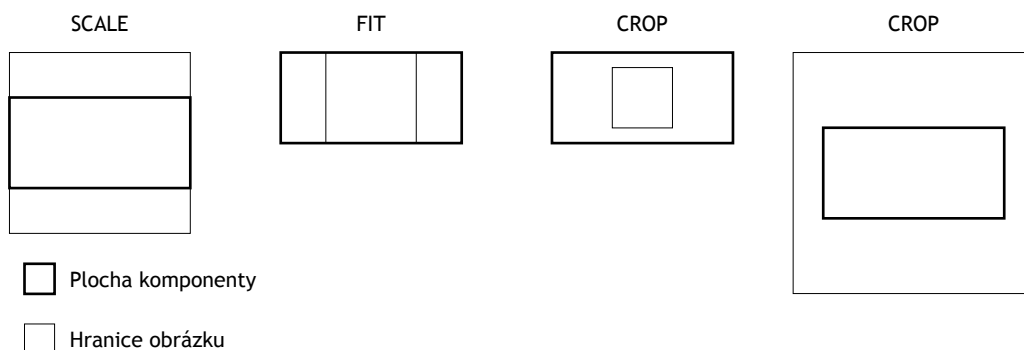
Uvedený příklad s přihlašovacím oknem obsahoval pět různých komponent: obrázek, tlačítko, statický text, pole pro editaci textu a pole pro zadávání hesla. Většina komponent má ale stejné základní atributy:

```
<button id="btnOk">
  <caption>OK</caption>
  <position>
    <x>164</x>
    <y>175</y>
  </position>
  <size>
    <x>75</x>
    <y>23</y>
  </size>
</button>
```

Tyto atributy reprezentují jednoznačný identifikátor (id), pozici a velikost komponenty. Vlastnost caption, tj. titulek lze nalézt jak u tlačítka, tak u statického textu. Editační pole má vlastnost text, které obsahuje editovaný text. Obrázek má pak vlastnost kde je uveden zdroj obrázku (jeho url) a taky je tam upřesněno jak se má chápat velikost:

```
<image>
  <source>banner.png</source>
  <position>
    <x>0</x>
    <y>0</y>
  </position>
  <size>
    <x>411</x>
    <y>93</y>
  <style>CROP</style>
</size>
</image>
```

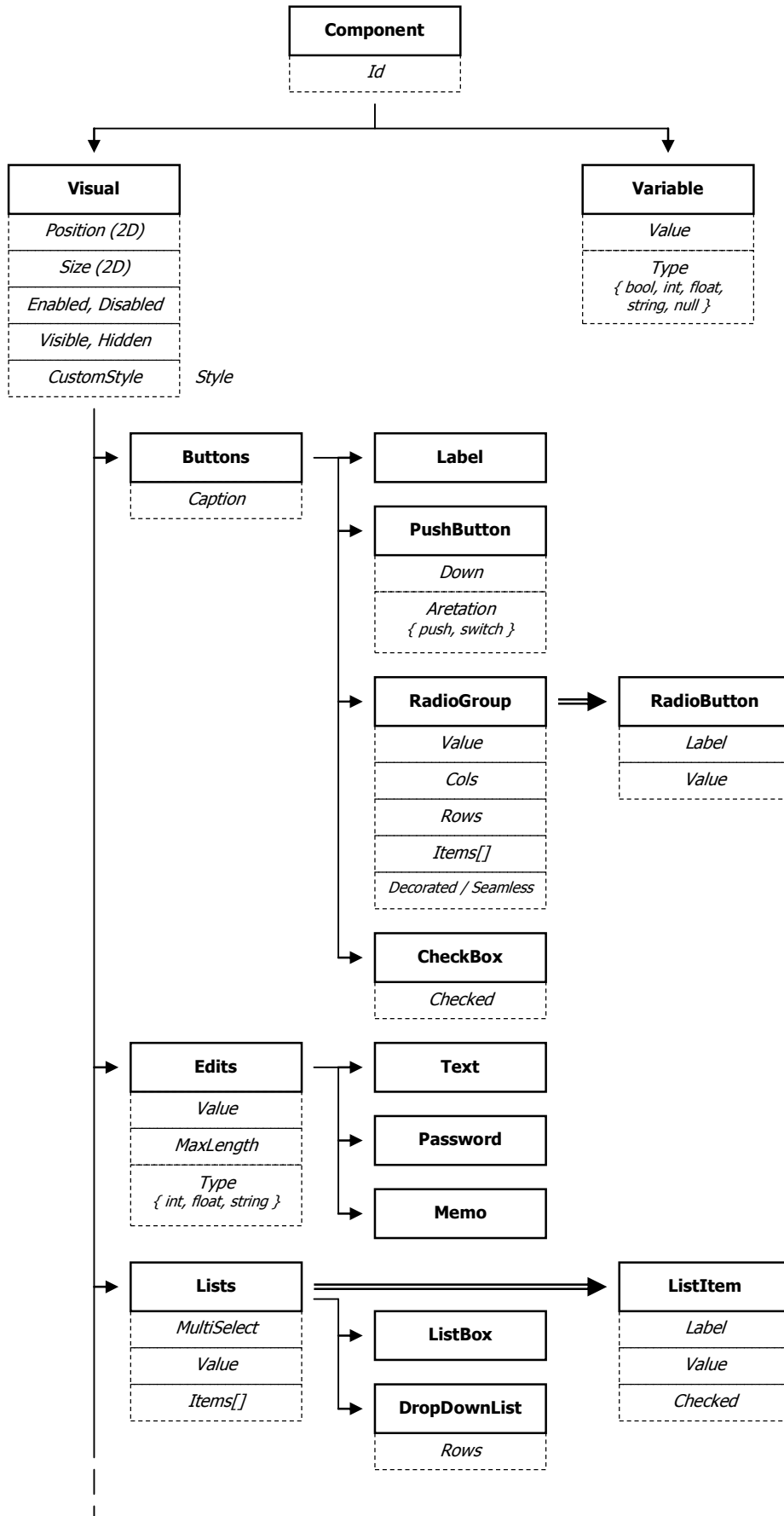
Možné varianty jsou SCALE, FIT, CROP. První dvě zmenšují nebo zvětšují obrázek a třetí nechává velikost nezměněnou. V specifikaci není definováno horizontální a vertikální zarovnání u obrázku a taky způsob změny velikosti co se týče kvality (algoritmu).

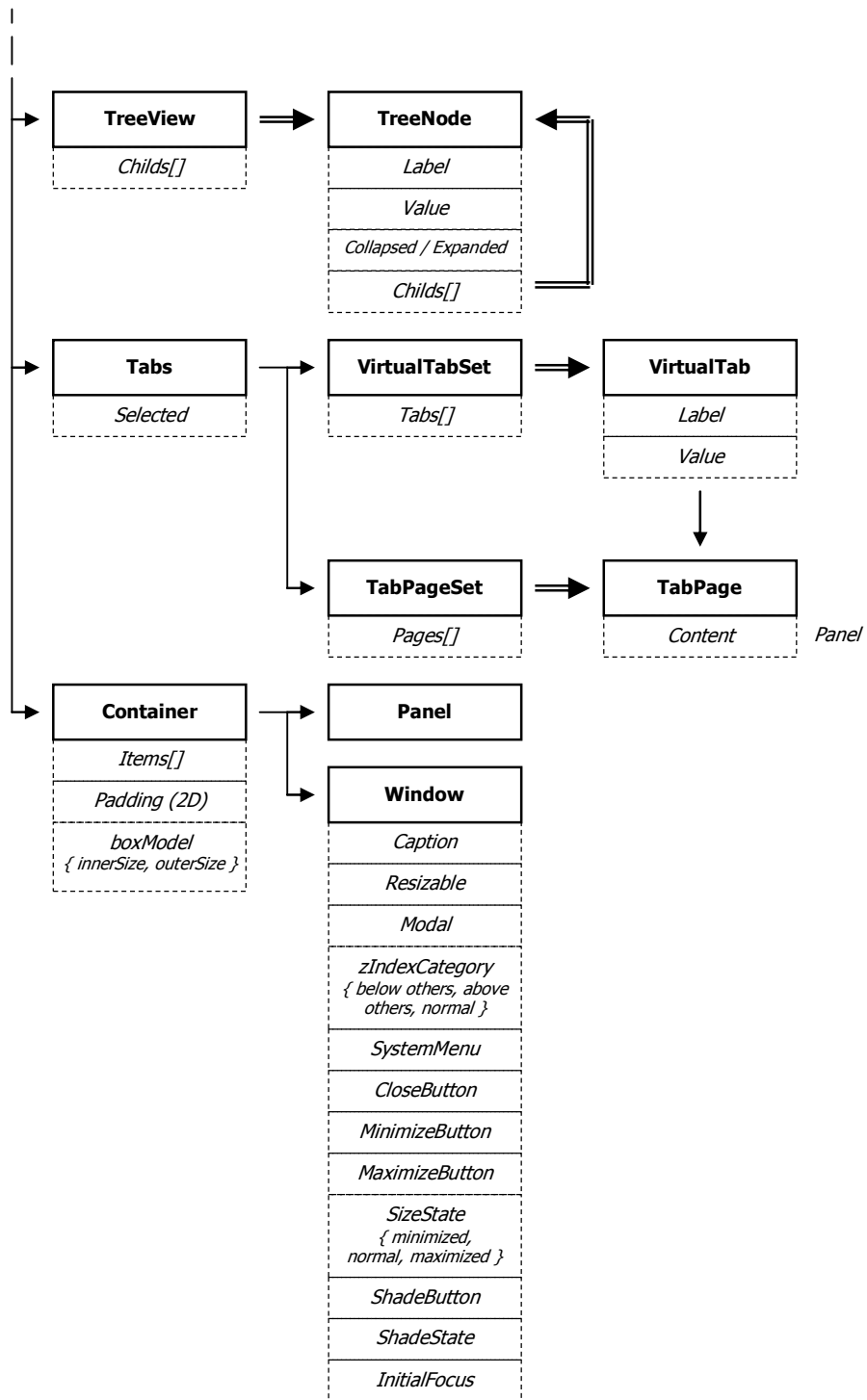


Zarovnání bylo v příkladu podstatné jenom pro textové popisky - obdobně lze zavést chybějící zarovnání u obrázků. U textu pro změnu chybí vlastnost určující zalamování nebo ořezání.

```
<label>
  <caption>Heslo:</caption>
  <position>
    <x>9</x>
    <y>139</y>
  </position>
  <size>
    <x>110</x>
    <y>20</y>
  </size>
  <alignment>
    <x>LEFT</x>
    <y>MIDDLE</y>
  </alignment>
</label>
```

# Strom komponent





## Závěr

Tento projekt je maličká součást omnoho většího projektu, který plánuji realizovat v rámci diplomové práce (vývojové prostředí nezávislé na programovacím jazyku, grafický zápis algoritmů), takže jakákoliv možnost realizovat alespoň kousek přišla vhod. Na chybějící vlastnosti jsem se snažil alespoň upozornit aby bylo pokračování v projektu jednodušší, protože k praktickému využití máme ještě daleko. Zbývá zejména:

- dokončení stromu komponent, upřesnění sémantiky, atributů
- definování událostí a způsobu vazby mezi uživatelským rozhraním a kódem
- návrh programových rozhraní pro jednotlivé činnosti
- návrh objektového modelu komponent
- implementace modelu pro různé platformy

Některé kroky je sice možno provést ihned, ale některé závisí také na ostatních částech projektu, zejména implementační fáze bude možná nejdříve až po definování formátu na zápis algoritmů v XML, nebo až když bude existovat ten grafický editor algoritmů.

## Použitá literatura

- [1] **Extensible Markup Language (XML)**  
<http://www.w3.org/XML/>
  
- [2] **Čaderský, P., Rozsnyó, D.: Web Application Framework**  
Final Project Thesis at Faculty of Nature and Technology, Hogeschool van Utrecht  
<http://www.rozsnyo.com/diary/2004/06/18/WebApplicationFramework.pdf>