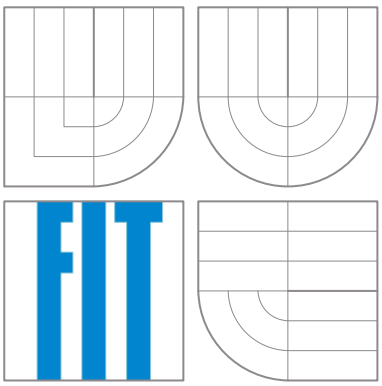


Fakulta informačních technologií
Vysoké učení technické v Brně

Dokumentace projektu pro předmět ITU

Mind Mapper

Autoři:

Vendula Poncová	`xponco00`
Marek Salát	`xsalat00`
Tomáš Trkal	`xtrkal00`

17. prosince 2012

Obsah

1 Úvod

V této zprávě dokumentujeme návrh, realizaci, testování a vyhodnocení uživatelského rozhraní programu pro tvorbu a úpravu myšlenkových map. Aplikaci jsme implementovali v jazyce Python s využitím knihovny wxPython.

2 Analýza problému

V této kapitole se zamýšlíme nad účelem editoru myšlenkových map, jeho využitím a potenciálními uživateli.

Jaké jsou požadavky? Účelem projektu je vytvořit aplikaci pro tvorbu a úpravu myšlenkových map. Máme klást důraz na jejich zobrazování a uživatelsky příjemné a intuitivní ovládání. Mezi základní požadavky na aplikaci patří: tvorba mapy, editace mapy, skrývání a zobrazování různých podčástí, ukládání do souboru, nahrávání ze souboru a export do obrázku.

K čemu je to dobré? Myšlenkové mapy jsou vhodné pro jakoukoliv formu plánování. Jedná se o efektivnější a lépe zapamatovatelnou alternativu k seznamům. Lze je použít při řešení problému a dekompozici na podproblémy, pro utřízení myšlenek, brainstorming nebo seznam úkolů. Možnosti využití jsou široké a záleží na uživateli.

Kdo to bude používat? Editor myšlenkových map může sloužit téměř každému, kdo upřednosťune pracovat s myšlenkovými mapami. Předpokládáme, že nejhojněji tento typ aplikací používají studenti, manažeři a pracovní kolektiv.

2.1 Studium existujících řešení

Vyzkoušeli jsme si práci s aplikacemi: Mindjet Mindmaper Pro, FreeMind, ConceptDraw MindMap, MindGenius a MindVisualizer. Další aplikace, které jsme prostudovali byly například: MindMeister, BrainMine, DropMind a MindBerry. Snažili jsme se pokrýt všechny aplikace od freewarových po komerční pro různé systémy a operační prostředí.

Všimli jsme si, že většina aplikací nabízí nepřeberné množství nástrojů pro vytváření map a už tolik neřeší jejich snadné zobrazování a procházení. Domníváme se, že editor myšlenkových map by neměl degenerovat na klasický editor pro tvorbu grafů, neboť tím zastiňuje výhody myšlenkových map. Uživatel se pak více soustředí na práci s programem než na tvorbu mapy a tím odklání svoji pozornost od řešeného problému.

Na základě těchto závěrů jsme se rozhodli odklonit od tohoto trendu a nabídnout uživatelům jednoduchý a efektivní nástroj.

2.2 Formulace cílů

Na naši aplikaci jsme si stanovili tyto požadavky:

Jednoduchost
Nevytvářet všeobecný nástroj pro tvorbu grafů.
Minimalizace nabídky nástrojů pro vytváření a editování map.

Elegance
Možnost zobrazování podmap, skrývání větví a poznámek.
Implicitně pěkný vzhled map.

Intuitivnost
Snadné a efektivní vytváření a procházení myšlenkových map.
Podpora víceúčelovosti a komplexnosti map.

3 Návrh řešení

Navrhovanou aplikaci jsme pracovně označili Neuron. Náš návrh řešení a implemetace je následující:

3.1 Návrh aplikace

V návrhu aplikace jsme se zaměřili na cíle, které jsme si stanovili. Řešili jsme tedy zejména způsob práce s myšlenkovou mapou, její vzhled a minimalizaci nastavení.

Rozhodli jsme se, že aplikace nebude podporovat otevírání více souborů v rámci jedné instance a práci s více mapami v rámci jednoho souboru. Pokusili jsme se navrhnout myšlenkové mapy tak, aby se tím uživatel necítil limitován a abychom podpořili ideu myšlenkových map. Jsme přesvědčení, že jeden uživatel je schopný všechny oblasti svého života zahrnout do jedné myšlenkové mapy. Proto má každá mapa vždy alespoň jeden uzel (root).

Myšlenkové mapy by měly umožňovat snadné skrývání a zobrazování větví. Stejně tak zobrazení větve jako myšlenkové mapy (podmapy). Tento problém jsme se rozhodli vyřešit zobrazováním tzv. pohledů. Pohled je definovaný svým kořenovým uzlem a užvatel si může v rámci každého pohledu nastavit, co má být skryto a odkryto a umístění jednotlivých uzlů. Toto nastavení zůstává zachováno pro všechny pohledy. Rozhodli jsme se podporovat změnu možnost rozmístění jen u uzlů první úrovně (následníky uzlu root) a ostatní úrovně zobrazovat v seznamech, abychom aplikaci zjednodušili.

V nastavení uzlů jsme ponechali: popisek, barvu, prioritu, ikonku a poznámku. Uzel je dále možné přesunout, změnit mu rodiče, skrýt nebo zobrazit jeho následníky, skrýt nebo zobrazit jeho poznámku, vytvořit a smazat.

S ohledem na zobrazování podmap jsme se rozhodli zobrazovat v bočním panelu výčet uzlů celého stromu a zvýrazňovat uzel, který je aktuálně zobrazený jako kořen.

Pro nástroje a možnosti nastavení jsme se změřili na tři oblasti: panel s nastavením uzlu, toolbar na horní liště a toolbar uzlu. Panel s nastavením uzlu zobrazuje nastavení vybraného uzlu a umožňuje jej modifikovat, toolbar na horní liště zpřístupňuje nejčastěji prováděné operace nad mapou a toolbar uzlu zpřístupňuje operace uzlu. Chceme navíc podpořit co nejefektivnější procházení a zobrazování pomocí tlačítek myši.

3.2 Návrh implementace

Při návrhu implementace jsme dodržovali návrhový vzor MVC, který je typický pro aplikace s uživatelským rozhraním. Pro implementaci jsme si vybrali objektově orientovaný jazyk Python, proto jsme se zaměřili na návrh jednotlivých tříd a modulů. Pro tvorbu grafického uživatelského rozhraní jsme používali komponenty z knihovny wxPython.

4 Popis implementace

V této kapitole popisujeme implementaci programu Neuron. Rozhodli jsme se program vytvořit v jazyce Python verze 2.6 nad knihovnou wxPython. Volba tohoto jazyka nám umožnila rychlý vývoj prototypu a snadné úpravy do finální podoby. Knihovnu wxPython jsme zvolili pro její multiplatformnost a nativní vzhled dle systému uživatele. Vývoj probíhal na operačním systému Ubuntu ve vývojovém prostředí Eclipse s modulem pro Python PyDev.

Aplikace je spustitelná souborem neuron.py ve složce src. Skript importuje balík neuron a spustí jeho funkci run, čímž se spustí a inicializuje uživatelské rozhraní. Balík neuron se skládá ze tří balíků: gui, lib a mindmap.

4.1 Modul gui

Grafické uživatelské rozhraní a jeho komponenty jsou popsané v tomto balíku. Základní kostra aplikace je definovaná v souboru template.py a její třída MainFrame je rodičem třídy MapEditor. MapEditor inicializuje grafické uživatelské rozhraní celé aplikace a vytváří instanci třídy MapControler z balíku mindmap, která je komponentou vykreslující a spravující myšlenkovou mapu. Součástí balíku je dále třída MapTree. Jedná se

o komponentu, která obsahuje stromový výčet všech uzlů myšlenkové mapy a zvýrazňuje uzel, který je aktuálne zobrazen jako kořenový. Poslední důležitou komponentou je třída NodeMenu, která obsahuje nabídku nástrojů pro uzel. Nabídka se vykresluje při najetí myší na některý z uzlů. Je vykreslována komponentou MapControler.

4.2 Modul mindmap

Model, komponenta uživatelského rozhraní a kontroler myšlenkových map je součástí balíku mindmap. V souboru model.py jsou třídy MindMap a MindNode sloužící jako model struktury myšlenkových map. MindMap má právě jeden kořenový uzel typu MindNode. Uzly si pamatují svého předka a přímé následníky. Každý uzel je jednoznačně identifikován svým identifikačním číslem. Součátí modelu nejsu žádné informace o způsobu zobrazení uzlů.

V souboru positioner.py jsou třídy pro pozicování prvků myšlenkové mapy. Zde jsou uložena nastavení související s vykreslením pohledu na mapu. Třída NodeView udržuje informace o pohledu na uzel. Třída MapView spravuje informace o všech pohledech na mapu, které byly vygenerované. Pokud je požadováno vykreslení pohledu, který neexistuje, vygeneruje se rozmístění prvků v tomto pohledu a uložé do struktury. Propojení mezi modelem myšlenkové mapy a pohledy se děje pomocí identifikačních čísel uzlů. Pohledy jsou spravovány ve slovnících.

Soubor drawer.py obsahuje jedinou třídu MapDrawer, která vykresluje uzly na základě aktuálního pohledu a informacích v modelu. Prochází od aktuálního kořene všechny uzly, které mají být zobrazené a dle jejich úrovně je vykresluje.

Konečně srdcem celého balíku je soubor controler.py a třída MapControler, která zajišťuje komunikaci mezi uživatelem a všemi vyjmenovanými třídami. Třída je potomkem třídy PyControl z balíku wx a slouží jako plátno pro vykreslení myšlenkové mapy a jako senzor uživatelských akci nad těmito mapami. Řídí tedy svým způsobem celou aplikaci.

4.3 Modul lib

Různé pomocné funkce a struktury jsou součástí balíku lib. Za zmínku stojí třída MacColor, kde je uložena implicitní paleta barev uzlů. Při vytváření uzlů se pak náhodně vybírá barva právě z této palety. Cílem této funkce bylo implicitní vytváření pěkných map bez nutnosti zásahu uživatelem.

5 Popis aplikace

Tato kapitola popisuje způsob spuštění aplikace a obsahuje několik ukázek z běhu programu. Aplikace má implementované jen základní rozhraní, které jsme chtěli otestovat. Položky menu a toolbaru jsou nefunkční. Aplikace podporuje pouze tvorbu, úpravu a zobrazování map.

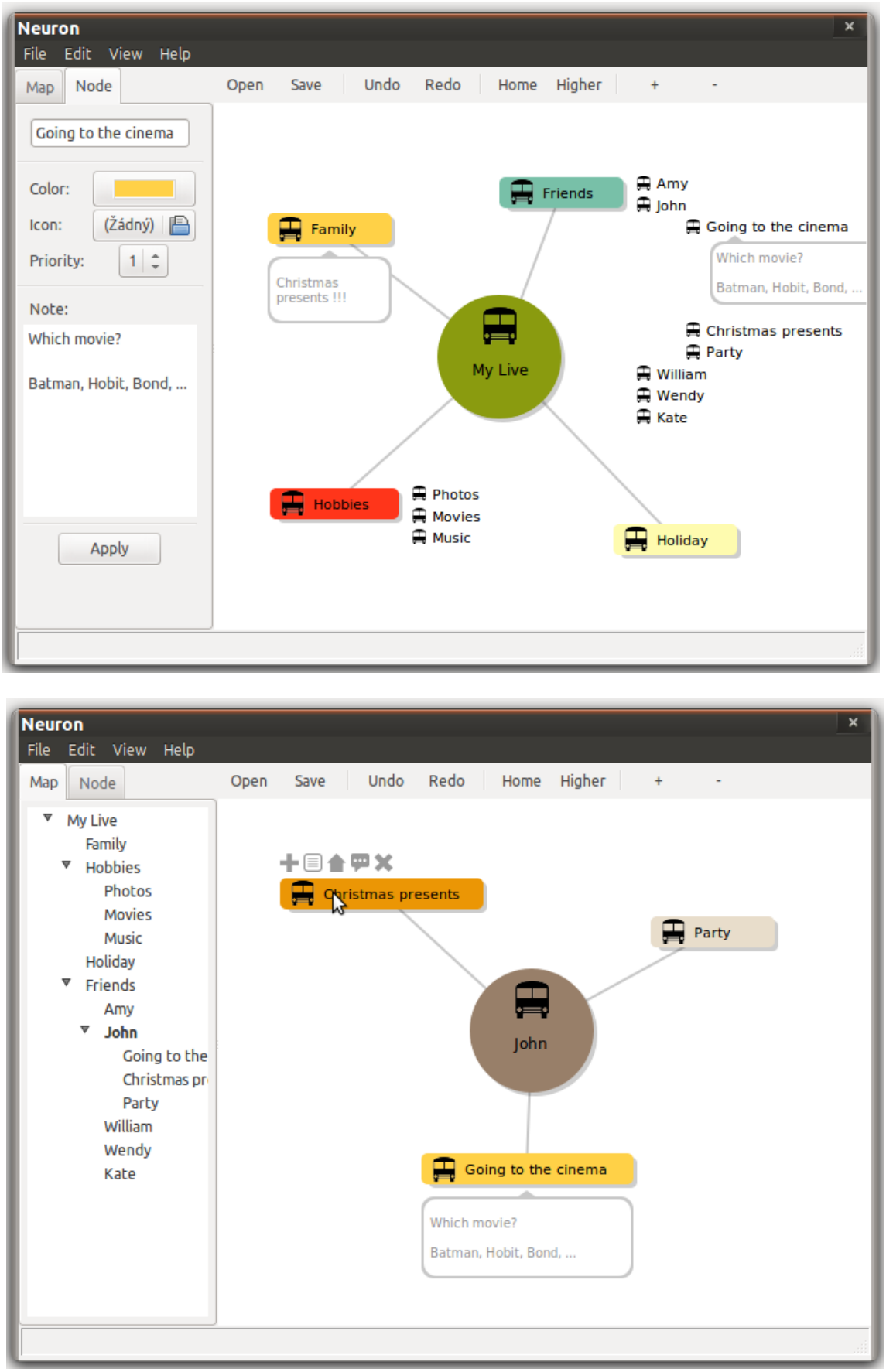
5.1 Zprovoznění a spuštění aplikace

Před spuštěním programu je třeba nainstalovat knihovnu wxPython 2.8, viz. *http://wiki.wxpython.org/How to install wxPython*. Například na linuxové distribuci typu Debian stačí spustit příkaz `apt-get install python-wxgtk2.8`.

Aplikace Neuron se spouští spuštěním souboru neuron.py ve složce src, případně spuštěním přes terminál příkazem `python neuron.py` ze složky src. Aplikace nebyla testována na jiném operačním systému než Ubuntu.

5.2 Ukázka aplikace

Screenshoty aplikace jsou na obrázku 1.



Obrázek 1: Ukázka programu Neuron.

6 Testování uživatelského rozhraní

V této kapitole se věnujeme testování uživatelského rozhraní naší aplikace.

6.1 Návrh experimentů

Rozhodli jsme se pro experimenty metodou přímé pozorování uživatele. Uživatelé měli za úkol plnit předložené úkoly a nahlas komentovat, co dělají a co si myslí. Cílem bylo srovnat naše uživatelské rozhraní s rozhraním podobné aplikace a odhalit její nedostatky. Ze studovaných aplikací jsme pro srovnání vybrali tu, která se nám zdála ve všech ohledech nejlepší, Mind Map.

Uživatele jsme řadili do skupin začátečník, pokročilý. Cílovou skupinou pak byli studenti, pracovníci na IT pozicích a manažeři. Uživatelům byla předložena sada úloh v podobě myšlenkových map, které měli za úkol postupně vytvářet a upravovat. Pro srovnání jsme nechali uživatele pracovat s testovanou aplikací a konkurenční aplikací Mind Map. Měřili jsme pak časy jednotlivých úloh a počet chyb, kterých se uživatelé v dané úloze dopustili.

V úlohách jsme se zaměřili na operace: vytvoření uzlu, smazání uzlu, vytvoření podúrovně, rozkliknutí podúrovně, skrytí podúrovně, změna rodiče, změna priority, vytvoření poznámky, skrytí poznámky, změna pohledu a návrat pohledu. Úlohy byly navržené tak, aby se dala odvodit intuitivnost rozhraní a jeho zapamatovatelnost.

6.2 Průběh testování

V rámci jednoho týdne jsme provedli testy s deseti uživateli různé pokročilosti (6 začátečníků, 4 pokročilí) s věkem v rozsahu 14 až 43 let zaměstnaných v různých oborech. Uživatelé plnili celkem 11 úloh. Průměrná délka testování byla 25 minut pro jednu aplikaci.

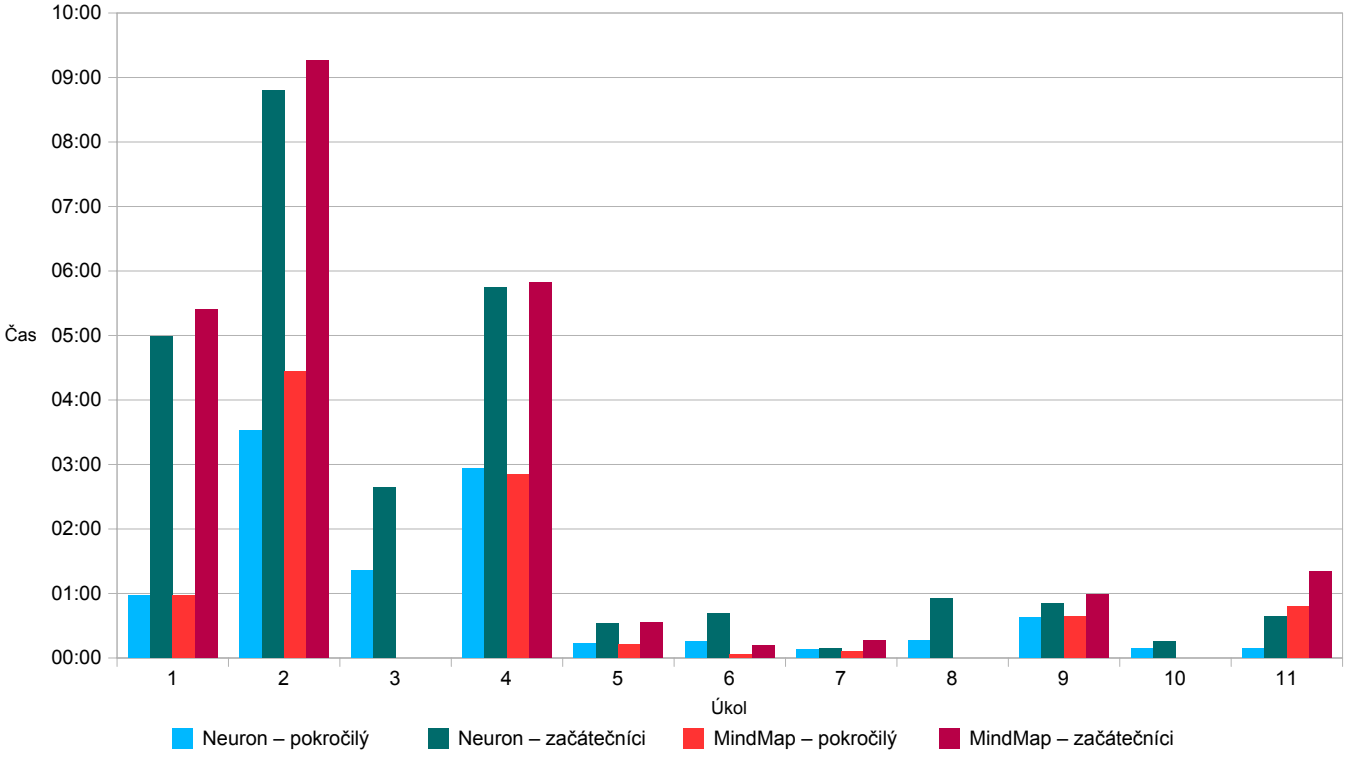
Uživatelé si nejčastěji stěžovali na:

6.3 Vyhodnocení výsledků testování

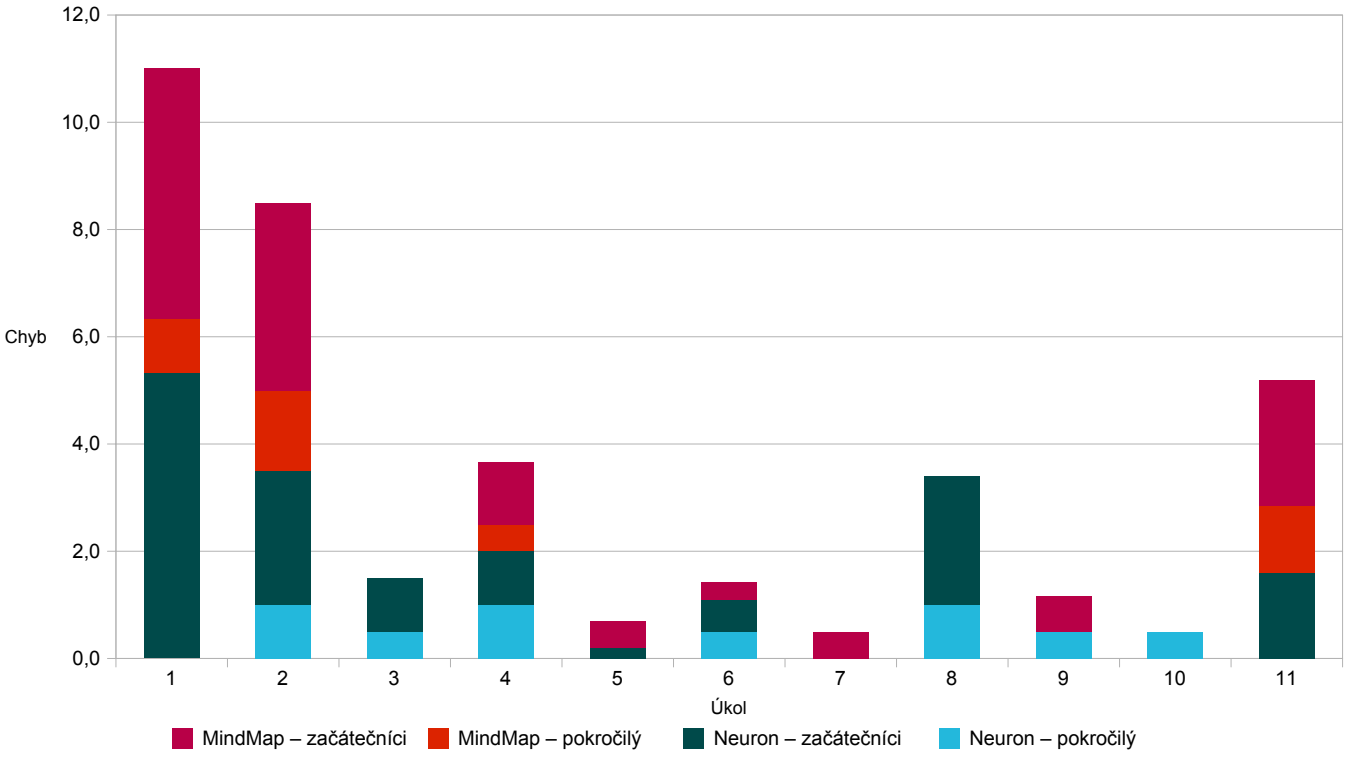
Výsledky testování jsou zpracovány v grafech 2 a 3, kde jsou srovnány průměrné časy a počet chyb začátečníků a pokročilých.

Z grafů je možné vyčíst strmou křivku učení. Lze z toho vyvodit, že je prostředí snadno osvojitelné. Pro začátečníky není prostředí úplně intuitivní, ale to je omluvitelné nedostatečným popisem dostupných nástrojů a neexistující nápovědou. Výsledky testů naší aplikace jsou přesto srovnatelné s výsledky konkurenčního programu, dokonce ho ve většině případů předstihují.

Výrazný rozdíl časů mezi začátečníky a pokročilými uživateli by byl brán v potaz v dalším vývoji. Zejména by se jednalo o přidání nápovědy a popisků nástrojů a začlenění editace uzlu do toolbaru uzlu, neboť se jedná o problémy, které uživatelé nejčastěji zmiňovali.



Obrázek 2: Srovnání průměrných časů začátečníků a pokročilých.



Obrázek 3: Srovnání průměrného počtu chyb začátečníku a pokročilých.

7 Závěr

Navrhli jsme, implementovali a otestovali aplikaci pro tvorbu myšlenkových map. Zaměřili jsme se na podporu snadného vytváření a zobrazování myšlenkových map v kontrastu s běžnými aplikacemi podobného zaměření. V testech se projevil nedostatečný nebo žádný popis nástrojů, které jsou v aplikaci k dispozici, ale po zaučení uživatelů se ukázalo naše řešení v porovnání s konkurenčními programy jako efektivnější. Navíc jsme implementovali vlastnosti, které nám přijdou pro myšlenkové mapy zásadní, jako přidávání poznámek, skrývání větví a zobrazování podmap, které většina nastudovaných aplikací nepodporovala.