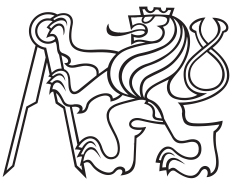
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Tvorba GUI pro nástroj FreeSurfer

Karolína Burešová
Program: Otevřená informatika
Obor: Informatika a počítačové vědy

Jaro 2014
Vedoucí práce: prof. RNDr. Olga Štěpánková, CSc.

Poděkování / Prohlášení

Chtěla bych poděkovat svému milému, že mě má rád, stojí při mě a často mi pomáhá najít chuť dělat něco užitečného.

Chtěla bych poděkovat své rodině za všechnu jejich podporu, za to, že jít na vysokou mohlo být jen mou volbou, i za všechny pochvaly a uznání za mé úspěchy.

Do třetice bych chtěla poděkovat své vedoucí za konzultace, rady i za její trpělivost.

Ráda bych přidala ještě nějaká poděkování nejspíš o kus překvapivější.

Děkuju Nice, holčině, s kterou jsme se sobě dávno ztratily, ale díky které jsem se kdysi na základce začala hrabat ve webech a tím si otevřela cestu k informatice.

Děkuju Kirarovi, Blackýmu, Glutexovi, Glurakovi, En-Cu-ovi, Nikešovi, Charlovi a vůbec všem těm klukům, kteří se mnou – často za nocí někde na skautských klubovnách – rozebírali všelijaké postupy a triky a tím podněcovali moji lásku k programování.

Děkuju všem svým svěřencům, zejména Mitchí, Glantovi, Pavlovi, Dominikovi a Terce, že i když jsem začala studovat dvě školy, chtěli mít dál kroužek a že mě každý týden, když jsem těch 90 kilometrů na schůzku přijela, nadšeně vítali. Možná i díky nim dnes nepovažuju za nemožné něco, co nemožné zdaleka není.

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 21. 5. 2014

.....

Abstrakt / Abstract

V neurologii se běžně zpracovávají data z magnetické rezonance mozku. Ke zpracování těchto dat existuje softwarový balík FreeSurfer. FreeSurfer ovšem poskytuje ovládání pouze z příkazové řádky, což je pro mnohé lékaře nevyhovující. Tato práce se proto zabývá návrhem a realizací příjemnějšího, grafického, rozhraní. Celá práce vychází z principů user-centered design, hlavním cílem je vyjít vstříc uživateli. Návrh a testování probíhá ve spolupráci s Neurologickou klinikou 2. lékařské fakulty Univerzity Karlovy a Fakultní nemocnice Motol.

Results from magnetic resonance imaging of brain are commonly processed in neurology. Software package called FreeSurfer is available for processing of such data. FreeSurfer however provides only command line interface which is inconvenient for many physicians. This thesis therefore focuses on design and implementation of a more comfortable, graphical, user interface. The whole thesis is based on user-centered design, the main goal is to make it enjoyable for the user. Design and testing is realised in cooperation with Neurology clinic of the 2nd Faculty of Medicine of Charles University and University Hospital Motol.

Title translation: Design and implementation of GUI for FreeSurfer

Obsah /

Obrázky

Kapitola 1
Úvod

1.1 Motivace

FreeSurfer, sada nástrojů ke zpracování dat z magnetické rezonance mozku, má zajímavý potenciál pro medicínu. Volumetrická data (data o objemu jednotlivých mozkových struktur) jsou důležitá v boji s Alzheimerovou nemocí a různými jinými formami demence, kde napomáhají ke kvantifikaci atrofie mozku. Tato možnost kvantifikace je důležitá hned nadvakrát. V první řadě pomáhá stanovit správnou diagnózu a následně vhodnou léčbu, ve druhé řadě, o nic méně významné, umožňuje lépe plánovat, realizovat, a zejména vyhodnocovat klinické studie zabývající se účinností nových léků, čímž pomáhá klinickému výzkumu.

Úspěšné používání FreeSurferu ovšem vyžaduje pohyb v prostředí UNIXového shellu a drobné skriptování. Pro správce systému na bázi UNIXu by nejspíš šlo o naprosto nezajímavou činnost, ani zkušení uživatelé takových systémů by to zřejmě nepovažovali za významnou komplikaci, ovšem pro většinu lékařů je toto prostředí cizí a svým způsobem velice nepřátelské. Mnozí z nich jsou zvyklí na prostředí OS Microsoft Windows a na různá klikací rozhraní. Práce s FreeSurferem je tak pro tyto lékaře nepohodlná a nepříjemná.

Lidé obecně to, co je pro ně nepohodlné a nepříjemné, dělají neradi, a jen proto, že z nějakého důvodu musí. Navíc práce s nepohodlným nástrojem bývá i nějakým způsobem neefektivní. V případě lékařů a snímků mozku je celá situace umocněná přirozenou obavou, že člověk v neznámém prostředí něco pokazí. Přitom ve chvíli, kdy se pracuje s lékařskými snímky, může mít takové pokažení neblahý vliv na něčí zdraví, a to může svádět k (byť podvědomému) vnímání vyhýbání se tomuto nástroji jako ochrany pacientů.

Získávaná data mají ale veliký přínos pro lidstvo, a proto by bylo vhodné nabídnout lékařům přívětivější nástroj, a tím je více motivovat k jeho využívání. Proto se tato práce snaží nabídnou prostředí, s kterým se bude pracovat co nejpříjemněji – alespoň v rámci možností toho, jak příjemná může být práce s nemocí a často i s nějakou ztrátou důstojnosti či lidskosti.

1.2 Cíl práce

Cílem této bakalářské práce je vytvořit uživatelsky co nejpříjemnější rozhraní umožňující realizaci úloh nejčastěji řešených na Neurologické klinice FN Motol. Výsledné GUI by mělo přinést efektivnější a příjemnější používání nástroje FreeSurfer. Jelikož cíloví uživatelé FreeSurferu, obvykle neurologové, představují už z principu malou skupinu lidí, která se navíc typicky pohybuje ve výrazně jiném prostředí než běžní programátoři, zdá se lepší přizpůsobit GUI konkrétním potřebám několika z nich (v případě této práce tedy neurologům z FN Motol a z 2. lékařské fakulty UK) než snažit se co nejobecněji popsat možné způsoby používání FreeSurferu a možné preference na skladbu GUI.

Součástí realizace měly být a byly průběžné konzultace právě s neurology z FN Motol a 2. LF UK a úpravy v souladu s jejich připomínkami. Nelze očekávat, že se hned na první pokus navrhne GUI vhodné pro každodenní používání, proto musí být používání GUI nebo alespoň jeho částí testováno průběžně.

1.3 Jazyk práce

Speciální poznámku si dovolím k jazyku této bakalářské práce. Ani mně, ani mé vedoucí nepřišel z výběru čeština nebo angličtina nějaký jazyk výrazně lepší, jelikož se ovšem jedná o práci vznikající v českém prostředí, v rámci české univerzity a ve spolupráci s českou nemocnicí, domluvily jsme se nakonec na vypracování v českém jazyce.

Program jako takový včetně dokumentace bude ale vypracovaný kompletně v angličtině, a to z několika důvodů. Předně, FreeSurfer je celosvětově rozšířený nástroj, což dává jistou šanci na pozdější distribuci GUI i do jiných míst než je FN Motol. Kdyby byl ovšem program a dokumentace v češtině, bylo by v případě rozšíření nutné vše lokalizovat alespoň do angličtiny. Zde považuji za důležité zdůraznit, že ti uživatelé, s kterými bude GUI v průběhu vývoje konzultováno, nejen ovládají anglickou terminologii a dokáží se orientovat podle anglického textu, ale obvykle spolupracují se zahraničními pracovištěmi, takže pro ně realizace v angličtině nepředstavuje komplikaci při používání programu. Mezi další důvody patří zejména to, že k samotnému FreeSurferu neexistuje lokalizovaná dokumentace, tedy by buď bylo třeba mísit v programu běžná česká označení s anglickou terminologií, nebo používat (typicky neustálené) české překlady, ke kterým by bylo těžké v případě potřeby dohledávat další informace v jiných materiálech.

Kapitola 2
FreeSurfer v nemocnicích

2.1 FreeSurfer jako softwarový balík

Pod označením FreeSurfer se skrývá soubor jednak několika algoritmů pro různé způsoby zpracování dat ze snímání mozku, jednak několika menších programů určených k prohlížení získaných dat, modelů mozku atp. To všechno je dohromady šířené jako jeden obsáhlý softwarový balík.

Bruce Fischl píše [1], že vývoj začal již v roce 1994. Dnes se o vývoj stará tým z Laboratoře pro výpočetní neurozobrazování (*Laboratory for Computational Neuroimaging*) v Centru pro biomedicínské zobrazování Martinos (*Martinos Center for Biomedical Imaging*). V současné době je FreeSurfer šířený jako open source software, přestože pro jeho využívání (nikoli samotnou instalaci) a plný přístup ke zdrojovému kódu se vyžaduje licenční klíč. K získání licenčního klíče slouží plně automatizovaná registrace; od uživatelů je vyžadováno jméno, adresa a jejich akademická či výzkumná instituce.

FreeSurfer má poměrně velké systémové požadavky (doporučen je procesor s frekvencí alespoň 2 GHz, alespoň 4 GB RAM a 3D grafická karta s vlastní pamětí) a běh konkrétních skriptů i při splnění těchto požadavků trvá velmi dlouhou dobu (částečné zpracování snímků z MRI jednoho pacienta běžně zabere okolo jednoho plného dne, tedy 24 hodin). Tento běh se navíc podle wiki FreeSurferu [2] moc nedá zrychlit pomocí paralelizace, resp. není možné rozumně paralelizovat výpočet pro jeden objekt (pro data z jednoho snímání jednoho pacienta). Co se udělat dá, je na stroji s vícejádrovým procesorem pouštět najednou zpracování několika objektů, taková zpracování dohromady zaberou jen o málo delší dobu (řádově o 10 %) než zpracování jednoho objektu. Při paralelním zpracování ovšem rostou požadavky na operační paměť, je důrazně doporučeno mít k dispozici 4 GB RAM na každý zpracovávaný objekt.

Většina konkrétních funkcí FreeSurferu vyžaduje data v nějakém specifickém formátu, případně již předzpracovaná, takže nedílnou součástí práce s FreeSurferem bývá i převádění dat do jiného formátu, různé úkoly navíc mají poměrně jasné postupy, které nelze výrazně obměňovat.

Navzdory pokroku technologií ovšem není možné FreeSurfer využívat plně automatizovaně, bez zapojení člověka. Například skripty pro odstranění lebky ze snímků občas kus lebky nechají, nebo naopak odstraní i kus mozku. Zrovna u odstraňování lebky často stačí změnit parametr skriptu, ale už je tu potřeba člověk, který pozná, že takhle by to asi vypadat nemělo, a pustí skript znovu s upraveným parametrem.

K FreeSurferu existuje relativně obsáhlá dokumentace na stránkách projektu [1]). Jak uživatelům, tak vývojářům je navíc k dispozici velmi aktivní e-mailová konference. Zejména dokumentace ve formě Wiki předpokládá uživatele odborníka, texty se velkou měrou zaměřují na medicínské pozadí používaných algoritmů.

Wiki FreeSurferu obsahuje i část věnovanou GUI, jedná se však ve všech případech o programy k prohlížení či editaci výsledných dat, nikoli o GUI umožňující samotnou přípravu dat či získání výsledků.

[1]) `https://surfer.nmr.mgh.harvard.edu/`

2.2 NeuGRID

NeuGRID[1]) se prezentuje jako webový portál určený zejména neurologům a neurovědcům. Tento projekt si klade za cíl několik věcí, mezi nimi nabídnout uživatelsky přívětivé prostředí neurovědcům, a tím jim umožnit lépe překonávat různé překážky ve výzkumu.

Pro neurology má velký význam zejména možnost využívat neuGRID ke zpracování dat z MRI na cloudu. K tomu je sice nejprve potřeba projít relativně složitým registračním procesem obnášejícím mimo jiné vygenerování a nahrání osobního certifikátu, ale následné využívání už nevyžaduje žádné speciální znalosti. Přímo s registračním procesem navíc může pomoci technik, případně nějaký jiný zkušenější uživatel.

NeuGRID nedokáže odstínit problematiku související s ovládáním FreeSurferu, představuje ale řešení některých jiných problémů zmíněných v následující části. Z tohoto hlediska by se tak mohlo jevit prioritní propojení GUI (které má odstínit právě problémy s ovládáním, ale ostatní problémy příliš řešit nemůže) s neuGRIDem. V době, kdy jsem na GUI začínala pracovat (únor 2014), jsem ovšem měla od spolupracujícího medika zprávy, že se uvažuje o ukončení projektu. I z tohoto důvodu byla možnost propojení s neuGRIDem ponechána jako hudba sladké budoucnosti.

2.3 Problémy využití v nemocničním prostředí

Pohyb v příkazové řádce a skriptování

Jak zmiňuji i v jiných částech této práce, zásadním problémem využívání FreeSurferu v nemocničním prostředí je nutnost pohybovat se v příkazové řádce a zejména nutnost různých úprav skriptů řídících běh programu. Pro lékaře se typicky jedná o neznámé prostředí a nezvyklý styl práce. Skriptům, které upravují, obvykle nerozumí, což je nutí postupovat podle přesně připravených návodů a kontrolovat, že nezměnili nic jiného než to, co mají v onom návodu uvedeno.

Práce s něčím, čemu člověk nerozumí, obvykle nebývá příjemná, právě kvůli tomu nechápání a související nejistotě. Zároveň povinností „běžných“ lékařů v žádném případě není rozumět skriptování, byť jednoduchému, v nějakém specifickém jazyku na specifickém systému. Jejich povinností je až porozumět datům, která dostanou ze speciálních programů, správně je interpretovat a získanou znalost použít při rozhodování o dalším postupu léčby, případně při dalším plánování výzkumu.

Přesně tento problém se snaží GUI pro FreeSurfer odstranit. Ovládání FreeSurferu prostřednictvím GUI by mělo být realizováno pouze na principu zadávání konkrétních informací a výběru z nabídnutých možnosti. V ideálním případě by měla být zcela odstíněna problematika používání konkrétního systému a skriptování.

Časová náročnost

Zpracování dat pomocí FreeSurferu trvá dlouho. Abych se vyjádřila o něco exaktněji, plné zpracování dat tak, jak ho využívají na NK FN Motol, běžně zabere okolo 55 hodin.

Po celou dobu zpracování musí samozřejmě běžet počítač, na kterém zpracování probíhá. Během zpracování je možné dělat na počítači i jiné činnosti při zachování rozumné rychlosti zpracování, přesto to znamená ohromné vytížení nemocničních počítačů. Množství počítačů na oddělení navíc bývá menší než množství pacientů, u kterých

[1]) `https://neugrid4you.eu/`

za těch 55 hodin lékaři provedou vyšetření. Tím vzniká „přetlak“ snímků, data se nestíhají zpracovávat.

Dalším nepříjemným důsledkem dlouhé doby zpracování je to, že zpracování se nedá zvládnout v jednom kuse. Lékaři musí neustále myslet na to, že existují data, nad kterými běží zpracování a která bude po doběhnutí tohoto zpracování potřeba zkontrolovat, možná dokonce nějak upravit.

Tento problém významnou měrou řeší dříve zmíněný neuGRID. NeuGRID pochopitelně neodstraňuje nutnost myslet na to, že se zpracovávají nějaká data, u kterých tedy bude potřeba zkontrolovat výsledky, ale umožňuje snížit vytížení nemocničních počítačů a zmíněný přetlak.

Malá schopnost zotavení z chyb

Při zpracování dat může docházet k různým chybám, které algoritmy provádějící zpracování někdy neumí ani detekovat, a typicky je neumí opravit. Lékaři musí výsledky zpracování kontrolovat a v případě problémů nějak upravovat skripty, které zpracování řídí.

Rozpoznání některých chyb vyžaduje odborné znalosti, řešení jsou ovšem často nějakým způsobem rutinní a automatizovatelná. Nutnost poznat chybu a reagovat na ni nejspíš dlouhodobě zůstane na lékařích, GUI by ale mělo i v případě řešení chyb odstínit problematiku skriptování a nabídnout rutinní postupy.

2.4 Úlohy běžně řešené na NK FN Motol

Návrh GUI se zaměřuje zejména na zpříjemnění často prováděných činností, samozřejmě těch činností, které jsou svázané s používáním FreeSurferu. K tomu je ovšem nezbytně nutné vědět, co vlastně mezi tyto často prováděné činnosti patří. Chtěla bych zde proto tyto činnosti shrnout a stručně popsat.

Zároveň bych na tomto místě chtěla zdůraznit, že podobně jako není povinností lékařů rozumět skriptování, nebývá povinností odborníků na informatiku rozumět medicíně nebo metodám medicínského výzkumu. Ani já jsem se s těmito oblastmi nemusela důsledně seznamovat, neboť skripty, pro jejichž vytvoření jsou tyto znalosti nezbytné, již existují a já ve své práci pouze buduji úroveň nad nimi.

> Veškerá má vysvětlení prováděných činností proto musí být považována za maximálně laická a neměla by být v žádném případě použita jako zdroj informací v pracích zabývajících se těmito oblastmi podrobněji.

Zpracování dat

Z dat získaných během MRI se zjišťují různé údaje o objemu jednotlivých mozkových struktur, zejména celkový objem subkortikálních struktur a průměrná tloušťka mozkové kůry (šedé hmoty).

Tato data se mohou principiálně zpracovávat dvěma způsoby, podle toho, zda jsou součástí *průřezové studie* (v angličtině *cross-sectional study*), nebo *longitudinální* studie (*longitudinal study*).

V případě průřezové studie jde opravdu o naměřené hodnoty, o absolutní čísla. Data pro takovou studii jsou získána v nějakém časovém okamžiku (*timepoint*), ke kterému patří, ale nespojují se s daty z významně jiného období. Pro potřeby výzkumu se s naměřenými hodnotami obvykle ukládají i další data, která potenciálně mohou být faktory ovlivňujícími tyto hodnoty – jedná se např. o pohlaví, věk, konkrétně v případě výzkumu Alzheimerovy choroby samozřejmě i o diagnózu atp.

Naopak v případě longitudinální studie se zejména sleduje změna mezi dvěma měřeními. Porovnávají se dva výsledky pocházející od stejné osoby, ale ze dvou různých časových okamžiků. Ty od sebe typicky bývají časově vzdálené cca 1 rok, ale přesná hodnota záleží na několika faktorech včetně toho, jak se povede danou osobu objednat a znovu jí MRI provést. Délka období mezi porovnávanými časovými okamžiky nijak přímo nevyplývá z fungování longitudinální studie, volí se s ohledem na zkoumaný jev, očekávanou dobu nutnou pro pozorování projevů změny atp.

Longitudální zpracování dat, alespoň ve FreeSurferu, ovšem vyžaduje předchozí zpracování dat stejným způsobem jako pro průřezovou studii. Teprve ve spojení s výsledky tohoto (průřezového) zpracování je možné provádět longitudinální analýzu.

Se zpracováním dat souvisí i nutnost data v nějaké fázi jejich použití pro výzkum anonymizovat. Např. při využití neuGRIDu musí být data anonymizována ještě před nahráním na Grid, při čistě lokálním zpracování v nemocnici, kde byly snímky pořízeny, se považuje za dostatečné data anonymizovat až při případném zveřejnění, např. formou článku.

GUI zatím možnost anonymizace nijak neřeší, data o pacientech se načítají ze souboru vloženého uživatelem (tj. typicky lékařem) a pouze na onom uživateli záleží, zda budou pacienti označeni jednoznačně identifikujícím způsobem, nebo např. nějakým přiděleným kódem. V době, kdy GUI umožňuje pouze lokální zpracování, není potřeba anonymizaci více řešit, ovšem při rozšíření funkcionality o podporu vzdáleného zpracování by jistě bylo vhodné mít tuto problematiku na paměti a případně se pokusit uživateli nabídnout možnost systematické anonymizace dat.

Kontrola získaných dat

Jelikož k problémům provázejícím využívání FreeSurferu patří i malá schopnost zotavení z chyb a občasné zcestné výsledky, mezi běžné činnosti patří i kontrola získaných dat.

Tato kontrola má typicky formu jednoduchého zhodnocení pohledem, kdy se lékař podívá na získaná data spolu s původními snímky a rozhodne, zda příslušná data dávají smysl, nebo ne (v takovém případě může rozhodnout buď o znovuprovedení analýzy, nebo o vyřazení dat).

V současné době již existují nástroje, které umožňují vizualizovat získaná data. Speciálně pro případ komplikací by ovšem bylo užitečné umožnit uživateli jednoduše zopakovat analýzu, tedy propojit tyto vizualizační nástroje se zbytkem funkcionality FreeSurferu.

Extrakce dat pro další zpracování

Data pro výzkum se (alespoň není-li výzkum ukončen bez nějakého zveřejňovaného výsledku) nemohou jenom tak zahodit, navíc už jejich sesbírání a zpracování obvykle zabere delší dobu. Je proto nutné data průběžně ukládat a uchovávat. Právě to ovšem při používání FreeSurferu může představovat další překvapivě velký problém.

Část získaných dat FreeSurfer vypisuje na standardní výstup, všechna data pak ukládá ve speciálním formátu na pevný disk. FreeSurfer sám o sobě poskytuje nástroj k uložení těchto dat do formátu, ve kterém se dají jednoduše importovat do jiných programů, ovšem opět se jedná o skript ovládaný z příkazové řádky, tedy by bylo vhodné udělat k němu nějaké příjemnější rozhraní.

Kapitola 3
Testování

Použitelnost softwaru se obecně dá testovat mnoha způsoby, které jsou často jen formalizací logických, přirozených postupů. Pro výběr konkrétní metody bývá důležité, zda máme možnost a zájem testovat s uživateli, nebo ne, a dále to, zda máme možnost se s uživateli fyzicky setkat.

Testování s uživateli je typicky po všech stránkách náročnější než testování bez nich, zároveň ale obvykle přináší mnohem relevantnější výsledky.

Právě z důvodu dostatečně malé náročnosti jsem všechny návrhy GUI testovala pomocí kognitivního průchodu a heuristické evaluace, což jsou dvě metody umožňující testování bez uživatele. Podrobněji je popisuji v následující sekci, v jejich popisu doslovně cituji popis z mé semestrální práce do předmětu A4B39TUR Testování uživatelského rozhraní [3].

Klasické testování s uživateli probíhalo až u implementovaného návrhu z běžných metod ale spíše vycházelo, než že by jim přesně odpovídalo. Důležité ovšem je, že testování probíhalo už během vývoje – jak ostatně píše i David Travis [4], při designu zaměřeném na uživatele je důležitý včasný a vytrvalý zájem o uživatele a jeho úkoly.

Přesto jsem všechny návrhy ukazovala spolupracujícím lékařům a bavila se o nich s nimi. Jak totiž píše např. Carolyn Snyder [5], třeba pro testování navigace nebo rozvržení stránky/okna stačí náčtrty či naklikaná rozhraní s nějakým popisem, jak by se to mělo chovat.

3.1 Metody testování

Pro úplnost si dovolím zdůraznit, že níže předložený seznam metod testování není zdaleka úplný. Představuje ovšem ty metody, které jsem nějakým způsobem řešila při vypracování této práce – metody, které jsem použila, nebo o jejichž použití jsem při vývoji a přípravě testování uvažovala.

3.1.1 Kognitivní průchod

Popis metody

Metoda kognitivního průchodu se používá v situacích, kdy je možné testovaný případ užití rozdělit na jednotlivé kroky, mezi kterými se postupně přechází.

Nejprve se stanoví cíl, kterého může chtít uživatel dosáhnout a který chceme otestovat. Někdy může být vhodné určit podmnožinu cílové skupiny, z jejíhož pohledu bude celé testování prováděno. Následně se provede právě rozdělení na jednotlivé kroky. Každý z těchto kroků se dál podrobí zkoumání spočívajícím v zodpovězení následujících tří otázek:

1. Bude uživateli jasné, co by měl v tomto kroku udělat?
2. Spojí si uživatel dostupné popisy kroku se svým cílem?
3. Dostane uživatel na svou akci dobrou zpětnou vazbu?

Na jednotlivé otázky se odpovídá „ano“, nebo „ne“. V případě záporné odpovědi je nutné přidat i komentář, proč byla zvolena tato odpověď, někdy může být vhodné přidat poznámku i ke kladné odpovědi.

Jednoznačnými výhodami metody kognitivního průchodu je velká rychlost a nízká cena testování. Kognitivním průchodem je navíc možné testovat i pouhý návrh, který zatím není plně implementován.

Kognitivní průchod se ovšem neprovádí se skutečnými uživateli, rozhodně tedy nemusí odhalit všechny problémy, a naopak může upozornit na potenciální problémy, se kterými by se ovšem skuteční uživatelé z nějakého důvodu nepotýkali.

Použití kognitivního průchodu může být problematické i v případě, kdy sa daného cíle dá dosáhnout mnoha různými způsoby, což např. u webových stránek nastává relativně často (v důsledku vyhledávacích funkcí, řazení stránek pod více kategorií atp.).

Použití při vývoji GUI

Jak jsem zmínila hned v začátku této kapitoly, pomocí kognitivního průchodu byly testovány všechny návrhy GUI, a byly tak testovány i postupně zapracovávané změny a úpravy. Výsledky testování v této práci uvádím pro testování prvotních návrhů (v kapitole 4 o těchto návrzích) a pro skutečně implementovaný návrh (v kapitole 5 o realizovaném návrhu).

3.1.2 Heuristická evaluace

Popis metody

Základem heuristické evaluace je předem stanovedá sada pravidel/zásad, které musí předmět testování splňovat. Při samotném testování se pak pro každé pravidlo zvlášť posuzuje, zda ho daný předmět testovaní splňuje, nebo ne. Minimálně případě záporné odpovědi se k výsledku testování přidává komentář, čím přesně předmět testování dané pravidlo porušuje (takových věcí může být i několik).

Sada pravidel používaná pro testování není a nemůže být plně univerzální, tedy ani jednoznačně stanovená. Ve své práci vycházím z definice použitelnosti podle Jakoba Nielsena [6], která zavádí tato pravidla, resp. tyto heuristiky:

- Viditelnost stavu systému
  Systém by měl uživatele vždy informovat o své stavu a o tom, co se v něm děje, a to pomocí vhodné zpětné vazby v rozumném čase.
- Shoda mezi systémem a skutečným světem
  Systém by měl mluvit jazykem uživatele, měl by používat slova, věty a koncepty, které jsou uživateli známé, spíše než pojmy vycházející ze systému samotného. Systém by měl následovat konvence skutečného světa a zajišťovat, že se informace objeví v přirozeném a logickém pořadí.
- Uživatelské ovládání a uživatelská svoboda
  Uživatelé často zvolí nějakou systemovou funkci omylem a potřebují možnost opustit nechtěný stav bez nutnosti procházet složitým dialogem.
- Konsistence a standardy
  Uživatelé by neměli být nuceni zjišťovat, jestli různá slova, situace nebo akce znamenají to samé. Systém by měl dodržovat konvence využívané platformy.
- Předcházení chyb
  Systém by měl co nejvíc bránit výskytu chyb pomocí pečlivě navrženého designu. Podmínky náchylné k chybám by měly být eliminovány, případně by se mělo jejich

nastání pečlivě hlídat a od uživatele by mělo být vyžadováno potvrzení před provedením jakékoli potenciálně nebezpečné akce.

- Preference rozpoznání před rozpomínáním se
  Akce a možnosti by měly být vidět. Uživatel by neměl být nucený pamatovat si informace z jedné části dialogu do druhé. Instrukce k použití systému by měly být viditelné celou dobu, nebo by je alespoň mělo být celou dobu snadné opět vyvolat.
- Flexibilita a efektivita užití
  Často prováděné akce by mělo být možné udělat rychle, zároveň by různé způsoby urychlení neměly příliš zatěžovat nezkušené uživatele.
- Estetika a minimalistický design
  Dialog by neměl obsahovat informace, které nejsou relevantní, ani ty, které jsou potřeba jen vzácně. Design by měl zajišťovat co nejlepší relativní viditelnost důležitých informací.
- Pomoc uživateli rozpoznat chyby, diagnostikovat je a zotavit se z nich
  Chybové zprávy by měly být sdělovány běžným jazykem, přesně určit příčinu problému a konstruktivně navrhnout řešení.
- Pomoc a dokumentace
  V jakékoli dokumentaci by se mělo dát jednoduše vyhledávat, popisovat jednotlivé kroky a nebýt příliš obsáhlá.

Použití při vývoji GUI

Podobně jako kognitivní průchod jsem heuristickou evaluaci použila k otestování všech prvotních návrhů i návrhu skutečně realizovaného, výsledky jsou uvedeny v příslušných kapitolách.

3.2 Testování v laboratoři použitelnosti

Popis metody

Testování v laboratoři použitelnosti poskytuje nejautentičtější obrázek o skutečné použitelnosti testovaného produktu. Probíhá s reálnými uživateli v kontrolovaném prostředí, takže upozorní i na situace, kdy by např. uživatel za jiných okolností použil vyhledávač a splnil úkol s pomocí nalezených informací.

Laboratoř použitelnosti tvoří dvě místnosti. První z nich je určená pro účastníka testování a moderátora, druhá pro pozorovatele z testujícího týmu (a případně další pozvané lidi, např. investory či vedoucí projektu).

V účastnické místnosti je připravené pracovní prostředí, zejména počítač s nainstalovaným softwarem určeným k testování. Dění v této místnosti nahrávají kamery, obvykle spolu s mikrofonem či mikrofony, dále se typicky zaznamenává obraz z displeje počítače, na kterém účastník plní úkoly.

Záznamy z účastnické místnosti se v reálném čase promítají do místnosti pozorovatelů, pozorovatelé tak mohou sledovat snahy účastníka o splnění úkolů, a zejména jeho reakce na chování testovaného softwaru. I pro tyto účely bývá vhodné, aby účastník nahlas komentoval své chování, čeho a jak se snaží docílit.

Použití při vývoji GUI

Tato metoda nebyla navzdory hodnotě výsledků při vývoji použita. Hlavním důvodem byl nedostatek lidských sil a zdrojů, neměla jsem k dispozici dostatek uživatelů

vhodných pro testování (takových, kteří mimo jiné rozumí pojmům ze zpracování dat z MRI) ani další lidi do testujícího týmu (striktně vzato by bylo možné provést celé testování v jednom člověku, ale je důrazně doporučeno, aby byl testující tým vícečlenný).

3.3 Vzdálené testování

Popis metody

Ve skutečnosti je vzdálené testování spíše skupinou metod, než konkrétní metodou samou o sobě, ale v metodě, kterou tímto pojmem aktuálně označuji, jde zejména o to, že účastník dostane seznam úkolů v elektronické podobě, pokusí se je splnit ve svém vlastním prostředí a následně opět elektronickou formou pošle nějakou zprávu o tom, jak se mu plnění úkolů dařilo.

S ohledem na to, že jde o nekontrolované prostředí, není nikdy jisté, zda měl účastník opravdu klid na plnění úkolů nebo zda nenastal nějaký neočekávaný problém nesouvisející s testovaným softwarem (v extrémním případě může jít třeba o pád operačního systému z nesouvisejícího důvodu).

Ne zcela spolehlivé jsou i zprávy od účastníka – ani ne tak proto, že by si účastník měl záměrně vymýšlet (ačkoli mnozí lidé mají tendenci příliš se nepřiznávat ke svým neúspěchům), ale proto, že ho třeba nenapadne zmínit něco, čeho by si pozorovatelé při testování v laboratoři všimli. Jak píše Donald Norman v úvodu druhé kapitoly [7] a jak bývá často zdůrazňováno na přednáškách z Testování uživatelského rozhraní [8], lidé mají tendenci své neúspěchy s technikou považovat za své selhání, nikolik za chybu té techniky, a málokdy jsou sami na sobě schopni objektivního zhodnocení, zda daný neúspěch opravdu byla jen jejich chyba.

Použití při vývoji GUI

Vzdálené testování jsem při vývoji používala. Dokud bylo GUI opravdu jen ve vývoji, bylo testování velice nekonkrétní („zkuste si projít chování GUI, zda vám dává smysl, nebo jestli pro vás něco bude neintuitivní“), později přibylo na konkrétnosti.

Z výsledků jsou pouze vybrány dosud neopravené problémy, ty jsou uveřejněné v sekci 5.4.

3.4 Testované případy užití

3.4.1 Průřezové zpracování dat

Modelová situace: Lékař získal data z MRI pro dva pacienty, vytvořil pro ně korektní soubor `.xls` a nyní chce zadat jejich průřezové zpracování.

Případ užití testuje zadání nového zpracování, nastavení všech potřebných údajů a průřezové zpracování naměřených dat.

Ve výchozí situaci má uživatel čerstvě spuštěné GUI, na počítači uložený soubor `.xls` a získaná data, na konci má data průřezově zpracovaná.

3.4.2 Extrakce dat

Modelová situace: Lékař již dříve provedl průřezové zpracování dat, nyní by ale ještě rád získal data z tohoto zpracování.

Případ užití testuje návrat k rozpracovanému zpracování a extrakci dat.

Ve výchozí situaci má uživatel čerstvě spuštěné GUI, na počítači uložený soubor `jobstatus.mat`, na konci má uložená extrahovaná data.

3.4.3 Srovnávací longitudinální zpracování dat

Modelová situace: Lékař má dokončené základní longitudinální zpracování dat a nyní chce pro každý časový okamžik každého pacienta provést ještě srovnávací longitudinální zpracování.

Případ užití testuje pokračování ve zpracování a provedení srovnávacího longitudinálního zpracování.

Ve výchozí situaci má uživatel spuštěné GUI, ve kterém právě skončilo základní longitudinální zpracování, na konci má hotové srovnávací longitudinální zpracování dat.

Kapitola 4
Návrh GUI

4.1 Základní principy

Vytvoření dvou (více) vrstev

V rámci tvorby GUI by měla vzniknout nikoli pouze vrstva samotného GUI, nýbrž i vrstva mezi tímto GUI a FreeSurferem. GUI jako takové představuje primárně vlastní grafický návrh a případná omezení průchodu, vynucené řetězení akcí atp. Mezivrstva pak zajišťuje komunikaci tohoto GUI s FreeSurferem, zejména spouštění jednotlivých skriptů.

Oddělení grafického návrhu od komunikace s FreeSurferem přispívá udržitelnosti celého programu a usnadňuje částečné úpravy. Při změně návrhu jsou stále k dispozici funkce z komunikační vrstvy, není tedy potřeba s přechodem k jinému grafickému rozvržení zahazovat i kód zajišťující vlastní spouštění FreeSurferu.

Naopak v případě, že by došlo ke změně ve skriptech FreeSurferu, například by se začaly spouštět jiným způsobem, stačí provést patřičné změny v komunikační vrstvě, do samotného grafického návrhu se vůbec nemusí zasahovat.

Rychlé reakce

Rozhodně není možné, aby se GUI zablokovalo na dlouhou dobu a nereagovalo na vstup uživatele, dokud například neskončí probíhající výpočet. Už pokud program nereaguje vteřinu, uživatel si toho všimne a obvykle značně znejistí, začne se obávat, že došlo k nějaké chybě, v jejímž důsledku program „spadl“ či brzy spadne.

Běžný uživatel navíc nemá žádnou možnost zjistit, zda program pouze potřebuje více času na dokončení složitého výpočtu, nebo zda při jeho provádění například nevznikl nekonečný cyklus, tedy program bude v každém případě nutné ukončit nestandardními prostředky.

Speciálně u FreeSurferu, kde se doba zpracování pohybuje v řádu hodin až dní, nepřipadá čekání na dokončení zpracování v úvahu. Zároveň ovšem příliš nevadí, pokud program oznámí konec zpracování s menším zpožděním (v řádu vteřin), protože v tomto případě je pro uživatele doba několika vteřin nerozlišitelná.

Střídmé možnosti nastavení

Při návrhu uživatelského rozhraní je vždy potřeba vycházet z konkrétní situace a konkrétní představy cílových uživatelů, mimo jiné z jejich očekávaných schopností a znalostí dané problematiky. Obecně by se ovšem mělo předcházet přehlcení uživatele různými možnostmi a nastaveními.

Obzvláště v případě, kdy se grafické rozhraní buduje nad programem v příkazové řádce, který má mnoho různých možných nastavení a jehož dokumentace může být pro nové uživatele značně matoucí, je naprosto nežádoucí vyměnit pro uživatele zmatenou, až děsivou dokumentaci k nástroji v příkazové řádce za zmatené, až děsivé grafické rozhraní ke stejnému nástroji.

Alespoň ve výchozím stavu by proto GUI mělo nabízet jen základní nastavení. Máli GUI umožňovat i nastavení složitějších akcí, přestože se počítá také se začínajícími uživateli, mělo by toto složitější nastavení buď být skryté pod položkou typu *pokročilé*, nebo např. zapnutelné v nějakém nastavení programu jako takového.

Dodržování zavedených konvencí

Uživatelské rozhraní libovolného programu by mělo respektovat zavedené konvence programů podobného charakteru. Nemělo by proto zavádět nová jména pro již existující označení, bez ohledu na to, zda se jedná o ovládací prvky, možné akce, ... Navíc by uživatelské rozhraní mělo dodržovat konvence platformy, na kterém běží, případně kultury, pro kterou je určené atp.

Jelikož FreeSurfer je velice specifický nástroj, a žádné GUI pro něj zatím neexistuje, konvence, které by se měly dodržovat (alespoň z hlediska samotného FreeSurferu), prakticky neexistují. Na druhou stranu by se o to větší pozornost měla věnovat zavádění různých označení, neboť právě ta by se teoreticky mohla později stát právě konvencí, kterou budou případné jiné programy dodržovat.

4.2 Výběr platformy

Při výběru vhodné platformy se obvykle bere v úvahu mnoho věcí, typicky třeba požadavky na rychlost, dostupné knihovny, přenositelnost apod. Cílem této práce od začátku bylo a je zejména vyjít vstříc lékařům, přizváni byli i k výběru platformy.

Jelikož přizvaní lékaři často pracují s Matlabem, a zejména v něm provádějí různé zpracování získaných dat, přáli si, aby i GUI pro FreeSurfer bylo vytvořené nad Matlabem. Mezi nejvýznamnější důvody, proč vybrat Matlab, patřil fakt, že se získaná data dají snadno dále zpracovávat, není potřeba řešit ještě export a import mezi GUI a Matlabem.

Primárně ve snaze vyhovět lékařům byl nakonec skutečně vybrán Matlab. Z toho plynou některé problémy, které blíže popisuji v pozdějších částech práce. Během vývoje se navíc ukázalo, že ani předávání dat z GUI není tak bezproblematické, jak se původně předpokládalo. Přesto jsem problémy považovala za dostatečně malé na to, abych platformu během vývoje už neměnila.

4.3 Prvotní návrhy

Všechny návrhy byly otestovány pomocí kognitivního průchodu a heuristické evaluace. Tyto metody jsou blíže popsány v kapitole 3 o testování, spolu s testovanými případy užití.

4.3.1 Souhrné GUI

Návrh GUI označený pracovně jako *souhrnný* vychází zejména z podkladů od MUDr. Zuzany Nedelské, mé hlavní konzultantky.

Tento návrh počítá pouze s jedním pracovním oknem a jednou pracovní obrazovkou, v níž uživatel může nastavit všechno, co potřebuje. Předpokládá se, že uživatel bude alespoň tušit, jak GUI používat, GUI žádným způsobem nepřipomíná softwarového průvodce. Přesto systém samozřejmě musí být odolný na uživatelské chyby.

Jako jednoznačná výhoda tohoto návrhu se jeví fakt, že uživatel může v kterémkoli okamžiku zkontrolovat libovolné nastavení (včetně takového, které aktuálně nevyužívá, a třeba ani využívat nemůže).

Naopak zjevnou nevýhodou je nepřehlednost, alespoň při prvním setkání. Ovládací prvky jsou nějakým způsobem uspořádané do jednotlivých celků, ve kterých spolu logicky souvisí, ale tyto celky nejsou nijak zvýrazněné, takže pro uživatele může být problematické si jich všimnout a pracovat s nimi.

Návrh je zobrazený na obrázku 4.1.

Testování kognitivním průchodem

1) Průřezové zpracování dat

Referenční průchod je následující:

1. Kliknout na Load
2. Vybrat správný soubor `.xls`
3. Vybrat pracovní adresář
4. Vybrat typ dat
5. Vybrat adresář s daty ke zpracování
6. Vyplnit e-mailovou adresu
7. Kliknout na Copy data
8. Vybrat typ zpracování
9. Kliknout na Local run

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	NE[1]	ANO	ANO
2	ANO	ANO	NE[2]
3	NE[3]	ANO	NE[4]
4	ANO	ANO	ANO
5	ANO	ANO	NE[5]
6	NE[6]	NE[7]	ANO
7	NE[8]	ANO	ANO
8	ANO	ANO	ANO
9	ANO	NE[9]	ANO

1. Nezná-li uživatel práci s GUI, není z ničeho jasné, že musí nejprve nahrát soubor s informacemi o datech
2. Jméno nahraného souboru se sice v GUI zobrazí, ale na úplně jiném místě, než na které se uživatel právě soustředí
3. GUI navádí na to, že je potřeba zvolit nějaký adresář, není ale jasné, že by mělo jít právě o pracovní adresář
4. Vybraný adresář se v GUI zobrazí, ale na jiném místě, než na které se uživatel soustředí
5. GUI nijak nepotvrdí volbu adresáře
6. Uživatel nemá odkud zjistit, že má vyplnit e-mailovou adresu
7. Políčko pro e-mailovou adresu je schované mezi spoustou jiného nastavení
8. Uživatel nemá odkud vědět, že GUI zásadně pracuje s kopií dat, a že má tedy data zkopírovat
9. Tlačítko pro spuštění zpracování je ztracené mezi velkým množstvím jiných tlačítek

I u kroků, u kterých bylo na první otázku odpovězeno ANO, je třeba myslet na to, že uživatel další kroky spíše odhaduje z toho, co mu GUI nabízí, než že by opravdu věděl, co má v danou chvíli udělat. Při odstranění dříve zmíněných závažnějších problémů by tedy jistě stálo za to zaměřit se i na větší návodnost GUI k průchodu.

2) Extrakce dat

Referenční průchod je následující:

1. Kliknout na Load
2. Vybrat správný soubor `jobstatus.mat`
3. Vybrat PVC/No PVC
4. Vybrat Volume/Thickness
5. Kliknout na Extract Data

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO	ANO	ANO
2	NE[1]	ANO	ANO[2]
3	NE[3]	ANO	ANO
4	ANO	ANO	ANO
5	ANO	ANO	ANO

1. Z GUI není nijak jasné, že se má nahrát zrovna `jobstatus.mat` z původního zpracování
2. Po nahrání souboru se data vypíší všude možně, ale i na místě, na které se uživatel zřejmě sousedí, takže bude mít jistotu, že se „něco stalo“
3. Není jasné, že se extrakce musí nějak nastavovat, natož přepínacími tlačítky nacházejícími se až pod tlačítkem, kterým se extrakce spouští

3) Srovnávací longitudinální zpracování dat

Referenční průchod je následující:

1. Vybrat Longitudinal-Long
2. Kliknout na Local Run

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO	ANO[1]	ANO
2	ANO[2]	ANO	ANO

1. Při návratu ke GUI by uživatel možná výběr zpracování musel nejprve najít
2. Podobně jako u 1), při návratu ke GUI by uživatel spuštění nejspíše nejprve hledal

Testování heuristickou evaluací

- Viditelnost stavu systému
  - V GUI není nikde vidět, když dojde např. k chybě při vytváření pracovního adresáře
- Shoda mezi systémem a skutečným světem

  Bez nálezů
- Uživatelské ovládání a uživatelská svoboda

  Bez nálezů
- Konsistence a standardy

Bez nálezů

- Předcházení chyb

  Bez nálezů

- Preference rozpoznání před rozpomínáním se
  - Možnosti a nastavení jsou vidět, ale je jich tolik, že si uživatel stejně musí pamatovat, v které části se nachází která
- Flexibilita a efektivita užití

  Bez nálezů

- Estetika a minimalistický design
  - Kvůli současné viditelnosti různého nastavení pro různé akce se ztrácejí opravdu důležité možnosti
- Pomoc uživateli rozpoznat chyby, diagnostikovat je a zotavit se z nich
  - Návrh GUI nepočítá s oznamováním jiných chyb než těch, ke kterým dojde v rámci zpracování dat prostřednictvím FreeSurferu
- Pomoc a dokumentace
  - Uživatel nemá z GUI možnost zjistit, kde najde podrobnější dokumentaci

Shrnutí výhod návrhu

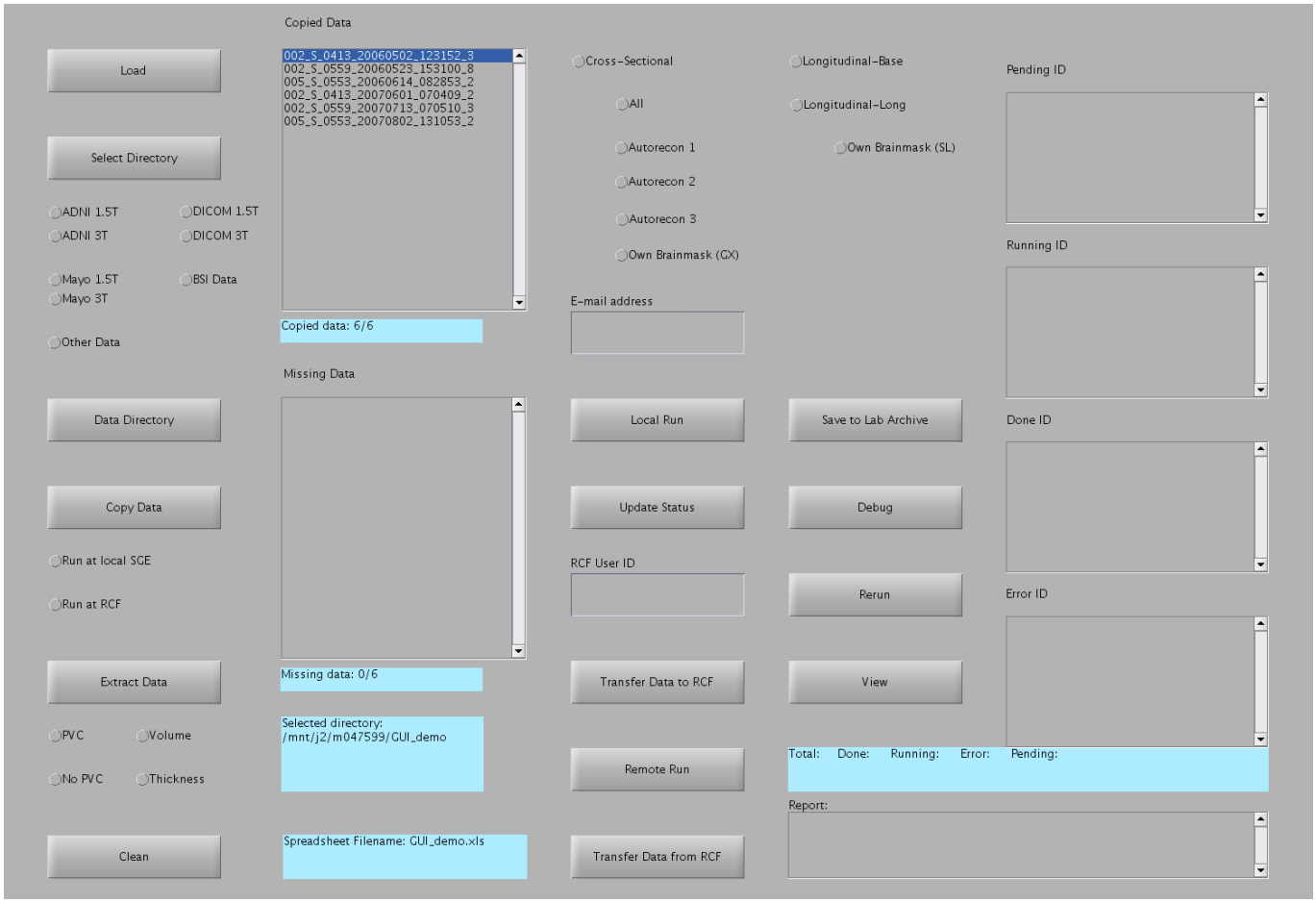
- Možnost kdykoli zkontrolovat všechny údaje

Shrnutí nevýhod návrhu

- Nepřehlednost
- Předpokládá se znalost práce s tímto konkrétním GUI

Prostor pro vylepšení

- Ohraničení jednotlivých celků by mohlo přispět celkové přehlednosti



Obrázek 4.1. Návrh souhrnné verze GUI

4.3.2 Průvodcovské GUI

Návrh označený jako *průvodcovský* ve své podstatě vychází z různých softwarových průvodců, uživatele krok za krokem provádí jednotlivými částmi a nastaveními práce s FreeSurferem.

U průvodcovského návrhu existuje v jistém směru ideologický problém, neboť softwarový průvodce by měl uživatele provádět nějakou konkrétní činností, zatímco možná činnost s GUI pro FreeSurfer není jen jedna. V návrhu je toto řešeno prostě, jako jeden z prvních kroků si uživatel vybírá, co by vlastně chtěl dělat. Nedostatek tohoto řešení ovšem spočívá v omezeném prostoru pro nabídku činností, tedy kdyby se mělo množství činností realizovatelných pomocí GUI zvětšit, musela by se minimálně v tomto konkrétním kroku zásadnějším způsobem měnit obrazovka GUI.

Na rozdíl od souhrnného GUI zde navíc uživatel nevidí všechny vyplněné údaje, a pokud si např. chce ověřit, že má správně zadanou e-mailovou adresu (ať už kvůli náhlému záchvatu nejistoty, ke kterému občas dochází i u pokročilejších uživatelů, nebo jen kvůli tomu, že se údaje nahrávaly ze souboru `jobstatus.mat`, musí se vracet přes jednotlivé kroky zpět a pak zase vpřed, neboť z dřívějších kroků není možné potvrdit akci.

GUI by ovšem mělo být intuitivní a pochopitelné i pro uživatele, který s ním zatím nikdy nepracoval, nemá prostudovanou dokumentaci ani případné jiné návody.

Obrázek 4.2 ukazuje základní podobu tohoto GUI, obrázek 4.3 výběr činnosti a obrázek 4.4 pak zahájení kopírování.

Testování kognitivním průchodem

1) Průřezové zpracování dat

Referenční průchod je následující:

1. Vybrat možnost Local SGE
2. Vybrat možnost Copy data
3. Vyplnit e-mailovou adresu
4. Vybrat typ dat
5. Vybrat zdrojový soubor `.xls`
6. Vybrat pracovní adresář
7. Vybrat datový adresář
8. Kliknout na Next

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO[1]	ANO	ANO
2	NE[2]	ANO	ANO
3	ANO	ANO	ANO
4	ANO	ANO	ANO
5	ANO	ANO	ANO
6	NE[3]	ANO	ANO
7	ANO	ANO	NE[4]
8	ANO	ANO	ANO

1. *SGE* není univerzálně známý pojem
2. Uživateli nemusí být jasné, že musí data nejprve zkopírovat
3. Není jasné, že má uživatel zvolit právě pracovní adresář (je trochu diskutabilní, zda tento nález patří spíše k první, nebo ke druhé otázce)

4. Zvolená cesta se uživateli nikde nezobrazí

2) Extrakce dat

Referenční průchod je následující:

1. Vybrat možnost Local SGE
2. Vybrat možnost Extract data
3. Kliknout na možnost Load job status
4. Nahrát správný soubor `jobstatus.mat`
5. Vybrat PVC/No PVC
6. Vybrat Volume/Thickness
7. Kliknout na Next

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO	ANO	ANO
2	ANO	ANO	ANO
3	ANO	ANO	ANO
4	ANO	ANO	ANO
5	ANO	ANO	ANO
6	ANO	ANO	ANO
7	ANO	ANO	ANO

3) Srovnávací longitudinální zpracování dat

Referenční průchod je následující:

1. Kliknout na Back
2. Kliknout na Back
3. Vybrat možnost Longitudinal-Long job
4. Kliknout na Next

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	NE[1]	ANO	ANO
2	ANO	ANO	ANO
3	ANO	ANO	ANO
4	ANO	ANO	ANO

1. Uživatel nemá důvod očekávat, že při pokračování ve zpracování se musí vracet

Testování heuristickou evaluací

- Viditelnost stavu systému
  - V GUI není nikde vidět, když dojde např. k chybě při vytváření pracovního adresáře
  - GUI neinformuje, kterou činnost uživatel právě nastavuje
- Shoda mezi systémem a skutečným světem

  Bez nálezů
- Uživatelské ovládání a uživatelská svoboda

  - Uživatel nemá jednoduchou možnost vrátit se do stavu po spuštění GUI
- Konsistence a standardy

  Bez nálezů
- Předcházení chyb
  - GUI umožňuje provedení činností, které ve skutečnosti potřebují, aby bylo nejprve provedeno něco jiného
- Preference rozpoznání před rozpomínáním se
  - Dříve zadané možnosti se uživateli nikde nezobrazují, uživatel si je tedy musí pamatovat
- Flexibilita a efektivita užití
  - Není možné jednoduše pokračovat ve zpracování po dokončení nějaké činnosti
- Estetika a minimalistický design

  Bez nálezů
- Pomoc uživateli rozpoznat chyby, diagnostikovat je a zotavit se z nich
  - Návrh GUI nepočítá s oznamováním jiných chyb než těch, ke kterým dojde v rámci zpracování dat prostřednictvím FreeSurferu
- Pomoc a dokumentace
  - Uživatel nemá z GUI možnost zjistit, kde najde podrobnější dokumentaci

Shrnutí výhod návrhu

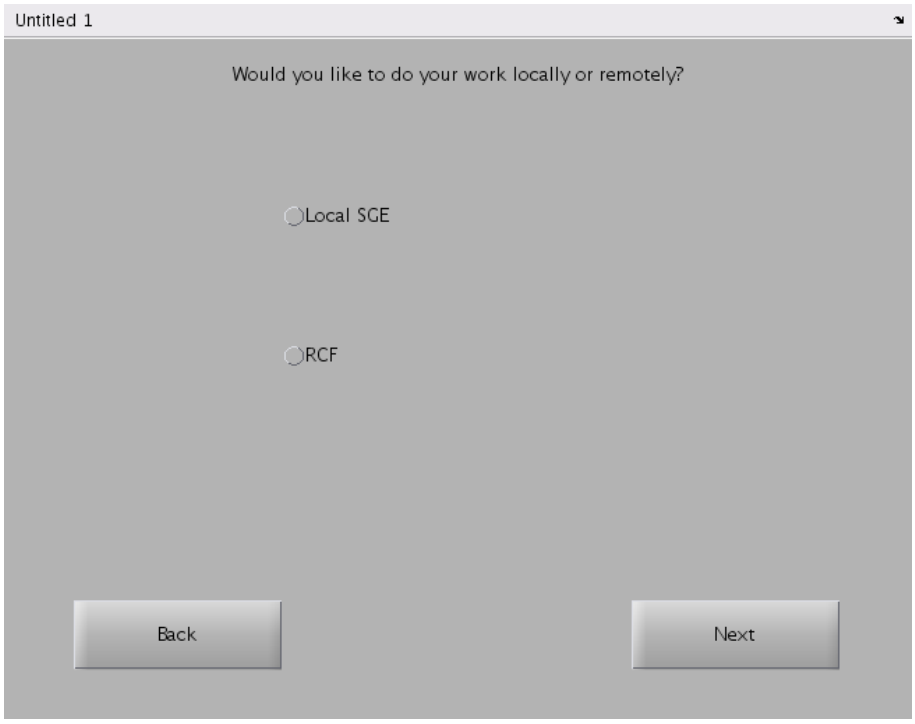
- Použitelné i bez předchozího seznámení s tímto GUI

Shrnutí nevýhod návrhu

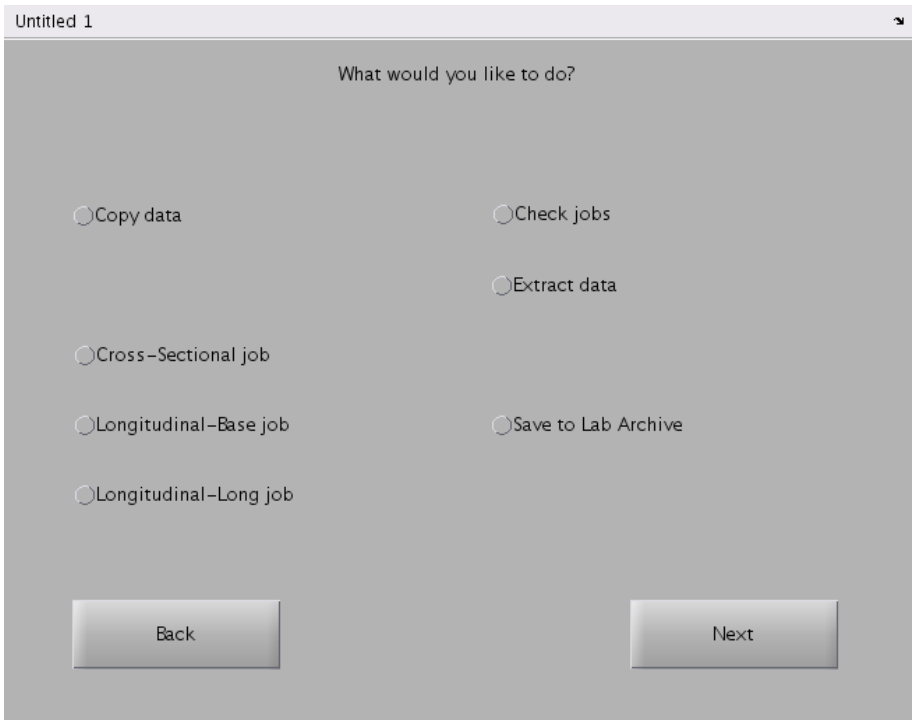
- Nutnost měnit zobrazení při rozšíření funkcionality
- Potřeba přecházet mezi kroky při kontrole údajů

Prostor pro vylepšení

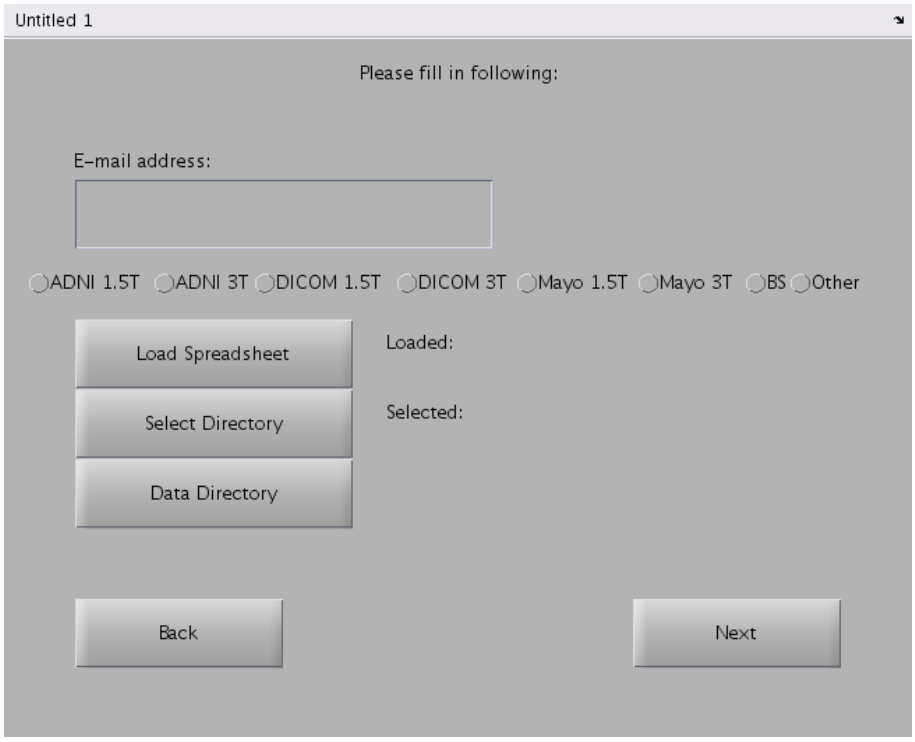
- Zobrazování údajů z dřívějších kroků by mohlo usnadnit jejich kontrolu
- Umožnit potvrzení akce z libovolného kroku by umožnilo nevracet se po kontrole zpět



Obrázek 4.2. Návrh průvodcovské verze GUI, úvodní obrazovka



Obrázek 4.3. Návrh průvodcovské verze GUI, výběr činnosti na lokálním počítači



Obrázek 4.4. Návrh průvodcovské verze GUI, kopírování dat

4.3.3 Kontextové GUI

Kontextový návrh GUI se snaží být zlatou střední cestou. Podle zaškrtnutých možností se mění zobrazované ovládací prvky, uživatel může vyplňovat pouze informace, které jsou relevantní pro danou činnost. Pro kontrolu nebo změnu údajů pomyslného dřívějšího kroku ovšem není nutné fakticky se vracet zpět, údaje jsou stále na obrazovce, dostupné pro čtení i úpravu, umožňuje-li to aktuálně zvolená činnost.

Podobně jako u průvodcovského návrhu by v případě rozšíření funkcionality GUI bylo potřeba významnějším způsobem měnit zobrazení.

Obrázek 4.5 ukazuje zadání nového zpracování v tomto návrhu, obrázek 4.6 pak pokračování v dříve rozpracovaném zpracování.

Testování kognitivním průchodem

1) Průřezové zpracování dat

Referenční průchod je následující:

1. Vybrat možnost New job
2. Nahrát soubor `.xls`
3. Vybrat pracovní adresář
4. Vybrat možnost Local SGE
5. Zadat e-mailovou adresu
6. Vybrat typ dat
7. Vybrat adresář s daty
8. Kliknout na Copy data

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO	ANO	ANO
2	ANO	ANO	ANO
3	NE[1]	ANO	ANO
4	ANO	ANO	ANO[2]
5	ANO	ANO	ANO
6	ANO	ANO	ANO
7	ANO	ANO	NE[3]
8	ANO	ANO	ANO

1. Není jasné, že má uživatel zvolit právě pracovní adresář (je trochu diskutabilní, zda tento nález patří spíše k první, nebo ke druhé otázce)
2. Pro uživatele může být matoucí, že se po zvolení možnosti nic nestane
3. Cesta ke zvolenému adresáři se uživateli nikde nezobrazí

2) Extrakce dat

Referenční průchod je následující:

1. Continue job
2. Kliknout na Load job status
3. Nahrát správný soubor `jobstatus.mat`
4. Vybrat možnost Extract Data
5. Vybrat PVC/No PVC
6. Vybrat Volume/Thickness
7. Kliknout na Extract

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO	ANO	ANO
2	ANO	ANO	ANO
3	ANO	ANO	ANO
4	ANO	ANO	ANO
5	ANO	ANO	ANO
6	ANO	ANO	ANO
7	ANO	ANO	ANO

3) Srovnávací longitudinální zpracování dat

Referenční průchod je následující:

1. Vybrat Longitudinal-Long
2. Kliknout na Run

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO	ANO	ANO
2	ANO	ANO	ANO

Testování heuristickou evaluací

- Viditelnost stavu systému
  - V GUI není nikde vidět, když dojde např. k chybě při vytváření pracovního adresáře
- Shoda mezi systémem a skutečným světem

  Bez nálezů
- Uživatelské ovládání a uživatelská svoboda
  - Není možné zpracování jednoduše přerušit (např. spustí-li ho uživatel omylem)
- Konsistence a standardy
  - Slovo „job“ se používá ve dvou významech, jednak pro celé zpracování, jednak pro dílčí zpracování jednotlivých dat
- Předcházení chyb

  Bez nálezů
- Preference rozpoznání před rozpomínáním se

  Bez nálezů
- Flexibilita a efektivita užití

  Bez nálezů
- Estetika a minimalistický design

  Bez nálezů
- Pomoc uživateli rozpoznat chyby, diagnostikovat je a zotavit se z nich

- Návrh GUI nepočítá s oznamováním jiných chyb než těch, ke kterým dojde v rámci zpracování dat prostřednictvím FreeSurferu

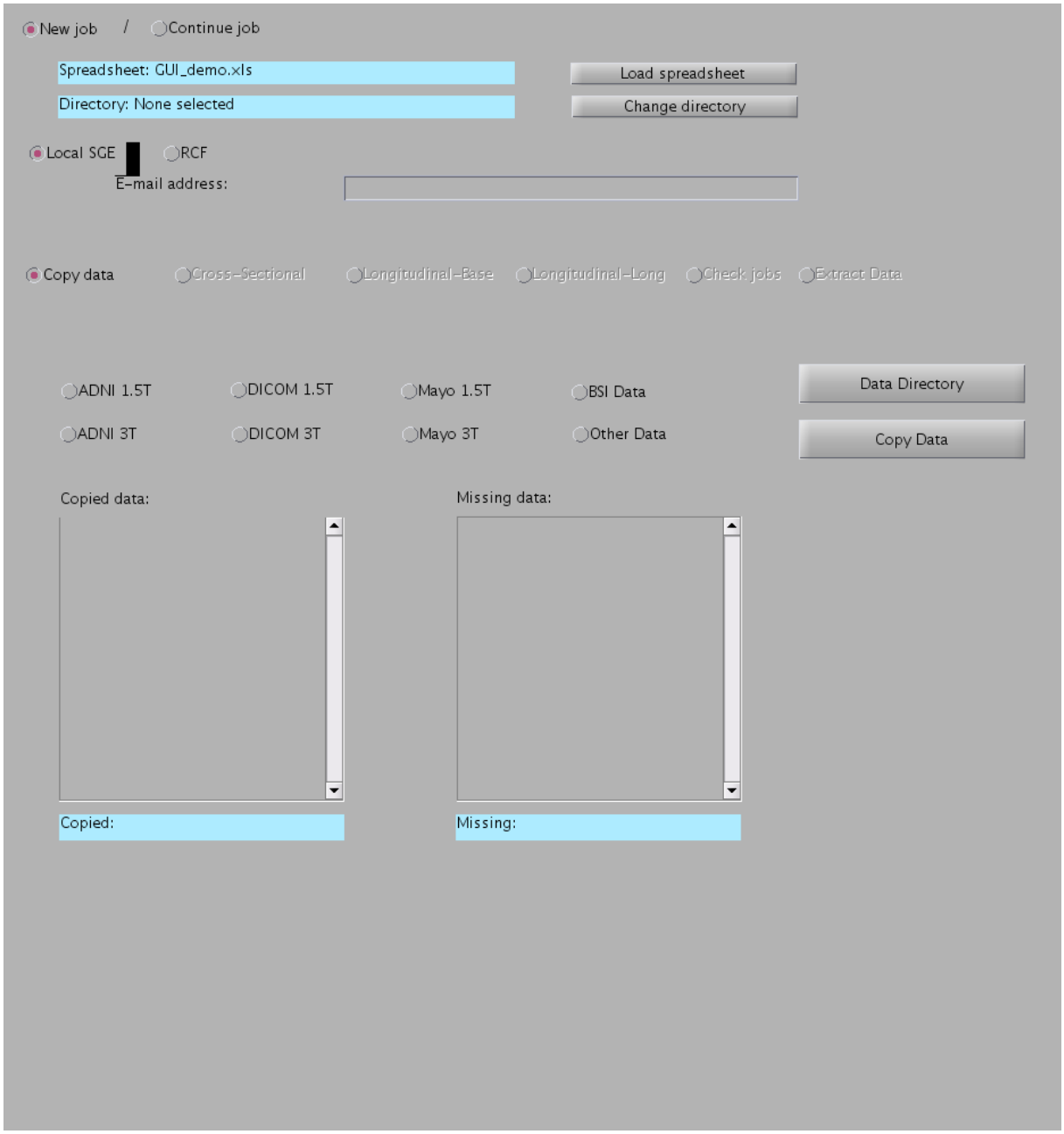
- Pomoc a dokumentace
  - Uživatel nemá z GUI možnost zjistit, kde najde podrobnější dokumentaci

Shrnutí výhod návrhu

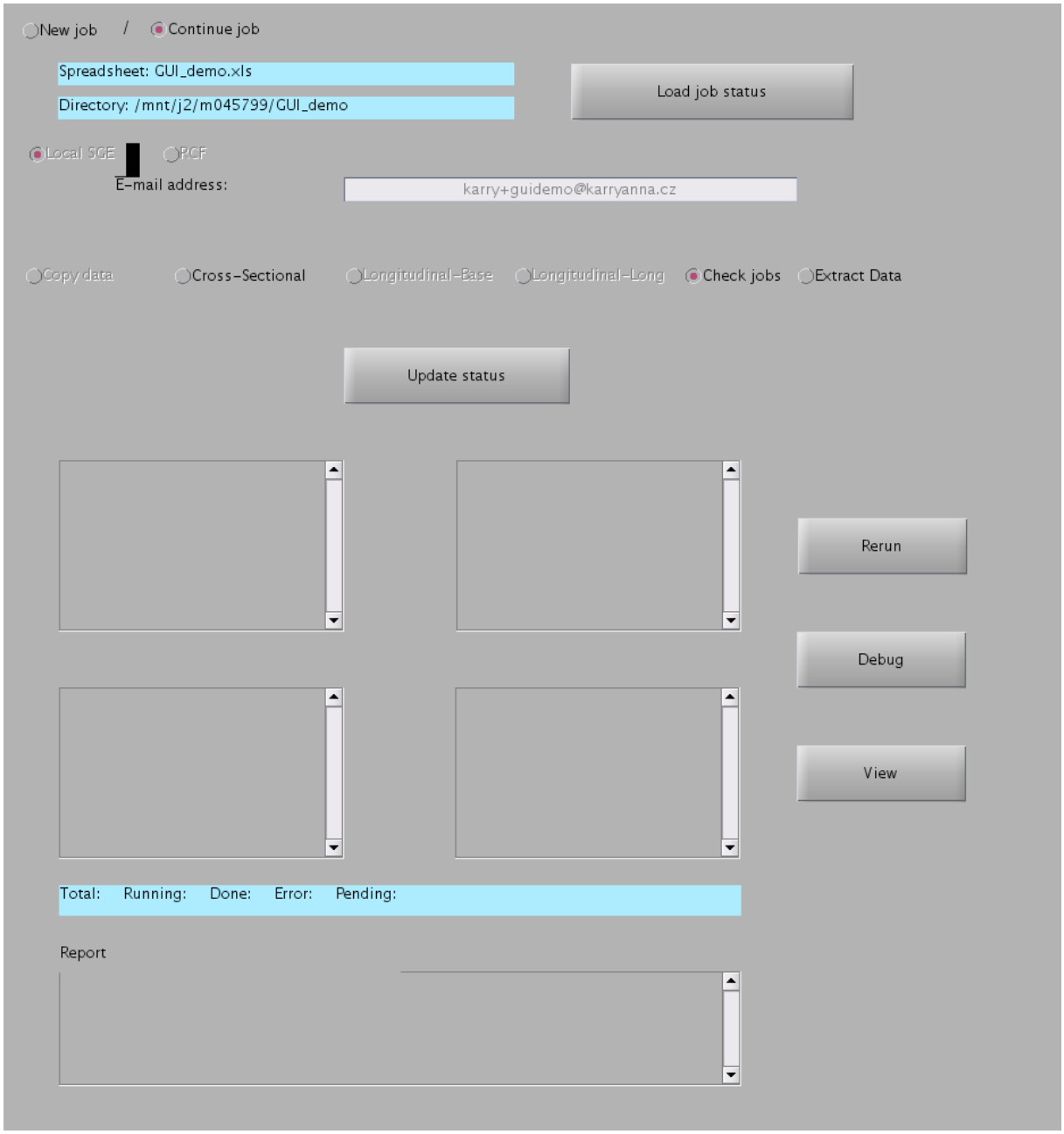
- V každém okamžiku jasné, co se zrovna děje

Shrnutí nevýhod návrhu

- Nutnost měnit zobrazení při rozšíření funkcionality



Obrázek 4.5. Návrh kontextového GUI



Obrázek 4.6. Návrh kontextového GUI

Kapitola 5
Realizovaný návrh

5.1 Charakteristika

Výsledné GUI bylo tvořeno pro FreeSurfer verze 5.3, ale není známý důvod, proč by nemělo běžet i nad jinou verzí FreeSurferu. Na verzi 5.3 jsme se domluvili zejména proto, že v ní jsou opravené některé chyby. Měnit používanou verzi FreeSurferu není tak snadné, jak by se mohlo zdát, neboť např. podle [9] verze FreeSurferu významně ovlivňuje získaná data, tedy se hodí neměnit verzi až do ukončení výzkumu.

Realizované GUI vychází zejména z kontextového návrhu a z jasného zřetězení potenciálně prováděných akcí. Uživateli se celou dobu zobrazují informace o tom, odkud byla vzata data, kde probíhá zpracování, a také o jeho vlastní identifikaci.

Podle aktuálně prováděné činnosti a činností doposud provedených GUI umožňuje a znemožňuje (aktivováním a deaktivováním příslušných políček a možnosti) výběr jednotlivých akcí, úpravu konkrétních vstupů atp, celé tedy do nějaké míry funguje stavově.

Oproti původním návrhům zmizely některé možnosti, tato zmizení by ale měla být pouze dočasná. Ukázalo se totiž, že implementování veškeré původně navržené funkcionality programu představuje příliš rozsáhlý úkol, který nebylo možné zvládnout v čase určeném pro realizaci bakalářské práce (rozhodně ne spolu s dopisováním dokumentace a průvodního textu), proto byla u jednotlivých činností stanovena priorita a zprovozněny primárně činnosti s vyšší prioritou.

Z konzultací vyplynulo zejména to, že možnost vzdáleného (remote) zpracování není tak důležitá jako možnost lokálního, proto bylo vzdálené zpracování vynecháno, stejně jako všechny přímo s ním související možnosti.

Vynecháno bylo zatím i kompletní nastavení extrakce dat. FreeSurfer má nástroje, které umožňují data extrahovat, a při vhodném nastavení umí extrahovat pouze zadanou podmnožinu získaných dat. V případě nutnosti si ovšem vybrat potřebná data ze všech mohou i sami lékaři, GUI jim tedy zatím umožňuje jen extrahovat všechna získaná data.

Mým cílem ovšem je, aby bylo alespoň základní nastavení extrakce doimplementováno v blízké době.

5.2 Implementace

Tato část práce si neklade za cíl nahradit podrobnou dokumentaci každého použitého triku a každého potenciálního problému, k tomu slouží komentáře přímo ve zdrojovém kódu. Následující text by měl zejména poskytnout ucelenou představu o způsobu, jakým bylo GUI implementováno, a podrobněji popsat některé řešené problémy.

V implementaci mi hodně pomáhaly Mathworks [10], Undocumented Matlab [11] a „programátorská klasika“ [12], občas jsem nahlédla do [13] a [14].

Jak již bylo v práci několikrát zmíněno, GUI je implementované v Matlabu. Tvoří ho několik `.m` souborů, hlavním souborem je soubor `freesurfer_gui.m`.

V tomto souboru je zavedena hlavní funkce `freesurfer_gui`, která zajišťuje sestavení a vykreslení GUI, v rámci této funkce je pak definováno mnoho pomocných funkcí. Pomocné funkce se starají o vykreslování jednotlivých částí GUI, o povolení či zablokování určitých prvků a funkcionality (čímž se zajišťuje dodržení zamýšleného průběhu zpracování), o správné reakce na uživatelské akce atp.

V dalších souborech jsou implementované jednak funkce zajišťující samotné zpracování (zejména tedy zpracování pro průřezovou studii a zpracování pro longitudinální studii), jednak pomocné funkce, ať již pomocné čistě z hlediska implementovaného GUI (načtení excelovského listu s daty o pacientech), nebo z obecnějšího hlediska funkcionality kódu (převod dat mezi formátem pro Microsoft Excel a pro Matlab).

GUI je stavové, jednotlivé stavy reprezentují průchod zamýšleným postupem zpracování a ovlivňují to, které činnosti jsou momentálně pro uživatele dostupné. Podrobněji jsou stavy popsány později v této sekci.

Proměnné GUI jsou typicky prefixované `fs_`, předpokládá se, že takovéto označení zajistí nekonfliktnost. Ta je potřeba zejména při komunikaci GUI s funkcemi řídícími zpracování; podrobnosti této komunikace jsou popsány v této sekci v části o obcházení vláken.

Všechna data důležitá pro zpracování jsou uložena ve struktuře `fs_data`. V dalších proměnných prefixovaných `fs_` jsou uloženy některé údaje důležité pro funkčnost samotného GUI, např. cesta k FreeSurferu. Je nicméně možné, že tyto proměnné budou v blízké době sloučené do struktury (zřejmě označené `fs_config`).

Zamýšlený postup zpracování a jednotlivé stavy GUI

Na základě poskytnutých materiálů a rozhovorů s lékaři jsem sestavila níže popsaný zamýšlený postup zpracování. Tento postup samozřejmě respektuje nutné podmínky plynoucí z vlastností softwaru a používaných algoritmů, ale zároveň se snaží co nejvíce respektovat přání vyslovená lékaři, ať již během úvodních konzultací, nebo během průběžného testování.

1. Zkopírování a převedení dat

   Spolupracujícími lékaři jsem byla požádána, aby zpracování neprobíhalo přímo na originálních souborech, nýbrž na jejich kopiích, proto se jako první krok data kopírují do pracovního adresáře, kde se dál zpracovávají tyto nově vytvořené kopie.

   Data z MRI se navíc v různých nemocnicích získávají v různých formátech, přičemž FreeSurfer umí zpracovat pouze některé konkrétní. Proto se data po zkopírování ještě převádějí do vhodného formátu.

   Do budoucna by jistě bylo dobré udělat tento krok nepovinným, aby uživatelé, kteří chtějí pracovat s původními daty, případně si raději data kopírují prostřednictvím jiných nástrojů, tuto možnost měli.

   Za úvahu by stálo i rozdělení tohoto kroku na dva. Takové rozdělení by jednak snížilo množství problémů vznikajících, pokud dojde k problému až při samotném převodu formátu, jednak by ušetřilo čas uživatelům, kteří potřebují provést skutečně jen jednu z těchto dvou činností.

2. Provedení průřezového zpracování

   Převedená data by se dále měla zpracovat metodami průřezového zpracování, jehož výsledkem jsou přímo data o objemu mozkové hmoty atp.

3. Provedení základního longitudinálního zpracování

   V případě, že bylo pro jednotlivé pacienty provedeno více měření, mohou být pomocí metod pro longitudinální zpracování nyní vytvořena data pro časové srovnávání.

4. Provedení srovnávacího longitudinálního zpracování

Po vytvoření základních dat pro srovnání může být provedeno samotné srovnávací zpracování pro všechna jednotlivá měření.

Kromě těchto jednoduše definovaných kroků existují další činnosti, které může mít uživatel zájem provést, a které nejsou jednoznačně zařaditelné do průchodu zpracování, neboť sice mají smysl až po dokončení nějakého konkrétního kroku, ale nic dalšího na nich přímo nezávisí, tedy mohou být provedeny i kdykoli později.

Momentálně do těchto činností patří následující:

- Visualizace dat
  Po provedení průřezového zpracování může mít uživatel zájem nechat si zobrazit získané výsledky a provést optickou kontrolu, zda získaná data dávají smysl, nebo zda nejspíš došlo k chybě (a případně i určit, k jaké chybě typově došlo).
- Extrakce dat
  Po provedení průřezového zpracování může uživatel také chtít extrahovat získaná data do podoby vhodné pro další, zejména statistické, zpracování.

GUI je stavové, jednotlivé stavy odpovídají jednotlivým krokům v rámci zamýšleného postupu zpracování (resp. tomu, že dané kroky jsou poslední zatím dokončené). Ke každému kroku patří dva stavy, jeden reprezentující situaci, kdy je daný krok rozpracovaný, druhý reprezentující situaci, kdy je dokončený. V současné době umí GUI správně navázat na dřívější zpracování pouze při přerušení mezi jednotlivými kroky, ale do budoucna je plánována možnost přerušit zpracování i v průběhu jednotlivých kroků.

Konkrétně tedy v současné době stavy GUI vypadají takto:

0	Základní nastavování
1	Spuštěné kopírování
2	Dokončené kopírování a převod dat
3	Spouštěné průřezové zpracování
4	Dokončené průřezové zpracování
5	Spuštěné základní longitudinální zpracování
6	Dokončené základní longitudinální zpracování
7	Spuštěné srovnávací longitudinální zpracování
8	Dokončené srovnávací longitudinální zpracování

Je ovšem třeba mít na paměti, že současný návrh stavů počítá s lineárním průběhem zpracování. Již teď ho trochu komplikují visualizace a extrakce dat, které do lineárního průběhu úplně nezapadají. Pro ty ovšem stačí definovat, po kterém kroku již mají smysl; nic dalšího na nich nezávisí.

Kdyby se ale funkcionalita GUI rozrostla do té míry, že by umožnila reálné větvení zpracování, a tedy by měl uživatel na výběr z více akcí, na které by třeba dál navazovaly různé možnosti, současný systém stavů by se mohl stát nevhodným.

Pojmenovávací systém funkcí a objektů v GUI

Jednotlivých funkcí a objektů definovaných v GUI je mnoho, a je proto nutné zavést nějaké konvence pro jejich pojmenovávání. Z těchto konvencí by mělo být rychle patrné, ke které části GUI tento objekt patří a co v GUI zajišťuje.

Identifikátory proto dodržují tvar `gui_část_typ_jméno`, kde `gui` je univerzální prefix naznačující, že jde o funkci či objekt existující pro potřeby GUI, `část` je kódové označení části GUI, ke které jsou daná funkce či objekt vztažené (to může být buď optická část

GUI, nebo část odpovídající nastavení nějakého konkrétního kroku), `typ` je typ funkce či objektu a `jméno` je přímo pojmenování, které nějakým způsobem odpovídá funkci či očekávaným datům.

Jak už jsem naznačila, části GUI jsou jednak části zejména optické (tedy ty, které odpovídají nějaké oblasti zobrazeného okna), jednak části reprezentující nějakým způsobem jednotlivé kroky (jednotlivé činnosti).

Optické části GUI jsou

- obecná část, označovaná v identifikátorech `general` – v této části se vybírá mezi novým zpracováním a pokračováním ve starém, spadá sem oblast pro oznamování chyb a také informace o GUI odkazující na uživatelskou dokumentaci; v blízké době by sem měla patřit i možnost přerušit zpracování
- nastavení, v identifikátorech `settings` – zde se nastavují všechny základní údaje, zejména tedy zdrojový soubor, pracovní adresář a adresář se snímky určenými ke zpracování, dále v současné době identifikace uživatele a jeho e-mail
- činnost, v identifikátorech `action` – v současné době tuto část představují jen dva řádky s výběrem činnosti, není v plánu tuto část výrazně rozšiřovat, resp. případné rozšíření by mělo být dáno pouze případným rozšířením funkcionality GUI o jinou logickou činnost
- nastavení činnosti, v identifikátorech označované označením dané činnosti

Možné činnosti se pak v současné době skládají z následujících:

- kopírování dat, označované `copy`
- převedení dat do formátu vhodného pro FreeSurfer, označované `convert` – trochu problematická činnost z toho hlediska, že z pohledu uživatele je přímou součástní kopírování dat, ale z pohledu GUI je od něj oddělená
- zpracování pro průřezovou studii, označované `cs`
- základní zpracování pro longitudinální studii, označované `longbase`
- srovnání dvou časových bodů pro longitudinální studii, označované `longlong`
- kontrola stavu zpracování, označovaná `check`
- visualizace zpracovaných dat, označovaná `visualize`
- extrakce získaných dat, označovaná `extract`

Jako typy funkce či objektu se v současné době používají tyto:

- `input`: uživatelský vstup, nikoli pouze ve smyslu editovatelného pole; typ `input` má vše, co uživatel nějakým způsobem zadává, tedy kromě zjevných editovatelných polí také různá rozbalovací menu či spouštěcí tlačítka
  Pozor: Poněkud neintuitivně tak nejsou označena přepínací políčka, ta mají místo `input` typ `groupinput` a jsou více popsaná u typu `group`
- `label`: informační popisky pro uživatele, typicky popisky editovatelných polí
- `info`: informace pro uživatele, které ovšem neslouží jako popis nějakého uživatelského vstupu – tento typ se používá např. pro textová pole vypisující zvolené adresáře či pro seznamy správně a chybně zpracovaných dat
- `callback`: funkce reagující na událost, její `jméno` je stejné jako `jméno` objektu, na kterém dochází k události vyvolávající tuto funkci
- `group`: pomocný objekt, skupina pro jednotlivé možnosti přepínacích tlačítek; identifikátory těchto tlačítek pak dodržují tvar `gui_část_groupinput_skupina_jméno`, kde `skupina` je `jméno` v identifikátoru příslušné skupiny
- `hide`: funkce skrývající některé prvky GUI

- `show`: funkce zobrazující některé prvky GUI
- `disable`: funkce blokující úpravu/vyvolání některých prvků GUI
- `enable`: funkce povolující úpravu/vyvolání některých prvků GUI
- `preset`: funkce přednastavující hodnoty některých uživatelských vstupů či informačních výstupů GUI; typicky může být volána při obnovení zpracování
- `timer`: funkce volaná při sepnutí časovače

Obcházení vláken

Jak jsem popisovala již v sekci 4.1, jednou z nutných vlastností GUI jsou rychlé reakce. Zejména je naprosto nepřijatelné, aby se GUI během výpočtu na několik hodin zaseklo a až do dokončení výpočtu nereagovalo na vstup od uživatele.

Potřeba rychlých reakcí GUI i během nějakého výpočtu se obvykle řeší tak, že jak GUI, tak samotný výpočet běží ve vlastním vlákně. Bohužel, Matlab bez speciálních rozšíření pro paralelní výpočty spuštění v samostatném vlákně neumožňuje (a to je něco, co jsem zřejmě měla vzít v potaz již v počátku, když jsme teprve řešili, zda GUI půjde udělat a jak by se toho mělo docílit).

Celý problém komplikuje fakt, že je žádoucí číst výstup a návratovou hodnotu spuštěných výpočtů. Striktně vzato to není nutné, neboť skripty řídící výpočet zapisují záznam o svém průběhu do `.log` souboru, přesto než velice často přistupovat k souboru a parsovat jeho obsah považuji za vhodnější nalezení způsobu, jak opravdu číst návratovou hodnotu.

Matlab totiž umí spustit program na pozadí, tedy bez zablokování hlavního vlákna (vlákna, v kterém běží GUI), ale neumí v takovém případě ani indikovat, že na pozadí spuštěný program již skončil, ani zpřístupnit právě jeho návratovou hodnotu a výstup.

Po prohledávání [10], [11] a různých internetových poraden a diskusí jsem zjistila, že obvyklé řešení spočívá v přistoupení k nižší vrstvě a přímém využití Javy, která je jinak volaná prostřednictvím Matlabu.

S využitím Javy, resp. třídy `Runtime` a její metody `exec`, je možné spustit samostatný proces a po jeho skončení přistupovat jak k jeho návratové hodnotě, tak k jeho standardnímu a chybovému výstupu.

Zřejmě by bylo možné vytvořit i instanci třídy `Thread`, která umožňuje lepší komunikaci mezi jednotlivými vlákny (zejména např. ve věci oznamování, že dané vlákno již ukončilo svou činnost) a využít tu, ale podle mého průzkumu se zdá, že běžným zvykem v prostředí Matlabu je právě využití tříd `Runtime` a `Process`, proto jsem je zvolila i já.

Problém tohoto přístupu spočívá ve faktu, že neexistuje způsob, jak původnímu vláknu oznámit, že proces již skončil. Proto je nutné se z původního vlákna periodicky dotazovat na návratovou hodnotu procesu a ošetřovat případnou výjimku vyvolanou tím, že proces ještě nedoběhl.

V mém návrhu GUI předává funkci seznam výsledků (balíků pořízených snímků) ke zpracování, a zpracovávající funkce sama pro jednotlivé výsledky volá zpracování. Ve zpracovávající funkci se proto vytvářejí nové procesy a kontroluje, zda původní proces už doběhl, navíc je třeba z GUI pravidelně zjišťovat, zda už doběhlo dílčí zpracování, neboli zda se má překreslit aktuální stav.

> Zde si dovolím malou vsuvku. Je nejspíš diskutabilní, zda by funkce řídící zpracování měla spíše přijímat nějakou identifikaci jedněch konkrétních dat a pouze ta zpracovávat, nebo zda by měla přijímat takto seznam a řešit nějaké dělení a postupné zpracování.

> Z principu, že funkce má být co možná nejjednodušší, by plynula spíše první možnost, naopak druhá více odpovídá pohledu uživatele. Možná by nakonec nejlepší bylo poskytovat funkce dvě, jednu, která přímo zpracovává konkrétní data, a druhou, která je nad ní nadstavbou, umí dat přijmout více a postupně na ně volá onu první funkci.
>
> Je možné, že taková úprava bude později provedena, ale zatím jsem se řídila právě pohledem uživatele, tedy funkce si bere seznam dat ke zpracování.

Při zpracování se proto používají dva objekty typu `Timer` (časovač). Jeden z nich je definovaný ve funkci pro zpracování, v něm se přímo tvoří procesy a kontroluje se, zda poslední proces již doběhl, spolu s tím se v něm řeší právě postupné zpracovávání více dat.

Druhý je definovaný v GUI jako takovém, tomuto se při zahájení zpracování nastaví perioda a funkce k vykonání. V rámci této funkce se následně kontroluje, zda dané zpracování již skončilo, nebo ne, případně zda skončila nějaká jeho část (a tedy je potřeba aktualizovat nějaké informace zobrazované v GUI).

Předávání této informace je realizované pomocí vyhodnocování proměnných v základním pracovním (jmenném) prostoru, ke kterému mají jak GUI, tak zpracovávající funkce přístup. V tomto prostoru se vyhodnocuje jednak proměnná `fs_processing` nesoucí informaci o tom, zda zrovna probíhá nějaké zpracování, jednak proměnná `fs_trasnp` obsahující všechna případně předávaná data.

Spouštění jednotlivých procesů a zjišťování, zda už proces skončil, je blíže naznačeno v šabloně funkcí volaných pro zpracování, zde ještě ukážu vzory funkcí implementovaných v GUI.

Funkce zahajující zpracování pro nějakou konkrétní akci je implementována podle následujícího vzoru:

```
 1 function gui_action_callback_start(source, eventdata)
 2     uni_timer.TimerFcn = {@gui_action_timer_function};
 3     uni_timer.period = ___;
 4
 5     fs_data.status = ___;
 6
 7     % Povolení a zakázání kroků nebo nastavení
 8
 9     assignin('base', 'fs_processing', 1);
10
11     % Příprava parametrů pro volanou akci
12     perform_action(fs_vars, fs_data.subjects, analysis_data);
13
14     start(uni_timer);
15 end
```

Na druhém řádku se nastavuje funkce, která se vždy při sepnutí časovače provede, na třetím řádku pak perioda, se kterou bude časovač spínat. Ta je (nelineárním způsobem) přizpůsobena očekávané době zpracování, např. pro kopírování dat je nastavena na 0.5 vteřiny, pro zpracování pro průřezovou studii na 5 vteřin (a pravděpodobně by mohla být nastavena i na delší dobu).

V rámci povolení a zakázání kroků nebo nastavení se obvykle zakazuje měnit nastavení k právě zahájené akci (voláním funkce `gui_action_disable_options()`), může být umožněno sledování průběhu zpracování atp.

Funkce při sepnutí časovače se pak řídí následujícím vzorem:

```
function gui_action_timer_function(~{}, event)
    processing = evalin('base', 'fs_processing');

    if (processing == 0) || (processing == 2)
        fs_transp = evalin('base', 'fs_transp');
        % Zpracování předávaných údajů

        % Aktualizace zobrazení v GUI
    end

    if (processing == 0)
        stop(uni_timer);
        gui_action_callback_finish();
    end
end
```

Hodnota proměnné `fs_processing` rovna 2 znamená, že existují data, která by měla být předaná do GUI; hodnota 0 pak znamená, že celé zpracování již doběhlo.

Předávaná data se vyhodnocují i v případě, že zpracování skončilo, neboť v opačném případě by bylo potřeba implementovat ještě mechanismus bránící situaci, kdy se zapíšou data k předání, ale před jejich vyhodnocením celé zpracování skončí.

Vzor pro funkci ukončující celé zpracování vypadá následovně:

```
function gui_action_callback_finish(source, eventdata)
    % Povolení a zakázání kroků nebo nastavení

    fs_data.status = ___;

    update_jobstatus();
end
```

Podobně jako při zahájení zpracování je obvyklé zakázat nastavení pro danou akci, při dokončení se již zakazuje vše, zejména tedy možnost zpracování opět pustit (k tomu slouží funkce `gui_action_disable_all()`). Dále se v tomto místě typicky povoluje další krok (existuje-li nějaký, resp. existuje-li nějaký, který dává smysl se zpracovávanými daty). Například při dokončení zpracování dat pro průřezovou studii se podle množství časových bodů v datech může povolit zpracování dat pro longitudinální studii.

Struktura fs_data a soubor jobstatus.mat

Všechna data pro zpracování jsou uložena ve struktuře `fs_data`, která je při spuštění GUI inicializována takto:

```
fs_data = struct( ...
    'spreadsheet', '', ...
    'wdir', '', ...
    'ddir', '', ...
    'odir', pwd, ...
...
    'email', '', ...
    'uid', '', ...
...
    'status', 0, ...
    'working', 0, ...
    'action', 'none', ...
```

```
13 ...
14     'datatype', 1, ...
15     'datatype_name', '', ...
16 ...
17     'cs_recon', '1', ...
18     'cs_notal', 1, ...
19     'cs_norunning', 1, ...
20     'cs_recon_done', 0, ...
21     'cs_recon_ip', 0, ...
22 ...
23     'longbase_notal', 1, ...
24     'longbase_norunning', 1, ...
25 ...
26     'longlong_notal', 1, ...
27     'longlong_norunning', 1, ...
28 ...
29     'num_missing', 0, ...
30     'num_copied', 0, ...
31     'num_converted', 0, ...
32     'num_notconverted', 0, ...
33     'num_pending', 0, ...
34     'num_running', 0, ...
35     'num_done', 0, ...
36     'num_error', 0, ...
37 ...
38     'missing', {{}}, ...
39     'copied', {{}}, ...
40     'converted', {{}}, ...
41     'notconverted', {{}}, ...
42     'pending', {{}}, ...
43     'running', {{}}, ...
44     'done', {{}}, ...
45     'error', {{}}, ...
46     'omitted', {{}}, ...
47     'visualizable', {{}}, ...
48     'extractable', {{}} ...
49     );
```

Položky `spreadsheet`, `wdir` a `ddir` odpovídají po řadě zdrojovému souboru dat, pracovnímu adresáři a zdrojovému adresáři dat. Znát pracovní adresář je nezbytné v celém průběhu zpracování, zdrojový soubor a datový adresář jsou z hlediska zpracování potřeba pouze při úvodním kopírování, ovšem je nutné mít je někde uložené, aby bylo možné tyto údaje zobrazit uživateli.

V položce `odir` je uložený původní adresář, ze kterého bylo GUI spuštěno. Během zpracování se totiž přepíná mezi tímto a pracovním adresářem, a je proto nutné vědět, kam se má přepnout zpět (mimo jiné proto, že uživatel nutně nemusí mít GUI nainstalované na cestě globálně dostupné z Matlabu).

Položka `status` obsahuje stav GUI, tedy informaci o posledním rozpracovaném/dokončeném kroku.

`Working` je příznak, zda aktuálně probíhá nějaké zpracování. V současné době se s ním v GUI dále nepracuje, ale je zamýšlené jako doplňující ochranný prvek k samotnému povolování a blokování spouštěcích tlačítek, aby nebylo možné spustit více kroků najednou (to by mohlo vést k nekonsistenci údajů zobrazovaných uživateli).

V položce `action` je uloženo označení právě aktivní činnosti, tedy té, pro kterou se zrovna zobrazuje nastavení a další informace. Hlavní důvod pro ukládání této informace je možnost při pokračování zpracování rovnou uživateli zobrazit posledně prováděnou činnost.

Zatímco do `datatype` se ukládá číselné označení typu zdrojových dat, do `datatype_name` se ukládá označení textové, tak, jak se zobrazuje uživateli v rozbalovacím menu. Textové označení se ukládá jednak pro zvýšení čitelnosti pro lidi, jednak pro možnost kontroly, že typ dat odpovídá v případných novějších verzích.

Položky prefixované `cs_`, `longbase_` a `longlong_` obsahují nastavení jednotlivých činností.

Do položek definovaných na řádcích 29–36 a 38–48 se ukládají informace o tom, která data se (ne)povedlo zpracovat v jednotlivých krocích a která se tedy (ne)mají používat v krocích následujících.

Tato struktura se při změně nastavení či dokončení nějakého kroku ukládá do souboru `jobstatus.mat` v pracovním adresáři. Načtením tohoto souboru je možné pokračovat v dříve zahájeném zpracování.

Šablona funkcí řídících zpracování

```
 1 function perform_action(fs_vars, subjects_data, analysis_data)
 2    t = timer;
 3    t.TimerFcn = {@action_timer_function};
 4    t.StopFcn = {@action_timer_stop};
 5    t.ExecutionMode = ___;
 6    t.period = ___;
 7
 8    processing = 0;
 9    processed = -1;
10    active = 1;
11
12    to_process = ___;
13
14    fs_transp = struct();
15    fs_transp.num_pending = to_process;
16    fs_transp.num_running = 1;
17    fs_transp.num_done = 0;
18    fs_transp.num_error = 0;
19
20    fs_transp.running = cell(1, 1); % Until multiprocess is supported
21    fs_transp.done = cell(1, to_process);
22    fs_transp.error = cell(1, to_process);
23
24    fs_transp.pending = cell(1, to_process);
25    pending = cell(1, to_process);
26    for i=1:to_process
27       fs_transp.pending{1, i} = ___;
28       pending{i} = ___;
29    end
30
31    runtime = java.lang.Runtime.getRuntime();
32    strs = get_cmd_strings(action, analysis_data);
33
34    % Dummy definitions
```

```
35    proc = 1;
36    cmd = 1;
37
38    next();
39
40    if (active)
41       start(t);
42    end
43
44
45    function action_timer_stop(~, event)
46       delete(t);
47    end
48
49
50    function action_timer_function(~, event)
51       cd(fs_vars.wdir);
52
53       if (processing == 1)
54          try
55             exit_code = proc.exitValue();
56
57             save(0, exit_code);
58          catch
59             err = lasterror();
60             if (strfind(err.message, 'exited'))
61             else
62                rethrow(err);
63             end
64          end
65       end
66
67       if (processing == 0) && (active)
68       % Fill in as needed
69
70          if (isfield('debug', 'fs_vars')) && (fs_vars.debug)
71             cmd = {'bash', ___};
72             disp('Invoking dummy command for debug');
73          else
74             cmd = {'bash', ___};
75          end
76
77          proc = runtime.exec(cmd);
78          processing = 1;
79       end
80
81       cd(fs_vars.odir);
82    end
83
84
85    function save(reason, code)
86       if (code ~= 0)
87          fs_transp.num_error = fs_transp.num_error + 1;
```

```
 88          fs_transp.error{fs_transp.num_error} = pending{processed+1};
 89       else
 90          fs_transp.num_done = fs_transp.num_done + 1;
 91          fs_transp.done{fs_transp.num_done} = pending{processed+1};
 92       end
 93
 94       processing = 0;
 95       next();
 96    end
 97
 98
 99    function next()
100       processed = processed + 1;
101
102       if (processed == to_process)
103          fs_transp.num_running = 0;
104          fs_transp.running = cell(1, 0);
105          assignin('base', 'fs_transp', fs_transp);
106          assignin('base', 'fs_processing', 0);
107          active = 0;
108          stop(t);
109          return;
110       end
111
112       fs_transp.num_pending = fs_transp.num_pending - 1;
113       fs_transp.running = fs_transp.pending(1:1);
114       if (fs_transp.num_pending > 0)
115          fs_transp.pending = fs_transp.pending(2:end);
116       else
117          fs_transp.pending = {};
118       end
119
120       assignin('base', 'fs_transp', fs_transp);
121       assignin('base', 'fs_processing', 2);
122    end
123 end
```

Na začátku se nastavují vlastnosti časovače, který se později bude starat o postupné zpracování jednotlivých dat. Dál se počítá, kolik dat je vlastně potřeba zpracovat.

Do proměnné `pending` se ukládají data potřebná při zpracování jednotlivých dat, do položky `pending` struktury `fs_transp` se ukládají identifikátory zobrazované uživateli v GUI.

Definice proměnných označené jako „Dummy definitions“ jsou mou snahou o zavedení lokálních proměnných funkce (viditelných i z v ní vnořených funkcí). Tyto konkrétní proměnné se používají až právě ve vnořených funkcích, ale kdyby byly zadefinované až v nich, budou lokální pro každou vnořenou funkci, což v žádném případě není záměrem.

Ideální by bylo v hlavní funkci tyto proměnné pouze deklarovat a žádnou hodnotu jim nepřiřazovat, nepovedlo se mi ale odhalit žádný způsob, kterým by toto bylo možné.

Na řádcích 54–64 se zjišťuje návratová hodnota procesu a ošetřuje případná výjimka způsobená tím, že proces ještě nedoběhl. Jelikož tento `try - catch` blok odchytí libovolnou výjimku, je nutné ověřit, že se opravdu jedná jen o nedoběhnutí procesu. V opačném případě došlo k neočekávanému problému a výjimka má být vyhozena dál. Toto ověření se provádí prostým pokusem o vyhledání podřetězce v popisu výjimky.

Při ladícím režimu se místo souboru, který by se pustil za normálních okolností, pouští soubor se stejným názvem a prefixem `dummy_`. Tímto způsobem je možné pohodlněji (a zejména rychleji) ladit např. detekci chyb v GUI, než kdyby bylo nutné data opravdu nechávat zpracovat.

Typografie

Zvláštní poznámku si dovolím ještě k problematice typografie a celkově k ladění estetického dojmu GUI. Při tvoření GUI jsem narazila na několik problémů, které z tohoto hlediska považuji za velmi vážné. Jako první příklad bych uvedla fakt, že zatímco texty v editovatelných polích se vertikálně zarovnávají nahoru, nápisy na tlačítkách se zarovnávají na střed, a Matlab nenabízí možnost, jak toto změnit. Druhým příkladem budiž to, že není možné text na tlačítku zalámat na více řádků.

V první řadě jde samozřejmě o vytvoření fungujícího nástroje, přesto si myslím, že když už se tvoří GUI, mělo by vypadat slušně, a to i z typografického hlediska.

Snažila jsem se proto najít možnosti, jak například rozdílné vertikální zarovnávání řešit. Zjistila jsem ovšem, že řešit tento problém na úrovni Matlabu skutečně není možné. Jediné možné řešení spočívá ve využití nižší úrovně a řešení problému již na úrovni Javy.

V době, kdy jsem tvořila samotný vzhled GUI (což bylo dříve, než jsem implementovala samotnou funkcionalitu, zejména kvůli komplikované situaci s konzultacemi a testováním, v jejímž důsledku bylo potřeba mít co nejdříve dostatečně ilustrativní návrh, tedy ideálně opravdu spustitelný kód pro Matlab), jsem ovšem tuto možnost nechtěla využívat, přišlo mi vhodnější držet se oficiálních nástrojů Matlabu.

Později při řešení vláken, resp. jejich náhrady, jsem stejně musela Javu využít, možná by tedy v budoucnosti stálo za zvážení doplnění kódu o využívání Javy již v tomto místě, a tedy třeba o lepší sladění právě u zarovnání.

5.3 Výsledky testování

5.3.1 Kognitivní průchod

Průřezové zpracování dat

Referenční průchod je následující:

1. Zvolit možnost New processing
2. Nahrát soubor `.xls`
3. Vybrat pracovní adresář
4. Vybrat adresář obsahující naměřená data
5. Vyplnit e-mail
6. Vyplnit ID
7. Vybrat možnost Copy data
8. Vybrat typ dat
9. Kliknout na Copy data
10. Vybrat možnost Cross-Sectional
11. V případě zájmu nastavit provádění *Talairach registration quality check*
12. V případě zájmu nastavit kontrolu, zda práce běží
13. Vybrat část zpracování, která se má provést
14. Kliknout na Perform cross-sectional analysis

Výsledky kognitivního průchodu:

Krok	Ví uživatel, co dál?	Ví, jak?	Zareaguje program rozumně?
1	ANO	ANO	ANO
2	NE[2]	ANO	ANO
3	ANO	ANO	ANO
4	ANO	ANO	ANO
5	NE[3]	NE[4]	ANO[1]
6	NE[5]	NE[6]	ANO[1]
7	ANO[7]	ANO	ANO
8	ANO	ANO	ANO[1]
9	ANO	ANO	ANO
10	ANO	ANO	ANO
11	ANO[8]	ANO	ANO[1]
12	ANO[8]	ANO	ANO[1]
13	ANO[8]	ANO	ANO[1]
14	ANO	ANO	ANO

1. GUI nijak nepotvrzuje, že dana jsou správně zadaná, to je ale běžné chování drtivé většiny formulářových aplikací
2. Pokud uživatel GUI nezná, netuší, že data musí být popsána v `.xls` souboru
3. Uživatel nemusí vědět, že má nějaký e-mail zadávat
4. Uživatel nemusí vědět, jaký e-mail má zadat; není jasné, že pole je nepovinné
5. Uživatel nemusí vědět, že má zadávat nějaké ID
6. Uživatel nemusí vědět, jaké ID má zadat; není jasné, že pole je nepovinné
7. Pokud uživatel GUI nezná, nemusí vědět, že data se musí nejprve zkopírovat (asi nejlepším řešením by bylo nutnost kopírování odstranit, kopírování dat „pouze“ povolovat a podporovat)
8. Uživatel musí mít zkušenosti buď s GUI, nebo se skriptováním pro FreeSurfer, aby věděl, že se dané údaje nastavují

Extrakce dat

Referenční průchod je následující:

1. Vybrat možnost Continue processing
2. Nahrát správný soubor `jobstatus.mat`
3. Vybrat možnost Extract data
4. Kliknout na Extract data

Výsledky kognitivního průchodu:

1	ANO	ANO	ANO
2	NE[1]	ANO	ANO
3	ANO	ANO	ANO
4	ANO	ANO	ANO

1. Pokud uživatel GUI nezná, netuší, co je `jobstatus.mat` a jak se s ním zachází

Srovnávací longitudinální zpracování dat

Referenční průchod je následující:

1. Vybrat možnost Longitudinal-long
2. V případě zájmu nastavit provádění *Talairach registration quality check*
3. V případě zájmu nastavit kontrolu, zda práce běží
4. Kliknout na Perform long longitudinal analysis

Výsledky kognitivního průchodu:

1	ANO	ANO	ANO
2	ANO[1]	ANO	ANO[2]
3	ANO[1]	ANO	ANO[2]
4	ANO	ANO	ANO

1. GUI nijak nepotvrzuje, že dana jsou správně zadaná, to je ale běžné chování drtivé většiny formulářových aplikací
2. Uživatel musí mít zkušenosti buď s GUI, nebo se skriptováním pro FreeSurfer, aby věděl, že se dané údaje nastavují

5.3.2 Heuristická evaluace

- Viditelnost stavu systému

  Bez nálezů

- Shoda mezi systémem a skutečným světem
  - Označení *Longitudinal-Base* a *Longitudinal-Long* jsou pro uživatele neznalého práce s FreeSurferem poněkud kryptická
- Uživatelské ovládání a uživatelská svoboda
  - Změnu nastavení není možné provést přímo v GUI
  - Změna nastavení se neprojevuje hned
  - Není možné přerušit zpracování dat
- Konsistence a standardy

  Bez nálezů

- Předcházení chyb
  - Nahrání souboru obsahujícího pro různé pacienty různý počet měření není detekováno jako chyba
  - Není možné zrušit vytvořený adresář při chybné volbě pracovního adresáře
- Preference rozpoznání před rozpomínáním se

  Bez nálezů

- Flexibilita a efektivita užití
  - Uživatel nemůže přeskočit část kopírování dat do pracovního adresáře
- Estetika a minimalistický design

  Bez nálezů

- Pomoc uživateli rozpoznat chyby, diagnostikovat je a zotavit se z nich
  - Chybí možnost zjištění detailů, dojde-li při zpracování dat k chybě
- Pomoc a dokumentace

  Bez nálezů

5.3.3 Testování s uživateli

Testování s uživateli probíhalo převážně prostřednictvím e-mailu, vždy jsem posílala novou verzi návrhu a dostávala nová hodnocení a nové připomínky. Závěrečnou verzi jsem se svou hlavní konzultantkou, MUDr. Zuzanou Nedelskou, testovala i prostřednictvím programu Skype. K výhodám testování pomocí Skype patří zejména rychlejší a v jistém směru autentičtější komunikace.

> Kromě již proběhlého testování se připravuje větší testování ve FN Motol, kdy by zejména měli program otestovat lidé, kteří jeho návrh nesledovali od začátku. Toto testování ale jednoznačně proběhne až po termínu odevzdání bakalářských prací.

Testující lékaři byli s GUI spokojeni, přišlo jim přehledné a srozumitelné.

Navzdory mému očekávání byl testujícími lékaři oceněn fakt, že kopírování a převod formátu probíhá v jednom kroku, uživateli tak odpadá jedna starost s pouštěním další části zpracování.

Lékaře potěšilo hlášení o stavu zpracování, které se umí samo aktualizovat. Mohlo by ovšem důrazněji dávat najevo, že skončilo celé zpracování (uvažuje se i o možnosti posílat uživateli při skončení zpracování e-mmail, o kterou lékaři požádali).

Obzvláště oceněna byla možnost spouštět průřezové zpracování po částech. Ideální by bylo do GUI doimplementovat možnost nechat zpracování opakovat pro vybranou část dat (typické užití by bylo pro data, jejichž zpracování skončilo s chybou).

Lékaři by ocenili přesnější informace k chybám, ke kterým dojde při zpracování FreeSurferem, není to pro ně ale prioritní.

5.4 Známé problémy

Známé problémy jsou rozděleny do následujících kategoriích, popis těchto kategorií je doslovnou citací z mé semestrální práce do předmětu A4B39TUR Testování uživatelského rozhraní [3]:

- Kritické chyby
  Chyby, které znemožňují zamýšlené využití systému a jejichž odstranění je nezbytné pro správné fungování.
- Závažné chyby
  Chyby, které významným způsobem komplikují využití systému, a které by proto měly být co nejdříve odstraněny.
- Chyby
  Chyby, jejichž odstranění by uživateli zpříjemnilo práci se systémem, ale jejichž přítomnost tuto práci nijak zásadně neomezuje.
- Kosmetické chyby
  Chyby, které mohou kazit příjemný dojem ze systému. Jejich odstranění není nutné.
- Připomínky
  Nálezy, které by neměly být považovány za chyby v klasickém smyslu, ale které mohou stát za pozornost.

Kritické chyby

Žádné kritické chyby nejsou známy

Závažné chyby

- Zobrazení stavu zpracování (*Check jobs*) se při zahájení nového zpracování aktualizuje se zpožděním

  Zadá-li uživatel nové zpracování (např. základní longitudinální po dokončení průřezového), zůstane ve stavu zobrazení několik vteřin stará informace. To může být pro uživatele velice matoucí, neboť neví, zda se zpracování skutečně zahájilo.

Chyby

- Není možné přerušit zpracování

  Zpracování není možné přerušit, např. po dokončení analýzy dat právě zpracovávaného pacienta.

  Tato chyba by se mohla zdát velice závažnou, je ovšem třeba mít na paměti to, že zpracování dat jednoho pacienta probíhá mnoho hodin (a přerušit zpracování dat pacienta „něžně“, tedy tak, aby se dalo v budoucnosti jednoduše navázat, nejde), takže by i při možnosti přerušení z GUI byly s touto možností spojené různé komplikace.
- Neprobíhá kontrola, že všichni pacienti mají stejný počet měření

  Při načítání dat z průvodního souboru `.xls` se nekontroluje, zda mají všichni pacienti stejný počet měření (uživatel je pouze prostřednictvím dokumentace upozorněn, že nedodržení tohoto požadavku může vést k neočekávanému chování).

  Informace, že počet měření musí být vždy stejný, pochází od spolupracujících lékařů. Tato podmínka ale zřejmě plyne mnohem víc z relevantnosti sady dat pro nějaký výzkum než z možností FreeSurferu (kde má smysl pouze odlišovat, zda měření bylo jedno, nebo jich bylo více, tedy zda dává smysl provádět longitudinální zpracování).

  Před doimplementováním kontroly tedy počítám s dalšími konzultacemi ohledně toho, co a proč je vlastně potřeba ve věci počtu měření kontrolovat.
- Nelze odstranit omylem vytvořený adresář

  Pokud se uživatel při výběru pracovního adresáře překlikne, vytvoří se mu nový adresář pro data v jiném místě, než si původně přál. Tento nový adresář ale není možné prostřednictvím GUI smazat.

  Otázkou k zamyšlení je, zda by GUI spíše mělo toto umožnit, nebo vyžadovat potvrzení před vytvořením adresáře (na jednu stranu to představuje zbytečné obtěžování uživatele, na druhou stranu uživatel tuto akci provádí jednou za celé zpracování).
- Označení *Longitudinal-Base* a *Longitudinal-Long* nejsou samopopisná

  Pro uživatele, který nemá zkušenosti s ovládáním FreeSurferu, vůbec nemusí být zjevné, co se skrývá pod možnostmi *Longitudinal-Base* a *Longitudinal-Long*, a to ani pod nápisy na spouštěcích tlačítkách, tedy *Perform base longitudinal analysis* a *Perform long longitudinal analysis*.

Kosmetické chyby

- Informační pole se automaticky nepřepisuje

  Dojde-li např. k chybě, na kterou je uživatel upozorněn prostřednictvím vrchního informačního pole, ani po vyřešení této chyby nezmizí z vrchního pole upozornění na chybu.

- Měnit nastavení je možné pouze přepisováním konfiguračního souboru

  Pro uživatele, který preferuje GUI před příkazovou řádkou, je typicky také příjemnější moci měnit nastavení prostřednictvím tohoto GUI než přepisováním souboru neodlučitelně spjatého s programem.

Připomínky

- Extrakce dat není nastavitelná

  GUI nabízí pouze export kompletních dat, přestože skripty poskytované v rámci FreeSurferu umí vybírat jen část těchto dat.

  S doimplementováním nastavení se v blízké době počítá.

Kapitola 6
Závěr

V rámci své práce jsem se soustředila na vytvoření uživatelsky co možná nejpříjemnějšího rozhraní k nástroji FreeSurfer pro neurology, primárně pro lékaře a mediky z Neurologické kliniky 2. lékařské fakulty a Fakultní nemocnice Motol.

FreeSurfer je sice velice mocný, ale zatím k němu neexistovalo žádné GUI (alespoň ne GUI pro ovládání FreeSurferu, např. GUI pro zobrazení výsledků zpracování existují již delší dobu).

V rámci své práce jsem se blíže seznámila s tímto nástrojem a vybudovala nad ním první verzi grafického rozhraní, které se bude lékařům příjemně používat. Abych se s FreeSurferem blíže seznámila, konzultovala jsem jeho použití s lékaři, kteří jej používají běžně, a docházela jsem do FN Motol, kde jsem měla možnost pracovat na neuGRIDu.

Vytvářené rozhraní jsem v souladu se zásadami pro uživatelsky orientovaný design průběžně konzultovala a testovala s lékaři a přizpůsobovala jejich požadavkům. K otestování správné funkčnosti GUI jsem využívala poskytnutá anonymizovaná data.

Podle vyjádření lékařů se cíl povedlo splnit. Lékaři jsou se současnou verzí GUI velmi spokojení a věří, že jim usnadní práci s naměřenými daty. Zatím existují nějaké nevyřešené problémy a také žádosti o nové funkce, ale jak ze strany mé, tak ze strany lékařů je zájem ve spolupráci pokračovat.

V současné době čeká GUI další testování a pak bude zřejmě opravdu nasazeno jako alternativa k příkazové řádce.

6.1 Shrnutí dílčích výsledků

6.1.1 Návrh základních operací

Navrhla jsem zřetězení operací, které nyní GUI respektuje. Operace není možné provádět v jiném než ve stanovém pořadí; pořadí pochopitelně odpovídá požadavkům nižší vrstvy (samotného FreeSurferu), ale odpovídá i požadavkům lékařů, s kterými jsem návrh během vývoje konzultovala. Navržené zřetězení těmto lékařům přijde intuitivní a srozumitelné.

Navržené zřetězení předpokládá lineární zpracování, maximálně s občasnou možností udělat nějakou akci „od někdy“ kdykoliv, s tím, že na této akci již nic dalšího nezáleží. Takové zřetězení je sice omezující, ale plně odpovídá způsobu, jakým se data zpracovávají na NK FN Motol.

6.1.2 Návrh více variant GUI

Na začátku práce jsem vytvořila tři základní návrhy GUI. Ty jsem dále konzultovala se spolupracujícími lékaři. Společně jsme v nich navrhovali možné změny a možná vylepšení. Nakonec jsme vybrali jeden návrh, který se zdál lékařům být nejpříjemnější na používání, a tedy nejvhodnější na realizaci.

6.1.3 Implementace a dokumentace vybraného řešení

Vybraný návrh jsem implementovala v MATLABu. V současné době vím o několika chybách a zaznamenala jsem několik proseb o rozšíření funkcionality, ale GUI je i s těmito chybami připravenou k používání.

Dokumentace má primárně podobu komentářů ve zdrojovém kódu, dalším zdrojem informací je popis implementace v této práci. Právě z popisu implementace bude možná vytvořena programátorská dokumentace.

Ke GUI pro FreeSurferu dále existuje uživatelská dokumentace, kterou přikládám k této práci.

6.1.4 Otestování vytvořeného SW

GUI bylo otestováno na reálných datech. Testování s uživateli probíhalo formou e-mailu a prostřednictvím programu Skype (mimo jiné proto, že v době dokončování práce byli hlavní spolupracující lékaři dlouhodobě mimo ČR). V době odevzdávání bakalářské práce domlouváme větší testování přímo na FN Motol, to by mělo být nejspíš cca týdenní a proběhnout někdy v červnu.

Do tohoto testování by se měli zapojit i lékaři, kteří s GUI zatím nemají žádnou zkušenost (mimo jiné neměli možnost sledovat vyvíjející se návrh), navíc se počítá s dlouhodobějším testování, a tedy např. možností důkladněji otestovat schopnost GUI spouštět více longitudinálních analýz.

6.2 Další možnosti

Dopadne-li velké pražské testování dobře, bude možné GUI nabídnout i mimo prostředí NK FN Motol. Jelikož byl program od začátku vyvíjený v angličtině, může být nabízen i mimo ČR.

Bez ohledu na míru šíření existuje několik zdokumentovaných problémů, které bych se z GUI ráda pokusila v blízké době odstranit.

Dále bude hodně záležet na tom, jak se GUI na NK FN Motol osvědčí dlouhodobě. Budou-li si to lékaři přát, bude možné GUI dále rozšiřovat, případně přizpůsobovat nějakému konkrétnímu použití.

Podobných balíků jako FreeSurfer navíc existuje více, a lékařům s kterými jsem GUI konzultovala, by se líbilo, kdyby podobné GUI, jaké jsem vytvořila pro FreeSurfer, někdy existovalo i pro jiné balíky. Možná tedy na základě této práce vznikne daleko obsáhlejší soubor programů, jednotlivá GUI pro jednotlivé balíky.

Ovšem bez ohledu na další balíky, výsledek této práce několika lidem zpříjemnil jejich práci.

Literatura

[1] Bruce Fischl. FreeSurfer. *NeuroImage*, 62:774–781, 2012. http://www.sciencedirect.com/science/article/pii/S1053811912000389.

[2] Kolektiv správců. *Wiki projektu FreeSurfer*. https://surfer.nmr.mgh.harvard.edu/fswiki.

[3] Karolína Burešová. Testování webových stránek korespondenčního semináře z programování. http://hcisemestralky.felk.cvut.cz/system/mems/6154/original/A2_TUR_bureskar.pdf.

[4] David Travis. *The Fable of the user-centered design*. Userfocus, 2009. http://www.userfocus.co.uk/fable/.

[5] Carolyn Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann, 2003.

[6] Jacob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

[7] Donald Norman. *The Design of Everyday Things*. Basic Books, 2002.

[8] Adam Sporka. Přednáškové slidy k předmětu a4b39tur. Dostupné online na https://cent.felk.cvut.cz/courses/Y39TUR/?page=slides.

[9] Ed H. B. M. Gronenschild, Petra Habets, Heidi I. L. Jacobs, Ron Mengelers, Nico Rozendaal, Jim van Os, and Machteld Marcelis. The effects of freesurfer version, workstation type, and macintosh operating system version on anatomical volume and cortical thickness measurements. *PLOS one*, 2012.

[10] *MathWorks, official MATLAB reference*. http://www.mathworks.com/.

[11] *MATLAB Undocumented*. http://undocumentedmatlab.com/.

[12] Stack overflow. http://stackoverflow.com/.

[13] Karel Zaplatílek and Bohuslav Doňar. *MATLAB tvorba uživatelských aplikací*. BEN – Technická literatura, 2004.

[14] Karel Perůtka. *Základy pro studenty automatizace a inform. technologií*. Univerzita Tomáše Bati ve Zlíně – Fakulta technologická, 2005.

Příloha A
Zadání práce

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Karolína B u r e š o v á

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Tvorba GUI pro nástroj FreeSurfer

Pokyny pro vypracování:

Seznamte se s prací nástroje FreeSurfer, který je běžně používán v neurologii pro hodnocení MRI mozku a slouží k volumometrii (měření objemu) jednotlivých mozkových struktur. Tento volně dostupný SW [1, 2] má z hlediska lékaře řadu nevýhod, neboť jeho použití je časově zbytečně náročné a navíc požaduje součinnost uživatele na úrovni psaní skriptů, což je pro běžného lékaře zcela nepřijatelné. Cílem bakalářské práce je navrhnout a implementovat uživatelsky efektivnější a jednodušší uživatelské rozhraní, které je schopno realizovat úlohy nejčastěji řešené na Neurologické klinice FN Motol.

1. Navrhněte typy základních operací realizovaných prostřednicím GUI, případně i způsob jejich parametrizace či zřetězení tak, umožňující realizovat výše uvedené úlohy.
2. Navrhněte více variant pro odpovídající GUI a vyberte ve spolupráci s neurologickým pracovištěm nejvhodnější řešení.
3. Vybrané řešení implementujte a řádně dokumentujte.
4. Vytvořený SW otestujte v reálném použití.

Seznam odborné literatury:

[1] Fischl Bruce: FreeSurfer. NeuroImage, Volume 62, Issue 2, 15 August 2012, pp. 774-781.
[2] Shneiderman, B., Plaisant, C.: Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition). Addison Wesley, 2004. ISBN 0321197860.

Vedoucí bakalářské práce: prof. RNDr. Olga Štěpánková, CSc.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Příloha B
Seznam použitých zkratek

FN	Fakultní nemocnice
GUI	Grafické uživatelské rozhraní (*Graphical user interface*)
LF	Lékařská fakulta
MRI	Magnetická rezonance
NK	Neurologická klinika
UK	Univerzita Karlova

Příloha C
Uživatelská dokumentace

FreeSurfer GUI aims to be an easier way to use FreeSurfer, a software package for MRI data analysis. To be able to run it, you have to have FreeSurfer (version at least 5.3) and Matlab (no specific version is required) installed on your machine. (FreeSurfer GUI also assumes you're using OS based on UNIX which, however, shall be ensured by having FreeSurfer installed.)

Installation

First of all, you have to download FreeSurfer GUI. You should download if from the following URL:

`http://www.karryanna.cz/projects/fsgui/fsgui.zip`

The GUI is packed in a ZIP archive so you need to extract it first. On most Linux distributions, typing `unzip -d` *destination* `fsgui.zip` in terminal should do the trick (assuming `fsgui.zip` is saved in your current working directory, otherwise you have to prefix the name with path, like `Downloads/fsgui.zip`. The option to extract archive will most likely be available also in context menu when right-clicking this file.

You might wish to modify `settings.conf` afterwards. It is necessary to have path to FreeSurfer set correctly in this file. Look for the line reading

`home /usr/local/freesurfer`

and change the path to the one where FreeSurfer is installed on your machine.

Running FreeSurfer GUI

To run FreeSurfer GUI, you have to be running Matlab so start Matlab first. In Matlab, using `cd` command switch to the directory to which FreeSurfer GUI has been extracted. Then start the GUI by typing `freesurfer_gui`.

If you'd like to be able to run the GUI from any directory, you either have to extract it somewhere on MATLAB path or add the path to FreeSurfer GUI to MATLAB path.

Accompanying spreadsheet

You have to create a special Microsoft Excel file for the data you wish to process. The structure of this file must be as follows:

subject id	scan 1	scan 2
SUBJ01	06/20/2011	07/01/2012
SUBJ02	06/30/2011	07/03/2012

That is, first row is a kind of heading and is meant only for your convenience. **Beware: The first row is not read by FreeSurfer GUI.** First column contains IDs of your subjects, each other column contains dates of scans (no special format of date is required but make sure that Microsoft Excel really considers those columns dates). All subjects should have the same amount of scans performed!

FreeSurfer GUI expects that each scan data are saved in folder named SUBJID_SCANDATE. Specific format of date is required here, YYYYMMDD. For example, the first scan would be looked up in folder named SUBJ01_20112006.

Beware: FreeSurfer GUI assumes that lexicographical order of scan images corresponds with the order in which those images were taken (this is most likely the default behaviour of your software).

Working directory

You are required to select a working directory. New directory named by the same name as provided spreadsheet will be created in selected directory. The whole proccess will be done in this newly created directory.

jobstatus.mat

Whenever you set something, or when a specific action is started or finished, FreeSurfer GUI saves some information to a special file called `jobstatus.mat`. This file is located in the created working directory. If you wish to continue processing, you have to load this file into FreeSurfer GUI.

Provided actions

- Copy data
  Copies data from data directory to working directory. Names of data folders must respect the format of SUBJID_YYYYMMDD. Data are automatically converted to FreeSurfer-compatible format. Original data are saved under orig folder in subject folder.
- Cross-Sectional analysis
  Performs cross-sectional analysis on all data that were found and successfully converted.
- Longitudinal-base analysis
  Performs part of longitudinal analysis, template creation. Templates for each subject are named like `SUBJID_base`.
- Longitudinal-long analysis
  Performs individual runs of longitudinal analysis for all scans meeting the following conditions:
  - Data of such scan were successfully copied and converted
  - Relevant SUBJID_base was successfuly created
- Check jobs
  Allows you to check status of individual jobs. Job listed as running is currently being processed by FreeSurfer, done jobs were successfuly processed, error jobs were processed but FreeSurfer reported an error and pending jobs have not been yet processed by FreeSurfer.
- Extract data
  Extracts data using `asegstats2table`. Extracted data will be saved under `stats.txt` in working directory.
- Visualize data
  Allows you to visualize data that have already been processed by FreeSurfer. Data are visualized either by TkMedit or by TkSurfer, depending on wheter you with to view volume or surface.

In case you have any questions, suggestions or bug reports, feel free to send them to `karry+fsgui@karryanna.cz`.

Příloha D
Vyjádření lékařů k významu GUI

Centrum pro poruchy paměti při Neurologické klinice 2. lékařské fakulty
Univerzity Karlovy v Praze a Fakultní nemocnice v Motole, Doc. MUDr. Jakub Hort, PhD –
vedoucí centra, Doc.MUDr.Petr Marusič, Ph.D. - přednosta
V Úvalu 84, Praha 5, 150 06, Tel: +420 224 436 801, Fax: 224 436 820

Zdůvodnění významu a praktického využití FreeSrurfer GUI – uživatelského interface ke kvantitativní volumometrii mozkových struktur v oblasti Alzheimerovy nemoci a dalších neurodegenerativních onemocnění.

Freesurfer je open source software pro analýzu MRI dat, vyvíjený právě v Martinos Center for Biomedical Imaging, Boston. Cílová MRI data a mozkové struktury jsou zejména volumetrická měření hippocampu, který slouží jako biomarker Alzheimerovy choroby (ACH), segmentace hipokampálních subpolí, měření tloušťky mozkové kůry, a fůzní studie s jinými MRI modalitami jako funkční MRI či difůzní tensorové zobrazení bílé hmoty.

Aplikace GUI: FreeSurfer adaptace do GUI bude extenzivně využita našimi neurology, mediky, a PhD studenty či postdoky při tvorbě databáze a archivu MRI dat pacientů s ACH a dalšími neurodegenerativními chorobami, např. demencí s Lewyho tělísky, fronto-temporální demencí, kortikobazální degenerací, dat pro hodnocení vaskulárních změn u demencí, etc. FreeSurfer GUI umožní provedení všch základních designů studií morfometrie a atrofie celého mozku s výběrem struktur relevantních konkrétní nemoci, t.j. kros-sekční i longitudinální region-of-interest based analýzu, měření kortikální tloušťky či selektivně jen volumometrii subkortikálních jader šedé hmoty. T.č. je kvantitativní volumometrie prezentována jako klinický výkum, avšak měření atrofie mozku je již akceptovaný „level of evidence“ v praxi s využitím principů evidence-based medicine, a atrofie hipokampu jako proxy ACH patologie je již součástí nových diagnostických doporučení. GUI sice není tolik flexibilní jako použití FreeSurfer příkazové řádky, ale má nespornou obrovskou výhodu v jednoduchosti použití pro kliniky v praxi a v každodenním klinickém výzkumu. Robustnost FreeSurfer lgoritmu je v literatuře bohatě dokumentována, takže jeho adaptace do GUI nenese riziko ztráty funkčnosti či přesnosti měření.

Neurologická klinika 2. LF UK v má v současné době velmi dobré zázemí a soubor několika stovek pacientů, kteří jsou každoročně retestovánicelou baterií vyšetření včetně MRI, s cílem sledovat klinickou progresi onemocnění a sbírat longitudinální data. Vzniká tak unikátní soubor pacientů a dat tzv „the Czech Brain Aging Study (CBAS) – první longitudinální studii ACH, dalších demencí a jejich rizikových faktorů s epidemiologickám podtextem v ČR. Kvalitu multidisciplinárního týmu a studie dokazuje množství článků v impaktovaných časopisech a běžících grantů, s účastí mezinárodních špičkových zdravotnických a výzkumných pracovišť jako např. Mayo Clinic Rochester.

Jsme velmi rádi a děkujeme za možnost spolupracovat na tomto velmi užitečném projektu se studentkou Karolinou Burešovou.

Blízká budoucnost: GUI v diferenciální diagnostiky demencí, charakteristiky disease specific patterns, zobrazování korelátů prostorové navigace jako biomarkeru ACH a dále pro zvýšení celkové kvality a komfortu práce.

Grantová činnost:

- Grantová agentura Univerzity Karlovy v Praze (GAUK č. 624012, vedoucí Mudr. Zuzana Nedelská „Poruchy prostorové navigace a jejich MRI koreláty u nedementních seniorů v reálném prostoru a v prostředí virtuálních PC testů“

Kontakt na konzultantku práce za Neruologickou kliniku FN Motol:

MUDr. Zuzana Nedelská, nedelskaz@gmail.com

Příloha E
Obsah přiloženého CD

- `text.pdf` – tato práce ve formátu PDF
- `fsgui.zip` – ZIP archiv obsahující zdrojové soubory GUI tak, jak byl v době odevzdání práce k dispozici ke stažení z Internetu