Slovenská technická univerzita

Fakulta informatiky a informačných technológií

Ilkovičova 3, 842 16 Bratislava 4

FIITKinect

(Dokumentácia riadenia)

Tím: TeamToo
Vedúci: Ing. Vanda Benešová, PhD
Kontakt: tim2_fiit@googlegroups.com
Ak. rok: 2012/2013

Autori: Bc. Ján Antala
Bc. Martin Čertek
Bc. Jakub Gondár
Bc. Ondrej Grman
Bc. Silvia Hudačinová
Bc. Michal Igaz
Bc. Richard Sámela

Obsah

1 Úvod

Dokumentácia obsahuje všetky dokumenty opisujúce riadenie tímu v rámci vývoja softvérovej aplikácie FIITKinect. Dokumentácia je členená na viaceré časti. Úvodná časť zahŕňa informácie o tíme, tímovú ponuku pri uchádzaní sa o pridelenie projektu, zároveň je v časti ponuka spomenutý aj spôsob ako by chcel tím vypracovať projekt a kvalifikácie všetkých
členov tímu.
Jadro tvoria dokumenty predstavujúce manažment jednotlivých zložiek projektu od plánu projektu v manažmente plánovania, cez manažment rizík zahrňujúci identifikovanie a návrh riešenia rizikových situácií pri vývoji, nasledovaný manažmentom kvality popisujúcim spôsoby riadenia procesu tvorby softvérového diela s ohľadom na kvalitu. Práca sa zaoberá aj manažmentom komunikácie a definuje aj postupy pri tvorbe dokumentácie. V časti riadenia podpory vývoja je uvedený spôsob práce s verziovacím softvérom a v časti monitorovania projektu je opísaný spôsob dohľadu nad projektom.
Na záver dokumentu sú zaradené záznamy zo stretnutí, opisujúce a dokumentujúce jednotlivé stretnutia tímu. Pri stretnutiach je uvedené čo bolo predmetom stretnutia a aké sú úlohy jednotlivých členov tímu do nasledujúceho stretnutia.

2 Ponuka

V tejto časti je vypracovaná ponuka tímu TeamToo. Táto ponuka bola vypracovaná v prvom týždni semestra za účelom pridelenia projektu Kinect.

2.1 Členovia tímu

Sme tím mladých ľudí, ktorý majú radi technológie, nové výzvy a objavovanie ciest ako ich riešiť. Náš tím tvorí viacero spolužiakov už zo strednej školy, na ktorej sme vypracovávali zadania a spolu ako partia vyrábali a následne riešili problémy. Našou tímovou výhodou je zohratosť a taktiež nemusíme toľko času tráviť spoznávaním sa a učením sa dôverovať druhým členom tímu. Zvyšní členovia tímu priniesli nové pohľady, skúsenosti z nového prostredia a tak si myslíme, že ako tím máme všetky ľudské a veríme, že aj odborné predpoklady na riešenie výziev, ktoré sú pred nami.

Bc. Ján Antala
Je absolventom bakalárskeho študijného programu Informatika na FIIT. Počas štúdia získal skúsenosti s programovacími jazykmi C a Java. Bakalársku prácu vypracoval na tému Bohaté internetové aplikácie, vďaka čomu nadobudol znalosti tvorby multiplatformových webových aplikácií pre mobilné zariadenia. Venuje tvorbe webových stránok v HTML5, CSS3 a Javascripte s využitím princípom Mobile first. Taktiež má skúsenosti s prácou s databázami MySQL a CouchDB..

Bc. Martin Čertek
Bakalárske štúdium ukončil na FIIT STU v odbore Informatika. Počas štúdia získal skúsenosti s prácou v programovacích jazykoch C a Java. V bakalárskej práci sa venoval tvorbe e-learningového systému založeného na PHP, pracoval aj sa databázovými technológiami MySQL. Má skúsenosti z oblasti dobrovoľníctva s vedením tímu ľudí.

Bc. Jakub Gondár
Absolvent bakalárskeho štúdia FIIT STU v odbore Informatika. V záverečnej práci sa zaoberal témou Elektronického hlasovania, kde analyzoval, navrhol a implementoval elektronický hlasovací systém.
Najväčšie skúsenosti pri vývoji aplikácii má s platformou Microsoft .NET (C#, ASP.NET). Ďalej má skúsenosti s vývojom pre platformu Java a modelovaním v jazyku UML.

Bc. Ondrej Grman

Bakalárske štúdium ukončil na FIIT STU v odbore Informatika. Pri práci na projektoch nadobudol skúsenosti s jazykmi C, JAVA, databázami MySQL. Bakalársku prácu :Aplikácia na vyhľadávanie notového zápisu vytvoril v C#. Venuje sa tvorbe webových stránok v HTML, PHP, Javascripte. Má skúsenosti so správou menšej firemnej siete.

Bc. Silvia Hudačinová

Prišila študovať na FIIT STU po absolvovaní bakalárskeho štúdia na UKF v študijnom programe Aplikovaná informatika. Počas štúdia sa venovala programovaniu v C++ a tvorbe webových stránok v PHP, Flashi. Má skúsenosti s UML, používaním Matlabu a prácou v tíme 4 ľudí.

Bc. Michal Igaz

Absolvoval bakalárske štúdium na FIIT STU. Téma jeho bakalárskej práce bola : Vývoj informačných systémov podľa princípov architektúry orientovanej na služby . Vďaka nej získal skúsenosti v oblasti SOA, konkrétne s prácou s webovými službami. Ďalej má skúsenosti s programovacími jazykmi Java, C a riešeniami IBM WebSphere. Pracoval v menších tímoch na školských zadaniach.

Bc. Richard Sámela

Bakalársky stupeň vysokoškolského štúdia absolvoval na FIIT STU v študijnom odbore Informatika. Má skúsenosti s programovacím jazykom Java, C a databázovými systémami SQLite a MySQL. Jeho bakalárska práca bola zameraná na tvorbu aplikácií pre mobilné zariadenia pracujúce na platforme Android. V praxi sa v tíme venuje tvorbe mobilných aplikácií pre Android.

2.2 FIIT Kinect

Chceme sa podieľať na vývoji riešenia, ktoré mení stereotypy a prináša zlepšenia do života ľudí.

Domácnosti ľudí sa v poslednej dobe zapĺňajú množstvom techniky určenej na poskytovanie informácií a zábavy – televízory, domáce kiná, hudobné prehrávače sú neoddeliteľnou súčasťou každodenného života. S nárastom počtu zariadení ale dochádza čoraz častejšie k potrebe ovládať zariadenia a využívať ich čo možno najefektívnejšie a najefektnejšie.

Jednou z našich motivácií vedúcou k uchádzaniu sa o túto tému je zefektívniť fungovanie domáceho multimediálneho centra a popasovať sa s problémom - ovládač na zariadenie sa vždy nachádza mimo dosahu používateľa, a to s využitím možností ovládania pomocou pohybov a gest, ktoré my ľudia bežne používame pri komunikácii, vyjadrovaní emócií.

Vývoj domácej techniky napreduje rýchlym tempom, no od roku 1955 , kedy bol použitý prvý diaľkový ovládač k TV sa v tejto oblasti veľa nezmenilo. Preto si myslíme, že nastal čas zmeniť zaužívané a pridať ovládaniu domácej techniky nový rozmer – ovládanie gestami, hlasom.

Za perspektívnu oblasť považujeme aj možnosť ovládať zariadenia pomocou inteligentných telefónov, kde má náš tím potenciál zúročiť skúsenosti nadobudnuté členmi tímu pri vývoji mobilných aplikácií v bakalárskych prácach a praxi.

Zaujímavou oblasťou pre nás je práca so senzorom Kinect umožňujúcim zachytávanie a rozpoznávanie gest od používateľa. Toto zariadenie predstavuje revolučný nástroj pre spracovanie a analyzovanie pohybu v 3D priestore, čo otvára nové a pre nás zaujímavé možnosti. Podieľať sa na vývoji a pracovať s týmto revolučným zariadením pre nás predstavuje obrovskú výzvu a motiváciu pre prácu s novými technológiami.

Náš tím je všestranne zameraný a kreatívny. V tíme sú študenti programujúci aplikácie pre mobilné zariadenia, máme skúsenosti s technológiami firmy Microsoft a programovaním v jazykoch C#, .NET. Počas bakalárskeho štúdia dvaja naši členovia realizovali v predmete IČP-HCI projekt, ktorý v súťaži tímov skončil na 2. mieste.

Našou snahou je prísť s aplikáciou, ktorá prepája rôznorodé technické zariadenia a integrovať ich do funkčného celku tak, aby používateľ v prostredí NUI mohol ľahko pracovať bez nutnosti znalostí technológií stojacich za riešením.

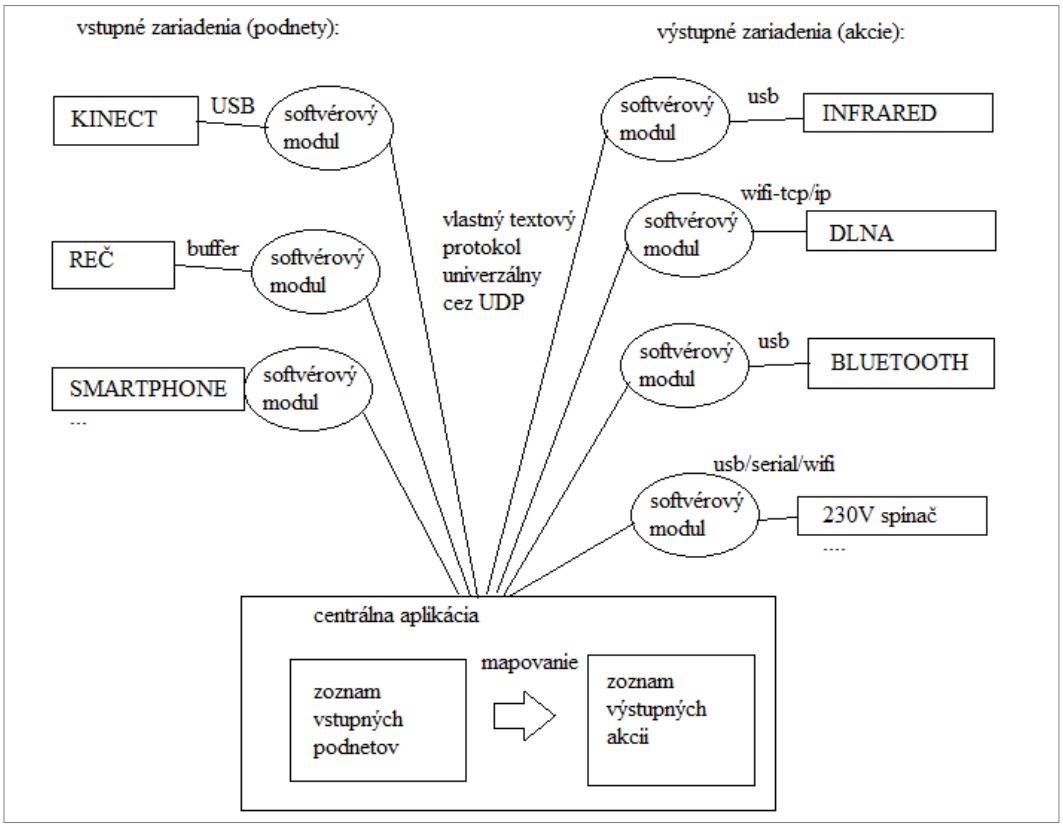
2.2.1 Návrh riešenia

Pri riešení sa chceme zamerať na vývoj modulárneho systému umožňujúceho pridávať vstupné a výstupné moduly zabezpečujúce príjem vstupných informácií (Kinect- gestá, mikrofón- reč, smartfóny- povely od používateľa) a výstupných zariadení ovládajúcich domáce spotrebiče -IrDA, bluetooth, wifi , DLNA s použitím centrálnej aplikácie spravujúcej pripojené rozhrania.

Konfigurácia ovládacích gest a nastavení od používateľa bude realizovaná v prostredí prehľadného webového rozhrania prístupného ako z desktopových, tak aj mobilných zariadení. Pri zisťovaní gest od používateľov plánujeme nadviazať a vylepšiť výsledky predchádzajúceho tímového projektu v tejto oblasti.

Prepojenie jednotlivých vstupných/výstupných rozhraní prostredníctvom centrálnej aplikácie plánujeme riešiť definovaním vlastného protokolu umožňujúceho komunikovanie zariadení a pridávanie zariadení bez ohľadu na platformu a poskytujúceho dobré možnosti rozšíriteľnosti pre ďalší vývoj systému.

Nemenej dôležité je vytvoriť intuitívne rozhranie pre tvorbu gest a priraďovanie akcií realizovaných na základe zadaných povelov, ktoré plánujeme vytvoriť ako webové rozhranie prístupné aj z mobilných aplikácií umožňujúce nastavovať a spravovať gestá bez ohľadu na platformu zariadenia.



Obr. 1.: Schematický nákres architektúry riešenia.

2.3 Simulácia demonštrácie v meste

Rozvoj občianskej spoločnosti prináša so sebou aj nárast počtu vyjadrovania sa jednotlivcov pri hromadných akciách- protestoch. Často ide iba o prejavovanie názoru a snahu napraviť a zmeniť situáciu za účelom dosiahnutia zlepšenia. No existujú aj skupiny ľudí, ktorých aktivity pri protestoch, demonštráciách nesmerujú k náprave situácie, ale naopak k radikalizovaniu a eskalácii.

Náš tím, ako skupina mladých ľudí podporuje snahy jednotlivcov o zmeny, hoci aj formou protestov, no na druhej strane je potrebné aby pri takýchto podujatiach nedochádzalo k roztržkám a účel protestu s dobrým zámerom nebol "pošliapaný" a protest sa nevymkol kontrole. Ak už k takémuto prípadu dôjde, je potrebné čo možno najlepšie vedieť odhadnúť čo sa bude diať a ako na takúto vzniknutú situáciu reagovať.

Simulovať správanie sa protestujúceho davu nie je užitočné iba pre silové zložky (polícia, armáda). Takýto nástroj na simuláciu by bol vhodný aj pre občianskych aktivistov – združenia, ktorým by pomohol pri organizovaní a plánovaní podujatí a poslúžil by aj pri jednaniach s úradmi v prípadoch povoľovania zhromaždení.

Zaujímavou oblasťou pre náš tím je možnosť reálneho nasadenia systému v prostredí polície a prepojenie akademickej sféry s reálnymi poznatkami z praxe, čo by nám prinieslo nové skúsenosti s aplikovaním znalostí z reálneho nasadania spolu so zlepšením komunikačných skúseností pri vývoji softvérových riešení pre klienta.

Našou motiváciou pre prácu na tomto projekte je poskytnúť možnosti na eliminovanie problémov, ktoré môžu vznikať pri protestoch a vytvoriť riešenie použiteľné v praxi napomáhajúce k ochrane poriadku a zdravia.

Zaujímavou oblasťou je práca s modelovaním emócií, kde sa otvárajú nové možnosti skúmania ľudských reakcií na vzniknutú situáciu a interakcie jednotlivcov v skupine – dave za pomoci simulačných nástrojov.

Náš tím má bohaté skúsenosti s programovacím jazykom JAVA zo školských projektov, ako aj z praxe, a tiež aj znalosti z predmetu Umelá inteligencia.

2.3.1 Návrh riešenia

V simulácii budú uvažované dve proti sebe stojace skupiny:
Protestujúci
Koná na základe rôzneho stupňa emocionálneho vypätia, čo je potrebné v návrhu zachytiť a pracovať s tým. Namodelovaním niekoľkých typov správania sa agentov – bežný človek, provokatér, výtržník zachytíme možné postoje protestujúcich.

Poriadkové zložky
Návrh ráta s vytvorením niekoľkých typov agentov zameraných na plnenie úloh dohliadania na protest. diverzifikáciu
Okrem typickej úlohy – policajta, resp. policajta „ťažkoodenca" je vzhľadom na diverzifikáciu síl potrebné rátať a navrhnúť aj ďalších agentov silových zložiek: policajta s koňom, správanie sa špeciálnych zásahových vozidiel – vodné delá, špeciálneho vozidla Božena Riot.

V návrhu riešenia uvažujeme aj nad emocionálnymi vplyvmi na jednotlivých agentov, v závislosti od odvíjajúcej sa situácie počas protestu.

Pri návrhu plánujeme riešenie vytvoriť tak, aby spolupracovalo s reálnymi mapovými podkladmi územia a simulovanie čo najvernejšie odrážalo reálnu situáciu priamo na mieste na ktorom je simulácia vykonávaná.

Popri spracovaní rôznych typov agentov poriadkových služieb a demonštrantov v návrhu plánujeme vytvoriť aj špeciálnu formu agenta - "lokalita demonštrácie", ktorá by umožňovala zachytiť aj špecifické vlastnosti daného miesta – povrch (dlažobné kocky, zámková dlažba), terénne nerovnosti (sochy, fontány, zeleň) mobiliár (smetné koše, lavičky, stojany na bicykle, smerové tabule) a pri simulácii by zohrávala spolu so statickými mapovými údajmi z OpenStreetMaps poskytujúcimi pohľad na miesto demonštrácie z perspektívy, ucelený a komplexný náhľad na miesto demonštrácie a potenciálne slabé stránky, na ktoré treba byť z pohľadu poriadkových zložiek pripravený.

V agentoch typu "lokalita" uvažujeme aj nad vytvorením agentov zachytávajúcich aj ďalšie aspekty miesta demonštrácie – obytná zóna, nákupná pasáž (sklenené výklady), reštauračné terasy (stoličky, stoly), ktoré by mohli byť pri demonštrácii v prípade eskalácie napätia použité.

Dôležitým pre správne simulovanie je aj zadefinovanie interakcií medzi agentmi rovnakého typu a ich správanie sa nielen ako jednotlivcov ale predovšetkým ako vzájomne sa dynamicky meniace zoskupenie.

Riešenie plánujeme implementovať v programovacom jazyku Java.

2.4 Personalizovaná TV

Televízia je súčasťou každodenného života mnohých ľudí a nemálo z nás pri nej strávi aj niekoľko hodín denne. Mnoho z používateľov sa však prehrabáva v zahlcujúcej ponuke, často si nevie vybrať alebo nezaregistruje program, ktorý by ich mohol baviť. Preto sme sa rozhodli viesť používateľa týmto bludiskom gumených tlačidiel a vytvoriť nástroj ponúkajúci mu obsah šitý na mieru podľa jeho životného štýlu a preferencií.

Televízne programy, tak ako ich poznáme v dnešnej podobe neposkytujú informácie o tom, čo chce divák, ale o tom, čo mu chce poskytnúť televízia. Prepínanie a volenie si obsahu je ponechávané iba na náhodné prepínanie, prípadne riadení sa nie vždy aktuálnym TV programom.

Našou snahou je zmeniť tento spôsob fungovania a postaviť používateľa do úlohy, kde si obsah *vyberá*, nie prepína. Poloha televízie – ako zariadenia sa v poslednej dobe čoraz viac odkláňa od pôvodného zámeru poskytovať obsah a stáva sa iba médiom na prezentovanie obsahu z rôznych zdrojov (domáce media centrá, streamovanie obsahu z počítačov).

Pri vypracovávaní tohto riešenia chceme využiť naše doterajšie skúsenosti s programovacími jazykmi, ako aj získať nové zručnosti a naučiť sa nové technológie. Veríme, že prínosom do tejto témy bude aj naše osobné nadšenie a zapálenie sa pre oblasť filmov, seriálov, ktorých sme dennodennými konzumentmi.

Problém prílišného trávenia času pred TV spôsobuje nemalé zdravotné problémy, ako mladí aktívni ľudia chceme používateľovi personálnej TV priniesť aj niečo, čo mu poskytne priestor pre relax aj mimo TV. Do nášho riešenia by sme preto chceli zapracovať popri odporúčaní priamo pre sledovanie TV aj odporúčania napr. na zájdenie si do kina na premietanie obľúbeného žánru filmu, prípadne na film s obľúbeným hercom, či prečítanie si recenzie o filme, ako priamo v prostredí aplikácie, tak aj v špecializovaných časopisoch.

Chceme vytvoriť TV, ktorá pozná svojich používateľov, vie, kto sú a rozpozná aký je ich obľúbený obsah a čo chcú sledovať.

2.4.1 Návrh riešenia

Pri vytváraní riešenia sa chceme zamerať na používateľa, jeho preferencie. Odporúčanie plánujeme založiť na analyzovaní jeho predchádzajúcich sledovaných programov a v prvotnej fáze najmä pomocou získania informácií priamo od používateľa prostredníctvom zadefinovania jeho obľúbených hercov, žánrov, atď.

Pre pohodlnejšie a interaktívnejšie získanie jeho preferencií by sme chceli vytvoriť mini hru, kde na základe typických scén z filmu, plagátu k filmu, postave herca, ktoré by boli zamaskované (rozostrené, inak upravené) by používateľ hádal a vyberal si z ponúknutých možností o čo sa jedná.
Odhadujeme pritom, že uhádnutie zamaskovaného obsahu by bolo rýchlejšie pri filme, hercovi, ktorý je používateľom preferovaný, čo by nám nemalou mierou naznačovalo záujem o film, herca, či žáner a pomohlo nám odporúčať obsah cielene.

Nemenej dôležitým pre riešenie je aj zachytávanie informácií o nových filmoch a seriáloch, získavanie informácií o ich obsahu. Plánujeme preto zapracovať do riešenia aj získavanie informácií z filmových webových databáz imdb.com, csfd.cz ktoré poskytujú pomerne dobrý zdroj informácií.

Systém bude pozostávať z časti serverovej, spracúvajúcej informácie potrebné pre optimalizovanie odporúčania a z časti klientskej, ktorá bude špecificky navrhnutá pre konkrétne zariadenie a platformu napr. smartfóny (rôzne operačné systémy), webová stránka, aplikácie pre inteligentné TV (SmartTV od Samsungu, atď.).

Príloha A - Poradie tém podľa preferencií:

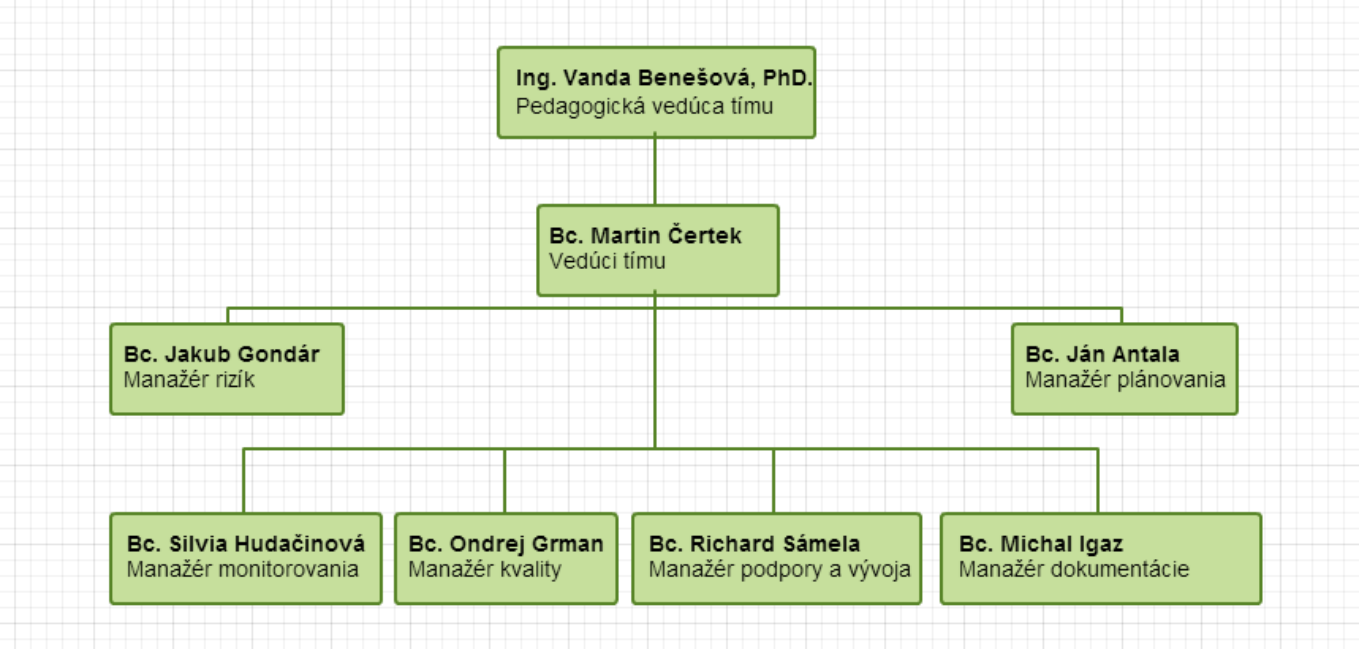
1. FIIT Kinect (KINECT)
2. Simulácia demonštrácie v meste (PROTEST SIM)
3. Odporúčanie pre inteligentnú TV (MY TV)
4. Inovatívna počítačová hra (GAME)
5. Odhaľovanie a hodnotenie vzťahov v oblasti vedy a výskumu (DIG LIB)
6. Offline Web (OFF-LINE WEB)
7. RoboCup – tretí rozmer (ROBOCUP)
8. Odhaľovanie emocionálneho stavu používateľa (EMOTION LOG)

3 Riadenie projektu a vývoja

Riadenie projektu je rozdelené na manažment projektu a manažment vývoja. Rozdelenie činností na manažmente projektu a vývoja je uvedené v diagramoch.

3.1 Manažment projektu

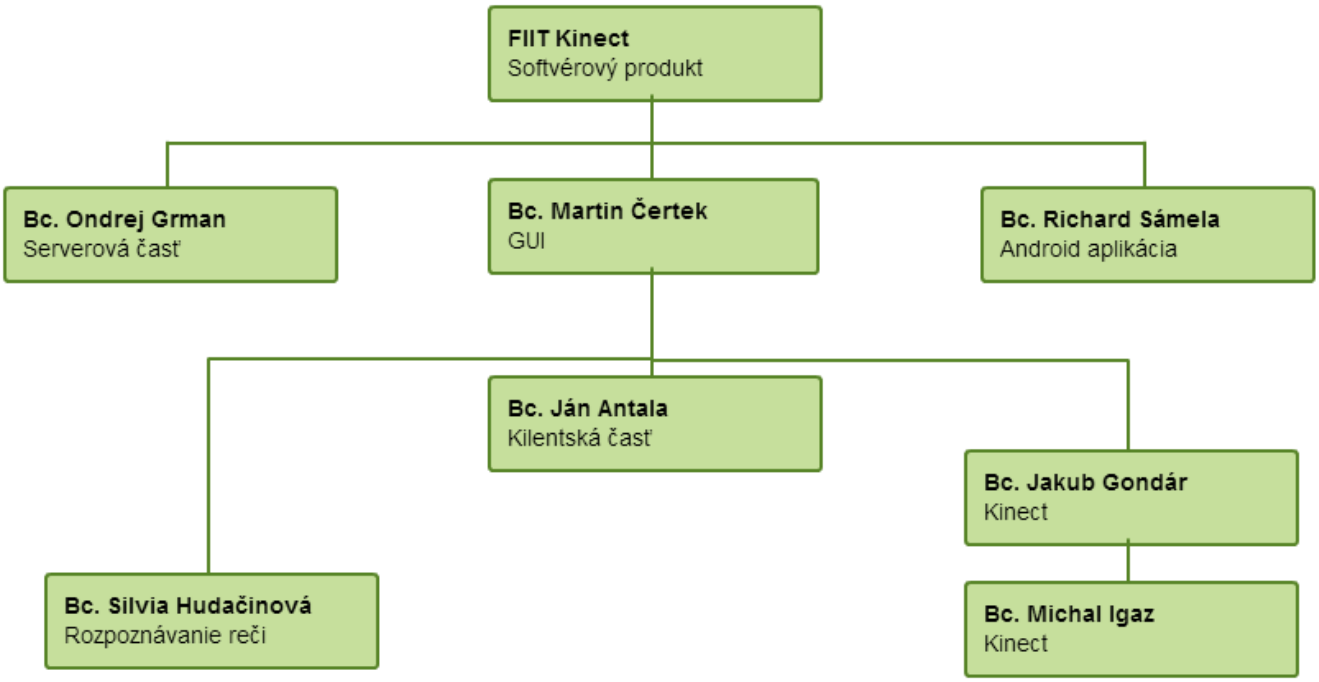
Manažment projektu pozostáva z manažmentu jednotlivých častí projektu. Pedagogický vedúci vystupuje v roli vlastníka produktu (v terminológií agilného vývoja – product owner). Za prácu celého tímu nesie zodpovednosť vedúci tímu. Ostatní členovia tímu sú zodpovední za jednu oblasť riadenia v závislosti od definovaného rozdelenia zodpovednosti. Rozdelenie zodpovedností a organizáciu je možné vidieť v diagrame na Obr. 2.



Obr. 2.: Manažment projektu.

3.2 Manažment vývoja

Zodpovednosti v rámci vývoja sú rozdelené na základe návrhu produktu a návrhu implementácie. Pre komplexnosť produktu a použitie rôznych technológií je manažment vývoja rozdelený na viaceré zložky. Diagram na Obr. 3 znázorňuje zodpovednosti členov tímu za jednotlivé zložky ako aj časti výsledného produktu.



Obr. 3.: Manažment vývoja.

4 Plánovanie

Dôležitou súčasťou úspešného projektu je jeho naplánovanie. Evidenciu plánu, jeho kontrolu realizujeme v prostredí systému Redmine. Nižšie je zachytené znázornenie plánu v spomenutom systéme.

Akceptacne testy	13.11.2012	20.11.2012
S2 - Analyza nalezitosti k projektovej dokumetacii a dokumentacii riadenia	29.10.2012	01.11.2012
Kupit sudok	28.10.2012	
Server - frontend	28.10.2012	13.11.2012
▸ S2 - Prevedenie GUI	28.10.2012	13.11.2012
▸ S2 - Mapovanie eventov na akcie, komunikacie s backendom	28.10.2012	13.11.2012
S1 - simulacia koncoveho zariadenia	17.10.2012	31.10.2012
S1 - simulacia posielania sprav	17.10.2012	31.10.2012
Server - backend	28.10.2012	14.12.2012
▸ S1 - prvý prototyp serveru		
▸ Pridanie pouzivatelskych roli	28.10.2012	
▸ Prijem videa	28.10.2012	
▸ Multipodnetove ovladanie	28.10.2012	
▸ Logovanie	28.10.2012	
▸ S2 - restful API pre klientsku cast	11.11.2012	
▸ GET experiments	11.11.2012	
DLNA Controller	28.10.2012	
▸ Logovanie	28.10.2012	
▸ Registracia na server	28.10.2012	
▸ Prijem akcii zo servera	28.10.2012	
▸ Vykonanie akcie	28.10.2012	
PC Controller	23.10.2012	30.10.2012
▸ Logovanie	28.10.2012	
▸ S1 - Registracia na server	23.10.2012	30.10.2012
▸ S1 - Prijem akcii zo servera	23.10.2012	30.10.2012
▸ S1 - Vykonanie akcie	23.10.2012	30.10.2012
Hudba Controller	28.10.2012	
▸ Logovanie	28.10.2012	
▸ Registracia na server	28.10.2012	
▸ Prijem akcii zo servera	28.10.2012	
▸ Vykonanie akcie	28.10.2012	
Svetla Controller	28.10.2012	
▸ Logovanie	28.10.2012	
▸ Registracia na server	28.10.2012	
▸ Prijem akcii zo servera	28.10.2012	
▸ Vykonanie akcie	28.10.2012	
AR.Drone Controller	28.10.2012	
▸ Logovanie	28.10.2012	
▸ Registracia na server	28.10.2012	
▸ Prijem akcii zo servera	28.10.2012	
▸ Vykonanie akcie	28.10.2012	
TV Controller	28.10.2012	

▸ Logovanie	28.10.2012	
▸ Rozpoznanie vstupov na eventy	28.10.2012	
▸ Navrh GUI	28.10.2012	
▸ Prevedenie GUI	28.10.2012	
▸ GUI - Pridanie web view	28.10.2012	
▸ Registracia na server	28.10.2012	
▸ Posielanie rozpoznanych eventov na server	28.10.2012	
▸ Upload videa na server	28.10.2012	
Kinect App	18.10.2012	13.11.2012
▸ S2 - Rozpoznanie pohybu tela	28.10.2012	13.11.2012
▸ S2 - Rozpoznanie zvuku (SAV softver)	28.10.2012	13.11.2012
▸ Kontinualne Ovladanie	28.10.2012	
▸ Pouzivatelske role	28.10.2012	
▸ Logovanie	28.10.2012	
▸ Rozpoznanie vstupov na eventy	28.10.2012	
▸ S1 - analýza GUI existujúceho riešenia, návrh a prototyp nového GUI pre časť Kinect	18.10.2012	22.10.2012
▸ Prevedenie GUI	28.10.2012	
▸ GUI - Pridanie web view	28.10.2012	
▸ Registracia na server	28.10.2012	
▸ S2 - Posielanie rozpoznanych eventov na server	28.10.2012	13.11.2012
▸ Upload videa na server	28.10.2012	
S1 - vypracovanie user stories	24.10.2012	30.10.2012
S1 - zapisnica zo stretnutia 3	24.10.2012	25.10.2012
S2 - Dokumentácia odovzdanie	27.10.2012	10.11.2012
▸ S1 - Manazment kvality	27.10.2012	10.11.2012
▸ S1 - Manazment rizik	27.10.2012	10.11.2012
▸ S1 - Manazment rozvrhu a planovania	27.10.2012	10.11.2012
▸ S1 - Monitorovanie projektu	27.10.2012	10.11.2012
▸ S1 - Manazment komunikacie	27.10.2012	10.11.2012
▸ S1 - Manazment dokumentacie	27.10.2012	10.11.2012
Prihlaska TP Cup	20.10.2012	26.11.2012
S1 - analýza požidaviek, špecifikacia	18.10.2012	04.11.2012
S1 - naštudovanie architekrúry, súčasného stavu kódu	21.10.2012	29.10.2012
▸ Inštalácia knižníc do VS2010	21.10.2012	29.10.2012
S1 - analýza riešení pre rozpoznávanie reči	18.10.2012	22.10.2012
Grant E-TALENT	16.10.2012	30.11.2012

5 Manažment komunikácie

Autor: Martin Čertek

Komunikácia medzi jednotlivými členmi tímu, ako aj tímu navonok je rozdelená do dvoch kategórii na formálnu (oficiálnu komunikáciu) a neformálnu komunikáciu (najmä medzi jednotlivými časťami vývojového tímu pracujúceho na spoločnej časti).

5.1 Formálna komunikácia

5.1.1 E-mailová komunikácia

Tímová adresa pre kontaktovanie je : tim2_fiit@googlegroups.com , kontaktovaním tímu na tejto adrese je zabezpečené poslanie mailu všetkým členom tímu súčasne. Táto adresa je používaná predovšetkým na:

- Dôležité veci ohľadne projektu
    - Architektúra, filozofia projektu
    - Metodiky, návody ako postupovať pri riešení problémov, inštaláciách
- Monitorovanie a oboznamovanie s termínmi, ich pripomínanie
- Informovanie o úlohách zaradených do daného šprintu s informovaním o pridelených aktivitách v systéme Redmine
- Potvrdenie oficiálneho odovzdania
- Zasielanie dôležitých a hodnotných informácií zo strany pedagogickej vedúcej tímu

Okrem komunikácie určenej všetkým členom tímu na základe požiadaviek prameniacich z vývoja je používaný aj spôsob komunikovania v menších vývojových tímoch spolupracujúcich spoločne na istej časti riešenia, produktu.
Pri mailovej komunikácii platí dohodnutá konvencia, kde ak je na začiatku dokumentu uvedený:

- Príznak [FYI] (z angl. For Your Information) – správa na informatívnu hodnotu, nie je potrebné reagovať na jej obsah priamou odpoveďou, pokladá za to, že čitateľ sa so správou iba oboznámil.
- Príznak [ACK] – správa z nutnosťou reakcie podľa požiadaviek, adresát má povinnosť po prečítaní reagovať a komunikovať v správe vyžiadané informácie

5.1.2 Komunikácia prostredníctvom zadávania úloh do systému Redmine

Pri bežnej práci na produkte je ako informačný kanál používaný aj systém Redmine, kde sú zhromažďované informácie o projekte (wiki, dokumenty, súbory). V záložke *wiki* sa nachádzajú všeobecné informácie nápomocné pri vývoji produktu.

Systéme slúžia aj na informovanie jednotlivých členov tímu o im priradených úlohách pri vývoji, kde sa pri priradenej úlohe nachádzajú aj presnejšie špecifikujúce informácie o náplni úlohy, o predpokladanej dobe vypracovania a čase, ktorý bol na riešení úlohy vynaložený.

Systém poskytuje popri priamej komunikácií aj možnosti s pridanou výpovednou hodnotou, ktoré pomocou grafov prinášajú informáciu o aktuálnom stave vytváraného projektu vo vizuálnej forme.

5.2 Neformálna komunikácia

Popri formálnej komunikácii tímu je pri vývoji dôležitou súčasťou komunikácia neformálna. Takáto neformálna komunikácia je realizovaná prostredníctvom IM nástrojov, predovšetkým prostredníctvom programu Skype, prípadne Google Talk. Používaná bude výhradne pri neformálnej komunikácii medzi vývojármi, ktorí spolupracujú na určitej prepojenej časti riešenia, prípadne pri vysvetľovaní zdrojových kódov, komentárov.
Cieľom je dosiahnuť stav, aby z tímu spolu komunikovali iba členovia, ktorí majú k danej oblasti čo povedať (rozdelenie tímu na skupiny: GUI, Kinect, softvérová časť, Android aplikácia). Komunikovanie týmto spôsobom nevyžaduje používanie formálnych náležitostí.

5.3 Tvorba používateľskej príručky

Dokument spracúva metodiku tvorby používateľskej príručky v prostredí dokumentácie v rámci vytvárania softvérového diela. Je určený členom tímu podieľajúcim sa na vytváraní a zdokumentovaní produktu. Ak nie je uvedené inak, všetky metodické pokyny, na ktoré sa v dokumente odvoláva sú interné predpisy spoločnosti, platí vždy najaktuálnejšia verzia daného dokumentu, ktorý je aplikovaný. Proces tvorby používateľskej príručky je zachytený v kontexte manažmentu zberu požiadaviek. Jej obsah je rozdelený na dve časti. Prvá sa venuje všeobecnému opisu metodiky spracovania používateľskej príručky, druhá popisuje konkrétnu činnosť pri využívaní tejto metodiky na vytvorenie používateľskej príručky softvérového riešenia ovládania multimediálnych zariadení prostredníctvom gest zaznamenávaných pomocou senzoru Kinect a inteligentných telefónov.

Zoznam súvisiacich metodík:

Metodika analýzy požiadaviek od klienta

Metodika práce v systéme RedMine

Metodika používania preberacích protokolov softvérového produktu

Metodika jazykových mutácií dokumentácii

5.3.1 Slovník pojmov a skratiek

Názov pojmu	Skratka	Popis, význam
RedMine		Systém podporujúci spoluprácu pri tímovom vývoji
Tester		Osoba zodpovedná za testovanie
Kinect		Senzor detegujúci pohyb v 3D
Textový procesor Micro-soft Word 2010	MS WORD 2010	Nástroj na tvorbu text, textovej dokumentácie

Tab. 1.: Slovník pojmov.

5.3.2 Manažment zberu požiadaviek

5.3.2.1 Používateľské roly

Rola	Zodpovednosť
Editor používateľskej príručky	Zodpovedný za zostavenie príručky
Manažér dokumentácie	Zodpovedný za všetku dokumentáciu k projektu a produktu
Programátor	Tvorca softvérového riešenia zodpovedný za vypracovanie podľa zákazníckych požiadaviek
Dokumentarista	Tvorca dokumentácie
Tester	Zodpovedný za overovanie funkcionality produktu, overovateľ napísaného postupu práce

Tab. 2.: Používateľské roly.

5.3.2.2 Proces získania požiadaviek na používateľskú príručku

Úlohou toho procesu je získať od klienta požiadavky na vypracovanie používateľskej dokumentácie podľa jeho preferencií.

	Krok	Kapitola
1.	Získanie požiadaviek na použ. príručku od klienta	3.1
2.	Spracovanie požiadaviek od klienta	3.2

Tab. 3.: Postupnosť krokov pri získavaní požiadaviek.

5.3.2.3 Proces zmeny požiadaviek na používateľskú príručku

Proces zachytáva požiadavku klienta na zmenu použ. príručky. Každá elementárna zmena na použ. príručku od klienta musí byť zaznamenaná, komunikovaná s klientom a posunutá na zapracovanie do dokumentu používateľskej príručky.

	Krok	Kapitola
1.	Prijatie požiadaviek na zmenu od klienta	3.3
2.	Zadefinovanie požadovaných zmien od klienta	3.4
3.	Zapracovanie zmien do použ. príručky	3.5

Tab. 4.: Postupnosť krokov pri zmene požiadaviek.

5.3.2.4 Proces priameho vytvárania používateľskej príručky

Proces spracúva priamy postup pri písaní dokumentu používateľskej príručky. Proces definuje formálne požiadavky na dokument a náležitosti, ktoré musí použ. príručka nevyhnutne spĺňať.

	Krok	Kapitola
1.	Definovanie formálnej štruktúry použ. príručky - dokumentu	3.6

Tab. 5.: Postupnosť krokov pri priamom písaní dokumentu.

5.3.2.5 Proces overenia používateľskej príručky

Po vykonaní každej iterácie práce na používateľskej príručke a vykonanej zmene je nevyhnutné overiť stav používateľskej príručky po daných úpravách.

	Krok	Kapitola
1.	Skontrolovanie dokumentu	3.7

Tab. 6.: Postupnosť krokov pri overovaní používateľskej príručky.

5.3.2.6 Proces ukončenia tvorby používateľskej príručky

Pred nasadeným produktu spolu s používateľskou príručkou je nutné vykonať záverečný postup fázy tvorby používateľskej príručky. Vykonanie tohto procesu je povinné.

	Krok	Kapitola
1.	Finalizácia dokumentu používateľskej príručky	3.8
2.	Odovzdanie používateľskej príručky klientovi	3.9

Tab. 7.: Postupnosť krokov pri ukončovaní tvorby používateľskej príručky.

5.3.3 Postup pri vypracovaní používateľskej príručky softvérového produktu

5.3.3.1 Získanie požiadaviek na použ. príručku od klienta

Vstup	Kontaktovanie klienta na účelom získania požiadaviek o použ. príručke
Výstup	Zozbieraná množina požiadaviek do klienta
Zodpovedný	Manažér dokumentácie

Účel	Špecifikovanie požiadaviek klienta na používateľskú príručku so zameraním sa na zachytenie preferencií klienta o prioritách v spracovaní dokumentu.

Postup:

1. Manažér dokumentácie osloví písomnou formou zástupcu klienta definovaného podľa zmluvy so žiadosťou na predloženie požiadaviek týkajúcich sa použ. príručky.
2. Na základe prijatého písomného dokumentu zostaví požiadavky klienta na príručku.

   V požiadavke musí byť definované:

   - akým spôsobom má byť realizovaná
   - aký je ďalší postup pri zmene požiadavky
   - časový rámec reportovania o požiadavke

3. Zozbierané požiadavky zašle editorovi používateľskej príručky pridelenému ku konkrétnemu projektu - produktu.

5.3.3.2 Spracovanie požiadaviek od klienta

Vstup	Zozbieraná množina požiadaviek do klienta
Výstup	Spracovaná množina požiadaviek od klienta
Zodpovedný	Manažér dokumentácie

Účel	Stanovanie postupu pri spracovaní požiadavky od klienta

Postup:

1. Manažér dokumentácie roztriedi požiadavky od klienta podľa štruktúry, ktorú definuje – *Metodika analýzy požiadaviek od klienta.*
2. Spracovanú množinu požiadaviek uloží vo firemnom systéme RedMine v danom projekte do záložky *Súbory.*
3. V systéme RedMine podľa metodiky – *Metodika práce v systéme RedMine* vytvorí úlohu pre editora používateľskej príručky.

5.3.3.3 Prijatie požiadaviek na zmenu od klienta

Vstup	Prijatá požiadavka od klienta
Výstup	Zaznamenaná žiadosť od klienta
Zodpovedný	Manažér dokumentácie

Účel	Stanovanie postupu pri prijatí požiadavky od klienta

Postup:

1. Manažér dokumentácie prijme písomnú požiadavku od klienta.
2. Zaeviduje prijatú písomnosť a oboznámi editora používateľskej príručky s prijatou skutočnosťou.

5.3.3.4 Zadefinovanie požadovaných zmien od klienta

Vstup	Zaznamenaná žiadosť od klienta
Výstup	Štruktúrovane spracovaná a zadefinovaná požiadavka
Zodpovedný	Editor používateľskej príručky

Účel	Stanovanie postupu pri komunikácií s klientom o zmene požiadavky

Postup:

1. Editor používateľskej príručky kontaktuje písomne zodpovednú osobu klienta za účelom vyjasnenia žiadosti klienta.
2. Podľa analýzy a komunikácie s klientom zostaví štruktúrovane spracovanú žiadosť o zmenu a zadefinuje požiadavku podľa - *Metodika analýzy požiadaviek od klienta.*

5.3.3.5 Zapracovanie zmien do použ. príručky

Vstup	Zadefinovaná žiadosť o zmenu
Výstup	Upravená použ. príručka podľa požiadavky
Zodpovedný	Editor používateľskej dokumentácie

Účel	Definovanie postupu pri zapracovávaní zmeny do používateľskej príručky zo strany klienta

Postup:

1. Editor používateľskej príručky pridelí dokumentaristovi úlohu zapracovať zmenu použ. príručky.
2. Dokumentarista na základe definovaného formátu požiadavky upraví tú časť použ. príručky, ktorá je ovplyvnená žiadosťou o zmenu.
3. Vykonanie zmeny reportuje testerovi, ktorý vykoná proces overenia dokumentu – *Proces 3.7 Skontrolovanie dokumentu.*

5.3.3.6 Definovanie formálnej štruktúry použ. príručky - dokumentu

Vstup	Množina požiadaviek klienta na použ. príručku, materiály o produkte od programátorov a testerov
Výstup	Štruktúrovane spracovaná používateľská príručka
Zodpovedný	Dokumentarista

Účel	Stanovanie postupu pri štruktúre písania textu použ. príručky

Postup:

1. Dokumentarista – pisateľ textu použ. príručky podľa požiadaviek klienta a podkladových materiálov z procesu vývoja a testovania produktu spíše text znenia použ. príručky.
2. Samotné úpravu textu spracuje podľa kapitoly 4 tejto metodiky - *Spracovanie používateľskej príručky.*

5.3.3.7 Skontrolovanie dokumentu

Vstup	Dokument použ. príručky po vykonaní zmeny
Výstup	Skontrolovaný dokument
Zodpovedný	Tester

Účel	Postup stanovuje spôsob a zodpovednosť pri overovaní dokumentu – použ. príručky

Postup:

1. Tester prevezme na spracovanie požiadavku od editora používateľskej príručky.
2. Overí konzistenciu – obsahovú, formátovú podľa definovaného štandardu - *Spracovanie používateľskej príručky - kapitola 4 tohto dokumentu.*
3. Ak v danom dokumente nie sú objavené odchýlky od normy, dokument uloží do systému správy dokumentov a potvrdí jeho kvalitu.
4. Ak dokument vykazuje známky odchýlenia sa od štandardu, oznámi to editorovi používateľskej dokumentácie.

5.3.3.8 Finalizácia dokumentu použ. príručky

Vstup	Overená verzia dokumentu použ. príručky
Výstup	Finálna verzia dokumentu použ. príručky
Zodpovedný	Editor používateľskej dokumentácie

Účel	Spracovane postupu vytvorenie finálnej verzie používateľskej príručky

Postup:

1. Editor použ. dokumentácie na základe neexistencie ďalších požiadaviek na zmenu použ. príručky a v súlade s časovým harmonogramom tvorby daného softvérového projektu potvrdí danú verziu používateľskej dokumentácie za finálnu.
2. Reportuje manažérovi dokumentácie dosiahnutie stavu finálnej verzie dokumentu pripravenej na odovzdanie.
3. Editor ukonči proces tvorby používateľskej príručky až do prijatia podnetu na prípadnú zmenu, opravu dokumentu.

5.3.3.9 Odovzdanie používateľskej príručky klientovi

Vstup	Overená a finálna verzia dokumentu použ. príručky pripravená na odovzdanie
Výstup	Odovzdaná použ. príručka s podpísaným preberacím protokolom
Zodpovedný	Manažér dokumentácie

Účel	Odovzdanie dokumentu používateľskej príručky klientovi po overení a vypracovanie preberacieho protokolu o odovzdaní dokumentu

Postup:

1. Manažér dokumentácie vyzve písomnou formou pred termínom stanoveného odovzdávania dokumentácie na zúčastnenie sa na preberacom konaní k dokumentu.
2. Manažér dokumentácie zašle verziu použ. príručky aktuálne považovanú za finálnu zodpovednej osobe zo strany klienta.
3. Ak klient uvedie výhrady k danému dokumentu, manažér dokumentácie iniciuje proces - *3.4 Zadefinovanie požadovaných zmien od klienta s klientom.*
4. Ak klient neuvedie výhrady, podpíšu zodpovedné osoby za stranu tvorcu používateľskej príručky a za stranu klienta dokument – *Protokol o prebratí používateľskej príručky*, s uvedením finálnej verzie príručky a elektronickou prílohou – samotnou Používateľskou príručkou na CD nosiči. Protokol má písomnú formu a je vypracovaný podľa – *Metodika používania preberacích protokolov softvérového produktu.*

5.3.4 Spracovanie používateľskej príručky

5.3.4.1 Obsahová štruktúra používateľskej príručky

Vymedzenie obsahu používateľskej príručky

Príručka musí byť z obsahovej stránky rozdelená na:

- Úvod – popis účelu dokumentu a oboznámenie čitateľa s popisom produktu
- Hlavná časť – popis jednotlivých funkcionalít produktu spolu s presne definovaným postupom, ako použiť zložky produktu
- Prílohy – súčasť dokumentu so samostatnou ucelenou hodnotou (obrazová, textová)
- Obsah – zoznam použitých materiálov pri odvolávkach, vymenovanie autorov používateľskej príručky
- Kontakt – údaje o firme vypracujúcej projekt a použ. príručku

Spracovanie použitia konkrétnej funkcionality produktu

Pri spracovaní konkrétnej funkcionality je potrebné uviesť:

1. Čo funkcionalita prináša používateľovi
2. Aké klientske požiadavky na produkt funkcionalita rieši
3. Postup pri použití tejto časti produktu
4. Usmernenia v prípade nastania neočakávanej situácie, postup pri nesplnení očakávanej funkcionality

1. Popísať v rozsahu max 100 slov funkcionalitu - poskytovanú množinu výstupov, aké možnosti poskytuje používateľovi produktu, v akej situácií sa daná funkcionalita používa, pri akých obmedzeniach sú je vlastnosti zmenené.

2. Zdôraznenie prínosu funkcionality z pohľadu naplnenia požiadaviek klienta.

3. Jasný, postupný, krokový návod ako pracovať s funkcionalitou – použiť konkrétne, číslované, po sebe nasledujúce podrobne popísané kroky s elementárnou vykonateľnosťou. Aplikuje sa číslovanie krokov návodu vo formáte :

1. Krok 1
2. Krok 2
3. ....

4. Riešenie vzniknutých neštandardných situácií sa uvádza pri danej funkcionalite po postupe objasňujúcom použitie funkcionality. Tento bod zahŕňa ošetrenie v prípade, ak:

- sa časť aplikácie, z ktorou klient pracuje nespráva podľa definovanej funkcionality
- funkcionalita, s ktorou sa pracuje nereaguje podľa postupu uvedeného v návode

Riešenie vzniknutých neštandardných situácií

Na základe zozbieranej množiny poznatkov získaných pri testoch používania produktu sa uvedú najčastejšie zlyhania a chybné používanie časti produktu vo formáte:

Používateľom zachytená chyba	**Možné riešenie**
Nefunguje pridanie nového projektu	Uistite sa, že máte dostatok voľného miest na disku pre vytvorenie nového projektu

Vzor – popis práce s produktom – príloha A

5.3.5 Definovanie úpravy dokumentu používateľskej príručky

5.3.5.1 Definícia štýlov

Používateľská príručka predstavuje ucelený, prehľadný dokument členený na kapitoly. Ďalšie delenie v rámci kapitol je založené na logickom rozčlenení obsahu kapitoly. Definovanie štýlov je realizované pre textový procesor Word 2010 za použitia štandardného štýlu:

Logický formát textu	**Použité formátovanie**
Kapitola	Nadpis 1
Podkapitola prvej úrovne	Nadpis 2
Podkapitola druhej úrovne	Nadpis 3
Podkapitola štvrtej úrovne	Nadpis 4
Bežný text	Font Calibri, veľkosť písma 12
Zvýraznený text	Font Calibri, veľkosť písma 12, italic
Hrubý text	Font Calibri, veľkosť písma 12, bold

Tab. 8.: Pravidlá použitia štýlov.

Použitie obrázkov, tabuliek v príručke

Vkladanie popisu obrázku, tabuľky

Pri vkladaní obrázkov je nutné daný obrázok vložiť v rozlíšení minimálne 800x600 pixlov vo formáte JPEG, PNG s centrovaním na stred strany. Vložení obrázok je slovne popísaný a očíslovaný na spodnej strane obrázka, vycentrovaní na stred obrázka s uvedením formátu:

Obr. <poradové číslo >, Slovný popis.

Poradové číslo obrázka je uvádzané v tvare <číslo kapitoly>.<číslo obrázka v kapitole>, „Slovný popis“ obrázka začína s veľkým písmenom a musí byť dlhý maximálne na šírku obrázka.

Rovnaký postup je aplikovaný v prípade práce s tabuľkami. Vložená tabuľka je vo formáte, centrovaná na stred, s číselným označením tabuľky a slovným popisom ako v prípade vkladania obrázku.

Tabuľka <poradové číslo >, Slovný popis.

Formát tabuľky

Tabuľka vložená do príručky je centrovaná na stred dokumentu, je použité jednoduché orámovanie, preddefinované v MS WORD 2010, hlavička tabuľky je farebne zvýraznená použitím – *ofarbenie biela, Pozadie1, tmavšia 15%.* Na text v hlavičke tabuľky je aplikovaný štýl – *hrubý text* , definovaný v tabuľke 8 tejto metodiky.

Dodržanie kontinuity jazyka

Text jednotlivých kapitol a odsekov je založený na rovnakom, vopred definovanom slovníku pojmov. Používanie rôznych pomenovaní na určenie tej istej veci je neprípustné. Rovnako je potrebné vyhnúť sa používaniu termínov a výrazov v cudzích jazykoch. Ak použitý výraz nemá slovenský ekvivalent v slovníku pojmov a skratiek sa uvedie daný výraz spolu s referenciou na zdroj opisujúci použitý termín.

Jazyková norma textu

Text používateľskej príručky je v slovenskom jazyku, riadi sa podľa platných pravidiel slovenského pravopisu. Pri písaní je použitá v prípade odvolávky na dokument 3.osoba jednotného čísla, aplikovaný ženský rod:
Príručka poskytuje ...

Pri písaní postupu sa používa prítomný čas, pri oslovovaní 2.osoba množného čísla:

Stlačte tlačidlo ...

Jazykové verzie použ. príručky

Vypracovanie používateľskej príručky v inej jazykovej mutácií sa riadi pomocou – *Metodika jazykových mutácií dokumentácii*.

Príloha A – Vzor popisu práce s produktom

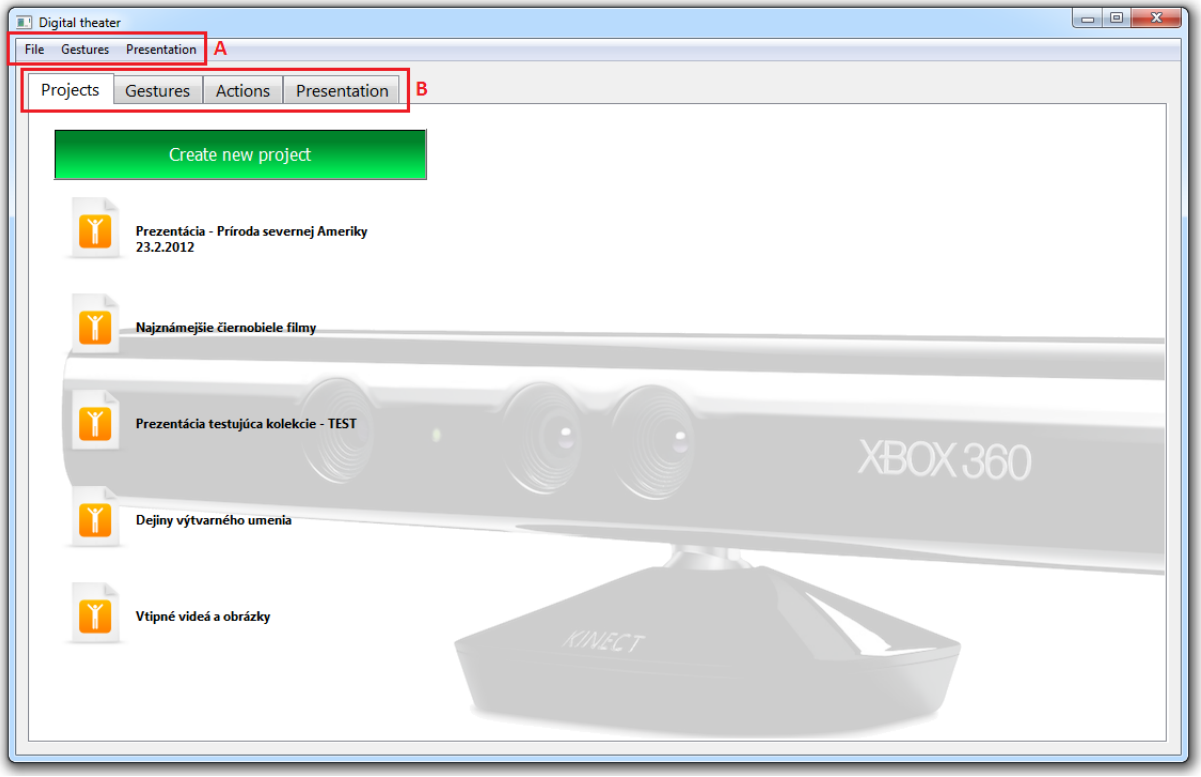
Spracované podľa používateľskej príručky - Tím: Art Quintet (č. 11), Digitálne divadlo

5.3.6 Projects (úvodná obrazovka)

V úvodnej obrazovke začína tvorba projektu. Obrazovka (Obr. 1) obsahuje tlačidlo *Create new project* pre vytvorenie nového projektu, a tiež zoznam posledných projektov, ktoré boli v programe otvorené.

5.3.6.1 Vytvorenie nového projektu:

1. Kliknite na tlačidlo *Create new project*.
2. V obrazovke sa zjaví textové pole- *Name*, do ktorej zadajte názov vášho projektu. Pod týmto názvom bude neskôr projekt uložený aj na disku.
3. Stlačte *Continue in presentation*.
4. Aplikácia Vás prepne do záložky *Gestures*, kde pokračujete vo vytváraní projektu (viď. podkapitola Gestures).



Obr. 2.1, Použitie aplikácie

6 Manažment kvality

Autor: Ondrej Grman

6.1 Konvencia písania kódu

Pri vývoji riešenia nadväzujeme na minuloročný tímový projekt Digitálne divadlo. Z dôvodu zachovania konzistentnosti sa vývoj v časti programovania v C++ riadi metodikou písania zdrojového kódu z projektu, na ktorý nadväzujeme a jeho časť používame.

Pri riešení projektu používame viacero programovacích jazykov a platforiem – javascript, aplikácia pre Android zariadenia. Takto vytváraný kód sa riadi príslušnými konvenciami pre daný jazyk, tak ako je to pre jazyk definované.

6.2 Testovanie

Účelom tohto dokumentu je oboznámenie sa s internými pravidlami testovania. Dokument je určený prevažne novým členom tímu, ale aj ako pomôcka stabilným členom zabezpečujúcim testovanie produktu. Obsahuje vymedzenie pojmov a metodické pravidlá pre riadenie testovania počas vývoja projektu.

6.3 Súvisiace dokumenty

[1] Metodika tvorby technických dokumentácií
[2] Metodika reportovania chýb
[3] Návod k tvorbe aplikácií v node.js
[4] Metodika písania zdrojového kódu
[5] Návod na inštaláciu a používanie testovacích nástrojov

6.4 Slovník pojmov

- unit test - časť programu, ktorá zabezpečuje otestovanie jednej metódy, funkcie systému
- akceptačný test – otestovanie jednej ucelenej funkcionality systému, prípadu použitia
- slovo „správny“ – znamená spĺňajúci koncept dohodnutý tímom (prípadne zástupcom manažmentu) pre konkrétny projekt (skupinu projektov)
- screenshot – statický obrázok obrazovky (monitora) zachytávajúci stav aplikácie v určitom čase
- black box testovanie – tester vie ako sa program ovláda, čo robí, aké ma vstupy a výstupy, ale nevie ako pracuje- nevidí do kódu

6.5 Zodpovednosti členov tímu v procese testovania

6.5.1 Manažér testovania

- vedie proces testovania
- kontroluje, či sa dodržujú stanovené postupy v procese testovania
- má prehľad o všetkých testoch, nariaďuje aké konkrétne testy kto vykonáva
- vypracováva metodické materiály
- vyberá nástroje použité pri testovaní
- väčšinou je to osoba zastávajúca manažment kvality

6.5.2 Návrhár – architekt systému

- vytvára akceptačné testy
- dohaduje so zákazníkom, kedy bude softvér dostatočne otestovaný aby bol akceptovaný a prebratý
- Bc. Michal Igaz, Bc. Martin Čertek

6.5.3 Tvorca unit testov

- v procese vývoja produktu vytvára unit testy (jednotkové testy)
- vo väčšine prípadov je to samotný vývojár, ale môže to byť aj osoba určená výhradne na túto činnosť, ktorá úzko spolupracuje s vývojárom- túto kompetenciu určuje manažér testovania
- unit testy tvoria programátori paralelne s vývojom softvérového produktu ak je možné tieto testy vytvárať, používajú na to dostupné nástroje daného vývojového prostredia

6.5.4 Tester

- je plnohodnotný člen tímu
- musí to byť fyzicky iná osoba ako tá ktorá testy vytvorila alebo inak sa podieľala na vývoji produktu
- uzatvára vyriešené úlohy ak testy prebehnú úspešne
- na konci testovania vypracováva dokument obsahujúci vyhodnotenie testovania v podobe určenej Metodikou tvorby technických dokumentácií [1]
- úlohu testera zastáva Bc. Michal Igaz a pokrýva všetky typy testerov opísaných nižšie (nevylučuje sa priradenie ďalších členov testovacieho tímu)

6.5.4.1 Tester databázy

- overuje či sa do databázy vkladajú správne dáta, v správnom formáte, konzistentné, na správne miesto
- prípadné chyby zaznamenáva v podobe určenej Metodikou reportovania chýb [2]

6.5.4.2 Unit tester

- vykonáva (spúšťa) unit testy vytvorené v systéme s dosadzovaním rôznych sád vstupov a výstupov
- prípadné chyby zaznamenáva v podobe určenej Metodikou reportovania chýb [2]

6.5.4.3 Hlavný tester

- jeho práca začína až keď testeri na zvyšných dvoch pozíciách ukončili testovanie s úspešným výsledkom
- vykonáva náhodné testovanie funkčnosti systému podľa vlastného uváženia
- jeho cieľom je systém pokaziť, zhodiť
- využíva sa black box testovanie
- vykonáva akceptačné testy
- prípadné chyby zaznamenáva v podobe určenej Metodikou reportovania chýb [2]

6.6 Procesy

Proces	Fáza	Kapitola
Príprava testovania	Návrh	5.1
Priebeh a dokumentovanie testovania	Vývoj a implementácia	5.2

6.6.1 Príprava testovania

Je to proces, ktorý predchádza samotné testovanie. Pred začatím prác na projekte či testovaní je nutné mať pripravené prostredia a nakonfigurované nástroje.

	Krok	Kapitola
1.	Konfigurácia testovacích nástrojov a vytvorenie testovacích projektov	5.2.1
2.	Vytvorenie testov a ich nastavenie	5.2.2
3.	Uloženie testovacieho projektu	5.2.3

6.6.1.1 Konfigurácia testovacích nástrojov a vytvorenie testovacích projektov

Vstupy	určené zodpovednosti a použité nástroje
Výstupy	správne nakonfigurované testovacie nástroje, vytvorené testovacie projekty
Zodpovedný	manažér testovania/ vedúci projektu/ programátor (v závislosti od projektu určí manažér testovania)
Popis	- *v závislosti od projektu a použitého programovacieho jazyka sa použije nástroj (doplnok, plugin) na tvorbu testov* - *určené nástroje sa nakonfigurujú podľa dostupných návodov* - *vytvoria sa testovacie projekty a adresáre*

6.6.1.2 Vytvorenie testov a ich nastavenie

Vstupy	nakonfigurované testovacie nástroje a moduly
Výstupy	hotové testy
Zodpovedný	tvorca testov, návrhár príp. iná osoba určená manažérom testovania
Popis	- *tvorca unit testov vytvorí jednotlivé testy funkcionality systému v určenom nástroji* - *návrhár vytvorí akceptačné testy, čo je proces spojený s konzultáciou so zákazníkom – zahrnuté sú pripomienky zákazníka a prípady použitia* ○ *v tejto fáze je zákazník zároveň upovedomený o tom, že tieto testy sú dôkazom správnosti softvérového produktu a po ich úspešnom prebehnutí preberá produkt*

6.6.1.3 Uloženie testovacích projektov

Vstupy	hotové testovacie projekty, sady testov
Výstupy	uložené projekty
Zodpovedný	manažér testovania
Popis	- *po predošlých krokoch je nutné testovacie projekty uložiť k prislúchajúcim softvérovým projektom a zapracovať do verzovacieho systému*

6.6.2 Priebeh a dokumentovanie testovania

Je to proces, ktorý sa pravidelne opakuje počas celého vývoja systému.

	Krok	Kapitola
1.	Testovanie	5.3.1
2.	Vyhodnotenie testovania	5.3.2
3.	Používanie podporného nástroja na zaznamenávanie testu	5.3.3

6.6.2.1 Testovanie

Vstupy	testovacie projekty, funkčné testy
Výstupy	výsledky testov
Zodpovedný	tester
Popis	- *samotné testovanie prebieha rôzne v závislosti na type testu (konkrétny príklad je uvedený v kapitole 6)* - *testeri dosadzujú vytvorené sady vstupov a porovnávajú ich s výstupmi, náhodne kontrolujú funkcionalitu systému alebo inak testujú projekt*

6.6.2.2 Vyhodnotenie testovania

Vstupy	výsledky testov
Výstupy	vyhodnotenie testov
Zodpovedný	tester
Popis	- *na konci testovania je tester povinný vyhodnotiť a zdokumentovať priebeh a výsledky testov* - *v prípade chýb sa riadi Metodikou reportovania chýb [2]* - *v prípade chýb aj úspechu spíše dokumentáciu podľa Metodiky tvorby technických dokumentácií [1]*

6.6.2.3 Používanie podporného nástroja na zaznamenávanie testu

Vstupy	vyriešené úlohy čakajúce na testovanie
Výstupy	uzavreté úlohy alebo znovu pridelené úlohy
Zodpovedný	tester
Popis	- *nástroj používaný na podporu riadenia projektu je redmine, ktorý nutne využívajú všetky tímy a tomu je podriadený aj manažment testovania* - *testeri si vyberajú na testovanie úlohy v stave „resolved“* - *po úspešnom ukončení testov označia danú úlohu ako „closed“ a priradia k nej spísanú dokumentáciu testovania* - *v prípade chýb sa riadia Metodikou reportovania chýb [2] a úlohy označujú naspať do stavu „assigned“*

6.7 Asynchrónne unit testovanie servra s REST API v node.js

Predpokladom je vytvorený projekt (server) v node.js, ktorý je funkčný a máme prístup k zdrojovým kódom. Vytváranie aplikácie v node.js nie je predmetom tejto metodiky, nakoľko sa tomu venuje Návod k tvorbe aplikácií v node.js [3] spolu s Metodikou písania zdrojového kódu [4]

6.7.1 Inštalácia testovacieho modulu a knižnice

- nainštalujeme moduly *nodeunit* a *nodeunit-httpclient* podľa Návodu na inštaláciu a používanie testovacích nástrojov [5]
- v projektovom adresári vytvoríme podadresár s názvom *test*.

6.7.2 Vytváranie testu

Každý tvorca testov sa riadi týmito pravidlami.

6.7.2.1 Základné pravidlá

Pri tvorbe testov je nutné dodržiavať nasledovné pravidlá:

1. Každý súbor s testami sa nachádza v adresári *test*.
2. Jeden súbor s testami prislúcha práve jednému konkrétnemu súboru (modulu) servra.
3. Názov súboru tvoríme skopírovaním názvu modulu a pridaním sufixu „_test“. (pr. modul sa volá events.js => test súbor nesie názov events_test.js)
4. Počet testov v súbore zodpovedá počtu funkcií v module.
5. Názov testu tvoríme skopírovaním názvu funkcie a pridaním sufixu „_test“

6.7.2.2 Štruktúra testu, predmet testovania

Každý test súbor obsahuje na začiatku nasledujúcu štruktúru premennej *api*:

```
var api = require('nodeunit-httpclient').create({
    port: 3000,
    path: '/',      //Base URL for requests
    status: 200,    //assert each response is OK
    headers: {      //assert that each response must have the-
se headers
        'content-type': 'application/json; charset=utf-8'
    }

});
```

Atribút *port* treba nastaviť na port na ktorom počúva server. Atribút *path* nesie v sebe url modulu ktorého funkcie ideme testovať. *Status*/*header* ponecháme v tejto štruktúre podľa vzoru- tie budú zmenené v konkrétnom teste.

Príklad testu:

```
//POST with data and custom header
exports.newEvent_test = function(assert) {
    api.post(
        assert,
        'fn',
        {
        headers: { 'content-type': 'application/json' , 'charset' :
'utf-8'  },
        data: { evt: 'postujem' } //Objects are serialised as JSON
automatically
        },
        {
            status: 200
        },
        function(res) {
            assert.equal(res.data.status,'ok');
            assert.done();
        }
    );
};
```

Funckia *post* premennej *api* vraví, že testujeme POST metódu. Ekvivalentným spôsobom sa testujú GET, PUT a DELETE. Reťazec „fn“ reprezentuje testovanú funkciu (pokiaľ je prázdny testuje sa index). Vstupné dáta sa nastavujú pre atribút *data* ako v štruktúre JSON súboru.

Testujeme:

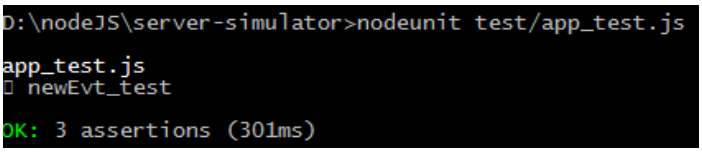
1. Či sa zhoduje hlavička prijatej odpovede s tou, ktorú očakávame (nastavená v atribúte *headers*).
    - štandardne je to hlavička JSON súboru s UTF8 kódovaním, môže byť zmenená v závislosti od konkrétneho projektu
2. Či sa zhoduje status kód http odpovede s tým ktorý očakávame (nastavený v atribúte *status*)
3. Samotnú odpoveď- výstup funkcie vo formáte JSON (v tomto prípade zisťujeme či kľúč *status* v odpovedi je zhodný s „ok“)

6.7.3 Spustenie testov

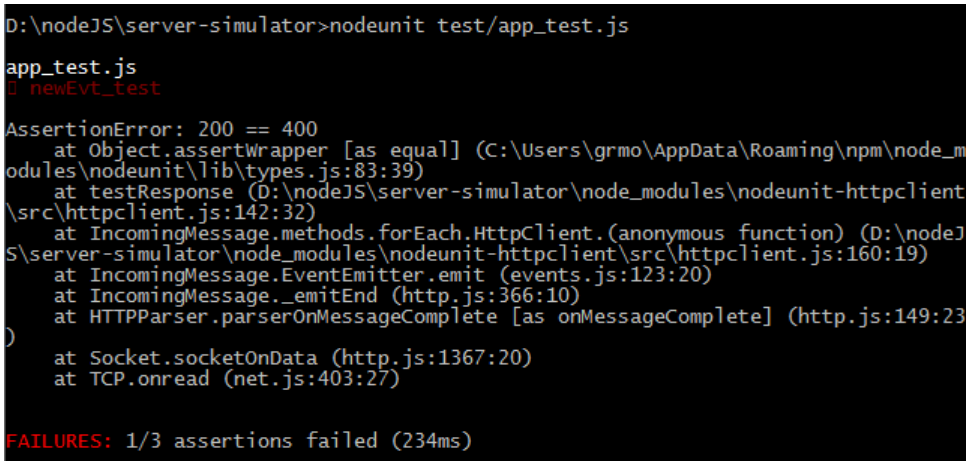
- podľa Návodu na inštaláciu a používanie testovacích nástrojov [5]

6.7.4 Vyhodnotenie testov

Po ukončení behu testov je k dispozícii výpis úspešných a chybové popisy neúspešných. Tieto je treba spísať do dokumentácie a priložiť do redminu podľa príslušných metodík – Metodika reportovania chýb [2] a Metodika tvorby technických dokumentácií [1].



Obr. 1.: Úspešný test



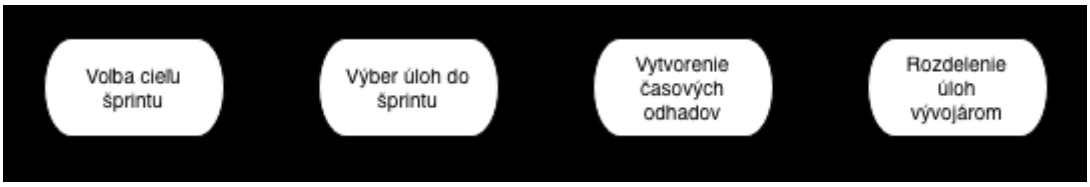
Obr. 2.: Neúspešný test

7 Manažment rozvrhu a plánovania

Autor: Ján Antala

7.1 Plánovanie šprintu

Plánovanie šprintu sa skladá z nasledujúcich fáz:



Obr. 1: Diagram plánovania šprintu

V nasledujúcej tabuľke je znázornené v ktorých kapitolách sú jednotlivé fázy zdokumentované:

Poradie	Fáza	Kapitola
1.	Voľba cieľu šprintu	1.1
2.	Výber úloh do šprintu	1.1
3	Vytvorenie časových odhadov	1.2
4	Rozdelenie úloh vývojárom	1.3, 1.4

7.1.1 Začiatok iterácie

Na začiatku každého šprintu sa naplánuje jeho cieľ a zvolia sa ´úlohy, ktoré je potrebné implementovať:

• Product Owner si zvolí cieľ šprintu

• Z Product Backlogu vyberie úlohy, ktoré by chcel mať v danom šprinte implementované

• Scrum Master ich preverí či sú konzistentné a či sú v súlade s plánom projektu

• Prípadné nezrovnalosti Scrum Master prekonzultuje s Product Ownerom

• Pri problémoch sa buď upravia úlohy zvolené Product Ownerom, alebo sa zmení

časový plán projektu

7.1.2 Časové odhady

Pre každú zvolenú úlohu sa vytvoria časové odhady:

• Scrum Master prinesie „pokrové kartičky" a rozdá ich členom tímu

• Následne prebieha rozhodovanie o jednotlivých úlohách

• Každý člen tímu si vyberie kartičku s takou hodnotou, ktorá zodpovedá jeho časovému

odhadu pre danú úlohu

• Členovia tímu zverejnia svoje kartičky

• Prebieha diskusia o danej úlohe, prečo by mala každému trvať tak dlho ako si zvolil

• V prípade zistenia nezrovnalostí v očakávaní a požiadavkách sa vývojárovi spoločne dohodnú na odhadovanom čase

• Zvolený odhad Scrum Master pripíše k danej úlohe Product Backlogu

7.1.3 Vytvorenie Release Backlogu

• Scrum Master predstaví zvolené úlohy do Release Backlogu tímu

• Následne si vývojári sami zvolia, ktoré z predstavených úloh by chceli implementovať

• Nepridelené úlohy následne Scrum Master rozdelí medzi vývojárov tak, aby mali celkový čas úloh približne rovnaký svojim schopnostiam

• Dĺžka trvania šprintu sa nastaví na dva týždne

• Úlohy pridané do Release Backlogu sa počas šprintu nesmú meniť

7.1.4 Nastavenie úloh v Redmine

• Scrum Master zmení stav rodičovskej úlohy z „Nová" na „Prebiehajúca" a k modulu priradí zodpovedného vývojára

• Ďalej zmení typ pod úlohy, ktorá sa ide vykonávať, z „Feature" na „Task" a zmení jej názov tak, že sa pridá prefix identifikujúci šprint, v ktorom prebieha, napr. „S1"

• Následne úlohu priradí vývojárovi tak, že sa zmení jej stav z „Nová" na „Priradená" a pridá k nej vývojára

• Scrum Master pre úlohy nastaví aj „Dátum začiatku" na dátum, kedy začína šprint a „Dátum skončenia" na dátum ukončenia šprintu

• Vykonávanie ´úlohy už zaznačuje vývojár podľa metodiky monitorovania projektu

A Znázornenie Product Backlogu v Redmine

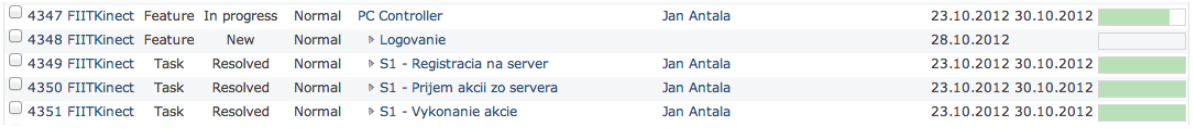
Príklad nového, ešte nepriradeného, modulu s pod úlohami v nástroji Redmine:

4332	FIITKinect	Feature	New	Normal	AR.Drone Controller		28.10.2012
4333	FIITKinect	Feature	New	Normal	Logovanie		28.10.2012
4334	FIITKinect	Feature	New	Normal	Registracia na server		28.10.2012
4335	FIITKinect	Feature	New	Normal	Prijem akcii zo servera		28.10.2012
4336	FIITKinect	Feature	New	Normal	Vykonanie akcie		28.10.2012

Obr. 2.: Nové úlohy v nástroji Redmine

B Príklad priradenej úlohy v Redmine

Príklad modulu s priradenými pod úlohami v nástroji Redmine:



4347	FIITKinect	Feature	In progress	Normal	PC Controller	Jan Antala	23.10.2012 30.10.2012
4348	FIITKinect	Feature	New	Normal	Logovanie		28.10.2012
4349	FIITKinect	Task	Resolved	Normal	S1 - Registracia na server	Jan Antala	23.10.2012 30.10.2012
4350	FIITKinect	Task	Resolved	Normal	S1 - Prijem akcii zo servera	Jan Antala	23.10.2012 30.10.2012
4351	FIITKinect	Task	Resolved	Normal	S1 - Vykonanie akcie	Jan Antala	23.10.2012 30.10.2012

Obr. 3.: Nastavenie úloh v nástroji Redmine

8 Manažment podpory vývoja

Autor: Richard Sámela

8.1 Odovzdanie verzie softvérového artefaktu

Typy rôznych verzií softvérových artefaktov používaných v našom projekte.

- Project init - prvotné odovzdanie verzie softvérového artefaktu
- Added functionality} - do projektu vnáša pridanú funkcionalitu
- Changed functionality} - v projekte mení funkcionalitu
- Removed functionality} - v projekte odstraňuje funkcionalitu
- Added GUI} - do projektu vnáša pridané grafické používateľské rozhranie
- Changed GUI} - v projekte mení grafické používateľské rozhranie
- Removed GUI} - v projekte odstraňuje grafické používateľské rozhranie
- Added resources} - do projektu vnáša pridané aplikačné zdroje
- Changed resources} - v projekte mení aplikačné zdroje
- Removed resources} - v projekte odstraňuje aplikačné zdroje
- Bug fix} - v projekte opravuje chybu
- Refactor} - robí projekt ľahšie pochopiteľnejším
- Merge} - zjednocuje súbory po vyriešení konfliktu

8.2 Ako odovzdať verziu softvérového artefaktu

Pre odovzdanie verzie softvérového artefaktu je potrebné do projektu pridať, upraviť alebo vymazať súbory, ktoré sa týkajú projektu. Následne vývojár musí odovzdať verziu softvérového artefaktu, ktorý musí obsahovať sprievodnú správu s nasledujúcimi vlastnosťami:

- musí byť napísaná v anglickom jazyku, pokiaľ nie je napísané inak
- musí byť napísaná malými písmenami
- musí byť napísaná bez interpunkcie
- všetky slová sú oddelené iba medzerov, ak nie je uvedené inak

V prípade ak daná verzia softvéru úplne rieši niektorú z úloh v Redmine, musí správa začínať slovom "resolves". V prípade, ak daná verzia nerieši žiadnu z úloh v Redmine, vývojár nenapíše nič.

Každé odovzdanie verzie softvérového artefaktu musí byť viazané s niektorou úlohou v Redmine a musí aspoň čiastočne riešiť alebo dopĺňať danú úlohu.

- správa nesmie byť uvedená bez platného identifikátora úlohy z nástroja Redmine
- správa musí obsahovať hodnotu z fronty (tracker) v Redmine z prislúchajúcej úlohy
- hodnota fronty musí byť lokalizovaná v anglickom jazyku
- za hodnotou z fronty musí nasledovať jednoznačný identifikátor prislúchajúcej úlohy v nástroji Redmine
- identifikátor úlohy musí začínať symbolom "\#"
- bezprostredne za symbolom "\#" musí nasledovať číselný identifikátor úlohy v nástroji Redmine v tvare celého čísla
- za identifikátorom úlohy musí správa obsahovať typ verzie softvérového artefaktu. Typ musí byť napísaný malými písmenami bez interpunkcie a môže mať len jednu z hodnôt popísaných v časti 5.1. Iné typy nie sú povolené
- identifikátor úlohy a typ softvérového artefaktu sú oddelené medzerou, pomlčkou a opäť medzerou " - "
- za typom verzie softvérového artefaktu musí nasledovať znak konca riadku
- v novom riadku správy musí byť stručne, vlastnými slovami napísané, čo sa v danej verzii projektu zmenilo oproti predošlej verzii
- jednotlivé popisy zmien v správe musia byť oddelené čiarkou

8.3 Príklady správ verzií softvérového artefaktu

resolves Task \#308 - added functionality}\\

it it now possible simulate continuous control with the app}

Idea \#4258 - added gui}\\

there was added a gui component to main user interface to control devices via DLNA}

Bug \#4136 - bug fix}\\

the app force close by change device orientation only if slovak localization is setted up, app works well if the english localization is setted up}

resolves Bug \#4136 - bug fix}\\

app works fine even the orientation of device is changed by every localization}

Idea \#3782 - refactor}\\

method used for copying stream changed to static for performance, class sender now doesn't extend from thread now it implement runnable interface, comments added}

9 Manažment rizík

Autor: Jakub Gondár

9.1 Identifikácia rizík

Identifikácia rizík prebiehala priebežne počas celého vývoja projektu. Zoznam identifikovaných rizík sa nachádza v nasledujúcej tabuľke spolu s odkazom na príslušný podnadpis.

Odkaz	Názov identifikovaného rizika	Dátum identifikácie	Zodpovedná osoba
Chyba! Nenašiel sa žiaden zdroj odkazov.	Nejasnosti v požiadavkách na výsledný produkt	10.10.2012	Martin Čertek
9.2.2 Neznalosť knižnice FiitKinect a projektu DigitalTheater	Neznalosť knižnice FiitKinect a projektu DigitalTheater	10.10.2012	Jakub Gondár
9.2.29.2.3	Zmena zákazníkových požiadaviek	10.10.2012	Martin Čertek
9.2.4	Nedodržanie plánovaných úloh	17.10.2012	Ján Antala
9.2.5	Nedodržanie termínu pre odovzdanie dokumentácie	1.11.2012	Michal Igaz
9.2.6	Nefunkčný prototyp do odovzdania	1.11.2012	Martin Čertek, Jakub Gondár

9.2 Analýza identifikovaných rizík

V tejto kapitole sa nachádza zoznam a podrobný opis identifikovaných rizík z časti 9.1

9.2.1 Nejasnosť v požiadavkách na výsledný produkt

Opis rizika	• Členom tímu nie je jasné, na čom presne idú pracovať • Nepochopenie požiadaviek zadávateľa projektu • Nesprávny výklad požiadaviek zadávateľa projektu • Chyba v zapísaní požiadavky do dokumentov • Zákazník sám nemusí vedieť čo presne chce
Spúšťače rizika	Konzultácia členov tímu so zadávateľom projektu a spisovanie požiadaviek do dokumentácie.
Pravdepodobnosť	Stredná
Dopad rizika	• Vyvíjanie nesprávneho produktu, ktorý zákazník nebude akceptovať • Vynakladanie úsilia ktoré bude úplne zbytočné
Priorita	Vysoká

9.2.2 Neznalosť knižnice FiitKinect a projektu DigitalTheater

Opis rizika	• Problémy pri kompilovaní importovanej knižnice minuloročného tímu • Knižnica je napísaná v C++, žiadny člen tímu nemá pokročilé znalosti s jazykom C++ • Projekt DigitalTheather závisí od mnohých knižníc, ich inštalácia môže byť problematická
Spúšťače rizika	Inštalácia a kompilácia projektu DigitalTheater
Pravdepodobnosť	Vysoká
Dopad rizika	• Zdržanie pri vyvíjaní • Problematické plánovanie • Nedodržovanie
Priorita	Vysoká

9.2.3 Zmena zákazníkových požiadaviek

Opis rizika	• Zákazník (zadávateľ projektu) podstatne zmení svoje požiadavky na výsledný produkt
Spúšťače rizika	Zákazník zmení požiadavky
Pravdepodobnosť	Nízka
Dopad rizika	• Zmárnenie časti doposiaľ vykonanej práce • Zmena plánovania, prepisovanie dokumentácii • Práca navyše
Priorita	Nízka

9.2.4 Nedodržanie plánovaných úloh

Opis rizika	• Členovia tímu nedodržia svoje pridelené plánované úlohy •
Spúšťače rizika	Časová tieseň členov tímu
Pravdepodobnosť	Stredná
Dopad rizika	• Nedodanie výsledného produktu v požadovanom termíne • Problémy so synchronizáciou práce členov tímu
Priorita	Vysoká

9.2.5 Nedodržanie termínu pre odovzdanie dokumentácie

Opis rizika	• V termíne odovzdania nebudú hotové požadované dokumentácie
Spúšťače rizika	Zlé plánovanie, nedodržanie termínov dodania častí dokumentácie od členov tímu
Pravdepodobnosť	Stredná
Dopad rizika	• Neúspešné dokončenie predmetu Tímový projekt
Priorita	Vysoká

9.2.6 Nefunkčný prototyp do odovzdania

Opis rizika	• Keď nastane termín odovzdania projektu, prototyp aplikácie nebude funkčný a nebude spĺňať akceptačné testy dohodnuté pre produkt na konci zimného semestra
Spúšťače rizika	Nedodržiavanie plánovania, výskyt neočakávanej udalosti (neidentifikovaného rizika)
Pravdepodobnosť	Stredná
Dopad rizika	• Znížené hodnotenie alebo až neúspešné absolvovanie predmetu Tímový projekt
Priorita	Vysoká

TEMPLATE pre tabuľku

Opis rizika	•
Spúšťače rizika	
Pravdepodobnosť	Stredná
Dopad rizika	•
Priorita	Vysoká
Reakcia na riziko	Dané riziko sa snažíme minimalizovať nasledovnými krokmi: • D

9.2.7 Zoradený zoznam rizík podľa priority

V nasledujúcom zozname sú zoradené identifikované riziká podľa priority. Prvé miesto znamená najvyššiu prioritu, zatiaľ čo posledné miesto v zozname znamená najnižšiu prioritu.

1. Nedodržanie termínu pre odovzdanie dokumentácie
2. Nefunkčný prototyp do odovzdania
3. Nejasnosti v požiadavkách na výsledný produkt
4. Nedodržanie plánovaných úloh
5. Neznalosť knižnice FiitKinect a projektu DigitalTheater
6. Zmena zákazníkových požiadaviek

9.3 Plán manažmentu rizík

Po identifikácii rizík a ich zoradení podľa priority je potrebné určiť reakcie na riziko a naplánovať akcie a činnosti, ktoré nám pomôžu minimalizovať identifikované riziká.

Číslo	Odkaz	Názov rizika	Reakcie na minimalizovanie rizika	Čo treba sledovať
1	9.2.5	Nedodržanie termínu pre odovzdanie dokumentácie	• Urgovanie členov tímu aby si plnili svoje povinnosti • Na termíny je kladený dôraz v plánovaní, kde sa ponecháva istá časová rezerva pre prípad núdze	Sledovať termíny Sledovať prácu členov tímu
2	9.2.6	Nefunkčný prototyp do odovzdania	• Dodržiavanie plánovania	Priebeh vývoja softvéru
3	9.2.1	Nejasnosti v požiadavkách na výsledný produkt	• Detailnou konzultáciou požiadaviek na prvých stretnutiach • Priebežnou konzultáciou a predvádzaním častí vyvíjaného softvéru	Treba sledovať či všetci členovia tímu správne pochopili deklarované požiadavky.
4	9.2.4	Nedodržanie plánovaných úloh	• Nepodcenením dôležitosti plánovania • Každý člen vyplní dotazník, kde sa odhaduje počet hodín pre každú úlohu, čím sa spresňuje odhad • Plán je nastavený tak, aby bola určitá časová rezerva na konci pre	Sledovať dodržiavanie stanovených plánov, úloh v systéme RedMine

			opravu chýb • Každý člen sa snaží dodržiavať plán	
5	9.2.2	Neznalosť knižnice FiitKinect a projektu DigitalTheater	• Konzultáciou s členom minuloročného tímu • Vyhradením dostatku času na zoznámenie s problémom	Priebeh vývoja a mieru zoznámenia a pochopenia knižnice.
6	9.2.3	Zmena zákazníkových požiadaviek	Dané riziko nedokážeme ovplyvniť a preto ho musíme akceptovať.	-

9.4 Monitorovanie rizík

9.4.1 Riziká, ktoré sa ukázali ako opodstatnené

9.2.2 Neznalosť knižnice FiitKinect a projektu DigitalTheater – Toto riziko sa ukázalo ako opodstatnené, pretože sa vyskytli problémy pri inštalácii knižníc, ktorých vyriešenie trvalo dlhšie ako očakávaný čas.

9.4.2 Riziká, ktoré neboli identifikované

... (na konci projektu)

10 Manažment monitorovania

Autor: Silvia Hudačinová

10.1 Úvod

Tento dokument opisuje manažment úloh, teda ako postupuje pracovník pri vývoji a tvorbe softvéru, aké nástroje využíva a spôsob ako komunikuje s ostatnými členmi týmu pri práci na svojej úlohe. Toto všetko zahrňuje aj kontrolu či členovia týmu vykonávajú svoje úlohy správne a včas. Túto kontrolu vykonáva manažér úloh, ktorý priebežne musí kontrolovať celý stav odovzdávania častí projektu od všetkých členov týmu. Existuje mnoho systémov pre sledovanie akú úlohu má daný pracovník vykonať, či ju už vykonal alebo je vo fáze vypracovávania. Jedným takýmto systémom je Redmine.

10.2 Dedikácia

Táto metodika je určená programátorom a manažérom alebo aj celým tímom, ktoré sa rozhodli pre Redmine ako pomocný systém riadenia projektu. Môže slúžiť aj ako prehľad pre ktoréhokoľvek záujemcu pri výbere pomocného systému riadenia.

10.3 Súvisiace metodiky

Táto metodika súvisí s nasledovnými metodikami a článkami:

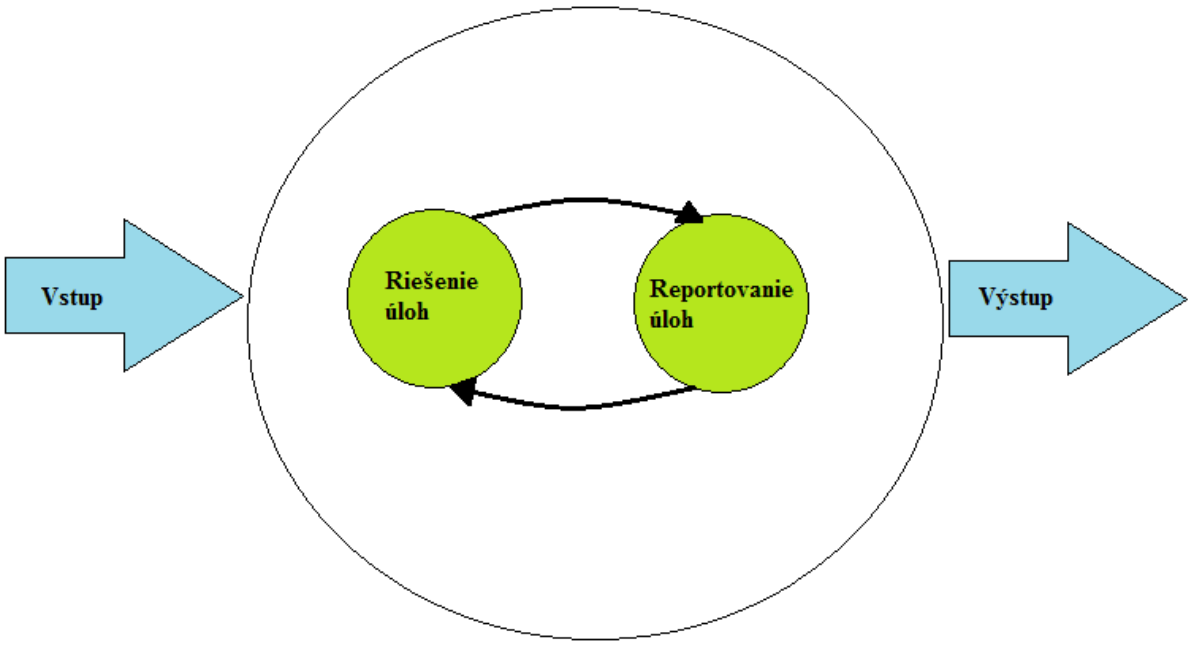
- metodika manažmentu chýb
- metodika manažmentu rozvrhu a plánovania

10.4 Slovník

Redmine	systém pre podporu riadenia projektov, je dostupný s otvoreným zdrojovým kódom.
Commit	činnosť, ktorou pracovník aktualizuje stav svojej úlohy v redmine.
Issue	výstup alebo výsledok, v redmine sú pod touto záložkou dostupné úlohy a ich stav spracovania.
Update	záložka v redmine, kde je možné zmeniť informácie ohľadom issue, teda úlohy.

10.5 Manažment úloh a reportovanie

Manažment úloh vyžaduje reportovanie od príslušných pracovníkov o stave úloh. Takéto reportovanie by malo prebiehať neustále ako je zobrazené na obrázku č. 1, až do ukončenia projektu.



Obr. 1.: Reportovanie úloh.

10.5.1 Úlohy v tíme

Člen tímu, programátor

- vykonáva zadané úlohy
- vyberá si úlohu, za ktorú prevezme zodpovednosť
- priebežne informuje o stave úlohy
- odovzdáva vypracovanú úlohu
- v prípade potreby konzultuje s manažérom

Manažér projektu

- kontroluje správnosť zadania
- kontroluje stavy odovzdávania
- v prípade potreby prideľuje úlohy
- konzultuje s členmi tímu
- konzultuje s manažérom komunikácie

Manažér komunikácie

- komunikuje a konzultuje so zákazníkom
- informuje manažéra projektu

Zákazník

- informuje manažéra komunikácie o tom čo potrebuje
- konzultuje s manažérom komunikácie

10.5.2 Postup pri manažmente úloh

	Popis	Kapitola
1.	Zadanie úlohy	5.2.1
2.	Príprava a riešenie úloh	5.2.2
3.	Reportovanie postupu prác na úlohe	5.2.3
4.	Finálne odovzdanie úlohy	5.2.4
5.	Manažment úloh a Redmine	5.2.5

10.5.2.1 Zadanie úlohy

Vstup: zadanie od zákazníka

Výstup: plán tvorby softvéru a úlohy

Vykonávatelia: zákazník a manažér komunikácie

V prvej fáze konzultuje zodpovedný vedúci so zákazníkom o jeho predstave projektu. Manažér komunikácie spracuje požiadavky zákazníka a tlmočí ich vedúcemu tímu. Ten vytvorí plán projektu a k nemu prislúchajúce úlohy.

10.5.2.2 Príprava a Riešenie úloh

Vstup: zadanie a potrebný softvér

Výstup: čiastočné riešenie úlohy

Vykonávateľ: programátor

Tu je v prvom rade nutné pripraviť sa na vykonanie úlohy. Inštalujú sa tu nové programy prípadne sa získavajú nové znalosti. Následne sa programátor púšťa do tvorby softvéru a svoje prvé výstupy zdieľa s ostatnými členmi tímu.

10.5.2.3 Reportovanie postupu prác na úlohe

Vstup: vykonaná práca

Výstup: odovzdanie úlohy

Vykonávateľ: programátor

Po tom ako programátor vykoná pridelenú úlohu, podáva informácie o vykonanej práci vedúcemu projektu ale aj spolupracovníkom, ktorý po ňom úlohu prevezmú. Tu existujú rôzne riešenia, ako dať vedieť o stave vykonanej práce, a tie sú rozličné softvéry určené pre reportovanie.

10.5.2.4 Finálne odovzdanie úlohy

Vstup: čiastkové riešenie úlohy

Výstup: finálna verzia úlohy

Vykonávateľ: programátor

V poslednej fáze programátor odovzdá svoju finálnu verziu úlohy. Finálne odovzdanie sa tiež odzrkadľuje v príslušnom programe, tak aby aj manažér aj ostatní programátori videli v akej fáze vykonávania úlohy je daný pracovník. Toto odovzdávanie by sa malo uskutočňovať včas, aby nebrzdilo ďalší vývoj projektu. Na finálnu verziu jedného programu môže nadväzovať ďalší program od iného programátora.

10.5.2.5 Manažment úloh a Redmine

Na vykonanie všetkých vyššie spomenutých aktivít existuje množstvo systémových riešení. Jednou z nich je Redmine, je to systém pre manažment a monitorovanie úloh, kde sa zaznamenávajú úlohy a stav, v ktorom sa aktuálne nachádzajú.

10.6 Redmine

Systém umožňuje jednoducho sledovať celý priebeh projektu, teda kto, kedy a v akom stave odovzdal pridelenú úlohu. Je tu možné sledovanie posledných aktivít v rámci tímu. Celý systém Redmine nie je vyhradený len pre jeden projekt môže sa tu súčasne realizovať viacero projektov, ktoré si zvolí užívateľ hneď po prihlásení.
Nasledujúca tabuľka zobrazuje postup práce so systémom Redmine, kde sa v uvedených kapitolách opisuje práca so systémom. Teda pridávanie, aktualizovanie a dokončenie úloh.

	Popis	Kapitola
1.	Nová úloha	6.1
2.	Riešenie úloh	6.2
3.	Aktualizovanie stavu úlohy	6.3
4.	Dokončenie úlohy	6.4

10.6.1 Nová úloha

Vstup: zadanie úloh

Výstup: úlohy pridané v systéme

Vykonávateľ: manažér alebo programátor

Keď sú už naplánované úlohy, treba ich sprístupniť programátorom, tak aby bolo jasné kedy a čo majú spraviť a koľko majú na danú úlohu času. Zadanie úloh je možné spraviť jednoducho v záložke New issue, kde vyplníme všetky dôležité údaje, kde nesmie chýbať:

- o akú úlohu ide
- názov úlohy
- termíny, od kedy do kedy sa má vykonať
- vykonávateľ
- priorita
- charakter úlohy, či ide o novú, opravu tej starej a pod.
- popis čo všetko treba spraviť a na čo sa nemá zabudnúť
- stav vykonávania úlohy v percentách

10.6.2 Riešenie úloh

Vstup: zadanie úloh zadané v redmine

Výstup: vykonaná úloha

Vykonávateľ: programátor

Po zadaní úloh si to môže programátor pozrieť. V záložke Issues má možnosť vidieť zadanie ostatných pracovníkov ale predovšetkým svoje zadanie. Ak má pokračovať v úlohe ktorej začiatok vykonáva iný programátor, má možnosť mu po kliknutí na danú úlohu, možnosť pripísať svoje odporúčania alebo pripomienky.

10.6.3 Aktualizovanie stavu úlohy

Vstup: vykonaná úloha

Výstup: commit úloh v systéme

Vykonávateľ: programátor

Tu sa vykonávajú zadané úlohy ale je samozrejme dôležité aby sa to odrážalo v systéme, aby ostatný spolupracovníci a manažér vedeli v akom stave je rozpracovaná úloha. V záložke Issues nájdeme zadané úlohy a ich percentuálny stav spracovania. Po kliknutí na danú issue sa zobrazí jej detail, kde môže prebehnúť commit, teda úprava zadania na koľko percent je úloha vykonaná. Vykonáva sa po kliknutí na Update, ktorý je v pravo hore, tesne nad onom úlohy. Túto činnosť vykonáva programátor priebežne počas riešenia úlohy, tak aby mal manažér a ostatný spolupracovníci prehľad o stave úlohy.

10.6.4 Dokončenie úlohy

Vstup: dokončená úloha

Výstup: konečný commit v systéme

Vykonávateľ: programátor

Dokončenie úlohy je sprevádzané update-om do systému. Robí sa rovnako ako commit, po kliknutí na úlohu sa zmení jej percentuálny stav na sto percent. Po konečnom kliknutí sa stav zobrazí všetkým pracovníkom.

10.6.5 Možnosti Redmine-u

Samozrejme systém ma mnoho ďalších funkcií, v záložkách ktoré nájdeme pod ich anglickým názvom, a to:

- Overwiev – tu sa nachádzajú informácie o projekte, vymenovanie všetkých členov tímu, sú tu zhrnuté úlohy, teda počet dokončených a rozpracovaných úloh
- Activity – tu sa zobrazujú všetky vykonané aktivity podľa času, prvé sú tie, ktoré boli vykonané naposledy
- Charts – používateľ má možnosť vidieť grafické zobrazenie teda graf podľa vlastného určenia
- Gantt – tu sa zobrazujú všetky úlohy projektu, znázornenie je grafické podľa času a znova podľa toho ako si nastaví používateľ
- Calendar – v tejto záložke sa nám zobrazí kalendár príslušného mesiaca, používateľ si môže prezerať aj predchádzajúce a nasledujúce mesiace, pričom pri každom dni sú zobrazené úlohy
- Settings – nastavenia

Nasledujúce záložky slúžia na dokumentáciu, ktorá súvisí s projektom :

- Documents
- Wiki
- Files

10.7 Záver

Metodika na vyššej úrovni popisuje priebeh vykonávania a reportovania úlohy a akú účasť na tom má programátor, manažéri a zákazník. Ďalej, na nižšej úrovni, metodika popisuje priebeh vykonávania a reportovanie pomocou konkrétneho systému Redmine. Zameriava sa tu na to, ako postupuje programátor pri reportovaní úlohy. Je tu možné zistiť, ktorá úloha a do akej miery je vykonaná. V poslednej časti sú zhrnuté všetky prehľady a zobrazenia týkajúce sa vykonávaných úloh, ktoré Redmine ponúka.

11 Manažment dokumentácie

Autor: Michal Igaz

11.1 Slovník pojmov

Java – objektovo – orientovaný programovací jazyk

Javadoc – nástroj na automatické generovanie komentárov kódu

Rozhrania – spôsob komunikácie medzi triedou a vonkajším svetom, rozhranie predpisuje spôsob komunikácie

Balíky – slúžia na organizovanie tried v objektovo – orientovaných jazykoch

Triedy – plán/prototyp vytvárania objektov

Metódy – funkcia, ktorá zapuzdruje istú časť funkcionality

Premenné – vyhradené miesto v pamäti, môže nadobúdať viacero typov

Konštanty – premenné, ktoré majú nemennú hodnotu počas behu programu

Architektúra systému – je celkový obrázok o projekte, definuje rozloženie balíkov a tried v rámci systému

Komponent – kus funkcionality, typicky je to trieda

Big Picture – znamená celkový prehľad o projekte

Jdbc – technológia na spojenie s databázou v jave

11.2 Roly

Pre technickú dokumentáciu je typické, že ju z väčšej časti tvoria programátori. Významné úlohy pri jej tvorbe však zastávajú aj analytici, architekti, manažéri dokumentácie a korektori.

1. Analytik
    a. Zodpovedný za špecifikáciu požiadaviek a funkčnú špecifikáciu
    b. Zodpovedný za udržiavanie tzv. Big picture
2. Architekt
    a. Zodpovedný za architektúru systému
    b. Zodpovedný za určovanie priorít komponentov tak, aby sa najskôr implementovali tie vlastnosti systému, ktoré sú požadované najskôr
3. Programátor

   a. Zodpovedný za funkcionalitu komponentov, ktorá vytvoril
   b. Zodpovedný za dostatočne okomentovaný kód
4. Manažér dokumentácie
   a. Zodpovedný za konzistentnosť a kompletnosť softvérovej dokumentácie
   b. Zodpovedný za odovzdanie finálnej verzie dokumentácie v stanovenom čase
5. Korektor (proof reader)
   a. Zodpovedný za gramatickú stránku dokumentácie
   b. Zodpovedný za dodržanie termínov ohľadom korektúr

11.3 Využitie javadocu pre automatickú tvorbu dokumentácie

Táto metodika predpokladá znalosť metodiky komentovania kódu a znalosť prostredia eclipse alebo netbeans.

11.3.1 Úvod

Javadoc je nástroj, ktorý generuje html dokumentáciu z javadoc-ových komentárov v zdrojovom kóde. HTML je využívaná z dôvodu schopnosti odkazovania na príbuzné dokumenty.
Vhodne okomentovaný kód s využitím komentárov nástroja javadoc poskytuje prehľadnú štruktúru technickej dokumentácie ku kódu. Na úplné využitie možností tohto nástroje je potrebné doplniť vhodné komentáre.

11.3.2 Javadoc komentáre

Javadoc rozoznáva špeciálne komentáre **/** …..*/**, ktoré sú zvýraznené tak, ako je uvedené v príručke používania jednotlivých vývojových prostredí.
Javadoc dovoľuje programátorovi pripájať opisy k triedam, konštruktorom, poliam (premenné), rozhraniam a metódam vo vygenerovanej html dokumentácii tak, že javadoc-ové komentáre budú vložené priamo pred ich deklarácie.
Všeobecný postup pre vkladanie javadoc komentárov do kódu.
Vloženie mena autora
Vloženie čísla verzie
Vloženie opisu relevantný typu objektu
Vloženie definície ďalších relevantných vlastností

11.3.2.1 Vloženie mena autora

Jednou z najdôležitejších častí v javadoc komentároch tried, metód alebo konštruktorov je meno autora daného objektu.

Meno autora sa zadáva podľa nasledujúcej konvencie:
<Titul> <Meno> <Priezvisko> <Titul>, <ID>, <Pracovná pozícia>, <Typ zmeny>
Titul – vloženie relevantného titulu pracovníka
Meno – krstné meno pracovníka
Priezvisko – priezvisko pracovníka
ID – identifikačné číslo pracovníka
Pracovná pozícia – pracovné zaradenie pracovníka
Typ zmeny – rozlišujú sa dva typy: vytvorenie a zmena

Pri zmene už existujúceho komponentu je povinný pracovník uviesť svoje meno ako ďalšieho autora so zodpovedajúcou zmenou opisu komponentu. Táto situácia je znázornená príklade 2.

Príklad 1

```
/** <<Opis komponentu>>
 * @author Bc. Michal Igaz, 64355, Analytik, Vytvorenie
 *
 */
```

Príklad 2

```
/** <<Opis komponentu>>
 * @author Bc. Michal Igaz, 64355, Analytik, Vytvorenie
 * @author Bc. Michal Igaz, 64355, Analytik, Zmena
 *
 */
```

11.3.2.2 Vloženie čísla verzie

Číslo verzie je ďalším potrebným údajom, ktorý podáva informáciu o tom, ako ďaleko postúpil vývoj daného komponentu. Okrem čísla verzie sa zadáva aj čas zmeny.

Číslo verzie a čas zmeny sa zadávajú podľa nasledujúcej konvencie:
<Verzia>, <Dátum>

Verzia – podáva informáciu o čísle verzie, zadáva sa ako:
v<Číslo hlavnej verzie>.<Číslo subverzie>
Dátum – podáva informáciu o dátume poslednej zmeny, zadáva sa ako:
<Rok>-<mesiac>-<deň>

Okrem vyššie uvedených je potrebné vložiť aj dátum vytvorenia komponentu. Dátum vytvorenia má rovnaký formát ako dátum zmeny:
<Rok>-<mesiac>-<deň>
Príklad

```
/** <<Opis komponentu>>
 * @version v3.45, 2012-10-28
 * @since 2012-6-20
 *
 */
```

11.3.2.3 Vloženie opisu relevantný typu objektu

Opis je posledným z dôležitých údajov, ktoré je nutné zadať do javadoc komentárov. Rôznymi typmi opisov sa zaoberá metodika o komentovaní zdrojového kódu v kapitole **Chyba! Nenašiel sa žiaden zdroj odkazov.**.

Opis musí byť čiastočne štrukturalizovaný a musí obsahovať určité základné náležitosti.
Typ objektu – autor musí explicitne deklarovať, aký komponent vytvoril, možnosti:
Balík
Trieda
Metóda
Opis funkcionality – autor uvádza, aké použitie má komponent a aké algoritmy boli použité, spolu s uvedením, kde sa tieto algoritmy použili

Príklad 1

```
/** Typ: Trieda
 * Trieda zahŕňa funkcionalitu zahŕňajúcu kontakt s databázami. Využíva bežnú technológiu
 * „jdbc“.
 * Nadviazanie spojenia – metóda estamblishConnection() vytvára spojenie s databázou
 * Transakcie – metóda getData(Object query) vracia výsledok dopytu do databázy uložený
 * v resultSet
 * Uzavretie spojenie – metóda closeConnection() zatvára spojenie s databázou
 *
 */
```

Príklad 2

```
/** Typ: Metóda
 * Metóda zahŕňa funkcionalitu triediaceho algoritmu quicksort.
 * Použitý je modifikovaný algoritmus quicksortu s využitím počítania mediánu v postupnosti
 * čísiel.
 * Metóda je funkčná pre polia celých čísiel.
 *
 */
```

11.3.2.4 Vloženie definície ďalších relevantných vlastností

Tento bod sa týka metód a konštruktorov. Pre ich opis treba dodefinovať vstupné a výstupné parametre. Pre zadávanie parametrov platia konvencie o pomenovávaní objektov v jave opísaných v „Metodike pomenovávania objektov v jave“.

Konvencie pre zadávanie:
Parametre – opis parametra, ďalej treba jednoznačne určiť typ objektu a opis jeho určenia, prípadne triedu, z ktorej prichádza
Návratová hodnota – do opisu návratovej hodnoty treba uviesť typ objektu a opis jeho určenia

Príklad

```
/** <<Opis komponentu>>
 *
 * @param Počet ľudí v banke, celé číslo, prichádza z triedy “CustomerCounter”
 * @return Potrebný ďalší pracovník, Boolean, potrebný pre zvýšenie počtu pracovníkov v
 *                 prípade                 veľkého                 množstva                 klientov
 *
 */
```

12 Záznamy zo stretnutí

V tejto kapitole sa budeme venovať záznamom zo stretnutí tímu TeamToo. V zázname z každého stretnutia sú dve tabuľky. Prvá obsahuje už vykonanú prácu jednotlivých členov tímu a v druhej sú úlohy, ktoré vykoná daný člen tímu do dátumu uvedeného v poslednom stĺpci.

12.1 Zápis 1. stretnutia tímu

Téma stretnutia:
Prvé stretnutie, všeobecné pokyny

Dátum stretnutia: 10.10.2012

Čas stretnutia: 9.00 – 11.30

Zapisovateľ: Michal Igaz

Účastníci:
Ján Antala
Martin Čertek
Richard Sámela
Michal Igaz
Silvia Hudačinová
Ondrej Grman
Jakub Gondár

Vedúca stretnutia: Ing. Vanda Benešová, PhD.

Priebeh stretnutia: Všeobecné pokyny ohľadom projektu Ustanovenie všeobecných pravidiel:

- Žiadne mobily na schôdzkach
- Dochádzka načas
- Každú neprítomnosť treba hlásiť
- Správy, na ktoré netreba response označovať ako FYI (for your information)

Dohodnutie o všeobecnej architektúre Dohoda o použitých technológiách Používa sa C++ na starom projekte, rozhodnutie o nadviazaní naň Stanovenie vstupov a výstupov projektu Stanovenie nového termínu na utorok 12.00

12.2 Zápis 2. stretnutia tímu

Téma stretnutia:
Druhé stretnutie, vytvorenie backlog

Dátum stretnutia: 17.10.2012

Čas stretnutia: 9.00-12.15

Zapisovateľ: Michal Igaz

Účastníci:
Michal Igaz
Ondrej Grman
Richard Sámela
Jakub Gondár
Silvia Hudačinová
Ján Antala
Martin Čertek

Vedúca stretnutia: Ing. Vanda Benešová, PhD.

Priebeh stretnutia:
Ujasnenie si základných princípov SCRUM.
Vytvorenie product backlog a release backlog.
Debata ohľadom architektúry aplikácie a modulárnosti riešenia.
Debata ohľadom protokolu.
Vytvorenie úvodnej špecifikácie užívateľských rolí spolu s ich stručným popisom.
Debata o implementačnom prostredí.
Kinect – C++.
Kvôli modulárnosti riešenia sú zvyšné časti implementované zvlášť. Menovite Android, ohľadom ostatných sa debatuje, pravdepodobný výber C#.
Rozdelenie úloh v tíme na nasledujúci šprint.
Ohodnotenie a prioritizácia zvolených lístkov v backlog product ownerom (vedúcou projektu).

Vyhodnotenie úloh z predchádzajúceho stretnutia:

Číslo úlohy	Úloha	Zodpovedný	Termín	Stav
1	Analýza existujúceho riešenia	Všetci	17.10.2012	Riešená
2	Prezretie dokumentácie z predošlého tímového protokolu	Všetci	17.10.2012	Riešená

Pridelené úlohy:

Číslo úlohy	Úloha	Zodpovedný	Termín
1	Analýza požiadaviek, špecifikácia	Igaz	23.10.2012
2	Spracovanie kódu z predchádzajúceho TP pre účely nášho projektu	Gondár	23.10.2012
3	Vytvorenie web stránky	Antala	21.10.2012
4	Konfigurácia RedMine a Gitbus	Sámela	23.10.2012
5	Analýza existujúceho GUI, návrh riešenia nového GUI	Čertek	23.10.2012
6	Dátový model a databáza	Grman	23.10.2012
7	Analýza riešení pre spracovanie zvuku	Hudačinová	23.10.2012

12.3 Zápis 3. stretnutia tímu

Téma stretnutia:
3. stretnutie

Dátum stretnutia: 23.10.2012

Čas stretnutia:13.00-14.45

Zapisovateľ: Silvia Hudačinová

Účastníci:
Michal Igaz
Ondrej Grman
Richard Sámela
Jakub Gondár
Silvia Hudačinová
Ján Antala
Martin Čertek

Vedúca stretnutia:Ing. Vanda Benešová, PhD.

Priebeh stretnutia:
Diskusia ohľadom webovej prezentácie.
Rozoberanie plánovania, Redmine.
SCRUM – online dokument, úlohy cez formulár, každý doplní a zašle mailom vedúcemu projektu.
Na konci stretnutia Rišo predviedol Android aplikáciu a konzultoval o jej zdokonalení.

Vyhodnotenie úloh z predchádzajúceho stretnutia

Číslo úlohy	Úloha	Zodpovedný	Termín	Stav
1	špecifikácia požiadaviek, use-case šablóna, analýzy	Igaz	23.10.2012	Riešená
2	Pokus o spojazdnenie predchádzajúceho projektu spolu s update-om	Gondár	23.10.2012	Riešená
3	Simulácia posielania správ z kinectu simulácia výstupného zariadenia web, server- vstup na výstup	Antala	23.10.2012	Hotová
4	Redmine, gitbus, rozbehaný server funkčný prototyp android gest	Sámela	23.10.2012	Hotová
5	analýza pôvodného GUI	Čertek	23.10.2012	Hotová
6	Analýza technológií, rozbehanie serveru, simulácia textových výstupov, kt. bežia pod GUI	Grman	23.10.2012	Hotová
7	Analýza systémov na rozpoznávanie reči	Hudačinová	23.10.2012	Hotová

Pridelené úlohy :

Číslo úlohy	Úloha	Zodpovedný	Termín
1	Finalizovať use-case, katalóg používateľov, definícia využitia modulov	Igaz	30.10.2010
2	Funkčný prototyp verzie z tohto týždňa	Grman	30.10.2010
3	Zdokonaliť a dokončiť software vektor ťahu prsta -> mapovanie	Sámela	30.10.2010
4	Spojazdenie predchádzajúceho projektu a pokus spojazdniť to na všetkých pc	Gondár	30.10.2010
5	Preštudovanie dokumentácie, prísť na prednášku v pondelok o 16tej v BC300, dohodnúť s M. Rus-kom	Hudačinová	30.10.2010
6	Úprava spraveného zadania, ako spracovať správy aj na iné zariadenie(pc)	Antala	30.10.2010
7	Gui - návrh zvlášť pre "experimentátora" a zvlášť pre používateľa bežného používateľa	Certek	30.10.2010

12.4 Zápis 4. stretnutia tímu

Téma stretnutia: 4. pravidelné stretnutie
Dátum stretnutia: utorok 30.10.2012
Čas stretnutia: 12:00
Zapisovateľ: Jakub Gondár

Účastníci:
Bc. Ján Antala
Bc. Martin Čertek
Bc. Jakub Gondár
Bc. Ondrej Grman
Bc. Silvia Hudačinová
Bc. Michal Igaz
Bc. Richard Sámela

Vedúca stretnutia: Ing. Vanda Benešová, PhD.

Priebeh stretnutia:
Diskusia k práci s televíznym DVB-T prijímačom. DVB-T prijímač bude v spolupráci s notebookom simulovať televízny prijímač. Predpokladá sa, že budeme prostredníctvom IrDA posielať signály a ovládať notebook s DVB-T prijímačom.
Postupné konzultácie s každým členom tímu, každý rozprával o tom, čo vykonal za uplynulý týždeň a každému boli pridelené nové úlohy.

Pridelené úlohy z minulého týždňa:

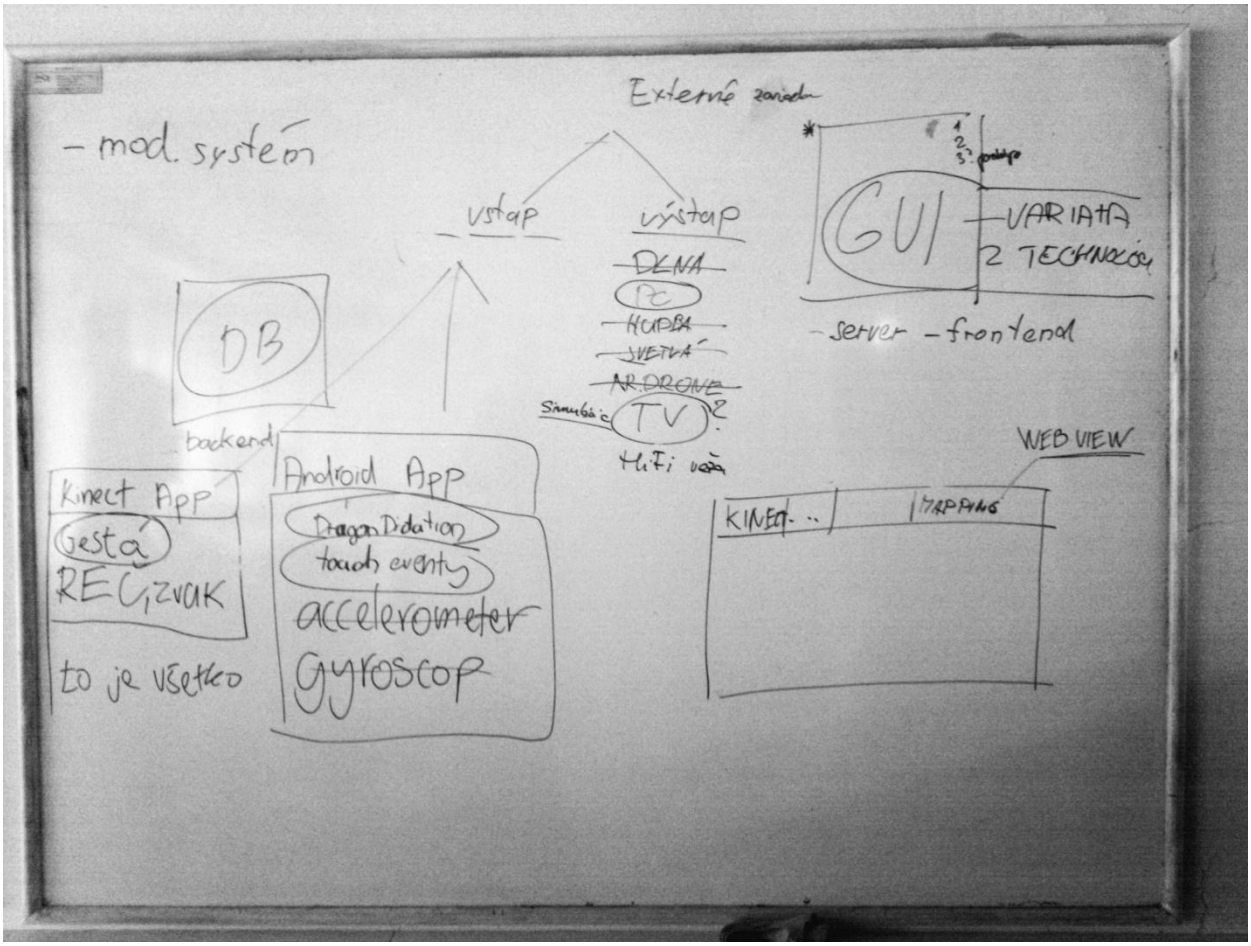
Číslo úlohy	Úloha	Zodpovedný	Termín	Stav riešenia
1	Pokračovať v dokumentácii – špecifikácia požiadaviek a analýza	Igaz	30.10.2012	Vypracované asi 40%
2	Finalizovať user stories.	Igaz	30.10.2012	Hotové. Vypracovaný 1. návrh, pripravený na konzultáciu
3	Inštalácia a konfigurácia knižníc pre kompiláciu DigitalTeather vo Visual Studiu 2010.	Gondár	30.10.2012	Hotové. DigitalTeather je možné skompilovať a spustiť.
4	Príprava aplikácie, ktorá bude ovládať počítač (simulácia kláve-	Antala	30.10.2012	Hotové. Pripravená aplikácia, ktorá ovláda prezentáciu

	sov a myši)			klávesmi a vie kontrolovať hlasitosť zvuku počítača.
5	Štúdium rozpoznávania gest na Android zariadeniach.	Sámela	30.10.2012	Zdokonalené rozpoznávanie v prototypovej aplikácii.
6	Riešenie kontinuálnych gest	Sámela	30.10.2012	Hotové. Predvedené kontinuálne ovládanie hlasitosti.
7	Vytvorenie náčrtov GUI v QT	Čertek	30.10.2012	Hotové. Prvé prototypy GUI nakreslené a vytvorené v QT.
8	Prvý prototyp serveru	Grman	30.10.2012	Hotové. Funkčné mapovanie eventov na akcie v databáze
9	Stretnúť sa s p. Ruskom ohľadom zvukového modulu	Hudačinová	30.10.2012	Bola na stretnutí, dokumentácia nespravená
10	Zápisnica z 3. stretnutia	Hudačinová	30.10.2012	Hotové. Zápisnica dokončená.

Pridelené úlohy na ďalšie stretnutie:

Číslo úlohy	**Úloha**	**Zodpovedný**	**Termín**
1	Dokumentáciu definitívne dokončiť.	Igaz	8.11.2012
2	Registrácia zariadení, príjem eventov a preposlanie vstupov na výstup.	Grman	6.11.2012
3	Spraviť implementáciu rozpoznávania „touch“ gest, naštudovať použité algoritmy.	Sámela	6.11.2012
4	Pripraviť odosielací modul, napojiť ho rozpoznané gestá Kinectu.	Gondár	6.11.2012
5	Spolupracovať s Igazom na dokončení dokumentácie.	Hudačinová	6.11.2012
6	Vytvorenie webovej aplikácie pre GUI konfiguráciu servera.	Antala	6.11.2012
7	Zápisnica z 4. stretnutia	Gondár	6.11.2012

Doladenie zodpovednosti za Kinect GUI aplikáciu. Čertek je zodpovedný za design aplikácie na úrovni GUI, Gondár je zodpovedný za implementáciu navrhnutej aplikácie.
Predvedené ovládanie počítača prostredníctvom mobilu so systémom Android. Pomocou gest na displeji telefónu Richard posúval priesvitky prezentácie spustenej na počítači. Potom bolo predvedené ovládanie hlasitosti počítača prostredníctvom kontinuálneho gesta. Mobilné zariadenie posielalo vertikálnu súradnicu a to sa mapovalo na úroveň hlasitosti na počítači.
Po prestávke sme sa pokúsili odhadnúť a definovať, ako má vyzerať výsledok nášho projektu na konci tohto semestra.
Najprv sme si na tabuľu nakreslili všetky vlastnosti, ktoré by mal mať výsledný produkt v letnom semestri. Potom sme sa pokúsili odhadnúť, ktoré vlastnosti stihneme spraviť v zimnom semestri a ktoré si necháme na letný.
Vlastnosti, ktoré majú byť funkčné (aspoň prototypom) na konci zimného semestra:
Grafické rozhranie (GUI) – varianty s dvoma technológiami (qt + webview)
Mapovací server s databázou
Vstupy:
rozpoznanie gest cez Kinect
Android zariadenie – gestá prstami
(?) rozpoznávanie reči a zvukov (?)
Výstupy:
Ovládanie počítača
Ovládanie pomocou IrDA – Televízor, Hifi veža a iné IrDA zariadenia



Obr.1.: Náčrt vlastností cieľovej aplikácie

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií
Softvérové inžinierstvo a Informačné systémy

PREBERACÍ PROTOKOL

Typ projektu: Tímový projekt

Členovia tímu: Bc. Ján Antala

Bc. Martin Čertek

Bc. Richard Sámela

Bc. Michal Igaz

Bc. Silvia Hudačinová

Bc. Ondrej Grman

Bc. Jakub Gondár

Názov projektu: FIITKinect

Počet strán: .....

Ing. Vanda Benešová, PhD., týmto potvrdzuje prevzatie Projektovej dokumentácie - analýzy problému, špecifikácie požiadaviek, návrhu riešenia a Dokumentácie riadenia projektu.

Podpis: .....

Bratislava, dňa .....