TÍMOVÝ PROJEKT TRACKS

Zápisnica

Dátum konania / Miesto konania	26.9.2016, 3.21
Zúčastnení	Ing. Martin Konôpka (MaKo) Ing. Karol Rástočný, Phd. (KaRa) Miriama Pomffyová (MiPo) Michal Slovík (MiSl) Peter Bobovský (PeBo) Peter Kučera (PeKu) Marek Mura (MaMu) Michal Kráľ (MiKr)
Neprítomní	Lukáš Račko (LuRa)
Zapísal	Miriama Pomffyová
Prílohy	-

PROGRAM PORADY

1. Zoznámenie sa tímu s vedúcim a konzultantom.

2. Oboznámenie sa s témou projektu. Pridelenie úloh.

NOVÉ ÚLOHY

P. č.	Názov úlohy	Popis úlohy	Zodpovedný	Termín splnenia
1.	Vytvorenie plagátu	Tím sa po stretnutí s vedúcim stretne a navrhne plagát – bude obsahovať názov tímu, mená, číslo tímu, názov projektu a tému vystihujúcu projekt.	celý tím	***27.9.2016***
2.	Teambuilding	Členovia tímu sa neformálne stretnú s vedúcim – dohodne sa termín – štvrtok (niekde v meste) alebo piatok (Stupava – Dni zelá).	celý tím	30.9.2016

3.	Stránka tímu	Vytvoriť webové sídlo projektu statické – informácie o tíme a o aktualizáciach práce na projekte (html , JavaScript, bootstrap).Peter Lacko dodá informácie.	MiKr	3.10.2016
4.	Prihlásiť sa na TFS	Všetci členovia tímu sa 1.-krát prihlásia do systému tfs.fiit.stuba.sk pod prihlasovacími údajmi do AIS. Heslo sa dá zmeniť. Je tu integrovaný kalendár, git, work – kto čo robí, koľko toho spravil; build – možné automatické pretestovanie, nasadzovanie na server; code - user story – kto čo programoval. Oboznámiť sa s prostredím a dať info vedúcemu, keď sa každý prihlási.	celý tím	3.10.2016
5.	Metodika gitu	Spísať metodiku používania gitu. Napr. vznikne úloha – vytváram branch – nazvem podľa úlohy – pracujem – mergnem s master brnach – otestujem či sa dá buildnúť, ak nie, vrátim do pôvodného stavu.	nepridelené	3.10.2016
6.	Rozbehanie projektu	Zbuildnúť projekt, editovanie a uloženie, spísanie k tomu dokumentácie. Prepojenie s TFS (Team Foundation Server). Naklonovať repozitár z gitu do našich kont. Spísať k tomu manuál.	MaMu, MiSl	3.10.2016
7.	Spísať Code convention	Spísať code convention, podľa možností priložiť tabuľku porovnania kódu Java do C#. Napr. používanie – namespace/ using, alebo o premennej var – nemusím definovať explicitne, vezme si ho sám pri vytvorení inštancie. Uvádzať aj príklady.	PeKu, PeBo	3.10.2016
8.	Spísať zápisnicu	Spísať zápisnicu zo stretnutia a vo formáte PDF priložiť na stránku, dávať relatívne odkazy na zápisnice. Uložiť na GoogleDisk.	MiPo	***28.9.2016***
9.	Vytvoriť úložisko	Vytvoriť úložisko na GoogleDisku pre dokumentácie a zápisnice.	MiSl	***27.9.2016***
10.	EA	Získať licencie na EA (Enterprise Architect). Poslať model v EA a dokumentáciu k RDF.	MaKo	3.10.2016

NOVÉ INFORMÁCIE

CIEĽ PROJEKTU

Vytvorenie systému TRACKS pre zber a prepájanie dát, ktoré po sebe zanechali vývojári pri vývoji softvéru, t.j. „Tasks, Code Reviews, Activities, Source Code, and Knowledge about Software“. Zber dát by mal byť vykonávaní pomocou botnet klientov na základe zadaní od hlavného uzla. Nasleduje prevedenie dát do spoločnej reprezentácie, ich samotné prepájanie a sprístupnenie tretím stranám. Jedným z hlavných cieľov je navrhnúť infraštruktúru tak, aby bola ľahko nasaditeľná ako na fakulte, tak aj interne v softvérovej firme, a bola využitá záverečnými projektami na fakulte.

SYSTÉM – PREPÁJANIE DÁT O VÝVOJI SOFTVÉRU

- V procese vývoja softvérových projektov používame rôzne podporné nástroje, v ktorých zanechávame stopy o svojej činnosti:
    - vývojové prostredia (Eclipse, Visual Studio),
    - systémy pre správu zdrojových kódov (Git),
    - úloh a nahlasovania chýb (Bugzilla, Jira, Redmine),
    - prehliadok zdrojového kódu (Gerrit),
    - alebo aj diskusné fóra a dokumentácie (StackOverflow, MSDN).
- Problém nastáva, keď sa snažíme nájsť súvislosti medzi nimi. Príkladom je prepojenie otázky v StackOverflow od vývojára pracujúceho na úlohe z Bugzilla a jeho commity v Git. Pri následnej prehliadke zmien v Gerrit má kontrolór pred sebou iba výsledný návrh zmeny v Git a nie odpoveď z SO, na ktorej vývojár svoje riešenie založil. Všetky uvedené dáta o vývoji softvéru sú v prípade open source projektov voľne dostupné cez API alebo web crawling. Každý zdroj dát je však samostatné riešenie, a tak sú dáta slabo alebo vôbec prepojené, napr. iba v komentároch (ID úlohy v commit message). Orientácia v množstve informácií je tak zbytočne náročná, znepríjemňuje analýzu dát výskumníkom, uplatnenie ich výsledkov v praxi, a zároveň obmedzuje vznik nových podporných nástrojov pre vývoj softvéru. A keďže nedokážeme upraviť všetky systémy, spravíme si vlastný TRACKS, ktorý nám a naším študentom umožní to pekne pospájať.
- Kľúčovými komponentmi systému TRACKS pre vytvorenie sémantickej vrstvy nad dátami o vývoji softvéru by mali byť:
    - hlavný (master) uzol pre plánovanie zberu dát,
    - distribuované botnet klienti pre zber dát, špecifické pre každý zdroj dát (napr. Git, Bugzilla, Gerrit),
    - repozitáre zozbieraných dát a odvodených metadát,
    - rozhranie pre vystavenie dát a rozšíriteľnosť systému o analýzu dát Príkladmi softvérových projektov, ktoré môžeme hneď zbierať a prepájať sú Eclipse, Android, Qt, OpenStack, i mnohé ďalšie na http://gerrithub.io, ku ktorým máme dostupné dáta rôzneho druhu.
- Každý systém má svoje dáta a samostatné prihlásenie používateľa – login (Git, tasky –úlohy, chyby) – chceme prepojiť súvis commitov.

ČINNOSTI:

- Master vetva – otestovaný zdrojový kód, vedľajšie vetvy – fixovanie chýb + vývoj -> merge.
- Unit testy majú mať čo najväčšie pokrytie.
- Prehľad kódov – code review – kontrola commitov, opráv chýb, či boli správne opravené – Gerrid – nadstavba na Git.
- Sledovanie programátora počas práce – nájsť, čo mu dlho trvá, s čím má problémy, nájsť dobrých a efektívnych riešiteľov pre vybraný problém.
- Zadávať tasky, robiť reviews, vykonávať activities, zdieľať a dokumentovať knowledges, ...
- Zápisnica obsahuje – čo sa riešilo na stretnutí, kto sa zúčastnil a kto nie, kto mal aké nápady a pripomienky, prejdú sa zadané úlohy, či sú splnené / nesplnené, zápis rozhodnutí, ktoré boli prečo ako urobené. Ukladať na GoogleDisk.
- Viesť agilný vývoj – konzultácia so zákazníkom = vedúcim projektu.
- Za každý šprint je zodpovedný 1 člen tímu.
- Každej úlohe definovať, kedy sa dá označiť za splnenú a dopredu odhadnúť ako dlho bude trvať jej vypracovanie – dobré pre vyhodnotenie Burn Down Chart.
- Dokumentácia – ukladať na GoogleDisk – odkazy z webového sídla projektu na PDF:
  - Z priebehu šprintov – aké úlohy boli zadané a ako dopadli – nahodiť do TFS.
  - O produkte BigPicture, odkazy na dokumenty o implementácii.
  - Zápisy zo stretnutí.

- Zozbierané dáta sú vo formáte RDF:
  - Resource Description Framework.
  - 3 zložky – objekt, predikát, subjekt.
  - funkcia podobná ako XML – Extensible Markup Language.
  - aby dáta vedeli čítať a premieňať na znalosti nie len ľudia, ale aj stroje.
  - objekt je URI, o čom hovoríme, predikát je čo o objekte hovoríme a subjekt je znalosť.
  - chceme dosiahnuť Link data – pospájať všetko so všetkým – súvislosti – vznikne graf. Napr. máme výpis – tento commit – tento autor – tento názov – tento čas, ... –> uložiť všetky dáta do trojíc (o, p , s).

ČO POTREBUJEME

- Desktop API:
  - Spracovanie dát – formuláre, a vizualizácia,
  - Sťahuje dáta,
  - Posiela dáta,
  - Beží u klienta, riešiť obmedzenie trafiku na sieti, výkonu – koľko jobov berie – možnosť „vypnúť“ alebo „pozastaviť“.
- Server – webový – prijíma dáta a spracováva.
- Balancer – dáva info Deamonovi, čo sa má kedy sťahovať, posielať, desktopy si pýtajú úlohy.

- Deamon po prijatí príkazu z Balancera spúšťa job – pošle klientovi správu – zo zariadenia – klientského PC stiahne všetky požadované dáta. Job je JSONovská informácia s parametrami pre nastavenie config file – univerzálnemu klientovi nastavím, čo chcem.

UŽ MÁME:

- Základný portál,
- Balancer – RepositoryTypeController – vráti typ repozitára podľa ID.
- TripleStore – ukladá dáta – vie naplánovať joby – JobExecutoryController – commit určujúci intervaly, od kedy do kedy chceme sťahovať.
- JobExecutionID dá potrebné informácie o tom, čo sa má vykonať.
- Job Executable – odkiaľ pokiaľ sa má vykonvať.
- model DB a model projektu (jeho komponenty: TripleStore, Balancer, Deamon,Job, Git Job, Buggzilla Job, Mylyn Job, Gerrid Job, Stack Overflow) – dodané v EA.
- V controlleroch sú controllery pre APIčka.
- Vytvorené libky na testy – rozchodené testy – podobné xUnit.
- Rozbehaný connect na databázu RDF.

TO DO:

- Rozbehať MS SQL server– databáza pre trojice v RDF, definovanie Jobov, intervalov, balancera – Express verzia zadarmo.
- Nainštalovať VIsual Studio – verzia Enterprise zadarmo.
- Pre connect na databázu potrebné nainštalovať lokálne pre vývoj Apache Jena Fuseki – downloadnúť 1 zip – rozbaliť, dá sa dať na Tomcat ako var súbor, ale lokálne stačí cez .bat súbor, beží na localhoste.
- Naštudovať SPARQL Query Language – po trojiciach: ?o ?p ?s (dokumentácia na W3C).
- Rozhodnúť stratégiu testovania – či pri commitoch, či pri celkoch, alebo ináč.
- Pri .NET RDF je chybný repozitár, neurobili ešte release s opravou, preto je potrebné naklonovať správny k sebe lokálne a po release možné stiahnuť správny vo vyhľadávaní online.
- Potrebný .Net Core – na stránke 2 veci vyskočia na stiahnutie – Visual Studio 2015 Update 3 a .Net COre 1.0.1.