切換導航 Farseerfc的小窩

- 繁體
- <u>简体</u>
- **English** 日本語

- **≜** About
- **≜** Links
- About
- <u>Links</u>
- \Box
- □ Import
- □ Life
- <u>□ Tech</u>

- <u>Import</u>
- <u>Life</u>
- □<u>Tech</u>
- Q

搜索

- 搜索
- 歸檔
- 9

ICSE 2012 <u>区</u>

- June 6
 - Keynote 1
 - Cost Estimation for Distributed
 Software Project
 - Characterizing Logging Practices in Open-Source Software
 - Combine Functional and Imperative
 Pgrm for Multicore Sw: Scala & Java
 - Sound Empirical Evidence in Software Testing
 - Identifing Linux Bug Fixing Patch
 - Active Refinement of Clone Anomaly Reports
- June7
 - Keynotes 2: Sustainability with
 Software An Industrial Perspective
 - Green IT
 - What can we do?
 - Green by IT
 - On How Often code is cloned across repositories
 - Graph-based analysis and prediction for sw evolution

- graph are everywhere
- predictors
- Conclusion
- What make long term contributors: willingness and opportunity in OSS
 - approach
 - summeray
- develop of auxiliary functions: should you be agile?
 - experiment
 - research questions
 - result
- Static Detection of Resource
 Contention Problems in Server-side
 script
- Amplifying Tests to Validate
 Exception Handling Code
- A tactic-centric approach automating traceability of quality concerns

June 6

Keynote 1

沒怎麼聽懂,只記得講到了finance is not money但 是沒聽懂這個和軟件有什麼關係。

Cost Estimation for Distributed Software Project

講到他們試圖改善現有的模型去更精確地評估軟件 開發的開銷。

他們會給PM建議之前的項目的歷史數據,然後對於新項目,他們建議歷史上已有的項目的數據,從而幫助PM得到更精確的評估。他們試圖儘量減少項目評估對PM的經驗的需求,從而幫助即使經驗很少的PM也能準確評估項目的開銷。

他們的觀點:

Context-specfic solutions needed!

我們需要更上下文相關的解決方案!

Early user paticipation is key!

早期用戶的參與是關鍵

Characterizing Logging Practices in Open-Source Software

Common mistakes in logging messages

在日誌記錄中容易犯的錯誤

他們學習了歷史上的log記錄,然後試圖找到重複修改的輸出log的語句,確定log中存在的問題。他們首先確定修改是事後修改。

通常的修改的比例(9027個修改)

45%	靜態文本
27%	打印出的變量
26%	調試等級verbosity
2%	日誌輸出的位置

他們發現有調試等級的變化,是因爲安全漏洞之類 的原因,或者在開銷和數據 之間的權衡。

大多數對log的變量的修改都是爲了增加一個參數。

他們之前的LogEnhancer是爲了 解決這個問題而提出的,通過靜態檢查,提醒程序員是否忘記了某個參數

對text的修改是因爲要改掉過時的代碼信息,避免 誤導用戶。

他們的實驗是採用了基於code clone 的技術,找到 所有log語句,然後找不一致 的clone,然後自動提 出建議。

Combine Functional and Imperative Pgrm for Multicore Sw: Scala & Java

趨勢:到處都是多核,但是併發程序呢?

他們研究的對象是Scala和Java,因爲可以編譯後確認JVM字節碼的語義。

Java:

- 。 共享內存
- 顯示創建的線程
- 。 手動同步
- Wait/Notify機制

• Scala:

- 高階函數
- Actors, 消息傳遞
- lists, filters, iterators
- while
- 共享狀態,00
- import java.* 能從java導入任何庫
- auto type inferance 自動類型推導

實驗的參與者都經過4周的訓練,實驗項目是工業等 級的開發項目

結果:

scala 的項目平均比java多花38%的時間,主要都是 花在Test和debug上的時間。

程序員的經驗和總體時間相關,但是對test和debug 沒有顯著影響。

scala的爲了讓編程更有效率的設計,導致debug更困難。比如類型推導,debug 的時候需要手動推導,來理解正在發生什麼。

scala的程序比java小,中位數2.6%,平均15.2%

- 性能比較:
 - 單核:scala的線性程序的性能比 java好
 - 4核:
- scala 7s @ 4 threads
- java 4si @ 8 threads
- median
 - 83s scala
 - 98s java
- 32core: best scala 34s @ 64 threads
- 結論
- java有更好的scalability
- scala類型推導
 - 45%說對攜帶碼有幫助
 - 85%說導致程序錯誤

- 調試
- 23%認爲scala簡單
- 77%認爲java簡單

multi-paradigram are better

Sound Empirical Evidence in Software Testing

Test data generation 測試數據自動生成

Large Empirical Studies - not always possible

For open source software - big enough

Identifing Linux Bug Fixing Patch

- current practice:
 - o manual
- Current research:
 - keywords in commits
 - link bug reports in bugzilla

Try to solve classification problem

- issue
- pre-identified
- post-identified
- data
- from commit log
- feature extraction
 - text pre-process stemmed nonstop words
- model learning

research questions

Active Refinement of Clone Anomaly Reports

motivating

- code clones, clone groups
- clone used to detect bugs
- anomaly: inconsistent clone group many anomaly clone are note bug, high false

positive

approach

reorder by sorted bug reports

June7

Keynotes 2: Sustainability with Software

- An Industrial Perspective

Sustainability

- Classic View: Idenpendent view with overlap
 - Social
 - Environment
 - Economic
- Nested viw
 - Environment
 - Social
- Economic

Triple bottom line

- economic
 - -global business, networks, global econ
- env
- natural res, climate change, population grow
- social
- awareness, connectivity, accountability

Green IT

- reduce IT energy
 - more than 50% cooling doing nothing
- mini e-waste: not properly recycled
 - 80% in EU
 - 75% in US
- foster dematerialization

In-Memory Technology: Expected Sustainable Benefits

What can we do?

- consider all software lifecycle phases in your design
- avoid energy expensive behavior in your codes
- design lean architectures

Green by IT

- 2% green IT
- 98% green IT

On How Often code is cloned across repositories

Line based hashing code clone detection
never do anything harder than sorting
hashing a window of 5 lines of normalized

(tokenized) code, dropping 3/4 of the hashing

把ccfinder一個月的工作縮短到了3,4天。沒有比較 presion和recall。

14%	type1
16%	type2
17%	type3 (not really type2)

Graph-based analysis and prediction for sw evolution

graph are everywhere

- internet topology
- social net
- chemistry
- biology

in sw - func call graph - module dependency graph

developer interaction graph - commit logs - bug

reports

experiment 11 oss, 27~171 release, > 9 years

predictors

- NodeRank
 - similar to pagerank of google
 - measure relative importance of each node
 - func call graph with noderank
 - compare rank with severity scale on bugzilla
 - correlation between noderank and BugSeverity
 - func level 0.48 ~ 0.86 varies among projects.
 - model level > func level

- ModularityRatio
 - cohesion/coupling ratio: IntraDep(M)/InterDep(M)
 - forecast mantencance effort
 - use for
- identify modules that need redesign or refactoring

EditDistance

- bug-based developer collaboration graphs
- ED(G1,G2)=|V1|+|V2|-2|V1交
 V2|+|E1|+|E2|-2|E1交E2|
- use for
- release planning
- resource allocation

graph metrics

graph diameter

- average node degree indicates reuse
- clustering coefficient
- assortativity
- num of cycles

Conclusion

"Actionable intelligence" from graph evolution

- studie 11 large long-live projs
- predictors
- identify pivotal moments in evolution

What make long term contributors: willingness and opportunity in OSS

OSS don't work without contributors form community

mozilla (2000-2008)

10^2.2 LTC <- 2 order -> 10^4.2 new contributors <- 3.5 order -> 10^7.7 users

gnome (1999-2007)

10^2.5 LTC <- 1.5 order -> 10^4.0 new contributors <- 3.5 order -> 10^6.5 users

approach

- read issues of 20 LTC and 20 non-LTC
- suvery 56 (36 non-LTC and 20 LTC)
- extract practices published on project web sites

summeray

- Ability/Willingness distinguishes LTCs
- Environment
 - macro-climate
 - popularity
 - micro-climate
 - attention

- bumber of peers
- performance of peers

regression model

newcomers to LTC conversion drops

actions in first month predicts LTCs

- 24% recall
- 37% precision

develop of auxiliary functions: should you be agile?

a empirial assessment of pair programming and test-first programming

can agile help auxiliary functions?

experiment

- pair vs solo
- test-first vs test-last
- students vs professors

research questions

- r1: can pair help obtain more correct impl
- r2: can test-first
- r3: dst test1 encourage the impl or more test cases?
- r4: does test1 course more coverage

result

- test-first
 - higher coverage
 - non change with correctness
- pair
- improve on correctness
- longer total programming time

Static Detection of Resource Contention Problems in Server-side script

Addressed the race condition of accessing database or filesystem of PHP

Amplifying Tests to Validate Exception Handling Code

異常處理的代碼不但難寫,而且難以驗證。各種組 合情況難以估計,尤其是手機 系統上。

A tactic-centric approach automating traceability of quality concerns

tactic traceability information models