切换导航 Farseerfc的小窝

文

繁體

<u>简体</u>

English

日本語

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

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

搜索

- 搜索
- <u>児档</u>
- *y*

ICSE 2012 <u></u> **区**

- 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

没怎么听懂,只记得讲到了finance is not money但 是没听懂这个和软件有什么关系。

Cost Estimation for Distributed Software Project

讲到他们试图改善现有的模型去更精确地评估软件 开发的开销。

他们会给PM建议之前的项目的历史数据,然后对于新项目,他们建议历史上已有的项目的数据,从而帮助PM得到更精确的评估。他们试图尽量减少项目评估对PM的经验的需求,从而帮助即使经验很少的PM也能准确评估项目的开销。

他们的观点:

Context-specfic solutions needed!

我们需要更上下文相关的解决方案!

Early user paticipation is key!

早期用户的参与是关键

Characterizing Logging Practices in Open-Source Software

Common mistakes in logging messages

在日志记录中容易犯的错误

他们学习了历史上的log记录,然后试图找到重复修改的输出log的语句,确定log 中存在的问题。他们首先确定修改是事后修改。

通常的修改的比例(9027个修改)

45%	静态文本
27%	打印出的变量
26%	调试等级verbosity
2%	日志输出的位置

他们发现有调试等级的变化,是因为安全漏洞之类的原因,或者在开销和数据 之间的权衡。

大多数对log的变量的修改都是为了增加一个参数。

他们之前的LogEnhancer是为了解决这个问题而提出的,通过静态检查,提醒程序员是否忘记了某个参数

对text的修改是因为要改掉过时的代码信息,避免 误导用户。

他们的实验是采用了基于code clone 的技术,找到 所有log语句,然后找不一致 的clone,然后自动提 出建议。

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

趋势:到处都是多核,但是并发程序呢?

他们研究的对象是Scala和Java,因为可以编译后确 认JVM字节码的语义。

Java:

- 。 共享内存
- 显示创建的线程
- 。 手动同步
- Wait/Notify机制

• Scala:

- 高阶函数
- Actors,消息传递
- lists, filters, iterators
- while
- 共享状态,00
- import java.* 能从java导入任何库
- auto type inferance 自动类型推导

实验的参与者都经过4周的训练,实验项目是工业等 级的开发项目

结果:

scala 的项目平均比java多花38%的时间,主要都是 花在Test和debug上的时间。

程序员的经验和总体时间相关,但是对test和debug 没有显著影响。

scala的为了让编程更有效率的设计,导致debug更困难。比如类型推导,debug 的时候需要手动推导,来理解正在发生什么。

scala的程序比java小,中位数2.6%,平均15.2%

- 性能比较:
 - 单核:scala的线性程序的性能比 java好
 - 4核:
- scala 7s @ 4 threads
- java 4si @ 8 threads
- median
 - 83s scala
 - 98s java
- 32core: best scala 34s @ 64 threads
- 结论
- java有更好的scalability
- scala类型推导
 - 45%说对携带码有帮助
 - 85%说导致程序错误

- 调试
- 23%认为scala简单
- 77%认为java简单

multi-paradigram are better

Sound Empirical Evidence in Software Testing

Test data generation 测试数据自动生成

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一个月的工作缩短到了3,4天。没有比较 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

异常处理的代码不但难写,而且难以验证。各种组 合情况难以估计,尤其是手机 系统上。

A tactic-centric approach automating traceability of quality concerns

tactic traceability information models