Diplomarbeit
CPC
„an Eclipse framework for automated clone life cycle tracking and update anomaly detection“

Valentin Weckerle
weckerle@inf

- Introduction
- Requirements
- Problems
- CPC
- Current Status
- Discussion

TOC

Clone Definition

- No universal definition
  - often “defined” as the class of similarities detected by a specific Clone Detection Tool
- Different Types
  - Simple Clones
    - parametrised / gapped / reordered
  - Structural Clones
- For Us
  - whatever you copy&paste :o)
- Similarity Metric
  - edit distance, parametric differences, metrics

Related Work (1/5)

"A Language Independent Approach for Detecting Duplicated Code"

S. Ducasse, M. Rieger, S. Demeyer, Univ. of Berne

ICSM 1999

- GCC (460 kLOC, C)
  - 8.7% clones
- DBS (245 kLOC, Smalltalk)
  - 36.4% clones
- Payroll System (40 kLOC, Cobol)
  - 59.3% clones
- Message Board System (6.5 kLOC, Python)
  - 29.4% clones
- *(Metrics based)*

Related Work (2/5)

"CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code"

Z. Li, S. Lu, S. Myagmar, Y. Zhou
University of Illinois
Symposium on OS Design & Implementation 2004

- Linux 2.6.6 (4,365 kLOC)
  - 22.3% clones
- FreeBSD 5.2.1 (3,299 kLOC)
  - 20.4% clones
- PostgreSQL 7.4.2 (458 kLOC)
  - 22.2% clones
- Apache 2.0.49 (223 kLOC)
  - 17.7% clones

Related Work (3/5)

"An Investigation of Cloning in Web Applications"

D.C. Rajapakse, S. Jarzabek
National University of Singapore
ICWE 2005

- 17 Web applications
- Open & closed source
- 33 – 1719 source files
- all major programming languages
- **16 – 63% clones**
- **average 41% clones**
- stddev 15%
- *(CCFinder, 20 tokens)*

Related Work (4/5)

"An Ethnographic Study of Copy and Paste Programming Practices in OOPL"

M. Kim, L. Bergman, T. Lau, D. Notkin
University of Washington
Symposium on Empirical Software Engineering 2004

- observe 9 experienced programmers for 60 hours
- 16 C&P actions per hour (median 12)
  - 74% <line, 17% block, 8% method, 1% class
  - 25% non-trivial => 4 per hour
- C&P is software "reuse"
- C&P actions may capture important design decisions
  - crosscutting concerns

Related Work (5/5)

And many more…

- Programmers do use Copy&Paste
  - and rightly so
- Cloning is inevitable
  - Programming language limitations
  - Conflicting design goals
  - Cloning may be intentional
    - performance, reliability
- Automated Clone Detection is problematic
  - "accidental" clones / false positives
- Refactoring of clones may not be beneficial
  - many clones are short lived
  - long lived clones are hard to refactor

TOC

In a perfect world...

- *we know* about all clones
  - even “legacy” clones *(import)*
- *we know* which clones are important
- *we know* which changes to propagate
- *we know* when to drop a clone
- *we know* when to yell *(and how)*

- in short
  - *we know* the users current intention
    - even if he/she doesn't?
  - *we know* how to be of use

- but this world isn't perfect...
  - so *we guess? we gamble?*

... back in 2007

- We *know* very little
  - But we want to learn...
    - Collecting C&P clone data
    - Collecting user feedback
  - ... about the real world
    - Lab/student experiments won't do

- Tool Support - A solution (?)
  - free, open source
  - integrated into IDE
  - long term, iterative improvement
  - basis for other tools / framework
  - real world data for future research
  - as many users as possible

Goals for this thesis

- Eclipse IDE plugin
  - for Java programs
- Framework approach
  - flexible and powerful API which allows extension and modification
  - basis for future work
- Suitable for a production environment
  - multiple developers, multiple workstations
  - best effort tracking of clones with graceful fallback
- Export of collected clone data
- Only very basic, simple heuristics
- Simple user interface

Not covered by this thesis

- Advanced heuristics
  - i.e. AI based
- Empirical analysis of clone data / user behaviour
  - no data
- Thorough evaluation / validation
  - no time
- Advanced GUI features
  - i.e. linked editing/change propagation, adv. visualisation
- Advanced conflict resolution
  - on external edit or merge
- Advanced import of legacy clone data
  - multiple static clone detectors, filtering, evaluation, ...

TOC

- Introduction
  - Clone Definition
  - Related Work
- Requirements
  - A perfect world...
  - ... back in 2007
  - Goals
  - Not covered
- **Problems**
  - A closer look
  - Eclipse
- CPC
  - Architecture
  - User Interface
  - Heuristics
- Current Status
- Discussion

A closer look (1/3)

- A Framework...
  - most potential future uses of CPC are unknown
    - API requirements are unknown
    - different levels of reuse
      - add a new view
      - just reuse clone tracking
      - reuse everything, “just” with another definition of clone
    - i.e. BP Tool, LCM, Refactoring, Templates
  - flexibility is key
  - leads to complexity

- being all things to all people
  - over engineering at its best
  - hard choices

A closer look (2/3)

- Clone Tracking
  - loose one character, loose everything
    - clone positions and source need to be kept in sync, at all times
    - System/Eclipse crash...
  - automated document modifications
    - refactorings, source reformats, code completion/generation, ...
    - save actions / paste actions
  - external modifications
    - anything other than Eclipse
  - revert/undo/redo
    - ...
  - performance
    - updating 100 clones, 100 times, on one key press?
    - ECG Sensor ...
  - ...

A closer look (3/3)

- Synchronisation
  - development teams
    - multiple developers, multiple workstations
    - concurrent modification
    - intermittent network connectivity
  - CVS / 2*SVN / ...
    - many team providers out there
  - merges / merge conflicts
  - revert / checkout of specific revision or branch
  - no central server besides the repository?
    - ease of use
    - open source projects
  - API...
    - what API?

Eclipse – Love it, Hate it (1/3)

- Going where no one has gone before...
  - Lots of documentation and discussions
    - for the common problems
- non-JavaDoc Documentation
  - does only cover part of the API
  - many things are only mentioned in the JavaDoc
- "creative" API usage
  - Naming schemes, Indexes, Headlines, ...
    - geared towards the main purpose of an API part
    - but often much more can be done...
- Nice API, but...
  - ... does every-(any?)one implement it?
- Legacy API
- Lots of exploratory Eclipse source code reading

Eclipse – Love it, Hate it (2/3)

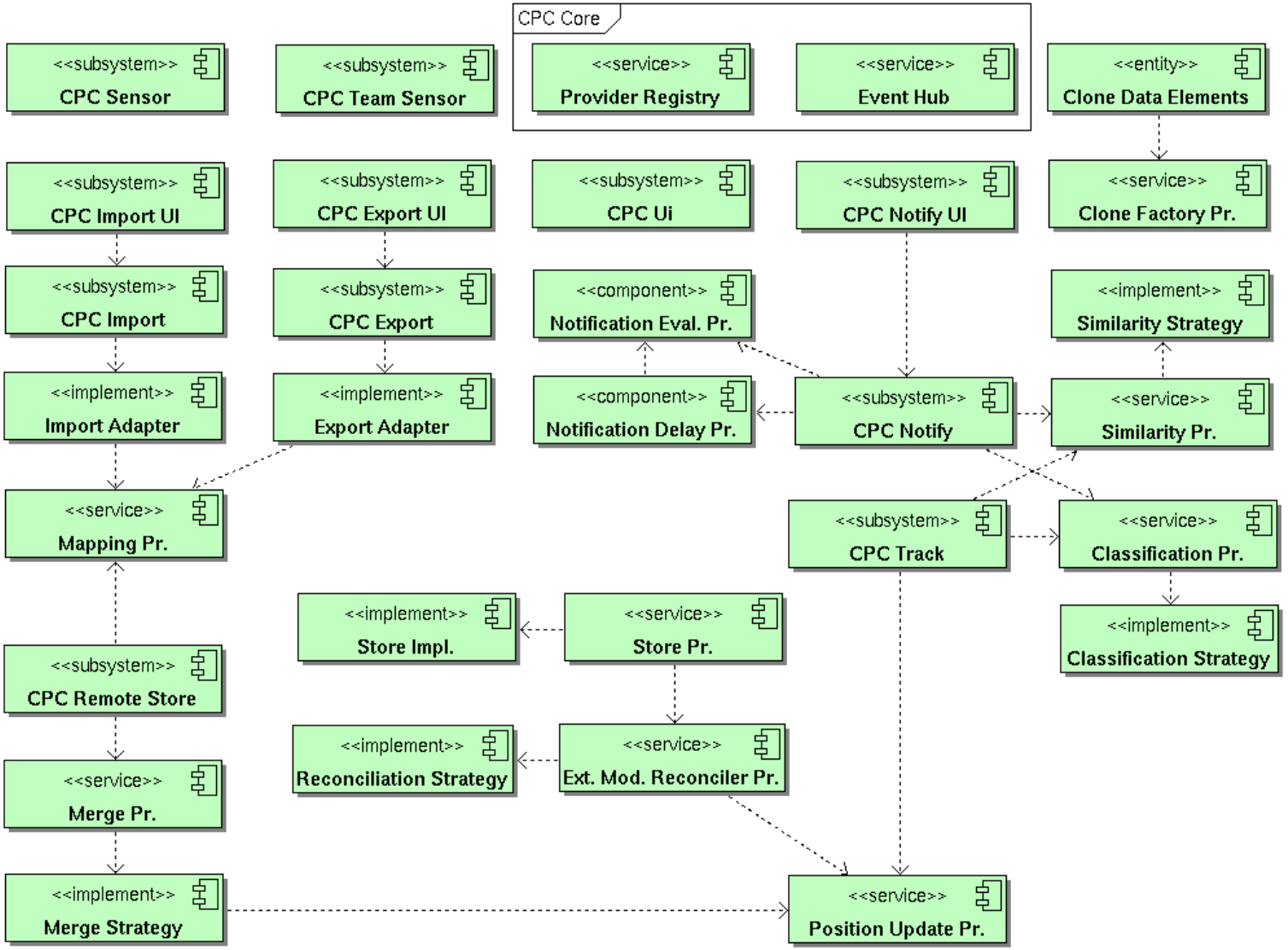
- Conservative development
  - If it is a simple bug with an indisputable fix
    - ~3-6 months
  - If there are multiple ways of fixing it
    - ~6-12 months
  - If it can't be reliably reproduced
    - open end
    - 150934: text editor synchronisation (2006-07-18)
      - gives me multiple text editor crashes a day
      - trying to get it fixed since 2007-09-04
  - If it affects the API
    - open end
    - 36418: make IMarkerImageProvider API (2003-04-11)

Eclipse – Love it, Hate it (3/3)

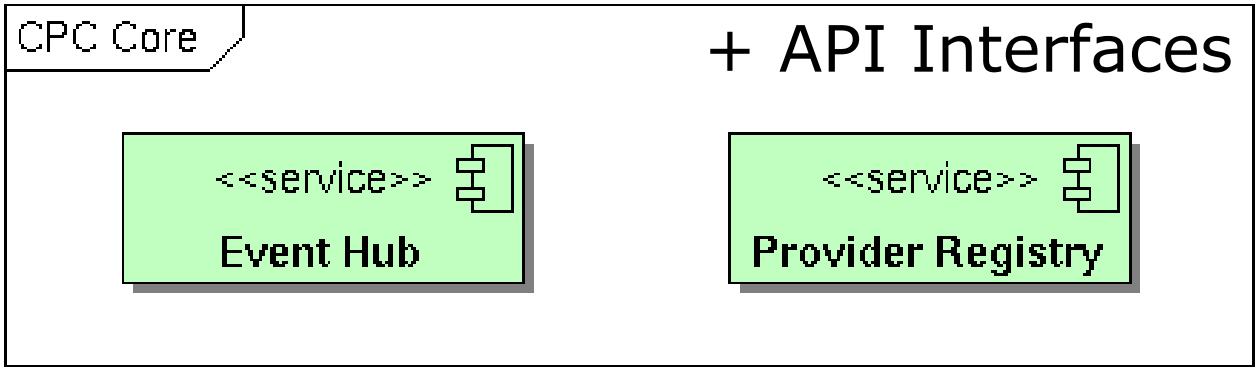
- One day during development...
  - you work on some new features
  - you make some changes to the core
- The next day...
  - Eclipse starts crashing with out of memory errors
- You waste 3 days trying to find “your” bug...
- Then you find the culprit...
  - an Eclipse auto-update introduced a new Eclipse bug
  - easy workaround exists
  - ... once you know about it

TOC

CPC Component "Overview"

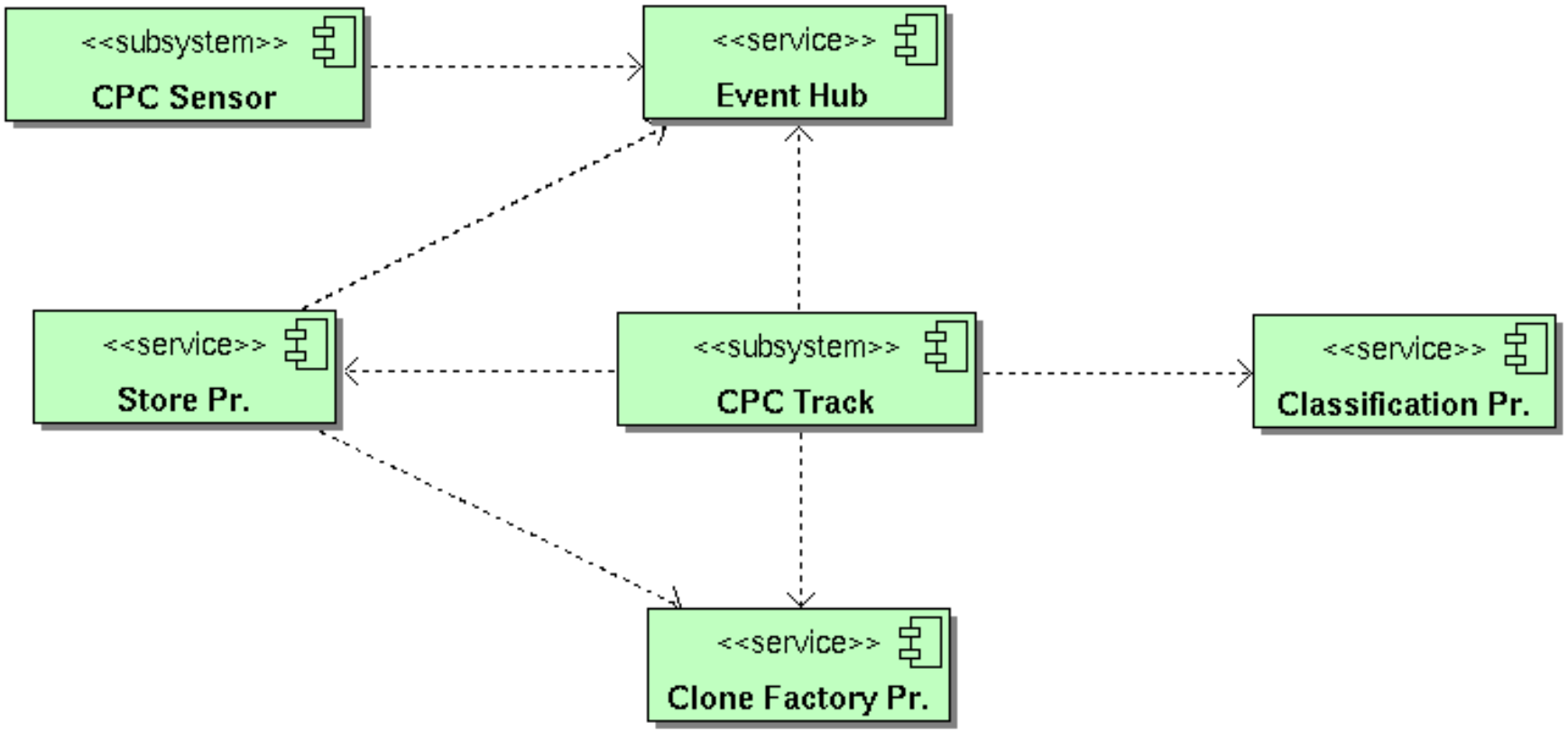


Core Components

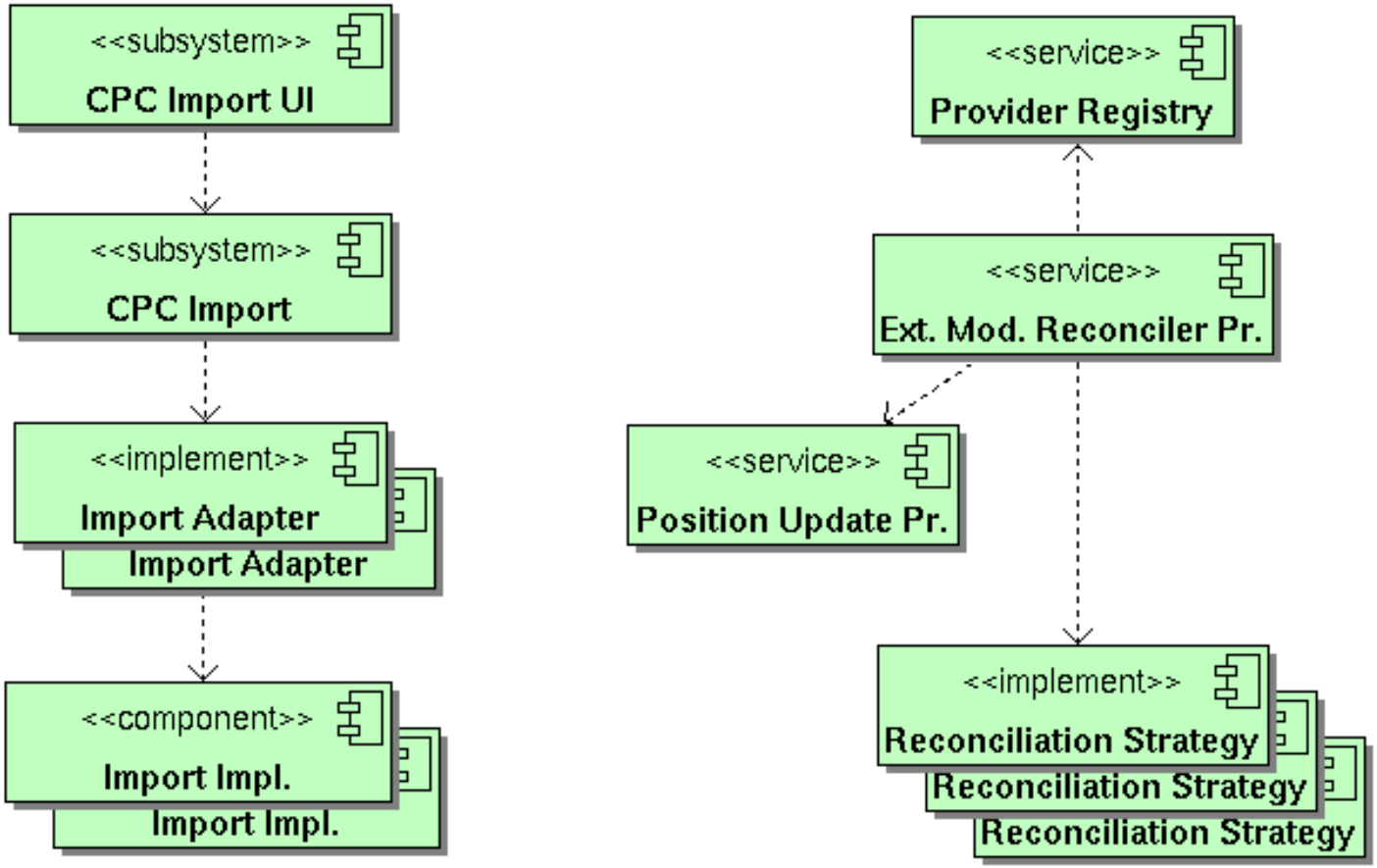


15 Event Types, 11 Provider

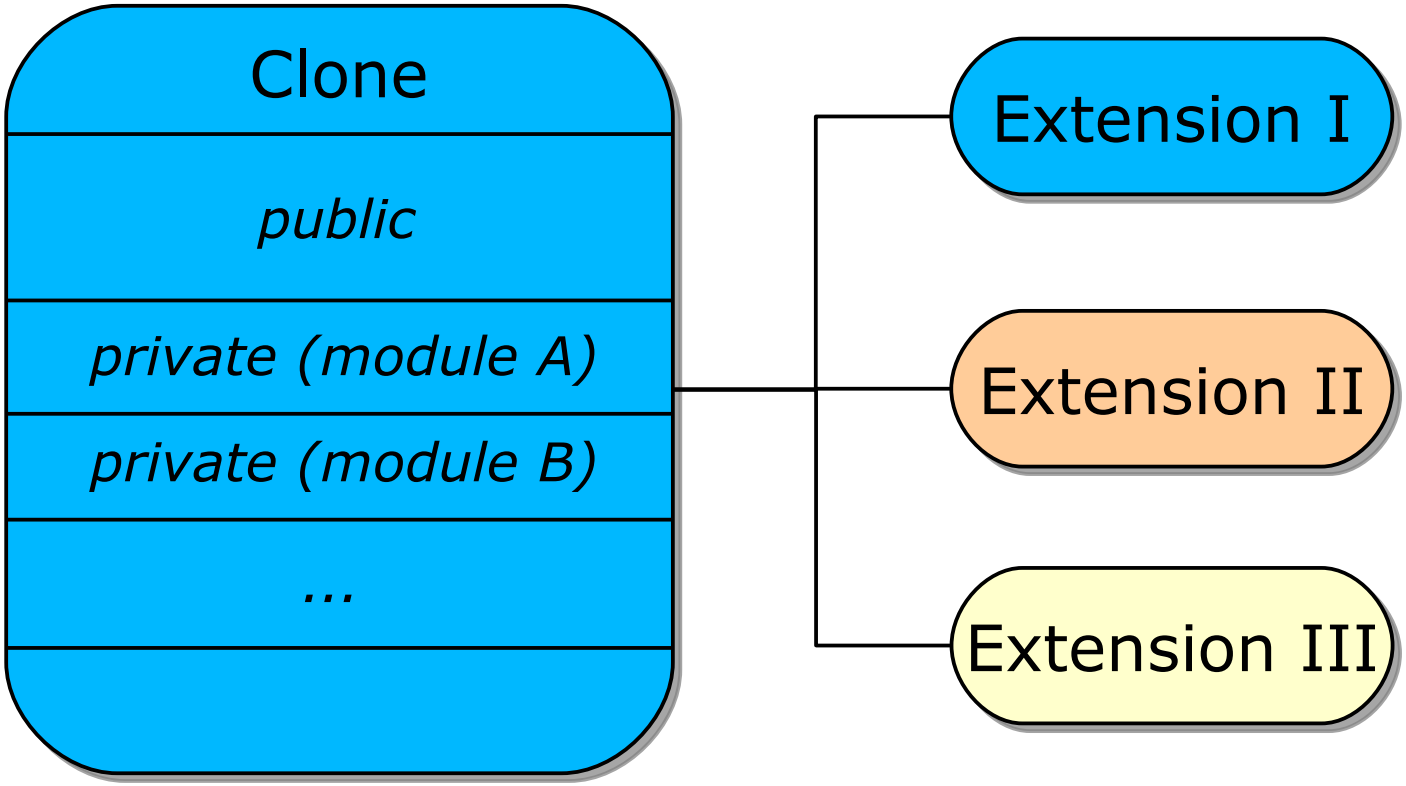
Basic Components



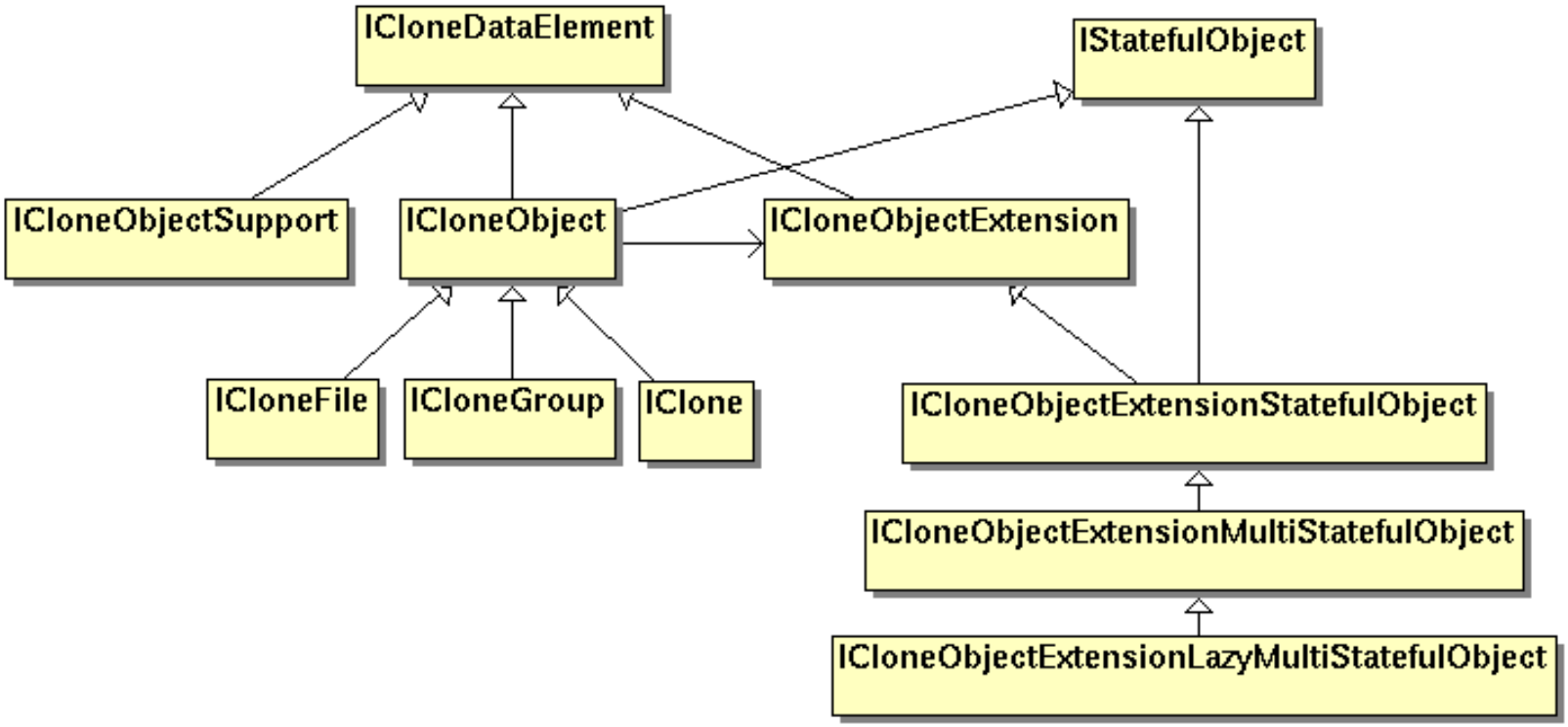
Modularisation Approach



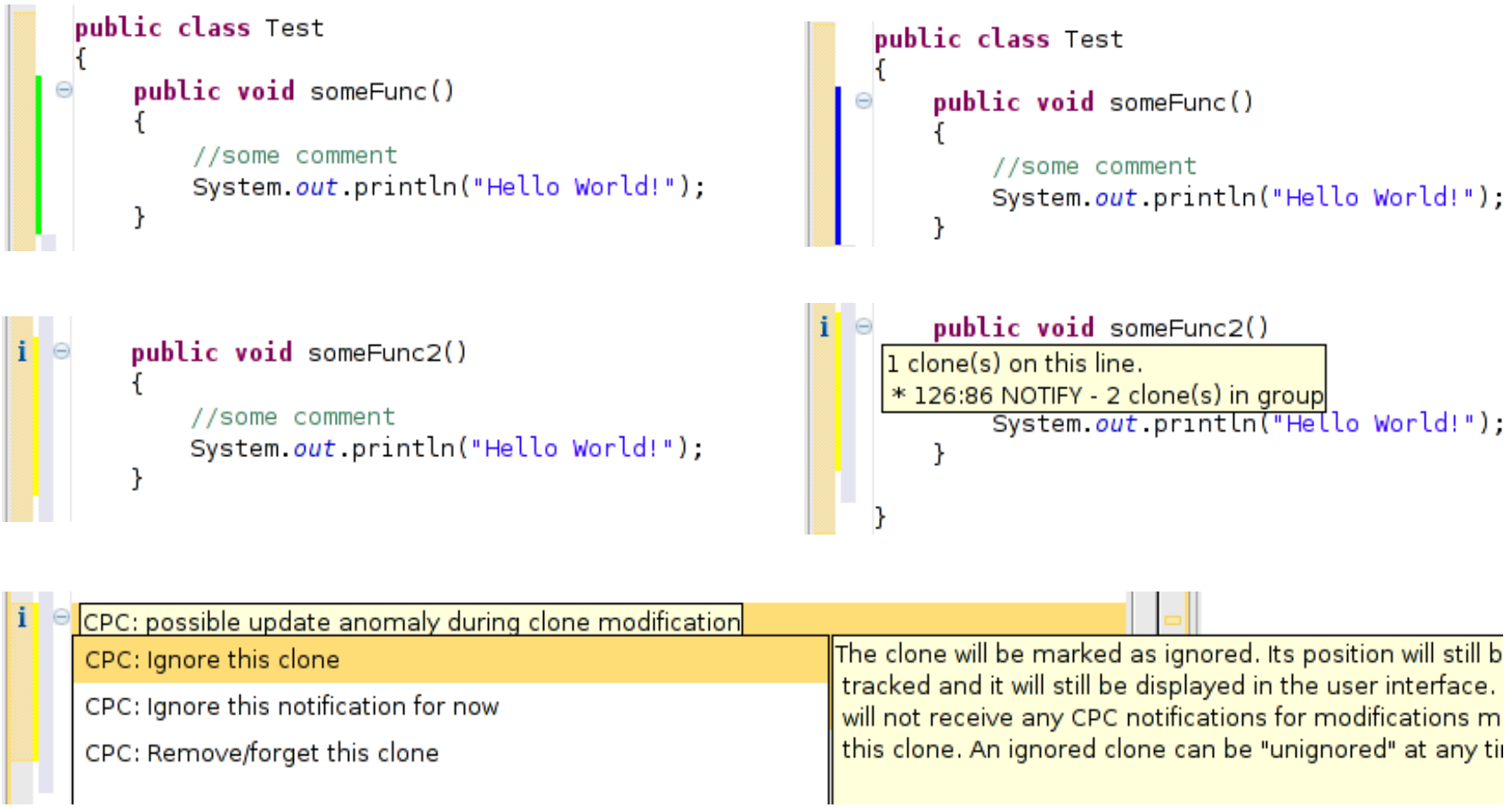
Clone Objects (1/2)



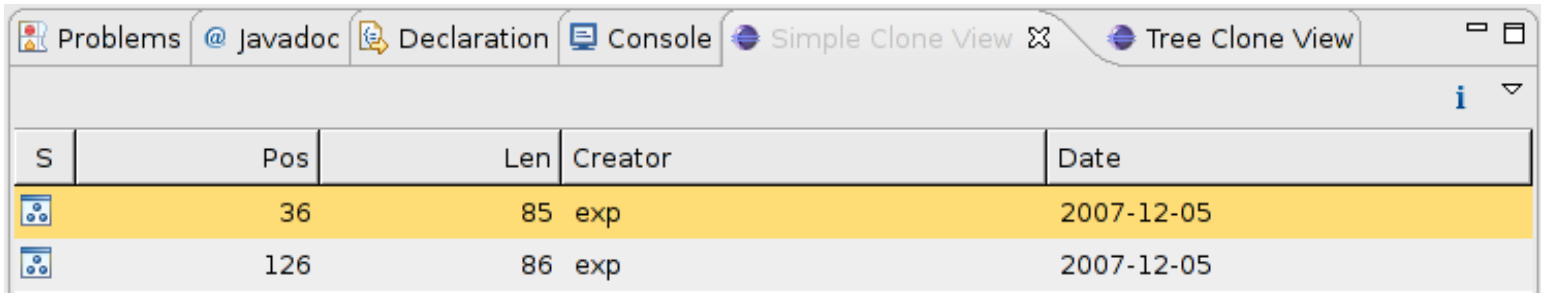
Clone Objects (2/2) (simplified)

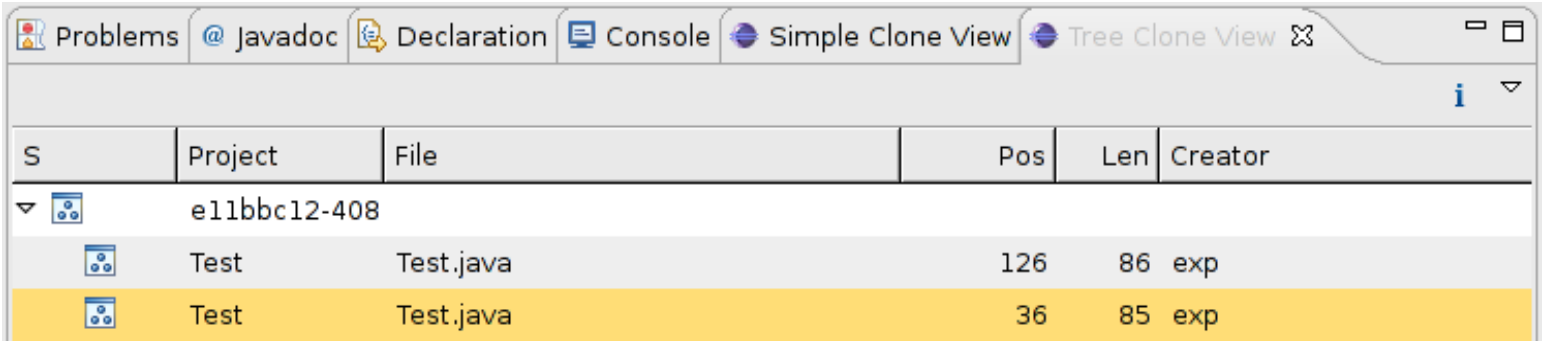


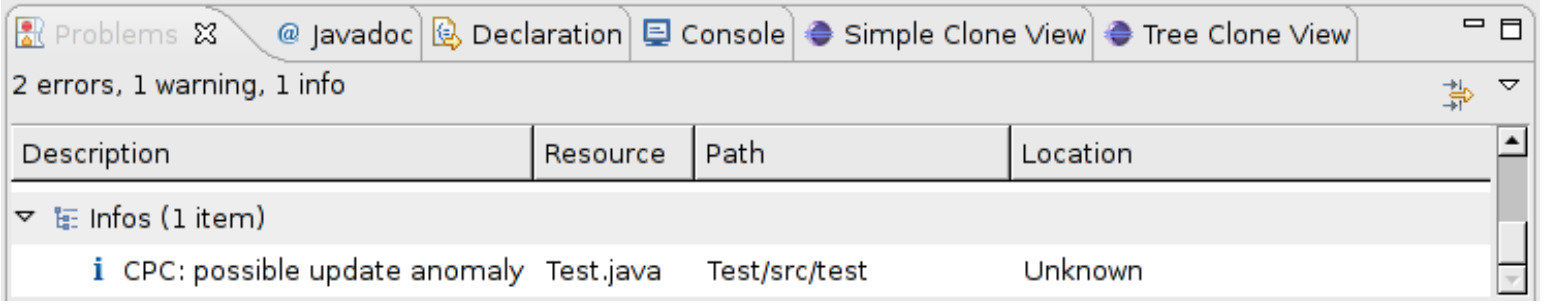
User Interface (1/3)



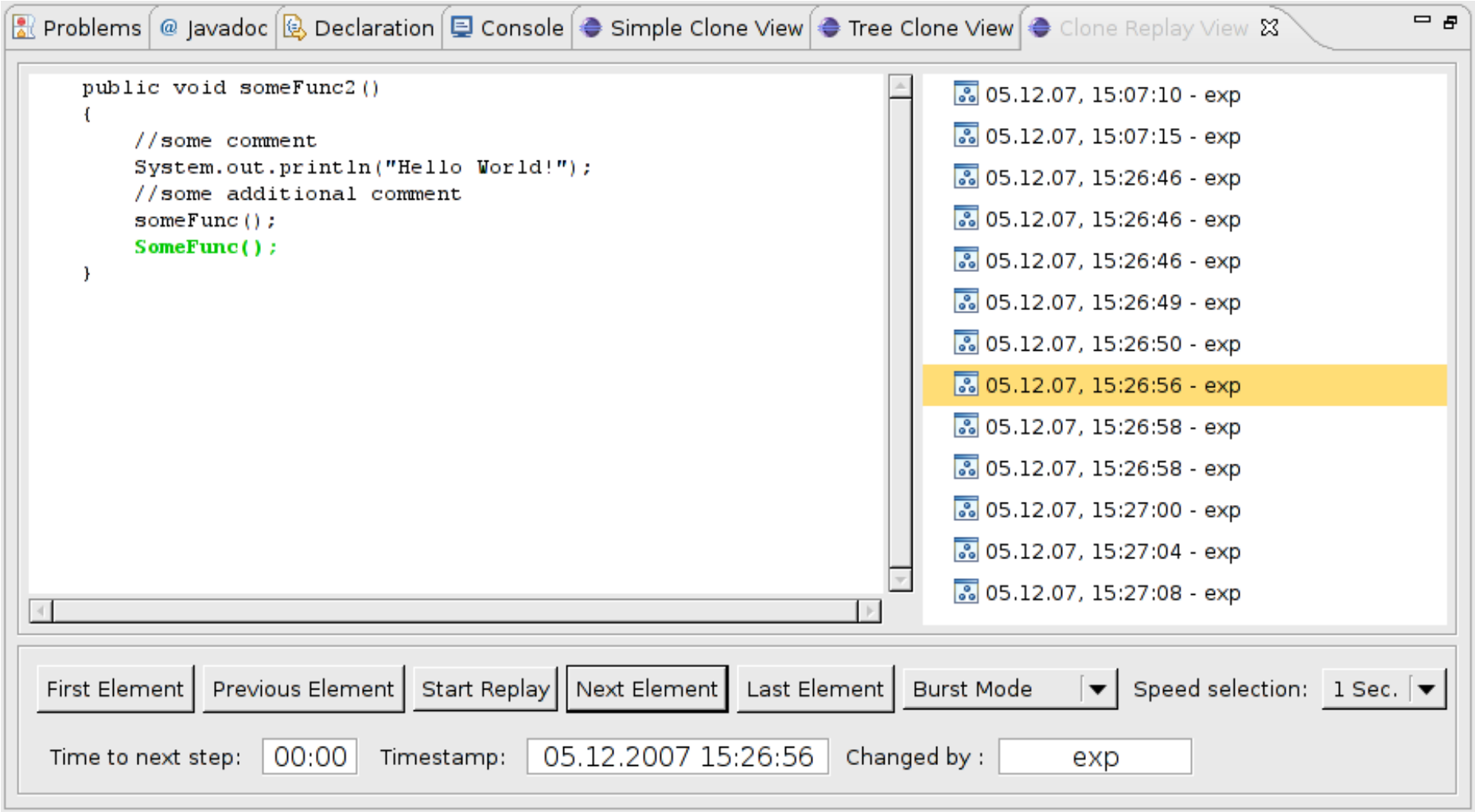
User Interface (2/3)







User Interface (3/3)



Heuristics (1/3)

- Classification
  - each clone instance is classified on creation
    - clones can be rejected by a classifier
  - classifications are arbitrary strings
  - multiple classifications are possible
  - reclassification is possible
  - rationale: performance
- Currently implemented
  - rejection: minimum token length
- Ideas
  - Class, Method, Control structure, Condition
  - Complex, Template

Heuristics (2/3)

- Similarity
  - given two clones, how similar are they?
  - result: 0 – 100 %
  - "preprocessing" and "compare" steps
- Currently implemented
  - preprocessing: generic whitespace normalisation
  - preprocessing: Java code tokeniser/normaliser
  - compare: Levenshtein Distance
- Ideas
  - preprocessing: Java parser, Identifier normalisation
    - problem: clones are not always complete code blocks
  - compare: ?

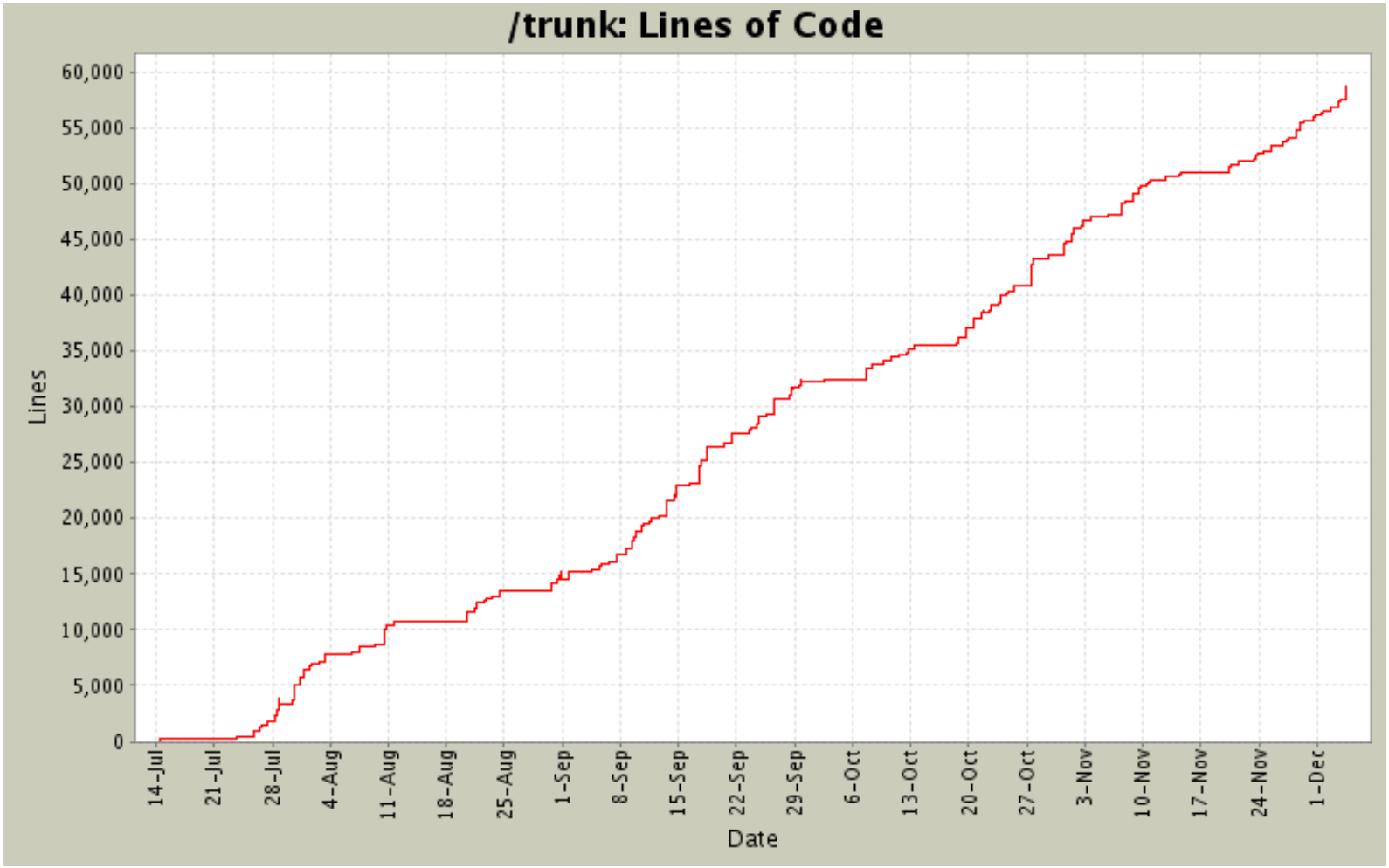
Heuristics (3/3)

- Notification
  - should a clone modification result in a notification?
  - right now or later?
- Currently implemented
  - ignore: clones with classification “Template”
  - ignore: whitespace only changes
  - ignore: “equivalent” to old state
  - ignore: “equivalent” to their origin/group members
  - delay all notifications by X seconds/minutes
- Ideas
  - ignore same class? ignore young clones?

TOC

Complexity

- 22 Plugins - 75 Interfaces - 308 Classes - 58,656 LOC



Pending Problems

- Remote Synchronisation
  - every repository provider does his own thing
    - even global APIs are often not implemented
  - no suitable APIs
    - No update/commit listener!
  - extremely time consuming
- Undo
  - performance issues
  - there maybe no good solution
- Heuristics
  - many possibilities
  - but which of them are good?

TOC

Discussion

Starters... :o)

- Who is programming mostly in Java?
- Who is using the Eclipse IDE?
- “What” do you copy & paste?
- Heuristics, Heuristics, Heuristics...
- What's next?
- 2015: Could C&P tracking save your day?