Diplomarbeit

CPC

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

Valentin Weckerle
weckerle@inf
http://cpc.anetwork.de

- Introduction
- Requirements
- Problems
- CPC
- Conclusion
- Demo / Discussion

TOC

Clone Definition

- Clone
  - No universal definition
    - in general: source code duplications / similarities
  - For Us
    - whatever you copy&paste :)
- Update Anomaly
  - inconsistent propagation of changes among members of a clone group

Related Work (1/2)

- Static Clone Detection
  - more than a decade of research
  - Cloning in Software Systems
    - pervasive phenomenon
    - lots of findings on cloning rates
      - Linux 2.6.6: 22.3% [1]
      - Closed source DMBS: 36.4% [2]
      - 17 web applications: 16 – 63% [3]

- Copy and Paste Cloning
  - very few publications so far
  - Kim et al. [4]
    - observed 9 experienced programmers for 60 hours
    - 16 Copy&Paste actions per hour
      - >= 4 per hour non-trivial

Related Work (2/2)

And many more…

- Programmers do use Copy&Paste
- Cloning is inevitable
  - Programming language limitations
  - Cloning may be intentional
    - performance, reliability
- Automated Clone Detection is problematic
  - "accidental" clones / false positives

TOC

A Problem / A Solution

- Copy and Paste Cloning
  - a serious concern
  - But: we know too little about it
    - more data needed
    - small experiments are not enough

- Tool Support - A Solution *(?)*
  - increase awareness of clones
  - detect update anomalies
  - real world data for future research
  - integrated into IDE
  - long term, iterative improvement
  - basis for other tools

Goals for this thesis

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

TOC

A closer look

- A Framework...
  - most potential future uses of CPC are unknown
    - API requirements are unknown
    - different levels of reuse
  - flexibility is key
  - leads to complexity

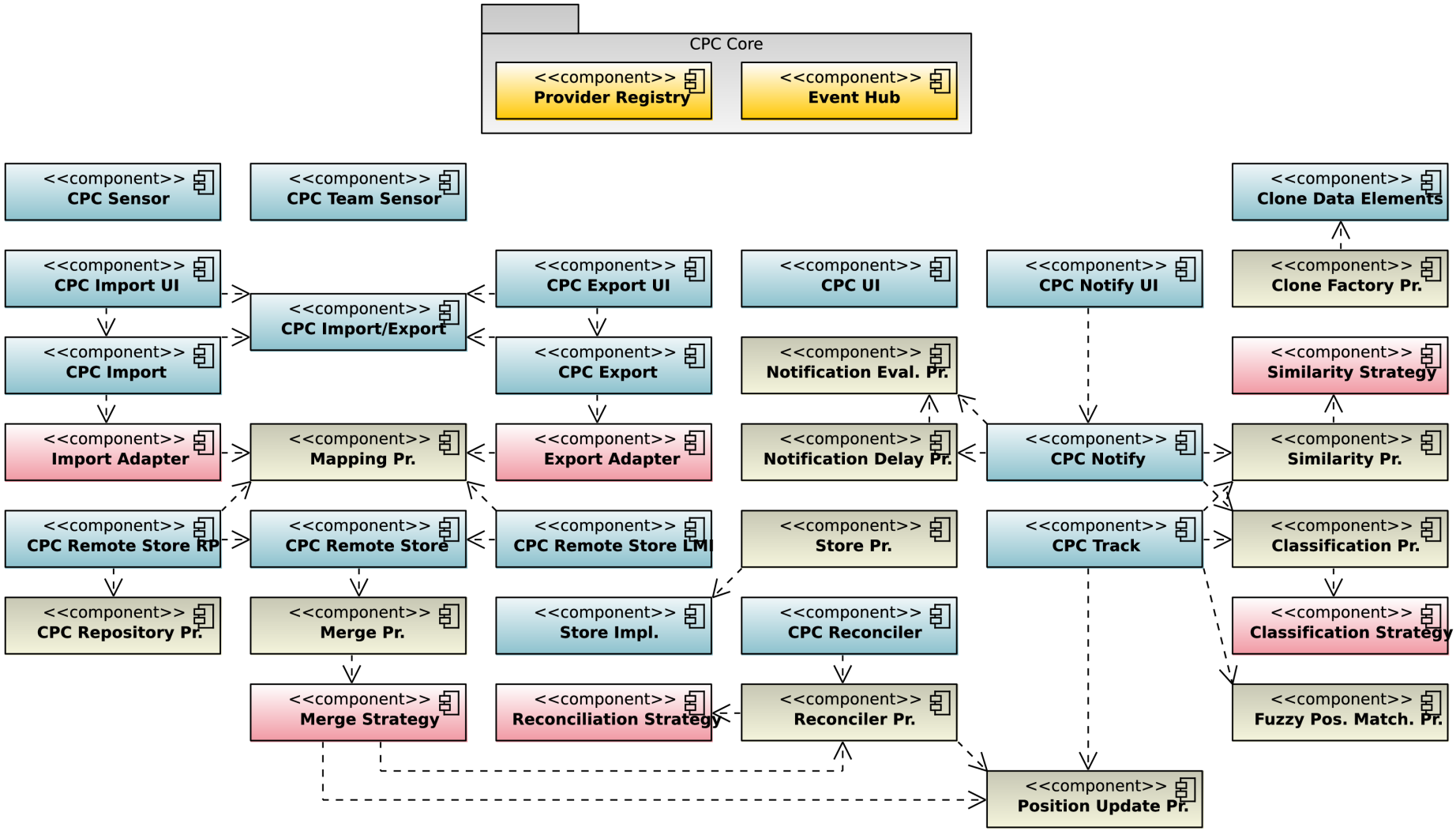
- Clone Tracking
  - loose one character, loose everything
  - automated document modifications
  - external modifications
  - revert/undo/redo
  - performance
  - *remote synchronisation*

Eclipse – Love it, Hate it

- General complexity
- Going where no one has gone before...
  - Lots of documentation and discussions
    - for the common problems
  - "creative" API usage
  - unimplemented APIs
- Lots of exploratory Eclipse source code reading
- Conservative development

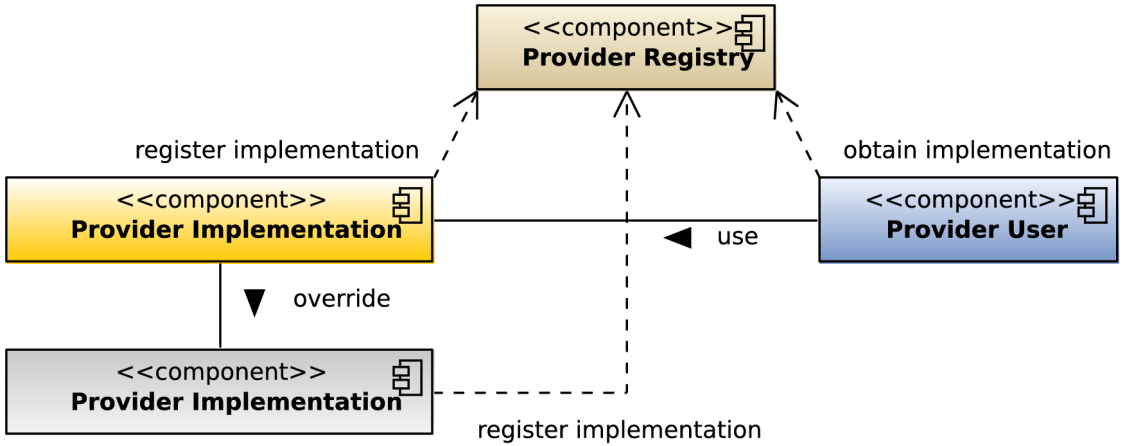
TOC

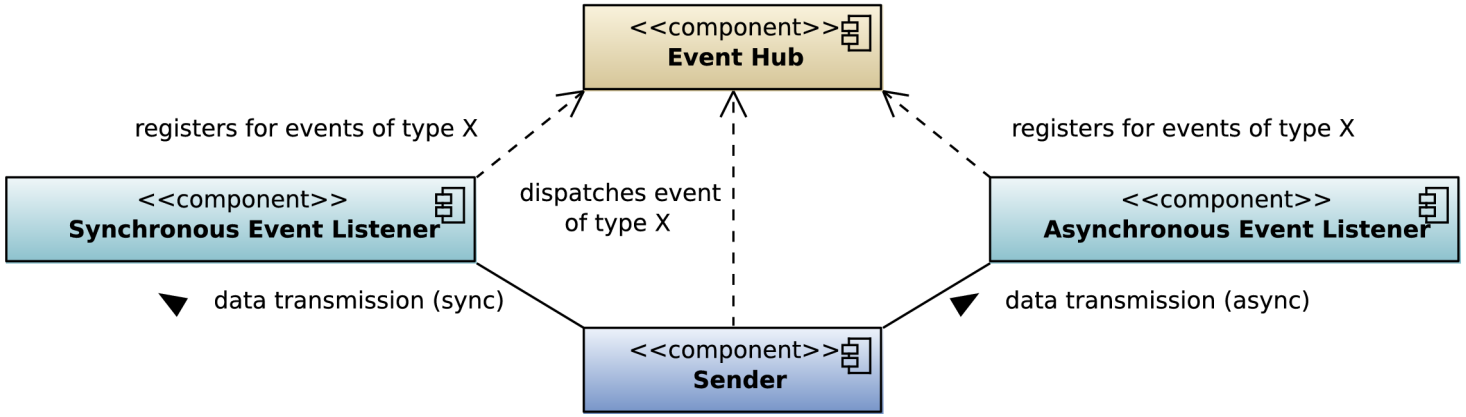
CPC Component "Overview" *(simplified)*



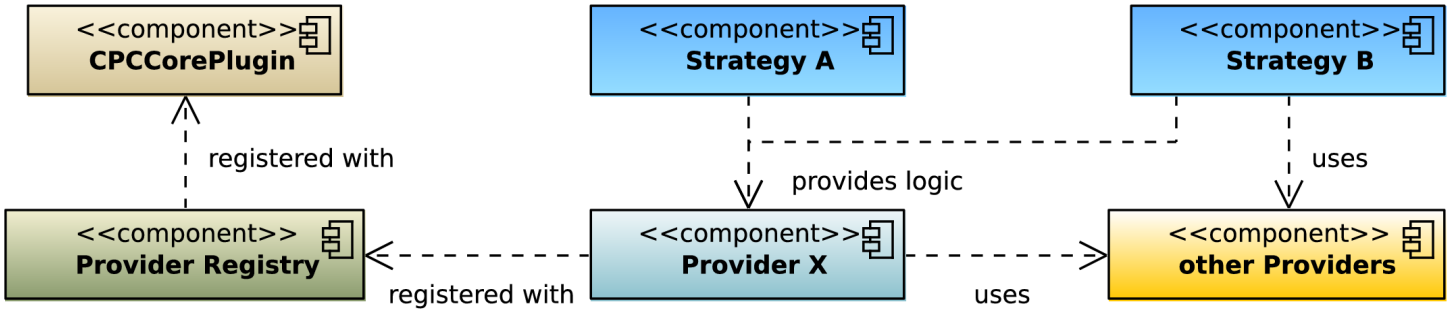
28 Plug-ins - 101 Interfaces - 355 Classes - 66,573 LOC

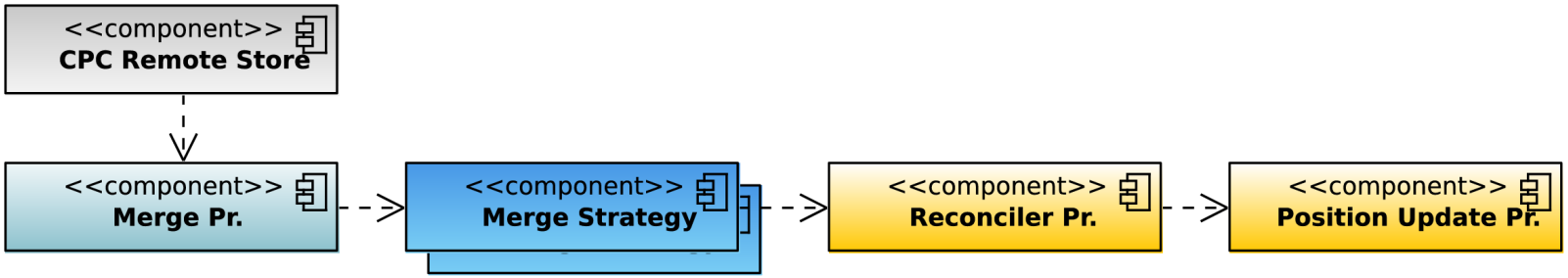
Core Components



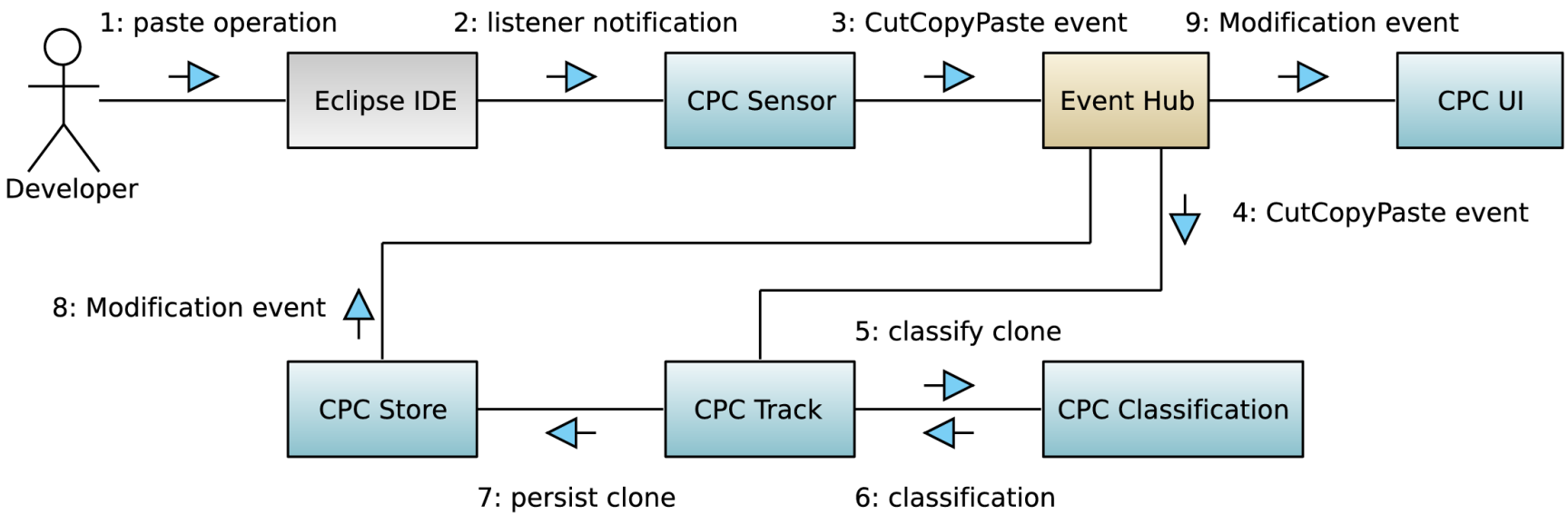


Modularisation Approach





Component Interaction *(simplified)*



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

- Currently implemented
  - rejection
    - by character count, line count or token count
  - classification
    - by content and complexity via AST
      - i.e. method, class, loop, ...

Heuristics (2/3)

- Similarity
  - given two clones, how similar are they?
  - result: 0 – 100 %
  - “preprocessing” and “compare” steps

- Currently implemented
  - preprocessing
    - generic whitespace normalisation
    - Java code tokeniser/normaliser
      - strip comments *(optional)*
  - compare
    - Levenshtein Distance

Heuristics (3/3)

- Notification
  - should a clone modification result in a notification?
  - right now or later?

- Currently implemented
  - ignore
    - clones with classification “Template”
    - whitespace only changes
    - “equivalent” to old state
    - “equivalent” to their origin/group members
    - clones in same class
    - “young” clones
  - delay
    - all notifications by X minutes

TOC

Challenges

- General Complexity / Documentation
  - many aspects not part of the specification
  - lots of source browsing
  - planning problematic
  - performance
- Inconsistent, Inappropriate or Missing APIs
  - increase complexity of CPC
- Team / Repository Providers
  - No commit / update listener API
  - Logical Model Integration API shortcomings
  - few commonalities

Test and Evaluation

- "Real World" Standalone
  - Setup
    - 3 developers, day to day development work
      - Linux, MacOS and Windows
    - started mid November
  - Results
    - no serious failure
    - 6,464 copy and paste clones
    - 40,606 clone modifications

    - most clones are small *(median size: 25 characters)*
    - most large clones are modified *(>64%)*
    - most modifications made at creation time *(median: 00:02:37)*
    - mostly small clone groups *(median: 2, avg: 2.39)*
    - most large groups created in one go *(median delay: 19s)*

Shortcomings

- Remote Synchronisation
  - Eclipse API shortcomings limit reliability
  - only prototype implementation
- Evaluation
  - only tested by 3 developers
  - larger test was postponed too long *(Remote Sync.)*
- Framework Complexity
  - Being all things to all people
    - over engineering
  - Initial steps easy, full overview difficult

Outlook

- Still lots of improvements needed
  - heuristics / clone visualisations
  - remote sync. once Eclipse APIs are improved
- Data collection and analysis
- Clarkson University
  - C&P tracking tool CnP *(work started end 2007)*
  - high overlap with CPC
  - CPC as a potential base for CnP
- National University of Singapore
  - potential use for next version of XVCL BP-Tool
  - use in clone research likely

Demo

Discussion

References

- CPC Website: http://cpc.anetwork.de
- [1] “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
- [2] “A Language Independent Approach for Detecting Duplicated Code”, S. Ducasse, M. Rieger, S. Demeyer, University of Berne, ICSM 1999
- [3] “An Investigation of Cloning in Web Applications”, D.C. Rajapakse, S. Jarzabek, National University of Singapore, ICWE 2005
- [4] “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

User Interface (1/3)

```
public class Test
{
    public void someFunc()
    {
        //some comment
        System.out.println("Hello World!");
    }
```

```
public class Test
{
    public void someFunc()
    {
        //some comment
        System.out.println("Hello World!");
    }
```

```
    public void someFunc2()
    {
        //some comment
        System.out.println("Hello World!");
    }
```

```
    public void someFunc2()
1 clone(s) on this line.
* 126:86 NOTIFY - 2 clone(s) in group
        System.out.println("Hello World!");
    }

}
```

CPC: possible update anomaly during clone modification

- CPC: Ignore this clone
- CPC: Ignore this notification for now
- CPC: Remove/forget this clone

The clone will be marked as ignored. Its position will still b
tracked and it will still be displayed in the user interface.
will not receive any CPC notifications for modifications m
this clone. An ignored clone can be "unignored" at any ti

User Interface (2/3)

Problems | Javadoc | Declaration | Console | Simple Clone View | Tree Clone View

S	Pos	Len	Creator	Date
	36	85	exp	2007-12-05
	126	86	exp	2007-12-05

Problems | Javadoc | Declaration | Console | Simple Clone View | Tree Clone View

S	Project	File	Pos	Len	Creator
	e11bbc12-408				
	Test	Test.java	126	86	exp
	Test	Test.java	36	85	exp

Problems | Javadoc | Declaration | Console | Simple Clone View | Tree Clone View

2 errors, 1 warning, 1 info

Description	Resource	Path	Location
Infos (1 item)			
CPC: possible update anomaly	Test.java	Test/src/test	Unknown

User Interface (3/3)

Problems | @ Javadoc | Declaration | Console | Simple Clone View | Tree Clone View | Clone Replay View

```
public void someFunc2()
{
    //some comment
    System.out.println("Hello World!");
    //some additional comment
    someFunc();
    SomeFunc();
}
```

- 05.12.07, 15:07:10 - exp
- 05.12.07, 15:07:15 - exp
- 05.12.07, 15:26:46 - exp
- 05.12.07, 15:26:46 - exp
- 05.12.07, 15:26:46 - exp
- 05.12.07, 15:26:49 - exp
- 05.12.07, 15:26:50 - exp
- 05.12.07, 15:26:56 - exp
- 05.12.07, 15:26:58 - exp
- 05.12.07, 15:26:58 - exp
- 05.12.07, 15:27:00 - exp
- 05.12.07, 15:27:04 - exp
- 05.12.07, 15:27:08 - exp

First Element | Previous Element | Start Replay | Next Element | Last Element | Burst Mode | Speed selection: 1 Sec.

Time to next step: 00:00 | Timestamp: 05.12.2007 15:26:56 | Changed by : exp