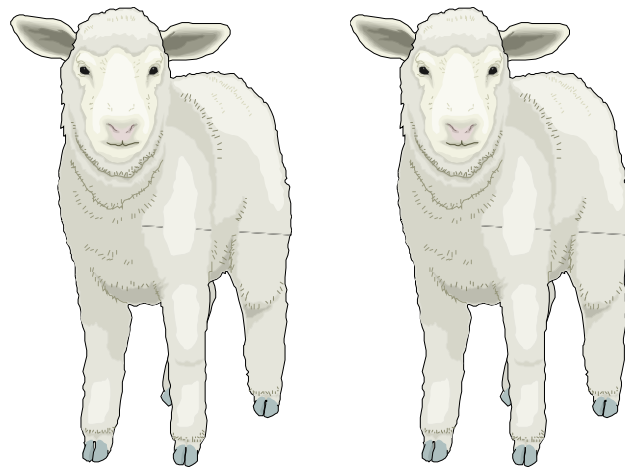


Advanced clone-analysis to support object-oriented system refactoring

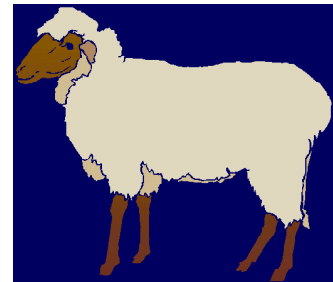
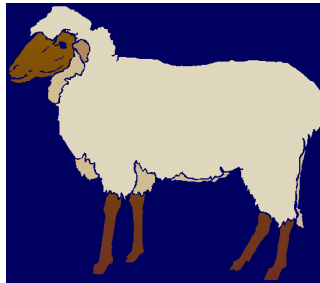
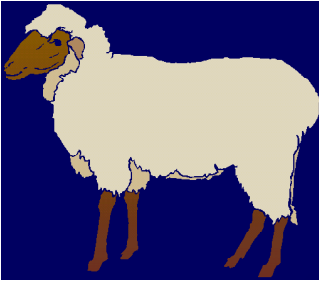
Magdalena Balazinska, Ettore Merlo, Michel Dagenais,
Bruno Lagüe and Kostas Kontogiannis

WCRE – October 2010



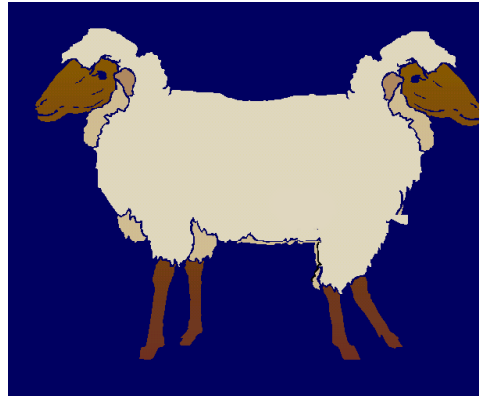
Clones

- Clones occur



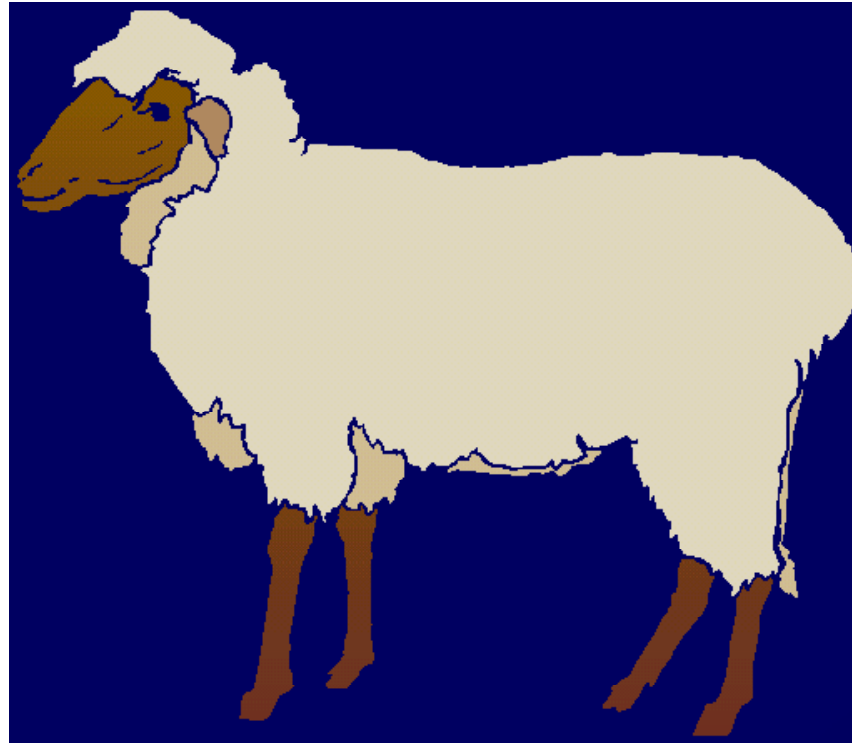
Refactoring

- ... casual refactoring?



Refactored clones

- That's better!
 - ... bigger, though ...



WCRE 2000 context

- Techniques
 - Metrics
 - AST
 - Strings matching
 - Token matching
 - Fingerprints

WCRE 2000 context

- Literature from 1994 to 2000
 - Baker B., Barson P., Baxter I., Bier L., Davey N., Demeyer S., Ducasse S., Fanta F., Field S., Frank R., Johnson J., Kontogiannis K., Leblanc C., Mayrand J., Merlo E., Moura L., Rajlich V., Rieger M., Sant'Anna M., Tansley S., Yahin A.
- Previous studies on software duplication and similarity analysis have been reported

WCRE 2000 approach

- Metrics-based similarity analysis produced clone clusters today called “classes”
- Clones in the same clone class were post-processed using token based DP matching
 - Insertions
 - Deletions
 - Substitutions

WCRE 2000 approach - 2

- Differences between clones were computed with respect to one random representative in a class
- Token based clone differences were projected over the corresponding ASTs

WCRE 2000 approach - 3

- Re-factoring opportunity were evaluated using
 - Classification of differences
 - Superficial differences
 - Signature changes
 - Type changes
 - Number of differences
 - Size of candidate clones

WCRE 2000 approach - 4

- Selected clones were automatically refactored using "strategy" and "template" design patterns
- Experimental evaluation had been performed on JDK1.1.5 from Sun Microsystems

Software Clones

- Definition
 - Two software code fragments are clones if they satisfy some similarity criterion
- Metrics based definition
 - Two code fragments are clones if their associated vectors of metrics satisfy some similarity criterion

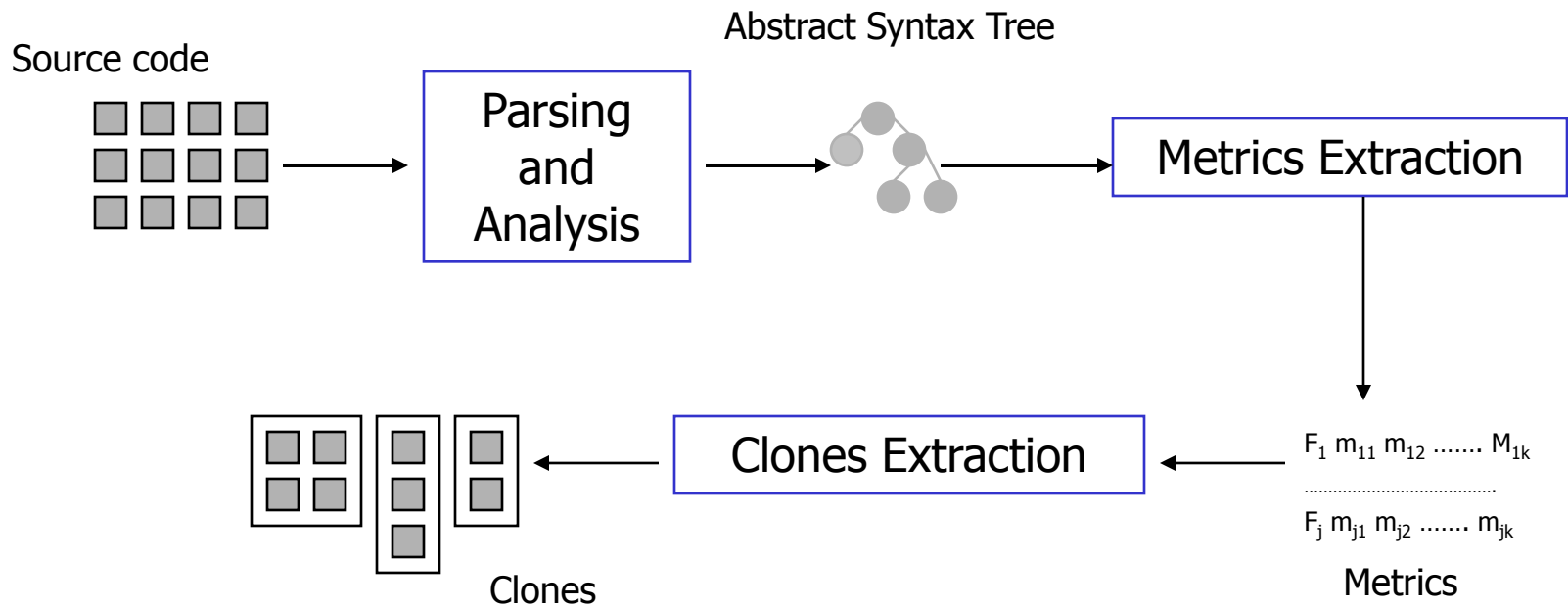
Clone Detection

- Identification of duplicated or near-duplicated components
- Duplicated components, often indicate:
 - Some sort of implicit software reuse
 - Some sort of implicit management practices (roles and responsibilities of different corporate structures)

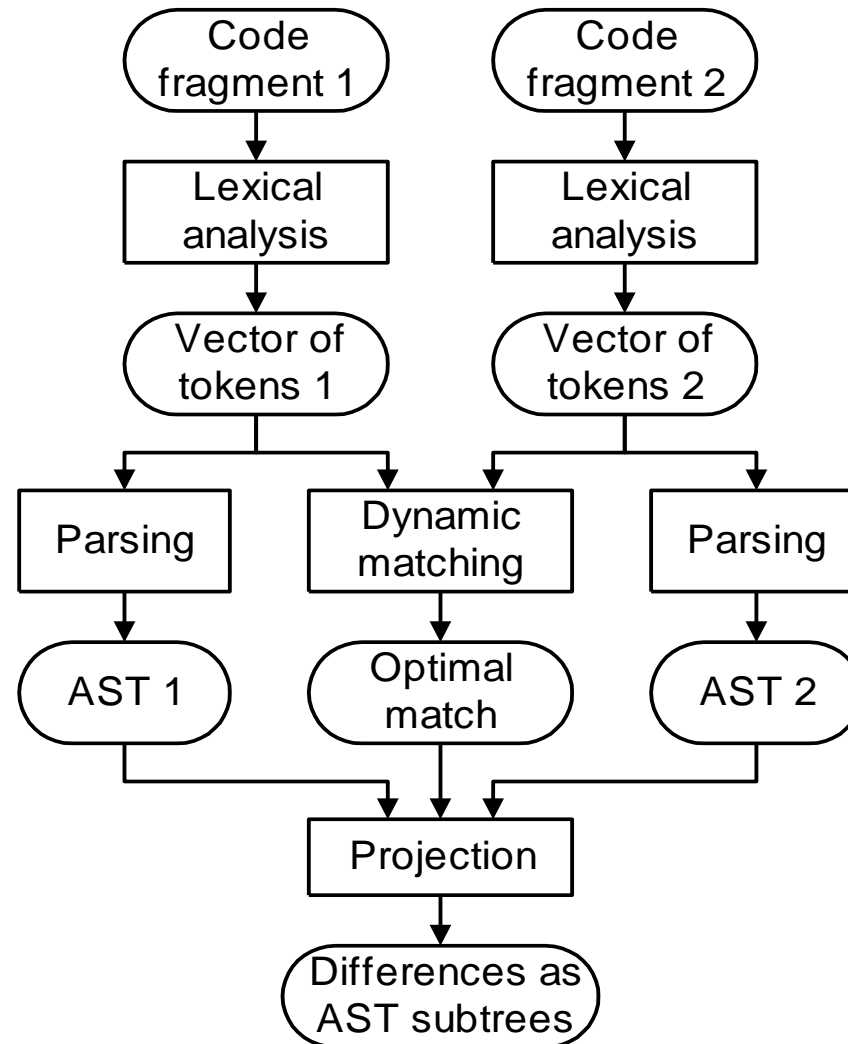
Metrics for Clone Detection

- Volume
- Complexity
- Module/function interface
- Call graph structure
- Local memory
- Global memory
- Dataflow

Clone Identification



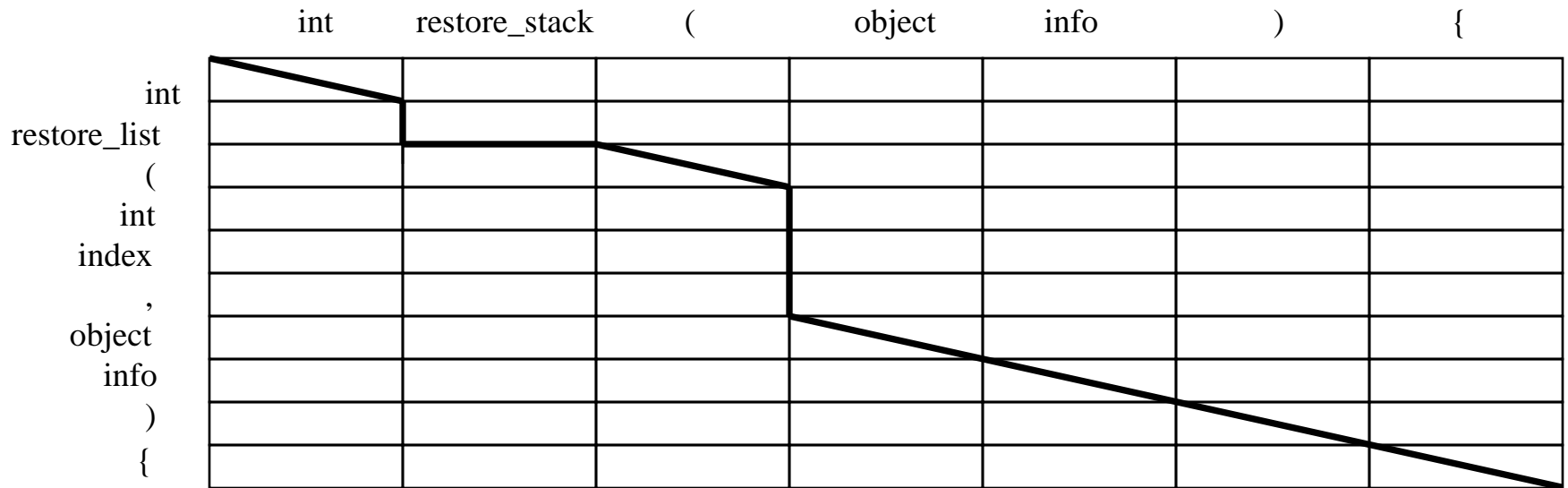
Clone Comparison



DP Matching Algorithm

- Compute the optimal sets of lexical changes using dynamic programming
 - Sub-optimal and heuristic ones exist

Matching Example



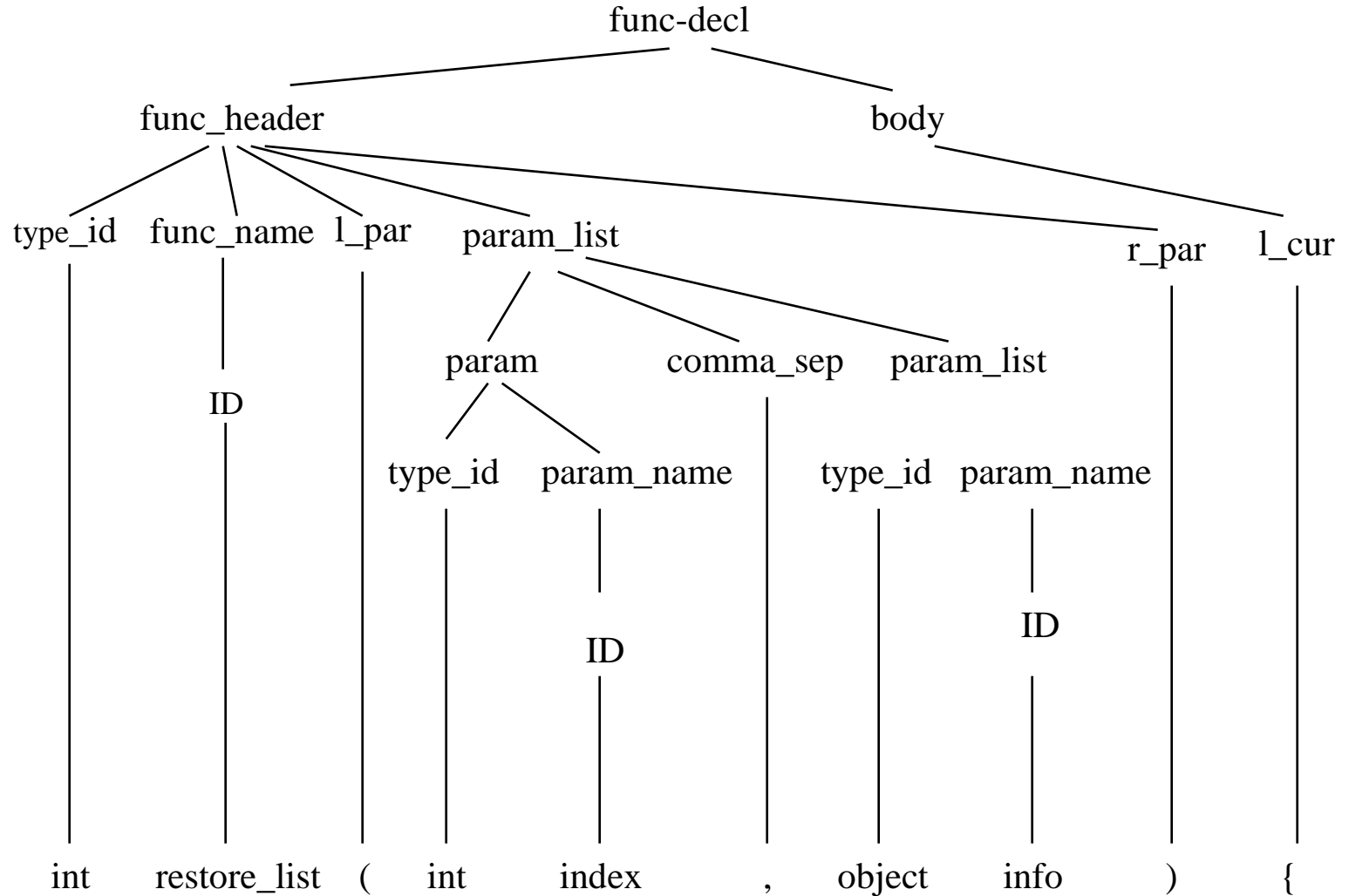
AST Comparison

- Project lexical changes onto AST's to obtain tree changes
- Definition:

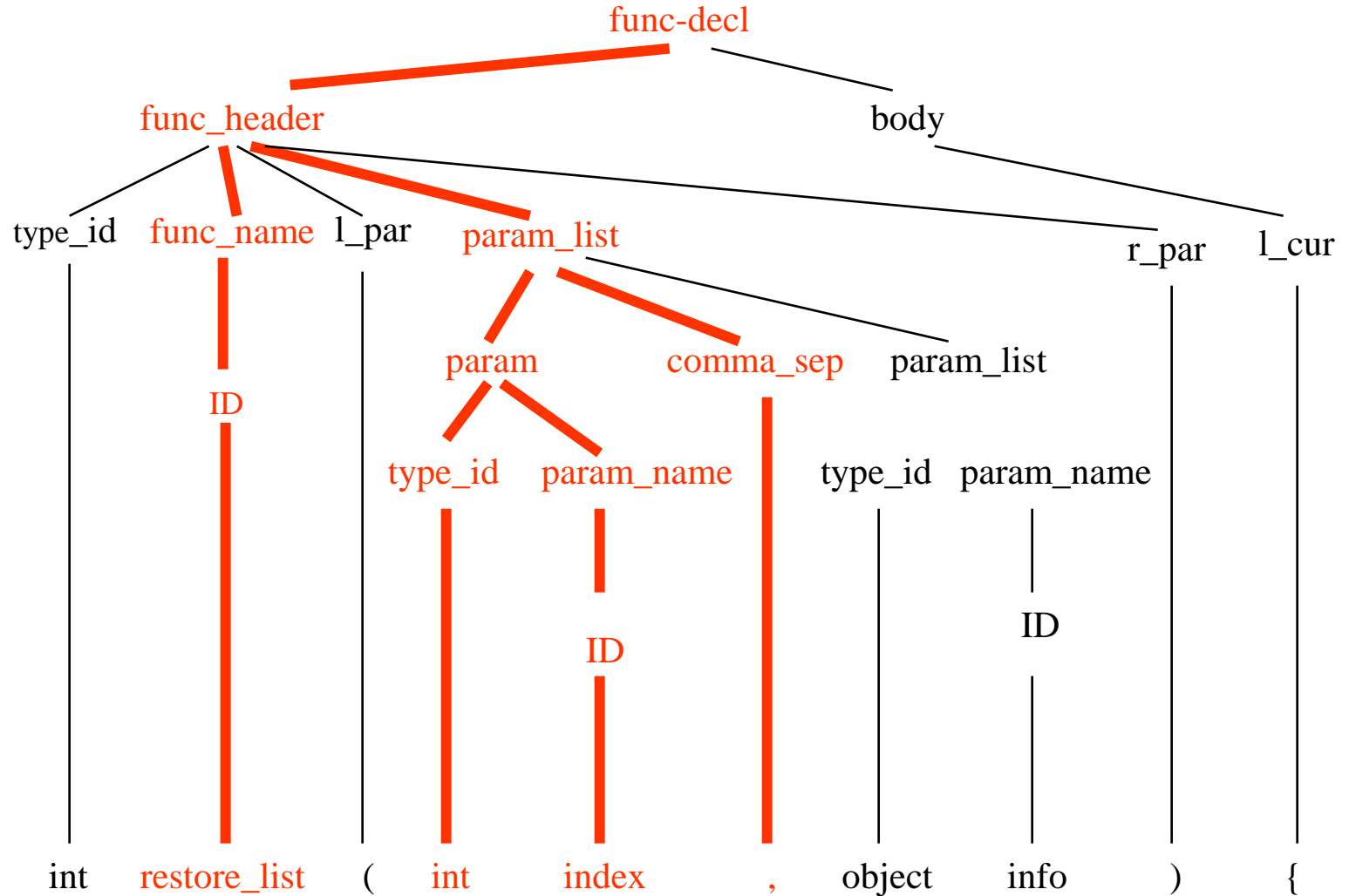
$$\begin{aligned} \text{tree} &= (V, E) \\ \text{ast_node_changed}(v) &\leftrightarrow \\ (\exists p = \langle v, \dots, v_i, \dots, v_t \rangle \mid & \\ (v_i \in V) \wedge & \\ (\text{lex_changed}(\text{token}(v_T)))) & \end{aligned}$$

- Comparison computation is linear in the size of V

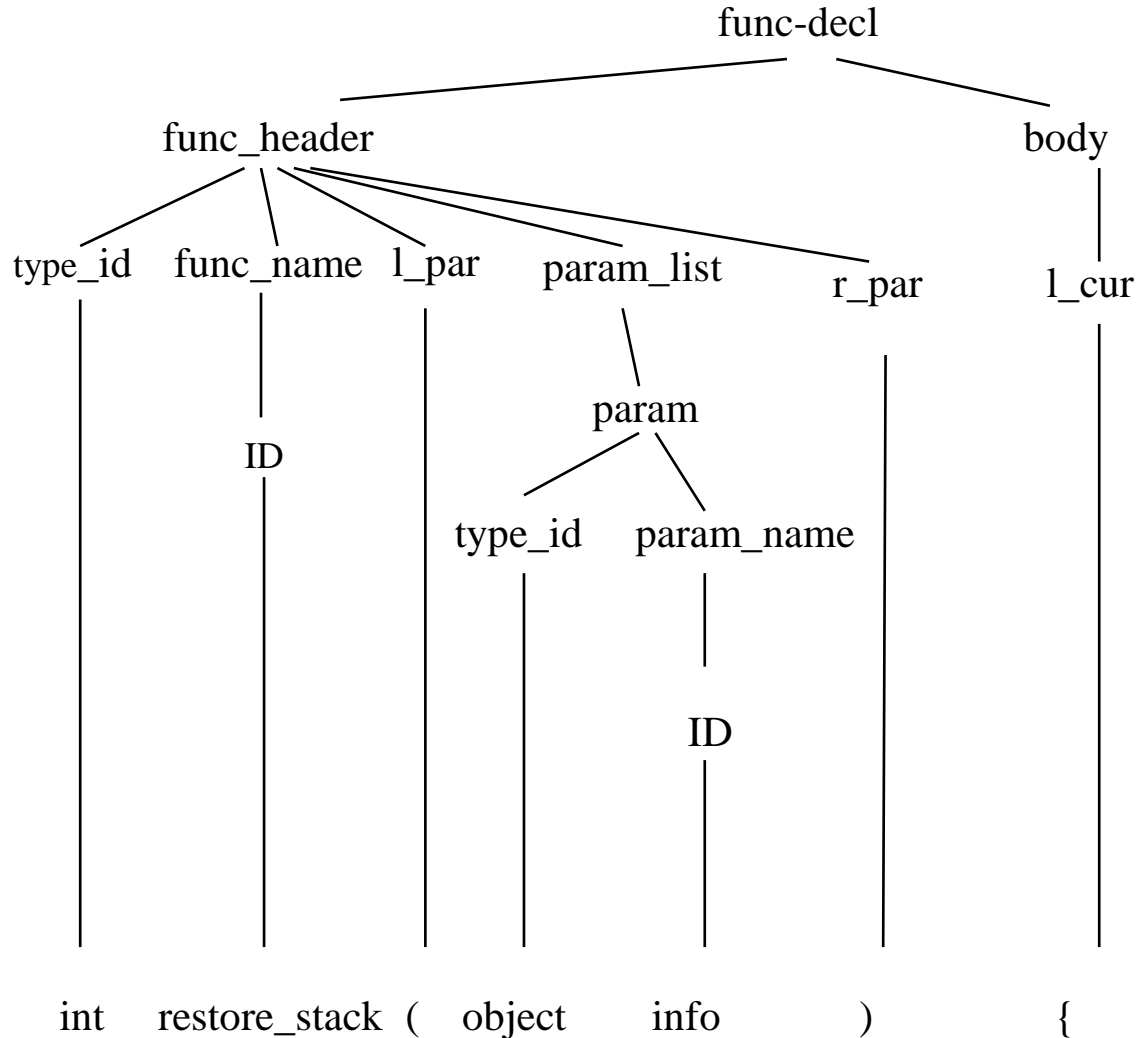
AST Projection



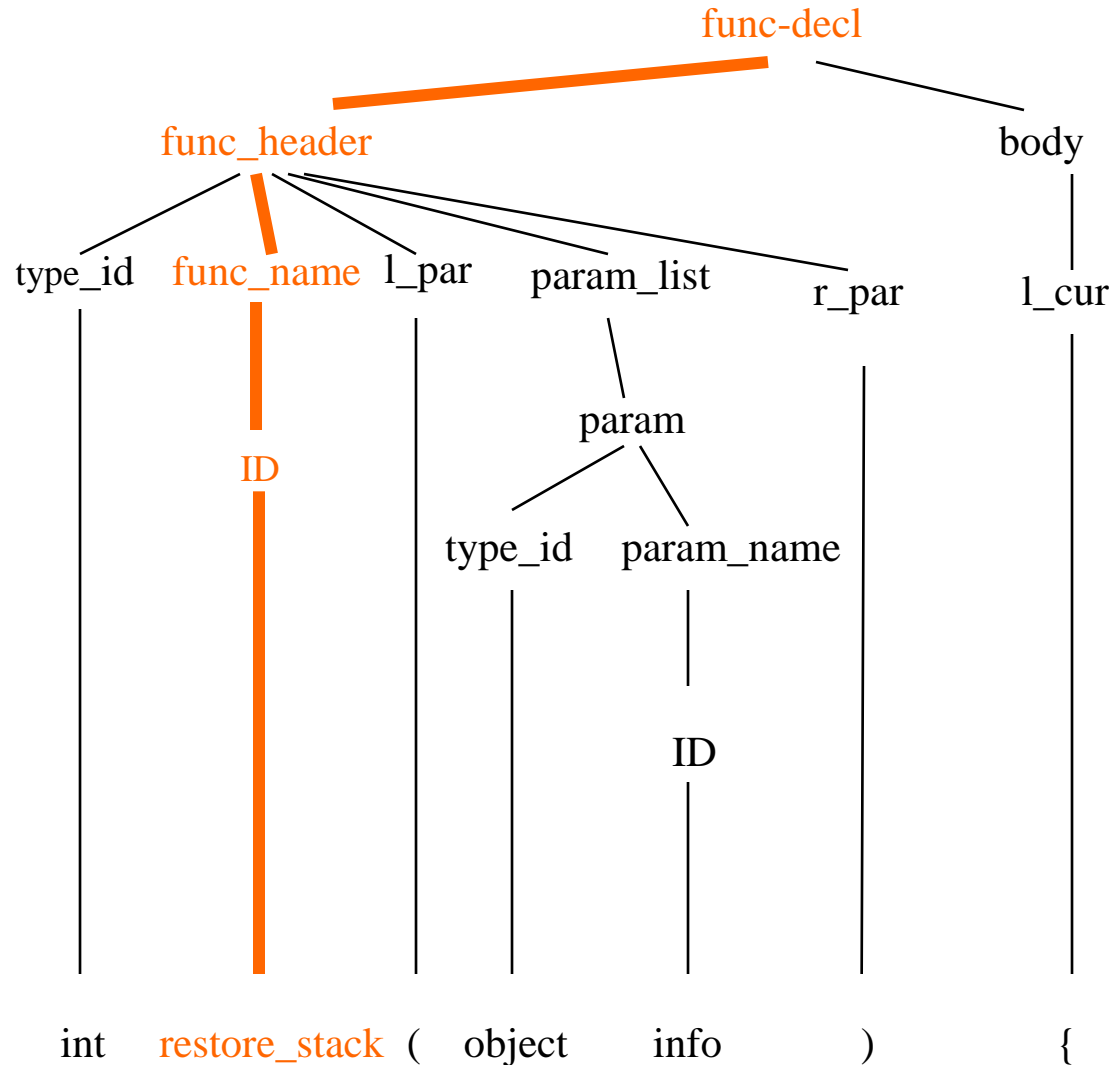
AST Projection - 2



AST Projection - 3



AST Projection - 4



Clone Classification Scheme

Category number	Type of clones
1	Identical
2	Superficial changes
3	Called methods
4	Global variable
5	Return type
6	Parameters type
7	Local variables
8	Constants
9	Type usage
10	Interface changes
11	Implementation changes
12	Interface and implementation changes
13	One long difference
14	Two long differences
15	Several long differences
16	One long difference, interface and implementation
17	Two long differences, interface and implementation
18	Several long differences, interface and implementation

Classification Results for JDK 1.1.5

Category	LOCs	Percent clones	Methods	Groups
1	432	6.0	42	11
2	42	0.6	8	4
3	464	6.4	36	15
4	198	2.6	26	8
5	54	0.7	4	2
6	229	3.2	24	9
7	0	0.0	0	0
8	146	2.0	6	2
9	14	0.2	2	1
10	12	0.2	2	1
11	135	1.9	16	5
12	326	4.5	43	10
13	106	1.5	8	3
14	130	1.8	11	4
15	33	0.5	2	1
16	828	11.4	65	24
17	158	2.2	18	8
18	224	3.1	19	7
Total	N/A	N/A	N/A	115

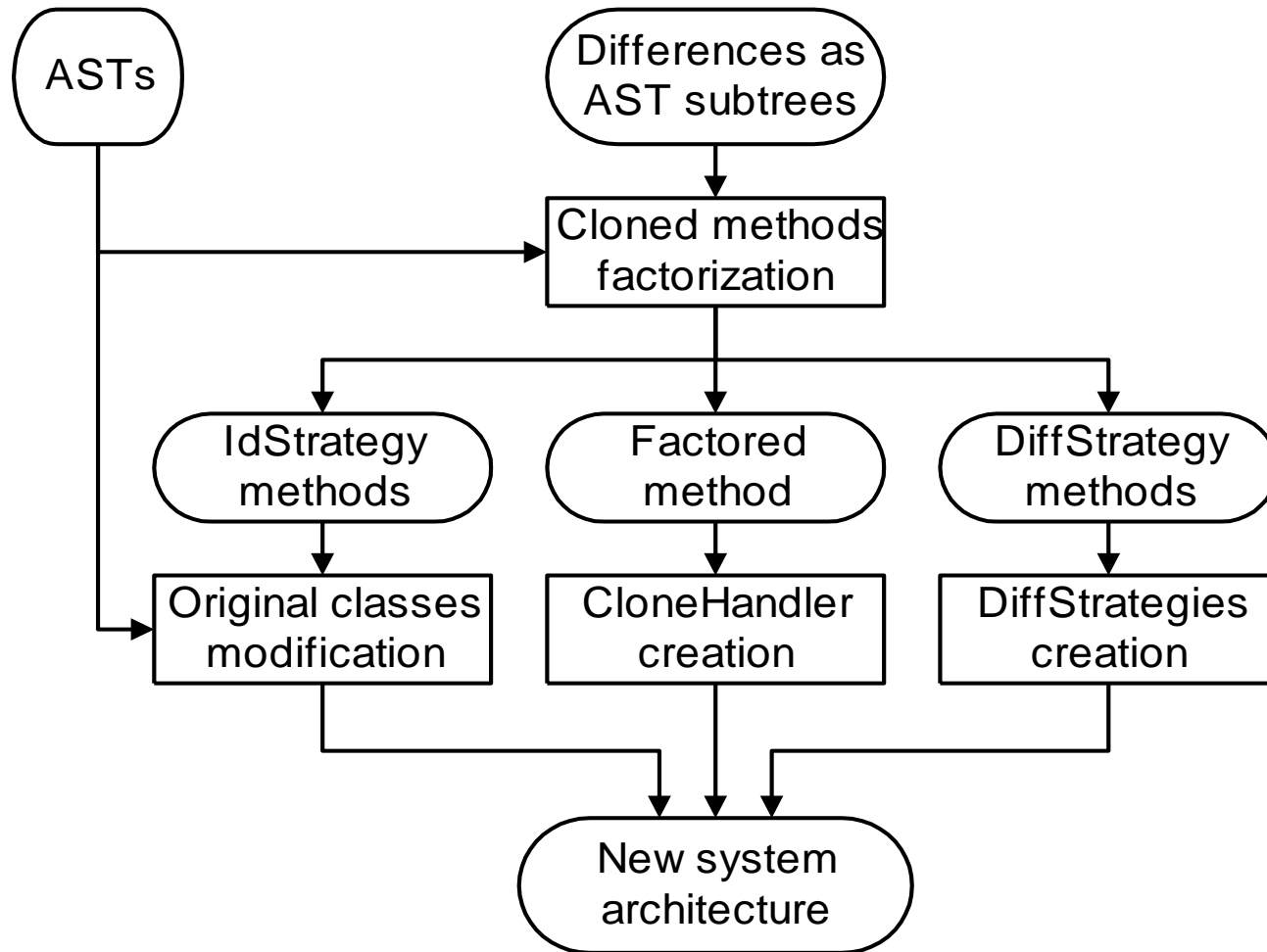
Classification Results

- Category 1 ("Identical") contains significantly more redesign opportunities than any other category.
- Categories 16, 3, 10 and 12 also contain the majority of opportunities.
- Category 7 ("Local variable type") is empty, in all considered systems.
- Clones may belong to more than one class

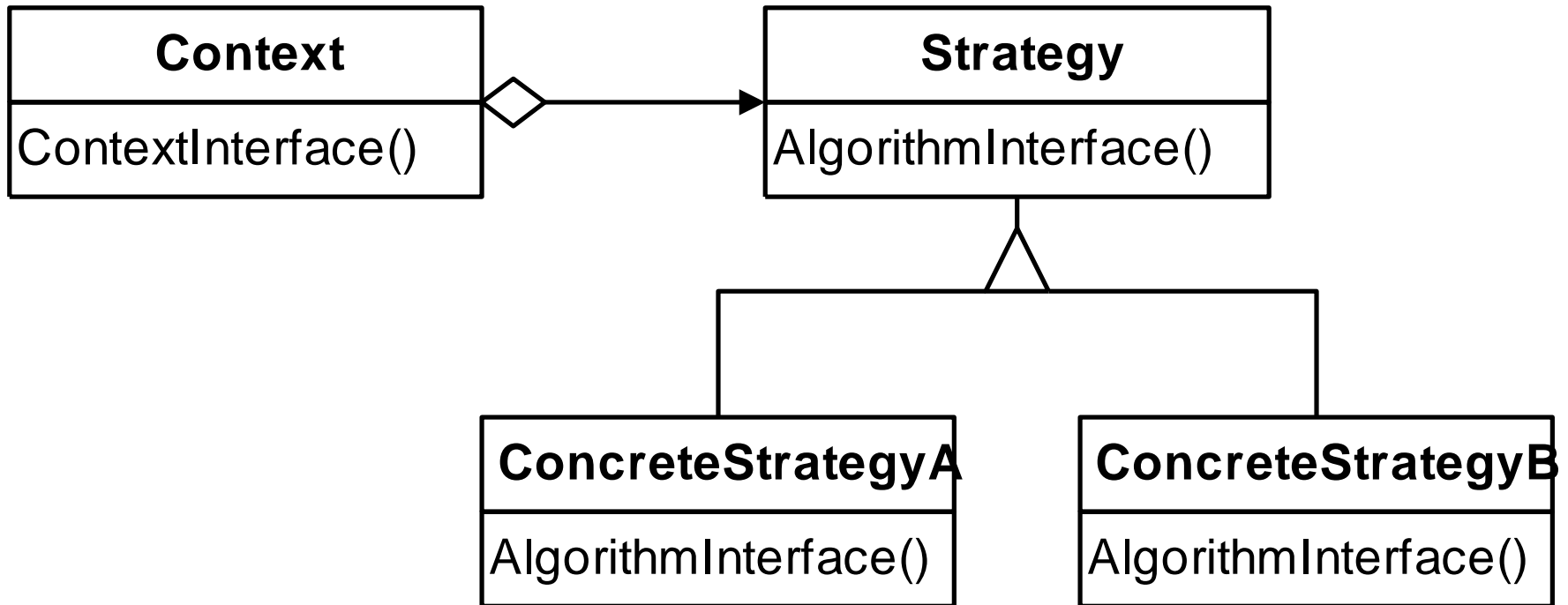
Redesign

- Factor out common parts of clones
- Encapsulate differences
- Decouple context
- Synthesize new design

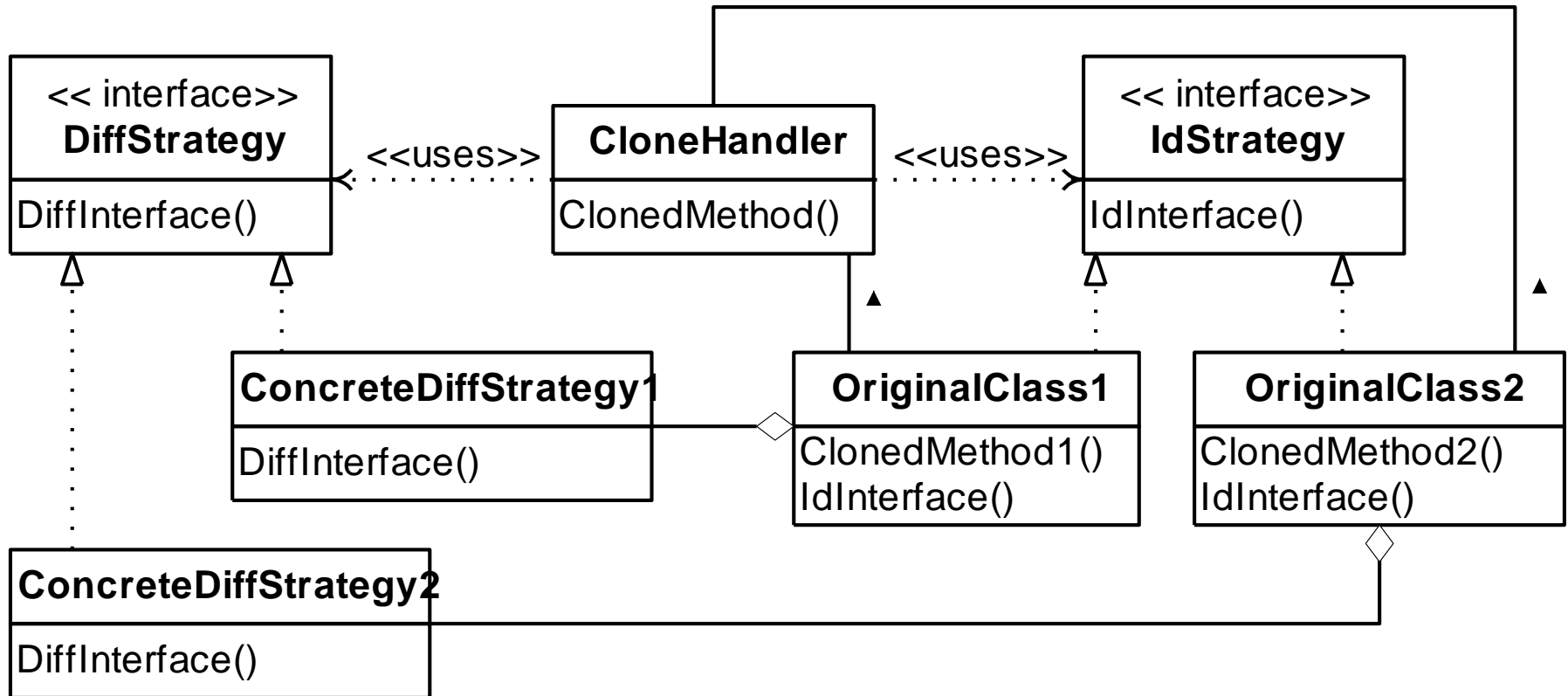
Automatic Redesign Process



Strategy Design Pattern



Use of « Strategy » in Automatic Redesign



High Impact Categories

- All opportunities are not equivalent
- High impact categories present a potentially better impact for redesign
 - Ease of manipulation of a fair amount of code
 - Large volume of code
 - Large number of methods
 - Highly clustered clones
 - Large methods

Applying the Redesign Process

- Selected categories
 - Identical clones (1)
 - Names of parameters (2)
 - Names of local variables (2)
 - Names of methods (3)
 - Use of global variables (4)

Experimental Context

- Candidate system: SUN's JDK 1.1.5
 - Java language
 - 145 KLOC (size)
- Platform
 - Pentium Pro 180 MHz
 - 64 Mbytes RAM
 - Linux 2.0.27

Redesign Results

Partial redesign of Jdk 1.1.5.

Candidate clones	120 clusters
Redesigned clones	11 clusters (28 methods)
Source code variation	+1057 lines
Percent source code variation	+0.7
Methods created	84

Advantages of Automatic Redesign

- Software maintainability may be increased
- Further reuse may be facilitated
- Source code reuse is made explicit
- Common parts of clones are factored
- Differences are parameterized in the shared component
- Context is decoupled

Disadvantages of Automatic Redesign

- Powerful analyses are necessary
- Many redesign opportunities occur with a very low frequency
- The size of the system can be slightly increased

Limitations

- Syntactic approach
 - Tokens
 - AST
 - Metrics
- Metrics matching
 - post-processing (DP matching)

Limitations - 2

- Semantics
 - Clone definition
 - Method
 - Given a definition, then search
 - Given human matched clones, then compare algorithms
- Good precision, but moderate recall

State of art

- In recent years, new relevant research contributions were published
- Several interesting surveys can be found in the literature together with a list of problems many of them are still open

Surveys

- Publications and open problems
- C.K. Roy and J.R. Cordy, A survey on Software Clone Detection Research, Technical Report 2007-541, School of Computing, Queen's University, November 2007, 115 pp.
- R. Koschke, Survey of Research on Software Clones, in Duplication, Redundancy, and Similarity in Software, Dagstuhl Seminar Proceedings 06301, 2007

Surveys - 2

- Duplication, Redundancy, and Similarity in Software, R. Koschke, E. Merlo, A. Walenstein (Eds.), Dagstuhl Seminar Proceedings 06301, 2007
- Mens T., Tourwe T., A survey of software refactoring, IEEE TSE, V. 30, No. 2, pp. 126-139, 2004
- <http://students.cis.uab.edu/tairasr/clones/literature/>
- <http://www.refactoring.com/sources.html>

Topics

- Scalability of clone detection approaches
 - Complexity
 - Execution time performance
 - Memory usage
 - Incremental approaches
- Clone visualization

Topics - 2

- Latent semantic analysis
- Prefix and suffix trees
- Clone identification using program dependence graphs
- Bugs caused by inconsistent modifications
 - Clones in a systems
 - Fragments in several software releases

Topics - 3

- Empirical studies
 - Somehow controversial empirical findings
 - Evaluation of clone detection approaches
- Web applications
- Evolution aspects
 - Evolution of clones and their lifetime over several versions of a system
 - Software evolution by computing some similarity measures between versions
 - Product lines
 - Genealogies
 - Origin analysis

Topics - 4

- Domain specific clones
 - Automotive
 - Business
- Clone management
 - Harmfulness of clones
- Intellectual property issues
 - License infringement
 - Plagiarism detection
- Malware analysis
 - Similarity of malicious code

Topics - 5

- Hybrid approaches
- Scripting languages
- Canonical representation of clones used for matching and comparison
- Similarity of structured software artifacts
 - Trees
 - Kernel based approaches
 - Graphs
- High precision hashing schemes

Topics - 6

- Specialized workshops and conferences
 - 4th IEEE International Workshop on Software Clones (IWSC 2010)
 - 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)
 - 3rd ACM Workshop on Refactoring Tools (WRT'09)

Confirmed points

- Taxonomies of clones
- Taxonomies of differences
 - Tree based
 - Priorities
- Re-factoring
- Trade-off precision and recall vs. performance
 - Need for fast performance
 - Scalability

Confirmed points - 2

- Metrics based approaches
 - Similarity measures (similarity functions) are useful
 - Precise and efficient with respect to memory usage and execution time performance
 - Robust, and stable, and inherently comparable against thresholds
 - Hashing on metrics is intrinsically easy to be incrementally computed
 - Possible agile integration in Integrated Development Environment s (IDE)

Confirmed (negative) point

- Optimal choice of representative for computing differences
 - Multiple pattern alignment
 - Many clones are very similar
 - Optimal strategy was not worth the computational cost

Current issues

- Formal definition of clones
 - Representations
 - Strings
 - Tokens
 - Syntactic
 - Semantic
 - Etc.

Current issues - 2

- Formal definition of clones (cntd)
 - Similarity definitions
 - Representation
 - Metrics
 - Feature-based
 - Behavior
 - Semantics
 - Definitions and measures are open problems
 - Execution similarity
 - Traces
 - Profiles
 - Dynamic analyses
 - Distances and thresholds

Current issues - 3

- Types III and IV clones
 - Taxonomies
 - Differences
 - Patterns of uninteresting clones
 - Low interest automatic filtering
 - Statistics
 - Statistics of intentional clones
 - Frequencies
 - Intentions
 - Multilingual flexibility

Current issues - 4

- Applications to software engineering
 - Refactoring
 - Development process
 - Management
 - Harmfulness of clones
 - Impact of code clones on software quality
 - Inconsistent modifications
 - Industrial adoption

Current issues - 5

- Evolution of clones
 - Evolution in industrial systems
 - Evolution and the development organization
 - Reliable predictors of clone evolution
- Empirical studies
 - Reference data bases
 - Comparison data

Current issues - 6

- Standard benchmarks
 - Pairs
 - Classes
 - Boolean data
 - Degrees of confidence
 - Relevance ratings
 - Type III and IV measures of similarity
 - Extensibility
 - More systems
 - More languages

Current issues - 7

- Need for effective time and memory performance
 - Low complexity algorithms
 - Scalability to large software
 - Ultra-scalable approaches
 - Distributed approaches

CLone ANalysis (CLAN) tools

- Family of software similarity analysis and related applications
 - Metrics
 - Syntactic
 - Centroids
 - DP matching

CLAN applications

- Clone detection
- Software evolution
 - Analysis of versions and releases
- Plagiarism detection
- Refactoring
 - Re-design
 - Libraries identification
- Development process

CLAN applications - 2

- Clone detection
 - Cloning ratio
 - % of clones in a system using different and increasing thresholds
 - Take “noise” around the origin of metrics into account

CLAN applications - 3

- Software evolution
 - Comparison between different versions of the same system
 - Subsequent versions
 - Reference versions
 - Issues
 - Define fragment identification
 - A function signature is not enough, neither is a position in the file system
 - Code motion across files and directories
 - Systematic renaming

CLAN applications - 4

- Software evolution (cntd)
 - Comparison between different versions of the same system
 - Subsequent versions
 - Reference versions

CLAN applications - 5

- Plagiarism Detection
 - Comparison of sets of syntactic blocks
 - Spectral analysis of similarity
 - Increasing thresholds
 - Spectral shape parameters are computed
 - Projects are ranked by similarity spectrum
 - The most similar projects are considered as candidates for plagiarism

CLAN applications - 6

- Refactoring
 - Redesign
 - Pattern based re-design
 - Automatic source code synthesis
 - Make the reuse explicit
 - Libraries identification

CLAN applications - 7

- Development Process
 - Problem mining
 - Issues
 - Bugs may be propagated by copies
 - Bug fixes may not be propagated in copies
 - Approach
 - If a modification is required in a code fragment, check also clones for modification opportunities (useful for bug fixes)

CLAN applications - 8

- Development Process (cntd)
 - Preventive control
 - Issue
 - Control the clone propagation
 - Approach
 - When a code fragment is delivered, check for existing similar fragments to prevent clones from being added to a system (if appropriate)

CLAN current research

- Definition of clones
- Type III (similar) and simple type IV (semantic) clones
 - Detection algorithms
 - Performance and scalability
 - Classification
 - Taxonomies
 - Statistics
 - Frequent patterns

CLAN current research - 2

- Properties of clone detection
 - Scalability
 - Performance
 - Time
 - Memory
 - Asymptotic
 - Practical
 - Incremental aspects
 - Robustness (tolerance to less frequent variations of representation patterns)

CLAN current research - 3

- Properties of clone detection (cntd)
 - Stability (small variations imply small differences in output)
 - Precision
 - Discretization error under thresholds
 - Recall
 - Clone maximality issues under thresholds
 - Clone redundancy
 - Clone subsumption

CLAN current research - 4

- Architectural analysis
 - Similarity + AST projection
 - API similarity
 - UML similarity
 - API rationalization
 - Refactoring
 - Assisted re-factoring
- Software evolution
 - Inconsistent modifications of clones
 - Taxonomy of some identifiable bugs

CLAN current research - 5

- Inconsistent modifications of clones
 - Bug reporting
 - Consistency of replacements
 - Identifiers
 - Identifier vs. constant
 - Identifier vs. identifier
 - Numeric constants
 - String constants
 - User messages
 - Error messages

CLAN current research - 6

- Inconsistent modifications of clones (cntd)
 - Patterns of inconsistent changes
 - Statistics on relevance

CLAN current research - 7

- Plagiarism detection
 - Spectral clone analysis
 - Relation with clone normalization problem
 - Web services
- Parallel approaches
 - Graphical Processing Unit (GPU)
- Distributed approaches

CLAN current research - 8

- Distribution of CLAN
 - Licensing parsers
 - Many sources
 - Licensing software (university and students issues)

CLAN current limitations

- Discretization error in the multi-dimensional space of fragments
 - Thresholds and neighboring classes
- False positives exist (noise)
 - Noise increases with increasing thresholds

CLAN current limitations - 2

- Limits of metric based approaches
 - Difference between metrics equivalence and representation distances
 - Similarity of metrics doesn't necessarily imply representation similarity of clone
 - The converse hold (somehow): similarity of representation implies similarity of metrics

CLAN further research

- Increasing the precision
 - Increasing the number of highly discriminating dimensions
 - High data dimensionality
 - Complexity issues
 - Neighboring clusters
 - Dimensions compression
- Increasing the recall rate
 - Increasing the thresholds
 - Precision issues

Conclusions

- A perspective on the WCRE 2000 paper has been presented
- Advancement during the last 10 years have been briefly outlined
- Current status of CLAN has been described
- Some open problems and research issues have been discussed

Acknowledgements

- The authors wish to thank Bell Canada for the original funding of the software quality project that included clone detection and re-factoring
- IBM Canada and CSER have supported related research projects
- Recent and current research has been funded by the Natural Sciences and Engineering Research Council of Canada under the Discovery Grants Program

Further Contacts

Ettore Merlo
Ecole Polytechnique de Montreal
tel: (514) 340-4711 ext. 5758
fax: (514) 340-3240
ettore.merlo@polymtl.ca