# Discoverable Digital Clone Repository for Improved Knowledge Transfer and Productivity

Moses L. Gadebe and George E. M. Ditsa

***Abstract*—Code clone is a result of copy and paste of source codes to solve similar problems. Currently most automated clone detection techniques produce indexed statistical reports showing locations of clones. This paper propose a Discoverable Digital Clone Repository to enhance Clone Wrapper Detection Technique to detect, extract and transform clone source code metadata into Clone Family Tree Ontology model stored in Fedora Repository. This is aimed at improving reuse, shareability and reliability of potential clones by software developers and maintenance programmers. The preliminary results show that the Discoverable Digital Clone Repository can store valid Clone Family Tree Ontology which is accessible to developers via Simple HTTP SPARQL queries.**

***Index Terms*—Clone family tree ontology, clone wrapper detection technique, discoverable, shareability.**

## I. Introduction

Software developers repeatedly copy and paste source codes to solve similar problems within organizations. Copy and paste development is intended to meet software project deadlines instigated by marketing teams who demand just-in-time solutions. This type of software development is lucrative to developers but sore to maintenance programmers, because they need to go through all duplicate codes to make required changes. Currently automated clone detection tools proposed indicate clone locations as indexed statistical reports that lack design patterns, reusability and shareability features [1], [2]. Thus potential clones lose their knowledge and consequently become meaningless to software developers: as most developers understand design patterns rather than index statistical reports.

Design patterns are high level conceptual models of software application design. Currently Object Oriented Design pattern is the dominant design pattern, which follows Abstraction process popularized by Object Management Group (OMG) [3]. The Abstraction process taxonomy (inheritance, composition and associations) of software application is currently formalized in OMG UML diagrams [3] [4]. Standard UML diagrams do not support variables to define design patterns as metadata that could be catalogued and published as a sharable, reusable and discoverable metadata. Resource Description Framework (RDF) and Ontology Web Language (OWL) provides cataloguing and publication of design patterns as high level conceptual model of software components that can be gathered and shared among software developers and maintenance programmers [4], [5].

The absence of discoverable, reusable and sharable structural clone reports diminishes knowledge transfer between software developers and maintenance programmers [3]-[5]. Hence this paper proposed Discoverable Digital Clone Repository (DDCR) that implements Clone Wrapper Detection Technique as discussed in [6] to detect and store structural clones in Fedora Repository as discoverable, sharable and reusable Clone Family Tree Ontology to support program maintenance and knowledge transfer between software developers and maintenance programmers.

The remainder of this paper is organized as follows. Section II presents the Background of the study and Section III the Implementation and the System Architecture. The benefits of DDCR are provided in Section IV and section V presents the Validation of DDCR. Lastly the Contribution, Conclusion and Acknowledgements are presented in Section VI, VII and VIII respectively.

## II. Background

Most of clone detection techniques detect clones and produce indexed statistical reports as textual file reports; indicating clone locations, line numbers of cloned source codes and clone pairs or clone classes (see Fig. 1) [2], [7].

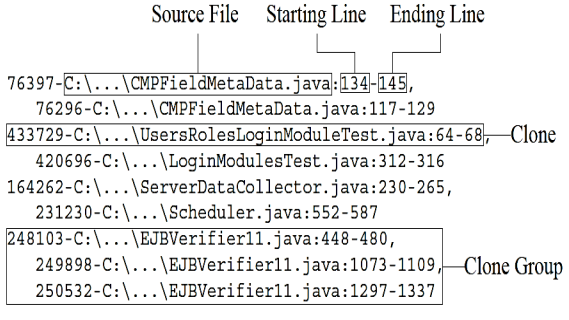



Fig. 1. Statistical indexed text clone report [7].

From the current observations of code clone detection research, there is little work done in detecting structural clones, which have potentials to be stored and reused through inheritance, composition and associations [2]. The issue of documentation in respect of source code's components shareability, interoperability and reuse capabilities is entirely not solved. Most software projects were developed using primitive use of copy and paste which neglects conceptual modeling and documentation of previous application source codes that can be catalogued and published [5]. Lack of conceptual model and repository prevents other developers to

Manuscript is received June 18, 2013; revised September 23, 2013. This work is supported by Tshwane University of Technology in Department of Computer Science to improve reuse of software components. The research title is Implementation of Discoverable Digital Clone Repository for Knowledge Transfer and Improved Productivity.

The authors are with Tshwane University of Technology, Pretoria, 0001, South Africa (e-mail: GadebeML@tut.ac.za, DitsaGE@tut.ac.za).

reuse existing source codes following proper design pattern methodology. Recently, clone visualization techniques have been developed to visualize clones metadata in a form of clones' genealogy. The genealogy depicts version history of clones' evolution to keep track of changes on clone pairs over a given period [8], [9].

An approach for software maintenance using Sematic Web techniques was proposed by [10]. The approach represents software components and information metadata in Resource Description Language (RDF) graph, to enable language-neutral relationship navigation and to facilitate ease navigation and comprehension of software components to improve maintenance. Hyland-wood *et al*., [10] developed an automated system to automatically generate ontology represented in RDF graphs from existing software component sources. This technique generates RDF graph of all replicated software components without filtering most commonly used software component that can be reused and shared by both software developers and maintenance programmers. Thus maintenance programmers are overwhelmed by the existence of clones that are not properly documented. Kleinvaloo *et al*., [11] proposed a much improved technique called Source code ECOsystem Linked Data (SECOLD) framework to provide source code facts that are usable by both human and machines for browsing and querying.

The above technique uses Linked Data to support interoperability and sharing of open datasets that allow on the fly inter-linking and using basic layers of Semantic Web and HTTP protocol. SECOLD provides Uniform Resource Locator (URL) generation schema that supports ontological representation for the interlinking of data extracted from source code ecosystems. The SECOLD lacks collaboration of which is the core for knowledge transfer between software maintenance programmers and software developers [11]. Software developers and maintenance programmers can use shared open metadata without knowing the originator of the software components, thus create the chances of software copyright violations and software plagiarism, as no-one can confirm the origin of software component.

## III. Implementation and System the Architecture

The Discoverable Digital Clone Repository (DDCR) comprises the Clone wrapper Detection Technique (CWDT) [6], Fedora Clone Repository, Development Server and two users - who are maintenance programmer and software developer as depicted in Fig. 2. Both users interact with DDCR system using the web browser and Owl protégé viewer as shown in Fig. 3. The maintenance programmer firstly detects commonly used structural clones by reading Java project sources from the development server. Then the structural clones are transformed into Clone Family Tree Ontology and stored into MySQL RDF Triple Store.

To detect commonly used structural clones from the development server, the maintenance programmer uses CWDT detection tool in the following orderly steps:

Firstly the Pre-processing step of the clone detection tool reads Java project source code files from the development server. Then converts each Java project source code files method's statements into equivalent Rectangular Matrix of multi-dimensional array of zeroes and ones, and then stores it into a data structure called Class Vector as shown in the pre-processing column in a Table in Appendix 1.

Secondly in the Transformation step, each Class Vector data structure of Rectangular Matrix is aggregated into a reduced Class Column Vector of matrices based on system of linear equations. Each method Rectangular Matrix $A$ is multiplied by its transposed method Rectangular Matrix $A^{-1}$ to calculate a method hash code and stored in the Class Column Vector (CCV) data structure as shown in the Transformation column in a Table in Appendix 1.

Thirdly, the Matching process compares CCVs of aggregated method hash codes based on intersection matching relation according to proposition $z_n = [x_n, y_n]$ as indicated on the intersection matching truth as Table I below. Two class column vectors are structural clones if they evaluate to be Perfect set or Subset or Proper sets and not Disjoint set based on the given threshold. Both their original source code are stored in a temporary storage for further processing.

In fourth process, all stored structural clones metadata are extracted and interwoven into a hierarchy called Clone Family Tree Ontology (CFTO), which is based on Abstraction process principles Inheritance (IS-A), Composition (HAS-A) and Associations (USES-A). This is transformed into OWL Lite as an aggregated hierarchical logical file called Digital Object Model (DOM). DOM is a group of related entities identifiable by a persistent ID shown in Fig. 4.

In the fifth and last process, the interwoven Digital Object Model is stored into a CFTO RDF triple store in Fedora Clone Repository shown in Fig. 5. The CWDT connects to the Fedora Repository Server via the Access API to store the CFTO on RDF Triple store. The CFTO RDF triple store is implemented in MySQL database as shown in Fig. 5. All generated CFTO are stored in the RDF triple database based on their persistent Uniform Resource Identifier (URI) as depicted in Fig. 4 and 5. All users query the CFTO using this URI to access the online structural clone metadata. The CFTO is stored in the RDF triple store as DOMs as shown in Fig. 4, whereas the persistent ID is the URI and the data streams are all interwoven structural clone's metadata and their relationship as part digital object model items as in [12].

TABLE I: Intersection Matching Truth Table

| The proposition $z_n = [x_n, y_n]$ is true if CCV1 and CCV2 contains: | Intersection Relation |
|---|---|
| All elements $\{x_1, x_2, x_3 \ldots\ldots, x_n\}$ in CCV1 are contained as elements $\{y_1, y_2, y_3, \ldots, y_n\}$ in CCV2. | *Perfect set CCV1=CCV2.* |
| All elements $\{x_1, x_2, x_3 \ldots\ldots, x_n\}$ in CCV1 are included as part of all elements $\{y_1, y_2, y_3, \ldots, y_n\}$ in CCV2. | *Subset* $CCV1 \subseteq CCV2$. |
| CCV1contains the same set of elements $\{x_1, x_2, x_3 \ldots\ldots, x_n\}$ in as in $\{y_1, y_2, y_3, \ldots, y_n\}$ of CCV2, but CCV2 contains at least one element $y_n$ not in CCV1. | *Proper subset* $CCV1 \subset CCV2$. |
| All Elements $\{x_1, x_{2,x_1}, x_{3x_{1,}} \ldots\ldots, x_n\}$ in CCV1 are not elements in $\{y_1, y_2, y_3, \ldots, y_n\}$ of CCV2. | *Disjoint set.* |

The maintenance programmer can retrieve and view the generated CFTO using the OWL Protégé viewer or the Web browser by sending a simple HTTP SPARQL query to the RDF triple store. The maintenance programmer also can

initiate a collaboration session with the software developer to pose questions about the digital objects model of CFTO.

The software developer responds back to any questions posed by the maintenance programmer online using a web browser. Moreover the software developer can reuse existing software components to solve similar problems. Both the software developer and maintenance programmer accesses the stored Clone Family Tree Ontology (CFTO) by sending a simple HTTP SPARQL query via the Access API of the Fedora repository server.

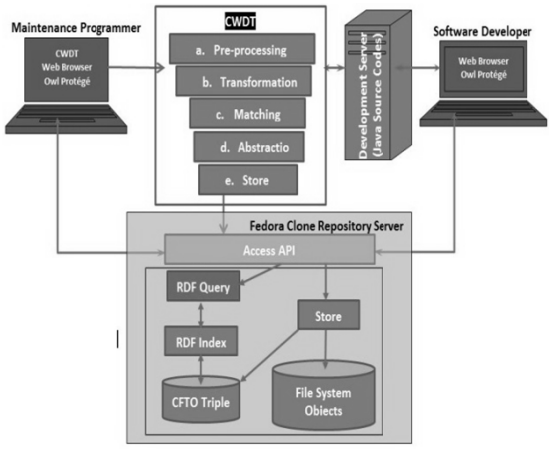



Fig. 2. DDCR system architecture.

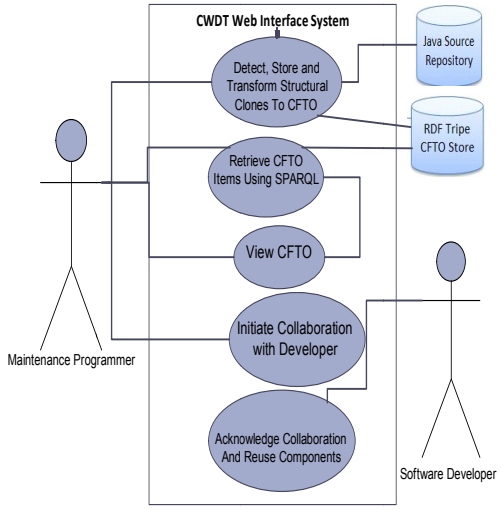



Fig. 3. DDCR use case diagram.

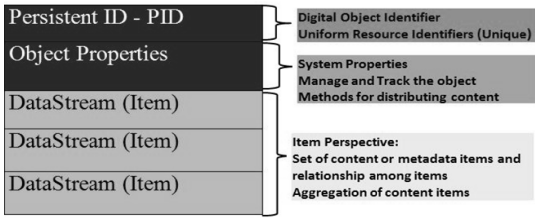



Fig. 4. Fedora digital object model [12].

| FTO_ID | FTCO |
|---|---|
| 3 | <?xml version="1.0" encoding="UTF-8"?> <rdf:RDF... |
| 4 | <?xml version="1.0" encoding="UTF-8"?> <rdf:RDF... |
| 5 | <rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl... |

Fig. 5. CFTO triple store MySQL database table.

## IV. Benefits of the DDCR

Most in-house software applications that perform similar responsibilities are usually replicated within the organization network to support overlapping business units. These software applications are tightly created to meet a specific business unit. Maintaining this replicated applications can take a long time especially when the maintenance team are novice programmers. Knowledge transfer between software developers and maintenance programmers is compromised due to unavailability of online comprehensive reusable and sharable conceptual metadata model that is publishable.

The DDCR brings the following benefits: 1) detection of reusable structural source codes; 2) online publishable, reusable and sharable conceptual models (i.e. CFTO metadata); 3) reliability, simple to understand conceptual model; and 4) the online collaboration between software developers and maintenance programmers. These will subsequently improve information sharing, knowledge transfer and cohesion among team members. Thus bridging the gap between time to develop and time to market and therefore improve productivity of software developers and software maintainers.

## V. Validation of DDCR

The DDCR was tested using different Java Development Kits versions. The CWDT implemented by the DDCR managed to detect a number of true positive structural clone pairs. The results indicate a balanced precision of 0.64% to 0.81% with a relative recall of 0.80% to 0.96% as shown in Fig. 6.

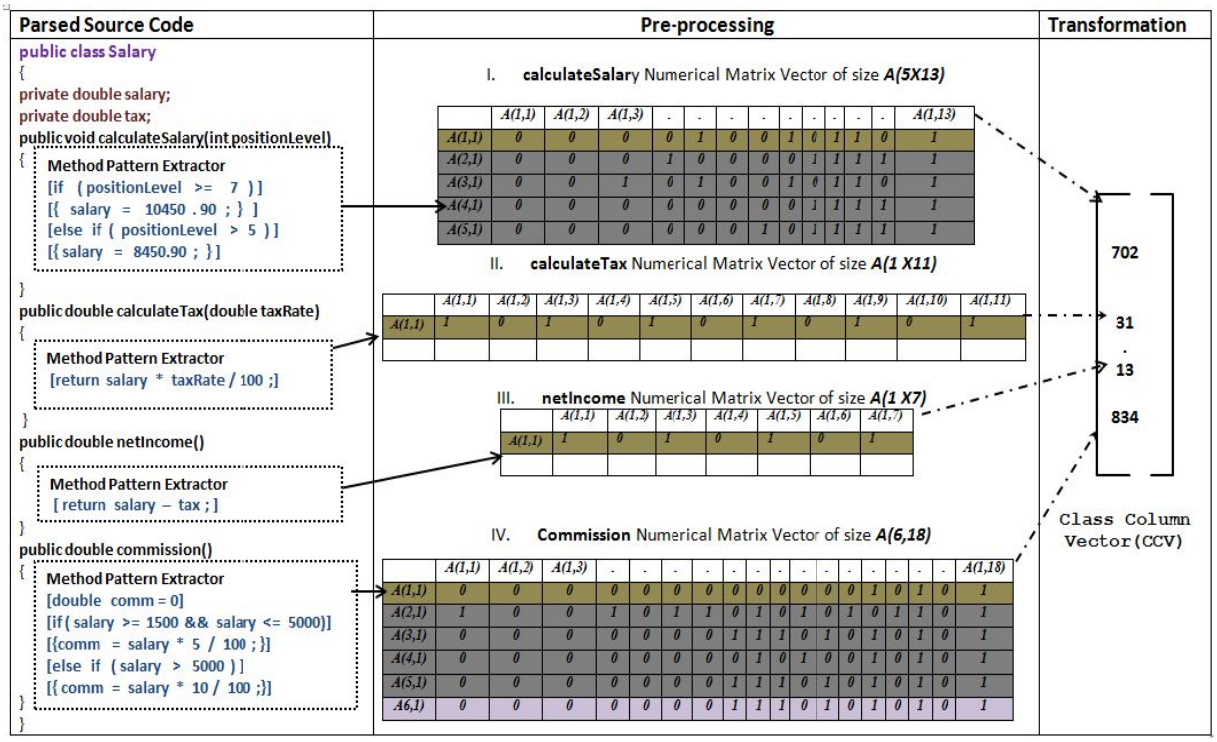
The true positive structural clones are transformed to Clone Family Tree Ontology (CFTO) represented as Ontology Web Language Lite and stored as Fedora Digital Object Model. The resultant CFTO must be valid, correct and consistent hierarchical metadata that can open and be processed in OWL Protégé viewer. OWL Protégé in Fig. 7 portrays hierarchical structural clone metadata descending from class "Thing".

## VI. Contribution

The main goal of this research is to provide a universal persistent identification (ID) Digital Object Model for each reusable and sharable structural source codes as online conceptual model called Clone Family Tree Ontology (CFTO). The Objective is to provide an accessible CFTO in a collaborated manner for both software developers and maintenance programmers. Commonly used structural source codes within the organization are mined and transformed using the CWDT. Each structural clone sets are interwoven according to their hierarchy level using Abstraction principles (inheritance, composition and associations) identified by unique Uniform Resource Identifier (URI) (i.e. persistent ID) as discussed in [10]-[12] research. Then the transformed CFTO is stored in a discoverable Fedora repository. The vision is to provide an online conceptual interlinked metadata to support software developers and maintenance programmers within an organization to collaborate in order to improve software component reuse,

software components and reliability to eliminate software plagiarism.

## APPENDIX



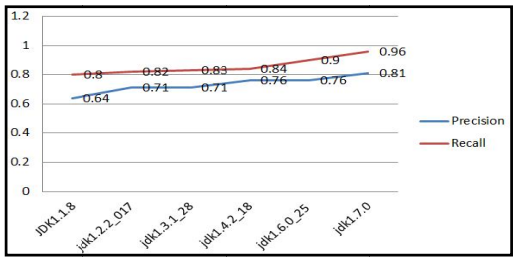




Fig. 6. CFTO triple store MySQL database table.

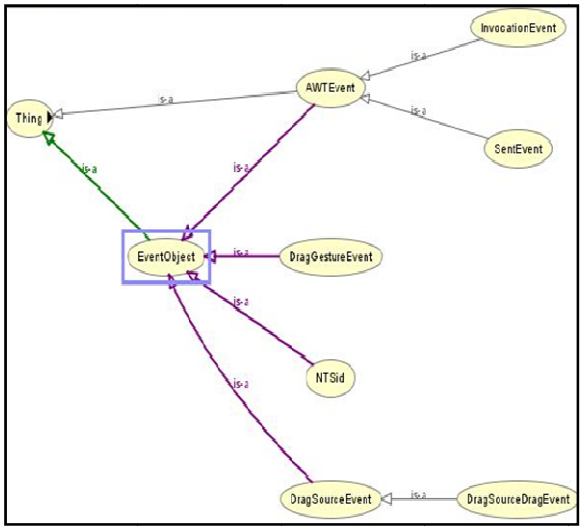



Fig. 7. Protégé clone family tree ontology viewer.

The CFTO would be accessible over the Internet via HTTP Protocol using SPARQL query to retrieve interlinked metadata based on their unique URI and to view the interlinked digital object's super classes, sub classes, composites classes and associated classes. The conceptual interlinked metadata model also act as online discoverable structural clone documentation of software components.

## VII. CONCLUSION

This paper presented the Discoverable Digital Clone Library (DDCR) system to help maintenance programmers and software developers to collaborate with the aim of transferring knowledge between them and to eliminate software plagiarism and to improve software reuse and ease of maintenance. The paper provided system architecture implemented in Fedora repository to store a Clone Family Tree Ontology (CFTO) as digital object models that define super classes, sub classes, composite classes and associated classes as online hierarchical metadata.

The CWDT is robust and accurate in detecting structural reusable clone transformed into valid CFTO. In the future we intend to complete and implement the DDCR in an organization to perform usability testing. It will also be interesting to find out the acceptance level and usage level of DDCR after implementation. Finally we intend to extend the DDCR to allow the software agents to access and reuse CFTO to support the business units within the organization as a central processing point.

## ACKNOWLEDGMENT

We thank Tshwane University of Technology the Department of Computer Science for the financial and other logistical support for success of this project.

## REFERENCES

[1] S. Thummalapenta, L. Cerulo, L. Aversano, and M. Di Penta, "An empirical study on the maintenance of source code clones,"

*International Journal of Emprical Software Engineering*, vol. 15, no. 1, pp. 1-34, 2010.
[2] K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools," *Journal Science of Computer Programming*, vol. 74, no. 7, pp. 470-495, 2009.
[3] A. H. Eden, "A theory of object oriented design," *Journal of Information Systems Frontiers*, vol. 4, no. 4, pp. 379-391, 2002.
[4] G. Ganapathy and S. Sarayaraj, "To generate the ontology from Java Source code: Owl creation," *International Journal of advanced Science and Applications*, vol. 2, no. 2, pp.111-116, 2011.
[5] L. Pavlic, M. Hericko, V. Podgorelec, and I. Rozman, "Improving design pattern adoption with ontology-based repository," *Journal of Informatics,* vol. 33, pp. 189-197, 2008.
[6] M. L. Gadebe and G. E. Ditsa, "Clone wrapper detection technique: Clones family tree ontology," in *Proc. 2nd International Conference on Computer and Software Modeling*, Cochin, 2012, pp. 20-31.
[7] R. A. Tairas, "Representation analysis and refactoring techniques to support code clone maintenance," Ph.D. dissertation, Dept. Computer Science, Birmingham Univ. Alabama, 2010.
[8] M. F. Zibran and C. K. Roy. "The Road to Software Clone Management: A Survey," A Technical Report, Dept. Computer Science, Saskatchewan. Univ., Canada. 2012.
[9] L. Barbour, F. Khomh, and Z. Ying, "Late propagation in software clones," in *Proc. 27th IEEE International Conference on Software Maintenance*, 2011, pp. 273-282.
[10] D. Hyland-Wood, D. Carrington and S. Kaplan, "A semantic web approach to software maintenance," presented at ACM EKAW, Poderbrady, Czech Republic. Oct 2-6, 2006.
[11] I. Keivanloo, C. Forbes, J. Rilling and P. Charland, "Towards sharing source code facts using linked data," in *Proc. ACM SUITE*, Waikiki Honolulu, 2011, pp. 25-28.
[12] A. Grigorov, A. Georgiev, and P. Anagnostou, "Building OWL ontology in fedora digital repository," in *Proc. Modelling and Control of Information Processes*, Sofia, 2010, pp. 314-2771.

**Moses Lesiba Gadebe** is a Lecturer at Tshwane University of Technology in the Department of computer science. Mr Gadebe holds a BTECH degree in information technology and currently lecturing technical programming subjects. He is a member of SAICSIT an ACM affiliate. He published a number of conference papers

**George Ditsa** is currently an associate professor at the Tshwane University technology in Pretoria, South Africa. He holds a B.Sc. (Hons) degree in computer science, an MBA (IS) and PhD (IS) degrees. Dr. Ditsa worked for many years as a programmer/analyst and project team leader in various organizations before joining the academia. Dr. Ditsa currently lectures and researches in Information Systems (IS) and related disciplines. Dr. Ditsa has won a number of research grants and awards. His current research interests include Strategic IS Management, IS Project Management, Cultural Issues in IS management, Knowledge management & Knowledge Management Systems, Mobile & Pervasive Computing, ICT for Development and Human Computer Interaction (HCI). Dr. Ditsa has supervised a number of postgraduate students in his research interest areas. Dr. is currently an Associated Editor of two journals and he is on the editorial review board of five journals. He also serves as an External Theses Examiner for some universities. Dr. Ditsa has published a number of scholarly articles including a book, book chapters, journal papers and many refereed conference papers.