2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering

Towards Implementation of an Integrated Clone Management Infrastructure

Minhaz F. Zibran
Department of Computer Science, University of New Orleans, LA, USA
Email: zibran@cs.uno.edu

***Abstract*—Despite more than two decades of research on software clones, a dependable integrated clone management system is still not available and active clone management has also remained far from industrial adoption. This paper proposes a design of an integrated clone management infrastructure and points to the challenges involved in realizing the design to a dependable implementation.**

I. Introduction

Code clones are recognized as a serious kind of code smells. In addition to exposing the malign effects of clones, the research community also acknowledges the benefits from code cloning. Thus, clone management has drawn interests in the community in relatively recent years than in farther past. Towards clone management, initiatives from leading software giants such as Microsoft, Google and other industrial bodies in addition to the clone research community have been encouraging, indeed.

Support for clone management, although with very limited functionalities, has been available in popular IDE's (Integrated Development Environments) such as Microsoft Visual Studio and Eclipse while some stand-alone research prototypes have also been developed, only a few of which intend to offer clone management support beyond detection [6], [9]. A few software quality assessment tools [1], [2] have also been available, which include features to support clone analysis. Nevertheless, clone management has rarely been adopted in software development process on daily or periodic basis [3], [9]. We believe, the following two issues play a decisive role behind this.

Lack of Interoperability and Integration: Although many techniques and many prototypes have been developed for solving important problems in different areas of clone management, those separate tools and techniques are rarely interoperable. There is no feature-rich dependable tool that can be easily integrated with the development process.

Organizational Priorities: In industrial settings, a typical software development project is planned for completion in a very tight schedule, because, from the business perspective, revenue from such a project typically remains inversely proportional to the development time (in terms of man-hour) spent in completing the project.

Indeed the benefits from code cloning (e.g., code reuse for speed-up development process) are immediate while the majority of probable negative impacts of clones (e.g., program faults, increased maintenance effort) are uncertain and deferred until (and if) those are encountered in future. Anyways, the developers strive in writing and maintaining test suits expected to reveal software defects no matter whether the faults appeare from cloning or not. Thus, a software development team do not want to spend significant portion of current time and efforts for proactively managing clones whose harmful effects are uncertain and not immediate. They would rather save that effort, and are prepared to spend that, if necessary, to fix any issues that may become visible in future.

In this paper, we propose an infrastructure for clone management, which also addresses the aforementioned two influencing factors.

II. Proposed Infrastructure

The proposed architecture for a clone management system is summarized in Figure 1, in which, the distinct functional components of the infrastructure are presented in rounded rectangles. To resolve the issues with interoperability described above, the architecture emphasizes on complete integration of diverse clone management features and techniques under a common infrastructure. The solid bidirectional arrows in Figure 1 represent integration while the dotted unidirectional arrows denote the directions of flow of data and information. The complete integration is necessary to minimize the need for human efforts in clone management, which consequently, will also help in addressing the issues with organizational priorities described before.

The organizational priorities, in practice, are unlikely to change in favour of clone management, because software development projects will continue to be dictated by financial and business constraints. We, therefore, argue that clone management must adapt to it by incorporating features developed on top of clone-based techniques for solving additional problems in software engineering, beyond merely clone management. Thus, the proposed infrastructure also includes functional components enriched with customizable and parameterized features for additional applications. For example, the clone analysis part (Figure 1) is meant to support product line engineering by the extraction of reusable components detected from clones. The infrastructure also aims to incorporate clone-based information into software quality reports, refining developer evaluations, aid in version/branch merging, origin analysis, and the like.

Effective clone management must support both proactive and reactive means for dealing with the clones. Hence, the

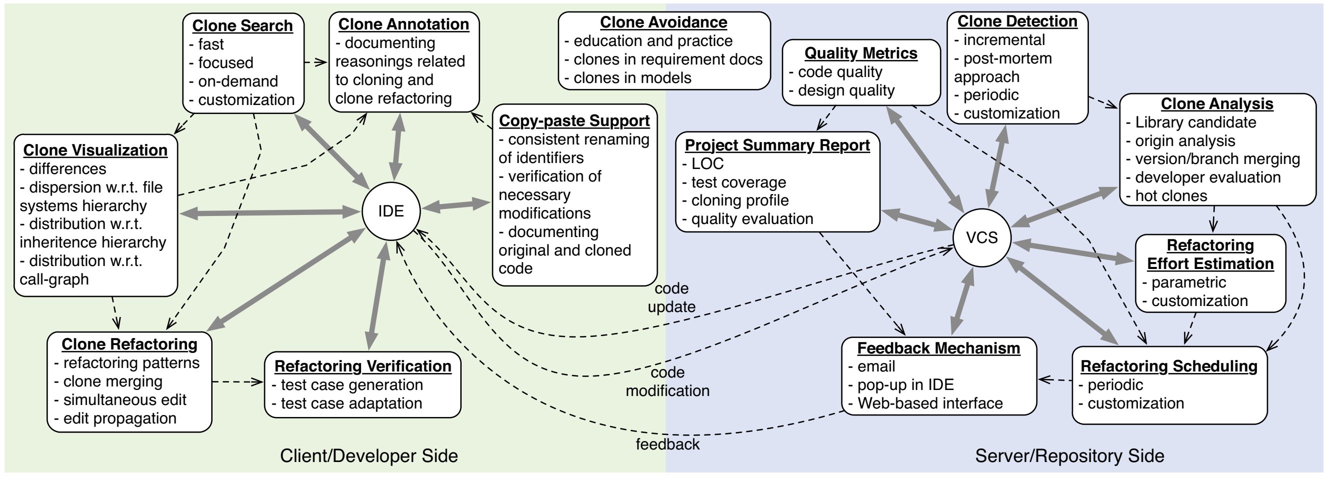

Fig. 1. Proposed Infrastructure for Integrated Clone Management

infrastructure must be integrated with the daily software development process by plugging in the clone management techniques to a popular IDE [7], [8] and also to a popular Version Control System (VCS), as proposed in Figure 1. Proactive clone management will help in dealing with clones right after those are created, while reactive clone management will allow periodic detection and management of long-lived clones, which are left intact for some documented legitimate reasons. An integrated recommendation system must keep propagating feedback and recommendations from the VCS to the developers in IDE without disrupting their natural workflow.

III. Challenges

Till date, we, as a clone research community, have already developed many innovative and powerful techniques for separately carrying out clone management activities such as the detection, documentation, tracking, visualization, analysis, merging and refactoring of clones. The capabilities and limitations of those tools are discussed elsewhere [5], [9]. Instead of "reinventing the wheels", it would be wise to reuse existing techniques and tools in the implementation of the proposed infrastructure. However, we see significant challenges in realizing the proposed infrastructure by integrating existing techniques and prototype tools.

First of all, the definitions of clones vary depending on use cases. With respect to any of the clone management activities mentioned above, there is no single tool that can operate in accordance with all different task-specific definitions of clones. The diversity in the definitions of clones and tuning parameters used in different tools constraint their interoperability. In addition, the variations in the formats and contents of the inputs and outputs of the individual tools add to difficulty in information sharing between them. While the formulation of RCF (Reach Clone Format) [4] meant for documenting clone information can help in information exchange among individual tools, we do not see wide adoption of this particular data format in existing tools.

An approach for integrating the independent tools can include extensive glue code for transformation of data between one format to another possibly involving substantial disk operations, which may hinder the scalability and performance of the proposed infrastructure and make it inappropriate for practical use. Moreover, such an approach will be fragile and susceptible to break when the individual tools change independently.

A better approach can use collective efforts from the clone research community. We may choose a common operating platform, on which the tool authors can develop their tools. The popular Eclipse platform having inherent plug-in architecture can be such a suitable platform. Tool authors can develop their tools as plug-ins to Eclipse and publish their tools with exposed extension points. Individual tools then interact with each other through the available extension points.

IV. Conclusion

This paper proposes an integrated infrastructure for effective clone management. We have discussed the existing deficiencies and challenges against the implementation and adoption of the proposed system. Towards realization of such a versatile infrastructure, this paper makes a call for collective efforts from the software clone research community.

References

[1] ConQAT, https://www.cqse.eu/en/products/conqat/overview/, last access: Dec 2015.
[2] PMD: Source code analyzer, https://pmd.github.io, last access: Dec 2015.
[3] I. Baxter, M. Conradt, J. Cordy, and R. Koschke. Software clone management towards industrial application (dagstuhl seminar 12071). *Dagstuhl Report*, 2(2):21–57, 2012.
[4] J. Harder and N. Göde. Efficiently handling clone data: RCF and cyclone. In *IWSC*, pages 81–82, 2011.
[5] C. Roy and J. Cordy. Scenario-based comparison of clone detection techniques. In *ICPC*, pages 153–162, 2008.
[6] C. Roy, M. Zibran, and R. Koschke. The vision of software clone management: Past, present, and future. In *CSMR-18/WCRE-21 Software Evolution Week (SEW'14)*, pages 18–33, 2014.
[7] M. Zibran and C. Roy. Towards flexible code clone detection, management, and refactoring in IDE. In *IWSC*, pages 75–76, 2011.
[8] M. Zibran and C. Roy. IDE-based real-time focused search for near-miss clones. In *ACM-SAC (SE Track)*, pages 1235–1242, 2012.
[9] M. Zibran and C. Roy. The road to software clone management. Technical Report 2012-03, Department of Computer Science, University of Saskatchewan, Canada, http://www.cs.usask.ca/documents/technical-reports/2012/TR-2012-03.pdf, 2012.