Continuous Maintenance

Candy Pang, Abram Hindle
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
{candy.pang, abram.hindle}@ualberta.ca

***Abstract*—There are many "continuous" practices in software engineering, for example continuous integration (CI), continuous delivery (CD), continuous release (CR), and DevOps. However, the maintenance aspect of continuity is rarely mentioned in publication or education. The continuous practices and applications depend on many repositories and artifacts, such as databases, servers, virtual machines, storage, data, meta-data, various logs, and reports. *Continuous maintenance* (CM) seeks to maintain these repositories and artifacts properly and consistently through automation, summarization, compaction, archival, and removal. For example, retaining builds and test results created by CI consumes storage. An automated CM process can remove the irrelevant artifacts and compact the relevant artifacts to reduce storage usage. Proper CM is essential for applications' long term sustainability. There are two sides of CM: pre-production and post-production. During the pre-production phase, CM maintains the health of the development environments and the relevant processes. Then during the post-production phase, CM maintains the health of the applications. This paper defines CM for developers. CM complements and completes continuous practices.**

Keywords—Software engineering; Continuous Integration; Continuous Delivery; Continuous Release; DevOps; Continuous Maintenance

I. Introduction

Continuous maintenance (CM) is an extension of *continuous integration* (CI). The CI concept was first practiced and described as "do everything in parallel, with frequent synchronizations" in the 1998 book Microsoft Secrets [1]. Microsoft employs multiple teams to implement different features in parallel, and relies on CI to keep software development synchronized. Companies have been practicing CI for at least two decades. Normally integration happens from end to end, and in practice it is not divided into *continuous integration* (CI) [2], *continuous release* (CR), and *continuous delivery* (CD) [3]. Therefore, the authors combine CR, CD, and other continuous practices into CI for discussion in this paper.

CM focuses on the processes that maintain applications' long term sustainability during development and after production, while CI focuses on enhancing development. For example, during development, many CI processes rely on version control repositories, and these repositories need maintenance. The CI build process creates artifacts such as executables, meta-data, and various logs. These artifacts need maintenance. The CI test process creates artifacts such as virtual machines, containers, database instances, test data, test results, and various kinds of logs. These artifacts also need maintenance. In production, applications create artifacts such as data and execution logs that need maintenance. CM maintains these artifacts through automation, summarization, compaction, archival, and removal. For example, an automated maintenance process can analyze and summarize execution logs, then compact and archive the logs for a retention period, before removing the logs.

Imagine an application that creates temporary files during a process. The temporary files should be retained for a period of time, then be eliminated under normal circumstances. In this case, the programmers implement the source code that creates the temporary files. The operational analysts allocate the storage where the application writes the files. The programmers specify the life span of the temporary files and any exceptional conditions. An automated CM process will monitor the temporary files, compress the files to use less storage during the retention period, and eliminate the files under normal circumstances. The CM process will also watch for exceptional conditions, and carry out predefined handling procedures. Furthermore, the CM process can analyze and summarize the temporary files to provide additional knowledge. Likewise, CM can maintain bug reports, crash reports, and other non-source code artifacts created by applications.

As IT practitioners, the authors have been observing the end-to-end CI practices in enterprise for more than 10 years in the fields of health care, law enforcement, Internet service, customer management, e-Commerce, telecommunication, finance, investment, and environment. In the IT industry, programmers have embraced many of the CI practices. However, the authors observed that the maintenance processes are often overlooked. Many of the maintenance processes are retrofitted for development or in production after encountering sustainability issues. For example, no CI publication has introduced any continuous process to clean up repositories. When companies deploy CI, they should deploy CM at the same time to maintain CI. By describing various CM processes, the authors hope to raise discussion and awareness about application maintenance. In this paper, the authors describe *continuous maintenance* (CM) as the continuous processes that maintain development (pre-production) artifacts, and operations (post-production) artifacts through automation, summarization, compaction, archival, and removal.

The remainder of this paper is structured as follows. Section II reviews existing continuous practices. Section III lists CM processes. Section IV concludes the paper and sketches future work.

II. Related Work

In the existing literature, the maintenance processes for applications' long term sustainability are rarely mentioned. Fitzgerald *et al.* [4] tried to list all continuity related activities from business strategy to development, which "include continuous planning, continuous integration, continuous deployment, continuous delivery, continuous verification, continuous testing, continuous compliance, continuous security, continuous use, continuous trust, continuous run-time monitoring, continuous improvement (both process and product) … [and] continuous innovation". Shamieh [5] added continuous engineering. However, the maintenance aspect of continuity is missing. Some documented continuous practices among practitioners are described below.

A. Development Management

During development, continuous practices have been well defined. In the book "Continuous Integration", Duvall *et al.* [2] have thoroughly described the following CI practices and their corresponding tools:

- Continuous code integration / continuous build
- Continuous testing
- Continuous inspection
- Continuous feedback
- Continuous delivery
- Continuous database integration

Continuous code integration is the core of the continuous development process. Continuous code integration frequently builds applications from repositories, so that integration problems can be identified early. There are many automated build tools, such as Ant [6], Travis CI [7], and Jenkins [8].

For large projects, compiling a complete build can be very time consuming. A hierarchical build process can be employed to reduce the building time of individual components.

Continuous build can identify integration problems, and **continuous testing** can identify implementation errors. After each build, predefined unit tests and regression tests will be run to identify implementation errors. Regression tests can also be organized into hierarchy to reduce testing time and lower resource consumption. There are many automated testing tools, such as JUnit [9] and other "xUnit" products.

Testing can increase programmers' confidence in their source code. However, having integrated source code that passes testing ensures neither code integrity, nor bug free status. Therefore, **continuous inspection** evaluates the quality of the source code after each change, and looks for potential issues according to previous knowledge.

Continuous inspection uses a variety of tools to examine different source code aspects. For example, Checkstyle [10] makes sure source code adheres to code style standard, EMMA [11] measures code coverage, and FindBugs [12] analyzes code to look for potential bugs. In addition, there are tools that identify code duplication, measure unit tests coverage, estimate energy consumption, and more. Overall continuous inspection safeguards source code quality.

Continuous code integration, continuous testing and continuous inspection can all identify problems. But they rely on **continuous feedback** to alert the responsible personnel.

Continuous delivery orchestrates deployment in different environments, such as individual machines, local servers, or the cloud. Humble *et al.* [3] in their book "Continuous Delivery" describe the continuous delivery practice in details. However, they did not consider the maintenance of artifacts described in this paper.

Last but not least, **continuous database integration** rebuilds databases and test data according to changes in repositories [2]. Continuous database integration is necessary to support continuous code integration and continuous testing.

Many continuous practices have been identified in software engineering. However, they have not considered any maintenance processes. DZone has published a Continuous Delivery Maturity Checklist [13], which does not take into account artifact maintenance either. CI focuses on launching applications to production. CM cares about the health of the applications after they are in production. CM also cares about the sustainability of the CI processes.

Programmers learn how to practice CI through these publications, which do not talk about maintenance, but maintenance is essential to sustain applications. The authors believe that programmers should learn about CM as they learn about CI.

B. Operations Management

In large companies, applications are implemented by the development team, but run by the operations team [14]. The operations team follows pre-defined IT Service Management (ITSM) activities to maintain quality IT services, separately from the development team or the development process. A few popular ITSM frameworks are listed below:

- *Information Technology Infrastructure Library* (ITIL) [15]
- *Control Objectives for Information and Related Technologies* (COBIT) [16]
- International Service Management Standard for IT Service Management (ISO/IEC 20000) [17]

These frameworks identify processes, roles and responsibilities for IT practitioners to sustain production environments and increase uptime. These frameworks promote circular processes to continuously improve IT services. However, these frameworks do not specify application maintenance processes, or promote coordination between programmers and operational analysts.

It is generally known among IT practitioners that "there was, is, or has been a problem within corporate IT departments between Development and Operations" [18]. DevOps [14] was introduced "as a culture, movement or practice that emphasizes the collaboration and communication of both software programmers and other IT professionals while automating the process of software delivery and infrastructure changes" [19]. DevOps focuses on automating infrastructure, but the literature is not explicit about maintaining applications in operation. CM

covers maintenance processes necessary to sustain applications in operation.

CM can also benefit DevOps by purging containers and virtual machines that are not needed anymore. For example, a system may use Docker containers [20] for testing and deployment. In some cases, the containers are made, but left behind. An automated CM process can clean up dangling containers to maintain resource availability.

In the rest of the paper, we will describe the CM processes that enhance development management and operations management.

III. CONTINUOUS MAINTENANCE (CM)

As described in the previous sections, CI creates many artifacts, such as executables, meta-data and logs from code integration; results, data and logs from testing; warnings and reports from inspection. CI also depends on many repositories and artifacts, such as databases, storage and virtual machines. In addition, applications running in production also create and depend on many artifacts. *Continuous maintenance* (CM) consists of the processes that maintain these repositories and artifacts properly and consistently through automation, summarization, compaction, archival and removal.

CM aims to improve productivity, sustainability, and efficiency through automation. CM seeks to improve consistence and continuity within an IT department. The goal of this paper is to make companies aware of the CM processes, which complement and complete the continuous practices. Not all processes are applicable to all applications. CM processes can be classified into two phases: pre-production and post-production.

A. Pre-Production CM

For *Commercial Off-the-Shelf* (COTS) applications, the pre-production phase includes all the IT processes before the applications are released to the public. For Web applications, the pre-production phase refers to the IT processes before the applications are deployed to production. The pre-production phase includes all CI practices described in Section II.A.

Above and beyond, the pre-production phase also includes processes such as feature verification, bug fixing, user acceptance testing, and staging. CM is responsible for maintaining the long term wellness of repositories and IT artifacts that support the pre-production phase. Some of the pre-production CM processes are listed below.

1) Repository Archival: Repositories are the core of many CI processes, such as automated build and testing. There are many version control repository software, such as *Concurrent Versions System* (CVS) [21] and Git [22], Perforce [23], and *Team Foundation Server* (TFS) [24]. Each repository has its advantages and disadvantages, and organizes data in its own way. For example, Git accumulates data continuously. Without maintenance, the size of Git repositories will keep growing. In large companies, ever growing repositories may cause performance and management issues. CM monitors the content, size, and performance of the repositories, and keeps check of the relevant content. Older file versions that are no longer relevant to the product development can be archived. Nonetheless, these repositories often handle binary files poorly. For example, Git requires special tweaks to manage large binary files [25]. CM can automate theses corresponding processes.

CM can help maintain application branching policy. Such a policy enable pruning of branches that are no longer needed. According to the policy, CM can archive the branches that are abandoned or no longer relevant to the product line. In some cases, it may be worthwhile to apply some forms of data warehousing to dead branches, especially those with many binary materials and assets.

CM automatically archives content regularly. Archival does not simply removing artifacts forever. It may just be marking artifacts as historical and no longer relevant to the product development. CM is particularly important for companies using multiple types of repositories to support different technologies. CM will ensure that all types of repositories are maintained consistently.

2) Storage Cleanup: Besides repositories, many CI practices consume a lot of storage. For example, the continuous code integration process builds applications from repositories regularly and executes tests. For large applications, a complete build may take hours. When continuous code integration fails, the build's artifacts should be retained for investigation, while avoid filling up storage. Even if all tests pass, the IT department may retain the executables for additional assessment, such as performance and energy consumption. On top of the executables, the following details about each build should be retained:

- List of source code files used and their versions
- List of configuration files used and their versions
- List of binary files used and their versions
- Build and application execution log

If any tests fail, programmers need to know the exact conditions in which the tests fail. These additional test artifacts should be retained:

- Test logs and test results
- Databases and data used in the tests
- Virtual machines and containers

These artifacts are also retained from user acceptance test and staging, where written failure reports are provided to programmers after testing.

CM manages metadata and retains relevant artifacts. When needed, CM can help recreate any executables from the repository according to the metadata. Retaining build and test artifacts devour a lot of storage. CM can deduplicate, compress, and/or archive large files to use less storage. CM can also support the business process that resolves integration problems. Once a build has served its business purpose, CM can eliminate irrelevant artifacts.

3) Management Systems Automation: Continuous code integration identifies conflicts; continuous testing identifies bugs; continuous inspection identifies quality issues. Companies may use different management systems (e.g. bug tracking system, task tracker, quality management system) to

Fig. 1. Temporary Data Cleanup and Logs Archival[1]

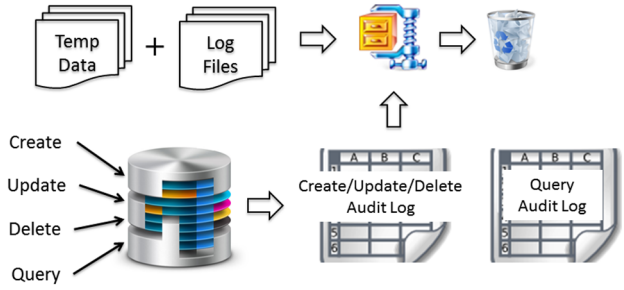
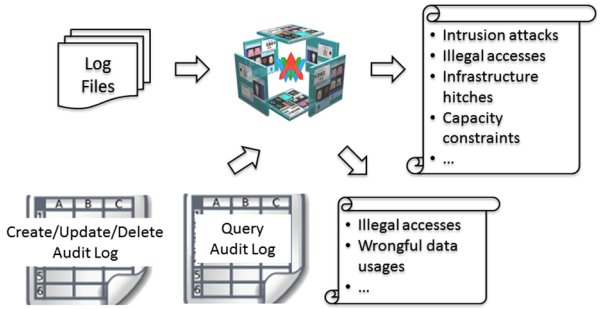


Fig. 2. Exception or Violation Analysis[1]



manage these affairs. Since the CI processes are being invoked continuously without human's involvement, CM should filter out redundant information for project stakeholders.

CM can simplify and enhance the management processes. For example, when automated tests fail, CM can use bug-dedupers and bug-clusterers [26] to determine whether the errors have been reported in the bug tracking system. If not, CM will automatically initialize bug reports on behalf of the programmers. CM can also employ bug repair, crash-dedupers, and crash-clusterers [27] to propose solutions for the programmers. CM can also close duplicated bugs.

Moreover, CM can utilize search based software engineering [28] to generate additional unit tests, which may reveal errors. In the presence of errors, tool derived from search based software engineering can mutate and search possible programs for mutations that repair the errors. CM can bring these potential errors and solutions, not identified by continuous inspection, to the attention of the programmers.

In the pre-production phase, CM maintains a healthy development environment by managing artifacts used and generated during the CI processes. CM ensures all development artifacts are in good condition, so that programmers can concentrate on their source code.

B. Post-Production CM

For COTS applications, the post-production phase refers to the time after the applications are installed, configured, and have started serving users. For web applications, the post-production phase refers to the time after the applications are deployed to the production environments and are accessible by users.

Maintenance is necessary to keep applications sustainable. This section describes the automated CM processes that maintain applications in the post-production phase, involving both programmers and operational analysts.

1) Temporary Data Cleanup: Many applications create temporary data or files. The temporary data may be created intentionally, or left dangling because of abnormal termination. Regardless, programmers should know where temporary data may exist. Programmers need to work with operational analysts to identify potential temporary data. Then CM can automatically monitor, summarize, archive, and clean up temporary data, as shown in Fig. 1.

2) Logs Archival: Many applications keep execution logs about connection requests, service requests, executed processes, exceptions, and errors. Under normal situation, the logs will not be examined. When problems occur, the logs will be inspected. CM can analyze the logs and store the relevant statistic data, then compress and archive the logs for a period of time before eliminating them, as shown in Fig. 1.

Many applications also keep audit logs. Audit logs contain metadata about data *creation, retrieval, update, and deletion* (CRUD). CM can analyze and summarize the audit logs as benchmark. Audit logs can grow rapidly, especially the data retrieval audit logs. Therefore, CM should monitor the size of the audit logs, then compress and archive the logs as needed. Audit logs are very important for many businesses. CM should keep the archived audit logs easily accessible for inquiry or analysis.

3) Exception or Violation Analysis: Execution logs and audit logs contain valuable information. CM can analyze them before archiving.

Analyzing execution logs can identify potential problems such as intrusion attacks, illegal accesses, infrastructure hitches, and capacity constraints. CM can bring these potential problems to the responsible personnel. Defining and continuously enhancing the exception analyzing rules can prevent potential problems in the future, as shown in Fig. 2.

Analyzing audit logs can identify wrongful data usages. For example, there were incidents where health service providers had been illegally browsing patients' medical records [29] [30]. CM could have identified the wrongful behavior through analyzing the retrieval audit logs. Subject matter experts may be involved in defining the rules for audit logs analysis. The rules should be reviewed and enhanced continuously.

4) Data Elimination: In some businesses, there are legal obligations to eliminate obsolete data. For example, the law enforcement systems may need to remove juveniles' records once the offenders become adults. E-commercial systems may need to purge cardholder data after retention period [31]. Subject matter experts and legal advisors may involve in defining the data elimination rules for CM to remove obsolete data for legal compliance.

5) Data Warehousing and Analytics: Proper data warehousing and analytic improve business value, and speed time to insight [32]. Sometimes production data is too

[1] Clip art © Microsoft Office Media library.

complicated and sensitive for data warehousing. CM can employ the *Extract, Transform and Load* (ETL) tools provided by the *database management systems* (DBMSs) to periodically extract analyzable data from production databases, then filter, convert, mask, and replace sensitive data before loading data into data warehouses for analysis. Subject matter experts and data analysts may need to review the data conversion rules to make sure sensitive information cannot be mined from the data warehouses.

In the post-production phase, CM maintains applications' sustainability by managing operational data and logs. So that operational analysts can focus on handling exceptions, and will be more aware about potential problems.

IV. Conclusion

Many artifacts, such as executables, meta-data, test data, logs, reports, virtual machines, and containers, are used and generated by the continuous integration processes during development. In addition, data and logs are generated by applications running in production. Managing repositories and these artifacts is onerous and not adequately addressed in the continuous literature. Thus ***continuous maintenance*** is proposed to maintain these artifacts and extract values out of them through analysis, summarization, compaction, archival, and removal. This paper identifies specific automatable maintenance processes needed before production in development, and after production in operation that are benefited by continuous maintenance.

Continuous maintenance attempts to automate many manual maintenance processes to reduce maintenance costs. This allows companies who deploy continuous maintenance to allocate a bigger portion of their IT budget to innovation [33], hence increase return on investment.

Since this is a relatively new topic, there are many research questions to be answered:

- How many companies have implemented what CM processes?
- How do companies deploy the CM processes?
- Who is accountable for the CM processes?
- Have companies implemented or purchased CM tools?
- Where do programmers and operational analysts see the needs of cooperation for maintenance?
- What are the challenges in their cooperation?
- What other CM processes exist?
- Can CM policy, guideline or checklist be defined?

References

[1] M. A. Cusumano and R. W. Selby, Microsoft Secrets: How the world's most powerful software company creates technology, shapes markets, and manages people, New York: Simon and Schuster, 1998.

[2] P. M. Duvall, S. Matyas and A. Glover, Continuous Integration – Improving Software Quality and Reducing Risk, Addison-Wesley, 2007.

[3] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Pearson Education, 2010.

[4] B. Fitzgerald and S. Klaas-Jan, "Continuous Software Engineering and Beyond: Trends and Challenges," in Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, Hyderabad, India, 2014.

[5] C. Shamieh, Continuous Engineering for Dummies, IBM Limited Edition, John Wiley & Sons, Inc., 2014.

[6] The Apache Software Foundation, "Another Neat Tool (Ant)," [Online]. Available: http://ant.apache.org.

[7] Travis CI, GmbH, "Travis CI," [Online]. Available: https://travis-ci.org/.

[8] Jenkins, "Jenkins," [Online]. Available: https://jenkins-ci.org/.

[9] JUnit, "JUnit," [Online]. Available: http://junit.org/.

[10] Checkstyle, "Checkstyle," [Online]. Available: http://checkstyle.sourceforge.net/.

[11] Vlad Roubtsov, "EMMA," [Online]. Available: http://emma.sourceforge.net.

[12] University of Maryland, "FindBugs," [Online]. Available: http://findbugs.sourceforge.net/.

[13] DZone Research, "Continuous Delivery Maturity Checklist," The DZone Guide to Continuous Delivery 2015 Edition, p. 32, 2015.

[14] M. Loukides, What is DevOps, O'Reilly, 2012.

[15] AXELOS Ltd., "Information Technology Infrastructure Library (ITIL)," [Online]. Available: https://www.axelos.com/best-practice-solutions/itil.

[16] ISACA, "Control Objectives for Information and Related Technology (COBIT)," [Online]. Available: http://www.isaca.org/cobit/pages/default.aspx.

[17] ISO, "ISO/IEC 20000-1:2011 Information technology-Service management-Part 1: Service management system requirements," [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=51986.

[18] R. Lowery, "What You Need to Know about DevOps," 19 04 2016. [Online]. Available: http://www.solutionsiq.com/what-you-need-to-know-about-devops/.

[19] M. Schmidt, "DevOps and Continuous Delivery: Not the Same," Mediaops, LLC., 08 04 2016. [Online]. Available: http://devops.com/2016/04/08/devops-and-continuous-delivery-not-same/?mc_cid=30427ff599&mc_eid=411e0719f2.

[20] Docker, "Docker," [Online]. Available: https://www.docker.com/.

[21] Derek Robert Price & Ximbiot, "Concurrent Versions Systems (CVS)," [Online]. Available: http://www.nongnu.org/cvs/.

[22] L. Torvalds, "Git," [Online]. Available: https://git-scm.com/.

[23] Perforce, "Perforce," [Online]. Available: https://www.perforce.com/.

[24] Microsoft, "Team Foundation Server (TFS)," [Online]. Available: https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx.

[25] J. Narebski, Mastering Git, Packt Publishing, 2016.

[26] A. Alipour, A. Hindle and S. Eleni, "A Contextual Approach Towards More Accurate Duplicate Bug Report Detection," in Proceedings of the 10th Working Conference on Mining Software Repositories, San Francisco, CA, USA, 2013.

[27] J. C. Campbell, E. A. Santos and A. Hindle, "The Unreasonable Effectiveness of Traditional Information Retrieval in Crash Report Deduplication," in Proceedings of the 13th International Conference on Mining Software Repositories, Austin, Texas, 2016.

[28] J. C. Campbell and A. Hindle, "The charming code that error messages are talking about," PeerJ PrePrints, vol. 3, p. e1388, 2015.

[29] R. Southwick, "Hospital staffer fired for snooping into patient, employee health files," Edmonton Journal, p. A8, 08 10 2014.

[30] T. Hopper, "Mount Sinai staff access Ford's records - Two employees disciplined for taking a peek," National Post, p. 9, 17 10 2014.

[31] Security Standard Council, "Payment Card Industry (PCI) Payment Application Data Security Standard v1.1," 2008.

[32] C. Howson, J. Parenteau, R. L. Sallam, T. W. Oestreich, J. Tapadinhas and K. Schlegel, "Critical Capabilities for Business Intelligence and Analytics Platforms," Gartner, 2016.

[33] M. Zetlin, "How to balance maintenance and IT innovation," 21 10 2013. [Online]. Available: http://www.computerworld.com/article/2486278/it-management/how-to-balance-maintenance-and-it-innovation.html.