Improving Product Quality and User Satisfaction through Early Customer Feedback

Matias Cabral, Domingo Gonzalez, Dan Hirsch, Cesar Martinez, Andres More, and Victor Rosales

Intel Software Argentina - Argentina Software Development Center (ASDC)
{matias.a.cabral, domingo.c.gonzalez, dan.hirsch, cesar.martinez, andres.more, victor.h.rosales}@intel.com

Abstract. New users of cluster computing find a high entry barrier due to the daunting complexity of deploying and managing a cluster. The Intel® Cluster Ready program aims to lower such entry barrier by setting standards for the cluster integration process and providing tools to simplify the deployment process. The Intel Cluster Checker tool, a key component of the Intel Cluster Ready program, is an automated and flexible tool that validates the cluster settings against the Intel Cluster Ready specification, and checks the general wellness of the cluster. This experience report shows how the product engineering team of Cluster Checker faced the challenge of having a small team and be able to provide a cutting-edge high-quality product in less time, maximizing resource usage, and increasing and securing early validation with key stakeholders.

1 Introduction

The *Intel® Cluster Ready Program* [1] aims to reduce the time-to-market and minimize the risk involved in selecting a collection of hardware and software components for High Performance Computing environments, providing a thoroughly tested solution stack and ensuring the interoperability of its components out-of-the-box.

Key to this program, the Intel Cluster Checker [1] is a powerful and flexible tool for helping customers to integrate clusters by validating the compliance of a new cluster solution with the Intel Cluster Ready Specification [1]. The product includes 31K lines of source code distributed in more than 120 independent test modules, executing over +10 officially supported operating systems. Although Independent Product Validation is done using four 4-node cluster environments over the latest Intel and third-party technologies, the actual testing goes beyond that, covering cluster scale-outs up to 256 server nodes; heterogeneous server platform combinations; and pre-release hardware and software component compatibility (both, for Intel and third parties) to support the time to market launch of Intel's new technology. Furthermore, as the Intel Cluster Checker is integrated by the program partners (Original Equipment Manufacturers, Platform Integrators and channel members) in their cluster manufacturing processes, there is a

need for the tool engineering team to efficiently and timely address partner requirements that are critical for their business continuity and the launch of their own Intel Cluster Ready certified products.

This experience report shows how the Intel Cluster Checker engineering team, located at Intel's Argentina Software Development Center, has implemented reliable, efficient and high quality processes and infrastructures, coping with the complexity of the validation space and the critical time to market business requirements, through the integration of Agile development methodologies with a *CMMi* process initiative.

2 Background

2.1 Intel Cluster Ready Program

In recent years, cluster computing has emerged as a scientific tool to obtain additional computational power in the commercial (manufacturing and services), health, finance, and educational areas [2]. Furthermore, as Figure 1 shows, the cluster market is growing faster than the non-cluster servers market [5]. It is estimated that the High Performance Computing (HPC) server market revenue will grow past $12 billion in 2013, from $9 billion in 2009, with clusters making up for more than 70% of this amount [6].

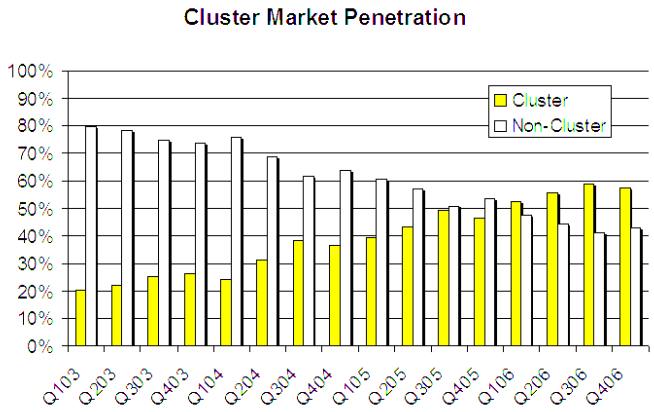


Fig. 1. Cluster vs. Non-Cluster Server Market Penetration

The comparatively low cost and scalability of clusters is pushing more companies to enter in the cluster's arena. However, the capability of each cluster involves an exchange of hardware costs for software costs [7]. A complex combination of software is required to configure and maintain multiple distributed machines that make a single clustered system. Therefore, new users are forced

to do extensive research to establish hardware and software requirements, build the cluster, install the required software, and then tune the components to obtain a healthy system with a competitive performance. Given the complexity of building a cluster, different users often develop ad-hoc procedures for automated cluster deployment and management. The associated lack of standards have kept the cluster ownership costs high as custom procedures should be developed from scratch on virtually every installation. To ameliorate this situation, some of the "best practices" techniques for cluster deployment were integrated in easy-to-use toolkits called provisioning systems [7, 8]. However, the solution offered by provisioning systems only considers certain software stacks, hopefully over a small set of hardware platforms.

In order to overcome the complexity of this environment, Intel developed the *Intel Cluster Ready Program* [1] to simplify the design, building, and deployment of clusters.

With more than 100 partners around the world, the Intel® Cluster Ready Program is a collaborative effort between Intel, Original Equipment Manufacturers (OEMs), Platform Integrators (PIs), channel members, and Independent Software Vendors (ISVs), that signup as Intel Cluster Ready partners, in order to establish an architecture and specification to be used as a common basis for performing clusters. Cluster solutions that comply with the Intel Cluster Ready Specification [1] are certified by Certification Authorities of the Intel Cluster Ready program. Also applications produced by ISVs can be registered as compliant with the Intel Cluster Ready specification.

The main idea behind the Intel Cluster Ready certification process it that applications written to run on one certified cluster can run on any certified cluster with a high confidence. Conversely, a certified cluster will be able to run any registered application without requiring major efforts.

To complete the certification of a cluster solution, the Intel Cluster Ready partner (OEMs, PIs, channels) may follow the steps indicated by a Cluster Reference Implementation (a.k.a. recipe) [1]. These recipes are the product of a cluster engineering process that will allow the Intel Cluster Ready partner to manufacture compliant clusters. The next step in the process is to use the Intel Cluster Checker Tool to automatically verify the cluster health and compliance with the Intel Cluster Ready Specification. The tool used for the automatic verification is the Intel Cluster Checker, which is introduced in the next section.

2.2 Intel Cluster Checker Tool

The Intel Cluster Checker [1] is a software tool that helps the verification of cluster compliance with respect to the Intel Cluster Ready Specification. It can verify the cluster at the node and cluster-wide levels, ensuring that cluster nodes are uniformly and optimally configured. The tool is customizable and extensible, letting users develop their own tests by specifying commands to be executed and their expected results.

In a nutshell, it consists of an engine that loops over a set of test modules (see Figure 2). Prior to entering the loop, the engine must figure out which test

modules to run, which nodes to check, and setup the infrastructure to process the test modules.

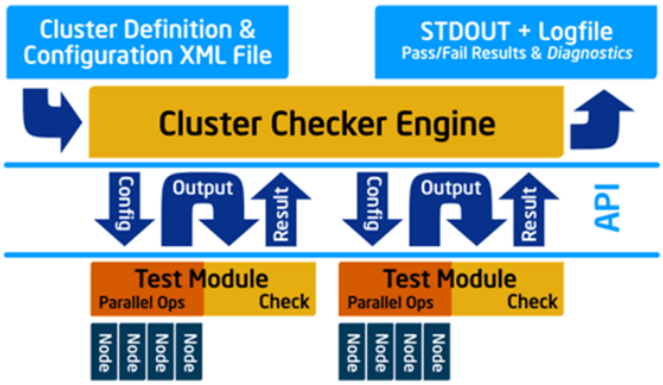


Fig. 2. Test Modules Execution flow.

The modules themselves do not execute on the cluster nodes; a module runs one or more commands on a node. So, there is no needed for any specific support at each node of the cluster. A module has two functional steps: it first collects information and then determines if the information is correct. Correct may mean matching a certain value, uniformity compared to other values, or something else. Thus, modules must have a gather step where it collects information and a test step where it determines if that information is correct.

As shown in Figure 3, there are four different classes of modules to handle special cases of this two-step gather/test process, but all classes respect the aforementioned structure.

Class	Coverage	Description	Example
Unit	Individual node	Checks 'correctness' of a node property compared to a value.	Does `/tmp` have the correct permissions?
Vector	Cluster-wide	Checks the 'uniformity' of a node property across the cluster.	Is the same version of `gcc` installed on each node?
Span	Cluster-wide	Checks a 'cluster-wide' property compared to a value.	Does a simple MPI program run successfully on the cluster?
Matrix	Pair-wise	Checks a node-to-node ('pair-wise' or 'all-to-all') property for every possible node-to-node combination.	The network latency and bandwidth for all node pairs meet a threshold.

Fig. 3. Classes of Modules

There are two types of checking: the wellness checking of a cluster and the compliance mode with respect to the Intel Cluster Ready Specification.

Wellness checking verifies the functional and non-functional characteristics of a cluster, like performance or disk space. It can be used for on demand validation

of the cluster. This feature of Intel Cluster Checker helps the maintenance of the cluster health providing diagnostic data to identify the issue.

Compliance checking is used during the Cluster Certification process of the Intel Cluster Ready Program. The goal of applying this checking is to determine if the cluster configuration follows the Intel Cluster Ready Specification.

3 Implementing combined CMMi and Scrum methodologies

As we have mentioned in the introduction, although Independent Product Validation of the Intel Cluster Checker is done using the four top 4-node cluster environments, the actual testing goes beyond that, covering cluster scale-outs up to 256 server nodes; heterogeneous server platform combinations; and pre-released hardware and software component compatibility (both, Intel and third party) to support the time to market launch of Intel's new technology. Also, as the Intel Cluster Checker is integrated by the program partners in their cluster manufacturing processes, there is a need for the development engineering team to efficiently and timely address partner requirements that are critical for their business continuity and the launch of their own Intel Cluster Ready certified products.

The product is made up of 31K lines of source code distributed in more than 120 independent test modules, executing over +10 officially supported operating systems.

Currently, the product is being developed at the *Argentina Software Development Center (ASDC)* in Córdoba. The engineering team consists of 2 developers, 1 tester and 1 independent product validation engineer.

The main challenge faced by the reduced engineering team has not been only to provide a cutting-edge high-quality product in less time, while maximizing resource usage, but also to increase and secure early validation with key stakeholders. Moreover, last minute requirements requested by Intel Cluster Ready partners should be included based on their feedback.

To achieve these goals, the engineering team has been driving process and infrastructure improvements over the last year and a half, involving three releases of the tool.

3.1 Infrastructure

On the infrastructure side, the project has included an aggressive effort for building an automated development and testing infrastructure (Figure 4), avoiding recurrent effort on repetitive manual tests, while supporting the required software and hardware coverage against a small time to market window.

The engineering infrastructure includes 2 servers (based on different platforms) for development, 4 clusters dedicated to testing (15 nodes on 4 different server platforms), and 1 server with 18 virtualized GNU/Linux distributions for automatic operating system testing coverage. For product validation the team

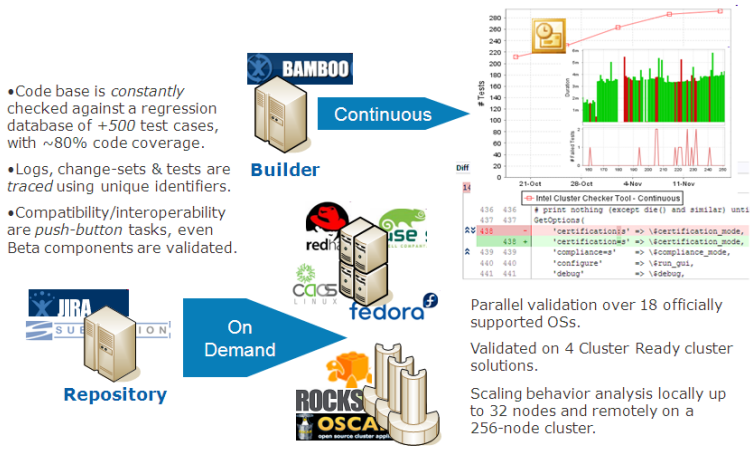


Fig. 4. Development, Testing and Validation Infrastructure

uses, on demand, a laboratory of more than 300 servers that allows up to 32-nodes cluster scale-outs over 4 different multi-core and multi-processor Intel platforms; and remote access for scale-out validation up to 256 nodes. This infrastructure allows to cover a wide range of the key hardware and software stacks for High Performance Computing.

The development and testing teams rely on several tightly coupled tools and a flawless working environment for its day-to-day work. All issues are tracked in a dedicated tool with the information needed for their planning and solution. Every issue is a branch in the versioning system keeping the trunk as stable as possible. Once the development team finishes the fix and adds at least one unit test, then a script is executed on that branch that triggers a regression framework with over 500 test cases and a static code analyzer. The script makes sure that the branch is up-to-date and it generates a detailed report that is sent to the testing team. The testing team will not accept the fix unless the quality of the code is greater or equal than before, at least one unit test was added to the framework, a code peer review was performed, and the regression framework has passed 100% of the tests. This testing cycle is performed over the issue branch and once it is finished, a testing report is sent back to the developer, who then merges the branch back into the trunk. Once the merge into the truck is done, it automatically triggers a building script that runs the same regression framework over the trunk. Then, all resulting data is gathered and reports are sent back to the team.

Two more build plans complement the development cycle. A nightly build plan gets executed which runs the regression framework over the trunk and builds the product leaving it ready for distribution. Then, the regression framework is ran over the binary package generated in the previous step and, if the build succeeds, a third build plan is executed exercising the binary package in parallel

over 18 virtual machines with all supported operating systems, ensuring binary compatibility.

3.2 Processes

Based on the infrastructure improvement and automation, the key component that allowed the team to achieve their goals has been the changes applied to the Product Development Process through the implementation of the Capability Maturity Model® Integration (CMMi) [3] - Level 2 and Level 3 Best Practices (as part of a certification strategy at the ASDC center) combined with the adoption of an agile development methodology [4], concretely the iterative incremental development process called SCRUM [9] (see Figure 5). This revolutionary change, combining the CMMi process improvement model and an Agile development methodology, was fully implemented for the last three releases (versions 1.3, 1.4 and 1.5) of the Cluster Checker Tool, delivering a working and validated product after each development sprint.

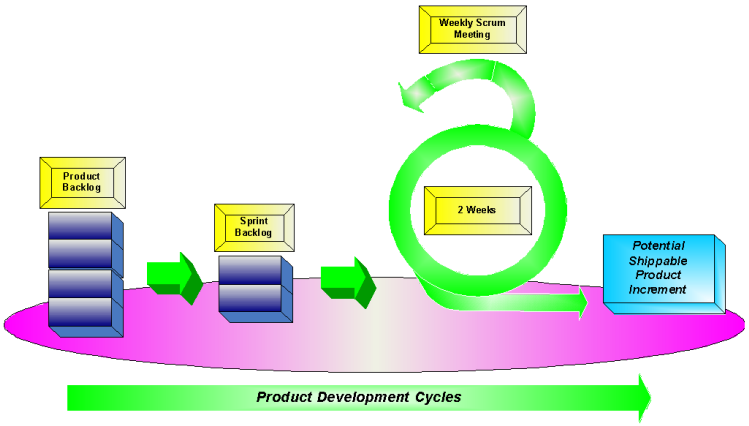


Fig. 5. Development Methodology

The different engineering areas of the team, that is, Software Development, Testing, Product Validation and Product Packaging, worked together to improve product quality through the implementation of an innovative process to increase automatic testing coverage while keeping track of all documents/source code revisions. All of these enhancements were successfully validated by internal and external audits performed by the *Process and Quality Services* area of ASDC and aligned with the CMMi process initiative in place.

The Product Development Process to build the product was completely migrated from a traditional Waterfall to a *SCRUM* agile software development process. Moreover, it is worth noticing that the release approval process followed

the standard Intel approval product lifecycle for a release. As a first step to integrate these processes, the Project Manager in charge was certified as *SCRUM MASTER* to be able to oversee all the procedure. The process started when the product version was approved at *Implementation Plan Approval* phase by the *Software Product Planning Committee* of the *Developer Product Division* of Intel (including the complete Product Requirements Backlog prioritized).

The agile process framework, defined by the software engineers, established a set of *Development Sprints* where the *Change Control Board* committee defined which features are going to be delivered at the end of each sprint. Once a sprint achieved all the pre-established quality targets, the product was distributed to key stakeholders with the purpose of getting early feedback on the features committed for that phase. Each Sprint cycle is executed every two and a half weeks in order to keep the key stakeholders warm and involved in the progress of the tool. Additionally, a closure meeting is held after each sprint by the *Change Control Board* committee to check status, implement any requirement changes and define next sprint planning. Due to the very aggressive schedule and changing requirements, the process requires a high automation level through all the development, testing and product validation phases. Therefore, the set of new test cases and regression test cases have been continuously integrated into an automatic Quality Assurance framework which is executed against each new change, validating the pass/fail rate criteria established at the beginning of the development phase. This setting allows the engineering team to have a constant and hard tracking on the product development behavior; maintaining an always working product state.

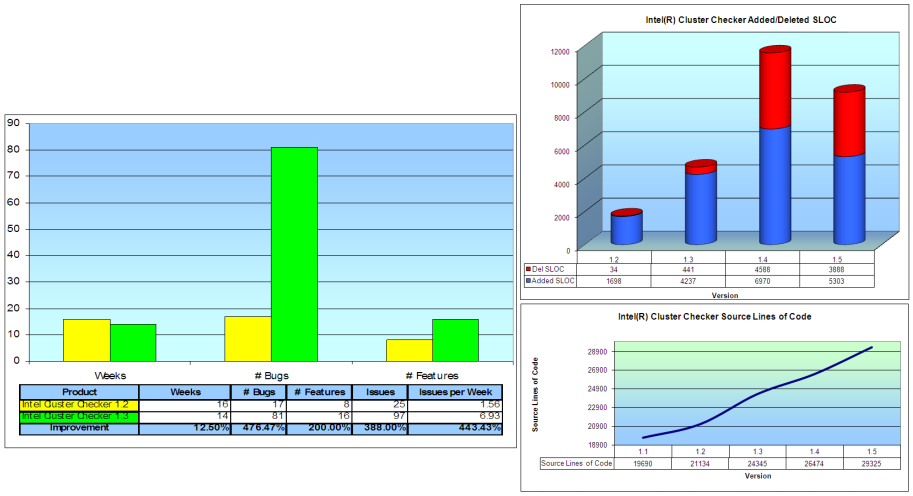
4 Results and Conclusions

Early customer feedback and testing automation allowed Intel Cluster Checker last three versions (versions 1.3, 1.4 and 1.5) to be released achieving sustained higher quality and in a time frame shorter than previous versions. As a result of the process started in Cluster Checker 1.3, the product had key stakeholders validating not only all the requested features but extra features additional to the originally planned ones (Figure 6).

In cold numbers, the process transition in Cluster Checker 1.3 showed an initial 12.5% effort reduction, a 400% improvement in backlog resolution, and 200% on new features included in the release cycle (Figure 6). Following a quality-oriented approach, the size of the automated test case framework was increased from 80 to 500 test cases which were enforced after each code change, resulting in a constant 80% global coverage in the current release version 1.5 (Figure 7), presented through a dynamic drill-down web report which identifies gaps and directs next steps (Figure 8). Furthermore, static source code analysis over both source code and documentation were integrated to guarantee that all known coding pitfalls are avoided and also documentation formatting is homogeneous.

As the final result, the product team was able to release, in less time than before, higher quality versions, validating with key stakeholders on the progress

and the implementation maturity for all the requested features and allowing even extra features outside the planned ones to be included. It is worth mentioning the integration of the agile methodology within the CMMi initiative at ASDC, resulting in Cluster Checker being one of the projects certified for both Level 2 and Level 3 action plans (details of this are beyond the scope of this paper). Also, very positive impact with excellent feedback was received from internal/external key stakeholders for the well-defined product development process. The practices and methodologies established in this process have been shared as *Best Known Methods* with other teams, where the added value of this experience has been the increased quality and customer satisfaction without impacting on available project resources.



Product	Weeks	# Bugs	# Features	Issues	Issues per Week
Intel Cluster Checker 1.2	16	17	8	25	1.56
Intel Cluster Checker 1.5	14	81	16	97	6.93
Improvement	12.50%	476.47%	200.00%	388.00%	443.43%

	1.2	1.3	1.4	1.5
Del SLOC	34	441	4588	3888
Added SLOC	1698	4237	6970	5303

	1.1	1.2	1.3	1.4	1.5
Source Lines of Code	19690	21134	24345	26474	29325

Fig. 6. Improvements achieved after process transition

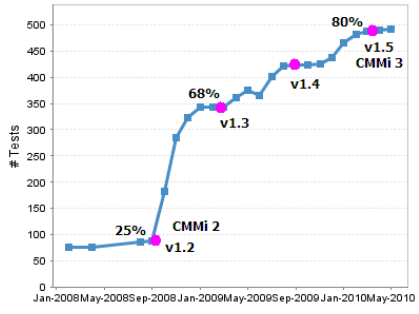


Fig. 7. Growth of test cases and global coverage

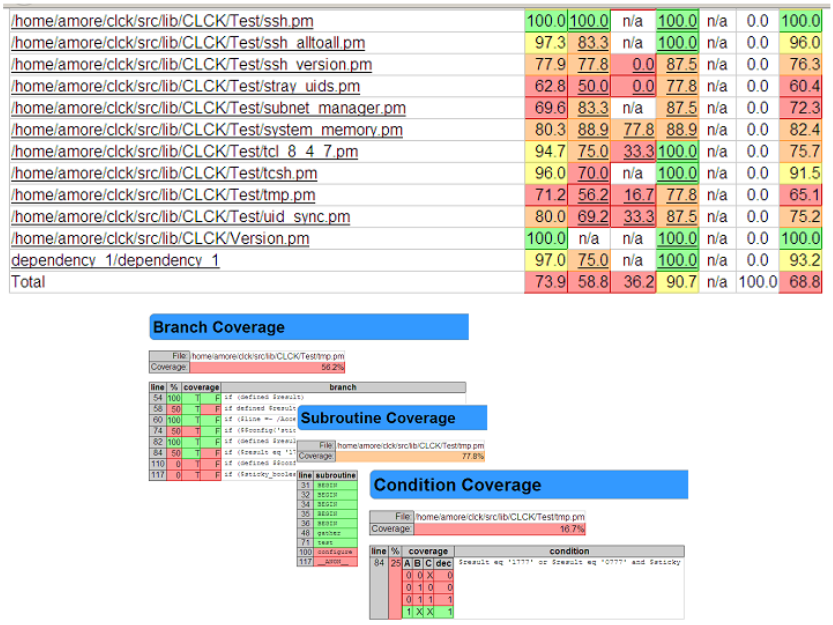


Fig. 8. Assisted coverage extraction and gaps identification

References

1. http://www.intel.com/go/cluster
2. http://www.top500.org
3. http://sas.sei.cmu.edu/pars/
4. Cohen, D., Lindvall, M., Costa, P.: An introduction to agile methods. In Advances in Computers. 62, 2–67 (2004)
5. Joseph, E.: HPC cluster market trends. Tech. rep., IDC Corp. (May 2007)
6. Joseph, E., Conway, S., Walsh, R., Wu, J., Lee, D.: Worldwide technical computing server 2008 top 10 predictions. Tech. rep., IDC Corp. (January 2008)
7. Mugler, J., Naughton, T., Scott, S., Barrett, B., Lumsdaine, A., Squyres, J., des Ligneris, B., Giraldeau, F., Leangsuksun, C.: OSCAR clusters. In: Linux Symposium (June 2003)
8. Papadopoulos, P., Katz, M., Bruno, G.: NPACI rocks: Tools and techniques for easily deploying manageable linux clusters. In Concurrence and Computation: Practice and Experience. 15(7,8), 707–725 (2003)
9. Schwaber, K.: Agile Project Management with Scrum. Microsoft Press (2004)