

aselsan

www.aselsan.com.tr

Radar and Electronic Warfare Systems Business Sector

ASELSAN is a Turkish Armed Forces Foundation company.



Software Quality Improvement Practices in Continuous Integration

İlgi KESKİN KAYNAK, Evren ÇILDEN, Selin AYDIN
Sep 18, 2019

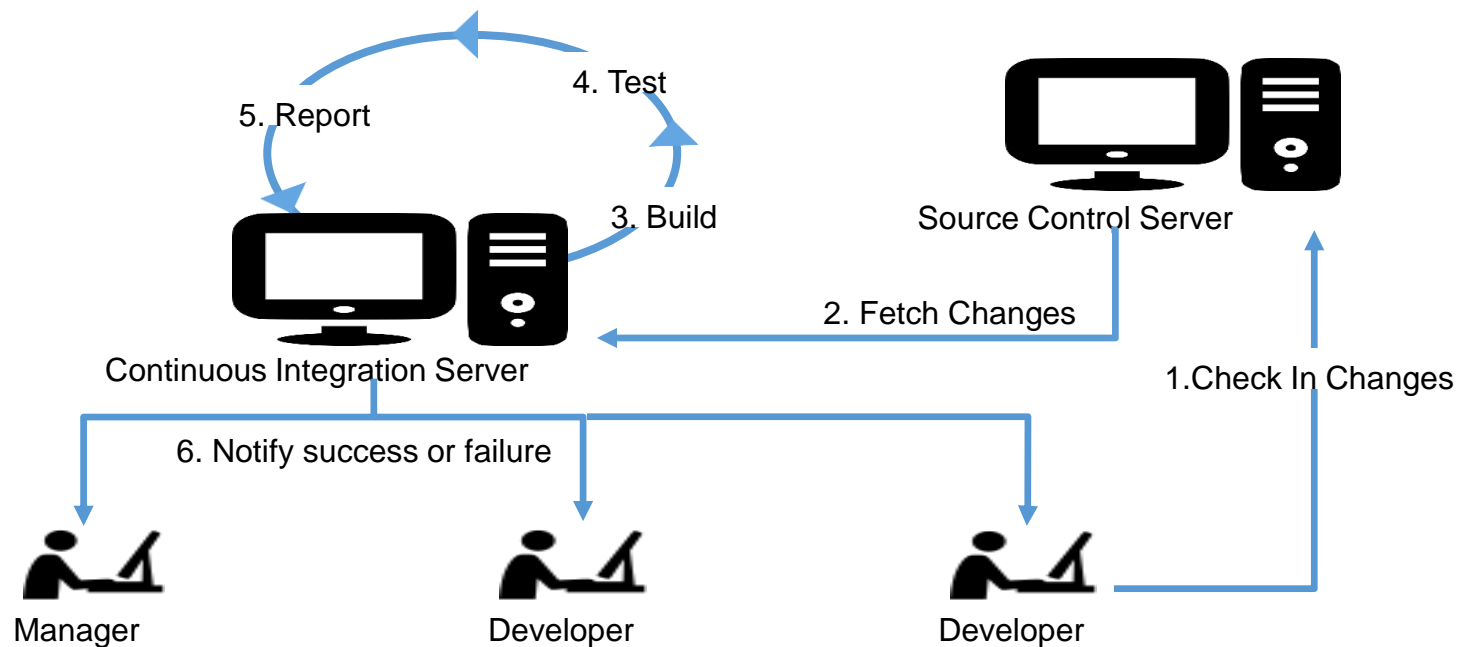
- ☐ Introduction
- ☐ Continuous Integration Setup
- ☐ Measurement Automation
- ☐ Software Metrics
- ☐ Defect Detection Effectiveness
- ☐ Defect Density
- ☐ Evaluation
- ☐ Conclusion
- ☐ Future Works
- ☐ References

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early. By integrating regularly, errors could be detected quickly, and located more easily.

The Automation Studies, covered in this study, have the following sections: Compilation, Unit Test, CSCI Test, Static Analysis, Measurement and Release Management. Process automation is needed to automatize the measurement process which has the following advantages:

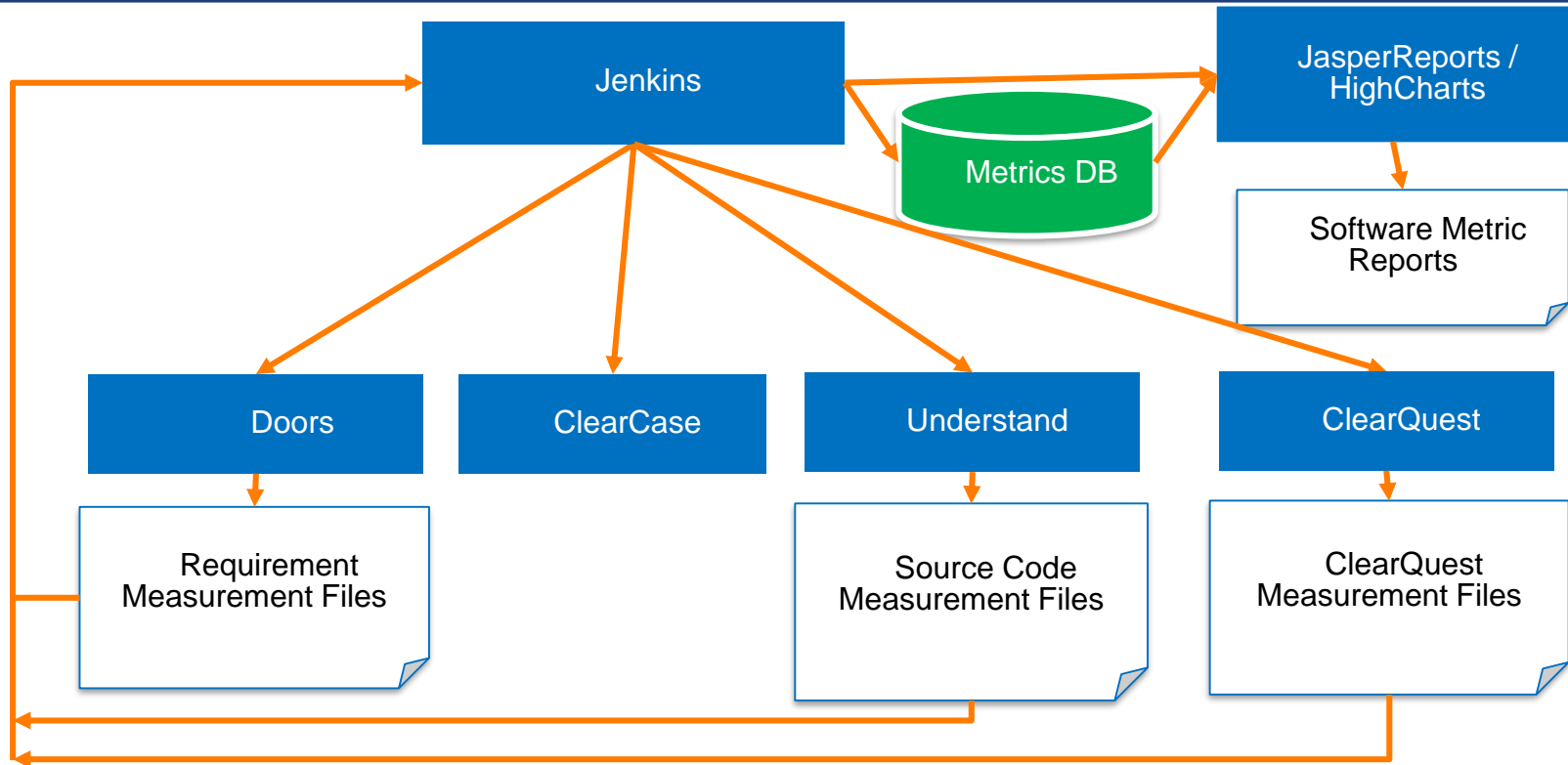
- ✓ Repeatability
- ✓ Reliability
- ✓ Practicality
- ✓ Effectiveness
- ✓ Efficiency

The process automation over Continuous Integration provides an automated QA (Quality Assurance) infrastructure.



To setup the CI infrastructure, a virtual server is allocated to run Jenkins and other tools required. Jenkins is then customized to our needs by using several plug-ins, most important ones are listed below:

- Version Control plug-in (e.g.; ClearCase)
- Management plug-ins (e.g.; JIRA)
- Feedback plug-ins (e.g.; Email extension)
- Analysis plug-ins (e.g.; Coverity)
- Test plug-ins (e.g.; JUnit)



The process of metric collection is fully-automated with the use of Jenkins, which is responsible of the tool orchestration within the process.

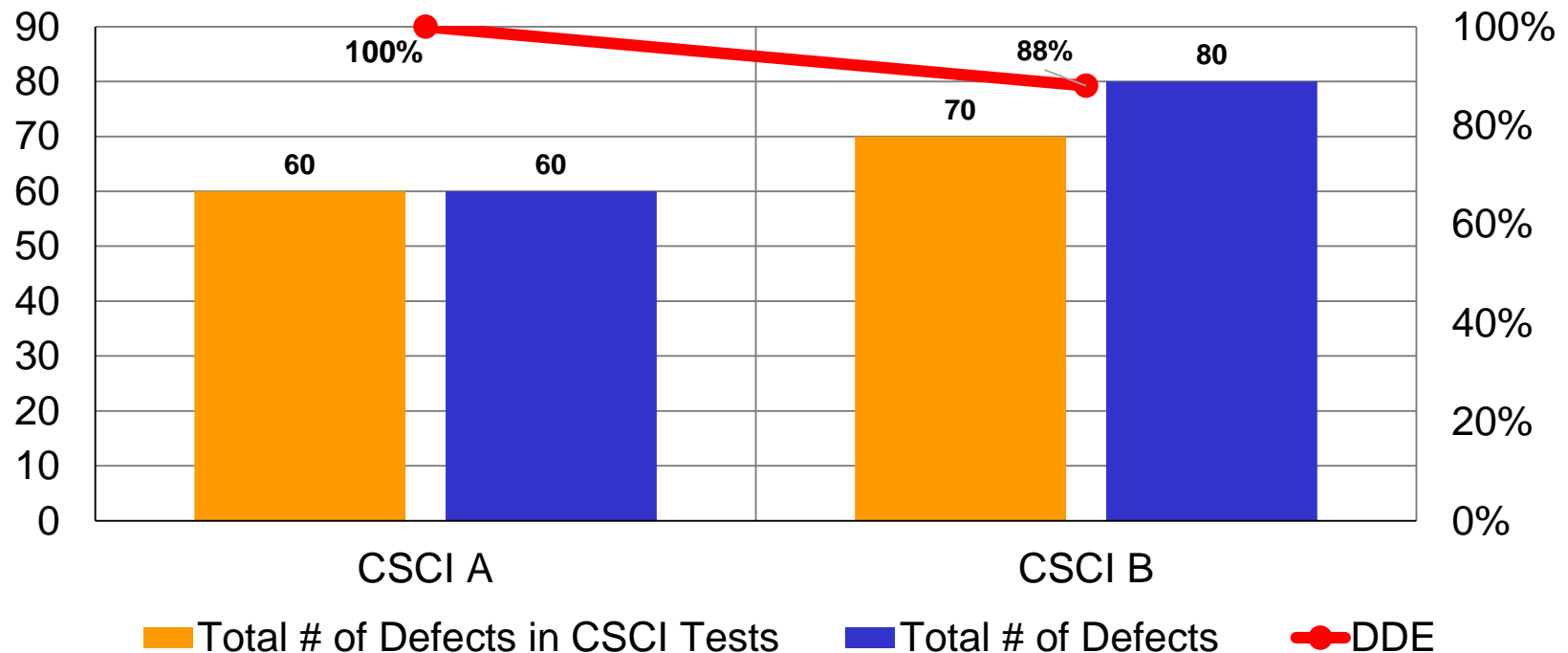
Jenkins is used for;

- Automating the collection of software metrics
- Continuous integration
- Automated release management

Id	Software Metric	Source
SM1	# of requirements	DOORS
SM2	# of requirements that are implemented	
SM3	# of requirements that are verified	
SM4	# of requirements for engineering use cases	
SM5	# of requirements that are verified by unit tests	
SM6	# of requirements that were added/removed/changed since the last baseline	
SM7	# of requirements that contain "To Be Determined Later"	
SM8	# of requirements that contain vague words	
SM9	# of classes	Source Codes
SM10	# of source code files	
SM11	# of functions	
SM12	KSLOC (Source Lines of Code/1000)	
SM13	Percentage of Comment/Implementation	
SM14	# of functions that have cyclomatic complexity more than 10	
SM15	Maximum cyclomatic complexity	
SM16	# of components	
SM17	Total # of Defects detected in CSCI and system integration tests	ClearQuest
SM18	# of defects that were resolved	
SM19	Total # of Improvements	
SM20	Total # of Improvements that were resolved	
SM21	Total # of Defects reported in CSCI Tests	
SM22	Total # of Improvements reported in CSCI Tests	
SM23	# of release announcements that were delivered for test	Release Automation Database
SM24	# of release announcements that were delivered for system integration test	

Defect Detection Effectiveness (DDE)

DDE = SM21 (Total # of Defects reported in CSCI Tests) / SM17 (Total # of Defects detected in CSCI and system integration tests)



For CSCI A, 100% of defects were detected in CSCI tests prior to the system integration tests. For CSCI B 88% of defects were detected in CSCI tests. As a result of the process automation based on CI, we realized 12% improvement in DDE measurement results in CSCI Tests.

Software	SM12 (KSLOC)	SM17 (Total # of Defects)	Software Defect Density
CSCI-A	66,5	60	0,9
CSCI-B	42,2	80	1,9

DD is the number of defects detected in CSCI and system integration tests per the size of the software (in terms of KSLOC).

$DD = \text{SM17 (Total \# of Defects detected in CSCI and system integration tests)} / \text{SM12 (Software Size)}$

Improvement in DDE and DD measurement results depicts that the CI based process automation lets the test and QA engineers to be involved in the development, continuously.

The e-mail feedback informs the test and QA engineers about the current status of the developed software throughout SDLC. The relevant QA process and product audits are triggered meanwhile. Those QA activities are carried out to evaluate the process performance by the help of process automation.

CSCI tests are carried out by software test engineers regarding the software product quality. The introduced framework provides a basis for software QA and test teams synchronously. For that reason, all the defects are detected till the end of CSCI tests for CSCI-A.

An evaluation is made based on SPI (Software Process Improvement) Manifesto principles:

1. **Know the culture and focus on needs:** The needs were elicited from all relevant stakeholders (i.e.; Software developers, software test engineers, software quality engineers).
2. **Motivate all people involved:** Teamwork is carried on with the participation of all stakeholders. Periodical meetings were conducted to keep all the team motivated.
3. **Base improvement on experience and measurements:** The improvement is observed in a quantitative manner by means of measurement and analysis (via DDE and Defect Density metrics).
4. **Create a learning organization:** Other development teams are informed about the study in the in-house design technology conferences.
5. **Support the organization's vision and business objectives:** To satisfy the customer needs and expectations, it is important to release bug-free software. This improvement study aims to establish an automated DevOps infrastructure for this purpose.

6. **Use dynamic and adaptable models as needed:** Scrum is applied in this study.
7. **Apply risk management:** Risks are identified and mitigated regarding their severity and likelihood.
8. **Manage the organizational change in your improvement effort:** The study is carried on regarding the Unfreeze-Move-Freeze model. Initially, the need for automation is realized. Then the CI based process automation is implemented. Then the development, test and QA activities are conducted on the framework, further.
9. **Ensure all parties understand and agree on process:** All the relevant stakeholders (i.e.; development, test and QA engineers) are involved.
10. **Do not lose focus:** The concentration on the main focus is kept by means of the measurements. Measurement results provided a quantitative monitoring approach to the improvement team.

Process Automation based on Continuous Integration brought the following benefits to ASELSAN:

- ✓ Improvement in the measurement results depicts that the CI based process automation lets the test and QA engineers to be involved in the development, continuously.
- ✓ Increased visibility among the relevant stakeholders
- ✓ Caught issues earlier
- ✓ Spent less time debugging and more time adding features
- ✓ An objective approach for quantitative evaluation of software progress
- ✓ Reduced integration problems
- ✓ Provided detailed insight from management perspective
- ✓ Increased software quality
- ✓ Decreased paperwork

- We plan to spread the improvement study to all software products.
- We intend to improve the software process and product quality in a quantitative manner.
- A systematic QA approach is needed to work in line with the software development and test teams.
- This improvement study results will be considered to establish a software quality improvement guideline.

1. Integrating Quality Assurance into the Software Development Life Cycle, Leslie Tierstein, STR LLC and Hilary Benoit, W R Systems, Ltd.
2. M.Fowler, "Continuous Integration", Thoughtworks, 2006, http://www.martinfowler.com/articles/continuous_integration.html
3. Ö.Acar, "Exteme Programming", 2008, <http://www.kurumsaljava.com>
4. Continuous Integration, https://en.wikipedia.org/wiki/Static_program_analysis
5. Regression Testing, www.softwaretestingclass.com
6. T. DeMarco, "Controlling Software Projects, Management, Measurement & Estimation, 1982, p.3
7. IEEE Standard for Software Reviews and Audits - IEEE Std 1028™-2008
8. Software Development Process Audits - A General Procedure, Stewart G. Crawford & M. Hosein Fallah, AT&T Bell Laboratories, Holmdel, New Jersey 07733. USA
9. Survey on Impact of Software Metrics on Software Quality, Mrinal Singh Rawat, Arpita Mittal, Sanjay Kumar Dubey, Boeing. USA
10. SPI Manifesto version A.1.2.2010, Jan Pries-Heje, Jørn Johansen

aselsan

www.aselsan.com.tr

Radar and Electronic Warfare Systems Business Sector

ASELSAN is a Turkish Armed Forces Foundation company.



Thanks... Questions?

İlgi KESKİN KAYNAK, Evren ÇILDEN, Selin AYDIN
Sep 18, 2019