The New Agile that brings Software Engineering Excellence in the Cloud

Software development practices have evolved over time and each has had its own limitations which have resulted in adopting new practices or processes every time. What many have missed is, combining the best practices of the past and current to drive software development. In this document, we will see some of the best practices in the past and current and how effective they are, when combined with software engineering excellence.

aspire SYSTEMS
attention. always.

History

Embracing Continuous Integration and Deployment in Agile Process

Businesses are always in the need for more agility, faster time-to market, tighter and faster feedback loops and reduction of work in progress while maintaining the software's quality throughout the process at the same time. The traditional agile methodology with sprints and scrum did not suffice these needs and hence Continuous Integration, where build processes are automated to enable small and frequent releases, along with Continuous Delivery to keep the software deployable throughout the lifecycle was adopted.

Typical Agile Process

The key principle of Agile is releasing working software frequently. The following exhibit showcases the typical agile process methodology in a non-cloud environment.
From the above diagram, we can say that CI/CD needs to function on a daily basis as daily scrum demands that working code should be merged and integrated every day.

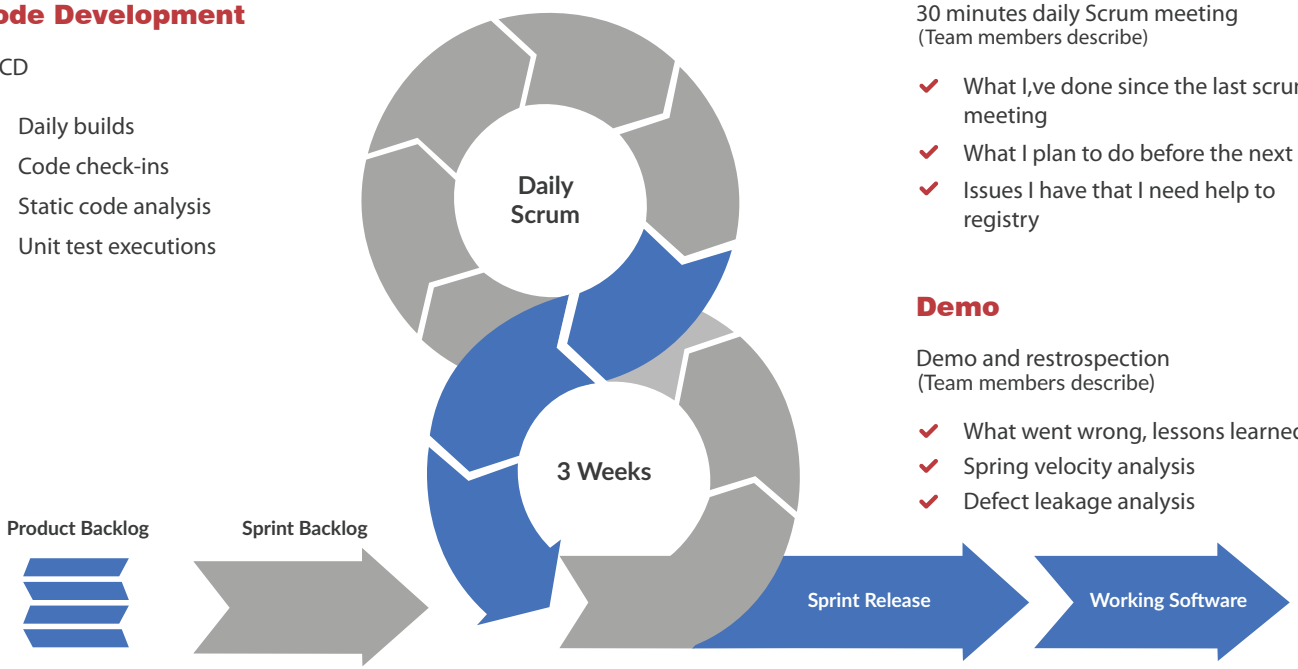
- Merging code on daily basis – This removes the risk of merging the code at a later stage which might result in bad code leakage, merge issues, build failures, fixing the merge conflicts and retesting.
- Automation of unit and integration testing – The practice of having to run the tests on the build to ensure that the tests behave as expected by the developers.
- Quality control – A continuous process to measure and monitor the code quality using automated static and dynamic review tools and to measure the performance of the code.

Agile Process

Code Development

CI/CD

- Daily builds
- Code check-ins
- Static code analysis
- Unit test executions



Daily Scrum

30 minutes daily Scrum meeting
(Team members describe)

- What I,ve done since the last scrum meeting
- What I plan to do before the next
- Issues I have that I need help to registry

Demo

Demo and restrospection
(Team members describe)

- What went wrong, lessons learned
- Spring velocity analysis
- Defect leakage analysis

CI/CD with Agile

- Enables small and frequent release of working software
- Reduces cycle time to get new features into production
- Software is made deployable throughout the lifecycle

Implementing CI/CD in the Cloud for a Distributed Agile Team

In today's world, changes are inevitable for businesses. In order to be competitive, changes have to be embraced quickly. One way of doing this is by having a distributed agile team which uses Feature Boxed Release and Branching Model to manage the increased number of builds and deployment. In a non-cloud environment, scaling and elasticity becomes challenging as the process to scale hardware, manage project resources becomes time consuming. Hence for the team to churn out results faster, businesses have started to use the power of cloud.

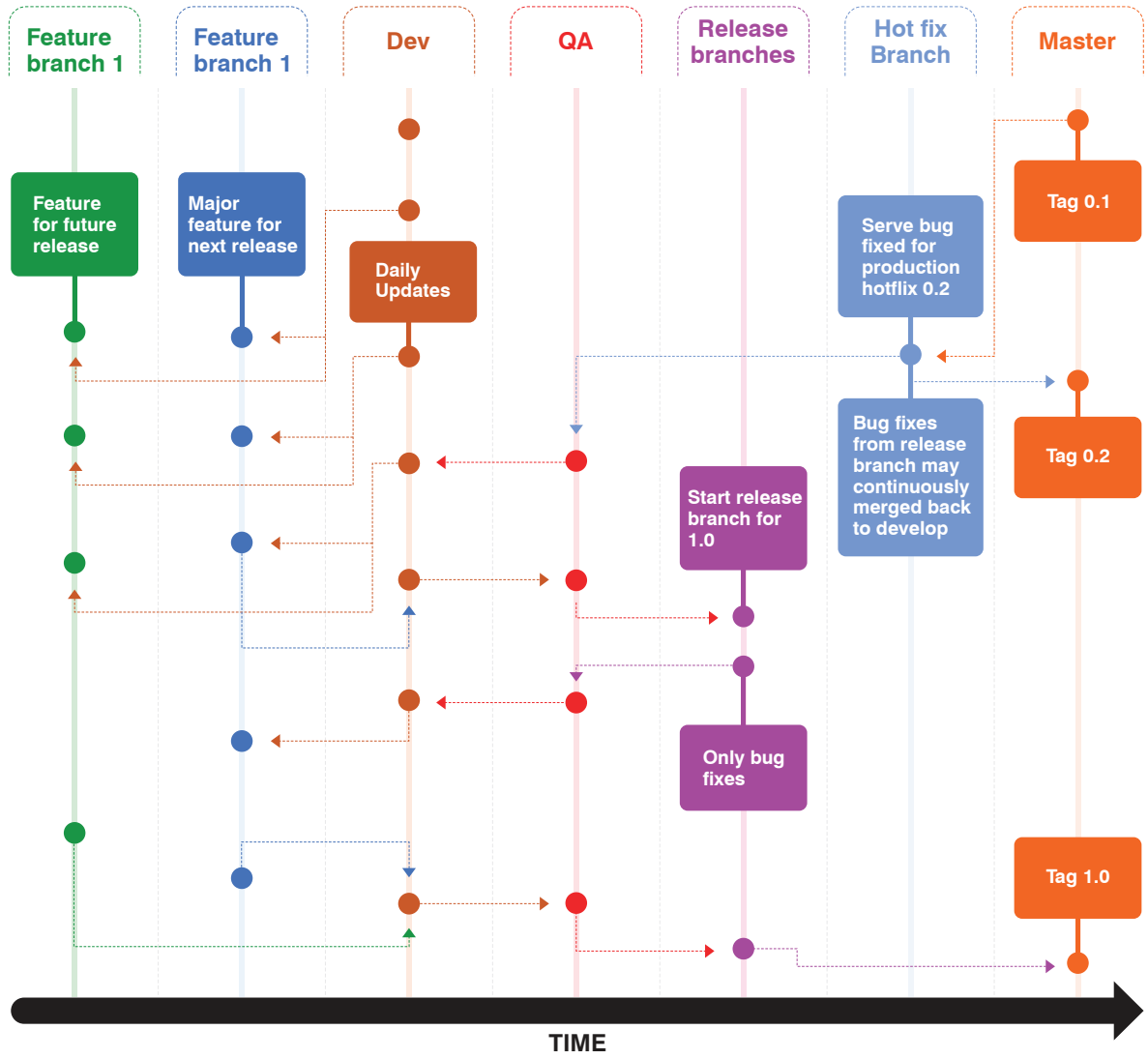
Feature Boxed Release and Branching Model for a Distributed Agile Team

Let us look at how code development process works for a distributed agile team which uses Feature Boxed Release and Branching Model.

Build Process

It is a known fact that a robust build process is required for agile development. For a small agile team, a high available and scalable build infrastructure may not be necessary, but for a distributed agile team and feature boxed release process, there will be many feature branches on which agile teams work. There will be many code check-ins and merging from and to the main branch and build will take place for each branch independently.

In this complex development environment, the build process should be completed within the acceptable build time limit. If the build process exceeds the limit, then achieving CI will be a challenging task.



Code Merge

Typically any code repository will have a trunk-active development branch, tags with shippable code and branches for hot fixes, customer specific code and so on. Here, code merging would be done in the trunk and syncing the code is challenging and implementing CI/CD is time consuming. To overcome these challenges, many have started moving towards feature boxed release and branching model. In this model,

- The trunk has code that is current and is unit tested, integrated and build done successfully on a daily basis
- Branches are created for features where each branch is available only for that feature
- Code gets merged to trunk on a successful build from the feature branch
- When a feature is completed it gets moved to the QA branch, release branch, tag and so on

Since there are several feature branches from a distributed agile team, CI/CD must be implemented on the feature and trunk branch so that the working code is merged and integrated with the live branch. Also in cases where the code has to be pushed to the dependency feature branch, an effective CI/CD implementation is required to ensure code sync is current.

To implement the above code development process, a scalable infrastructure is needed. Hence, embracing cloud becomes necessary.

Static Code Analysis

Code quality is a key area that needs to be checked on daily basis and most importantly after every code merge or successful build. With a distributed agile team it is very important that we have a consistent way of monitoring and checking the code quality, so that every developer checks a code which complies with the coding standards defined for the project.

There are many static code analyser tools which can be integrated with CI/CD environment so that the code is analysed for every build. With cloud providing better elasticity and scalability, it is easier to run these tools for every check in or for every build as required.

With distributed agile team and parallel development using featured boxed release process we need to have an automated way of reviewing the code and tools like Sonar and Ndepend provide required automation of code analysis.

Unit Test Case Execution

Running unit test cases as part of the build process reduces the leakage of bugs and also checks for any breakage of dependent functionalities. With distributed agile team and feature boxed branching model, automating the unit test cases for the integration build will ensure there is no functional breakage. Also, the coverage of test cases to be executed will be high with unit test case automation. The above items does provide a clear indication on why agile development needs be executed in a Cloud development environment

Agility in the Cloud

➜ Offers zero lead time for starting development work in a distributed environment ➜ Feature boxed release management for shorter release cycles ➜ Enhances Developer's productivity ➜ Optimizes development and operational costs

The Way Forward

Bringing Software Engineering Excellence in the Cloud

Agile development is prone for long term quality risk, if the quality is not checked in frequent intervals. In a cloud environment, prevention of quality repetition in upcoming sprints is very important, if not, the bad code from distributed sprint teams or feature based branches will be integrated during the CI/CD process. At a later stage, fixing the bad code becomes expensive due to the increased number of builds and integration.

What is Software Engineering Excellence?

Delivering working software that meets the quality aspects like design, architecture, coding, testing, release engineering, requirements, environment and usability ensures software engineering excellence. Following are the quality standards to be adhered to

Design Quality

- Early detection of flawed design pattern implementation
- Optimization of design for right data structure presentation and their performance (like list, set, map, tree)
- Concurrency assessment with immutable objects and thread safety

Code Quality

- Compliance of code with respect to defined language specific guidelines
- Discovery of redundant, un-optimized code in the implementation
- Assessment of character encoding, memory allocation, usage of namespace so that runtime classes generated are unique

Standards Compliance

- Standards compliance for code with respect to W3C, PCI, GLBA
- Product specific coding standards compliance

Automated Unit Testing

- 100% code path coverage for the implemented classes or methods
- Usage of mock references for the third party and non-available implementations

Usability

- Simplified information architecture flow with faster access to information
- Layout and positioning of widgets and support accessibility for all
- Browser compatibility, responsive web design

Security

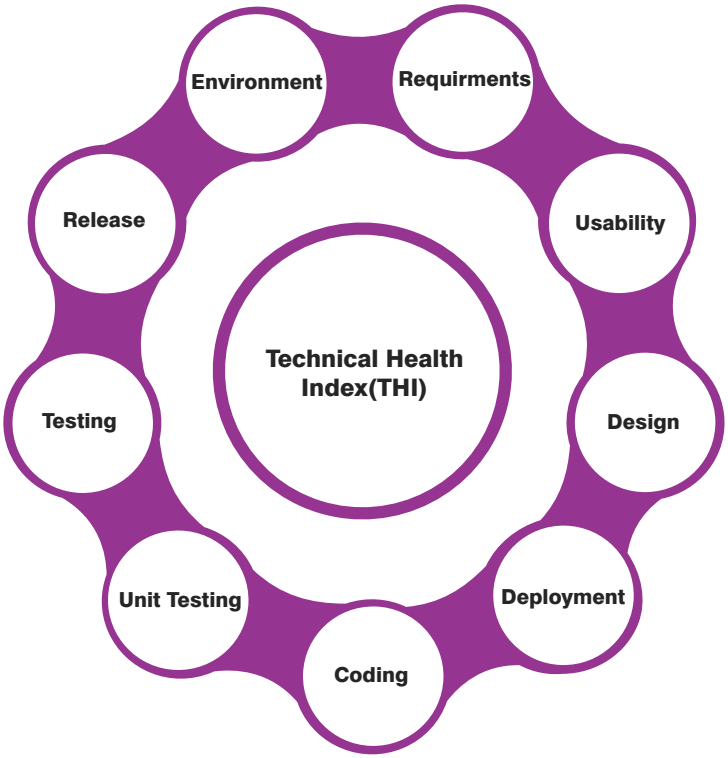
- Identification of potential security threats like SQL injections attacks
- Identification of cross site scripting possibilities to avoid phishing
- Buffer overflows in the code to eliminate exploitations by hackers.

How to Measure Software Engineering Excellence?

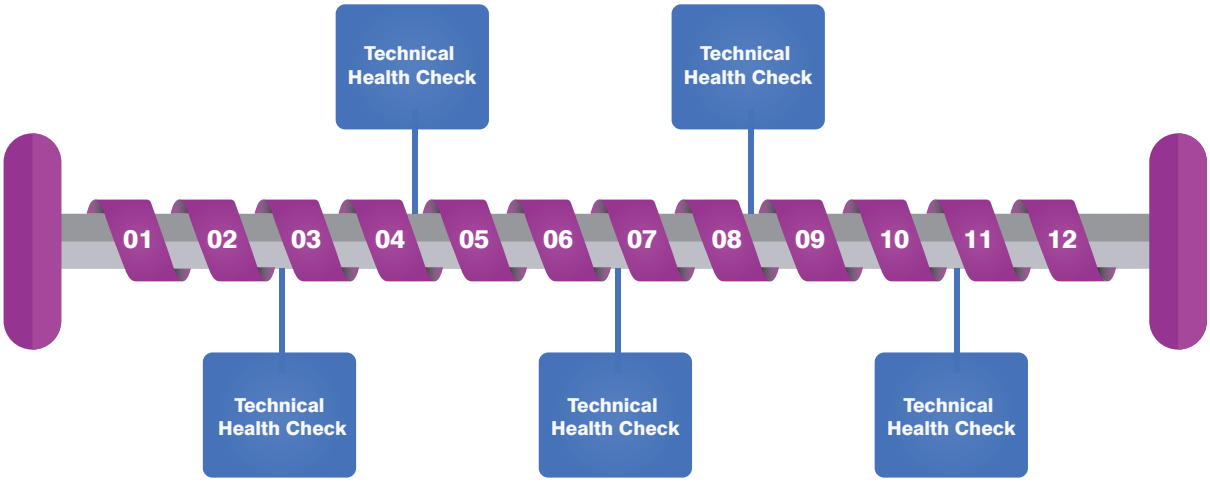
There must be a process in place to check the above mentioned quality aspects in order to measure the technical maturity level of the software. In other words, it means arriving at the Technical Health Index of the software. Thus, the software must be audited for technical quality compliance and assessed on defined technical criteria. Based on the level at which the software is checked across the criteria, the engineering maturity level must be calculated.

The process can cover the following areas in the software development lifecycle.

The below exhibit provides an overview of how the Technical Health Index check is planned and executed in agile development model and how the output can be used for analysis or as inputs for further process areas as mentioned in the right bottom box. The frequency of the audit can be determined based on the sprint cycle and tickets planned in each sprints. The left bottom box provides the various artifacts that will be considered for audit sampling.



Sprint Cycle



Input

- Sonar (static code analysis and code coverage report) input
- Jenkins build status
- Project workspace
- Coding standards

Technical Health Check

Output

- Technical health index report

- Quality indication of the software
- Technology standards compliance
- Standards compliance like PCI -DSS, web accessibility, etc
- Training input
- Input for performance appraisal of the developer

In an environment with CI/CD in the cloud and having a distributed agile development team using the cloud based development infrastructure, it is very important that the cloud environment goes through the Technical Health Index Check frequently to avoid any quality issues when integrating code from different teams and ticket branches frequently.
The audit can be sampled at any level starting from overall product, module level or particular version of the code, specific release or ticket level. In a distributed agile team this can be used for checking the quality of output from each agile team.

Conclusion

CI/CD with some static automated code quality tools or unit test automation takes care of the quality from functional and static code analysis perspective but there is no focus on checking the quality of the design, functional code, and data in unit testing and so on. Having a well-defined CI/CD process in the cloud alone will not guarantee the quality expected by the technology stakeholders. Only when it is combined with engineering excellence, the stake holders can be confident that the working software being delivered is within a certain quality level and decisions can be made based on the technical health index output.

Software Engineering Excellence

- ➜ Ensures software engineering compliance with coding and design standards
- ➜ Enables releasing working software with quality ➜ Reduces rework effort ➜ Controls defect leakage

ABOUT ASPIRE

Aspire Systems is a global technology services firm serving as a trusted technology partner for its customers. The company works with some of the world's most innovative enterprises and independent software vendors, helping them leverage technology and outsourcing in Aspire's specific areas of expertise. Aspire System's services include Product Engineering, Enterprise Solutions, Independent Testing Services, Oracle Application Services and IT Infrastructure & Application SupportServices. The company currently has over 1,400 employees and over 100 customers globally. The company has a growing presence in the US, UK, India, Middle East and Europe. For the fourth time in a row, Aspire has been selected as one of India's 'Best Companies to Work For' by the Great Place to Work® Institute, in partnership with The Economic Times.

NORTH AMERICA | UK | BENELUX | NORDIC | MIDDLE EAST | INDIA
+91 - 044 - 67404000, +1- 630- 368 - 0970, +44 - 203 170 6115

For more info contact
info@aspiresys.com or visit www.aspiresys.com