Our Technology. Your Business.

Process Maturity At Sosaley

SOSALEY

Process Maturity At Sosaley

M. Annamalai

CR. Venkataraman

Sosaley Technologies Private Limited

Chennai, India

www.sosaley.com

Process Maturity at Sosaley

February 2019
Chennai India.

About The Authors

M. Annamalai. Annamalai holds a Bachelor of Engineering degree in Electronics and Instrumentation. He has been trained in CMMI Level 3, Six Sigma and ISO 9000. He is a CMMI Appraisal Team member. Annamalai has over 16+ years of experience in project management. He is an expert on Agile project management and is a Scrum master.

C.R. Venkataraman is Director Marketing at Sosaley Technologies. He has over 35 years of experience in software development and marketing and has led many national and international projects in finance, manufacturing, networking, media, etc. Venkat holds a Master Degree in Economics from Kurukshetra University.

ACKNOWLEDGEMENTS

The authors have borrowed from multiple sources on the Internet.

(1) The Mozilla Foundation that set the trend for using a set of widely dispersed technologists for community development of a large scale project. In our opinion, they set the rules for such development and have fine-tuned it over the years. For more details on the Mozilla Foundation please visit https://www.mozilla.org/en-US/.

(2) Atlassian, owners of Jira, Bitbucket and other products that are used extensively in the Agile Methodology. Jira and other products of Atlassian are trademarks owned by Atlassian. For more details on Atlassian and their products please visit https://www.atlassian.com/

(3) The Scrum methodology from multiple sources:

   a. http://www.mountaingoatsoftware.com/agile/scrum

   b. https://www.softwaretestinghelp.com/agile-scrum-methodology-for-development-and-testing/

   c. https://www.guru99.com/agile-scrum-extreme-testing.html

Contents

Introduction

Working with a multitude of clients across the world, Sosaley has developed a process that is stable and dependable. It helps us meet time and project targets consistently. It helps us do the following on a regular basis:

- Satisfy our customers with quick and continuous delivery of software.
- Incorporate changes to the specifications and requirements.
- Deliver prototypes and working software for evaluation on a regular basis.
- Motivate and offer the necessary environments for our people to complete the tasks.
- Encourage research and curiosity to try something new.
- Continuous attention to details and excellence in technology.
- Encourage all team members to interact with clients and their representatives and reflect on the feedback received.

Service Positioning

The following points show how we position our services:

- We focus on customized solutions, product development, client/server, web, embedded and, wireless technologies.
- We have continuously added new toolkits, platforms, and technology to our knowledge repertoire. We have frequently demonstrated cross-platform expertise.
- We have acquired new technologies quickly through continuous learning and knowledge management. We have a time tested technology acquisition model in place.
- We believe that the success of outsourcing boils down to the clarity of communication, strictly adhering to process and quality standards.
- We have evolved our business models from practical interactions with clients. These follow client requirements as gospel.
- We have created business models and standards to deliver value to our clients. As client needs are unique, we have kept the models and standards flexible and adaptable.

Development Philosophy – Transparent Reporting™

Sosaley Technologies has followed this method of seamless interaction with the client since the founding. Over the years, we have evolved our practices to allow for more transparency into our operations. We have implemented the Agile methodology that delivers up-to-the-minute information on work-in-progress to the client.

This method of software development allows you, as a client, to keep a virtual eye on all stages of the SDLC. You can interact with the team and discuss technical issues. You can avail of the support of a Product / Project owner or Scrum Master. The transparency of reporting is enhanced by documentation at each stage of the development process.

Introducing STLCM

What is STLCM?

Sosaley's Total Life Cycle Management (STLCM) is a mature, collaborative development solution that spans software (and product) development from conceptualization to versioning and beyond. STLCM solutions deliver strategic advantages to both existing and new projects and products.

Companies today face the challenge of maintaining high levels of customer support and responsiveness. This has to be done in parallel with investing in the hardware/software to keep them competitive in terms of features and usability. Very often, the secret of success in software is the ability to maintain an existing customer base, protecting it against erosion, and adding new customers, all at the same time.

STLCM presents aggressive and cost-effective leverage to protect your investment in software.

Using a collaborative approach, STLCM brings together the strengths of the software architects and offsite, mostly offshore, development and maintenance teams. The STLCM team can focus on augmenting resource bandwidth for faster deliveries, automated testing, process orientation, and R&D to support the core strengths of the architectural team.

STLCM Value Propositions

The objective of the STLCM offer is to deliver strong value propositions that extend beyond dollar savings and enable overall competitive edge in the market place. With STLCM, offsite software development evolves from cost-effectiveness into a strategic advantage.

The advantages of Sosaley's TLCM are:

1. Scalable, dynamic bandwidth of trained and experienced resources
2. Reduced time to market, faster reaction to market needs
3. Reduced elapsed time in development through process innovation
4. Better quality and well-tested software
5. Process orientation for sustained benefits
6. More return per invested dollar – reduced risks
7. Dollar savings through global sourcing
8. Inexpensive prototypes for market testing

Scope of STLCM

The STLCM initiative is based on the software itself. The main determinants are shown below.

- The boundaries of an STLCM initiative are defined by the software itself - its attributes and the characteristics of its life cycle. These provide valuable insights into the software's positioning and it's future.
- The vision for the software and plans to deliver ROI to the buyer strongly impact the direction of an STLCM initiative.
- The possibilities for software evolution, the challenges, and its evolving targets have an equal impact on an STLCM initiative.

Software Visioning

The vision for the software has to be noted down in both tangible and non-tangible terms. The possibilities for value addition have to be listed in line with the vision for the product - to support 'top 5' operational environments, or, to have a backup solution for any user? These possibilities are to be prioritized, created as EPICs, and a Roadmap created for multiple phases of STLCM. Each EPIC then breaks down into definable Stories with specific deliverables. Each such Story is then broken into specific tasks with teams, infrastructure, schedules and deliveries assigned.

Sample Tasks

The following is a list of sample tasks that can be performed under STLCM.

- Development from scratch on new platforms
- Collaborative or completely outsourced development
- Evolution of design from concepts or broad-based specifications
- Porting existing software to new operating systems
- Backend migration
- Process orientation (QA) for existing products & documentation
- Automated testing
- Outsourced testing and bug fixing
- R&D, prototyping, and feasibility studies

Probable Team Compositions

The composition of the team depends on the nature of tasks allocated to the STLCM team. However, the following roles can be considered as basic for any STLCM initiative:

- Product / Project Owner
- Scrum Master
- Solution Architects
- Technical Leads
- Database Architects
- QA Lead – Process Manager
- QC Resources
- Programmers – working on a task basis
- Programmers – working on long term initiatives
- Programmers – working to automate Quality Control (testing)
- Documentation resources

SOSALEY

Process Orientation

The Need for Process Orientation

Process orientation in an organization enables it to deliver results to its customers and to its own growth on a consistent basis. Processes define the way the organization handles project management and problem-solving. Processes also help the organization to avoid repeating mistakes it has made before.

Put simply, process orientation increases the probability of success.

Why Processes? – Statistical Support

Companies and governments in the United States spend more than $250 billion each year on the development of approximately 180,000 software projects. Over 31% of these projects will be cancelled before they ever get completed in large companies, and only 9% of projects will come out on time and on budget. In 1995, American companies and governments spent $81 billion on cancelled software projects. Projects involved in designing and implementing software generally rely on development methodologies, such as computer-aided software engineering (CASE) tools. This approach, however, has not resulted in success as attested by a large number of failures described above. While tools and people are essential to software projects, experts now agree there is the need for well-defined, structured, and disciplined management processes that ensure results - a comprehensive management process that provides the necessary structure to support the design, implementation, and maintenance of software throughout its life cycle.

Process Orientation At Sosaley

At Sosaley, we have designed and tailor-made a program that is suited to support our ongoing activities. A primary feature of this life cycle management approach is its functional flexibility. At a minimum, the STLCM initiative will address the following standards. These are defined by the CMMI under 'Software Process Improvement':

- Project definition
- Requirements analysis
- Risk management plan
- Quality assurance plan
- General design
- Feasibility study
- System requirements
- Software requirements
- Detailed system design
- Construction
- System implementation and testing
- Training
- System operational support
- System maintenance support

Standards under STLCM

A framework of standards has been developed, practised, and evolved for software under STLCM. Usually referred to as the Software Quality Assurance Process (SQAP), these standards span the entire software development life cycle from requirement specifications to post-deployment maintenance. The philosophy of development - SAD, UML, RUP, or, XP - determine the processes for a standard software development project.

The following standards, in addition to development processes, are relevant for STLCM:

- Design standards
- Coding standards
- Performance criterion
- Behavioural specifications
- Installation and deployment standards

These standards ensure better control over the development process. They enable predictability and maintainability of the code. Transfer of knowledge across teams and resources becomes easier and faster. The activities in each phase are shown on the next page.

QA Process Orientation

Development Phase	Activity Description	Artefacts	Check Points
Project Inception	Project Kick off.	System Vision Document.	System Vision approval.
	Project Roadmaps.	Detailed road map with release dates.	**Review of Roadmaps**, Formation of CCB process.
Requirements Elicitation	Scope Definition. Detailed requirements gathering (User interview or any other technique as required by the project).	Scope requirements clearly documented in the Confluence.	Approval of business and product owners.
	Quality Gateway.	The Roadmap, Release, Epics, Stories, and tasks mapping in Releases.	Daily Scrum calls, periodic Sprint plan, and retrospective meetings.
	FPA	FP Document comprising FP Count, Effort & Cost Estimate and Story point effort and cost.	**FPA Review** (Internal) during every Sprint plan with the team.
	Document & Version Control.		Functional and physical configuration audit in Confluence.
	Miscellaneous	Project progress report, Revision of Project Schedule (if required).	**Periodic Reviews** of Daily Scrum, Sprint and retrospective plans.
Software Design	Design	Test Cases, HLD, LLD, Contract Notes, Database Design, UI Design, Prototype (if warranted), Traceability Matrix.	**Design Review.** Traceability Matrix between SRS, Design, and Test Case Documents.
Software Construction	Coding	Source Code	**Code Review.** Traceability Matrix between Design and Source Code.
Testing	Unit, Integration, System and User Acceptance Testing.	Test Report, Test Summary Report, Validation and Rectification Report, Defect Metrics Report.	Done in JIRA Board and results updated in Confluence.
Software Release	Software Release	Consolidated Test Report. Software Release Notes, Shipment Details Report.	Automatic **deployment** using DevOps.

Defect Management

Defect Management is one of the direct benefits of process orientation. Defect management can be divided into defect detection and development of solutions to clear the defects.

Defect detection can be from internal testing and from external sources like an end user forum. Under the STLCM initiative, a Jira ticket will be created in the JIRA Board. All defects identified by internal as well as external sources are posted in JIRA. The bug tracking tool allocates defect IDs to each instance. Defects identified by external sources will be channelled through the technical support group or by the field force in touch with customers. The quality control (testing) team constitutes an internal source for defects. QC as part of the STLCM initiative is explained below.

The objective is to avoid duplication and also prioritize defect correction based on the levels of severity. Defects with available diagnosis and perceived level of severity will be posted on a secure Extranet for access to the STLCM teams.

The workflow guidelines for defect allocation can be defined and implemented into the extranet. For example, a particular category of defects can be restricted for access to a group specializing in the relevant area. Or, a particular defect can be allocated to a certain number of resources to ensure the spread of effort.

Effective Quality Control

QC is the internal wall for defect identification. In an ideal situation, defects should not pass through the QC team at all. The effectiveness of a QC framework is based to a large extent on the Software Quality Assurance Plan of STLCM.

Test case designs, for example, depend almost entirely on the specifications and low-level designs. For a sound testing mechanism, it is imperative that the QC team works in tandem with development and starts functioning on the same day. The 'Design' phase comprises of:

- High-Level Design
- Low-Level Design
- Test Case Design
- Database Design
- Hardware design
- Firmware design
- User Interface Design

Design review for various Stories is the control point at this stage of development. An inherent attribute of this QA process is parallel progress on QC. This ensures that Test Cases act as control points for the software design. The EPIC stories cover specifications, designs, and the test cases ensuring complete coverage. At the end of the design phase, test case designs are converted to test scripts.

QC in STLCM

QC method in STLCM is designed to cut down on the elapsed time between completion of unit development and commercial release. This phase of work can be done through parallel and automated QC methods. The mechanism under STLCM ensures that a good amount of testing is automated and bug lists are published almost immediately after the unit development is completed.

QC in STLCM is a combination of:

1. Custom testing tools
2. Automated testing
3. Complimentary manual testing
4. Stress / Load testing

All of the above mechanisms will be based on test case designs and client requirements. The rendering of the design into actual testing will be automated in methods (1) and (2). Manual testing (3) will be called for when testing cannot be or need not be automated. Stress / Load testing (4) will be based on 3rd party commercial tools.

Automated Testing

Custom testing tools are standard under the STLCM initiative. These tools focus on specific aspects of QC or can be a bunch of test cases tied together with a PERL script for ease of testing operation.

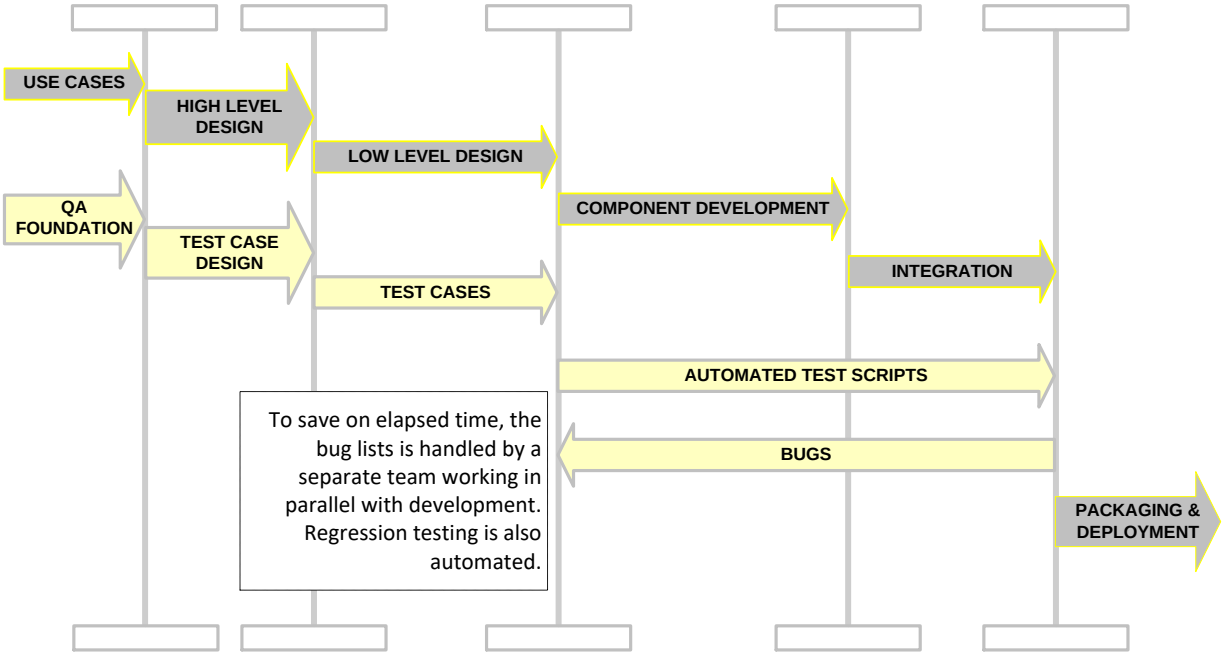
Under the STLCM initiative, code developed, modified, ported or re-engineered will be stored in a central code repository. The storage has been built to include a certain amount of code based testing. When a project milestone is achieved, the relevant set of automated tests will be triggered and the outcome of the tests will be made available to the stakeholders for that portion of STLCM.

A considerable amount of unit testing and a reasonable amount of integration testing can be covered using the code based testing method.

Savings on Automated Testing

In standard development, the bulk of the testing activity happens after the development is complete. This QC phase very often adds a quantum of elapsed time after the bulk of the development is complete. The development team spends a considerable amount of effort in a 'bug fix' mode after the development process.

In more matured models, unit testing happens in parallel with development followed by integration testing. Using a combination of a layered development model and automated testing, the elapsed time on development can be cut down considerably.



This mandates that the foundations for testing be laid parallel to design. The High-Level Design for software development (or version development) will be used as the basis to create test case design. Along with low-level design, test cases will be created. Upon LLD sign off, the test cases will be fine-tuned. The LLD normally includes a component breakup of the system.

Test cases will be assigned to a component or a set of components. If class diagrams are included in the LLD, test cases can be assigned to each class. In parallel with component development, test cases for each component will be automated. The test cases will be deployed as a QC layer on top of the code storage – version control system, such as Bitbucket, SVN etc.

At the end of each workday, components scheduled for development on that day will be checked in and the pertinent set of automated test cases executed. In line with the progress made, automated test cases will be loaded onto the version control system and the code checked is tested immediately.

Control & Workflow

Control and workflow define the way day to day management activities are executed under STLCM.

Centralized Code Storage

The storage server is always on the cloud with access rights defined. At the STLCM end, the Sosaley's Team will ensure that there are no copies of the code in workstations, the development server or, the testing server. The code will be checked out for a specific purpose and will be checked back at the end of the workday.

Central storage of code also prepares the STLCM team for collaborative development with other development teams spread out in different geographical locations.

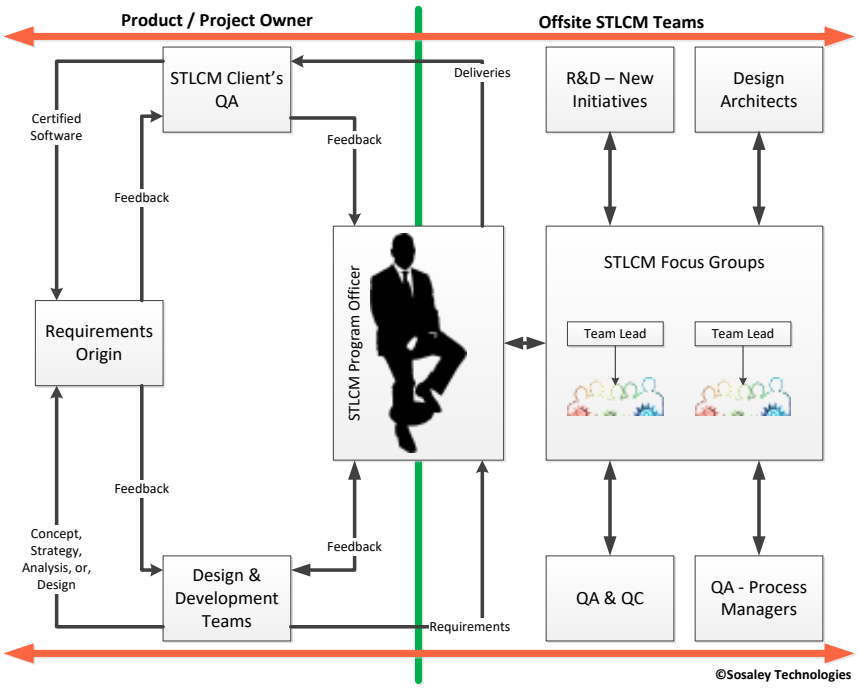
Matrix for Project Management

The amount of Sprint accomplished and the number of incremental releases constitutes an effective measurement of progress achieved. This is for any development cycle - day, week or milestone.

The performance and progress measurement is derived on the basis of EPICs and Stories completed. This can be expanded to cover other areas of project management including:

- Story or Functional estimation
- Milestone based delivery
- Stories & Task allocation
- Sprint accomplishments against goals
- Performance measurement & goal setting

The matrix for STLCM is derived from or based on popular matrices including Function Points Analysis.



SPA – Relevance to STLCM

Story points are a measure of the size of computer applications and the projects that build them. The size is measured from a functional, or user point of view. It is independent of the computer language, development methodology, technology or capability of the project team used to develop the application.

Development & Support Estimation

Story points have been used as an estimation technique. Estimation is obviously necessary for a cost-benefit analysis that justifies development. Even for strategic projects that need no quantitative justification, accurate estimation is required for proper staffing.

You can customize the estimation statistic you use (Story points, time, or issue count, for example) and the time tracking settings (remaining time estimate) to suit how you estimate and track work in your project.

Many Scrum teams separate estimation—used for measuring the size of a backlog and calculating velocity—from tracking—often the burndown of hours used during the Sprint to make sure that they are on track to complete the stories during the Sprint period—and use different units for each. A common approach is to estimate tasks in Story points and then track the tasks using hours. Jira gives you the flexibility to set your estimation and tracking statistics differently, depending on what suits your team best.

Software development needs to be estimated irrespective of the development methodology – fixed costs or man and material. This may not be easy as backlogs may stretch for a long time. Teams will be able to provide rough estimates, but accurate estimates are literally impossible even if time is spent breaking the requirements into tasks. However, from Sprint to Sprint, as they work through the Stories, the team will develop a way of completing a number of units of work they had roughly estimated at their speed.

In Jira, you can choose which type of units (Story points or time, for example) will be used for estimating and tracking issues. You do this by choosing an estimation statistic, then choosing to either use the same units for your tracking statistic or to use time tracking. Each Board can have a different type of estimation and tracking statistic.

Workflow Management - JIRA

The software development methodology we use for developing our software is AGILE. We use JIRA tool for AGILE. In JIRA, we do the following

- Sprint planning
- Daily Scrum
- Boards management
- Documents management
- Release management
- Agile reports

The workflow we follow is defined in the following points:

- Sprint planning is done for a period of two weeks. During this planning, we discuss the list of Stories and priorities for this Sprint release.
- For each Sprint, we pick important Stories and assign them to the respective resource. During this time we do a collective estimation on the task based on Story points or days.
- Once the Sprint is started, we have a daily Scrum call or a stand-up meeting. In these meetings, we discuss the tasks that are done, task planning, and identify blockers if any.
- Each task is linked to Bitbucket as a feature branch to check-in the developed code and to Atlassian Confluence for document reference.
- During the course of each Sprint, we have requirements discussion, design discussion, code review, unit testing, testing and release.
- For each task we have all documents done, design reviewed and task tested. For all these, we have a page created in Confluence.
- For version control, we use Bitbucket and we have an automated build system in which we have a static test and unit test inbuilt.
- The code, once reviewed, is pushed to the development branch from feature branch and the build pipeline process runs automatically.
- From the development branch, the tester pulls the code and tests. If the test passes, he pushes the code to the Master branch. The build process takes place executing the functional scripts and creating a report on tests passed and failed.
- We create a number of EPIC's and Stories and map them to the release for the particular version. The Scrum Board at any point gives us a clear view of the status of the task.
- At the end of each Sprint, we have a retrospective discussion on what went well and what failed. We also discuss possible improvements for the next or other Sprints.
- At the end of a Sprint, we analyze the Sprint report, Sprint velocity and Release burn down to see how we have fared in the Sprint.

Framework for Development

The scope of a framework is a skeletal foundation of the software that will be created with the adequate amount of cross-platform orientation built in. The framework is created to evolve with each phase of development. An evolved framework will mean a lesser amount of effort in the coding phase. The code can be created as classes or components in this framework to be implemented across a target set of platforms. This will include support for multiple operating systems and databases among other aspects of platform free development. The process of generation of the framework and code syntax can be automated. The programming phase of work under the proposed framework will be restricted to filling in the relevant portions to achieve the target code base. The code output generated through this process will have the syntax requiring for the logic to be filled in. This means that most of the decision points will be covered in the design phase.

Benefits of Framework

- Improved productivity
- Increased application availability
- Improved customer satisfaction
- Better requirements tracking
- Improved team coordination and project control

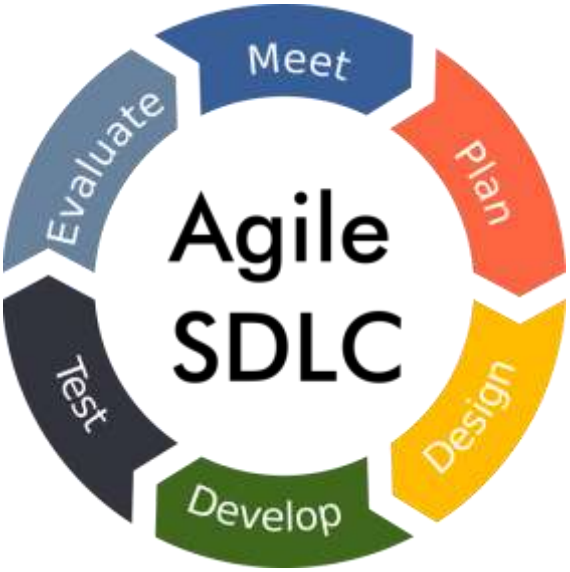
Versioning Workflow

The overall workflow under STLCM is as follows:

- A development branch is created from the Master branch
- A release branch is created from the Development branch.
- Feature branches are created from the Development branch.
- When a feature is complete, it is merged into the Development branch.
- When the release branch is done, it is merged into the Development and Master repositories.
- If an issue is detected in the Master, a hotfix branch is created from the Master branch.
- Once the hotfix is complete it is merged to both Development and Master repositories.

Agile Collaborative Development Methodology

Introduction to Agile Methodology



Agile software development is an approach to project management in software development. It is a combination of an iterative and incremental model with the focus on customer satisfaction through rapid delivery.

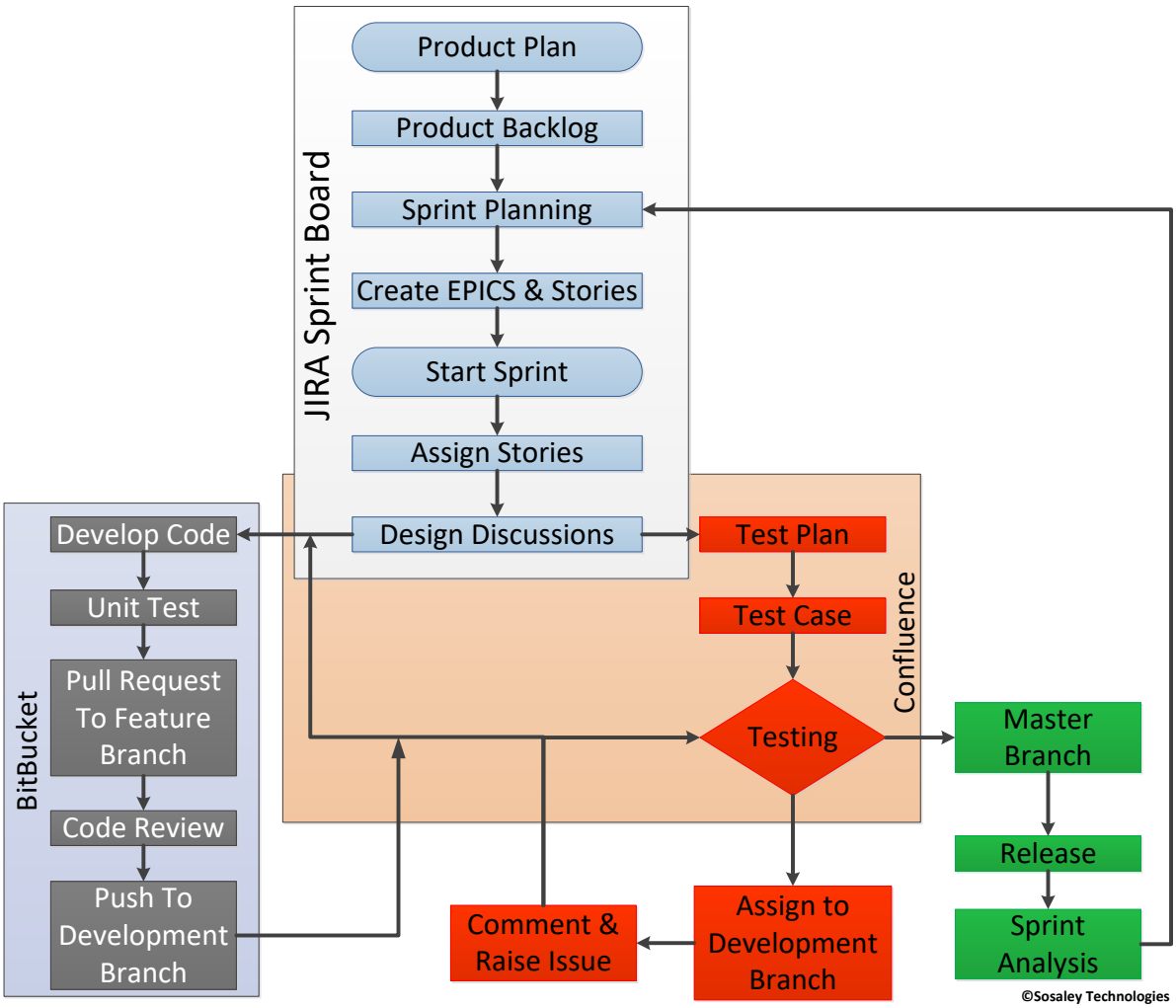
This method breaks the software into small incremental builds and is delivered in iterations. The whole software is broken into EPICs and Stories and listed in the product catalogue. Normally, deliverables are planned for the duration of a week or a month. In this duration, a list of task is reviewed and delivered. Each such duration is called a Sprint. Every Sprint is delivered as part of an iterative process.

The primary stakeholders are Product Owner, Scrum Master and team members comprising of the team lead, developer, tester, etc. The various stakeholders and components of Agile methodology are as shown in the table below

Component	Description
Scrum Team	A typical Scrum team has between five and nine people, with cross-functional expertise.
Product owner	The product owner is the product/project's key stakeholder and represents users, customers, and others. He is in charge of release, requirements, creations of EPICs and prioritization of the tasks.
Scrum Master	Facilitator for the agile development team, in charge of the Scrum Board, helps the team to stay focused on delivery.
Product backlog	A prioritized features list containing every desired feature or change to the product.
Sprint backlog	A list of tasks needed to complete the product backlog items the team has committed to fulfil in the Sprint.
Sprint planning	At the start of each Sprint, a Sprint planning meeting is held, during which the product owner presents the top items on the product backlog to the team. The Scrum team selects the work they can complete during the coming Sprint. That work is then moved from the product backlog to a Sprint Backlog.
Daily Scrum Meeting	Each day during the Sprint, a brief meeting called the daily Scrum is conducted. This meeting helps set the context for each day's work and helps the team stay on track. All team members are required to attend the daily Scrum.
Sprint review meeting	At the end of each Sprint, the team demonstrates the completed functionality.
Sprint retrospective	At the end of each Sprint, the team conducts a Sprint retrospection to understand the effectiveness of that Sprint.

How Does the Agile Methodology Work?

1. A Sprint is planned for 2-4 weeks. List of stories are discussed and priorities set for the Sprint release.
2. Important Stories are picked and assigned to resources. Collective estimation is done based on resource experience and Story points.



3. Once the Sprint starts, a daily Scrum call reviews progress, task planning and, roadblocks, if any.
4. Each task is linked to a Bitbucket as a Feature Branch to check-in the developed code, and to Confluence as a document reference.
5. The Sprint has requirements discussion, design discussion, code review, unit testing, testing and release.
6. For each task, there is a design review, documentation completed, and task tested. Updates are posted on the Confluence.
7. Version control is managed in Bitbucket that has a static test, a unit test, and an automated build system built-in.

8. Once the code is reviewed and tested, it is pushed to the Development Branch and the build pipeline.
9. From the Development Branch, the tester pulls the code and tests it. When the test passes, it is pushed to the Master Branch. Here an automated build process is executed and a report created.
10. Each version of the software has a number of EPICs and Stories. Posted on the Scrum Board, these present a clear view of the task statuses.
11. At the end of each Sprint, time is spent by the team to discuss what went well, what failed, and what improvements can be made in subsequent Sprints.
12. The end of Sprint activity also includes report analyses, analysis of Sprint Velocity and Release burn down, average delivery, and estimated vs actuals of the deliveries.