**Product Lifecycle Management**

**Introduction:**

A process that is suitable to carry out a software project usually consists of several phases going far beyond the implementation of the software. These phases include, among others, project management, requirements analysis and management, quality assurance, and of course the technical phases such as implementation, testing, and maintenance. A holistic development process should address all these relevant phases of the software lifecycle, which we also refer to as project disciplines. For example, consider a process based on Scrum with its practices (such as daily standup meetings, retrospectives, backlog management, and an onsite-customer) as a project management framework (also partially covering the requirements engineering). This process is extended with some eXtreme programming practices such as pair programming (for quality management), test-driven development to structure the implementation and testing, as well as CI/CD for the integration. Despite the number of methods and practices combined in this process, several further relevant project disciplines such as maintenance, configuration management, and risk management are still not addressed.

SDLC is a step by step procedure or systematic approach to develop software and it is followed within a software organization. It consists of various phases which describe how to design, develop, enhance and maintain particular software.

It consists of various phases like requirement, feasibility study, design, coding, testing, installation and maintenance.

**Phase 1: Requirement collection and analysis:**

In this phase mainly focus on gathering the business needs from the customer. Business Analyst collects the requirement from the customer and prepares the BRS (Business requirement Specification) which has the requirement in the business form. Then a group (BA, Project managers and customers) of people sits together and determines the requirements like; what should be input data to the system?  Who is going to use the system? What should be output data by the system?  These questions are getting answered during this phase. After this, a Requirement Specification document is created which gives the guideline for the upcoming phase of the model.

**Phase 2: Feasibility study:**

Once the BRS document is completed, a set of people like Human Resource department, Finance department, Business analyst, Architect and Project manager are sitting together and analyze if the project is do able or not. This decision is taken based on the cost, time, resources and etc.

**Phase 3: Design:**

In this phase system design specification is prepared from the requirement document once the project is feasible, this design specification give input for the next phase of the model. Design is a blue print of the application and it helps in specifying hardware and requirements of the system and helps in defining architecture of the system.

**Phase 4: Coding:**

Once the system design document is ready, in this phase developer’s starts writing the code using any programming language i.e., they start developing the software. Generally task is divided in units or modules and assigned to the developers and this coding phase is the longest phase of SDLC.

**Phase 5: Testing:**

Once the software is ready and is deployed in the testing environment, test engineers starts testing, if the functionality of an application is working according to requirement or not. During this phase test engineers may encounter some bugs/defects which need to be sent to developers, the developers fix the bug and sent back to test engineers for testing. This process continuous until the software is bug free/stable/working according to the requirement.

**Phase 6: Installation/Deployment:**

Once the product developed, tested and works according to the requirement it is installed / deployed at customer place for their use.

**Phase 7: Maintenance:**

When the customers start using the software, they may face some issues and needs to be solved from time to time means need to fix those issue, tested and handed over back to the customer as soon as possible, which is done in the maintenance phase.

**SDLC Models:**

The models of SDLC are the methodologies that are selected for the software development is depending on the project’s aims and goals. These models are mainly used to develop software, based on the requirement, cost, customer and time, decide which model to be followed to develop software. Each model follows sequential steps of its own type, to develop high quality software.

The types of SDLC models are:

* Waterfall Model
* Spiral Model
* Prototype Model
* V-Model
* Iterative model
* Agile Model

The combination of different (agile or plan-driven) methods and practices a so-called hybrid development method as in the example described before is state of the practice with three out of four companies using such a hybrid method. However, it is not yet completely clear, how suitable methods and practices can be identified to form such a hybrid method. According to the large-scale international HELENA study, 30% of the projects select the methods and practices on demand. Other processes are based on defined rules (17%), tailored by the project manager (16%), or according to customer demands (14%). Almost 80% of these development processes evolve over time based on learning from past projects. In addition, when devising such a hybrid method, several context factors have to be considered, including situational factors and tailoring criteria. Research provides first attempts to support the creation of a (hybrid) development method. “typical” hybrid method consisting of a subset of the methods Scrum, Iterative Development, Kanban, DevOps, the Classic Waterfall Process, eXtreme Programming, and Lean Software Development. These methods are extended by a variety of practices. However, it appears that even a selection of a huge amount of methods and practices does not adequately cover the whole software lifecycle. Nevertheless, a software process needs to address all the relevant phases to reduce the risk of project failure (for example due to a missing risk management). That is, if applicable for the respective project, a hybrid method needs to contain methods and/or practices for the different project disciplines, i.e., for the requirements analysis, the implementation, testing, maintenance, etc.

we aim to investigate whether development processes used in practice cover the whole software lifecycle. In particular, we analyze which methods and practices are used in which phase of a project. For example, we investigate whether Scrum is used for the project management only, or also for other disciplines. This way, we can uncover phases of the software lifecycle that can be hardly covered by using standard methods and practices only (and, hence, require more attention when devising a development process), and we can point to the (mis-)application of methods and practices throughout the process.

Based on a pilot study with 27 practitioners reporting on in total 1,257 applications of methods and practices in different project disciplines, we analyze which methods and practices are used in which phase of the software lifecycle. We report on the use of Scrum, DevOps, and Kanban as top-3 methods, as well as on the use of Code Reviews, Daily Stand-up Meetings, and Coding Standards as top-3 practices. Our initial results show that these methods and practices are used for almost all project disciplines covered by our pilot study. Furthermore, looking at project disciplines mentioned more than 100 times in our study, we observe a wide variety in the methods and practices used to support Implementation and Coding, Project Management, Integration and Testing, and Quality Management.

**Literature Review:**

**Ducalis.io** is a fast and simple tool for collaborative prioritization. It’s a decision-making mechanism for the entire company. With Ducalis you can:

* Evaluate and prioritize product features, blog posts, hypotheses, bugs, marketing tactics, and anything essential for your company.
* Use any prioritization framework like RICE, ICE, Kano Model—or create your own.
* Save dozens of hours on status sync meetings.
* Eliminate unnecessary work.
* Involve the whole team in the decision-making process and gather different views.
* Understand what topics the team is aligned on and what topics need more clarity.
* Create business-critical, custom evaluation criteria, or use our criteria templates.
* Set weekly reminders to make prioritization your team’s main habit.

Import Asana projects quickly and easily to get:

* A bird’s-eye view of your team’s backlog
* Real-time task syncs
* Immediate context for all of your work
* Business-critical criteria description at hand
* Team-wide asynchronous task prioritization
* Hotkeys for even faster evaluation

**Jira** is an application lifecycle management (ALM) tool from Atlassian that provides different packages to suit various customer needs. It is used for bug tracking, issue tracking and project management.

Jira is one of the company's main offerings along with Confluence, the company's team collaboration platform. Atlassian is an Australian enterprise software company that specializes in developing products for software developers, project managers and content management. Jira and Confluence are the company's two most popular products.

Jira is a flexible issue tracking tool that helps teams plan, manage and report on their work. Users of the tool will vary based on which package is being employed. For example, the Jira Software package is often used by software developers and engineers, Jira Core is popular among business and marketing teams and Jira Service Desk can be used by employees and customers. The flexible and scalable configuration of Jira also allows it to support a wide variety of industries, including healthcare, retail, manufacturing and information technology (IT).

**Jira packages**

The Jira packages facilitate work throughout the enterprise, especially in Agile environments. Atlassian offers three Jira packages to users: Jira Core, Jira Software and Jira Service Desk.

Jira Core is primarily used for business management. As a result, it is most frequently employed by business, marketing, finance, human resources (HR), legal and operations teams. Uses of Jira Core include:

* task management;
* general business project tracking and process management through workflows;
* project and task progress reports through a Dashboard; and
* overall task tracking and reporting.

Jira Software is mostly used for project and issue tracking. Users of this package include technical managers, software developers and engineers, product owners, DevOps teams, Scrum Masters and testing and quality assurance teams. The uses of Jira Software include:

* Software development
* Agile reporting
* Virtual Scrum and Kanban boards
* Dashboard reports on issues, sprints and release progress
* Issue assignment
* Backlog planning

Jira Service Desk is used for customer service and as an IT service desk. The primary users of this package are called Helpdesk Agents -- the people who work on tickets and communicate with the customers. Within Jira Service Desk, agents can access the agent interface and customer portal; add, delete and edit comments on issues, both customer-facing and private; control content within a connected knowledge base; and access queues, service-level agreement (SLA) goals, reports and customer lists. Other users of Jira Service Desk include Helpdesk supervisors, self-help users, employees and customers. Overall uses of the package include:

* Incident and problem management
* Tracking metrics such as SLA
* Helpdesk management
* A customer self-service portal that is managed with the knowledge base
* Change management

A fourth offering, Jira Align, focuses specifically on Agile software development. This service allows users to connect their business strategy to the technical execution process.

Jira Align is frequently used by project managers and release train engineers (RTEs), portfolio managers, transformation teams, product managers, executives and delivery teams. Uses include:

* the optimization of customer value by connecting strategic investments with customer value to reliably accelerate outcomes;
* collection of team-level data to provide real-time visibility across the entire enterprise; and
* the alignment of each team to a single strategy in order to determine the scope of the project, dependencies of each team and to create a roadmap.

**Jira used**

Jira is most commonly used for application lifecycle management, test management and project management. Specific tasks frequently include bug and issue tracking as well as the creation of user stories. A user story is a tool used in Agile software development to capture a description of a software feature from an end-user perspective. This creates a requirements management tool that can then be used to generate tasks.

In an Agile environment, user stories are often handwritten or printed on sticky notes or index cards and attached to a Scrum board. Jira enables its users to digitally record user stories and prioritize them, often using due dates or the MoSCoW criteria -- standing for the breakdown of must, should, could and would like to have. From here, the user story can be assigned to team members, given an estimated timeframe, tagged to a specific component level feature and assigned to a sprint that is required for the implementation of the story.

Despite Jira's initial focus on issue and project tracking, it is also becoming popular among teams responsible for test case management. This presents the advantage of allowing testing and development to remain in one system. QTest was developed as a Jira test management tool that is integrated at both the requirements and defects levels. The tool allows users to work faster and plan, track and test their work more efficiently. The QTest tool easily collaborates with custom issue types within Jira -- such as user stories and tasks -- to provide coverage for test cases. Users of QTest can also find real-time reports of their test run execution history and Jira issue coverage. Furthermore, QTest also lets users map test projects to various Jira projects, further increasing efficiency.

Teams using Jira can be assigned multiple existing workflows. For new projects or user stories, teams create new workflows by assembling the project's goals and established practices. The software helps underpin the tasks of each team by gathering user inputs, which drive the changes for new versions, facilitating the fast and regular release of new versions.

Jira incorporates an open REST API which allows users to integrate the tools into a wide range of systems. Users can choose to either create their own connector between Jira and the system or use one of the connectors that is provided by the Atlassian Marketplace -- a platform where Atlassian customers can distribute and sell their created apps to other Atlassian product users. Jira is commonly integrated with Salesforce, GitHub and GitLab, Microsoft Team Foundation Server and Visual Studio Team Services, Perforce and ServiceNow.

Furthermore, the Atlassian suite of services can be integrated across each product. This allows data to pass from one Atlassian application to another. For example, issues can be created in Jira using Confluence and Confluence can be used to view Jira reports and dashboards.

**Benefits**

Benefits of Jira include:

* Cost-effective -- A Jira license is frequently half the cost of similar services obtained from other competitive enterprise software companies and applications. The ability to easily support and maintain applications also makes customization and upgrade processes faster and simpler, consequently reducing the total cost of ownership.
* Interoperability -- Jira provides a large REST API library, over 2,000 add-on options and a plugin software development kit that enables companies to more efficiently develop, integrate and extend the use of Jira to create an enterprise-level platform.
* User-friendly -- Companies and users prefer Jira because it provides an intuitive and uncomplicated interface that is easy to use. Furthermore, the support provided through community forums and open-source documentation allows users to feasibly learn and adapt the tool to their needs.
* Easy deployment -- Jira is deployed as a web application on Windows or Linux; it is designed to have a small footprint.
* Scalable -- Jira can be used by small, medium or large organizations across a variety of teams and for basic or complex solutions.
* Customizable -- The ability to customize the products to best fit an organization's work makes Jira beneficial for a large range of companies and projects.

Furthermore, Jira supports the major databases -- including MySQL, Microsoft SQL Server and Oracle -- as well as standard web browsers -- such as Google Chrome, Internet Explorer and Firefox.

**Licensing**

If an organization is considering licensing Jira, they have the option to try the services for free for seven days without any team size limitations. If the company decides to buy Jira, they can either license it as a cloud service with monthly payments or as a self-managed, on-premises service with one flat payment. Prices for on-premises services vary based on the number of team members using Jira and whether the company is buying it for a server or data center.

**Pricing for the three packages when used through the cloud are listed below:**

**Jira Core options and prices:**

* Free -- Always free with support for up to 10 users. It includes 2 GB of file storage, and users have access to support from the Jira community. Users do not have anonymous access, advanced permissions or access to audit logs.
* Standard -- The starting price is $5 per user, per month with support for up to 5,000 users. It includes 250 GB of file storage, and users have access to standard support during the workday.

**Jira Software options and prices:**

* Free -- Always free with support for up to 10 users. It includes 2 GB of file storage, and users have access to support from the Jira community. Users do not have anonymous access, advanced permissions or access to audit logs.
* Standard -- Prices start at $7 per user, per month with support for up to 5,000 users. It includes 250 GB of file storage, and users have access to standard support during the workday.
* Premiums -- The starting price is $14 per user, per month with support for up to 5,000 users. Unlimited storage is provided, and users have access to 24/7 premium support.

**Jira Service Desk options and prices:**

* Free -- Always free with support for up to 3 agents and no limit on the number of customers who can access the service. Agents are not provided with audit logs. It includes 2 GB of file storage, and users have access to support from the Jira community.
* Standard -- The starting price is $20 per user, per month with support for up to 5,000 agents and unlimited customers. It includes 250 GB of file storage as well as access to standard support during the workday.
* Premium -- Prices start at $40 per user, per month with support for up to 5,000 agents and unlimited customers. Users are provided with unlimited storage and 24/7 premium support.

**Problem statement:**

When we work on a big project with team member so its very difficult to manage a project, we face following problem like:

**1. Integration Issues**

It can be challenging to integrate what you are making with tools your audience may already be using. You need to find ways to make your product compatible using application program interfaces (APIs) or partnering with these other tech brands on ways to work together.

**2. Communication Breakdowns**

Software development requires a team collaborating to address consumer needs. Every member has to know a project’s coding strategy, objective and goals. Otherwise, the fallout affects the manager’s reputation and the team’s output. A good manager will make sure each team member knows a project’s multiple facets and immediately educate new hires. This will increase synergy and productivity

**3. Unrealistic or Mismanaged Timelines**

One of the most common issues in project management is the infamous timeline. Being realistic in setting timelines with projects is critical. One of the best ways to avoid delay in timelines is to brainstorm at the beginning of the project. Ask the team, “What will break during this process?” When a bump in the timeline does occur, gather everyone and find the best way forward and around it.

**4. Feature Overload**

The largest obstacle that I’ve observed when it comes to launching a new software package is the desire to put too much into one application. To overcome this, be more specialized rather than more general look for sub-niches. Remember, less is more when we are trying to solve a problem.

**5. Lack of Alignment Between Sponsor and User Needs**

Your executive sponsor’s opinion about what should be built must be secondary to what the end user needs. Let’s face it, the chances that your boss is a Steve Jobs visionary are pretty slim, despite what they might think of themselves. Educate them on the importance of letting the market tell you what features to build.

**6. Underestimating the Task at Hand**

Developers know that there can always be bumps on the road to deployment. Save yourself some time and hardship by scheduling in some extra cushion time in case it is needed. In the worst-case scenario, that space is used to conduct tests on the project or take some much-needed time off. Many time developers underestimate the task at hand, and an extra cushion can alleviate pressure and anxiety.

**7. Not Pinpointing the Real ‘Why’**

The most common problem is building the wrong thing. Businesses need to understand not just what a customer is asking for, but “why” they’re asking for this specifically. The driver of a horse and cart may ask for a better whip to go faster; in reality, he needs a vehicle, but he doesn’t know it. Businesses need to determine the “why” to provide the most effective software for their customers.

**8. Underestimating the Importance of Quality Assurance**

To guarantee customer satisfaction, the importance of QA should never be underestimated. To obtain high-performing and secure products, code quality has to be reviewed and tested continuously throughout the development stages. Outsourcing companies can certainly alleviate this process, providing expert testers who can lead an integrated approach to maximize the quality of your project.

**9. Feature Creep**

Feature creep the excessive expansion of new features is a common obstacle I’ve seen software developers confront. The most effective way to combat this issue is to consistently and thoroughly conduct customer development interviews to validate each feature and make sure it meaningfully targets the issue you set out to solve. Continue to conduct these interviews even after a successful launch.

**10. Security-Related Release Delays**

Developers are often focused on getting good, working code (software) out the door. Often times, security is an afterthought that can really delay your ability to release code. It behooves developers and product managers to have security built into the development lifecycle and built into their go-to-market strategies, including any security certifications and/or other industry requirements.

**11. Not Defining A Target Audience**

Your new software might be cool, but not everyone will think it is. If you don’t define a target audience for your software, then you’ll waste a lot of time and money marketing it to consumers who have no interest or use for it at all. So, it’s important to define your target audience before launching your new software. Do thorough audience and market research to find the best audience.

**12. Underestimating the Demand**

One of the unforeseen problems or obstacles to releasing new software or services is being unprepared for a highly successful launch. Often companies underestimate the volume of traffic their infrastructure may experience with the launch of a new product. Building systems that can quickly scale on demand is critical in avoiding this obstacle and ensuring your site does not crash on release.

**Problem solution:**

Our platform is an agile project management tool that supports any agile methodology, be it scrum, Kanban, or your own unique flavor. From agile boards, backlogs, roadmaps, reports, to integrations and add-ons you can plan, track, and manage all your agile software development projects from a single tool. Pick a framework to see how Our platform can help your team release higher quality software.

**Agile tools for scrum**

Scrum is an agile methodology where products are built in a series of fixed-length iterations. There are four pillars that bring structure to this framework: sprint planning, stand ups (also called daily scrums), sprints, and retrospectives. Out-of-the-box, our platform comes with a comprehensive set of agile tools that help your scrum team perform these events with ease.

**Tools for sprint planning**

Sprint planning meetings determine what the team should complete in the coming sprint from the backlog, or list of work to be done. Our platform makes your backlog the center of your sprint planning meeting, so you can estimate stories, adjust sprint scope, check velocity, and re-prioritize issues in real-time. There are several tools within Our platform scrum template that can help your sprint planning run smoothly.

**Version management**

Track versions, features, and progress at a glance. Click into a version to see the complete status, including the issues, development data, and potential problems.

**Easy backlog grooming**

Easily re-prioritize your user stories and bugs. Select one or more issues, then drag and drop to reorder them in your backlog. Create quick filters to surface issues with important attributes.

**Sprint planning**

Make your backlog the center of your sprint planning meeting. Estimate stories, adjust sprint scope, check velocity, and re-prioritize issues in real-time with the rest of the team.

**Story points**

Estimate, track, and report on story points to help your team become more accurate in future sprints. Use story points, ideal hours, or your own method of estimating.

**Scrum board**

Scrum boards are used to visualize all the work in a given sprint. Our platform scrum boards can be customized to fit your team's unique workflow. You can also easily add things like swimlanes to separate epics, assignees, projects, and more. At the end of the sprint, get a quick snapshot of all the issues that were completed and any unfinished issues will automatically move into the backlog to be addressed in the next sprint planning meeting.

**Track and manage your sprints**

A sprint is a fixed-length iteration of work, typically one or two weeks in length. Work should be fully scoped and prioritized during the sprint planning meeting, so the team can get started as soon as the sprint begins. These features keep track of all the work being tackled in each sprint, so your team can stay focused on the tasks at hand.

**Sprint permissions**

Use sprint permissions to define what users can see or do in a given project. Sprint permissions allow you to better delegate permissions and responsibilities to team members, allowing the entire team to be more autonomous and agile.

**Custom issue types**

Develop custom issue types for bugs, stories, coffee runs, and everything in between. Configure a custom workflow and custom fields for each issue type, so your team can manage work according to its requirements.

**Workflows**

Create and assemble your own custom workflow states and transitions for every issue type (bugs, stories, epics, coffee runs). Leverage the powerful our platform workflow engine to update issues automatically based on events in other systems, or define criteria that must be met before an issue can progress.

**Release hub**

Quickly see what's shipping in your next release, and what's left to do to get it out the door. Easily reconcile your source code and issues: automatic warnings are generated for issues with broken builds, missing code, or open pull request