Daimler ICertis Delivery Framework

Executive Summary

Based on our interviewee feedback and internal working sessions – the current delivery framework supports a rigid and monolith release model. There are multiple factors impacting organizational ability to increase frequency of delivery model and improve product quality. Certain agile methodology practices are in place, but efficiency is limited due to poor adoption in planning, development, testing and delivery phases. The current delivery framework is built to orchestrate around key challenges related to application architecture, build / deploy architecture, ineffective development practices, lack of an enriched infrastructure capabilities, and limited testing capabilities.

Primary focus on the delivery framework assessment was on understanding current organization roles, responsibilities, processes and standards to support major release events related to customization updates to the iCertis platform. The target of this assessment is to baseline current delivery model, identify gaps, and provide a target delivery model. As the organization is working to streamline their development practices (merger of Core and Daimler branch) - the target state will encompass this change.

Current organizational release practices are constructed to support 2 primary release event types (quarterly and hot fix), supporting 3 applications developed by three different scrum teams. Quarterly release events are the primary release vehicle – driving comprehensive application delivery from planning to production in 3 month cadence. To allow for a continuous flow, an interim engagement model (2-2-2) have been introduced to allow for iterative planning, development and product testing of features prior to release certification phase. This model provides short-term benefits in allowing for early insight; the practice masks some of the key challenges to be addressed in sprint planning, development and testing phases. Furthermore this practice may have potential negative impact on adoption of Agile Scrum / XP best practices, DevOps and foundational services practices.

There are multiple breaking points in the current major release process and as delays are encountered, weekend work is required to complete production changes. Key areas of the delivery model do not follow industry best practices, which will be highlighted in detail in following sections.

- Lack of central Release Management function to be the liaison between product, portfolio and delivery teams ensuring Agile release train practices are standardized, enforced and measured. Development teams are organized as project teams that deliver comprehensive application systems vs. product teams focused on domain expertise. Minimal to no measurement on developer productivity, delivery quality, functional testing, and integration testing. Isolated development teams limiting transparency and integration dependencies until late into the release cycle (Sprint 5) - causing rework and lead times to address integration issues.
- Applications are design in monolith architecture patterns with extensive integration testing occurring late in the release cycle (Sprint 5) does not allow for smaller and modularize releases. Because of the monolith architecture; activities related to branching, developer workspace management, artifact versioning, build optimization, deployment architecture and test management are inadequately designed with long execution periods and ineffective practices. We were unable to get a deep dive into the application architecture as client was not comfortable providing more in depth information due to lack of non disclosure agreement in place.
- Quality engineering and management practices for both development and release related activities are lacking proper governance, planning, automation and execution. Because of the minimal test engineering capabilities, other elements of the delivery process have been altered to drive efficiency. The lack of these capabilities lead to inconsistent, unrepeatable and incomplete quality practices and potential unstable product delivery capability.
- Environment strategy and practices do not incorporate best practices related to infrastructure as code, configuration management and automated deployment strategies. Limited automated and repeatable infrastructure capabilities impact teams ability to support frequent deployments to QA and Product testing environments. Current practice of 2 deployments a week - introduce long lead times, potential rework for developers, and unstable environment practices.

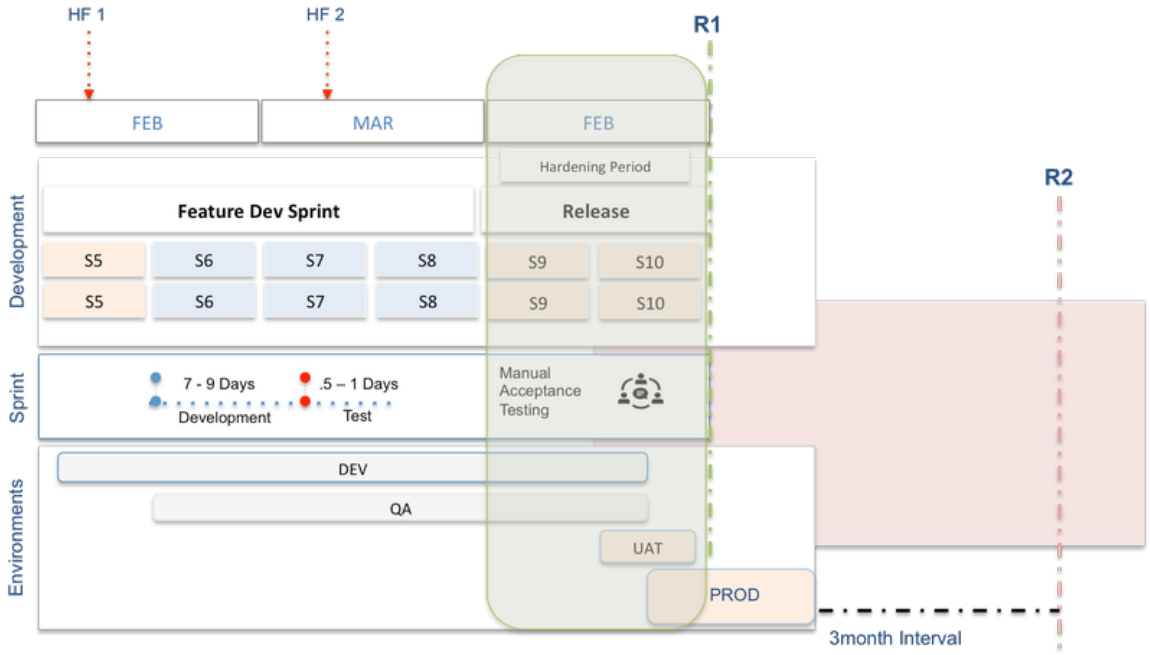
Release Type: Quarterly

- Major release cycles planned for 4 times a year (Quarterly)
- Release events are incorporated into the development delivery model, utilizing every 6th sprint to complete certification activities and deployment to production
- Limited to no governance and traceability of planned release features vs. delivered code packages
- Lack of centralized Release Management function to define, execute and improve delivery practice capabilities.

Current State

Primary focus on the delivery framework assessment was on understanding current organization roles, responsibilities, processes and standards events. Major release events are the primary release vehicle – driving comprehensive application delivery from planning to production. The relea six sprints, where the last two sprints are utilized for the release hardening and certification activities. In most cases the first sprint is utilized for a bug fixes, and addressing technical debt from the previous release. In some cases, developer activities in sprint 6 are split between bug fixes an development for sprint 7. This was not clearly articulated, as the current delivery model practice do not adequately support this capability. All rel activities occur in sprint 10 - where business acceptance testing is completed in a controlled UAT environment. Production deployment activities completed over the weekend - duration of 16 - 48 hours required for successful production update.

The diagram below depicts the comprehensive view of the release framework adopted for major release events.

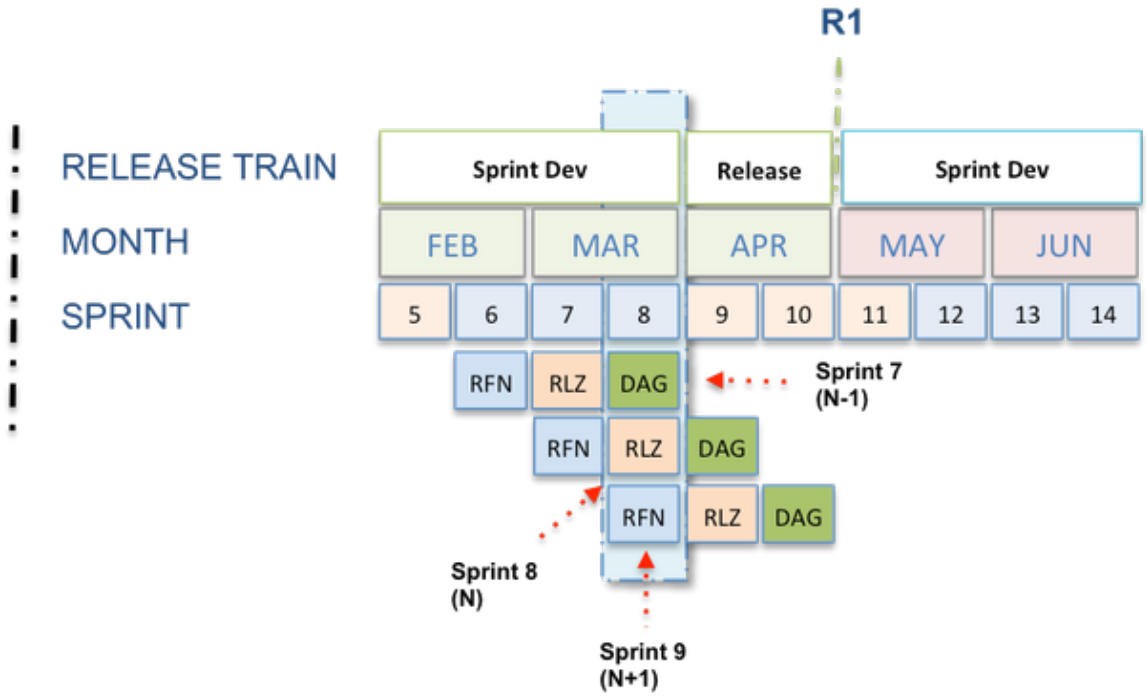


The current delivery model activities consists of 3 major phases –

- Refinement – For every quarterly release there is a 2 month period prior to start of the first release sprint - where the product organization cr backlog stories for engineering teams to review and provide high level estimates. Users stories that are ready for development are identified targeted release cycle. The planned release content (reviewed, estimated, agreed) is provided to 3 development scrum teams **(name the 3** realization. Most user stories are sized as 20 points, with the average developer delivery capacity per sprint is 13 story points. During the de continuous planning cycle is adopted to allow for new changes to be introduced, and other stories to be refined before the next sprint start.
- Realization - Implementation phase where development teams complete sprint planning, development of user stories and quality / business t deliverables. Two week sprint cadence is followed - where day 1 of sprint is utilized for sprint planning, and 7-9 business days to complete th stories. Developer teams are driven by deliverable date (story to be completed by end of sprint), and complete on development activities on Developer working on multiple tasks leverage TFS workspace and isolate their working files via change set. Developers completed activities trunk branch, with no code analysis, unit test and security testing gates. Code reviews are completed by dev leads but no automated and go present. In some cases quality assurance team members will validate the user story on the developer workstation before the change is com There are 4 code drop windows (Thursday, Tuesday) to a QA environment to allow for formal quality acceptance validation. In most cases th fully utilized - as large user stories will not be delivered until 7-9 business days. Most sprint related user stories are delivered to QA utilizing t day of sprint) - requiring quality assurance testing for sprint 5 to start at the beginning of sprint 6. It is common practice for business and qua happen in parallel on the QA environment - impacting the availability of the QA environment for sprint 6 activities.
- Release Hardening / Certification – Development scrum teams work in isolated on their application and are integrated as part of the hardeni sprint is utilized by scrum teams to identify and address functional, integration and regression issues on the QA environment (challenge if bu completing feature testing for Sprint 8). Once a good quality baseline has been met (which is not clearly defined), product testing will begin i first time product teams are able to access a fully integrated product planned for the target release and complete their acceptance testing.

Planning & Refinement Management

There is a continuous delivery engagement model created to support a more agile approach to planning, execution and delivery of the product in model allows for 3 sprint activities to be driven in parallel, where sprint N+1 goes thru story refinement, sprint N is in construction or realization ph acceptance testing is completed for sprint N-1. The diagram below illustrates the desired practice -



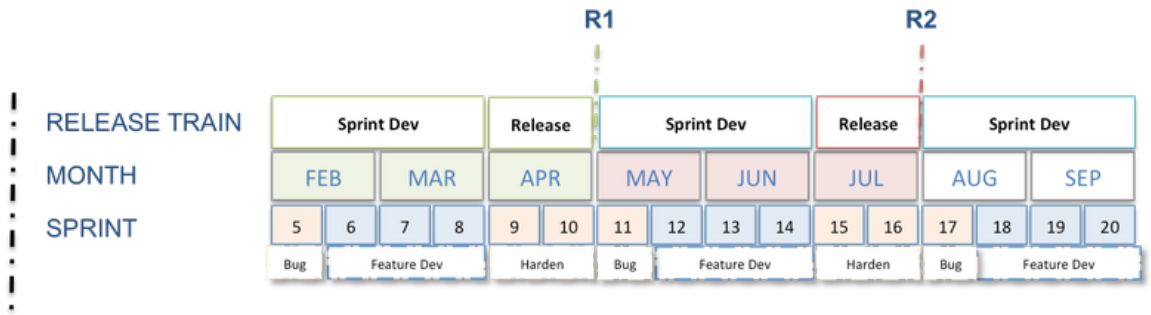
This model works well when there is a consistent delivery model (all sprints are weighed and executed in same manner) - however it does introdu when implementing in current model. By definition Sprint 5,9,10 should be development sprints but they are internally being utilized for bug fixes, activities. In this case, the refinement done for sprint 9 would not be realized until potential sprint 12 (introducing a 6 week gap). Furthermore spr would overlap with product hardening / integration testing cycles. Given the current environment strategy - there is a high risk the same environm for sprint 8 acceptance testing along with Spring 9 core activities related to integration testing. This practice also leads to inconsistent velocity an in refinement, realization and acceptance testing. Overlap of realization activities for Sprint 9 and hardening of R1 release can cause developer a (depending on quality of S5 - S8) as we are not taking into account integration issues that might arise during S9.

Product Development / Realization

Product development activities are performed utilizing the 2 week sprint agile model – becoming fully integrated 4 weeks prior to the release date a standard scrum cadence, however internal practices associated to the activities related to sprint planning, execution and completion are not cor stories are sized using Fibonacci sequence - elements considered in assigning a story point include the complexity of the story, the number of un potential effort required to implement it. It is not clear the development scrum teams is considering these factors in their story sizing exercise. In team story board a majority if the stories are sized at 20 story points - which are usually recommended to be broken down into smaller stories. O submitted late into development phase, which impact delivery dates, which then subsequently extend into business acceptance testing cycles. Tl change control process introduced to manage these changes ensuring clear understanding and agreement of impact to sprint deliverables.

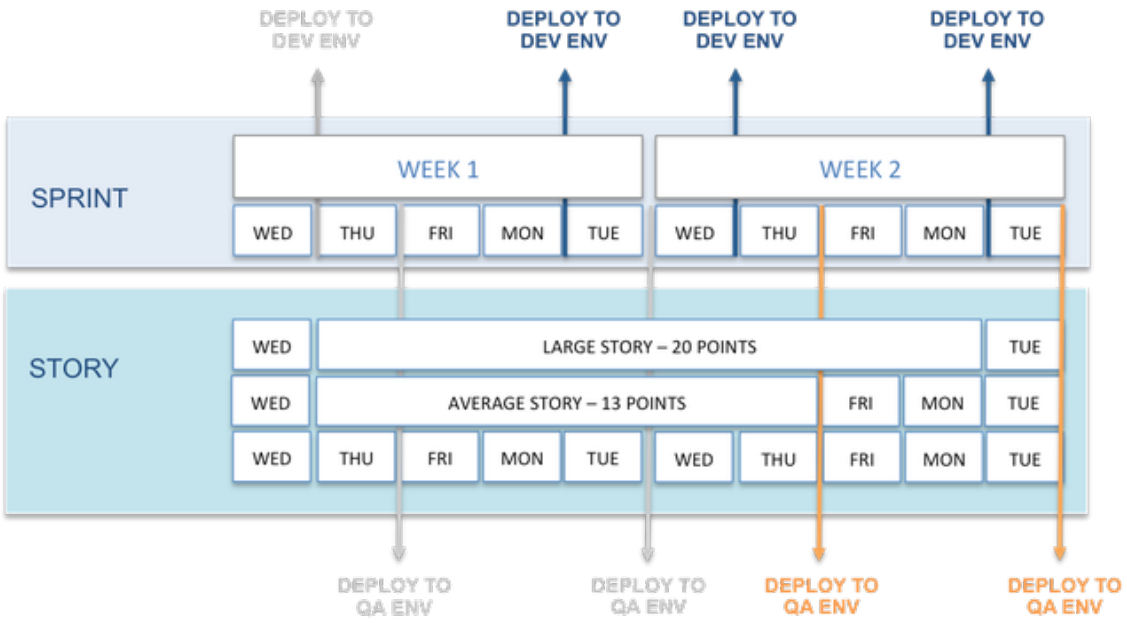
Development scrum teams are organized as project teams (focusing on complete product ecosystem) - which does not allow for concentrated an knowledge upscaling. Furthermore developers working on a monolith application might not be aware of other competing changes that can cause functional incompatibility issues. Unconventional practices are identified in development phase, related to source control systems, branching mod versioning, build optimization and packaging models. There is no clear evidence of unit testing, functional testing and code coverage during the d

The diagram below depicts the sprint sequence associated to the delivery framework adopted. A release cycle consists of 6 sprints, but 3 of sprir for bug fixing, integration testing and release readiness activities.



- Story at a minimum transitions over 3 sprints [Refinement, Realization, Daimler Testing]
- Current model only supports quarterly releases
- Last 2 sprints dedicated for integration hardening and release readiness
- First sprint of release is used to fix bugs from previous release. Start of feature based development usually starts in second sprint of release.
- Developer workload is not dedicated to just development activities because a percentage of time is required for refinement activities each sp
- On average, a Story's Refinement phase spans more than 1 sprint which delays start of Realization phase

Sprint activities related to build and deployment of completed user stories align to existing environment strategy (Dev, Qa) adhering to a fixed upc
Development team members work locally (workstation/laptop) and update their changes incrementally to the shared DEV environment. This proc
files (dll, config) from their local workspace to the target Dev environment. There is a planned QA drop on Tuesday / Thursday to support sprint c
Due to the large story size, most user stories are delivered to the QA environment on the 4th available drop window. This leads to sprint test acti
sprint activities.



- Actual development starts on the second day of the sprint (Thursday) due to sprint planning with Daimler
- Average story is 13 pts which means dev complete occurs late in the sprint and first opportunity for testing occurs late in the sprint when it is Environment.
- First opportunity for testing a large story (20 pts) in QA is at the beginning of next sprint since deploy is at the end of the sprint
- Inconsistent DEV environment due to developers overriding DLLs directly with versions compiled on their local laptop environment
- Inability to increase QA deployment frequency due to it taking at least 5 hours for deployment (P2P, code deploy, and manual sanity test)
- Manual sanity test takes 1.5 hours

Release Hardening / Management

The current release train model embeds release activities into specific sprint sequence - requiring the full scrum team to support release certificat
challenging approach which will lead to inconsistent resource allocation and reduce ability to accelerate N release activities in parallel with N+1 R
activities. The release phase begins with an integration phase in Sprint 9 where all 3 scrum teams will deploy their feature changes to an integrat
and complete their technical integration testing. This practice is not recommended as it introduces high risk to release related to code conflicts, fu
incompatibility, and environment instability. Release activities are date driven, requiring development teams to work weekends and late nights to
issues. Furthermore, the testing activities related to the release are manual driven and require heavy involvement of product management team t
feature testing and regression tests. It has been identified in many cases that regression bugs go unidentified due to inconsistent test practices a

Core release activities related to integration testing, performance testing and penetration testing occurs on the same environment (QA). This is m
team - but is not recommended practice as it may cause environment stability issues. As performance testing and penetration testing require sys
configuration updates, it may lead to false positive test cases or unreliable environments.

It is not clearly present if a centralized release management function existed to consistently govern and manage the end 2 end release train activi discussions around release management activities were discussed, artifacts validating key release practices are followed were not disclosed. En related to managing release train planning, release registration, infrastructure changes, platform updates, configuration management, database c penetration testing, certification quality indexes, environment strategy, environment readiness, release readiness, and production roll out artifacts review.

Finding Summary

Item	Finding	Summary
1	Lack of release management involvement in managing release train events. This may lead to inconsistencies in what features are planned / delivered in a release; with no central authority to conduct the release certification activities. Often causing uncertainty and delays in development and test activities late into the release cycle.	Introduce centralized release management function responsible for planning, governance, and execution of multiple release vehicle types. Release Management function should define the entry criteria, release train types (infrastructure, config, incremental, full) with associated quality, security and standard certification phase activities. Furthermore – they would be responsible for driving organization delivery of the release train to production.
2	Demand Management Tool (Jira) lacks traceability and transparency from actual work being delivered (TFS Work Item) to Release Train content. Lack of integration between the two tools, requires manual status updates to Jira tickets.	To allow for a continuous workflow – create integration point between TFS Work Item workflows and corresponding Jira Story workflows. This can be setup utilizing JIRA API's and a service account to update JIRA story stages as work items are transitioned based on internal development practices.
3	Story points allocated to a given developer are 20 a sprint; which does not follow Scrum guidelines. The level of change introduced requires 8-business days to develop the solution. Not allowing time for testing and bug fixing before completion of sprint.	Agile practice allow for a story to be small enough that it can be coded and tested in 1 iteration. If this cannot be achieved – stories should be split into smaller stories – separating multiple acceptance criteria to each single story having only a single acceptance criterion.
4	Not well-defined product backlog – requires last minute change requests into realization phase. Stories delivered don't meet the done criteria – introducing bugs and rework.	The product backlog should be a very active list of user gathered, where the product owner is responsible for making sure the content of the user stories are well defined in detail level. All user stories in product backlog should be enough in sizing to fit in one sprint. Build standard practice to incorporate use case scenarios, acceptance criteria should be provided. If product backlog story is not ready – scrum team to communicate back with product owner and work on user stories that are ready for development.
5	2-2-2 model does not adhere to Scrum principles – in most cases sprint deliverables are not tested before requiring product testing. Business testing occurs on a shared QA environment – which impact integrity of environment and limits efficiency of next sprint deliverables.	Sprint planning, development, testing, and product acceptance activities need to be completed in one iteration. Development, testing and product testing activities to be managed in separate environments ensuring the integrity of the environment readiness.
6	Limited to no quality testing during sprint – require spill over and reactive testing activities to catch up on sprint end date.	Adhere to test driven development – where story automated test cases are built as the first step (where all cases will fail) that defines the desired improvement or new feature, then produce the minimum amount of code to pass that test. This will help drive for higher quality products, a faster delivery cycle, and risk free development process.
7	Lack of integration testing until sprint 6 – leads to delays and potential rework, which can be identified before if proper CI/CD practices are enforced.	Early integration of different application components allow for higher quality product delivery. Implement a continuous integration / delivery model to support integration automated tests at sprint, weekly, daily and on-checkin. Improve frequency of CI/CD – based on organization and tool maturity.
8	Testing activities during Sprint 6 require manual testing from product teams – leading to poor practices and standards. In some cases, comprehensive testing cycles are limited to focus areas (new feature changes) which might allows for regressions bugs to go unidentified.	Sprint 6 business testing should be focused primarily on business acceptance / product testing – once all other quality criteria has been met. This will require incorporating CI/CD for new development features, and a fully automated regression suite, performance testing, penetration testing executed with results analyzed and a Silver build criteria met before business team involvement.

9	Incur technical debt – as releases are date driven and quality issues are not identified or accepted to allow for release train to complete.	Technical debt is a concept in software development – where design and quality is compromised to meet deliverable dates. There are three areas to focus on to eliminate this risk; application development, integration points and release certification. To avoid technical debt – recommendation is to incorporate a shift-left model to development practices, early and frequent integration testing, and a fully automated comprehensive regression with analysis.
10	Build / Deploy to QA environment is scheduled for 2 times a week - introducing potential 24-48 hours lead time before user story can be validated. Features / User stories delivered by development teams require long lead time and resource overhead to deliver for test activities. Practices related to code analysis, static analysis and functional testing activities are delayed or non existent.	With the adoption of behavioral driven development and automated continuous integration and delivery capabilities, product quality gates will be enforced consistently and reduce rework and resource overhead. Need to invest in incorporating automated test activities in sprint cycles allowing individual components to be validated on demand.

Target State

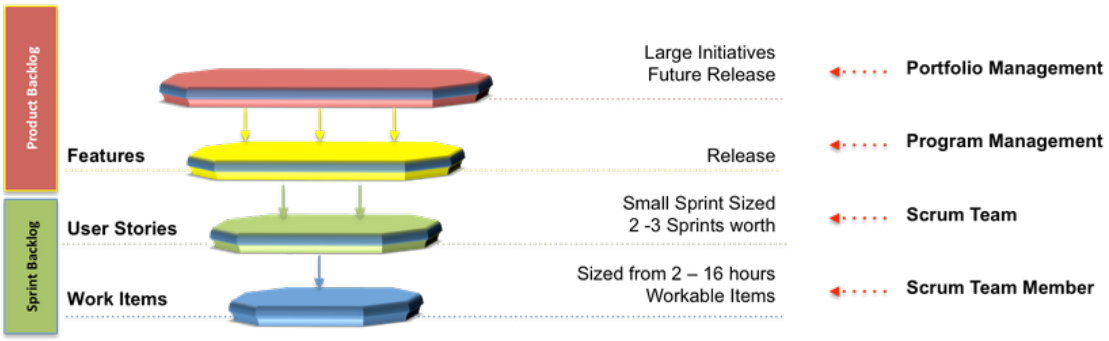
Target delivery framework is constructed on providing multiple delivery vehicles based on complexity of application change and maturity of engine on driving a culture of continuous planning, integration, deployment, and monitoring. The recommended approach to achieving this continuous flo delivery practices is adopting best practices related to Scrum and Lean principles. Through agile iterative development, we would break down lar activities and introduce shift left methodologies related to quality and security practices. With the implementation of lean manufacturing principles sub-tasks to identify the value associated with each one of them.

Planning / Demand Management / Refinement

Portfolio management considers the business goals of the larger organization and approves funding for the various initiatives in which the organiz has approved resources and funding that must be appropriately managed, the portfolio manager is facilitating the overall understanding of organi assign people and resources, and how projects will be interleaved to optimize business outcomes given a limited set of people and resources, an than the capacity to deliver. Portfolio management teams organize major initiatives and plan the delivery of features/stories in high frequency rele govern the change control with Product teams for scheduling of Epics while taking into consideration capacity and deliverable timelines. They wor stakeholders in grooming the product backlog, ensuring user stories are completed and ready for realization.

The program management function is responsible for a collection of feature changes that must be integrated over time to achieve a larger busines multiple releases of a single product and require a longer-term, more strategic perspective on the life of the project. Each release has to be self-c any subsequent releases that are in scope at the program level. Program managers are still dealing with time, cost, scope, and risk associated wi even greater concern. Program management teams work across scrum teams to confirm the sequencing of stories / toggle switching / and infras Scrum teams have the flexibility to register, execute, and deploy changes to production with Release Management oversight. Scrum of scrum sta sequencing, story completion dependency, and improvement opportunities across scrum teams.

The scrum approach is an iterative approach utilizing 2 week sprint cadence to deliver releasable products on a regular basis. The roles, respons each sprint, there is sprint planning, daily stand-up, sprint demo and sprint retrospective. There are task boards and burn-down charts to follow up receive incremental feedback.



The diagram above depicts the correlation between Product backlog and Sprint backlog, providing an outline of the different teams roles and resp model allows for product owners to decompose large initiatives into prioritized feature stories and then sprint stories so the team can execute as r teams, invalid stories, and unhappy developers and stakeholders.

Development / Sprint Model / Realization

Development teams are organized into scrum teams (7-9 resources) focusing on domain specific ownership. All delivery teams adhere to a stand management leads work with their scrum teams to ensure planned deliverables for a sprint are completed and meet the done criteria. All prioritiza deliverables across scrum teams are to be managed by Program Management lead. Features are broken down into stories in planning phase wit dependency on other scrum team dependencies. User stories are sized using Fibonacci sequence of 1, 2, 3, 5, 8, 13, and so on. The benefit of greater than the previous one (with the obvious exception of 1 and 2, of course). This streamlines the conversation by preventing debates about v difference between a 6 and a 7 would be small enough as to not be worth the time needed to come to consensus on it. So long as the team can d 5 or an 8 (which is a more significant difference), the estimation process will result in the desired outcome.

All sprint activities related to sprint planning, development, quality testing and product acceptance must be completed within the sprint cycle.

Release Train Construction and Definition

The Agile release train provides alignment and helps manage risk by providing program level cadence and alignment. It is based on agreement a practices, which are followed by all teams who will take part in a particular release train. The release train rules should contain -

- Frequent, periodic planning and release dates for release construction, execution and delivery
- Teams follow a standard sprint iteration model
- Continuous Integration / Delivery practices and gates are implemented and enforced at multiple levels related to development, testing, and re
- Release hardening iteration should take part in short intervals (3 days) as part of release certification phase
- Infrastructure, database changes and other middleware changes may track ahead of the release

Considerations for defining and determining type of release trains must take into consideration:

- Trains should be focused on a single, primary product or feature objective
- Teams with features and components that have a high degree of interdependencies should plan and work in the same train
- Whenever possible, train teams should be collocated - simplifying planning logistics

Registration to a specific release requires development teams to execute and pass Continuous Integration / Delivery quality gates related to unit t Release trains are design to support 4 types of release events – On Demand, Green, Major, and Hot Fix. Goal of target delivery framework will en

- Release more frequently to production
- Publish higher volume of changes to production while mitigating risk and complexity
- Allow for infrastructure and operational maintenance releases

Release Types

On Demand -

The on-demand release events consist of deliverables that make up a story/feature delivered by 1 scrum team. Change is backward compatible – stories do not require performance and penetration tests. Release events may be invoked as the end of a completed sprint – with fully automated

Green Release -

Green release events empower the scrum teams to manage their changes from planning to production – change set is isolated to one feature del release events require scrum teams to meet release criteria; component isolation, automated CI/CD, infrastructure as code, automated test suites

Major Release -

Monthly release events support major release activities where multiple program teams engage to deliver a comprehensive product version to proc for registration, governance, certification activities execution, and deployment change window activities. Changes in monthly release are consider or may not be customer facing. In agreement with Product Management - every 3rd monthly release may be labeled as a market release (custom their Market Release cadence to monthly if required for business demands. Requires integration and regression testing along with Blue / Green d

Hot Fix -

Post production release event – supporting production severity 1 bug fixes, security remediation or infrastructure findings.

Agile Release Train

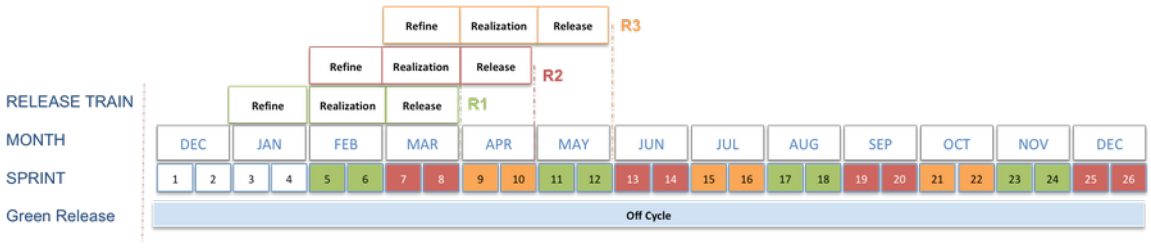
Working across geographically distributed teams the release train provides a cohesive medium for organizing deliverables (completed over X spri teams. The train concept aligns with the sprint model (can be affiliated with 1 – 2 week cycles) to enable 3 multiple parallel release trains at a time certification, N+1 is being developed in sprint cadence, and N-1 is deployed in production and properly monitored. Enabling 3 parallel release tra adopt modern DevOps practices related to continuous integration, continuous delivery, shift-left methodologies related to quality and security, adv capabilities, and leverage modern pipelines enabling advanced deployment strategies.

The release train framework to be managed by a cross-functional leadership team (Release Management) in ensuring preparation work, release and release readiness practices are enforced. The release train governance and deliverables consist of documentation, infrastructure, configurati activities. Parties with components registered to a release train, must be full-time engaged in the delivery activities and successfully deployment c central delivery function consisting of delivery management, infrastructure setup, build/deploy engineers, test and security engineers, performanc are strict gated criteria across the release delivery cycle; from registration to production roll out. The criteria to be defined and enforced by Releas management alignment.

Sample Release Train Framework

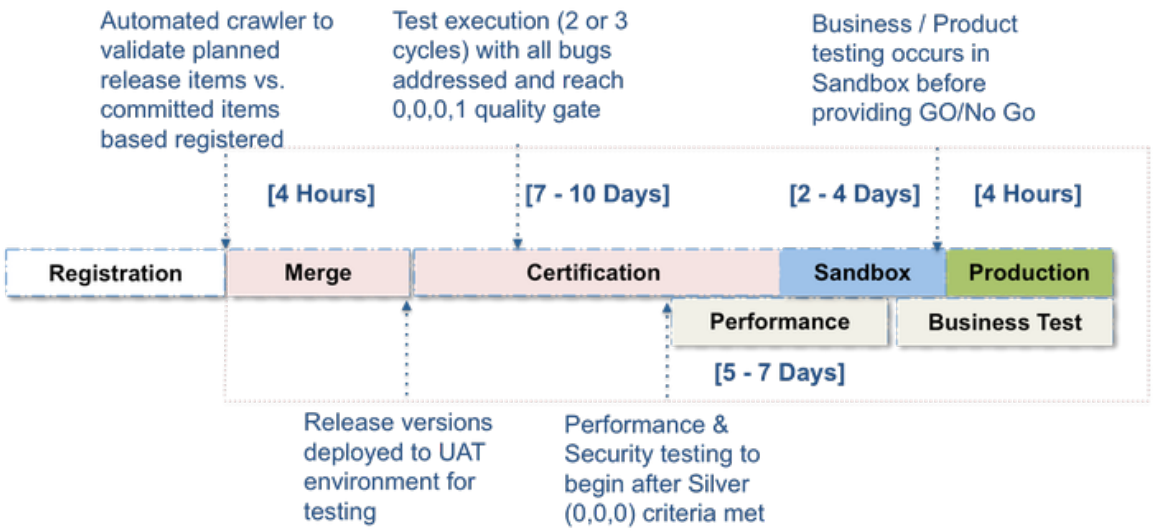
The concept behind a release train to allow for a standard and controlled vehicle where multiple applications / components can be registered, cer consistent and frequent manner. The diagram outlines the release train concept with 3 release trains running in parallel with deployment to produ depicted below, the release train framework consists of 3 phases which are executed in parallel. This approach will require a defined and allocat management function responsible for planning product backlog refinement and release train planning. In realization phase, the program manage ensuring committed deliverables and quality metrics are met following Agile Scrum and Lean practices. In the Release phase - a shared central f management and site reliability team members would drive the execution of all certification activities.

In the recommended release train model, it allows for off cycle releases at sprint cadence with release management oversight.



Teams registered for a monthly release train will be required to take part in the release merge, certification, sandbox, and production release activ 4 weeks (1 week for infrastructure and environment readiness, 2 weeks for certification activities, 1 week for sandbox and production roll out and infrastructure and application architecture matures; we may further modify the release train "release" phase activities to delivery at a higher frequ

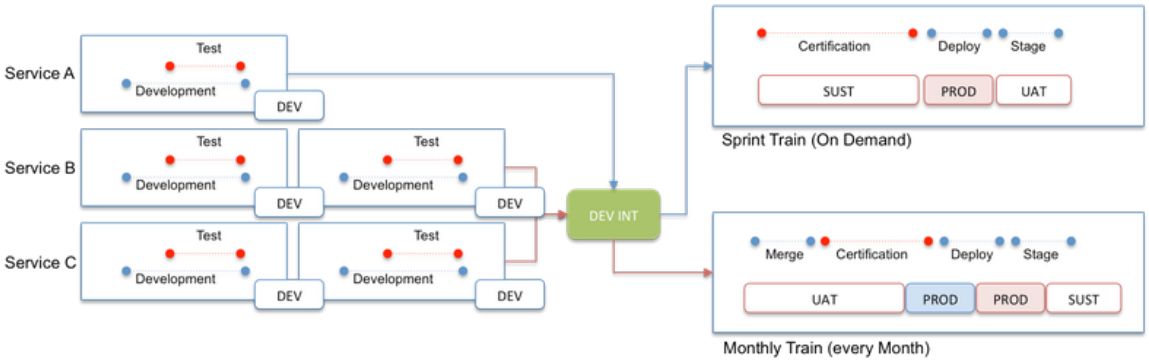
As development sprints end on the same time, registered components will be validated on DEV-INT (development integrated) environment and d vehicle by producing their component release labels with all quality gates passed. Within 4 hour period, the necessary governance, tracking, and release train are executed into UAT environment. Certification phase would consist of 2 – 3 certification cycles (integration, regression, and perfc more than 2 – 3 days. Test engineering teams determine release readiness by reviewing test results; determining release readiness. In parallel, p managed in PERF environment.



Once Silver (0 critical, 0 high, 0 medium, 2 low) build baseline has been achieved, release train is deployed to sandbox environment (production - would have 24 – 48 hours to test and resolve any issues identified in production environment. When gold (0 critical, 0 high, 0 medium, 0 low) bas activated in production as part of blue/green deployment.

Delivery & Environment Strategy

Delivery agility and quality can be attained with the adoption of an effective environment strategy that supports the corresponding delivery framew two types of logical environment regions (Non-Production, Production) with various environment types and ownership. Non-production environme (non-managed, managed), where Development teams will have full read/write access non-managed environments and Release Management / E environments.



Transformational Roadmap