Q1: Which steps in the Value Stream should be the main focus when prioritizing improvement items?

A1: Steps with long lead time and short process time in the current-state map

**Explanation**

<https://www.scaledagileframework.com/identify-value-streams-and-arts/>

**Lead time** – The time from the trigger to the delivery of value is the lead time. Shortening the lead time reduces the time to market. The easiest way to shorten lead time is to identify and reduce (or remove) non-value added activities and wasteful delays. That’s the primary focus of Lean thinking.

Q2: What work is performed in the Build activity of the Continuous Delivery Pipeline?

A2: Compile

#### Explanation

https://www.scaledagileframework.com/continuous-integration

Purpose: **Compile source files** into deployable binaries, verify that code functions as the developer(s) intented, and merge dev branches to trunk.

**Build the Solution**

During the build phase, teams continuously integrate new code. This can be accomplished by automating the build and test tools to run upon code commit. Passing versus not-yet-passing and broken automated tests are the real indicators of progress. Automating code building enables teams to fix problems quickly, before they affect larger parts of the system. Addressing a broken build should be the highest priority. A ‘gated commit’ ensures software has passed the gate (e.g. unit tested, performance tested, and free of known defects, etc.) before being checked into the main codebase or trunk. Code that passes the gate is automatically integrated into the trunk; which removes the complications of managing multiple branches. This trunk-based development helps to ensure the code can be reliably released on demand without the need for costly code freezes or hardening iterations.

There are five practices which can help build the solution:

\* **Continuous code integration** – Code commit should automatically trigger compilation and testing of changes. Ideally, this happens on each commit but should happen at least several times a day.

\* **Build and test automation** – The compilation process should be automated and include unit- and story-level tests to verify the change. These tests often use test doubles to replicate other parts of the systems and enable fast builds.

\* **Trunk-based development** – Long-lived branches must be avoided. Teams should merge back as quickly as they can, at least once per day, and all teams should work off a single trunk.

\* **Gated commit**– Committing to a single trunk is risky, as broken changes can impact many teams. This is why only the changes that have been validated through the build and test process are merged into the trunk.

\* **Application security** – Code analysis tools inspect the code and third-party packages for known vulnerabilities.

Q3: Scanning application code for security vulnerabilities is an important step in which aspect of the Continuous Delivery Pipeline?

A3:

#### Explanation

<https://www.scaledagileframework.com/continuous-integration/>

Build the Solution During the build phase, teams continuously integrate new code.

There are five practices which can help build the solution:

**Continuous code integration** – Code commit should automatically trigger compilation and testing of changes. Ideally, this happens on each commit but should happen at least several times a day.

**Build and test automation**– The compilation process should be automated and include unit- and story-level tests to verify the change. These tests often use test doubles to replicate other parts of the systems and enable fast builds.

**Trunk-based development** – Long-lived branches must be avoided. Teams should merge back as quickly as they can, at least once per day, and all teams should work off a single trunk.

**Gated commit**– Committing to a single trunk is risky, as broken changes can impact many teams. This is why only the changes that have been validated through the build and test process are merged into the trunk.

**Application security** – Code analysis tools inspect the code and third-party packages for known vulnerabilities.

A4:

#### Explanation

<https://www.scaledagile.com/devops/>

**Value Stream Mapping**

When implementing DevOps, the problem is that even after a good action plan is created, the team is not sure how to act on it or whether the right people are even working on it. Value Stream mapping is a multifaceted tool for identifying problems with the flow of value and building a better future map and action plan. This draws clarity to where the bottlenecks lie and where the improvements are most needed.

Why map the Value Stream?

► Understand how work flows through the organization from concept to cash

► Measure process quality and organizational efficiency

► Identify bottlenecks to the flow of value

► Understand how we can improve the flow of value

A5:

#### Explanation

"https://www.scaledagileframework.com/continuous-exploration/

SAFe describes four activities of Continuous Exploration. One of them is Hypothesize (Lean UX).

Hypothesize describes the practices necessary to capture ideas and the measurements needed to validate them with customers.

Accordingly, the first set of practices are associated with the ability to leverage hypothesis based development:

\* **Lean startup thinking** – Defining Minimum Marketable Feature (MMFs) and/or Minimum Viable Products (MVPs) helps evaluate hypotheses quickly before too much investment has been made. Both MMFs and MVPs represent the smallest thing that can be built to evaluate whether the hypothesis is valid.

\* **Innovation Accounting** – Evaluating hypotheses requires different metrics than those used to measure end-state working solutions. Innovation Accounting focuses on how to measure the intermediate and predictive business outcomes of a hypothesis both during initial incremental solution development and evaluation of the MVP. (Read more in the Innovation Accounting article.)

A6:

#### Explanation

https://www.scaledagileframework.com/calmr/

SAFe’s CALMR approach to DevOps is a mindset that guides ARTs toward achieving continuous value delivery by managing simultaneous advancements in delivery culture, automation, lean flow, measurement, and recovery. Successful DevOps hinges on an approach that unites everyone in the value stream toward achieving extraordinary business outcomes. In SAFe, CALMR provides such an approach. When everyone in the value stream thinks and acts with continuous delivery in mind, the result is: Increased frequency, quality, and security of product innovation Decreased deployment risk with accelerated learning cycles Accelerated solution time-to-market Improved solution quality and shortened lead time for fixes Reduced severity and frequency of failures and defects Improved Mean Time to Recover (MTTR) from production incidents

A7:

#### Explanation

"https://www.scaledagileframework.com/continuous-deployment

Feature toggles – a technique to facilitate dark launches by implementing toggles in the code, which enables switching between old and new functionality

Before being released to end users, deployments must be verified for functional integrity and robustness. When they’re coupled, deployment and release have to happen almost instantaneously, as decisions must be made immediately about whether to rollback or not. When they’re decoupled, however, there’s room to test new functionality extensively in production before approving it for release. Immediately following the migration to production, solutions undergo a final round of testing. Typically, this is conducted through smoke testing and/or light user acceptance testing, but also as a stress and performance test, which can only be done in production. This provides a critical sanity check that tests the behavior of the solution in an actual production environment.

A8:

#### Explanation

https://www.scaledagileframework.com/built-in-quality/

**Unit Testing and Test-Driven Development**

The unit testing practice breaks the code into parts and ensures that each part has automated tests to exercise it. These tests run automatically after each change and allow developers to make fast changes, confident that the modification won’t break another part of the system. Tests also serve as documentation and are executable examples of interactions with a component’s interface to show how that component should be used.

Test-Driven Development (TDD) guides the creation of unit tests by specifying the test for a change before creating it. This forces developers to think more broadly about the problem, including the edge cases and boundary conditions before implementation. Better understanding results in faster development with fewer errors and less rework.

**Pair Work**

Pairing has two developers work the same change at the same workstation. One serves as the driver writing the code while the other as the navigator providing real-time review and feedback. Developers switch roles frequently. Pairing creates and maintains quality as the code will contain the shared knowledge, perspectives, and best practices from each member. It also raises and broadens the skillset for the entire team as teammates learn from each other.

#### A9:

#### Explanation

<https://www.scaledagileframework.com/devops/>

A Paradigm Shift IT organizations worldwide are plagued by a core, chronic conflict: technology delivery processes that are reliant on teams with opposing goals and incentives. Development teams must deliver changes quickly to keep pace with business needs. Operations teams must regulate the flow of changes to maintain the stability of solutions that run the business. Security teams must institute policies to prevent changes from introducing vulnerabilities that can cause data breaches. To correct this, a new delivery system is needed—a ‘software factory’ that aligns teams and increases delivery speed while simultaneously increasing solution quality, security, and stability. Only then can the needs of all teams and the needs of the customer be met predictably and effectively.

A10:

#### Explanation

<https://www.scaledagileframework.com/continuous-integration/>

Trunk/main will not always be in deployable state

Trunk-based development – Long-lived branches must be avoided. Teams should merge back as quickly as they can, at least once per day, and all teams should work off a single trunk.

A11:

#### Explanation

<https://www.scaledagileframework.com/continuous-deployment/>

**Self-service deployment** – when automation deployment is not fully implemented, self-service deployment allows a single command to take solutions from staging to production

A12:

#### Explanation

https://www.scaledagileframework.com/continuous-delivery-pipeline/

The pipeline consists of four aspects: Continuous Exploration (CE), Continuous Integration (CI), Continuous Deployment (CD), and Release on Demand.

A13:

#### Explanation

Boundaries and Limitations:

Improvement items that are outside of our control. They require significant involvement from other people and/or teams.

A14:

#### Explanation

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

Lack of alignment impedes progress

► Different groups in the organization have different goals and directions

► The lack of alignment means their different efforts cancel each other out

► This creates a feeling of constant work with little or no progress

A15: ?

#### Explanation

**Test automation**

Many types of testing need to be run:

- Functional testing

- Integration testing

- Regression testing

- Performance testing

- Security testing

- Exploratory testing

- Penetration testing

**Release On Demand - Measure**

► Purpose: Determine actual business value delivered through feedback collected from production

► Application telemetry creates a way to evaluate the business results of a hypothesis

A16:

#### Explanation

<https://www.scaledagileframework.com/release-on-demand/>

**Stabilize and Operate**

The changes to the solution have been verified after they were deployed, but once customers have access to them, new problems might arise. These new problems not only may be due to the increase in usage, but also due to unexpected usage patterns. Incidents and security threats must be resolved quickly and within agreed to Service Level Agreements (SLAs). Four practices help operate the solution:

\* **Cross-team collaboration** – A mindset of cooperation across the Value Stream to identify and solve problems as they arise is crucial. This involves building Agile Release Trains that can operate the solution, as well as develop it.

\* **Failover/disaster recovery** – Failures will occur. It’s vital to develop a failover mechanism to allow service to resume quickly, or even avoid service interruption. Disaster recovery must be planned, architected into the service, and practiced.

\* **Continuous security monitoring** – Security as code and penetration testing focus on preventing known vulnerabilities from getting to production. But it’s also important to test services continuously for newly discovered and reported vulnerabilities and to detect intrusions and attacks on services and infrastructure.

\* **Architect for operations** – Operational needs must be considered. High loads, security attacks, and responding to incidents motivate a range of options from downgrading or removing services to adding capacity. Telemetry and logging capabilities enable organizations to better understand and further tune their architecture to meet actual usage patterns.

\* **Monitor nonfunctional requirements (NFRs)** – To avoid service disruptions, system attributes such as reliability, performance, maintainability, scalability, and usability must be continuously monitored.

A17:

#### Explanation

Develop on Cadence, Release on Demand – ARTs apply cadence and synchronization to help manage the inherent variability of research and development. However, releasing is typically decoupled from the development cadence. ARTs can release a solution, or elements of a solution, at any time, subject to governance and release criteria.

Each ART builds and maintains (or shares) a Continuous Delivery Pipeline with the assets and technologies needed to deliver solution value as independently as possible. The first three elements of the pipeline work together to support the deployment of small batches of new functionality, which are released to meet market demands.

A18:

#### Explanation

https://www.scaledagileframework.com/continuous-deployment/

**Deploy to Production**

\* Blue/green deployment – a technique that permits automatic switching between two environments, one for deployment and one that is live

A19:

#### Explanation

Correct answer: Surprises are found in deployment that lead to significant rework and delay

**Create Alignment to Achieve Fast Flow**

https://www.scaledagileframework.com/built-in-quality/

Alignment and shared understanding reduce developer delays and rework, enabling fast flow. Behavior-Driven Development (BDD) defines a collaborative practice where the Product Owner and team members agree on the precise behavior for a story or feature. Applying BDD helps developers build the right behavior the first time and reduces rework and errors. Model-Based Systems Engineering (MBSE) scales this alignment to the whole system. Through an analysis and synthesis process, MBSE provides a high-level, complete view of all the proposed functionality for a system, and how the system design realizes it.

A20:

#### Explanation

https://www.scaledagileframework.com/continuous-deployment/

**Respond and Recover**

The ability to respond to and recover from unforeseen production issues is critical to supporting continuous deployment and streamlining the Continuous Delivery Pipeline.

Six practices support the ability to respond and recover from production issues:

\* **Proactive detection** – a practice for proactively creating faults in the solution to identify potential problems and situations before they occur

\***Cross-team collaboration** – a mindset of cooperation across the Value Stream to identify and solve problems as they arise

\* **Session replay** – the ability to replay end-user sessions to research incidents and identify problems

\* **Rollback and fix forward** – the ability to both rollback a solution quickly to a previous environment, or to fix a problem quickly through the pipeline without the need to rollback

\* **Immutable infrastructure** – this concept recommends that teams never change the elements of the production environment in an uncontrolled manner, but instead manage infrastructure changes through the Continuous Delivery Pipeline

\* **Version control** – environments should be maintained under version control in order to rollback quickly

A21:

#### Explanation

<https://www.scaledagile.com/devops/>

Value Stream mapping is a multifaceted tool for identifying problems with the flow of value and building a better future map and action plan. This draws clarity to where the bottlenecks lie and where the improvements are most needed.

A22:

#### Explanation

<https://www.scaledagileframework.com/continuous-deployment/>

When verification reveals critical defects, deployments must either be rolled back or fixed quickly to prevent them from contaminating the production environment or disrupting the flow of business.

A23:

#### Explanation

<https://www.scaledagileframework.com/wsjf/>

**Weighted Shortest Job First**

Weighted Shortest Job First (WSJF) is a prioritization model used to sequence jobs (eg., Features, Capabilities, and Epics) to produce maximum economic benefit. In SAFe, WSJF is estimated as the Cost of Delay (CoD) divided by job size.

Economic prioritization

► In a flow system, job sequencing is the key to economic outcomes.

► Give preference to jobs with shorter duration and higher cost of delay (CoD) using weighted shortest job first (WSJF)

► WSJF provides a way of understanding the cost of delay and focusing on items that provide the best cost of delay reduction in the shortest time

A24:

#### Explanation

"https://www.scaledagileframework.com/wsjf/

**Weighted Shortest Job First**

Weighted Shortest Job First (WSJF) is a prioritization model used to sequence jobs (eg., Features, Capabilities, and Epics) to produce maximum economic benefit. In SAFe, WSJF is estimated as the Cost of Delay (CoD) divided by job size.

A25:

#### Explanation

<https://www.scaledagileframework.com/continuous-integration>

**Continuous Integration (CI)** focuses on taking features from the Program backlog and implementing them. In CI, the application of design thinking tools in the problem space focuses on refinement of features (e.g., designing a user story map), which may motivate more research and the use of solution space tools (such as user feedback on a paper prototype). After specific features are clearly understood, Agile Teams implement them. Completed work is committed to version control, built and integrated into a full system or solution, and tested end-to-end before being validated in a staging environment.

A26:

#### Explanation

<https://www.scaledagileframework.com/continuous-deployment/>

**Deploy to Production**

Deployment is the migration of changes into a production environment. In the Continuous Delivery Pipeline, such changes are deployed continuously. Thus, partial functionality—i.e., deploying some of the stories that comprise a feature—can be implemented into production.

A27:

#### Explanation

\* Identify the role or team responsible for each step.

\* Evaluate the efficiency of your current Value Stream (Lead Time, Process time, etc)

A28:

#### Explanation

<https://www.scaledagileframework.com/devops/>

DevOps is a mindset, a culture, and a set of technical practices. It provides communication, integration, automation, and close cooperation among all the people needed to plan, develop, test, deploy, release, and maintain a Solution.

The goal is simple: deliver value whenever there is a business need. This is indeed achievable, as teams that excel at DevOps, on average, deploy 208 times more frequently, deploy 106 times faster, experience 7 times fewer failures, and recover from incidents 2,604 times faster than low performing teams.

A29:

#### Explanation

https://www.scaledagileframework.com/continuous-delivery-pipeline/

The pipeline consists of four aspects: Continuous Exploration (CE), Continuous Integration (CI), Continuous Deployment (CD), and Release on Demand, each of which is described in its own article.

A30:

#### Explanation

**Innovation accounting**

It is important to focus on Metrics that demonstrate real customer engagement and not on vanity Metrics

A31:

#### Explanation

<https://www.scaledagileframework.com/continuous-deployment/>

**Respond and Recover:**

Six practices support the ability to respond and recover from production issues:

\* **Proactive detection** – a practice for proactively creating faults in the solution to identify potential problems and situations before they occur

\***Cross-team collaboration** – a mindset of cooperation across the Value Stream to identify and solve problems as they arise

\* **Session replay** – the ability to replay end-user sessions to research incidents and identify problems

\* **Rollback and fix forward** – the ability to both rollback a solution quickly to a previous environment, or to fix a problem quickly through the pipeline without the need to rollback

\* **Immutable infrastructure** – this concept recommends that teams never change the elements of the production environment in an uncontrolled manner, but instead manage infrastructure changes through the Continuous Delivery Pipeline

\* **Version control**– environments should be maintained under version control in order to rollback quickly

A32:

#### Explanation

<https://www.scaledagileframework.com/release-on-demand/>

**Canary releases** – These provide a mechanism for releasing the solution to a specific Customer segment and measuring the results, before expanding and releasing to more customers.

A33:

#### Explanation

"https://www.scaledagileframework.com/continuous-exploration

Customers, Suppliers, partners, Business Owners, Agile Teams, Product Owners, and Lean Portfolio Management are among the internal and external stakeholders involved in this process. Their involvement may be indirect, such as through secondary research on market needs. Or it can be direct, as when Agile Teams are participating in an Innovation and Planning Iteration. The results of CE activities enable the organization to align to a shared Vision, a set of features in the backlog defined for implementation, and a Roadmap forecast of when those features might be delivered.

A34:

#### Explanation

https://www.scaledagileframework.com/lean-ux

**Lean UX**

Lean User Experience (Lean UX) design is a mindset, culture, and a process that embraces Lean-Agile methods. It implements functionality in minimum viable increments and determines success by measuring results against a benefit hypothesis.

**Benefit Hypothesis**

The Lean UX approach starts with a benefit hypothesis: Agile teams and UX designers accept the reality that the ‘right answer’ is unknowable up-front. Instead, teams apply Agile methods to avoid Big Design Up-front (BDUF), focusing on creating a hypothesis about the feature’s expected business result, and then they implement and test that hypothesis incrementally.

**Collaborative Design**

Traditionally, UX design has been an area of specialization. People who have an eye for design, a feel for user interaction, and specialty training were often entirely in charge of the design process.

Continuous Exploration takes the hypothesis and facilitates a continuous and collaborative process that solicits input from a diverse group of stakeholders – Architects, Customers, Business Owners, Product Owners, and Agile Teams. This further refines the problem and creates artifacts that clearly express the emerging understanding including personas, empathy maps, and customer experience maps.

A35:

#### Explanation

Value stream mapping metrics include following:

    \* PT = processing time (how long does it take to perform the process)

    \* LT = lead time (what is the latency between initiation and execution of a process)

    \* AR = activity ratio = PT / LT \* 100

    \* %C&A = % complete and accurate (determine in % if we needed to return to the previous process to re-ask and update information before we start our process)

Now perform a quick calculation:

    \* Total PT= sum of all PTs per process

    \* Total LT = sum of all LTs per process

    \* Rolled AR = product of all ARs = 28/100 \* 35/100 \* 87.5/100 \* …

    \* Rolled %C&A= product of all %C&A = 80/100 \* 71/100 \* 80/100 \* …

A36:

#### Explanation

<https://www.scaledagileframework.com/continuous-exploration/>

Collaboration and research are grounded in specific practices:

\* **Primary market research** – Product Management develops additional insights through primary market research, including surveys, focus groups, questionnaires, competitive analysis, and Innovation Games for Customer Understanding.

\* **Gemba walk**s – An experiential form of primary market research, a Gemba walk (‘Gemba’ is the place where the work is performed) is a process in which the product team observe how stakeholders execute the steps and specific activities in their operational value streams to better identify opportunities for relentless improvement.

\* **Customer visits**– There’s no substitute for first-person observation of the daily activities of the people doing the work. Whether structured or informal, Product Managers and Product Owners are responsible for understanding how people actually use systems in actual work environments. They can’t do that at their desk, so there is no substitute for “getting outside the building” and observing users in their specific Solution Context.

\* **Secondary market researc**h – To broaden their thinking, Product Management uses a variety secondary market research techniques to develop a comprehensive understanding of the customers and markets they’re serving. Staying abreast of market/industry trends is a critical outcome of secondary market research.

\* **Lean UX thinking** – Lean UX is a collaborative process of working with stakeholders to define Minimum Marketable Features (MMFs) and validate them quickly with customers. (The Lean UX article provides more information about this process.)

A37:

**Explanation**

**The percentage of time spent on value-added activities**

The activity ratio is a measure of how quickly work moves through a process. It is simply the sum of the process times divided by the total lead time. For example, if the processing time was 30 minutes and the lead time was 2 hours, the activity ratio would be . 25 or 25%

**Exampl**[**e**](https://www.scaledagileframework.com/development-value-streams/)**(Scaled Agile):**

[https://www.scaledagileframework.com/development-value-stream](https://www.scaledagileframework.com/development-value-streams/)s/

A picture containing diagram

Description automatically generated

From the data above, it’s clear there is substantial room for improvement. Only 5% of the lead time was value-added activity time. The other 95% was spent waiting. Also, there was substantial rework, as indicated by only 36% of the features making it through the process without revisiting prior activities.

A38:

#### Explanation

https://www.scaledagileframework.com/release-on-demand/

**Release on Demand**

Release on Demand is the process that deploys new functionality into production and releases it immediately or incrementally to customers based on demand.

**Release Value to Customers**

When the Solution is in production and has been verified as operable, the time has come to make it available to customers. This is a crucial business decision, as releasing value too early or too late can have negative economic repercussions.

A39:

#### Explanation

https://www.scaledagile.com/devops/

Approaches to implementing DevOps are diverse, without a standard playbook or central manifesto. But successful DevOps organizations do tend to have certain capabilities in common. They help inform the principles on which SAFe DevOps is based. SAFe calls these principles a CALMR approach to DevOps.

**Culture**

Establish a culture of shared responsibility for development, deployment, and operations.

**Automation**

Automate the continuous delivery pipeline.

**Lean flow**

Keep batch sizes small, limit WIP, and provide extreme visibility.

**Measurement**

Measure the flow through the pipeline. Implement application telemetry.

**Recovery**

Architect and enable low risk releases. Establish fast recovery, fast reversion, and fast fix-forward.

A40:

#### Explanation

"https://www.scaledagile.com/devops/

**Gaining alignment with continuous exploration**

Continuous exploration is the process of continually exploring the market and user needs, as well as defining a Vision, Roadmap, Architecture, and set of Features that address these needs. CE helps to build alignment across the organization about what should be built or what hypothesis should be evaluated.

**Building quality in with continuous integration**

Stakeholders need to see work turning into tangible, working solutions. Continuous integration ensures that development teams are not only busy, but productive, by delivering working software and systems each Iteration. Without CI, teams may be sprinting, but the system as a whole might not be iterating.

**Reducing time-to-market with continuous deployment**

Enterprises deploy continuously to get to market quickly and enable the business to release value on demand. In SAFe, deployment is decoupled from release. Therefore, continuous deployment focuses on deploying continuously to production, not going live to end users.

**Delivering business value with release on demand**

Releasing value on demand provides the business with the capability of getting value into the hands of users when there is market and user demand for it. The final part of every hypothesis learning cycle is evaluating with customers and users, and learning for further exploration. Responsibility for operating the solution is shared across the organization.

A41:

#### Explanation

<https://www.scaledagileframework.com/continuous-integration/>

**Continuous Integration**

Continuous Integration (CI) is the process of taking features from the Program Backlog and developing, testing, integrating, and validating them in a staging environment where they are ready for deployment and release.

But the fact remains: integrating and testing components together frequently is the only practical way to fully validate a solution.

Test end-to-end describes the practices necessary to validate the solution.

A42:

#### Explanation

<https://www.scaledagileframework.com/continuous-delivery-pipeline/>

Continuous Exploration (CE) focuses on creating alignment on what needs to be built. In CE, design thinking is used to ensure the enterprise understands the market problem / customer need and the solution required to meet that need. It starts with an idea or a hypothesis of something that will provide value to customers, typically in response to customer feedback or market research. Ideas are then analyzed and further researched, leading to the understanding and convergence of what is needed as either a Minimum Viable Product (MVP) or Minimum Marketable Feature (MMF).

A43:

#### Explanation

<https://www.scaledagileframework.com/release-on-demand/>

Release describes the practices necessary to deliver the solution to end users, all at once or incrementally

A44:

#### Explanation

<https://www.scaledagileframework.com/continuous-delivery-pipeline>

**Continuous Delivery Pipeline**

The Continuous Delivery Pipeline (CDP) represents the workflows, activities, and automation needed to shepherd a new piece of functionality from ideation to an on-demand release of value to the end user.

The pipeline is a significant element of the Agile Product Delivery competency. Each Agile Release Train (ART) builds and maintains, or shares, a pipeline with the assets and technologies needed to deliver solution value as independently as possible. The first three elements of the pipeline (CE, CI, and CD) work together to support the delivery of small batches of new functionality, which are then released to fulfill market demand.

A45:

#### Explanation

<https://www.scaledagileframework.com/wsjf/>

**Weighted Shortest Job First**

Weighted Shortest Job First (WSJF) is a prioritization model used to sequence jobs (eg., Features, Capabilities, and Epics) to produce maximum economic benefit. In SAFe, WSJF is estimated as the Cost of Delay (CoD) divided by job size.

In a flow-based system, updating priorities continuously provides the best economic outcomes. In such a flow context, it is job sequencing, rather than theoretical, individual job return on investment, that produces the best result.

To that end, SAFe applies WSJF to prioritize backlogs by calculating the relative Cost of Delay (CoD) and job size (a proxy for the duration). Backlog priorities are continuously updated based on relative user and business value, time factors, risk reduction and opportunity enablement, and relative job size. WSJF also conveniently and automatically ignores sunk costs, a fundamental principle of Lean economics.

EXAM2

A1:

#### Explanation

<https://www.scaledagileframework.com/continuous-delivery-pipeline/>

**Continuous Exploration**(CE) focuses on creating alignment on what needs to be built. In CE, design thinking is used to ensure the enterprise understands the market problem / customer need and the solution required to meet that need. It starts with an idea or a hypothesis of something that will provide value to customers, typically in response to customer feedback or market research. Ideas are then analyzed and further researched, leading to the understanding and convergence of what is needed as either a Minimum Viable Product (MVP) or Minimum Marketable Feature (MMF).

A2:

#### Explanation

Productivity is a business measurement which captures the rate of outcome by employees. Hence it is one of the business benefits of DevOps.

A3:

#### Explanation

"https://www.scaledagileframework.com/continuous-deployment/

**Respond and Recover:**

Six practices support the ability to respond and recover from production issues:

\* **Proactive detection** – a practice for proactively creating faults in the solution to identify potential problems and situations before they occur

\* **Cross-team collaboration** – a mindset of cooperation across the Value Stream to identify and solve problems as they arise

\* **Session replay** – the ability to replay end-user sessions to research incidents and identify problems

\* **Rollback and fix forward** – the ability to both rollback a solution quickly to a previous environment, or to fix a problem quickly through the pipeline without the need to rollback

\* **Immutable infrastructure** – this concept recommends that teams never change the elements of the production environment in an uncontrolled manner, but instead manage infrastructure changes through the Continuous Delivery Pipeline

\* **Version control** – environments should be maintained under version control in order to rollback quickly

A4:

#### Explanation

<https://www.scaledagile.com/devops/>

Value Stream mapping is a multifaceted tool for identifying problems with the flow of value and building a better future map and action plan. This draws clarity to where the bottlenecks lie and where the improvements are most needed.

A5:

#### Explanation

**Evaluate the efficiency of your current Value Stream**

\* Calculate the total lead time, total process time, activity ratio, and rolled % complete and accurate

Total lead time is defined as the time it takes for a customer to receive a good or service from the time the customer places the order or performs a purchase transaction.

A6:

#### Explanation

One aspect In Release On Demand it to determine actual business value delivered  through feedback collected from production.

**Measure the Business Value**

The first activity of continuous exploration is to hypothesize—and as value is released to customers, it’s time to use the application telemetry to measure the hypothesis and the business value delivered. Two practices support this effort:

    \* **Application telemetry** – Application telemetry is the primary mechanism used to track and measure data usage against the hypothesis.

    \* **Innovation Accounting** – Evaluating a hypothesis requires different metrics than those used to measure end-state working solutions. Innovation Accounting focuses on how to measure the intermediate and predictive business outcomes of the hypothesis during initial incremental solution development and evaluation of the Minimum Viable Product (MVP). (Read more in the Innovation Accounting article.)

A7:

#### Explanation

Continuous Deployment removes the manual step (one click deployment) from Continuous Delivery pipeline. Thus each step is automated in the pipeline with out any human intervention.

A8:

#### Explanation

<https://www.scaledagile.com/devops/>

DevOps is a mindset, an enterprise-wide culture and practice. Although it does not have a commonly accepted manifesto or playbook, core capabilities of successful DevOps implementations can be identified. The five core concepts of DevOps and Release on Demand include:

1- Value Stream Mapping

2 - Gaining alignment with Continuous Exploration (CE)

3 - Building quality in with Continuous Integration (CI)

4 - Reducing **time-to-marke**t with Continuous Deployment (CD)

5 - Delivering business value with release on demand

A9:

#### Explanation

<https://www.scaledagileframework.com/accelerating-flow-with-devsecops-and-the-software-factory/>

Security is an extremely important topic as vulnerabilities can lead to existential business threats. Hackers are trying to harm businesses in many ways, such as by stealing intellectual property and customer data or trying to take over systems to obtain a ransom.

Continuous security monitoring monitors systems to find potential security breaches. This is usually done by a security information and event management (SIEM) system that incorporates advanced analytics such as event correlation, user behavior analytics, network flow insights, artificial intelligence, and incident forensics. With the collected data, security analysts can derive the proper actions and generate applicable security compliance and audit data.

**Continuous Exploration practices**

\* **Threat modeling** analyzes a system to answer vulnerability questions such as, Where am I most vulnerable to attack? What are the most relevant threats? What do I need to do to safeguard against these threats?

\* **Penetration testing** (pen testing) attempts to breach IT systems by simulating any kind of hacker attack. This type of testing requires highly skilled professionals who can apply the same tools and techniques that real hackers would use.

A10:

#### Explanation

<https://www.scaledagileframework.com/continuous-deployment>

**Feature toggles** – a technique to facilitate dark launches by implementing toggles in the code, which enables switching between old and new functionality

Before being released to end users, deployments must be verified for functional integrity and robustness. When they’re coupled, deployment and release have to happen almost instantaneously, as decisions must be made immediately about whether to rollback or not. When they’re decoupled, however, there’s room to test new functionality extensively in production before approving it for release. Immediately following the migration to production, solutions undergo a final round of testing. Typically, this is conducted through smoke testing and/or light user acceptance testing, but also as a stress and performance test, which can only be done in production. This provides a critical sanity check that tests the behavior of the solution in an actual production environment.

A11:

#### Explanation

**Continuous Integration**(CI) focuses on taking features from the Program backlog and implementing them. In CI, the application of design thinking tools in the problem space focuses on refinement of features (e.g., designing a user story map), which may motivate more research and the use of solution space tools (such as user feedback on a paper prototype). After specific features are clearly understood, Agile Teams implement them. Completed work is committed to version control, built and integrated into a full system or solution, and tested end-to-end before being validated in a staging environment.

A12:

#### Explanation

https://www.scaledagileframework.com/release-on-demand/

**Measure the Business Value**

The first activity of continuous exploration is to hypothesize—and as value is released to customers, it’s time to use the application telemetry to measure the hypothesis and the business value delivered. Two practices support this effort:

A13:

#### Explanation

<https://www.scaledagile.com/devops/>

**Gaining alignment with continuous exploration**

Continuous exploration is the process of continually exploring the market and user needs, as well as defining a Vision, Roadmap, Architecture, and set of Features that address these needs. CE helps to build alignment across the organization about what should be built or what hypothesis should be evaluated.

**Building quality in with continuous integration**

Stakeholders need to see work turning into tangible, working solutions. Continuous integration ensures that development teams are not only busy, but productive, by delivering working software and systems each Iteration. Without CI, teams may be sprinting, but the system as a whole might not be iterating.

**Reducing time-to-market with continuous deployment**

Enterprises deploy continuously to get to market quickly and enable the business to release value on demand. In SAFe, deployment is decoupled from release. Therefore, continuous deployment focuses on deploying continuously to production, not going live to end users.

**Delivering business value with release on demand**

Releasing value on demand provides the business with the capability of getting value into the hands of users when there is market and user demand for it. The final part of every hypothesis learning cycle is evaluating with customers and users, and learning for further exploration. Responsibility for operating the solution is shared across the organization.

A14:

#### Explanation

<https://www.scaledagileframework.com/continuous-exploration>

Customers, Suppliers, partners, Business Owners, Agile Teams, Product Owners, and Lean Portfolio Management are among the internal and external stakeholders involved in this process. Their involvement may be indirect, such as through secondary research on market needs. Or it can be direct, as when Agile Teams are participating in an Innovation and Planning Iteration. The results of CE activities enable the organization to align to a shared Vision, a set of features in the backlog defined for implementation, and a Roadmap forecast of when those features might be delivered.

A15:

#### Explanation

Communication and collaboration is a critical success factor in the DevOps transformation. Because culture eats strategy for breakfast!

A16:

#### Explanation

Inertia means unwilling to change and remaining in the same state. Hence it is NOT encouraged in a DevOps transformation.

A17:

#### Explanation

https://www.scaledagileframework.com/calmr

**Recovery**

To support frequent and sustained value delivery, the continuous delivery pipeline must be designed for low-risk releases and fast recovery from operational failure. Techniques to achieve a more flexible release process are described in the Release on Demand article. In addition, the following techniques support fast recovery:

\* **Stop-the-line mentality** – With a ‘stop-the-line’ mentality, any issue that compromises solution value causes team members to stop what they are doing and swarm on the issue until it is resolved. Learnings are then turned into permanent fixes to prevent the issue from recurring.

\* **Plan for and rehearse failures** – When it comes to DevOps, failed deployments are not only an option, they are expected from time to time. To minimize the impact of failures and maximize the resiliency of solutions, teams should develop recovery plans and practice them often in production or production-like environments.

\* **Fast fix forward and roll back** – Since production failures are inevitable, teams need to develop the capability to quickly ‘fix forward’ and, where necessary, roll back to a known stable state. Fixes must flow through the same process as any feature or enhancement; therefore, it is imperative that the continuous delivery pipeline is able to accommodate any type of change at any level of severity.

A18:

#### Explanation

The SAFe DevOps Health Radar (Figure 5) is a tool that helps ARTs and Solution Trains optimize their value stream performance. It provides a holistic DevOps health check by assessing the maturity of the four aspects and 16 activities of the continuous delivery pipeline. The Health Radar is used to measure baseline maturity at any point in a DevOps transformation and guide fast, incremental progress thereafter.

A19:

#### Explanation

<https://www.scaledagileframework.com/inspect-and-adapt/>

**Problem-Solving Workshop**

For addressing systemic problems, a structured, root-cause problem-solving workshop is held by the ART. Root cause analysis provides a set of problem-solving tools used to identify the actual causes of a problem, rather than just addressing the symptoms. The session is typically facilitated by the RTE, in a timebox of two hours or less.

A20:

#### Explanation

**Bottle neck**. The operation or function with the lowest capacity, usually the operation with the longest cycle time per unit. The bottle neck sets the limit for the production pace and thus the capacity of the entire process. The process efficiency is uses to detect bottlenecks.Process Efficiency = Process time / Lead TimeIr our case, the lowest process efficiency time is 0,005 = 0,5 days / 10 days

A21:

#### Explanation

"https://www.scaledagile.com/devops/

Approaches to implementing DevOps are diverse, without a standard playbook or central manifesto. But successful DevOps organizations do tend to have certain capabilities in common. They help inform the principles on which SAFe DevOps is based. SAFe calls these principles a CALMR approach to DevOps.

**Culture**

Establish a culture of shared responsibility for development, deployment, and operations.

**Automation**

Automate the continuous delivery pipeline.

**Lean flow**

Keep batch sizes small, limit WIP, and provide extreme visibility. Identify and eliminate bottlenecks in the Continuous Delivery.

**Measurement**

Measure the flow through the pipeline. Implement application telemetry.

**Recovery**

Architect and enable low risk releases. Establish fast recovery, fast reversion, and fast fix-forward.

A22:

#### Explanation

https://www.scaledagileframework.com/continuous-delivery-pipeline/

**Percent Complete and Accurate** (%C&A) represents the percentage of work that the next step can process without needing rework. Often, delays are caused by poor quality in the upstream (prior) steps. The percent complete and accurate metric helps identify the steps where poor quality might be occurring and causing longer lead times, resulting in delays of value delivery. Figure 4 indicates that 20% of the time the work moving from the ‘Design’ step to the ‘Code’ step, needs to be reworked. Improving the %C&A metric is also essential to improving the flow of value. The %C&A of a single step is extended to rolled percent complete and accurate, a measure that captures the likelihood that an item will pass through the entire workflow without rework. With a cumulative rolled %C&A of 35%, this workflow is reworking more than half of its items.

A23:

#### Explanation

Test data management

► Data for all types of tests must be managed

► Store data in a repository for consistent testing

► Emulate production data to ensure tests reflect realistic situations

A24:

#### Explanation

Continuous Integration helps in executing automated test cases for every code change committed in to the version control system. Thus errors are detected and remediated before the product increment is deployed.

A25:

#### Explanation

[https://www.scaledagileframework.com/built-in-quality/](https://www.scaledagileframework.com/team-and-technical-agility/)

Collective Ownership and Coding Standards Collective ownership reduces dependencies between teams and ensures that any individual developer or team will not block the fast flow of value delivery. Any individual can add functionality, fix errors, improve designs, or refactor. Because the code is not owned by one team or individual, supporting coding standards encourages consistency so that everyone can understand and maintain the quality of each component.

All teams including software, hardware, operations, product marketing, legal, security, compliance, etc. share the goals and principles of built-in quality. However, the practices will vary by discipline because their work products vary.

A26:

#### Explanation

https://www.scaledagile.com/devops/

**Value Stream Mapping**

When implementing DevOps, the problem is that even after a good action plan is created, the team is not sure how to act on it or whether the right people are even working on it. Value Stream mapping is a multifaceted tool for identifying problems with the flow of value and building a better future map and action plan. This draws clarity to where the bottlenecks lie and where the improvements are most needed.

Why map the Value Stream?

► Understand how work flows through the organization from concept to cash

► Measure process quality and organizational efficiency

► Identify bottlenecks to the flow of value

► Understand how we can improve the flow of value

A27:

#### Explanation

DevOps is NOT a set of best practices. It is a cultural movement based on collaboration, automation, Lean, Agile, etc.

A28:

#### Explanation

Visual management tools like dashboards and Kanban boards helps in conveying the information to all stakeholders. This results in more communication, collaboration and trust between various stakeholders.

A29:

#### Explanation

Measurement is one of the five DevOps values. It insists on collecting all possible data in order to understand the current organization. Then with the help of this data improvement plan should be created.

A30:

#### Explanation

<https://www.scaledagileframework.com/release-on-demand/>

When the Solution is in production and has been verified as operable, the time has come to make it available to customers. This is a crucial business decision, as releasing value too early or too late can have negative economic repercussions.

A31:

#### Explanation

<https://www.scaledagileframework.com/continuous-integration/>

Build the Solution During the build phase, teams continuously integrate new code. This can be accomplished by automating the build and test tools to run upon code commit.

There are five practices which can help build the solution:

\* **Continuous code integration** – Code commit should automatically trigger compilation and testing of changes. Ideally, this happens on each commit but should happen at least several times a day.

\* **Build and test automation**– The compilation process should be automated and include unit- and story-level tests to verify the change. These tests often use test doubles to replicate other parts of the systems and enable fast builds.

\* **Trunk-based development** – Long-lived branches must be avoided. Teams should merge back as quickly as they can, at least once per day, and all teams should work off a single trunk. Gated commit – Committing to a single trunk is risky, as broken changes can impact many teams. This is why only the changes that have been validated through the build and test process are merged into the trunk.

\* **Application security** – Code analysis tools inspect the code and third-party packages for known vulnerabilities.

A32:

#### Explanation

Increased market share is one of the 'business' benefits of implementing DevOps. The other options represents the IT benefits of DevOps.

A33:

#### Explanation

Continuous Delivery insists on keeping the software in a releasable state more than working on new features.

A34:

#### Explanation

Primary stakeholders of DevOps include all the roles in the organization and also the suppliers.

A35:

#### Explanation

Continuous Delivery can be achieved by integrating code changes frequently in to version control system, triggering automated builds and automated testing.

A36:

#### Explanation

**Continuous Integration**

**==================**

There are five practices which can help build the solution:

\* Continuous code integration – Code commit should automatically trigger compilation and testing of changes. Ideally, this happens on each commit but should happen at least several times a day.

\* Build and test automation – The compilation process should be automated and include unit- and story-level tests to verify the change. These tests often use test doubles to replicate other parts of the systems and enable fast builds.

\* Trunk-based development – Long-lived branches must be avoided. Teams should merge back as quickly as they can, at least once per day, and all teams should work off a single trunk.

\* Gated commit – Committing to a single trunk is risky, as broken changes can impact many teams. This is why only the changes that have been validated through the build and test process are merged into the trunk.

\* **Application security** – Code analysis tools inspect the code and third-party packages for known vulnerabilities.

There are five practices which can help in end-to-end system testing:

\* Test and production environment congruity – Environment congruity assures that testing exercises the solution as it would behave in front of live users and decreases the probability that defects will escape into production.

\* Test automation – Many types of tests need to be run: functional testing, integration testing, regression testing, etc. The Agile Testing article details a testing matrix of what can and should be automated.

\* Test data management – To create stability, tests must be consistent and realistic, replicating production as much as possible, and under source control.

\* Service virtualization – Different kinds of testing require different environments. Service virtualizations allow teams to simulate a production environment without the costs and effort associated with creating and managing real environments.

\* **Testing nonfunctional requirements (NFRs)** – system attributes such as security, reliability, performance, maintainability, scalability, and usability must also be thoroughly tested. \* Continuous integration with suppliers – Suppliers bring unique contributions that can have a significant impact on lead-time and value delivery. Their work must be continuously integrated as well. It helps to adopt a shared integration cadence and establish objective evaluation mileston[es.](https://www.scaledagileframework.com/continuous-integration/)

<https://www.scaledagileframework.com/continuous-integration/>

A37:

#### Explanation

**Behavior-Driven Development**

Behavior-Driven Development (BDD) is a Test-First, Agile Testing practice that provides Built-In Quality by defining (and potentially automating) tests before, or as part of, specifying system behavior. BDD is a collaborative process that creates a shared understanding of requirements between the business and the Agile Teams. Its goal is to help guide development, decrease rework, and increase flow. Without focusing on internal implementation, BDD tests are business-facing scenarios that attempt to describe the behavior of a Story, Feature, or Capability from a user’s perspective. When automated, these tests ensure that the system continuously meets the specified behavior even as the system evolves. That, in turn, enables Release on Demand. Automated BDD tests can also serve as the definitive statement regarding the as-built system behavior, replacing other forms of behavioral specifications.

**Deploy to production**

**Dark launches**

►Separate deploy (to production) from release (to end users)

► Enables testing and monitoring system behavior in the actual production environment before exposing new functionality to users

**Release On Demand - Measure**

► Purpose: Determine actual business value delivered through feedback collected from production

► Application telemetry creates a way to evaluate the business results of a hypothesis

A38:

#### Explanation

<https://www.scaledagileframework.com/continuous-deployment/>

Traditional development practices treat deployment and release as the same activity: changes deployed to production are immediately available to users. This motivates behaviors that make applying certain design thinking practices, such as A/B Testing, hard to implement, and serve to inhibit the flow of value. Continuous deployment separates the deployment and release process.

A39:

#### Explanation

<https://www.scaledagileframework.com/operational-value-streams/>

Value streams are persistent, enduring for as long as customers continue to place orders for their products or services. They cut across departments and functions, and each contains:

\* All the steps necessary to convert the trigger to the delivery of value

\* The people who perform these steps

\* The systems they use to do their work

\* The flow of information and materials that are necessary to satisfy that request

A40:

#### Explanation

Value Stream Mapping depicts the FLOW of activities and information from request to delivery. Once defined the value stream map helps in identifying and eliminating waste and other inefficiencies from the value stream.

A41:

#### Explanation

Stabilize and operate is an activity of Release on Demand and Blue/green deployment is a technique used in the activity of Continuous Deployment.

A42:

#### Explanation

DevOps is NOT a one size fits all strategy. For every DevOps implementation the strategy needs to be defined based on the organization's vision and goals (WHY).

A43:

#### Explanation

DevOps helps in reducing the failure rate of new releases. This is achieved through collaboration of different teams, continuous integration, continuous delivery, etc.

A44:

#### Explanation

<https://www.scaledagileframework.com/test-driven-development/>

We never have enough time for testing, so let’s just write the test first. —Kent Beck Test-Driven Development Test-Driven Development (TDD) is a philosophy and practice that involves building and executing tests before implementing the code or a component of a system.

Beck and others have defined a set of Extreme Programming (XP) practices under the umbrella label of TDD:

1 - Write the test first, ensuring that the developer understands the required behavior. This could be a new test or a modification of an existing test.

2 - Run the test and watch it fail. Because there’s no code yet, this may seem pointless, but it accomplishes two useful objectives: it verifies the test works, including any testing harnesses and demonstrates how the system will behave if the code is incorrect.

3 - Write the minimum amount of code needed to pass the test. If it fails, rework the code or the test until it passes routinely.

4- Continue implementing new code until all tests pass. This step provides the developer with the confidence that their changes meet the current requirements and haven’t created an error in another part of the system.

5 - **Refactor as necessary** to ensure the design aligns with changing requirements (e.g., emergent design). Developers continually update their designs to ensure that evolving requirements and a growing codebase don’t lead to poor code quality.

A45:

#### Explanation

<https://www.scaledagile.com/devops/>

Value Stream mapping is a multifaceted tool for identifying problems with the flow of value and building a better future map and action plan. This draws clarity to where the bottlenecks lie and where the improvements are most needed.

EXAM3

A1:

#### Explanation

Monitoring and event management data in the form of dashboards can provide valuable feedback to the teams which can be then acted upon towards continuous improvement.

A2:

#### Explanation

Meaningful metrics and re-inforcing new behaviors helps in anchoring the results of DevOps transformation.

A3:

#### Explanation

Any DevOps professional should possess business, technical, soft skills and self management skills.

A4:

#### Explanation

While adopting DevOps the first and foremost thing to be considered is BUSINESS strategies and goals. This defines the approach and practices that can be implemented during DevOps transformation.

A5:

#### Explanation

Detection and remediation of errors before moving to production is NOT part of Continuous Delivery.

A6:

#### Explanation

https://www.scaledagileframework.com/continuous-exploration/

SAFe describes four activities of Continuous Exploration. One of them is Hypothesize.

Hypothesize describes the practices necessary to capture ideas and the measurements needed to validate them with customers.

Accordingly, the first set of practices are associated with the ability to leverage hypothesis based development:

\* **Lean startup thinking** – Defining Minimum Marketable Feature (MMFs) and/or Minimum Viable Products (MVPs) helps evaluate hypotheses quickly before too much investment has been made. Both MMFs and MVPs represent the smallest thing that can be built to evaluate whether the hypothesis is valid.

\* **Innovation Accounting** – Evaluating hypotheses requires different metrics than those used to measure end-state working solutions. Innovation Accounting focuses on how to measure the intermediate and predictive business outcomes of a hypothesis both during initial incremental solution development and evaluation of the MVP. (Read more in the Innovation Accounting article.)

A7:

#### Explanation

https://www.scaledagileframework.com/continuous-deployment/

**Respond and Recover:**

Six practices support the ability to respond and recover from production issues:

\* **Proactive detectio**n – a practice for proactively creating faults in the solution to identify potential problems and situations before they occur

\* **Cross-team collaboration** – a mindset of cooperation across the Value Stream to identify and solve problems as they arise

\* **Session replay**– the ability to replay end-user sessions to research incidents and identify problems

\* **Rollback and fix forward**– the ability to both rollback a solution quickly to a previous environment, or to fix a problem quickly through the pipeline without the need to rollback

\* **Immutable infrastructure** – this concept recommends that teams never change the elements of the production environment in an uncontrolled manner, but instead manage infrastructure changes through the Continuous Delivery Pipeline

\* **Version control** – environments should be maintained under version control in order to rollback quickly

A8:

#### Explanation

**Value Stream Mapping**

When implementing DevOps, the problem is that even after a good action plan is created, the team is not sure how to act on it or whether the right people are even working on it. Value Stream mapping is a multifaceted tool for identifying problems with the flow of value and building a better future map and action plan. This draws clarity to where the bottlenecks lie and where the improvements are most needed.

A9:

#### Explanation

"https://www.scaledagileframework.com/continuous-integration/

Build the Solution During the build phase, teams continuously integrate new code. This can be accomplished by automating the build and test tools to run upon code commit.

There are five practices which can help build the solution:

\* **Continuous code integration** – Code commit should automatically trigger compilation and testing of changes. Ideally, this happens on each commit but should happen at least several times a day.

\* **Build and test automation** – The compilation process should be automated and include unit- and story-level tests to verify the change. These tests often use test doubles to replicate other parts of the systems and enable fast builds.

\* **Trunk-based development**– Long-lived branches must be avoided. Teams should merge back as quickly as they can, at least once per day, and **all teams should work off a single trunk**. Gated commit – Committing to a single trunk is risky, as broken changes can impact many teams. This is why only the changes that have been validated through the build and test process are merged into the trunk.

\* **Application security** – Code analysis tools inspect the code and third-party packages for known vulnerabilities.

A10:

#### Explanation

There is no ideal structure for a DevOps team. Each organization defines their own structure which works for them.

A11:

#### Explanation

In DevOps organizations people are expected to possess business knowledge along with their technical knowledge. Also DevOps organizations use cross functional teams to facilitate collaboration.

A12:

#### Explanation

"https://www.scaledagileframework.com/continuous-delivery-pipeline/

**Percent Complete and Accurate (%C&A)** represents the percentage of work that the next step can process without needing rework. Often, delays are caused by poor quality in the upstream (prior) steps. The percent complete and accurate metric helps identify the steps where poor quality might be occurring and causing longer lead times, resulting in delays of value delivery. Figure 4 indicates that 20% of the time the work moving from the ‘Design’ step to the ‘Code’ step, needs to be reworked. Improving the %C&A metric is also essential to improving the flow of value. The %C&A of a single step is extended to rolled percent complete and accurate, a measure that captures the likelihood that an item will pass through the entire workflow without rework. With a cumulative rolled %C&A of 35%, this workflow is reworking more than half of its items.

A13:

#### Explanation

<https://www.scaledagile.com/devops/>

Approaches to implementing DevOps are diverse, without a standard playbook or central manifesto. But successful DevOps organizations do tend to have certain capabilities in common. They help inform the principles on which SAFe DevOps is based. SAFe calls these principles a CALMR approach to DevOps.

**Culture**

Establish a **culture of shared responsibility**for development, deployment, and operations.

**Automation**

Automate the continuous delivery pipeline.

**Lean flow**

Keep batch sizes small, limit WIP, and provide extreme visibility. Identify and eliminate bottlenecks in the Continuous Delivery.

**Measurement**

Measure the flow through the pipeline. Implement application telemetry.

**Recovery**

Architect and enable low risk releases. Establish fast recovery, fast reversion, and fast fix-forward.

A14:

#### Explanation

"https://www.scaledagileframework.com/continuous-integration/

The Four Activities of Continuous Integration

\* **Develop** describes the practices necessary to implement stories and commit the code and components to version control

\* **Build** describes the practices needed to create deployable binaries and merge development branches into the trunk

\* **Test end-to-end** describes the practices necessary to validate the solution

\* **Stage** describes the practices necessary to host and validate the solution in a staging environment before production

A15:

#### Explanation

https://www.scaledagileframework.com/devops-practice-domains/

Version Control

=============

Version Control stresses minimizing the amount and duration of open branches and ensuring that the code base always remains in a deployable state. Automation in this category is directed toward optimizing branching and merging strategies, facilitating distributed version control, and managing file types of all kinds.

Continuous Quality

===============

Continuous Quality includes specific practices like hypothesis-driven development, behavior-driven design (BDD), test-driven development (TDD), A/B testing, and exploratory testing. Automation is used to enhance the speed and accuracy of testing throughout the value stream

DevOps practice domain (technical practices and tools)

===========================================

\* Value Stream Management

\* Continuous Quality

\* Continuous Security

\* Version Control

\* Configuration Management

\* Infrastructure Management

\* Agile Planning and Design

\* Deployment Pipeline

\* Continuous Monitoring

\* Agile Product Management

\* Value Metrics

A16:

#### Explanation

**Respond and recover**

► Proactively detect and resolve production issues before they cause business disruption

► Skills:

-Chaos engineering

- Proactive detection

-Cross-team collaboration

- Session replay

- Rollback and fix forward

- Immutable infrastructure

- Version control (see Build)

**Netflix Example:**

This was our philosophy when we built Chaos Monkey, a tool that randomly disables our production instances to make sure we can survive this common type of failure without any customer impact . The name comes from the idea of unleashing a wild monkey with a weapon in your data center (or cloud region) to randomly shoot down instances and chew through cables-all the while we continue serving our customers without interruption.

A17:

#### Explanation

"https://www.scaledagileframework.com/release-on-demand/

**Canary releases** – These provide a mechanism for releasing the solution to a specific Customer segment and measuring the results, before expanding and releasing to more customers.

A18:

#### Explanation

During a DevOps initiative people who are open to experimentation and learning are preferred over others who work alone, asks for proof or good in following processes.

A19:

#### Explanation

DevOps teams don't have individual accountabilities. They provide shared services to the wider organization.

A20:

#### Explanation

DevOps extends beyond software developers and IT operations to include business stakeholders, clients, end users, suppliers, etc.

A21:

#### Explanation

There is no risk in delivering incremental changes. Also the cost and time are greatly reduced by delivering incrementally. Hence this option is NOT correct.

A22:

#### Explanation

The purpose and intent ofDevSecOpsis to build on the mindset that"everyone is responsible for security"with the goal of safely distributing security decisions at speed and scale to those who hold the highest level of context without sacrificing the safety required.

A23:

#### Explanation

https://www.scaledagileframework.com/continuous-integration/

**The Four Activities of Continuous Integration**

\* **Develop** describes the practices necessary to implement stories and commit the code and components to version control

\* **Build** describes the practices needed to create deployable binaries and merge development branches into the trunk

\* **Test end-to-end** describes the practices necessary to validate the solution

\* **Stage** describes the practices necessary to host and validate the solution in a staging environment before production

A24:

#### Explanation

Chaos Monkey is a tool used to introduce intentional disruption to the infrastructure so that teams are equipped to handle such situations more effectively in the future.

A25:

#### Explanation

**Lean startup**

► The lean startup cycle focuses on identifying the viability of ideas

► It follows the plan-do-check-adjust cycle (POCA)

►MVP - establish a baseline to test assumptions and gather objective data

►Evaluate the hypothesis:

    - If the benefit hypothesis has been proven true , the Value Streams will implement more Features

    - If the hypothesis is proven false , then a decision is made to either pivot with a new hypothesis or to stop work on the Epic

A26:

#### Explanation

"https://www.scaledagileframework.com/test-driven-development/

We never have enough time for testing, so let’s just write the test first. —Kent Beck Test-Driven Development Test-Driven Development (TDD) is a philosophy and practice that involves building and executing tests before implementing the code or a component of a system.

Beck and others have defined a set of Extreme Programming (XP) practices under the umbrella label of TDD:

1 - Write the test first, ensuring that the developer understands the required behavior. This could be a new test or a modification of an existing test.

2 - Run the test and watch it fail. Because there’s no code yet, this may seem pointless, but it accomplishes two useful objectives: it verifies the test works, including any testing harnesses and demonstrates how the system will behave if the code is incorrect.

3 - Write the minimum amount of code needed to pass the test. If it fails, rework the code or the test until it passes routinely.

4- Continue implementing new code until all tests pass. This step provides the developer with the confidence that their changes meet the current requirements and haven’t created an error in another part of the system.

5 - Refactor as necessary to ensure the design aligns with changing requirements (e.g., emergent design). Developers continually update their designs to ensure that evolving requirements and a growing codebase don’t lead to poor code quality.

A27:

#### Explanation

Development delays, overly tight architecture, complex or bureaucratic processes are examples of commonly occurring constraints.

A28:

#### Explanation

**Percent Complete and Accurate (%C&A)** ~ Percent of work that the next station could process as-is. Ask people responsible for the next step about the % complete and accurate of the current step.

A29:

#### Explanation

All the three options are characteristics associated with Continuous Integration

A30:

#### Explanation

**Gated commit**– Committing to a single trunk is risky, as broken changes can impact many teams. This is why only the changes that have been validated through the build and test process are merged into the trunk.

A31:

#### Explanation

People related issues are the biggest challenge for any organization that is implementing DevOps transformation.

A32:

#### Explanation

Capacity testing is used to find out the maximum number of users/sessions that can be supported by a product/application. Hence it falls under Non-Functional testing.

A33:

#### Explanation

**Visualize the Value Stream**

► Step 1: On the current state section of the canvas, model your existing value stream using sticky notes. Start at the first step and finish at the last step identified previously.

► Step 2: Identify the role or team responsible for each step.

A34:

#### Explanation

https://www.scaledagileframework.com/continuous-exploration/

**Synthesize** describes the practices that organize the ideas into a holistic vision, a roadmap, a prioritized program backlog, and supports final alignment during PI Planning

Defining a backlog with clearly written features – Clearly defining features that fit in a PI is critical for ARTs to align on what is needed and for teams to be able to plan. The backlog also reflects important security requirements.

A35:

#### Explanation

<https://www.scaledagileframework.com/continuous-deployment/>

**Deploy to Production**

Ideally, the deployment pipeline triggers the deployment process automatically following a successful build, integration, and validation. This makes the entire workflow, from code-commit to production-deploy, a fully-automated “one-click” process. Highly sophisticated enterprises can reliably deploy any time of day, any day of the week, and any week of the year—even during peak periods.

A36:

#### Explanation

All the factors mentioned here are applicable in a DevOps transformation initiative.

A37:

According to lean principles 'value' is always defined from end customer/user's perspective.

A38:

#### Explanation

All the mentioned factors pose challenges and risks for DevOps transformation initiative.

A39:

#### Explanation

**Lean startup**

► The lean startup cycle focuses on identifying the viability of ideas

► It follows the plan-do-check-adjust cycle (POCA)

► MVP - establish a baseline to test assumptions and gather objective data

► Evaluate the hypothesis:

      - If the benefit hypothesis has been proven true , the Value Streams will implement more Features

      - If the hypothesis is proven false , then a decision is made to either pivot with a new hypothesis or to stop work on the Epic

A40:

#### Explanation

In a DevOps culture the best way to determine the frequency of release windows is to talk to the business. Because the deployment frequency varies from industry to industry. For example frequency can be more in e-commerce versus a highly regulated industry (finance, healthcare, etc.).

A41:

#### Explanation

In relation to other options provided systems engineering that spans Dev and Ops is a critical success factor for DevOps initiative.

A42:

#### Explanation

**The problem to be solved**

► Every Enterprise must be able to quickly validate the wants and needs of Customers

► Every Enterprise must be able to release Features when Customers need them the most

A43:

#### Explanation

Shortened feedback loops help the teams to gather feedback quickly and also act upon them quickly. Thus the teams improve themselves continuously over a period of time.

A44:

#### Explanation

Training, continuous skill improvement, management commitment to culture change are critical success factors for DevOps transformation.

A45:

#### Explanation

**Value stream mapping metrics include following:**

\* PT = processing time (how long does it take to perform the process)

\* LT = lead time (what is the latency between initiation and execution of a process)

\* AR = activity ratio = PT / LT \* 100

\* %C&A = % complete and accurate (determine in % if we needed to return to the previous process to re-ask and update information before we start our process)

**Now perform a quick calculation:**

  \* Total PT= sum of all PTs per process

  \* Total LT = sum of all LTs per process

  \* Rolled AR = product of all ARs = 28/100 \* 35/100 \* 87.5/100 \* …

  \* Rolled %C&A= product of all %C&A = 80/100 \* 71/100 \* 80/100 \* …

EXAM4

A1:

#### Explanation

<https://v5.scaledagileframework.com/continuous-exploration/>

A2:

**Explanation**

Since the organization is facing both 'delay' and 'quality' issues the BEST approach is to balance the release frequency and quality of each release.

A3:

#### Explanation

Best DevOps practices in release and deployment management encourages business to determine the release frequency and also automating the environment provisioning.

A4:

#### Explanation

<https://v5.scaledagileframework.com/continuous-delivery-pipeline/>

A5:

#### Explanation

All the characteristics mentioned here are applicable for any DevOps professional.

A7:

#### Explanation

International Standards Organization (ISO) defines various public standards around quality management. ISO is not involved in the subject of DevOps.

A8:

#### Explanation

Value Stream Maps helps an organization to define the existing flow of activities and information. Then waste and other inefficiencies can be removed from the value stream.

A9:

#### Explanation

Culture, Automation, Lean, Measurement and Sharing are the five values of DevOps.

A10:

#### Explanation

Both 'Deployment Frequency' and 'Mean Time To Recover (MTTR)' helps in demonstrating the IT benefits of DevOps. Other metrics are Deployment Time and Deployment Success Rate.

A11:

#### Explanation

Rugged software development is a cultural approach to creating available, survivable, defensible, secure, and resilient software.

A12:

#### Explanation

All the above mentioned benefits can be achieved by integrating DevOps practices and IT Service Management practices.

A13:

#### Explanation

Continuous Delivery helps in frequent release of product increments. Thus business benefits like faster time to market, reduced risk/cost are achieved.

A14:

#### Explanation

All the mentioned DevOps practices are aligned with ITSM Change Management processes.

A15:

#### Explanation

In today's highly disruptive business environment organizations need both rapid responsiveness to change in the form of frequent releases and stability in the form of quality deployments.

A18:

#### Explanation

In high performing organizations 'trust' plays a vital role. Culture is the first step in DevOps transformation. Then comes tools, processes and automation.

A20:

#### Explanation

Continuous Integration ensures that code is committed multiple times in a day by the development team members. All code changes are integrated in to a common shared repository.

A21:

#### Explanation

In Theory of Constraints (ToC) identifying the constraint precedes exploiting the constraint.

A22:

#### Explanation

<https://v5.scaledagileframework.com/continuous-delivery-pipeline/>

A23:

#### Explanation

IT must understand the organization's purpose, cause and belief so that they can align themselves with the organization's vision to avoid silo culture.

A24:

#### Explanation

Out of the provided options, option C defines DevOps in the correct context which is about communication / collaboration between the Dev and the Ops teams.

A27:

#### Explanation

End to end use case testing is an example for 'shift left' approach. The practice of Continuous Integration (CI) can be used to achieve the same. In CI the automated acceptance test cases are executed every time code is changed in version control system.

A28:

#### Explanation

<https://v5.scaledagileframework.com/behavior-driven-development/>

Behavior-Driven Development

A29:

#### Explanation

**The Four Activities of Continuous Deployment**

SAFe describes four activities of Continuous Deployment:

\* Deploy to production describes the practices necessary to deploy a solution to a production environment

\* Verify the solution describes the practices needed to make sure the changes operate in production as intended before they are released to customers

\* Monitor for problems describes the practices to monitor and report on any issues that may arise in production

\* Respond and recover describes the practices to rapidly address any problems that happen during deployment

https://www.scaledagileframework.com/continuous-deployment/

A30:

#### Explanation

DevOps teams use shared tools to communicate/collaborate. Also they apply automation wherever possible.

A31:

#### Explanation

Manual testing and Testing in production are examples of 'shift right' approach. In DevOps organizations performance testing is done continuously throughout the delivery cycle. Hence it is an example of the 'shift left' approach.

A32:

#### Explanation

ChatOps is a DevOps practice where multiple teams collaborate in chat rooms of tools like Slack, HipChat, etc. Conversations between the Dev and Ops stakeholders will drive faster development, frequent delivery/deployment and effective support.

A35:

There is currently no ‘industry recognized’ job description or formal career track for a DevOps Engineer.

A37:

#### Explanation

<https://v5.scaledagileframework.com/devops/>

A39:

#### Explanation

The Four Activities of Continuous Deployment

SAFe describes four activities of Continuous Deployment:

\* Deploy to production describes the practices necessary to deploy a solution to a production environment

\* Verify the solution describes the practices needed to make sure the changes operate in production as intended before they are released to customers

\* Monitor for problems describes the practices to monitor and report on any issues that may arise in production

\* Respond and recover describes the practices to rapidly address any problems that happen during deployment

https://www.scaledagileframework.com/continuous-deployment/

A40:

#### Explanation

The purpose of NewOps is NOT to remove the need of human competencies. But to use the human competencies for better decision making, creating more automation, client interactions, etc

A41:

#### Explanation

Product Manager role belong to the Development stakeholders. Database Administrator, Release Engineer and Information Security Engineer roles belong to the Operations stakeholders.

A43:

#### Explanation

Goal of DevOps is to provide frequent quality deployments via communication and collaboration between Development and Operations teams

A44:

#### Explanation

Continuous Delivery encourages keeping the product in releasable state. Thus continuous feedback on production readiness is achieved.

A45:

#### Explanation

Introducing security as code and shift left strategy helps an organization to implement DevSecOps and Rugged DevOps.