PLUTORA

WHITE PAPER

Managing Risk
Using Quality Metrics in Value Streams

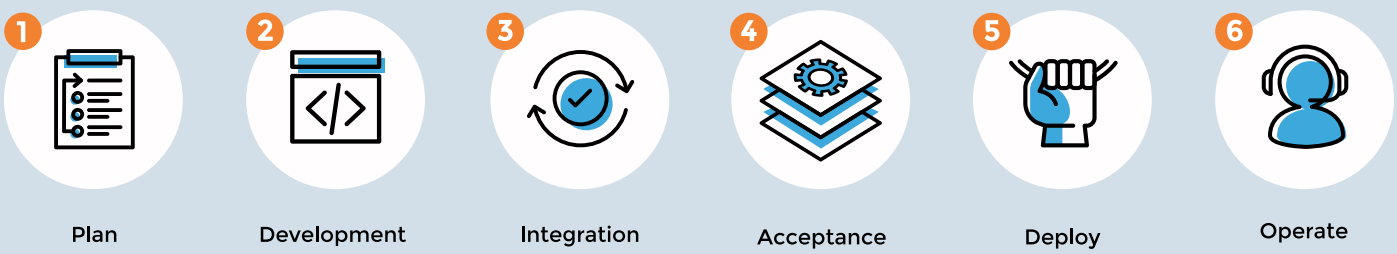
- What are the effective quality metrics for mitigating risks with DevOps value streams
- An example project illustrating how risks were managed using metrics for a real DevOps value stream

Organizations that do not take risks are doomed to fall behind more innovative competitors. Risks are necessary but need to be managed strategically. This white paper is concerned with using quality metrics as part of a strategy to manage risks which affect the **possibility of suffering business losses** due to failures in the DevOps value stream.

Multiple engineering steps are required to move a product from idea to production—a chain of processes steps in which each step adds value, but also adds the possibility of risks. A DevOps continuous integration (CI) / continuous delivery (CD) pipeline implements a value stream for software delivery. Figure 1—DevOps Value Stream Example illustrates this idea. This white paper explains:

- How DevOps affects the **choice of metrics** for managing business risks in value streams.
- **Recommended practices** to engineer risk management using quality metrics in DevOps value streams.
- A strategy to **set risk priorities** and select quality metrics for risk management in value streams.
- **An example project** illustrating how risks were managed using metrics for a real DevOps value stream.
- A **list of effective quality metrics** for mitigating risks with DevOps value streams.

Figure 1: DevOps Value Stream Example



WHY is managing risk using quality metrics in value streams important?

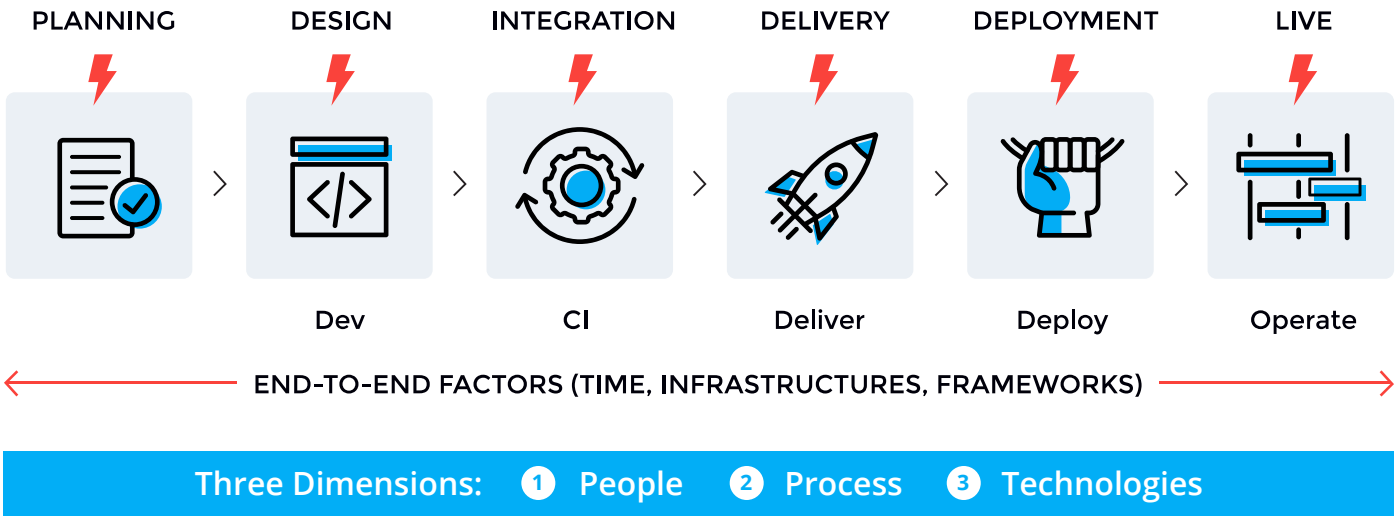
For the purpose of this paper, risk is defined as the possibility of suffering business losses due to failures in the DevOps value stream. Examples of such risks include:

- **Quality:** problems being introduced to products and services due to development of poor-quality software
- **Costs:** for software development and production can cause increased production cost.
- **Time:** the possibility of not being able to complete project on time.

In general, risks are caused by several types of value stream management deficiencies: lack of information, lack of control and/or lack of time related to actions by people, processes and technologies at each stage in the value stream. Figure 2 illustrates the sources of risks in DevOps value streams.

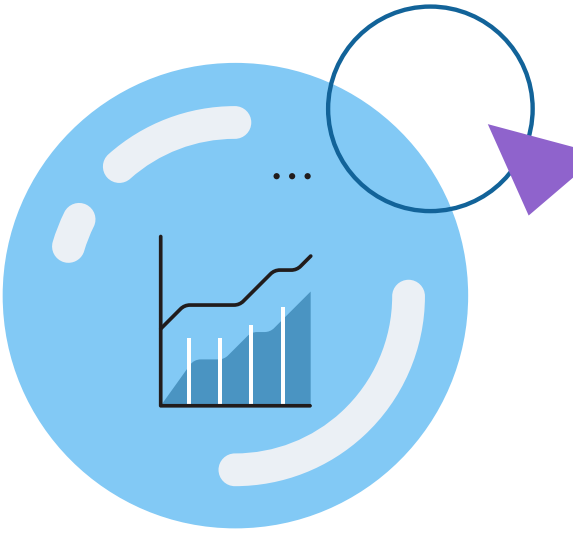
The primary goal of DevOps value streams is typically to accelerate the lead time for software changes. This is accomplished by removing time consuming bottlenecks in the value stream. If changes in the DevOps value stream do not adequately address people, processes and technologies considerations then risk is increased.

Figure 2: Sources of Risks in DevOps Value Streams



The following characteristics are particularly important for selecting quality metrics for managing risk with DevOps value streams:

- **DevOps aims to continuously improve** agility, stability, efficiency, quality, security, and satisfaction. These goals can conflict with each other, so it is important to have metrics that verify each goal stays within business requirements.
- **Development teams take more responsibility** for testing and QA. Metrics that verify that developers' code check-ins meet quality goals are important.
- **Customer product experience must be monitored continuously.** With DevOps, changes to production happen often and the customer experience may change quickly.
- The user experience of DevOps tools must be monitored continuously. With DevOps, **the DevOps toolchain becomes a critical 24/7 high availability requirement.**
- With DevOps the **Mean-Time-to-Restore-Service (MTRS)** is more important than Mean Time Between Failures (MTBF).
- **Increased technology complexity** such as federated micro-service networks, toolchains, and value stream portfolios indicate that metrics that validate activities across multiple pipelines is important.
- **Quality and security checks occur earlier in the lifecycle,** so it is important to have quality metrics providing visibility to early stages in the value stream.



HOW are risks managed using quality metrics in value streams?

Organizations that strive to innovate and compete must adopt a risk management orientation in which risk management is part of the culture. In this culture, teams must work to identify actionable business risk priorities.

Organizations with a strong risk management culture focus on contextual quality metrics (e.g., coverage of requirements by risk priority level) rather than "counting" metrics (e.g., number of tests).

Measurement of end-to-end (E2E) user experience is favored over application-specific or team-specific metrics. Monitoring and tracking high risk trends is used to proactively monitor the most important risk factors.

The book "Engineering DevOps – From Chaos to Continuous Improvement...and Beyond"[1] identifies "Nine Pillars of DevOps", as illustrated in Figure 3.1.

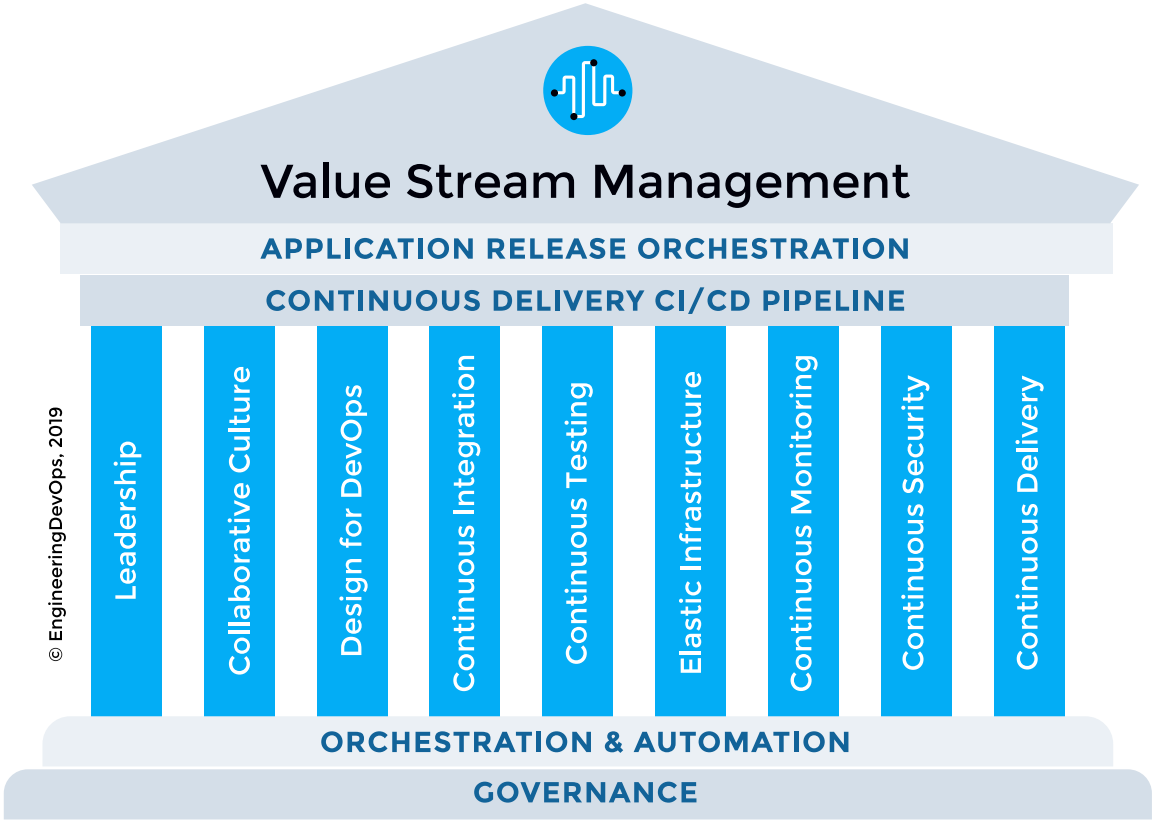
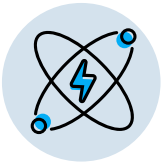


Figure 3.1: Nine Pillars of Devops



LEADERSHIP

Participation in DevOps training and incentives, decision response times, business risk priorities

CULTURE

Requirements for QA, infrastructure and deployment in backlog, response times to requests and defect reports

DESIGN

High risk problems with cause design, build success rate, Static Code Analysis (SCA) reports and trends, unit test success rate, unit coverage, number of code branches

INTEGRATION

Build and test success rates, check-ins include automated tests, image scan trends, requirements covered by API and functional tests

TESTING

E2E test coverage, percent of E2E tests automated, percentage passing trends, test planned coverage, risk priority identified

INFRASTRUCTURE:

Failure reports, infrastructure test results, Chaos test report trends

MONITORING

Application performance monitoring trends, log reports with high priority exception cause

SECURITY:

Scans, failure trends, event tickets

DELIVERY

Sire Reliability (SRE) practices with Service Level Objectives (SLOs) and error budgets, successful deployments. use of blue/green and progressive rollouts

In addition to the vertical pillars, shown in Figure 3.1, there are horizontal structures with E2E quality metrics that are effective for managing risk as shown below:

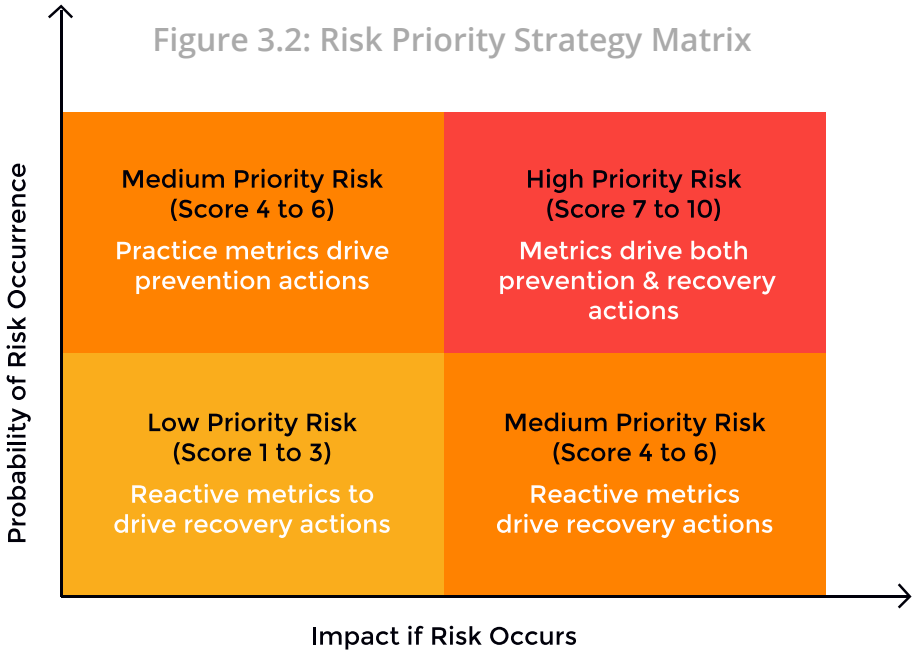
DEPLOYMENT METRICS
Time-to-deploy, deployment frequency, deployment success/failure, time spent fixing failed release environments, configuration drift. E.g. Deploy on demand

LEAD TIME METRICS
Individual productivity/ velocity, rework time, cycle time, time-to-value trends. E.g. < 1 hour

MTRS METRICS
Broken build time, time to triage, time to investigate, time to remediate. E.g. < 1 hour

As indicated in the prior paragraphs there are many possible DevOps quality metrics that are effective to manage risk but measuring all possibilities would only create "noise" and can be prohibitively expensive for most organizations to implement. A strategic approach is needed to determine an affordable and effective set of quality metrics that are most suitable for the specific application and organization priorities.

Figure 3.2: Risk Priority Strategy Matrix



This is an example of how to set risk priorities in accordance with a strategy. It should be noted that there is no standard approach. Organizations may have different ways of assigning risk priorities.

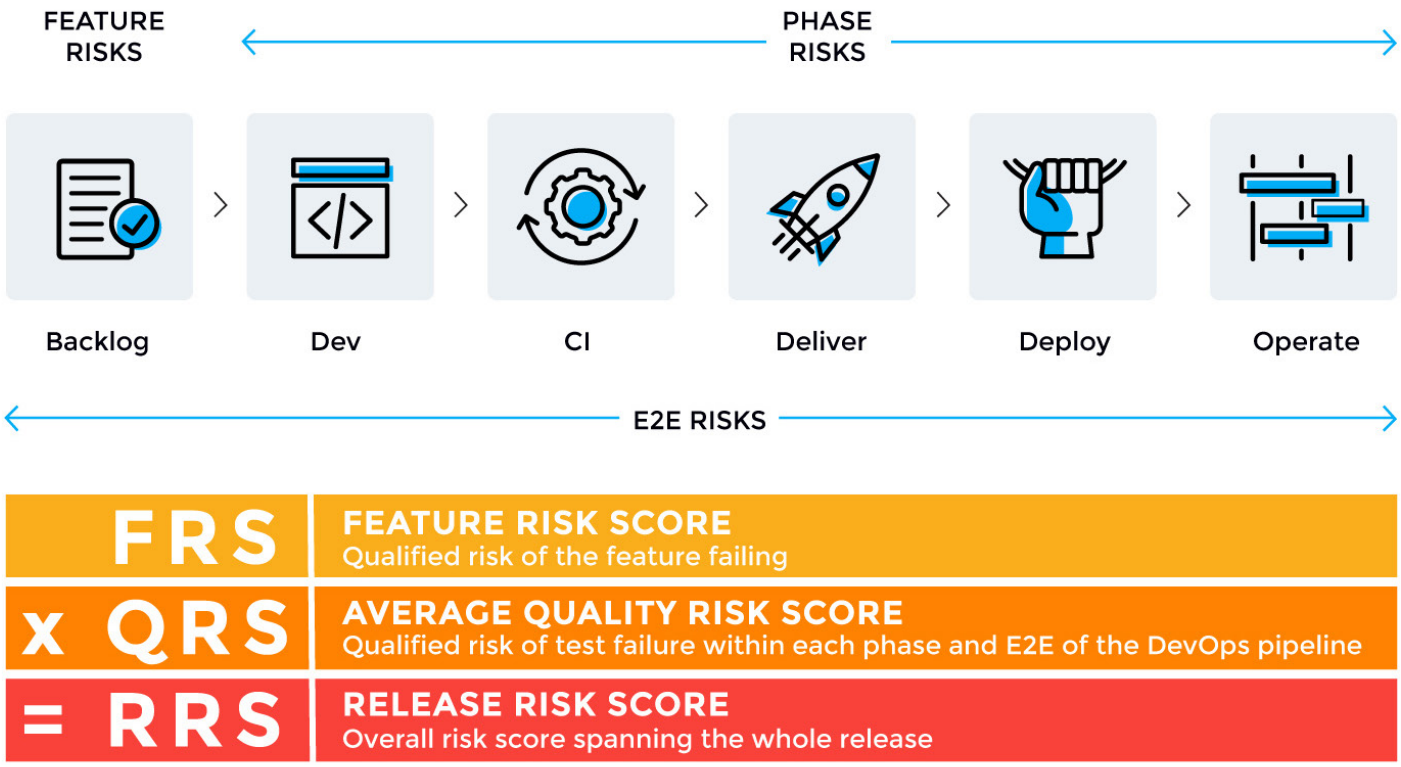
The lower left quadrant represents risks with low impact and low probability. The strategy for these is to assign low priority risk scores such as score 1 to 3. The quality metrics strategy for this quadrant include using reactive metrics that provide information to drive recovery actions. An example of such a quality metric could be the reporting of low priority defects occurring in part of the system that is not critical. If such a defect is detected, then the system would detect it and restart only the instance of the system that experienced the defect.

The lower right quadrant represents risks of high impact but low probability of occurrence. Usually the systems design strategy for this quadrant is to transfer such risks from the system to another system that can deal with the risk more effectively. These types of risks are labelled medium priority with scores in the range 4 to 6. The metrics strategy for this quadrant is to select and use reactive metrics that drive recovery actions. An example of such a quality metric would be a failure of a high priority test in the API connecting the system to another system that is not used very often. If such a defect is detected, then the recovery action would restart only the instance of the system that experienced the defect.

The upper left quadrant represents risks with low impact but high probability. These are also labelled medium priority with risk priority scores 4 to 6. The strategy for quality metrics in this quadrant is to use proactive metrics to drive prevention actions. An example of such a metric is a Feature Risk Priority (FRP) assigned during backlog planning. The action would be to make sure that QA tests for any feature tagged as high risk are fully implemented and validated prior to integration.

The upper right quadrant represents risks with high probability and high impact. These are the highest priority risks assigned risk priority score 7 to 10. In this case the strategy for quality metrics is to prevent the impacts by using metrics to drive preventative actions and also drive recovery actions a risk event occurs. An example of such a quality metric is MTRS. The trend of MTRS is used to drive actions to speed up recovery, and when an event occurs the actions are to determine the fastest way to implement recovery.

Figure 3.3: Risk Management Scoring



It is useful to score the risk of releasing each feature, to simplify release decisions. While there is no standard approach, each organization can develop their own method for assigning a **Release Risk Score (RRS)** for a feature. The following is an example approach.

- Assign a Feature Risk Score (FRS) to each feature that is being changed. For example use the Risk Priority Strategy Matrix shown in Figure 3.2
- Assign a Quality Risk Score (QRS) to each test failure
- For each feature, the RRS can be calculated as the FRS X average QRS

Figure 3.4 illustrates a set of quality metrics used to help manage risk for a DevOps value stream.

The metrics shown are a selection of reactive and proactive metrics for the 9 pillars and horizontal end to end structures listed earlier in this section.

Figure 3.4: Creating Risk Metrics

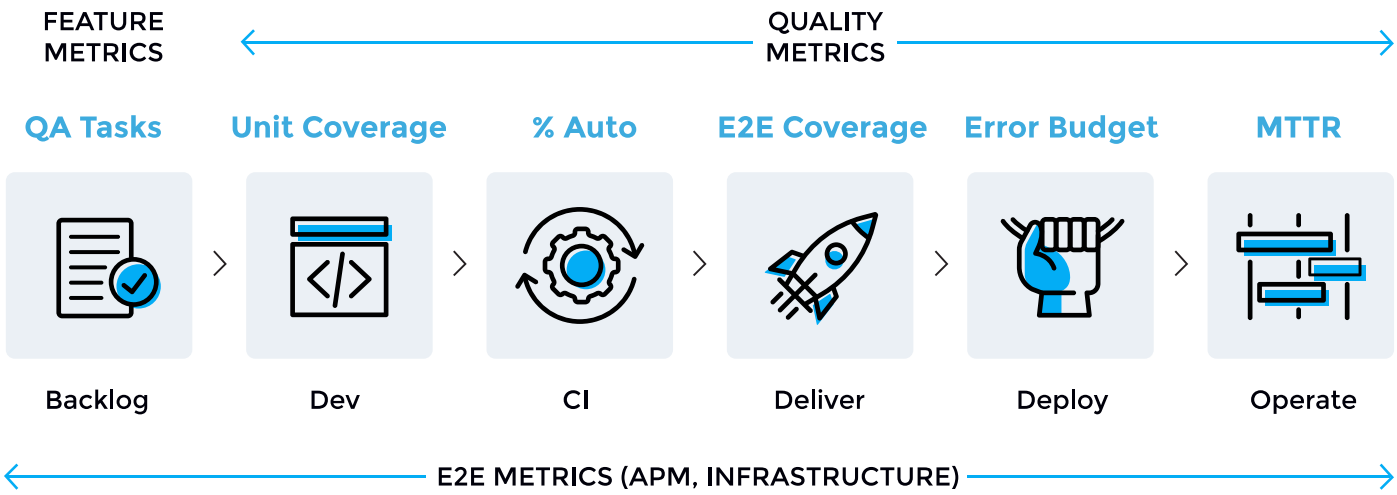
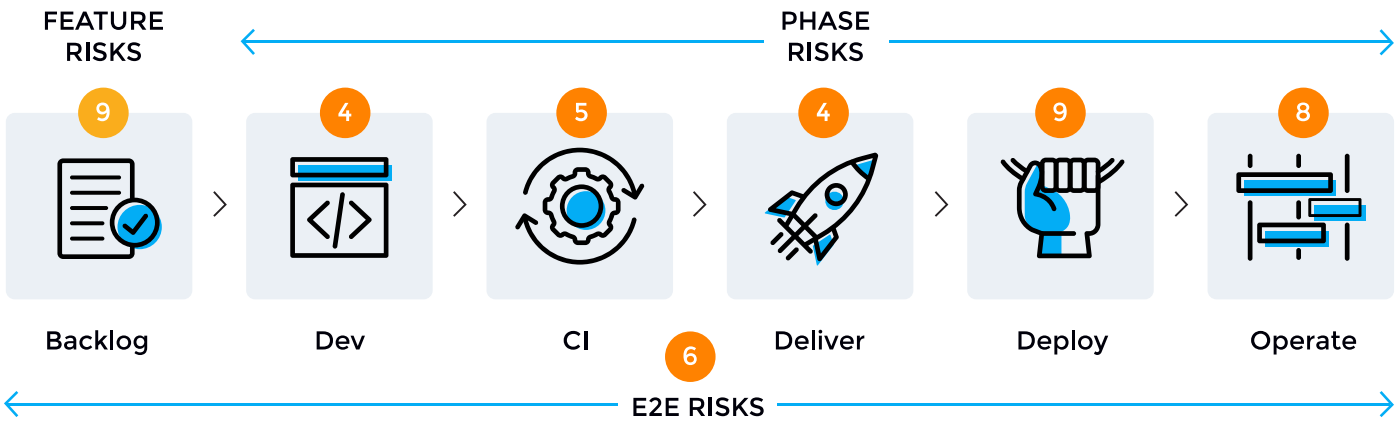


Figure 3.5: Risk Management Scoring Example

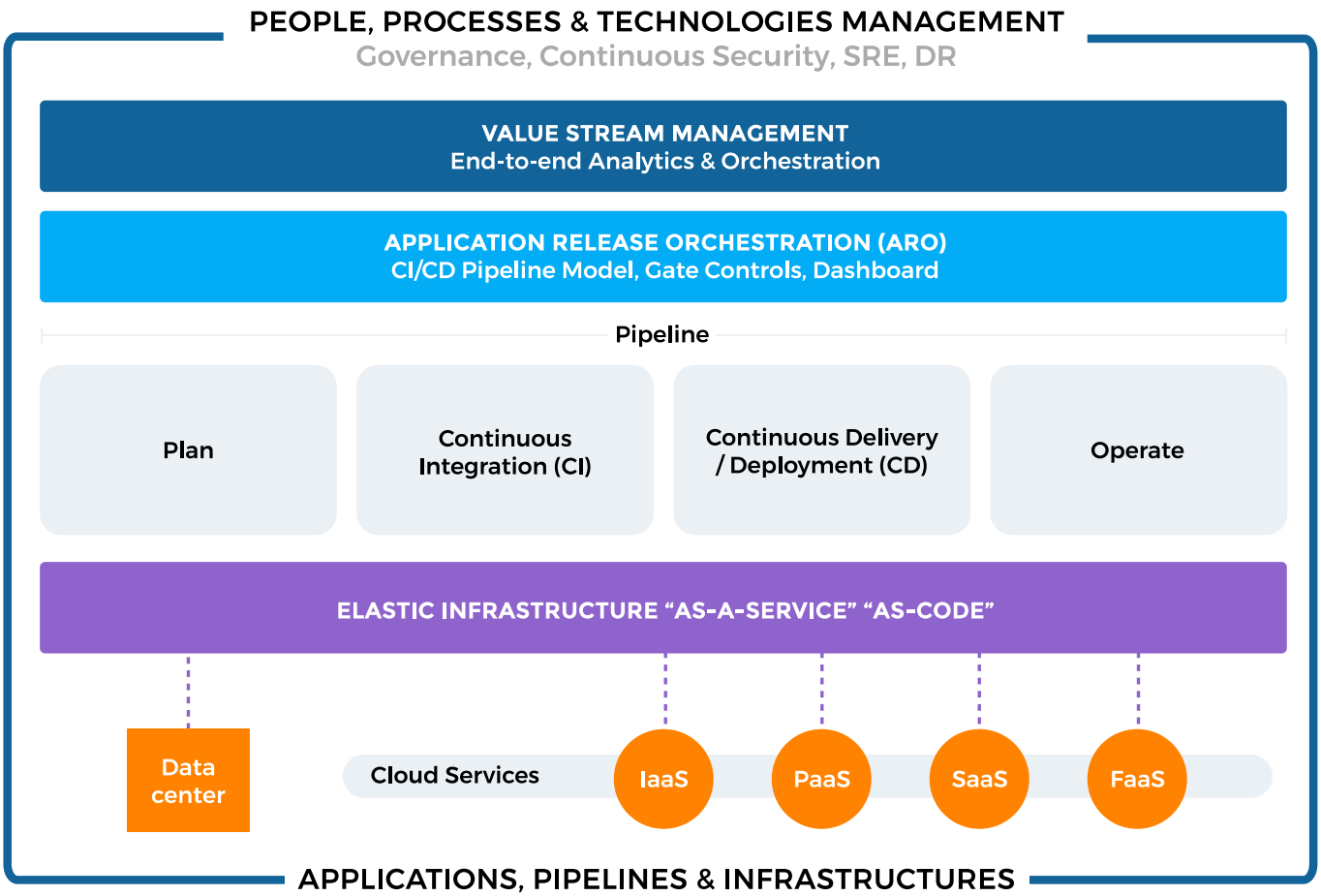


FRS	9
x QRS	AVG (4, 5, 4, 9, 8, 6) = 6
= RRS	54

Tools and engineering practices are needed to implement the strategy for managing risks using quality metrics.

Figure 3.6 DevOps Engineering Blueprint[1] illustrates that a well-engineered DevOps system requires management and governance capabilities presiding over multiple interacting tools configured as a CI/CD toolchain, all supported by infrastructures.

Figure 3.6: DevOps Engineering Blueprint



Examples of useful tools for implementing quality metrics are:

- Value stream management platforms (VSMPs) with the capability to ingest quality metrics data from other tools in the toolchain, provide analysis and orchestrate actions depending on the result of the analysis. An example of such a platform is Plutora.
- Ticket systems with cause and risk priority fields.
- Quality trend reports and dashboards which can convey metrics and risk priorities.
- System-generated reports logs that can be consumed by analysis tools.

Tools alone are not enough. The practices for using the tools to implement the chosen metrics and to indicate risk priorities are equally important. The following practices are especially useful for using quality metrics to manage risk in DevOps value streams:

- Set Service Level Objectives (SLOs) which define response time goals for metrics with exceptions and consider risk priorities. Refer to Site Reliability Engineering practices for best practices concerning SLOs and error budgets.
- Codify the entry/exit criterion between each stage in the value stream.
- Codify the release criterion using quality metrics data and risk priorities.

This section covered HOW risks are managed using quality metrics in value streams. The next section will cover WHAT is needed to accomplish this for your organization.

WHAT is needed to accomplish risk management using quality metrics in value streams?

As indicated in prior sections, a strategic approach is required to select quality metrics and implement them using tools and processes that match specific business goals and maturity level of the DevOps system itself. Figure 4.1 Seven-Step DevOps Transformation Blueprint[1] is a general prescription that illustrates how to approach and evolve any DevOps project strategically.

In accordance with the blueprint, the following is a summary of recommended actions to select, implement and operate quality metrics for risk management:

1. VISIONING

Strategic, Sponsors, Partners

Leadership sponsors quality metrics team to lead actions.

2. ALIGNMENT

Leadership, Team, Applications

Quality metrics team assigns risk priorities – people, process and technologies

3. ASSESSMENT

Discovery, 9 Pillars of Maturity, Deep-Dive, Value Stream Map

Select ACTIONABLE metrics that best match highest risk priorities – pillars and structural

4. SOLUTION

Future State, Roadmap, Epics, Realignment

Select a value stream management platform (VSMP) to perform a backbone for metrics tools and analytics

5. REALIZE

Projects, Stories, Tasks, POC, Validation,Training, Deployment, Governance

Select tools and coding projects that fit the selected metrics

6. OPERATIONALIZE

Monitor SLI/SLO/SLA, SRE Controls, Retrospectives

Put in place highest priority metrics with a model application

7. EXPANSION

Continuous Flow, Enterprise Adoption, Continuous Feedback, Continuous Improvement

Operationalize, measure, iterate.

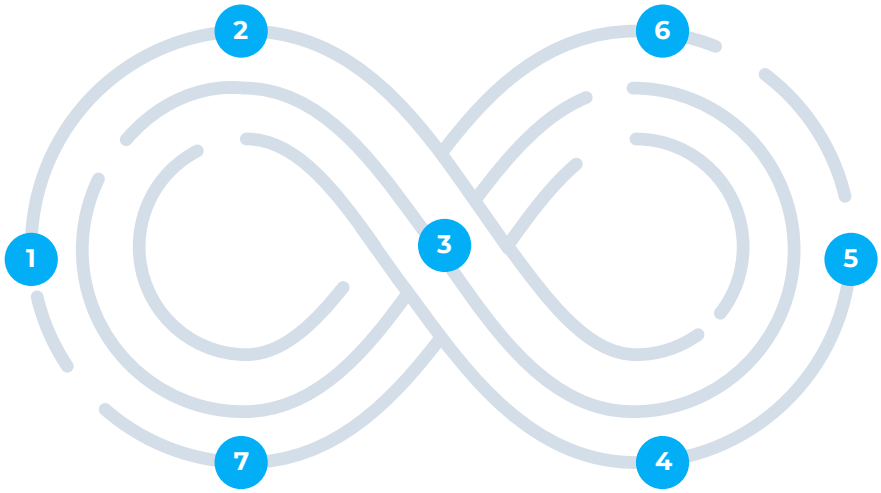


Figure 4.1: Seven-Step DevOps Transformation Blueprint

Here are some pitfalls to watch out for when setting strategies for quality metrics:

- Too many metrics can be as bad or worse than not enough metrics.
- Do not invest in specific tools until the metrics strategy and risk priorities are decided or you may find you have the wrong tools and no budget left for the right ones.
- Metrics without assigned risk priorities do not allow for adequate discrimination.
- Metrics without specific actions are meaningless.
- Do not assume the same set of metrics are the right ones for every application.
- Do not keep metrics in place if they are not yielding actions.

The choice and use of individual quality metrics for risk management are likely to evolve for each application as the application and the DevOps system supporting the application matures. Figure 4.2 illustrates a general approach to evolving DevOps where feedback, derived from metrics data, is continuously gathered and used to iterate the application and DevOps pipeline as the DevOps system matures.

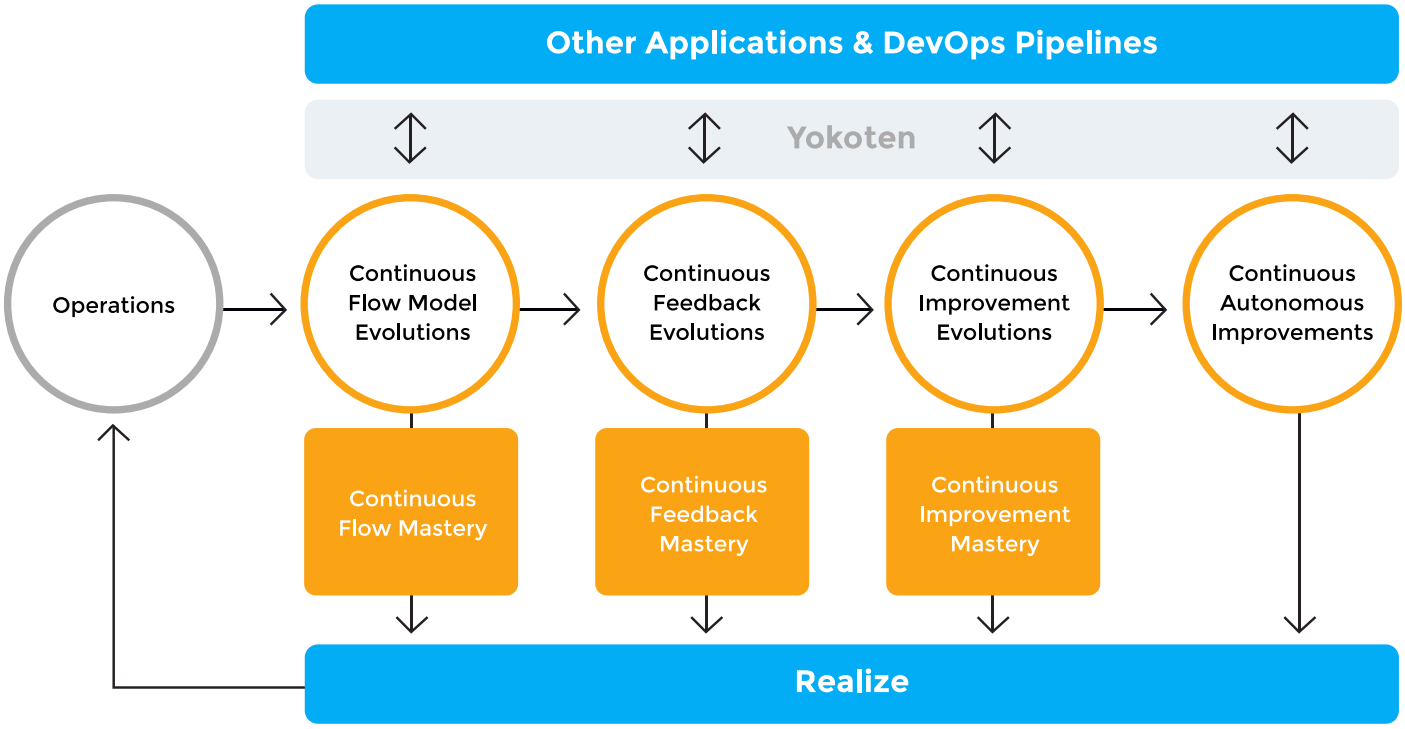


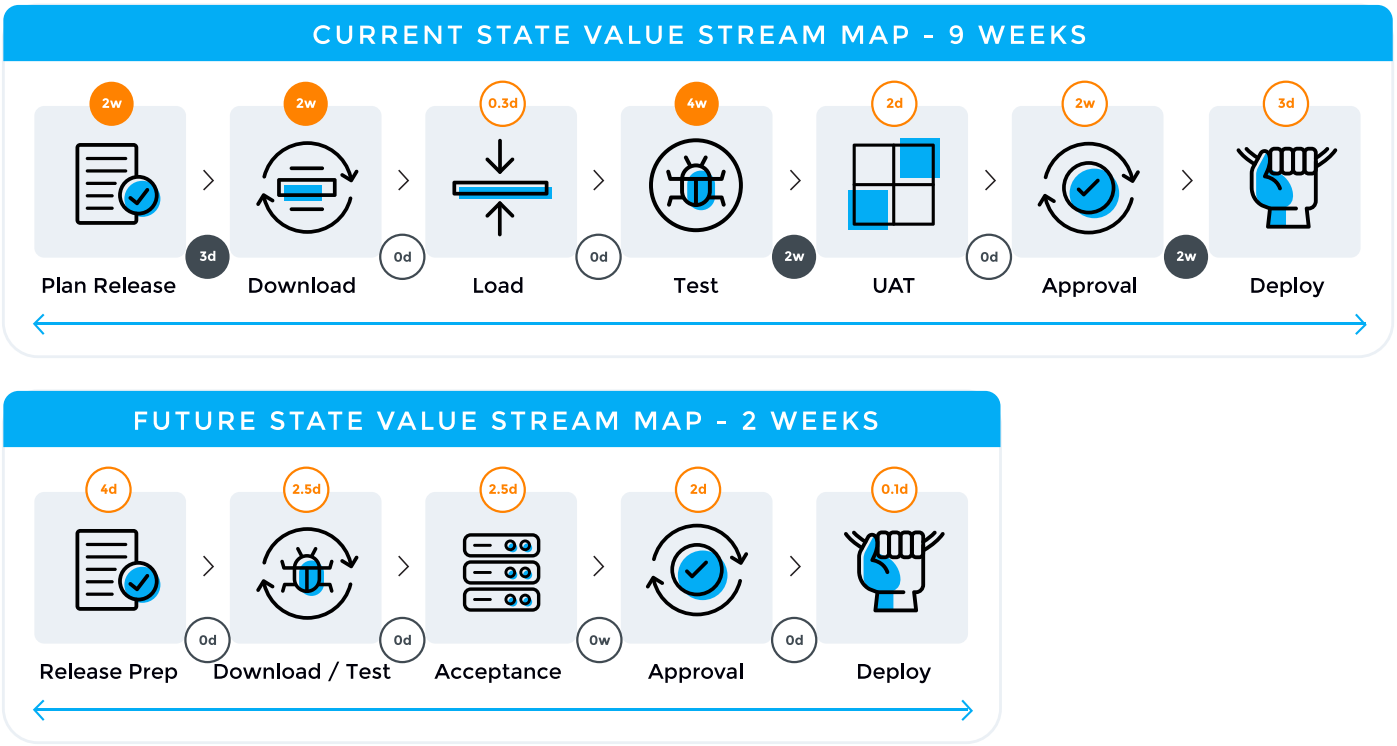
Figure 4.2 Continuous Improvement with DevOps

This section explained WHAT is needed to implement quality metrics for managing risks in DevOps value streams. The next section will show an example of applying the information in the prior sections, to realize a set of quality metrics for risk management.

Example Project

A government agency set for themselves a goal for their public facing application to use DevOps to reduce the lead time in their value stream from planning to deployment from 9 weeks to 2 weeks, while managing risks using quality metrics. The solution prescribed included eliminating wasteful emails, meetings and manual tasks, automating overall governance, and adding metrics to improve visibility of the value stream, test environment orchestration, approvals and deployment processes. Figure 5.1 shows a customer example with the current state and future state value stream map. Shortening the end to end value stream from 9 weeks to 2 weeks has the potential to add risk because of the shorter times to process and test changes. Quality metrics were selected to mitigate the risks.

Figure 5.1: Customer Example Project



The high priority risks were determined to be risk of quality problems resulting from inadequate E2E testing, release delays due to bottlenecks in the value stream, and user experience problems resulting from long recovery times. The primary quality metrics that were selected for the first phase of the project are follows:

HIGH PRIORITY RISK:
Quality problems resulting from inadequate E2E testing

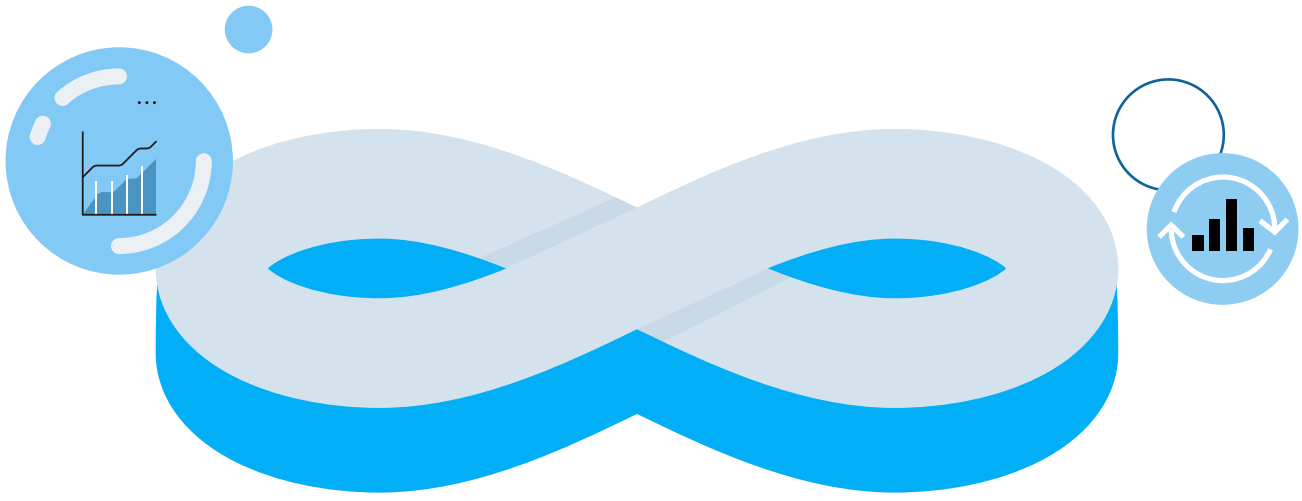
TESTING METRICS:
Percentage pass trends and test planned coverage to measure E2E testing performance

HIGH PRIORITY RISK:
Release delays due to bottlenecks in the value stream

DEPLOYMENT METRICS:
Time-to-deploy, deployment frequency, deployment success/failure, and time spent fixing failed releases environment to measure bottlenecks

HIGH PRIORITY RISK:
User experience problems resulting from long recovery times

MTTRS METRICS:
Broken build time, time to triage, time to investigate, time to remediate to measure recovery times



To implement these metrics, analysis and governance for the DevOps value stream the Plutora value stream management platform was recommended, configured as shown in Figure 5.2.

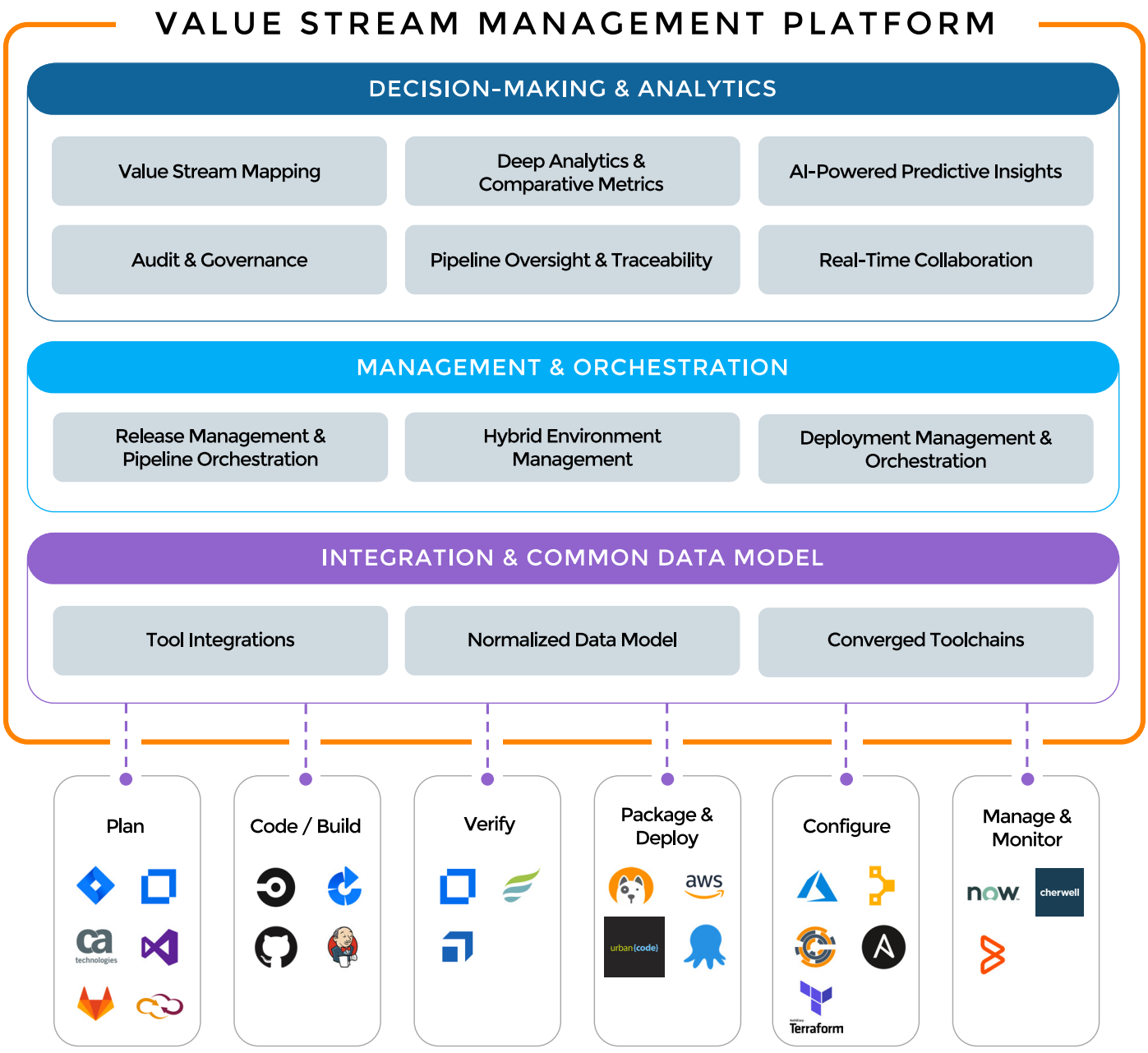


Figure 5.2: Software Application Value Stream Management Platform

Risk scores can be displayed on a value stream management dashboard such as the Plutora Insights dashboard shown in Figure 5.3.

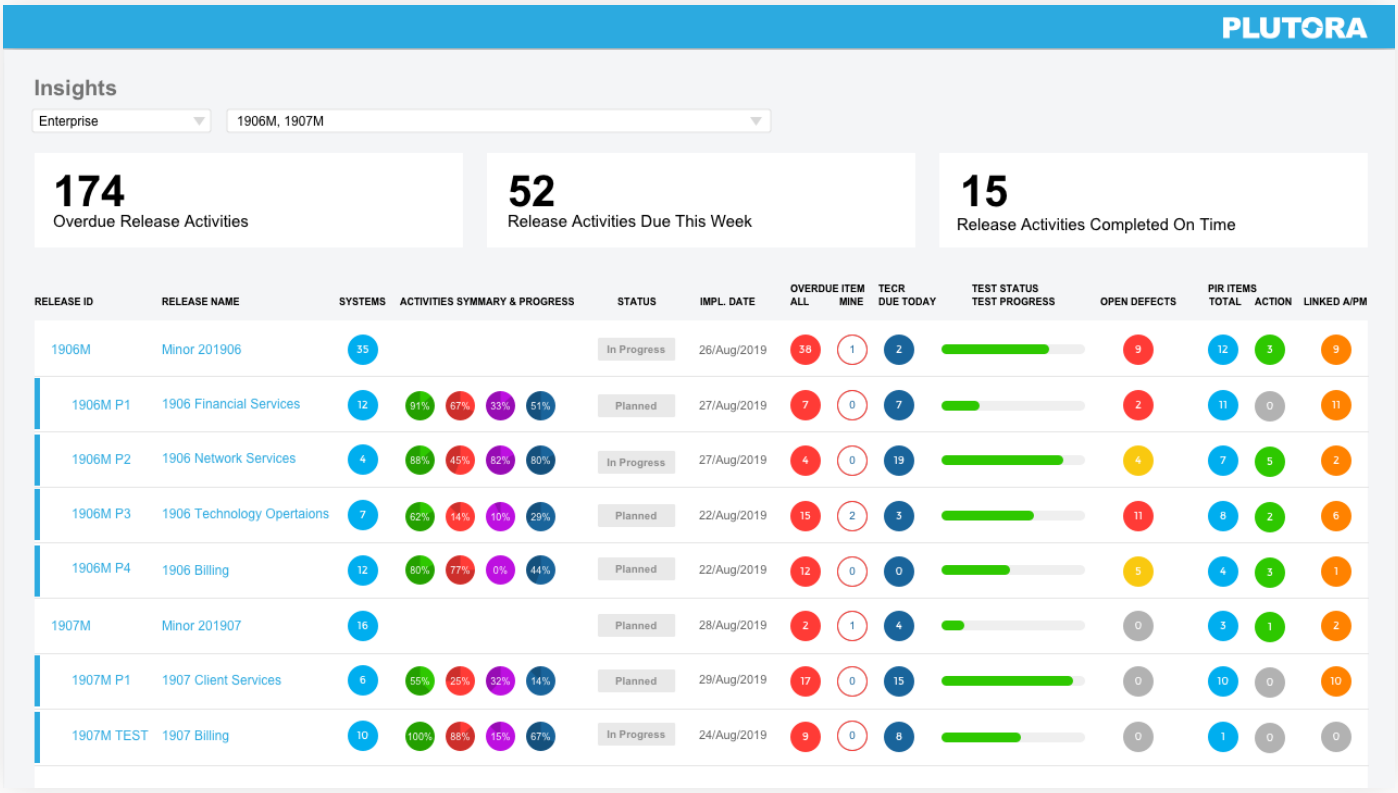


Figure 5.3: Example Value Stream Management Dashboard

Summary

This white paper explained that business risks are necessary but need to be managed strategically. This white paper explained the WHY, WHAT and HOW to select, implement and operate quality metrics to manage business risks for DevOps value streams.

The paper explained a strategic approach to set risk priorities and choose metrics from a list of metrics covering the Nine Pillars of DevOps, three dimensions People, Process and Technology, as well as E2E structures for the DevOps value stream.

The paper explained that it is critical to follow a strategic approach to decide which metrics, how to decide where to invest, how to implement and evolve the solution as the application and DevOps systems mature. Anything short of a strategic approach will likely not be effective.

The implementation of quality metrics requires a value stream management platform, such as Plutora to aggregate, analyze and display risk information.

REFERENCES
1. Hornbeek, Marc. 2019. *Engineering DevOps: From Chaos to Continuous Improvement... and Beyond.* BOOKBABY.

About Plutora

Plutora, the market leader of value stream management solutions for enterprise IT, improves the speed and quality of software creation by capturing, visualizing and analyzing critical indicators of every aspect of the delivery process. Plutora orchestrates release pipelines across a diverse ecosystem of development methodologies, manages hybrid test environments, correlates data from existing toolchains, and incorporates test metrics gathered at every step. The Plutora Platform ensures organizational alignment of software development with business strategy and provides visibility, analytics and a system of insights into the entire value stream, guiding continuous improvement through the measured outcomes of each effort.

PLUTORA

Learn more: www.plutora.com
Email: contact@plutora.com