# THE BEGINNER'S GUIDE TO AGILE API DEVELOPMENT:

## A look at DevOps, CI, and CD in API Delivery

SoapUI NG Pro

SMARTBEAR

## SoapUI NG Pro

SoapUI NG Pro is the “Swiss Army knife” of API testing that lets Agile developers and testers rapidly create and run automated functional, regression, and data-driven tests for SOAP and REST APIs in one environment.

LEARN MORE ABOUT **SOAPUI NG PRO**

# Contents

## Introduction

**API development is changing at a more rapid pace than it ever has in its short history. This pace is also impacted with a series of challenges that can impact an organization's ability to deliver software in a reliable and repeatable way. As new Agile methodologies emerge, such as Continuous Integration, Continuous Delivery and Continuous Deployment, organizations must adapt their delivery strategies or risk falling behind the pace of their peers. But how do you select the right approach for your organization?**

What tools should you use to enable API delivery in an Agile landscape? The evolution of modern software delivery raises many of these questions; the goal of this eBook is to provide clarity by covering the following topics:

- What shifts are occurring in the ways that software is delivered, and how this creates challenges as organizations adapt
- The roles and methodologies associated with DevOps
- The differences between Continuous Integration, Continuous Delivery and Continuous Deployment and where each of these strategies is most applicable
- The importance of automation and tooling to enable modern API development
- How to implement DevOps and Agile – best practices in knowledge management, tool selection

## Modern API Development Landscape & Challenges

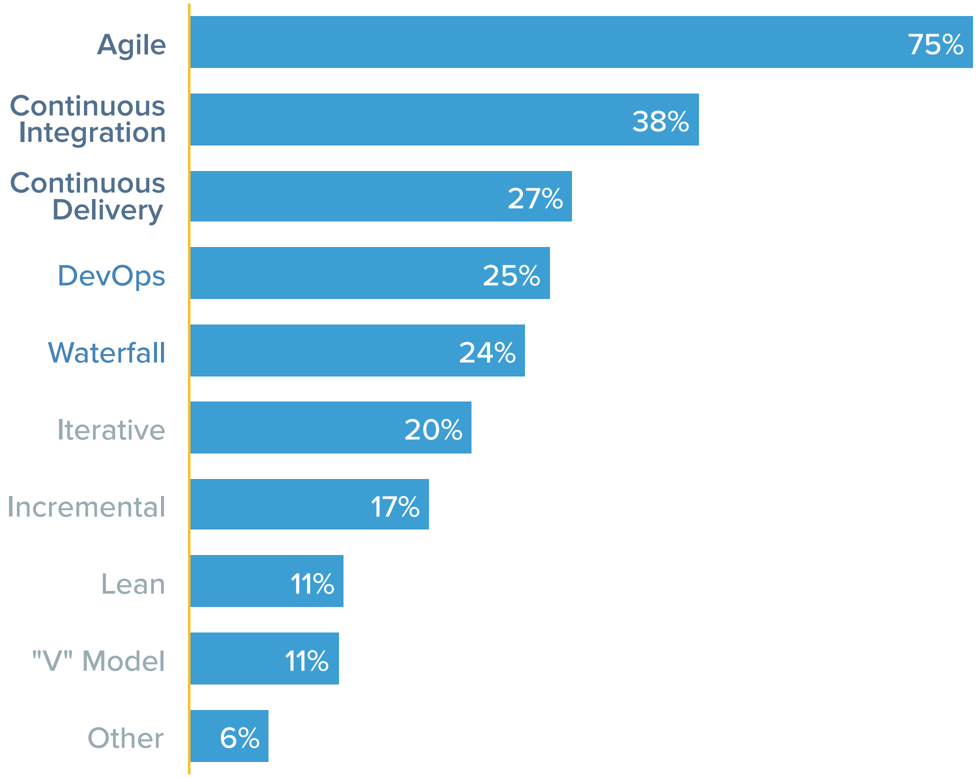
**Several factors have dramatically changed how software is developed, packaged and released for general consumption. These core influences are outlined in this section.**

### Iterative Methodologies

**Description:** *Software planning, tracking and effort is done with a focus on smaller software components and features performed in small time frames (weeks versus months).*

The release of software has changed dramatically over the past 6-10 years. Waterfall methodology was a common approach to developing software using a well-defined, sequential process that often would take 6, 12 or 18 months to deliver software to the customer with hundreds of capabilities/features. A significant challenge in this process is keeping the software in sync with customer's requirements and delivering what has been promised. Waterfall often deviated from the needs of the customer due to the long delivery cycle and changing concerns of the customer.

In a recent survey of over 2,300 API testing and development professionals it was clear that a majority of API teams are now embracing an agile approach to software development. Waterfall is still being used by a portion of teams but is outpaced by other delivery methods, including continuous integration (37.8%), continuous delivery (27%), and DevOps (25.4%).



Iterative methodologies provide an alternative philosophy and approach towards API development. A smaller amount of features or capabilities are tackled in two, four or six week increments, often called sprints. Iteratively releasing a small amount of features allows teams to stay more aligned to customer's requirements. Additionally, customers provide immediate feedback and often are direct participants in the process.

This shift in philosophy allows for far more rapid change and consumer input in the way that APIs are developed. One shortcoming of traditional waterfall approaches is that it's difficult – if not impossible – to predict customer needs months in advance. This approach allows organizations to be much more sensitive and reactive to the marketplace in a shorter timeframe.

In spite of the benefits of Iterative Methodologies, it's important to recognize the challenges that these introduce in API delivery. It's a complete change in process and mindset, requiring a different approach to specifications and documentation, new tooling, and an organizational emphasis on flexibility to change.

## Moving to the Cloud

**Description:** *Infrastructure, Platforms and Software-as-a-Service provide on-demand resources that compete with on-premise or managed service provider resources.*

Cloud computing has introduced a major shift in how organizations design, develop and provision software. The speed of provisioning software and infrastructure occurs within minutes versus the weeks, months or years for traditional hardware and software. In addition cloud infrastructure provides the flexibility of being able to scale on demand to meet the demands of an organization's customer usage.

The challenges introduced by this new model of computing are numerous. The first may relate to whether the flexibility the cloud provides can fit within regulatory, data privacy and service level agreements of an organization. For example, customers that have certain in-country data requirements for personally identifiable information (PII) may not find a vendor who can provide such services. Additionally, cloud computing requires a different set of organizational skills. Examples of this would be getting staffed trained in the platform being leveraged (Salesforce.com, Amazon Web Services, Workday, Microsoft

Azure). Finally, the cost considerations and how you operate within these constraints is a concern. Platforms such as Amazon Web Services cost are driven by the type of resource (e.g. larger servers vs. small servers) and cost model being leveraged. Different cost models are provided to allow for immediate access to a resource (On Demand), long term reservations (Reserved) and an auction marketplace (Spot). Generally, it's important to consider that the shift towards cloud computing changes the roles and tools used in API delivery, introducing migration challenges.

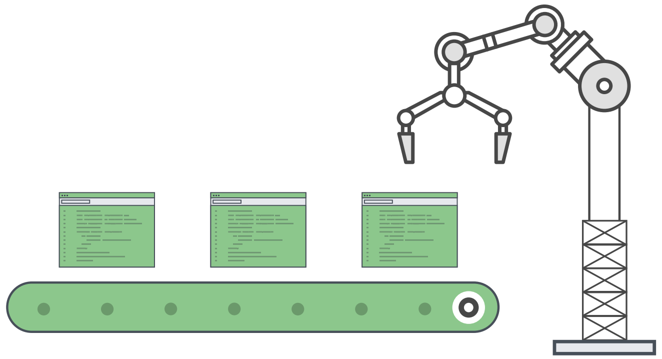
## Business Agility

**Description:** *Infrastructure, Platforms and Software-as-a-Service provide on-demand resources that compete with on-premise or managed service provider resources.*

Today's businesses are challenged with delivering functionality to their customers at much faster rate. Examples range from the automotive industry, which shrunk new car development from 5 years to 3 years – changing the standards by which all competitors needed to deliver new automobiles. Additionally, the consumer electronics industry has seen a similar emphasis on rapid delivery, with organizations such as Apple setting the consumer expectation that they release new products on an annual basis. The music industry delivers their product almost exclusively through an on-demand model.  Put simply – as organizations make advances in business agility, consumer expectations mold to these new timeframes making business agility the standard rather than the exception to the rule.

A differentiator for many organizations is their software and its ability to refresh hardware to support more and more capabilities. Tesla automotive for instance was able to introduce Autopilot (self-driving vehicle capabilities) via wireless updates to take advantage of on-board electronics and sensors. Nest is able to introduce new features for heating and air conditioning, smoke alarms and security video, in the same fashion.

Software's flexibility makes it a natural supporter for Business Agility and bringing compelling products to market.



## Stores/Marketplaces

**Description:** *Consumers demand simple access to vendors' software, APIs and applications.*

With the advent of iTunes and the App Store, Apple fundamentally changed how digital content / applications were found and consumed. Digital stores or marketplaces have become a requirement in today's organization to facilitate a simple and easy consumption model for their customers. Stores and marketplaces also provide a means of creating a rich eco-system with third party providers, an additional revenue stream for applications / APIs sold and increase the overall stickiness of a provider's solutions.

While APIs have enabled a piecemeal approach to application development – why develop a robust mapping capability inherent to your application when Google Maps already has an API for that? Why reinvent the wheel? – this creates a need for shift in focus. Extensibility is profoundly important, so developers have the challenge of creating better documentation, as well as including a developer portal in the delivery process. It's imperative to consider how your application or API could potentially be consumed or extended by others from the very beginning.

## Talent

**Description:** *Competition for and maintaining sufficient technical resources impacts organizational ability to deliver exceptional software.*

The competition organization's face in acquiring and maintaining talent to develop software solutions is ever increasing. As part of that challenge, knowledge management and solutions to capture and leverage that knowledge are critical to maintain team's velocity. In addition, alternative means to support projects such as outsourcing, distributed teams and open source software models can impact delivery due to distributed capabilities / knowledge. Companies are also reaching out to the developer community via Open Source Software solutions they are sharing on GitHub to draw interest, participation and even employees. Examples of this include Nike, Netflix, LinkedIn, Google and others.

The new social contract no longer dictates that individuals stay loyal to a company for the majority of his or her career – now, as tenures at different organizations shortens to just a few years, anticipating turnover and building replicable processes is a challenge.

## Development Operations (DevOps) comes to the forefront

### Example Roles

As the relationship between API Development and Operations teams tightens, traditional roles within these teams are changing. The following are example roles in an organizations delivering APIs using Agile Methodology. In this example, the team executes work in 2 weeklong sprints with an iteration planning meeting performed on Day 1 of the sprint and a retrospection (review) meeting performed on day 10 of the sprint. Between these two dates, a 15 minute stand up meeting is performed to summarize each roles work for the day and if any blocking actions are preventing them from completing their work. If this is the case, the Scrum Lead typically will be designated in removing the blocking activity so that the team can proceed.

**Scrum Lead** - leader of a scrum team that manages the backlog, iteration planning and managing blocking conditions for other team members. They also coordinate with the customer or act on the customer's behalf in organizing user stories (features / work to be performed) for the iteration.

**Developer** - responsible for taking user stories and developing technical capabilities in the form of application or API logic.

**Architect** - responsible for supporting the technical staff by providing guidance on best practices and working with the business on translating business requirements into technical strategy.

**DevOps** - this role focuses on the integrating the software solutions into available infrastructure using solutions that help build, package, deploy and test the implemented capabilities. They also are responsible for moving those features through the appropriate environments up to and including production. Finally monitoring / maintenance of the implemented solution in production is performed.

## Continuous Everything

DevOps often discusses Continuous Integration, Continuous Delivery and Continuous Deployment. What do each of these mean and when or why does an organization embrace one of them? The following sections go into further detail to help clarify each:

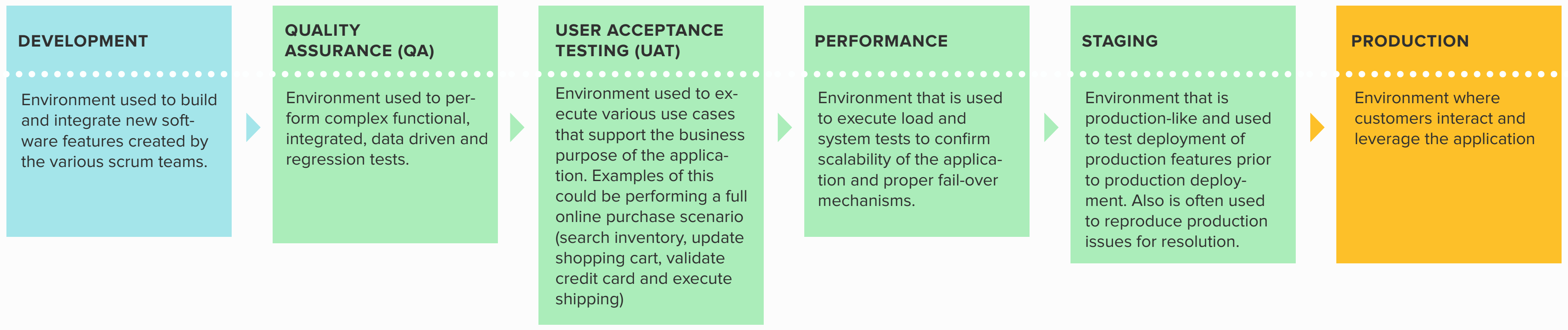
### Continuous Integration

Continuous Integration refers to using automation to incorporate developer features that are ready to merged into the software application. The goal is to perform builds for each new feature as soon as it is available and to identify issues early in the process. This allows developers to correct issues quickly as they occur. It also provides a means for team members to have visibility of changes that are occurring with each commit. In addition to confirming new features successfully execute and work as anticipated, test automation can also be used to help identify regression.

## Continuous Delivery / Continuous Deployment

Whereas CI focuses on the development environment, Continuous Delivery and Deployment, brought about changes in how software was delivered and deployed to the remaining environments within an organization. Continuous Delivery focuses on the getting software features to a production-like environment. In many organizations, this environment may be described as Staging. By vetting that the software features run successfully in a production-like environment, the producer has a high degree of confidence in the new features and can decide on when they will release it to the customers.

Continuous Deployment executes a similar process to Continuous Delivery however the software is eventually deployed to Production directly. In either case, both Continuous Delivery and Deployment rely heavily on automation of the deployment of the software and subsequent testing that confirms the software meets expected criteria for the environment that it is in. Example environments may included Development, Quality Assurance, Staging and Production. The following table articulates what each environment's purpose may be:



**DEVELOPMENT**

Environment used to build and integrate new software features created by the various scrum teams.

**QUALITY ASSURANCE (QA)**

Environment used to perform complex functional, integrated, data driven and regression tests.

**USER ACCEPTANCE TESTING (UAT)**

Environment used to execute various use cases that support the business purpose of the application. Examples of this could be performing a full online purchase scenario (search inventory, update shopping cart, validate credit card and execute shipping)

**PERFORMANCE**

Environment that is used to execute load and system tests to confirm scalability of the application and proper fail-over mechanisms.

**STAGING**

Environment that is production-like and used to test deployment of production features prior to production deployment. Also is often used to reproduce production issues for resolution.

**PRODUCTION**

Environment where customers interact and leverage the application

🟦 Continuous Integration 🟩 Continuous Delivery 🟨 Continuous Deployment

So when would one consider using Continuous Integration, Continuous Delivery or Continuous Deployment? This is up to the organization but the following table helps isolate which option makes the most sense based on delivery frequency.

| | Description | What does it impact | For whom? | Effort |
|---|---|---|---|---|
| **Continuous Integration** | New features are quickly integrated for confirmation. | Development Environment Only | All organizations | Requires setting up a build automation solution and having source code be stored in source code management. |
| **Continuous Delivery** | New features are propagated to all environments including a production-like environment. | QA and other environments up to and including a production-like environment. | Organizations that want to stage new features and release on a frequent schedule. CD provides a level of validation / quality. | Requires using build or CD solution to automate deployments. Test automation also needs to be leveraged to identify whether software meets exit criteria or not. |
| **Continuous Deployment** | New features are propagated to all environments including production. | QA and other environments up to and including production | Organizations that release new features on a daily even hourly basis. | Same as Continuous Delivery with added emphasis in testing once in Production. Ability to automate rollback of production features should also be considered. |

Continuous Delivery and Deployment are not simple and require fortitude to automate the pipeline. Examples of organizations that have implemented these complicated methodologies include Netflix and Yahoo.

## Automation is Key

Automation is the use of technology to facilitate execution of repeatable processes in a consistent manner. Similar to the creation of the assembly line by the Ford Motor Company in the early 20th century, automation in software is meant to provide similar benefits such as consistency, repeatability and speed. Areas of automation can include build automation, test automation, deployment automation and infrastructure automation. The following is a table describing some example solutions that support these types of activities that we often see being utilized by clients:

| Topic | Purpose | Example Solutions |
|---|---|---|
| **Continuous Deployment** | Keep track of all software being developed with an organization in order to support products and their various permutations. | Apache Subversion, GitHub, Atlassian Bitbucket, PVCS, Microsoft Team Foundation |

| Topic | Purpose | Example Solutions |
|---|---|---|
| **Build Automation** | Automate the building and packaging of software solutions and integration into other automation frameworks. (This is the brain that orchestrates what happens in the devops pipeline) | Jenkins, Hudson, Atlassian Bamboo, Microsoft Team Foundation, Apache Continuum |
| **Artifact Management** | Manage the lifecycle of built artifacts for consumption in later deployments or sharing with other teams / organizations. | JFrog Artifactory, Nexus, ... |
| **Test Automation** | Execute automated tests against applications, APIs or other backend components | JUnit, Selenium, Smartbear Ready!API |
| **Infrastructure Automation** | Manage the creation of infrastructure on demand. This can be locally or in the cloud. | AWS CLI, Docker, Puppet, Chef, Ansible, Azure Resource Manager, Shell scripts |

## Best Practices in supporting Knowledge Management, Continuous Integration, Delivery and Deployment

The following are scenarios that demonstrate some core recommendations when organizations are performing Knowledge Management, Continuous Integration, Delivery and/or Deployment.

### Artifact Sharing/Knowledge Management

How does one capture the knowledge of a team that is continuously iterating and developing solutions? All the artifacts and notes created for each capability tell a story of who, when, what and why. Using these tools we can go back and forward in sprints to learn, educate and remember.

- Agile tools such as Rally and JIRA capture document the user stories, participants, team communications, releases etc.
- Test Automation tools like Ready!API capture the API invocation flows via TestSuites and TestCases.
- SwaggerHub and Swagger provide a means to define, version and communicate APIs to team members, partners and consumers. Using the Swagger definition, one can incorporate the meta-data into Ready!API solutions, Amazon Web Services, Microsoft Azure, WSO2 solutions and other third party products.
- Source Code Management solutions keep track of all versions / submissions and associate comments.

### API Virtualization

When developing in an iterative manner, changes to backend systems will require days/weeks to complete. This makes it very challenging for teams that are dependent on APIs to perform their work including Front End Developers, DevOps / Test Developers and other Third Party Developers. A common approach which helps lessen the burden allow for parallel development is the use of Mock services or API Virtualization. Consider using the following solutions to increase your team's velocity:

- Smartbear ServiceV to simulate to be built endpoints / API logic.

- SwaggerHub generated Servers to expose a public simulated endpoint / API logic.

## Test Automation within Continuous Integration

A key component in Continuous Integration is the ability to verify the software being integrated provides both the new capability(s) and does not introduce regression in the existing capabilities. Test automation provides this capability at the User Interface, API and Data Store level.

- Have developers create Unit tests to be focus on new components.
- Have developers create TestSuites / Testcases to exercise the API functionality and use Assertions to confirm the Transport Codes, Responses and Headers match expected behavior in both the successful scenarios and failure scenarios.
- Have DevOps create Data Driven Testsuites / Testcases to flood the system with more realistic information to isolate / identify data related issues.
- Have DevOps execute Security scans to pinpoint unexpected behavior due to malicious consumers.

## Test Automation within Continuous Delivery/Continuous Deployment

Where continuous integration focuses on the integration of new features into the existing code base, continuous delivery and continuous deployment move the capability up to production or into production via automation. In order to control the movement of the capability from one environment to another, gating needs to be performed. Automating testing provides a means to introduce the gating logic. Tools such as Smartbear's Ready!API provide a rich set of capabilities to perform functional, data driven and security testcases. This, coupled with complex assertion support, allows teams to accurately define complex logic to gate Continuous Delivery or Continuous Deployment pipelines.

- Have DevOps leverage Smartbear Ready!API TestSuites/TestCases that are specific to the environment being validated. In the QA environment, Data Driven tests with appropriate assertions could be implemented.
- In another environment, Integration Testing may be relevant and thus require using Smartbear Ready!API to support invoking a series of APIs in

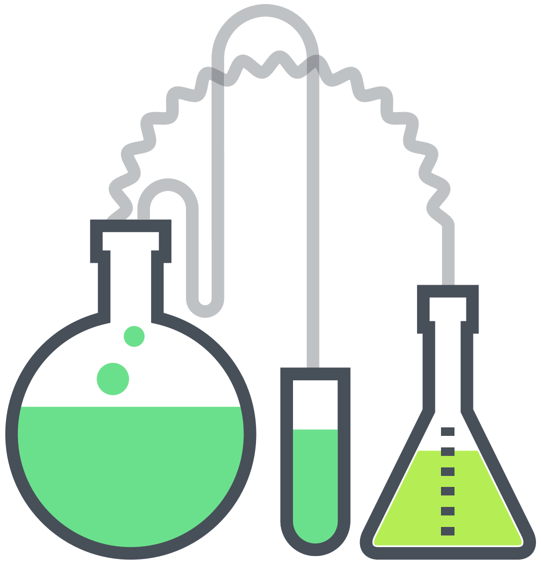
support of a business process. Ready!API provides a simple way to define numerous APIs, SOAP Services, Databases and JMS endpoints that can be used within a Testcase / Testsuite to accurately reflect the real process.

## System Testing of Automated Infrastructure

Organizations that have leveraged automated infrastructure in cloud providers such as Amazon Web Services, Microsoft Azure or Google Compute test to make sure their infrastructure configuration supports resiliency. Netflix has led the effort in developing open source tooling to kill / disable infrastructure (e.g. Chaos Monkey). These outages will often test features such as:

- Container self-healing
- Auto-scaling groups
- DNS Failover
- Elastic Load Balancing
- Multi-AZ failover
- Multi-Region failover

In order to verify applications that have been architected to fail, It is important to simulate production level traffic to exercise APIs and Applications with tools such as Smartbear LoadUI. LoadUI can generate accurate API requests and appropriate level of virtual users. This in combination with Chaos Monkey can allow teams to simulate system level outages and deeper understanding of how infrastructure behaves. These behaviors provide valuable information to DevOps personnel who can then develop a playbook and valuable experiences.



## About the Author

Chris Riley is a founding Partner at HKM Consulting LLC and has focused on the services and integration space since 1995 in a variety of pilot, implementation and training roles with companies such as PTC, IBM, Cape Clear Software and SOA Systems. In his various capacities, Chris has also been a speaker and contributor at QCon, SOA Symposium, ECAUG, Prentice Hall Podcasts, AWS Boston Meetup and Toronto JUG along with publications in SOA Magazine and SOA Design Patterns. Currently Chris is supporting Fortune 500 companies in APIs, DevOps, Cloud Computing, and Governance as part of their IT Strategy. Chris can be followed on twitter @hkmconsulting and the HKM Blog

SMARTBEAR

The most trusted API testing framework for Agile developers and testers

SoapUI NG Pro

TRY IT FOR **FREE** TODAY

SMARTBEAR

Over 4 million software professionals and 25,000 organizations across 194 countries use SmartBear tool

| 4M+ | 25K+ | 194 |
|---|---|---|
| users | organizations | countries |

See Some Succesful Customers >>

## API READINESS

Functional testing through performance monitoring

SEE API READINESS PRODUCTS

## TESTING

Functional testing, performance testing and test management

SEE TESTING PRODUCTS

## PERFORMANCE MONITORING

Synthetic monitoring for API, web, mobile, SaaS, and Infrastructure

SEE MONITORING PRODUCTS

## CODE COLLABORATION

Peer code and documentation review

SEE COLLABORATION PRODUCTS

SoapUI NG Pro