Agile Software Factory: Bringing the reliability of a manufacturing line to software development

Today's businesses are complex organizations that must be agile across multiple channels in highly competitive global markets. To succeed, their IT departments must work as efficiently and reliably as a manufacturing assembly line, where each stage of development and release process is clearly defined and automated to minimize errors and delays. To help businesses address these needs, Grid Dynamics offers Agile Software Factory (ASF) – a Continuous Delivery toolkit that uses pre-integrated tools to implement end-to-end development and release processes. Industry standards and best practices implemented in ASF serve as a solid foundation for assuring an efficient, agile, and structured DevOps culture. Offering a wide range of standard pre-configured features, ASF can be easily customized to address unique requirements and integrate with a business' existing processes and tools.

introduction

The fast-changing landscape of retail makes it extremely challenging for tier one retailers to maintain their competitor edge and market leadership. Today's consumer is more technically savvy than ever before and wants a unique online shopping experience customized to his or her preferences. The added complexity of supporting multiple channels, coupled with the increasing size of engineering teams, makes it more difficult to deliver the necessary software quickly and reliably. To succeed, retailers need to take a more integrated and structured approach to eCommerce built on well-defined processes, standards, and best practices. New research commissioned by CA Technologies and conducted between May and July 2013 by Vanson Bourne shows many organizations are achieving significant and measurable benefits from DevOps—anywhere from between 17% to 23% improvement in key business metrics such as revenue, time-to-market, and new

Grid Dynamics

SEPTEMBER 2014 WHITE PAPER

customer acquisition[1]. Other benefits outlined match what Grid Dynamics sees in the market (see Figure 1).

industry overview

During the early days of software development, building and releasing software involved relatively small teams that could effectively use informal processes involving ad hoc procedures and manual handoffs. In contrast, today's modern software engineering organizations tend to require the coordination of many teams composed of tens and hundreds of developers, QAs, architects, analysts, managers, and release engineers. As a consequence, software development processes have become much more complex, leading to longer release cycles and greater risk.

This is especially true for tier one retailers. Over the past several years, retailers have had to make significant investment in their digital omnichannel platforms to stay relevant in an increasingly competitive marketplace. Such investments lead to tremendous growth of retailers' IT departments, demanding fast and efficient innovation. Technology giants like Amazon, who implemented Continuous Delivery through many years of trial and error, are now able to deploy to production at blazing speed and with greatly reduced risk. Traditional retailers, in contrast, don't have the luxury of spending a lot of time to set up proper processes. Their IT departments need to achieve efficiency from day one.

problem description

With the increasing growth of engineering teams, businesses find themselves facing new kinds of software delivery issues:

- Development environments have become a scarce resource, and development and QA teams lose valuable time waiting to acquire the proper development environments. Furthermore, most engineers and managers don't know how many environments the team actually has or which projects are currently deployed there.
- Releases are constantly delayed due to either critical defects appearing in late design stages or lack of agreement about what functionality should go to production.
- Management doesn't have a clear picture of the completion status of each feature planned for release. This increases overall uncertainty and delivery risks.
- Defects take longer to fix due to lengthened feedback cycles. For example, because testing is often completed just before release, developers have lost the context of any defects found.

existing approaches

The value and importance of Continuous Delivery and a structured DevOps culture is generally recognized today. Businesses have a wide choice of tools to help with implementing very specific aspects of Continuous

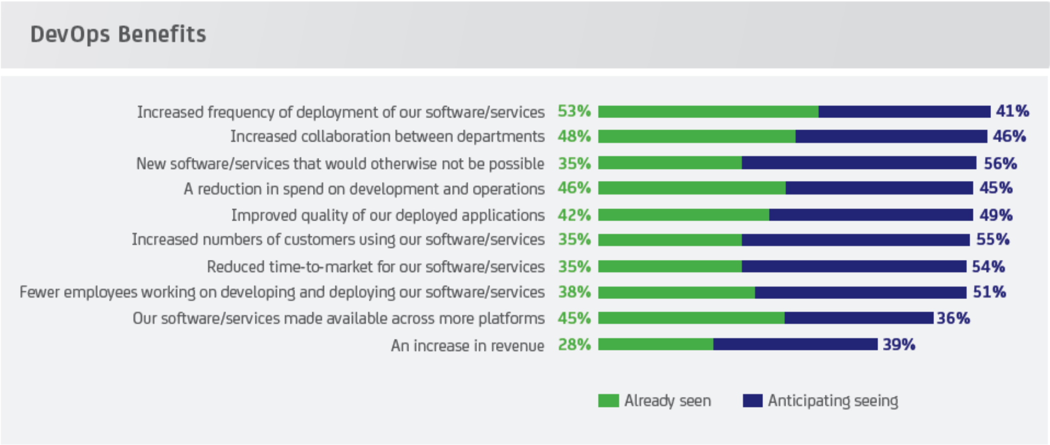

Figure 1: DevOps Benefits from Vanson Bourne

[1] Vanson Bourne. "TechInsights Report: What Smart Businesses Know About DevOps" CA Technologies, 2013. Web. September 2014. https://www.ca.com/us/register/forms/collateral/techinsights-report-what-smart-businesses-know-about-devops.aspx.

Delivery, including source code management, code quality analysis, continuous integration, deployment automation, environment management, and testing automation. However, having to select the optimal combination of tools and integrate them together makes the challenge of implementing a Continuous Delivery-based infrastructure even harder.

Even when a business can overcome these technological difficulties, its teams implementing Continuous Delivery often invent custom processes from scratch. Not many companies can afford to go through the long experimentation cycles of trial and error required to assure these processes are robust enough to trust critical business operations to. Moreover, such implementations often result in the creation of specialized development and release process knowledge that typically belongs to a small number of key individuals. This exposes businesses to the risk of these individuals leaving the company and taking their knowledge with them.

Agile Software Factory

ASF key features/concepts

To help retailers address these DevOps challenges, Grid Dynamics offers Agile Software Factory (ASF). Agile Software Factory provides a ready-to-use implementation of Continuous Delivery out of the box through self-deployable, integrated components that create well-defined development and release processes, standards, and best practices. Among the many benefits Agile Software Factory can bring to businesses are:

- **Speed**: ASF increases the rate and frequency with which new features can be implemented. Deployments that used to take days can be completed consistently and reliably in less than an hour through complete deployment automation.
- **Consistency**: ASF implements proven, standardized processes built upon well-documented standards including branching strategy, application properties management, database schema, and data management. This ensures not only correct usage of ASF tooling but also establishes efficient development and release processes and culture.
- **Visibility**: Environment and deployment management portals give teams up-to-date visibility into the status

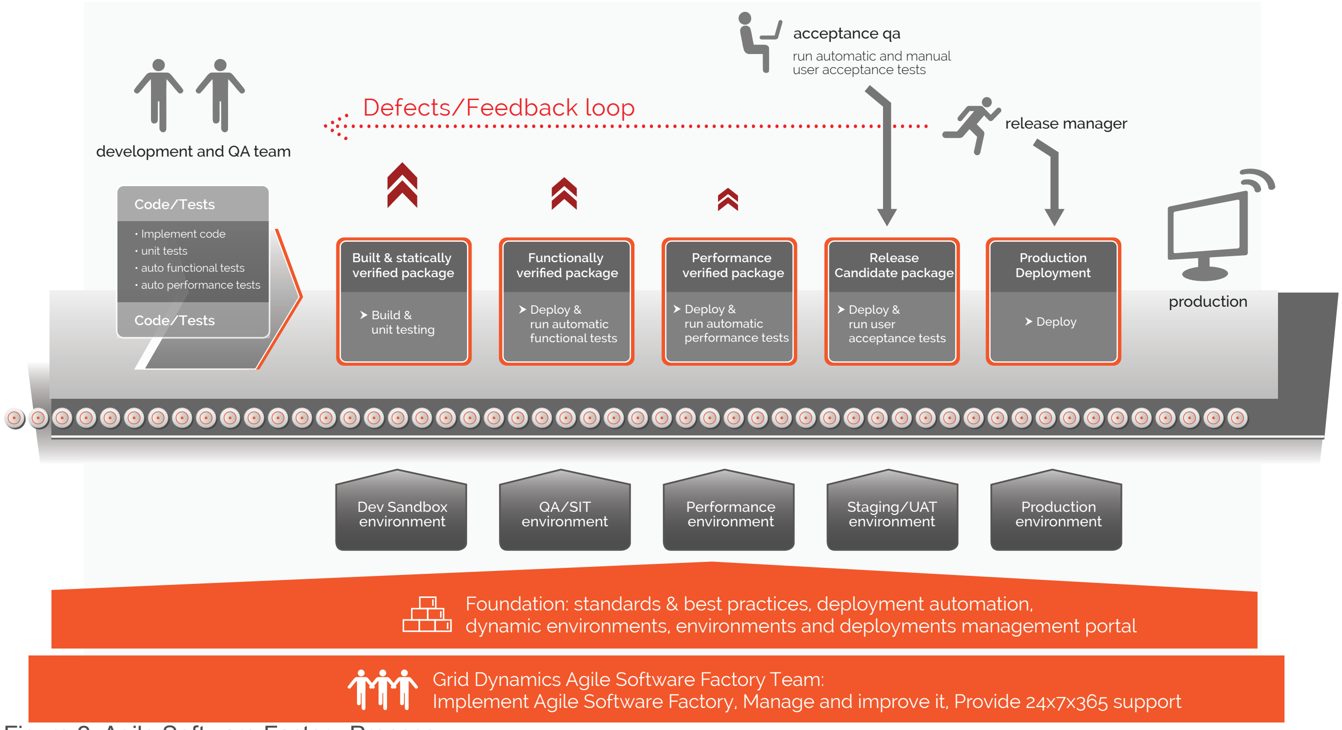
acceptance qa
run automatic and manual user acceptance tests
Defects/Feedback loop
release manager
development and QA team
Code/Tests
• Implement code
• unit tests
• auto functional tests
• auto performance tests
Code/Tests
Built & statically verified package
➤ Build & unit testing
Functionally verified package
➤ Deploy & run automatic functional tests
Performance verified package
➤ Deploy & run automatic performance tests
Release Candidate package
➤ Deploy & run user acceptance tests
Production Deployment
➤ Deploy
production
Dev Sandbox environment
QA/SIT environment
Performance environment
Staging/UAT environment
Production environment
Foundation: standards & best practices, deployment automation,
dynamic environments, environments and deployments management portal
Grid Dynamics Agile Software Factory Team:
Implement Agile Software Factory, Manage and improve it, Provide 24x7x365 support

Figure 2: Agile Software Factory Process

and usage of environments. Team always knows how many environments they have, what are they being used for, and what version of applications are deployed in each of them. Team can also create new environments on demand, destroy environments on schedule, and efficiently manage hardware resource utilization with quotas.

- **Flexibility**: Dynamic management of cloud-based resources leads to cost savings by creating environments when they are needed and destroying them when they are not.
- **Automation**: Automated setup of pre-integrated ASF components saves time on initial implementation. Components integration also ensures efficiency of end-to-end processes.

ASF processes and best practices

With its integrated components, ASF creates a balanced system that implements various best practices and supports customization to meet a business' specific needs. The Continuous Delivery process that ASF implements out of the box is shown in Figure 2.

ASF implements an assembly pipeline that manages code changes to production deployments reliably and quickly, so ideas transform into revenue faster. Proper functionality of the ASF assembly pipeline is ensured by industry-proven best practices:

- **Continuous Integration (CI)**: CI is the core of ASF. All ASF components and communications between components are built to provide effective continuous integration. Specifically, ASF components implement automated testing and deployment combined into effective pipelines that makes frequent integration of code changes possible while enabling large developers teams to work more effectively and deliver functionality faster.
- **Code change management**: Effective code change management is the cornerstone of every Continuous Delivery infrastructure. It is very important to keep code change management simple but at the same time enable parallel development. For example, to minimize the number of merges and reduce merging complexity, ASF implements shared trunk for its branching strategy.
- **Binary artifact management**: ASF implements versioned storage of binary artifacts. Every successful build (even of the same version of code) has a corresponding set of binary artifacts stored and versioned. That allows artifacts promotion through all testing environments as well as connects testing results to specific builds.
- **Database change management**: Consistent application of database migrations across environments makes CI-based processes consistent and reliable. As a result, at every moment there is a package consisting of all relevant source code, database scripts, tests, and binaries.
- **Continuous testing**: ASF testing is based on test pyramid concepts. To ensure an effective CI pipeline, ASF enforces automation of functional and performance testing using industry standard frameworks. Test automation also significantly reduces the time required to get certified artifacts and enables adoption of Acceptance Test Driven Development.
- **Environment management**: Automated environment provisioning makes it possible to create new environments on demand and destroy environments on schedule. This practice leads to effective cost management of cloud-based environments.
- **Configuration as code**: It is very important to manage dependencies between all changeable assets across a project. The tools used to manage dependencies – including deployment automation scripts, CI server jobs configuration, DB configuration, application configuration, and source code – are therefore stored and versioned. This enables ASF to provide a consistent set of assets at every point of time, a factor critical for reliable Continuous Delivery.
- **Application properties management**: Proper application properties management enables effective deployment to various kinds of environments. This practice streamlines the application deployment path from the development environment through test environments onto production. ASF implements a hierarchical approach to application properties management, allowing for customization of properties for every environment type or specific environment with dynamic assembly during the deployment process.
- **End-to-end visibility**: Visibility into feature status is one of the critical parameters of the Continuous Delivery infrastructure, as is immediate access to test results by developers. ASF components – like the project management and task tracking system, CI server, and code review server – are integrated to provide high visibility into feature status across the entire development, testing, and release lifecycle.

- **Incremental code review**: Every completed change of code should be reviewed. Incremental code review illuminates code issues at earlier development stages. This speeds deployment as well as identifies design defects earlier. Every single commit in ASF goes through the code review procedure, thus continuously improving code quality.

ASF architecture and infrastructure

The ASF architecture is designed to support components integration and best practices. Best practices are implemented via tight integration of tools across all the layers (see Figure 3).

ASF components

To provide the infrastructure for all necessary services and best practices, ASF contains a number of thoughtfully selected components. ASF uses only stable versions of components to ensure overall system reliability. ASF is built on top of industry standard components that have been proven invaluable in different aspects of Continuous Delivery. Most of the tools used in ASF are open source, thus significantly reducing the total cost of ownership (TCO) of the infrastructure. The list of major ASFO components includes:

- **Jira**: Project management, tasks, and issues tracker.
- **Sonatype Nexus**: Artifacts repository manager that supports binary artifacts storage and versioning.
- **Jenkins CI**: Extendable open source, continuous integration server. As a pipeline orchestrator, Jenkins automates continuous integration and continuous delivery pipelines consisting of build, deploy, and test stages.
- **SonarQube**: Open platform for managing code quality.
- **Gerrit Code Review**: Web-based code review system. Facilitates online pre-commit code reviews for projects using the Git version control system.
- **Jasig CAS**: Single sign-on service.
- **Selenium Grid**: User Interface (UI) testing automation tool. Test automation makes comprehensive CI-based processes possible and enables acceptance test-driven development.

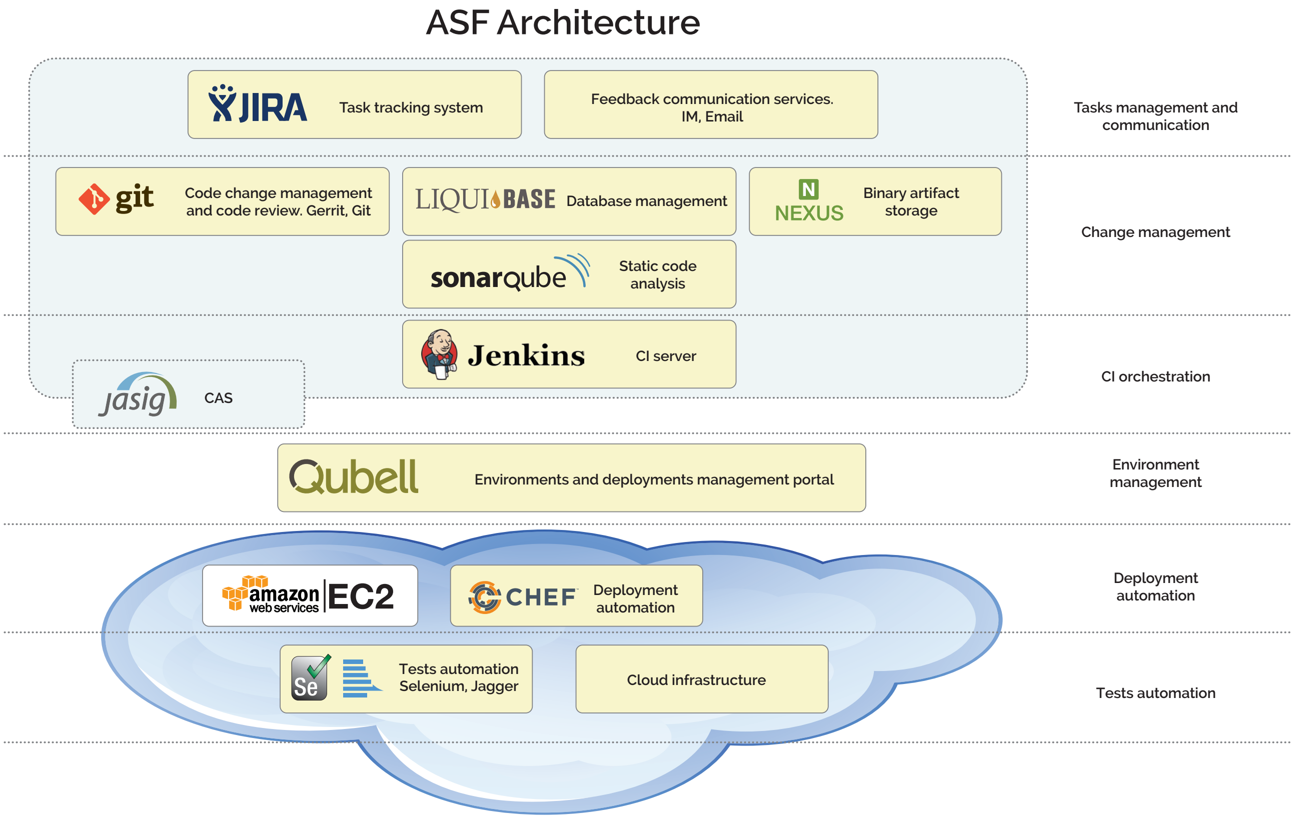

Figure 3: Agile Software Factory Architecture

- **Jagger**: Performance testing framework that, from the very beginning, was designed to meet the rigorous requirements of continuous performance testing.
- **Qubell**: Environments and deployments management portal. Provides control and visibility over environments, orchestrates complex provisioning and deployment tasks, and enables run-time dependency management for composite applications.
- **Chef**: Deployment automation tool that makes application deployment fast, consistent, and reliable.
- **LiquiBase**: Database management tool.

ASF components integration

To implement all of the best practices listed above, ASF components are integrated into a cohesive system. This integration also supports a consistent user experience and data representation across all tools. For example, while browsing a Jira ticket, a developer can see the links to the Jenkins build history and corresponding binary artifacts. All Jira tasks, Gerrit changes, and Jenkins builds have consistent information. Jenkins is the core element of ASF infrastructure, and most of the other components are integrated with Jenkins using special approaches to integration in each case.

ways to customize ASF

Best practices implemented in ASF can be used as a starting point for further customization. ASF components such as Qubell, Jenkins, and Jira support customization of environment management, how pipelines and process are establishment, and tasks and issues tracking. Build, deploy, and test run Jenkins jobs can be adjusted to particular customer projects. Environment management can be adjusted according to the preferred infrastructure as well.

ASF can be also customized to work with the tools a company has already implemented and is using. ASF can be customized to support Bamboo for CI; GitHub, GitLab, or SVN for source control; Ansible or Puppet for deployment automation; and GoGrid, VMWare, or Openstack cloud for cloud infrastructure.

conclusion

The key to success – and survival – in today's quickly evolving market is agility. Being able to deliver value quickly, reliably, and repeated is essential if businesses are to keep pace with technology. By bringing the reliability of a manufacturing pipe to software development, businesses can have visibility into design processes, avoid undesirable delays, and eliminate defects early in the design cycle when they are less expensive to resolve.

By building a structured DevOps culture founded on Continuous Delivery, Agile Software Factory brings consistency, flexibility, and automation to software design while accelerating the rate at which businesses can deliver new features and act on market opportunities. With tight integration of tools across all layers of the development process, Agile Software Factory provides a sound foundation upon which retailers can bring their value to market, reduce TCO, and transform the way they do business.

about Grid Dynamics

Grid Dynamics is a leading provider of open, scalable, next-generation commerce technology solutions for Tier 1 retail. With in-depth expertise in commerce technologies, wide involvement in the open source community and a modern, global workforce, Grid Dynamics helps great companies gain a sustainable business advantage by implementing and managing solutions in the areas of omnichannel platforms, product search, and continuous delivery. To learn more about Grid Dynamics, find us at www.griddynamics.com or by following us on Twitter @GridDynamics.