accenture technology

DEVOPS FOR SAP

HEAVILY LEVERAGING SOFTWARE TECHNOLOGY FOR AUTOMATING QUALITY, CONFIGURATION AND DEPLOYMENT ACTIVITIES

ABSTRACT

DevOps, the methodology which combines software development and information technology operations, relies heavily on enablement by technology. But what is the benefit of DevOps, and is it applicable in SAP technology stacks?

It's important to understand how DevOps generates benefits and which mechanisms are doing so. That allows IT leaders to then explore whether a mechanism can be brought to life using SAP technology. IT leaders will be better able to distinguish which elements of DevOps are essential for realizing the benefits associated with DevOps, and which are optimizations.

Can DevOps be applied in an SAP technology environment? To find out, this whitepaper looks at the necessary characteristics of DevOps and their feasibility for use with SAP technology. Characteristics are deemed "necessary" if they are needed for DevOps to work. This whitepaper discusses ways to implement necessary characteristics once they are found to be necessary.

At the time of writing, the DevOps community considered DevOps with SAP technology feasible. Typically, the community provides multiple technology patterns for implementation. A technology or implementation pattern consists of a defined set of technical components and their interaction. Accenture also provides an SAP cartridge for the Accenture DevOps Platform (ADOP), which encompasses pre-configured tools and scripts to jump-start an implementation.

Due to the extensive amount of detail needed to implement the patterns, this whitepaper provides references for further information. If your company wants to implement a pattern, it should be guided by an experienced DevOps consultant.

The purpose of this document is to provide an easy-to-read introduction to the topic that omits details in favor of explaining DevOps in its larger context and by its main characteristics.

Most IT organizations at large organizations are experienced with agile software development processes and practices and have replaced waterfall project management with them. While this transformation is underway, DevOps is emerging as the next big milestone on the roadmap of IT organizations.

DevOps uses software to automate quality, configuration and deployment activities, but its scope extends much further than that. Nonetheless, some organizations dismiss the feasibility of DevOps in some areas due to the technology side of DevOps.

For instance, many consider DevOps difficult or even unreasonable for SAP back-end systems. This hypothesis—that SAP back-end systems are not suited for DevOps—deserves proper evaluation, which we provide in this paper.

The technology aspect of DevOps, represented by the Deployment Pipeline pattern, and its realization with SAP back-end systems, is the focus of this paper. The first section elaborates on the value proposition and essential elements of DevOps. Readers familiar with these topics are encouraged to skip to the second section, which is an evaluation of applying DevOps technology practices to SAP back-end systems.

TABLE OF CONTENTS

DEVOPS VALUE PROPOSITION

When companies use agile development for software development, this typically results in software changes in regular intervals. In the language of Scrum, the iterations in the agile development model are called “sprints,” and the results of those sprints—e.g., the changes to the software—are known as “increments.” However, scrum as a process management framework doesn’t tell users how to develop the increments. The question of “how” is better described by other practices, such as “Extreme Programming.” Scrum also doesn’t tell how increments are to be taken into productive use.

Enter DevOps. DevOps effectively addresses the difficult challenges—whether they are technical, cultural or practical—associated with frequently and reliably deploying increments into production environments.

Why is it important to continuously and frequently adapt software in the production environment?

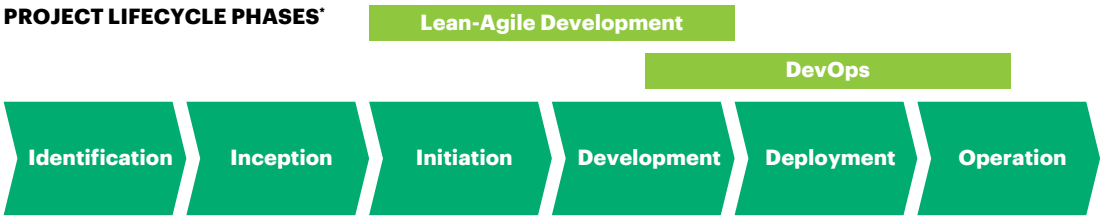
The main reason is to allow the business organization to see and learn from the effects of new or changed customer interactions, while keeping risk low. Most companies want to offer improved customer experience or new services, both of which require careful integration of changes into the production environment—regardless of whether a customer has frequent contact with the system or none at all. When small changes are made gradually, the risk associated with them is reduced.

> **Agile development without DevOps could be compared to revving up an engine in neutral gear—the engine is generating power which isn’t used.**

In many cases, teams and processes for deploying software are in place in IT organizations, they are just not operating at the same level of efficiency and scale as those on the agile development team. DevOps bridges the gap from the development environment to productive operation, and it does it efficiently and reliably.

According to the 2017 State of DevOps Report[1], compared to lower-performing peers, IT organizations using DevOps can perform 44 times more frequent deployments, have lead times for changes that are 440 times faster, they have a mean time to recover that is 96 times faster, and they experience one fifth the change failure rate.

Figure 1. Lean-Agile Software Development and DevOps complement each other to achieve fast-paced, frequent adaptation of productive systems



*2010; Jez Humble, David Farley, Continuous Delivery, Pearson Education

While the DevOps culture, principles and practices complement agile software development practices, the business case for implementing DevOps is the benefit to the organization of having fast and frequent adaptation of productive systems according to business demands.

KEY ELEMENTS OF DEVOPS

Continuous Delivery is the approach DevOps uses for achieving high-quality software deployments in an efficient, reliable and frequent manner. Continuous Delivery requires significant automation for many activities—from testing to deployment.

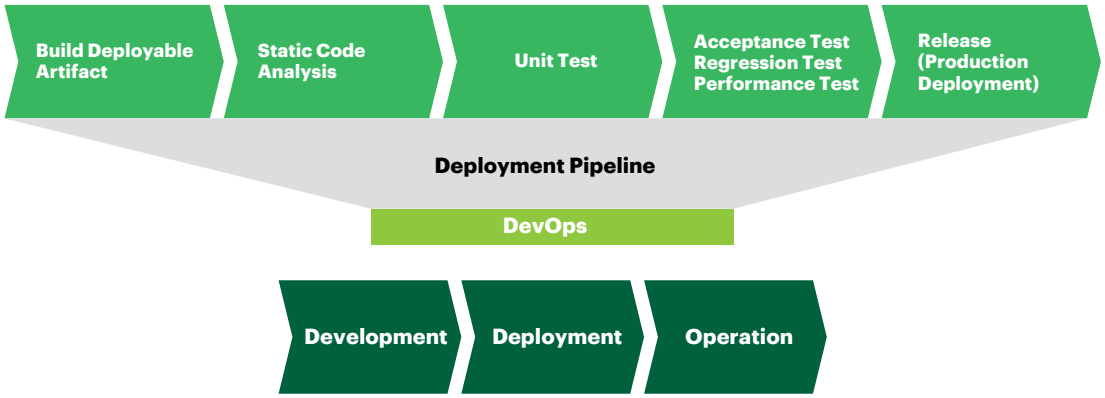
For example, companies can provide pre-configured environments to their development teams on demand by combining scripted automation with virtualization technologies. To some, this approach may seem feasible for web-technology stacks, but not for legacy back-ends. Thus, the question arises whether this capability is required, and if so, how easily can it be achieved.

Let's take a look at which elements are essential to DevOps and Continuous Delivery, and which are optional but useful optimizations.

> **On a side note, while technology is the focus of this paper, to make DevOps work, companies need more than technology and practices. They also need collaboration among different areas of the organization, something that can be difficult to achieve.**

In Continuous Delivery, companies can analyze key elements along the pattern of the Deployment Pipeline. As part of the Deployment Pipeline, almost all deployment activities are automated.

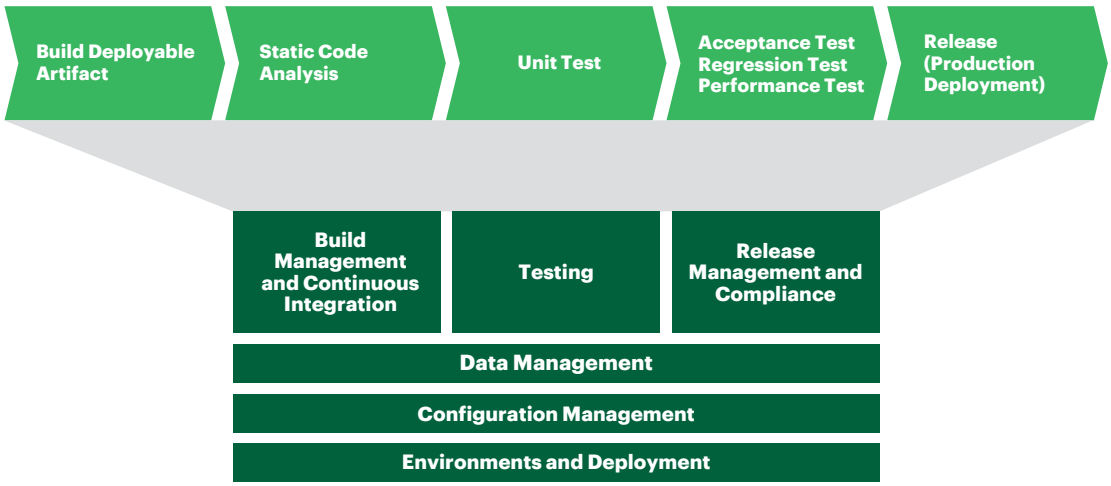
Figure 2: Deployment Pipeline pattern, adapted from the 2006 paper by J. Humble, C. Read and D. North[2]



The examination of essential elements will be conducted for individual capabilities, which are the pillars of the Deployment Pipeline. After listing the capabilities individually, based on the maturity model[3] defined by Humble and Farley[4], we can see that three capabilities resemble sequentially ordered waterfall phases: Build Management and Continuous Integration, Testing, and Release Management and Compliance. Three other capabilities support the Deployment Pipeline across all activities: Data Management, Configuration Management, and Environments and Deployment.

Even though the original paper describing the pipeline pattern was published in 2006 (by J. Humble, C. Read and D. North) and the DevOps movement has evolved significantly since then, these capabilities remain important and provide a common terminology accessible to a large audience.

Figure 3: DevOps capabilities used/defined by Humble and Farley



BUILD MANAGEMENT AND CONTINUOUS INTEGRATION

Continuous Integration, an Extreme Programming practice, aims to build and test-run applications after small changes have been made. This provides early feedback about undetected integration problems for developers. When practiced consistently, the application remains in a working state. With many other approaches, the application under development may be in a "broken" state for most of the release cycle. If anything gets broken, the defect is most likely caused by the latest change, which should be very small and therefore relatively easy to correct or reverse. If developers give themselves a time limit for analyzing the defect and fixing it, or else they must reverse the changes, the application under development will never be broken for long.

When building the software, developers compile source code, link dependent libraries, and create deployable packages. For Continuous Integration, this activity must be automated, which requires a repository for common source code that is version-controlled, an automation server and building script.

A test of the application includes static testing of the source code quality with tools called code analyzers. It must also cover dynamic testing with unit tests, dynamic testing of functional components and so-called smoke testing of the entire application.

Table 1: Requirements to the areas of process/practice, technology/tools and organization imposed by Build Management and Continuous Integration

PROCESS/PRACTICES	TECHNOLOGY/TOOLS	ORGANIZATION
• Build and run the application regularly • Create and maintain a comprehensive suite of automated tests • Keep run-time of build and test automation short • Time-box defect fixing after deployment to Continuous Integration environment	• Source code repository with version control • Static code analyzer • Unit testing framework • Framework or tool for automated component and acceptance testing • Build definition and automation • Automation facility, e.g., shell script and job scheduler (Orchestrator)	• Developers and testers part of one team—avoid separation of testers into separate team

ENVIRONMENTS AND DEPLOYMENT

A production environment is generally not well-suited for development or testing activities; similarly, a testing environment is not well-suited for development. Some activities generate side effects, which could, in the worst case, negatively impact business operations. Therefore, organizations need separate environments that isolate activities and their side-effects from each other when developing, testing and finally using software productively.

When an organization deploys new or changed software, it is installing new or changed executables or configurations to an environment. DevOps provides shorter development cycles, less effort and increased reliability is by addressing the risk and cost of human error that would lead to rework. DevOps does this by automating both the creation of environments and the deployment of new or changed software to such environments.

Configuration settings and scripts for automated provisioning of environments must be maintained in a source code repository. This practice provides documentation and increased reliability of the environment's configuration. The same applies to automation scripts for deploying software to such environments.

Finally, even if the provisioning of environments and deployment of software worked flawlessly, there might still be problems with the environments, e.g., a hard-drive failure. This is why organizations not only need to automate processes and codify configurations, but also monitor the environment.

Table 2: Requirements to the areas of process/practice, technology / tools and organization imposed by Environments and Deployment

PROCESS/PRACTICES	TECHNOLOGY/TOOLS	ORGANIZATION
• Completely automate all steps required for setting up new environments or changing existing environments • Completely automate all steps required for deploying software to any environment • Maintain all configuration and automation scripts in a version-controlled source code repository	• Source code repository with version control • Automation facility, e.g., shell script and job scheduler (Orchestrator) • Environment monitoring	• Application Operations members temporarily embedded into development team

RELEASE MANAGEMENT AND COMPLIANCE

When software is released, it is deployed to the production environment, which brings with it more risk to business operations than deploying to a test environment. In the worst case, business must stop operations until the software that is causing the problem is fixed or removed. Sometimes the best approach to such situations is to roll back to the last state known to be good.

When software is deployed to production and impairs business operations, it is the worst time to make plans for recovery. Such strategies and plans must already exist. The only concern an organization should have when dealing with production failure is the proper execution of their recovery strategy.

Release management begins with all involved stakeholders creating and agreeing to a strategy which includes how to deploy to production and what to do in case of failure. This strategy must then guide the creation of a release plan. None of this is specific to DevOps—it's part of the IT Infrastructure Library's (ITIL) "Service Transition" process.

What is specific to DevOps is the practice of using exactly the same automated deployment facilities which are used for deploying software to any of the previous environments. Deployment to production should be completely automated and fully traceable. In addition, an organization must have the ability to back out changes in the production environment. For other environments like development or QA (testing), this capability is not required. Releasing should also be available as a fully automated process.

Compliance will want to know if proper approval was received and recorded, and if all changes to the production environment are traceable. When deployment and roll-back are completely automated, organizations must enforce the proper logging of approvals, deployment and roll-back activities. They must also make sure that triggering them requires logged, digital authorization.

Table 3: Requirements to the areas of process/practice, technology/tools and organization imposed by Release Management and Compliance

PROCESS/PRACTICES	TECHNOLOGY/TOOLS	ORGANIZATION
• All items listed for Environments and Deployment • Create a release strategy and plan • Completely automate all steps for restoring the last state known as good (roll-back) • Remove authorizations for changing the productive environment manually	• All items listed for Environments and Deployment • Roll-back automation • Logging facilities for all deployment automation	• All items listed for Environments and Deployment • Form a "release working group," consisting of at least one person involved in build, test, deploy and release processes • Foster open communication across silos

TESTING

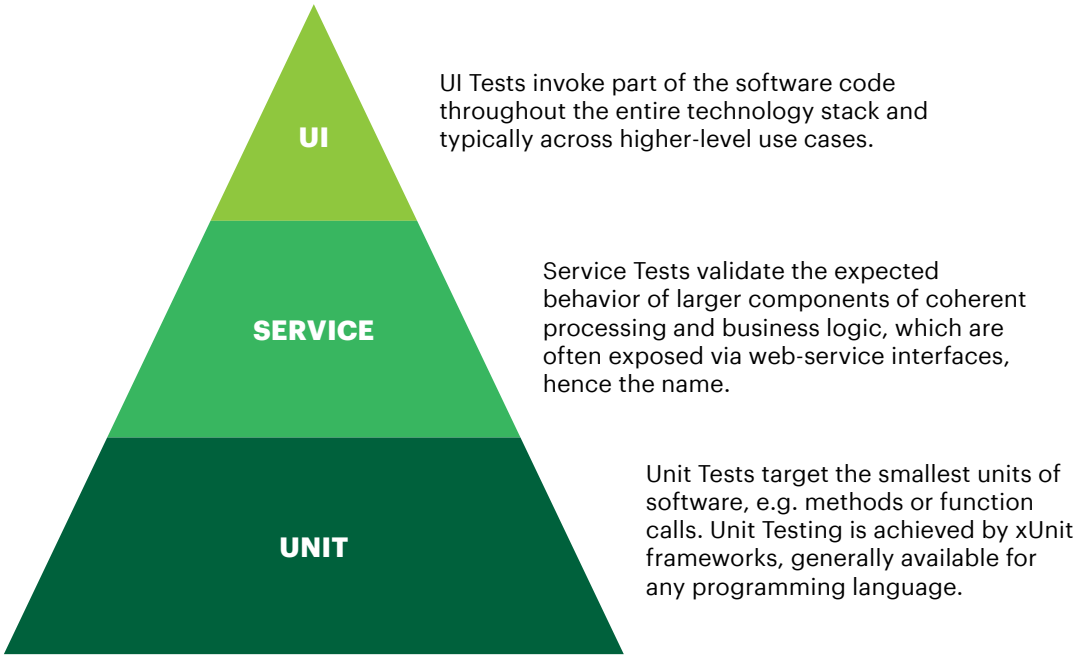
If an organization wants to make a new change in the production environment, those changes must pass tests stage by stage for being promoted from development towards production. Even in V-Model development, testing should start at the smallest scale—unit tests—and progress to the largest scale—an operational readiness test.

For fast, efficient testing, most tests should be automated, such as unit, integration and acceptance tests, which are essential. If load testing is necessary, it should also be done with an automation script.

A manual testing stage can and typically should be a part of the process, since not all tests can be automated. Typically, these are stakeholder demos, usability tests and exploratory tests. Those acceptance tests that traditionally dominated testing activities can and should be automated completely—thereby freeing testers' time for manual testing that has a different focus and brings more value.

Automated tests help keep production incidents from happening and must therefore be effective, cover the right areas and have results that are easy to understand. Testing provides much needed feedback on the quality and feasibility of the software that has been changed. Continuous Delivery is not feasible without a suite of very reliable automated tests which give stakeholders the necessary confidence to release to production.

Figure 4: The Test Automation Pyramid



Such automated tests should be done using guidance from the Automated Test Pyramid, which includes unit, service/API and UI-driven tests. The pyramid distribution guidance is based on the observation that lower-level tests provide better return on effort and are less prone to failure than higher-level automated tests. The higher levels are required because lower level tests fail at detecting defects related to the integration of larger building blocks and their interdependencies.

Table 4: Requirements for the areas of process/practice, technology/tools and organization imposed by Testing

PROCESS/PRACTICES	TECHNOLOGY/TOOLS	ORGANIZATION
• Effort distribution as guided by the Test Automation Pyramid • Consider the testing quadrants • Make tests idempotent	• Unit testing framework • Service/API testing tool or framework • UI-driven test automation tool or framework • Source code repository with version control • Automation facility, e.g., shell script and job scheduler (Orchestrator)	• Testers and Test Automation Engineers embedded into the development team

DATA MANAGEMENT

Application data is typically stored in databases and represents the persistent internal state of a productive application. This internal application state serves the organization as a memory for customers, transactions, prices, products and organizational characteristics, etc. Such business critical, confidential information must be protected and remain secure. Because of its sensitivity, production data should not be copied to development or testing environments. And because of its business criticality, any change to the structure must be carefully designed and tested.

Except for unit testing, tests may require the databases of development and testing environments to contain suitable test data. To meet this need, partial or obfuscated copies of the production database are commonly used, particularly in testing environments. The data in partial copies is highly sensitive and poses a risk, should protection fail. It is also unsuitable test data, if structural changes have been made. In the case of obfuscated copies, they, too, are unsuitable test data if structural changes have been made, and they also require work to search and identify data suitable for testing.

DevOps helps organizations get suitable test data with automation, virtualization, and test stubs. Automation is used to create synthetic test data—data designed specifically for the requirements of the test case. Ideally, automation also takes care of disposing of any data created for the test directly after the test is done. This practice achieves test isolation, which is needed for parallel, reliable and automated tests.
Database virtualization is used for creating a quick-to-use snapshot of the test data

which was created by automation. Instead of running a set of automation scripts every time a test database needs to be refreshed, a virtual image can be used instead.

Test stubs or mock interfaces simulate the external data source for the component being tested. If the data source is a database, ideally, it should be abstracted by a data access layer which can be replaced with a test stub or mock interface, which provides pre-defined values and simulates a database response.

The most delicate challenge when dealing with databases are structure changes. We recommend special techniques, like "branching by abstraction," which means using an abstraction layer of views and stored procedures for evolving data model structures. Such changes should be accomplished by completely automated, well-tested scripts. This approach makes safe the most critical step of modifying a productive database by performing this delicate task in a completely automated fashion which has been tested over and over in previous stages.

Table 5: Requirements to the areas of process/practice, technology/tools and organization imposed by Data Management

PROCESS/PRACTICES	TECHNOLOGY/TOOLS	ORGANIZATION
• Data Modeling Patterns • Scripts for automating test data creation and deletion • Scripts for automating data model changes and data migration • Database virtualization for efficient restauration of a known-good state	• Service/API testing tool or framework (for automating test data creation) • Test Stubs • Source code repository with version control • Automation facility, e.g., shell script and job scheduler (Orchestrator)	• Automation Engineers part of development team • Easy access to database admin and virtualization specialists

CONFIGURATION MANAGEMENT

Software applications have many dependencies, both internal and external. For instance, they depend on specific versions and specific configuration settings of dependent libraries, modules, application servers, databases, the operating system, network settings—and the list goes on. Configuration management is about documenting and considering all these dependencies, as well as keeping track of all involved artifacts, their versions, and storing them for retrieval.

When errors occur from unintentionally diverging configurations, this causes work to analyze them and correct them. Minimizing such effort is necessary for gaining efficiency and reliability. The DevOps approach relies on automation. Organizations may find it difficult to adopt DevOps because they must automate the process of setting up configurations to ensure that configurations are completely accurate and used correctly. And those automation scripts and configuration settings must be kept safe in a version-controlled repository

Table 6: Requirements to the areas of process/practice, technology/tools and organization imposed by Configuration Management

PROCESS/PRACTICES	TECHNOLOGY/TOOLS	ORGANIZATION
• Never configure manually, always change configurations with a script	• Source code repository with version control • Automation facility, e.g., shell script and job scheduler (Orchestrator)	• Application Operations infrastructure team member embedded into development team

DEVOPS CONTEXT

Digitization is having wide-ranging effects on organizations, and DevOps can help with only part of the changes taking place with operating models. It provides guidance on processes, technology and tools as well as the organizational setup for the part of the operating model dedicated to changes to software in production. Other aspects, for example, collaboration between business and IT are not addressed by DevOps.

What is the sense of delivering fast and frequently if there is no business value created by doing so?

Organizations need tools and technologies to achieve very high levels of automation, and they are available. To effectively benefit from such automation, companies need to understand why short cycle time and close inter-disciplinary collaboration are important goals. In the end, the characteristic trademark of DevOps is a culture founded in lean-agile values. It is one that continuously works to improve the current practices, technology, use of tools and organizational setup.

Many aspects of DevOps and its context are not technology-specific and therefore not the focus of this whitepaper. However, we consider them relevant for understanding and want to remind the reader of them.

> **To effectively benefit from such automation, companies need to understand why short cycle time and close inter-disciplinary collaboration are important goals.**

DEVOPS TECHNOLOGY PATTERNS FOR SAP

When evaluating the compatibility of the Deployment Pipeline pattern with SAP technology, consider what Gartner[5] says: Many companies have trouble coordinating their DevOps-related activities across the enterprise because the coordination affects multiple departments, and there are no industry-wide definitions for DevOps and the associated practices. Another point to consider: There are many ways to achieve the intended pattern. The challenge is in selecting and integrating tools that work.

> **Automation solutions, which are commonly considered essential for DevOps, depend heavily on the technology stack. Therefore, when thinking about the main pattern of the Deployment Pipeline, you will need to consider solutions tailored for the specific technology in use.**

SAP technology is often believed to require different approaches and tools than the open-source and open standards ecosystems, but that is not our view on the matter. Even though the first DevOps success stories originated from open-source and open-standard ecosystems, such success stories are also happening in the SAP ecosystem.

WHY SAP TECHNOLOGY IS DIFFERENT

What is commonly called "SAP technology" primarily encompasses technology based on the proprietary ABAP language, which means software executing on the NetWeaver Application Server for ABAP. SAP has also recognized the advantages of providing software solutions which rely on open programming languages and open standards. Thus, the technology mix is changing steadily towards such technology ecosystem, as can be observed with SAP Cloud Platform.

Nonetheless, a significant installed base of ABAP-based proprietary SAP technology stacks is in operation today and will be for the foreseeable future—S/4HANA, the new "digital core", is based on ABAP, the tool ecosystem for ABAP development and the proprietary execution environments for ABAP code. The proprietary programming language and mostly commercial software ecosystem creates a closed community with high barriers for entry, effectively limiting the crowd which can be leveraged for innovation and experimentation. Therefore, some consider DevOps for SAP technology more challenging.

Another common concern are the monolithic application architecture and the opaque implementation inherent in most packaged, commercial off-the-shelf (COTS) software, not just SAP software.

> **With monolithic application architecture, the concern relates to the impact of changes because of dependencies on shared artifacts, most notably the data model. A change might impact many teams.**

For example, even though a team might not require some planned data model change, the shared nature of the data model requires it to adapt a part of the application custom code they are responsible for.

Packaged software presents the challenge to accept much of the software architecture and implementation as immutable or even unknown. Changing the architecture or implementation is not possible, except for the parts specifically designed for being customizable.

WHY SAP TECHNOLOGY DOESN'T REQUIRE SPECIAL TREATMENT

While SAP technology—just like all other COTS software—limits what can be changed, IT organizations naturally focus on and work on only the parts that can be changed. These are the areas that provide context on the question of whether DevOps is feasible for developing and deploying such changes to production. There is no question that the capabilities described earlier apply to SAP technology at a high level. The main question is if the essential technology elements can be brought to life in a highly automated environment.

Now let's turn to essential tool and technology items which support DevOps capabilities. Figure 5 provides an overview of the tool and technology items previously identified as essential and maps them to associated capabilities, which were introduced in Figure 3. The stages of the Deployment Pipeline provide process context.

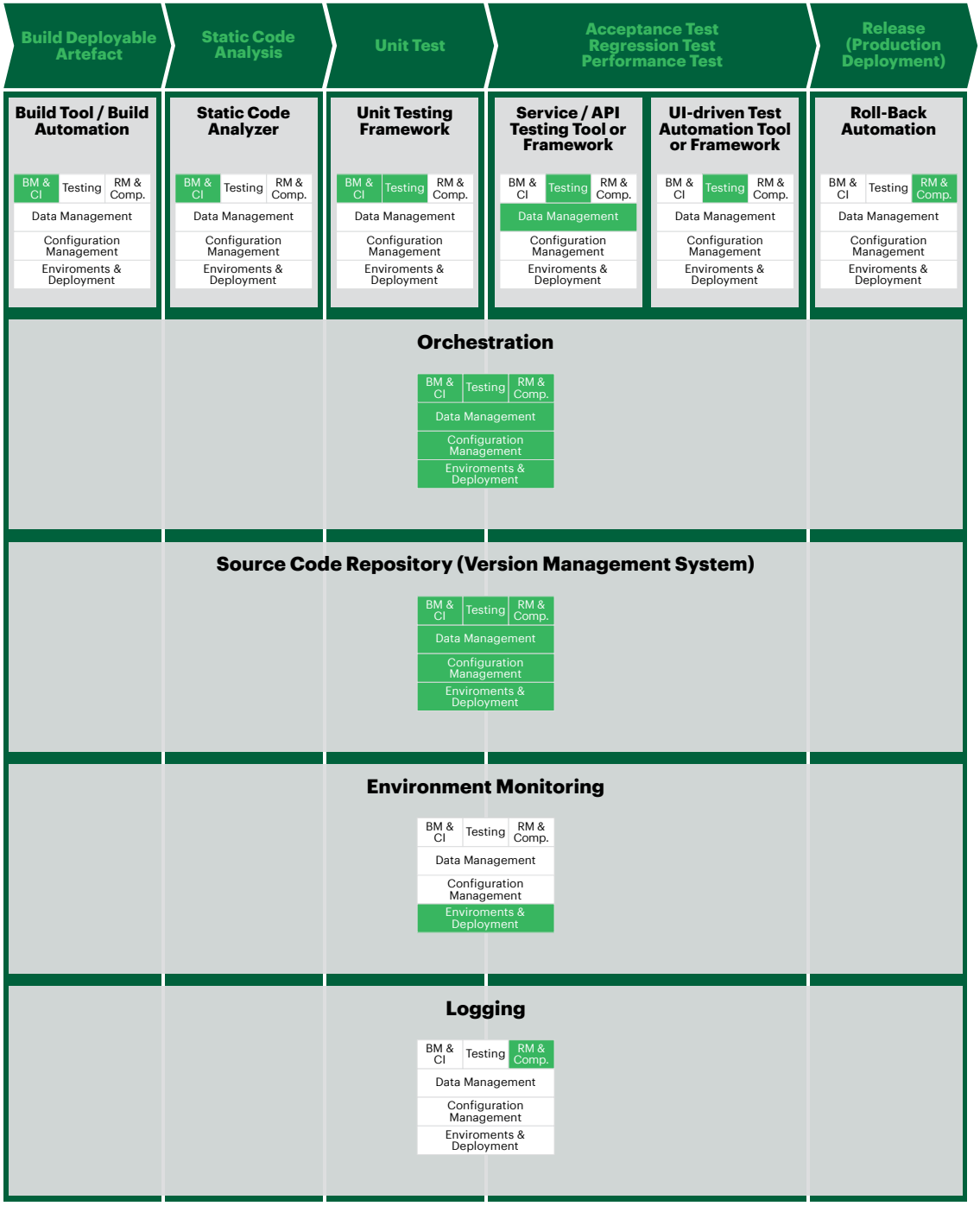
The figure to the right shows that Orchestration and Source Code Repository are essential technology elements required by all capabilities. Most other capabilities don't map to more than one essential technology element, except the Unit Testing Framework and Service/API Testing Tool, which are relevant for two capabilities.

> **Organizations need to evaluate the feasibility within the context of an SAP technology stack for these essential elements that relate to technology and tools.**

Further, you will see our evaluation.

You will also see that we conclude that DevOps for SAP may require tools besides what SAP provides. Fortunately, third-party or open-source community tools can fill the gaps nicely. This is good news, particularly for organizations that are running SAP technology together with software from other vendors or open-source projects. In such cases, the required tools can be used for all software, independent of vendor, which simplifies the tool landscape.

Figure 5: Overview of the technology elements supporting a Deployment Pipeline (depicted at the top, in gray), with each element linked to the capability it supports (highlighted in light green color).



TECHNOLOGY PATTERNS FOR THE AUTOMATED SAP DEPLOYMENT PIPELINE

Accenture's DevOps Professional Community is continuously evolving the technology patterns which can be used to implement an automated Deployment Pipeline for SAP technology landscapes. If you are in search of the most recent thought leadership and patterns, please reach out to us.

The technology patterns are mostly characterized by the tool choices for elements which are not covered by SAP technology. An overview of the technology patterns is given alongside the technology elements of a Deployment Pipeline. For cross-referencing, we marked each capability with the elements that are required to be in place.

When an IT landscape includes non-SAP systems, applications and software, should organizations use SAP software for addressing concerns that apply across the landscape? Or is it better to use independent third-party solutions for this purpose?

Often, we suggest using independent, third-party solutions instead of SAP software, when an IT landscape includes non-SAP systems, applications and software. We recommend this because independent, third-party solutions have broad applicability to other systems and applications in the landscape. Concerns usually include orchestration, source code repositories for non-ABAP code and scripts, environment monitoring, data management, some aspects of logging, and testing beyond unit level.

Two major patterns emerge for an SAP Deployment Pipeline: SAP-native and SAP-external. In the following sections of this paper, we give select, but not comprehensive, tool or software suggestions for both major patterns. We chose these based on our experience using them.

> **The SAP-external pattern of tooling involves many third-party, open-source tools which integrate with SAP application server technology. These tools are also commonly used for DevOps tool chains for technologies other than SAP, thereby providing a unifying, tool infrastructure as a foundation.**

What enables this pattern is the possibility of automatically retrieving relevant meta data, like transport information, from SAP application servers and storing such meta data in common source code repositories like Git. Combined with the possibility of remote-controlling SAP application servers via scripts and based on plans stored in external Orchestrators like Jenkins, the integration becomes complete. The following sections provide further details.

BUILD TOOL/BUILD AUTOMATION

Open-source languages have an overwhelming variety of Integrated Development Environments (IDE), compilers and tools for building deployable artifacts. For SAP technology, the SAP ABAP NetWeaver application server takes care of these considerations seamlessly, a major benefit of developing with SAP ABAP. Solutions for typical challenges like managing and committing source code to a version management system, fetching source code for compilation, linking libraries and building a deployable artifact are all provided out of the box.

The ABAP Application Server manages the definition of a build and automates the building and deployment process. A build is defined as the objects and object versions marked as "active". When changes to objects require a new build, these changes are assigned to deployment artifacts called "transports". The SAP Transport Management System (TMS) deploys transports along transport routes into target systems. So, even copying and installing artifacts from one environment to another is included via SAP TMS.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
• Build definition and automation	• Build Management and Continuous Integration	**SAP native** • Provided with SAP NetWeaver **SAP external** • *Not applicable*

STATIC CODE ANALYZER

SAP NetWeaver for ABAP provides the tool named SAP Code Inspector (SCI), which is capable of validating the syntax, performance, security and adherence to naming conventions of repository objects.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Static code analyzer	• Build Management and Continuous Integration	**SAP native** • ABAP Test Cockpit & SAP Code Inspector: Provided with SAP NetWeaver **SAP external** • SonarCube

UNIT TESTING FRAMEWORK

Unit testing frameworks exist for most programming languages and are commonly referred to as xUnit, where x is a placeholder for the programming language identification. For Java, there's JUnit, for .NET there's NUnit, etc. For ABAP, the unit testing framework is called ABAP Unit, which is provided with the SAP NetWeaver ABAP application server and is integrated into the ABAP Workbench.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Unit testing framework	• Build Management and Continuous Integration • Testing	**SAP native** • ABAP Unit: Provided with SAP NetWeaver **SAP external** • *Not applicable*

SERVICE/API TESTING TOOL OR FRAMEWORK

As Mike Cohn wrote in December 2009 in his blog "The Forgotten Layer of the Test Automation Pyramid"[6], the automation of tests should be done at the right level. One is frequently forgotten: The Service Test level (or layer). This layer is often overlooked by developers, who rather concern themselves with unit tests, as well as traditional testers, who only know how to test through the user interface because they don't have programming skills.

For the SAP ecosystem, this lack of consideration is manifested by the fact that there's no (dedicated) testing framework or tool provided by SAP for testing at this layer.

The preferred approach for Service/API testing makes use of the fact that up-to-date SAP applications expose functionality via OData Web services, which can be tested with third-party Service/API testing tools.

The functionality exposed via OData API also enables organizations to create test data by using Service/API testing tools to execute data input procedures, while using variables for reading data in looped execution.

In summary, for implementation of a Deployment Pipeline, the tools used in proven technology patterns are provided by third parties or are open-source.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Service/API Testing Tool or Framework	• Testing • Data Management	**SAP native** • *None* **SAP external** • SOAP UI • Postman

UI-DRIVEN TEST AUTOMATION TOOL OR FRAMEWORK

Powerful tools for automating tests driven by the UI are available both from SAP and third parties. Keep in mind that SAP GUI is not browser-based, which prevents the use of typical browser-based test automation and therefore requires SAP CBTA or similar third-party tools for UI-driven test automation.

Third parties offer many alternative tools for testing via browser-based GUIs such as SAPUI5.

Some organizations prefer to write tests first, e.g., before creating the UI, which is not possible with all tools. Such capability is typically provided by a Domain-Specific Language (DSL)—e.g., Gherkin—and requires writing test fixtures, e.g., programming code which handles the execution part. This means testers and developers will need more skill for test-first approaches.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
UI-driven test automation tool or framework	• Testing	**SAP native** • SAP CBTA **SAP external** • Worksoft Certify • HP UFT • Selenium

ROLL-BACK AUTOMATION

DevOps culture embraces learning from failures to minimize and reduce overall failure as a result. An important part of this is roll-back automation, which minimizes the fall-out from failed releases and resolves such situations swiftly and relatively painlessly. Unfortunately, SAP doesn't provide straightforward roll-back automation but requires failed functionality in the related transport request to be deactivated in a preceding environment on the transport route. This is followed by repeating the transport process, but this time transporting the deactivated functionality into the environment where roll-back is required. The procedure is easy to execute but requires custom scripts or third-party software for automation.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Roll-Back Automation	• Release Management and Compliance	**SAP native** • *None* **SAP external** • Custom-built scripts and Orchestrator • Basis Technologies Transport Expresso

ORCHESTRATION

Several capabilities require integrated automation of multiple activities in the right order. The technology or tool must be able to execute shell scripts, schedule automation jobs, and trigger automation jobs or automated testing scripts. Such a component is called an Orchestrator.

For Build Management and Continuous Integration, missing automation capabilities include triggering automated service and UI tests, raising resulting defect tickets, and automated sending of test result emails to developers. Continuous Integration/ Continuous Delivery (CI/CD) servers used for open standards development can be integrated with SAP technology to close this gap.

For Environments and Deployment, automation of environment configuration and provisioning is beyond the scope of the SAP technology stack. For these infrastructure aspects, SAP technology requires the same treatment as open standards technology.

For Release Management and Compliance, orchestration is required for Roll-Back Automation.

For Testing, orchestration triggers the automated tests and automation scripts which integrate third-party tools.

For Data Management, orchestration is required for creation of synthetic test data and automation of data model changes via scripting.

For Configuration Management, SAP technology is not concerned with parts of the technology stack which require automated changes of configuration, like the operating system or network configuration. This presents the same case as for Environments and Deployment.

While there are some challenges with orchestrating the entire Deployment Pipeline with SAP technology, this pattern is prevalent and represents an option.

Therefore, patterns are differentiated at the highest level when the decision is made to use an SAP-external or SAP-native Orchestrator for the technology pattern for creating an automated Deployment Pipeline with SAP technology.

SAP native orchestration can be achieved with Solution Manager and ChaRM, while SAP-external orchestration can use CI/CD servers such as Jenkins or Bamboo.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Automation facility, e.g., shell script and job scheduler (Orchestrator)	• Build Management and Continuous Integration • Environments and Deployment • Release Management and Compliance • Testing • Data Management • Configuration Management	**SAP native** • Solution Manager and ChaRM **SAP external** • Atlassian Bamboo • Jenkins

SOURCE CODE REPOSITORY (VERSION MANAGEMENT SYSTEM)

The high level of automation, which is the hallmark of the Deployment Pipeline pattern, requires strong configuration management across all source code, scripts and configurations, including not just the system or application, but also the Deployment Pipeline and environments.

For Build Management and Continuous Integration, the NetWeaver Application Server provides this functionality.

For Environments and Deployment, artifacts like operating system configuration require this functionality in the form of a third-party solution.

For Release Management and Compliance, this functionality is required by the artifacts relevant to roll-back automation and deployment automation. As these two elements also extend beyond the scope of SAP NetWeaver application server capabilities, a third-party solution is required.

For Testing, the scripts for triggering test automation require storage in a version-controlled code repository.

For Data Management, this functionality is required for storing scripts which automate creation of synthetic test data and automate changes to the data model.

For Configuration Management, this functionality is fundamentally required. Because Configuration Management comprises more than just the SAP technology stack, a third-party solution is required.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Source Code Repository with version control	• Build Management and Continuous Integration • Environments and Deployment • Release Management and Compliance • Testing • Data Management • Configuration Management	**SAP native** • NW, only for ABAP source code, DDICT, configuration and customizing entries • **Gap**: any non-SAP script or asset **SAP external** • Git

ENVIRONMENT MONITORING

When automation replaces manual work, the importance of monitoring increases. SAP Solution Manager provides monitoring facilities for SAP system landscapes. However, the scope of environments is larger than the scope monitored by SAP Solution Manager.

Particularly when moving towards infrastructure automation, the SAP native monitoring capabilities will have to be complemented with third-party solutions.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Environment Monitoring	• Environments and Deployment	**SAP native** • Solution Manager for monitoring SAP systems • **Gap**: Infrastructure monitoring **SAP external** • Nagios

LOGGING

One of the guiding principles of DevOps is learning from feedback. Root causes of failures or errors are an important source of feedback. For identification of root causes, organizations will need robust logging features. This is not a new kind of requirement and SAP technology provides respective features.

TECHNOLOGY ELEMENT	DEVOPS CAPABILITY	PATTERN AND REALIZATION
Logging facilities for all deployment automation	• Release Management and Compliance	**SAP native** • Provided with SAP NetWeaver **SAP external** • Various

EVALUATION SUMMARY

Our detailed evaluation of essential technology and tool elements revealed some gaps in native support by SAP technology. However, all these gaps can be addressed with third-party solutions, which is evident by our proposed solutions for SAP-native and SAP-external technology patterns. Therefore, we conclude that DevOps with SAP technology is certainly feasible.

The findings of this evaluation are summarized in Table 7.

Table 7: Evaluation results by DevOps capability

DEVOPS CAPABILITY	EVALUATION
Build Management and Continuous Integration	Most elements for build management and continuous integration are provided by SAP's proprietary application servers and development tools. A gap which requires third-party software is pipeline orchestration.
Environments and Deployment	The scope of environments and deployment expands beyond the systems provided by SAP, e.g., network and operating system configuration and scripts. This explains why this capability requires third-party software for all elements.
Release Management and Compliance	While release management is a core capability supported by SAP Solution Manager, gaps exist in conjunction with pipeline orchestration and extensive automation. The main gap which requires custom or third-party solutions is roll-back automation.
Testing	Automation of tests on all three levels of the test automation pyramid requires third-party tools. The Unit Testing level is enabled by SAP ABAP workbench functionality, but service/API testing and test-first UI-level testing are not.
Data Management	Strategies for data management may involve database virtualization, but the very minimum requirement is automated creation of test data. The automated creation of test data can be achieved with a variety of approaches, e.g., via API or UI. Depending on the approach, third-party software may be necessary.
Configuration Management	Like environment management, the scope of configuration management extends beyond SAP systems. Therefore, third-party tools are required for version control and automation.

CONCLUSION

“SAP back-end systems are not suited for DevOps” is a false statement. Indeed, DevOps technical practices sometimes use technical solutions specific for SAP back-end systems, making some adjustments when the required technical capability is available out of the box and doesn’t need to be provided by a tool.

For example, a source code repository with version control for application code is provided with any SAP application server.

Such specific adaptations to the SAP technology stack don’t mean the technical practices are more difficult or even unfeasible. They simply require the right knowledge and experience to bring these technology practices to life.

A big part the vision of DevOps is enabling teams to act as autonomously as possible for bringing changes to business stakeholders. This autonomy, for which the collaboration between development and operations roles is key, in turn enables true ownership and accountability.

Imagine you had your kitchen remodeled with extensive changes to the electrical wiring. If later you discovered there's something wrong with the wiring, who would you call? Who is in the best position to fix the problem? That would be the electrician who helped remodel your kitchen in the first place. Not only would this electrician most likely know what the problem is, based on his or her knowledge, but they also have their reputation on the line and a self-interest in resolving the problem.

The same should be true for all your applications—or in case of SAP technology, where applications are large, for coherent clusters of features. These clusters of features might even be called "products" in the agile sense of the word.

With this vision in mind, early successes and hands-on experience are key for gaining momentum among stakeholders in the organization. Accenture offers an open-sourced, openly available DevOps platform with pre-built assets for SAP, so-called DevOps cartridges. The cartridges are for kick-starting a DevOps transformation and achieving such early, hands-on experience.

Some challenges in implementing a Deployment Pipeline with SAP technology are harder than others. Specifically, data management, test automation, dashboards and the implementation of an Orchestrator can prove difficult. The difficulty is compounded with the concern that SAP systems are not the only technology stack in use. Enterprise-wide solutions in these areas provide synergetic benefits regarding required skills and complexity of the overall technology landscape, therefore careful selection and design of solutions is critical.

Accenture's DevOps transformation services can support overcoming such challenges based on ample experience and knowledge, both with DevOps for SAP and other technology stacks.

AUTHORS

DR. LEO LEHR

CMT SAP CoE Lead at Walldorf

leo.lehr@accenture.com

DANIEL PHILIPP

Business and Technology Innovation Principal Director

daniel.philipp@accenture.com

REFERENCES

1. Dr. Nicole Forsgren, Gene Kim, Jez Humble, Alanna Brown and Nigel Kersten; 2017 State of DevOps Report; https://puppet.com/resources/whitepaper/state-of-devops-report ; 2017, Puppet and DORA (DevOps Research and Assessment).

2. Jez Humble, Chris Read, Dan North; The Deployment Production Pipeline; Agile Conference 2006; IEEE 9055957.

3. The focus here is not the maturity model, but the definition of capabilities. DevOps has evolved significantly in the last years and so have maturity models.

4. Jez Humble, David Farley; Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation; Pearson Education, 2010.

5. "a lack of industrywide definitions of DevOps and its associated practices often inhibits the enterprisewide coordination of DevOps-related activities" from Principles and Practices of DevOps That I&O Leaders Need to Cultivate; 12.01.2017; G00320004; Gartner.

6. https://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test-automation-pyramid.

ABOUT ACCENTURE

Accenture is a leading global professional services company, providing a broad range of services and solutions in strategy, consulting, digital, technology, and operations. Combining unmatched experience and specialized skills across more than 40 industries and all business functions—underpinned by the world's largest delivery network—Accenture works at the intersection of business and technology to help clients improve their performance and create sustainable value for their stakeholders. With approximately 449,000 serving clients in more than 120 countries, Accenture drives innovation to improve the way the world works and lives. Visit us at www.accenture.com.