Global Knowledge®

Expert Reference Series of White Papers

The Intersection of DevOps and ITIL®

The Intersection of DevOps and ITIL®

Paul M. Dooley, MBA, IT Service Manager, ITIL® V3 Expert

Why All the Buzz about DevOps?

DevOps is more of a philosophy or way of working than it is a formal framework or standard. Nevertheless, the approach deserves merit, as it goes to the core of a tension in most IT organizations—the need to be responsive to business change, while maintaining a stable, highly available IT infrastructure, and delivering quality services that meet the needs of the business. What does DevOps have to do with ITIL®?

Let's start with ITIL. ITIL is the defacto global framework for delivering and managing technology as a service to the business. As a proven and well-adopted framework of IT best practices, ITIL includes the notion that a service is a means of delivering value to customers by facilitating the "outcomes" they want (achieving business results), without them having to deal with the cost and risk associated with that service. ITIL maintains that customers want outcomes—the result of carrying out the service—and they understand that a service provider is required to provide whatever IT services are necessary to help them achieve those outcomes, through the application of people, processes, and technology. ITIL is vendor neutral (doesn't require tools specific to a given vendor), not prescriptive (it must be adapted to the organization that wants to practice it), and a proven best practice for IT (it just works).

While DevOps emphasizes collaboration and communication, so does ITIL. In fact, there is no inherent conflict between the DevOPs movement and ITIL—upon examination, it becomes obvious that the two are very complementary. DevOps provides us with "new light" in which we can examine the ITIL framework in several key areas, improving core processes, functions, and principles within ITIL.

DevOps and ITIL—In Conflict or Complementary?

Key Elements of a DevOps Approach

DevOps is not a product or technology, but rather an emerging "methodology" or way of working between two key groups within an IT service provider organization. DevOps:

- Should be holistic—involving people, process, and technology
- Can be thought of as an "extension" of Agile
- Is about breaking down "barriers" between application development groups within organizations, and their service and support function
- Is about ways to improve collaboration between development and operations teams
- Includes the concept of automated testing, as well as CI monitoring

More a philosophy or way of working than it is a formal standard or framework, DevOps is short for Collaboration between Development and Operations. "DevOps" arose as a movement within IT best-practices a few years ago when IT managers began to realize that something needed to be done to close the communications and collaboration gap between development groups and support operations staff, and to speed responsiveness to the business and the delivery of service updates and changes.

With the trend toward mobile devices, the "economy of apps" was born. Work began its transition to a mobile workforce, and applications—which had a larger footprint and used to reside on desktops—began their transformation to smaller, portable "apps."

Office applications began to transition to the "cloud," and an ever-increasing amount of work began to be done on tablets, smartphones, and laptops. In conjunction with the trend toward mobile work, competition for customers and users continued to grow between enterprises. The balance of eCommerce shifted to mobile devices, and it became imperative for organizations to stay ahead of their competitors—with regular updates to "apps" that could empower their workforce, attract customers, and perhaps amount to a competitive differentiation.

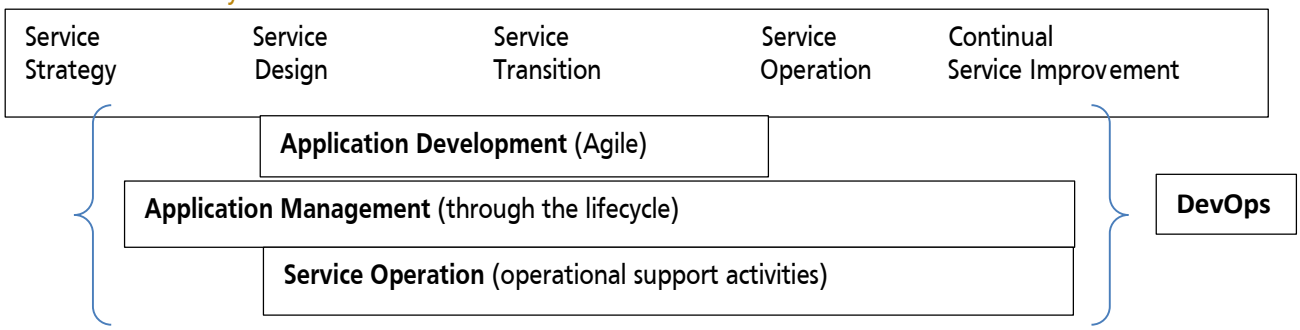
Developer organizations began to feel the pressure to be more response to business demands for functionality improvements in apps. The notion of "Agile" development teams was born, and development groups began to develop and deploy new, updated releases of code on a more frequent basis. Agile, which is a philosophy of how to carry out the development and deployment of apps, is also supported by methodologies such as Scrum. It provides the framework within which development organizations can work more productively to design, code, and ship smaller increments of code more frequently to meet customer demand.

How DevOps, Agile, and ITIL Relate

Whereas Agile is a methodology for speeding the development and deployment of applications (using techniques and processes such as Scrum), DevOps is a philosophy or way of working that has the potential to realize greater benefits and faster delivery through such Agile methods. DevOps is a "value add" approach to Agile development, emphasizing:

- Continuous involvement and engagement of operations teams with internal development teams, through the development lifecycle
- That operations teams should be engaged as early as possible, in fact right from the start—during the formulation of the vision and charter for the service solution
- Operations teams should also provide input to the technical and application requirements for the service, and advise on the feasibility of the proposed release schedule
- Improved deployment frequency, enabling faster time to market of new/improved application functionality changes
- Lower failure rate of new releases, more frequent fix updates, and faster recovery time in the event of a change failure
- The application of service automation to accelerate common "process models" such as a standard change, or routine release update

The ITIL Service Lifecycle



These notions of collaboration, optimized communication, and business responsiveness are not foreign to ITIL. In fact, the ITIL framework stresses the need for early involvement of technical management and application management functional teams in service strategy, service design, and service transition. ITIL is very clear that service operation functions—in particular, technical infrastructure teams—should provide early advisement to the design and development team on the supportability and fitness of the application to the live environment. They should also provide input on the technical architectures for the "service solution," including service architecture, application architecture, network architecture, information architecture, database architecture, and so forth.

Application management staff (being responsible for the entire lifecycle of all applications—not just those developed in house), should be engaged very early in the life of a service. In fact, they should be the ones to work with the business, the person assigned as the business relationship manager (BRM), and the service owner to identify, define, and document the complete "service solution" requirements, and ensure these are captured in the service charter for the new or changed service being considered. According to ITIL, application management teams should work closely with development groups, who stay focused during service design and service transition on rapid development and deployment of applications that are a part of the total service solution. While development focuses on development activities across service design and service transition, application management manages the app across the entire service lifecycle, from strategy, through design and transition (working with development), and on into operation and continual improvement. This approach ensures that the service solution (which contains the app), not only launches well, but continues to perform during live operations, meeting the needs of business customers and users.

The importance of defining and implementing effective cross-functional communications, along with automated support systems, is also underscored by ITIL. All of the various stakeholders, from front line service desk staff and technical and application support specialists, to developers and customers, must be linked through well-planned and efficient communications channels so that escalations can be handled quickly, and the IT support organization can truly achieve optimum flexibility and responsiveness.

Like DevOps, ITIL is also very keen on the use of "models," and the application of service automation to increase efficiency, speed execution, and lower costs. What ITIL emphasizes, however, is that to be successful, an IT service provider must first be strategy-driven, and process- before it can realize the maximum benefits of automation. The standard change, or release "model," must first be documented and analyzed, the "gaps" must be taken out, and the process and interfaces must be streamlined and automated. Only then will maximum benefits and throughput be achieved.

Four Reasons Why a DevOps Approach Adds Value to ITIL

The reason for DevOps, and its opportunity to add value to the ITIL framework, lies in the tension created by forces at work within most IT service providers.

First, on the one hand, development organizations are pressured by the business to deliver regular, frequent updates to apps, which are a key component of new and changed services. Small, frequent improvements in functionality potentially boost user productivity, but also carry the risk of destabilizing the production environment, affecting the availability of live operational services, or causing destabilizing side effects. Operations, on the other hand, is charged with maintaining a "balance" between being responsive to the business, and maintaining a stable IT infrastructure that provides high availability, and delivers the service value customers and users desire.

How DevOps adds value: by encouraging improved levels of communication, collaboration, and accountability between development and operations.

- Through improved communications and collaboration, a DevOps approach to ITIL can bring visibility and a greater appreciation to development teams of the challenges faced by service operations.

- As a result, greater attention can be paid during development to assessing the impact of the new feature/functionality on existing IT support resources, capabilities, and supporting infrastructure services. How these support services, resources, and capabilities should be augmented or adjusted can then be factored into the design plan and development activities, resulting in a service that not only features new and improved functionality, but also any improvements required in the area of IT supporting services, people resources and skills, and supporting systems and tools.

Second, development and operations teams often report to different senior managers within the organization. Development's focus is on building new and improved software applications, responding to customer business units at an ever increasing rate. Secondary thought is given to how another part of the same organization will end up supporting this new or changed app, and what they will need in order to be effective. Emphasis is on delivering incremental chunks of new and improved software on a continual basis, matched to the customer's specifications—without much regard to how those improvements work or perform in production. Often the attitude is, "The service desk will take care of that."

DevOps adds value by helping overcome organizational barriers between development and operations teams.

- By encouraging development to work more closely with operations, a DevOps approach helps overcome barriers, facilitating alignment between these internal teams. Development teams benefit from a broader exposure to the needs of operational support during design and development activities. Operational teams benefit by receiving early exposure to the design and development activity of new/changed services, and their supporting apps.

- As a result, operational supportability is more thoroughly considered in design and development activities, and the deployment and operation of the live "service" is much more effective.

Third, it important to realize that "value" is a product of functionality, as well as the performance of that functionality. Not only what the app does, but also "how it does it," is important. Is it available enough for the customer and user? Is it easy to use? Does it have enough capacity in terms of storage, bandwidth, and compute

power? Is it secure enough for the customer and users? Development's focus is primarily the software app, and the functionality delivered by that app. In production, however, the new/improved "app" is running on a number of supporting infrastructure services—network services, supporting platform services (mobile, smartphone, or other device), interacting with or supported by other apps or services.

The app is in reality an element (albeit an important element) in a "service," which is composed of multiple elements and supporting services or components. The challenge of operations is to ensure that the whole "service," of which the app is a key part, is delivering the value sought by the user— in terms of the desired functionality, but also in terms of its performance.

DevOps adds value by assuring a uniform understanding of "value" across IT service functions.

- DevOps helps assure that operations and development, as well as other functions, appreciate that it is the "service" that delivers the value to customers. According to the ITILframework, the "value" of a service is a product of the utility delivered by the service, but also the warranty of that service. Think of "utility" as the functionality of the service – what it does, its "fit for purpose". Users definitely want utility, or functionality. But they also need sufficient levels of "warranty" in order to use the service. Warranty is the assurance that one is able to use the service when required (in other words, it's "fit for use"). Aspects of warranty include acceptable levels of service availability, capacity, continuity, security, and usability. For example, you might have a great smartphone with a cool app that provides lots of functionality, but what good is that if you can't connect to a cellular or wireless network in order to effectively use that app? Without the desired levels of warranty (availability, capacity, continuity, security and so forth), the user will not be able to realize the full extent of the "value" from the service.

- ITIL, when used with a DevOps approach, places emphasis not only on the development and timely delivering of new and improved functionality, but also on the ongoing performance of that functionality when in live operation.

Fourth, development groups are typically focused on carrying out time-bound, project-oriented activities—developing discrete chunks of software code within limited timeframes, in order to meet deadlines and deliver increased functionality to apps in production. Their priority is on developing and delivering improved functionality—not so much as how that functionality will operate day in and day out while in production. Operations, in contrast, is charged with supporting both the functionality and the ongoing performance of the "service" of which the application is a major component. And the service must not only deliver the functionality the user seeks, but must also perform to acceptable service levels—levels of availability, capacity, and so forth—on a daily basis.

DevOps adds value by ensuring early participation of operations teams in design and development activities

- Often IT operations is not given the opportunity to provide input into design and development activities. The mindset is, "The service desk will take care of any supportability issues we haven't thought of." The problem is that supportability is an after-thought, and services become very labor intensive and time consuming to support when launched into production.
- A DevOps approach to service design supports the notion of getting early involvement of both development and operations teams in design activities. By encouraging collaboration between development and operations, DevOps also helps to provide operations with early visibility to new and changed apps so they can be positioned to deliver effective support when they go live.

The Intersection of DevOps and ITIL, Enabling Improved Processes

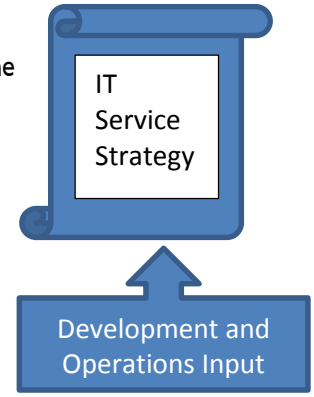
DevOps can indeed be a positive influence on ITIL, enabling improvements in various processes across the service lifecycle. A more collaborative DevOps approach adds value in service strategy, by ensuring that operations and development functions are included during the development of overall IT business strategy. A DevOps approach also adds value during service design, by ensuring that design coordination includes active participation from both operations and development—resulting in the potential for designing improved functionality and performance, as well as operational support.

During service transition, a DevOps approach facilitates the effectiveness and efficiency of both change management, and the process of release and deployment management. By encouraging greater collaboration between development, operations, and indeed all IT functions, a DevOps approach can facilitate knowledge management as well. During service operation a DevOps approach can help ensure closer collaboration between the service desk and backline support and development teams—resulting in more effective incident and problem management. It also emphasizes the need for applying automation to change management as well as release and deployment management "models—thereby facilitating the frequency of deployments, more consistent delivery of new and changed releases, and improved quality.

Continual service improvement (CSI) also benefits from a greater level communication and collaboration between development and operations teams, resulting in a higher level of teamwork, synergy, and creativity. The result is a greater likelihood of identifying improvement opportunities across the lifecycle, many of which will be turned into formal CSI improvement proposals – some of which will be approved and chartered for implementation. The result: higher performance, lower costs, and improved service quality.

DevOps in Service Strategy: Strategy Management for IT Services

This process—normally carried out by IT executives—defines, documents, and communicates the vision, mission, and goals of the IT service provider across the organization. It ensures all departments and functions have a sense of common purpose, shared goals, and a sense of what constitutes "quality" services for customers and users.



Without this hi-level guiding process, along with its direction, many IT service providers find an absence of common goals and objectives across all groups, including development and operations. Hence various IT functional groups are often not assisting one another in achieving common goals, and may even be working at cross-purposes!

How a DevOps Approach Adds Value to Strategy Management for IT

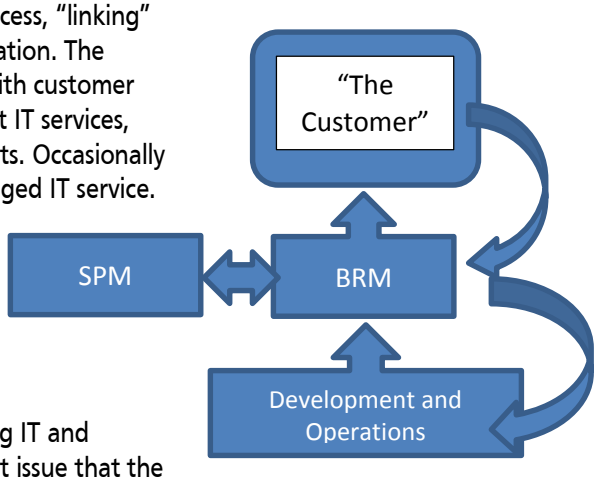
First and foremost, IT executives must plan, communicate, and implement a strategy for the IT service organization and its services, defining a set of shared goals and objectives that span all teams—including development, operations, and other functions. A DevOps approach here would encourage IT executives to consider input from operations and development teams during the creation of or updates to the overall IT business strategy and direction. In this way, development and operations priorities are factored into the go-forward plan for IT; and they in turn are also on board with, and are working toward, achieving common hi-level organizational goals during design, development, and deployment of services.

Secondly, it's important that the strategy management process define and document what constitutes "quality" and success across the service lifecycle. Without a common definition and understanding of what constitutes "quality"—whether that be a quality design, a successful transition, or a successful deployment during service

transition—development, operations, and other IT functions will be left to their own devices to define "success." Once the criteria for what constitutes "quality" service is defined, IT executive management must communicate this broadly across all development, operations, and other teams. An evaluation of success at each stage in the service lifecycle can then be undertaken, based on the terms of this criteria.

Business Relationship Management (BRM) and Service Portfolio Management (SPM)

The BRM process is the ITIL "customer relationship management" process, "linking" customer business unit managers with the IT service provider organization. The BRMs (the IT customer relationship managers) must regularly meet with customer decision makers, providing feedback to them on the quality of current IT services, and seeking to understand any new or changing business requirements. Occasionally the BRM may detect and document the requirements for a new/changed IT service. But without appropriate involvement from design and development, the BRM may not accurately communicate to the customer the potential of new technology to meet these needs; or the BRM may not adequately convey the features and performance of projects underway which also might help meet new/changing business needs.



Operational support considerations are also left out of the loop during IT and customer "service review" discussions, unless there is a chronic support issue that the customer is raising.

How a DevOPs Approach Adds Value to BRM and SPM

The BRM process should find ways to include representatives from both development and operations during the execution of its responsibilities. For example, representatives from both development and operations could be included in periodic "service review" meetings with the BRM and the customer, where the BRM is reviewing the previous period's service activity, and discussing any changing business requirements. This would provide early visibility to both development and operations of any anticipated changes in the business environment, which could impact services currently in design, development, or deployment.

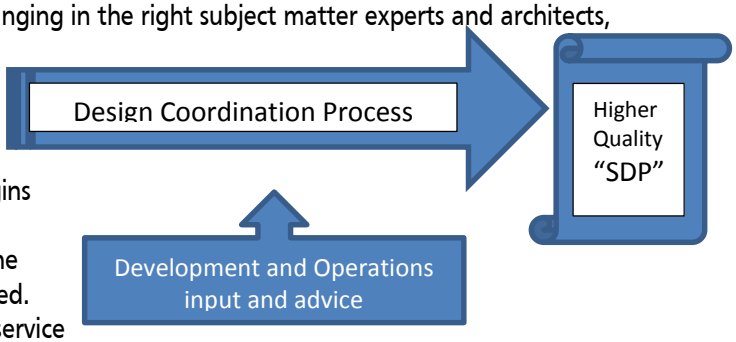
Being included in periodic "service review" meetings also allows development to provide feedback to the customer, when and where appropriate, on current development activities underway (such feedback should be coordinated with the BRM, the service portfolio manager, and application management rep). Operations also benefits from participation in service reviews, as they are able to provide direct feedback to the customer regarding any current operational support issues.

Closer ongoing collaboration and communication between the BRM, customer business unit managers, and operations and development teams will also facilitate a clearer definition of business unit needs and requirements for any new or changed services. This, in turn, will benefit service portfolio management, resulting in a better understanding of what the proper "mix" of services are at any given time, and what new or changed services might be needed from time to time to better serve customers.

Involving development and service operations in periodic "service review" meetings can ensure that feedback on any functionality short comings, or chronic support issues, is clearly conveyed and understood by both development, and operations. Development gets clear and direct feedback on application problems, or opportunities for improvement in functionality or performance. Involving operations is service review meetings also provides service operations with the opportunity to provide direct feedback to customers regarding any changes or improvements in operational policies and support procedures.

DevOps in Service Design: Design Coordination

Design Coordination is the process responsible for coordinating all of the activities during service design and development so that a comprehensive Service Design Package (SDP) can be documented and produced for the new or changed service. Normally a project manager or lead is assigned as the "design coordinator," driving the process of design coordination—scheduling meetings, bringing in the right subject matter experts and architects, and coordinating activities and resources—to ultimately produce the specifications for the new/changed service, or the SDP.



During design activity, a design team is formed, and begins focusing on designing "all" aspects of a new or changed service. Unless the right stakeholders are brought into the design team, important considerations may be overlooked. Design and development activity for a new or changed service is narrowly focused on that specific service—not necessarily taking into consideration how the new/changed service may impact essential supporting services (ie., various IT infrastructure services, operational monitoring, or service desk support). Unless operations is also included on the design team, important operational support considerations may also be overlooked. Likewise, unless development is mapped into the design process, optimized technical approaches may be neglected. The result would be a service design comprised of less than optimized application and technology components, as well as a lack of provision for adequate support during live operation.

How a DevOps Approach Adds Value to the Process of Design Coordination

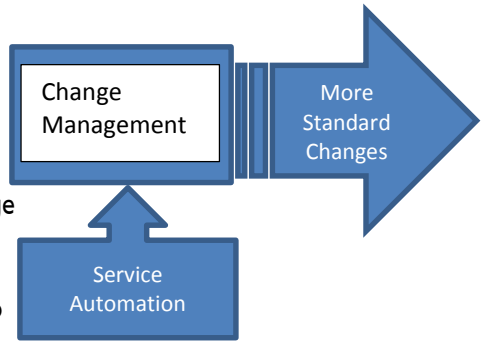
A DevOps approach adds value here by ensuring that the project lead driving design coordination includes representatives from both operations and development on the design team. In this way, both designing to satisfy operational support requirements and optimized approaches to technical and applications architecture can be assured.

Including members from both application development and application management teams can also ensure close collaboration between the management team (responsible for *overall lifecycle management* of apps), and the development team (responsible for the *development and coding* of the apps). Both teams are crucial to the successful definition, design, development, deployment, and ongoing operation of the apps, which are such an important component of most services.

DevOps in Service Transition

Change Management

According to ITIL, the process of change management controls "all" changes across the service life cycle—from strategic changes (new services and major changes), to design changes, to tactical and operational changes—resulting in a higher number and percentage of beneficial changes, while minimizing the impact of disruptive changes.



Unless properly planned, resourced, and executed, change management can degrade into a bureaucratic process, and end up being a "barrier" to carrying out responsive business or IT changes. Key factors contributing to such poor process performance include poor process policies and procedures, a failure to leverage process automation, and not bringing in the right stakeholders to change management meetings to review and advise on changes in a timely manner. In such cases, the process works too slowly to be truly responsive; change management become a reactive process, and primarily a scheduler—in contrast to what it should be: a change "management" process. In such cases, proposed changes are either accepted or rejected, without much information as to why, or what other information could have been provided in order to merit acceptance.

In addition, many of the smaller, frequent changes to "apps" are still considered normal changes, requiring submission to the weekly Change Advisory Board (CAB) in order to be approved for release and deployment.

How a DevOps Approach Adds Value to Change Management

To provide a basis for a DevOps approach to change management, start first with an effective change management framework consisting of a well-thought-out set of policies and standard operating procedures (SOPs). Factor in a DevOps approach when devising your policies and SOPs:

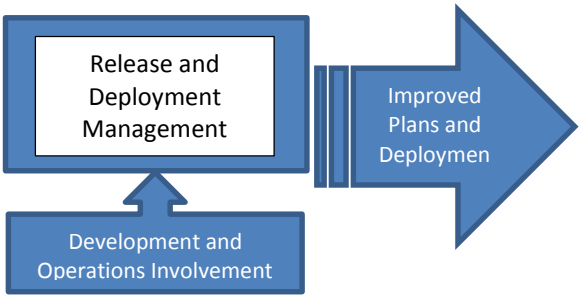
- Include clear criteria for what constitutes an emergency, normal, and standard change
- Develop a strategy to minimize the number of emergency changes, and maximize the number and percentage of changes that can be handled as "standard changes" by an operational team, leveraging automation to deliver the change rapidly and effectively
- Ensure that guidelines are set for the right "standing members," and which "temporary" members are brought into the CAB to provide advice on approving or rejecting proposed changes
- Improve participation of development and operations in change management, along with collaboration between these teams, by including representatives on the CAB when and where that makes good business sense. This will facilitate the impact analysis of a proposed change, and raise the quality of feedback to the change manager.

DevOps stresses that where possible, change management should leverage automation to speed the progress of a proposed change through the steps of the process. ITIL encourages this as well. Change management reviews and approvals can often be automated via workflow and automated tools. A DevOps approach would take advantage of automated tools to expedite the routing of RFCs and attachments, CAB review, and approval of normal changes to speed throughput. The result: less bureaucracy, higher process performance, and a higher level of stakeholder satisfaction.

In terms of the review and evaluation step, consider moving the emphasis from accept/reject, to "what information or actions are needed in order for me to approve?" When returning a rejection notice, include information that can facilitate re-submittal and future approval.

Finally, a DevOps approach would re-evaluate smaller, frequent updates to apps, and if they fall within documented "standard change" criteria—low risk, repeatable, and able to be carried out via an SOP with service operations resources and automation—delegate the handling of these changes to operations as "standard changes." Over time this will reduce the percentage of "normal" changes that require a CAB review, and increase the percentage of changes that are being handled as standard changes, carried out via a service operations team, SOPs, and automation.

Release and Deployment Management



The release and deployment management process is responsible for planning releases of new and changed services, seeing that they are properly built, tested and packaged, and releasing and deploying them into the live environment with minimal "side effects" so that maximum value can be delivered to customers and users.

Without adequate participate from development and operations in this process, incomplete release and deployment plans can be the result—lacking attention to required levels of testing (to validate functionality, performance, and usability). During this process, it's critical that the test environment mirror the live environment adequately; without the involvement of operations in this process, critical aspects of the live environment might be left out of testing—resulting in poor quality testing, and perhaps a failure to achieve service acceptance by the customer.

Another potential problem area is executing activities in release planning and deployment serially. When release and deployment planning is put off until the very final stages of development and testing, overall time to deployment and live production suffers dramatically.

How a DevOps Approach Adds Value to Release and Deployment Management

A DevOps approach encourages the integration of development and operational teams into release planning activities, so that optimized building testing and packing can be adequately planned for. DevOps would stress including development and operations into the service validation and testing process as well, so that proper types and levels of testing can be planned and carried out—resulting in a service and supporting components that meet the quality requirements of the business and the customer.

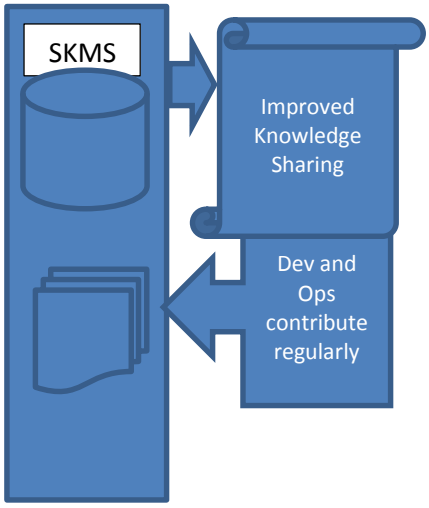
Development and operations support teams would also be integrated into deployment activities for a new or changed service, its applications, and supporting services. This makes sense for service operations, as they will begin taking on the production support of the service after it goes "live." But involving development in deployment activities also makes sense, as development is thus exposed and experiences firsthand the performance of the application in the hands of customers and users.

To reduce overall time to live operation, a DevOps approach would stress that release and deployment teams should be working in parallel as much as possible. In places where you can't break things into parallel efforts,

DevOps emphasizes looking for ways to start release planning and deployment earlier in the service life cycle. Unless there are clear dependencies between steps, a DevOps approach would stress working on deployment planning while release planning is still underway; and begin release planning while in the midst of completing testing and validation. When and where possible, a DevOps approach also stresses the need to engage testing and validation in parallel with development activities. The end result: total elapsed time involved in release and deployment activities is collapsed.

Knowledge Management

According to ITIL, knowledge management should be approached as a "process," not as a "tool"—although it uses a tool to collect, store, and share knowledge across the organization, called a Service Knowledge Management System (SKMS). As a process, knowledge management seeks to identify sources of data and information across IT, transforming this data and information into usable knowledge, enabling improved decision making, and minimizing the need to rediscover knowledge.



Although such a process seems straightforward and deserving of wide support across IT, few IT service organizations are able to successfully plan, effect, and maintain a robust knowledge management process along with a supporting SKMS. Why? IT teams are often knowledge "siloes." Without a collaborative approach between functions such as development, operations, and other teams, knowledge tends to become "siloed"—buried within various team databases, repositories, or intranets. Because IT technical, application, and support teams are usually organized around their particular technology, their respective knowledge repositories also become organized around their team—off limits to other IT groups, including service operations teams which are responsible for supporting live services, along with applications and supporting technology.

Without a DevOps approach that encourages collaboration and regular cross-functional communication, development teams often become isolated from operational support teams, unaware of the challenges faced in dealing with problems surfacing in the live environment due to software errors. To be effective in handling incidents and noticing problems, operations must be kept continually up-to-date on any changes to apps that are a part of a service. They must also be equipped with trouble-shooting techniques, workarounds to any "known errors," and be advised on what information is required in order to efficiently handle and escalated incident or problems.

How a DevOps Approach Adds Value to Knowledge Management (KM)

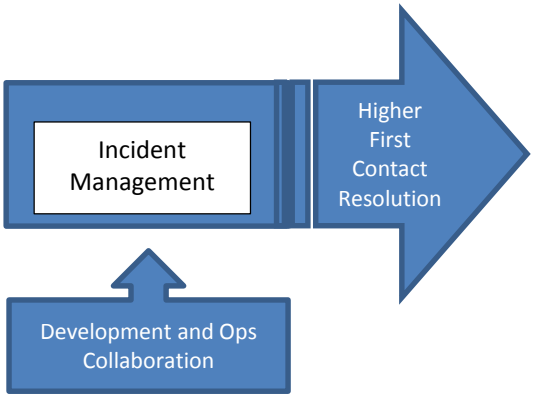
A DevOps approach to knowledge management (KM) would stress that development, operations, and other IT groups must ALL participate by contributing to and drawing value from the process, and the SKMS. It encourages breaking down barriers to storing and sharing knowledge by making it easy for development, operations teams, and other functions to capture knowledge "in the workflow," so that a knowledge article becomes a "by-product" of an activity. A DevOps-powered KM approach would engage subject matter experts in development, operations, and other technical and application support teams, to review and QA knowledge articles submitted before they are published and shared so that accuracy and completeness can be assured.

DevOps emphasizes communications and collaboration between teams, and supports periodic knowledge transfer and training in the form of blogs, videos, and webinars from development to operations—ensuring a continuous knowledge transfer from development to operations on any new or changed applications. This will provide continuous training on the stream of frequent, small changes that are deployed into the live environment, better equipping operational teams to handle questions, incidents, and problems related to new and changed apps. The result is fewer escalations, faster incident resolution, higher uptime, and more satisfied customers and users.

Operations must not only be made aware of the changes, but also of any "known errors" that may be present—along with ways to work around reported "problems." Regular participation in the problem management process by development can help ensure that known errors and workarounds are added to the Known Error Database (KEDB), and made available to service operations in a timely manner.

DevOps in Service Operation

Incident Management



According to ITIL, incident management is responsible for handling the lifecycle of all "incidents," restoring service as quickly as possible, minimizing the impact of service disruption or degradation to customers and users. Incidents are detected by users, at the service desk when a call is received, or at times in IT operations when a service monitoring tool has detected an "event" that signals a disruption or degradation in the performance of a service (or a supporting component). As a process, incident management is normally managed by the service desk, the dedicated functional group that is the "single point of contact" for users of IT services.

The performance of operations, and the incident management process, can be compromised if this function is not able to take advantage of optimized techniques to work around reported incidents, or is not able to access a knowledgebase of "known errors" with effective workarounds. The result will be a less than effective service desk and incident management process—one that does not resolve incidents as quickly as it should, and one that repeatedly "reinvents" solutions due to the lack of knowledge sharing across IT groups.

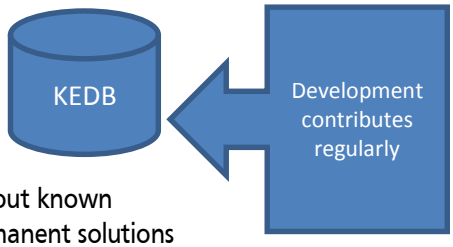
The cause of such issues? Often developers are in another "silo" within the organization, far removed from operational support. Due to management priorities, they are also normally focused on developing the next application, rather than supporting applications in live production. Issues arise when operations escalates incidents to development teams, where the incident has been identified as having to do with some aspect of code in the team's application. In many cases this is compounded by a lack of shared organizational goals, as well as a shared policy and procedure for how incidents are to be handled across IT teams.

How a DevOps Approach Adds Value to Incident Management

Improving regular communication between development and operations can go a long way in facilitating knowledge transfer, troubleshooting skill development at the service desk, and building teamwork and collaboration between front line support (i.e., the service desk), and backline support teams (i.e., development and other technical and application functions). For example, a regular weekly/bi-weekly "operational support" meeting or conference call might be instituted, to review escalated incidents/problems, discuss optimized ways to expedite handling and resolution, and to continually improve the process. As part of their support for problem management, development teams should also be contributing to the "Known Error Data Base" (KEDB). Their regular contribution and support of the KEDB makes information and workarounds available to the service desk on problems and known errors, facilitating service desk performance, and the resolution of incidents.

Another priority should be to clearly communicate and get buy-in from all groups on any IT operational support goals, especially when it comes to maintaining a high level of customer and user satisfaction. Shared incident management policies and procedure which cut across all functional groups within IT should be adopted, including the service desk, technical and application development teams, and other support teams.

Problem Management



According to ITIL, the problem management process has two primary purposes. First: prevent problems from happening in the first place by removing defects in hardware, software, and other components from the IT infrastructure. Second: for those problems that cannot yet be prevented, provide timely information to support staff about known problems, "known errors," workarounds (temporary solutions), and permanent solutions (fixes/corrections).

With the increasing frequency of small, incremental releases in many organizations, and the increased use of automated testing, the potential for software defects even in small release updates is still present.

Problems detected in the development environment, and during test (along with any recommended workarounds), must be channeled into the problem management process for recording and inclusion into a KEDB, and then transferred to front line support teams through a knowledge sharing or knowledge transfer activity. Apart from this, operational support teams will be "blind sighted" by problems introduced to the live environment along with the release update.

How a DevOps Approach Adds Value to Problem Management

A DevOps approach would stress that development, technical, and application support teams be clear that their responsibilities include participation in the problem management process. The policies for this process should make it clear that problem management extends not just into live service operations, but also into the development and test environments.

DevOps would "make it easy" for developers to capture information about problems, workarounds, and "known errors" in the workflow—as a byproduct of development activity. DevOps would remove barriers and make it easy for developers to submit problem reports, with associated workarounds, to problem management for inclusion into the KEDB.

DevOps in Continual Service Improvement: CSI: Improving People, Process, and Technology across the Lifecycle

CSI is all about identifying major improvement opportunities across the service lifecycle—in all stages, across all processes, in people practices, and with respect to the use of technology. A CSI program identifies improvement opportunities by enabling services and process owners, managers, and practitioners with a good measurements and reporting framework, models for identifying and management improvement opportunities such as PDCA, the CSI Approach, and a vehicle for submitting improvement opportunities to CSI. An organization's CSI program then reviews these opportunities on a periodic basis, and acts to transform major improvement opportunities into proposals for improvement (SIPs). When implemented, these SIPs result in major improvements to services, processes, supporting technology, and people practices.

How a DevOps Approach Adds Value to CSI

By encouraging a higher level of ongoing communications and collaboration between developers, operational support teams, and other functions across IT, a DevOps approach will result in a higher level of teamwork and synergy between teams. This results in a higher level of "out-of-the-box" thinking and creativity. Provided that CSI has enabled functional teams with common models for improvement, such as PDCA, and the CSI Approach, and a good measurements and reporting framework, teams operating in service design, service transition, and in operations should be equipped to identify improvement opportunities more quickly and easily, implementing minor improvements themselves, and submitting major proposals to CSI for consideration.

Why Does DevOps Matter? It Adds Real Value to ITIL

Change is the order of the day, and if anything, the pace of business and technology change is accelerating. The business and customers are looking to IT service providers to be more responsive, delivering more frequent service changes with higher quality—resulting in services that deliver more value to the business. In order to continue to be relevant and of high value, ITIL must continue to benefit from other complementary best-practices for IT. DevOps, an approach that encourages improved communication, collaboration, and teamwork across development and operations, can have a positive influence in improving ITIL processes across the service life-cycle:

- Due to the collaboration and communication that DevOps promotes, the strategy processes of strategy management for IT, and business relationship management, benefit with more sound, long-range strategic planning, reduced risk, and improved communication and transparency with customers.
- Service design benefits from a more effective design coordination process that includes all key stakeholders in design and development activity, resulting in a more "holistic" service design—development, operational support teams, and other stakeholders contribute in a collaborative fashion to arrive at a complete design, including optimized architecture and supportability.
- Service transition benefits from a more collaborative approach to change management, resulting in less bureaucracy, a faster process, and more standard changes. Release and deployment benefits though improved planning, and more effective builds, testing, and deployments. And knowledge management is improved through improved collaboration and support for sharing knowledge across IT development and support teams.
- The Service operation processes of incident and problem management benefit through improved collaboration and communication between development and operations groups, resulting in improved capability for the service desk, fewer escalations, higher availability of services to customers and users, and lower costs to IT.
- Continual service improvement benefits because a DevOps approach, when applied to processes across strategy design, transition, and operations, results in a more collaborative approach that leads to more frequent discovery of key improvement opportunities. These can be submitted to CSI in the form of improvement proposals, which CSI can then review, assess, and potentially action as SIPs.

There is no question that ITIL is the core of industry best-practices for IT. But IT service providers must benefit from and incorporate the best complementary practices as well—including DevOps—that have shown to improve alignment with the business and customer, raise the quality and performance of IT services, improve the throughput of delivery, and lower overall IT costs.

Sources

- "How are Lean, Agile and DevOps related to each other?" Jul 12, 2012 - Agile Web Development & Operations - *http://www.agileweboperations.com/lean-agile-devops-related*
- "ITIL Guide to DevOps" – ScriptRock. http://www.scriptrock.com/blog/ITIL-guide-to-devops
- "A Pragmatic Guide to Getting Started with DevOps" – CA Technologies: https://www.ca.com/us/register/forms/collateral/dysfunction-junction-a-pragmatic-guide-to-getting-started-with-devops.aspx
- "DevOps and Agile" April 18, 2014 – the Scrum Alliance: *https://www.scrumalliance.org/community/articles/2014/april/devops-and-agile*
- "What is DevOps?" - The Agile Admin: *http://theagileadmin.com/what-is-devops/*
- IBM DevOps: *http://www.ibm.com/ibm/devops/us/en/*
- "ITIL Invented DevOps" – by Brian Johnson, Founder, ITIL Users Group, and original ITIL Author - *http://www.ca.com/us/lpg/ca-technology-exchange/ITIL-invented-devops.aspx*

Learn More

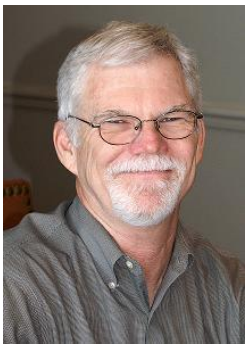
Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge through training.

DevOps Implementation Boot Camp

ITIL® Foundation

Visit **www.globalknowledge.com** or call **1-800-COURSES (1-800-268-7737)** to speak with a Global Knowledge training advisor.

About the Author



Paul is the president and principal consultant of Optimal Connections LLC. With over 30 years of experience in planning and managing technology services, Paul has held numerous positions in both support and management. Corporate experience includes working for such companies as Motorola, FileNet, and QAD. He is also experienced in service desk infrastructure development, support center consolidation, deployment of web portals and Knowledge Management Systems, as well as service marketing strategy and activities. Currently Paul delivers a variety of services to IT organizations, including Support Center Analyst and Manager training, ITIL Foundation and Intermediate Level training, Best-practice Assessments, Support Center Audits, and general IT Consulting. His degrees include a BA and MBA. Paul is certified in most ITIL Intermediate levels and is a certified ITIL Expert. He is also on the HDI Faculty, and trains for ITpreneurs, Global Knowledge, Quickstart Intelligence, Phoenix TS, and other IT training organizations. For more on Paul, visit www.optimalconnections.com.