Overcoming Inefficiencies in Multiple Work Management Systems

Helping Your Enterprise with Their DevOps Transformation

IT REVOLUTION
DEVOPS ENTERPRISE FORUM
2018

25 NW 23rd Pl
Suite 6314
Portland, OR 97210

Overcoming Inefficiencies in Multiple Work Management Systems:
Helping Your Enterprise with Their DevOps Transformation

Produced in the United States of America

Cover design and interior by Devon Smith

For further information about IT Revolution, these and other publications, special discounts for bulk book purchases, or for information on booking authors for an event, please visit our website at ITRevolution.com.

PREFACE

In March of this year, we at IT Revolution once again had the pleasure of hosting leaders and experts from across the technology community at the DevOps Enterprise Forum in Portland, Oregon. The Forum's ongoing goal is to create written guidance to overcome the top obstacles facing the DevOps enterprise community.

Over the years, there has been a broad set of topics covered at the Forum, including organizational culture and change management, architecture and technical practices, metrics, integrating and achieving information security and compliance objectives, creating business cases for automated testing, organizational design, and many more. As in years past, this year's topics are relevant to the changing business dynamics we see happening across all industries and the role technology has to play within those changes.

At the Forum, as in previous years, participants self-organized into teams, working on topics that interested them. Each team narrowed their topics so that they could have a "nearly shippable" artifact by the end of the second day. Watching these teams collaborate and create their artifacts was truly amazing, and those artifacts became the core of the Forum papers you see here.

After the Forum concluded, the groups spent the next eight weeks working together to complete and refine the work they started together. The results can be found in this year's collection of Forum papers.

A special thanks goes to Jeff Gallimore, our co-host and partner and co-founder at Excella, for helping create a structure for the two days to help everyone stay focused and productive.

IT Revolution is proud to share the outcomes of the hard work, dedication, and collaboration of the amazing group of people from the 2018 DevOps Enterprise Forum. Our hope is that through these papers you will gain valuable insight into DevOps as a practice.

—Gene Kim
June 2018
Portland, Oregon

INTRODUCTION

Large enterprises are traditionally organized by function and managed to optimize vertically for specific outcomes. In IT, this often means organizations specialize in functions such as design, development, QA, and operations. Many decisions are made in the context of those functional silos as opposed to the end-to-end flow of delivery across those teams.

This mode of decision-making affects work management practices as well as tool selection for each group. In traditional operating models of siloed teams, this is certainly an issue when creating an environment of task-driven queues. We view this as a variant of Taylorism,[1] in which the concept of work specialization and tool choices for work management have negative impacts on overall service delivery goals in knowledge-based work. Common countermeasures usually involve the introduction of other groups (release management, change management, project management) to help manage the "flow" of work, but these groups yield marginal returns, if any. (See Figure 1.)

As enterprises adopt Agile and DevOps, this mismatch between SDLC and ITIL practices, as well as tool silos, impedes both flow efficiency and work understanding within individual team and across teams. Additionally, business stakeholders are continuously frustrated because work disappears into the IT "black box," and IT leaders are unable to provide timely delivery, let alone estimates and visibility into the work that it takes to both build and run the service. This dysfunction erodes trust between the stakeholders and the teams that are delivering and running critical services for the business.

This paper is focused on enterprise practitioners and management leaders who have multiple work management systems that their teams deal with every day and who struggle to provide visibility to the work as well as an improvement model to make the work and systems better.

[1] Production efficiency methodology that breaks every action, job, or task into small and simple segments which can be easily analyzed and taught. Taylorism aims to achieve maximum job fragmentation to minimize skill requirements and job learning time. When taken to the extreme in software and operations org design you often see silos organized by skill.

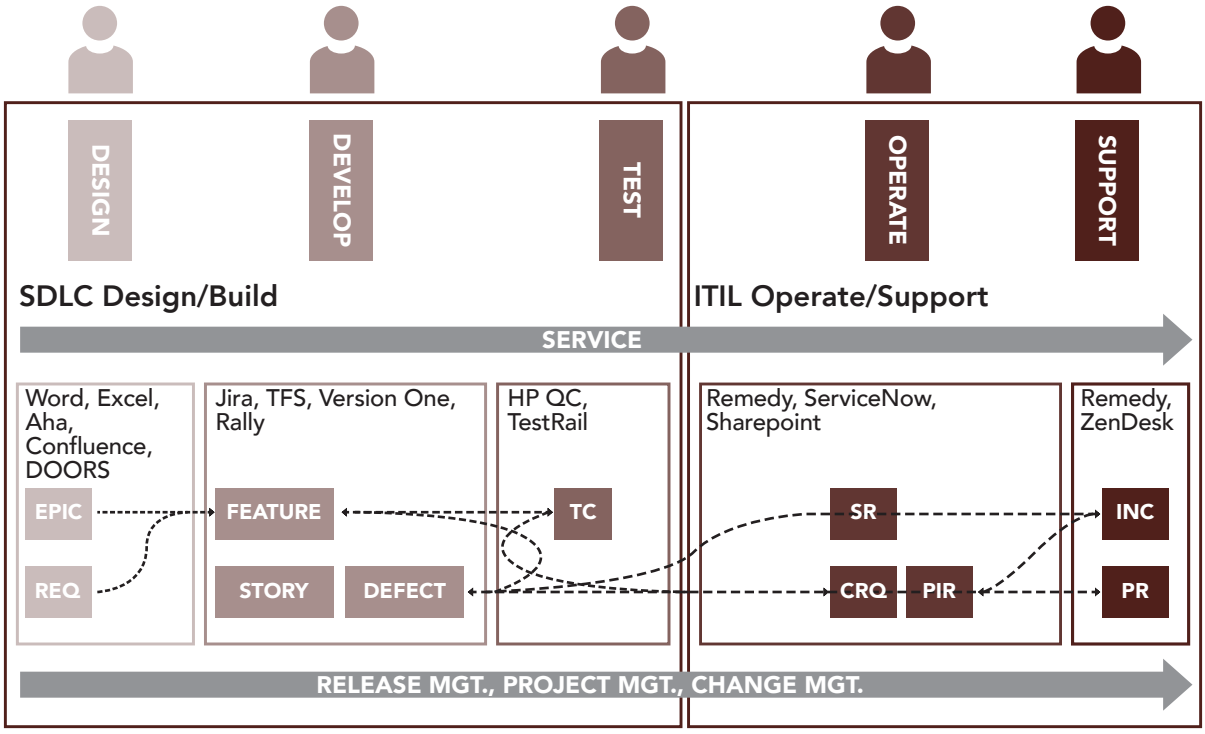

Figure 1: Workflow as Influenced by Release Management, Change Management, and Project Management

THE PROBLEM

Multiple work management systems that originate in functionally oriented silos or independent teams create long-term inefficiencies at the enterprise scale. Communication between business people and IT people has long been an issue. It can seem as if the two groups speak different languages. Lack of visibility into IT work leaves business people confused and guessing as to what IT is doing. They think to themselves, "IT is a black box. We have no idea what goes on there." Furthermore, IT people struggle to articulate operational risks and problems to business people—particularly, the question of why IT appears to rarely meet the needs of the business. If meeting expectations is a measure of success, IT frequently comes up empty handed.

No doubt, multiple causes contribute to this lack of understanding. One cause that stands out is the use of different work management systems. Some examples of different work management systems are as follows:

- A product management tool to manage roadmaps and requirements.
- An SDLC tool to manage features/user stories/defects.
- A test tool to manage test cases and regression testing.
- An ITSM tool to manage and approve service requests and change.
- A help desk tool to manage incidents, problems, and knowledge.

When each functional team uses their own set of tools, long-term inefficiencies can occur. Many software delivery organizations waste time on manual data entry, handovers, status meetings, spreadsheets, and chasing down missing information. This leads to high coordination costs, often because of dependencies.

As stated in Dominica DeGrandis' book *Making Work Visible*, unknown dependencies make up one of five costly time thieves related to multiple work management systems.[2] The other four thieves are too much work in progress, unplanned work, conflicting priorities, and neglected work. Let's look at each of these thieves briefly.

Too much work in progress (WIP) matters for a number of reasons. When we start a new task before finishing an older task, the work takes longer to complete. The more items that are worked on at the same time, the more doors open up that allow dependencies and interruptions to creep in, which leads to context-switching and delays. These in turn increase costs, decrease quality, and contribute to conflicting priorities.

Unplanned work impacts time measures as well, because unplanned work steals time away from planned work. When urgent issues pull people away from planned work (such as an unexpected malfunction with a heavily used program) uncertainty and variability get added into our everyday work. Because of this interruption, something else is going to take longer than expected. Furthermore, unplanned work is often invisible (because it's hidden in another team's toolset), and invisible work is hard to manage.

Conflicting priorities raises its ugly head when people are bombarded with messages like, "Do all the things now." A team of ten people working on fourteen initiatives at the same time don't have sufficient focus to give any one initiative the attention it deserves in order to deliver a quality product or service. Multitasking is

[2] DeGrandis, Dominica. *Making Work Visible: Exposing Time Theft to Optimize Work & Flow*. Portland, OR: IT Revolution, 2017.

an effective way to get less done. When you try to do too many things at one time, you won't do any of them great, and sometimes you won't finish them at all. Trying to do too much at once delays everything. If you want to get important work done, decide what things you're *not* going to do.

Last, there is **neglected work**. This is work that doesn't get the necessary attention it needs. Neglected work is constantly deprioritized, losing resources or funding. It sits on the bench like a second-string sports player watching the other players get all the attention. Fixing technical debt is often a victim of neglected work because the business frequently prioritizes new feature releases over fixing technical debt. New features equal revenue generation, while fixing technical debt equals revenue protection. This is an example of how IT people struggling to articulate operational risks to business people.

Together, these five time thieves delay the delivery of value to customers.

It is frustrating for all involved to see time wasted on keeping colleagues and tools in sync instead of doing the actual value-adding work that the business needs. When work is visible across an organization's work systems, it's easier to see the single source of truth of how business value is flowing from inception through production support.

SOLUTION-GUIDING PRINCIPLES

A necessary condition for dialogue among IT and business stakeholders is a visualization of the work. Visualizing work helps others see risks and problems. Visual landscapes provide a level of tangibility that might otherwise be considered too abstract to stakeholders. Mapping value streams and defining feedback loops within the value streams helps to optimize flow. Because a focus on flow improves the delivery of business value, organizational objectives can begin to be met.

Given the importance of work visibility, we suggest using the following guiding principles to form an improvement cycle for both the work management system and the nature of work[3] being done:

[3] "Nature of work" refers to the actual work being done in the teams that can now be improved or automated. A good example is a standard change that might be manually executed on a recurring basis, like a service request to change a URL in 100s of files or an incident to restart a job when it fails. Once these types of work are exposed, the team can look to change or automate this work. Without visibility, it is unlikely that these types of work improvements will be made.

1. Make work visible.
2. Use visibility to improve system behavior.
3. Adjust work management to make work visible.

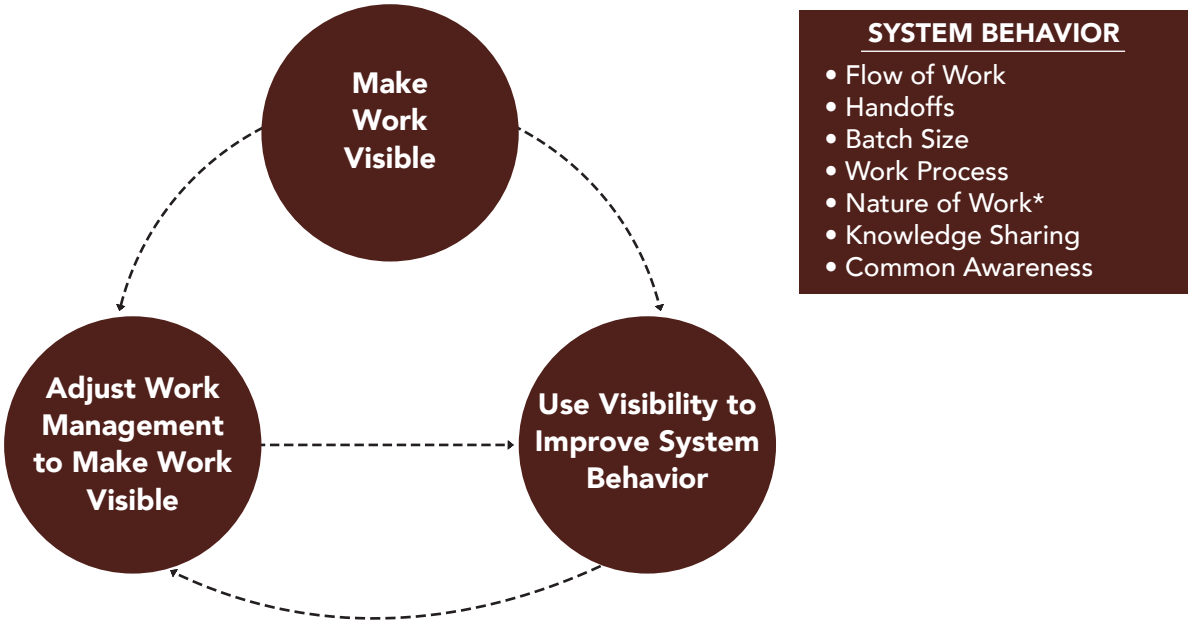

Figure 2: The Work Management Improvement Cycle

COUNTER MEASURES

There are multiple ways to address the problems caused by lack of visibility of work across the entire value stream for a product. This section will describe a number of the approaches that have been used by different companies. Each has its own advantages and disadvantages. The goal of these strategies is to provide examples of ways to make all the work visible across all teams, including the "black box" of operations work.

This section is primarily focused on the tooling used in the work, as it is the tools that interfere with seeing end-to-end work and overall problems. Once all the work is visible, then improvements to the flow and system mechanics can begin.

The first step is to identify the different tools used to capture the work. This will include development tools such as Jira, TFS, RTC, Word documents, and emails; help-desk tools such as Remedy and Servicenow; incident tools such as HP ALM, Jira, and Remedy; and testing tools such as HP ALM, RQM, Jira, and email. With all of the solutions, an additional requirement is to move out of email and non-trackable sources.

One other important factor with any of the choices to consider is that other tools and other processes will be discovered, and those will need to be included and addressed as part of the update.

Table 1. Countermeasures

Countermeasure	Description
Off-the-Shelf Replication	Replicate entities across tools using off-the-shelf products such as Tasktop.
Consolidation, Replication, and Strangulation	Choose a foundational system to replicate and consolidate work items to and strangle off non-strategic options.
Migrate	Wholesale migration of one or many work items to desired target, may include eventual depreciation of original source system.

The first of the choices described here is a syncing methodology. There are tools, such as Tasktop, that allow you to sync multiple work item processing tools. With this choice work fields are mapped and then synced across the different tools. This allows teams to see the entire work; however, this increases the complexity and cost in the system by adding yet another tool to be synced with instead of removing tools. This option has been used most effectively as a starting point to make work visible. This should not be seen as an end point in the migration but as an early step in an overall plan.

The second of the choices is to use a consolidation, replication, and strangle approach. For this approach, you select the primary tool that will be used to show work across all teams. Most organizations select the development workflow tool for this consolidation point. This is based on the fact that Operations needs to move to a more

development-based methodology for their efforts that includes capabilities such as infrastructure and code. Those doing operations work will need to track their software updates as changes, manage their scripts in the SCM, and, as this is already set up and integrated, move all work into the development pipeline to make this process simpler.

In the consolidate and strangle countermeasure, help desk tools such as Remedy or Servicenow generally remain as help desk tools, but incidents created are synced into the development backlog. This approach still leaves multiple tools in place (but generally fewer) and it provides a realistic approach that can be implemented over time to gain value. With this approach, once the work is visible the improvements start to flow and system mechanics can be implemented primarily in the single manager of work item managers.

The last of the choices described here is referred to as the rip-and-replace strategy. This is the most risky of the options. With this option, you have to have the executive support and funding to complete the activities, and you have to be prepared to change the processes and the tools at the same time. The approach is to select the desired end-state tool for work items and then move to that tool for all teams. Organizations that have selected this approach generally don't have modern tools in use in the development area from which they're selecting. Finding and migrating all the existing data into the new environment is important to seed the system with the current work. With a new tool for everyone, there is a single place for visibility, but it requires training and a conscious effort to support the change in processes.

With all of these approaches, you also need to work with the teams to capture all the hidden work that is currently not tracked in any tool or that appears only in email or other less integrated sources. This is a change, but by having teams see all their work, it becomes easier to do the proper prioritization and allows the organization to see what people are spending their time on.

In this proposal, you see that we are recommending you select a single tool, which conflicts with a common general principle that teams should be able to select their own tools. Flexibility for teams is important, but there are a few key areas where standardization brings more value than any gain by teams using different tools. Those key areas include the work item management system, so all work can be visible, and the source code manager, so that the work created is visible to all with the task of tracking it.

BENEFITS

Our brains are wired to process information visually. Visual communication is a must-have skill for organizations, because it's often the only way to make sense of the work being done. By taking the approach of tool consolidation, data replication between systems, and toolchain strangling, we can achieve better visibility into the work being done, improve flow and system mechanics, and gain better customer visualization of capacity, velocity, and IT priorities.

Visibility into the work being done within the "black box" provides product owners with an appreciation for all of the patching, compliance, and other activities that IT does to keep the lights on. It also allows teams across the value stream to see how they are involved in the work and how their work affects other teams. The result of this visibility can help get teams out of the mindset of "this is how I've done it for years."

Improving flow within a pipeline allows us to eliminate the manual processes between tools, avoids overlapping functions between tools, and reduces complexity for team members, resulting in faster delivery. Improving system behavior by connecting, reducing, or passing data between systems allows us to reduce or eliminate handoffs, wait times on requests, and multiple sources of records.

- Visualizing work allows customers to see IT capacity and velocity.
- Visualizing work creates the ability to build a shared awareness between stakeholders and teams executing work.
- Awareness of the work unlocks the ability for teams and stakeholders to begin to make both system and process improvements.

CASE STUDIES

Financial Organization

This case study is an example of the migrate countermeasure (described previously) in a scenario where it was the most appropriate option. This case study represents a financial institution with an IT workforce of approximately 2,000. Due to mergers, the organization had a very diverse set of tools, and those tools had not been the focus of improvement or updating in years. This had led to a mess of out-of-date

capabilities that were not satisfying any part of the organization. At the same time, there was a drive to Agile at an organizational level, including concepts of DevOps (though DevOps was not really a fully formed principle yet).

The organization started with an assessment of all the different ways work was being captured and tracked, from the initial requirements through the running of the system. They discovered that their departments and teams used the following programs and systems:

- Business analysts: Word, email, shared drive
- Technical leads and project managers: Word, Excel, Project, email, shared drive
- Developers: Word, Excel, email, shared drive
- Testers: HP QC, Word, shared drive
- Deployment team: Word, Excel, shared drive
- Operations: Remedy, Word, shared drive

Requirements were specified in large Word documents, the titles of which were manually copied with a number into an Excel spreadsheet for tracking. The documents were put on a shared drive for the business to see. There were standard templates for these, but the template was different for each of the original companies.

When the business analysts worked with the technical leads to translate the requirements into a form that Development could work on, new technical specification documents were created. The title of the requirement was copied into the technical specification document for some level of traceability, and again, a spreadsheet was created to list the new development level of the requirements and to facilitate breaking down the work into assignments. These documents were also stored on a shared drive, but at a different location accessible to the Development teams. The original documents were not generally accessible by the Development teams. The project managers would create a project file with the titles of the requirements and lay them out in the plan to be completed.

The description of each requirement to be developed was then sent to the developer responsible via email for some teams and via the shared drive for other teams.

Status meetings were the only way information about progress was shared.

The business analysts would also work with the test team to share the requirements, and the test teams would create yet another document on test cases to be created. The manual tests were also created in HP QC for tracking.

Once development was complete, the code was delivered to the test team. When the test team found problems, they would open HP QC defects for the developers to work on.

While testing was being completed, Development would create a run book for the deployment of the code, as well as a run book for the Operations team on how to deal with any problems or issues. These were created and stored in yet another shared drive common between Development, Deployment, and Operations.

A change request had to be created in Remedy for the production deployment and to have the official run book updated for operations. This change request was a long form requiring the input of much of the information that was already put in the Word document but which had to be specified again in the change request.

You can imagine the number of problems that can come from such a process. By the time development was done, it was very hard to trace back to the original requirements documents, and there was no automated way of doing the trace—it was all manual work.

This organization knew they needed to improve, but they really had no existing work tracking capability. They selected a single work item planning tool, Rational Team Concert, as the foundation. With this, they could have everyone collaborating in a single tool, all work would be visible in the dashboards, documents would no longer be lost in shared drives but could be tracked, and traceability was clear from the original requirement to the deployment request.

The transition to RTC was done at the same time as a transition in the development process to Agile. The tool change helped support the cultural transformation aspect of the overall transformation within the organization. This transformation was done on a product-by-product basis within the organization to allow for training and support for each of the teams.

Tasktop

The following list of case studies are examples of the off-the-shelf countermeasure option described previously. These demonstrate how companies can gain visibility through replication of the various tools.

This first example, "Helping a Center of Excellence Live Up to Its Name" demonstrates how a personal insurance company used synchronization from Tasktop to bring all the work visibility to the testing center of excellence while allowing the individual teams to work with their own tool of preference.

The second example, "Gasoline Systems-Electronic Hybrid Customer Interface with Tasktop Sync," describes how managing change requests in one tool and defects in a different tool can result in workflow inconsistency, errors, conflicts, out-of-sync data, and manual rework. The setup created inefficient communication between a major automotive supplier and the OEM, adding unnecessary cycle time to each project iteration. The big takeaway here is that the problems of managing change requests and defects in different tools can be solved.

- "Helping a Center of Excellence Live Up to Its Name." Tasktop Sync CaseStudy | Tasktop Value Stream Management for Software Development. https://www.tasktop.com/case-study/microsoft-tfs-ibm-rtc-hp-alm-servicenow-excellence.
- "Gasoline Systems-Electronic Hybrid Customer Interface with Tasktop Sync." Tasktop Sync Case Study | Tasktop Software Integration for Delivery Teams. https://www.tasktop.com/case-study-hpqc-jira-rtc-gasoline-hybrid.

CONCLUSION

By following the guidance provided in this paper, enterprises can break down inefficiencies that exist in typical vertical teams and eliminate tooling exhaustion that often creates disconnects in visibility to tasks that are being carried out. The "black box" of IT is now visible by product owners, and those product owners can prioritize the efficiency work related to the process improvement, resulting in better awareness and visibility, happier and more fulfilled staff, and more consistent delivery.

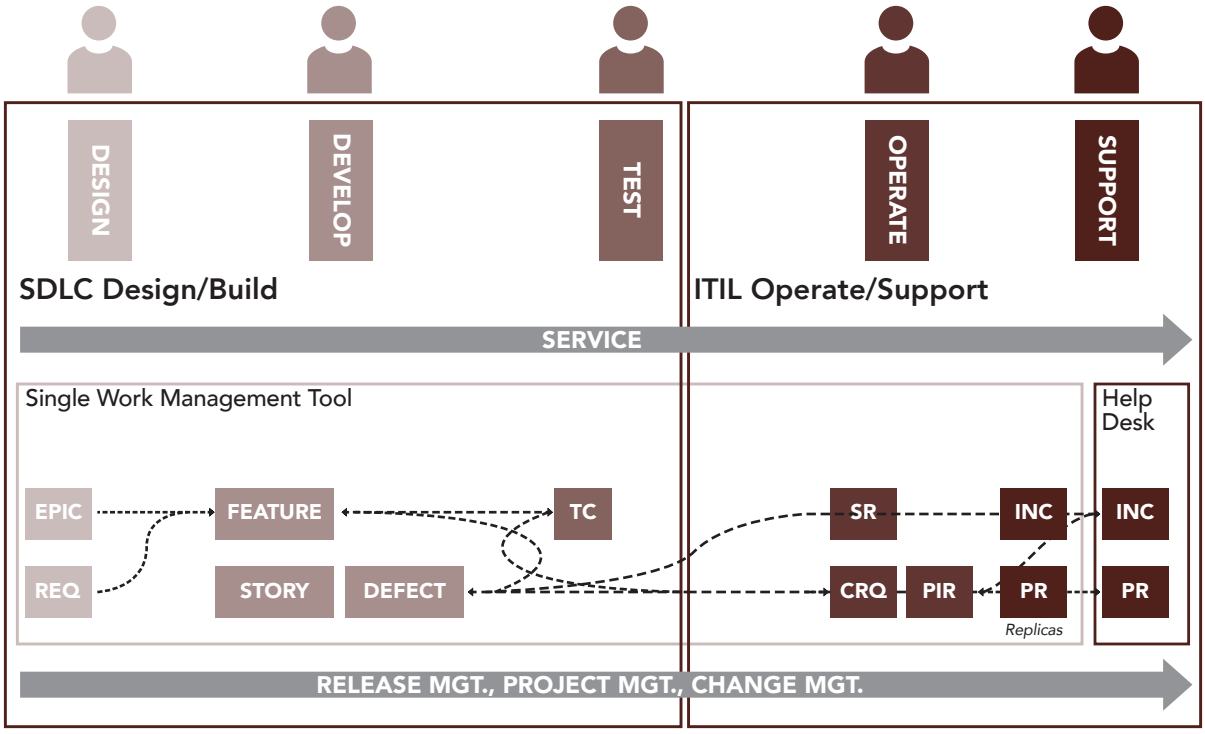

Figure 3. Traditional Model with Collapsed Toolchain

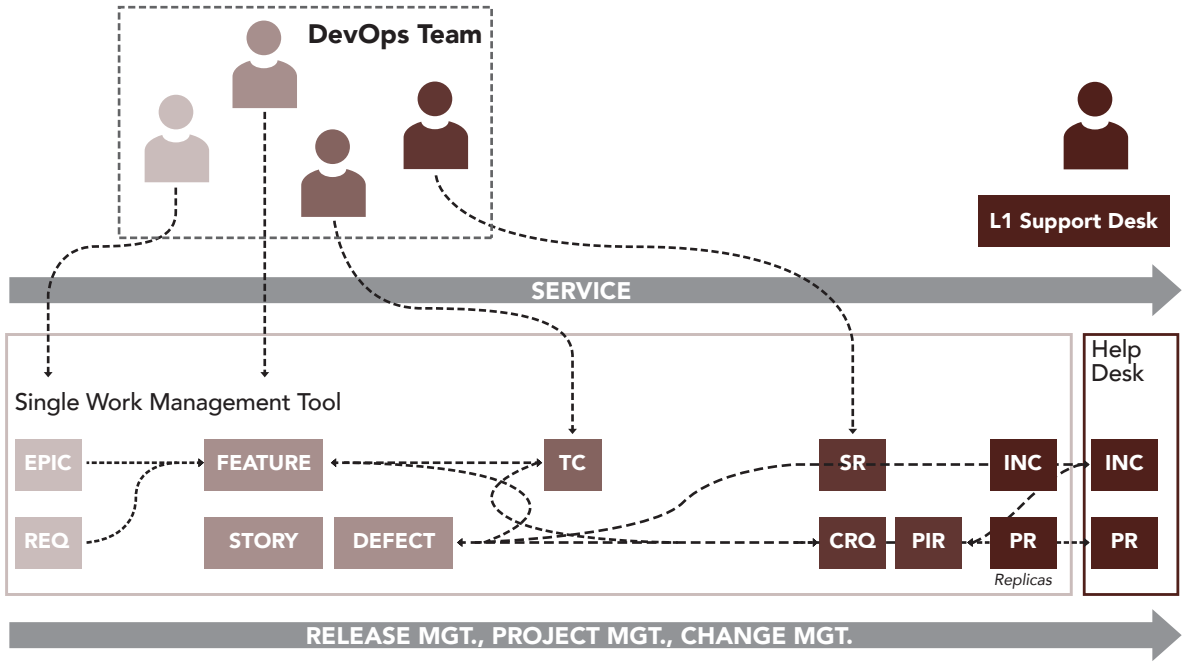

Figure 4. Build/Run Teams with Collapsed Toolchain

Collaborators

- Scott Prugh, SVP Engineering, CSG International; @ScottPrugh
- Dominica DeGrandis, Director Digital Transformation, Tasktop; @dominicad
- Rosalind Radcliffe, Distinguished Engineer for DevOps for IBMz, IBM; @RosalindRad
- Pat Birkeland, Business Analytics/Intelligence Compentency Center Lead, Boeing
- Keanen Wold, DevOps Transformation/DevOps CoE & Developer Practice Leader, Delta Airlines, @KeanenWold