Microsoft Core Services and Engineering (CSEO) is implementing a modern engineering vision that creates a culture, tools, and practices focused on developing high-quality, secure, and feature-rich services to enable digital transformation at Microsoft. The Modern Engineering initiative is helping CSEO be customer-obsessed, accelerate the delivery of new capabilities, and improve engineering productivity. To learn more about how we’re transforming, refer to Inside the transformation of IT and operations at Microsoft.

Our journey so far

Our move to the cloud enables us to increase the overall agility of the development process and accelerate value delivery for approximately 600 services comprised of about 1400 components to new cloud technologies that require less specialized skill sets to use and provide quicker access to additional infrastructure. This enables spinning up environments and resources on demand, which further enables an engineer to respond more quickly to evolving business needs. However, we still need to address several structural issues, including inconsistency between teams in basic engineering fundamentals like coding standards, automated testing, security scans, compliance, release methodology, gated builds and releases, and similar. We did expect this, given that some teams have portfolios with as much as 40 percent legacy code that consists of large components packaged together. This makes the management of storage, scale, availability, and operations difficult, so we’ve begun driving consistencies by implementing organization-wide service reviews.

We lacked a centralized common engineering system and associated practices. Recognizing that we could not continue to evolve our engineering system in a federated way, we invested in a central “Fundamentals” team. This team’s charter is to innovate on a common engineering system based on Microsoft Azure DevOps, while driving consistency across the organization with respect to how they design, code, instrument, test, build, and deploy services. We are bringing a product-engineering mindset to our services by defining a vision for each service area, establishing quarterly priorities, and executing on these via a defined cadence of sprints. The resulting engineering system and consistency promotes developer efficiency and cross-team mobility.

We needed to mitigate risk in our code by incorporating industry-leading development practices for accessibility, security, and compliance. Achieving compliance has been very challenging, forcing us to change from legacy processes and tooling, and requiring that we actively respond to our technical debt in these areas. Onboarding to new practices was slow, and there was friction because of the volume of technical debt that we had to clear. This hampered our efforts to set automatic policies on our code commits, which ensure we stay clean moving forward. We understand that we need to work towards enabling delivery of accessible, secure, and compliant software and services in a frictionless manner.

We also lacked a consistent level of telemetry and monitoring that allowed us to obtain key insights about service health, features, customer experience, and usage patterns. Today, CSEO runs at an average of 110,000 support tickets a month. We must perform deeper analysis, however, to identify opportunities for prevention and sustained remediation. There also is an opportunity to fully automate our current end-to-end process, which has some manual aspects due to gaps in our tools. We are driving towards a Live Site culture so that we can comprehensively monitor service health, proactively identify issues, and quickly remediate incidents systematically, while focusing on root cause to ensure sustained improvements in service quality. We established common CSEO-wide measures for service health, such as Time to Detect, Time to Mitigate. We discuss these at our service-review ceremonies across CSEO teams, and this drives culture change and improved visibility of issues that could result in incidents, such as expired security certificates. Telemetry capabilities have been improved through the ability to do synthetic monitoring and the ingestion of data from a wide variety of data sources.
We are implementing a unified approach for emitting telemetry data, managing this telemetry and insights regarding our business processes, applications, and infrastructure to facilitate cross-correlation. With the right instrumentation, the system should support basic monitoring scenarios for each app and service and enable metrics and cross-correlation across different applications. This will:

- Generate more actionable insights.
- Eliminate the need for learning and maintaining multiple, disparate telemetry solutions with little or no out-of-the-box integration.
- Provide a common experience to Engineering and Business teams to improve service, infrastructure, and business process health.

To address our previous high level of dependency on suppliers, we’ve implemented a new workforce strategy, whereby we’ve increased hiring of full-time employees and brought more work in-house. This allows us to transform and modernize how we deliver services, and furthermore, this workforce strategy makes it imperative that there is full-time employee oversight of any supplier deliveries. Our strategy now requires that the developer who owns the service or contract being delivered assumes end-to-end accountability for the quality of what the supplier delivers and also ensures they adhere to processes, standards, and regulatory requirements, including security, accessibility, and privacy. Additionally, as we’ve increased our hiring, we’ve also implemented a common bar for hiring across all teams and a common onboarding program to ensure all new hires receive a consistent level of training on all key tools and technologies.

Vision for modern engineering

Microsoft’s digital transformation requires us to deliver capabilities and solutions at a faster pace and with high quality, reliability, and security. To achieve this, we’re modernizing how we build, deploy, and manage our services to get new functionality in our users’ hands as rapidly as possible. We’re re-examining every part of our engineering process and instituting modern engineering practices in a way best summarized by Satya Nadella, our Chief Executive Officer: “In order to deliver the experiences our customers need for the mobile-first, cloud-first world, we will modernize our engineering processes to be customer-obsessed, data-driven, speed-oriented and quality focused.”

We want to ensure our engineers keep customers front and center in their thoughts, so we’re capturing feedback in a common tool, thereby providing our engineers with a deep understanding of the customer experience. We’re surfacing and viewing this feedback in conjunction with telemetry measures that will help us build an understanding of how our users are utilizing our products and the features that are causing them issues. We want to implement a higher fidelity service and business-process monitoring so that we’re alert to problems and fix them before our customers are even aware of them. We want to enable our engineers to develop reliable and secure services, and reduce deployment lead times by streamlining the process from committing of code to deployment while integrating compliance checks seamlessly in the build and release pipeline. We are the first customers of our commercial offerings, which enables us to identify and address the engineering needs of the enterprise operating in a cloud-centric architecture. We constantly work with our product engineering groups across the company, creating a virtuous cycle that makes our products such as Azure DevOps and Azure services even more enterprise ready.

We’re building a culture across CSEO that supports modern engineering practices and rapid delivery of new capabilities, and we’re increasing our ability to deliver faster, validate new capabilities with customers via stage gated rollouts, refine them via tightly integrated feedback loops, and provide regular releases, thereby enabling the Microsoft business to respond faster.

While increased velocity is critical to digital transformation, it must not negatively affect reliability. Continuous data collection, monitoring of health signals, moving toward proactive detection, and quick remediation are also critical in ensuring reduced downtime and creating a high-quality customer experience. Implementing Modern Engineering practices is a long-term journey that requires fundamental changes to mindset, culture, and tooling, including:

- Shifting to a modern cloud architecture; building microservices by componentizing monolithic applications; publishing APIs, NuGet, and similar in a central repository to enable integration; ensuring greater than 90 percent
automated test coverage; and partnering with other teams by taking dependencies and integrating with their service rather than attempting to duplicate it.

- Continuously integrating tested code and only deploying from a master branch of the code.
- Shifting from fixed cadence releases to continuous integration/continuous delivery by keeping the product in a releasable state, allowing fast release and rapid response to failures supported by an automated deployment pipeline.
- Conducting experimentation in production using a ring-deployment model instead of having user acceptance testing (UAT) and relying on dedicated UAT environments.
- A Live Site culture that comprehensively monitors service health, proactive identification of issues, introduction of self-healing mechanisms, fast remediation of major incidents, direct contact with end users, laser focused on root cause to ensure sustained improvements in service quality and by extending focus from service health to business process health.
- Proactively managing acknowledged technical debt and reserving engineering resources so that we can address it before it generates unplanned work or impacts the quality of service.

Investing in modern engineering
We have established a strong foundation for CSEO with common tools and platforms for service-portfolio management and integrated this data with our incident management, telemetry, and common metric dashboards such as those for incident health and compliance, which we review monthly in a CSEO-wide Service Review. This foundation enabled us to drive common practices that improve the overall experience with CSEO services by focusing on fundamentals such as security and compliance, and then also enforcing better incident management and service health from a customer’s perspective. Our ongoing investments in modern engineering practices and technology build on this foundation, and they reflect our vision and support our cultural changes. We have three key pillars on which we’re basing these investments:

- Customer obsession
- Rapid delivery
- Engineering productivity

Customer obsession
We are focusing on increasing our effectiveness in service and live-site incident management. We’re merging service-management platforms, rolling out a standard incident-management process, and measuring continual improvements against key metrics. Even as we continue pushing in this direction, we’re driving toward a better understanding of business process health, and we’re extending our tools and processes to improve business-process health monitoring. A business process can span multiple services and technologies, ranging from modern to legacy to third party. For example, procuring a service from a supplier, and then providing transparency to a supplier regarding the payment status for that service, requires integration across multiple systems and manual work by agents to support the process. We’re bridging the connection between service and business health by working to gain deeper insights on key business processes that span multiple services and technologies. This enables us to collect and aggregate data to provide a holistic picture of service and process health. We’re proactively detecting process bottlenecks to help decrease response time and we’re using end-to-end process monitoring to extend our visibility beyond individual services. This ensures our entire business processes are functioning effectively.

Using a telemetry platform
We’re using a unified telemetry platform built on Azure Monitor that helps us implement continuous improvements in the quality of our services. This platform integrates with heterogeneous data sources such as Kusto, Azure Cosmos DB, Azure Application Insights, and Log Analytics to collect, process, and publish data from applications, infrastructure, and business processes. A unified telemetry platform helps us obtain end-to-end views and generate more actionable insights about our service management, and it also enables us to better examine raw data and Application Insights data via common visualizations we use to identify correlations at team, Live Site, and service
reviews. We’re working toward delivering highly connected insights that aggregate the health of component services, customer experience, and business processes. This produces contextual data that not only identifies events but also identifies root causes and recommended next actions. We’re using business process monitoring (BPM) to monitor true availability and performance by tracking successful transactions and customer impact across multiple services and business groups.

Data enhanced tickets will provide a business-impact prioritized view of issues, supplemented with potential causes including those identified via Machine Learning to assess severity, enable smarter attribution of a change to a given incident, and similar. These data-enhanced tickets allow teams to focus on the most important tickets and reduce mitigation time.

We are integrating synthetic monitoring into the unified telemetry platform pipeline to help service engineers visualize and track the performance of their service, reduce the time it takes to detect issues, and pinpoint bottlenecks in the system. To achieve a sustained level of quality, we’re leveraging synthetic monitoring for all critical services, especially those with a relatively low volume of business transactions. To meet the needs of the disconnected and heterogeneous solutions across CSEO, we’re using synthetic monitoring to test new features and performance of third-party applications and smoothly handle various authentication protocols, including platform as a service (PaaS) components in Microsoft Azure, corporate firewall connectivity, and multi-factor authentication. We’re building the platform to provide a mechanism to enable load, stress, and availability testing and provide a portal and API to enable CSEO teams to onboard and manage their configurations.

Service health reporting
We’re monitoring service health metrics and key performance indicators (KPIs) across the organization to understand customer sentiment and ensure services are reliable, compliant, and performing well. We’re using consistent standards, which helps ensure that we can aggregate data at any level in the service hierarchy and compare it across different team groups. Monitoring and reporting on service health requires onboarding and integrating across our unified telemetry platform, custom dimensions, and service-health dashboards built on Power BI. We’re building a more integrated experience on top of Azure Monitor, enriched with contextual data from the unified telemetry platform, and we’re creating a set of defined service-health measures and an analyzer to track events that can affect service reliability, such as upcoming planned maintenance or compliance-related changes. This enables us to proactively and quickly detect and resolve issues. Defined service-health measures make it easier to enable service-health reporting across various technologies, including Application Insights, custom-service monitoring, and third-party services.

We know that we must connect service health to business-process health, and how we prioritize issues, so that engineers can address them in a way that reduces the negative business impact. The experience we’re building enables visualization of end-to-end business-process health and the health of the underlying services by analyzing their telemetry. We’re templatizing how visualizations are built for business-process monitoring, and we’re scaling these templates to simplify the visualization of other related metrics.

We’re also simplifying the flow of service-health and engineering-fundamentals data to the engineer and reducing the number of dashboards and tools they use. We’re using an internal tool as the key repository for all service owners to view service health and other relevant KPIs. The tool’s integrated notification workflow informs service owners when a service reaches a defined threshold, making it more convenient to prioritize any needed remediation into their backlogs.

Embracing a Live Site culture
Increasing scale and agility in our services and processes requires integrating into our corporate culture more of a focus on making customers’ experiences better. We’re establishing a Live Site culture, and pursuing excellence, via customer-obsessed, data-driven, multidisciplinary teams that embrace potential failure by using honest observation, continuous learning, and measurable improvement targets. We host an organization-wide, live-site review, during which we perform postmortem reviews on incidents, examine long-term remediation plans, and guide service teams through modern engineering standards that will help them perform robust reviews at a local level. We base these reviews on standard and actionable reports that contain leading indicators for outages or failures based on the
analysis of telemetry, synthetic monitoring, and other data. We’re also working toward broadening these reviews, factoring in quantified business impact derived from business-process monitoring.

**Using customer feedback to drive development**

We’re keeping the customer experience at the center of the engineering process via feedback loop mechanisms. Feedback loops serve as a foundation for hypothesis-driven product improvements based on actual sentiment and usage data. We’re making feedback submission as easy as possible by using the same tool that the Microsoft Office product suite uses. The Send a Smile feature automatically gathers feedback consistently across multiple channels and key user touchpoints. We’re also using this tool as a centralized data system for storing, triaging, and analyzing feedback, and then aggregating it into actionable insights. We’re delivering best practices, training, and tooling to help our engineers understand this feedback and encourage adoption of feedback loops and experimentation methods, such as feature flighting and ring deployment, to help measure the impact of product changes. With these foundational components in place, we’re exploring ways to correlate feedback data with related telemetry, so that we can better understand product-usability issues and the impact of service issues on customers. We’re also using a controlled rollout to eliminate the need for UAT environments, which slow down overall delivery.

**Rapid delivery**

To be customer-obsessed, we’re acquiring and protecting customer trust in every aspect of our relationship. However, to drive this, our primary focus must be the quality of our services. We are tracking delivery metrics so that we can shorten lead times from ingestion of customer requirements to customer usability and feedback, all the while ensuring service reliability. We’re helping engineers achieve this objective by checking for issues earlier in the pipeline and providing a way to rapidly experiment and mitigate risk. We are building feedback-loop mechanisms to ensure that we can understand the user experience as new functionality gets deployed, and we perform automated rollbacks if customer reaction or service-health signals are less favorable than we anticipated.

To be customer-obsessed, we must have high-quality services. We’re shortening the lead time from when we receive a customer requirement to the time the solution is in the customer’s hands. Reduced lead time allows us to give and receive feedback while ensuring that we’re delivering secure, compliant, accessible, dependable, and high-quality services. This is critical to building trust with our customers. Our engineers are checking for issues earlier in the pipeline, and we’re enabling them to experiment rapidly while limiting potential negative effect on the release process.

**Integrating security, accessibility, and fundamentals**

We are advocating a “shift-left” process, in which work happens as early in the development process as possible. This enables us to avoid carrying debt from sprint to sprint. We’re also implementing gates in the developer workflow that help build security in a streamlined way and auto-onboarding services to ensure continuous compliance via static and dynamic security tools. We’ll log bugs in Azure DevOps that we discover during the scanning process, so developers can fix them there directly rather than having to first triage from security tools. Furthermore, we plan to apply machine-learning capability so that these bugs are found in real time and surfaced for action immediately at build. This allows for developers to use the same engineering system as functional bugs for triage, prioritization, and tracking.

We have an independent vendor who assesses accessibility within our applications, but this happens late in the development process. To move this further upstream, we’re driving the adoption of accessibility-insights tooling during development and will also expose accessibility-related bugs as part of the pipeline workflow.

Additionally, we’re enabling engineering teams to utilize the guardrails we’re implementing by integrating fundamentals into the pipeline, and we’re evangelizing continuous integration practices so that all production releases (including hot fixes) come from master builds and have all appropriate compliance steps applied consistently. Each pull request must have a successful build to ensure that the master is golden and always production ready. Maintaining high-quality code in the master will minimize build failures that ultimately slow our time to production.
Deploying safely to customers
We’re creating an environment where teams test ideas and prototypes before building them, and we’re really focusing on what drives customer outcomes in a way that encourages risk-taking with a fail-fast, fail-safe mentality. Central to increasing the velocity of service updates to customers is a consistent, simple, and streamlined way to implement safe deployments to customers. Progressive exposure and feature flags are key in deploying new capabilities to users via controlled rollouts, so we can quickly start receiving customer feedback. We’ll implement checks and balances in the process by leveraging service indicators such as latency and faults within the pipeline, thereby catching regressions and allowing initiation of automated rollbacks when predefined thresholds are exceeded. Implementing safe deployment practices, combined with a streamlined and well-managed pipeline, are two of the key elements for achieving a continuous integration, continuous deployment (CI/CD) model.

Enabling code reuse
While a low volume, in less than 5 percent of our services, we’re still supporting applications that use on-premises servers and domain-joined Azure virtual machines. This results in ongoing effort to patch servers, upgrade software, and perform basic infrastructure-maintenance tasks. It also impedes our ability to scale apps to accommodate growth. We’re continuing to invest in transforming these applications to Microsoft Azure platform-as-a-service (PaaS) and software-as-a-service (SaaS)-based solutions, thereby leverage the scale and availability of Azure. We’re enabling this by providing architectural guidance and tools to migrate data, refactor existing functionality as APIs, and build lightweight applications by reusing APIs that others have already published.

Promoting data and code reuse to build solutions more rapidly and align with a service-oriented architecture requires that developers can easily publish and discover APIs. We’re building an API economy by creating a common set of guidelines for developing coherent APIs, and a central catalog and search experience for discovery. We’re integrating validation against API guidelines and enabling our teams to integrate API publishing into their Azure DevOps pipelines, and we’re defining and providing a set of common API health analytics. We’ll also continue working on practices to enable growth of “inner-source” in which sharing of code outside of APIs is achieved. This will help us extend our modern engineering practices to other organizations within Microsoft, where business-led engineering or “shadow IT” occurs today.

Engineering productivity
We’re providing our engineers with best-in-class unified standards and practices in a common engineering system, based on the latest Azure tools, such as Azure DevOps. A consistent development environment allows our engineers to transition smoothly between projects and teams. Improved automation, consistency, and centralized engineering systems enable engineers to better focus on the core role of developing. This also reduces onboarding time and allows our engineers to be more flexible across projects.

Increasing conformity
We’re creating and supporting standards to normalize our practices as an organization while continuing to let teams self-manage within these boundaries. This helps ensure that we have a more streamlined team and more productive engineers. These standards include using the following:

• Common work-item taxonomy and path structures within Azure Dev Ops. By standardizing work-item taxonomy and path structures for area and iteration, we’re allowing engineers to focus on engineering by providing a consistent mechanism to connect each team’s work back to CSEO goals and priorities. We’re facilitating CSEO-wide querying from Azure DevOps for deliverables such as compliance, accessibility, and delivery planning.

• Naming standard for production environments. Having a standard naming convention for different environments (dev, test, prod, and similar) will make them easy to differentiate and allow for automation of environment-specific policies across the organization.

• Standards for Azure DevOps tags to provide a lightweight method in which to mark related items. We do this by creating and socializing a finite set of tags that we use in conjunction with common iteration paths to provide an easy way to track completion progress over time.
Integrating developer tooling
We’re making organizationally mandated and recommended code analysis and compliance tools accessible directly within the development environment, thereby helping our “shift-left” goal. Our developers typically have a few ways in which they learn about the tools they need to install, such as from their team’s OneNote or wikis, or by searching through public-marketplace extensions. This takes them out of their primary development-environment context and can be very time-consuming. We’re investing in self-service capabilities to manage access, set policies, and make changes to Azure DevOps artifacts such as area paths, work items, and repositories. We’ll make it easy for engineers to create, update, or retire services, components, and subscriptions, minimizing the time spent managing such resources. We are streamlining user onboarding and artifact management by automating key workflows across tools, such as those to improve engineering productivity while ensuring cross-tool data integrity and compliance. For any investment in this work, we partner closely with the Visual Studio, VSCode, and Azure DevOps teams, as well as many others. We want to extend our “shift-left” goal to also examine optimization of our Azure service design and surface recommendations for configuration optimization so that these occur early in the deployment cycle and allow us to right-size our configurations and avoid unnecessary Azure costs.

Making it easy to discover resources
We constantly check in with our engineers on our engineering system’s health and benchmark against other Microsoft product groups. Survey feedback indicates that finding engineering related assets has been a challenge. We’re working toward easy and reliable ways to share functionality and reduce redundant design. We want our best practices to be easy to find within a central location, which will help our engineers understand the end-to-end processes and tools that they need to use to keep applications healthy, from a reporting perspective. We’re addressing many of these opportunities by the investments that are underway, such as building an API economy, scaling service health, and implementing a common reporting tool for service-health data. Furthermore, we’re building a mechanism to manage and maintain end-to-end engineering-process documentation and defining a way to publish, maintain, and discover engineering best practices within the organization; this will also be made available to the shadow-IT organizations.

Universal design system
We’re leveraging Microsoft’s product design system to engineer solutions that look and behave like other Microsoft products. We want CSEO products to meet the expectations of product quality that today’s consumers expect, meaning that every piece of the user interface (UI) and user experience (UX) should be engineered with accessibility, responsiveness, and familiar behaviors, states, motion, and visual styling. On complex but common components like headers, navigation menus, and data grids this can mean weeks of engineering time multiplied across every CSEO team. To achieve the “shift-left” goal, our developers typically have a few ways to discover engineering related assets such as from their team’s OneNote or wikis, or by searching through public-marketplace extensions. This takes them out of their primary development-environment context and can be very time-consuming. We’re investing in self-service capabilities to manage access, set policies, and make changes to Azure DevOps artifacts such as area paths, work items, and repositories. We’ll make it easy for engineers to create, update, or retire services, components, and subscriptions, minimizing the time spent managing such resources. We are streamlining user onboarding and artifact management by automating key workflows across tools, such as those to improve engineering productivity while ensuring cross-tool data integrity and compliance. For any investment in this work, we partner closely with the Visual Studio, VSCode, and Azure DevOps teams, as well as many others. We want to extend our “shift-left” goal to also examine optimization of our Azure service design and surface recommendations for configuration optimization so that these occur early in the deployment cycle and allow us to right-size our configurations and avoid unnecessary Azure costs.

Onboarding program
Providing newly hired personnel with an understanding of how we engineer is not only important, so that they have a good onboarding-integration experience, but it’s vital to ensuring that our cultural change keeps its momentum. A three-day orientation that was designed by the Fundamentals team helps new personnel understand the organization at a high level, providing key concepts regarding our culture, technologies, and modern engineering practices including two days of Azure training. We’re supplementing this with a 90-day self-paced set of onboarding training
sessions that we can tailor by geography, role, and team. Additionally, 30 days after the initial onboarding training, we'll provide another round of training to ensure a concrete understanding of the CSEO organization. We're also planning supplemental training including:

- Project Manager and Engineering Days to highlight and celebrate the progress in our transformation.
- A specific track for program managers to educate new employees on the expectations of a program manager in CSEO.
- Agile training to introduce Agile Scrum and deepen the understanding of Agile concepts.
- A communications track to build communication and storytelling skills. We're piloting an "Influence through Story Telling" training session to help our employees develop key messages that are persuasive, drive relevant insights for stakeholders, and teach how to be impactful in impromptu conversations.

**Conclusion**

We’re making our vision for modern engineering a reality at Microsoft. We’re promoting a “Live Site first” culture, using data to provide service and business-process health signals to inform the rapid iteration on new ideas and capabilities with customers. We’re supporting this by moving to a “DevOps” model of continuous integration and continuous deployment governed by a standardized engineering pipeline with automatic policy enforcement. This culture, and the tools and ceremonies that support it, have increased visibility into engineering processes, improved the quality and delivery of our services and improved our insight into our customer experiences, all of which ensures we are continually improving and adapting our set of services and processes to support digital transformation now and into the future.

**For more information**

**Microsoft IT Showcase**

[www.microsoft.com/itshowcase](http://www.microsoft.com/itshowcase)

Read about the transformation and major initiatives shaping Microsoft

© 2020 Microsoft Corporation. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.