‹packt›

1ST EDITION

Industrializing Financial Services with DevOps

Proven 360° DevOps operating model practices for enabling a multi-speed bank

SPYRIDON MANIOTIS

Chapter 1

Images

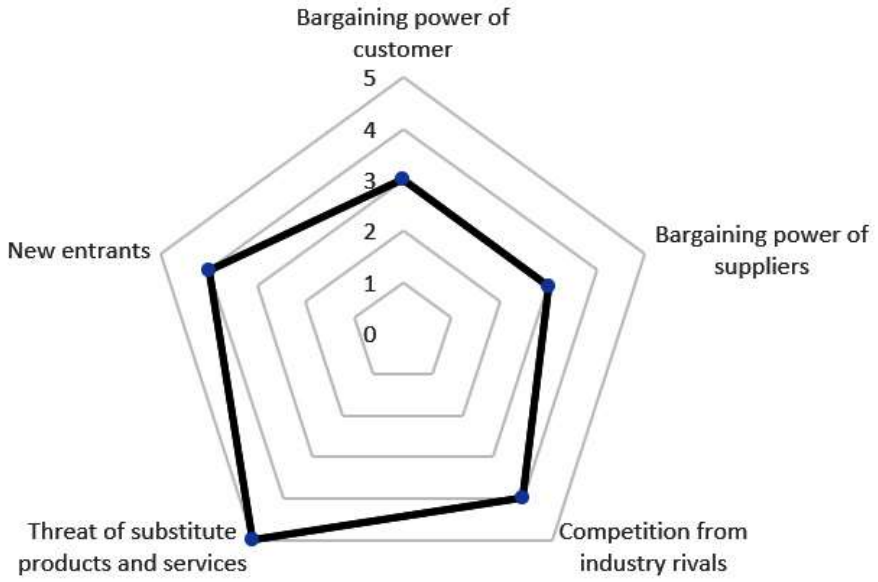


Figure 1.1 – Porter's Five Forces (1979) adapted to the financial services industry

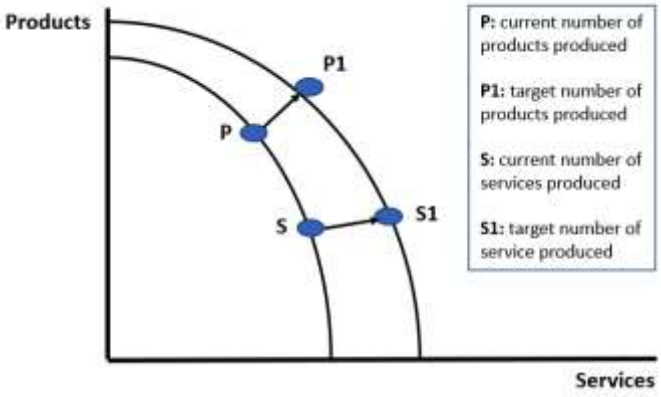


Figure 1.2 – Production possibilities frontier, adapted to incumbent banks

Figure 1.3 – Relevance example

Chapter 2

Images

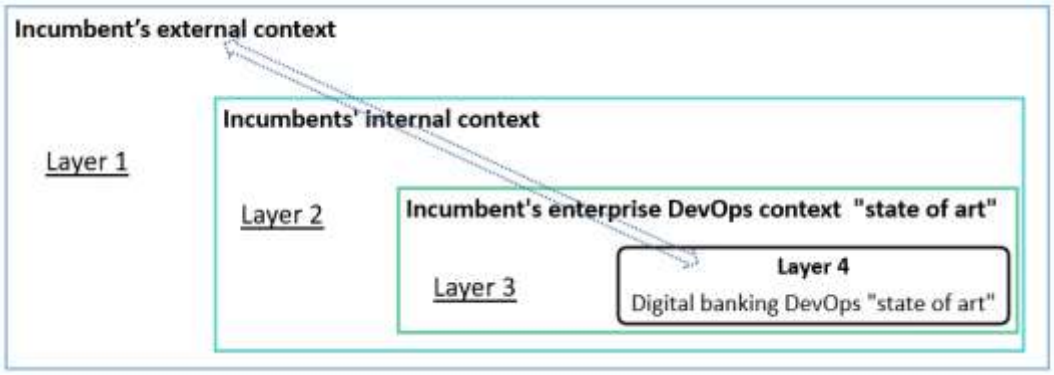


Figure 2.1 – The contextual layers of an incumbent that influence the DevOps adoption

Figure 2.2 – Diagram of the main forces and qualities that influence the DevOps vision

Corporate/Technology Strategic Objectives Reconciliation	Run the bank	Act commercially	Modernize and deal with legacy	Improve WoW and efficiency	Build or aquire skills	Compliant and secure by design	Plan ahead in alignment	Simplify and standardize	Deliver on commitments
Accelerate digitalization	R		R	R	R			R	
Best-in-class customer experience	R		R	R		R	R	R	
Operational and capital efficiency	R	R	R	R				R	R
Preferred partner for customers	R					R			R
Compliant and secure	R				R			R	R
Data driven	R	R		R			R	R	
Simplicity and standardization	R	R	R	R				R	R
People first					R				

Figure 2.3 – Corporate and technology strategy objectives reconciliation ("R" stands for reconciliation point)

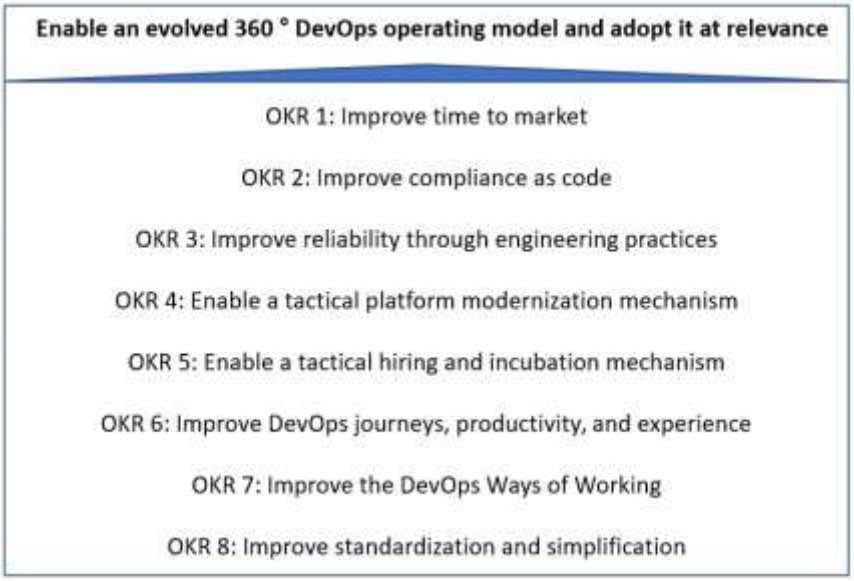


Figure 2.4 – DevOps enterprise adoption OKRs

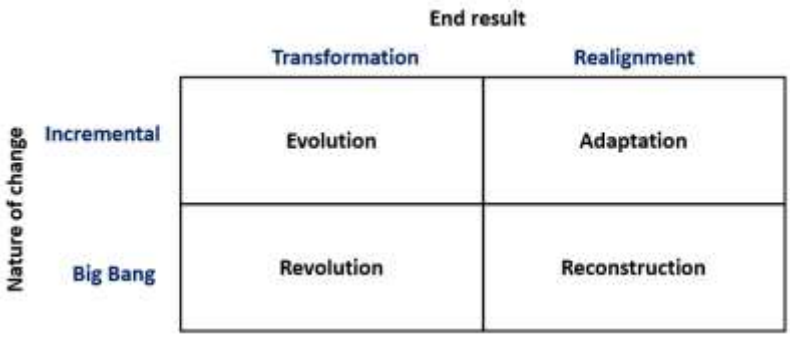


Figure 2.5 – Types of Change Source by Balogun and Hope Hailey (2004)

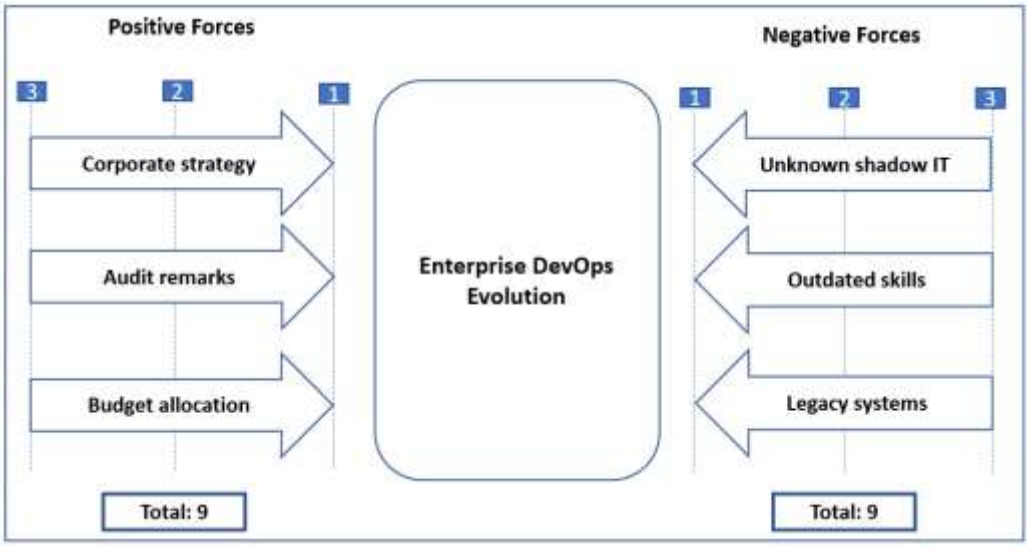


Figure 2.6 – Sample DevOps enterprise adoption Force Field Analysis

Tables

Play to Win (Home Markets)	Play Not to Lose (Challenger and Growth Markets)
First in the market advantage	Second mover advantage
Desire to shape the market	Desire to follow the market
Radical product innovation	Incremental product innovation
Cutting-edge greenfield products	Build on top of legacy products
Acquire new entrants	Collaborate with new entrants
Market share increase	Market share maintenance

Table 2.1 – Play to win versus play not to lose strategy differences

Objective	Key Results	Strategy Link	Motivating Factors
Improve time to market across the portfolio, especially for core digital services. This includes the following: 1) Mobile banking 2) Online banking 3) Open banking 4) Payments 5) Electronic trading	KR 1: Achieve "on-demand" releases to production for core digital services KR 2: Measure release velocity, reliability, and compliance balance KR 3: 70% of the modern application portfolio is using the standard CI/CD pipeline KR 4: Change management process automation	Accelerate digitalization at scale and deliver on commitments	1. Proliferation of new entrants, which introduces new products faster 2. Loss of customers due to inability to respond fast to demand

Table 2.2 – Sample of the improve time to market enterprise OKR

Objective	Key Results	Strategy Link	Motivating Factors
Strive toward compliance as code across our portfolio	KR 1: A minimum viable adherence IT control mechanism is enabled KR 2: Improvements in the policy and evidence engineering mechanisms KR 3: All audit remarks for business-critical applications are closed	Continuous compliance by design	1. Improve our efficiency in addressing the ECB and local FSA's audit remarks on IT risk management, to avoid reputational and capital loss 2. Minimize the compliance impact on time to market and productivity

Table 2.3 – Sample of the improve compliance as code enterprise OKR

Objective	Key Results	Strategy Link	Motivating Factors
Embed reliability engineering practices in the SDLC	KR 1: The reliability targets across the application portfolio are revised KR 2: Reliability engineering practices are implemented in the top 20 business-critical services KR 3: Reliability engineering practices are implemented in the common DevOps platforms	Create a best-in-class customer experience, run the bank, and deliver on commitments	1. FCA audit remarks on service fragility and domino effects 2. The top 20 business-critical services face instability 3. Fragility in our common DevOps platforms, which disturbs software delivery

Table 2.4 – Sample of the improve reliability through engineering practices OKR

Objective	Key Results	Strategy Link	Motivating Factors
Achieve platform modernization on business applications and their core infrastructure	KR 1: All newly built applications fulfill cloud-native, microservices, APIs, and event-driven architecture principles through reference architectures KR 2: Utilize PaaS in the public cloud and increase IaC utilization in the private cloud KR 3: A portfolio classification and modernization mechanism is defined KR 4: Interoperability issues in the DevOps tech ecosystem are resolved	Accelerate digitalization at scale and deal with the legacy	1. Legacy business applications pose an operational risk 2. Public cloud will require platform re-engineering 3. Lack of clarity on application classification and evolution 4. Interoperability gaps across the technological ecosystem

Table 2.5 – Sample of the tactical platform modernization enterprise OKR

Objective	Key Results	Strategy Link	Motivating Factors
Focus on people skills and incubation	KR 1: The hiring process is diversified and new near-shore locations are created KR 2: Engineers and developers are trained on our strategic technologies and platforms, enabling Π-shaped profiles KR 3: Internal job mobility is increased	Focus on people by building or acquiring skills	Stay relevant, support the DevOps transformation with the required skills, and retain and attract talent Reduce dependency on consultants

Table 2.6 – Sample of the tactical hiring and incubation enterprise OKR

Objective	Key Results	Strategy Link	Motivating Factors
Improve DevOps journeys and experience	KR 1: The DevOps infrastructure and tools service fulfillment and onboarding lead times have been reduced KR 2: The IaC capabilities of the private cloud have been enriched KR 3: Value streams across DevOps platforms are defined KR 4: The public cloud onboarding process is reduced to 2 weeks	Improve operational efficiency and minimize cost	1. Simplify the way we deal with each other at three levels: 1) client to bank, 2) internal business units to IT, and 3) IT to IT 2. Increase DevOps productivity through technological advancements 3. Gain value streams flow visibility across the DevOps journeys, continuity, and experience

Table 2.7 – Sample of the improve DevOps journeys and experience enterprise OKR

Objective	Key Results	Strategy Link	Motivating Factors
New WoW	KR 1: The organizational silos across the SDLC's stakeholders collapse KR 2: Operations shift left on the SDLC KR 3: The business agility, SRE, ITIL, and DevOps models are reconciled KR 4: The daily DevOps organizing principles and topologies are aligned KR 5: The portfolio planning mechanism is advanced KR 6: The SDLC flows are harmonized and advanced	Simplification, collaboration, and operational efficiencies	1. Shift toward common WoW to ensure value streams have end-to-end integration and visibility 2. Eliminate existing continuity gaps among the implemented concepts 3. Modernize our outdated SDLC to stay relevant 4. Ensure alignment on daily operations to stay compliant

Table 2.8 – Sample of the improve the DevOps WoW enterprise OKR

Objective	Key Results	Strategy Link	Motivating Factors
Standardize and simplify from a DevOps 360° perspective	KR 1: Common DevOps platforms are standardized through a capability menu and technology radar and shadow IT is eliminated KR 2: The SDLC is simplified for cloud-native apps KR 3: Usage of standard performance mechanism to measure the realization benefits KR 4: DevOps tools, platforms, and processes are examined based on their future purpose KR 5: The service registry mechanism is standardized KR 6: The DevOps capabilities consumption in an "as a service" model is simplified	Standardization, simplification, cost optimization, and operational efficiencies	1. Reduce waste and duplications and improve the economies of scale to achieve operational efficiency 2. Reduce shadow IT and technical debt to reduce operational risk 3. Speed up DevOps adoption by utilizing solution templates and recipes

Table 2.9 – Sample of the improve standardization and simplification enterprise OKR

Chapter 3

Images

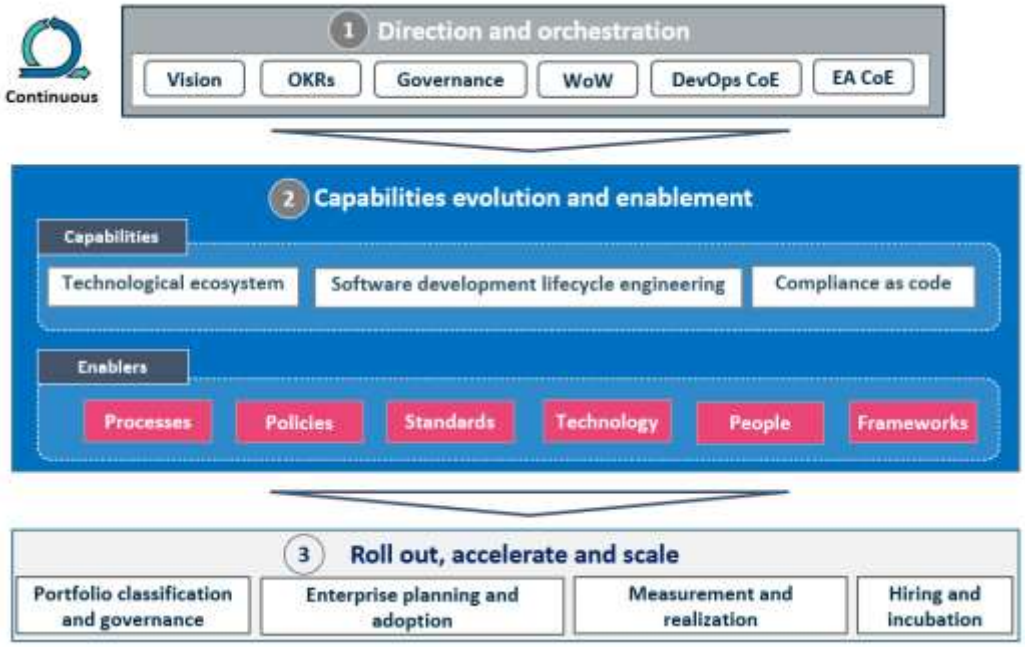


Figure 3.1 – The DevOps evolution 360° operating model skeleton

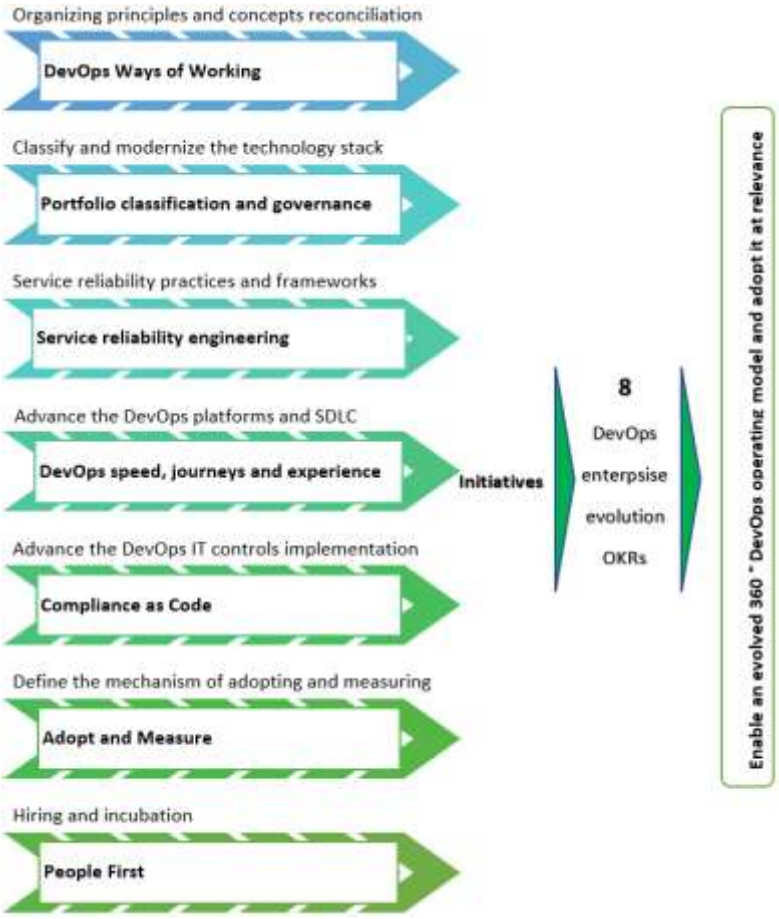


Figure 3.2 – Overview of the DevOps enterprise evolution workstreams

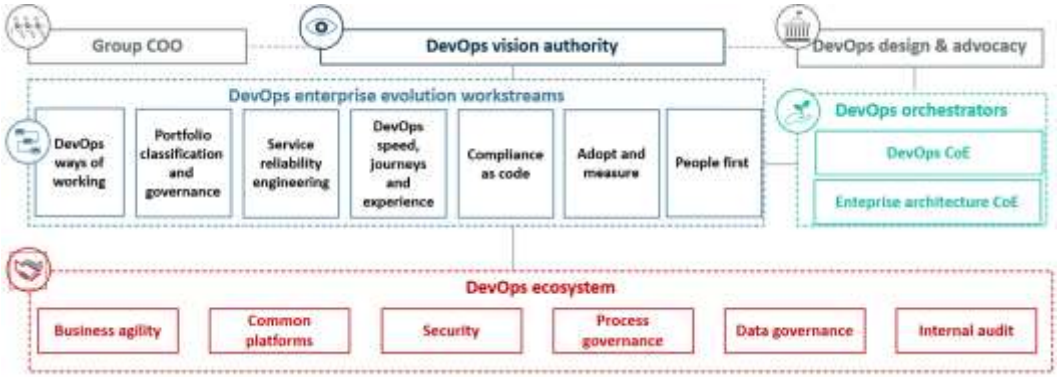


Figure 3.3 – Governance bodies and stakeholders illustration

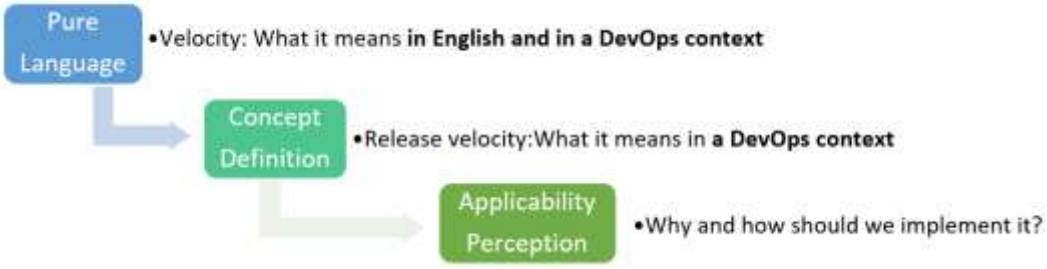


Figure 3.4 – The DevOps language tree

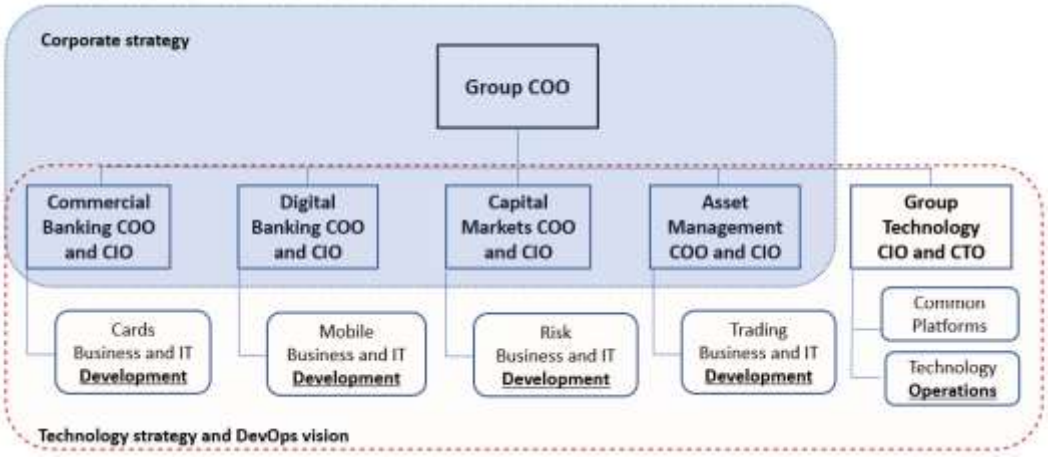


Figure 3.5 – Segregated business technology lines and group technology structure

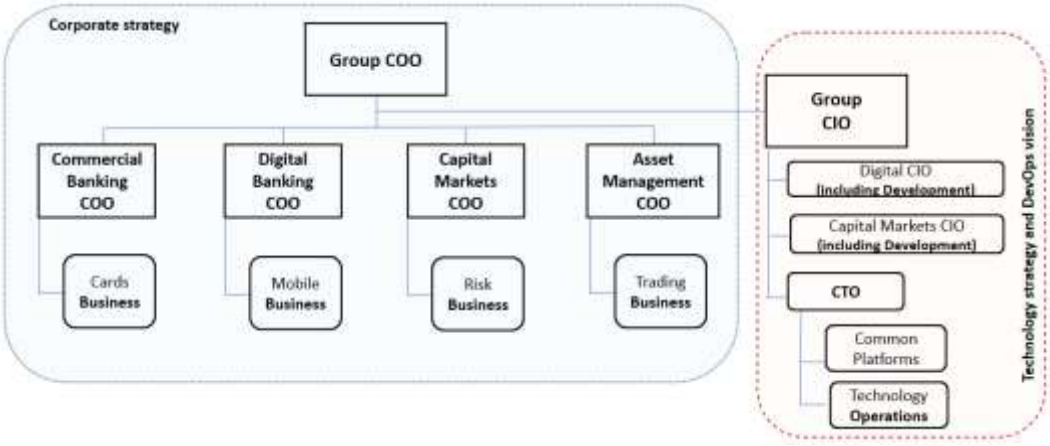


Figure 3.6 – Segregated business lines and technology

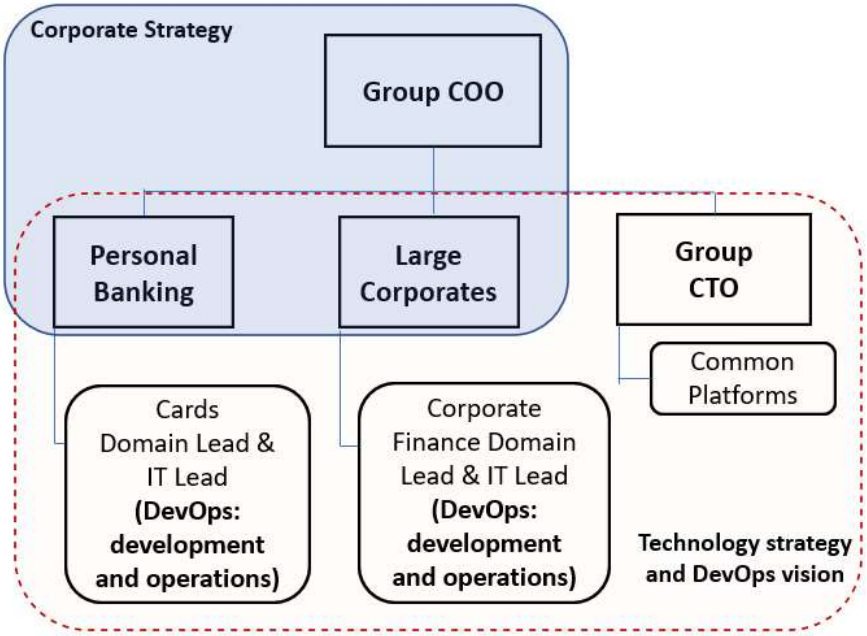


Figure 3.7 – Consolidated business and IT domains segregated from the core technology

Tables

DevOps Domain/Stakeholder	Value Proposition
Primary	
Software Development and Operations	Builds and runs software. The primary consumers of technology utilities and DevOps capabilities.
Business Units	Owns and uses products, applications, and services. Manages DevOps priorities for business applications and funding.
Enterprise Architecture	Forms the governing mechanism for architecting your portfolio of applications, services, platforms, domains, and flows.
DevOps CoE or CoP	Sets your organization's DevOps standards.
Control Center	This is the first line of service operations and continuity.
Enterprise Service Desk	Handles business inquiry fulfillment on IT applications.
Process Governance	Owns the service life cycle operational processes.
Common Platforms	Comprises utility teams offering technological DevOps capabilities.
Cyber Security	Enables security policies across your SDLC.
Service Portfolio Governance	Form the application and service registry mechanism.
IT Risk and Controls	Implements the IT risk management framework.
Business Agility CoE or CoP	Oversees the business agility model of your organization.
Data Governance	Enacts data classification and protection policies.
Quality Assurance	Creates quality assurance strategies and plans.
Core infrastructure	Provides the core technology infrastructure capabilities.
Data Analytics	Provides telemetry and reporting capabilities.
IT Audit	Provides an overview of open IT audit remarks.
Talent Acquisition and HR	Manages recruitment, mobility, incubation, and job descriptions.
User Experience	Manages DevOps journeys, experience, and value streams.
Regulators	Manage the alignment of regulatory demand responses.

Table 3.1 – Primary DevOps ecosystem stakeholders

DevOps Domain/Stakeholder	Value Proposition
Secondary	
Legal	Supports regulatory negotiations
Vendor Management	Manages the renegotiation of contract terms
Third-Party Vendors	Offer technological foundation utilities
Managed Services	Offshore teams that support development and operations
Friendly Customers	Customers that support the piloting of new products
Partnerships	Partnering incumbent banks, technology innovators, and management consulting firms

Table 3.2 – Secondary DevOps ecosystem stakeholders

Roles	Description
Owner	A vision authority group member who has expertise in the workstream's domain of focus. This member also represents the business partners of the incumbent.
Lead	A senior member of the design and advocacy group who has expertise in the workstream's domain of focus and who has coordination experience. This member, along with the owner, represents the business units.
Integrator	A person ensuring reconciliation with other workstreams running in parallel.
Squads	Permanent people who are appointed to be part of the workstream and come from the broader DevOps ecosystem. They are fixed for each workstream.
Satellites	Non-permanent members who join only when their domain of expertise comes into focus. They are not permanent and support several workstreams in parallel.

Table 3.3 – Roles within workstreams

The Organizationally Aligned	
Differentiation	Clear segregation of duties and responsibilities in the team
Centralization	Managers are fully aligned with central organizational priorities and decisions
Standardization	Tools, processes, and policies follow central standardization
Supervision	"Do not trust, get evidence and verify"
The Organizationally Autonomous	

De-differentiation	Freedom when it comes to the allocation of tasks and responsibility
De-centralization	People have the space and mandate to go in their own directions
De-standardization	Flexibility in bypassing central tools, processes, and policies
De-supervision	“Trust, do not verify”

Table 3.4 – Organizationally aligned teams versus autonomous teams

Chapter 4

Images

Figure 4.1 – Core business criticality domains example in banking

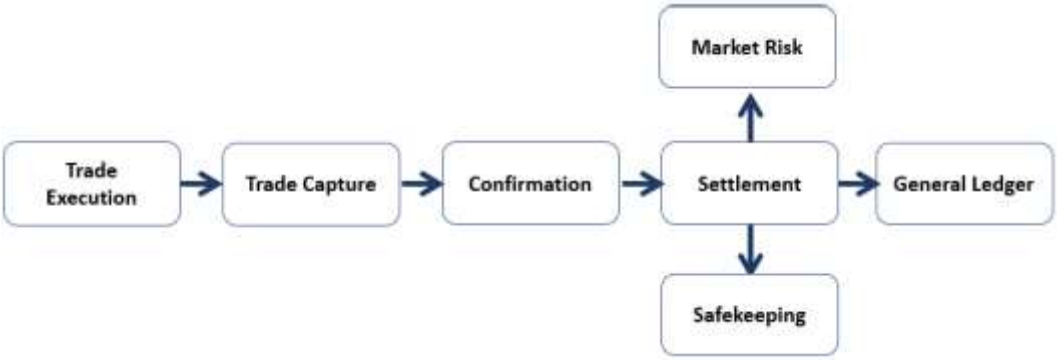


Figure 4.2 – Trade life cycle flow example

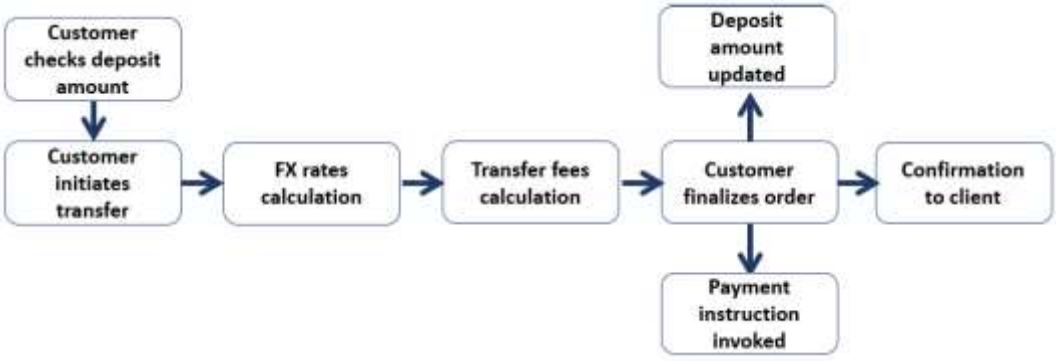


Figure 4.3 – Cross-border payment flow example

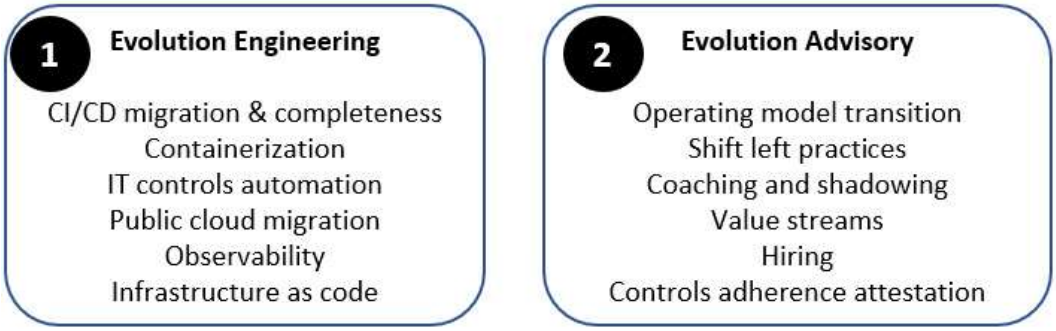


Figure 4.4 – CoE engineering and advisory services catalog example

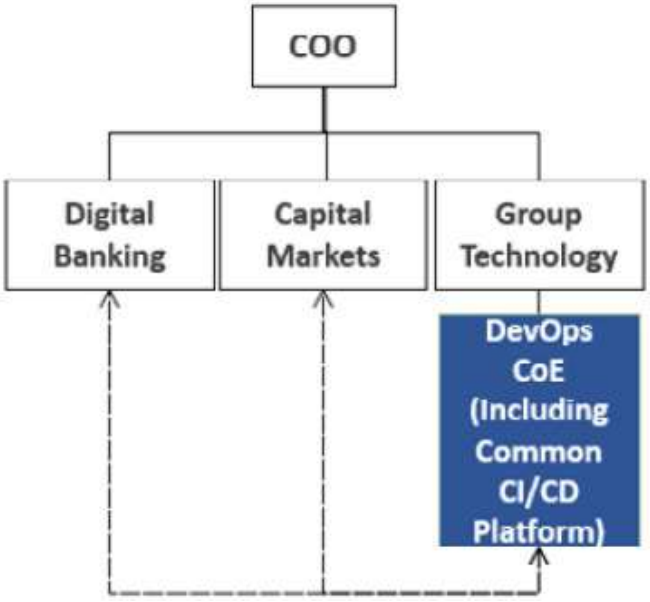


Figure 4.5 – The three-angles modus operandi CoE

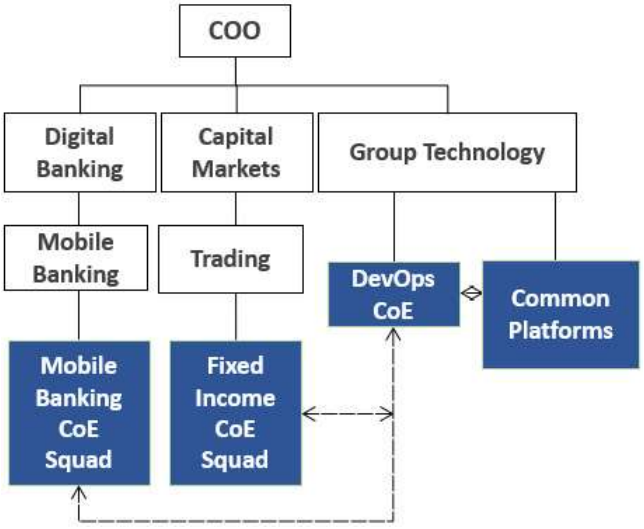


Figure 4.6 – The model owner and tactical adoption enabler CoE

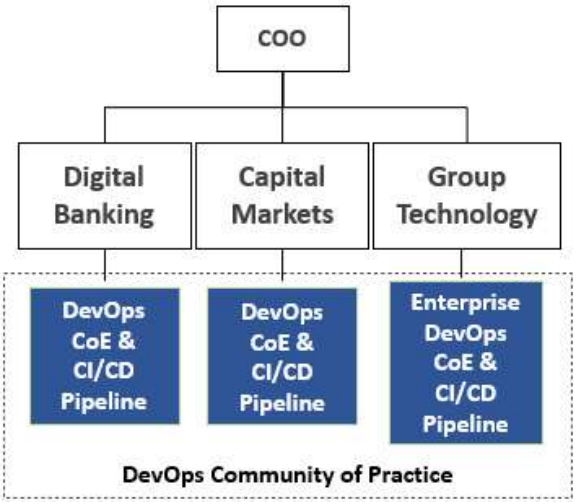


Figure 4.7 – The Enterprise CoE as part of a DevOps CoP

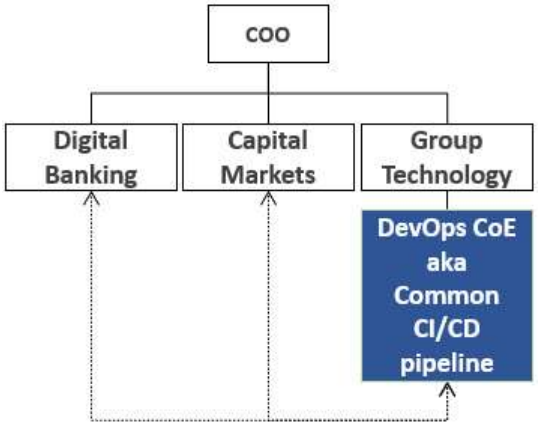


Figure 4.8 – The CI/CD DevOps CoE

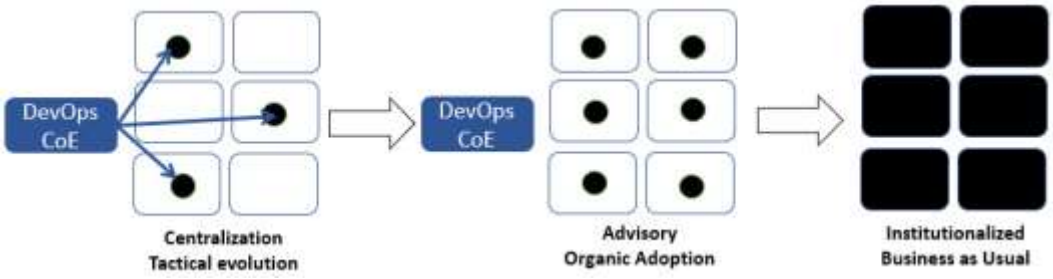


Figure 4.9 – The main phases of the CoE's life cycle toward scaled DevOps institutionalization

Tables

	Do nothing and sustain	Upgrade or incrementally improve
Business case elements	The platform is considered to be: • Fit for future purpose • Sustainable • Cost-effective • Low to no operational risk fragility	The platform is considered to be: • Made fit for future purpose via gradual improvements and/or upgrades • Sustainable • Cost-acceptable • Low operational risk fragility

Portfolio examples	1. CRMs, HR, and payroll systems 2. CI/CD pipelines 3. Non-client-facing applications and tools, used by small internal business teams 4. Very low criticality and stable non-mainframe applications with a release frequency of twice a year	1. Highly reputable vendor platforms, especially in the trading and group functions domains, with vendor lock-in and no intention or expertise to self-build (examples: Front Arena, Calypso, Murex, and SAP) 2. Self-built applications and platforms that are not part of the core or digital frontier and do not go through frequent code changes (examples: reference data and FX)

Table 4.1 – The "Do nothing and sustain" and "Upgrade or incrementally improve" approaches

	Refactor and/or decouple	**Full replacement, consolidation, or divestment**
Business case elements	The platform is considered to be: • Not fit for future purpose • Unsustainable from maintenance, efficiency, and cost perspectives • To pose short- or long-term operational risk On decoupling, a full replacement is considered operationally risky or cannot be funded.	The platform is considered to be: • Not fit for future purpose • Unsustainable from the perspectives of operations, meeting business needs, and cost • Under pricy soon-to-expire vendor contracts • Not able to cope with innovation growth The platform will either be replaced or terminated.

Portfolio examples	1. Migration and replication from HPS and PL/SQL business logic to Java, primarily in back and middle office functions, such as accounting and reporting 2. New platforms; replacing legacy logic with new business logic; built-in microservices; eliminating data dependencies through APIs 3. Cloud-native transformations and business logic migration to a public cloud 4. Ecosystem refactoring for regulatory purposes	1. Core banking platforms, primarily replaced by well-reputed vendor solutions (example: Temenos) 2. Legacy country-specific platforms, whose business logic is recreated and consolidated under one global platform, both vendor and self-built 3. Lines of business being terminated, consolidated, or divested 4. Legacy data transmission channels and storage 5. Legacy observability, automation, and identity and access management tools

Table 4.2 – The "Refactor and/or decouple" and "Full replacement, consolidation, or divestment" approaches

Engagement Name	Market and Credit Risk Flow DevOps Acceleration and Compliance
CoE Members	Three engineers: Michal, Alex, and Aurimas; two expert advisors: Tomasz and Mario
Objectives & Benefits	• Migration of home-grown CI/CDs to the central CI/CD pipeline: *Improve time to market and reduce maintenance overhead* • Microservices and containerization: *Platform modernization, data decoupling, and technical debt reduction* • Monitoring and logging standardization: *Enhance visibility and proactive capacity management* • SDLC controls completion and shift operations left: *Enhance compliance as coded coverage and improve reliability*
Funding	The **Fundamental Review of the Trading Book** (**FRTB**) program

Table 4.3 – Sample engagement template

Pros	Cons
Standardization and simplification of DevOps practices across engagements.	The CoE could not scale outside the pre-defined client engagements.
Effective and tactical utilization of the CoE's resources.	The "DevOps gap" between those supported and not supported by the CoE widened.
Economies of scale through reusability.	The CoE could not capture all the business IT context-related feedback and specialties.
A single recognized owner and point of entry for the DevOps operating model.	High capability concentration and density occasionally resulted in CoE unilateralism.
Career mobility within the CoE.	The client engagement rotations had a mixed effect due to switching context and focus frequently, as well as leaving gaps behind.
DevOps model frameworks and CI/CD technology alignment.	Feeling of exclusion in the broader organization due to fixed CoE engagements.
The CoE's engagements dominated the backlog priorities for the model and CI/CD evolution. (*This is a positive as those teams were the most advanced in the organization*.)	The CoE's engagements dominated the backlog priorities for the model and CI/CD. (*This is a negative due to the feeling of exclusion from the rest of the organization and a lack of focus on their requirements*.)

Table 4.4 – Pros and cons of the three angles modus operandi CoE

Pros	Cons
Standardization and simplification of DevOps practices across engagements.	The CoE could not scale outside the pre-defined client engagements.
Effective and tactical utilization of CoE resources focused on building the business IT area solutions.	The "DevOps gap" between those supported and not supported by the CoE widened.
Economies of scale through reusability.	Priorities and authority conflicts arose between the CoE and platform teams.
A single recognized owner and point of entry for the DevOps operating model.	The organization continuously and mistakenly thought that the CI/CD platform team was under the CoE, which resulted in communication and expectation complexity.

The CoE's engagements dominated the backlog priorities for the model. (*This is a positive as those teams were the most advanced in the organization*.)	The CoE's engagements dominated the backlog priorities for the model. (*This is a negative due to feeling of exclusion from the rest of the organization and lack of focus on their requirements*.)
Cross-business IT areas experienced a certain degree of synergy driven by the CoE.	Due to their dedication to business IT areas, the CoE people lost the sense of belonging to the CoE.

Table 4.5 – Pros and cons of the model owner and tactical adoption CoE

Pros	**Cons**
Flexibility and focus on the lines of business, allowing them to evolve as required by the business context.	Cross-organizational standardization was made very complex.
A degree of sharing practices across the organization was achieved through the CoP.	Lack of transparency about the enterprise DevOps adoption "exceptions."
Economies of scale and cost management within business IT lines and group technology were achieved.	The CoP lost attention from its members in the long run, while unilateralism increased between the lines of business and the group technology.
The central ownership of the model was broadly recognized.	Lack of enablement teams in the group technology to support adoption.

Table 4.6 – The Enterprise CoE as part of a DevOps CoP pros and cons

Chapter 5

Images

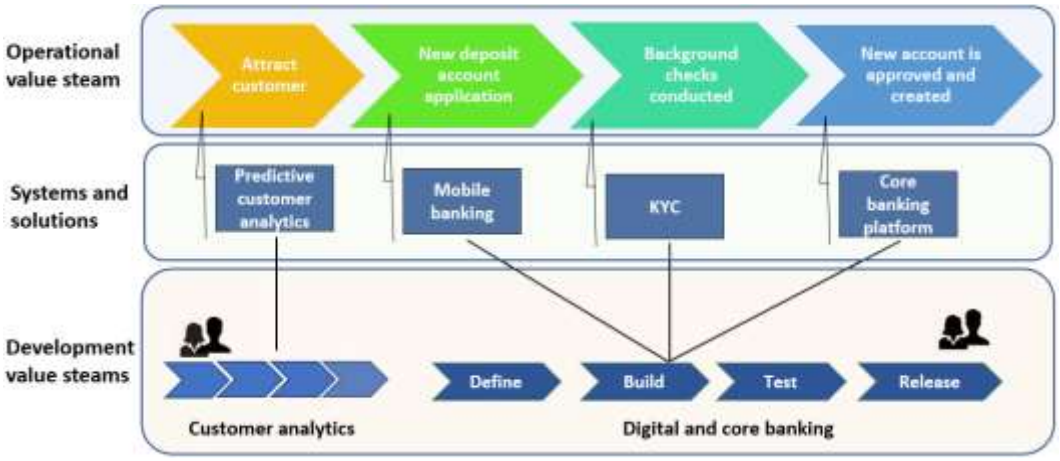


Figure 5.1 – Diagram of new deposits account opening value streams

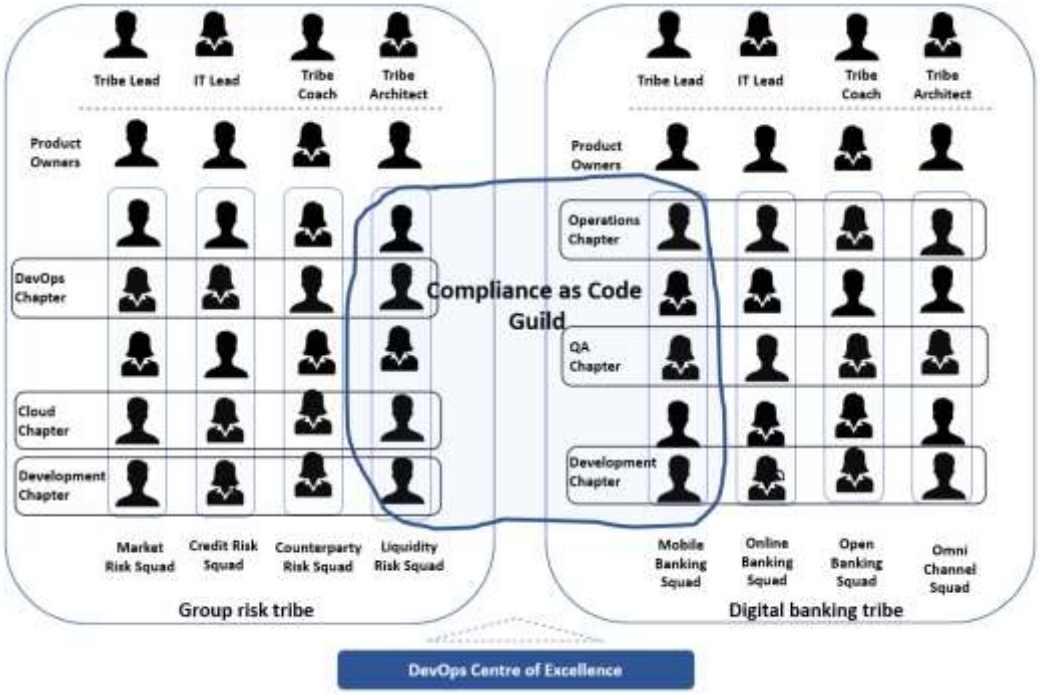


Figure 5.2 – Group risk and digital banking tribes representative example

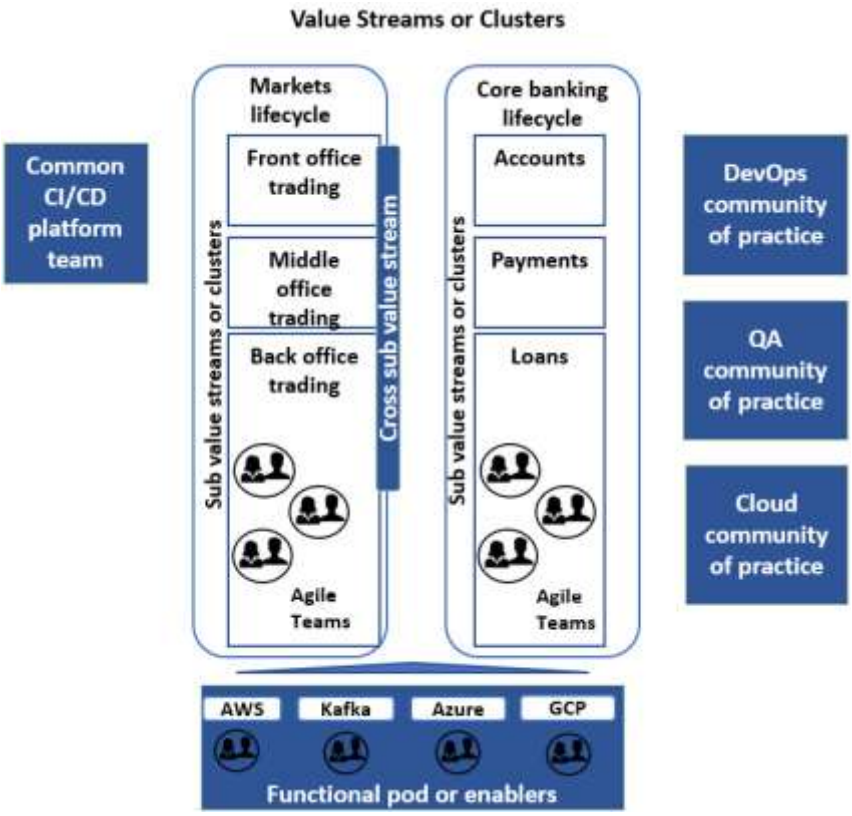


Figure 5.3 – Visual representation of the value streams or clusters organizing structure

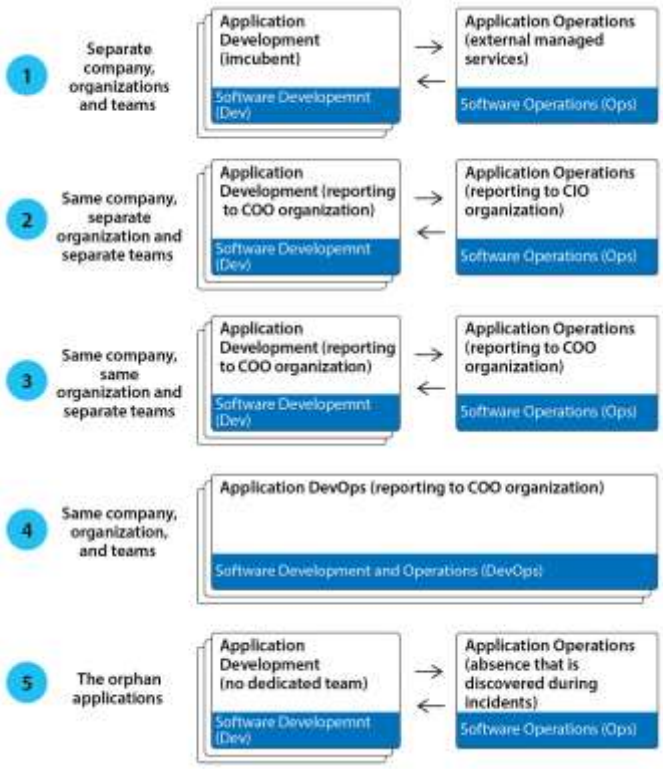


Figure 5.4 – The common models of the primary actors that you can find coexisting in an incumbent

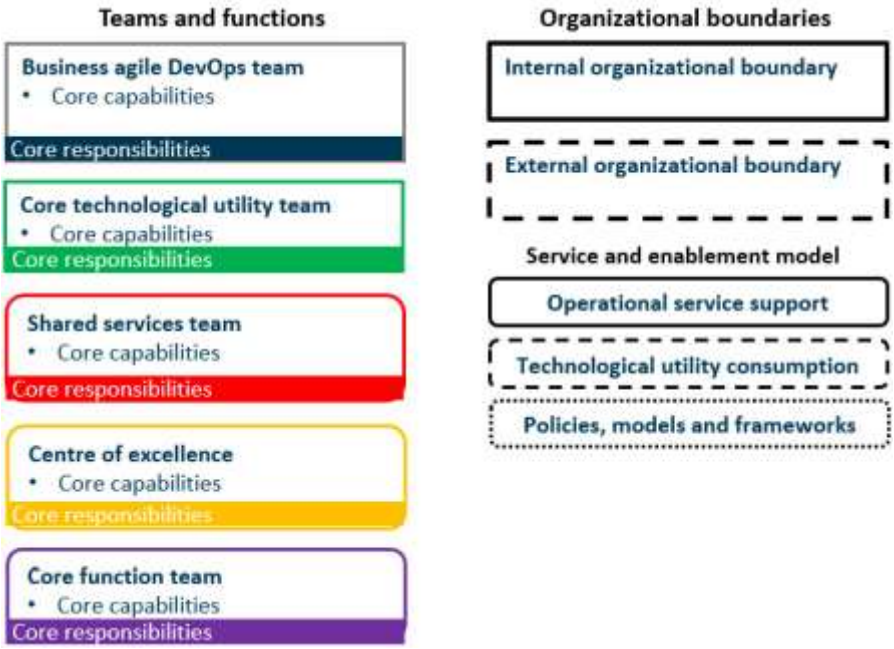


Figure 5.5 – Spyridon's topologies and organizing principles notation

Figure 5.6 – The dynamic/rotational model – separation/segregation of tasks

Figure 5.7 – Fixed roles model – separation/segregation of duties

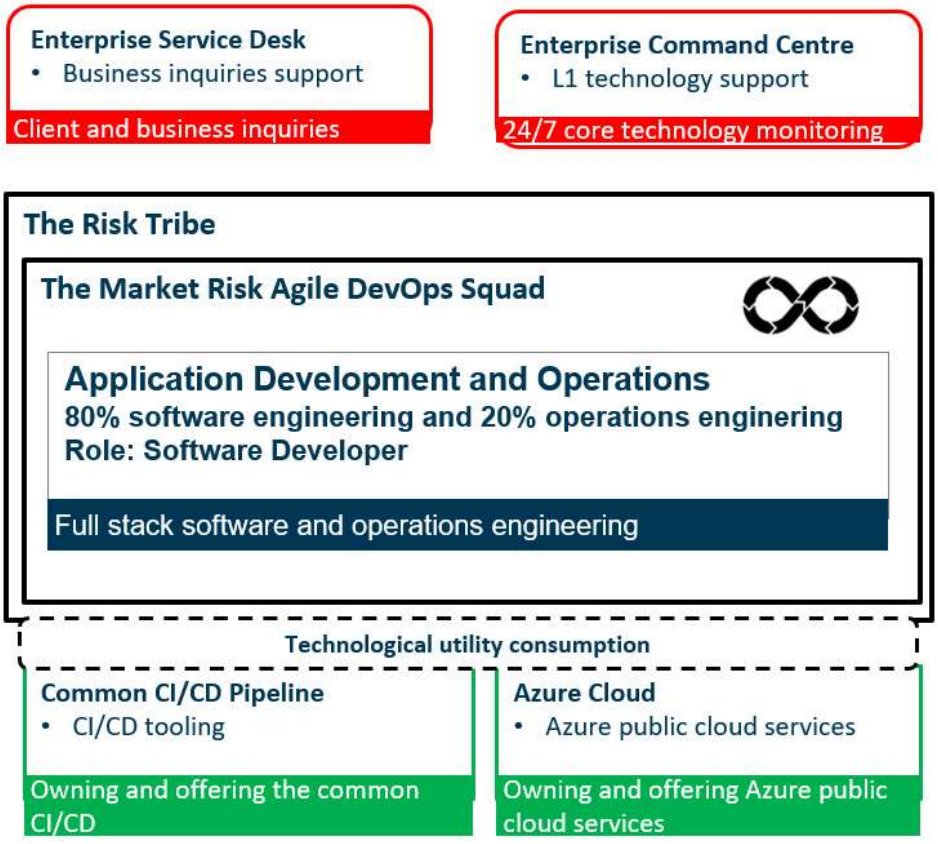


Figure 5.8 – View 2 of the dynamic model

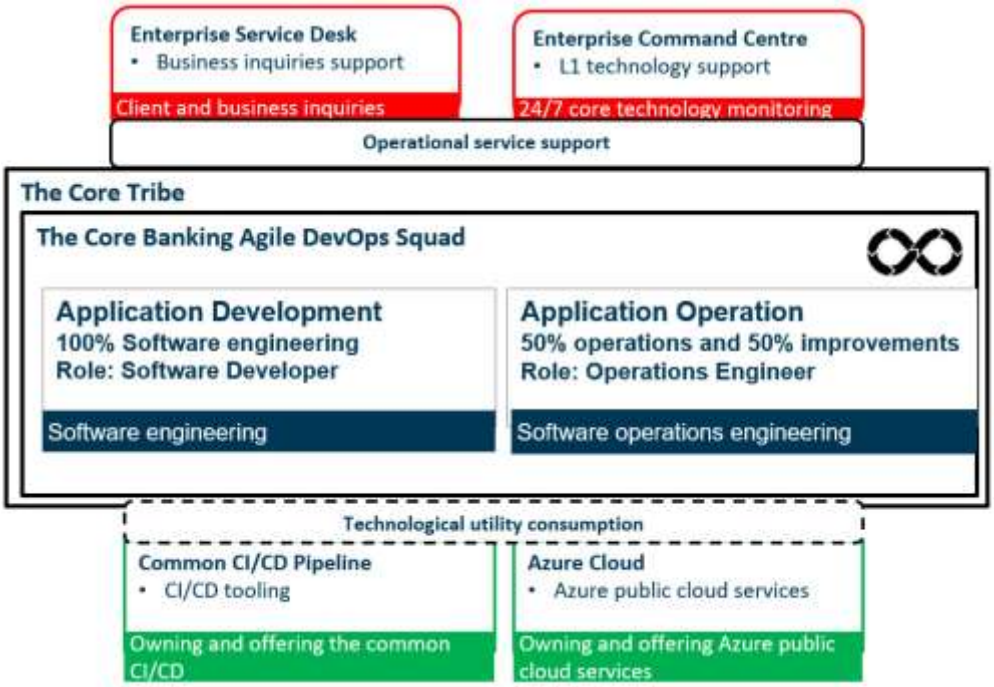


Figure 5.9 – View 2 of the fixed-roles model

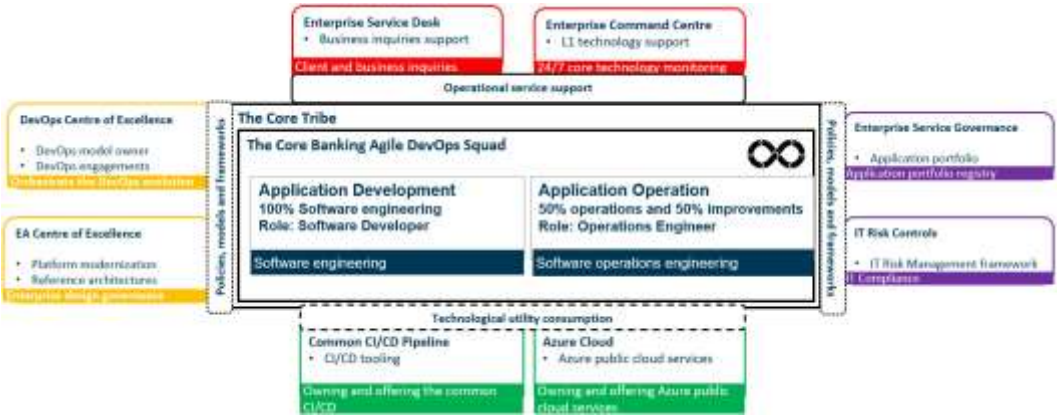


Figure 5.10 – Model expanded with CoE and function team examples

Tables

Role	Access
Developer	Non-production data
DevOps engineer	
Test engineer	
Operations	Production data
Application specialist	
Reliability engineer	

Table 5.1 – Family hierarchy and basic access principles

Chapter 6

Images

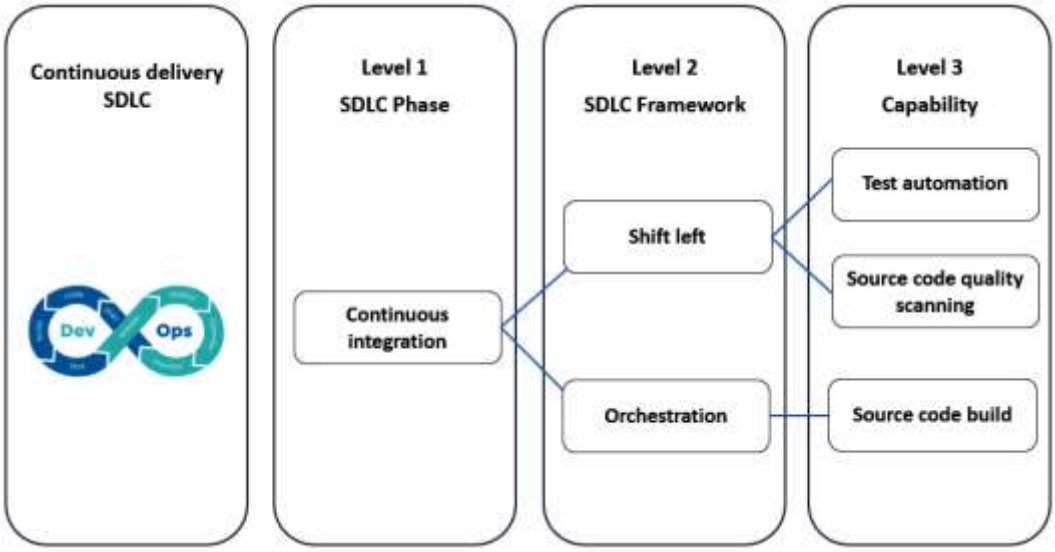


Figure 6.1 – The SDLC levels anatomy example

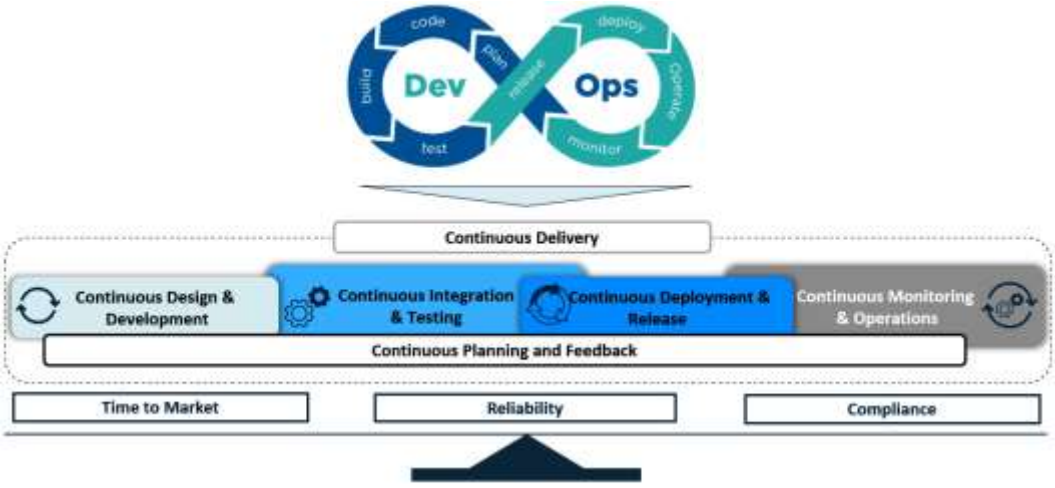


Figure 6.2 – The DevOps SDLC continuity phases and value equilibrium

Framework	Continuous integration orchestration

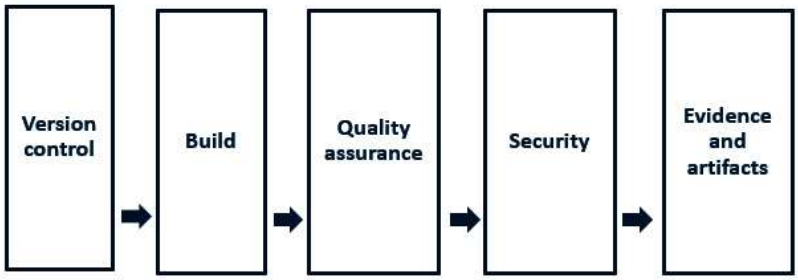
Orchestration steps

1. Code developed and stored in version control
2. Code build triggered
3. Unit and regression tests
4. Static code quality scans
5. Security vulnerability scans
6. Build complete: artifacts and evidence generated

Relationship to other DevOps SDLC frameworks

- **CI/CD provisioning and onboarding –** Continuous development
- **Autonomous IT service management –** Continuous deployment
- **Shift security left –** Continuous design, development, and integration
- **Compliance as code –** Continuous delivery

Flow conceptualization



Value proposition	Orchestrate through engineering practices the continuous integration phase, supporting the Agile DevOps teams to integrate their code in shared repositories and produce the respective artifacts and evidence
DevOps SDLC phases	**Continuous Design & Development** · **Continuous Integration & Testing**

Figure 6.3 – Continuous integration framework anatomy example

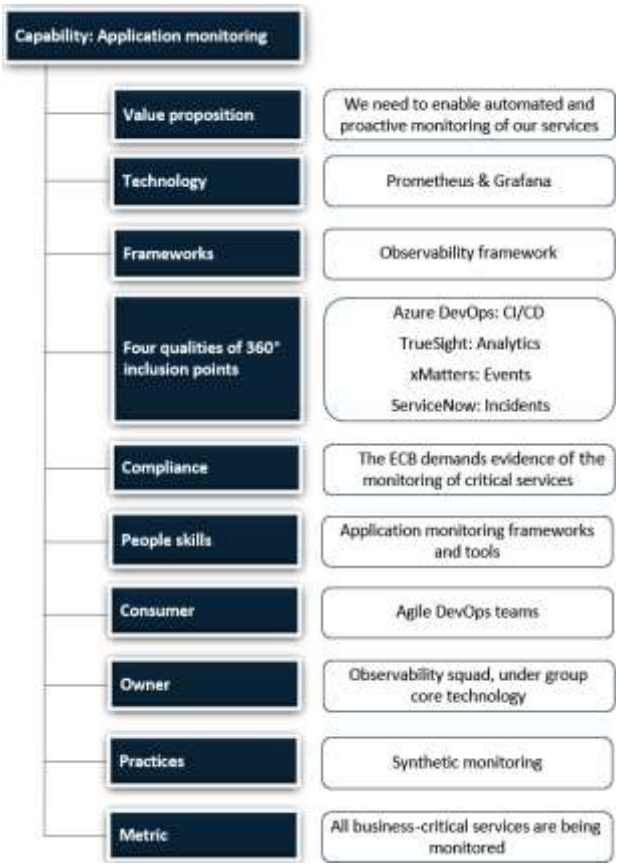


Figure 6.4 – The DevOps complete capability anatomy

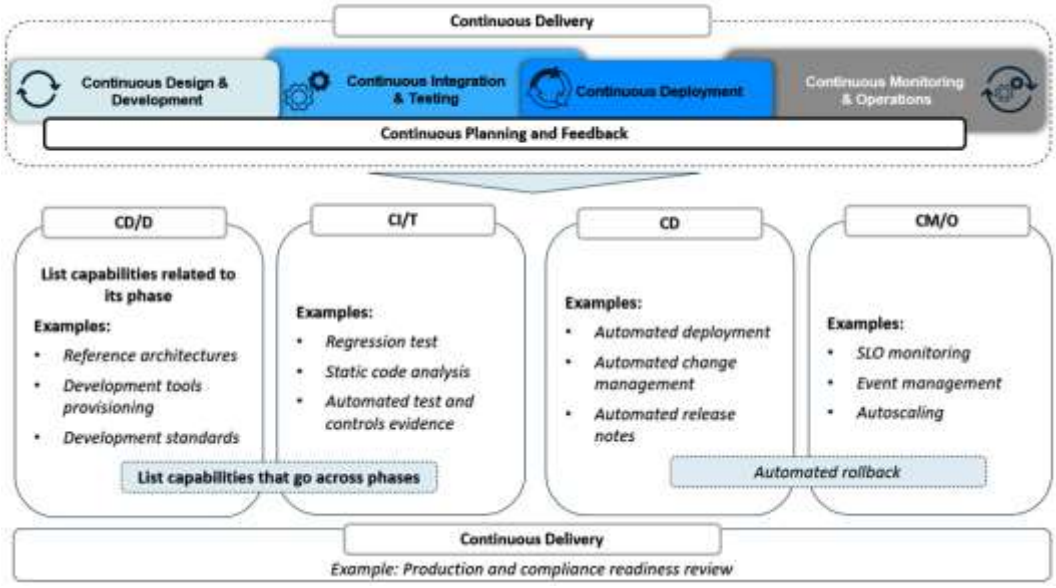


Figure 6.5 – Evolved DevOps SDLC capture template

Figure 6.6 – DevOps SDLC notation

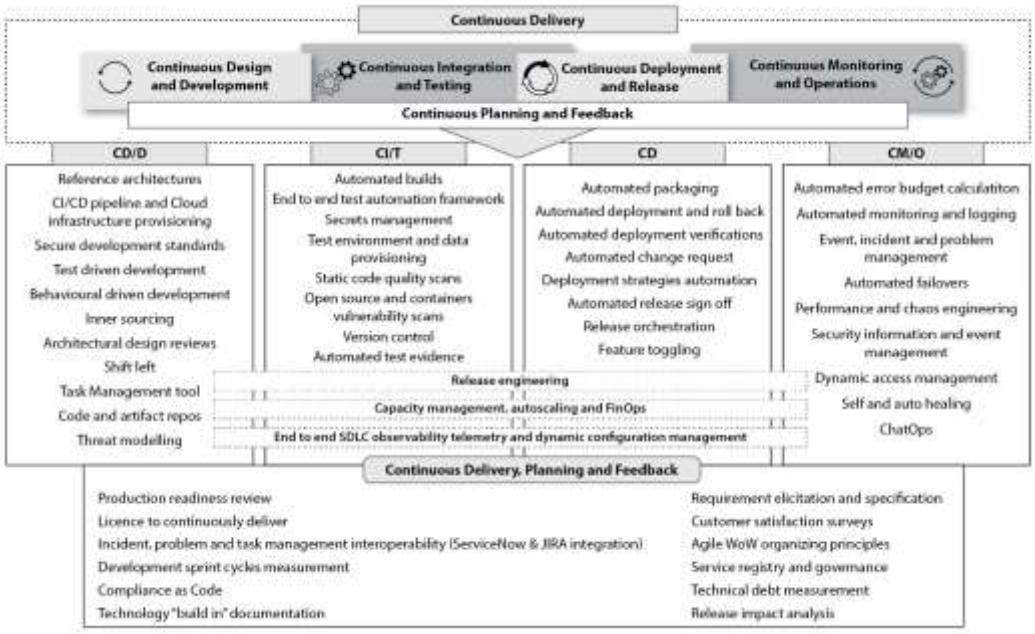


Figure 6.7 – DevOps SDLC catalog sample (indicative and not exhaustive)

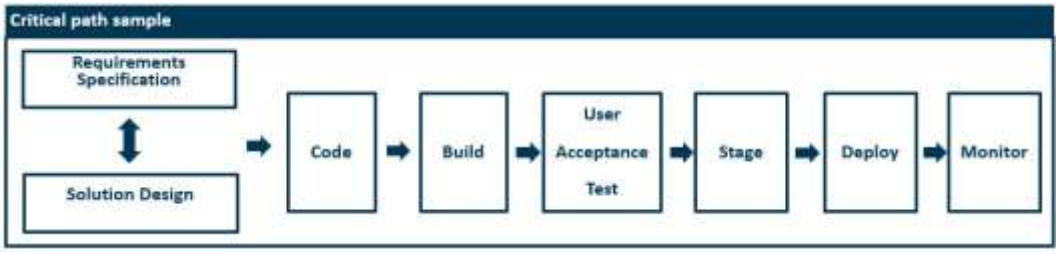


Figure 6.8 – Simple and representative SDLC critical path sample

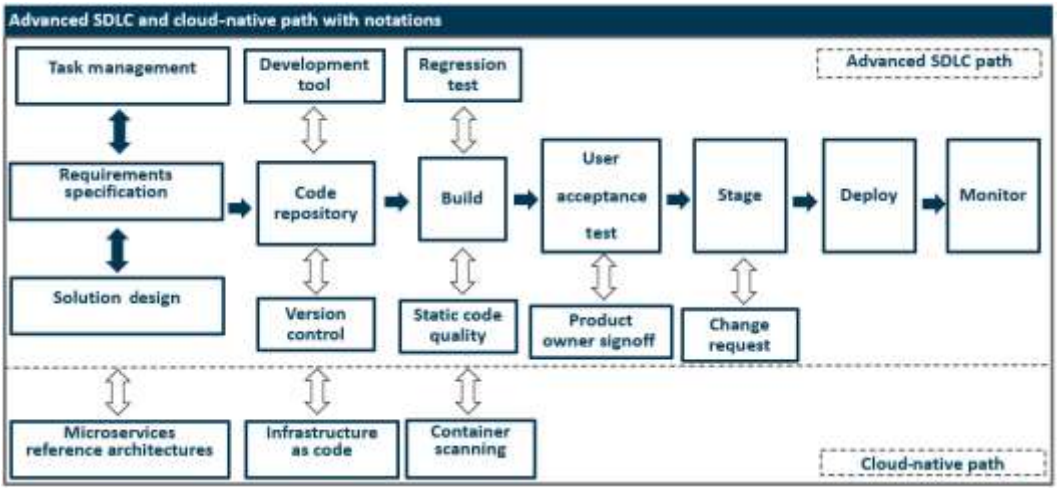


Figure 6.9 – Advanced SLDC and cloud-native path

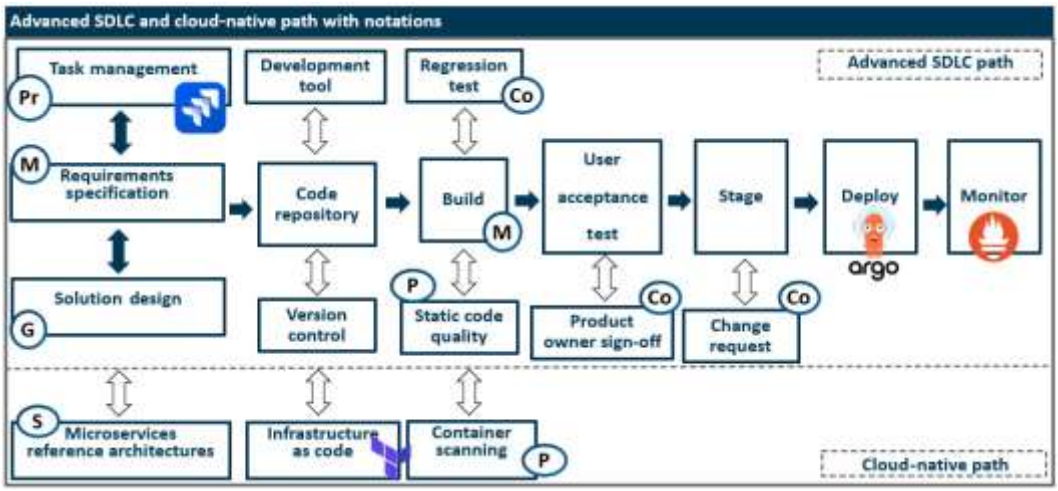


Figure 6.10 – Enriched advanced SDLC and cloud-native path

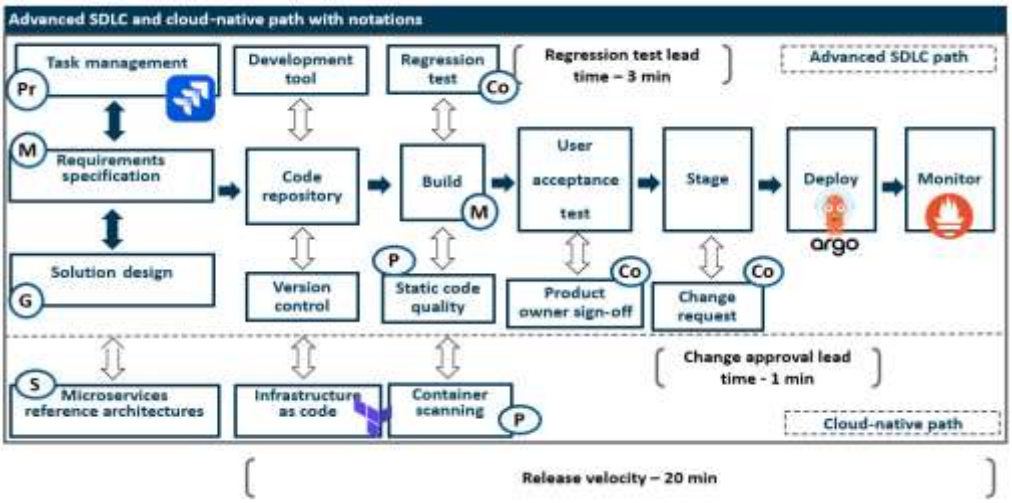


Figure 6.11 – Value stream enriched advanced SLDC and cloud-native path

Tables

Capability/Framework/Standard	Ownership Demarcation
Secure development standards	Cybersecurity
Shift-left standards	Quality assurance CoE
Public cloud infrastructure as a service	Public cloud platform team
Production readiness review	DevOps CoE
Task management	Agile WoW CoE
Reference architectures	Enterprise architecture
Deployment strategies	DevOps CoE

Table 6.1 – Example of DevOps 360° SDLC demarcation

Modernization strategy	Approach to DevOps SDLC evolution
Mainframe to be decommissioned	Out of scope
Mainframe to stay	Focus on the reliability and compliance capabilities
Decoupled legacy to stay	Focus on critical path adoption
Decoupled to modernize	Adopt advanced SDLC on the new yet-to-be-built parts of the portfolio

Table 6.2 – DevOps SDLC adoption tactics for mainframes and decoupled legacy components

Chapter 7

Images

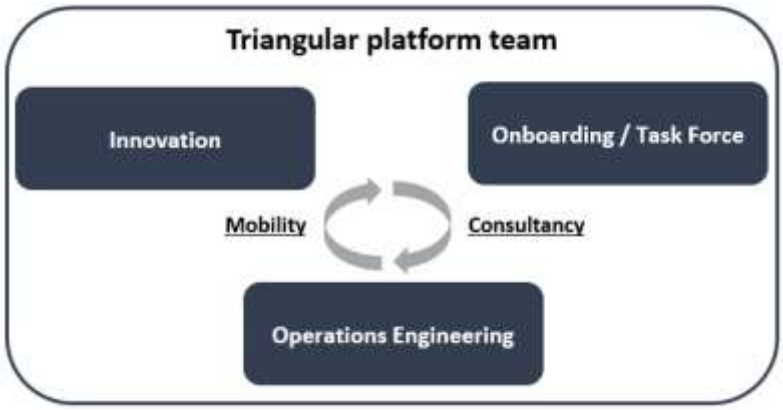


Figure 7.1 – Triangular platform team operating model

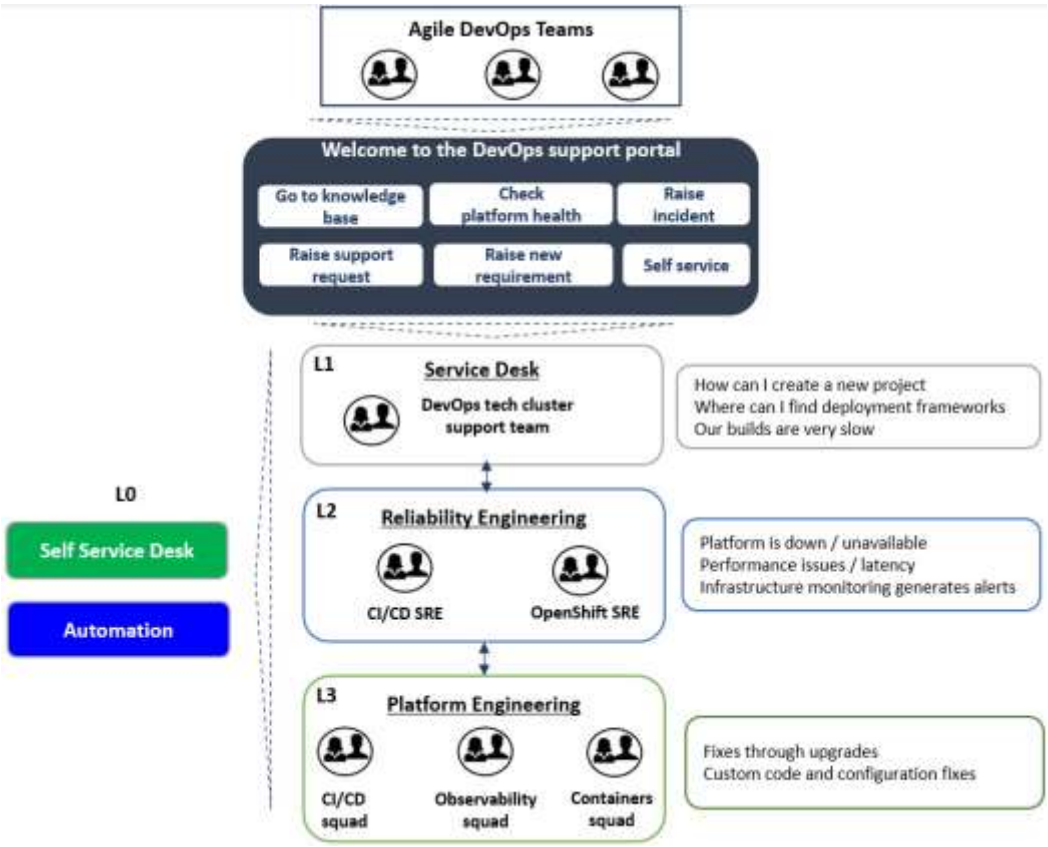


Figure 7.2 – Sample support model for the platform team

Figure 7.3 – Example technology menu

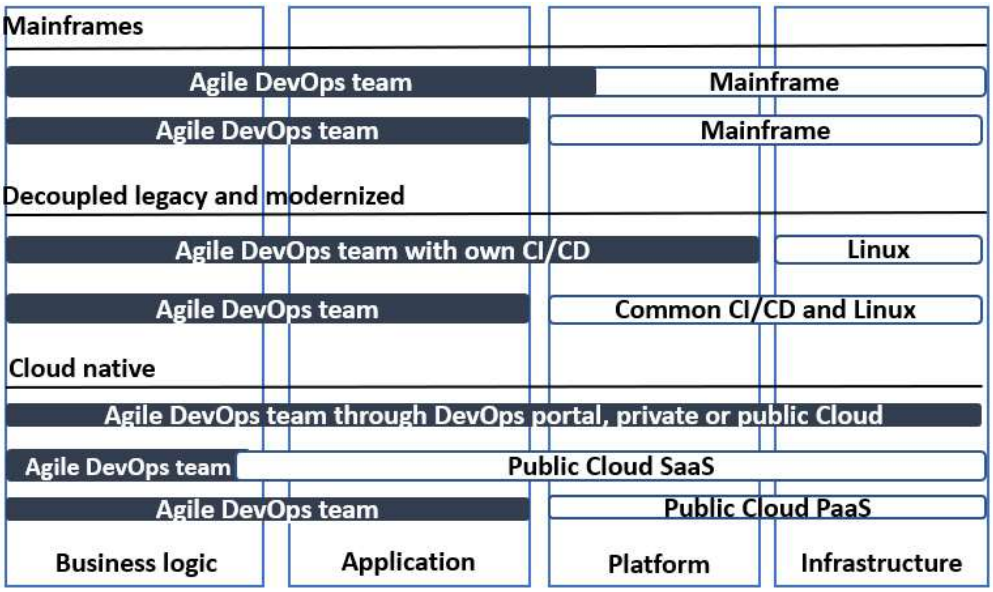


Figure 7.4 – Agile DevOps teams organizing principles illustration based on technology consumption

Tables

Technology product	DevOps 360° SDLC capability engineering enablement	Platform team service
Azure DevOps	Task management	Platform maintenance and onboarding
	Version control	Standards, methodologies, and proven practices
	Code build	CI/CD control adoption
	Code deploy	Ecosystem interoperability
	Test management	Knowledge base
SonarQube	Static code analysis	Central compliance and adoption evidence collection
Artifactory	Build and deploy artifacts' storage	

Table 7.1 – Example of a CI/CD pipeline platform team's product and service catalog

Activity/scenario	Agile DevOps team	CI/CD platform team	DevOps CoE
Tool ownership and offering		R	
Build plans onboarding	R		
Scans and remediation	R		
Control policy establishment			R
Tool operations and maintenance		R	
Integration with standard CI/CD		R	
Knowledge base		R	
Adherence evidence on the team's repository level	R		
Adherence evidence on the tool level		R	

Table 7.2 – Example of technological capability responsibility demarcation for static code quality scans

Domains	Criteria
DevOps fit for purpose	Should fulfill the value propositions of the DevOps 360° SDLC catalog.
Required skills	The required skills are either available in the firm or can be acquired from the market.
Availability and performance	Ultra-available version is available and the performance test is passed.
Security	The security test profiling exercise across capabilities is passed.
Interoperability	Integration points to the rest of the DevOps technological ecosystem are provided.
Maintainability	Is of relatively low effort and new upgrades come frequently.
Adaptability	Can be adapted to different platforms.
Exit	Minimum technology and/or vendor lock-in.
Commercial support	Enterprise commercial version and vendor support are available.
Internal demand capture	Enterprise versus localized.
Customization	Can be configured and scripted.
Data accessibility and visualization	Can have access to data for observability purposes.
Service model	Provides self-service capabilities.
Compliance	Can support the IAM, SoD, data, policy engineering, and IT controls.

Table 7.3 – Non-functional/quality requirements technology assessment sample

Use case	Vendor choice
Collaboration and office tools	Microsoft Azure
Data center offloading	Amazon Web Services
Scaled and rapid calculation engines	Microsoft Azure
Deployment of CI/CD pipelines	Microsoft Azure, Amazon Web Services, and Google Cloud Platform
Big data and machine learning	Google Cloud Platform
Composable banking	Amazon Web Services
Anti-money laundering and know your customer	Google Cloud Platform
Customer resource management and analytics	Amazon Web Services
Greenfield mobile banking	Amazon Web Services and Google Cloud Platform
Payments	Microsoft Azure and Amazon Web Services

Table 7.4 – Public cloud use cases and vendors in the FSI

Chapter 8

Images

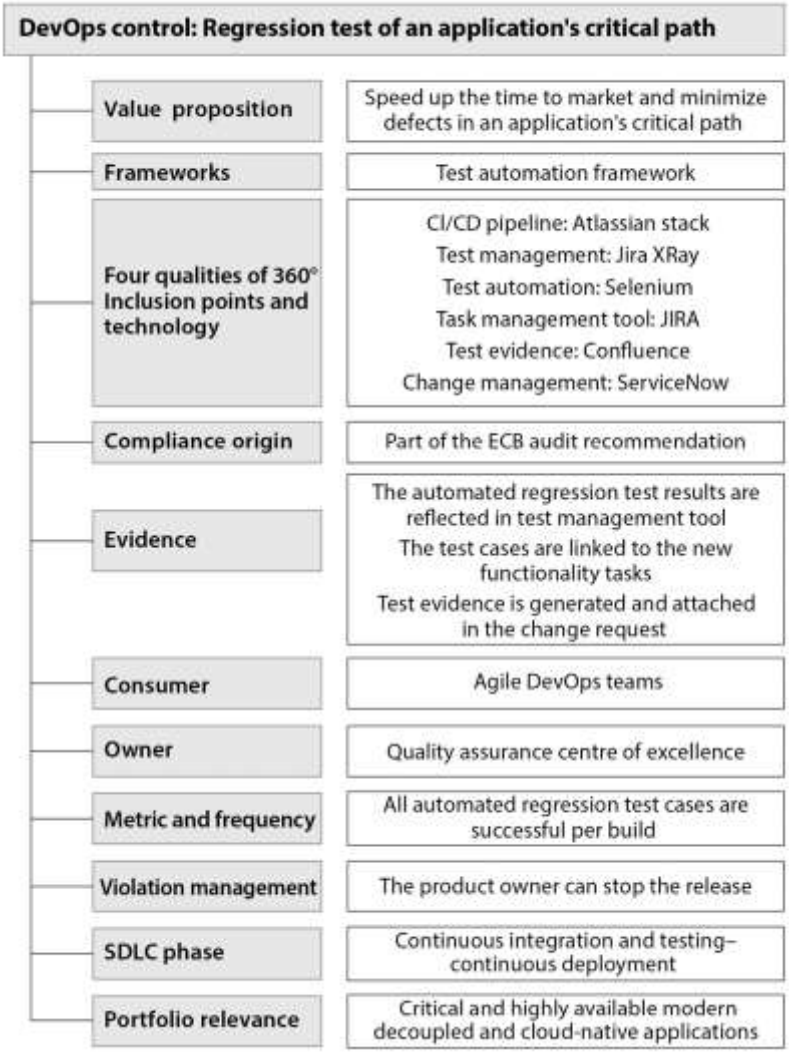


Figure 8.1 – DevOps control anatomy – critical path regression test example

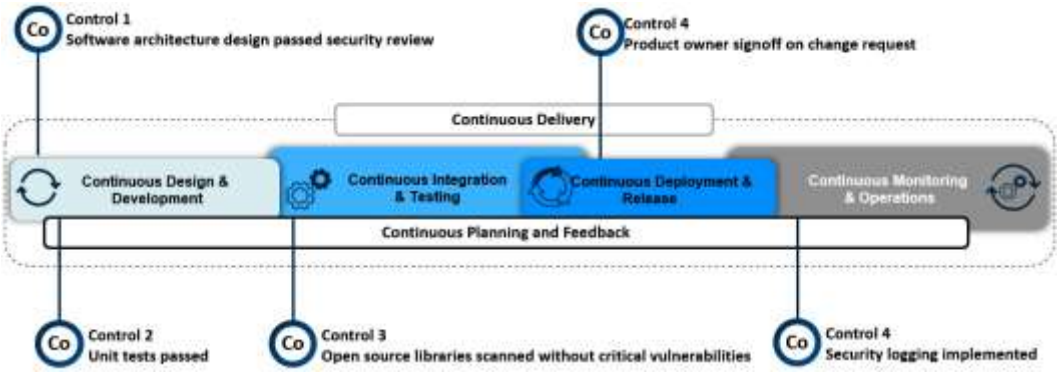


Figure 8.2 – Balancing DevOps controls across the DevOps 360° SDLC phase example

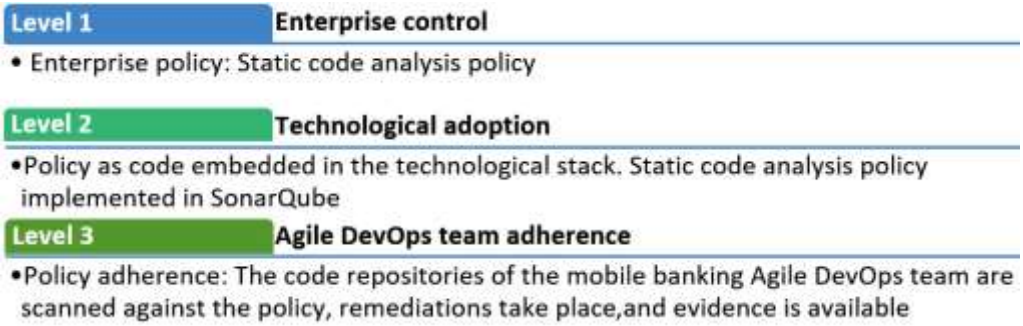


Figure 8.3 – The IT controls enablement, adoption, and evidence tree for static code analysis

Figure 8.4 – The dynamic/rotational model – separation/segregation of tasks

Figure 8.5 – Fixed roles model – separation/segregation of duties

Tables

Role	Data access	Duty performed
Software development	Non-production environments and data	• Can access and develop business logic • Can access and develop operational logic • Cannot deploy to production
Software operations	Non-production and production environments and data	• Can access but not develop business logic • Can access and develop operational logic • Can deploy to production

Table 8.1 – Roles of segregation/separation and basic access principles

Role	Data Access	Duty Performed
Software developer	Non-production environments and data	• Can access (read & write) and develop business logic • Can access (read & write) and develop operational logic • Cannot deploy to production
Rotational developer supporting production	Non-production environments and data Production environments and data, using just in time	• Can access (read & write) and develop business logic • Can access (read & write) and develop operational logic • Can deploy to production

Table 8.2 – Rotational model sample SoD and IAM policy

Role	Data Access	Function Performed
Software developer	Only non-production environments and data	• Can access (read & write) and develop business logic • Can access (read & write) and develop operational logic • Cannot deploy to production
Operations engineering	All environments and production data, using permanent access	• Can access (read) but not develop (write) business logic • Can access (read & write) and develop operational logic • Can deploy to production

Table 8.3 – Fixed-model sample SoD and IAM policy

Chapter 9

Images

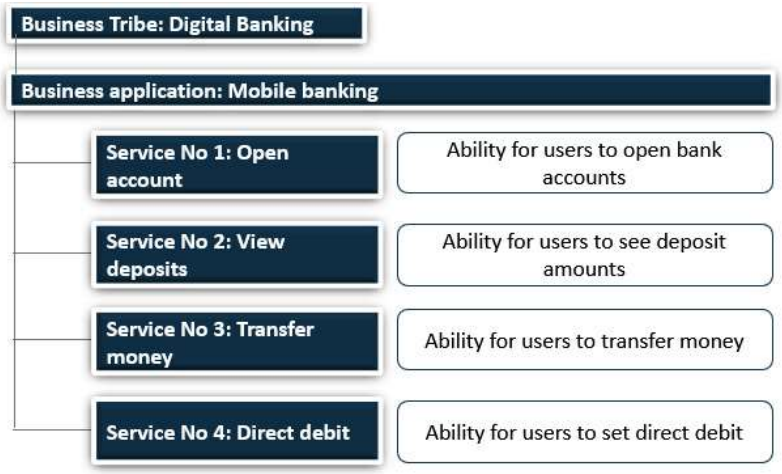


Figure 9.1 – Business application and services for mobile banking example

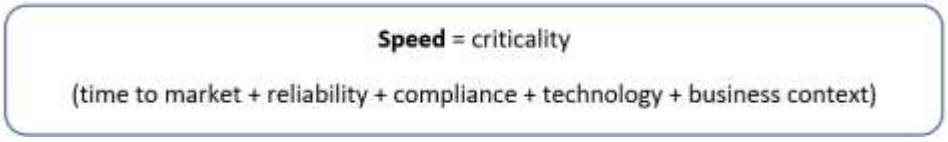


Figure 9.2 – The DevOps speed formula

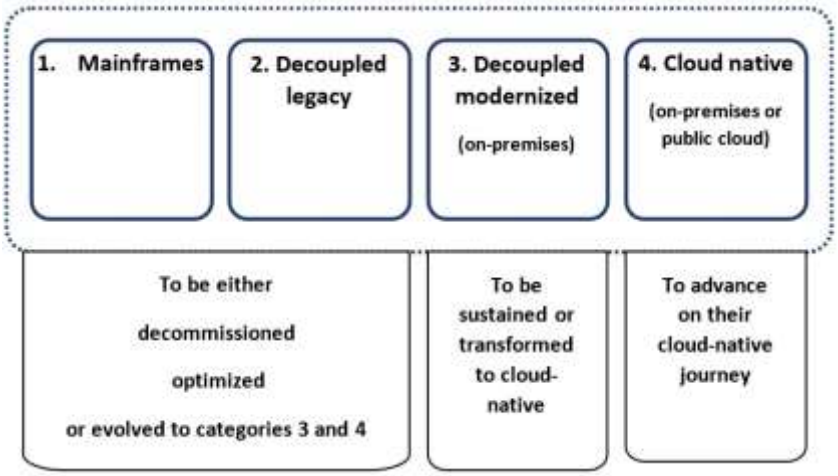


Figure 9.3 – Business portfolio classification based on a technological and architectural foundation

Figure 9.4 – Technological classification portfolio composition evolution sample

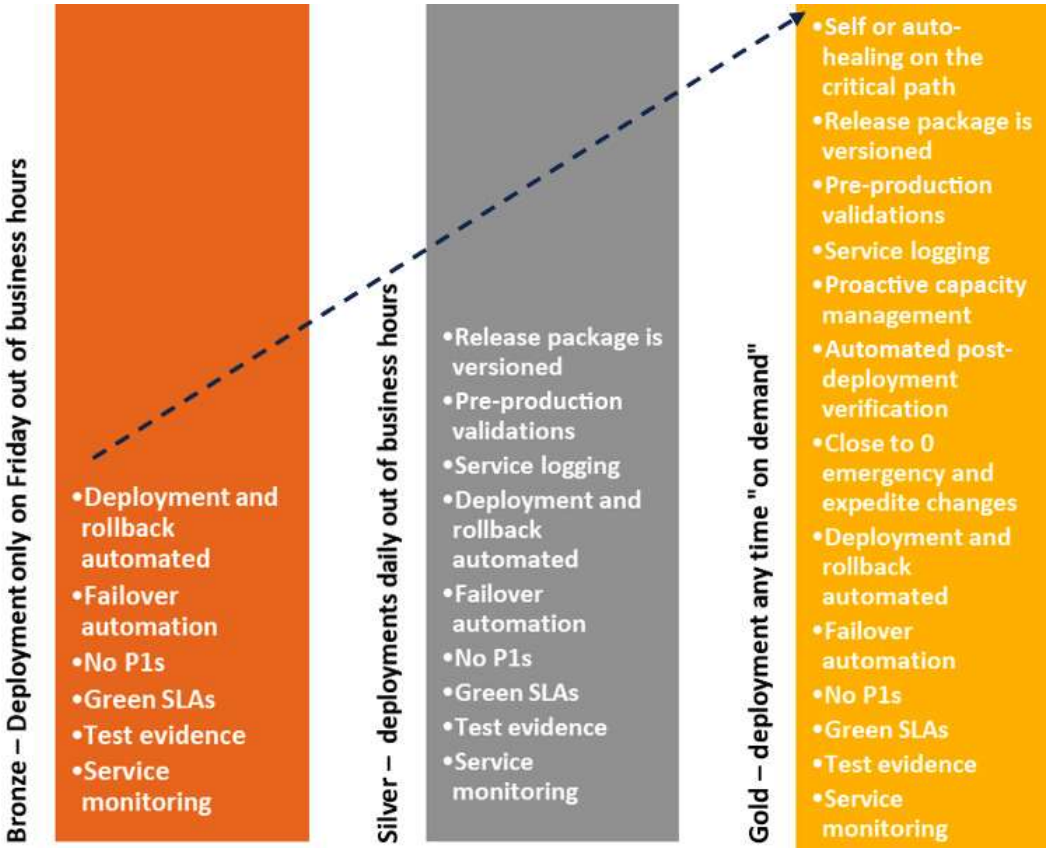


Figure 9.5 – LCD criteria per license type

Figure 9.6 – Level 1 portfolio registry and governance for mobile banking application and services

Figure 9.7 – Level 2 portfolio registry for mobile banking IT assets allocation

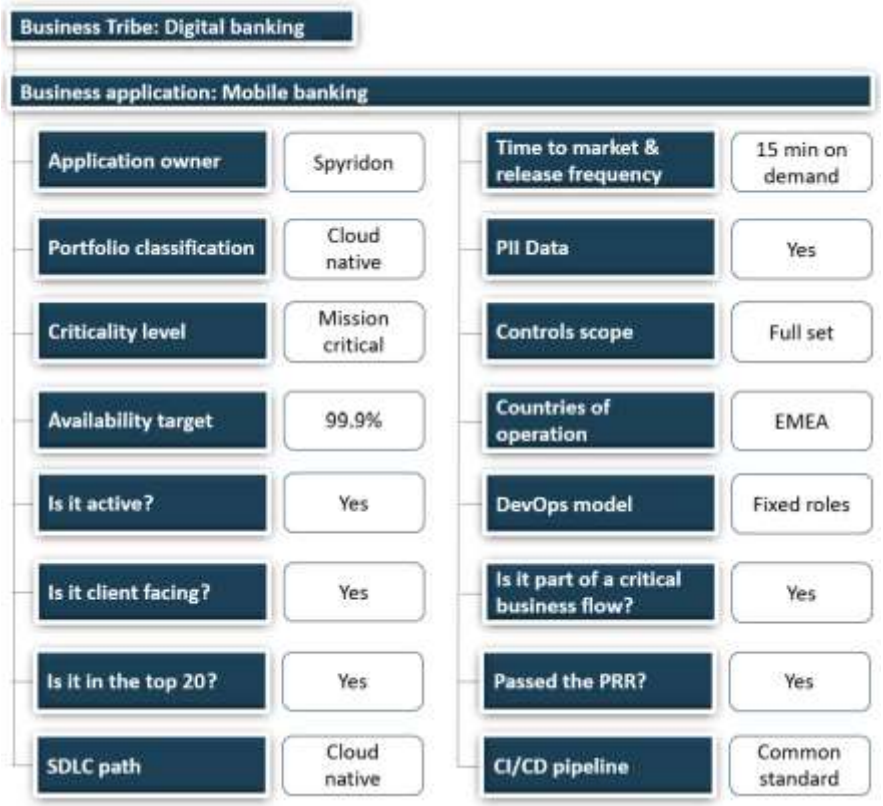


Figure 9.8 – Level 3 portfolio registry for mobile banking DevOps attributes

Tables

Criticality level/characteristics per impact category	Business and client impact (availability and resiliency driven)	Regulatory impact (regulatory compliance driven)	Availability and recoverability targets	Examples of business applications
Mission/system critical	Significant disruption or total termination of vital	Results in license to operate risks and fines.	Availability target: 99.9% **Mean time to restore**	Online and mobile banking Payments

	business functions. Can result in broader economic ecosystem effects.	Adherence to all the relevant regulatory compliance is a requirement.	(**MTTR**) in case of natural disaster: 2 hours Restoration priority 1	Regulatory and liquidity reporting Trading
Business critical	Considerable disruption to business functions. Could disrupt the society and economy.	Can result in fines. Adherence to all the relevant regulatory compliance is a requirement.	Availability target: 98% MTTR in case of natural disaster: 4 hours Restoration priority 2	Lending Market risk reporting Anti-money laundering and know your customer
Highly available	The impact on both the business functions and customers is noticeable but less severe.	Can result in fines. Adherence to certain regulatory compliance is a requirement.	Availability target: 97.5% MTTR in case of natural disaster: 8 hours Restoration priority 3	Safekeeping Invoicing
Standard	No impact on the end users and society and limited impact on internal business units.	Mostly descoped from the regulatory compliance work.	Availability target: 95% MTTR in case of natural disaster: 1 day Restoration priority 4	Business intelligence Data reconciliation Customer resource management

Table 9.1 – Business/client and regulatory impact criticality levels

Criticality level/characteristics per impact category	Business context	Time to market and release cycles	Business and client impact (availability and resiliency driven)	Regulatory impact (regulatory compliance adherence driven)	Technology modernization strategy
Mission/system critical	Real time and client facing	15 min on demand	Availability target: 99.9% MTTR in case of natural disaster: 2 hours Restoration priority 1	Strict adherence to all the relevant regulatory compliance requirements	Cloud-native incremental evolution
Highly available	Batch processing and non-client facing	Monthly	Availability target: 97.5% MTTR in case of natural disaster: 8 hours Restoration priority 3	Partial adherence to the regulatory compliance requirements	Mainframe – sustain with non-frequent enhancements

Table 9.2 – DevOps speeds criticality parameters

	Tactical rollout (3 months)	Organic/scaled rollout
LCD framework ownership	Enterprise portfolio governance function	
License ownership and maintenance	Business application product owner	
Business areas in scope	Market trading and digital banking	All the rest
Assessment tool	LCD portal questionnaire and interviews	
Application assessment method	Self-assessment and interview	Self-assessment and spot interviews
	1. Application request sent and questionnaire filled out. 2. Results available. 3. Evidence evaluation. 4. Final interview and decision made.	Gold: Interview process to finalize license granted. Silver: License granted through self-assessment to all scoring it. Periodical spot checks take place. Bronze: License granted through self-assessment to all applying. Random spot checks take place.
Assessment result calculation	Criticality weighting is assigned to the various questions. The accumulation of points results in one of the three licenses: for example (0 to 40 points – Bronze, 41 to 80 – Silver, 81 and above – Gold).	
Request for a license upgrade revision by applicants	On demand: For the mission- and business-critical applications. Quarterly: For the highly available and standard applications.	
Inability to maintain status consequences	Lower quality observed in production - > Falls one tier. Inaccurate/manipulated input -> Automatically falls to Bronze.	

Table 9.3 – LCD governance sample for inspiration

Core operational aspects

Have SLAs been defined and agreed upon with all the relevant stakeholders?

Has any potential knowledge handover to an operations team or third party been conducted?

Have the access rights and permissions been agreed upon among the relevant stakeholders and implemented in the identity intelligence system?

Has the operating support model been agreed upon with all the stakeholders?

Is the deployment and rollback procedure executed through a CI/CD pipeline?

Security aspects

Has the firm's passwords and secrets policy been implemented?

Are the code base, artifacts, and infrastructure free from critical security vulnerabilities?

Has the application passed the "secure design" assessment as part of its software architectural review?

Data aspects

Does the application contain PII or banking secrecy data? If yes, have the necessary access management and obfuscation rules been applied?

Monitoring and logging aspects

Has application and services monitoring been implemented for the critical services as a minimum requirement?

Has security events logging been implemented?

Quality assurance

Has the application been tested for each build and, ideally, automatically?

Are the automated test cases, artifacts, and data stored and versioned in code repositories?

Is test evidence across test levels available?

If the application is client facing, has a penetration test been performed?

Is static code analysis executed per build?

Reliability and continuity

Has a disaster recovery plan been created and tested successfully?

Have the data backup and restoration procedures been tested?

Have the non-functional requirements been documented and tested?

Table 9.4 – Example PRR questionnaire

Domain	**Question**	**Origin**	**Responsible**	**Accountable**	**Evidence**
Continuity	Has a disaster recovery plan been created and tested successfully?	Regulatory compliance requirement DevOps 360° SDLC requirement	Agile DevOps teams	Product owner	**Test requirements and recovery document** (**TRRD**) Disaster recovery test results

Table 9.5 – PRR criteria fields sample

PRR domain	DevOps evolution capability	DevOps controls domain
Access rights	SoD and IAM model	Access integrity and protection
Test evidence	Test automation	Quality assurance
Monitoring	Observability	Service health visibility

Table 9.6 – Interrelation of PRR domain, DevOps evolution capabilities, and DevOps controls domains

Chapter 10

Images

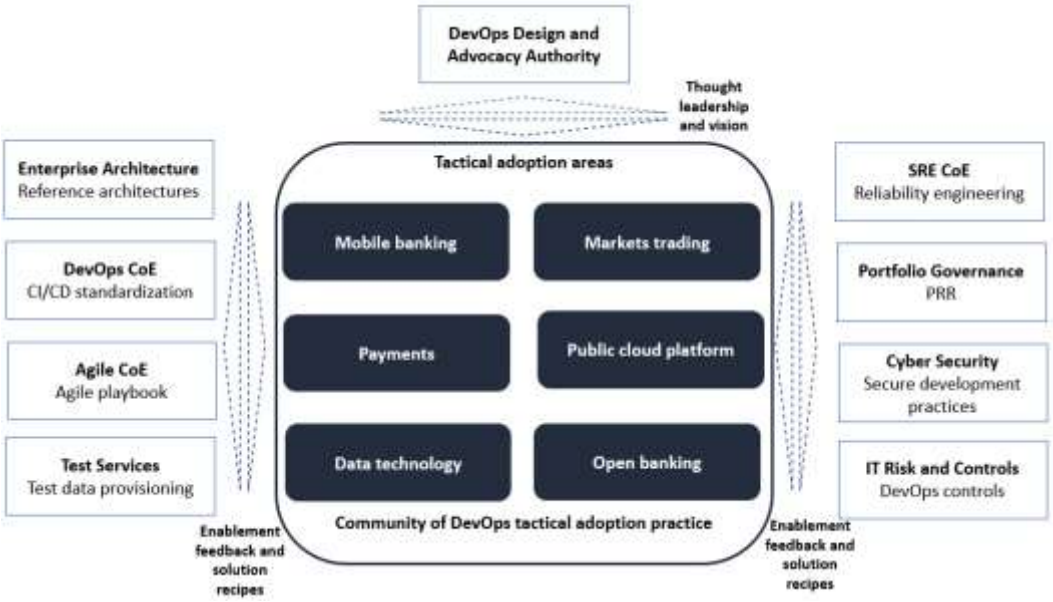


Figure 10.1 – The tactical adoption all-hands-on-deck ecosystem

Figure 10.2 – Adoption breakdown for the example of compliance

Figure 10.3 – Adoption breakdown for the example of DevOps models

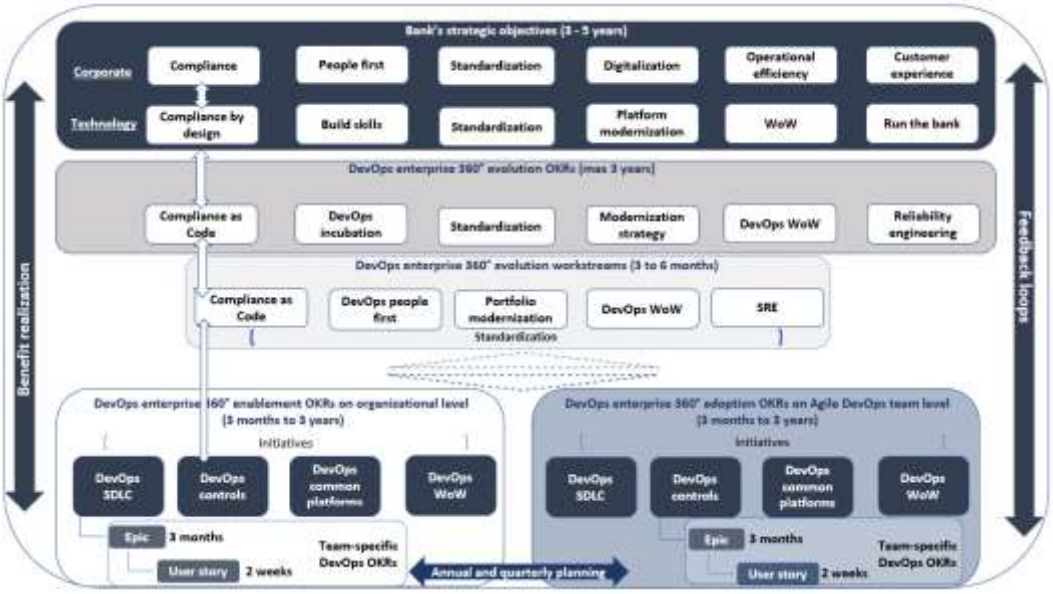


Figure 10.4 – Enterprise portfolio planning mechanism overview

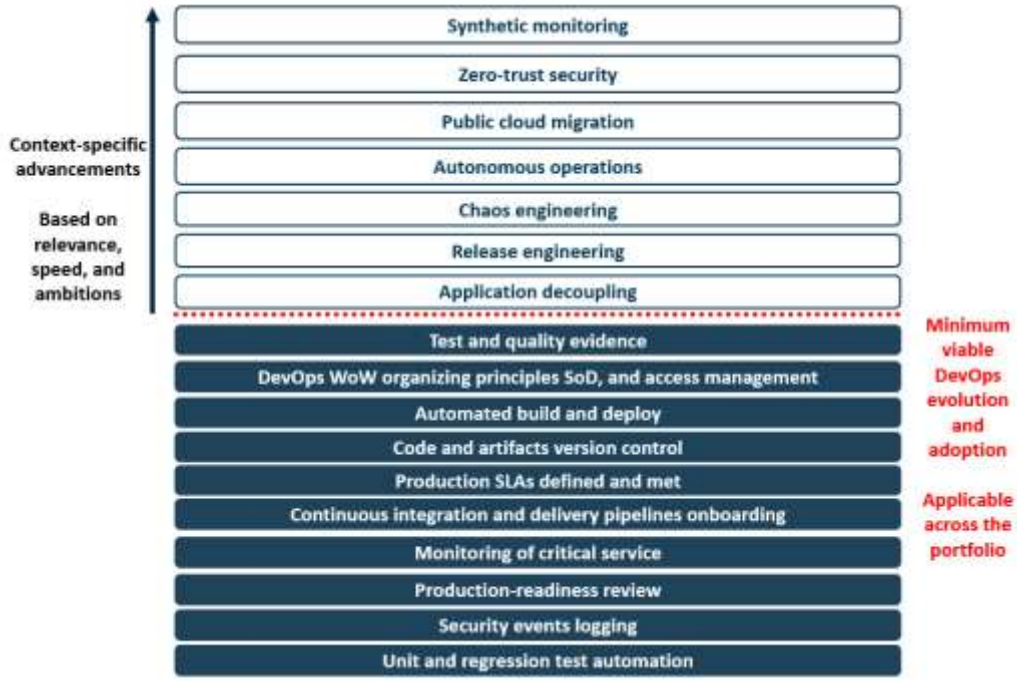


Figure 10.5 – Minimum viable adoption framework representation

Tables

Strategic domain	Examples
High-speed business domains and applications	Mobile and online banking, core banking, trading, asset management, and payments
End-to-end business-critical value streams and flows (either within or across business domains)	Trading life cycle, open banking ecosystem, know your customer, and customer account life cycle
Regulatory compliance ecosystems	FRTB, MiFID, Basel, PSD 2, group risk, capital and liquidity reporting warehouses, and monthly and year-end reporting
Strategic technological capabilities	Standard CI/CD, public cloud platforms, developer self-service portals, and big data engineering platforms
“Burning platforms”	Areas that have severe reliability issues and/or accumulation of unfulfilled regulatory demand

Table 10.1 – Strategic domains and examples of primary candidates for tactical adoption

Differentiator	Degree edge 1	Degree edge 2
DevOps WoW principles	Rotational	Fixed
Availability	99.9%	95%
Time to market	On demand	Weekly
Release size	Incremental	Bulk
Platform	Cloud native	Mainframe
Hosted	Public cloud	On-premises
People skills	Cross-functional advanced	Skills gaps

Client facing	No	Yes
Regulatory impact	High	Low
Future life cycle	To continuously evolve	To incrementally advance
Service nature	Shared service	Business area dedicated
Programming language	Java	HPS
Vendor dependent	No	Yes
CI/CD onboarded	Yes	No
Passed PRR	No	Yes
Architecture	Distributed	Monolithic
Data transmission	Real time	Batch
Compliance gaps	Several	Limited
PII data	Yes	No
Market and advantage	Regional and new	Global and existing
In production	No	Yes

Table 10.2 – Tactical adoption portfolio differentiation parameters

Chapter 11

Images

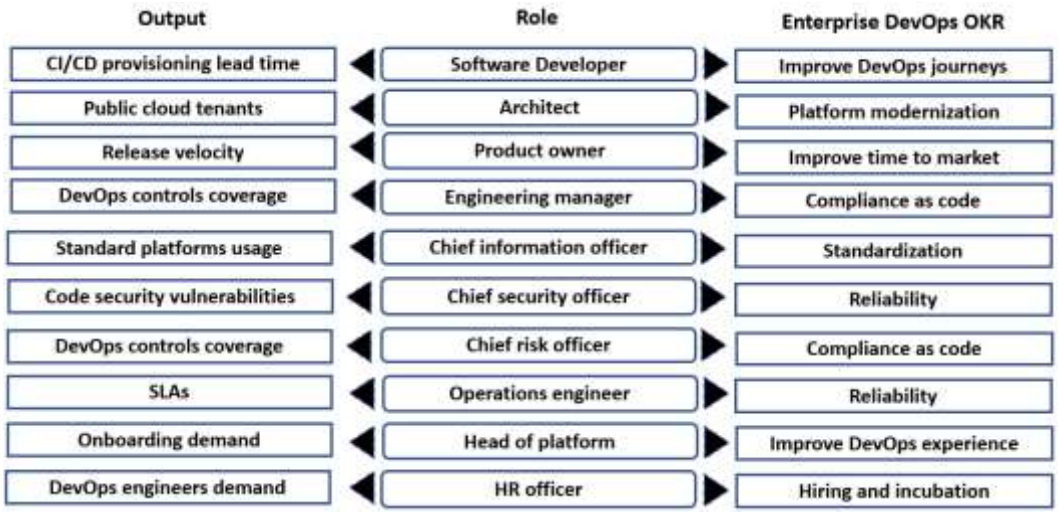


Figure 11.1 – Example desired metrics and linkage to enterprise DevOps OKRs

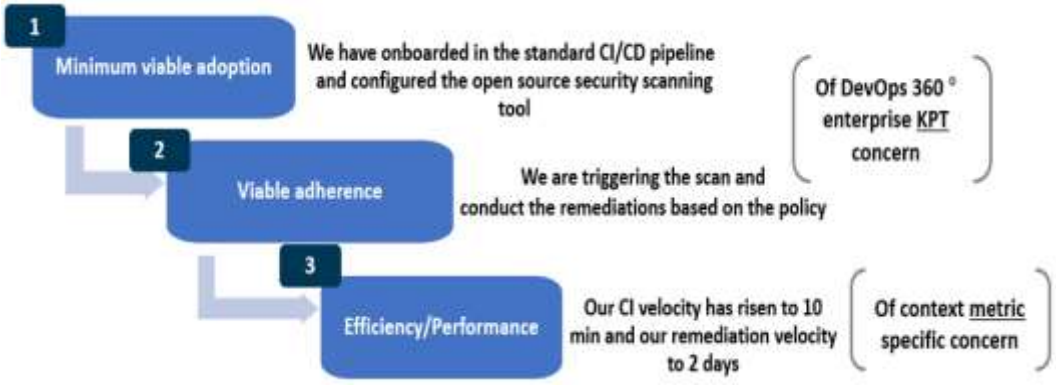


Figure 11.2 – The three-tier approach to KPT and metrics

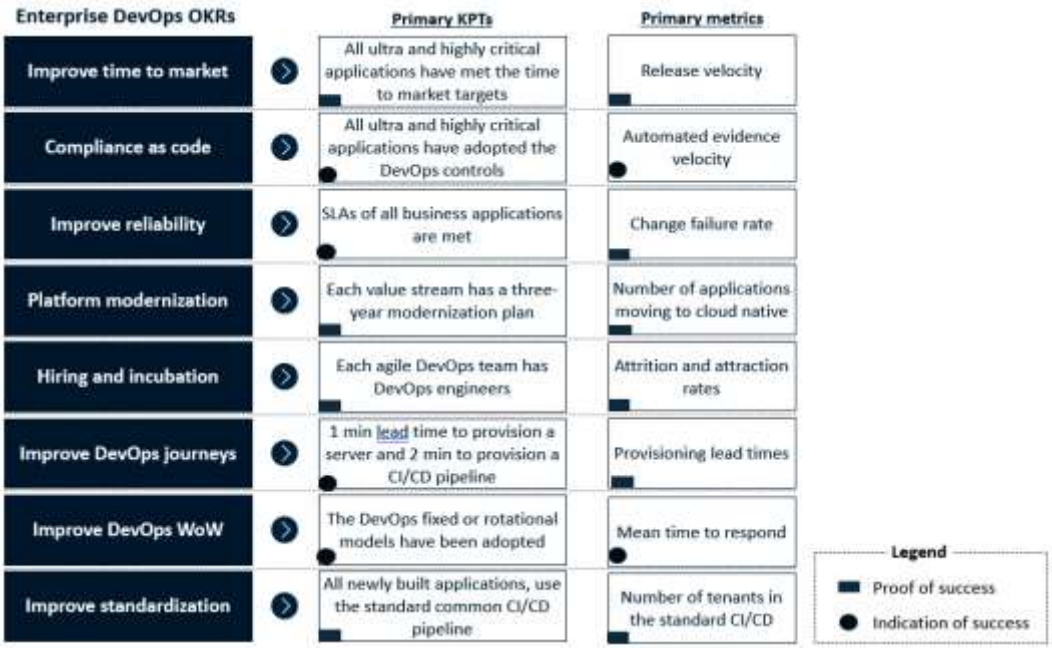


Figure 11.3 – Proposal for the top KPTs and metrics to adopt

Tables

KPT	Metric
Focused on enterprise DevOps OKRs	Focused on team-specific DevOps OKRs
High-level target perspective	Low-level progress perspective
More harmonious across the portfolio	More at relevance per business application
Used for DevOps enterprise evolution decision making	Used for DevOps local evolution decision-making

Table 11.1 – KPT and metric major differences

Chapter 12

Images

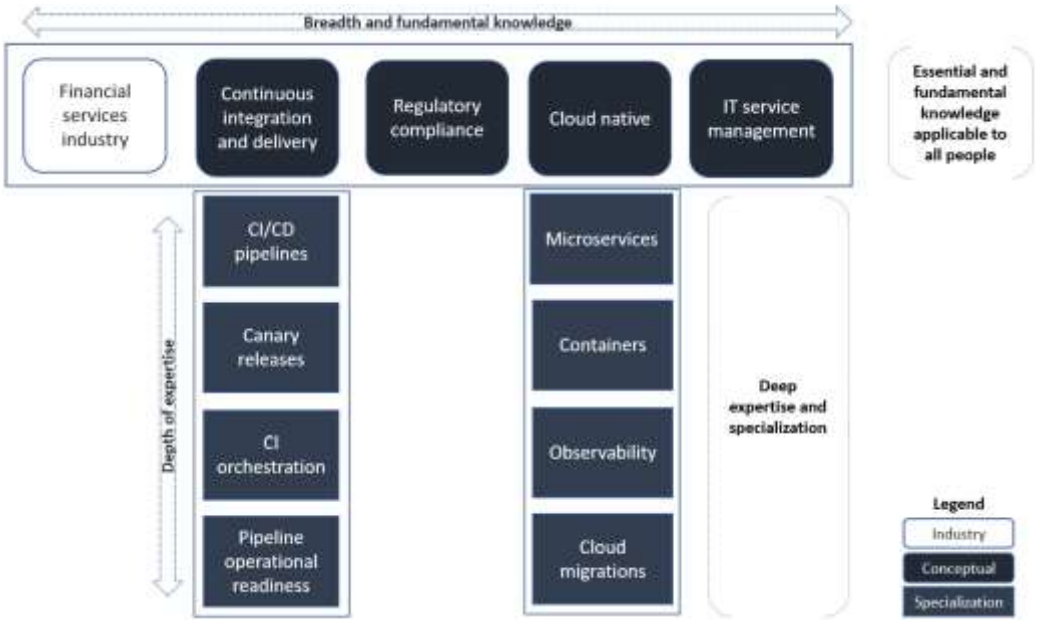


Figure 12.1 – Examples of Π-shaped profiles for a DevOps engineer

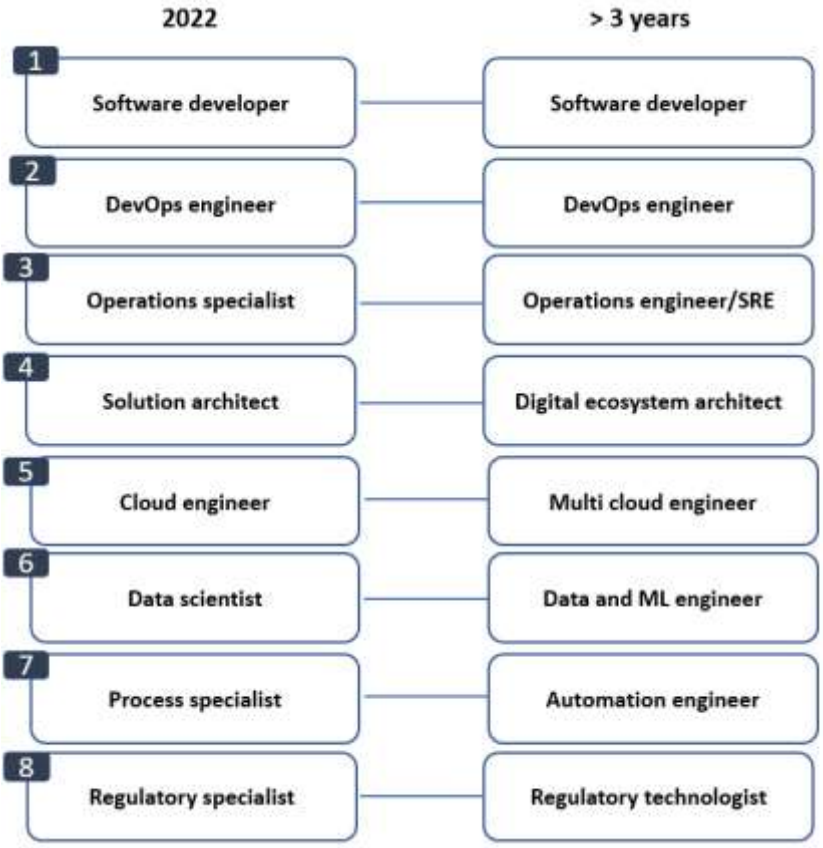


Figure 12.2 – How roles involved in the DevOps evolution can develop

Tables

Shape	I-shaped	T-shaped	Π-shaped
Characteristic	Only specialization	Broad knowledge and a single specialization	Broad knowledge and several specializations
Drawback	Lacks broad knowledge and therefore struggles to	Only one area of specialization and	Too many areas of specialization,

get the big DevOps picture	therefore limited mobility and fatigue	turning into a Swiss army knife

Table 12.1 – The disadvantages of different shapes for DevOps practitioners and professionals

Parameter	Considerations
Location	**Home countries**: Focus on highly experienced, specialized, former chief or senior people, where proximity to business teams is very important. **Nearshoring**: Focus on less experienced, specialized, and people where proximity to business teams is not important. **Offshoring**: Focus on the scale in terms of numbers without the need for high specialization and experience. **Note**: Avoid nearshoring lockup. Circumstances in developing or newly developed countries change rapidly and might force you to exit those locations.
Skill priorities (example)	**Necessity**: Java and .NET developers and DevOps engineers. **Tactical**: Cloud engineers and automation engineers. **On-demand**: Site reliability engineers. **Stop hiring**: Process governance analysts.
Internal domain priorities	You are not only competing in the market with other companies, but also competing internally. You may agree on who has priority internally, even though it might be perceived as discrimination. It is irrational, for instance, to hire DevOps engineers in non-critical domains, when you might have a scarcity of DevOps engineers on the tactical adoption teams.
Areas excluded from a hiring freeze	Hiring freezes are typical in organizations, especially in periods of cost cuts. Some of your areas, such as the DevOps CoE and the tactical adoption domains, need to have a hiring freeze "bypass card."
Mass consulting conversion process	This is when you have hired large numbers of consultants from a certain consulting firm, and eventually, you wish to convert them to full-time employees. You need to agree on which teams get conversion priority.
Dynamic salaries, rates, and trade-offs	Apart from defining what salaries and rates you are willing to pay in certain locations, you will also need to employ location trade-off tactics. Scenarios such as: "a DevOps engineer in location A gives us two DevOps engineers in location B. However, location B has more market competition than location A." You will phase through this repeatedly.

Table 12.2 – An overview of a sample hiring tactic

Chapter 13

Images

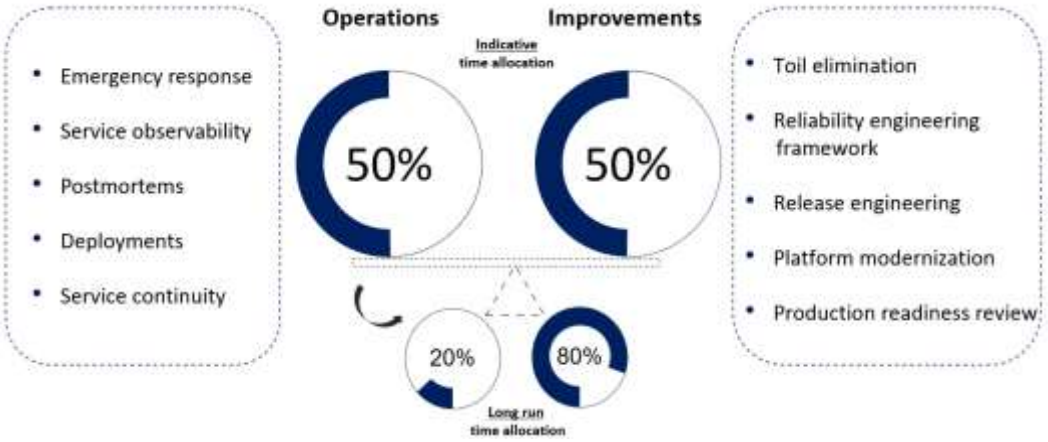


Figure 13.1 – Sample of the SRE tenets balance

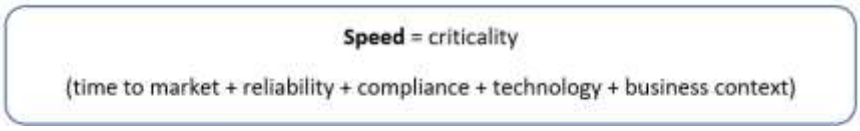


Figure 13.2 – The DevOps speed formula

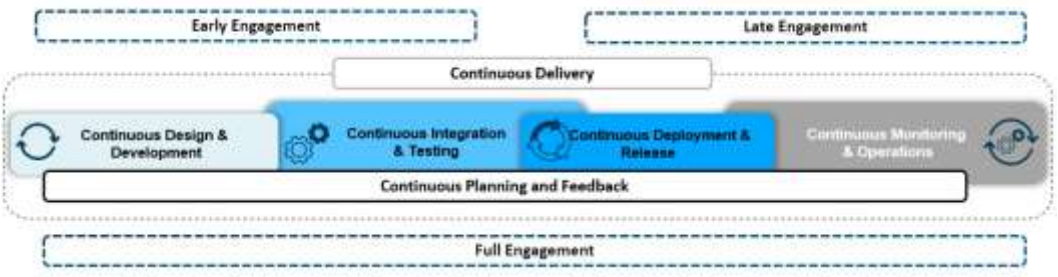


Figure 13.3 – SRE engagement models visualized in the DevOps 360° SDLC

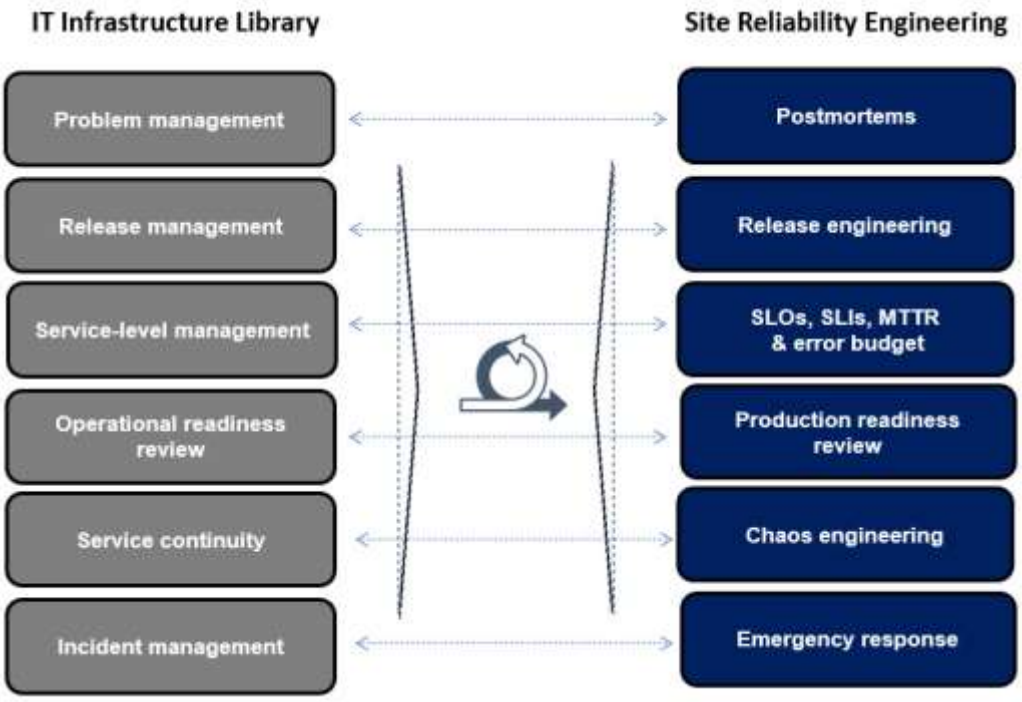


Figure 13.4 – Example of SRE and ITIL reconciliation

Figure 13.5 – Overview of the SRE roles on the technological stack

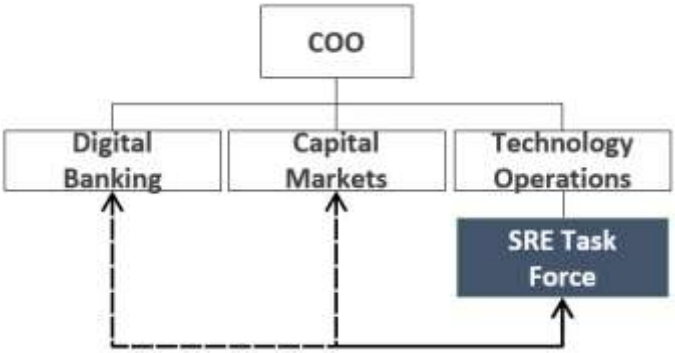


Figure 13.6 – The SRE task force topology

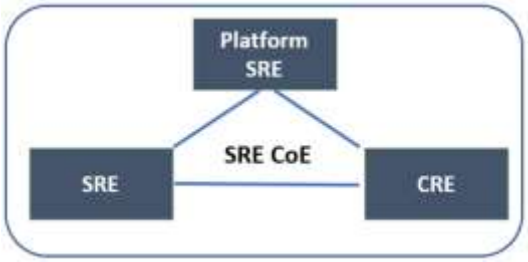


Figure 13.7 – The three parts of the triangular SRE CoE

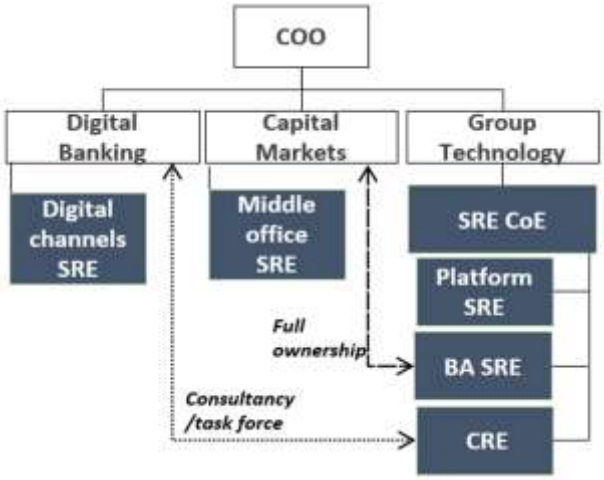


Figure 13.8 – The triangular SRE CoE and tactical topology

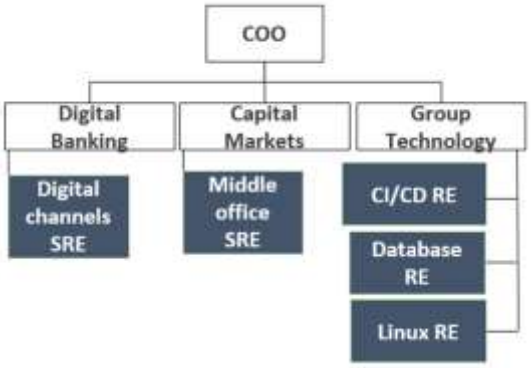


Figure 13.9 – Business application and platform SRE topology

Tables

DevOps	SRE
Commonalities	
Embracing change and risk	
Breaking the organizational silos	
Automation	
Technology utilization	
Visibility and visualization	
Release velocity and engineering	
Gradual service changes	
Fail fast and learn	
Shift operations and quality left in the SDLC	
Specific Focus	
Production operations are part of it	Production operations are at the center of it
The focus is on releasing “fast”	The focus is on releasing “reliably”

Focus on functional requirements	Focus on non-functional requirements

Table 13.1 – DevOps and SRE mainstream commonalities and specific focus

Reliability engineering framework elements
Automated monitoring of error budgets
Backup and restoration procedures
Reliability assessment on system design
Chaos engineering recipes
Microservice observability framework
IT service management automation
Deployment plans and strategies
Technological solutions' interoperability
Automated failover and restoration procedures
Event-triggered evidence generation
Launch coordination engineering practices

Table 13.2 – Example of elements in a reliability engineering framework

	Early engagement	Late engagement	Full engagement	Advisory
SDLC phase of the service	Ideation or early design	Either first "go live" or already in production	Ideation or early design to sunsetting	Any SDLC phase
Objective	Establish the fundamental practices of reliability engineering from day 0	Assess production readiness Support with the gradual deployment of	Ensure service reliability across its life cycle	Provide consultation on adopting the reliability engineering framework

		the service in production Address severe instability Operate the service in production		
Ownership level	Limited: Set up the fundamentals and move on to another service	Limited: Release or resolve the reliability issues and move on to another service	Full ownership of the service across its life cycle	No ownership

Table 13.3 – SRE engagement model parameters

SRE operating model ownership	SRE task force
Engagement model	Late – solve reliability issues and move on Rotational
SRE service ownership	None
Level of centralization	Highly centralized

Table 13.4 – The SRE task force operating model parameters overview

Pros	Cons
"Baptism" was avoided. The organization knew that those who were called site reliability engineers had the necessary qualifications.	The SRE task force, after a certain number of engagements, could not scale anymore.
The SRE people were not misused and were placed tactically in	Exiting certain engagements became challenging due to the absence of people to hand knowledge over.

certain areas, delivering decisive work.

The rotations across various areas provided extensive organizational awareness to the SRE task force team.	The rest of the organization, due to the absence of hands-on support, struggled to implement more than the basics of the reliability framework.
There was a single area owning the “reliability framework,” which was enriched through task force engagements.	The engagement rotations had a mixed effect on the site reliability engineers. Switching context and needing to constantly be onboarded to new setups create frustration.
Internal career mobility for the site reliability engineers was promoted through the rotations.	In certain engagements, site reliability engineers ended up performing more than 50% of the operations work, which resulted in frustration, especially in periods of significant “on-call” involvement due to constant firefighting.
A certain level of standardization of reliability practices was achieved in the task force engagements.	In several cases, it was challenging for the SRE team to influence the agile DevOps team’s backlog on prioritizing reliability matters over new functionality.
	Onboarding of the site reliability engineers in terms of domains and technological stack knowledge as well as access rights management prolonged the start of the engagements.
	A lack of certain technical skills of the site reliability engineers did not make them a good fit for certain engagements.
	With all SREs in engagements, it was challenging to focus on keeping the reliability model updated.
	Several disagreements over the responsibility “white paper” arose, for instance, who was

performing the deployments or who was responsible for L1 (emergency response).

Table 13.5 – SRE task force operating model pros and cons

SRE operating model ownership	SRE CoE and tactical SRE areas
Engagement model	Early, late, and full rotational and fixed
Service ownership	Full ownership of certain parts
Level of centralization	Hybrid

Table 13.6 – The "triangular" SRE CoEoperating model and tactical hiring parameters

Pros	Cons
The SRE CoE through dedicated focus delivered decisive work.	Due to its portfolio versatility and hybrid setup, leading the CoE was perceived as "empire building" by certain parts of the organization. That generated cooperation resistance in certain areas.
The visibility across various areas provided extensive organizational awareness to the SRE CoE team.	Running the daily operations of the CoE and aligning the three teams over priorities and communication was quite a complex task.
There was a single area owning the "reliability framework," which was kept updated. Scaled reliability engineering recipes were also added.	At a certain point, the CoE struggled with capacity due to parallel runs (old and modernized setups).
Internal career mobility for the SREs was promoted through the rotations across the three teams.	Only a few of the CoE people had a software development background and it was challenging to improve reliability through business logic changes.
There was a very high level of standardization of reliability	In several cases, it was challenging for the SRE team to influence the agile DevOps team's

engineering across business applications and platforms.	backlog on prioritizing reliability matters over new functionality.
Due to the magnitude of decisive deliveries, a “high desirability” SRE momentum was created.	In time, there were conflicts of interest between the platform SRE and business application SRE teams on the platform’s priorities and deliveries.
The platform modernization of the SRE CoE-engaged areas moved fast, eliminating dependencies on legacy platforms and applications.	Operational confusion rose among the business application site reliability engineers as, functionality-wise, they were referring to the agile product team PO and, HR-wise, to the SRE CoE team leads.
The SRE-owned platforms and tools were ultra-reliable; they were piloted in flagship business applications and direct feedback was captured.	It took time to assemble and make effective the CoE team as it was a combination of external hiring, internal incubation, and moves.
The tactical areas that did not have direct SRE involvement could advance through their own hiring and the CRE interaction.	

Table 13.7 – Triangular SRE CoE operating model pros and cons

Pros	Cons
The platform reliability engineering teams on the platforms made decisive reliability improvements that hugely benefited the broader organization.	Due to a lack of funding and coordination, the site reliability engineers hired in the business applications teams could not do decisive reliability work.
	Incubation through “baptism” was also observed.

Mobility, sharing of reliability engineering practices, and standardization were performed in the platform teams.	There was a significant variation in the SRE operating model and practices among the business application's site reliability engineers but also among the platform ones.
The cloud-first central capabilities' delivery was accelerated.	A lack of a single source of reliability engineering practices for business applications.
The platform onboarding and service support procedures were improved dramatically and the agile DevOps teams were brought closer to influencing the platform's backlogs.	
It was the first time that the platform teams conducted an enterprise capacity planning exercise.	

Table 13.8 – Business application and platform SRE operating model pros and cons

SRE operating model ownership	Non-existent
Engagement model	Each team could specify its own
Service ownership	Hybrid in either the business applications or platform areas
Level of centralization	Decentralized

Table 13.9 – The "baptized" and "random" SRE parameters