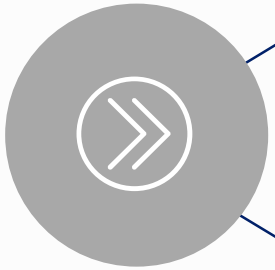
Beyond the expected.®

PEKIN®

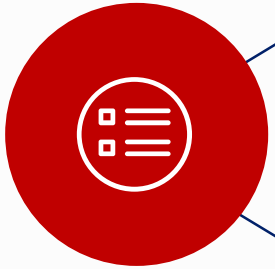
INSURANCE

DevOps Strategy

Executive Summary



Drivers for transformation: Rise of the digitally aware customers demanding anywhere/ anytime access to services and information and a higher pace of innovation in the industry that cannot be supported by traditional technology



Our Recommendation: A fundamental shift in technology teams, systems and processes for improved transparency, higher automation, seamless integration and continuous delivery through adoption of Agile, DevOps and Cloud technologies

Impact to Pekin: Technology that consistently delivers high quality value driven products and services for internal and external customers with speed and efficiency

Current State Assessment and Opportunities for Pekin

A current state assessment on Pekin's people, process and tools was conducted to baseline, understand and document key capabilities that will drive and shape the organization's DevOps enablement

Key Findings

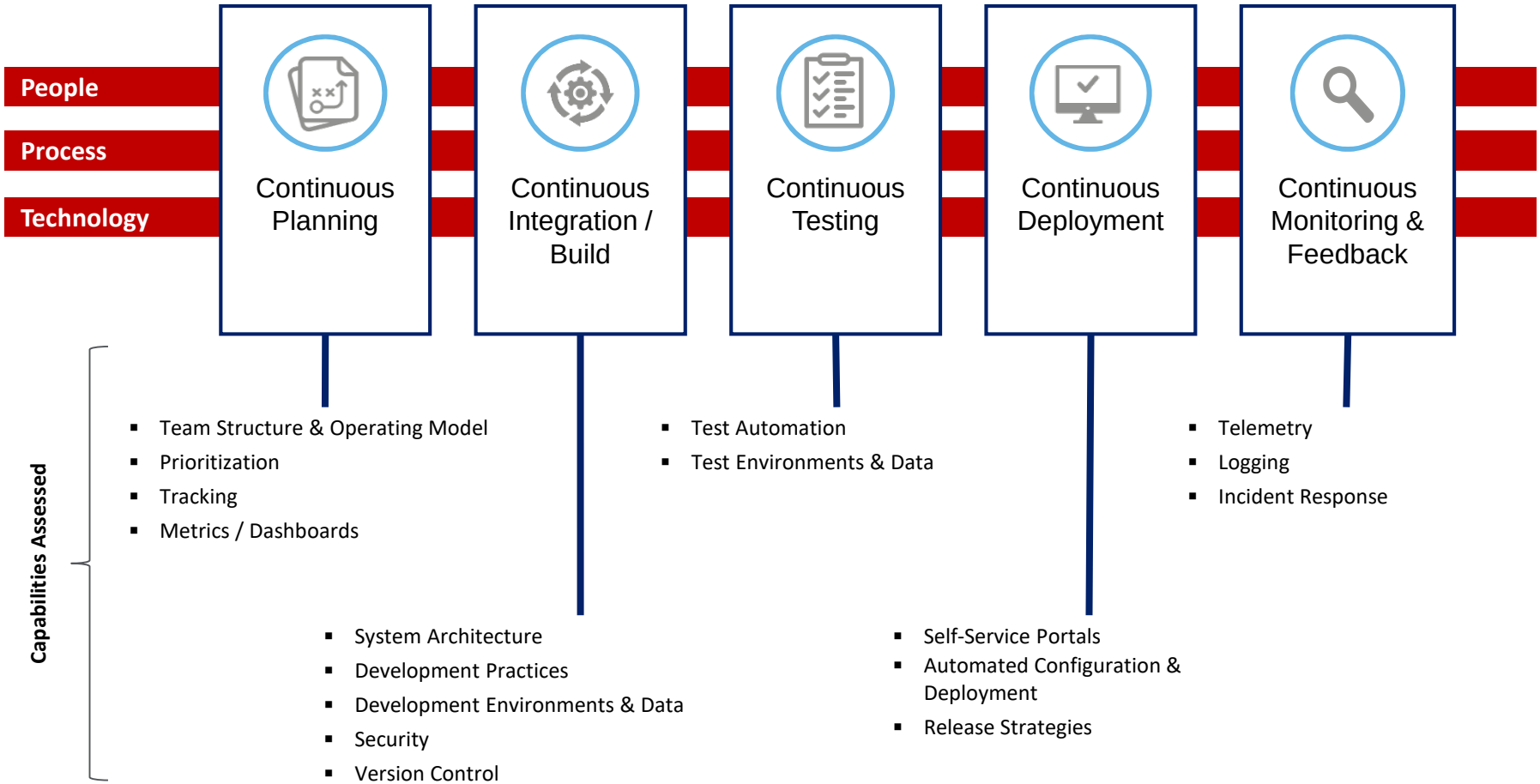
- Current processes followed are highly manual leading to low productivity and lack of transparency
- Agile methodology has been adopted by some teams but is followed loosely
- Automation has been implemented for limited SDLC activities (e.g., build, deployment, testing)
- Multiple tools are used across the enterprise to achieve similar work goals
- Resources are not enabled with the right knowledge and training on the new processes and tools
- Existing monolithic architecture leads to various delivery challenges
- Low focus on quality of product development leads to technical debts
- Limited metrics to guide teams to improve productivity during development and deployment
- Several initiatives are planned and ongoing to improve the organizational process (e.g., test data strategy, agile adoption)

Recommendations

- Adopt agile methodologies across the organization and establish an agile CoE to provide governance, standards and encourage adoption of agile by teams. Build an operating model to support agile delivery
- Define and track organizational metrics across the software development lifecycle
- Expand DevOps capabilities beyond continuous integration and deployment CI/ CD and set up end to end continuous delivery processes
- Identify and procure cloud suitable DevOps tools across the software development life cycle with a focus on complete automation and built in quality
- Design new applications and refactor old monolithic application to move to a componentized architecture
- Equip and coach business and technology resources to work in a more collaborative manner and reorient teams to own product delivery and quality

DevOps Capabilities

The Deloitte DevOps Maturity assessment evaluates an organization across 5 capability domains:



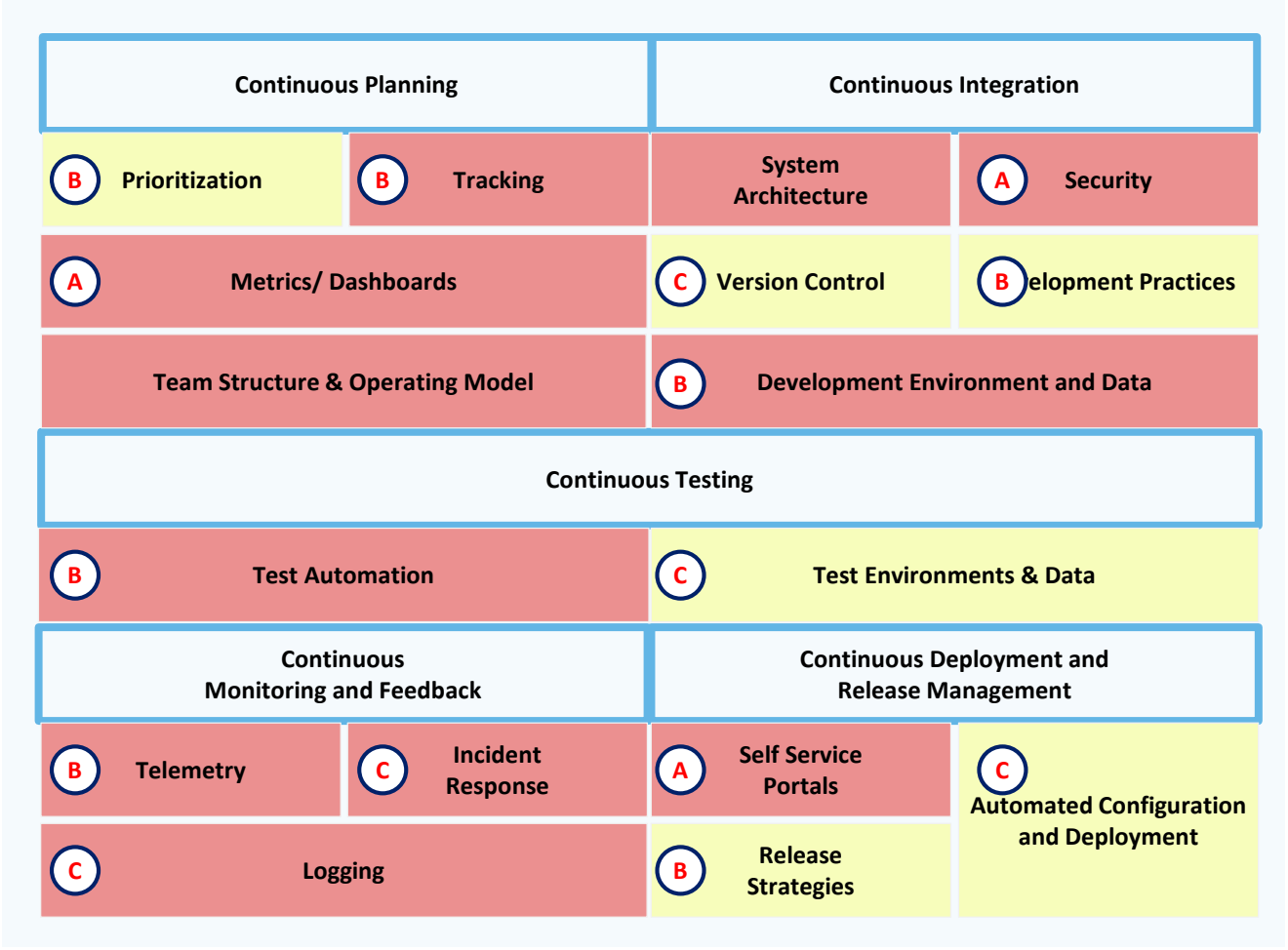
Current State - What we do well

Pekin has already kick started selected DevOps processes and introduced key DevOps tools

Planning	The technology both from Pekin and Deloitte are **enthusiastic, positive and willing to embrace** the new culture and methodology	Teams have adopted some **agile practices aligned to scrum** for faster delivery and feedback
Development	The end to end build and deployment processes **are partially automated** for few projects (e.g., PIVOT)	Approximately **70% of all open systems changes** are completed, and promoted via version controls tools Git/GitHub
Testing and Deployment	**Nearly 95%** of smoke and regression testing is automated for the PIVOT project	A majority **of the open systems use a single tool** - TeamCity - for automating the deployment process
Monitoring and Feedback	Major incidents tied to development team have **reduced by 90% since** 2014	

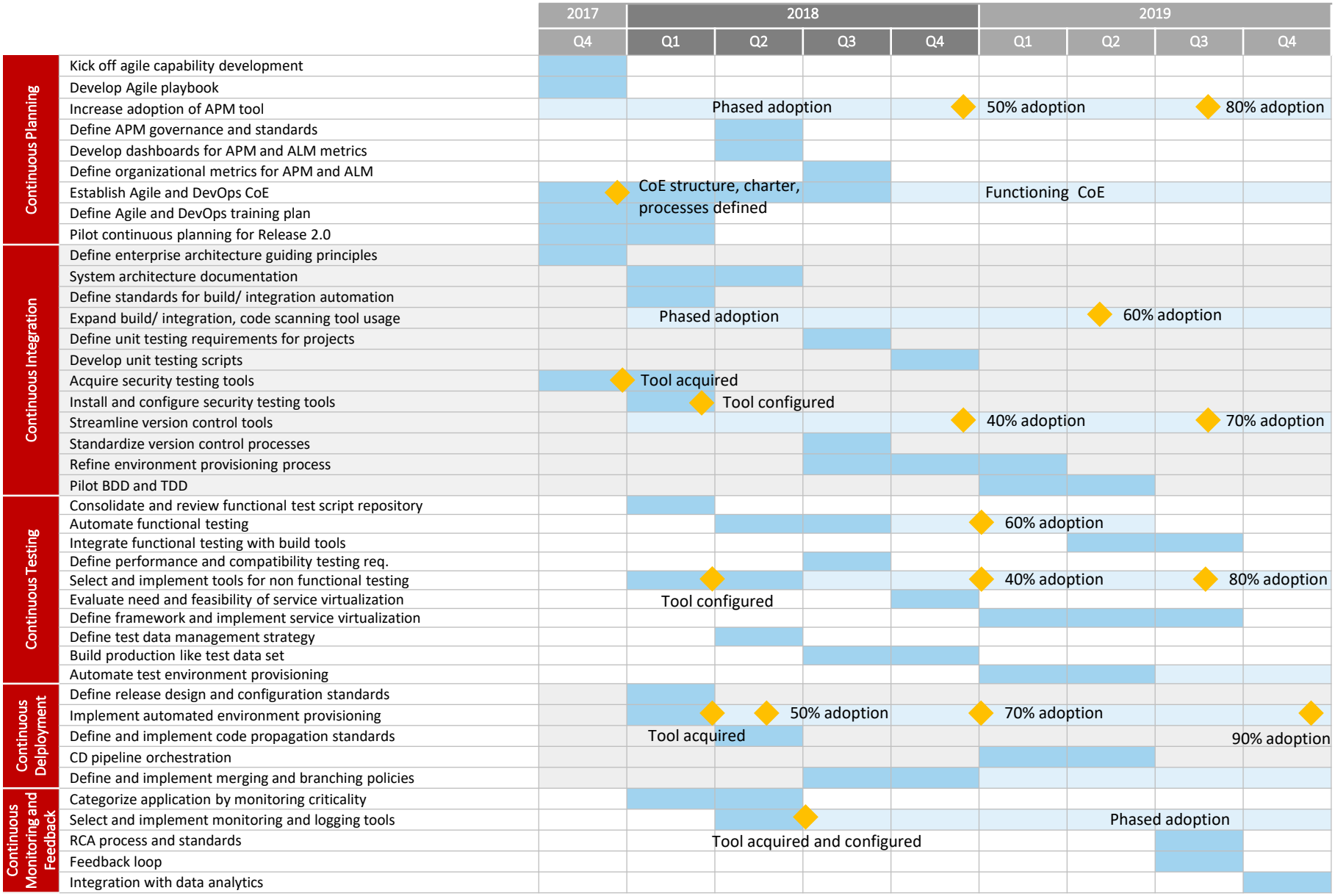
Current State - Capability Maturity Map

Overall DevOps capabilities are in nascent stages, though isolated teams have more mature processes or tools in place



The heat map indicates overall capability maturity as of 15th September 2017, levels may vary across teams

DevOps Transformation Roadmap



Pekin DevOps Key Development Milestones

The DevOps roadmap delivers incremental transformation of Pekin's technology over the next 3 years

	2018	2019	2020 and beyond
Continuous Planning	Agile operating model and methodology defined and adopted by 50% of technology teams*, training needs identified and plan defined Application portfolio management (APM) governance and standards are defined. APM tool streamlined and adopted by all agile teams Enterprise DevOps and agile center of excellence (CoE) established Key organizational DevOps metrics are defined	Agile adopted by 80% of technology teams APM tools adopted by 80% of teams to increase portfolio transparency Regular tracking of approved key metrics and generation of productivity and quality reports 50% of technology teams have resources (dev, QA, etc.) trained in DevOps processes and tools	Agile and DevOps implementation progress reviewed and goals refined Agile adopted for all suitable teams, teams mature on agile implementation Complete integration of requirement management processes with ALM tools
Continuous Integration / Build	Defined enterprise architecture guiding principles and document current architecture for 50% of applications Continuous build and integration enabled for 40% of teams Continuous code quality and security testing adopted for 40% of applications Standardized environment provisioning process	Complete architecture documentation for 100% of applications 80% of build and integration automated Code scanning and security testing adopted for 70% of applications Automate environment provisioning process for 70% of teams	Build status, code coverage and security dashboards available publicly Pilot conducted for BDD and TDD Security governance and code signing is enabled for 80% of teams 80% of teams have the capability to provision ad hoc environment for development
Continuous Testing	Functional and non functional testing standards are defined. Smoke, regression, integration and performance testing automated for 60% of teams Tests data management guidelines are established Defined enterprise service virtualization standards	Automated smoke, regression, integration and testing automated for 80% of teams Build triggered functional testing enabled for 50% of teams Implement automated test data management for 70% of open systems Automated test environment provisioning available for 30% of teams	Fully automate Smoke, Regression, integration and Performance testing with 80% adoption by teams Build triggered Functional/Integration/Regression testing enabled for 80% of teams Automated test data and environment provisioning available for all teams
Continuous Deployment	Coordinated release design, build and configuration capabilities Automated environment provisioning tools adopted by to 50% of open systems teams Process defined and implemented for code propagation on lower environments	Expand automated builds and self-service and environment provisioning tools adopted by to 70% of open systems Fully automated process for code promotion from pre-production to production environments	Improve process and orchestrated CD pipelines Implement automated builds, self-service and environment provisioning tools to 90% of agile teams
Continuous Monitoring & Feedback	Telemetry tools identified and purchased Application monitoring implemented for 20 critical applications Standardize ITSM tools used by developers, tester and IT operation across 100% of teams	Expand monitoring to selected non critical applications and servers Telemetry dashboards generated for production environments Establish monitoring and root cause analysis for infrastructure incidents	Telemetry dashboards integrated with data analytics Automated threat detection and alert and self-healing enabled

Deloitte.

* Considers teams supporting systems that are suitable for agile development

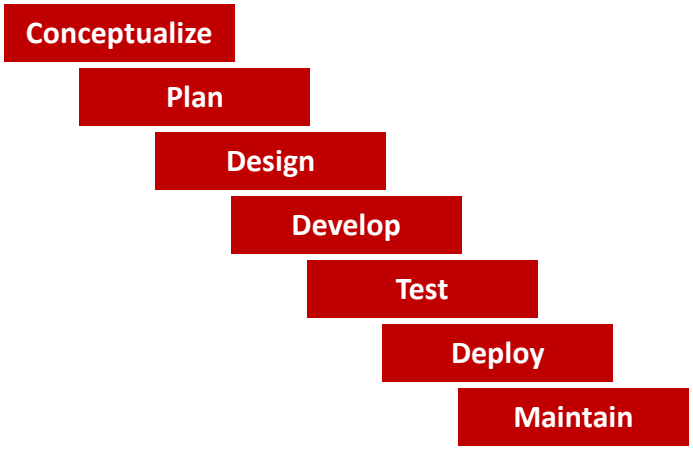
Beyond the expected™ PEKIN INSURANCE

Pekin DevOps Capability Maturity Milestones

Implementation of the roadmap will build maturity across all DevOps capabilities

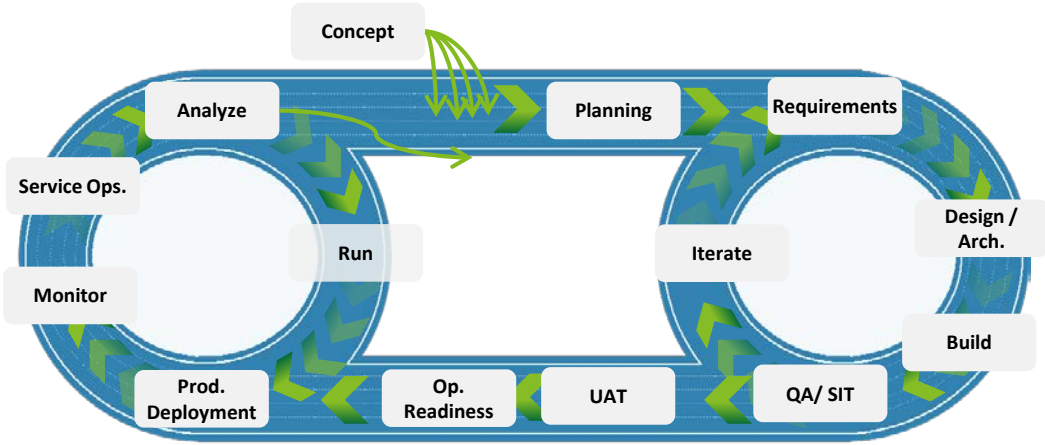
Modernization of the SDLC Framework

Current



Conceptualization to size project scope and effort and to determine resources and teams involved (e.g., architect, data, infrastructure)	Goal is to have scope freeze 1 month prior to release cycle
Gather and refine requirements	2-3 weeks/ varies with project complexity
Develop code to fulfill requirements	6 weeks
SIT to test developed functionality Configure testing environments Conduct regression testing	
Conduct user acceptance testing	After completion of SIT
Product deployed in production	1 week
Feature used by end customer*	**3 - 6 months**

Target

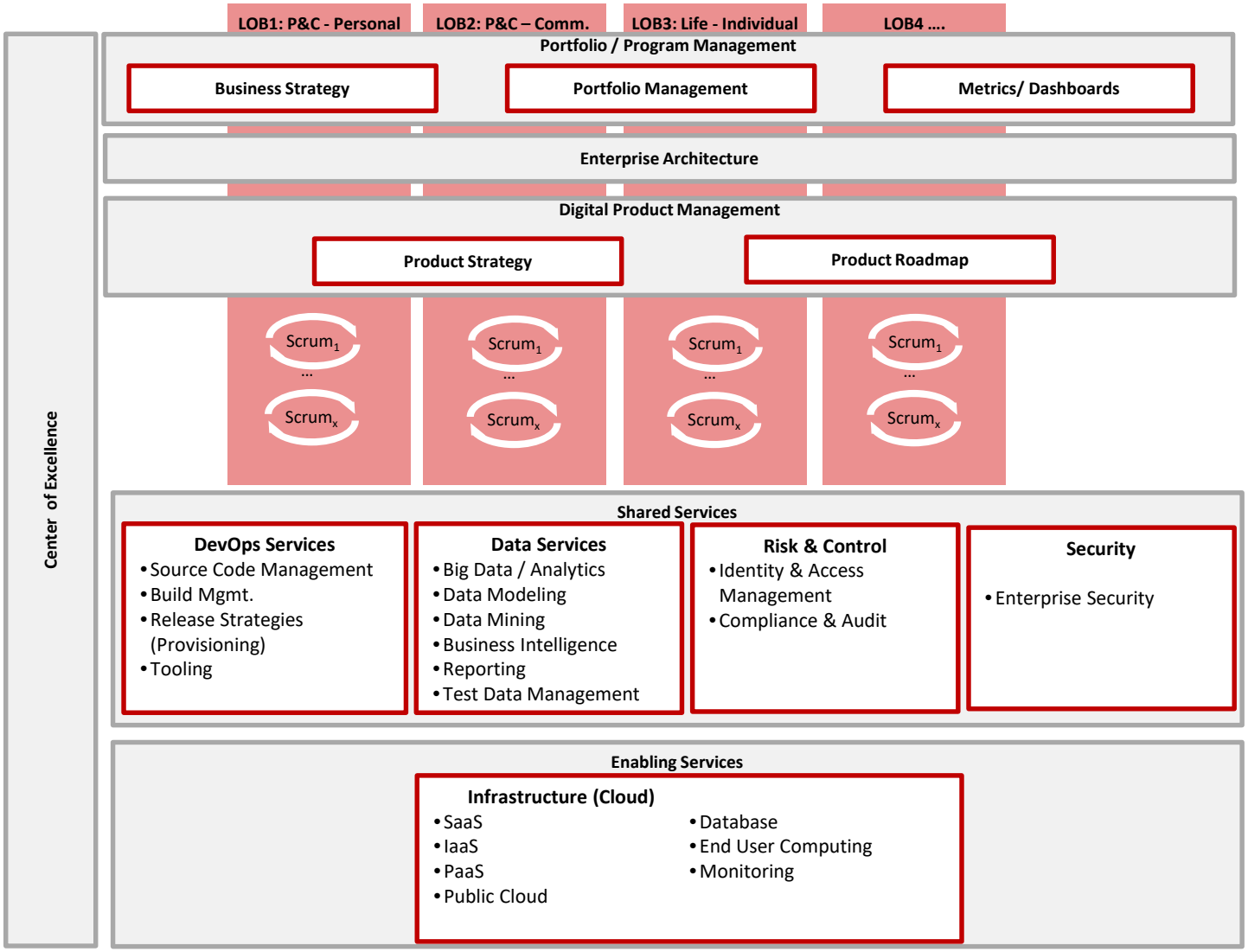


Conceptualization (due to business needs, regulatory compliance, new ideas, innovation) feeds the release planning process (1-2 meetings)	1 month prior to release planning	Quarterly
Plan, identify, prioritize and groom the features (1-2 meetings)	2 weeks prior to release planning	Quarterly
Release planning - Feature requirements are documented for each sprint and broken down into stories for development	3 days prior to sprint 1 start	Quarterly
Refine and groom user story for the upcoming sprint	2 days prior to sprint start	Each sprint
Development, build , and QA/ SIT/ UAT go through multiple iteration to develop a feature (Each sprint)	3 weeks	Each sprint
UAT determines readiness for use	Based on completed features	When ready
Feature deployed in production and is used by end customers	**Anytime within 3 months**	**When ready**

* Estimated time to delivery, exact data not available

Proposed Operating Model – Key Areas

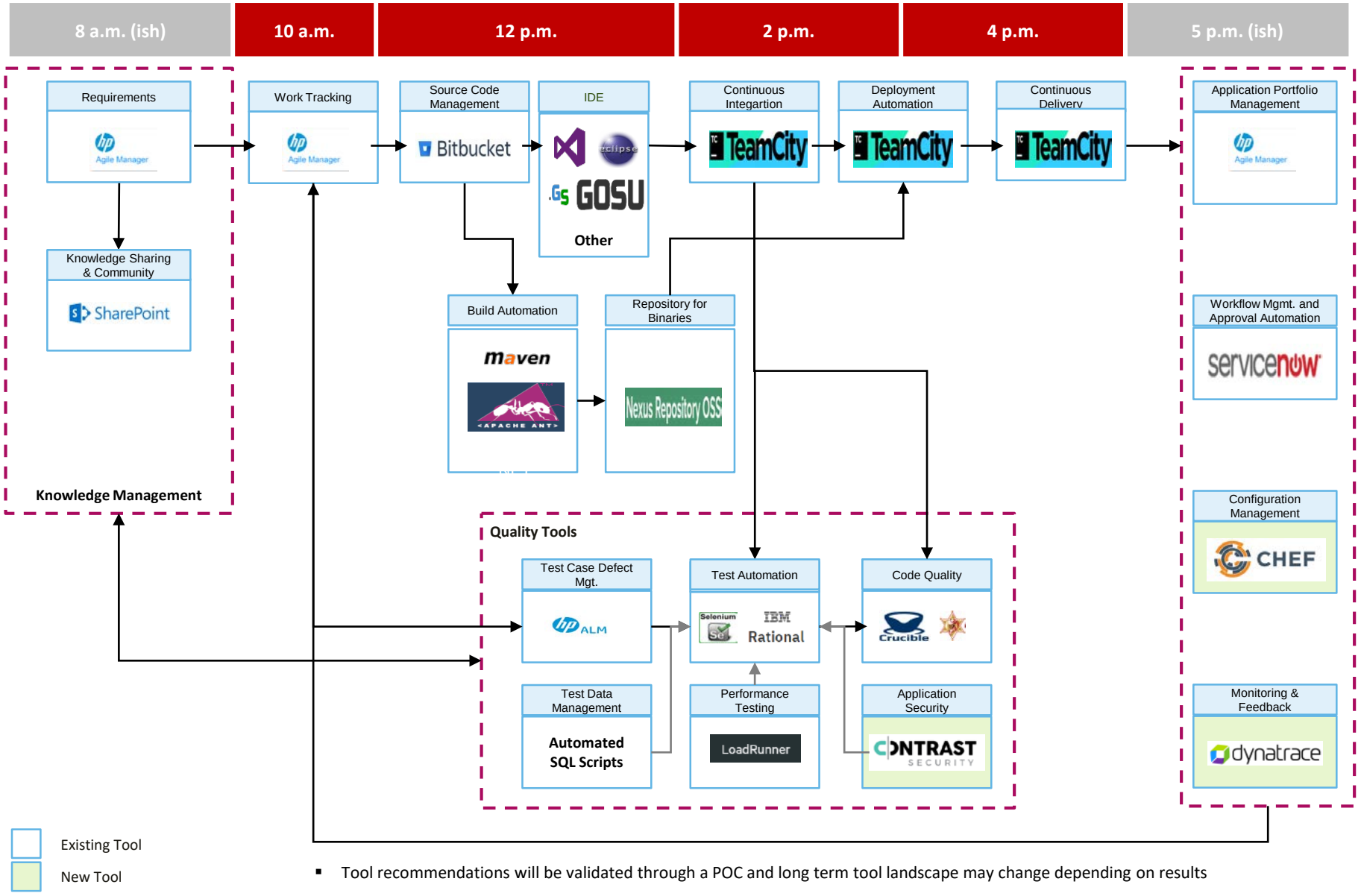
This operating model represents an organization with high DevOps maturity, rapidly responding to changing market needs



DevOps Enterprise Metrics

Categories	Drivers	Metrics
Strategic Value	How IT proves its value to the company	▪ Feature Usage ▪ Deployment Success Rate ▪ Requirements Coverage Ratio
Customer Perspective	How IT appear to customers and partners	▪ Customer Ticket Volume ▪ Incident/defect volume ▪ Mean Time to Restore Coverages (MTTR)
Operational Excellence	In which services and processes IT excel to satisfy the users	▪ Frequency of Deployment ▪ Speed of Deployment ▪ Speed of Build Verification (QA) ▪ Frequency of Build Verification (QA) ▪ Code Scanning Detection Rate ▪ Security Test Pass Rate

Future State Tools View - A day in the life of an Engineer



- Tool recommendations will be validated through a POC and long term tool landscape may change depending on results
- Mainframe tools are not captured and are out of scope for this effort

High Level Cost Projections

Target State Cost - DevOps Capability Building				
	2018		2019	
	Capex	Opex	Capex	Opex
Capability				
Continuous Planning	$ -	$ -	$ -	$ -
Continuous Build	$ -	$ -	$ -	$ -
Continuous Testing	$ 260,000.00	$ 78,000.00	$ 20,000.00	$ 220,000.00
Continuous Deployment	$ -	$ -	$ -	$ -
Continuous Monitoring and Feedback	$ 13,000.00	$ 3,900.00	$ 1,000.00	$ 11,000.00
Annual	**$ 273,000.00**	**$ 81,900.00**	**$ 21,000.00**	**$ 231,000.00**
Total Annual		**$ 354,900.00**		**$ 252,000.00**

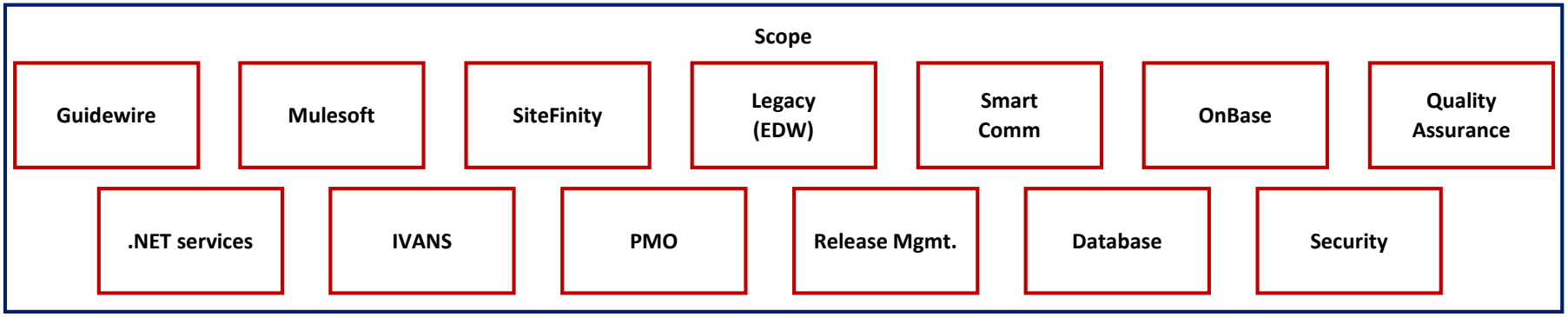
Assumptions:

- Includes cost estimates for new tools only, existing tool cost is accounted for as part of current opex
- HP ALM, AGM, LoadRunner are provided as part Deloitte PIVOT and cost is not included
- Capex includes license cost and initial investment on infrastructure (On Prem and/ or Cloud), and configuration of new tools only
- Opex (support, administration, resources, training) is 30% of Capex for Y1
- Capex post year 1 accounts for new licenses and repeat configuration due to increased tool usage
- Opex post year 1 includes license renewal with 10% price increase
- See appendix for cost estimate details by tool

Risks, Dependencies and Assumptions

1	Cloud strategy implementation
2	Conduct an end to end POC for recommended DevOps tools by end of 2017 to ensure fit with existing tools
3	Tools acquisition to enable DevOps capabilities
4	Restructuring and enablement of technology resources
5	Onboarding coaches for Agile and DevOps

Scope and Approach



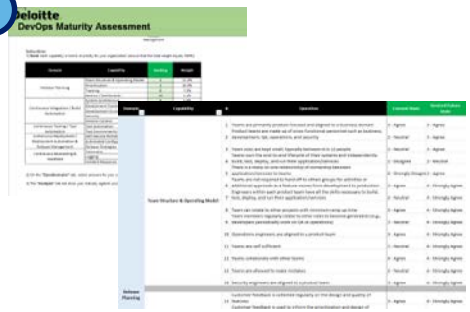
Approach to DevOps Strategy

Assess Current Capabilities	Future State Workshop	Update to Roadmap and Strategy
Collect	**Categorize**	**Summarize**
Understand current state processes followed within the organization Key activities: ▪ Interview various stakeholders identified in the key capabilities ▪ Review documentation and survey on the current state for people, process and tools whenever available. ▪ Deep dive discussion with Enterprise DevOps team to understand core DevOps practices and activities	Summarize key findings aligned to the capabilities Key activities: ▪ Group current state inputs according to key DevOps capabilities ▪ Identify key challenges and opportunities faced by each capability ▪ Organize the currently most use tools to provide transparency	Summarize the current state findings for each capability and determine the organizational opportunities and impact on driving key investments

Scope and Maturity Assessment and Interview Process

The DevOps Maturity Assessments will be accompanied by interviews in order to address gaps and answer questions resulting from analyzing results of the DevOps maturity questionnaire

1

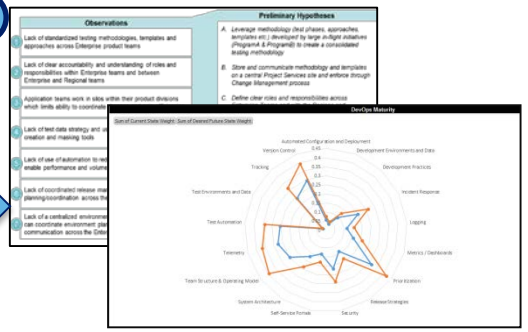


Completed Assessment Questionnaire Template

2

Collated interview data

3



Assessment results

Pre-Interview

- Key stakeholders identified
- Head's up expectations set
- Analysis of DevOps maturity assessment questionnaire results to identify interview themes
- Interviews scheduled on calendars

Interview Themes

- Typical duration 30 minutes
- Focus, by interviewee team
  - Planning
  - Development and Testing
  - Deployment and Release management
- Establish follow-up as necessary for additional info or drill-down into key subject areas

Key Outcomes

- Key interview findings (Strengths, Challenges)
- Preliminary maturity ratings
- Combine interview findings with assessment questionnaire results

Interviewees

- Product Owners / Business Analysts
- Development leads
- Build engineers
- Release coordinators
- Testing leads

Deloitte.

PEKIN INSURANCE

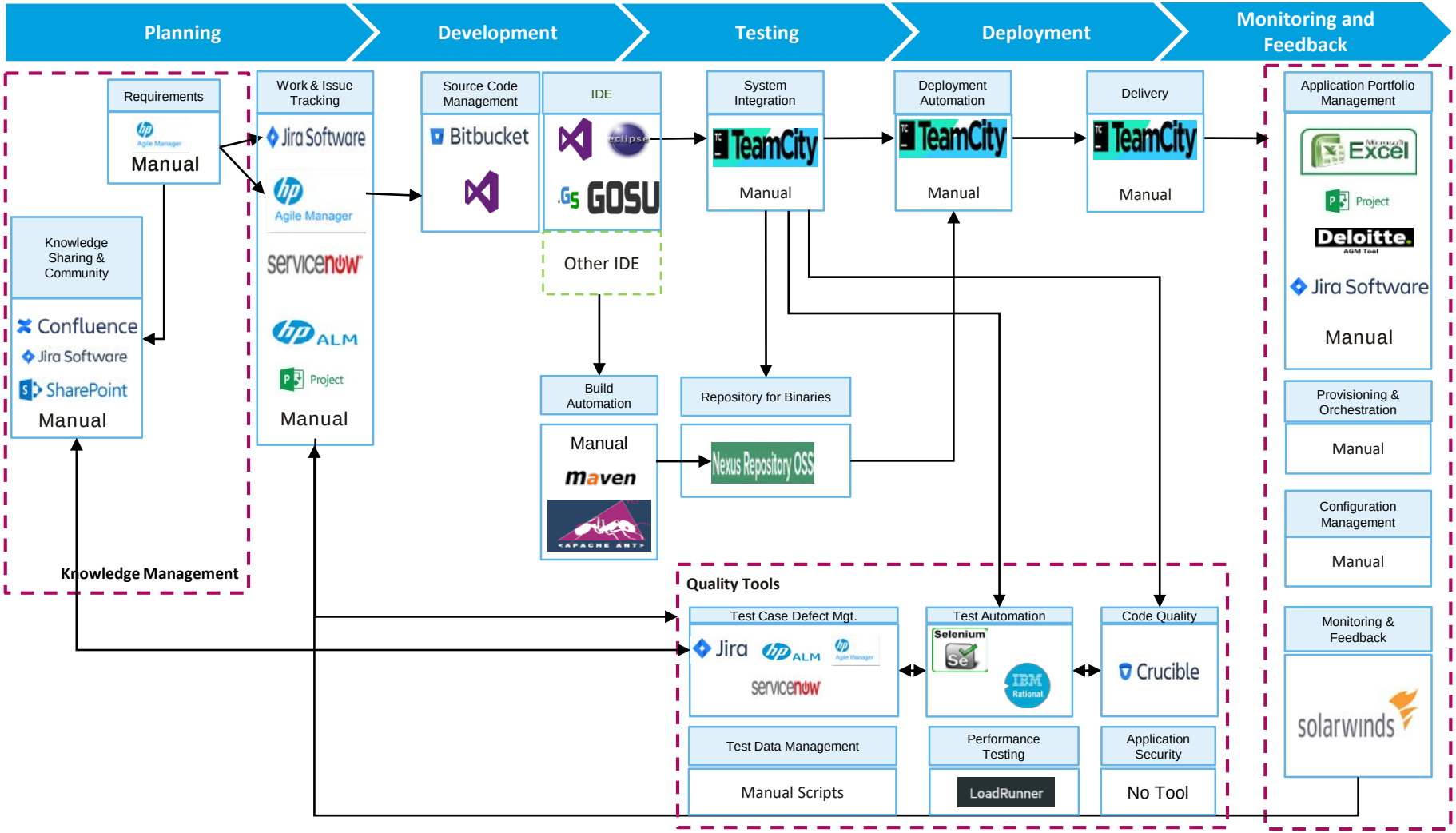
Current State

Current State – Challenges

Key challenges that impact organization's ability to achieve transparent operation and flawless execution

High-level Challenges	Details
Lack of end to end transparency	▪ Teams have **limited visibility** to the enterprise wide vision - IT work is achieved in siloes ▪ **Single view of the portfolio** is not available and **hand over of work** across teams is not a standard practice ▪ Dashboards to provide visibility on project build and deployment status are generated manually
Varying levels of process adherence	▪ SDLC process and check gates laid out by COE for non PIVOT projects is **not fully embraced** by all teams ▪ Processes for requirement documentation and test planning **vary across towers**
Inefficient cross functional collaboration	▪ Level of **business–IT collaboration varies** (very high to low) by tower ▪ Awareness on work cadence is limited, dependent teams do not have the **right information at the right time** ▪ Cross functional project **dependencies are identified late** and are handled through verbal/ mail communication
Low priority on building quality-in	▪ Developing **high quality code is not a priority**, little time spent on quality checks (e.g., code reviews) Legacy code is **complex and fragile** leading to significant IT effort to keep the "lights on" ▪ **No security, performance, and compatibility testing** integrated in the SDLC process ▪ Metrics for code quality (e.g., SIT defects) are monitored but not through a standard process
Low adoption of automation process	▪ Component and Unit testing is limited and manual ▪ Build, deployment and environment configuration is **largely manual and time consuming** ▪ TeamCity is used primarily for build automation and is not integrated with testing and deployment tools ▪ **Dependencies on small group of resources** to manage the build and deployment process manually ▪ Current branching process is manual and complex
Lack of tool standardization	▪ **Standard tool are not used** for planning of non PIVOT projects. Teams use Microsoft project, excel, word etc. ▪ **Multiple repositories** are used for storing documents, repository structure is project specific ▪ QA team to **trace requirements to test scripts manually** ▪ Tools in use currently may not be **cloud compatible**
Limited implementation of security strategy	▪ Security is **not built in during code development**, code scanning is not a standard practice across teams ▪ **Production data is used for testing** in some cases creating a security risk
Minimal application monitoring	▪ **Application monitoring and feedback loops are not in place**. Monitoring of servers, DBs is reactive in nature, feedback loop is manual ▪ RCA is limited to production defects
Limited development and testing environments	▪ SIT/ UAT **use the same environment**. SIT environment is too burdened to conduct performance testing ▪ **Single path to production** for non PIVOT systems, release, break fix and ad hoc work use the same servers

Current State Tool Stack



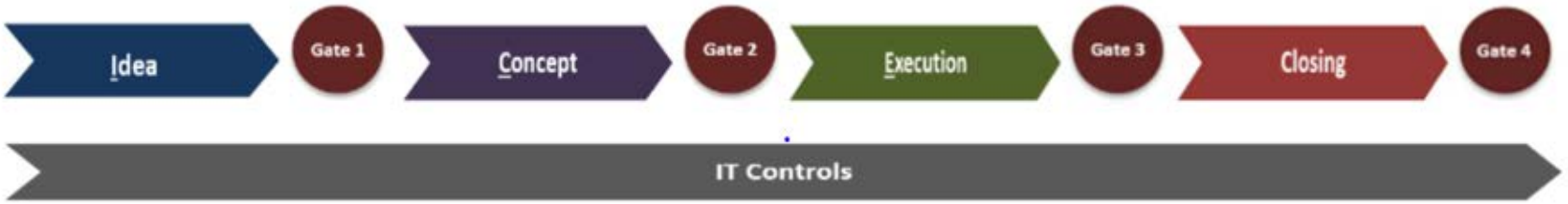
Legend

Standard offering — Engineer preference — Supporting services

Current state as of 15th September 2017

Mainframe tools are not captured as out of scope for this effort

Current State Process



Gate 1

- Idea Submission Document is approved by the IT Steering Committee or appropriate IT Tower Director.
- The IT Steering Committee or IT Tower Director assumes responsibility of Gatekeeper in the Idea phase

Gate 2

- The Project Manager records the Concept phase of the project complete and can move on to the Execution phase.
- The Steering Committee Moderator assumes responsibility of Gatekeeper in the Concept phase

Gate 3

- The Project Manager and Business Analyst record the Execution phase of the project complete and can move on to the Closing phase.
- The Project Manager/Business Analyst assume responsibility of the Gatekeeper in the Execution phase

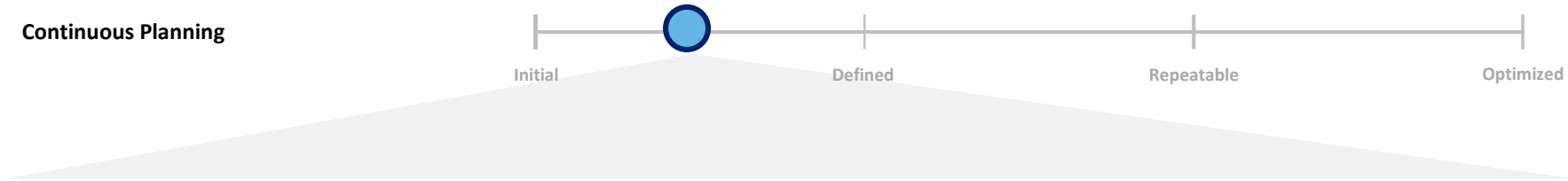
Gate 4

- The Project Manager records the Execution phase of the project complete and can move on to the Closing phase.
- The Project Management Office will ensure all deliverables have been completed through all phases of the project

Capability Maturity – Continuous Planning

Planning phase processes are not standardized across teams, inputs to the process and tracking are manual

Capability	Current State Maturity



Observations/ Rationale

Prioritization
- Releases occur every 6 weeks
- Work planning and prioritization process is defined but not fully adopted
- High level requirements are used for effort estimation, estimates may be revised later in the release cycle
- Requirement documentation and management is highly manual

Tracking
- Tracking of productivity and quality metrics during SDLC is limited
- Work tracking is manual and visibility across teams is low

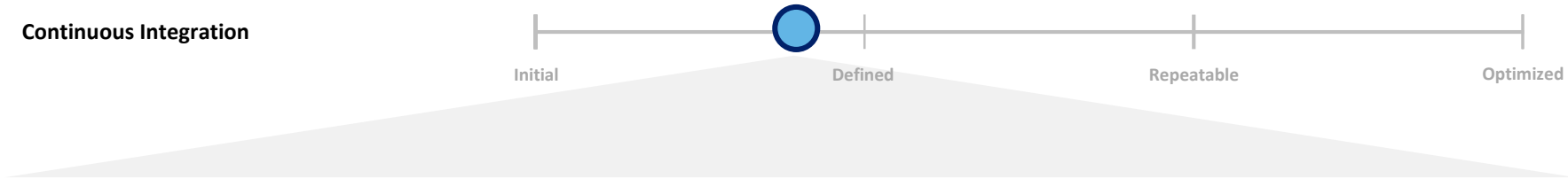
Metrics and Dashboard
- Metrics for measuring quality and productivity of releases are not used consistently across teams
- Information for dashboards is manually collated and presented

Team Structure & Operating Model
- Collaboration level across teams varies
- Teams are aligned to products/ lines of business

Capability Maturity – Continuous Integration

Tools for version control and build provide partial automation and are adopted by a majority of the teams

Capability	Current State Maturity
Continuous Integration	Initial · Defined · Repeatable · Optimized



Observations/ Rationale

System Architecture
- Legacy architecture is monolithic, systems are tightly coupled
- PIVOT projects use ESB for loose coupling
- Enterprise and solution architecture functions are still maturing
- SMEs and developers take on solution architect roles when required

Security
- Code scanning is not a standard practice
- Security is not built in during code development
- Security is considered as the responsibility of the enterprise security team alone

Version Control
- Source code is versioned, tools are available for version control
- Branching strategy is not uniform across teams and is complex

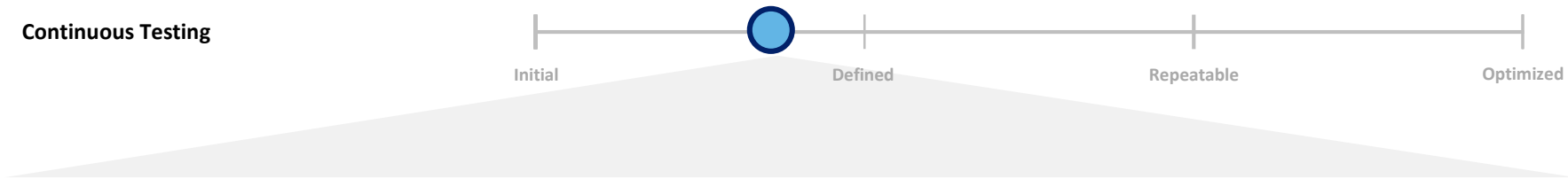
Development Practices
- Static code analysis is not conducted
- Unit testing is partial and manual
- Build process is automated but still requires some manual intervention

Development Environment and Data
- No production like integrated environment

Capability Maturity – Continuous Testing

Emphasis is on functional testing, non functional testing lags behind in maturity

Capability	Current State Maturity



Observations/ Rationale

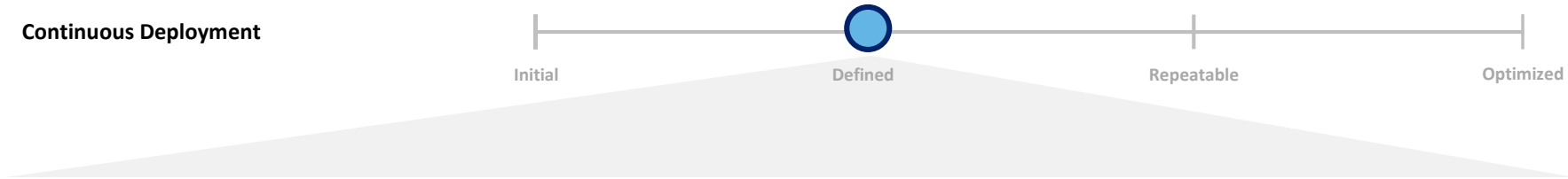
Test Automation
- In some cases, functional testing does not accompany coding and begins after development is completed
- Regression suite is not componentized and complete regression run required for isolated changes
- Non functional (i.e., performance, compatibility) testing is not a standard practice
- QA team solely owns the testing efforts

Test Environment and Data
- Limited test environments available, SIT/ UAT use the same environment
- Production data used for testing selected sub systems (e.g., billing)
- Test data generation is not fully automated
- Provisioning of testing environment requires coordination with multiple teams and is not owned by QA

Capability Maturity – Continuous Deployment

Deployment is partially automated, challenges exist in provisioning environments

Capability	Current State Maturity



Observations/ Rationale

Self-Service Portals
- No self service tools available for developers for build automation. Development and CI/ CD teams coordinate build schedules
- Environment configuration is manual
- Progress of build and deployment monitored and reported manually

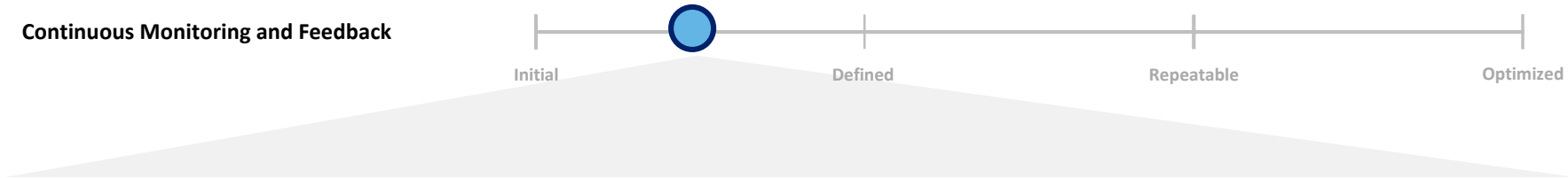
Release Strategies
- Release strategy is defined for agile and waterfall SDLC but standardized across the organization
- Automation for code propagation is in initial stages. Process for pre prod to prod is manual and varies by team

Automated Configuration & Deployment
- Code deployment is manual for all environments
- Source code is versioned, tools are available for version control
- Branching strategy is not uniform across teams and is complex

Capability Maturity – Continuous Monitoring and Feedback

Application monitoring is not adopted by most teams leading to reactive issue resolution

Capability	Current State Maturity



Observations/ Rationale

Telemetry
- Application and environment monitoring is in nascent stages
- Reports are manually generated at defined frequencies. Dashboard data is collated manually

Incident Response
- ServiceNow is the standard tool for incident management. Incidents, problems and change requests are created manually
- Testing captures most defects before production releases, number of defects shows a reducing trend
- Operations, developers and QA coordinate ITSM activities manually
- RCA is conducted manually and not for all incidents/ problems

Logging
- Server monitoring tools are available but used for selected servers (e.g., critical production servers) only – follow up on alerts is reactive in most cases
- Security events are logged for selected systems

Interview Participant List

Area	Name
Pekin Insurance	
Database	Crystal Kilgus
IVANS	Judith Fowler
Legacy Systems	Jerry Lorentzen, Reed Harvey , Randy Dray Michael Stauffer, Sue Jones
.NET services	Mike Vacca, Ryan Smothers, Brett Nye, Thomas Bollard, Darrell Johnston, Mike Rowe
PMO/ ITIL	Gary Finamore
QA/ Application Testing	Sean Horack, Latha Uppugonduri
Release Management/ ITSM	Darrell Johnson, Leigh Hess
SiteFinity	Chad Tiezzi
Smart Communications	Julia Sutherland
Deloitte	
Data/ GW data	Varun Kutty
CI/ CD	Sudhakar Srivastava
Infrastructure	Pankaj Punjani
Guidewire	Akshay Heroor
IVANS & SiteFinity PMO	Prashant Karande
QA/ Application Testing	Saurav Kumar

Current State Details - Continuous Planning (1 / 2)

Plan
Capacity Management: ▪ Yearly planning process lead by business, followed by reviews through the year ▪ Project evaluated by business and IT steering committee to check for resource availability and prioritization ▪ Tower specific projects managed by tower IT, PMO is involved in the estimating, planning and execution of large or cross functional projects ▪ Frequency of reviews varies by tower and is dependent on level of business involvement ▪ Accommodate ad hoc and emergency changes using capacity available after planned work or reprioritize ▪ Regulatory changes and associated deadlines, fragile legacy code (e.g., raters_ and spill over from previous releases lead to hot fixes in addition to releases ▪ Prioritization for PIVOT is based on business needs and for release 1 saw several additions (e.g., IVANS as a CR) ▪ IT can push back on portfolio priorities but this is limited due to regulatory compliance needs **Team Structure:** Partially federated with teams aligned to product line (i.e., PL, CL, Life) and enterprise services In addition have a COE (BA, QA,). BA, QA are managed by tower but dotted line to COE for guidelines Team size varies by tower and is often a mix of in house and vendor resources DevOps resources scattered across towers, have varying capability maturity levels DevOps as a practice is reactive and focused more on CI/CD **Application Portfolio Management:** ▪ Defined application lifecycle stages ▪ Several legacy application already being replaced by Guidewire, other third party systems or in process of upgrading to current versions ▪ Cloud suitability of applications under assessment **SDLC Methodology:** ▪ PIVOT teams follow agile methodology ▪ COE sets SDLC process and standards for the entire org. and is trying to standardize it across the towers. People like to pick and choose areas to implement. Total rate of adoption of standard process: PNC 50%, Life 20-30% ▪ SDLC Process documents available on single SharePoint site ▪ Non PiVOT projects follow waterfall SDLC with 6 week release cycle. However teams need to follow an agile mindset on requirements as changes/ additions to are common within a release and may come in late in the release cycle ▪ The aim is to have all towers and projects go through same 4 step process and check gates. Adoption and understanding of process is increasing but still not full

Current State Details - Continuous Planning (2 / 2)

Plan

- Plan is to move all projects to move to scaled agile approach. Ran a SA (Safe agile) and SPC training one month ago, 10 people (10 BA, dev)
- Waterfall SDLC process and check gates are based on ITIL processes. (add four step here)
- PIVOT and supporting projects follow agile or a mix of SDLC and agile. Some vendors follow a different release cycle from Pekin (e.g., IVANS follows agile but vendor is 6 week + 6 week cycle)
- Agile also appears as waterfall as often requirements are defined in one sprint, development in the next and testing in the following sprint

Requirement Management:

- No formalized process for requirement documentation and management, COE process followed by some teams only
- effort estimation involves BA, Dev and QA but often is made based on Initial requirements are generated by business and provided to BAs
- Detailed requirements are often flushed out later in the sprint/ release
- Management of cross functional requirements is through verbal communication or through mails. Involves coordination with multiple teams
- PIVOT requires sprint grooming and coordination with centers for cross functional stories
- Central repository for requirements exists but adoption and sue is low
- Teams use different areas to store requirements (e.g., I drive) , directory structure is not defined
- Moved from TFS to Service Now and several processes have been updated due to this change
- Quality of requirements is mixed and with varies Bas experience and knowledge levels
- Business and IT BA roles overlap is some towers and leads to multiple handoffs and slower requirement flow

Risks and Issue tracking:

PMO/ BA for risks and issues on individual projects and escalation point is tower IT director

Tools:

- Excel
- AgM
- Planned to use Doors for requirement management but never implemented
- Rational Tool Suite (RFT, RQT))
- Word documents (SIR)

Current State Details - Continuous Build/ Integration (1/2)

Build/Integrate
Enterprise Architecture: ▪ Recently formalized the enterprise architect role and responsibilities and set up an EARB ▪ EARB suggested closer collaboration with business teams. Both PL and CL business teams are currently involved closely in the IT development process ▪ No formal solution architect role, a number of people have legacy knowledge and are SMEs. Developers double up as solution architects. Most changes are within the existing framework and not major revamps. **System Architecture:** ▪ Legacy architecture is monolithic ▪ Attempt to use MuleSoft ESB to encourage loose coupling of applications ▪ There are opportunities to refactor code and improve architecture effort but limited implementation tight deadlines, also QA is not involved in the refactor effort so testing changes is a challenge ▪ GW generates a single war for each center, Mulesoft war mapped by services **Security:** ▪ Security testing: None ▪ TeamCity provides inbuilt features for code scanning. Current Code scan is limited to checks for failures in build due to the code changes and not the code itself **Build Management:** ▪ TeamCity used as the primary tool for build management, not adopted by all teams ▪ For non PIVOT teams, there are no daily builds, Components are usually pulled out of prod, changed and deployed , process is long ▪ Ideally want to get daily builds – smoke test – run, deploy ▪ Builds are automated only for PIVOT programs. This too is partially automated and not triggered at every code check-in. PIVOT has two scheduled build every 24 hours that are manually triggered ▪ Ad hoc builds are also covered by the DevOps team though dev managers also can trigger the build. Usually see ad hoc builds following a demo of features to business/ IT leadership ▪ Team city sends mail notifications to selected groups on build status ▪ Tracking of build status is manual to a large extent and requires coordination across dev, CI/CD and QA teams

Current State Details - Continuous Build/ Integration (2/2)

Build/Integrate
Development Practices: : ▪ Small teams have attempted TDD but most of enterprise not aware of practice and does not follow ▪ Unit testing is limited and manual, varies with tower. TeamCity has a the ability to configure unit testing and is planned after November ▪ Unit Testing goal is to be defect free once code gets to SIT, individual developers should be responsible for the code and defects For PIVOT, OOB functionality s is not unit tested, customizations are unit tested ▪ Some teams implement mandatory code reviews . The review is limited to adherence to coding guidelines, best practices, and standards and not actual functionality ▪ In the past some teams have tried coded unit tests, also experimented with paired programming, both stopped due to lack of time

Current State Details - Continuous Testing (1/2)

Test
▪ Pekin has a centralized QA group that is part of the COE. Test teams also use vendor resources (Deloitte, Value Momentum) ▪ More robust testing has been put in place as move from mainframe to a more distributed platform ▪ Testing usually follows the completion of development for a component and is not continuous. Attempt to get to scope freeze a month to 5 weeks before the beginning of a release cycle and begin creating test scripts after code freeze ▪ PIVOT SIT is in phases to allow testing of components as they become available ▪ Rigor of testing varies by tower and between projects. In the past used test plans, test cases, review meetings, test case review but is not implemented by all towers today ▪ For projects larger than 70 hrs. a test plan is created and provides traceability to defects. Smaller efforts are deployed as part of break fixes. Each tower may have a varying process for test planning ▪ The life LOB is currently developing a test framework training for the team ▪ Once code in SIT, dev is not part of the testing effort ▪ Traceability of requirements to test scripts is manual and not standardized across teams (e.g., PIVOT uses story cards, non PIVOT uses manually created matrix) **Test Automation:** ▪ Test automation is partially adopted using Selenium*. QA uses HP ALM (PIVOT) and Rational Test manager and RUP (Non PIVOT) - test cases are written in Excel and then load to Rational suite ▪ Scripts are automated but not plugged in with TeamCity. Automation of test scripts for PIVOT is difficult in the initial phases due to multiple UI changes that would need frequent changes to scripts ▪ Time required to complete QA differs across towers ▪ Limited or no code scanning ▪ No security testing ▪ Limited or no performance testing, load testing , and compatibility testing ▪ Regression is automated but not robust. Updates are made to the regression suite after each release ▪ Single regression suite used and entire suite is run even if changes are made to selected modules. Regression by module (e.g., CL only) would be easier. One reason for the consolidated suite may be that cross functional dependency is high (e.g., rater, web raters) ▪ UAT for non PIVOT often is dependent on SME resources and script documentation is limited ▪ No testing for mobile applications ▪ Analysis of current test script inventory is planned * Testing automation levels vary across teams. See following slides for details of PIVOT testing coverage and automation

DevOps Maturity Framework

Current State Maturity Definition: Continuous Planning

Depending on post-assessment maturity for each capability, the below guidelines provide high level areas to improve maturity in each DevOps capability

Capability	Maturity Spectrum			
	Initial	**Defined**	**Repeatable**	**Optimized**
Prioritization	▪ High Ceremony change control ▪ Resistance to change control ▪ Frozen and infrequent releases ▪ Verbose use cases ▪ Large statements without context	▪ Time bound production releases and occur every few months ▪ Features can be added and reprioritized within release cycle. ▪ Requirements broken down into epics ▪ Estimation is inaccurate	▪ User stories define business requirements ▪ Requirements are broken down into technical tasks. ▪ Accurate estimation ▪ Active involvement of business in delivery ▪ Bi-weekly/monthly release cycles	▪ Ability to frequently reprioritize requirements ▪ Multiple releases/week ▪ Requirements not derived from formal documentation ▪ Business need drives business requirement process
Tracking	▪ No SDLC tracking for product quality and release process	▪ Capture product release and process to some extent	▪ Organized quality and productivity reports, but not publicly displayed	▪ Quality and productivity reports are readily available ▪ Publicly-displayed dashboards to track products flow and features
Metrics / Dashboards	▪ Minimum or no emphasis on Dashboard and metrics	▪ Technical debt information is readily available ▪ Scattered dashboard for code coverage and security metrics	▪ Publicly display charts and or dashboard for failures or defects ▪ Customized Security Metrics dashboard	▪ Established dashboards available for code coverage metrics, product and release quality ▪ Stabilized dashboard for security metrics is available ▪ Dashboard are regularly used in planning decisions
Team Structure & Operating Model	▪ Developers have specific roles with limited ability to make changes ▪ Development teams have access to dev environments but limited knowledge ▪ Low collaboration between teams ▪ High ramp up time	▪ Team sizes are kept small, typically between 6 to 12 people ▪ Pair programming encouraged	▪ Teams are primarily product and customer focused ▪ Teams pair up for requirement execution ▪ Cross functional pairing (Dev, QA, BA) etc.	▪ Teams are self sufficient and members are self motivated ▪ Real time pairing with business ▪ Full cross team collaboration

Current State Maturity Definition: Continuous Integration (1 / 2)

Capability	Maturity Spectrum			
	Initial	**Defined**	**Repeatable**	**Optimized**
System Architecture	▪ No application and infrastructure architecture design ▪ No integration and enterprise architecture design	▪ Limited functions for applications/services to serve specific business domain ▪ Minimum Enterprise architecture design	▪ Represent Application and System Architecture design for all the systems ▪ Applications readily available for cloud migration ▪ Scalable, robust and tested applications and services in changing environment	▪ Application and system design aligns with enterprise architecture design ▪ Service-oriented architectures, such as Micro-services, are primarily used
Security	▪ Perform initial security assessment after integration Testing ▪ No security test automation and encryption	▪ Practice initial security assessment during code review ▪ Moderate security governance ▪ Security checks are performed for code commits but with limited analysis	▪ Implements comprehensive security governance ▪ Begin Initial security assessment during inception phase	▪ Automated security test and encrypted communication ▪ Comprehensive security governance ▪ Developers run automated local security scan before committing code
Version Control	▪ No or limited version control exists	▪ Versioned source code but merging is painful and dreadful ▪ Developer can create branches at an ad-hoc basis	▪ Strong version control ▪ Well established merging and branching policies ▪ Use Version control for automated and manual test scripts	▪ Automated version control for code and all configurations ▪ Everything versioned, reproducible and auditable ▪ Version control for production application code and application configuration

Current State Maturity Definition: Continuous Integration (2 / 2)

Capability	Maturity Spectrum			
	Initial	**Defined**	**Repeatable**	**Optimized**
Development Practices	▪ Manual build ▪ No Automation for code build and testing ▪ No Static Code analysis ▪ Teams writes testing scripts after writing code	▪ Defects and features undergoes code reviews and approvals ▪ Build process is automated and repeatable ▪ Security is not a part of SDLC ▪ Build enables Static code analysis	▪ Versioned builds and artefacts ▪ Developers rebase code before merging to master and unit tests developed in conjunction with new code ▪ Build triggered with each code change ▪ Implement role based security across SDLC ▪ IDE integrations for static code analysis	▪ Build triggered with each code change and deployed to environment. ▪ BDD and TDD practices followed by dev teams ▪ Security can be audited ▪ Security governance and code signing process enabled
Development Environments and Data	▪ No production like integrated environment ▪ Developers do not have environment to debug acceptance test failure	▪ Production-like testing environment is available during early stages ▪ Development environment can be can created and destroyed at ad-hoc level ▪ Separate environment to debug acceptance test failure	▪ Development environment be created and destroyed on need basis ▪ Developers debug acceptance test failures on production-like environments	▪ Dev/Test, Staging are available to build and test production like environment. ▪ Developer has access to production like environment for build and test. ▪ Environment can be build and destroy with complete automation

Current State Maturity Definition: Continuous Testing

Capability	Maturity Spectrum			
	Initial	Defined	Repeatable	Optimized
Test Automation	▪ Takes long time to complete smoke, Regression, Integration and Performance testing ▪ Manually Unit test cases and improperly documented ▪ Owned by QA ▪ Functional/Non-Functional/Integration performed at the end of Lifecycle	▪ Manual process for Smoke, Regression, integration and Performance testing ▪ Limited functional and non-functional tests for deployment ▪ Test cases can be done without requiring integrated test environment ▪ Functional testing not integrated into build process	▪ Smoke, Regression, integration and Performance testing with limited automation ▪ Build process can trigger functional testing ▪ Owned by QA/Dev	▪ Fully automated Smoke, Regression, integration and Performance testing ▪ Automated functional and non-functional tests for deployment ▪ Developers or QA engineers on product teams create and maintain automated smoke, acceptance and integration tests for their application/services ▪ Functional/Integration/Regression testing integrated with Build process ▪ Owned by QA/Dev/Business
Test Environment and Data	▪ Poor management of test data ▪ No Integrated, production-like test environments are available for testing	▪ Deployment to test environment requires downtime ▪ Integrated, production-like test environments are available for testing on ad-hoc level	▪ Test data is either auto-provisioned or available on demand for test environments ▪ Test data is very similar to masked production data	▪ Test data is provisioned to the test environment automatically ▪ Integrated, production-like test environments are always available for testing

Current State Maturity Definition: Continuous Deployment

Capability	Maturity Spectrum			
	Initial	**Defined**	**Repeatable**	**Optimized**
Self-Service Portals	▪ No or very limited self-service tools for developers and testers ▪ Environment provisioned manually	▪ Set-up Self-service tools and environment specification for each environment ▪ Lower environment available for dev testing but out of sync with prod	▪ Partially automated and self-serviced tools and environment provision also require IT intervention	▪ Automated environments provisioning with Immutable architecture ▪ Self-service tools are available for developers to configure new build procedures and pipelines on demand
Release Strategies	▪ Lack standardization and automation for promoting pre-production to production environments	▪ Features toggles on or off in production using configuration changes ▪ Standardized and manual process of promoting pre-production to production environments	▪ Standardized and partially automated process to upscale from pre-production to production environments	▪ Standardized and fully automated process of promoting pre-production to production environments
Automated Configuration & Deployment	▪ Deployments performed by different team ▪ Manual deployments	▪ Automated code deployment and configuration	▪ Automated code and application configuration across environments. ▪ Orchestrated CD pipeline for all releases	▪ Rollbacks are possible with every deployment ▪ Code merge triggers automated build, test, and deploy jobs to acceptance and performance testing environments

Current State Maturity Definition: Continuous Monitoring and Feedback

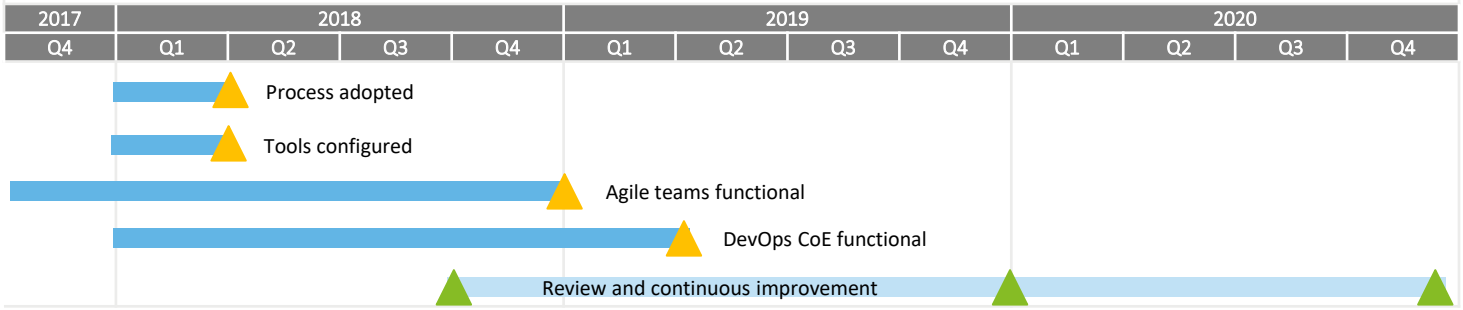
Capability	Maturity Spectrum			
	Initial	Defined	Repeatable	Optimized
Telemetry	▪ Require standard toolset for centralized dashboard ▪ Manually generated monthly, weekly or daily reports	▪ Telemetry dashboards are available for production and some pre-production environments but only limited some systems ▪ Telemetry tools are available to alert development teams and operators for data to meet pre-configured conditions	▪ Built-in telemetry dashboards available for production and some pre-production environments ▪ Perform statistical analyses telemetry data to identify anomalies at ad-hoc level	▪ Perform statistical analyses on telemetry data to identify anomalies ▪ Telemetry dashboards are available for production and some pre-production environments
Incident Response	▪ Operation team manually generates incident, problems and change request through ITSM tools ▪ Alerts, incident, problem and change management are in place for all services	▪ Operation team coordinate with Developer and tester to resolve defects and alerts ▪ Operation team manually generates incident, problems and change request through ITSM tools	▪ Most of the defects, alerts and issue handle through automation ▪ ITSM tools facilitate communication between Developer, tester and IT operation	▪ Testing captures all defects before a change gets released to production ▪ Automated threat detection and alert ▪ Automated process for self-healing
Logging	▪ No Monitoring in place for Infrastructure incidents ▪ Generating logs takes lot of time and effort	▪ Develop custom tooling to generate system logs ▪ Security event logs are available for limited systems	▪ Monitoring tools used in production environment ▪ Implement automatic root cause analysis through telemetry logs	▪ Application Performance Management is continuously monitored ▪ All environments monitored according to SLA''s defined ▪ Monitoring Dashboards published and used in RCA and Incidence Management as a practice

DevOps Level 2 Roadmap

Continuous Planning – Activity Detail

Continuous Planning

Initiative Timeline

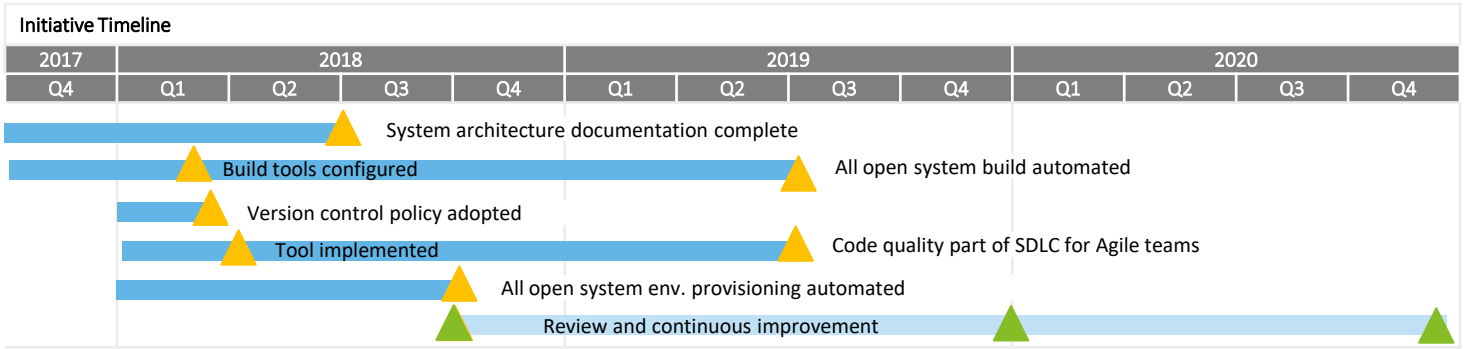


	Success Indicators
1	APM Tool implementation for 50% of technology teams by Q1 2018
2	Agile CoE kickoff by Q1 2018
3	Agile implementation initiated for 50% of teams by Q2 2018
4	DevOps CoE setup by Q2 2018

Initiative	Description	Complexity		Value		Dependency
Application Portfolio Management (APM)	• Increase adoption of existing APM tools • Define APM processes aligned with the capabilities of the tool	MED	Align processes across teams	HIGH	Increase portfolio transparency and improve planning	Tool implementation
Application Lifecycle Management	• Select single existing ALM tool for enterprise use • Define and adopt a consistent structure for project SharePoint • Define ALM processes aligned with tool capabilities • Align project and requirement management across the organization • Decommission other legacy APM tools	MED	Align processes across teams	HIGH	Present single view of portfolio	Tool implementation
Agile Adoption	• Define Agile standards, operating model and processes for the organization • Restructure teams to implement agile • Plan agile training paths and coach teams • Adopt agile methodology for all open systems	HIGH	Transition from existing waterfall SDLC needs long term changes	HIGH	Enhance delivery speed	Availability and training of resources
Project Portfolio Management (PPM)	• Establish consistent program and project governance • Define top 5 metrics for project tracking and success	MED	Aligning PPM with agile requires more involvement from PMO	MED	Ensure project success and quality	Cooperation from technology teams
DevOps CoE Setup	• Establish core DevOps group • Define DevOps standards and processes • Streamline DevOps tools	HIGH	Availability of resources	HIGH	Enable continuous delivery	Availability and training of resources

Continuous Build and Integration – Activity Detail

Continuous Build and Integration

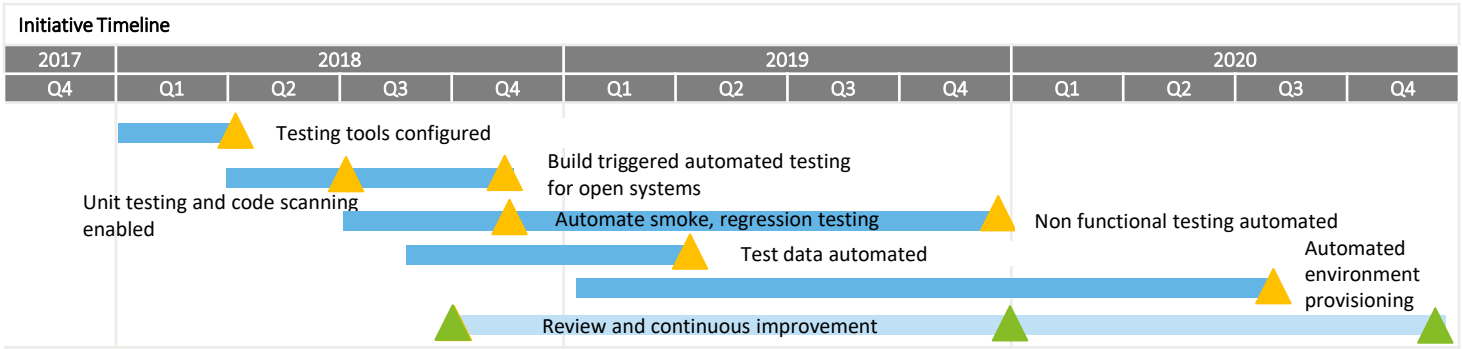


Success Indicators	
1	System architecture documentation for 100% of applications by Q2 2018
2	Build automated for 50% teams by Q2 2018
3	Code quality tools available by Q1 2018
4	Automated environment provisioning for 80% of open systems by Q3 2018

Initiative	Description	Complexity		Value		Dependency
System Architecture	• Define enterprise wide architecture guiding principles including componentized architecture (where applicable) and cloud enablement • Complete documentation of current architecture templates and blueprints for all applications	MED	On going changes to architecture	HIGH	Promote componentized architecture	Updated information on application architecture
Build automation	• Increase adoption of build automation tools and refine the build process • Implement automated build and track through dashboards • Define unit testing standards and automate testing	MED	Tool configuration and process improvement	HIGH	Provide continuous build \	Tool selection and implementation
Version control	• Streamline version control tools • Extend version control to requirements and test scripts • Define branching and merging policies for open systems	HIGH	Transition from existing tools	HIGH	Maintain code quality, reduce rework	Tool selection and implementation
Code Quality	• Increase adoption of code scanning tools and implement security testing tools. • Enable security governance and code signing • Develop, customize and publicly share code coverage and security dashboards	MED	Additional time and resource requirements	MED	Maintain code quality, reduce rework	Planning to include additional effort hours
Environment Provisioning	• Define environment provisioning process • Equip DevOps groups to implement quicker environment provisioning • Facilitate ad hoc environment provisioning for development	HIGH	Strategic decision on cloud	HIGH	Allow end to end continuous delivery	Strategic decision pending

Continuous Testing – Activity Detail

Continuous Testing

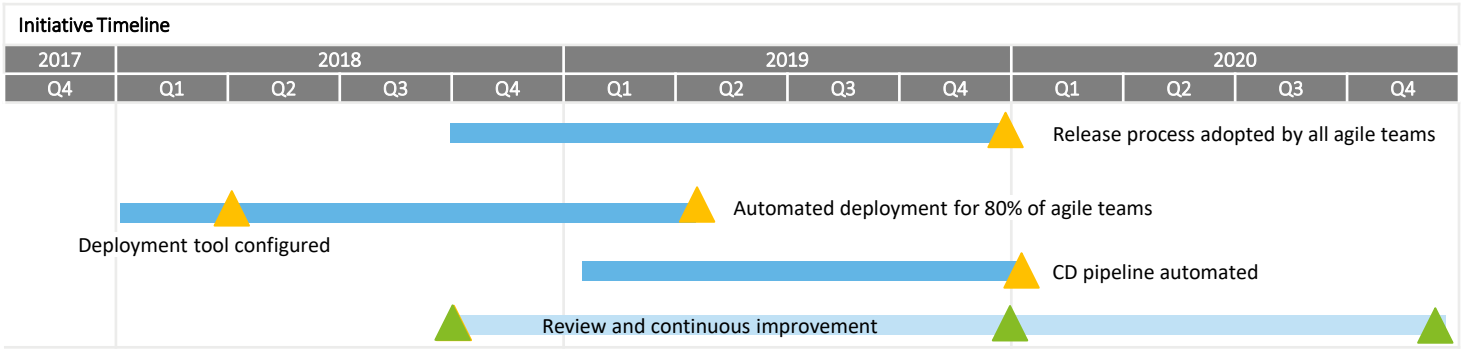


Success Indicators	
1	Functional testing standards defined by Q1 2018
2	Smoke and regression testing automated for 50% of teams by Q3 2018
3	Gold data set for testing available by Q3 2018
4	Automated test environment provisioning capabilities available by Q2 2019

Initiative	Description	Complexity		Value		Dependency
Testing Automation	• Define functional, non functional and specialty testing standards • Streamline testing tool stack and implement new tools • Automate Smoke, Regression, Integration and Performance testing . Increase adoption of security testing as part of the shift left process • Enable build triggered functional testing, • Define and implement service virtualization standards	MED	Large number of test scripts	HIGH	Continuous testing available	Feasibility of automating test scripts
Test data management	• Implement test data management • Develop gold data set of production like data for testing • Automate test data provisioning	MED	Tool configuration and process improvement	HIGH	Provide continuous build \	Tool selection and implementation
Test environment provisioning	• Standardize and automate the testing environment provisioning process	HIGH	Transition from existing tools	HIGH	Maintain code quality, reduce rework	Tool selection and implementation

Continuous Deployment – Activity Detail

Continuous Deployment

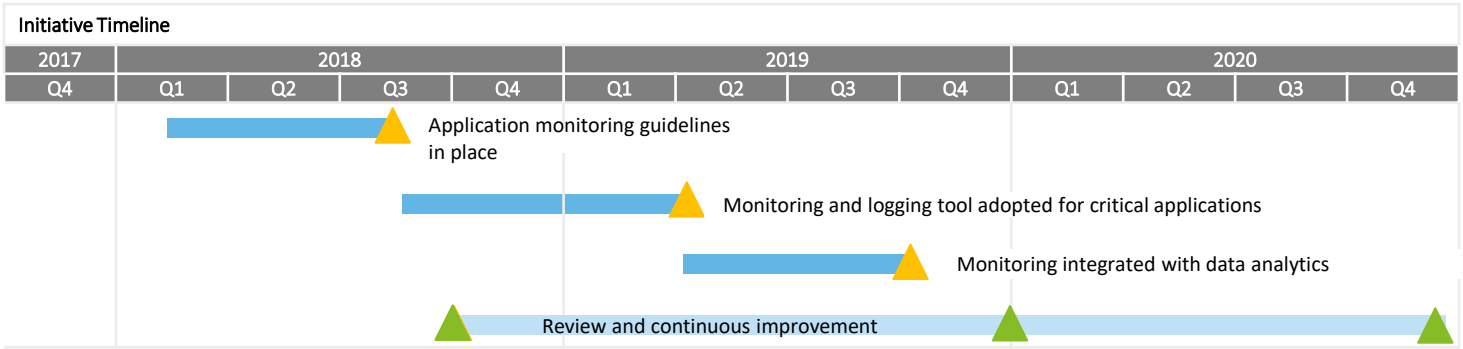


	Success Indicators
1	Release acceptance criteria defined by Q1 2019
2	Tool implemented for automated deployment by Q1 2018
3	CD pipeline orchestration started by Q2 2019

Initiative	Description	Complexity		Value		Dependency
Release management	• Streamline design, building and configuring of releases • Define release acceptance criteria and communication plans • Plan release coordination trainings for teams	MED	Align release process across teams	HIGH	Higher success rate for releases	Cross functional coordination
Deployment automation	• Define automated build standards • Increase adoption of tool for automated deployment • Integrate continuous build and testing tools with deployment tool stack	MED	Tool configuration and process improvement	HIGH	Continuous delivery without manual intervention	Tool selection and implementation
CD pipeline orchestration	• Implement automated process for code propagation initially on lower environments, followed by pre prod and production • Define automate workflows for CD pipeline • Integrate workflow tools with continuous deployment tools	HIGH	Transition from existing tools	HIGH	Continuous delivery without manual intervention	Tool selection and implementation

Continuous Monitoring and Feedback – Activity Detail

Continuous Monitoring and Feedback



	Success Indicators
1	Categories of application monitoring defined by Q2 2018
2	Monitoring tool procured by Q3 2018
3	Feedback loop and data analytics integrations begins by Q2 2019

Initiative	Description	Complexity		Value		Dependency
Application Monitoring	• Define monitoring guidelines across applications • Select and install application monitoring tool • implement application monitoring for selected applications • Incorporate server monitoring processes as part of monitoring • Enable automated threat detection and alert and self-healing	MED	Integrate cloud and on premise application monitoring	HIGH	Proactive incident management	Categorize of applications by application monitoring levels
Feedback	• Define guidelines and standards for root cause analysis • Enhance feedback loops • Generate and analyze monitoring dashboards for root cause analysis and incident management as a practice • Integrate monitoring systems with data analytics	HIGH	Continuous process is defined and adopted	HIGH	Proactive incident management	Resources and skills for RCA

DevOps Operating Model

Current State

Pekin's current state IT Operating model is a combination of the component centric and value chain centric model

Key Features:

- Leadership is horizontal and ownership is at the application/ component level (e.g., legacy, distributed systems)
- Approach to decision making is top down in most cases, participation from all stakeholders in initial planning is limited
- Individual teams are structured around the delivery value chain. Team responsibilities are divided between plan and building (i.e., BA, developers, QA) and run (i.e., deployment, database and infrastructure)
- The CoE supports project management for large and cross functional projects
- Solution architecture responsibilities are fulfilled by application SMEs or by developers
- EARB reviews and approves architecture design and implementation at defined project stages

	Component Centric	Value Chain Centric	
	IT Components	Plan/Build	Run
	IT Business Management	IT Business Management	
	Program Management	Program Management	
	Infrastructure & Shared Services	Infrastructure & Shared Services	
Description	▪ Organize by IT component ▪ Within each component, responsible for all elements of value chain	▪ Organize by IT Value Chain ▪ Organize by combination of Plan/Build/Run	
When the model is applicable	▪ Need to manage all resources within defined IT Service Components ▪ Resource constraints across value chain ▪ Need for increased responsiveness	▪ Effective with a clear enterprise IT platform strategy ▪ Need for separation of value chain and for increased accountability and monitoring of business value ▪ Useful as a foundational model for transformative IT organizations	
Pros	▪ Increased standardization ▪ Clear responsibility and accountability	▪ Centralized structure for establishing direction ▪ Enables evolution of new services into and from the IT organization ▪ Pushes down decision making	
Cons	▪ No coordination between business units ▪ Encourages silos ▪ Encourages a one size fits all approach	▪ Reduces flexibility ▪ Not fully aligned to multi-speed IT ▪ Too many layers	

Proposed Customer and Product Centric IT

Defining a Product

- A product transforms a market opportunity into a service or product available for sale, it may be tangible or intangible
- Product development may involve modification of existing product features, addition of new features or formulation of an entirely new product that satisfies a newly defined customer want or market niche
- Product development phases are conceptualization, design, development, marketing and management

Pekin Product View

P&C - Personal	P&C - Commercial	Life

Proposed Operating Model Changes

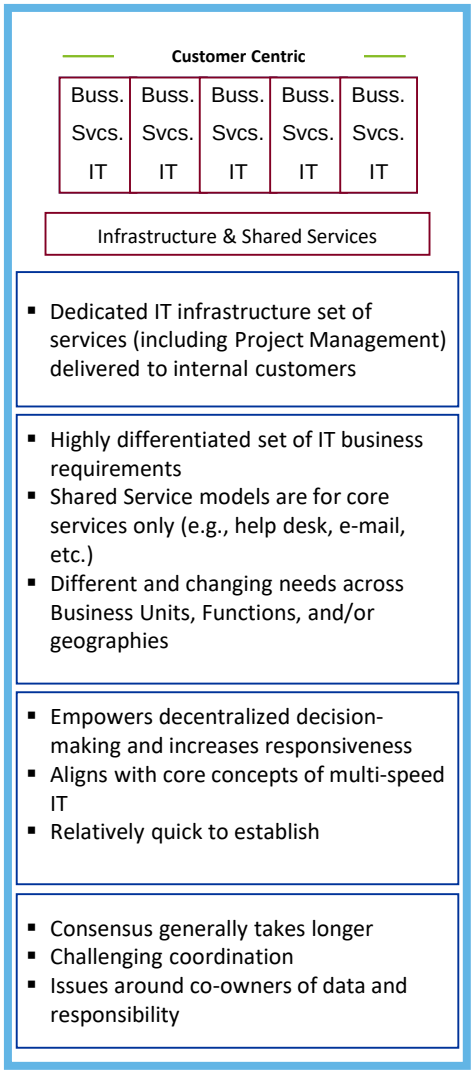
Based on Pekin's desired positioning in the marketplace, we propose a customer-centric IT Operating Model as the foundation for the Business Accelerator

A customer-centric approach is best aligned to Pekin's objectives of delivery with quality and speed:

- Ability to act and make decisions aligned to the business segments to empower customer-centric decision making
- Accountability to develop, deploy and support applications is centered on delivery teams
- Oversight and governance activities are streamlined and applied at enterprise level when necessary

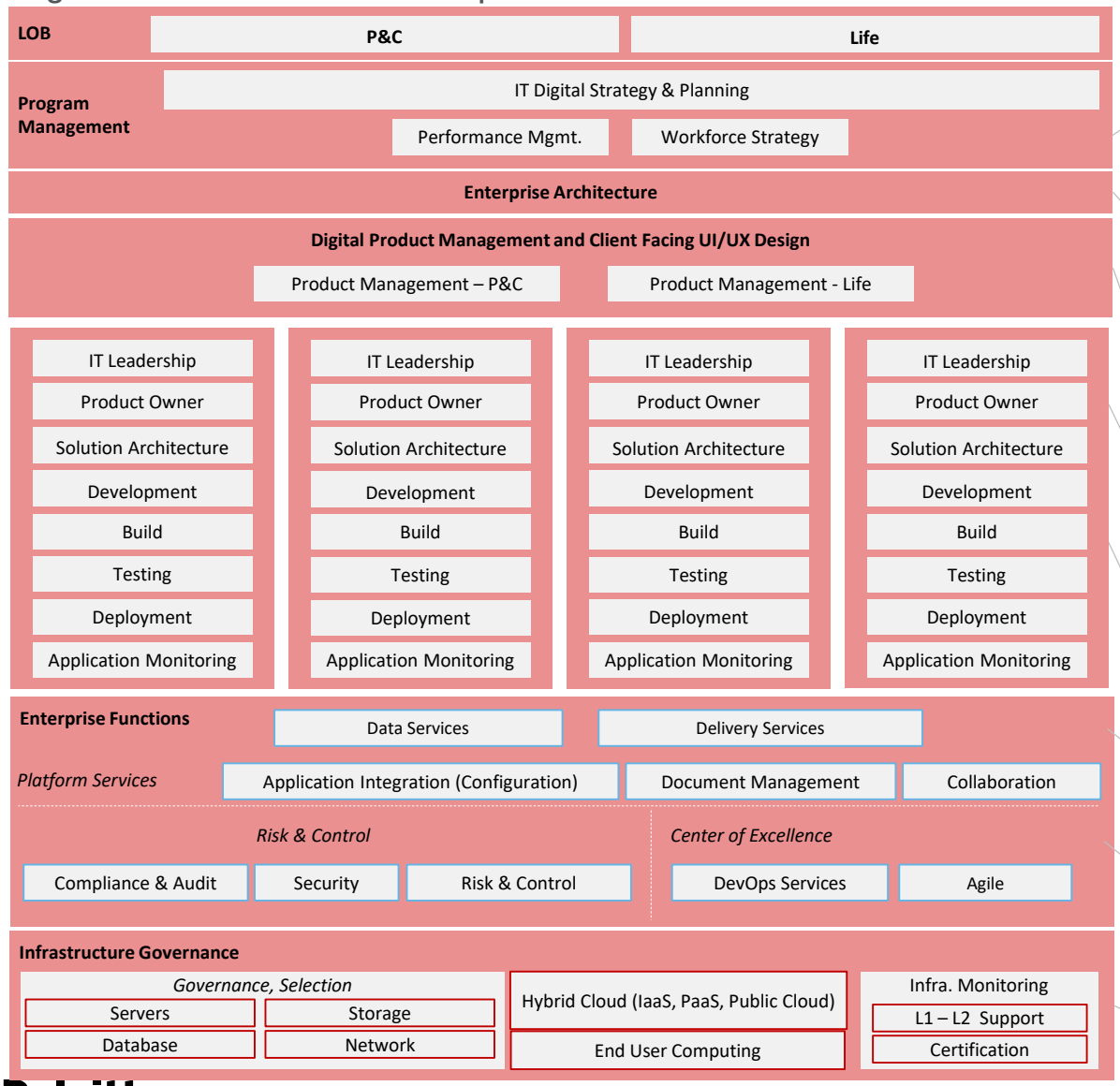
Proposed Changes:

- Teams are structured vertically to support individual lines of business
- Each team should be self contained and should include:
  - Product Owners
  - Solution architects
  - Developers
  - QA/Testers
  - Deployment and release resources
  - Operations
  - Application and infrastructure support group
- Center of Excellence (CoEs) are horizontal and cut across lines of business. CoEs define standards for agile delivery, release management, continuous delivery, cloud, etc.
- Enterprise services are made available for application integration, infrastructure governance (including cloud), security, and other roles



Proposed Operating Model – Function View

The function view of the proposed operating model summarizes high level activities and alignment across the enterprise.



Strategy
The enterprise project management office (PMO) enables realization of Business and IT strategy by prioritizing, budgeting, and planning investments and associated delivery. The group also manages the workforce mix, project metrics and reporting

Enterprise Architecture
Designing, governing, integrating, and managing technology architecture policies, provide guidelines to solution architects. Review project design and implementation through the EARB

Product Management and UI/UX Design
Provide customer/business requirements, align with the LOBs to provide agents/ customers with a consistent and unified user experience

IT Leadership
Manage and analyze the intake/ demand from the business, triage, assign work, and ownership of end to end application development

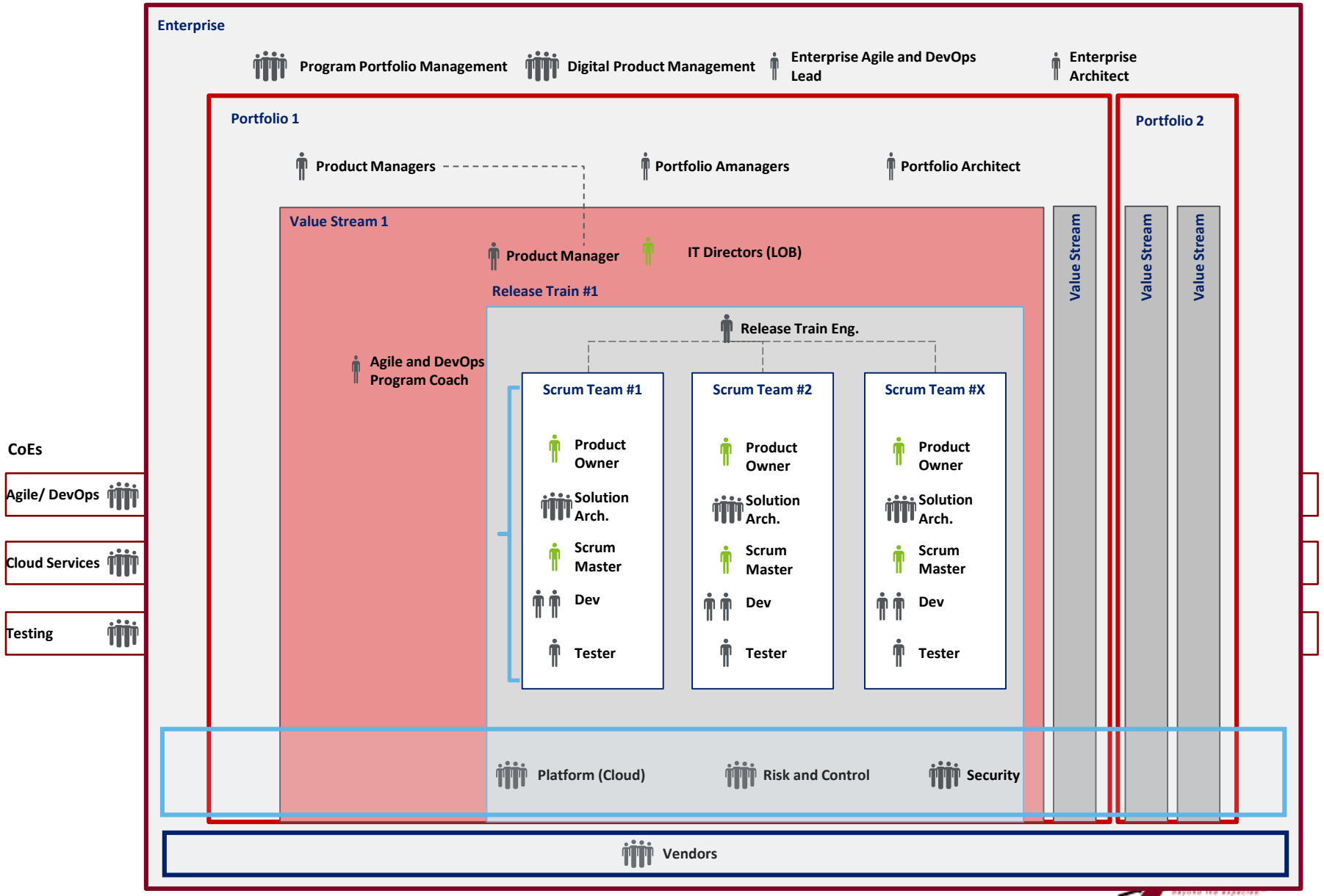
IT Delivery
Liaison with business to identify and analyze business requirements. Prioritize and translate requirements into solutions including associated development, build, test, deploy, release and run activities

Enterprise Functions
Provide streamlined enterprise functions and support for a successful product development

Centers of Excellence
Provide governance over standards, methods, tools, and performance metrics; confirm all teams are delivering according to best practice

Infrastructure
Provide cloud and infrastructure services to enable delivery teams to create and use environments for dev, test, and production

Proposed Operating Model – Role View



Proposed Operating Model – Roles Defined (1/2)

The roles identified to fulfill the target operating model are defined below

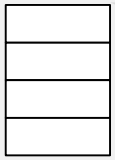
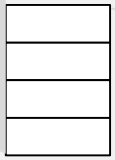
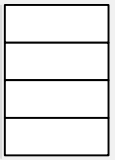
Level	Role	Role Description
Enterprise	Enterprise Agile Coach	Leads the agile CoE that defines processes and standards for agile delivery and coaches the program agile coaches
	Enterprise DevOps Coach	Leads the DevOps CoE that defines processes and tools for continuous delivery (development, build, test, deploy) and coaches the program DevOps coaches
Portfolio	Digital Product Manager and UI/UX Design	Product managers own the line of business and are the key provider of business requirements
	Enterprise Architect	Enterprise architects maintain a high-level, holistic vision of enterprise solutions and development initiatives for a given segment and provide guidance to solution architects and owns the EARB
	Portfolio Manager	Portfolio Managers have primary responsibility for strategy and investment funding, program mgmt., and IT and business governance within a line of business. Act as epic owners
	Infrastructure	Infrastructure provisions, configures and manages infrastructure components (compute, storage, networks and includes cloud)
	Platform Services	Platform services and tools provide application integration and collaboration
	Risk and Control	Risk and control organizations understand the risks the organization is exposed to, and put controls in place to counter threats
	Security	Security defines and enforces enterprise wide policies for protecting infrastructure, information assets, customer data, financial information and other critical IT information
Value Stream	IT Director	The director is accountable for end to end IT delivery and operations specific to a value stream
Release Train	Product Manager	Product manager are responsible for defining and communicating the program vision and backlog. Product managers participate in UAT.
	Release Train Engineer	RTE facilitates program level processes and execution, escalates impediments, and drives release train to continuous improvement

Proposed Operating Model – Roles Defined (2/2)

The roles identified to fulfill the target operating model are defined below

Level	Role	Role Description
VS, ART, scrum team	Program Agile Coach	An Agile coach identifies and coaches on Agile best practices and principles, contributes to the agile CoE and aligns on standards
	Program DevOps Coach	A DevOps coach identifies and coaches teams on continuous delivery processes and tools, contributes to CoE and aligns on standards
Scrum Team	Product Owner	The Product Owner has the vision of what a scrum team intends to build and communicates that vision to team. The PO performs analysis and interfaces with the business.
	Solution Architect	SAs have the technical responsibility for the overall architecture of the solution or an application and ensure it aligns with the enterprise architecture
	Scrum Master	The Scrum Master is responsible for conducting the ceremonies of the scrum team and works to remove any impediments to the development efforts
	Developers	Developers work with the shared services and infrastructure to develop functionality in accordance with the user story. In addition are responsible for monitoring applications in all environments and ensure built in quality
	Testers	Testers create functional, performance, and security test scripts and monitor testing execution as part of the continuous integration process
Center of Excellence (CoE)	Agile	The agile CoE defines processes and standards, ensures agile maturity and defines measures of success
	DevOps	The continuous delivery CoE defines processes and owns tools for continuous delivery, ensures DevOps maturity and defines measures of success
	Cloud Services	The Cloud Services CoE provides governance, standards and coaching on the use of cloud as infrastructure and use of cloud services
	Testing	The testing CoE establishes testing/ quality assurance processes and ensures adoption of the processes
	Vendor	A *vendor* is an internal or external organization that develops and delivers components, subsystems or services that help Solution Trains provide Solutions to their customers.

Taxonomy of Work

Portfolio / Program Backlog		Release Backlog	Sprint Backlog
			
Portfolio Epic (Years)	**Program Epic (Features)** (Quarters)	**User Story** (Days)	**Sub-Task** (Hours)
• Long term initiatives that align to the strategic business objectives/ themes • Has measureable target performance metrics to determine realization of business value	• Medium term initiatives that enable the implementation of the Portfolio Epic • Enable iterative delivery of business value desired by the Portfolio Epic • Has measureable target performance metrics to determine realization of business value	• Small, independently valuable work items that make up an Epic and can be delivered by one Tech Product • Will need to meet the Tech Product Team's Definition of Ready (DoR) to be worked on in a sprint	• Activities needed to complete an User Story • May be repetitive "chores" to support or maintain the product
Bounded by: Enterprise/ Portfolio	**Bounded by:** Portfolio Manager	**Bounded by:** Product Owner	**Bounded by:** Team
Owned by: Portfolio Manager/ Product Manager	**Owned by:** Product Manager	**Owned by:** Product Manager	**Owned by:** Product Owner

DevOps Tools

DevOps Tools Ratings (1/2)

Tool Name	Overall Rating	Technical Competence	Opportunity to leverage	Solution fit to expectations	Integration with other tools	Learning Curve	Migration cost	Migration time	Operating cost	Delivery Model	Ease of use
AppDynamics	26	3	2	3	3	3	0	3	3	3	3
APPLOADER	19	3	2	3	2	2	0	0	2	3	2
Atlassian Bamboo	25	3	2	2	3	3	3	3	2	1	3
Atlassian BitBucket (GIT)	25	3	2	2	3	3	3	3	2	1	3
Atlassian BitBucket Cloud	28	3	2	3	3	3	3	3	3	2	3
Atlassian Crucible	17	3	1	1	1	1	3	2	2	1	2
BluePrint (ENTERPRISE)	21	2	2	3	2	3	2	2	2	1	2
CA Test Data Manager	22	3	2	3	3	2	2	2	2	1	2
Chef	22	3	3	3	3	2	1	2	2	1	2
Confluence	26	3	2	2	3	3	3	2	2	3	3
ContrastSecurity	24	3	3	3	3	2	0	2	2	3	3
Delphix for Test Data Management	26	3	3	3	3	2	2	2	2	3	3
Doors	12	3	2	1	0	1	0	1	1	1	2
Dynatrace	30	3	3	3	3	3	3	3	3	3	3
Evident.io	23	3	2	3	3	2	0	2	2	3	3
Gatherspace (IND CONTRACTOR)	15	2	3	2	1	1	1	1	1	1	2
HP AgileManagement Tool	17	3	1	2	2	2	1	2	1	1	2
HP ALM	17	3	1	2	2	2	1	2	1	1	2
Jama (ENTERPRISE)	18	3	3	2	1	3	1	1	1	1	2
Jenkins	28	3	3	3	3	3	3	2	2	3	3
Jfrog Artifactory	30	3	3	3	3	3	3	3	3	3	3
JIRA	29	3	3	3	3	3	3	2	3	3	3
New Relic	23	3	3	3	3	3	0	0	3	2	3
Nexus Repository Manager	26	3	3	3	3	2	2	3	3	1	3

DevOps Tools Ratings (2 / 2)

Tool Name	Overall Rating	Technical Competence	Opportunity to leverage	Solution fit to expectations	Integration with other tools	Learning Curve	Migration cost	Migration time	Operating cost	Delivery Model	Ease of use
Nexus Repository Manager	26	3	3	3	3	2	2	3	3	1	3
Office 365 Application PPM	30	3	3	3	3	3	3	3	3	3	3
Puppet	24	3	3	3	3	2	2	2	3	1	2
Rational Performance Tester	15	3	1	2	1	2	2	0	2	1	1
Rational Tool Suite (RFT, RQT, etc))	15	3	1	2	1	2	2	0	2	1	1
Selenium	26	3	3	3	3	3	2	3	2	1	3
ServiceNow	25	3	3	3	3	3	0	2	3	2	3
Solarwinds	19	3	3	2	1	2	3	0	2	1	2
Sonarqube	26	3	3	3	3	2	3	0	3	3	3
TeamCity	26	3	3	3	3	3	0	2	3	3	3
Terraform	27	3	3	3	3	3	0	3	3	3	3
Vagrant	19	3	1	1	3	2	0	2	2	3	2
Veracode	18	3	3	3	1	1	2	0	2	1	2
Visual Studio Team Services	30	3	3	3	3	3	3	3	3	3	3
VMWare vRealize	18	3	2	2	1	2	1	0	2	3	2

Deloitte.

Recommended tool

PEKIN INSURANCE

DevOps Tools Estimated Cost

Target State Tool Stack Cost Details						
			2018		2019	2020
			License	Migration	License	License
Capability	**Key Area**	**Tool Name**				
Continuous Planning	Knowledge Sharing & Community	Sharepoint	$ -	$ -	$ -	$ -
Continuous Planning	Requirements:	HP ALM*	$ -	$ -	$ -	$ -
Continuous Planning	Work & Issue Tracking	HP AGM*	$ -	$ -	$ -	$ -
Continuous Build	Source Code Management	Bitbucket	$ -	$ -	$ -	$ -
Continuous Build	Continuous Integration	TeamCity	$ -	$ -	$ -	$ -
Continuous Deployment	Deployment Automation	TeamCity	$ -	$ -	$ -	$ -
Continuous Deployment	Continuous Delivery	TeamCity	$ -	$ -	$ -	$ -
Continuous Monitoring and Feedback	Application and Project Portfolio Management	HP AGM*	$ -	$ -	$ -	$ -
Continuous Monitoring and Feedback	Workflow Management and Approval Automation	ServiceNow	$ -	$ -	$ -	$ -
Continuous Monitoring and Feedback	Configuration Management	Chef	$ -	$ -	$ -	$ -
Continuous Build	Build Automation	Maven (for Java)	$ -	$ -	$ -	$ -
Continuous Build		MsBuild (for .Net)	$ -	$ -	$ -	$ -
Continuous Deployment	Repository for Binaries & Containers	Nexus	$ -	$ -	$ -	$ -
Continuous Testing	Test Case Defect Mgt	HP ALM, AGM*, ServiceNow	$ -	$ -	$ -	$ -
Continuous Testing	Test Automation	Selenium	$ -	$ -	$ -	$ -
Continuous Testing	Code Quality	Crucible, Codenarc	$ -	$ -	$ -	$ -
Continuous Testing	Test Data Management	Delphix for Test Data Management	$ -	$ -	$ -	$ -
Continuous Testing	Performance Testing	HP LoadRunner*	$ -	$ -	$ -	$ -
Continuous Testing	Application Security	ContrastSecurity	$ 200,000.00	$ -	$ 220,000.00	$242,000.00
Continuous Monitoring and Feedback	Monitoring & Feedback	DynaTrace	$ 10,000.00	$ -	$ 11,000.00	$ 12,100.00
Total			**$ 210,000.00**	**$ -**	**$ 231,000.00**	**$254,100.00**
Assumed increase in product licensing cost per year	***10%***					