
accenture technology
Key Accenture Learnings on Scaled and Distributed Agile Delivery
Andrew Ball, Ajay Nair & Mirco Hering
High performance. Delivered.
Strategy | Digital | Technology | Operations

# Focus for today

- Many companies struggle to implement Agile at *scale* when they have *distributed* workforces.
- Accenture has a range of experience in implementing large-scale, distributed Agile programs using the Scaled Agile Framework.
- The intention of today's session is to give some insights into Accenture's solutions for implementing *scaled* and *distributed* Agile.

# Challenges for distributed and scaled Agile

# Challenges for distributed and scaled Agile

Accenture has been able to leverage *Scaled Agile Framework* and a range of tools to address a number of these challenges

## Process

- Solution misalignment between teams
- Timeline misalignment between scrum teams
- Integration of Agile with Waterfall
- Status tracking misalignment

## Organization

- Different locations
- Different working hours
- Different holidays
- Different customs
- Cross-team activities

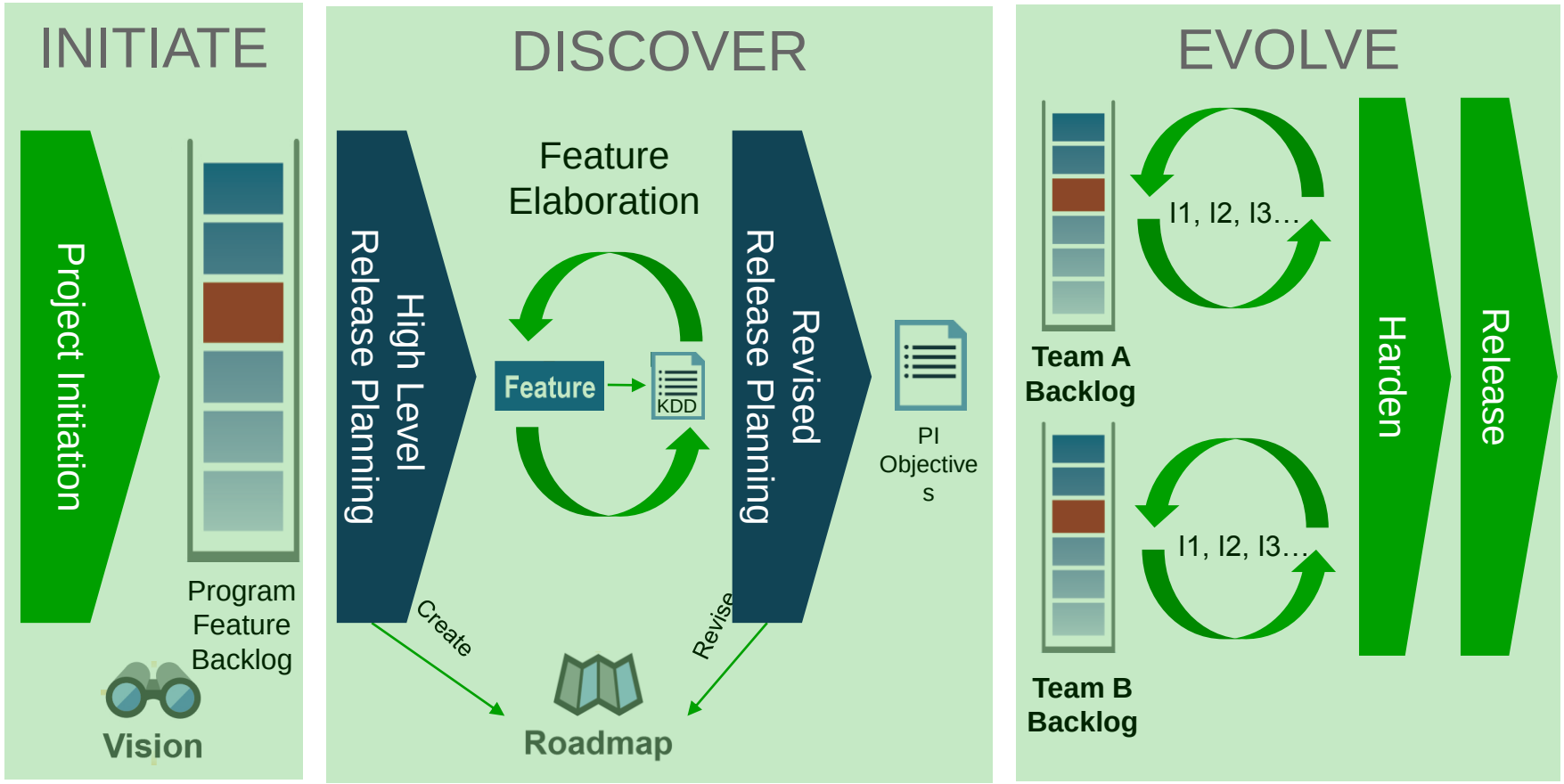
## Tools

- Different DevOps Tools between teams
- Some tools don’t scale well
- Offshore system performance
- Poor collaboration technologies

# Suggested solutions for these changes

# Process – Aligning solutions between teams

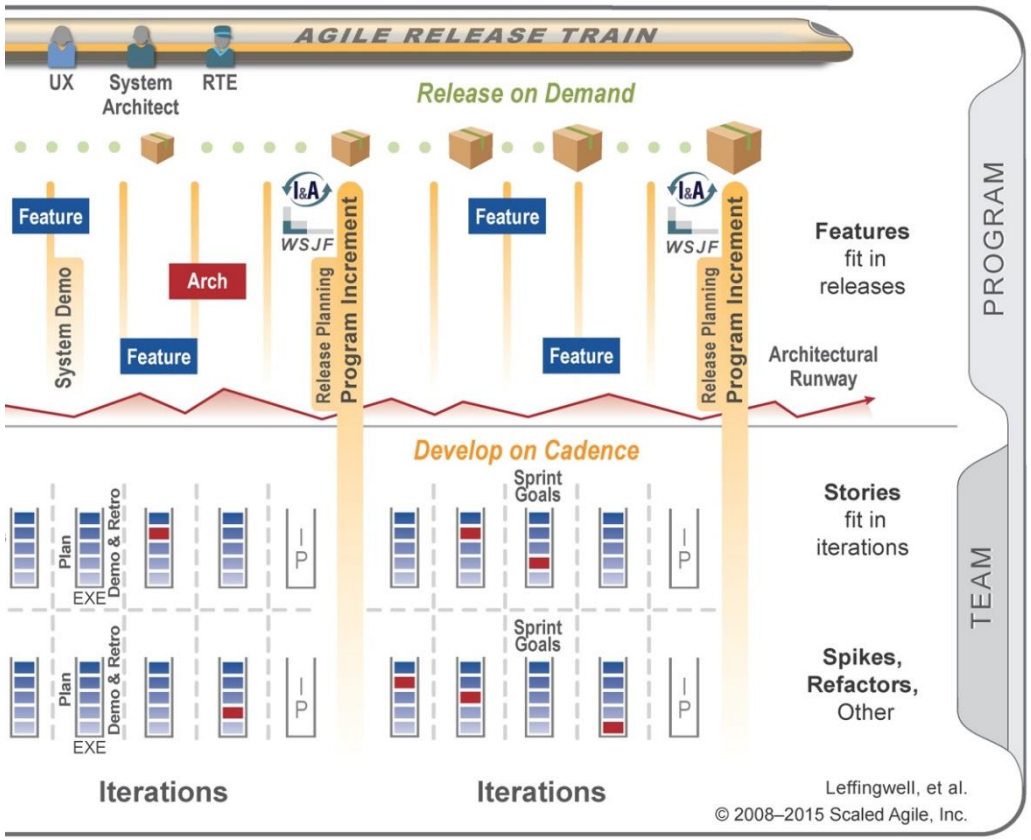
Enhanced SAFe processes are key to attaining solution alignment between different scrum teams



# Process – Ensuring timeline alignment between teams

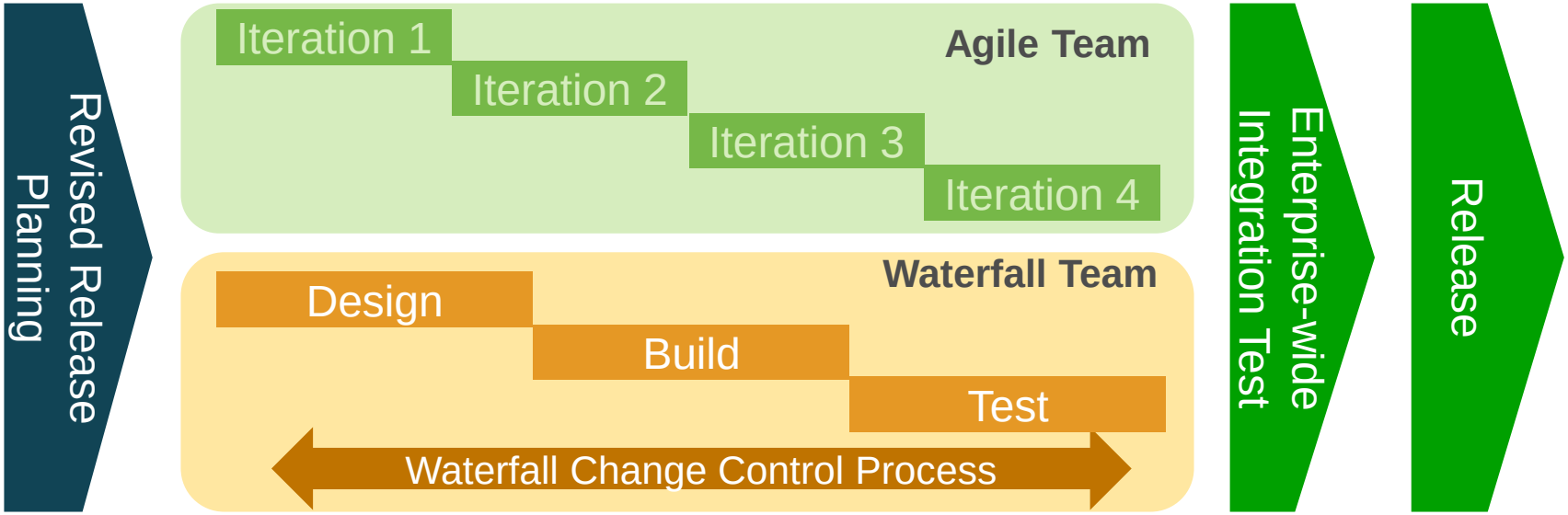
SAFe is critical to the alignment of delivery timelines

- Synchronize iterations
- Group iterations into common Program Increments (PIs)
- Aim for synchronized iterations and PIs across enterprise



# Process – Integrating Agile and Waterfall

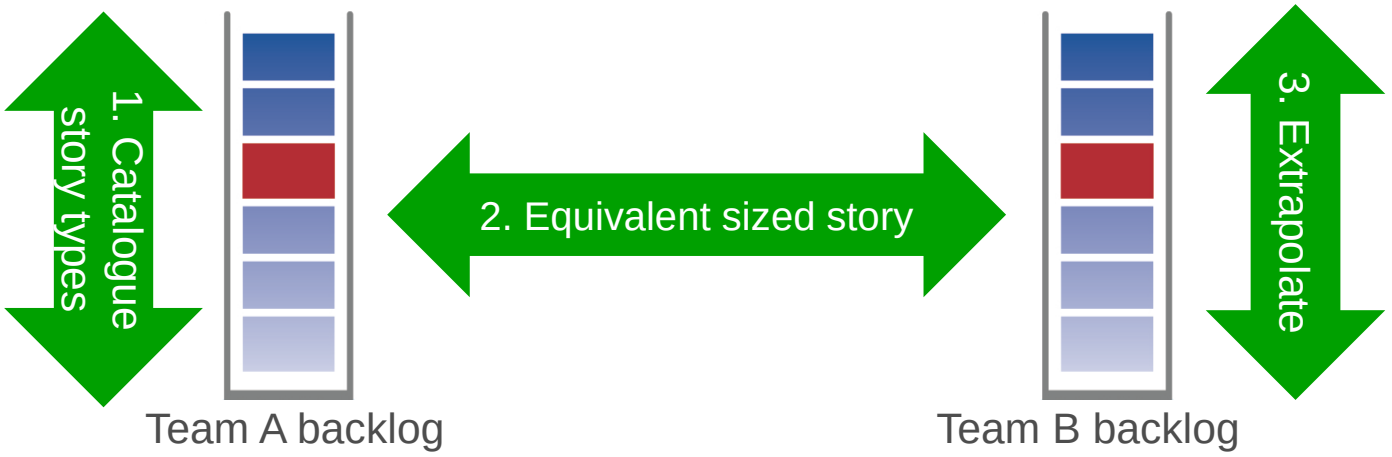
- Upfront identification of features/scope requiring Agile and Waterfall delivery
- Synchronize these activities for impacted features:
  - Discovery (Agile) and Solution Definition (Waterfall)
  - Release Planning
  - Integration Test
  - Release
- Manage any further changes in release via change control process for Waterfall team
- Consistent tooling and common DevOps team



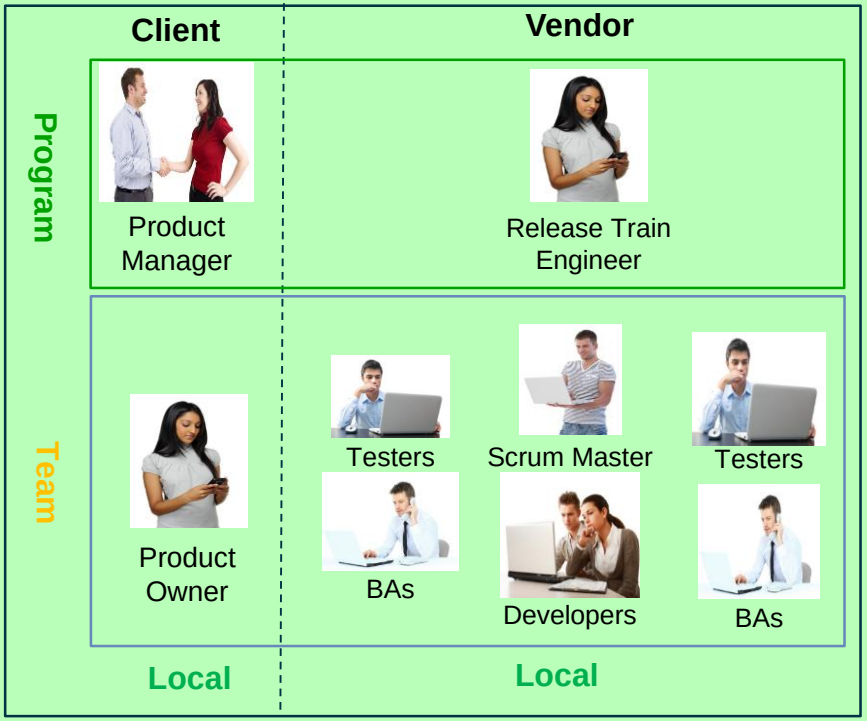
# Process – Status tracking alignment

Aligned tracking across scrum teams requires:

- Consistent definition of ‘Done’
- Consistent sizing method through the use of:
  - Ideal hours
  - Standardized story points – this requires:
    1. Creating a catalogue of story types for each team
    2. Agree on equivalent stories across teams
    3. Using equivalent story types as guides, extrapolate sizes for other story types

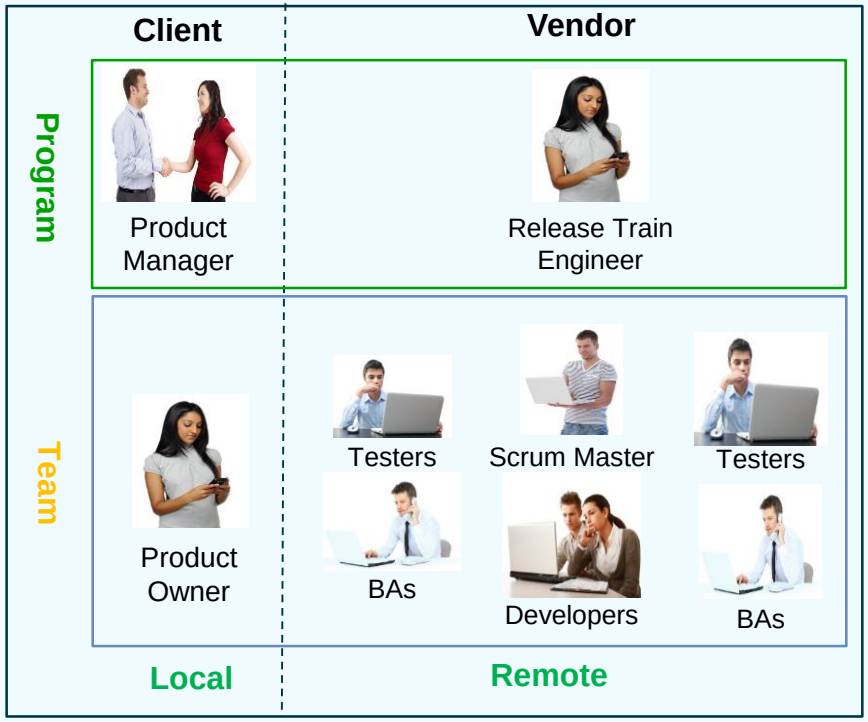


# Organization – Possible different structures




**Selection Criteria**

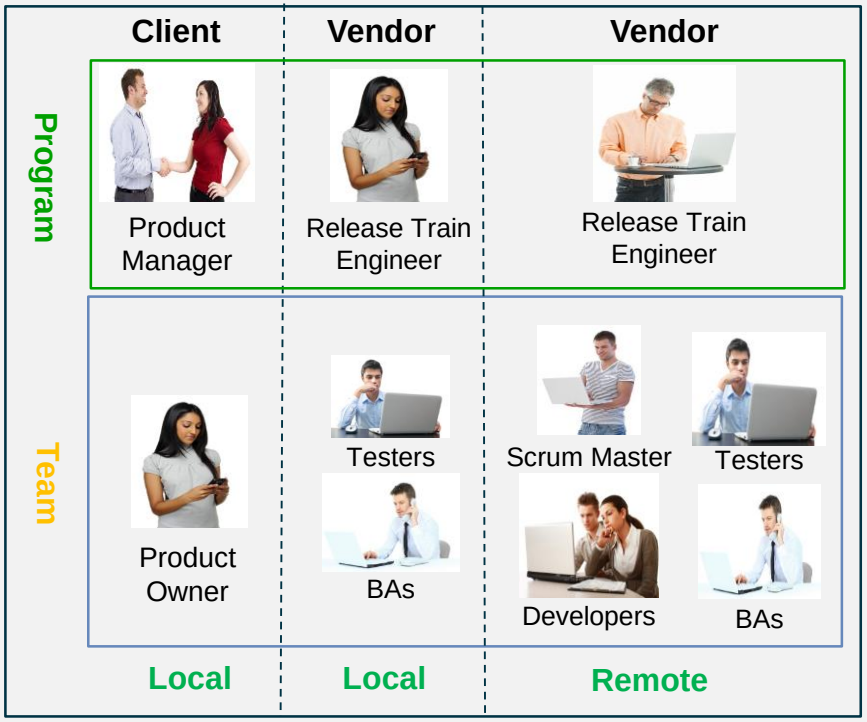
- Business requirements are too critical or confidential to be offshored
- Team has limited knowledge of business functionality
- Limited availability of Business Users/Product Owner
- Adequate budget




**Selection Criteria**

- Simple business requirements
- Remote team has good knowledge of business functionality
- High availability of Business Users/Product Owner

# Organization – Possible different structures




**Selection Criteria**

- Business requirements are complex or not very well-defined
- Remote site team has limited knowledge of business functionality
- Limited availability of Business Users/Product Owner

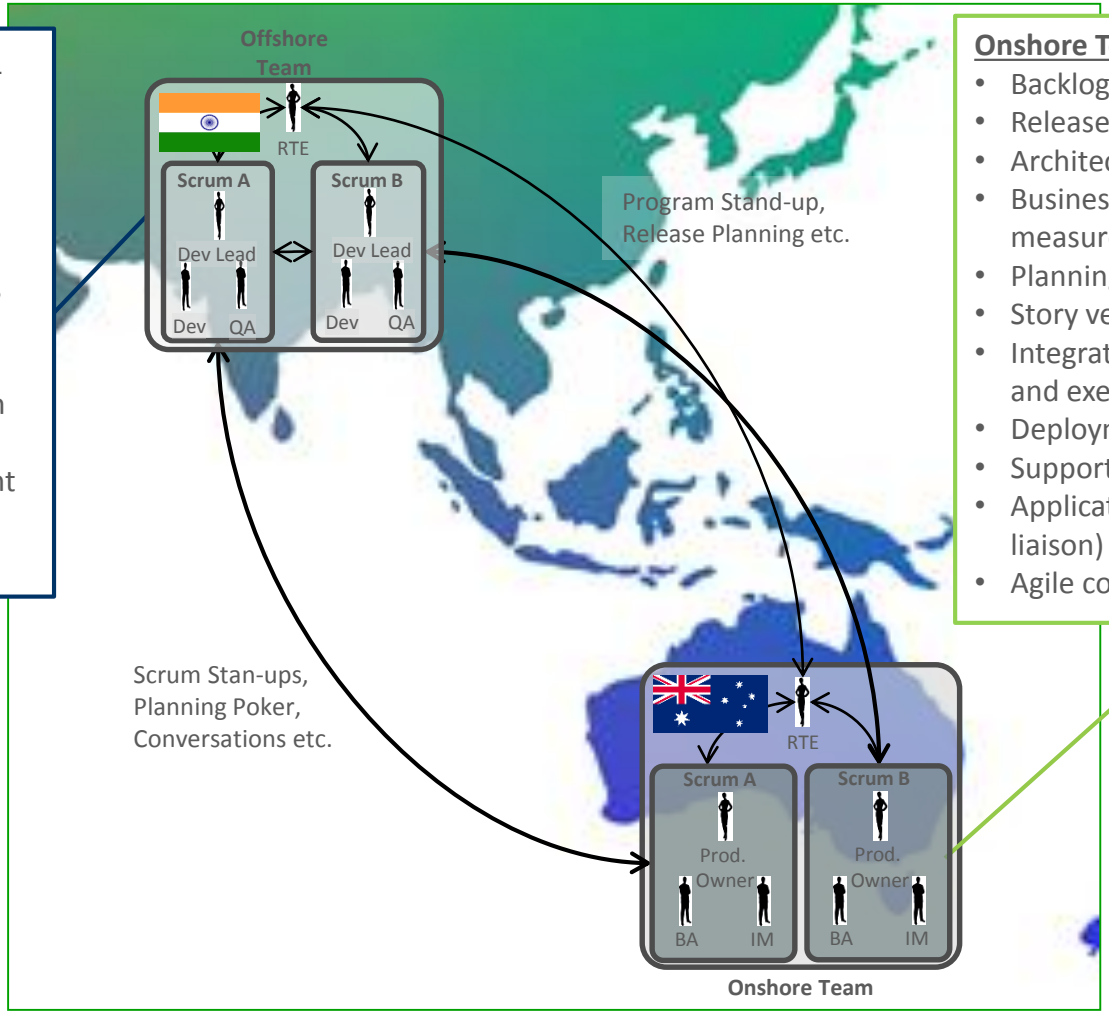

**Selection Criteria**

- Business requirements are complex and not well-defined
- Remote team has limited knowledge of business functionality and availability of Business Users/Product Owner
- Development of some application areas cannot be shifted to remote locations

# Organization - Example Activity Split

**Offshore Team Activities:**

- Release Planning input
- Planning Poker and estimation
- Build and unit test stories
- Application test stories
- Test planning, preparation and execution – Integration Test, SVT and ORT
- Application deployment
- Application support

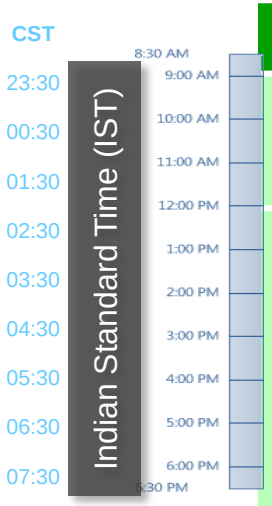



**Onshore Team Activities**

- Backlog definition
- Release Planning
- Architecture
- Business value definition and measurement
- Planning Poker and estimation
- Story verification
- Integration (E2E) test planning and execution support
- Deployment support
- Support readiness
- Application support (client liaison)
- Agile coaching and training

# Collaboration in time zone

## Onshore - Offshore model provides the ability for the team to work around the clock

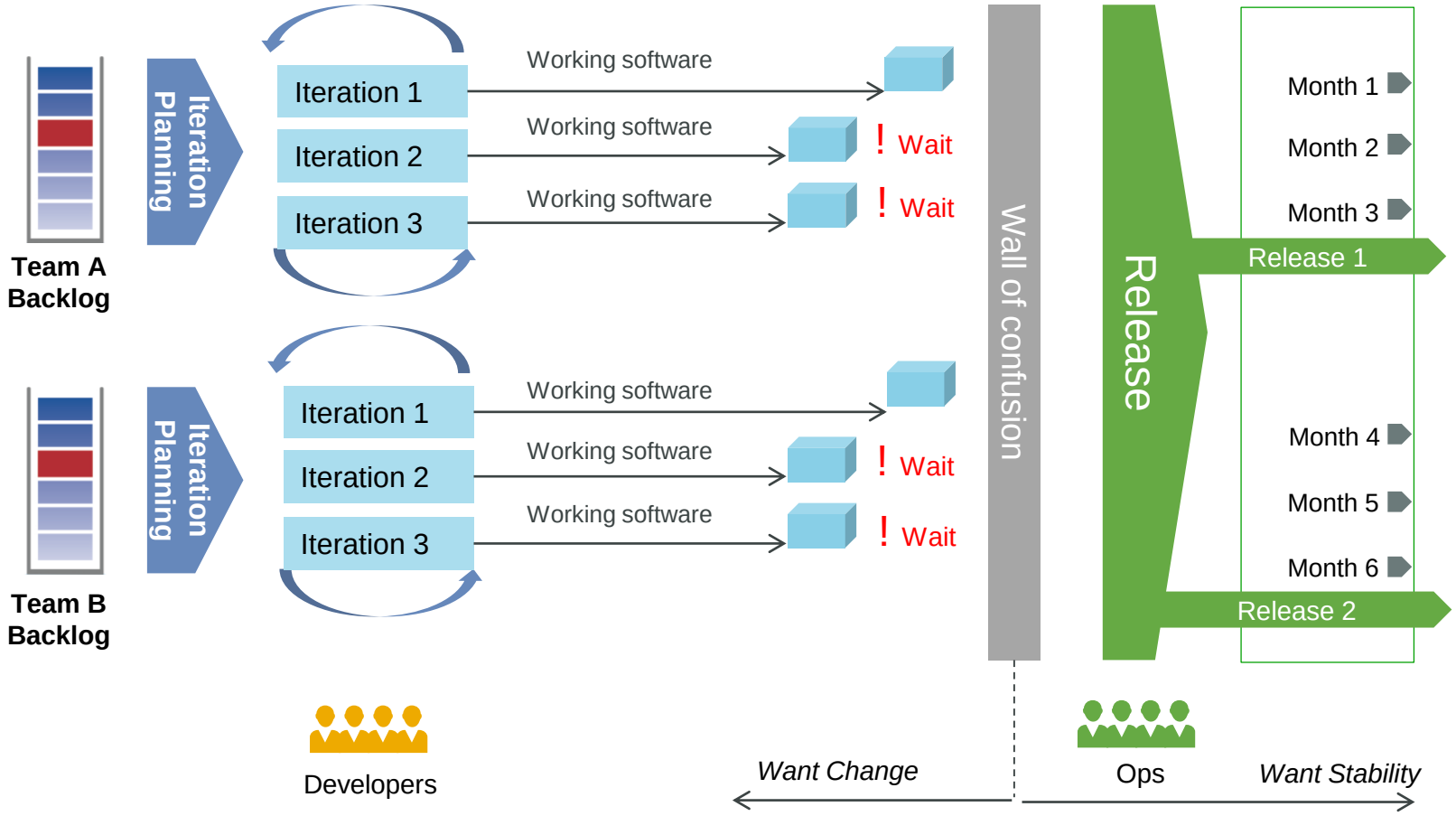



| Time | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| **Start of Day Indian Standard Time (IST)** | Offshore team connects with onshore UX Lead receive inputs from SMEs | | | | |
| **Offshore Factory Workday** | Offshore team works on UX planning and execute UX delivery and delivery management activities.<br>Peer review of UX deliverables.<br>Uses Vr controlled shared location to park the deliverables. | | Use virtual collaboration tools (tele-conference, Lync screen sharing, video conferencing) to facilitate standup meetings between distributed teams. | | |
| **End of Day IST / Start of Day US** | Standing call on project update | Standing call on project update Offshore Factory/US Team Meeting | Standing call on project update | Offshore Factory/ US Team Meeting | Standing call on project update |
| **US Workday** | Work on UX delivery planning activities<br>Review user stories and query logs with SME<br>Review session with Business Partners and Stakeholders<br>Work on UX deliverables | | | | |
| **End of Day US** | Provide next day instructions to offshore UX team | | | | |

# Agile & DevOps

# Why DevOps – example in an Agile delivery context

## Adopting DevOps principles compliments Agile delivery to enable faster delivery and speed to market



# DevOps principles

DevOps uses automation techniques to optimize collaboration across development and operations, enabling faster, more predictable and more frequent deployments to market

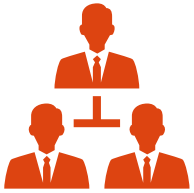
Strong Source Control

Deliver small increments

Automate everything

Test early and often

Experiment frequently

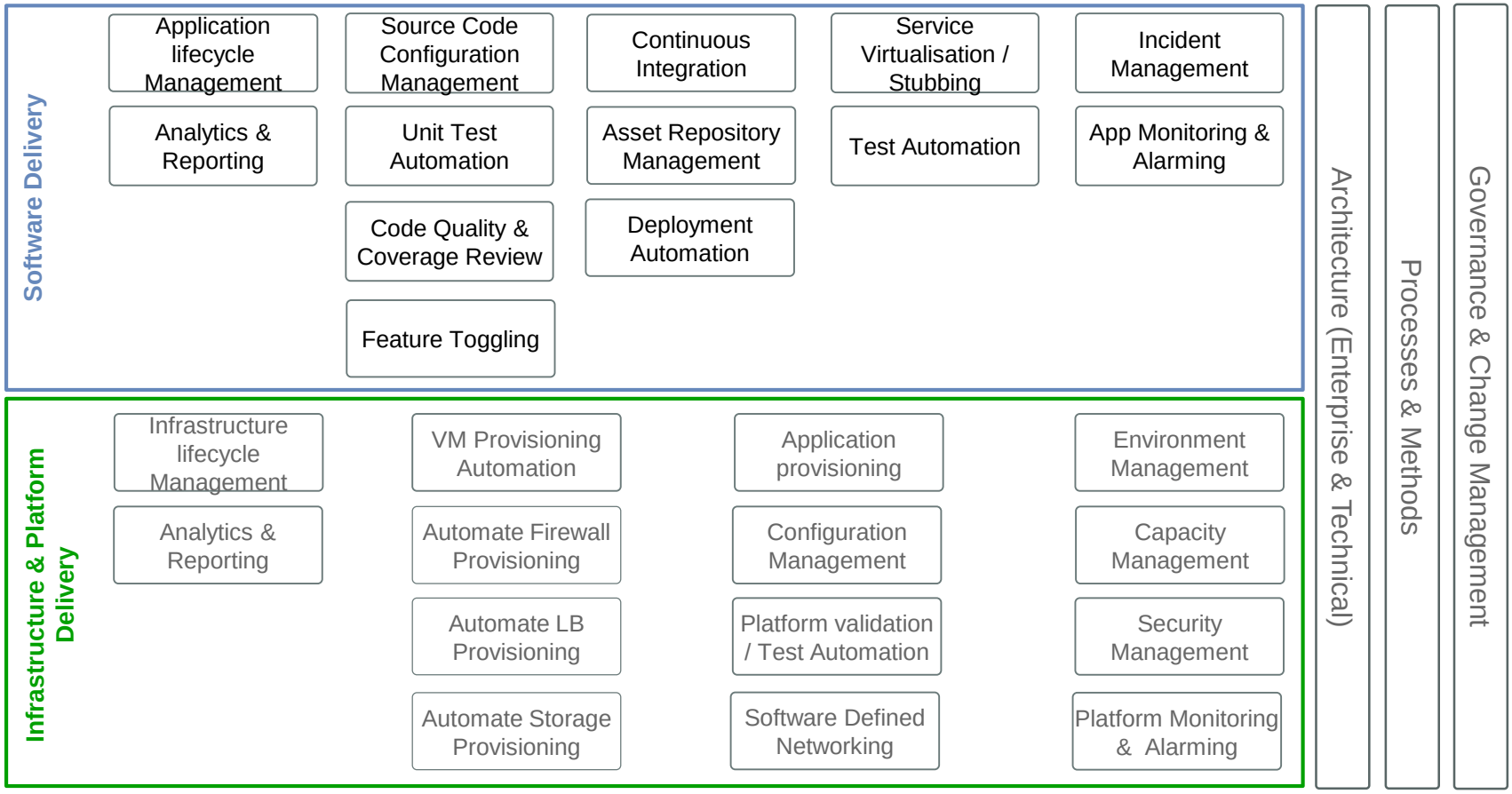


Cohesive teams

Improve continuously

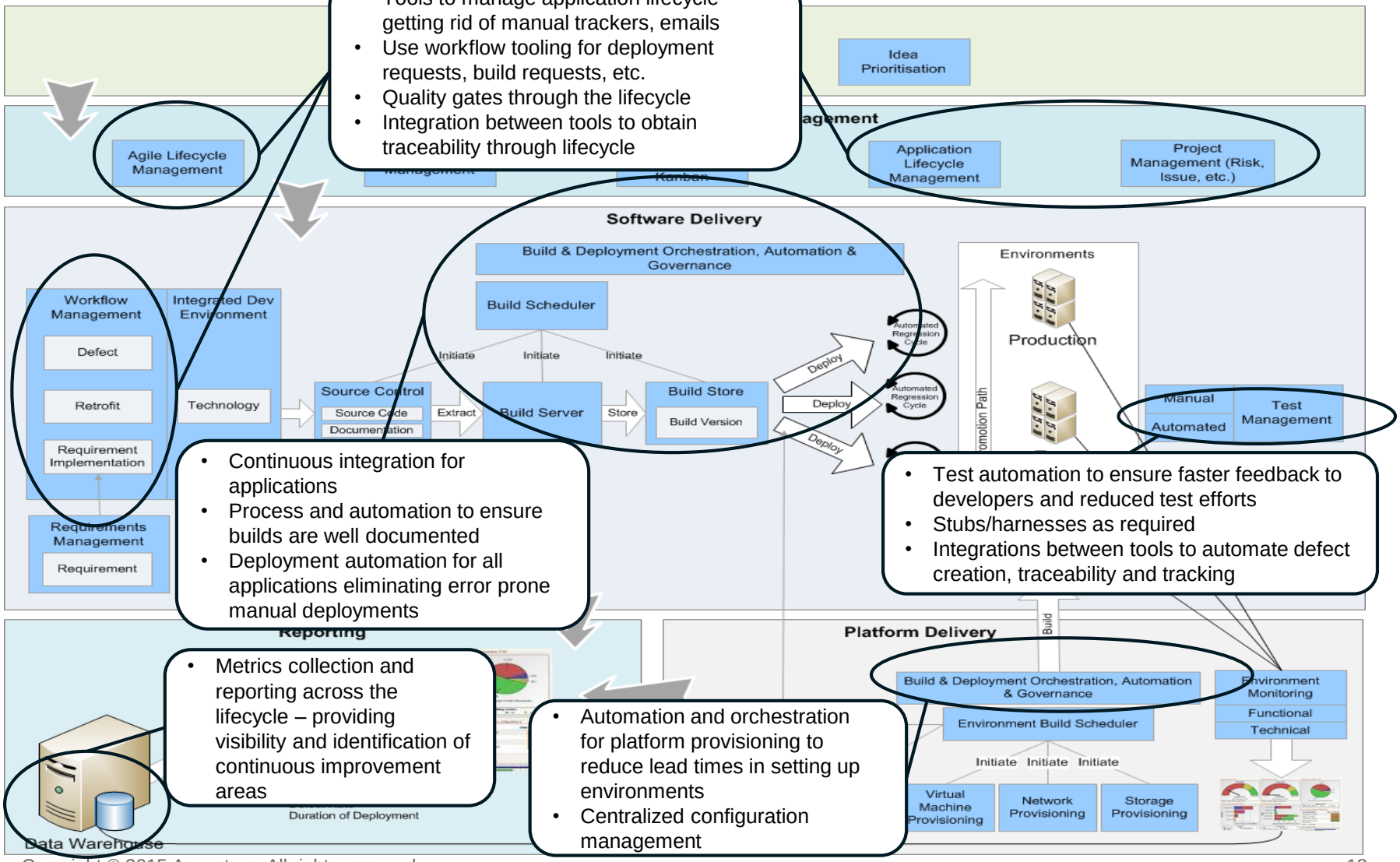
# Key building blocks

## Some of the key building blocks in moving to a DevOps model are below




Increase in maturity can be via further improvements within each capability or targeting additional capabilities

# Example Reference Architecture




- Tools to manage application lifecycle getting rid of manual trackers, emails
- Use workflow tooling for deployment requests, build requests, etc.
- Quality gates through the lifecycle
- Integration between tools to obtain traceability through lifecycle

- Continuous integration for applications
- Process and automation to ensure builds are well documented
- Deployment automation for all applications eliminating error prone manual deployments

- Test automation to ensure faster feedback to developers and reduced test efforts
- Stubs/harnesses as required
- Integrations between tools to automate defect creation, traceability and tracking

- Metrics collection and reporting across the lifecycle – providing visibility and identification of continuous improvement areas

- Automation and orchestration for platform provisioning to reduce lead times in setting up environments
- Centralized configuration management

# Benefits

# Early benefits

## Quantitative Benefits

| Activity | Measured as | % Improvement | Comments |
|---|---|---|---|
| **Merge and retrofit** | % of the build effort across releases | 50% | • Based on the actual effort tracked |
| **Software configuration management** | Effort to support SCM activities | 63% | • Elimination of manual trackers [workflow lists, baselines, objects]<br>• Maintenance of two tools [CVS, CC], scorecards and run books not required<br>• Unnecessary environment sync-up eliminated |
| **Cost of poor quality** | % of defects attributed to SCM, deployment | 59% | • Elimination of defects introduced because of previous<br>• Inefficient SCM processes<br>• Incorrect deployments |
| **Build and deployment** | Process and effort to raise deployment requests | 90% | • No. of deployment requests (DR) reduced - enterprise build [not incremental or individual builds]<br>• Effort per DR reduced because DR not in spreadsheets [now raised on tool – RTC] and elimination of extra step in the flow of DR |

## Qualitative Benefits

- Improved demand management and traceability from portfolio through to Agile delivery teams
- Granular configuration management and traceability
- Integration with agile lifecycle tooling to allow story based configuration management driven from meta data
- Real-time traceability of status for build and deployment
- Automated build and deployments – “one-button deployment”
- Developer efficiencies as a consequence of improved tool interaction times, processes