TOP DEVOPS BOTTLENECKS, CONSTRAINTS AND BEST PRACTICES

Speakers:

JP Morgenthal, CSC (@jpmorgenthal)

Mike Kavis, Cloud Technology Partners (@madgreek65)

Agenda

- **DevOps Perspective**

- **Top 5 Bottlenecks**
- 1) Inconsistent environments (Mike)
- 2) Long provisioning times (Mike)
- 3) Doing more with less (Mike)
- 4) Manual gates
- 5) Organization silos

- **Top 5 Constraints**
- 1) Auditing & Compliance
- 2) Technical debt
- 3) Misaligned incentives (Mike)
- 4) ITIL / Change Control (Mike)
- 5) Lack of metrics (Mike)

- **Top 5 Best Practices**
- 1) Governance / Security (Mike)
- 2) Continuously Deploy into UAT
- 3) Reduce Work-in-Progress / Deliver more frequently with less features
- 4) Fail Fast
- 5) Testing Center of Excellence

- **Top 5 Things You Can Do Starting Today**

CSC Cloud Technology Partners

DevOps Perspective

- Why DevOps?
  - How do we become a High Performing Culture?
- Leads to…
  - Better products and services
  - Increased customer satisfaction
  - Improved profitability
- DevOps is not…
  - A person, role, or team
  - A fix for IT

Definitions

- Bottleneck
  - Something that hinders flow and progress of activities, but does not limit completion. Bottlenecks can be eliminated or work-arounds can be instituted
- Constraint
  - An activity or rule that must be adhered to in order for progress to continue in an approved manner
- Best Practice
  - An activity or guideline that has demonstrated to foster improvement in continuous delivery activities

Bottleneck: Inconsistent Environments

- Creates unnecessary defects, rework, lowers quality and reliability, and increases risks of missing commitments

- .Reasons for inconsistency
  - Manual intervention
  - Lack of asset tracking
  - Poor patching process

- Mitigation
  - Automation
  - Configuration management
  - Immutable infrastructure
  - Infrastructure as code

Bottleneck: Long Provisioning Times

- Creates stoppage of WIP for long periods of time
- Impacts of wait time
  - Work stoppage between dev, test, and stage activities
  - Inconsistent environments
  - Sub optimal infrastructure
  - Project delays
- Mitigation
  - Automation
  - Immutable infrastructure
  - Modernize request management services

Bottleneck: Doing More With Less

- Technology teams are under intense pressure to deliver more features with greater agility and smaller budgets

- Common mistakes
  - Increase WIP
  - Sacrifice architecture, take shortcuts
  - Endless hot fix mentality

- Mitigation
  - Value stream mapping to identify bottlenecks
  - Prioritize technical debt (“waste management”)
  - Change mentality of the meaning of “Done”

Bottleneck: Manual Gates

- Manual gates introduces latency into release and delivery processes
- Manual gates are represented by the need for human intervention to move artifacts from one stage to the next
- Types of gates
  - Approvals
  - Environments
  - Tools
- Mitigation
  - Automation
  - Testing
  - Culture

CSC Cloud Technology Partners

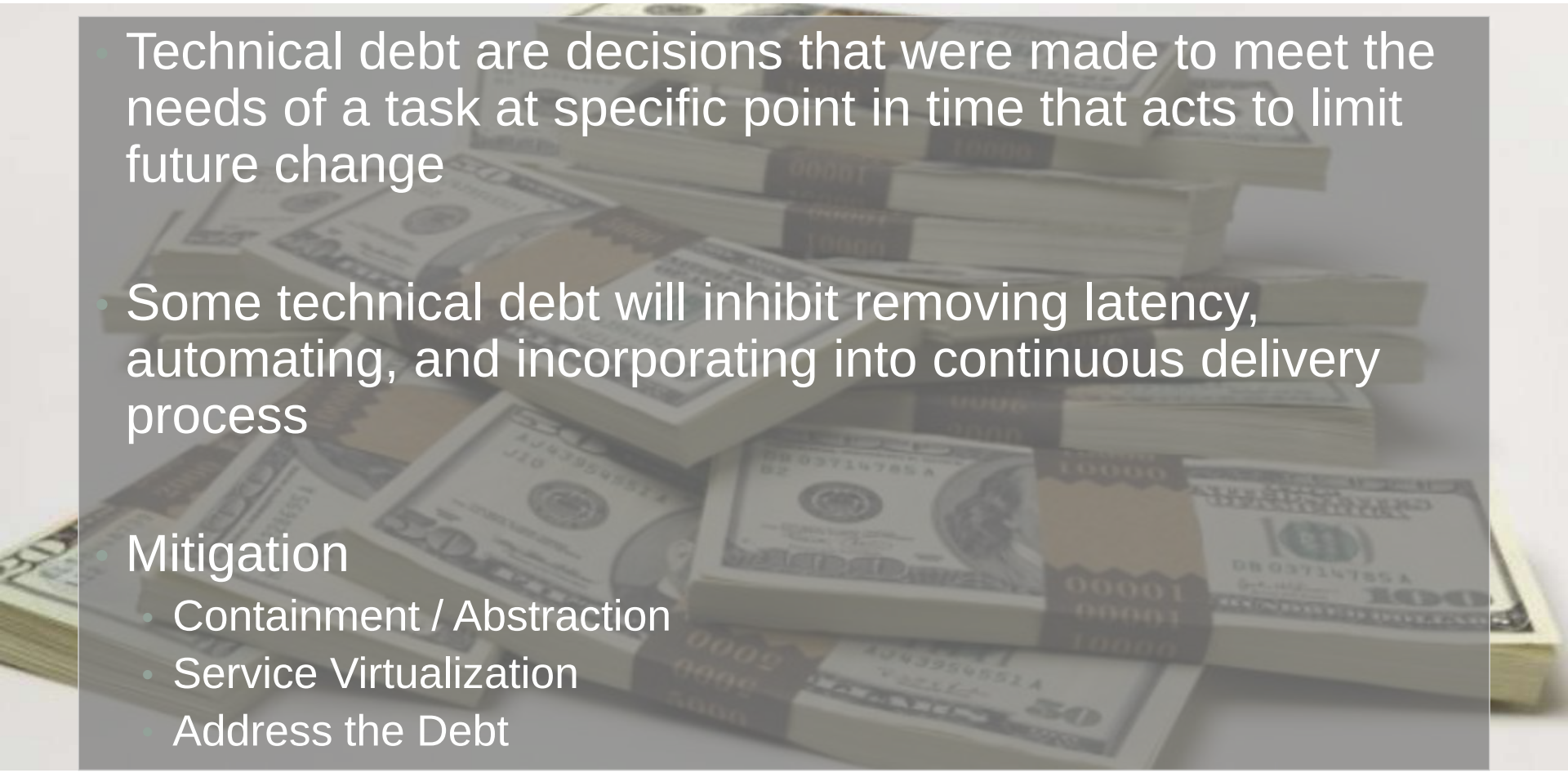
Bottleneck: Organization Silos

- Silos are not inherently good or bad
  - Silos isolate capabilities from each other
  - Silos act to limit and/or filter communications
  - Silos tend to have unique or inwardly-focused incentives and leadership
  - Silos are not easy to dispel
- Mitigation
  - Cross-functional leadership
  - Consolidation
  - Communications & management tools
  - Shared accountability

Constraint: Auditing & Compliance

- Compliance requirements are derived both internally and externally
  - External compliance often impacts ability to enact business
  - Internal compliance more pliable but still difficult to change in large-scale enterprises
- Auditing ensures compliance is adhered to and can add overhead to development, logging, deployment and operations
- Mitigation
  - Meet with auditors to identify acceptable methods of meeting compliance
  - Drive auditing left along with other DevOps initiatives
    - Approval of design may just require automated means to ensure design was adhered to in delivery
  - Metadata capture and management is critical to regulated environments

Constraint: Technical Debt



- Technical debt are decisions that were made to meet the needs of a task at specific point in time that acts to limit future change
- Some technical debt will inhibit removing latency, automating, and incorporating into continuous delivery process
- Mitigation
  - Containment / Abstraction
  - Service Virtualization
  - Address the Debt

Constraint: Misaligned Incentives

- If incentives don't change, behavior won't either
  - Incentives need to be shared across boundaries
  - Everybody owns quality, security, reliability
  - Moving to SaaS - Services model vs Product model
- Product owners must own product/service end to end
  - - What would happened if automobile product owners were not accountable for safety?
- Mitigation
  - Evaluate business model and map incentives
  - Assign ownership at the right place but incent appropriately to share goals
  - Tear down silos when they inhibit progress

Constraint: ITIL/Change Control

- Processes must be agile too
  - What good is continuous integration and delivery when we have to wait for a CAB review every 7 days to deploy?
- ITIL still works, but it needs to be modernized
  - Built during the waterfall era
  - Gates replace trust, often rubberstamp
  - Different apps have different risk profiles, don't put a web app through the same rigor as a payment system
- Mitigation
  - Value stream mapping across service catalog
  - Remove waste, auto approve where possible
  - Use metrics and log data to automate decisions

Constraint: Lack of Metrics

- Big part of DevOps is continuous improvement
  - Measure what matters
  - Be transparent with metrics so people can contribute
- Move from reactive to proactive
  - Establish baselines and raise alerts when deltas occur
  - Allows for fixing issues before customers notice
  - Measure processes to, they often get in the way
- Mitigation
  - Define KPIs for different actors within the system
    - Product, Finance, Security, Dev, QA, Ops, Sales, Customer, etc.
  - Design logging and monitoring framework with single pane of glass
  - Customer views, publish/suscribe

Best Practice: Continuously Deploy into UAT

- For many businesses it is not feasible to continually be releasing to production
- The cornerstone of Continuous Delivery is always being in a state that is ready to release
- Establish a User Acceptance Test (UAT) area that is continually being updated with the most recent released software
  - UAT should closely model production as much as possible
  - Upon release to UAT business users should be notified of availability of a new release and which features are included
  - Releases do not overwrite one another, but should exist in tandem

Best Practice: Governance/Security

- Design in security and regulatory controls up front
- Enforce controls through automation, self-service capabilities, risk profiling, continuous inspection
- Non-App specific rules, policies, controls should be abstracted from development (as a Service)
  - Centrally stored and managed
  - Configurable
  - Auditable
  - Visible

Best Practice: Reduce Work-in-Progress

- Too much Work-in-Progress (WIP) affects quality and predictability of completion
  - Reducing WIP produces greater predictability for average lead time
  - Focus on a single task until completion increases quality of output
- Only start new work once existing work is complete
  - Agile methods facilitate shorter durations for tasks
- Bottlenecks more likely to be addressed if they result in resources sitting idle than if they can switch to other tasks
- Leverage tools like Kanban to visualize WIP and manage backlog effectively
  - Capture and analyze metrics regarding velocity

Best Practice: Fail Fast

- Leverage Minimal Viable Product (MVP)
  - Qualify that work effort will lead to a usable output
  - Increase success velocity by quickly eliminating efforts that do not satisfy stated outcomes
- Design for limited release
  - Test new features in production among subset of entire consumer base
- Leverage Continuous Delivery to limit latency in release
  - Long feedback loops impair “fail fast”
- Leverage cloud for speed and economics
  - Investments in infrastructure and software enforces “must make it work” mentality
  - Cloud keeps costs down and eliminates capital investments necessary to test innovations simplifying the decision to terminate

Best Practice: Testing Center of Excellence

- QA and Testing are not synonymous
  - QA qualifies release meets stated goals for release capabilities and features
  - Testing occurs throughout the SDLC
- Embrace failure as inevitable
  - Too much time and money spent attempting to avoid failure
  - Focus on Mean-Time-to-Repair (MTTR)
- Center of Excellence (CoE) responsibilities
  - Practice Management
  - Governance
  - Organization
  - Environment

Top 5 Things To Go Do Today

1. **Find a Problem Area**
   - Achievable scope
2. **Identify Bottlenecks**
   - Value stream mapping
3. **Gather Metrics**
   - Establish baselines
   - Set targets
4. **Reduce WIP**
   - Improve flow
5. **Deliver Small/Quick wins w/Business Impact**
   - Increase trust & transparency
   - Create value

CSC Cloud Technology Partners

Thank You!

Questions?