The DevOps Handbook – Parts 5 & 6

– Part 5: The Third Way – The Technical Practices of Continual Learning and Experimentation;

1. Introduction
   1. Goal – practices to enable learning as quickly, frequently, cheaply, and as soon as possible
      1. Institutionalize rituals to increase safety, continuous improvement, and learning
      2. Create mechanism to rapidly spread learning throughout the organization
2. Ch. 19 – Enable and Inject Learning into Daily Work
   1. Complex systems are impossible to predict for all outcomes
      1. Dr. Steven Spear - resilient organizations are “skilled at detecting problems, solving them, and multiplying the effect by making the solutions available throughout the organization.” They are self-healing
      2. Netflix – April 21, 2011; entire Amazon AWS US-EAST availability zone went down; Netflix was unaffected. Since 2008 they were building and testing with the Chaos Monkey; they architected for failure, tested for failure, and evolved beyond it
   2. ESTABLISH A JUST, LEARNING CULTURE
      1. Unjust responses to incidents
         1. Impede safety
         2. Promote fear over mindfulness
         3. Create bureaucracy rather than carefulness
         4. Cultivate secrecy, evasion, and self-protection
      2. Dr. Sidney Dekkar – *Bad Apple Theory* – you cannot eliminate error by eliminating the people who caused the error.
         1. “Human error is not our cause of troubles; instead, human error is a consequence of the design of the tools that we gave them.”
         2. Accidents are due to the inevitable design problems in complex systems that we build; they are system problems – not individual problems
      3. Effective practices
         1. Blameless post-mortems
         2. Controlled introduction of failures for practice
   3. SCHEDULE BLAMELESS POST-MORTEM MEETINGS AFTER ACCIDENTS OCCUR
      1. Blameless Post-Mortem – meeting to examine “mistakes in a way that focuses on the situational aspects of a failure’s mechanism and the decision-making process of individuals proximate to the failure.” – John Allspaw
      2. Blameless Post-Mortem – Actions:
         1. Construct a timeline and gather details of the failure from multiple perspectives
         2. Empower all engineers to improve safety by allowing them to give detailed accounts of their contributions to the failures
         3. Enable and encourage people who do make mistakes to be the experts who educate the rest of the organization on how not to make the mistakes in the future
         4. Accept that there is always a discretionary space where humans can decide to take action or not, and that the judgment of those decisions lies in hindsight
         5. Propose countermeasures to prevent a similar accident from happening in the future and ensure these countermeasures are recorded with ta target date and an owner for follow-up.
      3. Blameless Post-Mortem – Stakeholders:
         1. The people involved in decisions that may have contributed to the problem
         2. People who identified the problem
         3. People who responded to the problem
         4. People who diagnosed the problem
         5. People who were affected by the problem
         6. Anyone else who is interested in attending the meeting
      4. Guidance
         1. Pull all factual evidence (chat logs, etc.) to help build the timeline; any specific metrics observed, investigative paths taken, results, and other resolutions considered
         2. Don’t allow fear of punishment or retribution creep in due to words or behaviors – find a facilitator to start
         3. Explicitly disallow the phrases “would have” or “could have”
            1. These are counterfactual statements
            2. Frames the problem as *the system as imagined* rather than the *system that actually exists*
         4. Focus on – “Why did it make sense to me when I took that action?”
         5. Brainstorm on real, implementable countermeasures – not *Be more Careful*
   4. PUBLISH OUR POST-MORTEMS AS WIDELY AS POSSIBLE
      1. After the meeting, widely publish the minutes, artifacts, and results; share and encourage others to learn from what happened
      2. Share results outside of organization – even back to customers
   5. DECREASE INCIDENT TOLERANCES TO FIND EVER-WEAKER FAILURE SIGNALS
      1. As the organization gets better at finding and resolving problems, decrease the threshold of a problem definition to keep learning and improving
      2. Continue to amplify signals to help avert the next catastrophe
      3. Michael Roberto, Richard M.J. Bohmer, & Amy C. Edmondson (Harvard Business Review): organizations are typically structured as:
         1. Standardized Model – where routine and systems govern everything; including strict compliance with budget and schedule
         2. Experimental Model – every day every exercise and new piece of information is evaluated and debated; more similar to R&D lab.
   6. REDEFINE FAILURE AND ENCOURAGE CALCULATED RISK-TAKING
      1. Leaders reinforce the culture through their actions
      2. Roy Rappaport, Netflix – a single engineer had been responsible for taking down Netflix twice in 18 month span. He wasn’t fired, he had also helped move their operations and automation forward by “light-years” and had performed huge number of production deployments.
   7. INJECT PRODUCTION FAILURES TO ENABLE RESILIENCE AND LEARNING
      1. Crumple zones in cars – build in failure modes that keep issues away from critical areas.
      2. Michael Nygard, author of *Release It! Design and Deploy Production-Ready Software*, “If you do not design your failure modes, then you will get whatever unpredictable—and usually dangerous—ones happen to emerge.”
      3. Resilience – requires defining failure modes and then testing to ensure they operate as expected
      4. Great Amazon Reboot of 2014 – 10% of Amazon EC2 servers had to reboot for Xen emergency security patch. At Netflix, zero downtime, no one actively working incidents. They were at a Hollywood party celebrating an acquisition milestone.
   8. INSTITUTE GAME DAYS TO REHEARSE FAILURES
      1. Game Days – exercise designed to increase resilience through large-scale fault injection across critical systems
      2. Simulate and rehearse accidents for practice
         1. Schedule the event
         2. Give teams time to prepare, make changes, and establish procedures
         3. Execute
      3. Expose latent defects in the systems and create progressively more resilient systems with higher degrees of assurance
3. Ch. 20 – Convert Local Discoveries into Global Improvements
   1. USE CHAT ROOMS AND CHAT BOTS TO AUTOMATE AND CAPTURE ORGANIZATIONAL KNOWLEDGE
      1. ChatOps pioneered at GitHub – put automation tools (Hubot) into the middle of their chatrooms
         1. Everyone saw everything that was happening
         2. New engineers could see what daily work and how it was performed
         3. People were more likely to ask for help when others helped
         4. Rapid organizational learning was enabled and accumulated
      2. Public knowledge versus private knowledge from emails
   2. AUTOMATE STANDARDIZED PROCESSES IN SOFTWARE FOR RE-USE
      1. Don’t store standards and processes in Word or non-actionable documents; leads to fragmentation, fragility, and unmaintainable outcomes
      2. Transform and use executable forms of standards and processes
      3. Justin Arbuckle, Chief Architect at GE Capital, “the actual compliance of an organization is in direct proportion to the degree to which its policies are expressed as code.”
   3. CREATE A SINGLE, SHARED SOURCE CODE REPOSITORY FOR OUR ENTIRE ORGANIZATION
      1. Firm-wide shared source code repository is powerful way to share local discoveries to the entire organization
         1. Configuration standards for libraries, infrastructure, and environments
         2. Deployment tools
         3. Testing standards and tools, including security
         4. Deployment pipeline tools
         5. Monitoring and analysis tools
         6. Tutorials and standards
      2. Google – Single repository with over 1B files and over 2B SLOC, over 25K engineers for every Google property
   4. SPREAD KNOWLEDGE BY USING AUTOMATED TESTS AS DOCUMENTATION AND COMMUNITIES OF PRACTICE
      1. Ensure automated tests demonstrate use and behavior of libraries and components
      2. Test suite becomes the living documentation of the system specification and represent working examples of API use
   5. DESIGN FOR OPERATIONS THROUGH CODIFIED NON-FUNCTIONAL REQUIREMENTS
      1. Designing for fast flow, deployability, and operations will derive NFRs. Codify these into the tests and pipeline
   6. BUILD REUSABLE OPERATIONS USER STORIES INTO DEVELOPMENT
      1. Goal – make recurring work as repeatable and deterministic as possible; standardize and automate (rinse and repeat)
      2. Collectively define handoffs, for non-automated work, as clearly as possible to reduce lead times and errors
   7. ENSURE TECHNOLOGY CHOICES HELP ACHIEVE ORGANIZATIONAL GOALS
      1. Balance team productivity against organizational goals; let operations influence components used in production or have the ability to not be responsible for unsupported platforms
      2. Systematically review production infrastructure and services for items that are causing disproportionate amounts of failure and unplanned work; plan for elimination or replacement
      3. Create “buoys, not boundaries”; navigate the channel, mark the channel, and allow people to explore past it.
         1. The marked channel provides safe, supported passage
         2. You can go beyond the buoys, if you follow organizational principles
4. Ch. 21 – Reserve Time to Create Organizational Learning and Improvement
   1. Toyota Production System has *improvement blitz* – dedicated and concentrated period of time to address a given issue
   2. Target DevOps Dojo and 30-Day Challenge – teams work with dedicated Dojo coaches and engineers, execute 2-day sprints to generate breakthroughs. Teams regularly achieve results in days that previously would have taken 3-6 months
   3. INSTITUTIONALIZE RITUALS TO PAY DOWN TECHNICAL DEBT
      1. Regularly schedule day- or week- long improvement blitzes with Dev & Ops teams. Focus on problems they care about. NO FEATURE WORK
      2. Goal is focused improvement on daily work, not experimentation and innovation
      3. Demo back improvements at the completion of the blitz
      4. Empower those closest to the work to continually identify and solve their own problems. The cultural norm is for everyone to find and fix as part of their daily work
         1. Facebook HipHop PHP compiler – resulted from hack day project. Small team refined it over 2 year period and allowed Facebook to handle 6X production load compared to native PHP.
   4. ENABLE EVERYONE TO TEACH AND LEARN
      1. Dedicate organizational time to let teaching and learning happen. Nationwide Insurance – provides 2 hours each week with expectation that everyone is to teach or learn something in that timeframe. Place high value on mentoring
      2. Encourage life-long learning. Our field is only progressing at faster rates
      3. Pair with individuals from other organizations & teams
      4. Participate in peer reviews
   5. SHARE YOUR EXPERIENCES FROM DEVOPS CONFERENCES
      1. Cost is always a consideration. BUT, encourage attendance at conferences, give talks, and then sharing back to the organization what was learned
      2. Attend Meetups!
   6. CREATE INTERNAL CONSULTING AND COACHES TO SPREAD PRACTICES
      1. Capital One – SMEs hold open office hours
      2. Google Testing Grouplet and *Testing on the Toilet* periodical
         1. Started Test Certified Levels for teams and products to measure themselves against
         2. Provide Test Certified Mentors and Test Mercenaries to kick start testing efforts

– Part 6: The Technical Practices of Integrating Information Security, Change Management, and Compliance

1. Introduction
   1. Goal to simultaneously achieve Information Security goals and create high degree of assurance for confidentiality, integrity, and availability
   2. Don’t inspect security in at the end, it’s integrated as part of our daily work
      1. Make security part of everyone’s job
      2. Integrate preventative controls into our shared repository
      3. Integrate security with our deployment pipeline
      4. Integrate security with our telemetry for better detection and recovery
      5. Protect our deployment pipeline
      6. Integrate our deployment activities with our change approval process
      7. Reduce reliance on separation of duties
2. Ch. 22 – Information Security as Everyone’s Job, Every Day
   1. Many DevOps organizations have better results than when security is organized as a separate silo; fully integrate Security in the same manner as QA and operations
   2. Compliance checking is the opposite of security engineering
   3. INTEGRATE SECURITY INTO DEVELOPMENT ITERATION DEMONSTRATIONS
      1. Bring Infosec left; incorporate them into demos and help them understand team goals in the context of organization goals, provide feedback and guidance as early as possible
      2. Awareness and involvement provides better business context for risk-based decisions
   4. INTEGRATE SECURITY INTO DEFECT TRACKING AND POST-MORTEMS
      1. Track all open security issues in the same backlog to ensure visibility and enable prioritization
      2. Conduct post-mortems for security incidents to resolve, prevent, and transfer security knowledge to the teams
   5. Integrate preventive security controls into shared source code REPOSITORIES and shared services
      1. Add mechanisms & tools
      2. Add security’s pre-blessed libraries, implementations, etc.
      3. Collaborate with any shared services to provide prebuilt, secured platforms for OS, VMs, authorization, logging and other security/auditing relevant requirements
      4. Reduce the friction by providing application/infrastructure stacks that are pre-approved and appropriately configured and secured.
   6. INTEGRATE SECURITY INTO OUR DEPLOYMENT PIPELINE
      1. Hardening the application after development completes is not acceptable. Left issues unaddressed due to schedule and budget constraints
      2. Automate as many security controls and tests as possible and incorporate into the pipeline
      3. Enable fast feedback on potentially insecure changes
   7. ENSURE SECURITY OF THE APPLICATION
      1. Focus on the *sad paths* or *bad paths* to effectively address QA, Infosec and related concerns. Incorporate these paths into automated tests
         1. Static Analysis – inspect program code for all possible run-time behaviors and seek out coding flaws, back doors, and potentially malicious code
         2. Dynamic Analysis – tests executed while a program is in operation. Monitor items such as memory, functional behavior, and other elements. *Testing from the outside-in*
         3. Dependency Scanning – inventory the dependencies for vulnerabilities or malicious binaries
         4. Source code integrity and code signing – all contributors should have their own key and sign all commits to version control. All created packages should be signed and hash recorded for auditing
   8. ENSURE SECURITY OF OUR SOFTWARE SUPPLY CHAIN
      1. Were often assembling applications from third-party components and integrating them with our business logic. We inherit the vulnerabilities of these 3rd party components
      2. Examine dependencies for known vulnerabilities and consolidate multiple versions of the same library
      3. 2014 Verizon PCI Data Breach Investigation Report – studies over 85K cardholder breaches. 10 vulnerabilities accounted for 97% of the exploits used. 8 of the 10 exploits were over 10 years old.
   9. ENSURE SECURITY OF THE ENVIRONMENT
      1. Once a hardened, risk-reduced environment is put in place, it must be monitored to ensure it stays in known good states
      2. Generate automated tests for appropriate settings and use the tests to scan environments for vulnerabilities
      3. 18F Cloud.gov (uses AWS GovCloud –
         1. Created platform addressing bulk of compliance concerns driven by ATO requirements
         2. Automating framework for generation of system security plans
   10. INTEGRATE INFORMATION SECURITY INTO PRODUCTION TELEMETRY
       1. Often production breaches are detected months after the event. A primary reason is due to no one regularly reviewing production logs
       2. Use telemetry to help fulfill infosec objectives and integrate security telemetry into the same tools
   11. CREATING SECURITY TELEMETRY IN OUR APPLICATIONS
       1. Create relevant telemetry
          1. Successful/unsuccessful logins
          2. Password resets
          3. Email resets
          4. PII or private changes
   12. CREATE SECURITY TELEMETRY IN OUR ENVIRONMENT
       1. Create environment telemetry
          1. OS changes in production, build infrastructure, etc.
          2. Security group changes
          3. Changes to configurations
          4. XSS, SQLi attempts
          5. Server errors
       2. Consider blocking and storing source of events when attacks are detected to facilitate mitigation strategies
   13. PROTECT OUR DEPLOYMENT PIPELINE
       1. CD pipelines represent a new attack surface. If not properly secured, credentials could be compromised or malicious code & changes injected.
          1. Harden CI/CD servers and ensure they can be reproduced in automated manner
          2. Review all changes through pair programming or code review
          3. Instrument the repository to detect suspicious code (API calls from certain types of test code)
          4. Ensure every CI process is in an isolated container
          5. Make the version control credentials of the CI system read-only
3. Ch. 23 – Protecting the Deployment Pipeline
   1. INTEGRATE SECURITY AND COMPLIANCE INTO CHANGE APPROVAL PROCESSES
      1. Effective change management recognized different risks associated with different types of changes, to be handled differently
         1. Standard changes: lower-risk following an established process; can be pre-approved
         2. Normal changes: higher-risk changes that require review or approval from change authority. Often times this a board which lacks the required expertise to fully understand the change; leads to long lead times
         3. Urgent changes: emergency and potentially high-risk that must be done immediately. Key goal of DevOps – streamlined normal process that is suitable for emergency changes
   2. RE-CATEGORIZE THE MAJORITY OF OUR LOWER RISK CHANGES AS STANDARD CHANGES
      1. A reliable deployment pipeline demonstrates that changes can be low risk and pre-approved.
      2. Use previous metrics to demonstrate high success rates and low MTTR
      3. Link and provide traceability from planning to version control to production implementation for visibility and auditing
   3. WHAT TO DO WHEN CHANGES ARE CATEGORIZED AS NORMAL CHANGES
      1. If change approval board is required, aim for quick deployment even if manual steps are required
      2. Use tooling to generate the artifacts needed for approval. Share evidence and artifacts from the pipeline to support.
      3. Continue to build successful track record to gain agreement for automated changes to become standard changes.
   4. REDUCE RELIANCE ON SEPARATION OF DUTY
      1. Separation of duties impedes feedback and limits ability to take responsibility for quality of work.
      2. Enable controls such as pair programming, continuous inspection, automated testing to achieve outcomes intended with controls.
   5. ENSURE DOCUMENTATION AND PROOF FOR AUDITORS AND COMPLIANCE OFFICERS
      1. DevOps challenges the traditional notions of auditing, controls, and risk mitigations.
      2. Bill Shinn, principle security solutions architect for AWS – auditors traditional training of sampling with screenshot evidence doesn’t really work in cloud, container or similar environments with infrastructure-as-code and auto-scaling. Must create alternatives methods of providing the data to show auditors controls are in place and operating.
         1. Work closely to identify the evidence needed for a given control and then show/enable the ability to retrieve on demand
         2. Help them understand the visibility and transparency that’s provided by telemetry and monitoring systems