10 Steps to Mainframe Agile Development

Mark Schettenhelm, Sr. Product Manager
September 28, 2017

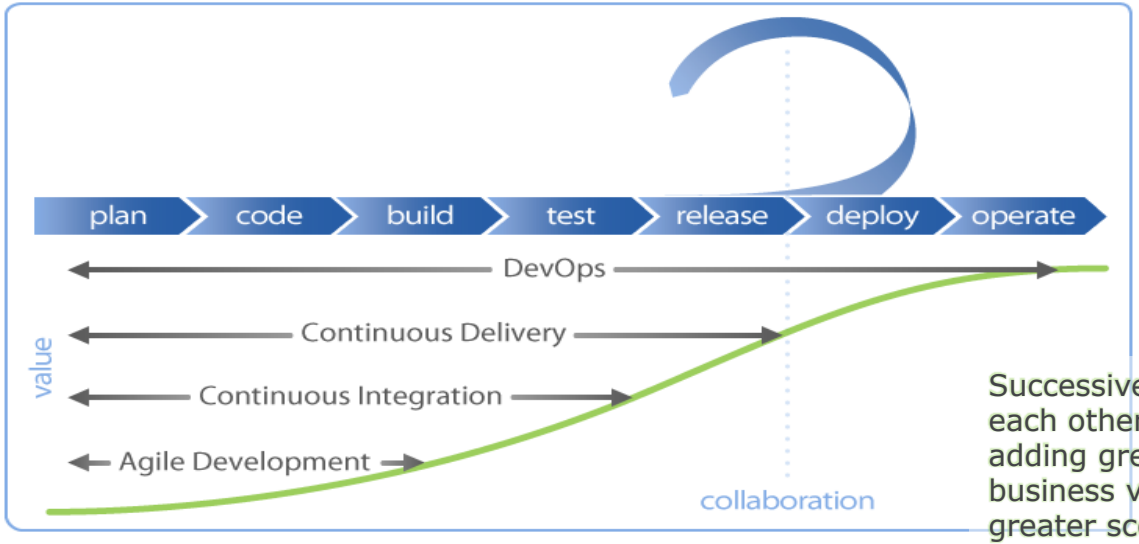
dev·ops *noun* \'dev-äps\
“a term used to describe a framework/collection of best practices and tools to deliver Systems faster and with higher quality”

2

The DevOps Transition

Improves Quality
Improves Productivity
* if done correctly

- Instant feedback to developers on quality issues
- Aids unit test automation on every build
- Supports Agile development
- Pre-cursor to Continuous Delivery and DevOps



Successive practices build on each other, with each practice adding greater and greater business value and having greater scope over the software development process.

10 Steps to Mainframe Agile Development

STEP 1
Define the Desired State

STEP 2
Modernize the Mainframe
Development Environment

STEP 3
Adopt Automated Unit Testing

STEP 4
Gain Graphical, Intuitive Visibility
into Existing Code and Data Structure

STEP 5
Enable Earlier Detection of Application
Quality Issues and Establish Quality KPIs

STEP 6
Initial Training in and
Adoption of Agile Processes

STEP 7
Use Operational Data Throughout the
Development, Testing and Production Lifecycle

STEP 8
Agile-enable Core Source
Code Management Functions

STEP 9
Automated, Intelligent Deployment

STEP 10
Cross-platform
Continuous Delivery

Step 1
Define Desired State

Create specific goals in terms of:

Agility	Enables frequent, rightsized code changes to fulfill business needs
Confidence	Ensures successful mainframe code changes without unintended consequences
Efficiency	Maximizes time and skills across dev, testing and ops
Ease of Use	Empowers mainframe-inexperienced developers to work on updates and enhancements
Integration	Connects mainframe and non-mainframe systems to achieve DevOps

Define Desired State

- Visualize what you want out of this
- Where you want to be
- Sample goals
  - One process
  - Reusable, easy to discover code
  - Data that is understood and available
  - Increased quality
  - Faster, more frequent delivery

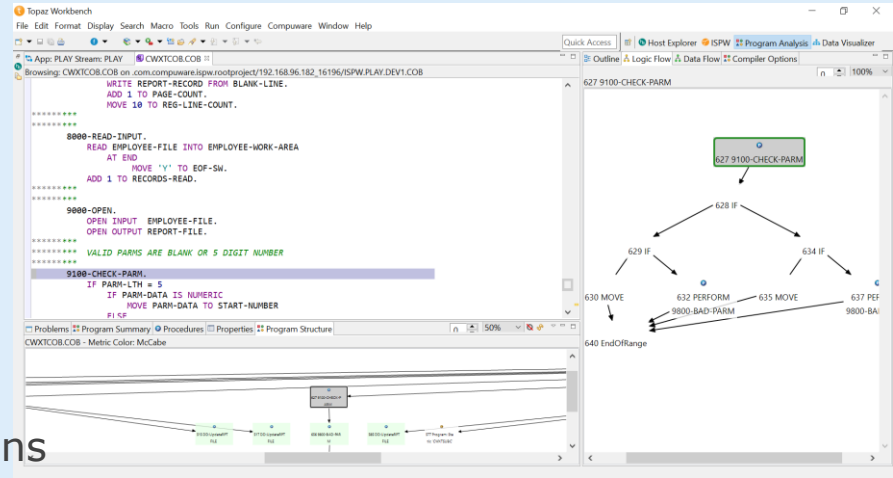
Step 2 Modernize the Mainframe Development Environment

Compuware Topaz®

Compuware iStrobe®

Modernize the Mainframe Development Environment

- Many still work as if it were 1980
- Yes, you may be lightening quick in ISPF, but…
  - Would new people be that proficient?
  - Can you use modern analysis and graphical tools?
  - Could you be even faster in a graphical interface?
- I believe we are at that "Windows Moment" where there is enough, more than enough, to move
- Having one interface for all code available means that you can work on multiple languages, platforms
- "Different only in syntax"



Step 3 Adopt Automated Unit Testing

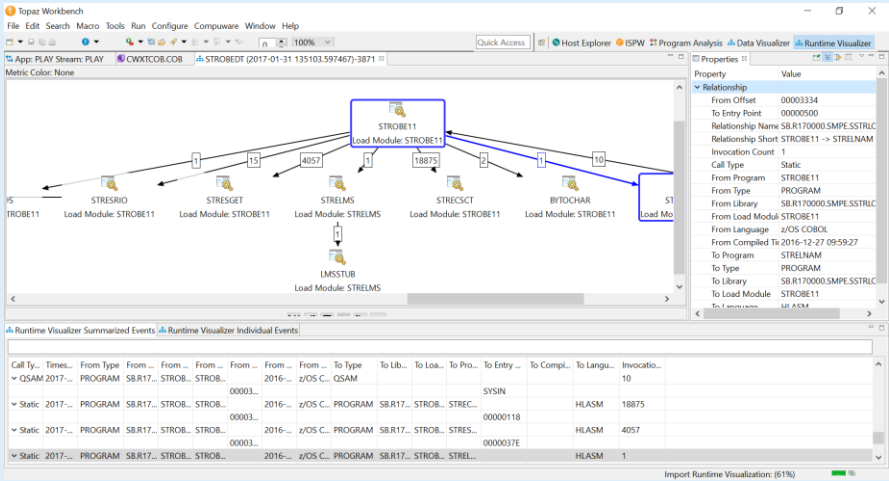
Adopt Automated Unit Testing

- But wait! If we go faster, quality goes out the window
- Well, yes, if you do it like you do today
- Testing is something we all want to do, should do, but don’t do well
- This could be the most important part in getting the process to speed up and actually improve quality, find defects faster, shift left
- Unit testing – testing small sections to make sure that section works
- Automated – do a build each night, run the tests, alert on failure, fix in the morning
- It is different from these
  - Regression testing – it didn’t break anything
  - Volume Stress testing – it can handle the load

Step 4 Gain Graphical, Intuitive Visibility into Existing Code and Data Structure

Gain Graphical Visibility into Code and Data Structure

- Remember Flow charts? We did them in the beginning to understand logic
- How do you understand logic today? New applications, application knowledge leaving, how can you be flexible and adapt applications?
- You need ways to tear into your applications, your programs, your data
- Make decisions faster, with confidence
- If you can't understand it, you can't change it



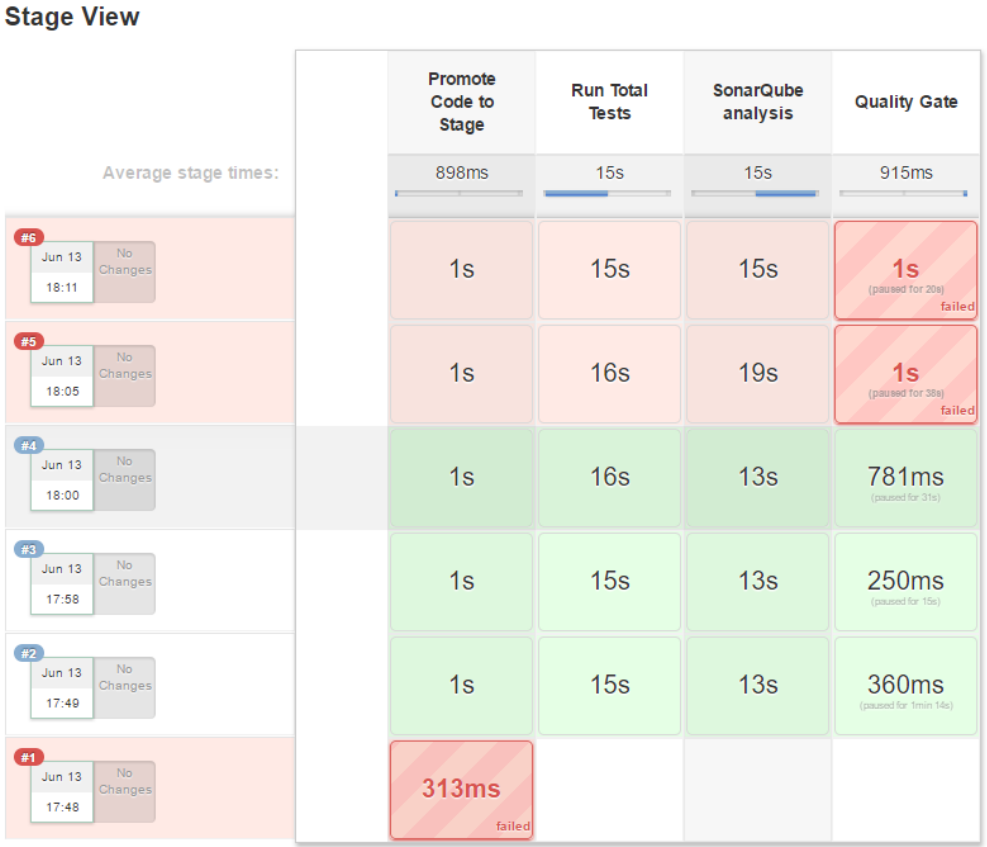
Step 5
Enable Earlier Detection of Application Quality Issues and Establish Quality KPIs

Enable Earlier Detection of Application Quality Issues and Establish Quality KPIs

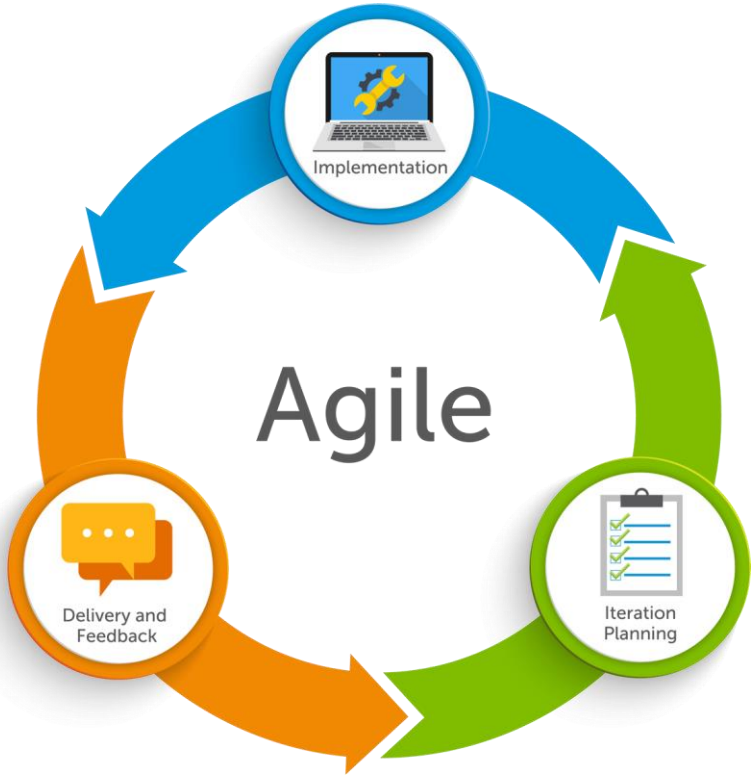
- You can’t improve what you can’t measure
- Saying quality is bad is not helpful
- Earlier Detection is key, code will always be tested, do you want it tested
  - By the end user
  - At the final go/no go test
  - Integration or Regression testing
  - Right after the nightly build so you can fix it in the morning
  - Right after you’ve committed the change

Set up your DevOps tool chain

- If it is automated, it will be done, if it's not, it won't be
- This is key to DevOps, the automation of common repeatable tasks, with notification
- This is how you can speed up, it frees you to focus on coding AND you increase quality



Step 6 Initial Training in and adoption of Agile Processes



Initial Training in and adoption of Agile Processes

- Adopting Agile Development goes hand and hand with adopting DevOps
- My advice?
  - Get everyone trained first
  - Go all in! Don’t try it in one group for a few months – it shows lack of commitment
  - Have top down commitment – burn the bridges
  - Live with the pain, it will ease over time. There will be temporary disruptions but benefits will start to build. Your end state will be better
  - Learn from the experiences of others, share successes
  - Continue to talk and to retrain
  - This is continuous improvement

Step 7 Use Operational Data Throughout the Development, Testing and Production Lifecycle

Use Operational Data Throughout the Development, Testing and Production Lifecycle

- Using operational data in testing helps identify production-type errors earlier
- Operational tools foreshadow code issues
- Feedback loops are crucial as pre-production issue identification and resolution avoids costly abends and potential app issues
- In short - knowing more about your code, sooner in the process, is a good thing

Step 8 Agile-enable Core Source Code Management Functions

Compuware Topaz®

Compuware ISPW®

Agile-enable Core Source Code Management Functions

- You need your Source Code management to allow for concurrent development
- With a two week sprint, you can’t wait for the other person to finish
- You need your Source Code management to fit into a DevOps pipeline – promote code, syntax check, compile, deploy to test environment, unit test, regression test
- It needs REST APIs to easily connect into your existing toolchains
- It can and should be on the mainframe to allow for platform differences and efficiency, but have a familiar modern interface and API connectivity with notification

Compuware
Topaz®

Compuware
ISPW® Deploy

XebiaLabs

Jenkins

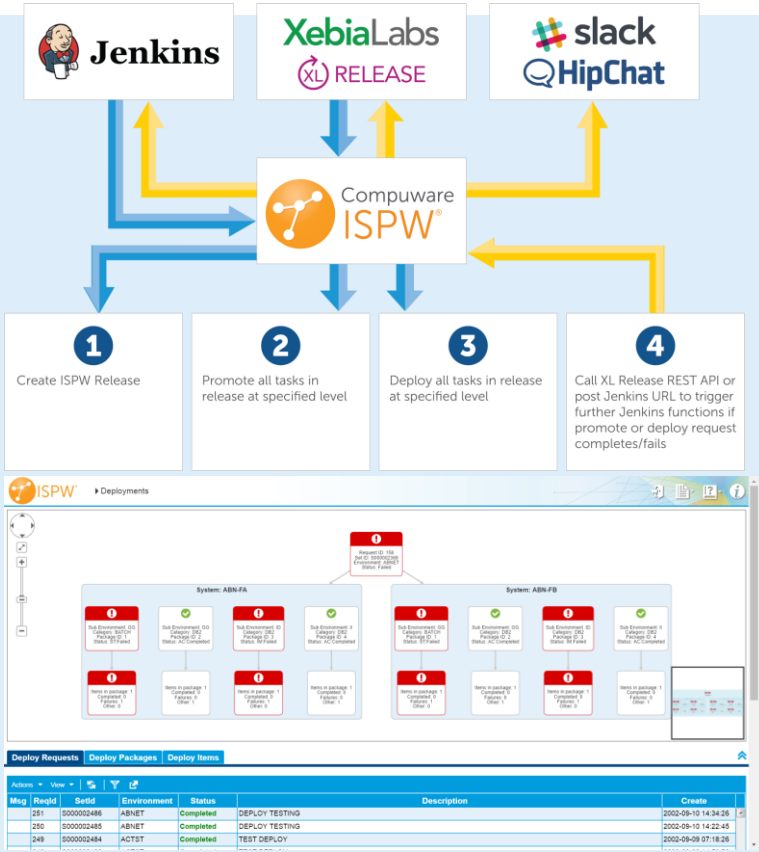
Step 9
Automated Intelligent Deployment

Is your deploy like this?

- Is it too large? Should you break it up into more manageable pieces
- How many people are involved?
- Do you have insight into the entire process or is it siloed?

Automated Intelligent Deployment

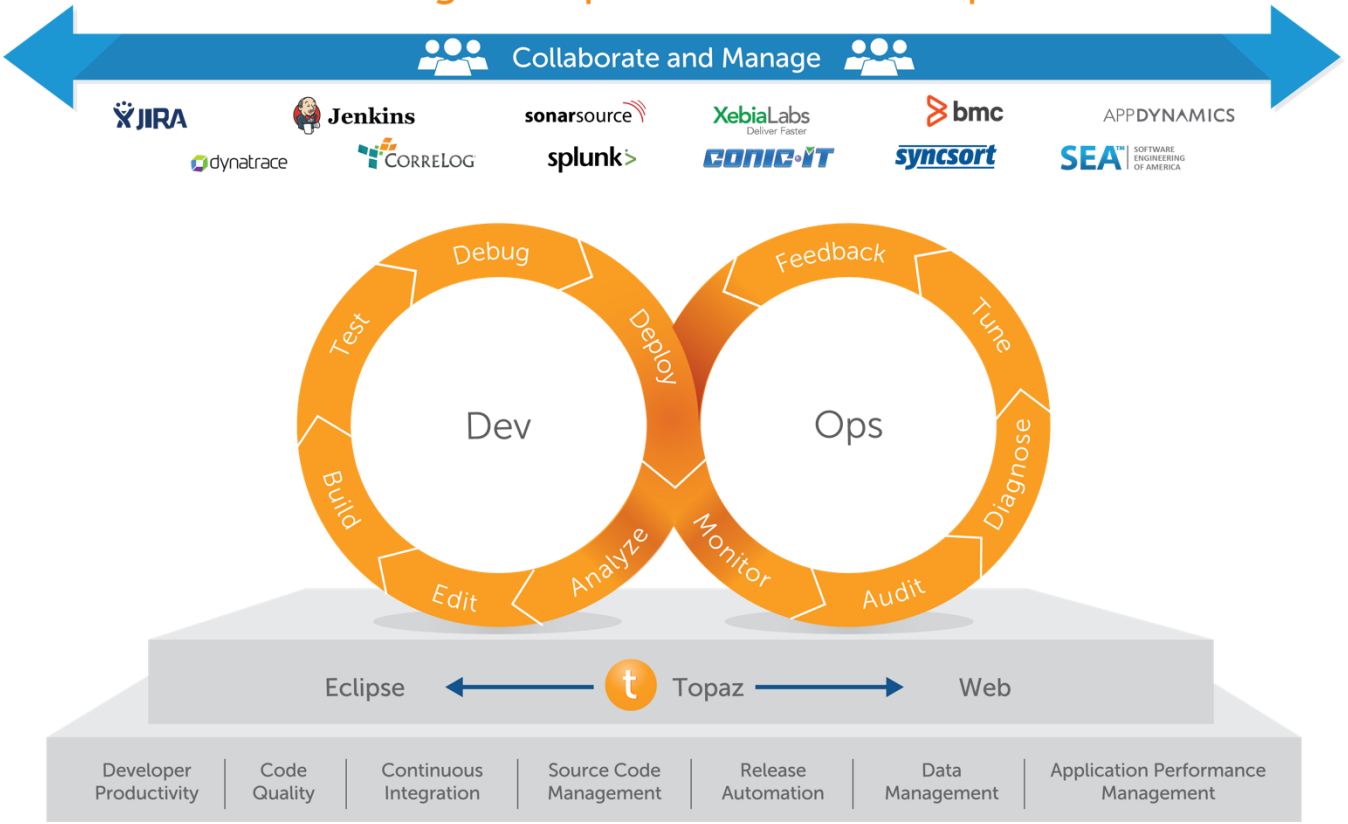
- Deployment is where the payback begins and is often where these efforts start
- Right now you probably have two separate deploys, how can that work? How can it mesh?
- The Deploy can be platform specific and should be, but the orchestration and control should be in one place
- Leverage REST APIs in your deploy to get distributed and mainframe deploy talking together



Step 10 Cross – Platform Continuous Delivery

Blended Ecosystem

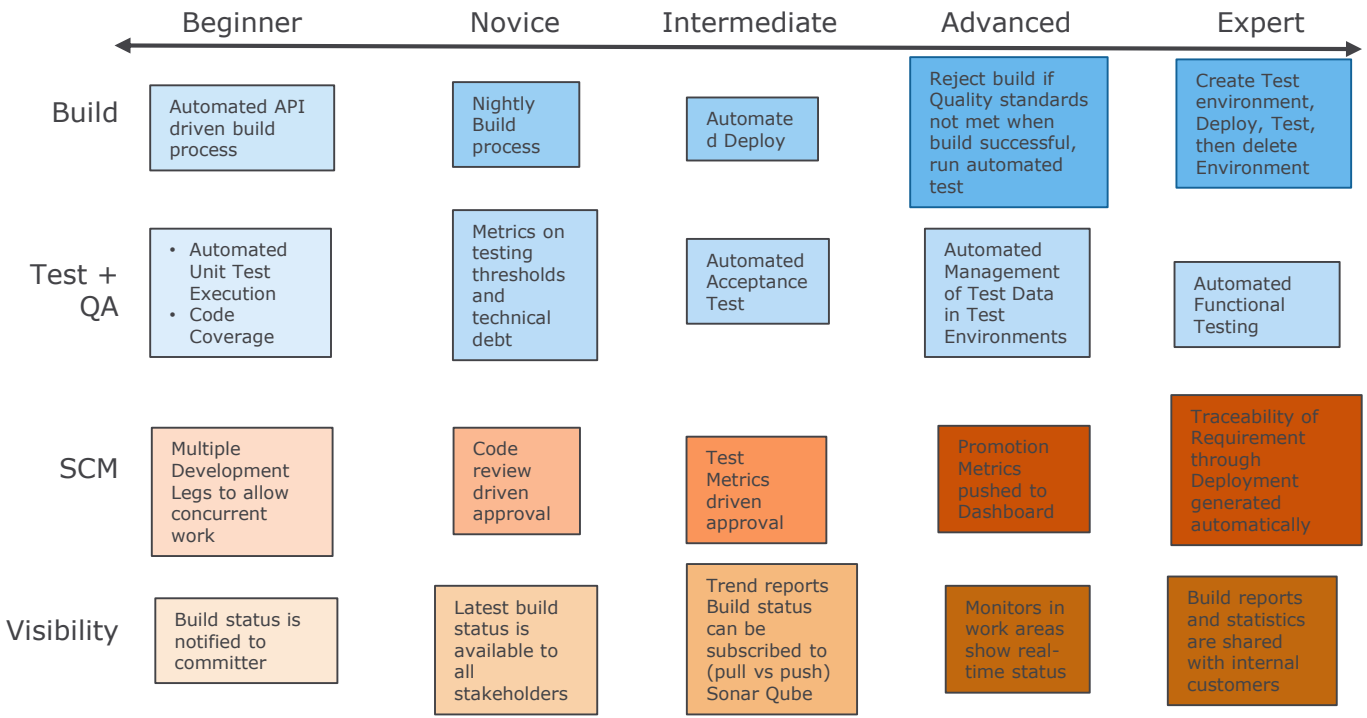
Enabling DevOps Across the Enterprise



This is about where the panic sets in, if it hasn’t already

- Relax, it can be done, it has been done
- Leverage your own experiences and those of others
- Break it into small, bite sized pieces
- I’ll say that again, break that into small bite sized pieces
- So you can “fail fast” and more importantly “learn fast”
- Each piece should deliver incremental value, you gain as you go

Continuous Delivery Maturity Matrix



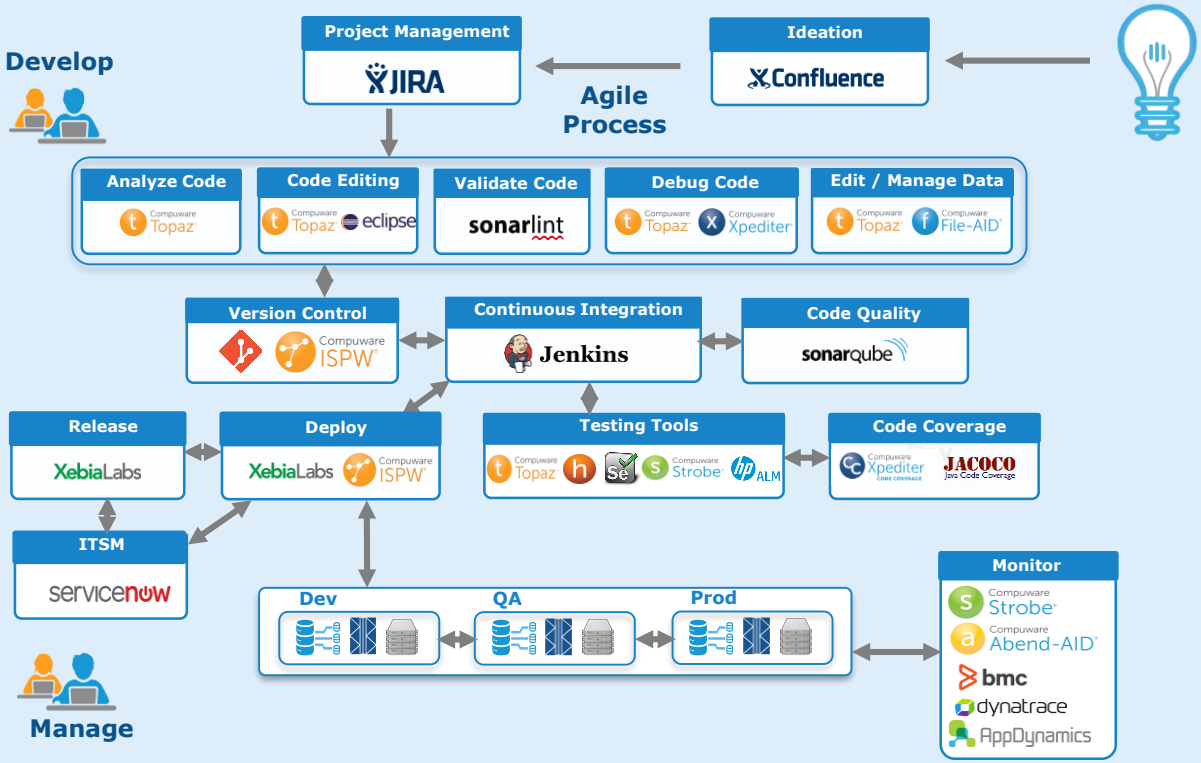
Journey to the Desired State

Step	2 Modernize Mainframe Dev Environment	3 Adopt Automated Unit Testing	4 Gain Graphical, Intuitive Visibility into Existing Code and Data Structures	5 Enable Earlier Detection of App Quality and Establish Quality KPIs	6 Initial Training in and Adoption of Agile Processes
Focus and Goal	• Eliminate need for green screen and highly specialized knowledge • All experience levels easily work on mainframe and non-mainframe dev, test and maintenance tasks	• Incremental testing allows devs to quickly and continuously adjust to better align with goals • Reduce reliance on manual testing • Empower those not accustomed to working this way to change mentality	• Large, complex, undocumented mainframe apps impede transformation • Highly dependent on tribal knowledge of senior mainframe staff • Prepare new devs to quickly "read" existing app logic, program inter-dependencies and data structures	• Mainframes power core business processes with low tolerance for error • Increased possibility for human error as less-experienced devs work on mainframe tasks • Error rates for experienced devs may increase as speed and frequency increase	• Now have dev environment and teams in place • Shifting to incremental model allows teams to collaborate • Moving from waterfall to Agile presents significant changes in culture • Craft mainframe and Agile-experienced devs to aid transition
Tools	**File-AID, Xpediter, Topaz Workbench**	**Topaz for Total Test**	**Topaz for Program Analysis and Topaz for Enterprise Data**	**SonarLint and SonarQube**	**Atlassian JIRA, HipChat and Confluence**
Success Indicators	1. Empirical productivity metrics (ex. Delivery cycle times, etc.) 2. Positive anecdotal feedback 3. Motivate non-mainframe devs to work on mainframe-related activities	1. More frequent code drops 2. Fewer errors found later in lifecycle 3. Tight synchronization across mainframe and non-mainframe dev	1. Devs work independently on unfamiliar programs 2. Experienced devs confirm benefits of program visualizations 3. Measurements of incremental improvements in dev productivity	1. Higher rates of pre-compile error detection 2. Positive trends in quality 3. Reduced number of error-related cycles	1. Certain percentage of dev/test staff completes Agile training (with goal = 100%) 2. First delivery of artifacts from initial Agile teams 3. Discovery of obstacles to broader adoption 4. Evidence of cross-team collaboration

Journey to the Desired State

Step	7 Use Operational Data Throughout the Dev/Test/Prod Lifecycle	8 Agile-enable Core Source Code Management Functions	9 Automated, Intelligent Deployment	10 Cross-platform Continuous Delivery
Focus and Goal	• Using operational data in testing helps identify production-type errors earlier • Operational tools foreshadow code issues • Feedback loops are crucial as pre-production issue identification and resolution avoids costly abends and potential app issues	• Modern, end-to-end Agile SCM, release and deployment automation enables all skill levels to fulfill business requirements, optimize code quality and improve dev productivity • Automated change management eliminates manual steps, empowering quick iteration through dev, test and QA	• Quickly and reliably getting new code into production is crucial to keeping pace • Automate and coordinate deploy of all related dev artifacts into all target environments in highly synchronized manner • Pinpoint deployment issues immediately, taking instant corrective action	• Including mainframe in enterprise DevOps is critical to achieving agility when changing multi-platform apps to fulfill business needs • Craft de-siloed environment where mainframe can be quickly and appropriately accessed to meet business needs by whichever staff resources are available
Tools	**Abend-AID and Strobe**	**ISPW SCM**	**ISPW Deploy**	**XebiaLabs XL Release**
Success Indicators	1. Earlier detection of avoidable CPU consumption 2. Reduction in Abends in production 3. Reduction in cost per error measures	1. Different Agile teams work on different stories in parallel 2. Devs with all skill levels can quickly understand scope of changes before diving in 3. Reductions in code approval delays	1. Faster rollouts of compiled code 2. Reduction in code promotion failures 3. First successful automated rollback from failed deployment	1. Related code is worked on in parallel on multiple platforms 2. Increased communications and collaboration between previously siloed devs with different skill sets 3. First successfully automated cross-platform release rollout

Mainframe Inclusive DevOps Toolchain



QUESTIONS?

mark.schettenhelm@compuware.com

@MarkSchett

The Mainframe Software Partner
For The Next 50 Years