# The Modern Mainframe Podcast Transcript

## Building a Better Software Development Platform: Chapter 12

## AUTOMATING SOFTWARE DELIVERY I

**Matt:** Hey everyone, I'm Matt DeLaere and this is Chapter 12 of the Building a Better Software Delivery Platform podcast series. Each week, Compuware Executive DevOps Solution Architect Rick Slade tackles a different aspect of the modern software delivery system. And this week is no different, so let's say hi to Rick. Hey Rick.

**Rick:** Hey, Matt, how are you?

**Matt:** I'm good, thanks. So, we have a special guest today. Compuware Vice President of Product Engineering, David Rizzo. Way back in Chapter Two, David joined us to talk about the architecture of an SDS. Today, he returns to discuss the role of automation in the software delivery life cycle. So, David, you're our first returning guest, we obviously didn't scare you off. Thanks for joining us day.

**David:** Thanks for having me back, I appreciate it.

**Matt:** So, Rick, you've discussed automation in probably every episode in the series, can you give us a quick overview of why such an important component of modernizing your software delivery system and maybe some of the benefits that are gained?

**Rick:** Yeah, I'm glad to, Matt. Yeah, I do mention it a lot. I think that in organizations, based on my experience, and this is an opinion, different people have different opinions on what constitutes the most effective way to deliver software. Everything's important, so I don't want to think that one less important than the other. But there's two critical components in my mind, and I'm going to ask this question to David, so David don't feel like you've got to respond the way I did—I'd love to hear your perspective on it. But I think there are the two most critical components to effective and efficient software delivery first begins with the adoption of Agile techniques, being able to manage the development and the delivery process using Agile methods. I think that is so absolutely critical to faster software delivery, but faster software delivery without sacrificing quality. I actually believe that Agile methods in the software delivery process actually produce a better product and in a faster fashion. So, I think that Agile is one capability that I think every organization needs to be looking at and figuring out how to incorporate into their development processes regardless of platform.

The second critical component to a modern software delivery ecosystem or system, as I like to call it, is automation. Looking at the work in your software delivery processes and trying to leverage the machine or automation to do as much of that work as is possible. And I say that because I'm a realist in that there may be work that can't be automated. I think what we're

finding, and maybe David will talk about, is that we are seeing that there's very little that can't be automated. I think you're only limited to… I think the architecture of the application has a part in defining what can and can't be, but sometimes it's just a limited imagination with regards to what can be done and leveraging tools and capabilities in different ways. And I'm sure David will talk about some of the things we're doing at Compuware with regards to automating the work that he is responsible for.

But I think those two things, the adoption of Agile, the incorporation of automation, and there are other things, but I think those two things can have the biggest impact from a business standpoint to an organization's abilities to deliver software faster and again, without sacrificing quality.

So, David is the first to do a second appearance on the podcast, so we're really appreciative of that. David is so knowledgeable, and I'll let him talk about his role, but he continues to provide great leadership in our labs and with our developers in the creation and delivery of our products. So, we're very glad to have David.

David, can you take a few minutes and tell us again about your role at Compuware and maybe some of the things that you guys are working on right now?

**David:** Sure, thank you very much, Rick. And again, it's great to be here. So, I'm David Rizzo, I lead the engineering organization for Compuware, so responsible for the Compuware product portfolio suite, new features, ongoing maintenance, and all the work that goes into that and delivering those products to our end users. And right now, we have a lot going on as we continue delivering every quarter. We recently launched our 23rd quarterly delivery with new features and updates to our classic offerings, so we are continuing to implement those changes and deliver new innovations for our customers.

## Incorporating Automation

**Rick:** Totally awesome, and it's just amazing what you guys are doing in the time periods that you've done it in. I wanted to start the discussion off with a question about your memory of when Compuware and your development organization started to incorporate automation into the processes. About when did that start?  What was it that you did? Do you remember how you started and why you started with what you started with? So, if you could maybe just go back into the memory banks and tell us a little bit about how we got started with automation in our software delivery life cycle.

**David:** Sure, so when I talk about 23 consecutive deliveries, we did our first delivery January 2nd of 2015. And we had been doing Waterfall for many years at Compuware and delivering in a very methodical way, and that delivery process was very manual. And when we delivered on that release January 2nd, we spent a lot of time making sure that everything was tested, that it was built properly, installing, and we spent literally hundreds of man-hours to making sure that that was delivered correctly, as we had done in the past with our waterfall deliveries. We

quickly realized that that would not scale to deliver every quarter. Doing it every 90 days, there's no way we would be successful if we invested that much time. So, we started at that point, took a critical look, and said, "How do we reduce the time?" You can't reduce the steps, you have to embrace automation, and we realized very quickly at that point that automation was going to be the key to our success, and we looked at it... From my perspective, I took a very pragmatic approach to get people to embrace automation, because oftentimes automation is equated with trying to eliminate people or eliminate positions or do something different. And our goal has always been to get as many people and resources working on delivering new functionality because that's where the biggest value is for our end users. So, automation freed time to use those resources in other ways to deliver value.

So, we began by looking at every step of the process from when code was finished and a developer said, "This code is ready to go," from that point until it was in our delivery system and a customer could download it. So, we started and looked at everything we could do to introduce automation, found the low-hanging fruit and then built upon that over time.

**Rick:** Yeah, do you remember what some of that low-hanging fruit was? I know it's been a long time ago, but....

**David:** Sure, so some of the, I'll say easier things were moving files. To get to our ultimate delivery system, we have PTFs as we deliver or product files, and somebody manually copied those to the production environment. So, a pretty simple one was create a piece of JCL or a copy script that could move those files and it could be executed by anyone. It started out that we would execute the script, one of somebody moving 100 files, they executed one job. So, we took that 30 minutes down to two minutes, and then further on down the road, of course, we introduced things like Jenkins and the whole pipeline as we built upon it. But those initial low-hanging fruit was those simple repetitive manual tasks and how we could automate that.

**Rick:** Yeah, and I love that because it's what I encourage folks to do for a long time. The first thing you've got to do is understand what steps, what work are you actually executing, how is it being initiated, how is it being managed, how is it being monitored from a manual standpoint. And with that understanding, you can start to think about things that you can automate very easily, and typically there's a lot of opportunities in that cycle to do that.

You mentioned JCL, and one quick point before I get to my next question, I tell folks all the time when I'm talking to clients who are just starting on their journey with regards to modernizing their software delivery system or software delivery environment that the mainframe developer actually, conceptually, has leveraged automation for 50 years. And sometimes I'll get a blank look and I'll say, "What's JCL?" And it's no more than a workflow of steps, work to be executed by different programs by different utilities, and it's essentially automation in and of itself. And so, that understanding of batch workflow management through JCL, simple JCL, I think has significant roots for the automation work that we're seeing today in today's modern pipelines.

We talked about it at first. This is a two-part series on automation because I think it's just that important. And there's a lot of parts to automation. Typically, at its highest level, we talk about build automation, we talk about testing automation, we talk about deployment automation from a DevOps standpoint, and all the work that's a part of that. We're going to focus on test automation more next week, because I think it deserved its own time slot.

## The Build Process

So, I wanted to ask you to define or describe the typical build process for the majority of our mainframe components in our products. The word build... I remember when I was at IBM at first hearing the word "build" thinking, "What the heck is that?" Coming as a mainframe guy, we talk about compiles and we talk about link edits, as far as building a binary that can be executed. And "build" wasn't a term, at least until recently, that at least shops that I had been involved in, had used a lot. So, could you take just a minute to describe what the build is, at least for copier, and how it's leveraged, or how it's automated in the delivery of our products?

**David:** The build process for us delivering to our end users for our mainframe products is either PTF, which we do a lot of our delivery through a PTF, so that a complete new install is not done, or a build could be a complete product that would be installed as a new release. And then for our tools like Topaz, the non-mainframe, we do using Eclipse through P2 repos for updates to them, and again, complete builds for an install. And the process of getting... We have a one file or one PTF that is done and we have multiples of those that make up a whole release that we're working on or an update. And so, the build is taking all those individual pieces, whether they're individual PTFs or individual components, compiling them, putting them all together to get into a file or file set that is deliverable as installable. So, for example, for the way we deliver PTFs, we take all the PTFs we have for a product and the build process takes all the individual PTFs, creates a single file so that those can be received, SMP/E received by the customer with one job, and the automated process we have takes all those PTFs, puts them in the file, applies them as that new package to an existing environment to validate the install, and then moves into the automated testing phase. And then once it's passed the testing phase, it automates the move of those to a file location where it would be picked up by the production environment, release environment.

**Rick:** Gotcha, so the build from your perspective is how I think most people see it, is the creation of an executable and then it can be deployed to a production environment. And the creation of that executable, in our case—and tell me if I'm right or wrong on this—is not just the compile and the link edit to create a load module, but also any potential testing that we can include in that pipeline. Is that accurate?

**David:** That is accurate. We put in the pipeline as much testing as we can, and that is automated as well, so that we can have confidence when we deliver something that it is ready to go and has been tested.

## Automation Technology

**Rick:** Gotcha. Can you talk a little bit about the automation technology that we use to support that process that you've just described?

**David:** Sure. So, our build process, again, once we have something that's approved to go, we use as our source code manager, we use ISPW and ISPW also has build and deploy capabilities, and we take advantage of both of those, So, as source code is promoted within ISPW to a level where it's ready to be released, then the build portion of ISPW takes over, will compile those, will put them into a structure that is deliverable, whether that is PTF or a file. It creates those deliverable pieces through an automated process and then it deploys those pieces to, what we have is a test system and test LPAR that all the changes are deployed to, which then allows in the process, the automated test scripts to be executed so we can run regression tests and integration testing, is all automated. It gets run once that deploy is finished and then it can be moved into the production-ready environment, so that it can be released to our customers and be available for them.

And in addition to using the ISPW build and deploy, it's also integrated with Jenkins. So, we use Jenkins to create the pipeline to initiate the testing once the build is done, so when code is promoted within ISPW, the Jenkins agent is aware of that. It will initiate the automated tests and it can initiate the automated tests on both mainframe and non-mainframe systems for changes that impact both environments or where tests need to be run in either. And so, Jenkins will initiate that and get the testing done. And then once that's passed, our test results and build results go into Sonar and all of that is validated in an automated fashion to meet certain criteria, and then it's made available for production.

**Rick:** So, we're using ISPW as our build engine and our deploy engine, and we're using Jenkins to manage that workflow as we move from the build through deployment, and we include different types of testing work in that Jenkins pipe. Do I understand that right?

**David:** You do. That is correct.

## Automated Testing

**Rick:** Good deal. So, I know testing is part of that process, and I know that we're going to discuss this in much more detail next week with Steen, but can you offer your thoughts about test inclusion in this delivery process from an automated standpoint? Talk about some of the testing that we are doing for, if not, maybe not all our products, but for the majority of them?

**David:** Sure. As a look back, testing, when you talk about automating the build process and having a good build process, testing is probably the biggest key and the most complex component of it, it certainly does warrant its own time and talking points. But that is one of the most complicated pieces to get put in, is to have that automated testing. But building the files is rather straightforward and creating the automation for that can be rather straightforward. The big question is, do you have the right code, does the code work when it's all put together,

because having the files is less than half the equation. You want to make sure that you have good quality and you can't sacrifice quality. And as you talk about, our goal is to get new code updates, changes, whatever it may be, into the hands of our end users as quickly and efficiently as possible. And I like to use the word "efficiently" because it just isn't about how fast can you get new code out, it's how fast can you get new code out that's been fully tested and is of good quality.

So, when we go through the build process, as I said, Jenkins kicks off our automated test suites, and we have different levels of testing, we have regression testing that runs through and makes sure all the functionality of the product is working, we have integration testing that tests the integrations with other products as Compuware products integrate with other Compuware products, make sure those integrations are still working, and then we also have a level of testing that's a little bit higher level of system testing that runs some more extensive tests on just the overall… how the product and how the system is running. And how we have our automation set up is when changes are made, every night we run, we can run through all of those tests, they're all initiated in an automated fashion for any and all of our products.

So, as part of the build process, once the file is built, then it is deployed and then the automated test sequence is executed, so we execute our Hiperstation scripts and other tests that are run and things within Topaz if there's integration and we have all that automatically initiated. The results come back, and based on the results of those tests, we then can see what percent of tests pass, what failed, what percent executed, and we have criteria set up that says if we run 100% of our tests and 100% of them pass, move it to production, and no manual intervention for our customers to have access to that updated file. And we have our criteria level set up based on the product and the type of change, so that process can go through in an automated way. It doesn't always have to be 100% of the tests run because there may not have been changes in parts of the product. If it does not meet our threshold for automation, then we have dashboards that are populated with all those results, and then somebody can go in. We have our release team that can go in in the morning and say, "Okay, these are the files that didn't get released," and then determine why they were not automatically released. If there was an error found or what needs to be done to make that available. It will go back to the development team to make any changes if needed, and then start the process over. And if it did for some reason, it was validated that everything was okay, there's a click of a button to automatically deploy it to production, so it would be available. So, that's the process that we go through to test and validate what we release is achieving the quality levels that we have set and we want.

## Responsibility

**Rick:** It really is a thing of beauty to see from a technology standpoint. Who in the organization, David, is responsible for maintaining those… I'm going to use the word "pipes" that manage all of that work that you just described? Is that the work of testers? Is that the work of a separate team? Who is responsible for maintaining those automated workflows?

**David:** So, we have a dedicated, what we call "release engineering team" that is responsible for the infrastructure of the workflow and that process. And then on the individual Scrum teams, we have... the team is responsible for updating the test cases and ensuring that the test cases that are being executed for certain changes are correct and that the result criteria is correct. So, the overall pipe would be a general team and then ensuring that it's running the way each team wants is up to the team and making sure that there are tests for everything that we need to have tested before we can confidently release something.

**Rick:** That's awesome because that aligns with my beliefs on how to manage a software delivery, and I use the air quotes "system," there has to be dedicated resources, I believe, and there are different thoughts in the industry about this, but I think that having dedicated resources who are responsible for the effectiveness and the efficiency of that pipe is critical, but it's up to testers and resources within those Agile teams, for the quality of the test cases and the testing that occurs within that pipe. Very cool.

## Automating Deployment

I want to move to... And we'll talk a lot more about testing next week, folks with Steen, because it is so important. I want to move a little bit to the deployment side, and I know it's not as splashy, maybe, as some of the other stuff. I want to talk about automating deployment and what that means and how that's accomplished at Compuware. Can you describe what deployments means to you and to Compuware, and how is automation leveraged in that deployment process within our company?

**David:** So, deployment is ultimately, once we've created a change, it's been built, it's been tested and validated, now we have something that we want to put into use. So, there's deployment to test, and then there's deployment to production, so we use a similar process. Again, we use the ISPW deploy functionality, which takes those individual executables and puts them on the correct system, deploys them out to different systems. And again, first deployment is for testing, so it goes to a testing system, and then next deployment is, once that passed, deployed to the production systems. So, for us, when we think about Compuware, we like to use our own software and we very much embrace the quicker we have people using it, the better we validate that everything we've done is working and we find opportunities for improvement. So, before as we would deploy something out to our customers to be available, it gets automatically deployed onto our development systems at Compuware. So, once the testing is passed, the next deployment phase is to deploy to those LPARs which are used by all developers at Compuware, so they have access to the latest and most up-to-date software. And then those files that we deploy then get moved into what is our release area so that they would be available for download by our end users.

## The Impact of Automation

**Rick:** Gotcha. Kind of talk to me a little about the impact, from a business standpoint, that this automation that you've described today has had on us as a company, maybe more importantly as well, on our customers.

**David:** Yeah, so I can't say enough about how important automation is from not only allowing us to focus what is most important, but also getting those things which are important to our customers in their hands. So, if I think back two, three, four years ago, if we had just a simple fix for a bug fix that we had to deliver, we would make a change, it would go through a manual process to make sure the testing was done to ensure that it was available and ready to go, and then we would be able to send it out to a customer. And we did monthly or quarterly updates to the file so that every other customer could get it. So, we'd make it individually available to someone who reported it, but to the broader audience, it would take up to 90 days to get out for everyone to make it available.

And very shortly, we will be to the point where our customer support center online will be updated every night once a fix is completed and has gone through the automated testing. So, from the time a developer says, "This fix is good in my eyes," we will have it out on the support center, assuming all the automated testing passes, within 24 hours. So, we're able to say that any changes we make today, we can deliver tomorrow. And that will be a completely automated process. So, if all the tests pass, and all the gates as we go are passed in that automation, and criteria is met, it will move into our production environment and a customer who has a problem or wants to get updated maintenance will be able to go out every day and get what is potentially new.

**Rick:** I was going to say, that's amazing because that truly defines in the purest form what continuous integration and continuous deployment can be for an organization. That truly is amazing.

**David:** It does define it. And I remember thinking back when we talked about this five years ago, and we started down this and started our first piece of automation, and we talked about being able to test and deliver overnight, and it seemed like a daunting task for many. And we didn't get there overnight, but it can be achieved, and that's one of the things, is taking one bite at a time, one piece, and moving methodically. Don't try and get it all done at once, but just keep moving methodically along. And the more time you invest in it, the more benefit you'll get.

**Rick:** Music to my ears, it truly is a journey that organizations should undertake in maximizing the effectiveness of software delivery within their organization. David, this has been great. I see our time is up. I can't thank you enough for joining us today. Matt, you want to wrap us up and then I'll close us out?

**Matt:** Yeah, thanks Rick. And thanks again, David, for taking the time to join us.

To everyone listening out there, be sure to join us this Friday at 11 AM Eastern Daylight Time for our live online Q&A session, Office Hour with Rick Slade. Bring any questions you might have about automation or about any part of the SDS.

You can also submit questions via Twitter. Just use the hashtag #goodcoding. To set a calendar reminder for the Q&A, or to check out any of our previous episodes, including David's first appearance, visit https://www.compuware.com/goodcoding.

So, Rick, that about does it. I'll send it back to you for some last words.

**Rick:** Thank you, Matt. And David, thank you again, I can't thank you enough. Your insights are incredibly valuable. Hopefully useful to our listeners. There is a lot of work to be done, but the business benefits of this type of investment in your software delivery efforts will pay significant business benefits. It'll create an environment that your developers will want to participate in. And the learning opportunities are tremendous. So, it's good from every perspective that I can see. It will definitely make a difference in the speed and the quality at which you deliver software within your organization.

David, thank you again. Thank you all for listening. We appreciate it. If you've got questions about anything that was talked about today or other questions regarding automating mainframe-targeted software delivery, if a child join us for the live Q&A on Friday. And with that, we'll close it out. Again, if you've got questions or concerns, as Matt indicated, you can go to Compuware.com/goodcoding to get information about these podcasts and about the live Q&As. And again, we hope that you'll join us. Take care, everyone. Stay productive. Stay safe. Have a great week. Good coding, everybody.