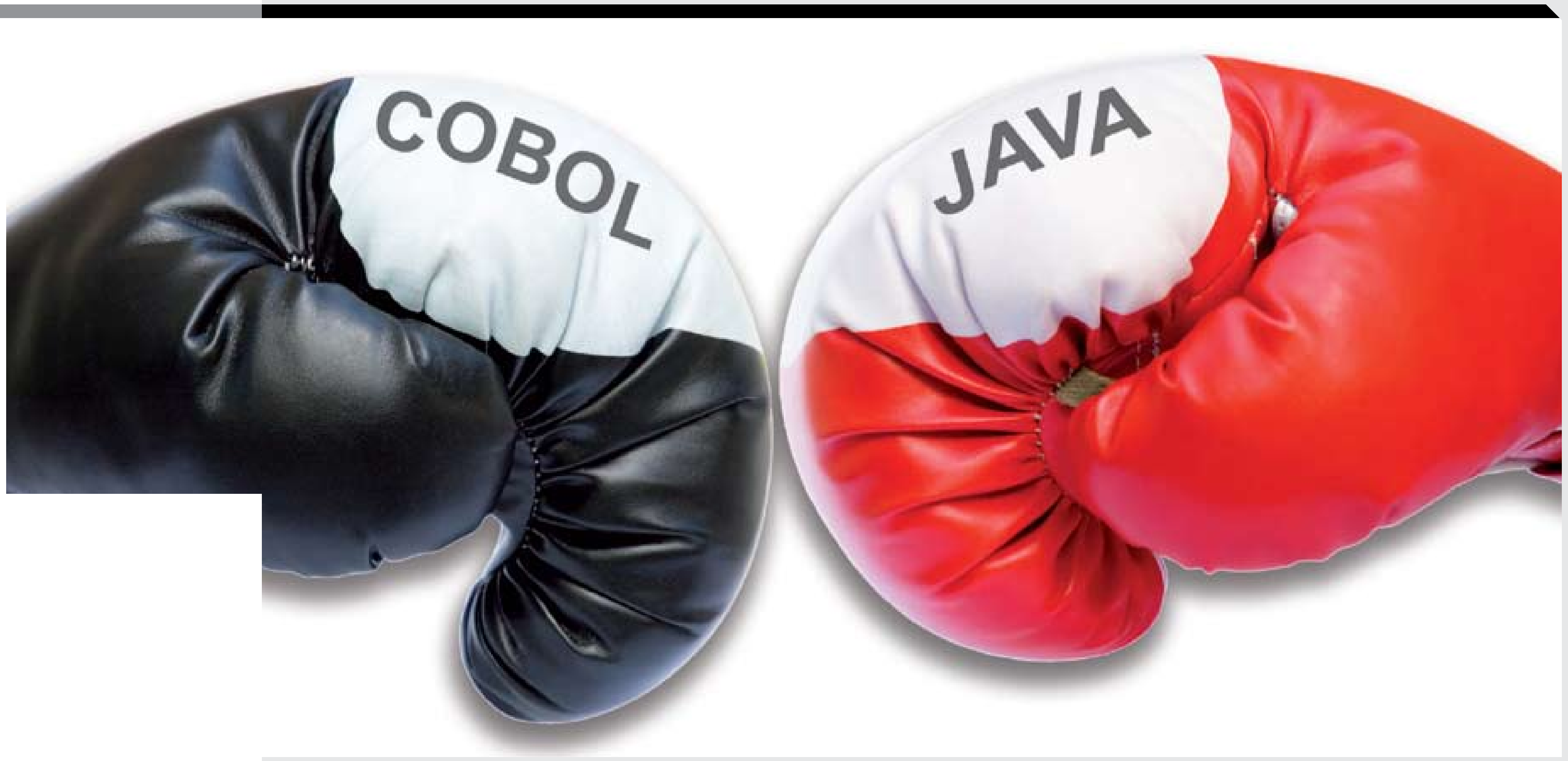
When two worlds collide

The difficulties with testing enterprise applications are compounded by the fact that many core systems span language and even platform boundaries. For example, many major mainframe applications, running COBOL, are presented to end-users by a front-end Java application interface. So, what's the problem here. They're both well known languages albeit with different histories. One, COBOL, is deemed to be rooted in the past, while the other, JAVA, is currently stealing all the headlines.

Ian Barrow is currently a programmer who works with JAVA, while Charlie Grant works in COBOL in a financial company which has recently re-hosted to a Windows environment. Derek Britton of Micro Focus referees an intriguing interview, as our two contestants slug it out – discussing the merits, or otherwise, of both languages.



Derek: We are here to discuss what's different, good, or complementary about your two different languages. So who would like to lead with a question?

Ian: I would start by saying the big difference that exists in our two working environments is agility... the fact that I use rich, productive tools. JAVA is a very visual and intuitive test environment... that's got to be something you envy, Charlie.

Charlie: Not at all, I already have what you have. I'm probably working on the same IDE as you? My preference is Eclipse but I also use Visual Studio. It just happens that the Apps I'm working with are COBOL-based. They were moved off the mainframe with very little disruption, and so now my interface is totally visual, with all the bells and whistles you have.

Ian: So, do you have Intellisense?

Charlie: Yep. Intellisense, auto-completion, background parse, red squiggles, the full IDE. For COBOL, it's the same as any other new language. And anyway, COBOL was the original write-once run-anywhere language and had multi-platform support long before Java was born.

Ian: Well, JAVA has grown up it's definitely now the programming language of the future.

Charlie: Of course Java is superb for building front ends.

Ian: Yes, and the code runs anywhere, it runs on tons of platforms, and it's high-quality code. We're talking about the difference between object-oriented versus the old-fashioned procedural techniques of COBOL. Who really does that anymore?

Charlie: Well that's a lot of claims. I said JAVA is ideal for front ends but that doesn't mean COBOL isn't better at running the back end. It always has been. And as for the quality of code, well, we've been churning out high-quality enterprise systems for many more years than you guys have.

Ian: It's still a bloated, verbose language though, admit that.

Derek: ...that is an argument that is often levelled at COBOL, Charlie.

Charlie: Sure, many people think it's a verbose language. That's probably because we have billions of lines of code out there. But it's only as wordy as you need, and more to the point it is readable. And also, as it's syntax driven, you can do a one-liner in a COBOL program like DISPLAY "Hello World" and it compiles and runs... interestingly Ian you could be up and running with COBOL in a couple of hours flat: because you can read it, understand it, and use the same IDE to code it, there's nothing stopping you.

Ian: Well it might be easy to pick up but, look, you basically can't conceive and implement Apps quickly using a small team with COBOL. JAVA does that and that can only be good for our business users.

Charlie: What's really important for the business is the value of the business logic and data that already exists, getting at that, and using it in different ways. Who would choose JAVA to try to build an entire new banking application for example?

Derek: Ian sorry to interrupt, I want to move you on to another important area, and that is testing for mobile and cloud, anything to be said here?

Ian: Well obviously JAVA will be on mobile and in cloud. It's a mainstay language there for development and test.

Charlie: You're right there, Ian. Java is designed for that and that's why COBOL sticks to the back end business logic. You can now leverage COBOL systems wherever they might be running by interfacing with new technologies like web services, for example. So your back-end core business system can be accessed by a web portal, a mobile app, or whatever.

Ian: You make that sound so easy, yet it can't be.

Charlie: OK, you're thinking 'how do I call COBOL from Java when a mainframe App has all of these weird data types I can't use from Java'? Well, we can talk language-to-language. You can create interfaces to COBOL programs that Java developers can use without worrying about what they are calling.

Derek: That's an interesting proposition. Just to give you both a chance to finish this off, I'd like to ask what the future holds employment-wise for developers such as yourselves.

Ian: That's easy, I think the JAVA developer has never been more in demand.

Charlie: And that's actually true of COBOL too – no Ian it is! – really. There's a lot of major enterprise software projects out there. And I think programmers and testers who have COBOL and new programming language skills such as C# or Java are well placed to take a lead role in that.

Ian: So who knows we might even up working in the same team!

Charlie: Really, it's not so crazy as you think. I know I'm collaborating more now with other programming teams, and using testing technology that bridges COBOL and Java environments, there are a lot of new composite applications out there.

Derek: Well let's shake hands on that thought please, gents. Sounds to me like there is space for both languages to flourish, and start to cooperate more, both for developers and testers. Thank you for giving your time to this debate.