Don Estes, Senior Principal, Maven Wave an Atos Company

Avoiding the Perils of Digital Transformation

Too often, people conflate 'mainframe' with the legacy applications that run on the platform. But experience has taught us that 'mainframe', 'legacy' and 'technical debt' should not be considered as synonyms. We have seen 5 year old, heavily maintained Java applications groaning under a load of technical debt as bad as if they were 30 year old COBOL applications.

Getting to Java (or C#) is not a panacea by itself, just as extending your legacy applications to the cloud by itself will not capture the benefits of digital transformation or insure you against future technical debt.

In order to fully reap the benefits of digital transformation, you should first and foremost distinguish between business logic and technical logic in your code. Most programmers push back on this distinction, because to them all the logic is technical. But by changing our thinking, we can reach insights that translate into hard business benefits.

We define business logic as decisions made using only external data, i.e. data from an input message or from a database. Next, the changes to data that result from each decision are also stored externally. Everything else is technical logic.

This is a critical distinction, because the business logic changes when the business processes themselves change. Technical logic changes when the application changes in some way, such as transforming a COBOL application to Java and re-deploying it on the cloud. By fully acknowledging this distinction, we can better manage the risk to the transformation project itself as well as the risk to the business.

Two-pronged approach

A digital transformation is a modernization project that incorporates digital technology wherever possible to increase productivity. We want to change the technological implementation to modern standards, and we want to change the business process to fully leverage the opportunities of digitalization. However, changing both at the same time is a recipe for excessive project cost, delivery overruns and for functionality shortfalls – or even outright project failure – especially if your business is one that runs on complex business logic.

This distinction provides a potential solution to this conundrum, implementing the digital business vision without taking on the risk of failing to deliver a fully functional system on time and on budget. The only complete and correct specification of that intricate business logic upon which your business depends is contained within those mountains of legacy spaghetti code. All subject matter experts combined will be unable to specify all that logic to a 100% standard. It is simply too complex, sometimes mind-numbingly complex, and has built up over the lifetime of the application. It was never completely written down.

But in a modernization project, we care very little for the technical logic. That is mostly or entirely to be replaced, so we don't need to understand very much about it. Since technical logic can easily be 80-90% or more of your legacy code base, business rule extraction that can largely ignore the technical logic proceeds at a rapid pace. Once we have the business rules, and we have them 100% complete and 100% correct as delivered by our patented dynamic business rule extraction, then we can change them to support the new business process design without incurring the risk penalty discussed above. The cardinal rule is: first do no harm.

Parallel tracks

The secret here is to conduct the design for the new digital business operations in parallel with the business rule extraction process, so that they both finish at about the same time. New transactions and new user experience can be rapidly constructed with modern tools, especially if using a combination of a rules engine for the business logic with a drag-and-drop low-code or no-code system for programming the technical logic. Getting a project ready for delivery has much less to do with developing the new system than it does with getting the results of the new system correct in all cases, but this problem has already been solved by dynamic business rule extraction.

Business rule engines and low code platforms assist, but are not necessary, to fulfill this modernization approach. The goal is to separate business logic and technical logic – and to keep them separate in the future. These benefits can be captured with a coded solution (e.g., Java, Python, etc.) with microservices encapsulating the business logic, as well as with a low-code/no-code solution that essentially eliminates code in its entirety. Regardless, both solutions fully mitigate the primary risk of a digital transformation, while ensuring fully correct business functionality in the new system.