I have always had a soft spot for CASE (Computer Aided Software Engineering, although I’d prefer to make that “Systems” engineering, myself). CASE modelling, done properly, can be a robust basis for systems development. It lets you think through the “essential” model of the business system - that subset of its characteristics that makes a widget processing system, say, a widget processing system and not something else – and remove the most fundamental defects of all (misunderstandings of what the business does) before you invest in expensive technology. Note, that I’m not bothering with any distinctions between “lower” and “upper” CASE tools here – the Wikipedia link covers that if you are interested.

There were real issues with early large-scale CASE tools – which were usually associated with mainframes – including lock-in to expensive proprietary technology, a need for computing power that often wasn’t easily available, rigidity (the inability to cope with the idea that not every detail of the system is known when you start) and so on; and these brought the CASE concept into disrepute in many quarters. But these are really implementation issues – companies like Uniface managed to make model-driven development work, and are still around today, producing web applications for large enterprises. And companies like OutSystems and Mendix have updated model-driven development into the PaaS world – they wouldn’t use the term CASE, but you could think of them as smaller, more agile CASE tools.

Perhaps the time has come to revisit the CASE world on the mainframe. CA Gen is a case [sorry] in point. Originally, it was IEF (Information Engineering Facility), which was used successfully by Texas Instruments and marketed to a select group of large-enterprise customers, especially in government. It has been re-branded several times in its life (initially as Composer) and sold, first to Sterling Software and then to CA Technologies, where it became CA Gen. It is still used and still being updated (CA Gen 8.5 Incremental Release 2 was delivered in December 2014). While still largely a mainframe tool (half its customer base continues to rely on the mainframe), in addition to the DB2 DBMS and Cobol, it now supports, for example, Linux, Oracle, Microsoft SQL Server, ODBC, JDBC, C# and Java code generation and ASP.NET web clients. It supports component-based development of modern web applications, as well as the creation of client/server applications, so there’s a mainframe modernisation story here, re-using tried-and-tested mainframe assets.

It is worth noting that model-driven development (which is more or less what CASE is) does require a certain discipline – you really need to maintain applications at the model/repository level. This is despite the availability, sometimes, of “round trip engineering” these days, which can be useful but is tricky if you rely on it, because you can make changes in the code which compromise the original model when converted back. If you generate code and then maintain it at the code level you can easily produce chaos and the model will probably become worse than useless, because it will either become unreadable or no longer correspond to the code and will thus be misleading. You will then blame model-based development or your tools, instead of your own inability to follow its disciplines.

Model-based development requires an appropriate level of commitment and discipline; and if this doesn’t suit your culture (luckily, since we are talking mainframe modernisation here, mainframe cultures are more likely to be appropriate than some others) then you had best try something else. Nevertheless, model-driven development can be very productive and it can quickly and efficiently generate low-defect, mission critical, systems, if used properly.

CA Technologies now sees CA Gen as a major mainframe application modernisation facility - development of new mainframe applications is declining but mainframe applications can be a fruitful source of robust services for, especially, modern mobile-based business automation environments.

This is achieved by linking the robust CA Gen environment (with its repository basis and built-in configuration management) with other CA Technologies tools. Notably, these will include the CA API Gateway and the CA App Services Orchestrator, to manage and combine existing services into new RESTful services; and CA Application Life-cycle Conductor, to mange a DevOps-style, feedback-controlled Design - Develop - Build - Deploy - Monitor - Re-factor the Design lifecycle.

I think the mainframe CASE tool concept is coming of age now – again – but please don’t call it CASE, that sounds like it is old-fashioned, and it really isn’t. Just take a look at CA Gen for yourself, here – and other model-driven development tools such as Uniface, Outsystems Platform and Mendix, for that matter – and see what you think.

David Norfolk
Practice Leader,
Development and Governance