The Benefits of Model Driven Development

MDD in Modern Web-based Systems

Written by: Michael Azoff

Published March 2008

▶ EXECUTIVE SUMMARY

Model Driven Development, CA Gen, and Mission-critical Applications

Model Driven Development (MDD) is the next step of abstraction in writing software applications. In the history of computing, each higher level of abstraction has offered improved productivity and ease-of-writing applications. This has enabled the creation of far more complex applications without increasing the project management work load, or conversely, building like-for-like applications faster, cheaper, and with higher quality, compared with lower-level techniques.

Thus with MDD there is a separation of the model from the code, the user works on a platform independent model, selects the specific target platform, and the tool generates the code. Productivity is raised because the auto-generation takes care of all the plumbing code. There is also only a single language to use in building the model, the auto-generation will take care of SQL database calls, Java or .NET platform specific code, XML, and JavaScript for Web clients etc. This makes it easier to rapidly build applications and avoid the code testing cycle, as code is 100% generated. Naturally, testing still needs to be performed to test the design against the project requirements (and in mature testing environments the requirements are also tested).

CA Gen is the MDD tool from CA that has a long history of successfully delivering applications: many of the early mainframe applications written in CA Gen are now benefiting from this choice of tool as models can remain unchanged and the code re-generated for newer, distributed computing platforms. The out-of-the-box model lifecycle management functionality means that best practice is in-built. For example the instant impact analysis feature ensures that the impact of changes is understood and leads to better model design. CA Gen has an easy-to-use business logic language that developers find robust and quick to pick up.

The strongest case for CA Gen is made when there is a need to build mission-critical applications that will work first time. The development team sizes are small, even for the larger-scale projects, making development more agile and saving on human resource costs. There is benefit to the business in lower lifetime costs, as maintenance costs are relatively modest and CA Gen applications are less likely to fail.

There is a move in IT towards Web-based applications and Service Oriented Architectures (SOAs) and having already assisted numerous customers in making this architecture transition, CA is expanding its commitment to SOA with the next release of CA Gen r8. There is also a transition process in play to move the tooling to the Eclipse platform, with the option of launching Adobe Dreamweaver from the new CA Gen Studio. The CA Gen WebView project (an internal r8 project name) is also promising direct modelling of Rich Internet Applications (RIAs) and Web services.

Application development moves with the needs of business: to deliver faster to market, at lower costs, and on the latest infrastructures. CA Gen is addressing these issues by ensuring that if the user interface is not part of the model then it can integrate with other client-side solutions. Web services are playing a role in this respect and the next release will offer greater options.

CA Gen deserves to be better known and increasing awareness should not be difficult for what is in my opinion a gem of a product. One way to achieve that is through its involvement with the MDD community, as it evolves its standards and tools: CA Gen is a good example of how to do MDD right.

▶ THE EVOLUTION OF APPLICATION DEVELOPMENT

Introduction

In the many years I have worked in IT I had never heard of CA Gen until CA invited me to be briefed on the product. As I learned more about CA Gen, talking to users, partners, and the CA team behind it, I was quite surprised that I had not heard of the product before given its proven track record in the field. In attempting to understand why, one particular analogy struck me about the application development world: that of an unruly classroom where the teacher's attention is focused on the misbehaving kids, while the ones that work do so quietly in the background, unnoticed. The 'bad' kids of application development are the C/C++ coders who have pointer problems or memory leaks. If a C/C++ application does irrational things then you know a memory location has been over-written, even adding a comment line to the source code will move the memory alignment and cause the problem behaviour to change. Java, Microsoft .NET languages, and other memory-managed languages improve the situation, but the high skill levels and resources needed to create hand-written code lead to the common situation today of software failure write-offs amounting to billions of dollars. This loss to business comprises the cost of dead-on-arrival software, end-users having to work around faults, developers having to re-work faulty code, and the impact to business and its bottom-line due to mission-critical application failure. The well-behaved kid is the code generation tool that works at the model level – it generates rock-solid code that works first time: in my opinion, CA Gen should receive greater attention for these strengths.

The idea of code generation and fourth generation Computer Aided Software Engineering tools has not taken hold in application development beyond niche areas. One exception in successfully delivering on its promise and not suffering from the lock-in and proprietary disadvantages of this type of tool is CA Gen, based originally on a product first launched in 1987 by Texas Instruments, Information Engineering Facility. The tool today has three prime aspects: it is a lifecycle tool, so management of the development lifecycle is out-of-the-box; it is model-driven, so it abstracts development to a higher level than code; and thirdly, a consequence of being model-driven, it is platform independent, so CA Gen can generate 100% Java code, 100% .NET code, and target a variety of platforms, including Windows, UNIX/Linux, and mainframe platforms.

MDD in general also emerged successfully from the fourth generation tool era, to the point where today there is a unity amongst a wide number of modelling experts and tool vendors around the Unified Modelling Language (UML), led by the Object Management Group (OMG) and its Model Driven Architecture (MDA) standard. However, MDD does not always prove popular with hard core developers, as many of them prefer to code rather than build models. It is in the large enterprises with large scale project requirements and certain verticals, such as defence and aerospace, that one is more likely to find modelling activity. Having understood CA Gen's approach, I offer some insights in this White Paper into what is hindering MDA and what has been successful about CA Gen's approach to modelling.

The structure of this White Paper is to provide some background on how application development has evolved, describe the approaches in MDD, explain how CA Gen fits into the larger picture of application development, and relate customer experiences to underpin the quiet success I alluded to above. The move within IT in general towards adopting SOA and Web services provides CA Gen with a tremendous opportunity, as the product fits in perfectly with a world of loosely-coupled composite applications, and allows users to mix the latest and trendiest on the front-end to a back-server code of high reliability. I shall expand and elucidate these statements.

The question arises of how CA can generate greater interest and awareness in CA Gen: as a number of people in this research pointed out, when Texas Instruments first launched the product it was hot and trendy, today developers at large want Java and .NET on their CV. There is an answer to this question, and it goes beyond just creating awareness for CA Gen, it addresses the endemic problem of software failure: Frederick Brooks said there was no silver bullet to this issue but my contention is that modelling can elevate abstraction in application development to the next level that mitigates many of the existing problems with manual coding (the misbehaving kids). CA Gen is an example of how this has been successfully done and I shall explain what aspects of the tool give me most reason for optimism about the future of CA Gen.

From Mainframes to Distributed Systems

In the late 1980s when CA Gen was first launched (under its original name and owner) the PC was still in its first decade (having been launched in the summer of 1981) and the Internet was only to take off in the early 1990s. Thus the mainframe held sway in corporate IT and the ruling IT infrastructure paradigm was that of centralised systems and applications, and client sides that were essentially user interfaces (the green screens). The introduction of client/server and then distributed systems led to a peer-to-peer paradigm that created the foundation for today's ruling paradigm and one that led to the growth of the Internet.

Today, there is a role for all three classes of systems: mainframes and client/server/multi-tier – these two categories now coalesce as the client side is now standard and the server side is mainly a question of scale – and Web-based systems running in SOA.

Application development paradigms have also evolved, the most recent prevailing one being object-oriented design and programming, but we are now witnessing a new paradigm in the ascendant, that of service orientation. Following this evolution, and with it also the progression of languages from assembler to today's high-level, memory-managed languages, it is apparent that each significant advance raised the level of abstraction one notch higher, by automating what was previously performed manually, and by making what was left as a manual task one that could be done more productively. This raising of abstraction is what modelling can achieve – the key is to get this done right.

"Our developers find CA Gen easier to use than a third-generation language, it has English-like syntax and the focus is on coding the business logic." – Stacy Pickett, ENTARCO USA.

In tandem with these evolutionary trends, high productivity tools were introduced, named fourth generation language (4GL) tools that often introduced proprietary languages. However, the lock-in to the vendor's proprietary platform meant these tools never gained widespread use. CA Gen grew from the 4GL movement but differed in one important respect, it was designed to be platform independent from the start: the code generation was separated from the model, and this meant that as platforms gained popularity, CA simply needed to add the code generation capability to the tool in the next release, and customers could generate on the new platform without changing the model. This approach led CA Gen customers to significant savings, as changing platforms normally require considerable investment, particularly for large organisations, running to millions of dollars.

The Era of Web 2.0 and Service Oriented Architecture

The introduction of Asynchronous JavaScript and XML (Ajax) methods and technology has led to Rich Internet Applications (RIAs), providing a rich, desktop-like user interface experience to applications delivered via the Web browser. RIAs are Web-server based applications with numerous benefits:

- A single location to maintain and administer the application;
- "Rolled-out' via the Web enabling easy deployment;
- Can be accessed anywhere with Internet connection, providing easy reach – whether for office-bound workers behind the firewall, or mobile workers in the field.

The latest generation of RIAs can now work out-of-the-browser, so have the added benefit of working in offline mode and can fully interact with the local machine. The concept of Web 2.0 was enabled by the RIA technology that underpinned it, and this has also led to the transfer of Web 2.0 ideas for business use – hence the Enterprise Web 2.0 label. Web services play a key role in Web 2.0 and are also key to SOA – although not wholly, as composite applications can rightly also be built from distributed components.

One of the key business drivers for moving to Web service applications, whether as part of a comprehensive SOA rollout or as part of a piecemeal adoption of Web-based techniques, is the drive to reduce cost. A Web interface is cheaper than bricks-and-mortar, a customer interacting with an automated screen service is cheaper than one involving client-facing staff.

"Our SOA implementation is acknowledged as a global first on an enterprise scale (1997), built using CA Gen. SOA has led to significant service re-use and consequent efficiencies." – Paul McRae, Queensland Transport, Australia.

The rapid change that the Web and SOA are introducing can create confusions about: which platform to adopt, which strategy and tools to use, and what architecture to build. With any new approach the scope to get things wrong or fall into pit-holes is higher. Thus while the scope for cost savings is high, the risk factor can also lead to equally high failures. Mitigating risk therefore needs to be on the agenda.

How to Reduce Risk with Increasing Application Complexity

The track record of traditional styles of application development is poor, especially for the larger scale projects. According to a report from the US National Institute of Standards and Technology (2002), the economic impact of defect ridden software amounted to US$22-60 million annually, resulting from end-users taking mitigation action, such as work around, and from development re-work. This estimate is conservative as it does not include the impact on the business, for example, failure in mission-critical applications and consequent revenue loss.

The introduction of Enterprise Web 2.0 and SOA compounds the risk factor as issues of Internet security, lack of maturity in the space (and hence uncertainty of best practice) play a part. New Web service / SOA-based applications and services are also more difficult to test, as external and third-party Web services are consumed by a development unit. Questions of how secure, reliable, and tested a service is are difficult to answer without IT governance in place, all adding to the risk factor.

There are a number of ways to reduce risk. One way is to focus on process and methodology, for instance adopting an Application Lifecycle Management (ALM) process, which includes practicing Requirements Management and Change and Configuration Management (CCM) – being able to trace a requirement to the developed code and being able to automatically see the impact of a change in the requirement or a piece of code. Many new practices stem from software project and construction fault lines. Agile software development aims to mitigate the effects of change by building adaptability in to the process, as well as reducing requirements to a minimum set of core essentials that allow coding to start immediately. Test-driven development is another approach designed to improve the code testing regime.

"Since adding a Web-interface to our CA Gen application, 85% of the business is now generated online. The switch to a Web interface was easily accomplished through use of Web services." – A CA Gen customer.

While these trends in process and methodology help traditional manual code development, either the risk factor remains too high or the cost of investment in mitigating these problems is prohibitive. MDD can play a significant role in reducing risk and this is examined next.

▶ THE DIFFERENT APPROACHES IN MODEL DRIVEN DEVELOPMENT

The Role for Modelling

Traditional application development faces a number of obstacles: there are too many programming languages in use per project, some may use as many as four or five, and most organisations with in-house development units will build using a variety of languages, with Java and .NET being the most popular. There is also a multiplicity of platforms: UNIX/Linux, Windows, Mac, mainframe etc, all with multiple versions and releases. The churn rate on platforms and languages is also quite high, for example, we are witnessing a decline in C/C++ and increase in Java and .NET, plus the various dynamic scripting languages are jockeying for position: PHP, Python, Perl, Ruby, and JavaScript. In contrast, there are mission-critical mainframe applications built in COBOL and PL/I that are still running after decades, but the need for modernising these is growing.

MDD is one approach that mitigates against the high risk of application development, the complexity of the live environment, and the multiplicity and churn rate of platforms and languages. It does this chiefly by separating out the concept, or model, of what is to be built from the specific detail of the implemented platform. Advanced forms of MDD such as MDA and CA Gen go one step further, as they generate the code automatically from the model. There is therefore no longer a need to invest resources in traditional coding, all the human effort can be focused on the requirements, design, modelling, and high-level business logic coding – the more complex lower-level coding is left to automation. CA Gen goes one step further than MDA: it has out-of-the-box lifecycle management built-in (some segments that are missing are supplied by the CA Gen eco-system partners).

MDD has the potential to lift application development to the next level of abstraction. Models hide 'plumbing' details, while highlighting the important concepts and essential business logic. This aids in understanding, in communicating intentions among the various stakeholders: the business customer, end-users, designers, and developers. High-level models can also provide a visual way for business people to understand how IT applications are built, and create opportunities for novel business processes and applications to arise as a result of clearer understanding. The separation of model and implemented code is perhaps the single most advantageous aspect of MDD, and the OMG and CA Gen way of achieving this is examined next.

The Object Management Group Way

The OMG's MDA standard is important by the simple virtue of being one that the majority of experts and tool vendors support. This means for example that the workforce can train in UML and that skill will have currency and be transferable across a wide range of tools, creating a ready resource pool. Standards also ensure compatibility and interoperability of tools.

The OMG and its standards are crucial for harmonising the evolution of open standards, such as the model repository, model syntax, and the Platform Independent Model (PIM) and the Platform Specific Model (PSM), all based on the Meta-Object Facility (MOF) – a meta language from which other languages, such as UML, are derived. The latest trends within MDA are to develop Domain Specific Language (DSL) capability, and to include Object Constraint Language (OCL) capability (this is the UML expression and query language).

The take up of MDA has not been universal, and there are quite divided opinions between modellers and developers who prefer coding. There is variation amongst vendors on the extent to which custom code needs to be added to complete the application, and how this code is managed. The existence of custom code means that the customer has to support PSM coding capability, e.g. Java or .NET. The UML 2.0+ version is a significant forward step, as OCL is part of the specification. The capability to combine UML and OCL means that the model is sufficient to define the application and no custom coding is necessary. The introduction of DSL and OCL within MDA, and the capability to execute models and test them before generating PSM code, takes MDA to a level where it has truly raised the abstraction level beyond traditional programming.

Perhaps the biggest handicap holding back MDA is that it is just a part of the lifecycle. To complete the ALM tooling, customers have to invest in an ALM suite provided by the MDA vendor in order to gain maximum benefit of the integration between modelling and other lifecycle stages. The alternative is to buy best-of-breed solutions, but the question of inter-operability creates new problems.

There is another aspect to MDA, which is that it is still an evolving community effort, and the investment by tool vendors to deliver tools is not happening at a uniform pace: some vendors are taking a wait-and-see approach and not always releasing to the latest UML version, while others are whole-heartedly engaged in pushing to the next level. As a result, the style of MDA tools available today is targeted more at the experts, and easy-to-use tooling, the kind typified by the old Visual Basic, is lacking. Given that the aim is to attract a mass army of developers, a more intuitive tool interface approach is necessary – thus the time to speed in using the tool is an important factor.

In conclusion, MDA is important in that if modelling is to progress beyond niche areas and attract universal adoption, raising development to the next level, then an open standards approach is crucial. It is also clear that MDA needs to be steered in the right direction. We look next to see how CA Gen approaches modelling.

The CA Gen Approach to Modelling

In many respects CA Gen parallels MDA and in other equally important respects it does not. The idea of separating out model and code started with CA Gen before MDA existed. Like the more advanced executable form of MDA, CA Gen generates all the necessary code from the model (this being its key feature from first launch). Platform independence is key to the longevity of CA Gen models, so as platforms change CA Gen simply re-generates the code for the new platform. Where CA Gen differs is that the use of diagrams and a native business logic language were in use from the beginning – with MDA the need for OCL has only recently been addressed.

CA is a participant member of the OMG and MDA committees, and its plan going forward is to standardise where possible along the MDA standards. This means, for example, that CA plans to add UML diagramming in CA Gen and give end users the option to switch between native and UML modes. Moving CA Gen towards MDA standards is important because only standards-based modelling products have any chance of gaining mass acceptance in the developer community. There are UML add-ons available from the CA Gen partner eco-system so it is possible to model components in UML and have these models generated in CA Gen today.

With CA Gen, diagramming and a natural language are used to express business logic. The low-level code is fully generated and for over 95% cases there is no custom coding necessary. When there is a need to call upon external services there is a code stub facility (called External Action Block) that allows external services or applications to be called.

CA Gen customers find that new team members can be trained up in Gen development within 6 months, having only a basic background in programming.

Similar to MDA, the PSM segment in CA Gen targets popular platforms such as Java and .NET. One of the key issues in application development is building the user interface. In the past CA Gen took a common denominator approach in ensuring that a model could be generated equally well to all the supported target platforms. As many platforms proliferated in the past this approach made sense, however, there is today a move towards Web-based interfaces running against Java or .NET (using a possible combination of Ajax and browser players such as Adobe Flash). As there are unique features that are particular to each of Java and .NET, taking a common denominator approach can lose look-and-feel. For this reason CA is moving CA Gen towards generating platform-specific models that exploit the full features of key target platforms.

In the pipeline for CA Gen r8 is extensive new Web functionality to support Web services and SOA projects, offering three options. First, hand-written UI code will be supported, for example, built with Adobe Dreamweaver and connected to CA Gen through proxies or Web services. Second, fully CA Gen modelled and maintained Web applications will be supported for Java and ASP.NET Web clients. Third, the WebView project is a new type of CA Gen Web client that will offer RIA features built with Dreamweaver, and be able to generate and consume Web services in CA Gen models. The new Eclipse-based CA Gen Studio will integrate Dreamweaver for the UI creation.

Customers have the flexibility to build the server-side of their applications with CA Gen and use Web services to connect to Web-based clients built with external tools. Future releases of CA Gen will offer more options for model-based Web client creation.

CA Gen Studio is the forward plan for the new generation of CA Gen tools, and will offer an environment familiar to many developers, as well as provide the opportunity to plug into the wider Eclipse platform eco-system. CA Gen is making an important transition to the new mode of application development, addressing RIA, Web services, and SOA compatibility.

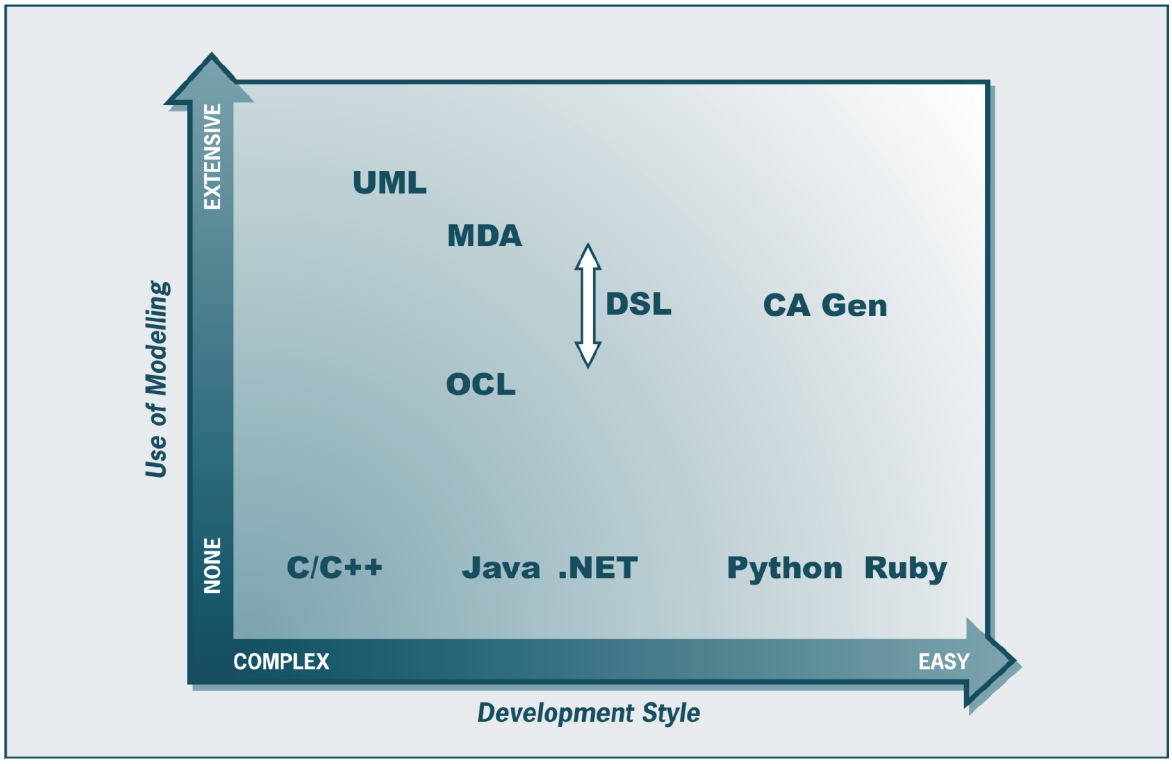
The CA Gen Partner Eco-system

The CA Gen partner eco-system also plays an important role in supporting the tool, providing add-on products with features that enhance CA Gen, improve integration with other tools, add ALM segments not available in CA Gen, or add new functionality for building CA Gen applications. CA Gen partners also provide consulting expertise and have a local presence that allows them to reach out to the business as well as IT, and address the differing styles of development solutions that are dependent on geography or vertical industry.

CA Gen's partners provide tools that enhance CA Gen functionality, for example: IET offers GuardIEn, a CCM tool for CA Gen modules; Jumar has a Jumar:Links for migrating UML models and linking CA ERwin Data Modeller to CA Gen; COOLProfs offers WebserviceConnect to consume Web services in a CA Gen application; QAT provides WebDaptive for building rich-Web clients with CA Gen, and QAT Wizard builds CA Gen applications in an interview style.

Thus CA Gen is not just a development tool, it offers a value proposition – delivering value over the application lifecycle: longevity of models that can be easily transitioned to new platforms, productivity of the CA Gen development environment, and the ALM functionality available out-of-the-box. Figure 1 maps the various development approaches, and shows that CA Gen's combination of easy style of development and MDD auto code generation marks it out in the field.

Figure 1: Mapping Application Development Approaches



Source: Butler Group DATAMONITOR

▶ THE FUTURE OF MDD AND CA GEN

Opportunities for MDD

Looking at the evolution of application development, when Fortran, COBOL, and C languages first appeared, no one carried on writing in assembler, bar niche embedded software developers – the move to higher level languages was compelling and universal. The move to MDD needs to be equally compelling and universal, and that means getting the approach right: the MDD tool needs to be as versatile as writing in the lower-level language, but have a productivity factor that makes using it compelling. MDA is still evolving and while keeping to its standards is essential to maintain critical mass in modelling, there are strong attractions in using a proven product with a long history.

Application development today embraces a number of areas:

- Modernisation of mainframe applications.
- Custom integration to SOA and Web service integration.
- Mission-critical applications.
- Application component re-use, Component Based Development.
- Server-side back-office applications.
- Enterprise Web 2.0.

It is possible for a business to dissect the styles of development according to many factors, including the pace of change, longevity of an application, and developer skill sets. Thus one may deliver portions of an application that have a shorter life span and require leading edge technology (for example Web interfaces which are largely about cosmetics and ease of interface use) using a manual coding approach and use MDD for the back-end servers that run the mission-critical business logic. The challenge for MDD tool suppliers is what functionality has sufficient market interest to justify investing in providing a PSM. End users can decide to not run with the latest client-side trends but rather build proven and stable clients – MDD tools can then offer a variety of client-side options out-of-the-box.

"In our industry margins are low and turnovers are large, so we cannot risk large jumps in IT – any changes must be stable and work first time, this is the strength of CA Gen." – A CA Gen customer.

Naturally, any manual coded portions in a project lose out on the agility that MDD offers when changes to platform or business logic occur (and such change is virtually guaranteed), as MDD can quickly re-generate code. However, end users have the flexibility to choose which strategy to adopt that meets their business needs.

There is another factor: making the MDD tool compelling means enabling the majority of programmers to be able to build first-class applications, and not require the skills of a top few elite. The advantage of MDD and CA Gen in particular is that it is more difficult to generate bad code, preventing for example manually entered typos that are syntactically correct and hence difficult to detect – CA Gen will never generate syntactically incorrect code. Where you can go wrong is in the application design, e.g. does the design correspond to the requirements, but that is where the focus should be, instead of having to yet again write plumbing code and risk introducing bugs. Tools like CA Gen that are robust, easy-to-use, and let the developers focus on the business logic, are what make MDD compelling.

Opportunities for CA Gen

In the early mainframe days the original CA Gen product, as launched by Texas Instruments, was heralded as an innovative and leading product in MDD – what is interesting is that the product through its evolution still maintains an advanced position in MDD. Although some developers may prefer to code directly in the Java and .NET platforms, partly because they feel this strengthens their CVs, it is from a business perspective that the benefits of MDD need to be considered when making application development decisions.

"We built a service interface layer with CA Gen that sits between the data model and the front-end, so changes to the data model can be made without affecting the front-end, and similarly changes to the front-end do not affect the back-end services." – Shibashis Mukherjee, Con-Way, USA.

The overall cost *benefits* of adopting CA Gen are significant over the lifetime of an application, when the maintenance and support phases are included: customers have talked about over 50% reduction in cost compared to manual coded development, and far smaller project teams to manage development and maintenance of large-scale projects, with consequent resource savings. This longer term value naturally benefits the business, and IT departments have the challenge of balancing short-term budget constraints with long-term total cost of application ownership. The problem with taking the cheaper upfront option is that the business suffers over the long term with poor quality issues, high maintenance costs, and high human resource requirements.

The pattern of tool adoption also provides clues about how to increase market awareness. Speaking with CA Gen customers, the decision to purchase the product often initiates from word-of-mouth recommendations from other customers, followed by pilot projects; as a result there is a tendency for usage clustering to occur, as has happened around public sector and government projects, and certain verticals, with the patterns varying across geographies. The decision making is generally made either by a consultant partner familiar with the product or by the customer's IT department.

Example CA Gen customer team sizes: a team of 10 to 13 manages 400 online screens, with about 443 batch programs and 677 non-batch programs; another customer has a maximum team size of 25 comprising 10 working on CA Gen and 15 engaged in non-Gen related activities such as database admin, and OS running – this project has over 2.75m lines of code; a third case has about 7 Gen developers and 4 business analysts working on a project with 2000 database tables; and a final example customer has a 25 strong team managing 1000 client-side screens and 150 database tables.

Thus there are a number of issues here: the business often leaves the decisions to the IT department and if the IT department takes a purely short-term view then the business will suffer. There is therefore a clear case for approaching the business as well as the IT department in getting the CA Gen message across. For instances where the business understands the proposition, there is an opportunity for a meaningful dialogue about how to reduce IT costs and gain high-quality application development capability.

The second benefit of talking to the business is that the human resource question now is not hidden within the IT department's decision-making, which as pointed out, is often led by CV considerations. Rather, the business can now initiate suitable human resource fulfilment for CA Gen. This in turn creates a demand for developers skilled in the tool and leads to a growth spiral that brings CA Gen and MDD skills to the attention of the CV concerned developers, making CA Gen an enhancement to career prospects.

"The Dutch market is a huge success for CA Gen and it is noteworthy that the approach there is not about selling a development tool but in offering a solution with a high degree of confidence; providing tooling, training, support, and sometimes a deliverable as well." Darius Panahy, IET.

Going forward there are challenges for CA in widening awareness of CA Gen, one suggestion mentioned by a customer is to see books on the tool and to have more tutorials downloadable from the support site. Adopting OMG MDA standards will certainly help as UML is strongly supported by books and training courses, and benefits from a good general awareness. The pressure from no-upfront-cost OSS development tools as well as availability of reduced fee or free tools for college students are tactics that are playing in the market. There are a number of possible options that CA can take in this respect, but the key is to create an upward path that will lead to awareness of the lifecycle benefits of the tool and its approach, and ensure that CA Gen is on the shortlist when development project decisions are being made.

▶ CONCLUSION

CA Gen Delivers Mission-Critical Applications

The point of strength where CA Gen starts from is in delivering high-quality, large-scale, mission-critical applications. It does this with fewer human resources than comparative projects using manual techniques. As a result of its MDD approach and built-in application lifecycle features, application lifetime costs are lower and resulting code is of higher quality and more reliable – as compared to manual techniques.

The next release of CA Gen will be an important one as CA is introducing major changes: moving tooling to the Eclipse platform, and providing multiple options for Web-enabling applications being paramount. The participation of CA on the OMG MDA committees is also important, as aligning CA Gen with open standards will win it new friends.

"Over a period of 20 years CA Gen has consistently delivered mission-critical applications while application development styles and technologies have changed all around it. It has only been able to do this because of the separation of model from technology." Doug Michael, Jumar Solutions.

In conclusion, CA Gen has maintained its leading capabilities in the MDD market, keeping abreast of the latest trends in Web-based and SOA IT environments. It deserves greater awareness and should be on the shortlist for development tools more often. There are challenges in an increasingly commoditised tools market, but when the total costs of ownership over the lifetime of major applications are taken into consideration, the CA Gen approach in particular and MDD in general, wins out.

▶ ACKNOWLEDGEMENTS

The research that went into this White Paper has benefited greatly from conversations with CA Gen customers, and CA Gen eco-system partners (in particular IET, Jumar Solutions, and QAT), as well as the team behind CA Gen at CA.

Contact Details

CA
One CA Plaza
Islandia
NY 11749
USA

Tel: +1 (0)800 225 5224

www.ca.com

CA European HQ
Ditton Park
Riding Court Road
Datchet, Slough, Berkshire
SL3 9LL, UK

Tel: +44 (0)1753 577733
Fax: +44 (0)1753 825464

E-mail: ukchannel@ca.com

Butler Group
a Datamonitor Company

Headquarters:

Europa House,
184 Ferensway,
Hull, East Yorkshire,
HU1 3UT, UK

Tel: +44 (0)1482 586149
Fax: +44 (0)1482 323577

Australian Sales Office:

Butler Direct Pty Ltd.,
Level 46, Citigroup Building,
2 Park Street, Sydney,
NSW, 2000, Australia

Tel: + 61 (02) 8705 6960
Fax: + 61 (02) 8705 6961

End-user Sales Office (USA):

Butler Group,
245 Fifth Avenue, 4th Floor,
New York, NY 10016,
USA

Tel: +1 212 652 5302
Fax: +1 212 202 4684

For more information on Butler Group's Subscription Services please contact one of the local offices above.