Publishing & Design

Celebrating 20 years | 1997-2017

Blog Post

Systems design with results, not software, in mind

An emerging methodology for designing, developing and deploying applications, Model-driven development (MDD) promises a faster, more efficient approach by going modular

Executive summary

Cost effective, future proof application development and deployment feels like a losing battle. Every year there are more providers offering more formats and solutions, there's more software under your roof that needs to work seamlessly with every other system, and you have to do it all with ever-tightening budgets and compete with ever-more agile opponents in your market. It might be time to look into model-driven development, an established and proven technique that's never been more relevant.

We tend to think of software and application development as being about lines of code – constructing functions from the ground up and finding the unique balance between tools that will fulfill business goals while staying inside the scope of budgets and manhours.

Model-driven development (MDD) instead gives you the capability to use 'building blocks' of functionality, assembling them into the applications you need according to very specific functionality wishlists, all without developers having to deal with software code on a line-by-line basis.

Think of pre-web terms like macros or libraries. MDD uses a similar thing – created and curated modules that slot together to build functions. It's a low-code, big picture solution that frees you up to meet goals rather than wrangle software.

Unlike object-oriented development where developers hand-write functions from scratch in languages like Swift, .Net or Java, MDD is a high level construction method, assigning needs that then filter down to the minutiae of code automatically.

A new paradigm that's anything but

As the methodology suggests, the biggest selling point for many is speed. When you're not writing code but creating or selecting from objects that represent the functionality of lines of code, the development process can be compressed from weeks or months into days or hours.

Ease of use is just as important to many users. Switching from one database system to another, for example, can be done in a far more automated way that impacts on your development budget.

And while longevity of performance is often a secondary consideration, an application codebase that crumbles a little more with each tweak or rebuild to fold in new functions can end up affecting your bottom line far more than you might think.

MDD offers robust maintenance. Curated software objects and modules rather than individual lines of code are carefully road tested as platforms are upgraded or expanded, so you can be sure new applications are going to play nice.

It also offers the means to design and deploy an application that just works rather than wrestle with multiple versions for screen sizes, devices, use cases or environments. The right MDD platforms contain parameters to automatically distribute the code base of your application out to any system or device without you having to develop multiple versions, letting you focus on the core functions you need to meet your business needs.

Client:

Altitude Marketing

Content type:

Blog Post

Brief:

To introduce readers to the concept and explain the advantages of an emerging business methdology.

Deliverable:

A written piece of fixed word count in the appropriate voice of the company presenting the idea.

Many MDD providers today can offer you modular building platforms that generate language-specific code, but that only leads you closer to the territory you're trying to avoid – of having to engineer software one line at a time when there's a better way.

Look instead for an enclosed ecosystem, a toolset that uses its own protocols and languages that already connect successfully to the system you need.

There are plenty of MDD systems that act merely as a programming layer, producing code in proprietary languages or formats at the user end, but doing so introduces all the pain points and risks associated with publishing multiple code across multiple deployments and devices. ■