Feedback: Jason.Stamper@computerwire.co.uk

Super models

While many development teams and their managers see modelling as a time-consuming, irrelevant chore, the demands on today's businesses mean they may no longer have an option. Jason Stamper investigates.

Anders Lidbeck, Telelogic president and CEO

Application development today bears little resemblance to typical software development strategies of just 10 years ago. Under increasing pressure to deliver higher quality applications in shorter and shorter timeframes, application development teams are looking to modelling to ease the pain of the application development lifecycle. At the same time, advances in both the unified modelling language (UML) and the tools that support it, have given software modelling a new lease of life.

Contrary to a popular misconception, modelling is not in itself a difficult concept to rationalise. Just as when building a house an architect will produce detailed plans, software modelling means visualising the design of the software in advance of actually getting started on the application development. But much of the image problem that application development has suffered has been down to standards. While any architect or builder has long been able to read and understand any architect's plans for a building, it has only been fairly recently that a standard way of modelling an application has emerged.

That standard is UML. Before UML a number of companies had modelling tools aimed at development teams, but they each had their own proprietary way of modelling a particular application. So, for example, while one modelling tool would signify an object with a triangle, and a

process with an arrow, another tool would signify an object with a square and a process as a line. Once the entire range of different items – known as software artefacts – was converted into the requisite symbol to suit each particular modelling tool, the models themselves became indecipherable to all but those familiar with the modelling tool with which they were created.

That meant that models were only useful to those with skills in that particular modelling tool, and that models could not be shared between teams or partners using different tools. Perhaps most crucial of all, however, was the fact that once a company had skills with one modelling tool, it was 'locked in' to that tool because it produced the only models its developers could understand. Similarly, if the skills required for a particular modelling tool left the company, all models – both historical and current – were rendered useless unless the company could replace those skills.

Changing expectations

The proprietary nature of modelling tools in the pre-UML era – and the fact that modelling tools were extremely expensive – left many companies turning away from modelling altogether, preferring to go straight to the build stage immediately after the requirements were set out. For a time, that may have been an acceptable way of proceeding. But no longer.

As Jim Rumbaugh, one of the three leading designers of UML and a Rational Software methodologist, says: "The brave new e-world has turned previous assumptions on their head, and old approaches to business or software will no longer succeed. The e-world is now distributed, concurrent and connected. Concurrent, distributed systems have extremely complex interactions that can be hard to understand, let alone predict. Vague specifications are a major problem.

"In the past, the specifications for a monolithic system only affected the single system, and if it didn't work exactly as specified, nobody really cared. But now a business system may have to interoperate with another system halfway around the world; both are written by people who have never heard of each other. A failure to follow specifications can introduce errors that propagate around the world."

The world has indeed changed, placing new, more stringent demands

Paul Levy, Rational Software CEO

on software development teams. Rumbaugh describes an 'e-world' that is distributed, concurrent and connected, and he is right. It is distributed, because information vital to a company's business can be located all over the world. It is concurrent, because business processes are no longer centralised and rarely simultaneous. As Rumbaugh points out: "Neither business decision making nor software programs can live with a single thread of control." Finally it is connected, because an action in one place can produce effects anywhere else within the organisation today. Put succinctly, the basic computer systems, languages, and models of the past are simply inadequate for today's needs.

So how can modelling with UML tools help to solve the problem? "The key driver for today's modelling tools is the ability to close the gap between business people and technical people," says Computer Associates' (CA's) Shirief Nosseir, AllFusion consultant. "Modelling helps to ensure that they are speaking the same language. If you build a house, the customer needs to speak to an architect and the architect needs to speak to the builder. If you ask for a three-bedroom house you have no idea how an architect will build it unless you sit down and look at the blueprint. The blueprint is all about taking a customer's requirements and putting them into a picture that all parties can understand."

"Businesses are realising that they have to model from a business perspective first and then drive that to IT.

Martin Owen, Popkin head of consultancy services

Business perspective

It is fairly obvious why technical people within a company need to understand what the business people are asking them to build. Building quality software means building software that meets the original business requirement. If the software development team does not understand the business need, it is likely to produce software that does not meet that need. In these times of belt-tightening, that kind of wasted investment is unacceptable.

Says Martin Owen, head of consultancy services at modelling specialist Popkin: "Businesses are realising that they have to model from a business perspective first and then drive that to IT. The challenge is that you first need to model your business processes and then model IT, and you need to bridge that gap. That's where UML-based business and software modelling tools come into their own."

It is a message that even today most companies are failing to heed. As Rational's Iain Gavin, marketing director, Northern Europe, says: "Only around 15% of software development projects use UML to model the project first." So what are the other 85% using? "Mostly, they're just not modelling at all," says Gavin. "Some might just try to map it out in Word or PowerPoint, others will scribble something on a whiteboard. A lot of people have a hangover from earlier modelling techniques like SSADM [structured systems analysis and design method], and don't understand the benefits of connecting the model to the code."

SSADM was a set of standards developed in the early-1980s for systems analysis and application design. It uses a combination of text and diagrams throughout the whole lifecycle of a system design, from the initial design idea to the actual physical design of the application. But it had its problems. The method was entirely separate from the physical code, so changes to the code were not always reflected in the model, and vice versa. Today, so-called 'round-trip engineering' built into UML modelling tools means that changes to the code are immediately

reflected in the model, and vice versa. This means that developers are less likely to get out of step with one another or with the business people driving the project.

Modelling companies also integrate far more closely with either the most popular integrated development environments (IDEs) from third parties (Rational now integrates tightly with IDEs from Microsoft, IBM, Borland and others), or with their own IDEs (TogetherSoft, for example, has integrated its own IDE with its modelling tools).

Knowledge retention

Another advantage of UML-based tools over SSADM and other methodologies is that because UML is a standard with far greater market penetration, models from one developer or one tool can be read and manipulated by other developers and within different modelling tools. "This helps an organisation to retain the knowledge of the people within the company," says CA's Nosseir.

"Modellers and developers are often changing at a company. With UML the model can always stay the central point – everyone knows where they are. That in turn increases productivity. We know from the analysts that 30% of application projects get scrapped, costing companies billions of dollars. A lot of those are cases were the software has not met the business objectives and the user requirements," says Nosseir.

But while UML is undoubtedly a major leap forward for software modelling, it is not by any means perfect. UML development began in 1995 with the combination of the Booch and OMT methods at Rational by Rumbaugh, Grady Booch and Ivar Jacobson. A co-operative effort by Rational and a score of companies led to a specification adopted by the Object Management Group (OMG) in 1997, being made available as a public standard. It served its purpose well: it was open, standard, and well supported by modelling tools vendors.

But standards must evolve as needs change, and after one or two updates the OMG started to accept proposals from interested parties as to what should go into UML 2.0. The four Requests for Proposals (RFPs) that specify the requirements for UML 2.0 were defined and issued in 2000, initial proposals for UML 2.0 were submitted in 2001, and final revised proposals are scheduled to be completed by the end of 2002. However, most expect 2.0 to be finalised late, around the first or second quarters of 2003.

UML suffers from one of its great advantages – the fact that it has many contributors and that the OMG tries to run the standard in a democratic fashion. But as Rumbaugh says: "This forces compromise, which often leads to some bloating of content... the UML is messier than it might have been had it conformed strictly to the views of a single company, but it is more comprehensive and universally adopted."

Keeping UML 2.0 relevant to today's application and ebusiness modelling requirements, without allowing it to become too bloated, is a significant challenge for the OMG. As Cris Kobryn, chief technologist at Telelogic – a modelling and application development company and a contributor to the UML 2.0 work – says: "Since most of these new language features are generally recognised as being essential for updating UML, they raise an interesting dilemma for the UML 2.0 language designers. Given that UML 1.x is frequently and fairly criticised for being gratuitously large and complex, how can we add major new features without exacerbating its unwieldiness?"

"With UML the model can always stay the central point – everyone knows where they are. That in turn increases productivity."

CA's Shirief Nosseir, AllFusion consultant.

So just what are the parties driving UML 2.0 trying to add, and what impact will it have on modelling? According to Popkin's Owen, the biggest new change is a move from being able to model objects to being able to model components. "With earlier versions of UML you could model objects, but if an application spanned multiple business processes you might find that that object has a large number of relationships with other objects," he says.

"The ability to model multiple objects as components will make a modeller's job far easier, since components will be able to span value chains across a business. These components are critical to modellers – UML 2.0 will help them to expose those components, and enable them to be reused by other modellers within the enterprise. That kind of reuse would be far more time-consuming if the original model consisted of tens or hundreds of objects instead of one component."

At the same time, Owens has his doubts about the validity of certain attempts at expanding UML in version 2.0. "I foresee a big problem with UML 2.0 being the Model Driven Architecture (MDA)," he says. "It means that instead of using UML to model business processes, objects and transactions, UML becomes a meta language to model absolutely anything." (For more on MDA see feature, *In the driving seat*, in this month's internal supplement.)

Why is this a problem? Because with earlier versions of UML, a box, for example, represented a class, and that was standard to all UML models. With MDA Owens believes that modellers will be able to define their own notations, also known as stereotypes, for a class, actor or other artefact. "If you can create your own stereotypes, for example make an actor into a car, then you no longer have a standard notation, and nobody understands each other's notations any more."

Standards evolution

CA's Nosseir agrees that stereotypes could become a headache: "Stereotypes are going to require careful management from the OMG," he says. "But this always happens in standards when new things are introduced. In this case I think that industry groups are going to have to collaborate on their open standards for stereotypes, and then liaise with the OMG. Either way, it is right to try these things. Standards cannot stand still."

CBR OPINION

UML is a useful and well-established language for modelling application development today, and in the current climate modelling is often the only way to achieve the required levels of quality, complexity and timeliness in application development projects that businesses expect. UML can even help to bridge the divide between the business and the IT department, and integration with the forthcoming business process modelling language (BPML) could strengthen that bridge even further. Meanwhile, UML 2.0 promises to bring new features to UML that should help to maintain its popularity and relevance. The OMG has its work cut out, however, to ensure that the specification does not become too bloated as a result of the democratic nature of the standard.