

Extract from Jim Lee's 'midrange matters' column in iSeries365 and iSeries News UK

COLUMN: Midrange Matters with Jim Lee: Agile development: practice, don't preach

There is no point blinding your customers with science, says Jim Lee.

Fundamental issues of contract in software development projects always relate to expectation over specification. The customer has an expectation while the developers and deliverers have a specification. If conflict, usually of interpretation, arises, then the undertaking is in trouble.

This fact is the same for projects that remain entirely in-house and don't involve third parties and lawyers. There are still customers (end-user departments) and suppliers (the IT developers). The expectation of users -- and user management -- is rarely met by achieving the specification. This specification determined the project and all progress, and even completion, is measured against this rather than against the business objectives achieved.

There is one other variable: change. The needs of a business today change during the lifetime of a development project. At the very least, the recognition of priorities changes as delivery approaches.

In the past, IT projects have been large, monolithic-style projects spanning months from design through approval, then development, testing, delivery and finally implementation. Procurement and project management, typically, followed the same principles as those processes required for building a car or a house. But house building and designing integrated software are not the same. Indeed, in some senses they are the antithesis of each other.

Success in software delivery has always been about agility. Agility is reflected in the ability to accept, embrace and deliver change instead of standing your ground in terms of contracted specification and change control.



What is different today is that programming languages and development studios allow us to build software in re-useable chunks. These chunks are functionally oriented. They never need to be rewritten. They are classes -- the only way, and therefore a consistent way, of delivering the function for which they were developed. There follows a dramatic change in today's environment over that of years ago. Testing is not left to the end of development; it is current and perpetual. If a class is changed, the test that reflects that the change has worked is embedded in the test bed of all future change and integration.

All of this background brings new commercial problems. Contracts are less easy to deliver. The lawyers like to deal with 'The supplier shall deliver to the specification by the date required. The supplier shall...'. In the new environment, we do not have a detailed specification in terms of programs and files. Instead, there is, at best, a list of functions to be delivered. Loose definitions are not easy for lawyers.

It might be argued that the technical evolution of an undertaking is exactly equivalent -- old versus new methodologies. In modern currency, 'functions' translate to objects and classes in much the same way as, in the past, programs were defined as performing specific operations on data in files and databases.

So what is different? Well, there is little doubt that nowadays, specifying, defining, drawing and generally completing every definition before programming would be treated as ludicrous. This is because the tools available allow us to enter continuously interactive evolution of the development without 'loss'. Writing detailed definitions takes longer than developing the code to perform tasks. No matter how far wide of the mark a process is in the eyes of a user, it can be corrected. We don't start again from scratch.

What we in IT need today is controlled user interaction. If we have a primary contact who really understands the business objectives and a pilot user team who are briefed to identify and even cause change, we can deliver to expectation more rapidly than ever.

It is important to recognise that this is a significant change to the old rules. Imagine being a user who had to sign-off a specification (half of which he could not understand). Imagine being responsible for acceptance-testing a new system which entirely meets specification but which you know is effectively unusable in terms of your work processes. These problems established huge conservatism in executive users' minds. As a consequence, the great divide that has resulted between user departments and IT is rarely bridged.

Now, the tag for all this change in delivery of modern IT solutions is 'agile programming'. But I don't recommend that you run along the corridors of your company announcing this revolution. As a director of a third party who undertakes development projects, I have found that threatening customers with a brave new approach only serves to delay things -- and I still can't resolve the lawyers' issues. Instead, focus on something relatively small. Evolve the solution with the users and deliver benefit. After the event, document the success and the procedures followed. Effectively, you are bridging the divide. You are creating trust and mutual respect. Starting with another IT slogan is not to be recommended.

One other thing: you need to be confident and you also need to cast off old practices and blame cultures. It may help bolster your confidence to know that the two largest development projects undertaken by my company this year both conformed to the agile programming model.

Initial interaction between our team and the customer team saw a web services model of development, so users could experience and comment on the interactive processes. We then undertook back-office conversion which saw us deliver 'engines' to service business rules. In one case, integration was then completed using an enterprise service bus (ESB) -- so it was service-orientated architecture (SOA) as well, but we didn't tell the users this either.

Both projects have outstanding user acceptance. Both were completed on time and within budget; so no work there for the lawyers. Neither project went through a



Midrange Matters

www.campbell-lee.co.uk

campaign involving the semantics of agile programming. Our customers needed trust while we needed commitment. What we sought was to obtain the trust and provide the commitment.

We did not take the work away and develop in some laboratory. We worked every day, on site, beside the users. Not offshore. Not even just local. Right beside the users. Now, how agile is that?

Jim Lee is Managing Director of Campbell Lee plc.

Contact him at Jim.Lee@campbell-lee.co.uk or call 0141 557 6400.

Ends.