# SOFTWARE DEVELOPMENT RISK 2.0

**By George S. Takach**

Many years ago, developing software from scratch was a very risky business. Customers (i.e., large organizations that use enterprise strength software products) were having some digital applications built, customized or installed for the first time, which is always a scary proposition. And many tech services suppliers (especially system integrators) were building, customizing or installing the particular software for the first time – also a highwire act of the first order.

But that was 25-30 years ago. Surely the customers are smarter now and the suppliers more experienced. And today are not the software development standards and protocals are more advanced, even standardized? If you think, however, that software development is no longer risky….well, you would be wrong.

A good example of what can go wrong can be seen in the recent Statement of Claim of a large customer that retained a leading supplier to build a state-of-the-art website for e-commerce. The customer wanted the new website to be able to give clients a superior – and digitally modernized – user experience (clients would use the site to order the core services of the customer's business). As well, the website had to be able to integrate with the back office enterprise systems of the customer. The result of the $32 million fixed price contract? After working on the project for a year, the customer terminated the relationship with the supplier for cause, and the statement of claim followed shortly thereafter. (I don't mention the names of the parties because, frankly, their identities actually don't matter that much; plus, until we see the Statement of Defence from the supplier, we will not know their side of the story (i.e., none of the allegations in the Claim have been proven in court). But in the meantime, there are some lessons that can be learned even from the Claim.

## *People Still Matter*

One of the allegations made by the customer is that there was significant turnover of key personnel on the supplier's team. This is a common complaint from customers in tech development deals that go sideways. The problem is this – even if the initial senior team members of the supplier are replaced by the supplier with equally stellar individuals, all of whom have solid general experience with the tools of the supplier and the technical parameters of the software being built, customized or installed for the customer, the problem is that the new team members simply will not have the institutional memory of the customer that the former staff had.

In other words, the initial senior implementation team of the supplier got to know intimately the requirements, and idiosynnacies, of the customer. If that initial team leaves half way through the project, that knowledge of the customer will be largely or completely lost. And that ends up putting the success of the project at risk – or at least causing a material delay. Therefore, in your contract with a supplier, it is highly advisable to address this risk head on. You should identify the key supplier personnel (by name), and provide that they cannot be pulled off your project unless you give your prior written consent. And to guard against a scenario where the supplier contemplates breaching this covenant, there should be a meaningful liquidated damages amount that will really make the supplier think twice about not living up to this critical obligation.

***Integration is Central – and Hard***

The Claim describes how the customer wanted to make sure the new software worked well on a traditional PC or laptop, but also on a smart phone, as well as on a tablet. The customer, however, alleges in the Claim that the functionality for the tablet wasn't delivered. This was one of the material deficiencies cited in the Claim – a seemingly simple feature that didn't come to fruition. You would think a feature such as this would be straightforward nowadays – we'll have to see what the supplier says about it in their reply.

On the other hand, one deficiency highlighted in the Claim that is usually acknowledged as difficult to do centres around the alleged failure of the supplier to integrate the new software with the other computer systems of the customer. This is a huge area of risk in these enterprise software development deals. That is, the core exercise is not simply to build the new functionality – in this case the client-facing website that would be the customer's primary e-commerce platform (not to downplay the difficulty of achieving that objective). But in addition, the supplier has to integrate the new software with myriad systems of the customer, involving finance, inventory control, database marketing, loyalty systems, and so on.

This type of enterprise level integration is so difficult because while the supplier is probably quite expert in the new software it is building (in the case of the Claim, the software for the new website), the supplier invariably is not as experienced with the back office systems of the customer. And therefore, however much the supplier knows the parameters of the new components of the project, it always has to get up the curve on the legacy environment into which the new software must be integrated. This is typically hard work – and in the Claim the customer alleged that it was not done well.

So, how to mitigate against this "integration risk"? One way is to de-couple from the main, fixed price contract the exercise of having the supplier learn everything they have to know about the integration challenge – that is, make the integration exploration effort a separate small contract, perhaps even priced on a time and materials basis. Then, once a very solid understanding of the integration effort is captured by the supplier through the initial small project, the supplier can bid the fixed price development work with a lot more confidence (which should let you breathe a lot easier).

***Tough Custom Requirements***

In many software development projects, the customer says at the outset, "I just want something basic, something out-of-the-box, something "vanilla" – meaning straightforward, simple. This is, indeed, the best practice approach, because if you can keep your requirements as "standard" as possible, you do go a long way to de-risking the project. But here's the thing – it seems that no matter how committed the customer is to "keeping it simple and vanilla", in every major project there is always something (and usually more than one thing) that turns out to be anything but straightforward.

In the Claim case, the difficult deviation from norm was the fact that the customer had several "sister operating divisions" in the same line of business. Thus, the customer wanted the initial website built in a manner that, after it was completed, it could be "cloned", such that with not much additional effort, it could be modified so each clone of the system could be used for another affiliate of the customer. This makes a lot of sense, and you can certainly understand why this design feature was asked for by the customer. Delivering on it, however, turned out to

be very difficult, apparently, to such an extent that the Claim argues this critical functionality was not delivered at all.

As noted above, until we see the Defence, and get a more thorough sense of what exactly happened, we'll never know (and if this particular case settles, we'll then we may truly never know). Nevertheless, the broader question of what to do about particularly tough custom requirements invariably arises in every material software development or system integration deal. And the first objective is for both sides – the customer and supplier – to admit that such tough hurdles will invariably exist, and therefore that they need to be ferreted out and surfaced right at the beginning of the mandate, in order that appropriate estimates can be made in the required budget and skillsets to make good on them. There is nothing to be gained by one or both parties to these transactions refusing to face reality, especially early on in the mandate.

The Claim concludes that the net effect of the deficiencies noted above (together with some others) caused the customer to have to cease work with the supplier, and then to pay $10+ million to a new supplier to finish the work (on top of the $32 million paid to the first supplier). Then there is the cost of being late to market with a meaningful web presence, which loss is very hard to quantify (i.e., how many potential clients went elsewhere, etc.?). Hence, the customer in the Claim asks for an award of all of its damages caused by the supplier's alleged failures. Moreover, I can affirm that the scenario described in the Claim is not that rare.

Therefore, to summarize my overall message – the lesson is fairly clear from the Claim. There is still plenty of risk (I call it Risk 2.0) in the world of software development, customization and implementation where large or complex IT systems are involved. And you will certainly want to manage your risk by using a fixed price contracting mechanism, even if your day-to-day development methodology utilizes "agile" practices. But at the same time you have to conduct sufficient due diligence on the technical and user requirement aspects of the project, so that the fixed price contracting approach does not come back to haunt you.