

Re: Design documents versus prototypes

Source:

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.non-technical/2006-05/msg01513.html>

- *From:* "Murdoc" <murdoc_0@xxxxxxxxxxxxx>
 - *Date:* 10 May 2006 01:43:17 -0700
-

Scout wrote:

I'm frequently given tasks by a huge corporate customer with a heterogenous IT system and more Business Analysts and consultants that you could shake a stick at (even if you were particularly adept at stick-shaking).

At a project review today which covered multiple projects, the customer acknowledged that I had delivered the required code in less time than the already very tight time-frame, and that the cost was under what they had budgeted for.

Then I got a bollocking for not producing design documents so that the BAs and consultants could see what they were going to get.

The only thing that I could think of saying in my defense was 'I did give you the design documents, but instead of being written in Word, they were written in Delphi, they worked, and showed you precisely what you were going to get'.

I had taken the requirements document and used the RAD power of Delphi to show exactly how the system would work in less time than it would have taken me to come up with a bunch of words that conveyed much less information.

Since the data access layer didn't exist when I created the prototype, I used things like:

```
procedure TMyForm.OnBarcode(sender : TObject; const Barcode : string)
begin
  {$IFDEF FAKE}
  // code that responds with some hard-coded values
  // and shows what should happen
  {$ELSE}
  // no code here, but this is where I'll write the real stuff
  {$ENDIF}
end;
```

Re: Design documents versus prototypes

...and put some real logic into the FAKE part so that all eventualities could be tested as long as they stuck to the hard-coded barcodes.

I know that I'll never win the battle of words versus work. The Project Manager even said "This is the same problem we've had with you for the last 14 years, but you always deliver on time (or before time) and on or under budget", so she's not going to win either.

What amazes me is that few corporates ever take advantage of the incredible power that Delphi provides in being able to produce working prototypes in a very short time.

I also got a bollocking for refusing to commit to a delivery date. As I said, this is an heterogenous system, and when I built the prototype, the data access layer didn't exist and was provided by another party. I was happy to commit to "3 days after the data is available, you can have your Delphi system", but project managers often don't see that as a commitment.

Some of the parts of the system were written in Java, others in COBOL, and the rest is Delphi. The Java part slipped under the radar 'coz none of the BAs actually understand what it does, and it seems that for COBOL stuff you really need to write documents specifying what is going on (or is planned) while the Delphi stuff just works.

Don't get me wrong: if there is a need to write database design documents to illustrate how the data relationships are going to work, then I'm your man. But if the proof in the pudding is whether or not the solution is going to fulfill the requirements, then I reckon that a prototype wins hands down every time.

And it seems that amongst all of the tools available to all of the people that work with these projects, only Delphi is able to produce working prototypes within the sort of timeframes that we are given.

The other thing that I love about Delphi is that there's not too much trouble in turning a prototype into a working system. Yeah, we can mess about with the prototype till the cows come home, but when the customer finally agrees that the prototype does what is required, there's very little work left to do.

What's your experience?

Scout

As a programmer (not Delphi, but that is irrelevant), I used to have a similar view, until I had to

Re: Design documents versus prototypes

maintain/modify someone else's UNDOCUMENTED system. As design document doesn't just specify WHAT the system will do, but generally HOW the system is designed and WHY it was designed this way (and potentially, why other designs were discarded).

These items are extremely relevant when using a non-standard design (e.g. not using a fully-normalized database design). There may be a perfectly legitimate reason for deviating from standard and conventional wisdom, but without documentation, that reason is lost.

Sure, well commented code does part of that, but not everyone that needs to know these things is a programmer.

On the flip side, a design document allows both you and the customer to agree on WHAT the system will do, before any code is written. Working out the requirements beforehand allows

(a) a clear understanding of what needs to be built, and hence ability to provide clearer estimate; and

(b) less chance of scope-creep. If you show the customer a prototype without having a clearly defined spec, then they will start saying things like "Hey, if you're doing that, can we also have it do ... " and "It would be really good if ..."

—

.