IncrementalMigration agile **7 July 2008** Reactions

Like any profession, software development has it's share of oft-forgotten activities that are usually ignored but have a habit of biting back at just the wrong moment. One of these is data migration.

Most new software projects involve data that's lived somewhere else and now needs to be moved into the new system once it's live. A system replacement might have to move all the old data, new functionality may lead to data being loaded from some other system.

It's common to not take this task very seriously. After all, it's just reading some data, munging it a bit, and loading into the new system. Furthermore the code only ever has to be run once, so there's no point making particularly fast or pretty. Once the migration is done the code can be safely chucked away.

And of course there's no need to worry about it till the end of the project since you only want to run the migration just before the new system goes live.

I have a high opinion of my readers, if only for their taste in software writing, so I'm sure I can see the wistful smiles. Data migration often looks easy from the safety of whiteboard abstractions, but is usually full of nasty details to trip you up.

- You may suspect that the existing data is somewhat messy, but everyone is usually taken aback at how dirty the data really is. As a result the whole exercise is often far more complicated than it ought to be.
- Because it's single use, throw-away code people don't tend to put much design effort into migration code since they assume it's below the DesignPayoffLine. That assumption is often wrong, especially with the previous bullet point.
- Doing an activity that balloons into something harder than you think is never fun, but when you leave it till close to the ship date you're offering trouble a big signing bonus.

There's a soundbite I like to use in an agile context: *if it hurts do it more often*. Its surface illogicality makes it memorable, and there's a real truth in there. Many difficult activities can be made much more straightforward by doing them more frequently. XPers are particularly well known for applying this principle to testing, integration, design, and planning - so it shouldn't surprise anyone to

Links
home
bliki
feed
Translations
Japanese
Spanish
Korean
Chinese
Thai
Categories
agile
design
dsl
leisure
refactoring
ruby
thoughtWorks
tools
uml
writing
Blog Roll
ThoughtBlogs
TW Alumni
Nicholas Carr
Steve Cook
Brian Foote
Simon Harris
Gregor Hohpe
Andy Hunt
Ralph Johnson

see it applied to data migration.

I first saw this done by my colleague Josh Mackenzie on a moderately sized project (dozen developers for one year) with two failed attempts in its recent past. He decided he would migrate data with every two-week iteration. Each iteration the team figured out what data they needed to add to support the new functionality that was being built and updated the data migration system to migrate that extra data from the live system.

As is often the case with these things it ended up being much less impossible than people feared and the resulting reduction of risk and stress made it a worthwhile choice. They appreciated the obvious benefits, which boiled down to a distinct lack of hasty panic close to going live.

The most interesting benefit, however, was the one they didn't expect. Incremental migration made a significant improvement in communication with the domain experts. Usually when you want to talk about use cases with domain experts, you make up some pretend scenario. By using incremental data migration the team got into the habit of using real examples, which were much easier for the domain experts to relate to. Furthermore when the development made builds available for the domain experts to look at, it included a copy of the live data. As a result the domain experts could investigate how the new system worked with tricky cases they had run into recently. Particularly juicy predicaments could easily be copied over into the test environment.

Even without the improved communication it's worth the effort to do incremental migration. If you do, be prepared to take advantage of the opportunity to use real data to talk to domain experts.

Patrick Logan
David Ing
Brian Marick
Jeremy Miller
Jimmy Nilsson
Samuel Pepys
Keith Ray
Johanna Rothman
Kathy Sierra
Dave Thomas