Why FileMaker requires development standards

I posted this comment on the main page of the Coding Standards and I'd love to hear what others have to say. This blog post is a better place to have the conversation. So, to start it off, here is what I had to say.

> *I'm all for the ideal. The biggest problem in the path of an ideal, that satisfies all the various environments, is FileMaker itself.*
>
> *Sadly, my BIGGEST gripe against FileMaker is the SERIOUS lack of organizational/documentation tools. I think FileMaker has been one of those applications which had a lot of early adoption, started heading in a direction and hasn't taken the time to refactor.*
>
> *Here are only some examples of what I'm talking about.*
>
> - *No filtering in manage fields or tables (thank goodness it's in scripts). Finding code fast should be a priority. Draco Ventions has a great tool called Developer Assistant for this type of thing.*
> - *No indicators on TO's or relationships of the attributes (e.g. allow creation, deletion and sorting)**
> - *No tightly-bound associative comments on tables, table occurrences, relationships, value lists, security and other areas. (it's great to have on fields, but why stop there)*
> - *Inflexible code editor (no syntax highlight, code folding, changing editor font, etc.)*
> - *The Relationship Graph is flat. No creation of logical groupings of TO's, no search feature, no grid or snapping support and various other missing features I would put in.*
>
> *Granted, recent improvements like the Inspector palette and improved Status bar provide more information, but there is still a ways to go in aiding the developer.*
>
> *So why say all of this to address why I promote breaking out the various technologies on a file by file basis? Because, if you're seriously developing a FileMaker solution then you're likely using a separation model. In this case, the data file is clean and won't be hindered by FileMaker specific conventions (such as the tilde and percent). Accessing with ODBC/SQL shouldn't be an issue.*
>
> *I agree, this does add a bit of overhead (due to potentially duplicated script functionality), but the upside is that your whole system becomes a bit more self-documenting.*
>
> *Some day, I'll have to sit down and document my development philosophy with FileMaker. I'd love to have a public chat about it. I'll make blog post about it and we can carry the conversion on over there.*
>
> ** FileMaker, as the "easy-to-use" database is great. The problem for advanced developers is that it's a bit of dialog hell.*

So here's the real issue. FileMaker's relational capabilities were added in because they could be. It was an evolution from the file to file heritage of the 5/6 days (if you were around back then). The capabilities realized were awesome. The concept of table aliases (which is what Table Occurrences are) is a great concept. However, I don't think organization of a growing solution was a forethought. They probably had to wait and see what developers did with the tool.

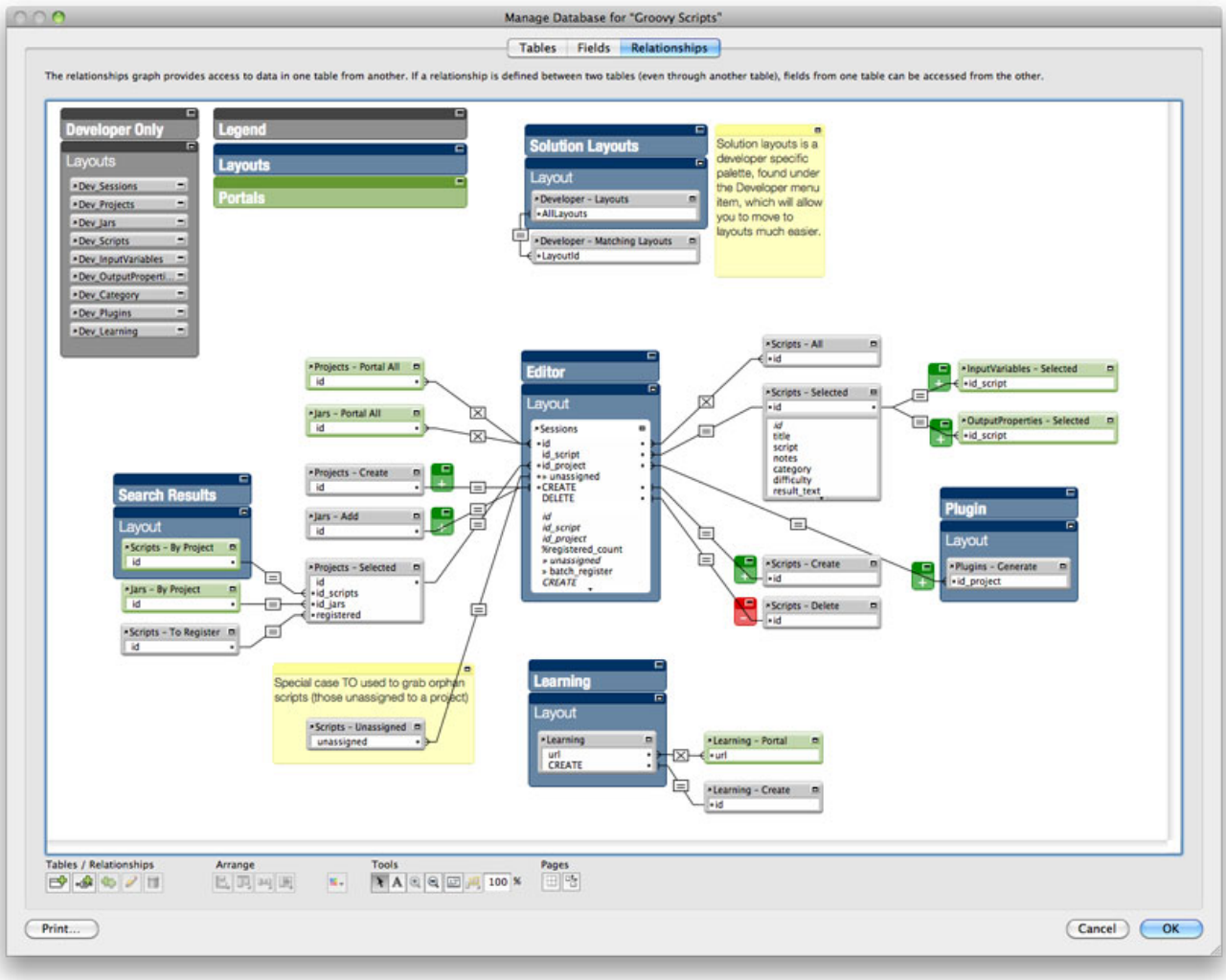
The problem is, as I see it, that developers have had to create development models to address the documentation/organizational shortcomings. Here's the biggest one of them all.

Did you know the Anchor/Buoy method of development is simply a way to organize your data structure so it's easier to understand? It really is just that. There's nothing wrong with that (I guess), but there's very little advantage to it other than its organizational aspect (at least as I understand it - school me if there's more to it). It allows a developer to have a left-aligned row of table occurrences which match to physical representations of layouts that users interact with. Yes, it does break down the solution in functional areas and "avoids" the claimed "dreaded spiderweb" - but... did you know that spiderwebs are actually really cool? 😉

Anchor/Buoy introduces extra table occurrences (yes, at very little data cost - if any), but adds to the "perceivable weight" of the solution. It also forgoes any advantages offered by the bidirectional nature of the relationships.

I know the anchor/buoy model is popular because I've seen a lot of developers use it. Personally, I don't use it. I rely on the limited degree of organization and documentation tools I've been given - until FileMaker adds in more.

Here's an image of a very small graph of one of my personal tools.



While this is not a major solution and it hasn't been converted to all the standards here, I don't see why this method of visual organization along with the **FunctionalArea » Basetablename** occurrence naming won't address the primary issue that Anchor/Buoy addresses - field selection. Basically, what we're talking about here is making head or tails of what is going on within a solution. Disclaimer: I typically work on smaller focused tools and not super large solutions - although I may at some point in the future.

As far as I can tell, I don't think Anchor/Buoy is that great of a solution because it inevitably introduces a lot of "waste" due to accidental duplication. This, in-turn, adds to the confusion of a solution.

And don't even get me started on the 001, 002, 003 prefix naming for table occurrences. We're not doing accounting here. Why should anyone have to decipher something when they should just be able to read it? Adding a numerical prefix because FileMaker's method of presenting Table Occurrence selections as a monolithic popup menu is not a good solution as far as I'm concerned.

So, I'm going to wrap up my rant for now. I'm interested to hear what you have to say.

Please, comment at will!