Flying over Yugoslavia

My mind tends to wander when I'm on lengthy airplane flights. Normally I can let some inoffensive movie wash over me on my personal video screen, but on a recent flight from London to Tehran this was not possible. The plane was somewhat outdated, for example, each seat had its own tiny ashtray, and like all planes of that vintage there was just a single screen at the front of the cabin. I was trying to recall my last long haul flight without personal entertainment when I noticed that there was something odd in the "travelling map" being projected on the front wall. It took me a moment to realise what it was. The country that we were flying over was clearly Yugoslavia. It was as if, as far as this particular airplane was concerned, the whole sorry mess in the Balkans had never occurred. Indeed, if it never intended to land in Belgrade what difference does it make? There is an infinite list of concerns that have very little impact on the way this airplane would operate: the correct name for Macedonia; the current relationship between Croatia and Montenegro; and whether Kosovo is part of Serbia; absolutely none of those matter in this context.

The out of date passenger map will, of course, have no impact on the flight planning. The appropriate air traffic control centres were being contacted, presumably. So the sole impact was to amuse some of the passengers (well, one of the passengers anyway). This got me musing on how system designers decide which information is relevant.

Inexperienced systems developers often attempt to combine as much data as possible. This is in the hope that "what is important" will emerge, organically. When constructing a simple solution this approach often works. When integrating input from many different disciplines my experience is that it almost never does. The approach usually fails for two different reasons. Primarily there is the issue of definitions, new participants will often use the same labels to refer to competing concepts. We are all aware, for example, of the issues that arise because there is no general agreement about what the word "well" actually means. I've covered that issue enough times, there is no need to revisit that topic.

The other danger is that as more and more elements are incorporated the data structures becomes Byzantine. This leads to systems that are convoluted and stacked with "special cases". Such systems tie you to the original vendors, no one else can ever hope to learn their complexities, and some vendors feel that is a good thing. But in the end these solutions will always fail, often because the incremental cost of extending (or even just maintaining) the system just continue to grow until the case for reform becomes overwhelming. A good Data Architect can significantly increase the lifetime value of a solution by addressing this early, or as someone[1] once said "perfection is attained not when there is nothing more to add, but when there is nothing more to take away".

[1] More precisely, since he was French, Antoine de Saint Exupéry actually said "Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher"