The re-rise of the question

And so another year ends. Another decade starts, and an accident of history and human nature - name one religion that doesn't have some form of midwinter feast - has given us all some time to stop and reflect. Except the people working the busiest time of year in the shops and hospitals, or doing shifts keeping everything running. But for all those people in the offices and factories, there's a break in the normal routine, and a time to catch our breath a little, even if it is between another round of games or drinks, or another run down the piste.

So since we're all in end-noughties reflective mood, I thought I'd join in. The did-I-meet-my-new-years-resolutions-post is later. I've been thinking about what marked the end of this decade for me, and me it's that people are starting to question again. Not "how do I find Waitrose", but the bigger questions, and a lot of what I'm seeing is "are we doing the right thing"?

One of the things that I work on are the points where strategy, management and technology meet. And over the years, I've been watching each of these strata - I'd say 'disciplines' but most mid-level tech jobs now are or should be a combination of all three - evolve. And with the short attention spans and ever-faster development cycles of the dot-com boom and its followers came techniques for business, management and engineering that each at their heart had the simple question "are we doing the right thing", and in the case of systems engineering, its correlate "are we doing this thing right": see under validation and verification respectively. One of my favourite grumpy engineers has convinced me to always start with "what is the problem that we're trying to solve here", but that's a point for a different day.

So most of us computer scientists started the decade with a toolbox of tried-and-tested strategy and management techniques that worked fine on very stable requirements and timespans that were never quite long enough for each project without a hockey-stick panic, but were at least long enough to be managed in recognisable and sizeable phases (or find stakeholders, gather requirements, create an architecture etc etc). We've ended it with agile methodologies, morning standups and scrum masters, where we pick features off a list, build them and check that we're doing the right thing and thing right by regularly reviewing the system with our customers and users. Except there's a mismatch... the agile methods don't give us the larger framework to build complex systems in, and the old requirements-based methods don't build enough user feedback as these systems are being built. So the big question for now is how to be both agile and situated, when the system is more than just a slightly-clever website.