There's an interesting debate raging around the software world: Is Web development the same as traditional software project development?

Some say "yes" (we'll talk more about the manifestations of that viewpoint in a minute), and some say "no" (ditto). There seems to be little middle ground among the adversaries. People even say, vehemently, that all the things we've learned about developing software over the years should be used on Web projects. Or they say, just as vehemently, that all that "old-fashioned" stuff is all wrong for Web projects.

Some important things are at stake here, of course. How should we educate Web developers? With what tools should we support them? What should management expectations, and planning, be for Web projects? Can we exchange Web developers and traditional developers across projects?

I've been interested in this debate for some time now. That interest stems mainly from the fact that I've never participated as a technologist on a serious Web project. Whereas I've been a part of a lot of significant traditional software projects, and I think I understand that field fairly well, I don't have that same basis for understanding Web work. And I strongly believe that, to truly understand a software field, you have to have worked on major projects—preferably as a practitioner—for at least the complete duration ("inception to deception," some practitioners like to call it!) of a project.

Because of all that, it was with high expectations that I attended a panel discussion on this very subject at the 25th Annual Information Systems Research in Scandinavia (IRIS) conference last August near Copenhagen, Denmark. Jan Pries-Heje of the University of Copenhagen took the position that Web work was different. Karlheinz Kautz of the Copenhagen Business School argued there was no difference between the two. And, just to add spice, Erling Havn of the Technical University of Denmark took the position that he didn't really care! (That's the last we'll hear of Havn in this column.)

Why did Pries-Heje say things were different? He had conducted a survey of Web developers, asking them how they felt about the issue. Respondents to Pries-Heje's survey indicated differences between Web and traditional software development included:

A Mugwump's-Eye View of Web Work
Choosing a point of entry into a contemporary software development debate.
Information systems have always been developed differently from scientific systems, and critical projects have been treated differently from more mundane ones.

- market environment;
- lack of developer experience;
- product rush to market; and
- negotiability of quality as a goal (maintainability, respondents said, often gets sacrificed).

Pries-Heje summed things up by saying “the principles are the same, but the processes are different.” That position pretty well summed up my own conclusions from listening to people like Don Reifer [3], Ed Miller [2], and M.J. Taylor [4], as well as attending Web presentations at practitioner conferences. I summarized my views in articles such as [1] and in a previous column, “Searching for the Holy Grail of Software Engineering” (May 2002). The bottom line of those views: methodologies/standards/best-practice guides may be inappropriate for Web development.

So it was with a biased attitude that I listened to the Kautz presentation on why Web work was basically the same as traditional software work. His approach was to do a review of the software engineering literature, trying to match the quirks and idiosyncrasies of Web work with what the literature has told us about more traditional approaches. For each quirk or idiosyncrasy, Kautz found something in the traditional software field to match it:

- Hectic projects? That commonly happened in the traditional field in the 1970s, and continues to occur today.
- Different methods/practices? Traditional software engineers don’t tend to use the things the literature tells them about right out of the box, either.
- Skewed life cycles? The waterfall life-cycle model has long been in disfavor in the traditional world. Practitioners were using the spiral life cycle before anyone called it that.
- Observed practice different from “best practice”? Well, duh! That’s also true traditionally.

Kautz’ bottom line was “we’ve seen all of this before.”

Now, the challenge arose for my biased view. What did I do with the views of Kautz, given that my initial position was that his perspective was the incorrect one? Here, I found things getting interesting. I couldn’t help but agree with him! Certainly, there’s a kind of “been there, done that” flavor to all those quirkinesses of Web work. For example, the instability of requirements—something cited by most of the authors on Web work, as noted in the references at the end of this column, was something we commonly experienced in the early days of traditional software development. How could users know what their requirements were when the software solution approach had never been available to them before? What Kautz had to say resonated well with my personal, old-timer, recollections.

I puzzled over that mugwumpishness for awhile (for those who’ve forgotten their history, a “mugwump” is a politically indecisive person who sits with his mug on one side of the fence, and his wump on the other!). How could I possibly agree with both sides of an important, watershed issue like this one? In the notes I was taking on the panel, I actually added some personal thoughts about this. In the end, I scribbled another note to myself, my on-the-spot resolution of my personal dilemma, and resumed…

1 I also relied on some personal communication in defining Kautz’ position, a draft copy of Persistent Problems in Information Systems Development: The Case of the World Wide Web, by Jacob Norbjerg and Karlheinz Kautz.

2 Webster’s Ninth New Collegiate Dictionary offers two definitions of mugwump: a bolter from the Republican party in 1884; and an independent in politics.
listening to the rest of the conference presentations.

What did I write? There have always been important differences between software projects. Diversity is a key part of what makes the software world go ’round (a point I made in greater detail in my May 2002 column). Information systems have always been developed differently from scientific systems, and critical projects have been treated differently from more mundane ones. Why should we be surprised that the same condition holds for Web and non-Web projects? There are important differences between the two, I concluded, but those degrees of difference have always been a vital and even healthy part of what the software world is all about.

So my bottom line for the debate between Pries-Heje and Kautz? You both won. Take a bow! 🎉

**REFERENCES**


*Robert L. Glass* (rlglass@acm.org) is the publisher/editor of the *The Software Practitioner* newsletter and editor emeritus of the *Journal of Systems and Software*.

© 2003 ACM 0002-0782/03/0800 $5.00