CS 428 Webster #6 Part III

WINTER 2023

BRUCE F. WEBSTER

Conceptual pitfalls

- Adopting a new technology or methodology for the wrong reason
- Thinking a new technology or methodology comes for free
- Thinking a new technology or methodology will solve all your problems
- Confusing buzzwords with concepts
- Confusing tools with principles
- Confusing training with skill
- Confusing prototypes with finished products
- Confusing approach with results
- Asking the wrong questions

Adopting a new technology or methodology for the wrong reason

- **Issue**: silver bullet syndrome leads you (or your managers) to hope the TOM will magically solve various problems.
- **Symptoms**: new expectations based on the TOM are not materializing.
- **Consequences**: project slip or failure. Finger pointing. Heads rolling.
- **Detection**: ask all involved to write down their expectations for the new TOM. Look for 'magic thinking' or unrealistic expectations.
- **Extraction**: debunk magic thinking and reset expectations.
- **Prevention**: do the last two steps *before* adopting the TOM.

Thinking a new technology or methodology comes for free

- **Issue**: too often, management and/or developers will think that they can switch to a new TOM without going through the usual learning curve of time and practice.
- **Symptoms**: reluctance to devote the time and effort necessary to actually come up to speed on the new TOM.
- **Consequences**: failure to achieve most or all of the expected TOM benefits.
- **Detection**: ask: “If we were going to compete against a group of TOM experts, how would we do?”
- **Extraction**: recognizing that you may already be in the middle of a project, reset expectations (and schedule) to accommodate for coming up to speed.
- **Prevention**: ask yourselves the question above in Detection, and focus on people, time, education, tools, and practice.

Thinking a new technology or methodology will solve all your problems

- **Issue**: the full range of activities in the software lifecycle is quite lengthy; how many will the TOM really have a positive impact on? Especially in the short term?
- **Symptoms**: when people think the TOM will solve problems that it won't.
- **Consequences**: project slips or even failure.
- **Detection**: identify where the TOM could actually help and how much time it will take to become that proficient in the TOM.
- **Extraction**: reset expectations among both management and enthusiastic developers.
- **Prevention**: do the Detection and Extraction activities before starting the project and/or committing to adopt the TOM

Confusing buzzwords with concepts

- **Issue**: every TOM tends to have its own jargon, but just because you use the jargon doesn’t mean you’re actually doing the key thing.
- **Symptoms**: constant repetition of key words w/out asking hard questions.
- **Consequences**: lack of benefits, negative impact on schedule or project.
- **Detection**: can be hard to tell who really knows what she’s talking about vs. someone who knows the buzzwords and basic ideas.
- **Extraction**: admit there is a problem and get help from an independent (and possibly outside) source.
- **Prevention**: educate yourself and others ahead of time; read criticisms and negative articles; set expectations appropriately.

Confusing training with skill

- **Issue**: for many organizations, adopting a new TOM is often as "simple" as sending one or more developers or managers to a week-long class. That's exposure, not skill.
- **Symptoms**: managers and/or engineers who think the TOM can be adopted quickly and provide benefits right away.
- **Consequences**: negative impact on project and schedule; abandonment of TOM.
- **Detection**: ask your engineers & managers to rate their own expertise on the TOM.
- **Extraction**: for a small and/or non-critical project, use the project as a learning experience; otherwise, you may need to set aside the TOM for now.
- **Prevention**: hire skills; use pilot projects; minimize reliance on the TOM at first.

Confusing prototypes with finished products

- **Issue**: we can prototype so quickly that we may overestimate our own progress.
- **Symptoms**: thinking we're "80% done" and being stuck there for months.
- **Consequences**: project delays; frustration from management; poor moral; loss of trust and credibility.
- **Detection**: ask those involved: "If you had to bet $1000 of your own money, when would you bet that we will ship? How about $10,000?"
- **Extraction**: reset expectations and do so very conservatively. "Take no small slips."
- **Prevention**: Don't show prototypes, except to elicit feedback.

Asking the wrong questions

- **Issue**: management will often focus too much on two questions:
  - Why isn’t someone coding yet?
  - When will we ship?

  Those are important, but they can distort all that needs to be learned and done, especially when adopting a new TOM.
- **Symptoms**: See Issues.
- **Consequences**: coding starts too soon; disposable prototypes become actual products; project never stabilizes.
- **Detection**: go up the management chain asking, “What are your questions and expectations with regards to this project?”
- **Extraction**: hard and painful, but reset expectations in line with reality.
- **Prevention**: also hard and painful, but education management ahead of time and set realistic expectations.