

CS 428

Webster #6

Part I

WINTER 2023

BRUCE F. WEBSTER

Webster #6: Pitfalls of Modern Software Engineering

- ▶ Derived from my 1995 book **Pitfalls of Object-Oriented Development**
- ▶ These initial chapters are universal and apply to adopting *any* new technology or methodology (“TOM”)

Managerial pitfalls

- ▶ Using the wrong developers
- ▶ Using the wrong metrics (or none at all)
- ▶ Lying to yourself and others
- ▶ Not identifying and managing risks
- ▶ Adopting a technology or methodology without well-defined objectives
- ▶ Misjudging relative costs
- ▶ Allowing new features to creep (or pour) in
- ▶ Allowing the specification to drift or change without agreement
- ▶ Attempting too much, too fast, too soon
- ▶ Abandoning good software engineering practices

Using the Wrong Developers

- ▶ **Issue:** gaps in TEPEs (talent, experience, professionalism, education, skills), particularly with relation to the TOMs in use
- ▶ **Symptoms:** constant core issues with architecture, design, code quality
- ▶ **Consequences:** inability to ship or poor quality of delivered product
- ▶ **Detection:** you need to have someone who is qualified and whose opinion you trust
- ▶ **Extraction:** really hard, but you need to find the right people and/or train up the ones you have
- ▶ **Prevention:** hire better and monitor more closely

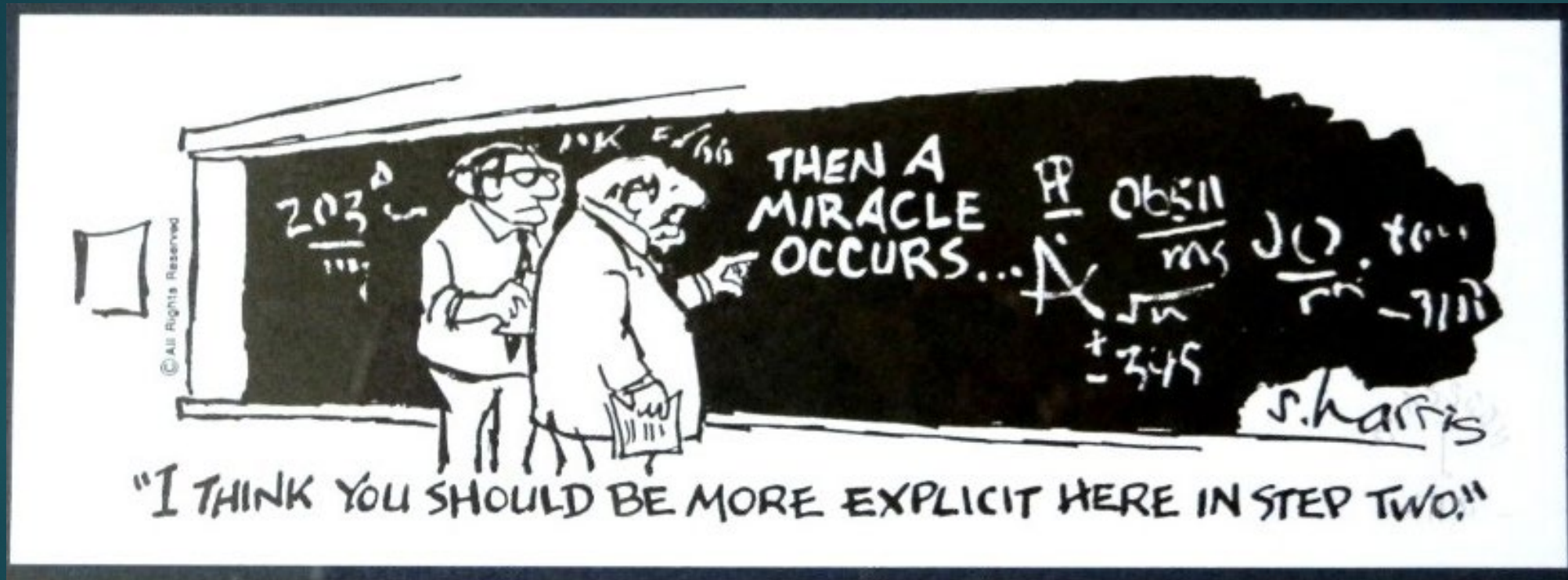
Using the wrong metrics

- ▶ **Issue:** most metrics are of dubious value; more so for a new TOM
- ▶ **Symptoms:** lack of correlation between metrics and actual progress; use of metrics as a management cudgel.
- ▶ **Consequences:** time and effort are spent gather useless or misleading metrics. Developer effort is focused on the wrong things.
- ▶ **Detection:** find out which metrics are being used and whether they have any predictive or informative value.
- ▶ **Extraction:** drop all metrics and investigate which, if any, would inform you.
- ▶ **Prevention:** define what metrics (if any) will be used at the start. Remember: they should be automated, objective, and informative.

Lying to yourself and others

- ▶ **Issue:** self-delusion and group delusion are far too common in software projects, due to optimism, positive thinking, and bad management. A new TOM often encourages such thinking.
- ▶ **Symptoms:** answer, irritation, disbelief when someone questions the delusion.
- ▶ **Consequences:** constant schedule slips, unexpected roadblocks, internal dissention.
- ▶ **Detection:** ask “What are we fooling ourselves about?”
- ▶ **Extraction:** need to re-plan and reschedule.
- ▶ **Prevention:** do a “pre-mortem” at the start of the project, asking all the ways in which it could be late or fail.

How project planning often goes:



Not Identifying and Managing Risks

- ▶ **Issue:** overlooking the risks involved in adopting a new TOM
- ▶ **Symptoms:** no one wants to talk about the risks. Lots of time spent putting out fires and explaining problems (and slips) to upper management.
- ▶ **Consequences:** slipped schedules, missed milestones, project failures, lost jobs.
- ▶ **Detection:** ask everyone on the project what risks they think the project faces. Build a list. Discuss it frankly.
- ▶ **Extraction:** prioritize the list of risks and address the most serious ones first.
- ▶ **Prevention:** actively and aggressively manage risks from the very start.

Adopting a new technology or methodology without well-defined objectives

- ▶ **Issue:** often a TOM is adopted just because it's new or interesting and not because it actually solves a known issue or roadblock.
- ▶ **Symptoms:** lack of progress, late deliverables, confusion about direction.
- ▶ **Consequences:** projects drag on forever and/or fail to achieve goals.
- ▶ **Detection:** as a group, describe exactly how this TOM is supposed to be helping and why it's not, i.e., how would things look if the TOM really were useful?
- ▶ **Extraction:** work backward from that goal and see if there is a clear and useful path.
- ▶ **Prevention:** use pilot projects first and determine feasibility and utility of TOM.

Misjudging Relative Costs

10

- ▶ **Issue:** failing to consider the extra time needed to adopt a new TOM and/or thinking the TOM will shorten the time required for different lifecycle efforts.
- ▶ **Symptoms:** all software lifecycle tasks are taking longer than planned/expected.
- ▶ **Consequences:** slipped schedules, missed deadlines, and rude surprises.
- ▶ **Detection:** apply Brooks' breakdown of tasks and see how that matches your schedule.
- ▶ **Extraction:** throw out your schedule. Replan from the ground up.
- ▶ **Prevention:** schedule conservatively from the start.

Allowing new features to creep in

11

- ▶ **Issue:** scope creep, especially if you assume the new TOM will let you do more things and/or do things faster.
- ▶ **Symptoms:** focus on adding new features (esp. in prototype form) rather than getting old ones working completely.
- ▶ **Consequences:** incomplete features, unexpected slips when milestones come up.
- ▶ **Detection:** review all planned features as a team and prioritize both the features themselves and the initial extent of each feature.
- ▶ **Extraction:** drop features until you can fit within the 'drop-dead' deadline.
- ▶ **Prevention:** do the Detection and Extraction steps before starting the project.

Allowing the specification to drift or change without agreement

- ▶ **Issue:** vague definition of features can lead to serious undetected scope creep.
- ▶ **Symptoms:** lack of detailed requirements. Constantly showing off “new features” before old ones are complete. Missed milestones.
- ▶ **Consequences:** schedule slip and lack of customer acceptance of product.
- ▶ **Detection:** do you have a features list? How detailed are they? What is being worked on that is *not* in the features list?
- ▶ **Extraction:** write a user’s manual for the 1.0 release and stick to it for feature completeness.
- ▶ **Prevention:** create, review, modify, and enforce the specification.

Attempting too much, too fast, too soon

13

- ▶ **Issue:** adopting a new TOM and then pushing full speed ahead with a mission-critical project.
- ▶ **Symptoms:** the project gets bogged down.
- ▶ **Consequences:** schedule slips and possibly project failure. Sometimes loss of confidence in or even abandonment of the TOM.
- ▶ **Detection:** do a hard-nosed match-up of actual progress vs planned schedule.
- ▶ **Extraction:** stop development, scale down the project, train developers, set realistic deadlines.
- ▶ **Prevention:** start out stupid, and work up from there.

Abandoning good software engineering practices

- ▶ **Issue:** adoption of a new TOM can sometimes lead management to think they can abandon best practices in software development.
- ▶ **Symptoms:** management thinks that development time will be shortened and some classic practices can be skipped because of the TOM.
- ▶ **Consequences:** lack of benefits of the TOM, leading to disillusion and abandonment.
- ▶ **Detection:** unreasonable demands and expectations from management.
- ▶ **Extraction:** very hard without getting upper management educated and enrolled.
- ▶ **Prevention:** education before the fact at all levels.