Food for Thought™

An e-newsletter published by
Software Quality Consulting, Inc.

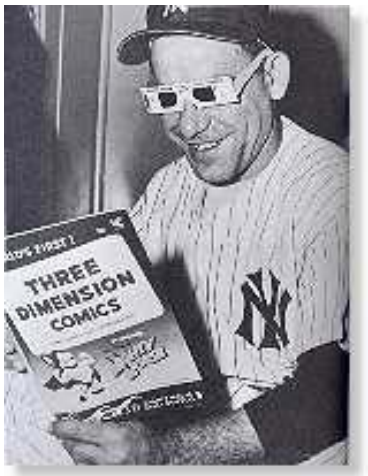
February 2007
Vol. 4 No. 2

Welcome to **Food for Thought™**, an e-newsletter from Software Quality Consulting. I've created free subscriptions for my valued business contacts. If you find this newsletter informative, I encourage you to continue reading. Feel free to pass this newsletter along to colleagues by using this link. If you've received this newsletter from a colleague and would like to subscribe, please use this Enter New Subscription link. If you don't wish to receive this newsletter, use the SafeUnSubscribe link at the bottom of this newsletter, and you won't be bothered again.

Your continued feedback on this newsletter is most welcome. Please send your comments to **steve@swqual.com**

In This Issue

In This Months' Topic I discuss what we can learn from sports legend Yogi Berra…



This Month's Topic

Déjà vu All Over Again

Yogi Berra is an 82 year-old American icon. He's a 15-time All Star who has won the American League MVP three times. He played in 14 World Series and holds numerous World Series records including most games by a catcher (63), hits (71), times on a winning team (10), first in at bats, first in doubles, second in RBI's, third in home runs and walks. In 1947, Yogi hit the first ever pinch hit home run in World Series history.

In addition to baseball, Yogi is known the world over for his uncanny ability to turn a catchy phrase. He's recently starred in commercials for an insurance company (you know, the one with the talking duck). So let's look at a few of Yogi's phrases and see what we can learn about software projects…

"It's déjà vu all over again."

Have you ever worked on a project that was released late, was far too buggy, and had key features dropped at the last minute? Has this happened to you more than once? Do you get the feeling that you've been down this road many times?

At some companies, after one of these late, buggy software projects is eventually released they hold some kind of a post-mortem. Usually, the post-mortem is held with little or no preparation and is led by someone with really poor meeting facilitation skills. Pretty soon, the discussion quickly turns into a gripe session where fingers are pointed and blame is laid. Amidst all the blame and finger-pointing there may be some good suggestions for things to improve for the next project. More often than not, however, these suggestions fall on deaf ears. And we wind up using the same process that led to failure again and again. And each time when the results are the same, management is surprised, concerned, even outraged. Well, there is a better way…

"You can observe a lot by watching."

Yes you can Yogi. I've observed many projects at many organizations. What I have seen is that one of the most difficult challenges software development companies face is their inability to accurately estimate and schedule software projects.

Inaccurate estimates lead to bad schedules. A bad schedule leads to poor quality, frustrated employees and dissatisfied customers. Maybe if we finally admit that **we need to learn how to estimate accurately**, we can break this cycle…

Steve McConnell [1] observed:

> "The fact that someone is an expert developer doesn't make them an expert estimator."
>
> "Creating accurate estimates is straightforward – once you understand how to create them."

People who are responsible for creating project schedules often have had little or no training in estimating and scheduling techniques. As a result, the estimates and schedules these people create are not accurate. How could they be?

Once again, there is a better way…

Steve McConnell [1] identified several terms that need to be defined in order to learn how to develop accurate estimates…

What are estimates?

The dictionary [2] defines an estimate as:

- A tentative evaluation or rough calculation
- A preliminary calculation of the cost of a project
- A judgment based on one's impressions and opinions

Steve McConnell says that "A good estimate […] provides a clear enough view of the project to allow the project leadership to make good decisions about how to control a project in order to hit its targets." [1]

Estimates are never negotiable. Do you understand why?

A word of caution - When your manager asks for **estimates**, they are often interested in something else… Do you know what?

We need to be wary of estimates that are:

- Derived using "gut feel" and "finger in the wind" methods
- Not supported by data or factual information
- "The answer the boss wants to hear."

Remember what Fred Brooks said:

"It is difficult to make a vigorous, plausible, and job-risking defense of an estimate that is derived by no quantitative method, supported by little data, and certified chiefly by the hunches of managers". [3]

What are targets?

- Targets are descriptions of desirable business objectives…
- Targets are determined by making business decisions…
- Targets are internal to the business

Unlike estimates, **targets are always negotiable**. Do you understand the difference between estimates and targets?

What are commitments?

We make commitments all the time. Every time you say something like, "Yeah, I'll have that for you by tomorrow" you've made a commitment.

Sales people often make commitments to customers that are not reasonable. Why do they do that? Ask a sales person at your company and see what they say…

Steve McConnell [1] defines commitments as:

- Promises to deliver defined functionality at a specific level of quality by a specific date…
- Promises made specifically to customers (internal or external)

Unlike estimates, **commitments are always negotiable**…

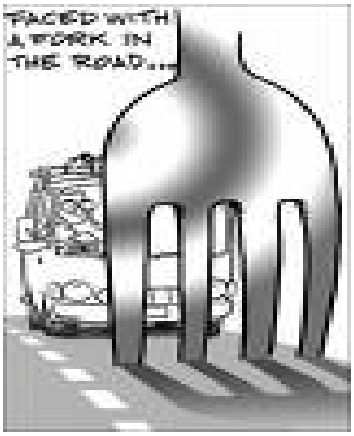
What are schedules?

A schedule is a **plan** that lists a project's **deliverables** and the **tasks** needed to accomplish those deliverables. **Resources** (people and equipment) are associated with tasks in a schedule. Have you even seen schedules where resources are the ubiquitous "TBD" (to be determined) or "TBH" (to be hired)? How can anyone think that a schedule based on fictitious resources is accurate? We do this all the time…

A schedule is created by looking at **dependencies** between tasks as well as the resources **available and qualified** to work on those tasks.

To create an **accurate schedule**, you need to identify:

- Deliverables
- Tasks required to accomplish the deliverables
- Available resources and their skill sets
- Required equipment for development and testing
- Dependencies between tasks
- Duration - how long it will take people to complete their assigned tasks



The last item – **duration – can only be determined by the people who will be actually be working on the task.** Is it a surprise that this is the case?

Remember: **accurate schedules can only come from accurate estimates.** And unlike estimates, **schedules are always negotiable…**

"When you come to a fork in the road, take it."

Forks in the road… It seems that managing a software project is a little like trying to find your way to a destination in a place that seems at once both familiar and strange. The journey is fraught with obstacles, potholes, and forks in the road.

The decision about which fork to take is often made by default, without the project team even realizing that a decision was made or what the ramifications are.

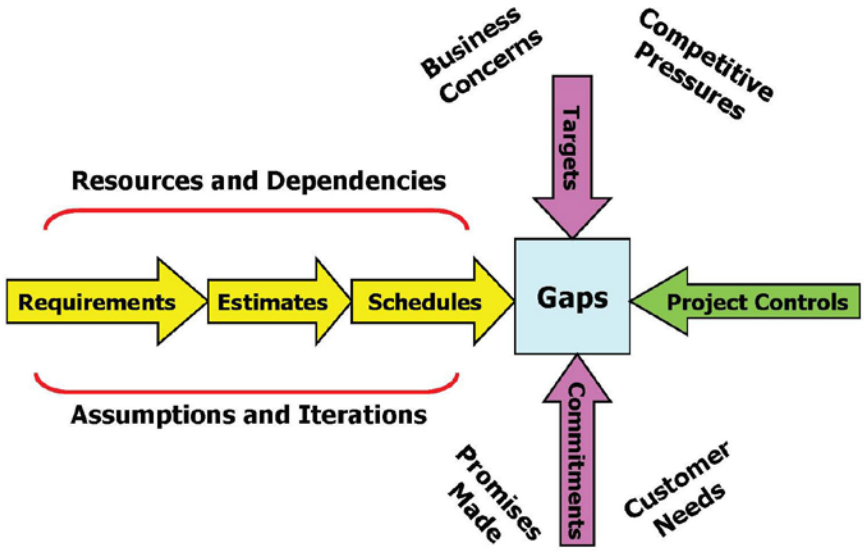
Example: A marketing person wants to add a new feature

late into the project. On the surface, the change seems simple. But lurking below the surface are many potholes, and a few major obstacles. A fork in the road…

Here's one decision a project team can make that will help the project be successful:

- Start the project with clearly written requirements
- Identify deliverables and tasks associated with those requirements
- Have people who will be doing the work estimate the tasks
- Build schedules based on factual information derived from these estimates

Here's another way of thinking about the process of estimating and scheduling…



Requirements are used to develop good estimates by people who will actually be doing the work. These estimates are then used together with resources and dependencies to build good schedules. This is an iterative process and takes into account project assumptions.

Based on this information, there will always be differences or gaps between the **schedule** and the **targets** that the business would like to hit, as well as potential commitments made to customers. This is where good project management skills can make a difference – by negotiating changes to targets and commitments, the project team can greatly increase the likelihood of success – which is defined as delivering a high quality product with promised features in the promised timeframe.

Remember, **estimates are not negotiable.** Everything else is…

Taking this fork in the road will definitely result in a more pleasant journey.

"If the world were perfect, it wouldn't be."

If only we could decide which features to include in a release – all of our problems would be solved, right? No, we'd certainly have other problems to deal with. The way we decide which features to include in a release is something we can certainly do better.

The age-old dilemma software companies are often faced with can be summed up by this conversation between Marketing and Development:

Marketing: "How can I know if I want that feature in the next release if I don't know what it costs?"

Development: "I can't tell you what it will cost until we have some requirements."

Fortunately, there is a relatively simple solution to this dilemma and it's called…

T-Shirt Sizing

Steve McConnell [1] has a very simple technique that can help resolve this impasse. Developers and testers estimate the work to develop and test each feature as Small, Medium, Large, or X-Large – like t-shirt sizes. **The effort is relative.** At the same time, Marketing gauges each feature's business value using the same scale. When all groups are done, we combine the information into a table like this…

Feature	Business Value	Development Cost	Testing Cost
Feature A	Large	Small	Medium
Feature B	Small	Large	Large
Feature C	Large	Large	Medium
Feature D	Medium	Medium	Small
…			
Feature Z	Small	Small	Medium

It now becomes easier for Development, QA, and Marketing to have an intelligent and factual conversation (wouldn't that be interesting?) about what features to work on. Which feature in the table above is the one to work on?

"It ain't over 'til it's over."

Testing is a process that is wrought with uncertainty. Testers have no way of knowing how many defects their tests will find. Clearly the defects found will be a direct result of the specific tests developed by testers.

As a result, it is often very difficult to accurately predict when testing will be done. One thing is for sure – as Yogi said, testing ain't over 'till it's over. What does that mean? Well it means that you need to have clearly defined and measurable **criteria** that indicates when testing is done. For example:

- All tests have been executed at least once
- All defects found as a result of testing have been dispositioned by the Triage Team and placed into one of several pre-determined states
- Regression testing based on those defects that were fixed has been performed

The specific criteria for testing being done should be identified early on in the project and agreed to by all stakeholders.

Summary

I'll end with these final points:

- Software complexity is increasing at a rate that's faster than our ability to

understand, develop, and test this software. If we don't invest in providing project teams with new skills, we will soon sink into the infamous tar pits described in the Mythical Man-month…

- Those who ignore history are doomed to repeat it. If your last project was not successful, something must change in order to avoid the same outcome on the next project. Remember…

If you always do what you've always done,
you'll always get what you've always gotten…

A remembrance…

Doug Ross, one of the pioneers of the software engineering profession, recently passed away. Doug developed several programming languages and is probably best known for his Structured Analysis and Design Technique (SADT) also known as IDEF0. SADT was one of the first techniques for modeling complex systems. Doug was the founder of SofTech, Inc. a software engineering technology company. During the 1970s and 80s, his company was actively involved in developing compilers and in providing independent verification and validation services. I had the privilege of working for SofTech for several years during this time and remember him as an inspirational leader who was incredibly bright.

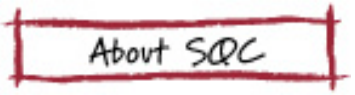
Every month in this space you'll find additional information related to this month's topic.

- **References:**

[1] McConnell, S., Software Estimation, Microsoft Press, 2006.

[2] American Heritage Dictionary, Second College Edition, 1985.

[3] Brooks, F. P., The Mythical Man Month and Other Essays on Software Engineering, Addison-Wesley, Anniversary Edition, 1995.



Software Quality Consulting provides consulting, training, and auditing services tailored to meet the specific needs of clients. We help clients fine-tune their software development processes and improve the quality of their software products. The overall goal is to help clients achieve Predictable Software Development™ – so that organizations can consistently deliver quality software with promised features in the promised timeframe.

To learn more about how we can help your organization, **visit our web site** or **send us an email**.

Graphic design by **Sage Studio**

Forward E-mail link
Safe UnSubscribe Link

Constant Contact logo…