Software Measurement

Why is Software Measurement Hard?

Prologue

- The presentation will explore and illustrate
  - The evidence of a problem establishing software measurement programs
  - The history of measurement - Including the emergence and description of the Pantometric paradigm
  - The Importance of different modeling Software Engineering Dynamics; Systemic, empirical & metaphorical. Some techniques for modeling Software Engineering Dynamics A new focus for software measurement

Acknowledgements

- The contents of this presentation represent the synthesis of much exposure to the subject of software measurement. Wherever possible I have acknowledged the sources of information, but I believe that it's inevitable that I have missed some, and I have certainly forgotten some. Therefore I make no claim to the originality of this work, and instead acknowledge the contribution that all those working in the field of software measurement, had to this effort.

D R Pitts
Phoenix
1998

Software Measurement

Evidence of a problem.

Measurement Justification

There are many motivations for software measurement, Almost without exception everyone “buys in” to at least one of these:

- **Hackneyed Case**
  - “You can’t control what you can’t measure.”

    Tom DeMarco, Controlling Software Projects, 1982
  - “What is not measurable make measurable”

    Galileo Galilei (1564 - 1642)
  - “Projects without clear goals will not achieve their goals clearly”

    Gilb, Principles of Software Engineering Management” 1987
- **Intuitive Case**
  - Measurement is prevalent not only within most scientific and engineering disciplines, but also within the day to day experience of everyone. It is how we run our lives.
- **The Benefits Case**
  - Tangible
    - Elimination of rework
  - Intangible
    - Greater Credibility for Software Development
  - Accountability
    - Justification of our own existence!

State of the Practice

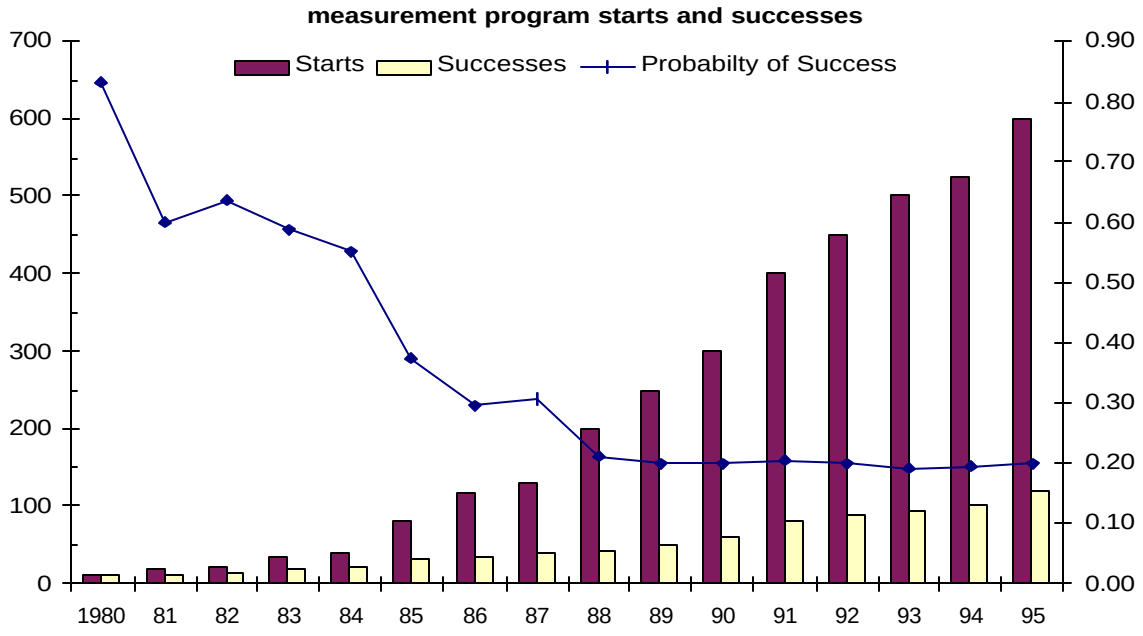


Figure 1

(Source: Dr. Howard A. Rubin, Rubin Systems Inc.)

- Observation
  - Since 1989 the ratio of starts to successes has remained remarkable consistent at approximately 5:1, so if you were to start a metrics program by following the industry practice there is about a 20% probability that the program will be a success.

State of the Practice

- ESI
  The Previous type of industry data is supported by other findings, for example in an European Software Institute (ESI) study only 40% of European companies surveyed admitted to employing metrics
    - » Software Practice questionnaire, EC, 1995 &1996 Data. Presented in: Improve, the Newsletter of the European Software Institute -Issue 6 -1997.

State of the Practice

- There is no shortage of literature on the subject
  - a Yahoo Search Result Found
    - 1175 web pages for “software metrics”
    - 576 web pages for “software measurement”
  - an Amazon.com Search Found
    - 40 Titles for “Software Metrics”
    - 50 Titles for “Software Measurement”

Conclusion

- Although measurement continues to demand increasing attention, measurement initiatives continue to exhibit a high failure rate,and the value of measurement goes unrealized.
- Many reasons are conjectured for this lack of success, mostly concentrating on organizational and management issues.
- It is my belief that there is a fundamental lack of understanding of the underlying measurement paradigm, and consequently that there exists an incorrect focus with regard to software measurement

Measurement is easy, Right?

I don’t understand why you
software guys don’t get it!

The Epiphany

- A colleague on mine vented his frustration one day with regard to software measurement by articulating something like:-

"Why isn't it like telling the time? We all know there are 24 hours in a day, and we just get on with it!"

- Which got me to thinking, Perhaps the problem is a lot more like measuring time than we realize.

TIME

“I know well enough what it is,
provided that nobody asks me…”

St Augustine (354-430)

Concepts of Time

- Relational Time
  - the original concept of time was that it was relational. Time is a function of something other than itself. Events, processes, superhuman powers etc. Aristotle generalized this by intrinsically relating time with the movement of the heavens.
- Potential Time/Perceptive time
  - Some philosophers experienced a conflict with their belief in the Bible in which there are reference to God stopping the sun, which contradicts the claim that celestial bodies always move. Consequently the concept of Potential time was introduced. This was a time that could be filled with events, even if nothing happened. Alternatively time and movement were linked such that if the celestial bodies slowed so would time, and so would mans perception, therefore the problem was insignificant.

Concepts of Time

- Absolute Time
  - In 1649 the concept of absolute time which "… flows with equal tenor whether anything is at rest or in motion, whether it moves faster or slower." was published, and In 1687 Newton published Principia in which he describes a duality of time, one absolute and one relational.
  - The absolute time was a necessary "invention" in order to to express the mathematical principles he had developed. Debate continues as to whether Newton's Absolute time was developed by him as an abstract concept useful in calculation but not in accordance with reality.
  - Newton's theory of Mechanics hindered the analysis of space & time for over two hundred years. Einstein's relativistic time is only of interest for movements approaching the speed of light. It is beyond the general experience not in accordance with what we consider today as common sense.
  - Absolute time continues to dominate general science and common thinking.

Units of Time

- The units of time based upon groupings of things easily divisible by 12 & 60 harks back 5,000 years to ancient Sumerian and Mesopotamian numerology, where occult meaning was perceived in numbers
- In general Hours were the smallest units of time that people were concerned with. Jesus himself reinforced the duodecimal system stating in John 9:9 “are there not twelve hours in the day?” (the implication being twelve hours for the night also). Europeans implemented a system of unequal hours that expanded and contracted with winter and summer to ensure 12 hours for daytime and nighttime.
- In addition hours were canonical (initially 3 then 5 then 7) They were not moored to “clock” time, but were breadths of time. Noon is derived from the canonical hour None, which was originally rung at approximately 3:00pm. Subsequently the ringing of None migrated back until it rested at midday. This was probably driven by the monks who on fast days could not eat before None had been rung.

Units of Time

- The French proposed in the late 1700's that a new time and calendar system be defined. The proposal was that the year be divided into 12 months of 3 weeks, each of which would be 10 days long. Each day was to be divided into 10 hours, each hour into 100 minutes, and each minute into 100 seconds. The "deciday"(2.4 hours) the "milliday" (86.4 seconds) and the "microday" (0.864 seconds) never overcame public opposition.

Mechanization of time

- Clock time
  - Mean Solar time was introduce in the 16th C in which the irregularities of the sun due to the earth's orbit were smoothed out. True solar time is as much as 16 min ahead or 14 min behind mean solar time. The difference is called the equation of time. The sun was still considered the decisive regulator of time, but with the introduction of pendulum clocks the discrepancy between mean and true solar time became apparent. Attempts to were made to solve the issue either by building clocks to show both times or tables to covert mean time to solar time.
  - The debate continued for over a century, but eventually clocks won. Mean Solar Time was made standard in Geneva 1780, London 1792, Berlin 1810, Paris 1816. Time was now embodied in Clocks.

Mechanization of Time

- Standardization
  - Even with the change to Mean Solar Time, people clung to nature in as much as they wanted mid-day to occur when the sun was at its highest in the sky. Consequently every community own had its own time. The need to evaluate, plan and organize activities, chronologically as well as geographically. demands two standardization's of time; one internal and one external.
  - With the development of faster communications a system of local times were substituted for standardized ones. In England Railways applied Greenwich-time in the late 1840s and in 1880 there was a “Definition of Time Act” prescribing Greenwich time for all activities in Britain.

Mechanization of Time

- Resetting of Time
  - in 1916 Germany introduced summer-time (Daylight Savings Time), followed within months by most European countries (who believed that this was a “secret weapon” to further the war effort. Public reaction was fierce, indicating that nature still played an important part in the realization of time
  - This however is probably the last vestige of the natural view of time. Indications are that more and more a abstract view of time is being established, as is indicated by the practice of clock stopping in political assemblies if a decision has to be made by a certain date (e.g. the UN conference on peace-initiative in Stockholm 1986).

Conclusion

- Our Current concept of time has been developed over a period of 700 years. During that period the concept has been consistently adapted to the practical problems at hand, to the point that hegemonic concept of time has changed from a natural/relational model to an abstract/absolute model. i.e. that which was considered the common sense view changed fundamentally.
- The current hegemonic view of time is not the currently accepted scientific view of time (Einstein's relativistic time).
    - For more infromation on the concepts of time
        - Crosby, A W. 1997, The Measure of Reality. Cambridge University Press, Cambridge, United Kingdom.
        - Haken H, Karlqvist A, Sveden U. 1993 The Machine as Metaphor and Tool. Springer-Verlag. Berlin

Why is Software Measurement Hard?

- Given that Software Engineering as a discipline has existed for approximately 50 years, why would we expect to have a Quantitative model of Software as well established and accepted as disciplines with histories centuries old.
- Measurement as a paradigm has become so well established that for most people it now lies below the level of consciousness. It has become a metaphor we live by, it is learnt, but it is not taught explicitly, and has been overlooked as an essential element of the software discipline.
- The key is to apply the same Paradigm to the discipline of software.

The Pantometric Paradigm

The Catalyst of Modern Western Society

Measurement Revolution

- The origin of Measurement as an activity can not readily be determined, but certainly precedes the emergence of mathematics (there is evidence that even Neanderthals made tallies)
- Measurement as a philosophy began to take hold in Western Europe circa 1200 AD. The venerable model which it superseded, is characterized by symbolism, mysticism and heterogeneity:
  - Numbers were chosen because they appealed to the intellect
  - Things can only be known if God deigned to let it be known. Predictability derives not from the reality itself but from God
  - Whatever the circumstance today would not preclude other circumstances yesterday or tomorrow.

Pantometry

- Pantometry is defined in the OED and Websters as: "Universal Measurement"

  http://machaut.uchicago.edu/cgi-bin/WEBSTER.page.sh?PAGE=1037

- Pantometry itself has fallen in disrepute due to misuse of measurement particularly in the classification and ranking of people according to their supposed genetic gifts and limits (Polygeny, Craniology IQ etc). Consequently other definitions include "belief in the possibility of and zeal for extending measurement to all phenomena”

The Pantometric Paradigm

- The Pantometric Paradigm, is to produce a purely visual and quantitative model of perceiving the material environment.
- To do this,
  - Reduce what you are thinking about to the minimum required by its definition;
  - Visualize it on paper (or in your head).
  - Divide it in fact or imagination into equal quanta. Then measure it (count the quanta).
  - Consequently you possess a quantitative representation of your subject which is precise in your definition.
  - Subsequently, you can manipulate it, and experiment with it.

The Key Concept

- No data can be interpreted without reference to an underlying model.
- In order to reason about a system, a model must be developed. This model documents the interactions within the system in a consistent manner. Subsequently appropriate strategies for improvement may be implemented based upon the expected behavior predicted from the model.
- There are three main formats for documenting the model.
  - Textual
  - Diagrammatic
  - Mathematical
- An affective model will utilize elements of all three.

Textual Models

- Text models are probably the least functional representation available. However this is the representation most commonly used for the documentation of the software engineering domain. Many heuristics concerning the dynamics of the software development practice exist most commonly in this medium. In addition all software engineers through experience develop their own “hunch base” which may never be expressed other than verbally
- The problem with textual representation is that it tends to be linear in construct. It is difficult to represent system feedback dynamics, and the resultant perceived model is dependent upon the interpretation of the prose.

Software Dynamics Heuristics

- The majority of the software dynamic is held in anecdotal form as rules of thumb, metaphors, and heuristics.
- Metaphors are extremely important because they shape our view of a problem in a fundamental and subconscious manner. This unknowingly entrenches our view and limits our ability to problem solve creatively.

Examples

- Metaphors of software development
    - The Wild West
    - Software Penmanship (writing)
    - Software Farming (Growing a System)
    - Software Construction
  - Heuristics
    - Brook's Law
      - Adding manpower to a late project makes it later
    - Weinberg's "Self Invalidating Model"
      - Belief that a change is easy makes it likely that it will be made incorrectly
  - Rules of Thumb
    - Good customer relations double productivity
    - Prototyping cuts the work to produce a system by 40%
      - » Larry Bernstein, Bell Communications Research

A Textual Software Model

Effort

The time required to develop a product, expressed as increments of person development time. e.g. person months, man hours.

In general Effort is a function of **Size**, & results in **COST**

Schedule

The total development time. Completion times for principle milestones

In General Schedule is a function of **Effort** and **Resources**

Size

The magnitude of product to be developed.

In general size is a function of **Features**

Defects

The incompleteness of the product.

In general Defects are a function of **Size**, & **Schedule**

Features

The Requirements of product to be developed.

Resources

The Number of developers applied to the product development

Diagrammatic Models

Diagrammatic Models

- Diagramming techniques are perhaps the most powerful tool available for establishing understanding. There are many techniques available, and two are documented here.

Diagramming Techniques

- Diagram of Effects
  - Gerald Weinberg in his book Quality Software Management Vol. 2, use the diagram of effects technique for exploring the systems dynamics of software.
  - This also has a simple notational convention, and is useful for translating models expressed in a text format.
  - This technique is more appropriate for exploring lower level project dynamics.
- System Diagram
  - Peter Senge in his book the Fifth Discipline, introduces the concept of system diagramming. He introduces a very simple diagramming notation for exploring complex dynamic systems.
  - In addition he identifies a number of System Archetypes, and management principles to deal with each.
  - This technique is more appropriate to the exploration of high level organizational dynamics.

Diagram of Effects

Diagram Of Effects

The diagram of effects uses a very simple notational convention, and consists primarily of nodes connected by lines.

- If the quality or attribute is conceptual. it is represented as a cloud.
- If the quality or attribute is actually being measured it is represented as an ellipse.
- A arrow between nodes indicated that quality or attribute A has an effect on quantity of quality B.
- The direction of the effect of A on B may be indicated by the presence of a filled dot on the line.
- A filled dot on the line indicates that as A moves in one direction B moves in the opposite direction Otherwise as A moves in one direction B moves in the same direction.

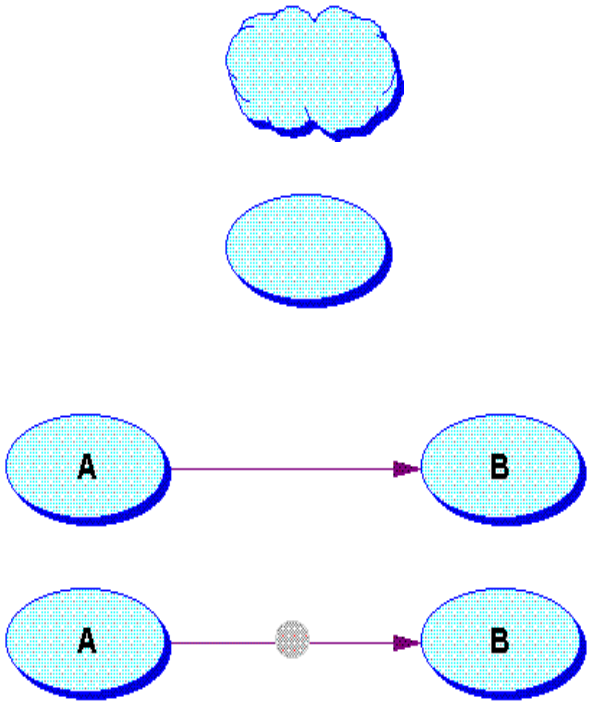


Diagram of Effects

- A square on the effects line indicates that human intervention is determining the direction of the effect
- A filled square indicates the intervention is making the affected measure move in the opposite direction to the case
- An unfilled square indicates that the intervention is making the affected measure move in the same direction as the cause.
- A half filled square indicates that the intervention can make the affected measure move either way dependent upon the intervention

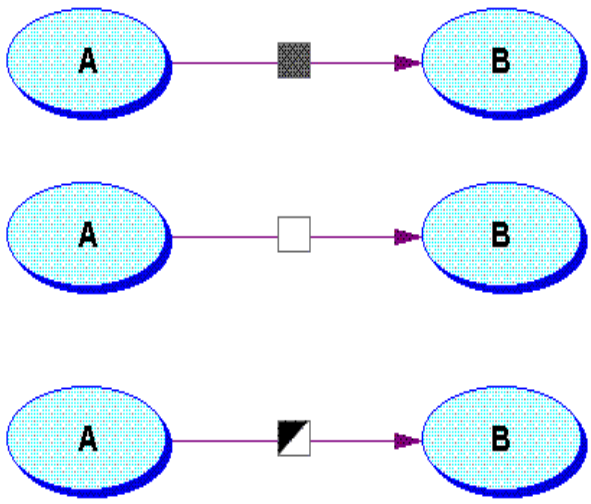


Diagram of Effects

- Consider the following representation of the textual software model described earlier

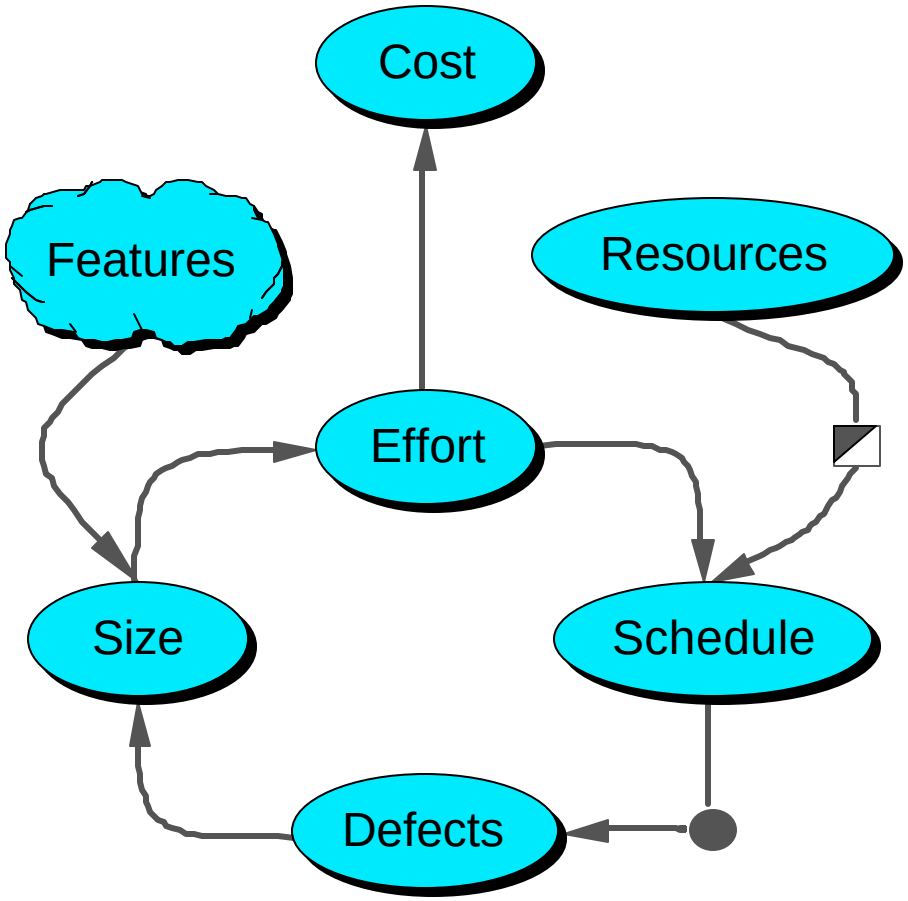
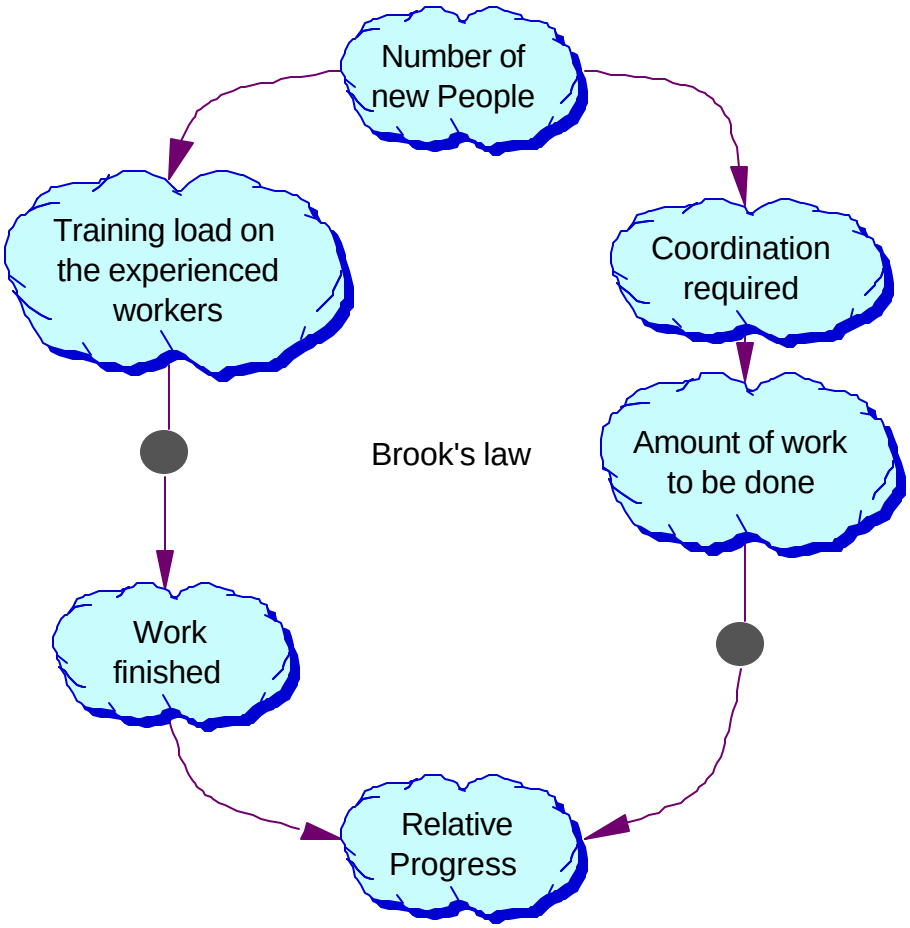


Diagram of Effects

- Example
  - Brook's law is usually explained in terms of the effect that the untrained personnel have on the trained personnel, and the increase in the amount of coordination required.
  - This can be understood more completely by the following diagram



Mathematical Models

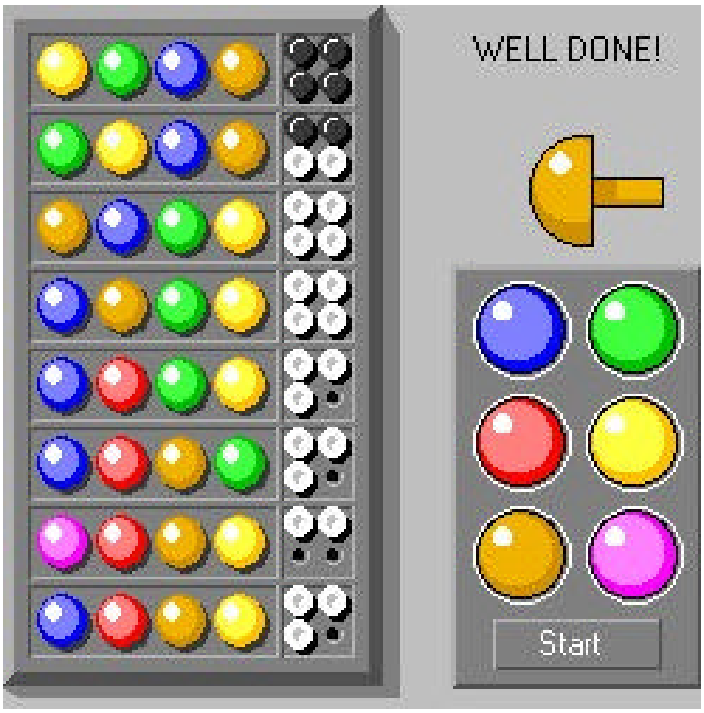
- Mathematical Models or Parametric models of software development are used exclusively for estimation and planning of software development project attributes.
- All use very similar models, and have been developed empirically from analysis of historical project data.
- All use a measure of size as the primary input, and accommodate various other project attributes.
- It is important to realize that all these mathematical models have an underlying system model of software development. Each is subtly different, but it is important that these are understood and are congruent with the actual management behavior.

Refine the model

- While ever the model provides sufficient explanation and control of the variability leave it alone.
- As options are exhausted or become inappropriate. Refinement is required to identify further drivers in the model.

Strategy to succeed?

- Remember this Game?



  - The more things that are changing simultaneously the harder it is to interpret or ascribe results to a particular cause

    http://www.mit.edu/people/mnoel/mastermind.htm

How do I know what to do?

- Top three remedies applied to “Runaway Software Projects”:
  - 85% Extending the schedule
  - 54% Better Project Management Procedures
  - 53% More People
    - Glass, R L. 1998 Software Runaways. Prentice-Hall Inc.Upper Saddle River NJ.
- Did they Succeed?
- Most software is being developed commercially consequently few valid experiments have been conducted to test scientifically the relative effectiveness of processes and tools. Case studies and endorsements are generally the only basis for selecting one conduct over another. By definition there are no "Best Practices" in the same way as there are medical cures.
- A way to overcome the non clinical nature of the software engineering knowledge base is to adopt an empirical or a Bayesian approach. This can be best defined as, commencing from an arbitrary opinion, the probability of an event is modified proportionally to the observed happenings.

What's a Baysian Approach

- If a new software development methodology/tool/technique has resulted in a software development cycle time decrease per project over 10 projects. Would It be justified to conclude ?-
  1. There is a mechanism at work here which causes software cycle time to improve with each project.
  2. The probability that the next project will also improve its cycle time is exceedingly high.
  3. But there is an unknown probability that it will not.
  4. If I fully understood the nature of my software development process, I maybe able to calculate the probability of my new technique not improving the next project.
  5. The fact that the nature of the software development process is not understood does not prohibit me from taking advantage of the fact that cycle time is better than it was 10 project ago.
- The key to success is being able to think rigorously about the problem.

Of course we think rigorously !

- Well the evidence for the population as a whole is not too hot.
  - 49% of Americans believe in ESP
  - 55% believe in the Devil (10% believe they have talked to or have been talked to by the Devil)
  - 46% believe in Psychic or spiritual healing (25% believe they have healed their body using mental power alone)
  - 25% Believe in Astrology
- There are 34 principles of knowledge reasoning, and evidence that you can use to enhance your problem solving skills and sharpen your judgement
    - Schick, T Jr., Vaughn A. 1995 How to Think About Weird Things. Mayfield Publishing Co., Mountain View, CA

How to think Rigorously

1 **Just because something is logically possible doesn't mean that it's real**

2 **Just because a claim hasn't been conclusively refuted doesn't mean that it's true**

3 **Just because a claim hasn't been conclusively proven doesn't mean that it's false.**

4 **Just because something seems physically impossible doesn't mean that it is**

5 **Just because something is physically possible doesn't mean that it's real**

6 **Just because something seems (feels, appears) real doesn't mean that it is**

7 **It's reasonable to accept personal experience as reliable evidence only if there's no reason to doubt its reliability**

8 **Just because you believe something to be true doesn't mean that it is.**

9 **Just because a group of people believe that something is true doesn't mean that it is**

10 **There is such a thing as objective truth.**

11 **We are justified in believing a proposition when we have no good reason to doubt it.**

12 **There is good reason to doubt a proposition if it conflicts with other propositions we have good reason to believe.**

13 **The more background information that a proposition conflicts with, the more reason there is to doubt it.**

14 **When there is good reason to doubt a proposition, we should proportion our belief to the evidence.**

15 **The is good reason to doubt a proposition if it conflicts with expert opinion.**

16 **Just because someone is an expert in one field doesn't mean that he or she is an expert in another.**

17 **If we have no reason to doubt what's disclosed to us through perception, introspection, memory, or reason, then we're justified in believing it.**

18 **Personal experience alone generally cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

19 **Case studies alone generally cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

Ibid.

How to think Rigorously Cont'd

20 **When claims of a treatment's effectiveness are based solely on case studies or personal experience, you generally cannot know that the treatment is effective.**

21 **Scientific evidence gained through controlled experiments- unlike personal experience and case studies - generally can establish the effectiveness of a treatment beyond a reasonable doubt.**

22 **Single medical studies generally cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

23 **When the results of a treatment conflict, you cannot know that the treatment in question is effective.**

24 **New study results that conflict with well-established findings cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

25 **Test-tube studies alone generally cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

26 **Animal studies alone generally cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

27 **Observational studies alone generally cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

28 **Clinical Trials with any of the limitations of; Lack of control group, Faulty comparison, & small numbers, generally cannot establish the effectiveness of a treatment beyond a reasonable doubt.**

29 **A hypothesis is scientific only if it is testable, that is, only if it predicts something other than that what was introduced to explain.**

30 **Other things being equal, the best hypothesis is the one that is most fruitful, that is , makes the most novel predictions**

31 **Other things being equal, the best hypothesis is the one that has the greatest scope, that is , that explains and predicts the most diverse phenomena.**

32 **Other things being equal, the best hypothesis is the simplest one, that is , the one that makes the fewest assumptions.**

33 **Other things being equal, the best hypothesis is the one that is the most conservative, that is , the one that fits best with established beliefs.**

34 **We should accept an extraordinary hypothesis only if no ordinary one will do.**

Ibid.

Software Model Evaluation

Each organization must develop and confirm a set of models appropriate to their own circumstance.

How much evidence do you need?

But it’s Got to be Right!

Any measure is better than no measure, and any measure is good if it consistently leads to the right control action.

- To a certain extent, mental power can compensate for observational weakness. To a certain extent, observational power can compensate for mental weakness
  - Weinberg. Gerald M. 1993, quality software management, vol. 2, first-order measurement, dorset house

The Pitts Postulate

- Expands on Weinberg in that it asserts that

  Bad data is better than no Data.

- In the physical world...
  - Ignorance
  - indifference
  - and misinformation
- … can all get you killed!. Wrong data can lead you into making wrong decisions If you survive, understanding what was wrong is key to improving your future chance of survival.
- A posteriori evaluations color our view of these states e.g is it better to die because of ignorance, or misjudgment ?
- Risk Management is the mitigation for bad data. Not ignorance or indifference

Necessary Measurement Attributes

- The necessary attributes of a metric in priority order are:-
  - Consistency
    Which is the reliability or uniformity of successive results,
  - Accuracy,
    Which is the next desirable attribute of a measure, and should not be confused with precision. Accuracy is how well the measure represents the specific attribute in question
  - Precision
    Which is simply the number of significant digits a measurement possesses

FALSE PRECISION IS THE ENEMY OF ACCURACY

Do not be seduced by precision into believing that a measure is accurate.

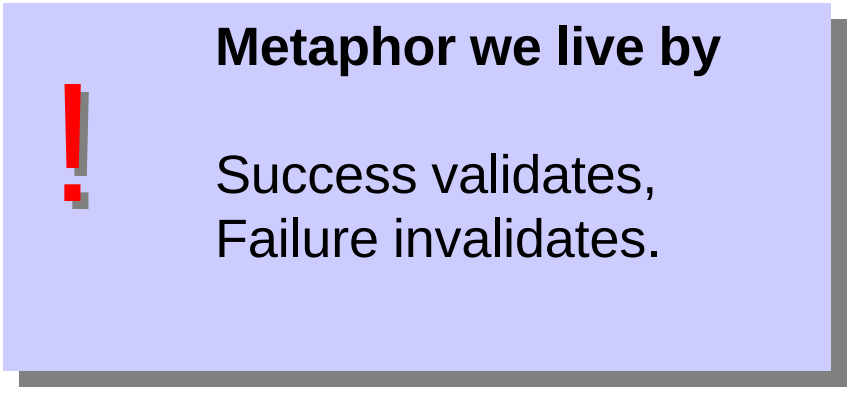
  - McConnell, Steve. 1996. Rapid development. Microsoft press

Summary

- Measurement is a way of representing the material environment quantitatively.
- Measurement is not and has never been easy.
  - We have forgotten this
- Measurement requires us to develop and change conceptual models of the subject.
  - Sometimes in a manner that is not intuitive.
- Measurement in Software is not as mature as measurement in all other disciplines, but we expect the same maturity, and are frustrated that we do not have it.
  - Even to the extent that we will not use it.
- The importance is not in the measurements themselves but in how these help you understand and manage the material environment.

Measurement Paradox

- Often the biggest obstacle to overcome when implementing a software measurement program, is our own history of poor or failed implementation.
  - Our inability to successfully implement software measurement programs, does not invalidate the principle of software measurement.
- Those that could benefit most from measurement programs are the least likely to accept the measurement paradigm because of previous failures.



David R. Pitts

David R. Pitts is a member of the Honeywell World Wide Software Initiative. This group's mission is to enable business growth by championing software competency improvement throughout the Honeywell Corporation. The focus of the group is on process and software capability improvement, and Mr. Pitts specialization is in the area of Quantitative Management. Mr. Pitts has worked at all levels of software development from programmer to software development manager both in the USA and Europe. Mr. Pitts received his education in the United Kingdom obtaining a bachelor's degree in Applied Statistics from Sheffield City Polytechnic (Hallam University), and a master's degree in Computer Science from Bradford University.