

Testing and Measurement: Transparency, Guidance and Tension 2012

Without the knowledge of cause and
effect, there is no way to improve.
Cory Doctorow

Tom Cagley

t.cagley@davidconsultinggroup.com

(440) 668-5717 – Cell

Software Process and Measurement Podcast (www.spamcast.net)

@tcagley - Twitter



Dashboard



Why



Tale



Methods



Exercise



Agile
Metrics



Food For
Thought



Questions



david consulting group



Why Measure

A GOAL OR A TOOL

Why Measure

- Software test measurement provides visibility into product and process quality.
 - Test metrics are facts to help a team, coach or project manager understand their current position.
 - Provide an objective measure of the effectiveness and efficiency of testing.
 - Identifies risk areas

Measurement Provides Awareness

- Knowing something is only the beginning of an equation that culminates in action.
- Awareness helps provide a spotlight of attention that filters unwanted information.
- If you are not able or interested in taking action, what value is there in knowing?



A Dialog About Measurement

- Good measurement requires an internal conversation about testing performance and goals.
- Measurement helps make better decisions.
- If you don't use performance measurement data, then do not bother measuring at all. Measurement shelfware wastes money.



The Basics

- All numbers begin life as good and useful tools.
- Act as a steward of the numbers and a high priest of information.
- Information rich world but very little structure and few filters.
- Metrics are a tool to fight Continuous Partial Attention (mostly).
- Defining what is important to the organization and what to measure is critically important and is not a truly democratic event.

Effective Measurement Is A Balance

- Effort
- Cost
- Interference
- Conflict



- Insights
- Actions
- Change
- Transformation

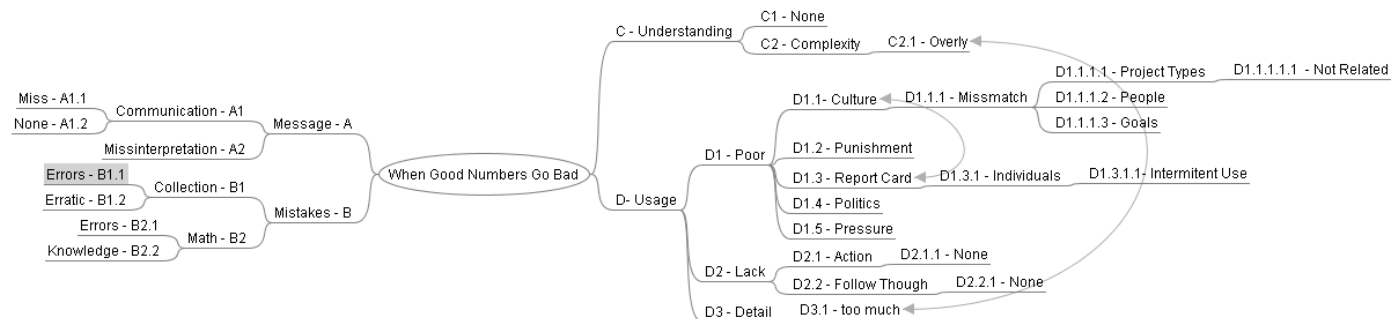


Good Numbers Go Bad

A METRICS CAUTIONARY TALE

A Cautionary Tale

- Message Messes
- Mistakes, Errors and the Like
- Lack of Understanding
- Lack of Use or Poor Usage



Message Messes: Communication

- Communication
 - Field of Dreams:
Un-validated vision
 - Monologues:
Unidirectional
communication
 - Beliefs: Powerful filter

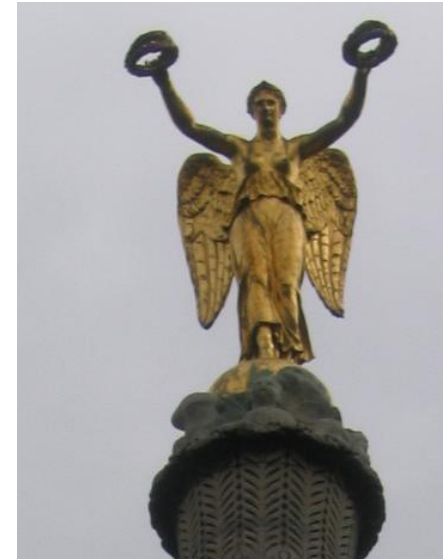
“A metric program is ineffective unless it is linked directly to a set of goals, mission or vision.”

Michael Sanders,
Past CIO of Transamerica Life

The Solution is . . .

Solutions:

- Validate how goals have been translated into metrics.
- Actively address misinformation and interpretations by providing neutral interpretations.
- Involve measurement users in analysis, interpretation (take a page out of Agile).



Message Mess: Misinterpretation

- Misinterpretation
 - Lack of education and knowledge: Missing the know how or frame of reference to analyze or interpret metrics data
 - Active dissemination: Making up a story . . .

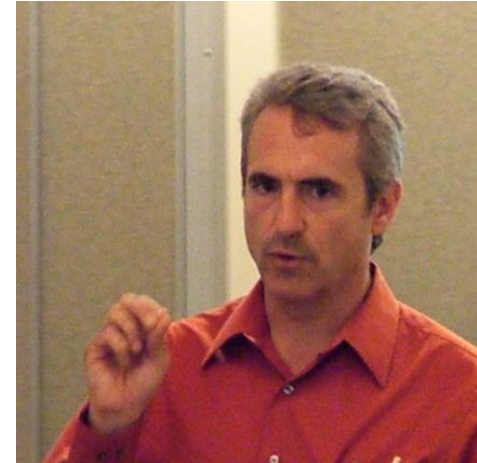
“It is of paramount importance for an organization to ensure that the proper decisions are made based upon the best (most accurate) data available.”

David Herron,
David Consulting Group

Metrics Analysis Should not be About Spin . . .

Solutions:

- Communicate and educate early and often.
- Keep interpretations neutral.
- Deal with misinterpretations as soon as they are identified.



Mistakes: Collection

- Collection
 - Errors: Collecting the wrong information or not collecting it at all (including all of their variants ‘garbage in . . .’).
 - Erratic: Collecting data when the urge (or boss) hits you.

“In order to capture metrics the procedures, guidelines, templates, and databases need to be in sync with the standard practices.”

Donna Hook, Medco

Make Sure You Collect the Right Stuff . . .

Solutions:

- Do not sweep problems under the rug.
- Make sure data specification is at a level that will allow you to actually collect it correctly.
- Collect data as specified in the measurement plan.



Mistakes: Math

- Math
 - Errors: Mistakes happen in logical definition of the metrics, the data collected and the equations.
 - Knowledge:
 - “I never took statistics in college but the graph looks pretty” syndrome
 - “I can prove anything by number syndrome”
 - “Equation exhaustion”

“We accidentally used \$88 instead of \$66. Now our stakeholders ask for a second source.”

Rob Hoerr,
Fidelity Information Services

One Plus One Equals . . .

Solutions:

- Have a professional statistician (or trained amateur) review your graphs, equations, assumptions and logical use of math.



Understanding: None

- None
 - Assuming: Don't make the assumption that users and providers understand what is being measured and know how to use the measures, or the data are contributing will be used for.

“What many people fail to realize is that metrics need to be tracked over time and ANALYZED.”

Iris Trout, Bloomberg

Educate Your Users . . .

Solutions:

- Communicate and EDUCATE early and often. Remembers awareness does not equate to knowledge.
- Use case studies to train your users and contributors.



Understanding: Complexity

- Complexity
 - Overly Simple: Failure to ensure explanative power of the measures and metrics
 - Overly Complex: “Baffle them with bulls...syndrome”

“Keep it simple enough. Ensure that the measurement is meaningful to both process actors and managers.”

S. J. Sanders, BOT International

Complexity Leads to Uncertainty . . .

Solutions:

- Leverage a statistician to review your graphs and equations. Are they explanative? Are they predictive?
- Simplify, simplify then do it again, but do not violate step one.
- Involve metrics users in the analysis of the metrics and measures.



Usage: Poor

- Poor Relevance
 - Culture Mismatches: Measures and metrics linked to unrelated items combined with the logical backing of studious people result in interesting ramifications. Type of mismatches include:
 - Types of work
 - People
 - Goals

“Good numbers go bad when, middle management dictates what the metrics program will report in order to improve or make a less than stellar project look better than it really is.”

RaeAnn Hamilton, TDS Telecom

Make the Measures Relevant . . .

Solutions:

- Review the measures you are accumulating and reporting.
Ask the following questions:
 - Do the measures work for all the types of work they are measuring?
 - Do the measures address all of the roles that participate in the work?
 - Are the measures and metrics aligned?



Usage: Poor

- Poor
 - Punishment: Leads to risk aversion or worse.
 - Report Cards: Comprehensive or jaded view?
 - Politics: “We can’t challenge that, it is too political”.
 - Pressure: Incent behavior outside of the norm?

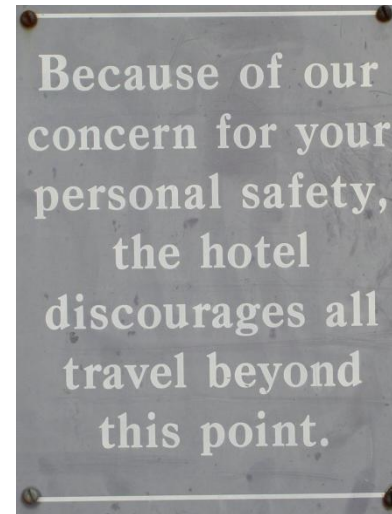
“One characteristic of a bad metrics program is to ‘Beat People Up’ for reporting true performance.”

Miranda Mason, Accenture

Your Report Card is in the Mail . . .

Solutions:

- Recognize the level of granularity each measure can be used to (person, team or organization) explain performance.
- Create balanced scorecards linked to business goals and the behavior you want people to exhibit.



Usage: Lack

- Lack of:
 - Action: Data is collected, then nothing. Someone forgot that the “Some action is required here” block on the flow chart.
 - Follow Through: Inaction is a message about the perceived importance of the behavior being measured.

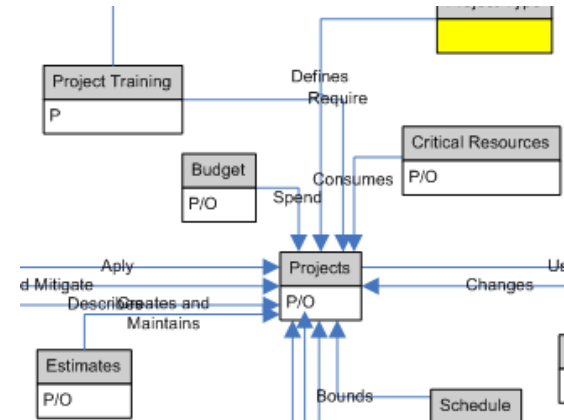
“The key is that there is no point to taking measurements and deriving metrics if they aren’t part of some (planned) decision making process.”

Jack Hoffman, Wolthers Kluwer

Now That You Have Data. . .

Solutions:

- USE THE DATA YOU ARE COLLECTING.
- Report the measures (publicly) and take actions based on the data.



Usage: Detail

- Detail
 - Too Much: ‘If a little information is good, then more is better’
 - Information Overload: Contributing to organizational ADD (Continuous Partial Attention, CPA).

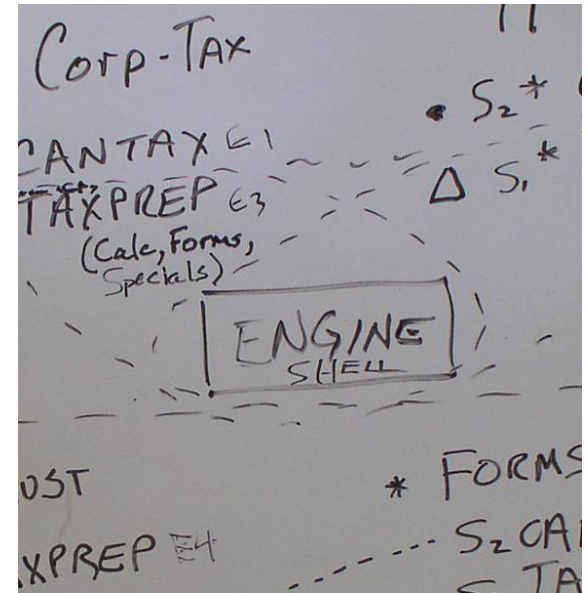
“I believe regular customer review and involvement will significantly increase the chance that we will provide what our customer(s) want.”

Mark Smith, Diebold

What Level of Detail . . .

Solutions:

- Link information needs to your organizations business goals as an anchor to collect and report *only what* is required.
- Discipline is required to make business goals an anchor.



End Of The Cautionary Tale

- Good numbers do not go bad all by themselves. Problems can stem from many sources including:
 - Lack of planning,
 - Lack of knowledge (on many fronts),
 - Politics and/or
 - Mere mistakes.

In the short term, it might be easier to let your numbers go bad, even to run wild.

Do not wake up late one night to see your numbers featured in a good-numbers-go-bad infomercial.

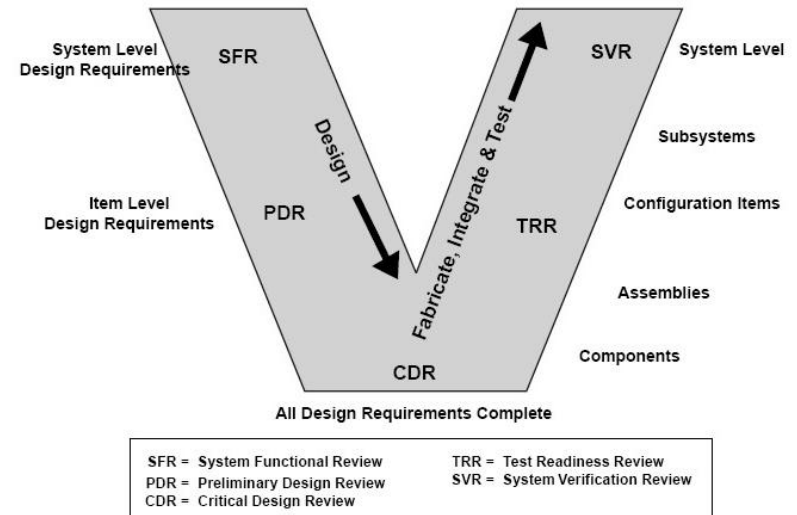


Methods

WATERFALL AND V-MODEL AND ALTERNATIVES

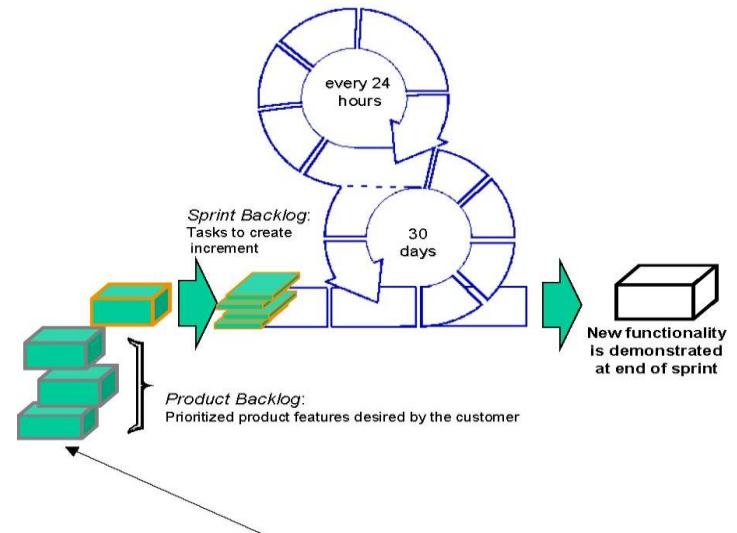
The Venerable V-Model

- Developed for managing!
- Salient Features
 - A simplification of the complexity of development
 - Waterfall ish
 - Provides a uniform procedure for development
 - Shows the decomposition of requirements paired with verification and validation steps



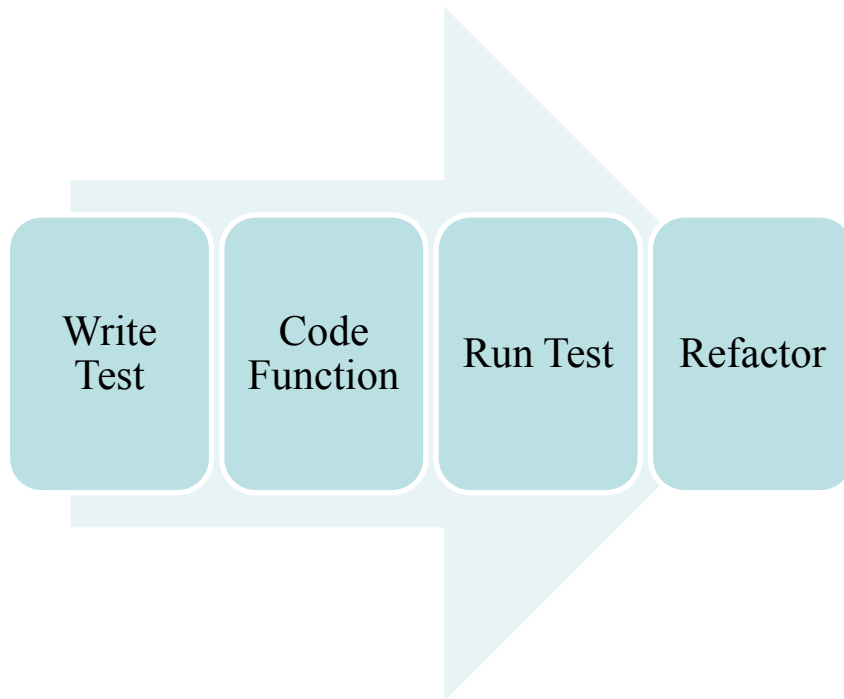
Scrum At A Glance

- Developed for managing!
- Salient Features
 - Product and Sprint backlogs (card wall)
 - Iterative planning
 - Time box
 - Standup Meetings
 - Definition of done
 - Sprint Demos
 - Retrospectives
 - Self directed and organized teams



- Roles
 - Scrum Master
 - Product Owner
 - Scrum Team

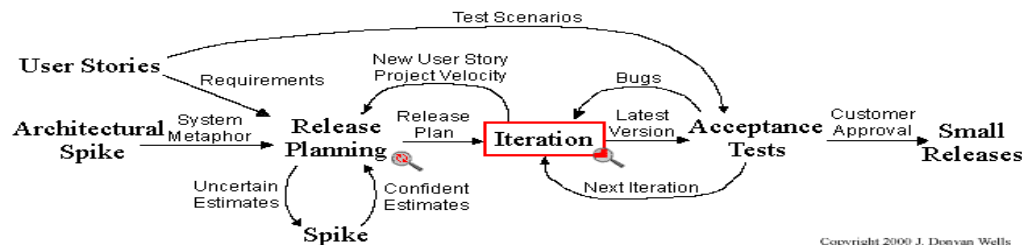
Test Driven Development At A Glance



- Salient Features:
 - Focus on meeting need expressed in test through code
 - Reduces technical debt via refactoring
 - Testing is an enforced regression test
 - Iterative
- Role(s):
 - Developer

xP At A Glance

- xP is a full methodology: Project management and technical components
- Salient Features
 - Release plan
 - Iteration plan
 - Acceptance test
 - Stand Up meeting
 - Pair negotiation
- Salient Features, part 2
 - Unit test
 - Pair programming
 - CODE
- Roles
 - Customer
 - Developer
 - Tracker
 - Coach



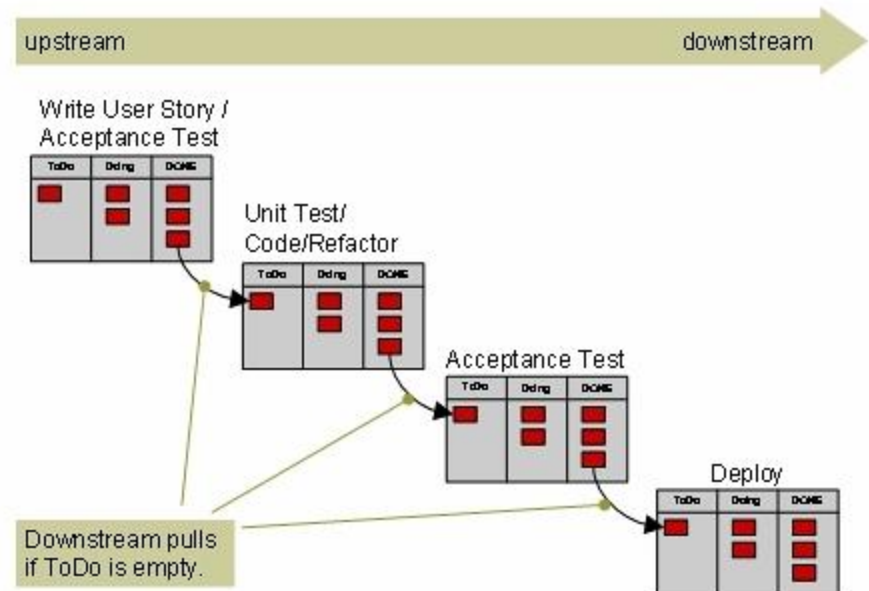
Copyright 2000 J. Donovan Wells

Kanban At A Glance

- Pull methodology
- Salient Features:
 - Work in process limits
 - Continuous Flow
 - Iterative releases
 - Self directing
 - Identifies when work sits
 - Does not require that you change your methodology

Kanban can be used anywhere so let's talk about it more =>

- Example



david consulting group

What Is Kanban

- Kanban means “visual card”
- Originally part of the Toyota Production system, Kanban cards limit the amount of inventory tied up in “work in progress” on a manufacturing floor
- Excess inventory is waste, time spent producing it is time that could be expended elsewhere
- Kanban represent how WIP is allowed in a system

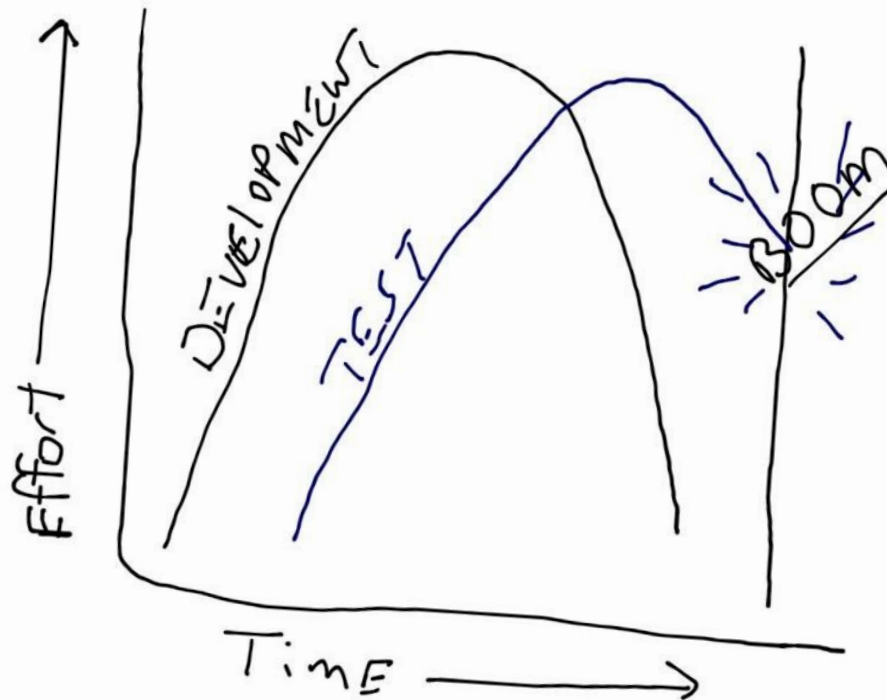
But . . .

But we are doing
incremental development
and testing. Shouldn't
everything be fine?

Common Time Box Development Issues

- Short time-boxes force development items to be smaller
- Smaller development items are often too small to be valuable and difficult to identify
- Quality of requirements suffers as analysts rush to prepare for upcoming cycles
- Quality of current development suffers when busy analysts are unable to inspect software or answer questions during development
- Quality often suffers as testers race to complete work late in the development time-box

Inside an iteration, effort across roles is uneven



Specialization
can
exacerbate
this issue

Development work often continues throughout a cycle while testing starts late and never seems to get enough time

Why Do Anything?



Why

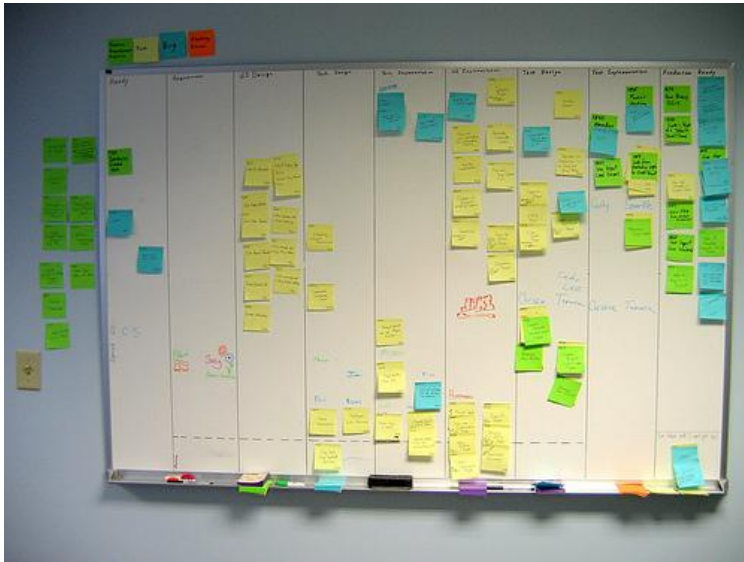
- Using a Kanban approach in software shifts from time-boxed iterations in favor of focusing on continuous flow.



Characteristics of Kanban

- Visualize the workflow
- Limit WIP (work in progress)
- Measure & optimize flow
- Explicit policies (definition of Done, WIP limits, etc)?

Flow: More or Less Complex

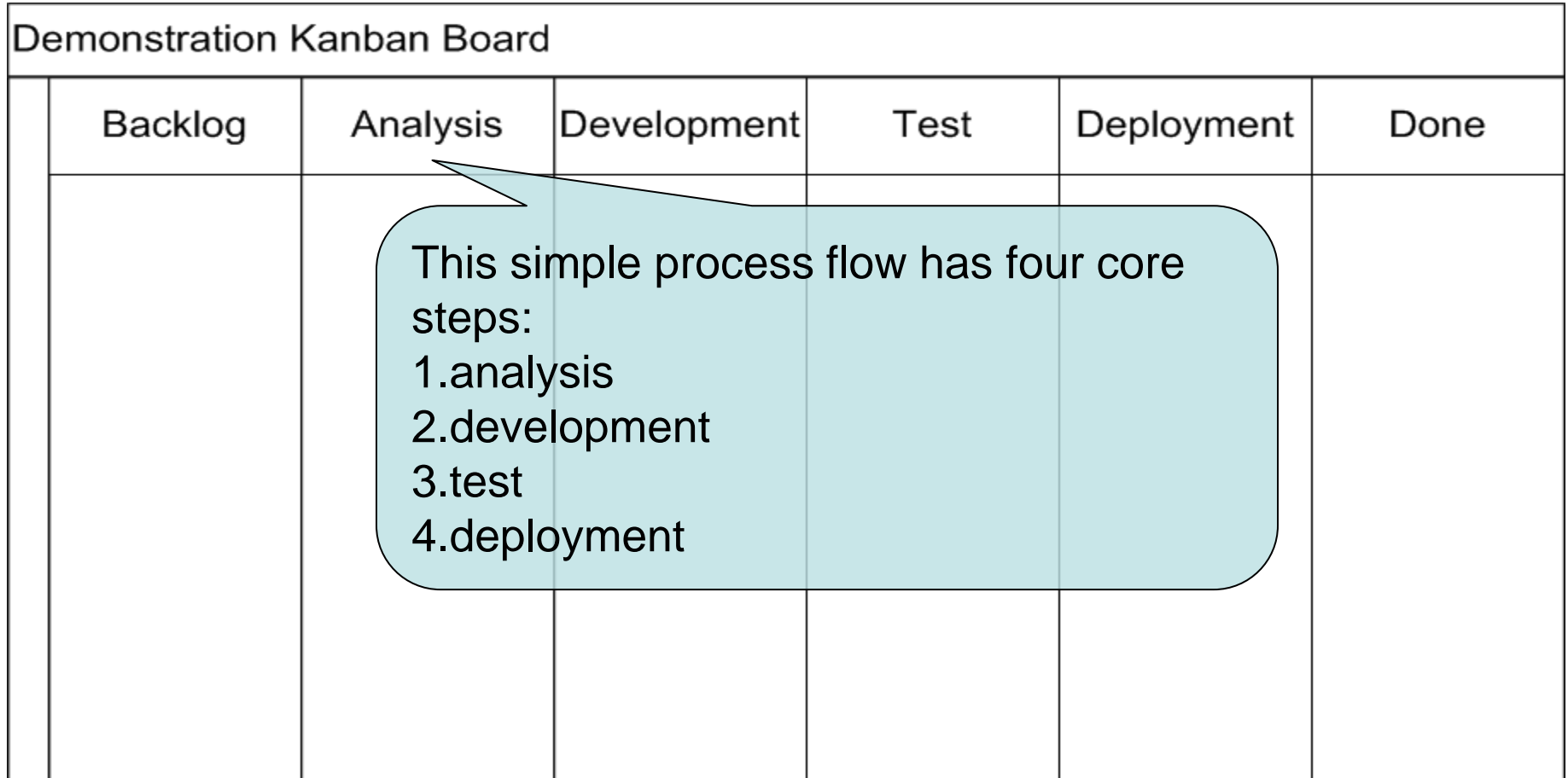


<http://flic.kr/p/4yvFP2>



<http://flic.kr/p/7xD6wF>

A Demonstration



Work In Process Limits

- WIP is **Work In Progress**. Work that has been started but not yet completed (acronym: WIP). In Kanban, each column has a limit of allowed work. It's called WIP limit. How to create a WIP limit:
 - Ask how many people do you have?
 - Start low and observe bottlenecks
 - Use size (function points or story points)



Goal:
Reduce
WIP

Making Explicit Policies

- Kanban Board Itself
- Work in Process Limits
- Coding Standards
- Definition of Done
- Exit Criteria

Making policies explicit is a key enabler of evolutionary, collaborative change in a Kanban System.

Why does this work?
Your role is to:
Observe, Challenge and Change



Using Kanban To Talk About Testing Flow

EXERCISE

Instructions

? Classic Dart Paper Airplane Folding Instructions

DIG. 1



1 Take an A4 sheet and fold it in half [DIG. 1](#)

DIG. 2



2 Fold the short edge of one side down to the first fold (ie produces a 45Degree angle). Do This for the other side too. [DIG. 2](#)

DIG. 3



3 Fold down the new fold you have created to the original fold you did in (1). Repeat for the other side. [DIG. 3](#)

DIG. 4



4 Do step 3 again for both sides [DIG. 4](#)

DIG. 5



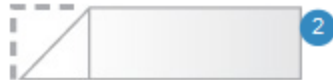
5 Hold Centre and open wings out. Now Throw!!! [DIG. 5](#)

Instructions Iteration One...

DIG. 1



DIG. 2



DIG. 3



DIG. 4



DIG. 5



- Split into groups of 13
 - 3 Four person development teams
 - 1 Independent tester
- Developers
 - One person will make fold one
 - One person will make fold two
 - One Person will make fold three
 - One Person will make fold four and add a logo then hand the plane to the tester
- Independent Tester
 - Inspect (no defects) and test. Test in the order the plane is received.

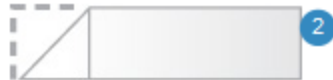


Instructions Iteration Two...

DIG. 1



DIG. 2



DIG. 3



DIG. 4



DIG. 5



- Split into groups of 15
 - 3 Four person development teams
 - 1 Independent tester
- One team . . .
 - One person will make fold one
 - One person will make fold two
 - One Person will make fold three
 - One Person will make fold four and add a logo then hand the plane to the tester
 - One person inspect (no defects) and test. Test in the order the plane is received.

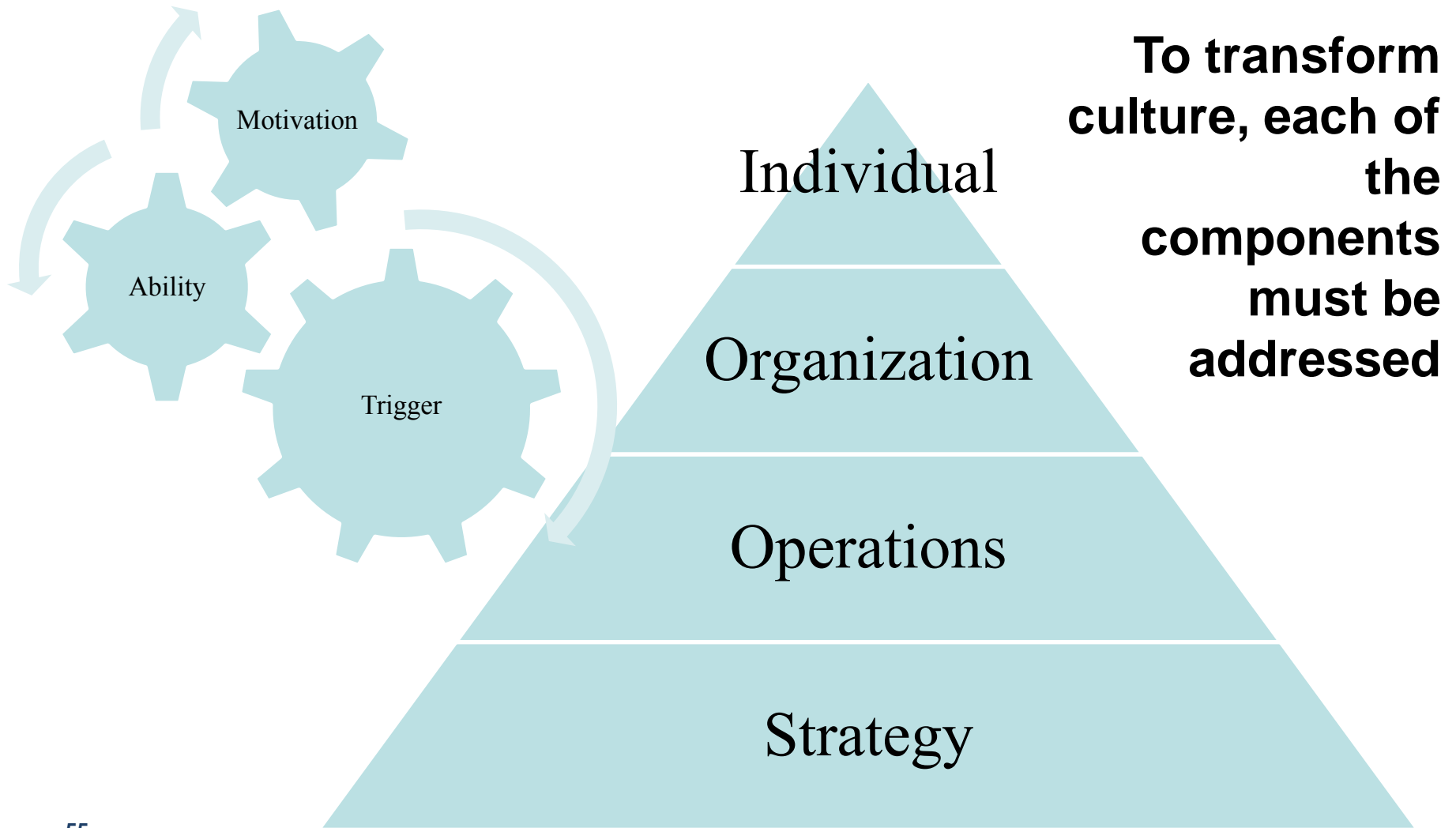
- Which method delivered more totally completed planes?
- Which method had more work in process when the iteration was completed?
- Was the quality the same for both?
- Who should answer the “quality” question?



Quality

AGILE INFLUENCED METRICS

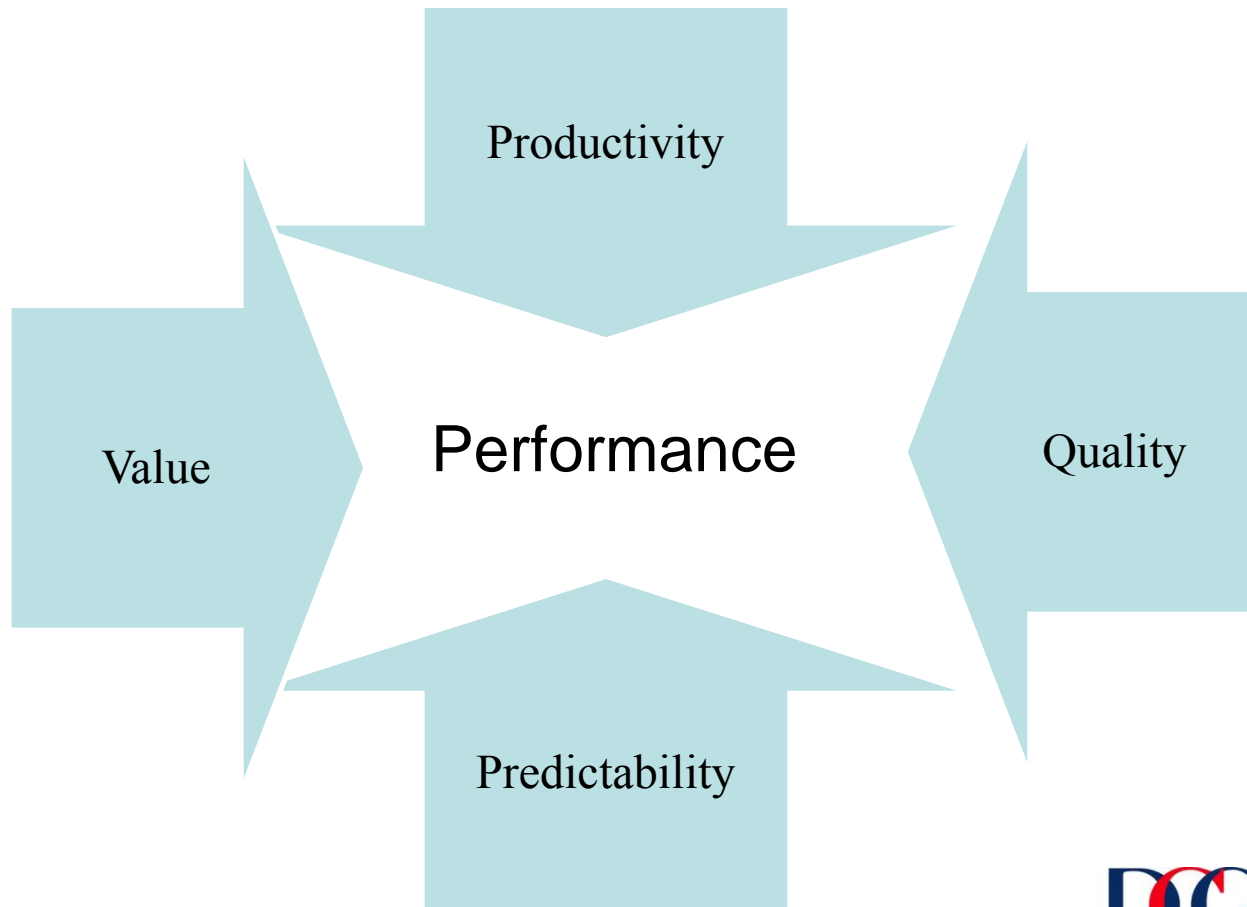
Requirements For Change



Agile Influenced Metrics

- Reinforce desired behavior
- Focus on results
- Measure trends
- Easy to collect
- Includes context
- Creates real conversation
- **ONLY WHAT IS ABSOLUTELY NEEDED**

No Measure To Rule Them All



A Palette

- ROI (value)
- Customer Satisfaction (value)
- Tests Passed (quality)
- Defect Count (quality)
- Technical Debt (quality)
- Work-In-Process (productivity)



Pallet suggests only using those that are absolutely needed.

ROI

A performance measure used to evaluate the efficiency of an investment or to compare the efficiency of a number of different investments.

Indexed Customer Satisfaction

- A customer satisfaction survey contains a number of questions aimed at assessing the customers' view of the team and the work the team is delivering. While the questions are mostly qualitative and individual answers subjective, surveys taken regularly and across a variety of participants will yield useful trends.
- Recommended frequency is with or after each release of the product. If releases are infrequent then perhaps each sprint, but too often will become annoying for the participants. Around every 6 weeks to 3 months feels about right.

Automated Tests

- Cumulative number of passing tests over time is a proxy for quality based on the theory that running more (i.e. passing) tests reflects a positive measure of quality.
- The higher the percentage of automated testing the better.

Defect Counts

- Two metrics to track quality improvement:
 - Post-Sprint Defect Arrival (leading indicator)
 - Post-Release Defect Arrival (lagging indicator)
- Plotted against time (Sprints). The trending of these curves independently and relative to one another can tell us a great deal about the effect of the team's attempts to improve quality ab initio and about its ability to drive down the open defect count.

Technical Debt

- Technical debt is 'undone' work. In other words work that will have to be done in the future in order to bring the code base or other required deliverables to the required quality level.
- Technical debt is always added to the Product Backlog and is prioritized by the Product Owner and team in relation to all the other work.
- The units are story or function points (as for other Backlog items) and these are tracked against time (Sprints).

- Work In Process is a lean metric that helps a team track whether they are working collaboratively or not. The idea in an Agile team is for the whole team, as far as is reasonably possible, to collaborate on a single work item until it is 'done'. This increases the rate of output, quality and cross-learning. It decreases the risk of unfinished items at the end of the Sprint, which results in waste.

Your Pallet

- Keep it simple!
- Focused on organizational goals.
- Measure only what you want to predict and ONLY if you are going to do something about what you will learn!





Software Quality

NUMBERS, PROCESS AND FOOD FOR THOUGHT

Basic Definitions

SOFTWARE QUALITY

Software that combines the characteristic of low defect rates and high user satisfaction

USER SATISFACTION

Clients who are pleased with a vendor's products, quality levels, ease of use, and support

DEFECT PREVENTION

Technologies that minimize the risk of making errors in software deliverables

DEFECT REMOVAL

Activities that find and correct defects in software deliverables

BAD FIXES

Secondary defects injected as a by product of defect repairs

Basic Software Quality Pallet

- Defect Potentials
 - requirements errors, design errors, code errors, document errors, bad fix errors, test plan errors, and test case errors
- Defects Removed
 - by origin of defects
 - before testing
 - during testing
 - during deployment
- Defect Removal Efficiency
 - ratio of development defects to customer defects
- Defect Severity Levels (Valid defects)
 - fatal, serious, minor, cosmetic

Basic Software Quality Pallet (cont.)

- Duplicate Defects
- Invalid Defects
- Defect Removal Effort and Costs
 - preparation
 - execution
 - repairs and rework
 - effort on duplicates and invalids
- Supplemental Quality Metrics
 - complexity
 - test case volumes
 - test case coverage
 - IBM's orthogonal defect categories

Basic Software Quality Pallet (cont.)

- Standard Cost of Quality
 - Prevention
 - Appraisal
 - Failures
- Revised Software Cost of Quality
 - Defect Prevention
 - Non-Test Defect Removal
 - Testing Defect Removal
 - Post-Release Defect Removal
- Error-Prone Module Effort
 - Identification
 - Removal or redevelopment
 - repairs and rework

Hazardous Quality Definitions (Jones)

Does Quality means conformance to requirements

Requirements contain > 15% of software errors.

Requirements grow at > 2% per month.

Do you conform to requirements errors?

Do you conform to totally new requirements?

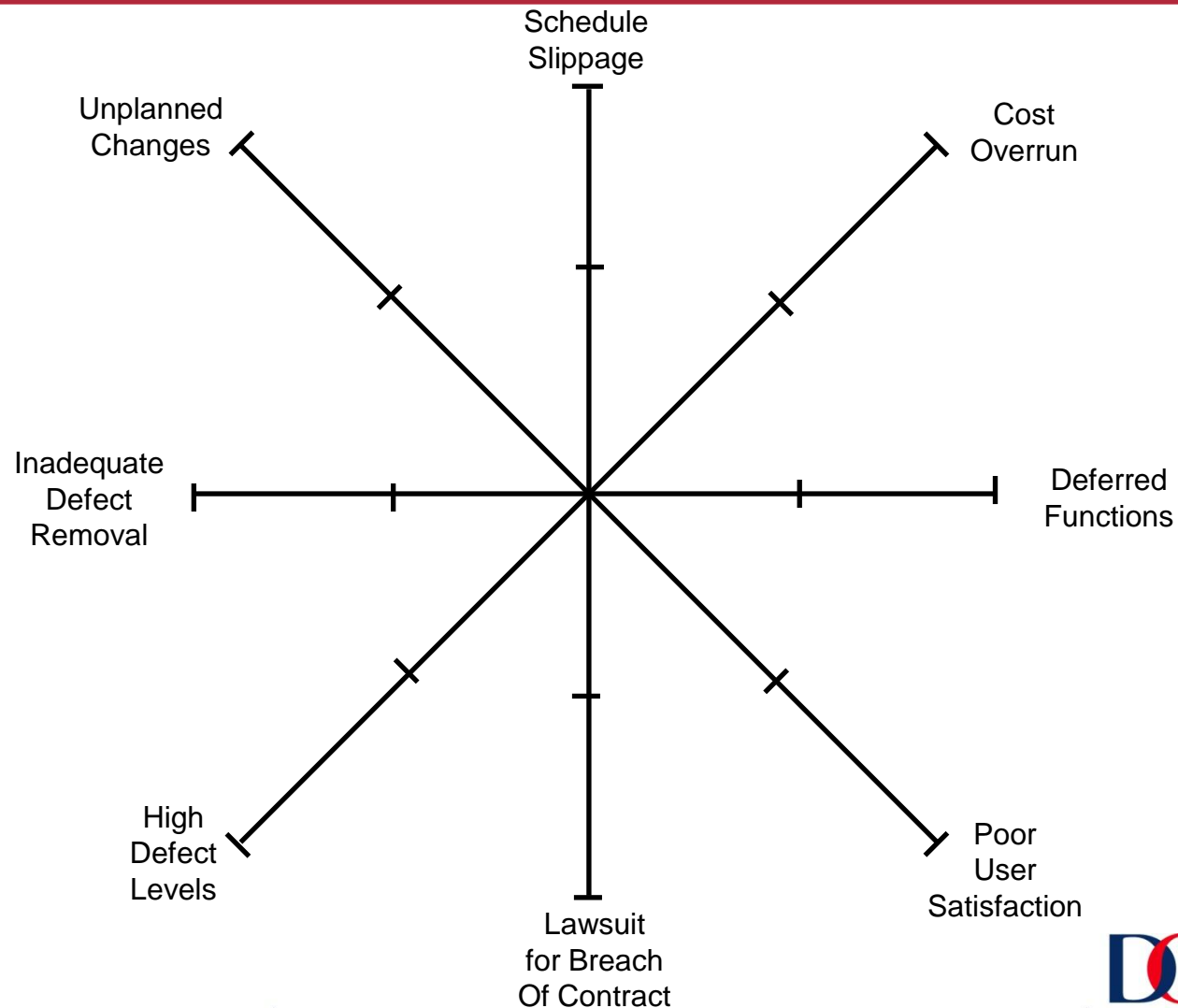
Whose requirements are you trying to satisfy?

Hazardous Quality Definitions (Cont)

Cost per Defect

- Approaches infinity as defects near zero
- Conceals real economic value of quality

Graph Of Major Software Risks



Quality Method Effectiveness And Costs - Jones

METHOD	EFFECTIVENESS	COSTS
• Formal Inspections	Very High	High
• Defect Estimation	Very High	Low
• Defect Tracking	High	Low
• Formal Testing	High	High
• QA Organization	High	High
• Independent audits	High	High
• JAD and QFD	High	Low
• Prototyping	High	Low
• Test Case Tools	High	Medium
• Change Tracking	High	Medium
• Informal Walkthroughs	Moderate	Medium
• Informal Testing	Moderate	Medium
• TQM	Moderate	Medium
• ISO 9000-9004	Marginal	High

Percentage Of Software Effort By Tasks

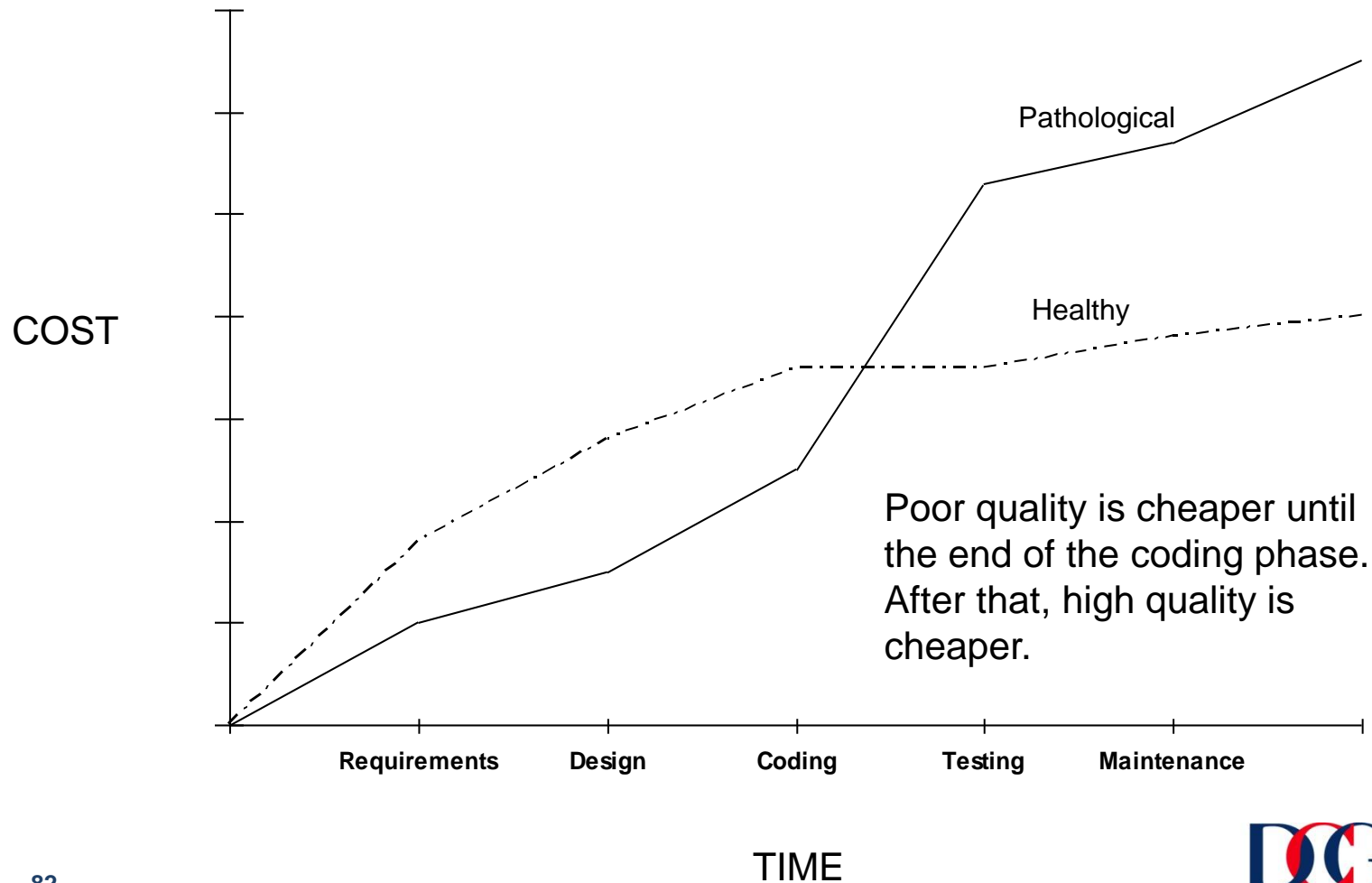
<u>Size in Function Points</u>	<u>Mgt./ Support</u>	<u>Defect Removal</u>	<u>Paperwork</u>	<u>Coding</u>	<u>Total</u>
2,580	16%	31%	29%	24%	100%
1,280	15%	29%	26%	30%	100%
640	14%	27%	23%	36%	100%
320	13%	25%	20%	42%	100%
160	12%	23%	17%	48%	100%
80	11%	21%	14%	54%	100%
40	10%	19%	11%	60%	100%

How do we reduce the amount of effort required defects?

Create less?

Make them easier to find?

How Quality Affects Software Costs



Defect Prevention Methods

	Requirements Defects	Design Defects	Code Defects	Document Defects	Performance Defects
JAD's ^A	Excellent	Good	Not Applicable	Fair	Poor
Prototypes ^A	Excellent	Excellent	Fair	Not Applicable	Excellent
Structured Methods	Fair	Good	Excellent	Fair	Fair
Design Tools	Fair	Good	Fair	Fair	Fair
Blueprints & Reusable Code ^A	Excellent	Excellent	Excellent	Excellent	Good
QFD	Good	Excellent	Fair	Poor	Good



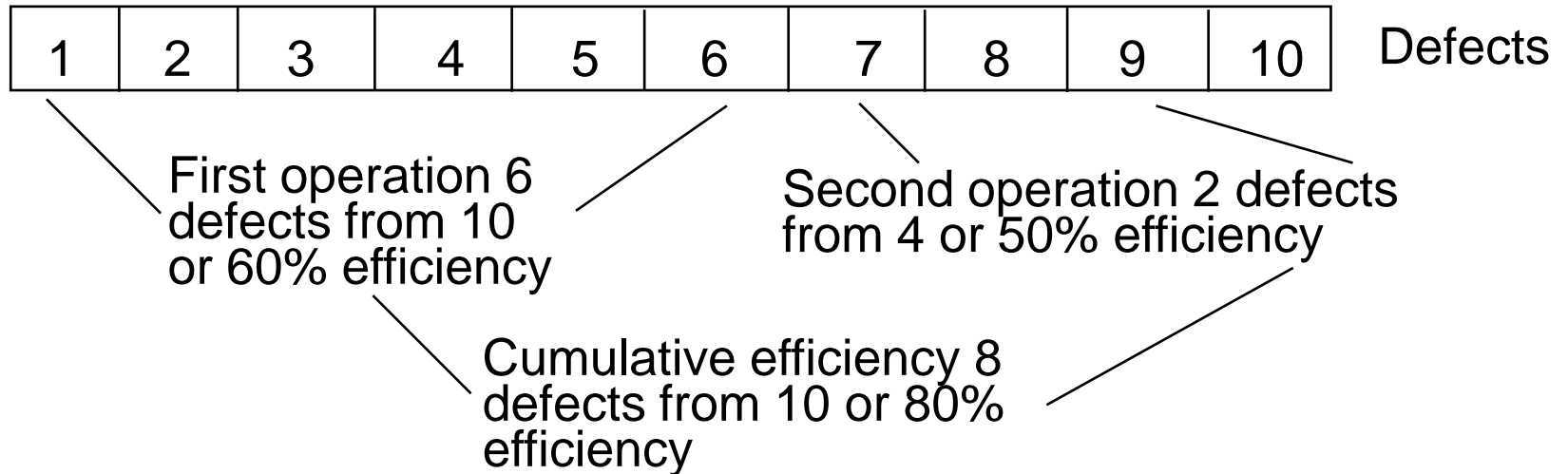
Defect Prevention Methods

	Requirements Defects	Design Defects	Code Defects	Document Defects	Performance Defects
Inspections/ Pair Programming ^A	Fair	Excellent	Excellent	Good	Fair
Stories ^A	Good	Fair	Fair	Not Applicable	Good
Testing (all forms) ^A	Poor	Poor	Good	Fair	Excellent
Correctness Proofs ^A	Poor	Poor	Good	Fair	Poor

Defect Removal Efficiency

- Removal efficiency is the most important quality measure
- Removal efficiency =
$$\frac{\text{Defects found}}{\text{Defects present}}$$
- “Defects present” is the critical parameter

Defect Removal Efficiency



Defect removal
efficiency =

Percentage of defects removed by a single
level of review, inspection or test

Cumulative defect
removal efficiency =

Percentage of defects removed by a series
of reviews, inspections or tests



Defect Removal Efficiency Example

DEVELOPMENT DEFECTS

Inspections	500
Testing	400
Subtotal	900

USER-REPORTED DEFECTS IN FIRST 90 DAYS

Valid unique	100
--------------	-----

TOTAL DEFECT VOLUME

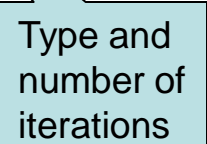
Defect totals	1000
---------------	------

REMOVAL EFFICIENCY

$$\text{Dev. (900) / Total (1000)} = 90\%$$

Number Of Testing Stages, Testing Effort, And Defect Removal Efficiency - Jones

Number of Testing Stages	Percent of Effort Devoted to Testing	Cumulative Defect Removal Efficiency
1 testing stage	10%	50%
2 testing stages	15%	60%
3 testing stages	20%	70%
4 testing stages	25%	75%
5 testing stages	30%	80%
6 testing stages*	33%*	85%*
7 testing stages	36%	87%
8 testing stages	39%	90%
9 testing stages	42%	92%



*Note: Six test stages, 33% costs, and 85% removal efficiency are U.S. averages.

Conclusions and Guidelines

- Measurements Should Directly Support Organizational Goals
- Start With Results Measurements
- Use Real time Scorecards
- The Measurement Set Must Be Internally Consistent
- Measurements Should Be Focused on Project Performance and Processes; Not on Individual Performance

Other Considerations

- You Must Deal With The Issue of Size
- Foundation Systems Are Important
- Collect And Use Data Close to Its Source
- Measurements Should Be a Natural By-product of Work
- Actual Use Is the Fastest Way to Identify Inaccuracies and Get Them Corrected
- Those Who Provide Data Should Use and Verify The Data

Know Where You Are Beginning

- Baseline: A Point-In-Time Inventory of an Organization Which Includes One or a Combination of:
 - Software Size
 - Processes
 - Capabilities
 - Hardware
- Benchmark: A Comparison of Performance against Standard. Typical Standards Include:
 - Industry Averages
 - Baselines



Questions . . .

Tom Cagley

t.cagley@davidconsultinggroup.com

(440) 668-5717 – Cell

Software Process and Measurement Podcast (www.spamcast.net)

“Call me, beep me if ya wanna reach me
When ya wanna page me it's okay
I just can't wait until I hear my cell phone ring
Doesn't matter if it's day or night
Everything's gonna be alright
Whenever you need me baby
Call me, beep me if ya wanna reach me”
- Kim Possible Theme Song

