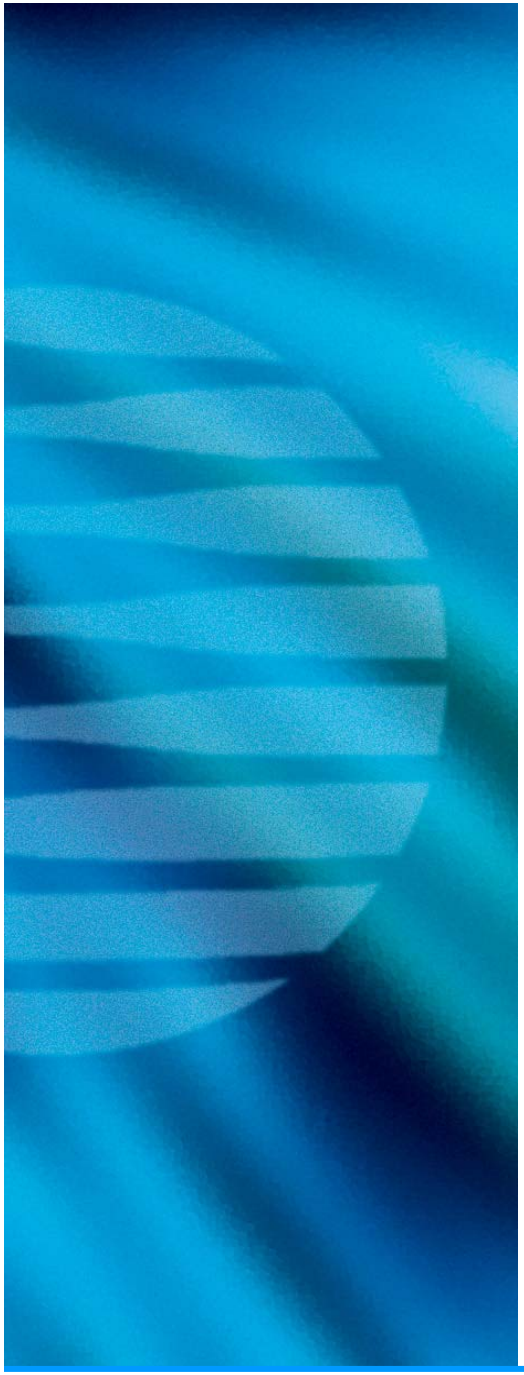


Software Metrics in Aggregate – The Big Picture View

Patricia Hinerman
AT&T
55 Corporate Drive
Bridgewater, NJ 08807
USA

Contributors
John Cirone - AT&T
Patrick Rhodes - AT&T

AT&T

The world's networking company SM

The Software Development Environment

The Big Squeeze

With the economy pressuring companies to cut costs and expense, IT budgets are being questioned because they are usually a large part of a company's internal spending.

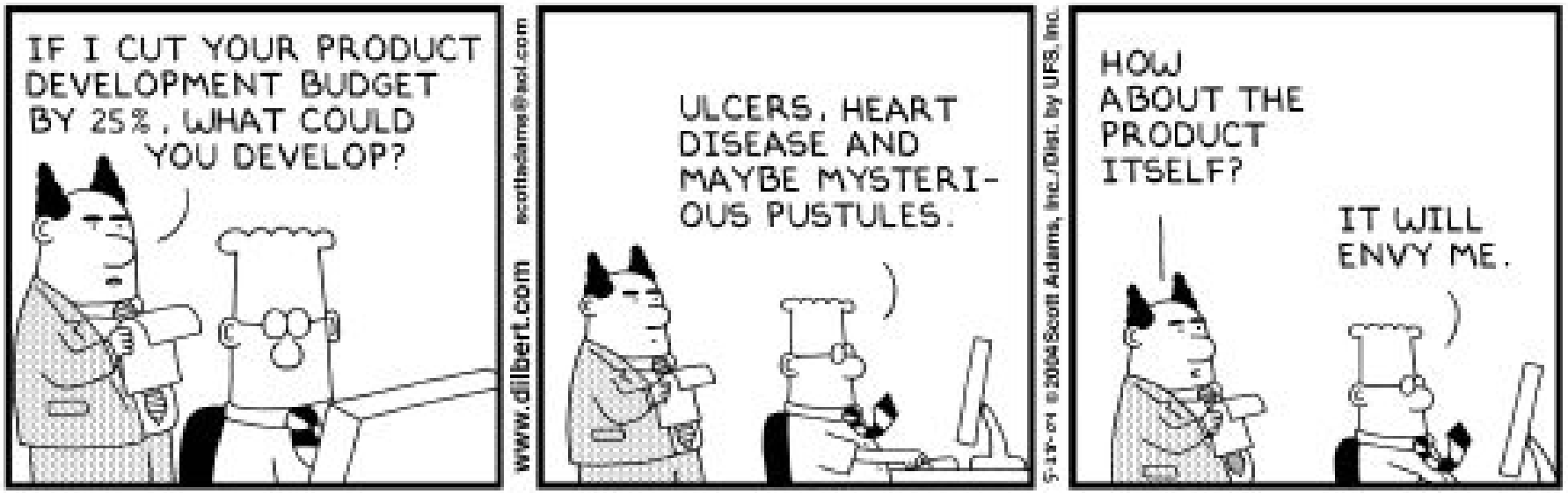
Full Court Press

In order to defend against excessive budget cuts or outsourcing, executives need to know how the IT budgets are being spent and how they compare to outsource competitors in delivering software solutions.

See the light

A metrics program can provide the information they need, but it won't work if you do not have sponsorship!

The Executive View of Software Development



DILBERT reprinted by permission of United Feature Syndicate, Inc.

Beginning a Software Measurement Program

Selling a measurement program to your executives…

What is in it for them?

- Having the ability to demonstrate their (CIO) contribution to the business
- Insight into their development shop's development and maintenance process
- Tools to plan and manage human resources more effectively
- Clarity around tensions between time-to-market imperatives and technical, quality and productivity constraints

Creating a Program that Works

Now that they are sold…

How do you make it work?

- A reasonable phased approach
- Communicate clear goals and definitions of measures and metrics (project dictionary, web site, etc.)
- Developer participation
- Align with current processes, i.e. project management, standard development policies and procedures, etc.
- Management commitment
- Separate & Dedicated program managers
- Regularly scheduled feedback
- Automated data collection
- Training

Creating a Program that Lasts

Now that it is working…

How do you make it stick?

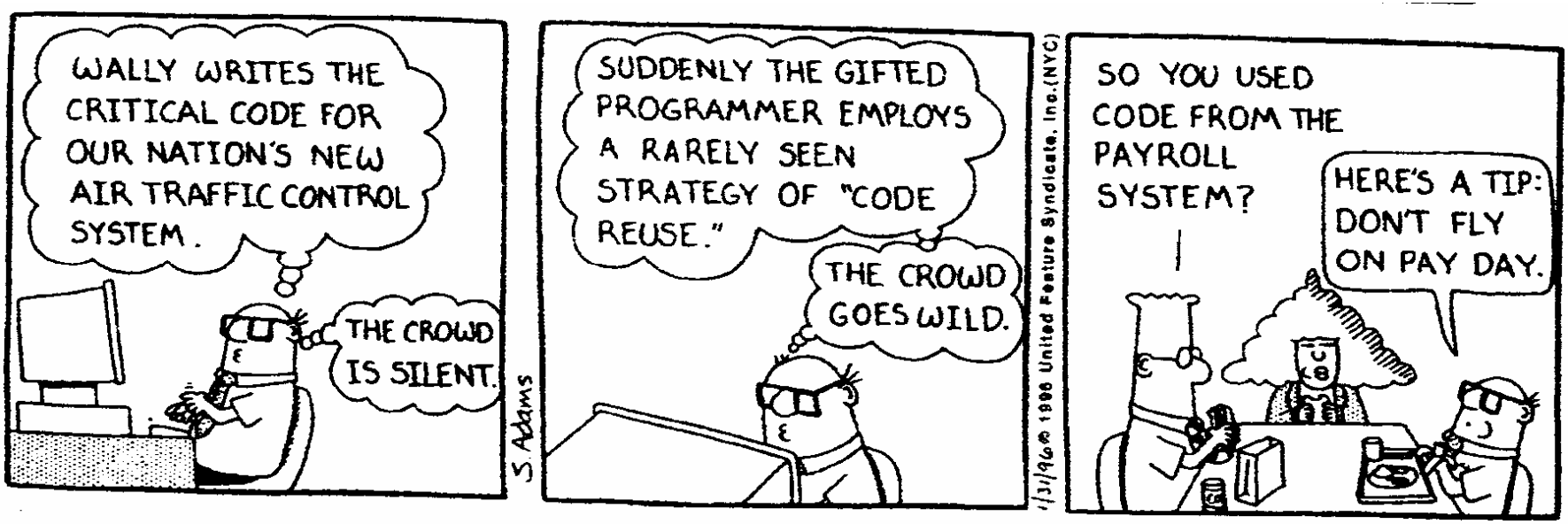
- A phased approach gets everyone used to the idea, and as you get better, it gets easier
- Use a consistent, well-understood measure like function points
- We used an outside vendor to count our function points to share risk, alleviate bias, and tap into an experienced skill set
- Understand that it will take time and practice to get numbers you can count on
- Create an automated measure collection process for capturing function points and resource time reporting

Maintaining a Metrics Culture

The phased approach…

- Prioritized applications into core (critical) vs. non-core and then into small, medium and large.
- We performed baseline counts on all core applications and estimates on the remainder.
- Each quarter, we performed release counts on the baselined applications and replaced estimates with actual counts (set goals each quarter until entire portfolio was baselined).
- When managing to a budget, we counted medium and large releases, and combined small releases over quarters.
- No backfiring; because of its inaccuracy, your metrics could lose credibility.

What Function Points Do Not Consider



DILBERT reprinted by permission of United Feature Syndicate, Inc.

100% Time Reporting is Invaluable

Our time reporting tricks…

- Categorize work effort into projects and activity into tasks
- Group like applications by customer and category (transactional internally developed vs. vendor developed vs. warehouse) into projects
- Define development vs. maintenance tasks
- Combine development into quarterly releases
- Limit the number of projects (process for new projects)
- Engage admin support for 100% time reporting

Projects

project #1	Customer group 1; Internally Developed; Maintenance
project #2	Customer group 1; Internally Developed; Development
project #3	Customer group 1; Vendor Developed; Maintenance
project #4	Customer group 1; Vendor Developed; Development
project #5	Customer group 1; Warehouse; Maintenance
project #6	Customer group 1; Warehouse; Development
project #7	Customer group 2; Internally Developed; Maintenance
project #8	Customer group 2; Internally Developed; Development
project #9	Customer group 2; Vendor Developed; Maintenance
project #10	Customer group 2; Vendor Developed; Development
project #11	Customer group 2; Warehouse; Maintenance
project #12	Customer group 2; Warehouse; Development
project #13	Cross Customer Unique Large Projects

Tasks

Invest in the business (Development)	Concept	
	Feasibility	
	Definition	Q1,2,3,4…
	Development	Q1,2,3,4…
	Service Test	Q1,2,3,4…
	Introduction	Q1,2,3,4…
Run the business (Maintenance)	Corrective	
	Adaptive	
	Perfective	
	Preventative	
	User Support	
Other	Operations	
	Administrative	
	Exceptions (Sick, Vacation, etc.)	

Engaging Software Developers

Developers are not going to be happy, however…

- You are asking them to track their time and provide information that takes time away from their “real work”
- Try to use data that already exists as much as possible
- Automate and simplify data collection
- Engage them when creating metrics process and procedures that affect them
- Provide them feedback on how their contribution to the program, is being reported

Leveraging the Data

We promised metrics, how do we present them…

- Time reporting with effort categorized provides a maintenance to development ratio and overhead which can be compared to plan.
- Create an average labor cost by employee type (employee vs. contractor) to be able to calculate: Cost (Labor Cost/Function Point) to provide insight into how much it costs to develop or maintain one function point (compared to benchmarks).
- Productivity (Function Points/Effort) illustrates how much functionality (how many function points) I can effectively deliver within an effort month or maintain with one full time equivalent on a quarterly basis (also compared to benchmarks).
- Cycle Time (End date – Start date) shows how long it took to deliver functionality as compared to benchmarks.
- As part of a quality program you can track (Defects/Function Points) during development or after release.

Reporting Productivity

Reporting productivity - an Executive "Bird's-Eye" View…

- Function Points per staff month is helpful for an application development manager but an executive wants to know if their development shop is doing well over all.

- We outline 4 ways to provide a "Bottom Line":
  - Trend your actual productivity
  - Calculate percentage of releases exceeding benchmark
  - Average the benchmarks
  - Weight the aggregate benchmark

The same approach can be used for development or maintenance; productivity, labor cost or cycle time.

A Development Shop Example

Table 1

Quarterly Software Release Data

Quarter #1				
A	B	C	D	E
Release #	Type of Application	Enhancement Size	Labor (in staff months)	Productivity
1	Transactional	10	1.2	8.33
2	Transactional	175	17	10.29
3	Data Warehouse	15	3	5.00
4	Data Warehouse	75	9	8.33
5	Transactional	100	7.5	13.33
Quarter #1 Total		375	37.7	9.95

Quarter #2				
Release #	Type of Application	Enhancement Size	Labor (in staff months)	Productivity
1	Data Warehouse	90	14	6.43
2	Data Warehouse	100	12	8.33
3	Transactional	80	8	10.00
4	Data Warehouse	75	9	8.33
5	Data Warehouse	10	2	5.00
6	Transactional	60	4.5	13.33
Quarter #2 Total		415	49.5	8.38

E = C/D

Sum of D

Sum of C

E = C/D

Trending Your Productivity

Table 2

Actual vs. Benchmark Productivity

Quarter #1		
A	**B**	**C**
Release #	**Type of Application**	**Actual Productivity**
1	Transactional	8.33
2	Transactional	10.29
3	Data Warehouse	5.00
4	Data Warehouse	8.33
5	Transactional	13.33
Quarter #1 Total		9.95

Quarter #2		
Release #	**Type of Application**	**Actual Productivity**
1	Data Warehouse	6.43
2	Data Warehouse	8.33
3	Transactional	10.00
4	Data Warehouse	8.33
5	Data Warehouse	5.00
6	Transactional	13.33
Quarter #2 Total		8.38

Difference from last quarter	-1.56
	-15.8%

Benchmarking Benefits

Now that you know your results, how do you compare…

- Benchmark data facilitates inspection of your results for accuracy and provides information necessary to set goals to achieve above industry average.
- Aligning data definitions to benchmark data definitions is no small task in order to get comparative data.
- Benchmarks provided insights into the data that showed that all releases are not equal based on size and type.
- When we were aligned, we identified a “sweet spot” where we would be most productive. It was a range of function points defined as medium developed within approximately a quarter’s time.
- We compared ourselves to industry average 1st quartile (top 25%) and best-in-class (top 5%)
- We were marginally successful working with our vendor to simulate a development shop benchmark

Benchmark Example

Table 3

Function Point Development Productivity Benchmark Data

A	B	C
Type of application	**Enhancement Size in Function Points**	**Benchmark Productivity**
Transactional	1-50	10
	51-150	12.5
	> 150	11
Data Warehouse	1-50	6
	51-150	7.5
	> 150	7

Benchmark productivity expressed as new function points developed (added or changed and adjusted for deleted functionality) per staff-month of labor.

Percentage Exceeding Benchmarks

Table 4

Actual vs. Benchmark Productivity

Quarter #1				
A	B	C	D	E
Release #	Type of Application	Actual Productivity	Benchmark Productivity	Comparison to Benchmark
1	Transactional	8.33	10	☹
2	Transactional	10.29	11	☹
3	Data Warehouse	5.00	6	☹
4	Data Warehouse	8.33	7.5	☺
5	Transactional	13.33	12.5	☺
Quarter #1 Total		9.95		

2/5 or 40% exceeded benchmark

Quarter #2				
Release #	Type of Application	Actual Productivity	Benchmark Productivity	Comparison to Benchmark
1	Data Warehouse	6.43	7	☹
2	Data Warehouse	8.33	7.5	☺
3	Transactional	10.00	11	☹
4	Data Warehouse	8.33	7.5	☺
5	Data Warehouse	5.00	6	☹
6	Transactional	13.33	12.5	☺
Quarter #2 Total		8.38		

3/6 or 50% exceeded benchmark

Averaging Benchmarks

Table 5

Actual vs. Benchmark Productivity

Quarter #1				
A	B	C	D	E
Release #	Type of Application	Actual Productivity	Benchmark Productivity	Productivity of Benchmark
1	Transactional	8.33	10	
2	Transactional	10.29	11	
3	Data Warehouse	5.00	6	
4	Data Warehouse	8.33	7.5	
5	Transactional	13.33	12.5	
Quarter #1 Total		9.95	9.40	105.9%

Average Column D to get benchmark aggregate

E = C / D

Quarter #2				
Release #	Type of Application	Actual Productivity	Benchmark Productivity	Productivity of Benchmark
1	Data Warehouse	6.43	7	
2	Data Warehouse	8.33	7.5	
3	Transactional	10.00	11	
4	Data Warehouse	8.33	7.5	
5	Data Warehouse	5.00	6	
6	Transactional	13.33	12.5	
Quarter #2 Total		8.38	8.58	97.7%

Average Column D to get benchmark aggregate

Weighting Aggregate Benchmark

Table 6

Weighted Average Aggregate Benchmark

Quarter #1					
A	B	C	D	E	F
Release #	Type of Application	Release Size	Actual Productivity	Benchmark Productivity	Labor Required to Achieve Benchmark
1	Transactional	10	8.33	10	1.0
2	Transactional	175	10.29	11	15.9
3	Data Warehouse	15	5.00	6	2.5
4	Data Warehouse	75	8.33	7.5	10.0
5	Transactional	100	13.33	12.5	8.0
Quarter #1 Total		375	9.95	10.02	37.4

F = C/E

Quarter #2					
Release #	Type of Application	Release Size	Actual Productivity	Benchmark Productivity	Labor Required to Achieve Benchmark
1	Data Warehouse	90	6.43	7	12.9
2	Data Warehouse	100	8.33	7.5	13.3
3	Transactional	80	10.00	11	7.3
4	Data Warehouse	75	8.33	7.5	10.0
5	Data Warehouse	10	5.00	6	1.7
6	Transactional	60	13.33	12.5	4.8
Quarter #2 Total		415	8.38	8.32	49.9

Weighted average benchmark

F = C/E

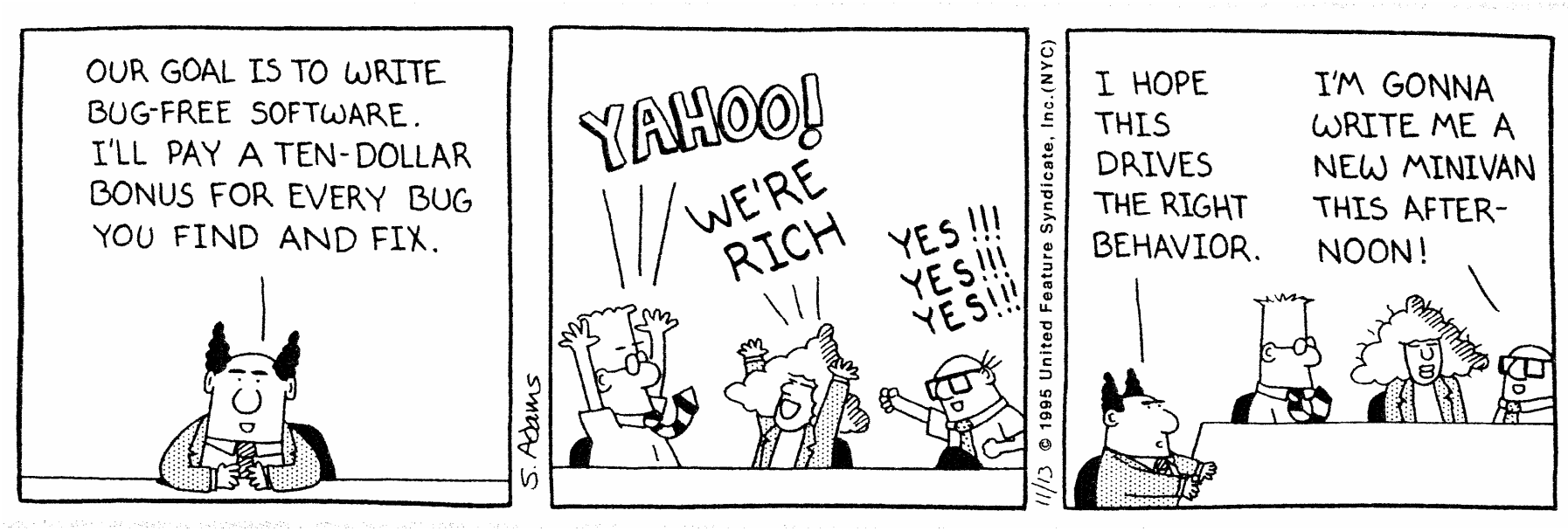
Productivity = Release Size/Labor Labor = Release Size/Productivity

Software Measurement Program

Approach	Trends data over time	Considers application type	Considers release size	Weight the benchmarks	Result
Trend Actual Productivity	*				-15.8%
Percentage of releases exceeding the benchmark	*	*			+10%
Average benchmark data to get comparative Aggregate	*	*			-7.70%
Weight average of benchmark data to get comparative Aggregate	*	*	*	*	**+1.4%**

The last approach takes into account: Release size and Application Type and weights them appropriately.

Metrics Can Be Tricky



DILBERT reprinted by permission of United Feature Syndicate, Inc.

Benchmarking Shortfalls

What we really wanted, but the data was not there…

- Our benchmarks provided us theoretical comparisons.
- Most companies pick out specific projects for tracking purposes rather than track 100% of a development shop.
- If development shops were tracked in whole we could come up with a benchmark that could place development shops in a range categorized into quartiles.
- We could then have a benchmark for a CIO that could say something like, “A first quartile shop exceeds average productivity by 20%”; And this is how you compare…

Continued Process Improvement

Taking it home…

- Validate the process at the fundamental level to ensure a solid foundation.
- Consistent definitions for system applications, documentation, and development vs. maintenance task code structures.
- Create a culture which will expect continued process improvements to to enhance the accuracy of data collected.
- Trending over time provides the opportunity to validate the accuracy of the metrics process. If the numbers look way out there, you may need to understand them better.
- The sound base will provide the opportunity to expand to the aggregate level for executive management reporting.
- Using benchmarks properly adds validity to your numbers.

Opportunities For Our Program

What is next for us…

- Our next phase is to expand further back into the development cycle. Currently we are focused on productivity following the completion of the development lifecycle.
- We have investigated estimating function points (based on detailed user requirements) with our time and costs to provide a tool to help us estimate more accurately for our customers.
    - Estimating assists with resource planning, improves cost estimates, provides more accurate delivery schedules and highlights scope creep.
    - It would enhance our ability to document user requirements well, and help manage user and executive management expectations.
- We also plan to improve defect tracking so that our quality and economic metrics are more closely aligned.

References

Garmus, D. & Herron, D., *Function Point Analysis*, Addison-Wesley, 2001.

Gopal, A., etal., Measurement Programs in Software Development: Determinants of Success, IEEE Transactions on Software Engineering, v. 28, n. 9, September 2002, pp. 863-875.

Hall, T. & Fenton, N., *Implementing Effective Software Metrics Programs*, IEEE Software, March/April 1997, pp. 55-64.

International Function Point Users Group, *Function Point Counting Practices Manual*, Release 4.1, IFPUG Standards, 1999.

Jones, C., *Applied Software Measurement*, Second Edition, McGraw-Hill, 1996.

Offen, R.J. & Jeffery, R., *Establishing Software Measurement Programs*, IEEE Software, March/April 1997, pp. 45-53.

Pfleeger, S.L., *Lessons Learned in Building a Corporate Metrics Program*, IEEE Software, May 1993, pp.67-74.

Wiegers, K., *10 Traps to Avoid,* Software Development, v. 5, n. 10, Oct 1997, pp. 49-53.

Questions and Answers

Future questions:
pburnell@att.com