Presented at the 2013 ICEAA Professional Development & Training Workshop - www.iceaaonline.com

cobec CONSULTING

QUANTIFYING THE FUTURE

The Use of Function Points in Earned Value Management for Software Development

Cobec Consulting
Mike Thompson, Director
Dan French, Senior Consultant
Ben Netherland

Background

- An on-going DOT software program was perceived to be behind schedule, over-budget, and high-risk
  - Program Management had low confidence in the development team's cost/schedule estimates
  - The Development team was being held to an early ROM estimate and the program office was suspicious of the numerous assumptions and qualifiers
  - A realistic, defensible, and repeatable way of reporting software development status was needed to assuage both parties fears

Proposed Solution Requirements

- The EVM solution should utilize an objective software size reporting metric
- Should leverage available data so reporting can start quickly
- Reporting must be easily understood by all levels of management and provide an accurate gauge of program progress
- Produce performance metrics that can be used with the existing EVM tool

The Solution

An International Function Point User Group (IFPUG) Function Point based EVM reporting solution

Function Point Background

- Developed by Allan Albrecht of IBM in 1979
- Created as an alternative to Source Lines of Code (SLOC) for measuring software size
- Counting Rules are established by the International Function Point Users Group (IFPUG)
- Current version is 4.3.1, Released in January 2010
- International Standards Organization (ISO) Standard for software functional sizing (ISO/IEC 20926 SOFTWARE ENGINEERING - FUNCTION POINT COUNTING PRACTICES MANUAL)

What are Function Points?

- Function Points are a unit of software size measure
- Measure the work product of software development
- Work product is measured in terms of functionality from user perspective
- Functions points do not measure internal architecture, effort, or technological complexity of an application

FP EVM Advantages

- When a customer purchases a software development product, they are purchasing functionality, not "Lines of Code."
- Able to establish and measure progress well in advance of full EVMS planning and implementation
- Function Points will not ebb and flow, as SLOC does – functionality earned will continue to increase with time
- Can implement without large investment in EVM processes, tools or personnel

Caveats & Constraints

- Relies upon status being reported by the developer, which can be subjective, and can only be verified by the completion of milestones
  - While providing progress status, the method becomes a modified “Milestone Complete” EVM method, which can be translated into percent complete and entered into an EVM tool
- Does not measure software quality
- System Engineering activities are not in scope

Implementation Background

- Our metrics team met with the program office to determine reporting requirements
  - Status updates established at two weeks after the end of each month
- The system for earning value was discussed with the development team and from those discussions a method of tracking CSCI milestone achievement was developed.

Software Performance Methodology

Use Function Points

- Function points measure how much software functionality is delivered
- Function points are an indicator of the effort required to complete the project
- Function points represent effort in software documentation, code & unit test, and functional lab test

Map FPs to CSCIs

- Function points are counted by requirement
- Requirements can be mapped to Computer Software Configuration Items (CSCIs)
- Result: Function points by CSCI, which provides a relative weighting of each CSCI

Code Reuse & how you go from Function Point to SLOC

- Reused code is taken into account, reducing gross function point/SLOC count to an "effective" FP/SLOC count. Effective Function Points/SLOC are denoted eFP/eSLOC
- Conversion factors enable the translation of function points to SLOC (Source Lines of Code)
- 117.8 SLOC/function point was derived after discussions with Development Team and referencing standard translation tables

Software Performance Methodology

Earned Progress

- Credit is given for completing intermediate milestones in the software process, CSCI Milestones, which include System Engineering and Software Engineering Milestones, this holistic approach to determining progress goes beyond relying solely on Function Points or SLOC as a means of measurement.
- Which means CSCI Milestones can be "earned" before code is created

Implications & Summary

- Value is earned in a way that is results oriented rather than by counting code/function points
- "Heavy hitter" CSCIs that require the most effort are identified early, in a systematic way – not just by gut feel
- Schedule progress is weighted by a factor (FPs) representing effort, presenting a clearer picture of true progress

How Milestones were Weighted

- An initial attempt at establishing weighting for program milestones was done by the metrics team
- The metrics team then conferred with the development team and refined the level-of-effort percentages to the following:

	= Systems Engineering Activities
	= Software Engineering Activities

Milestone / Activity	SW Effort Only	Complete Effort
	Incremental FP % Earned	Incremental % Earned
SSS	0.0%	5.0%
SRS	22.0%	20.0%
IER	0.0%	0.5%
Test Procedures	13.0%	11.0%
TVRTM	2.0%	1.0%
FER	1.0%	0.5%
Coding 50%	8.5%	6.5%
Coding 100%	8.5%	6.5%
Unit Test	14.0%	12.0%
Functional Test 50%	15.0%	13.0%
Functional Test 100%	15.0%	13.0%
Functional Test Report	1.0%	1.0%
Regression Test 50%	0.0%	4.5%
Regression Test 100%	0.0%	4.5%
Regression Test Report	0.0%	1.0%
SUM	100.0%	100.0%

cobec CONSULTING

CSCI Earned Progress from Development Team

1/23/2013	Date of Analysis

Actual Progress	CSCI													
Activity / CSCI	TTCS	SYS	TDCL	FDCS	Router	TSYS	TPGW	SDB	TCSP	DCL	BCI	STM	TMC	CHI
SSS	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
SRS	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
IER	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Test Procedures	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
TVRTM	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
FER	100%	100%	100%	100%	100%	100%	100%	100%	100%	0%	0%	0%	0%	0%
First Half of Coding	100%	80%	100%	100%	100%	100%	100%	17%	100%	40%	9%	0%	43%	40%
Second Half of Coding	100%	0%	75%	100%	40%	2%	100%	0%	100%	0%	0%	0%	0%	0%
Unit Test	100%	30%	66%	100%	52%	38%	100%	6%	100%	15%	3%	0%	16%	15%
Functional Test 50%	100%	0%	0%	100%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
Functional Test 100%	100%	0%	0%	100%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
Functional Test Report	100%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
Regression Test 50%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

- **Table represents current % of milestones achieved, as of 1/30/2013**
- **Table includes both Systems Engineering (SSS, IER, Regression Test) and Software Engineering (everything else) milestones**
- **Cells highlighted in blue have changed since last reporting period**
- **"Weighted % Earned" is the result of the matrix multiplication of the above table with the table from the previous slide**
- **This status is compared to planned progress on the following slide**

cobec CONSULTING

CSCI Planned Progress

Milestone / Activity	TTCS	SYS	TDCL	FDCS	Router	TSYS	TPGW	SDB	TCSP	DCL	BCI	STM	TMC	CHI
SSS	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
SRS	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
IER	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Test Procedures	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
TVRTM	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
FER	100%	100%	100%	100%	100%	100%	100%	100%	100%	0%	0%	0%	0%	0%
First Half of Coding	100%	100%	100%	100%	100%	100%	100%	0%	100%	0%	0%	0%	0%	
Second Half of Coding	100%	0%	0%	100%	0%	0%	100%	0%	100%	0%	0%	0%	0%	
Unit Test	100%	0%	0%	100%	0%	0%	100%	0%	100%	0%	0%	0%	0%	
Functional Test 50%	100%	0%	0%	100%	0%	0%	100%	0%	0%	0%	0%	0%	0%	
Functional Test 100%	100%	0%	0%	100%	0%	0%	100%	0%	0%	0%	0%	0%	0%	
Functional Test Report	100%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	
Regression Test 50%														
Regression Test 100%														
Regression Test Report														
Weighted % Earned	**90.0%**	**44.5%**	**44.5%**	**89.0%**	**44.5%**	**44.5%**	**90.0%**	**38.0%**	**63.0%**	**37.5%**	**37.5%**	**37.5%**	**37.5%**	**37.5%**

- To determine the planned values (how much progress should have been made), we:
  - Entered the scheduled Finish Dates for each CSCI milestone into a table
  - Created a second table that compared the scheduled finish date to the current date
  - If the scheduled date was earlier than the current date, 100% was assigned for that task

cobec CONSULTING

Earned vs Planned Comparison

- The "Weighted % Earned" value for each CSCI is multiplied by the total (when complete) function points for each CSCI to calculate the Earned or Planned function points at a point in time.
- The following slide details how the "earned" and "planned" function points compared

Program Software Metrics – Earned Function Points

CSCI	% Earned	Completed eFPs	Planned eFPs	Planned eFPs for 3/31/2013	eFPs when Complete
TTCS	100.0%	4	4	4	4
SYS	49.0%	31	29	50	63
TDCL	62.1%	4	3	7	7
FDCS	99.0%	27	27	28	28
Router	57.2%	19	15	32	32
TSYS	52.0%	75	67	111	144
TPGW	100.0%	3	3	3	3
SDB	40.3%	8	8	13	21
TCSP	69.0%	12	12	17	17
DCL	42.5%	77	67	86	182
BCI	38.3%	5	5	7	14
STM	37.0%	6	6	7	15
TMC	42.8%	74	64	77	173
TDLS CHI	42.5%	24	21	23	57
Total		369	332	463	759

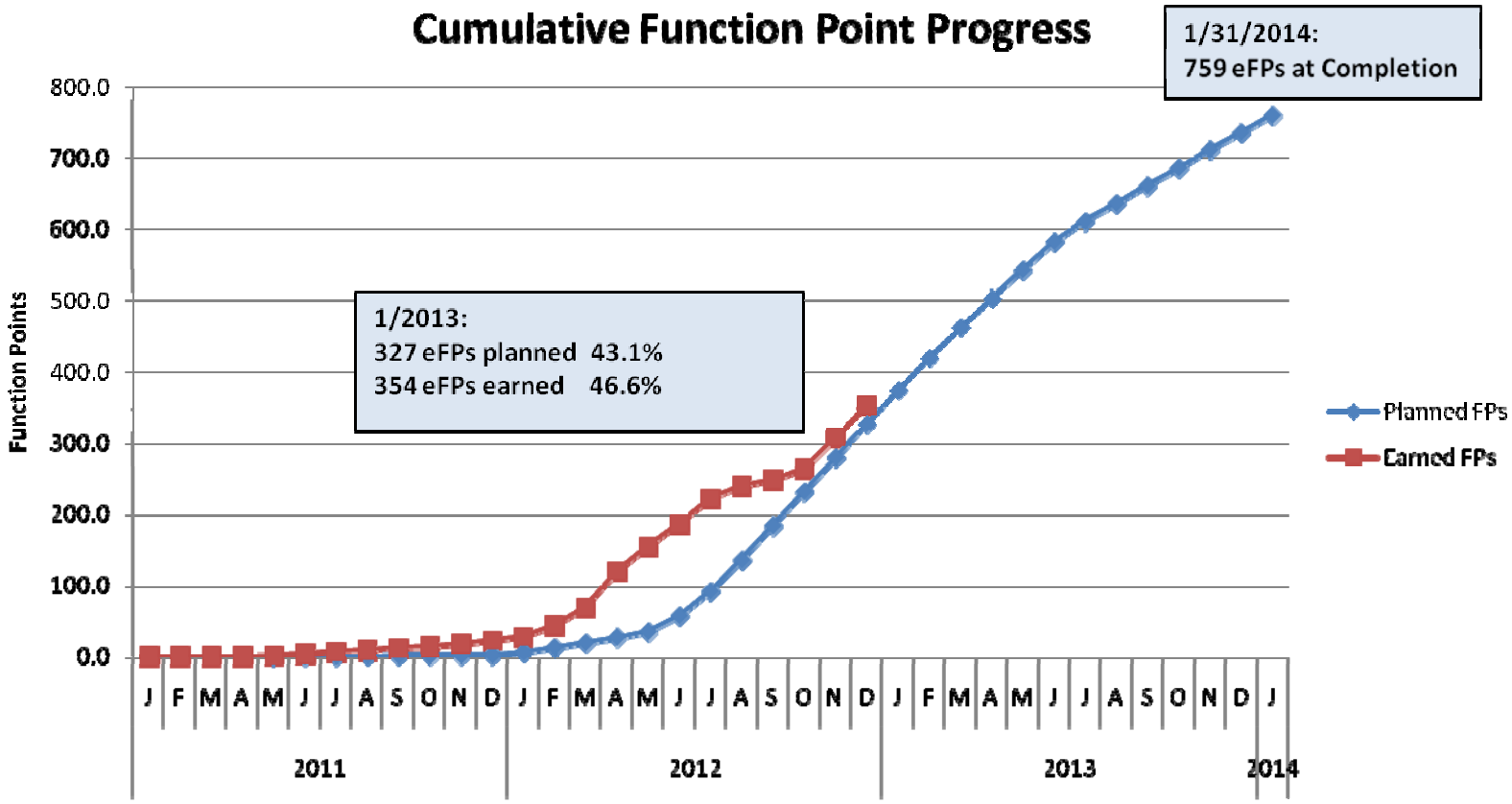
- Note: Progress on Systems Engineering activities is not captured by function points

cobec CONSULTING

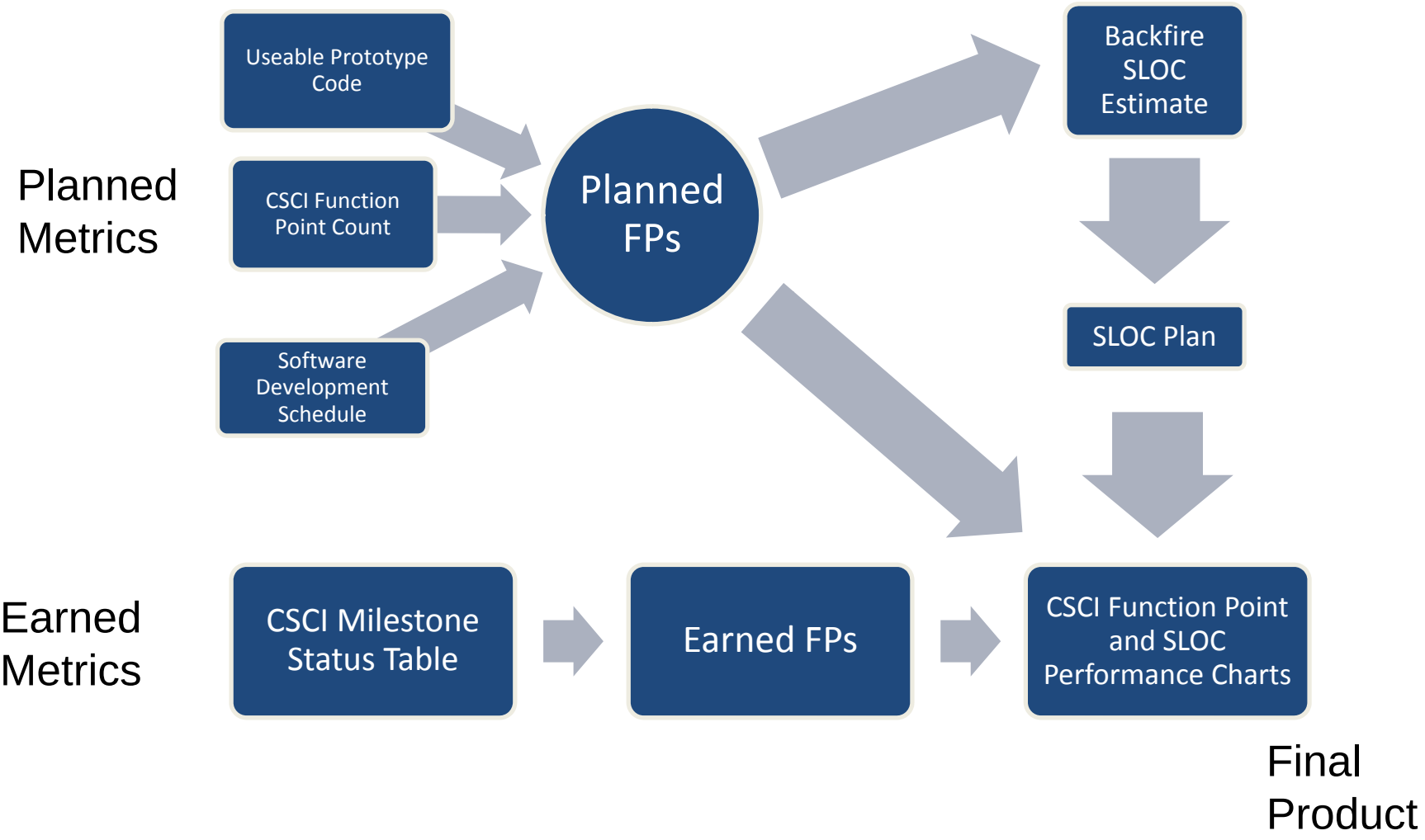
Charting Function Point Progress

- We wanted a more graphical way of displaying progress against the plan, so we decided to chart the cumulative planned and earned function point totals each month
- PLANNED PROGRESS CURVE
  - For each CSCI, the total function points [when complete] were weighted by milestones and allocated according to the Software Development Schedule.
  - When the total of all of the planned distributions was charted, the resulting composite curve looked much like a traditional S-Curve
- EARNED PROGRESS CURVE
  - The earned function points were recorded each month and the cumulative total was overlaid on the planned progress curve

Earned Function Points vs Planned Function Points



SW Metrics Flow Chart Summary



Results

- Performance was proven to be ahead of plan
  - 46.6% complete (vs. 43.3% planned) as of January, 2013
- Relationship between program office and development team improved
  - Program office became much more confident in the development team's ability to meet schedule
- Performance will continue to be tracked and reported monthly

FP based EVM Challenges

- Increased productivity resulting from software reuse must be accounted for:
  - Original size estimate based on the user requirements was roughly 1,400 function points – unadjusted for software reuse
  - Estimate was downgraded to 760 “effective” function points after development team identified requirements addressed by pre-existing code (COTS, open-source, reused)
- Need to account for activities not directly associated with code development (Systems Engineering, System Integration)

Contact Information

- Mike Thompson – mthompson@cobec.com
- Dan French – dfrench@cobec.com
- Ben Netherland – b_netherland@yahoo.com

Last Slide

Questions?

Backup

"Backfiring" - Function Point to SLOC Conversion

- **A number of organizations have made "backfiring" tables available to the public.**
  - These tables contain factors that convert function points to lines of code for various programming languages
- **Some tables are more complete and/or use more data points to come up with their backfiring rates.**
  - The QSM table was used for this estimate, because it used the most data points to derive its backfiring rates
- **In talking with Development Team, it was determined that the programming language blend could be approximated as follows: 75% C (107 SLOC/fp), 9% Ruby (21 SLOC/fp), 10% SQL (30 SLOC/fp), 6% Java (53 SLOC/fp),**
- **The resulting composite backfiring rate (BFR) was 88.32 SLOC/function point**
- **If we use risk ranges on the backfiring rates, the composite backfiring rate becomes 117.8 SLOC/function point**

Risk Range on Backfiring Rates

- Backfiring rates are not given as ranges
- However, a risk range could be determined by subtracting the largest published backfiring rate from the smallest published backfiring rate.
- Backfiring rates from Capers Jones, Cost Xpert, and the David Consulting Group have also been collected by Cobec and were used to calculate a variation of 91% around the chosen QSM rates

Benefits of Using Function Points

- Technology and language independent
- Consistent, repeatable, and verifiable
- Measures functionality the customer requests and receives
- Can use to derive metrics for cost, productivity, and quality
- Enables better management of project scope

Advantages of Function Points over SLOC?

- No consistent rules for defining what constitutes a Line of Code (blank lines, comments)
- SLOC are language and platform dependent, older languages and platforms tend to require more LOC to deliver the same functionality
- SLOC is dependent on the experience and coding style of the individual developer
- FP counts can an be developed earlier and more accurately in the project life cycle