Naval Center for Cost Analysis

Software Resource Data Report (SRDR) Analysis
August 2013 Dataset

Presented by: Nicholas Lanham

June 10-13, 2014

Purpose

- To analyze software productivity and growth relationships when compared to several variables included within DoD Software Resource Data Reports (SRDR)

- Discuss what SRDR variables should be considered when developing software cost estimates

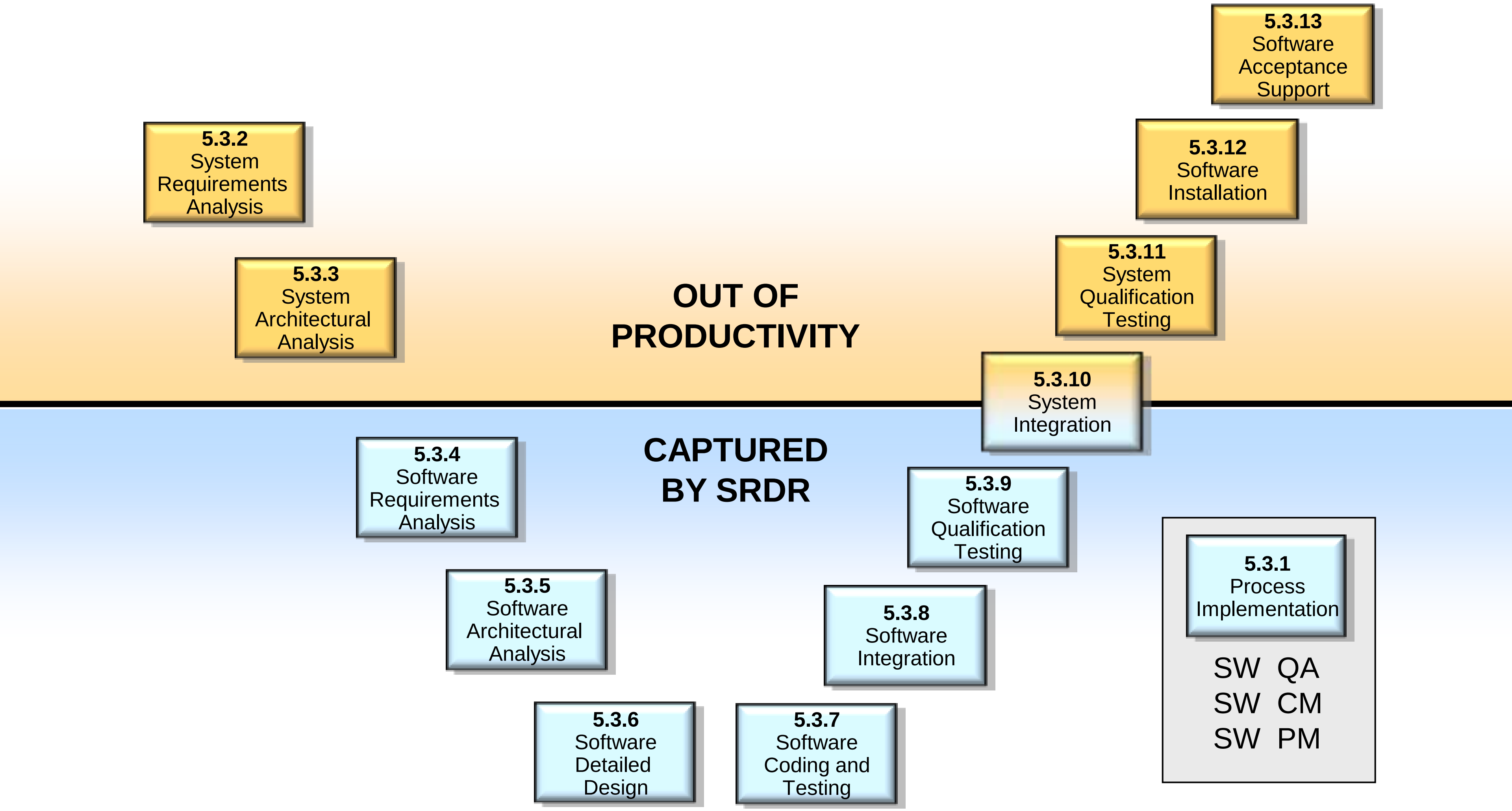
- Develop analysis that informs future SRDR Data Item Description (DID) updates

Table of Contents

What is an SRDR?

- As described on the Defense Cost Analysis Resource Centers' (DCARC) web portal, SRDR data reports are required for contracts meeting the following criteria:
  - All contracts greater than $20 million
  - High-risk or high-technical interest contracts below $20 million
  - SRDR requirements apply to all ACAT IAM, IAC, IC, and ID programs, as outlined below, regardless of contract type
- SRDRs include several performance and reporting variables that enable Government cost agencies to better estimate program software costs
- Examples of reported data variables include:
  - Software Lines of Code (SLOC)
  - Equivalent SLOC (ESLOC) conversion
  - Development hours by IEEE productivity elements
  - Team experience, and so much more!

What Effort is Covered in Reported Hours?



SRDR Data Overview and Progression

- SRDR data used in this analysis is through August 2013
  - Routinely updated to include the latest SRDR data submissions accepted within DCARC's Defense Automated Cost Information Management System (DACIMS)
- The SRDR database is available to Government analysts with access to the DCARC data portal
- Database includes the following SRDR data:

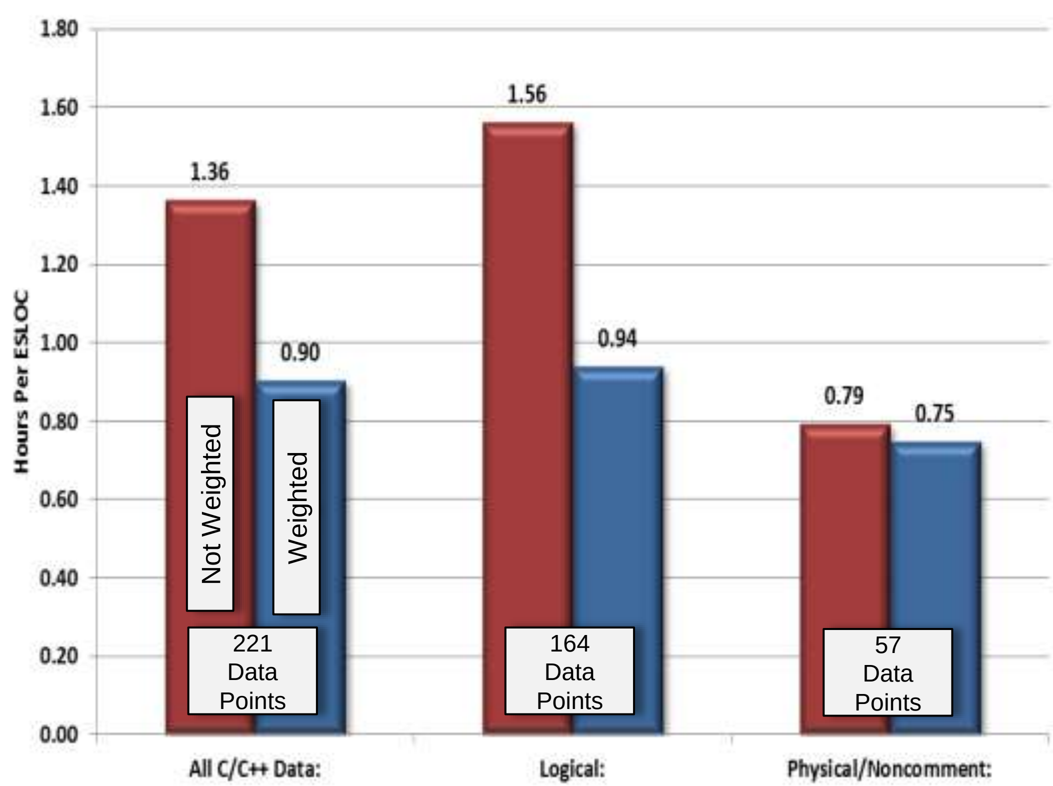
Data Segments	Dec-07	Dec-08	Oct-10	Oct -11	Aug-13
CSCI Records	688	964	1473	1890	2546
CSCI with hrs/ESLOC	N/A	896	1216	1548	2158
Completed program or actual build	88	191	412	545	790
Actuals considered for analysis, "2630-3" & "Good"	N/A	119	206	279	400
Paired Initial and Final	N/A	NA	78	142	212

Language	Data Points in Analysis
Ada	68
C/C++	257
C#	21
Java	46
Other	8

- NAVAIR is the primary reviewer of the SRDR database and conducts routine updates to the existing dataset
- Reasons NAVAIR may choose to reject an actual when updating database
  - Roll-up of lower level data (Did not want to double count effect)
  - Significant missing content in hours, productivity, and/or SLOC data missing
  - Interim build actual that is not stand alone
  - Inconsistencies or oddities in the submit
- ESLOC is calculated within the database using the NAVAIR derived values for new, modified, reuse, and autocode

Physical vs. Logical Productivity Analysis

- Analysis focused on weighted productivity values for logical and physical/non-comment counting conventions
  - "Not Weighted" productivity values include an average of individual CSCI productivity rates, "Weighted" values (preferred method) include total hours divided by total ESLOC
  - Productivity rates were also compared against the existing C/C++ dataset in order to scale against the largest available subset of C/C++ data
- Results indicate that the data includes a slight difference in overall productivity due to counting convention
  - However, various counting tools and inconsistent code counting methods make this method somewhat unreliable as a holistic productivity rate estimating metric
  - Analysts should consider the impact of counting convention as well as what tool(s) has, or will, be used within their given program

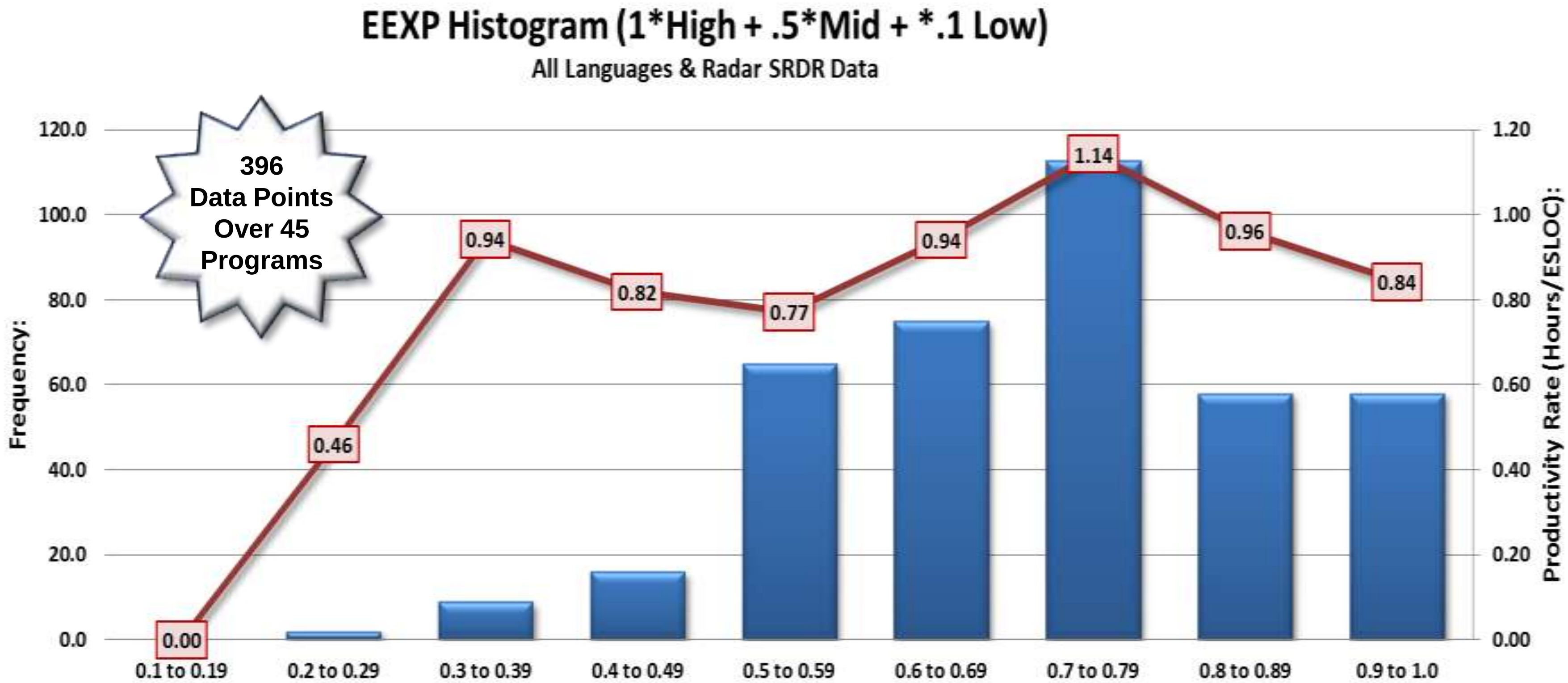


- Includes only C/C++ data, excluding "Radar" designations

Experience Level Productivity Analysis

- "Experience level" analysis used historical, three-level experience breakout (i.e. High, Nominal, and Low)
  - Data points that included "Very High" and/or "Entry" level experience values were added to their respective "High" or "Low" experience percentages
  - Majority of SRDR data points Do Not include experience levels within the "Very High" and/or "Entry" categories (Due to recently revised SRDR content requirements)
- Each category weighted to illustrate cumulative frequency distributions by calculating Equivalent Experience (EEXP) levels for each data point
  - EEXP = (High * 1.0) + (Nominal * .5) + (Low * .1)
  - Data points with large portions of staffing categorized as "High" will be closer to 1.0
- Based on this analysis, "experience level" does not represent a valid estimating variable for productivity rates
  - Staff turnover during lengthy development forces a guess on skill mix
  - Most contractors will default to "standard" reporting percent allocations
  - Programs (Contractors) tend to report similar mix of high, nominal, and low skill mix
  - Requires guessing by the cost analyst to "predict" experience level of team

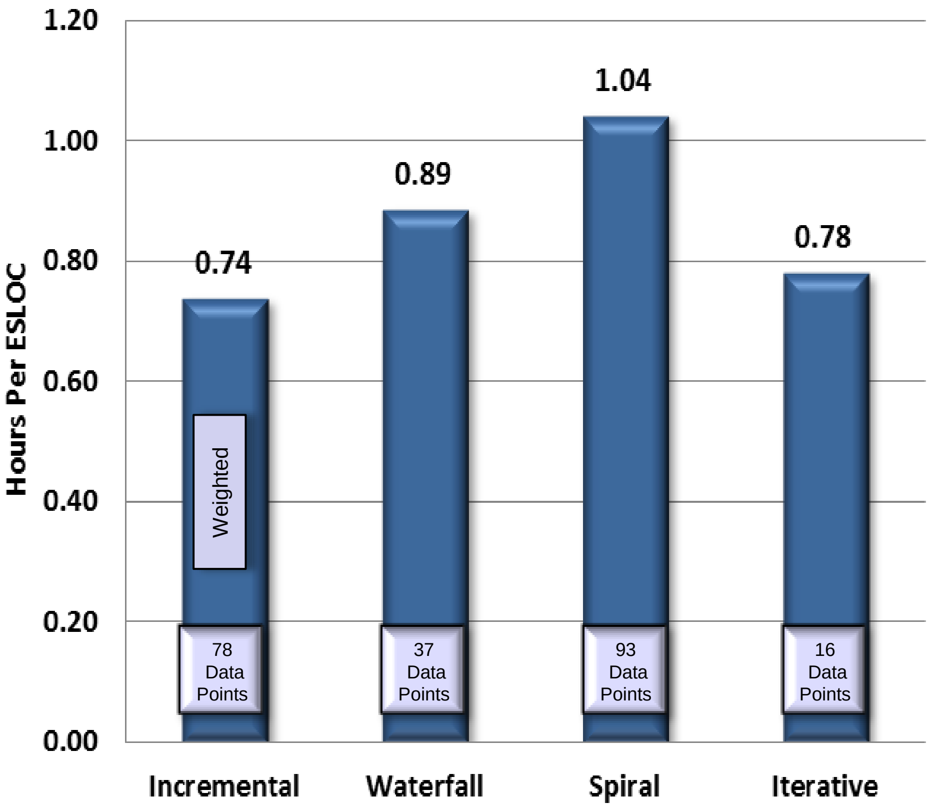
Experience Level Impacts on Productivity



- Data indicates <u>No</u> clear relationship between experience level and productivity-rate (Hours/ESLOC)
  - "Highly" experienced staffing levels resulted in similar productivity rates when compared to "low" and "intermediate" staffing
- Includes all language types and "Radar" data
- C/C++ dataset illustrates a very similar trend

Development Process Productivity Analysis

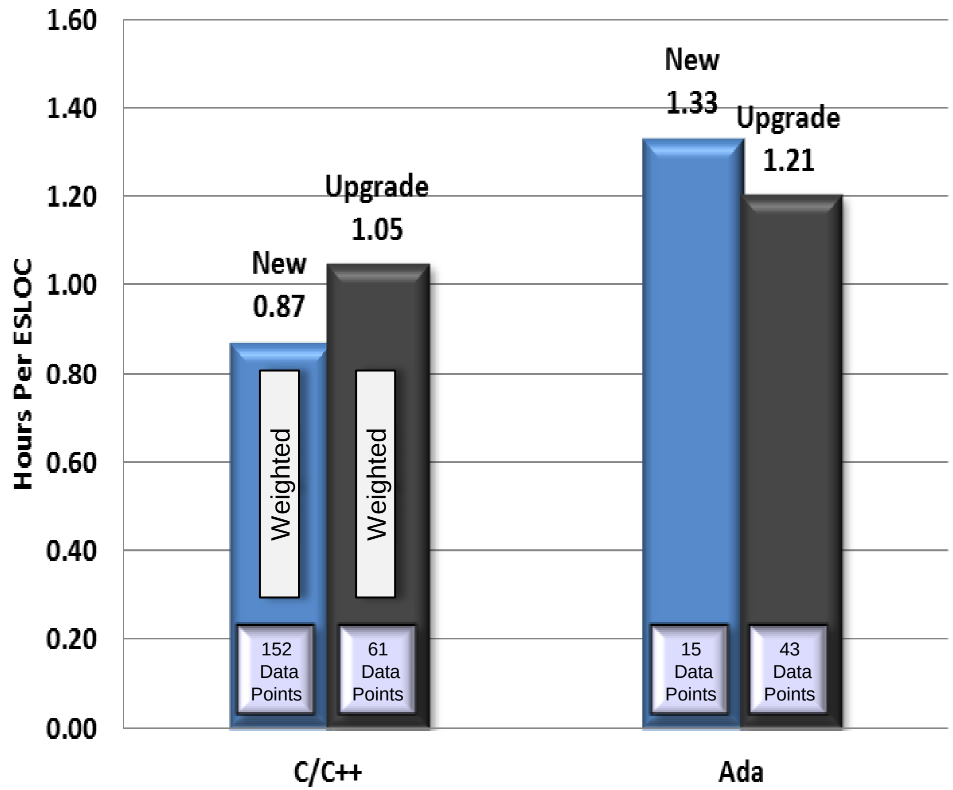
- Analysis examined Incremental, Spiral, Waterfall, and Iterative development processes to determine whether they influenced productivity
  - Other process did not have enough data to evaluate
  - Weighted productivity used to represent total hours per individual development process divided by total derived ESLOC
- Data indicates that developmental process does impact software development productivity
  - Consider development process productivity impacts when using language-focused SRDR-derived C/C++ estimating relationships



- Includes only C/C++ data, excluding "Radar" designations

New and Upgrade Productivity Analysis

- Analysis examined productivity behaviors resulting from “New” and “Upgrade” development efforts
- C/C++ provides adequate data to conclude that productivities differ for new efforts vice upgrade efforts
- ADA illustrates a similar trend
- JAVA includes a larger amount of “New” SLOC vice “Upgrade”
- C# did not provide adequate data to quantify impacts specific to “New” or “Upgrade” efforts
  - Illustrates the importance for analysts to request detail regarding the development type, especially if developers plan on leveraging C/C++ or Ada



- Includes only C/C++ and Ada data, excludes “Radar” designations

Productivity by Language Type Analysis

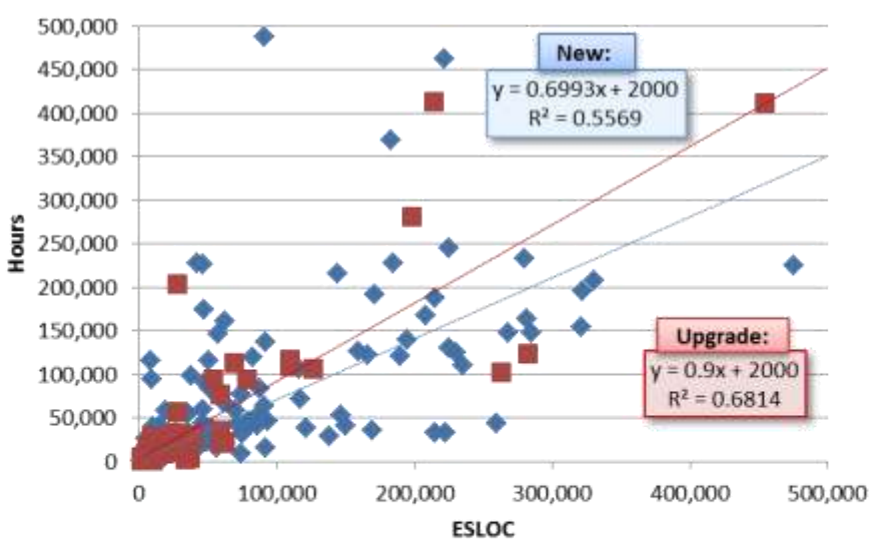
- Productivity by language type analysis focused on linear regression(s) with an intercept at 2000 hours
  - Equates to approximately one FTE
- Productivity by language-type results included within the table below, and highlighted within the following slides
- Even though Radar programs are not considered a "language type", Radar efforts do represent a distinct productivity behavior within the SRDR data
  - Combined all data regardless of language
  - Looked at results with and without two "outlier" data points
- In addition, radar CSCI's resulted in less efficient productivity rates than compared to other CSCI records

Language Type:	Productivity Hours / ESLOC:
C#	0.26
Java	0.84
Ada New	1.17
Ada Upgrade	1.12
C/C++ New	0.70
C/C++ Upgrade	0.90
Radar W/ Outlier	1.29
Radar W/O Outlier	1.50

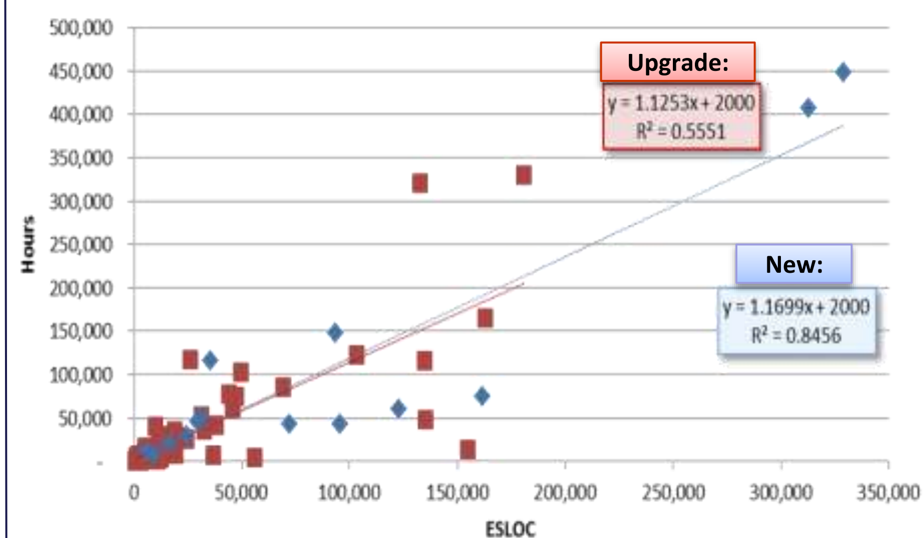
- **Values refer to regression relationships illustrated on the following slide(s)**

Productivity By Language Type

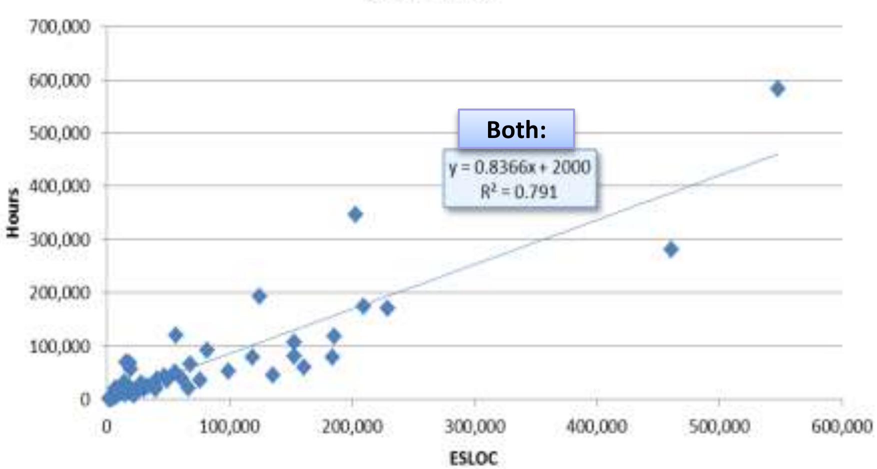
C/C++ Data Zoomed View:



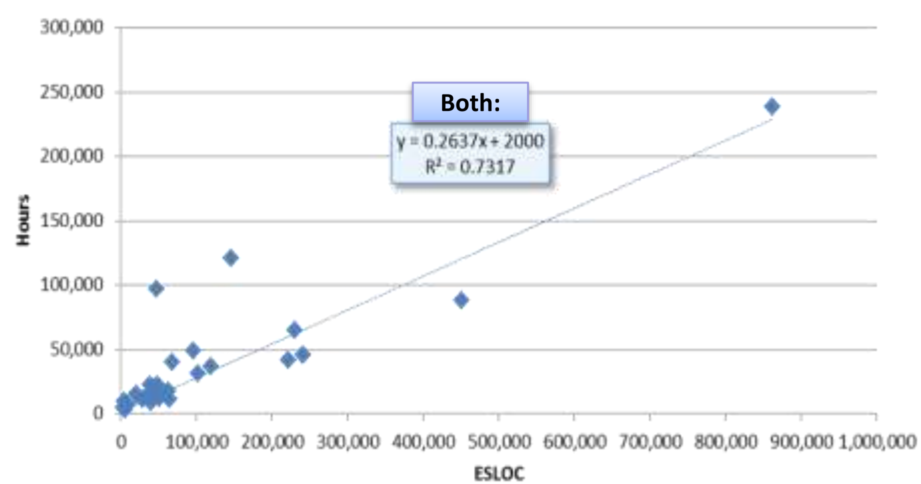
Ada Data:



Java Data:

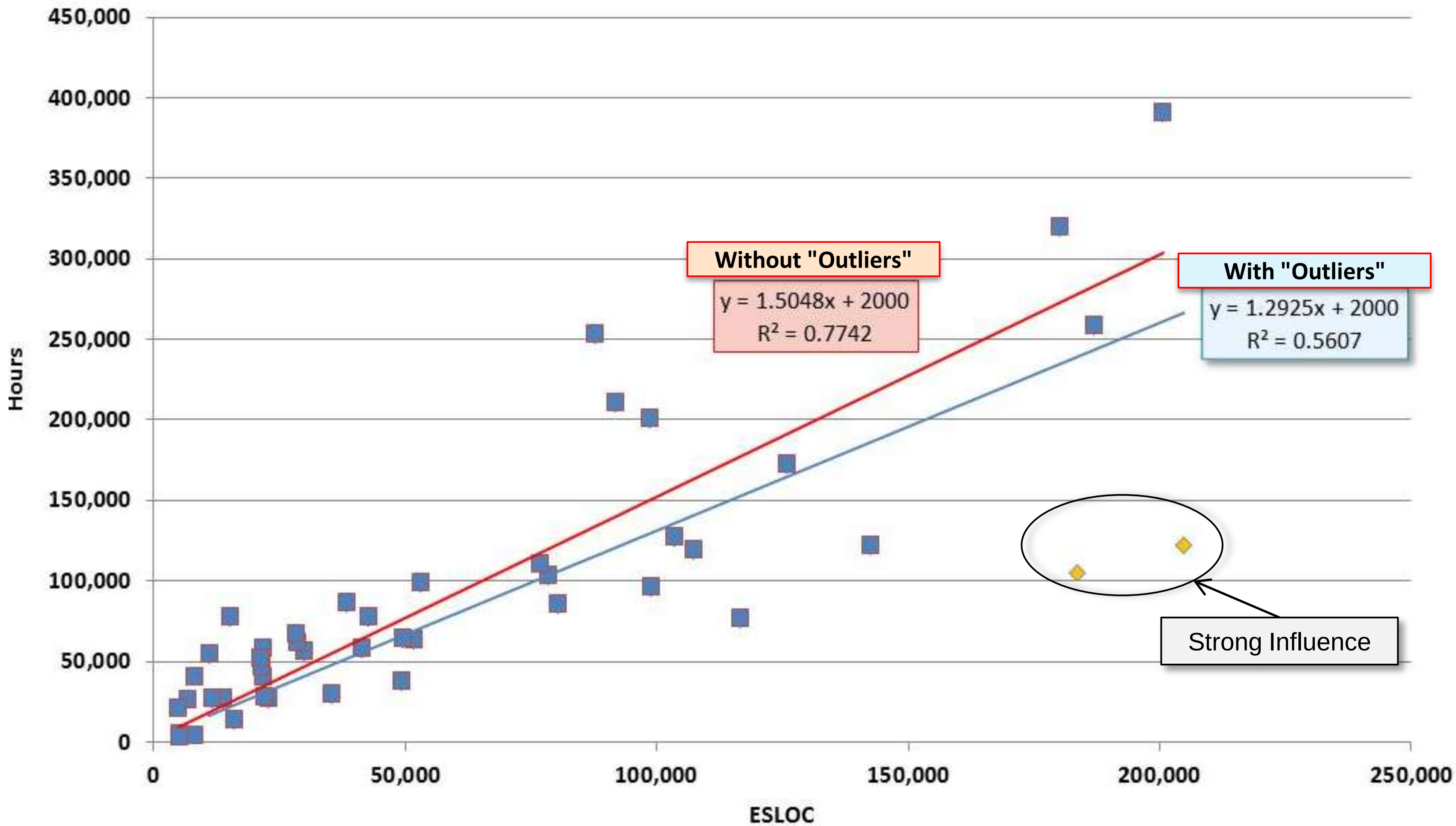


C# Data:



- Includes Final (2630-3) and "Good" data points

Radar Productivity Analysis



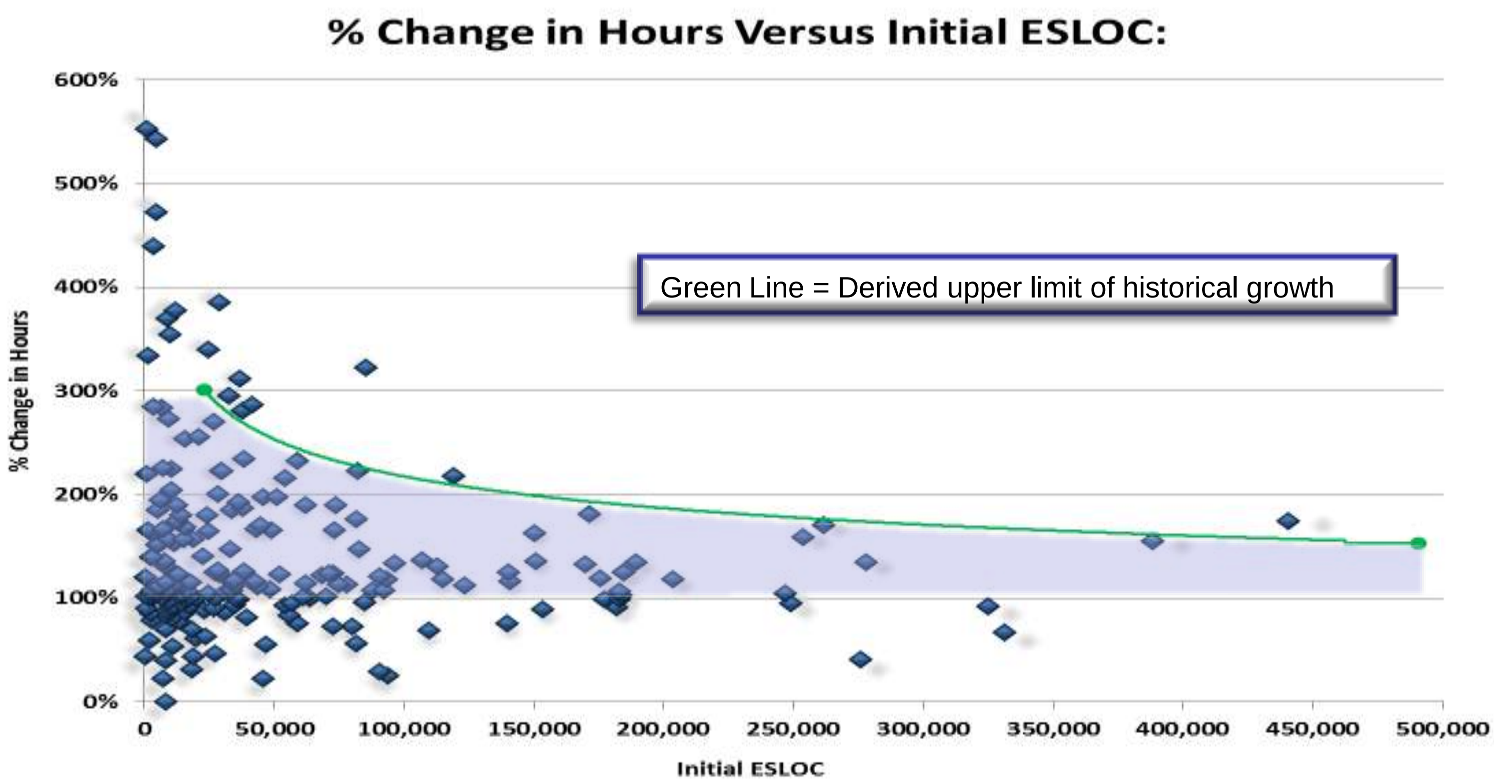
- Includes All language types designated as Radar within SRDR database (45 data points)
- Final (i.e. 2630-3) and “Good” records

Software Change From Initial to Final Reports

- Analysis illustrates that growth/change in hours behaves differently than growth in ESLOC
- However, change in software development hours should represent the primary focal point for cost estimating purposes
  - Historically software change has focused on ESLOC variations from initial to final reporting events
- Data indicated that change in hours could be modeled as a function of starting ESLOC size
  - Further described on the next slide

Percent Change in Hours and ESLOC

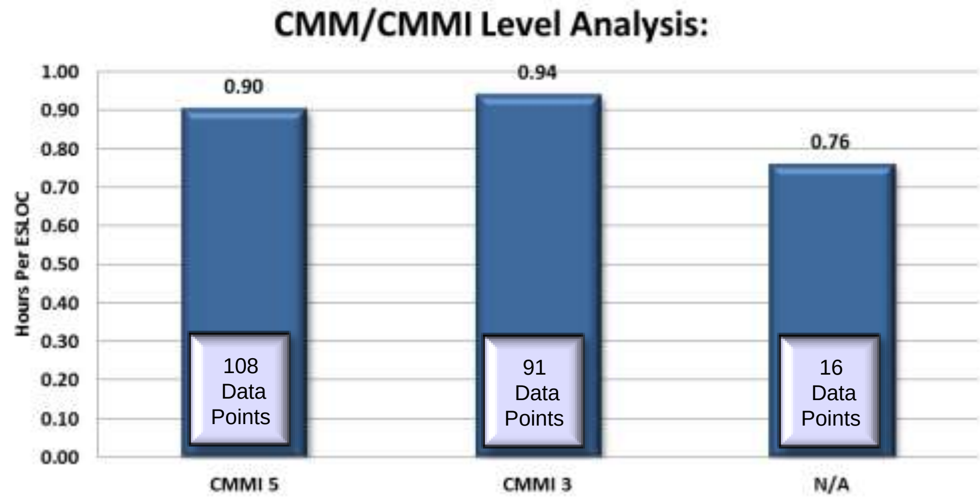
- Software development hour "growth" behaves in a discernable pattern when related to initial ESLOC size
  - Important to note that this analysis focuses on individual CSCIs that result in ESLOC values lower than 500K
  - Large programs experienced less growth, potentially due to higher maturity development process and increased estimating rigor



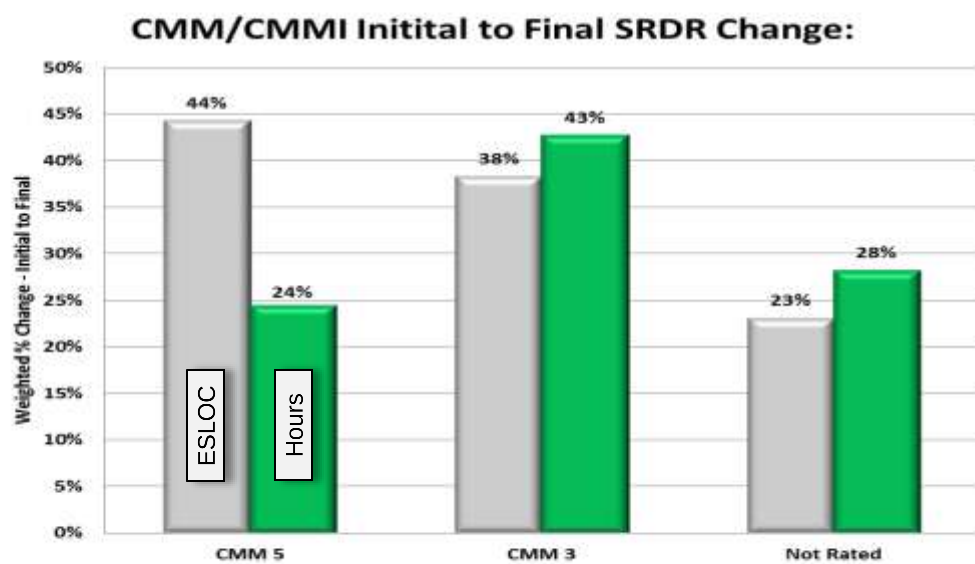
- 100% equals no change in this graph

CMM/CMMI Level Analysis

- Capability Maturity Model Integration (CMMI) provides a consistent measurement of process improvement across a reporting organization's individual division(s), development teams, or cumulative development enterprise
- Analysis indicates CMMI "level 5" and "level 3" organizations result in very similar weighted productivity values
- Additional analysis clearly highlights the CMMI level impact of future development hour growth from initial to final reports
  - Software size (ESLOC) remained relatively consistent from "Initial" to "Final" reporting events
  - The change in total development hours significantly decreased from CMMI "level 3" to "level 5" organizations



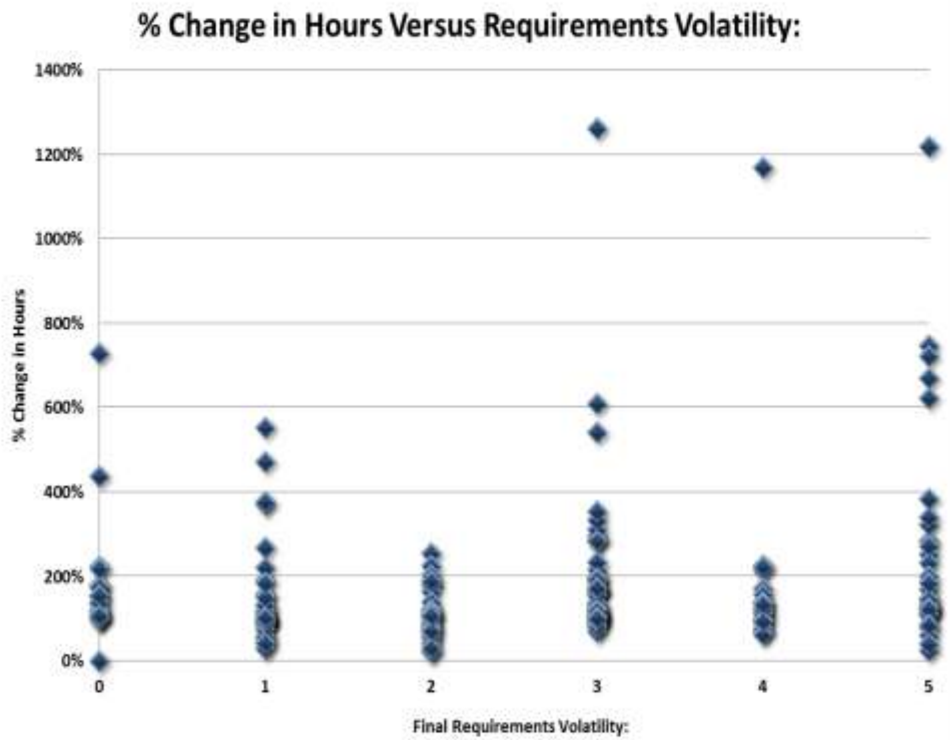
- Includes only C/C++ data, excluding "Radar" designations



- Data from August 2013 paired data set – all language types
- Variance in all groupings is so large that there is no statistical difference between the averages

Requirements Volatility

- Contractors typically provide subjective requirements volatility ratings
  - Volatility ratings based primarily upon estimated/perceived requirements change
  - Possibly related to unclear or inconsistent method of calculating requirements volatility from program to program
- Largest portion of data points are included under ratings 1 (no change), 3, and 5 (extreme change)
  - 10% of "paired" reports include no requirements rating
  - Scatter plot indicates similar percent change in hour groupings between individual volatility ratings
  - Largest portion of "paired" data points reported as "level 3" volatility



- Data from August 2013 Paired data set – all language types
- 100% = No growth

Analysis Summary

- SRDR analysis results provide cost analysts with several productivity variables to consider when developing future software estimates
- In addition, this analysis also highlights the need for Government agencies to collect and utilize SRDR variables that are relevant, and routinely tracked by contracting agencies
  - "Experience level" potentially represents a variable that is not consistently reported and/or tracked by contracting companies
  - Development process continues to drive slight impacts on overall program productivity rates
  - Radar programs continue to behave less efficiently (in terms of productivity rates) than language type analysis

Value of SRDR Data

- SRDR data provides analysts with a set of actual, DoD-specific, software productivity metrics
  - Significantly enhances the Government's understanding and negotiation position for future software development efforts
- SRDR data continues to provide the government with unprecedented insight into contractor software development efforts
  - Data supports some historical "benchmarks" while others are not supported
- Readily accessible to Government organizations with access to DACIMs, or FFRDCs
  - Greatly under utilized resource
  - You can use the NAVAIR compiled Excel file or individual SRDRs for deeper analysis
  - Allows analysts to make their own decisions based on the data and provides very flexible data tables for your own specific use

Future Analytical Efforts

- SRDR phasing by IEEE productivity element
- Analyzing and highlighting the need for Government required reporting of VHDL development efforts (i.e. Firmware)
- Additional relationships to software growth/change from initial to final reporting
- Contract-type relationships and potential impacts to overall productivity rates or total development hours
- Lower-level “Reuse” and “Modified” productivity rate impact analysis
- COTS integration productivity impacts
- Agile development process impacts on DoD software development efforts
- Software development trends further analyzed within 5-7 year ranges

If you have questions related to this presentation, please feel free to contact:

Nicholas Lanham
Naval Center for Cost Analysis (NCCA)
703-604-1525
Nicholas.lanham@Navy.mil