Get to the Point. What's the Deal with Different Function Points Methodologies?

A Preliminary Empirical Comparison

Authors and Collaborators

Anandi Hira
AHira@Tecolote.com

Katharine Mann
Katharine.Mann@hq.dhs.gov

Paul Cymerman
Joe VanDyke
{Pcymerman, JVanDyke}@quaternion-consulting.com

Ian Brown
Ibrown@avantusfederal.com

Dave Seaver
David.P.Seaver.civ@mail.mil

Kevin McKeel
Kevin.McKeel@logapps.com

Overview

Presentation Agenda

- Motivation and size methods explored
- Research methodology and dataset
- Results
- Conclusions

Table of Contents

Get to the Point (paper) – table of contents

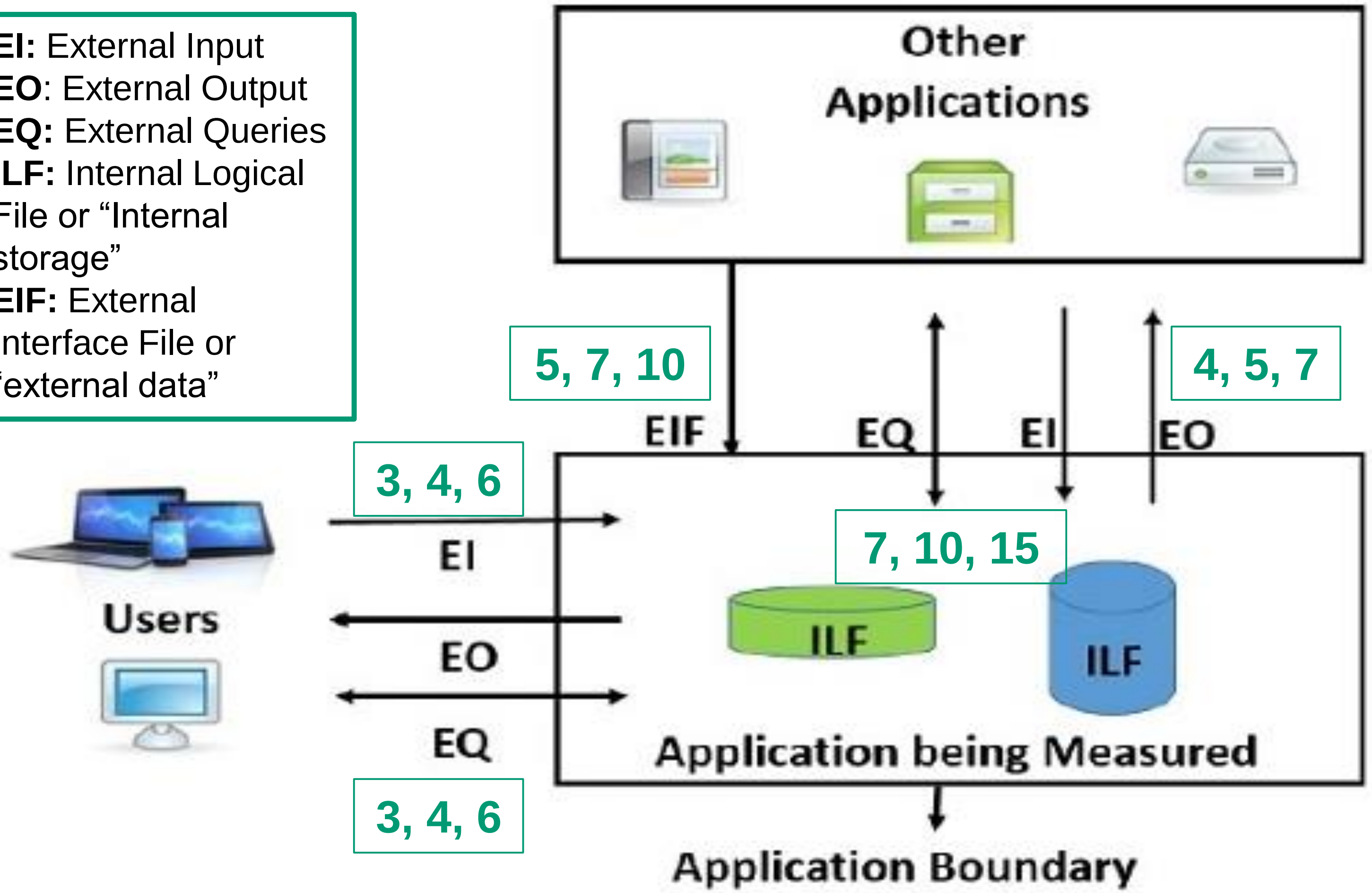
MOTIVATION AND SIZE METHODOLOGIES EXPLORED

Software Size Metrics

Source Lines of Code (SLOC)	Function Points	Agile Metrics (Story Points, T-shirt sizes)
Pros - Objective - Easy to calculate at completion	Pros - Objective - Easier to calculate early in lifecycle	Pros - Easy to calculate early in lifecycle
Cons - Difficult to estimate - Agile programs moving away from SLOC	Cons - Tedious to calculate - Difficult to get actual sizes at project completion	Cons - Highly subjective - Team-dependent

IFPUG Function Points (FPs)

- **EI:** External Input
- **EO**: External Output
- **EQ:** External Queries
- **ILF:** Internal Logical File or "Internal storage"
- **EIF:** External Interface File or "external data"



Issues with FPs

Tedious - Start	Granularity	Tedious - End
Calculating FPs requires: • Identifying all functional transactions • Determining correct complexity levels for each	Transactions are limited to low, average, and high complexities. • Very Low and Low get same sizes • Very High and High get same sizes	• Requirements, architecture, etc. documentation don’t match implemented solution. • Getting actual sizes requires updating doc’s
Solution: Simple Function Points	**Solution: COSMIC Function Points**	**Solution: Objective Function Points**

Effective Sizing

Standard Sizing

- IFPUG and COSMIC have methods to size enhancements: sizes of the changed functional processes
  - Does not account for amount of change required (% redesign, recode, retest)
  - Does not make the modified size equivalent to new development size

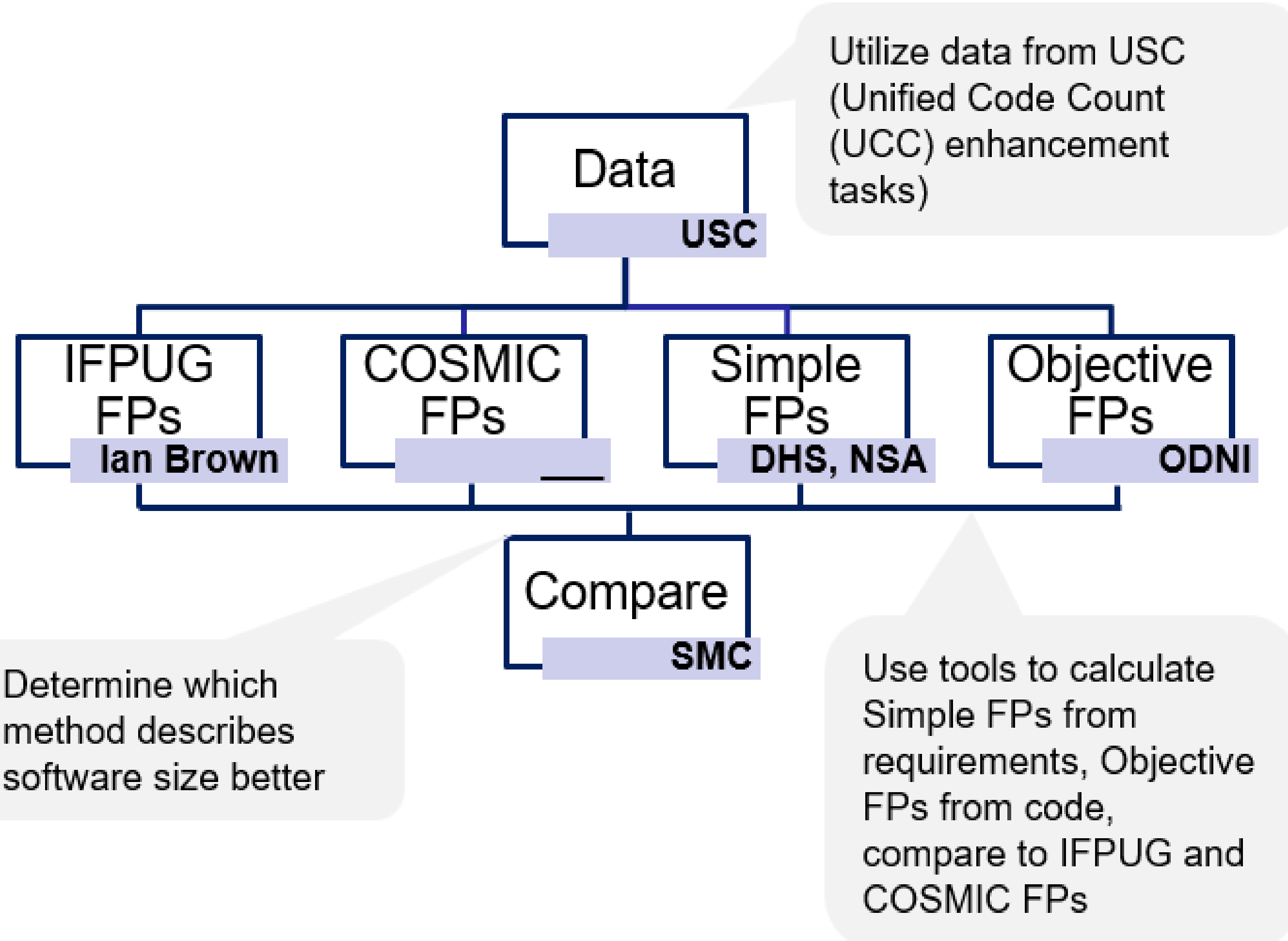
Effective Sizing

- Multiply FPs with weighted average of rework %'s
- Weights:

	Cadence /NSA	Ian Brown
% requirements	10%	
% redesign	30%	40%
% recode	30%	25%
% retest	30%	35%

RESEARCH METHODOLOGY AND DATASET

Research Methodology



Dataset – Unified Code Count (UCC)

Overview

- Enhancement projects
- Code metrics tool
- Command line program
- Implemented in C++, Java
- Each project by new team
- 32 data points

Groupings

- Enhancement Type
  - Add new features/modules (9)
  - Modify existing features/modules (23)
- Complexity Levels
  - Low/Average: Language Parsers, Differencing (12)
  - Very Low: Additional Metric, Input/Output (20)

Calculated Sizes

COSMIC	IFPUG	Simple		Objective
Actual Functionality	Requirements based on Actual Functionality			Actual Code
Anandi + Colleagues	Ian Brown (SME)	DHS	LogApps/NSA	ODNI*
Manual Process		Excel	Cadence	UCC-G

CFPs_AH	FPs_AH	FPs_IB	EFPs_IB	SFPs_DHS	SFPs_Cad	ESFPs_Cad	EOFPs	EOMPs

Several size metrics due to different inputs and methods or perspectives.

Sample Datapoint:

	CFPs_AH	FPs_AH	FPs_IB	EFPs_IB	SFPs_DHS	SFPs_Cad	ESFPs_Cad	EOFPs	EOMPs
Makefile Parser	5	4	12	2.49	20.8	16.2	4.5	28.44	4.88

* NRO provides Configuration Management (CM) for UCC-G. NGA has been backing the development of the Objective method by granting access to run UCC-G on a large SW effort which provided calibration opportunities.

What We're Comparing

CFPs_AH	FPs_AH	FPs_IB	EFPs_IB	SFPs_ DHS	SFPs_ Cad	ESFPs_ Cad	EOFPs	EOMPs

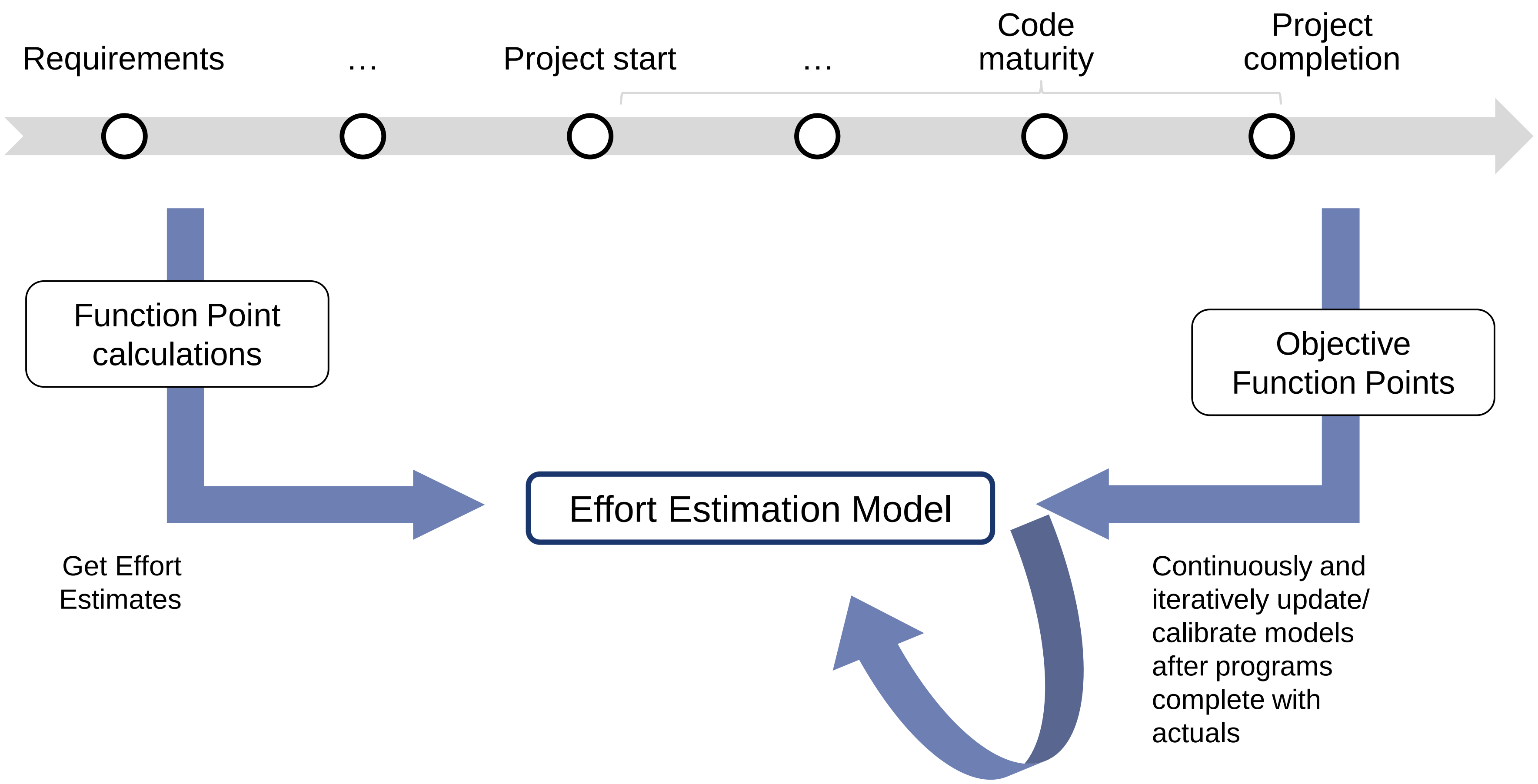
Compare to Effort

- How well do these functional size metrics correlate with effort (and therefore cost)?
- Does the loss/increase in detail used to calculate size hurt/improve effort estimates?
- Which of these methods is better/more accurate for effort estimation?
- If any, what are the drawbacks to using functional size metrics for effort estimation?

Compare to Actual Effective Sizes

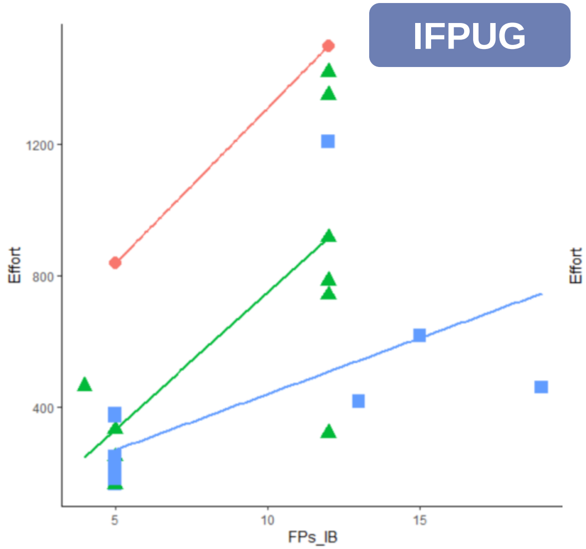
- Use actual reuse %'s for CFPs_AH, FPs_AH, FPs_IB, and ESFPs_Cad
- How well does this methodology predict actual, effective functional size?

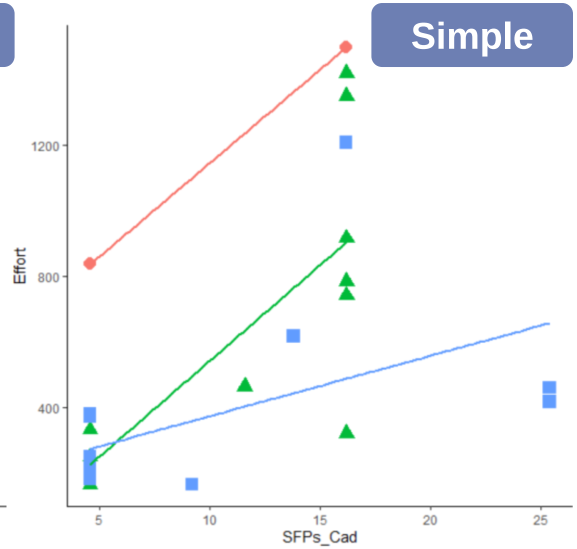
SW Estimation Life Cycle

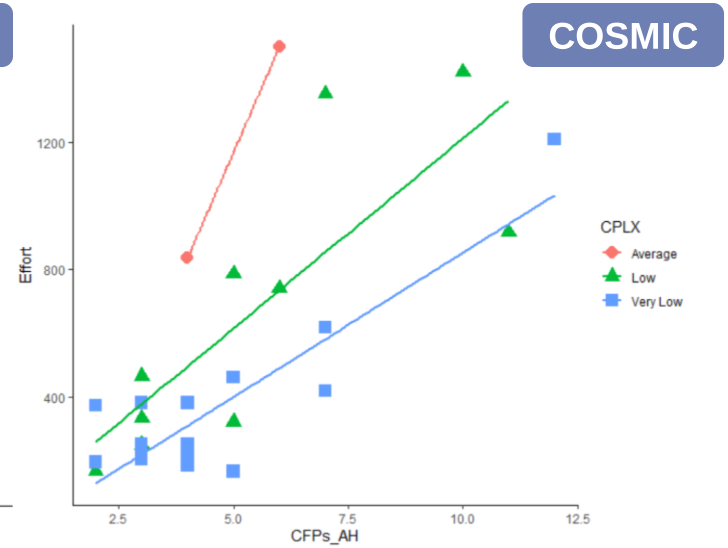


RESULTS

FPs Variants against Effort





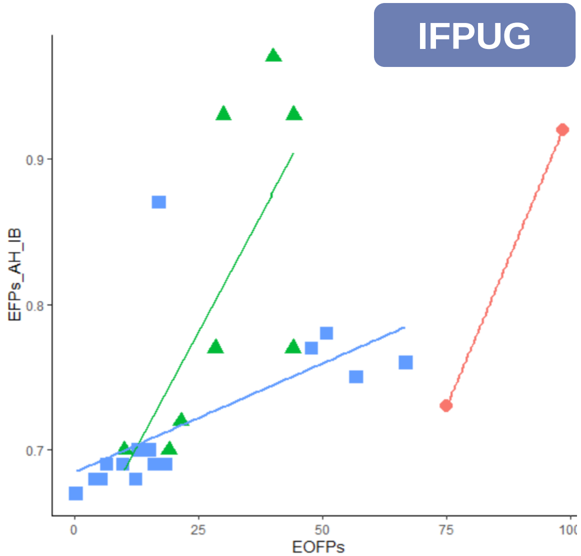


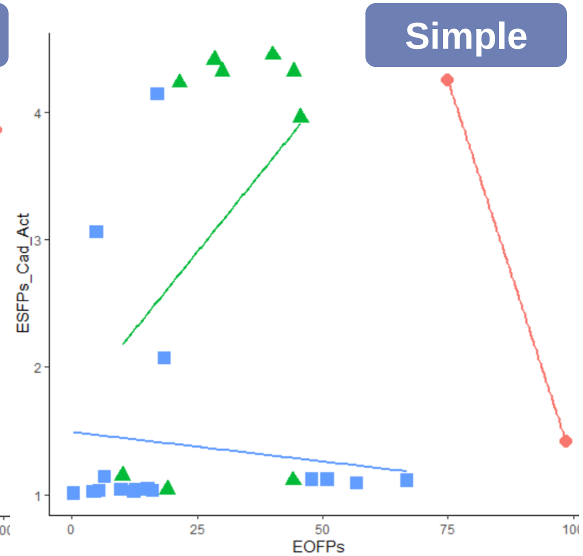
- Sizes stacked with large variance in effort
  - Outputs are of same size
  - Complexity and number of algorithms differ
- **Takeaway:** lack of distribution and accounting for algorithmic complexity → low correlation

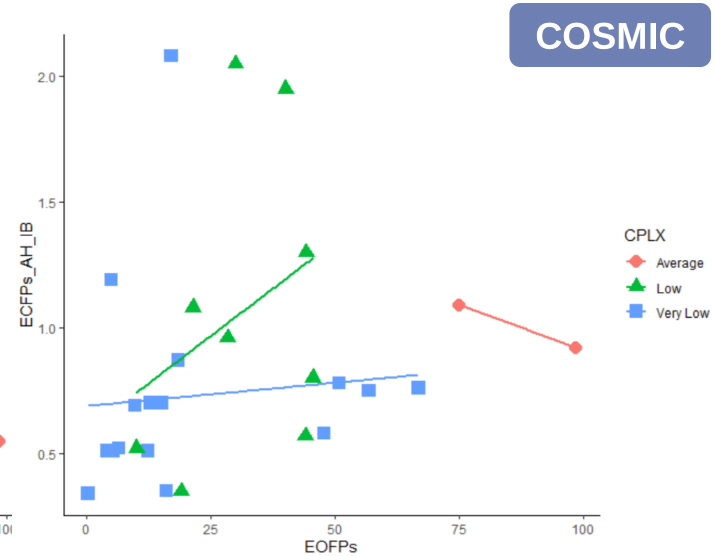
- Reduced granularity compared to IFPUG FPs caused insignificant reduction in correlations
- **Takeaway:** lack of distribution and accounting for algorithmic complexity → low correlation

- Stronger positive trend between size and effort due to higher distribution
- **Takeaway:**
  - better correlation (except for Low CPLX)
  - fewer outliers/anchor points

Objective FPs against FP Variants







- Removed 5 outliers (new code, input functionality), and Average complexity projects (only 2)
- Standard % Error: 6-15%
- **Takeaway:** Promising. Not enough data for types represented in outliers

- Lack of correlation even after removal of outliers
- Not surprising – not using similar counting methodologies
- **Takeaway:** lack of correlation due to difference in methodologies

- Lack of correlation even after removal of outliers
- Not surprising – not using similar counting methodologies
- **Takeaway:** lack of correlation due to difference in methodologies

CONCLUSIONS

Using Function Points for Effort Estimation

1. Useful? **Yes**, but reduced granularity and algorithmic complexity are problematic
   - Grouping by project/ complexity type helps
2. Simple Function Points – does the loss in granularity reduce effectiveness? **No**, not in this case
3. COSMIC Function Points – does increase in granularity increase effectiveness? **Yes**, except for the Low complexity group
4. Which is the best method?
   - **COSMIC** has the highest level of granularity
   - Automated counting from requirements for **Simple** Function Points simplifies estimation process

Using Objective Function Points for Actual Size

- Can the Objective Function Points method estimate actual functional size?
  - Group by complexity levels, and remove projects not reusing code or creating/modifying input options ← may need more exploration
  - Standard % Error for IFPUG between 6-15%
  - Lack of trend for Simple and COSMIC
  - Could be due to UCC atypical for Function Points
- Demonstrated the technique that would be used across a more general sample or within an organization
- Objective Function Points methodology still in development phase
  - Improve through exposure of different software types

Future Research

- Using Function Points methodologies for Effort Estimation
  - Continue comparing estimation effectiveness across larger, varied datasets
- Objective Function Points methodology
  - Continue calibrating the method with larger and varied software products (currently working with NGA)
  - Come up with general conversions from OFPs to FPs

Acknowledgments

Portions of this work were funded by the Air Force Space and Missile Systems Center (SMC), contract FA8802-16-F-0002, and the Office of the Director of National Intelligence (ODNI) RC&E, through the CARRS contract. The authors thank Ms Adriana Contreras, Mr. Raj Palejwala, Mr. Jim Fiume, and Ms Michal Bohn for their support of this effort.