

# A Software Sizing Toolkit: Choosing the right Sizing Approach(es) for your Program

Carol Dekkers, President, Quality Plus Technologies, Inc.





# Agenda

- 1. Introduction to software sizing**
  - 2. Sizing methods and units of measure**
  - 3. Factors in choosing the right sizing approach**
  - 4. Conclusion / recommendations**
-

# Highlights - Carol Dekkers, PMP, CFPS (Fellow), P.Eng. CSM



---

Proud moments: ICEAA 2022 Educator of the year & Lead Author of CEBOK-S, 2023 Global Leader in Consulting, IFPUG Honorary Fellow

---

U.S. expert and project editor for ISO/IEC JTC1 SC7 SW Engineering standards. IFPUG Past President & current Industry Standards Committee Chair

---

Published author, speaker (30+ countries), consultant

---

Mother of 2, YaYa of 1, event & volunteer coordinator, FL resident, and a passion for tennis, travel, craft beverages and gourmet food

---

Shout out to Dan French, COBEC Consulting Inc. my co-presenter at ICEAA 2024



# 1. Introduction to software sizing

---

# Background



In the early days of software, there was only one language and one platform, estimating projects and size was rudimentary (**Source Lines of Code (SLOC)**)



As new languages and platforms were developed, sizing and estimating software projects became more complex (**SLOC challenges**)

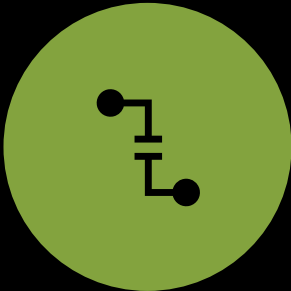


In the 1970's IBM was experiencing significant cost & schedule overruns of their SLOC-based estimates → developed Function Points (public release in 1979 (**International Function Point Users Group formed in 1984 & FPA v1.0 in 1986**))

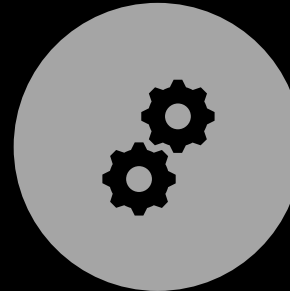


Today: > 700 software languages, diverse set of software platforms, many sizing options → further complicates software sizing and estimating (**Story Points, Simple FP, T-Shirt sizing, Use Case Points, ESLOC, RICEFW...**)

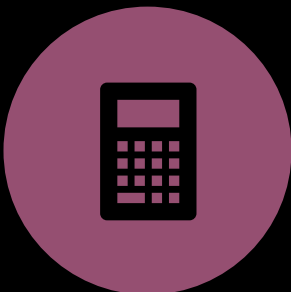
# Importance of Software Size



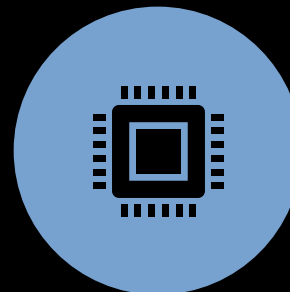
The sizing of software, regardless of method used, is critical to developing reasonable, defensible, and repeatable software estimates.



Size measures (estimates) the scope of the software project and is based on the functional requirements



Depending on the sizing measure used, up to 80% of a software project's effort is related to its size



Software sizing serves as the basis for any software estimate regardless of the technique used



## 2. Sizing methods and units of measure

# Summary: Software Sizing Toolkit

	Unit of Measure (Tool)	Rule-based?	Pros	Cons	Earliest phase for use
1	SLOC	Yes	Easy to count and quick to estimate, code counters will provide accurate installed count. Universal code count rules (Boehm CSSC). Availability of historical data, public CERs (COCOMO II), and commercial tool datasets.	Dependent upon language, platform and developer skill. Paradoxical relationship with productivity.	Budget, Implementation
2	IFPUG Function Points (indicative and traditional)	Yes	Rule-based, ISO std, language, platform and developer skill independent, can compare metrics across organization and industry. Availability of historical data (ISBSG), commercial tool datasets, CER (COCOMO II and III)	Labor intensive. Requires training, relies on specifics of functional requirements.	Design
3	IFPUG Simple FP (SFP)	Yes	Rule-based, only 2 functions, language, platform and developer skill independent, can compare metrics across organization and industry. Faster to count, doesn't require extensive training. Doesn't require detailed requirements.	Faster than standard IFPUG FP, but still takes some time. Requires some training (FUR)	ConOps (ROM), EPICS
4	COSMIC FP Wide Bands	Yes	Rule-based, ISO std, language, platform and developer skill independent, can compare metrics across organization and industry, measures internal functions	Requires development knowledge and details of functional requirements. Requires training.	ConOps (ROM), EPICS
5	Use Case Points	Yes	Rule-based, repeatable & verifiable	No governing body, restricted to use with UML & RUP, little industry data to compare metrics	ConOps (ROM), EPICS
6	Story Points	No	Quick estimating for Agile projects	Cannot be used with other methodologies, subjective, cannot be verified or audited, not Rule-based	EPICS or user stories (backlog)
7	T-Shirt Sizing	No	Quick estimating for Agile projects	Cannot be used with other methodologies, subjective, cannot be verified or audited, not Rule-based	EPICS or user stories (backlog)
8	RICE(FW) / Object Sizing	Yes	Useful for ERP (COTS-based) implementations. Based on SAP ABAP objects, but today is applied variously to other ERP-based programs. Some CERs developed.	Lack of consistency(beyond definitions) for RICE(FW) objects. Not convertible to other measures	Requirements / design
9	SNAP (Software Non-functional Sizing)	Yes	ISO standard. Useful to delineate / account for non-functional complexities	Lack of historical data (growing slowly). Requires training and design knowledge	Design



# Source Lines of Code (SLOC)

- SLOC is the original software size measure and still in use today
- Platform and language dependent
- Represents the physical number of lines of code written or expected to be written for the project
- Can also be expressed as eSLOC (effective SLOC) or KSLOC (thousands of Line of Code)
- No rules for counting nor is there a standard or organization to manage rule set
- Still commonly used because it is “easy”
- When estimating, it is a guesstimate of expected SLOC
- Can be easily counted once written with a code counter
- Hard to verify or repeat
- SLOC based metrics can be unreliable
- Influenced by developer skills and coding style

# International Function Point Users Group (IFPUG) Function Points

- First rules-based software sizing method (Counting Practice Manual v. 4.3.1)
- Sizes Functional User Requirements (FUR)
- First International Standards Organization (ISO) functional size measure ISO-IEC 20926:1998
- 3 transactional functions, EI, EO, EQ & 2 data functions, ILF & EIF
- Platform, language, development method and developer skill level independent
- Can be audited and replicated
- Highly defensible
- Industry certifications: CFPS and CFPP
- Does not measure non-functional requirements

# International Function Point Users Group (IFPUG) Simple Function Points (SFP)

- Based on IFPUG standard function points
- Developed in 2010 by Dr. Robert Meli of Italy.
- Measures Functional User Requirements (FUR)
- Elementary Process (EP) replaces transactional functions (EI, EO, EQ)
- Logical File (LF) replaces data functions (ILF, EIF)
- Platform, language, development method and developer skill level independent
- Can be audited and replicated
- Highly defensible
- Created to increase speed to count and simplify counting rules for inexperienced counters
- Industry certification being developed by IFPUG Certification Committee
- Does not measure non-functional requirements

# COmmon Software Measurement International Consortium (COSMIC) Function Points

- Evolved by combining Full FP (Quebec Canada) and Mark II FP (UK) methods, as an “academically founded” new method at the University of Quebec (Montreal)
- Rule based COSMIC Measurement Manual
- ISO Standard 19761
- Language, technology, and development method independent
- Counts functional requirements based on movement of data → Entry, Exit, Read, Write functions each count as 1 CFP
- Defensible, repeatable and auditable
- COSMIC Foundation Level and Early & Quick Size Estimating certifications

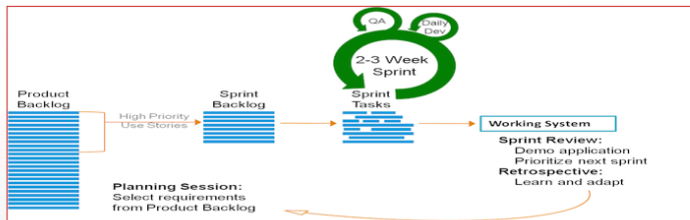
# Use Case Points (UCP)

- Developed by Gustav Kemer in 1993
- Used only when Unified Modeling Language (UML)/Extensible Markup Language (XML) & Rational Unified Process (RUP) object-oriented projects based on use cases
- Unadjusted Use Case Weight (UUCW) – the point size of the software that accounts for the number and complexity of use cases
- Unadjusted Actor Weight (UAW) – the point size of the software that accounts for the number and complexity of actors
- Technical Complexity Factor (TCF) – factor that is used to adjust the size based on technical considerations
- Environmental Complexity Factor (ECF) – factor that is used to adjust the size based on environmental considerations
- $UCP = (UUCW + UAW) \times TCF \times ECF$
- Rule based but there is no governing organization
- Never been calibrated via regression analysis and does not take into account diseconomies of scale

# Story Points (SP)

## B.3 Relative effort (1 of 7)

- Story points (SP) are the unit of measure used in Scrum and other agile methods to quantify the relative overall effort required to fully implement a “user story” or any other piece of work.<sup>1</sup>
- Relative effort size is used by agile scrum-development teams to estimate how much work (software requirements) can fit into a fixed-duration sprint



CEBoK-S

62

Adapted from [www.mountaingoatsoftware.com/agile/user-stories](http://www.mountaingoatsoftware.com/agile/user-stories)

## B.3 Relative effort (2 of 7)

### User stories:

- Are short, simple descriptions of a software feature, written from the perspective of the person who desires the new capability, usually a user or customer of the system
- Typically of the form “As a < type of user >, I want < to achieve some goal > so that < some reason >”
  - Example: as a **business traveler**, I want to make a **hotel reservation** so that I have a **place to stay**
- May include functional, non-functional or technical software requirements

- There are 3 primary relative effort sizing methods used to derive story points:
  1. planning poker, 2. t-shirt sizing, 3. time buckets

CEBoK-S

63

1. [www.mountaingoatsoftware.com/agile/user-stories](http://www.mountaingoatsoftware.com/agile/user-stories)

More information on all of the Software Sizing Methods can be found in  
**CEBoK-S: Lesson X Software Size**

# Story Points (SP)<sub>1/2</sub>

- “Relative effort” unit of measure used by Agile software development teams when assigning user stories during “Sprint planning”
- Based on user stories
- Subjective, relative effort unit of measure
- Each user story assigned a number of SP based on consensus, not rules
- Uses Fibonacci sequence instead of linear ranking/ratio to estimate
- Cannot be verified, audited and hard to defend
- Cannot be used for software metrics
- Cannot be compared to other teams or industry metrics
- Used to calculate a team’s velocity (SP per sprint)
- Quick and easy to estimate

# T-Shirt Sizing

- “Relative effort” unit of measure used by Agile software development teams when assigning user stories during “Sprint planning”
- Precursor to story points
- Used for release planning, early estimating & product backlog sizing
- Requirements are assigned a size XS, S, M, L, XL
- Subjective, relative effort unit of measure
- Estimates the number of sprints it should take to complete an Epic, 1 for XS to 7+ for XL
- If XL, may want to consider breaking down the Epic
- Cannot be verified, audited and hard to defend
- Cannot be used for software metrics
- Cannot be compared to other teams or industry metrics
- Used to calibrate a team’s velocity
- Quick and easy to estimate

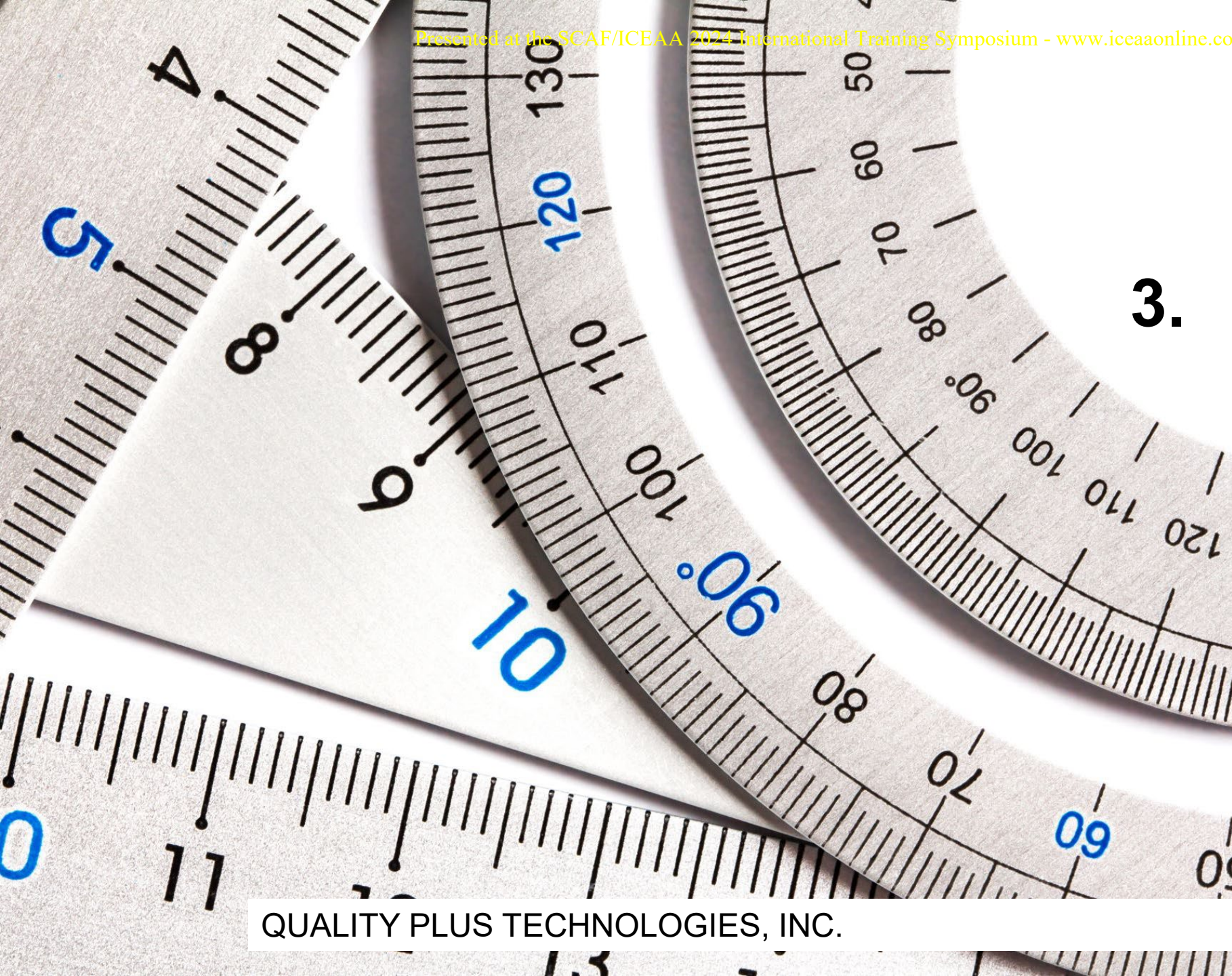


# Reports, Interfaces, Conversions, Extensions, Forms, Workflows (RICEFW)

- Often used for sizing Enterprise Resource Planning systems such as Oracle or SAP
- Physically counts the number of each item to estimate the software size
- Easy to count and verify
- Best used for COTS acquisitions
- May not have all the requirements available to identify each specific component
- Can be used to evaluate cost per RICEFW component when comparing multiple COTS solutions

# Software Non- Functional Assessment Process (SNAP)

- Introduced by IFPUG in 2009 to address to measure non-functional requirements that IFPUG FP does not size
- IEEE 2430-2019 standard Software Non-Functional Sizing Measurements
- Rules based, repeatable, defensible and auditable
- Adoption is slow but growing
- Parametric estimating tools do not currently support
- 4 categories with 14 sub-categories:
  - Data Operations
  - Interface Design
  - Technical Environment
  - Architecture

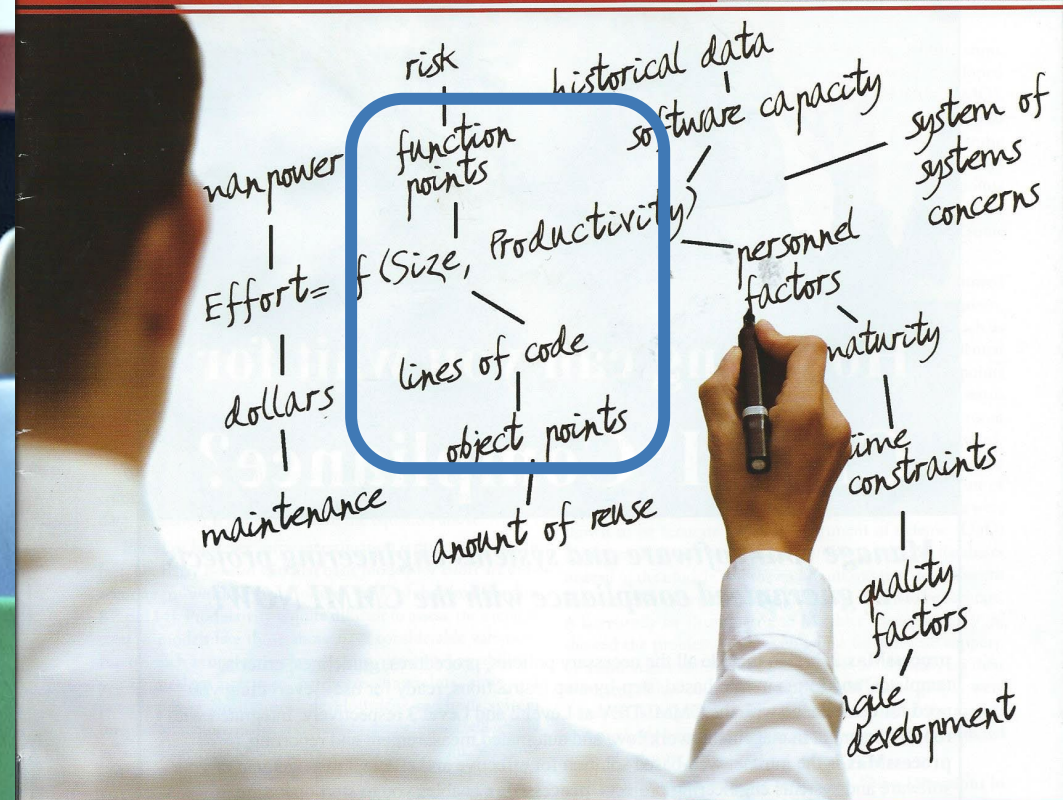


**3. Factors in choosing the right best sizing approach**

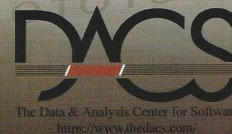
---

# Size is a main driver in the software estimation process

(See CEBok-S)



## New Directions in Software Estimation



The Data & Analysis Center for Software  
<https://www.thedacs.com/>

<http://iac.dtic.mil/dacs>

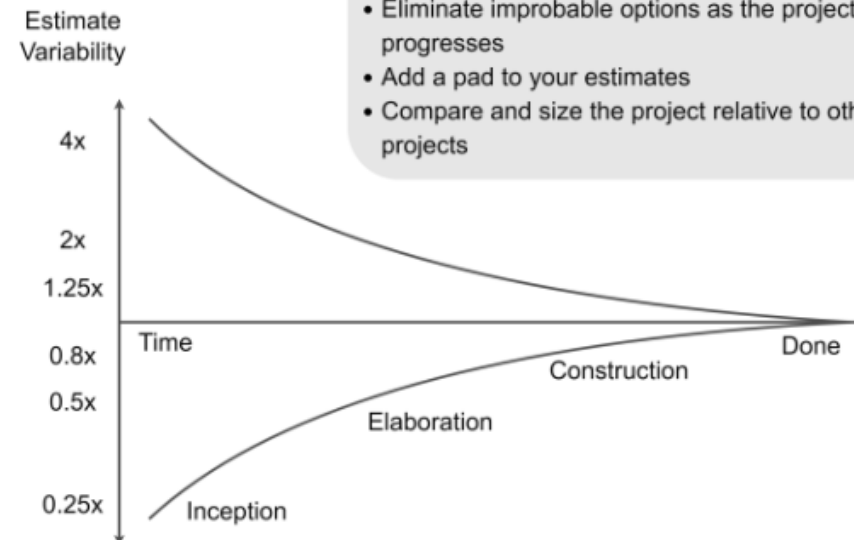
Unclassified and Unlimited Distribution

# CONE OF UNCERTAINTY

At the start of the project, there are a lot of variables that need to be taken into account. All projects have different requirements, different priorities, different people working, or different technologies. These are the reasons why the variables coming into the project are not clear at the beginning of a project. The variability of these factors is reflected in the project variability. The result is that estimates must include the variability. After the project starts and more is known, the variability is decreasing. This phenomenon is called the "Cone of Uncertainty". A significant narrowing of the Cone occurs during the first 20-30%.

## ESTIMATION TIPS

- Make a visible list of risks
- Inspect and update risks and estimates regularly
- Eliminate improbable options as the project progresses
- Add a pad to your estimates
- Compare and size the project relative to other projects



(Software estimation)  
cone of uncertainty

# 3. Factors in choosing the best sizing approach



## B. Understanding software size (2 of 2)

- The best sizing method depends on available scope information, historical data, and the chosen estimating technique(s)
- Ensure that the size is complete by:
  - Documenting exclusions (e.g., does not include ABC components which are outsourced);
  - Documenting any "To-be-determined" areas (e.g., reports were not yet identified)
  - Asking questions about the units of measure and their meaning
- Normalization of data is important for consistency and comparability – regardless of sizing method(s) used
- Always do cross-checks on the sizing estimate

CEBoK-S

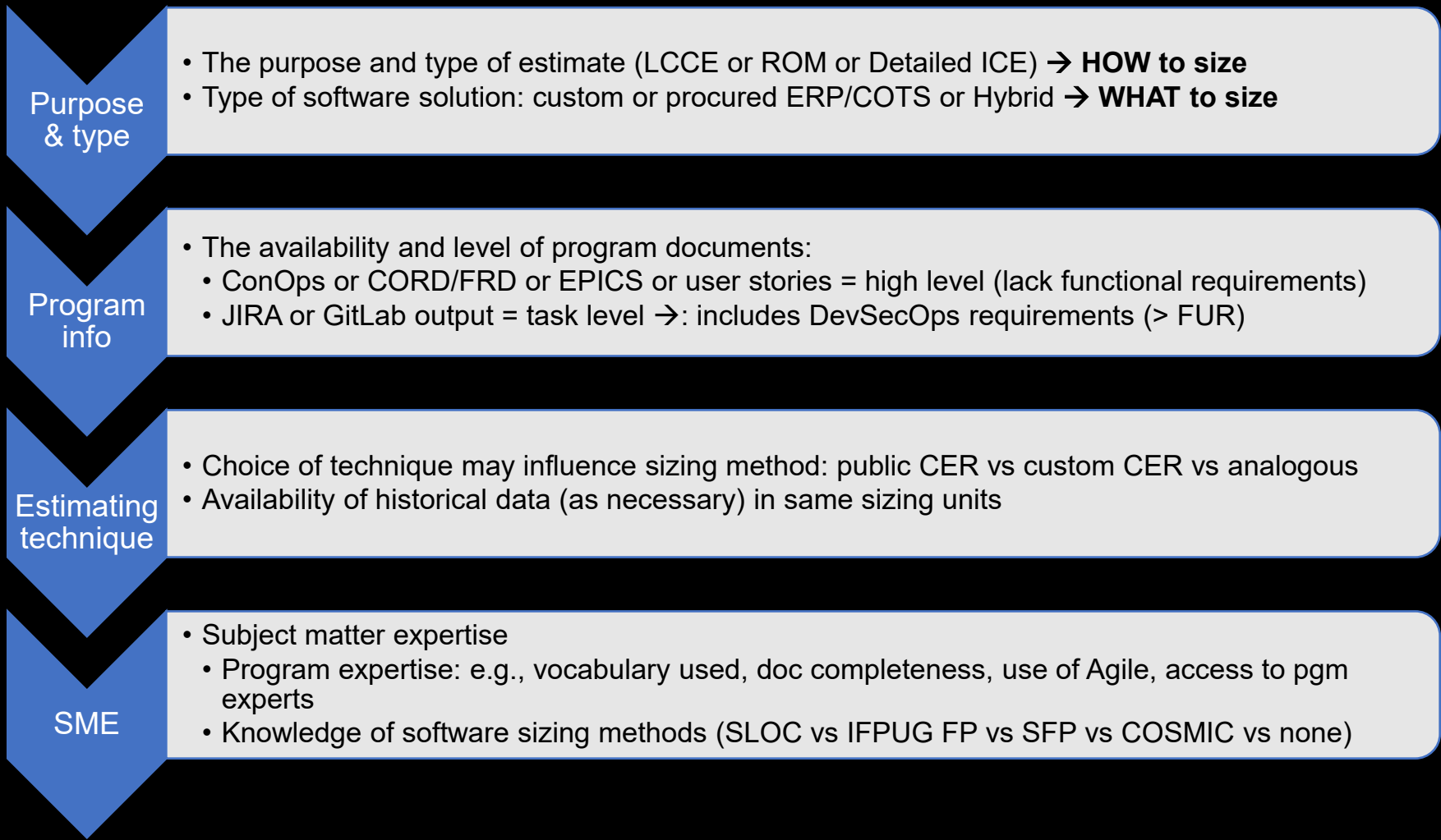
8

*CEBoK-S Lesson X: Software Size*

Factors to selecting the BEST sizing method (and measure) involve the purpose and type of estimate; availability of program documents; chosen estimating technique(s); historical data; software development paradigm (waterfall/agile/hybrid); analyst knowledge level; ...



# 3. Factors in choosing the best sizing approach



# 3. The best sizing approach? It depends...

Estimating Scenario	SLOC	IFPUG FP (traditional)	IFPUG SFP	COSMIC Wide Bands	Use Case Points	Story Points	T-shirt sizing	RICE (FW)	SNAP	DHS SiSE (SFP)
• ROM estimate based on ConOps			X	X						X
• ROM estimate based on Agile Roadmap			X	X						X
• Engineering build up based on past delivery	X	X	X		X			X		
• LCCE after several releases	X	X	X		X			X		
• 5 year Software Sustainment Estimate	X	X			X				X (contrib)	
• Analogous estimate for first release based on FRD	X	X	X	X	X			X		X
• MVP release estimate based on product backlog			X	X		X (sched only)	X (sched only)	X		X
• Sprint planning estimate						X	X			



# Additional sizing considerations:

## 1. CROSS CHECKS

- Use a 2<sup>nd</sup> software sizing method (and/or commercial tools, Capers Jones' Rules of Thumb,) as a size sanity check;
- Use a 2<sup>nd</sup> software estimation method (commercial tool, public/custom CER, analogy) ;
- Consider uncertainties related to function ambiguity/incompleteness with early requirements documents

## 2. SIZE UNCERTAINTY – Growth



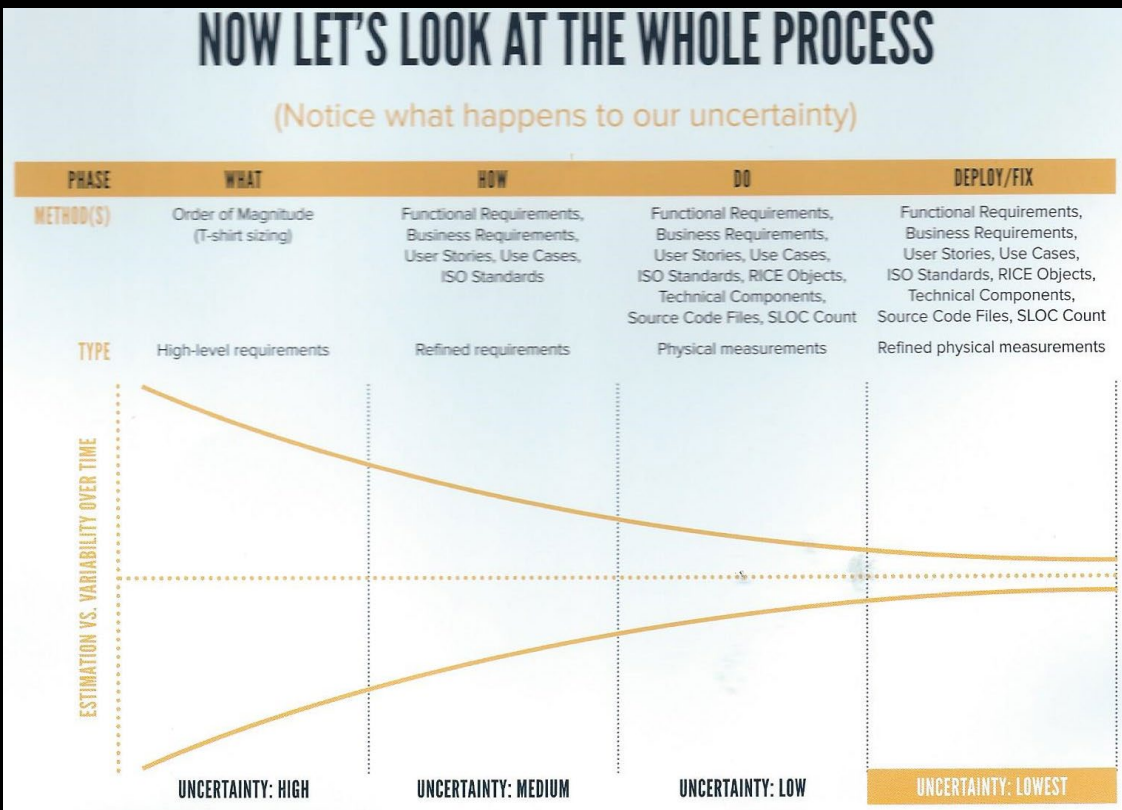
Step 4:  
Conduct  
Sensitivity,  
Risk, and  
Uncertainty  
Analysis

### Driver Uncertainty: Size Considerations of Size Growth

- Any credible software cost estimate should consider, and properly account for, anticipated growth (SLOC or Functional Size)
- Size is a major cost driver for software estimates, but until the project is completed, size itself is uncertain
- Size growth (from the original estimate) includes several types:
  1. **Code growth and reuse optimism** → driven in part by the view that a greater percentage of code (than is warranted by history) can be reused
  2. **Functionality growth** → driven in part by the requirements uncertainty and volatility
  3. **Agile scope growth** → driven by high-level requirements analysis and stakeholder priorities during development

*CEBoK-S Lesson 3: 4 Step SW Estimating Process*

# 4. Conclusion / recommendations



Excerpted from QSM's Software Sizing Infographic (circa 2018)

- There is no single BEST sizing approach for all programs (or all estimates)
- The BEST sizing approach for YOUR program depends...
- Use a second sizing approach /estimation method as a cross-check
- Don't forget about software growth!
- Software size is THE most important (and often overlooked) driver of software development cost

# A Software Sizing Toolkit: Choosing the right Sizing Approach(es) for your Program

Carol Dekkers, President, Quality Plus Technologies, Inc.

[Dekkers@qualityplustech.com](mailto:Dekkers@qualityplustech.com)

