QSM® The Intelligence behind Successful Software Projects

SOFTWARE SIZE MATTERS

Size is one of the 5 core metrics behind software estimation – what are the other four?

WHY DO WE CARE ABOUT SOFTWARE SIZE?

Ignorance of size leads to bad estimates.

Without software size, it's hard to estimate:

- How long a software project will take
- How much it will cost
- How many people we'll need
- How many defects we can expect to find during testing
- How productive we are likely to be

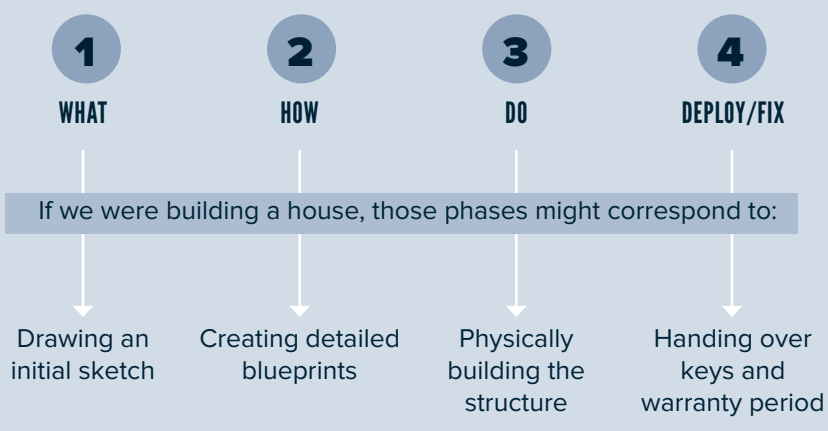
WHY? Because there's a **non-linear** relationship between size and schedule, effort (cost), and defects.

ESTIMATING SIZE IS EASY, RIGHT?

Unfortunately, no.

Estimators need to use different sizing methods, depending on where the project is in its life cycle, and what information is available.

Broadly, we can divide the software lifecycle into four stages:



At each stage, we're able to estimate size in new ways, and with greater clarity – sometimes referred as **"PROGRESSIVE ELABORATION."**

BUT HERE'S THE TRICK:
Since we'll be using different sizing methods together, we'll need a way to normalize or convert to a common measurement unit. This is important because it allows broad comparisons across different technologies, projects, industries and organizations.

For FUNCTIONAL SIZE, we can convert all our measurements into base units called FUNCTION POINTS. (The International Function Point Users Group (IFPUG) method is the most widely used ISO standard for functional sizing.)

For TECHNICAL SIZE, we can convert all our measurements into base units called IMPLEMENTATION UNITS. An IU is equivalent to writing one source line of code or one technical step in configuring a commercial package (COTS) package.

FUNCTION POINTS can be converted to IMPLEMENTATION UNITS using QSM's Function Point Languages Table: **www.qsm.com/function-point-table.**

WHAT'S IN A NAME?
Individual methodologies have their own names for the four basic stages:

METHODOLOGY	WHAT	HOW	DO	DEPLOY/FIX
WATERFALL	Concept	Rqmts. & Design	Construct & Test	Deploy
RUP	Initiation	Elaboration	Construction	Transition
AGILE	Initiation	Iteration Planning	Iteration Development	Production
SAP ASAP	Project Preparation	Business Blueprint	Realization & Final Prep	Go Live

WHAT DO WE MEAN BY "SIZE" ANYWAY?
There are two main types of software sizing: functional size and technical size.

FUNCTIONAL SIZE is the amount of software functionality delivered to the end user.

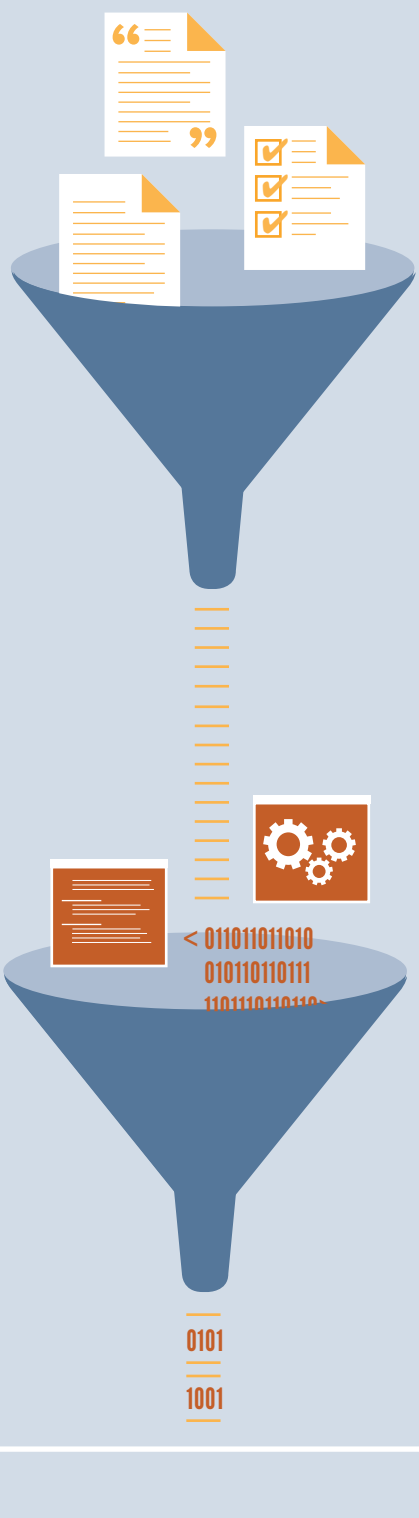
TECHNICAL SIZE is the amount of software logic that creates the functionality.

End users only care about FUNCTIONAL SIZE.

Developers care about TECHNICAL SIZE - how many logical source lines of code or configurations are required.

WHAT ARE THE MOST COMMON SIZING METHODS?

(WHAT) Order of magnitude size (T-shirt sizing)
Ranges (XS to XXL) can scale with sizes of completed projects (industry or corporate).



FUNCTIONAL SIZE (normalized to FUNCTION POINTS, then to IMPLEMENTATION UNITS)

(HOW) (DO) (DEPLOY/FIX)	
FUNCTIONAL REQUIREMENTS (A.K.A. FUNCTIONAL CAPABILITIES)	Testable "shall statements" that describe the intended functions of software; often maintained in a Requirements Traceability Matrix (RTM) spreadsheet or database tool; requires a consistent definition; most often used in traditional software development.
BUSINESS REQUIREMENTS	A higher level of abstraction that comprises multiple functional requirements; can be used for ballpark estimates, assuming a given number of functional requirements per business requirement; most often used in traditional software development.
USER STORIES	Placeholders for future conversations between developers and customers; equates to one scenario (i.e., thread of functionality) in the software that will satisfy a user goal; requires a consistent definition; most often associated with Agile projects.
USE CASES	Containers of multiple scenarios (i.e., user stories) that describe the interactions between a human (or external system) and a software system to achieve a common user goal; can be used for ballpark estimates, but requires a consistent definition; commonly associated with both traditional and Agile projects.
FUNCTION POINTS (ISO STANDARDS: IFPUG, MARK-II, COSMIC, FISMA, NESMA)	More robust but time consuming methods for measuring functional size; can be estimated using shortcuts early, but can only be counted after requirements are established.

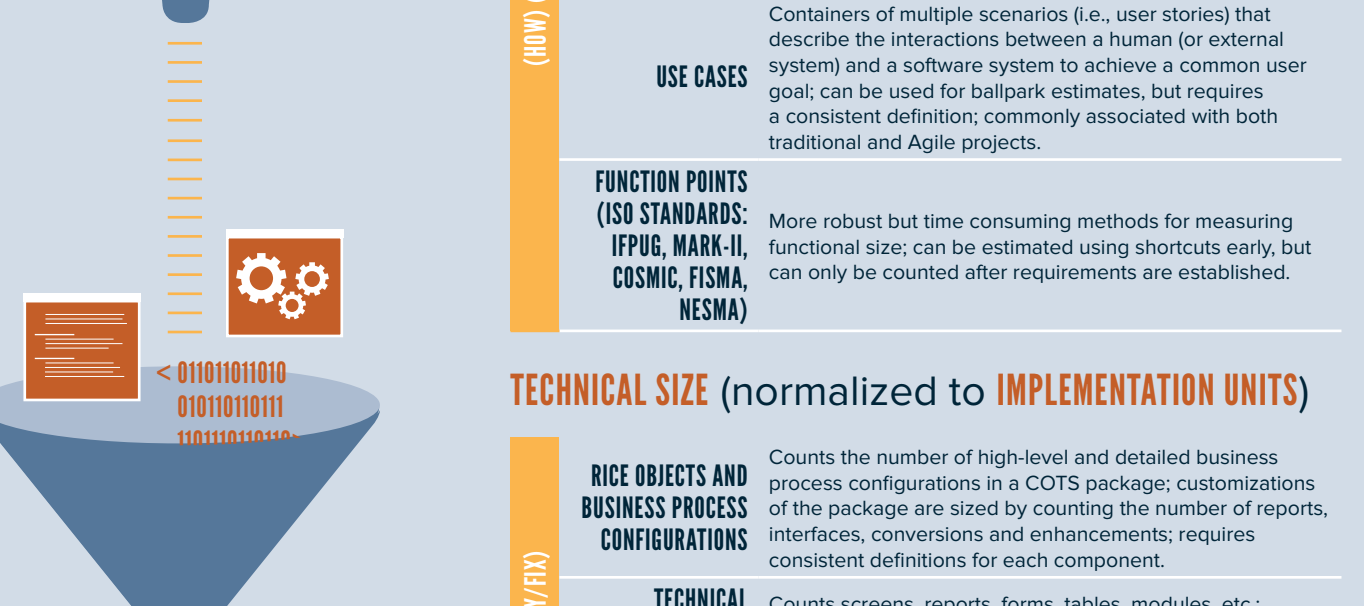
TECHNICAL SIZE (normalized to IMPLEMENTATION UNITS)

(DO) (DEPLOY/FIX)	
RICE OBJECTS AND BUSINESS PROCESS CONFIGURATIONS	Counts the number of high-level and detailed business process configurations in a COTS package; customizations of the package are sized by counting the number of reports, interfaces, conversions and enhancements; requires consistent definitions for each component.
TECHNICAL COMPONENTS	Counts screens, reports, forms, tables, modules, etc.; requires consistent definitions for each component.
SOURCE CODE FILES	A ballpark estimate of the number of Source Lines of Code (SLOC), assuming an average number of SLOC per source code file.
SLOC COUNTS	The number of logical source statements delivered to the customer, specific to the type of project and programming language; only measurable after coding.

NOW LET'S LOOK AT THE WHOLE PROCESS

(Notice what happens to our uncertainty)

PHASE	WHAT	HOW	DO	DEPLOY/FIX
METHOD(S)	Order of Magnitude (T-shirt sizing)	Functional Requirements, Business Requirements, User Stories, Use Cases, ISO Standards	Functional Requirements, Business Requirements, User Stories, Use Cases, ISO Standards, RICE Objects, Technical Components, Source Code Files, SLOC Count	Functional Requirements, Business Requirements, User Stories, Use Cases, ISO Standards, RICE Objects, Technical Components, Source Code Files, SLOC Count
TYPE	High-level requirements	Refined requirements	Physical measurements	Refined physical measurements



FROM THE SIZING AND ESTIMATION EXPERTS AT

QSM®

www.qsm.com

FOR MORE INFORMATION ON SIZING DOWNLOAD THE QSM SOFTWARE ALMANAC

WWW.QSM.COM/2014-ALMANAC