THE COSMIC METHOD OF SOFTWARE SIZING AND ITS USES IN MANAGING AND ESTIMATING SOFTWARE ACTIVITIES

BCS Advanced Programming Group meeting

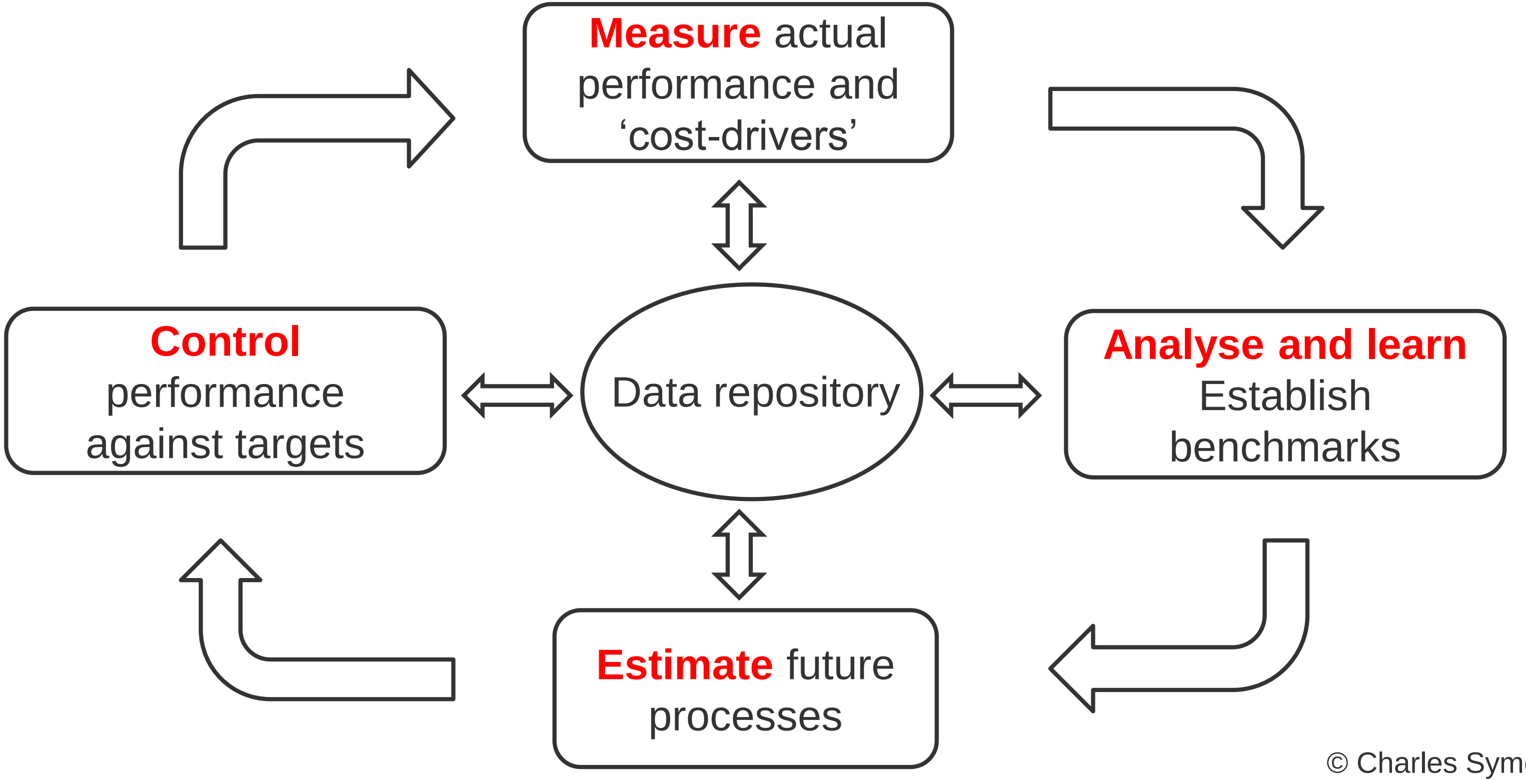
12th April 2018

Charles Symons

Agenda

- Goals: the importance of measuring software size
- Overview of the COSMIC method
- The acid test. Do COSMIC sizes correlate with effort?
- Conclusions and future

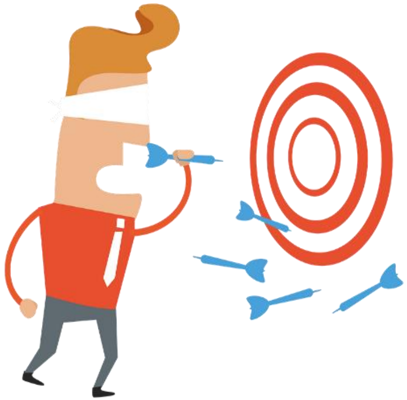
The goal: master the whole cycle of managing software processes



COSMIC

Few organizations really master the control cycle

- High proportions of software project failures and cost over-runs
- Who does best?
  - Commercial software suppliers – a matter of survival
  - Agile method practitioners – (maybe) but only at the team level



- Why the problems? Developing software is partly an <u>unpredictable</u> process

The performance of software processes has various aspects, and they are tradeable

Project achievement vs plan

- Actual vs. estimated: Effort, Duration, Size

Project productivity

- Size / Effort

Project speed

- Size / Duration

Product quality

- Defect density (# Defects/Size)
- Functional (e.g. business needs)
- Technical (e.g. maintainability, response time, etc.)

… and the performance of on-going **maintenance and enhancement** processes

Mastering the control cycle requires a sound method for measuring software size

Sizing method options:

Counts of Source Lines of Code:	X X	Can't estimate until software designed Technology-dependent, no standards
Development method specific, e.g. UCP, OOP, Story Points, etc.	X	No reliable standards; benchmark data possible only locally
Functional size:	✓ ✓ ▪	International standard methods Technology-independent 'First Generation' methods have limitations

Simple example: using the control cycle data to estimate effort for a project, iteration, etc.

Completed projects:

$$\text{Measure productivity} = \frac{\text{Software size}}{\text{Project effort}}$$

(Establish average 'benchmark' productivity for the type of project)

New project:

$$\text{'Typical' estimated effort} = \frac{\text{Estimated software size}}{\text{Benchmark project productivity}}$$

$$\text{'Best' estimated effort} = \left\{ \frac{\text{Estimated software size}}{\text{Benchmark project productivity}} \right\} \times \left\{ \begin{array}{c}\text{Adjustments for} \\ \text{project-specific} \\ \text{'cost-drivers'}\end{array} \right\}$$

A huge number of possible cost-drivers can affect performance

Hardware platform(s)

Programming language(s)

Software re-use

Development methods

SIZE

Staff problem-area experience

Requirements uncertainty

Non-functional requirements

Project-management method

Interfaces, dependencies

New, maintenance, re-development

Risk

Staff technology experience

Budget, time constraints

Staff turnover

Domain (business, real-time)

Industry

Number of implementations

Security

CHANGES

Summary: there are inherent challenges to implement the software control cycle:

- The performance of software processes has multiple, *tradeable* aspects
- There are so many variables, it is *impossible* to build general, statistically-valid estimation models for more than a few of them
- Conclusions:
  - Collect your own size, effort, etc., data
  - Establish your own size/effort relationships

Agenda

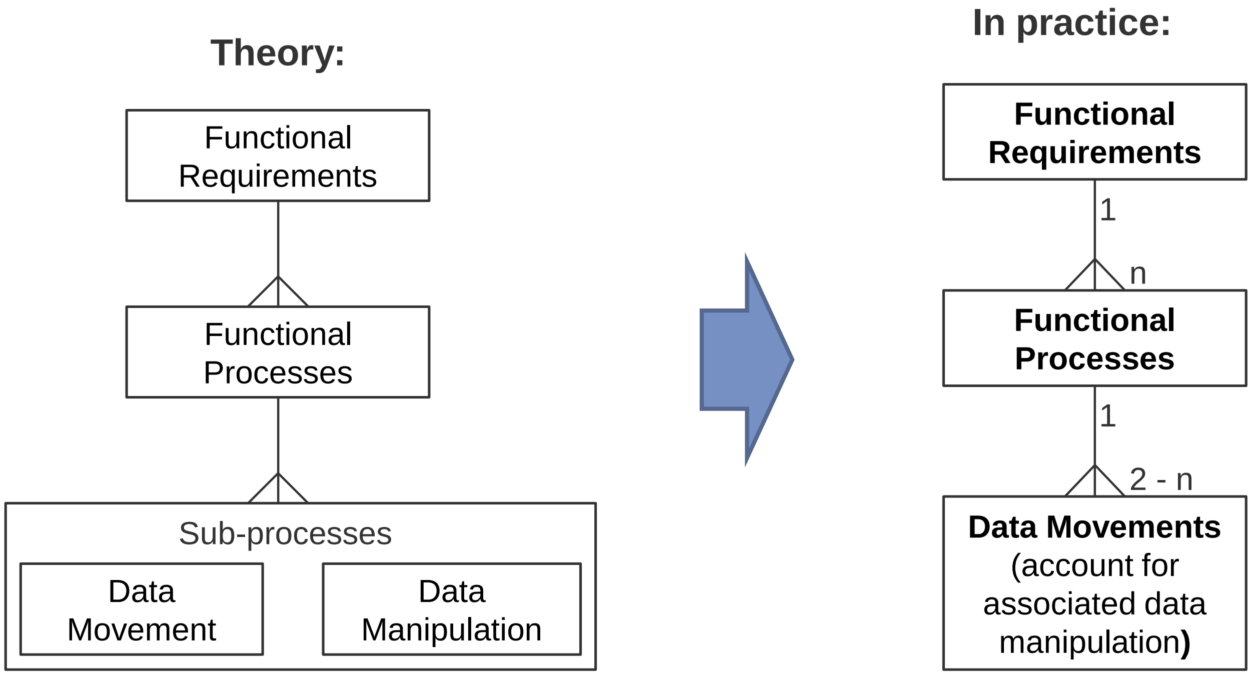
- Goals: the importance of measuring software size
- Overview of the COSMIC method
- The acid test. Do COSMIC sizes correlate with effort?
- Conclusions and future

COSMIC

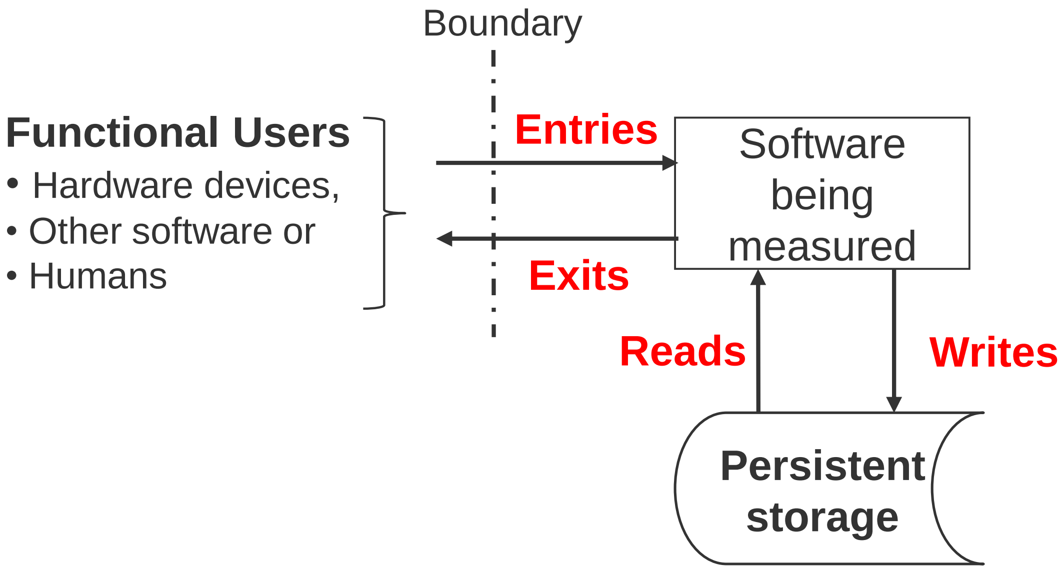
Method Goals

- A measure of functional requirements based on fundamental software engineering principles
- Applicable to business, real-time and infrastructure software
- Independent of technology or processes used for the software or project
- (Hopefully) produces sizes that correlate well with effort
- Open, free

All software functional requirements can be broken down into 'functional processes'

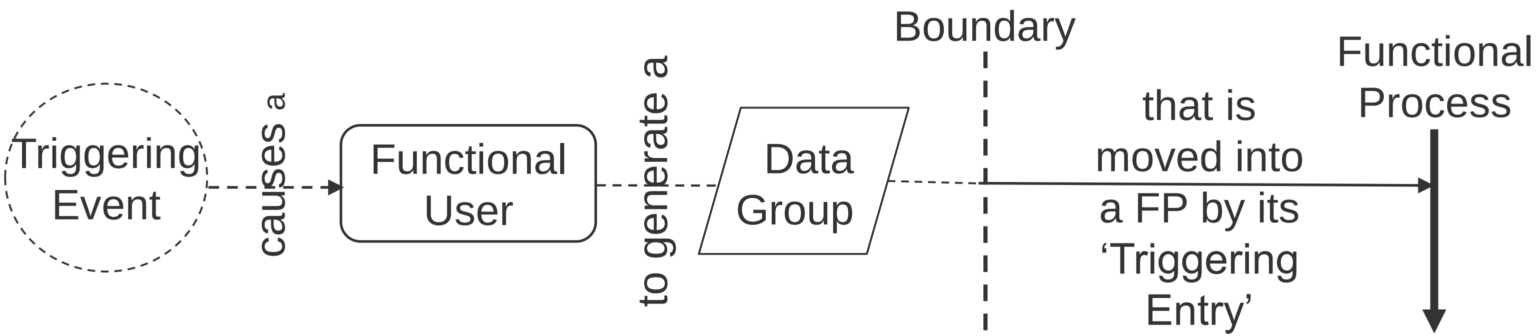


There are four types of 'Data Movement' sub-processes

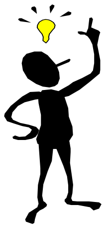


The 'Data Movement' is the unit of measure: 1 CFP (COSMIC Function Point)

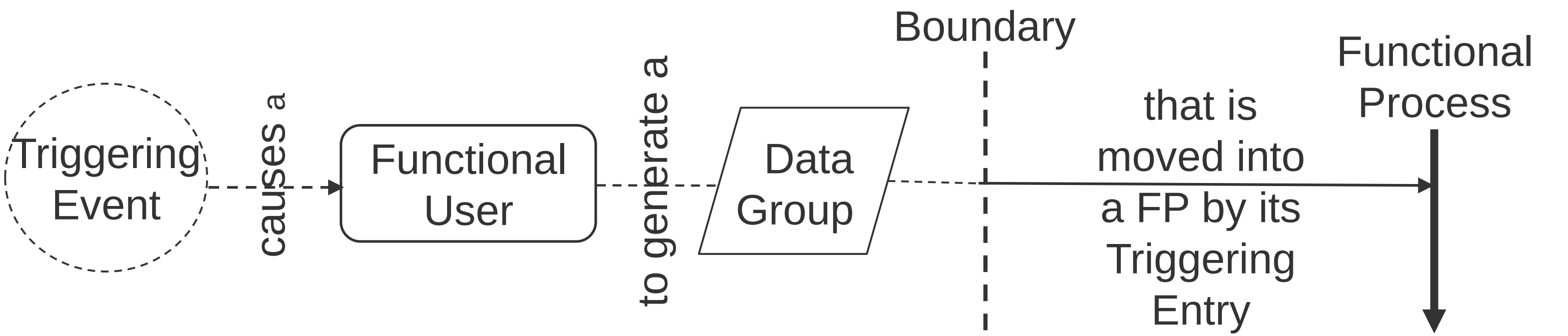
A Functional Process responds to an ‘Event’ that a ‘Functional User’ detects or generates



New employee starts work	Personnel Officer	types employee details	Entry DM with employee details
“I want to enquire”	Personnel Officer	types employee ID/name	Entry DM with employee ID/name



Some real-time examples



End of time interval	Clock	Clock tick	Entry DM of tick (= ‘start processing’)
Missile approaching	Aircraft radar	Message ‘Missile approaching’	Entry DM with radar info

Definition of a Functional Process (abbrev.)

a) A set of data movements … of the functional requirements …. being measured, …. that can be defined independently of any other functional process in those requirements.

b) … Each functional process starts processing on receipt of a data group moved by its Triggering Entry data movement.

c) The set of all data movements of a functional process is the set that is needed to meet its requirements for all the possible responses to its Triggering Entry.

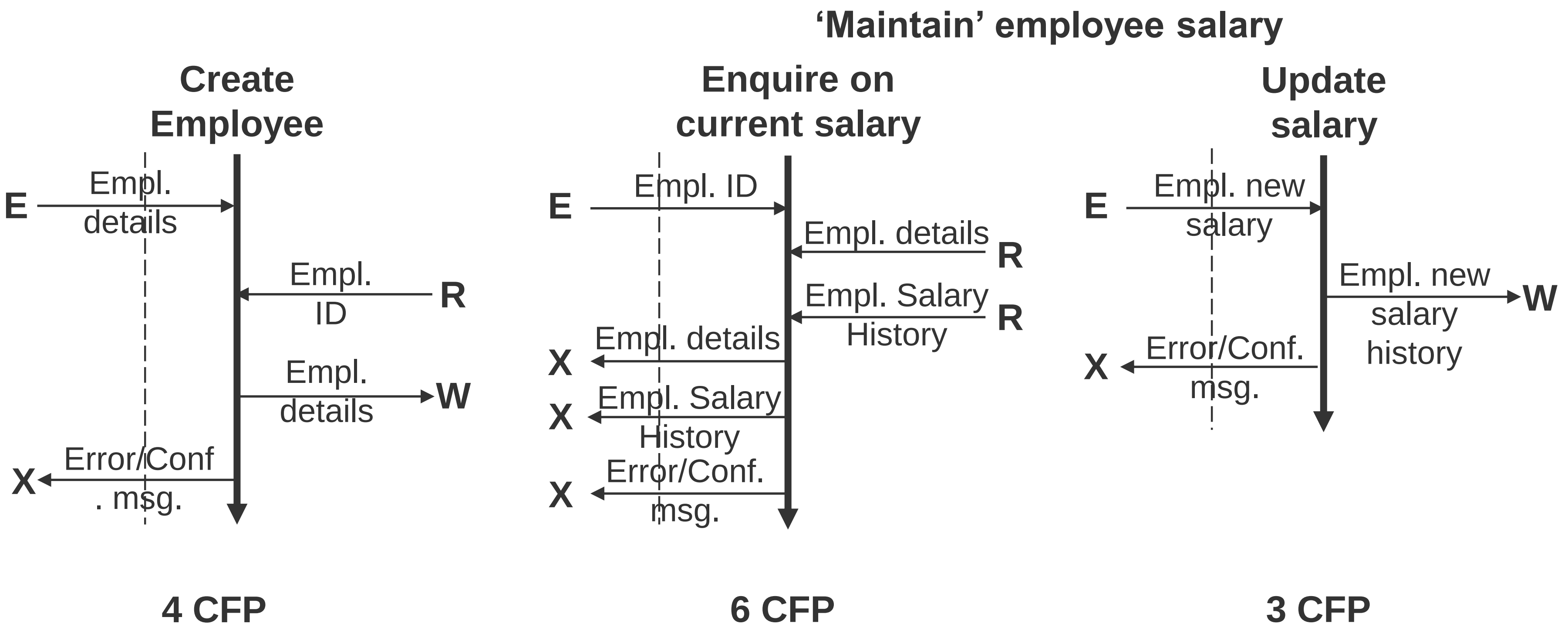
Some more definitions

A **data movement** (E, X, R or W) moves a single **data group**, where:

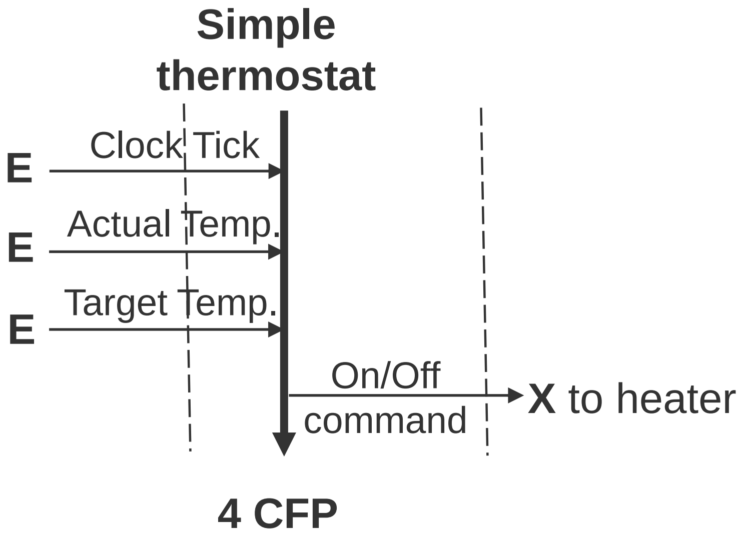
- A **data group** consists of one or more **data attributes** that describe a single **object of interest**
- An **object of interest** is any ‘thing’ (physical or conceptual) in the world of the **functional user**, about which the software being measured must process or store/retrieve data

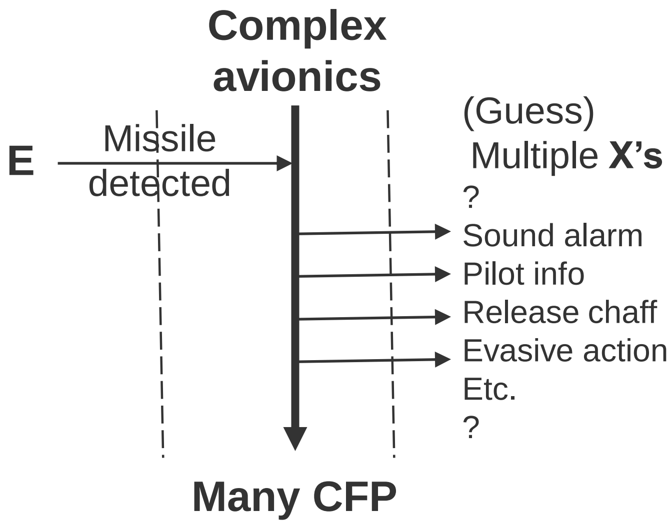
(Think of an entity-type, a relation in 3NF, or the subject of an object class)

Example business application Functional Processes



Example real-time Functional Processes

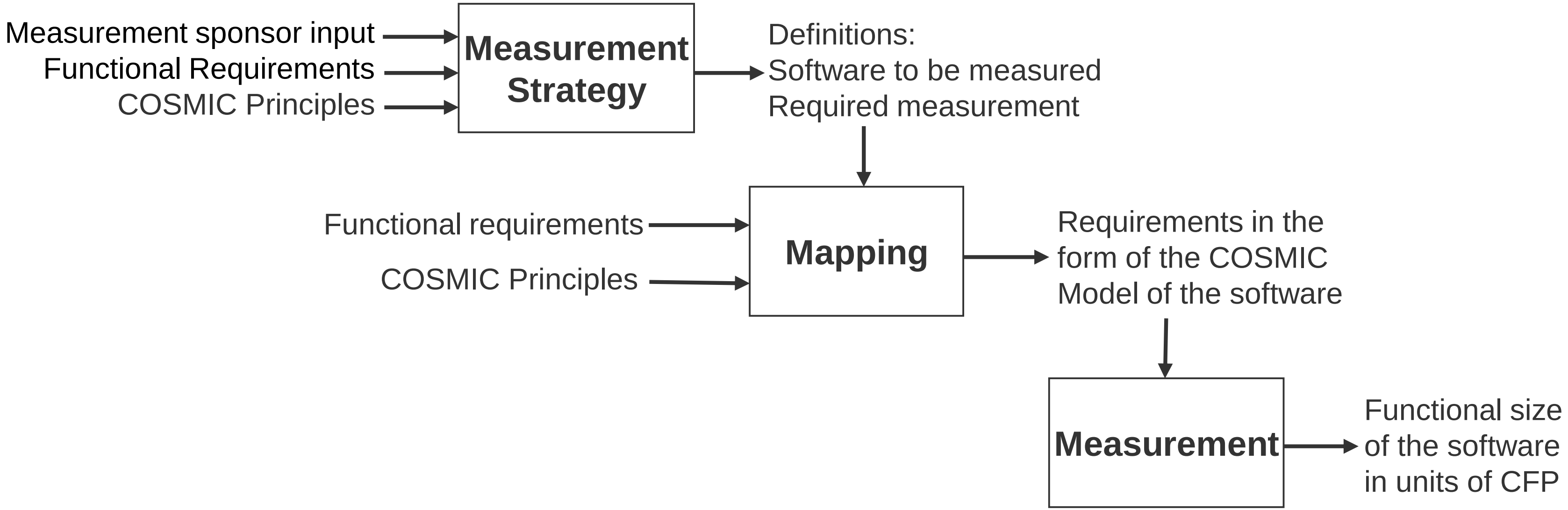




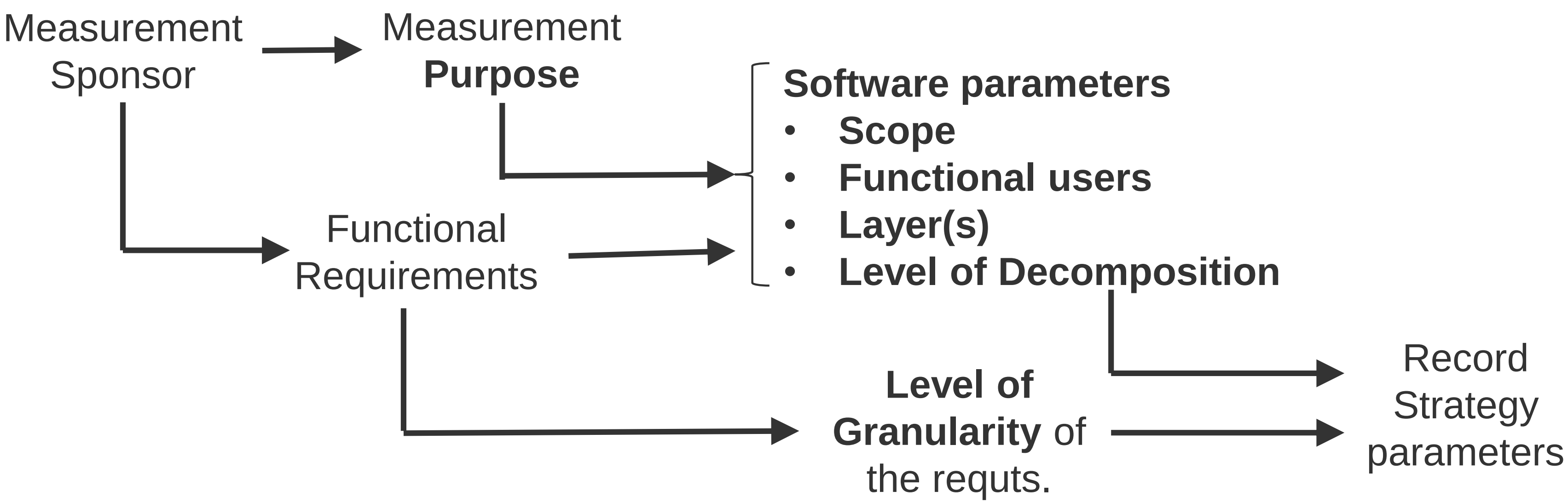
There is no upper limit to the size of a functional process

- A functional process must have at least 2 CFP
  - A triggering Entry
  - An 'outcome' – i.e. a Write or an Exit
- Largest reported functional processes?
  - In banking ~ 65 CFP
  - In avionics >100 CFP
- The smallest change to an existing functional process is 1 CFP

Measurement involves a three-phase process

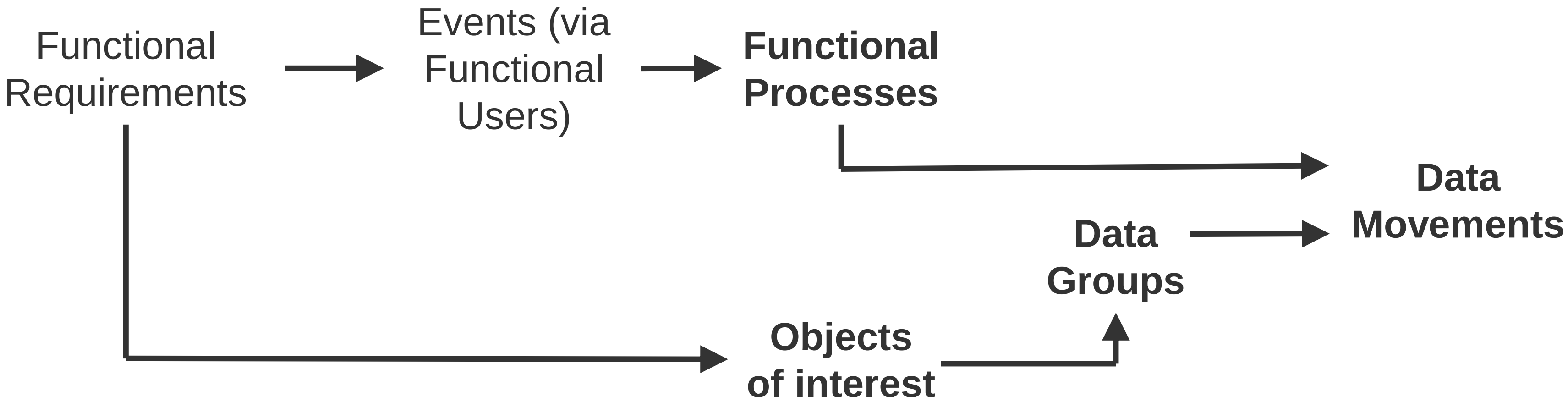


Measurement Strategy phase 1: define the measurement parameters



Recommendation: define 'patterns' for standard M'ment Strategy parameter sets

Mapping phase 2: map the requirements to the COSMIC model



Measurement phase 3: count the data movements

Within a defined Measurement Scope:

Software size	=	Sum of sizes of Functional Processes	=	Count of all their Data Movements

Size of a change to software	=	Count of DM's added	plus	Count of DM's modified	plus	Count of DM's deleted

An example result from a measurement

	Data Group Names										Nos. of Data Movements				
Acme Car Hire Functional Processess	Customer name	Customer Master Record	Customer name and address	Customer ID	Customer Summary Details	Customer Details	Customer latest Invoice	Exisitng Bookings	Booking Details	Error/Confirmation Message	Entries	Exits	Reads	Writes	Total
Search Customer by name	E	R	X							X	1	2	1		4
View Customer Summary details		R		E	X			X			1	2	1		4
View Customer Details		R		E		X					1	1	1		3
Update Customer details		W		E						X	1	1		1	3
Add new Customer		W				E				X	1	1		1	3
Print current Invoice		R		E				R, X		X	1	2	2		5
View Booking details				E					R, X	X	1	2	1		4
					Totals for Acme System:						7	11	6	2	26

‘What about?’ Common objections to COSMIC size measurement

Needs too much detail for early estimating	There are variants for approximate sizing
Non-functional requirements	Quality NFR evolve wholly or partly into functional requirements that COSMIC can measure. Other NFR affect cost, effort but not software size
Complexity	A COSMIC size closely measures the software ‘crude complexity’ of the functional requirements at the level of granularity of the data movements
Re-used software	Distinguish sizes of new and re-used software

In Agile processes, COSMIC sizes can be measured at any level of aggregation

System

Release

Iteration

User Story
(new &/or re-work)

Hence usable for:

- early total System sizing and effort estimation,
- User Story sizing and estimation,
- progress control, etc.

Agenda

- Goals: the importance of measuring software size
- Overview of the COSMIC method
- The acid test. Do COSMIC sizes correlate with effort?
- Conclusions and future

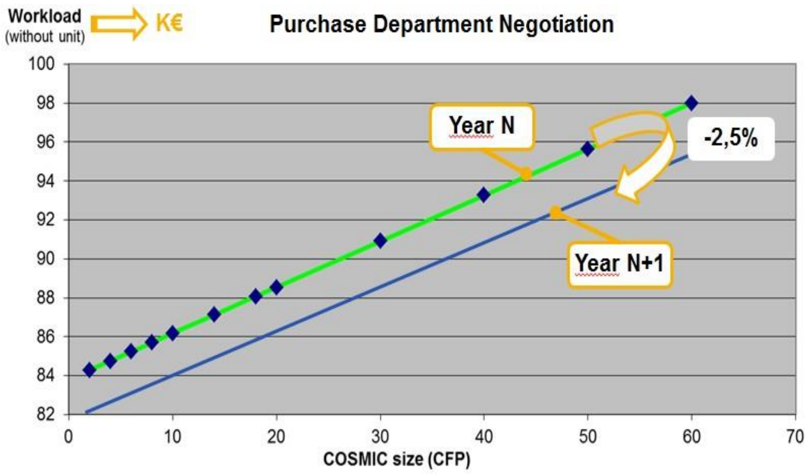
Case 1: Renault Automotive use in embedded software

Renault [1)] uses CFP sizing to control the development and enhancement of Electronic Control Units (ECU's)

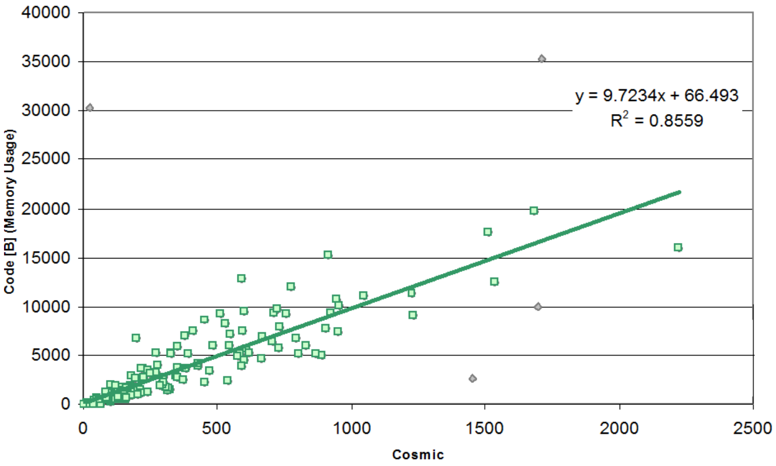
- tracks progress of ECU specification teams…
- who create designs in Matlab Simulink…
- which are automatically measured in CFP

Motivation for automation: speed, accuracy of measurement

Renault achieves remarkable cost estimation accuracy from its ECU designs

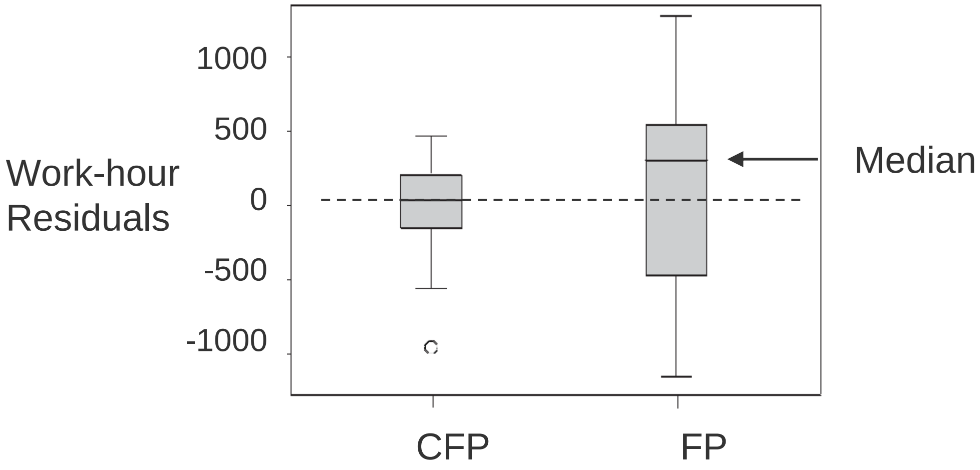


Cost vs size (CFP)



Memory size vs software size (CFP)

Case 2: Web effort estimation is more accurate with COSMIC than using ‘1G’ FPA



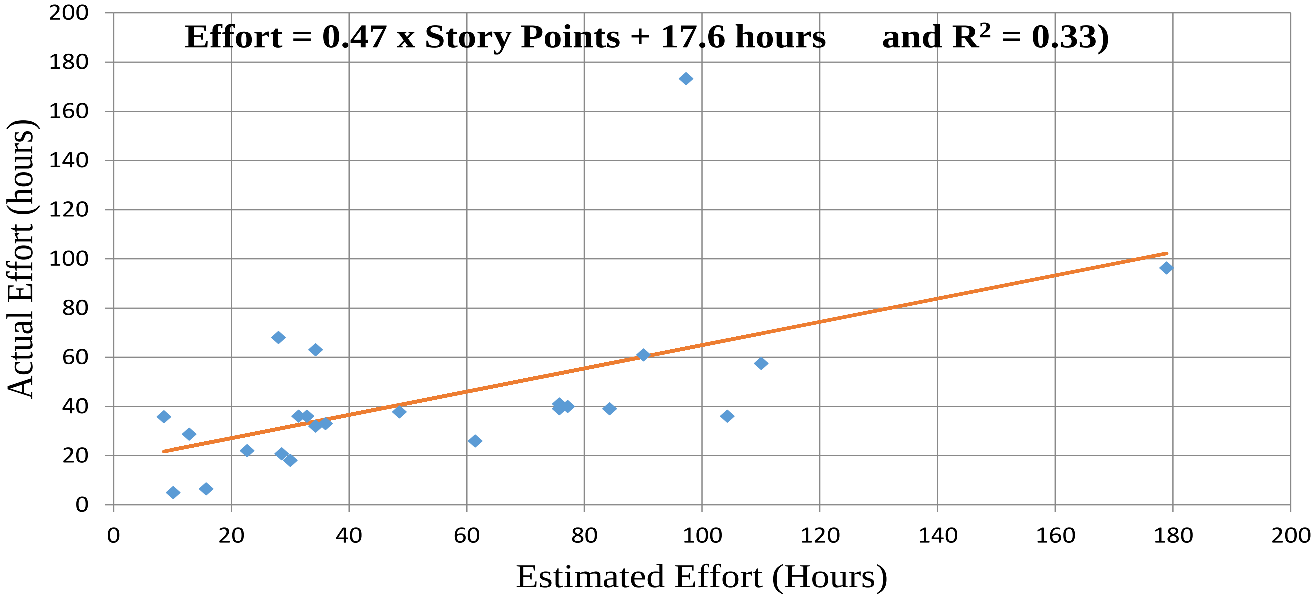
25 industrial Web applications 2)

Conclusions:
‘*The results of the … study revealed that COSMIC outperformed Function Points as indicator of development effort by providing significantly better estimations*’

Case 3: A Canadian supplier of security and surveillance software systems

- A customer request for new or changed function is called a ‘task’
- Scrum method used with iterations of 3 – 6 weeks
- Teams estimate tasks within each iteration in User Story Points, and convert directly to effort in work-hours
- CFP sizes were measured on 24 tasks from nine iterations, for which USP ‘sizes’, estimated and actual effort data were available [3)]

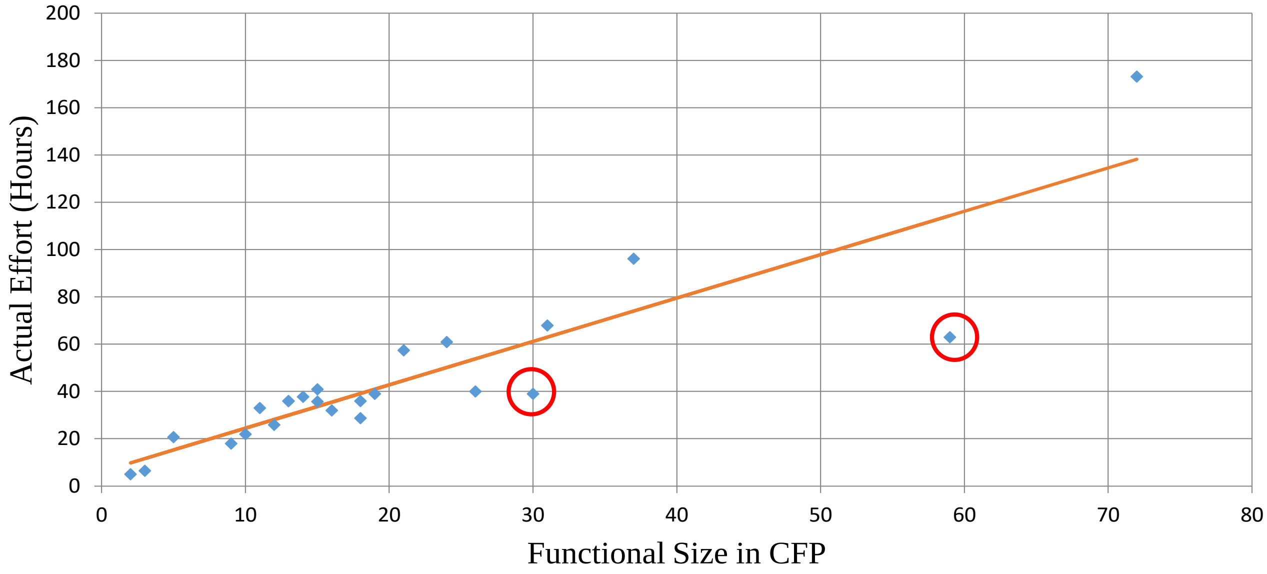
User Story Point sizes are a poor predictor of effort



Notice the wide spread and the 17.6 hours 'overhead'

The CFP vs Effort graph showed a good fit, but revealed two outliers

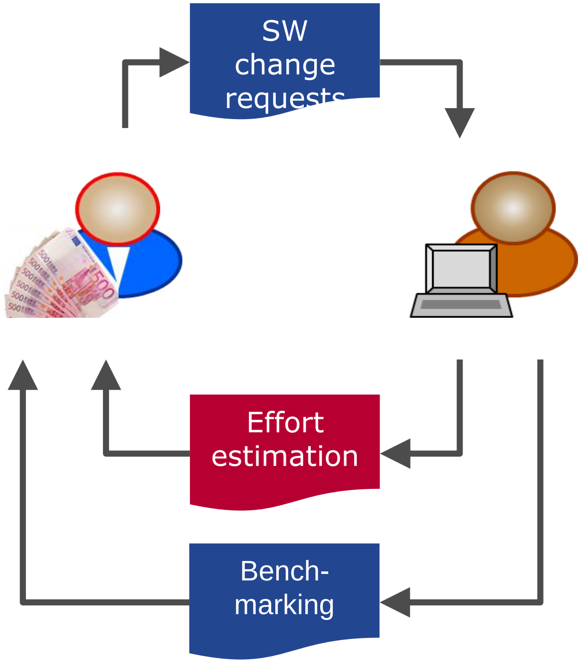
Effort = 1.84 x CFP + 6.11 hours and R^2 = 0.782



Two tasks with low effort/CFP had significant software re-use.
Removing these outliers improves the R^2 to 0.977

Case 4: A global automotive manufacturer improved estimating for maintenance changes

- Context: real-time embedded software
- Starting point: text/diagrams for required changes
- A COSMIC-based measurement program [4)] resulted in
  - Estimating precision of 10 – 20% within one year of starting
  - More disciplined, repeatable processes, internal benchmarks
  - Greater customer/supplier trust



Conclusions from case studies of size/effort relationships

COSMIC-measured sizes correlate very well with effort

Investing in COSMIC measurement and recording cost drivers should help improve:

- estimating accuracy
- organizational learning for process improvement
- quality control of requirements

Most accurate cost estimate → least cost project [5)]

Agenda

- Goals: the importance of measuring software size
- Overview of the COSMIC method
- The acid test. Do COSMIC sizes correlate with effort?
- Conclusions and future

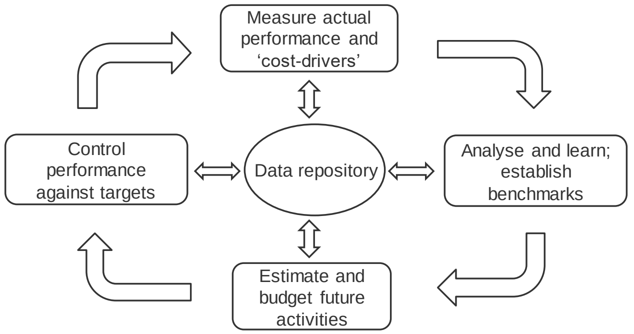
The COSMIC method has many advantages over other methods of measuring software size

- Based on fundamental software engineering principles, hence:
  - ‘future-proof’ (and stable)
  - relatively easy to automate
- Applicable to business, real-time and infrastructure software, at any level of decomposition
- ISO/IEC standard; endorsed by GAO [6)], NIST [7)], etc
- ‘Open’, freely available via www.cosmic-sizing.org [8)]

Estimating software processes can never be an exact science - so iterate!

Software development is partly mechanical, but partly creative and unpredictable

Repeat the control cycle *frequently*



AGILE!

using a proper size scale – ~~Story Points~~

COSMIC Function Points

Thank you for your attention

Charles Symons (www.cosmic-sizing.org)
cr.symons@btinternet.com

References

1. ‘Manage the automotive embedded software development cost & productivity with the automation of a Functional Size Measurement Method (COSMIC)” Alexandre Oriou et al, IWSM 2014, Rotterdam, www.ieeexplore.org
2. ‘Web Effort Estimation: Function Point Analysis vs. COSMIC’, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Federica Sarro, Information and Software Technology 72 (2016) 90–109
3. ‘Effort Estimation with Story Points and COSMIC Function Points - An Industry Case Study’, *Christophe Commeyne, Alain Abran, Rachida Djouab.* ‘Software Measurement News’. Vol 21, No. 1, 2016. Obtainable from www.cosmic-sizing.org
4. Private communication, Vector Consulting (Germany), 2016
5. ‘Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs, Government Accountability Office (USA), 2011, http://www.gao.gov/new.items/d093sp.pdf
6. ‘A rational foundation for software metrology’, National Institute for Standards and Technology (USA), NIST IR – 8101, https://doi.org/10.6028/NIST.IR.8101, 2016
7. ‘Introduction to the COSMIC method of measuring software’, v1.1, https://cosmic-sizing.org/publications/introduction-to-the-cosmic-method-of-measuring-software-2/