SUCCESSFUL ESTIMATING FROM REQUIREMENTS USING COSMIC FSM

UCL CREST Open Workshop

March 2017, London

Charles Symons

My interest in this topic

- At the 'Micro' level
  - Interest in improving methods for measuring performance & estimating for software activities

- At the 'Macro' level
  - Helping customers control price/performance of software suppliers and delivery to time & budget

Agenda

- Objectives of software size measurement: FSM methods
- The COSMIC method – key features
- Evidence that the COSMIC method achieves its goals
- Conclusions, and future measurement challenges

Objective 1: control and compare performance of project activities

Delivery to budget & time:
- Actual vs. Estimated Cost
- Actual vs. Estimated Duration

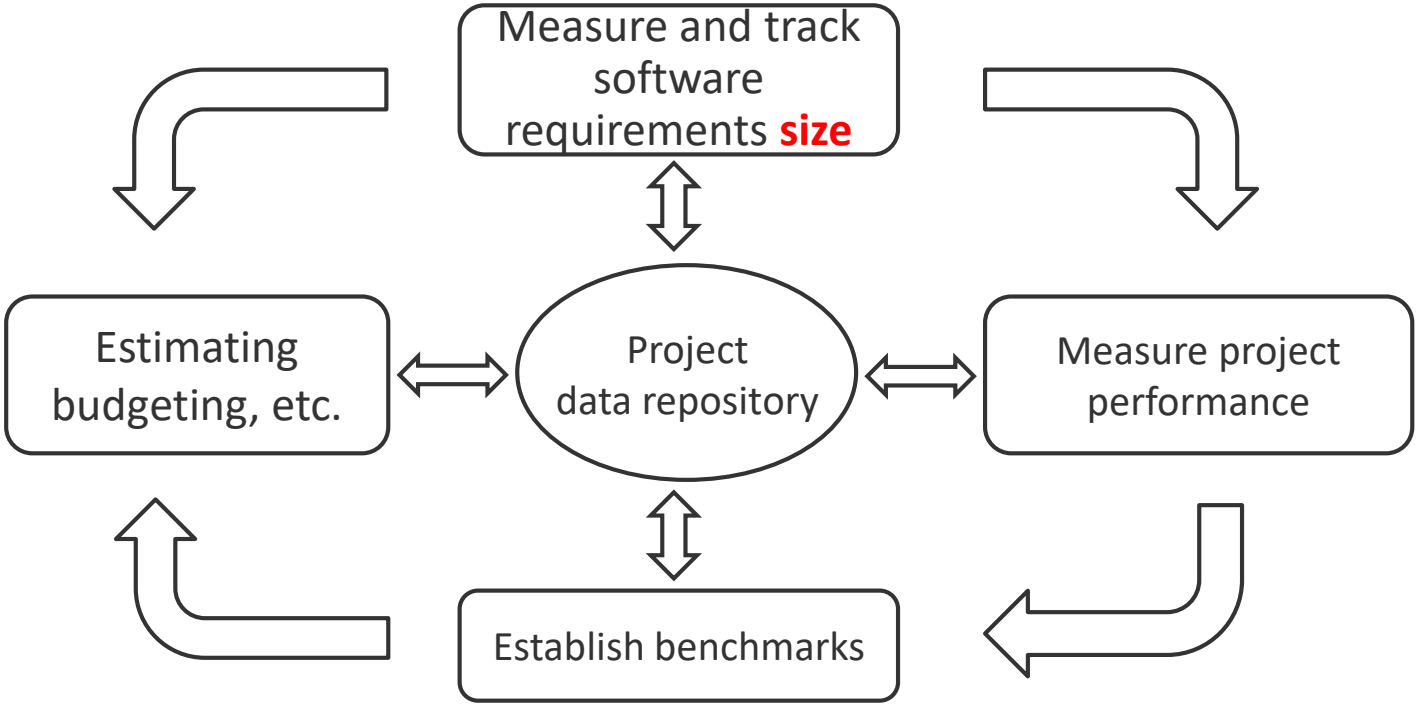
Project speed
- **Size** / Duration

Project productivity
- **Size** / Effort

Product scope and quality
- **Size**
- Functional (e.g. business needs)
- Technical (e.g. maintainability, post-delivery defects density)

Also consider the performance of on-going **maintenance and enhancement** activities

Objective 2: use performance data for estimating future projects



There are three types of requirements for a software project or iteration, etc.

Functional User Requirements (FUR)
- what the software must do

Non-Functional Requirements (NFR)
- quality, technical and environmental constraints, etc.

Project Requirements and Constraints (PRC)
- targets, processes & tools, languages, resources, dependencies, etc.

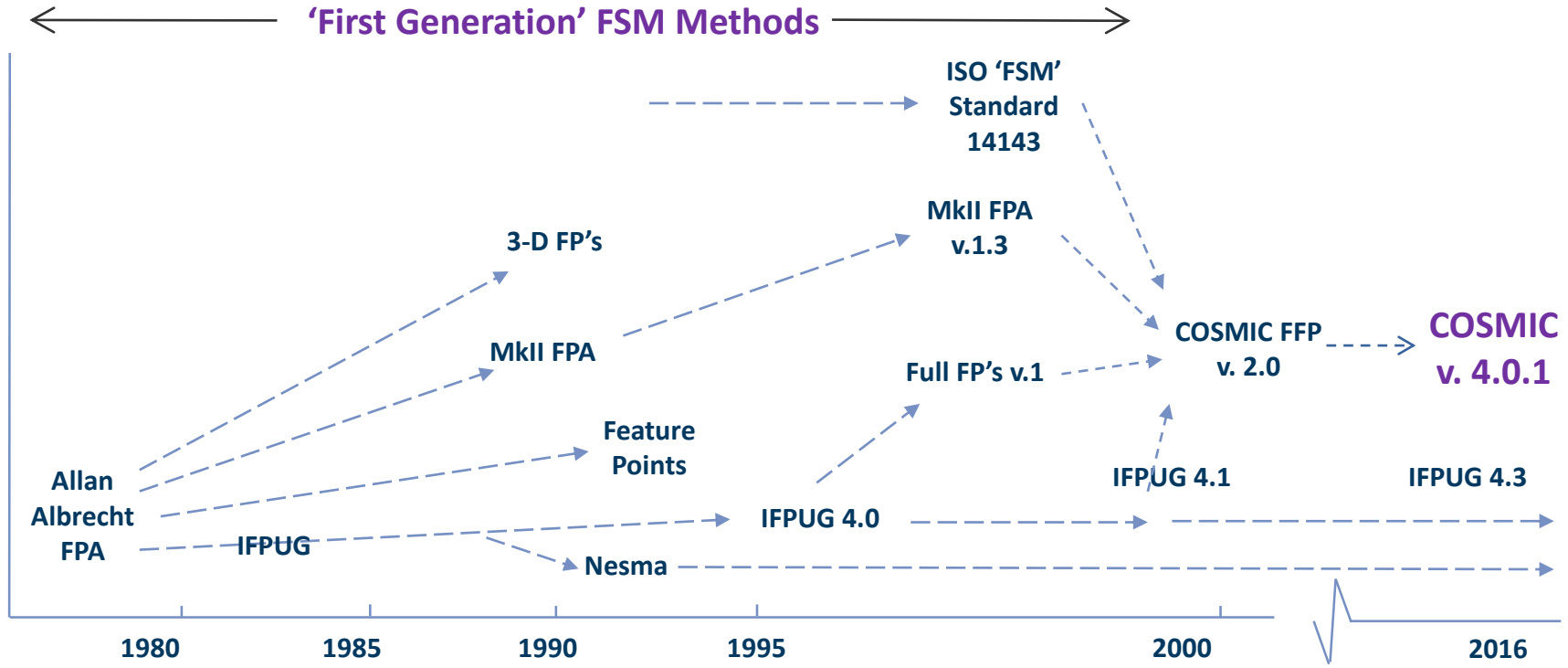
The size of the task

Convert size to estimated effort

Only Functional Size Measurement (FSM) methods can help achieve both objectives

Counts of lines of code:	X Cannot be estimated until software designed X Technology-dependent, no standards ✓ Accounts for all requirements that are delivered
Functional size:	✓ International standard methods ✓ Technology-independent ▪ What about 'Non-Functional' Requirements?
Other sizing methods: e.g. UCP, OOP, Story Pts. (?) etc:	X No reliable standards; only local benchmark data possible X What about 'Non-Functional' Requirements? X Early total effort estimation?

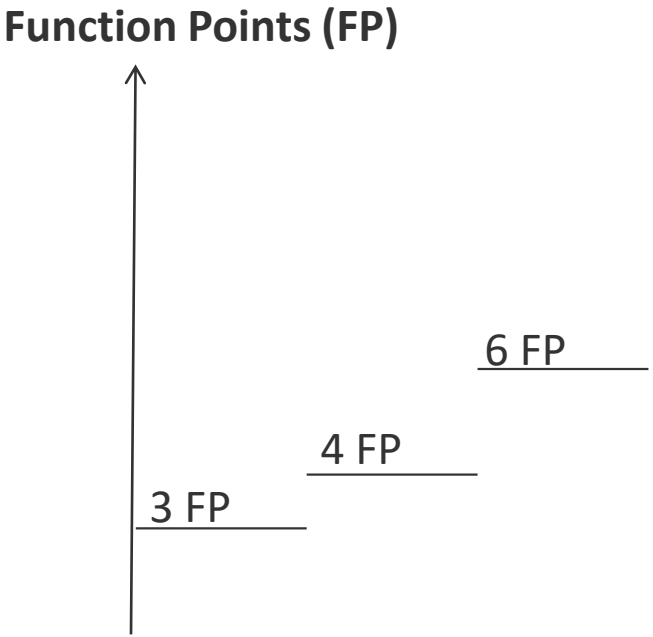
There are several models for Functional Size Measurement

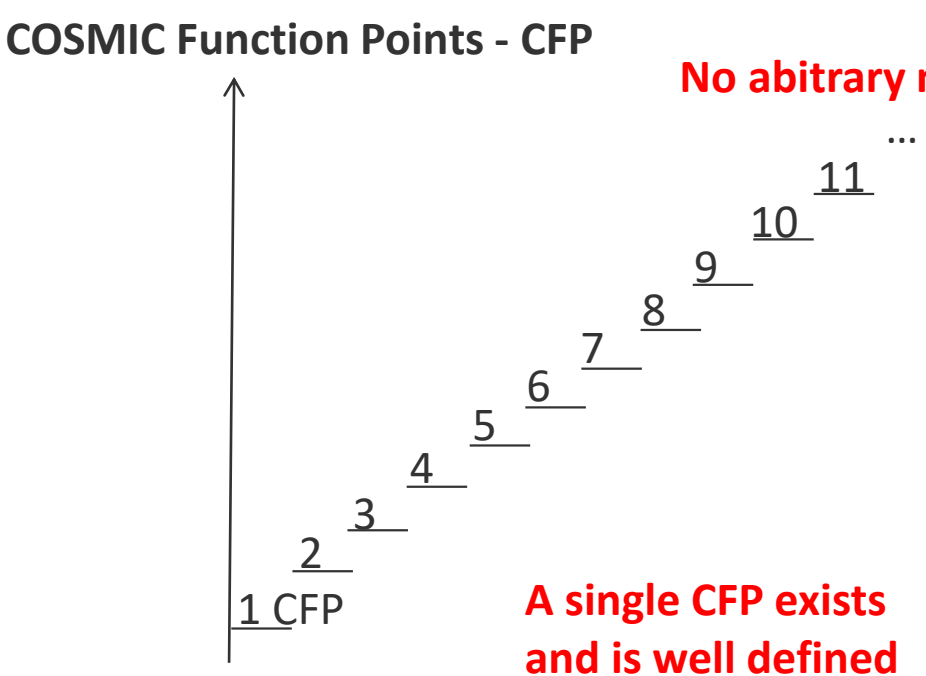


‘Traditional’ FP methods have several weaknesses

- Base models of functionality
  - Difficult to reconcile with modern requirements engineering and development methods (e.g. agile)
  - Designed to measure ‘whole’ business applications
- Sizes of functional components
  - Limited size ranges, no well-defined unit of measure
  - Calibrated on relative effort to develop, hence ‘frozen’ in a particular technology era

Traditional FP vs. COSMIC FP measurement scale – a key difference





Agenda

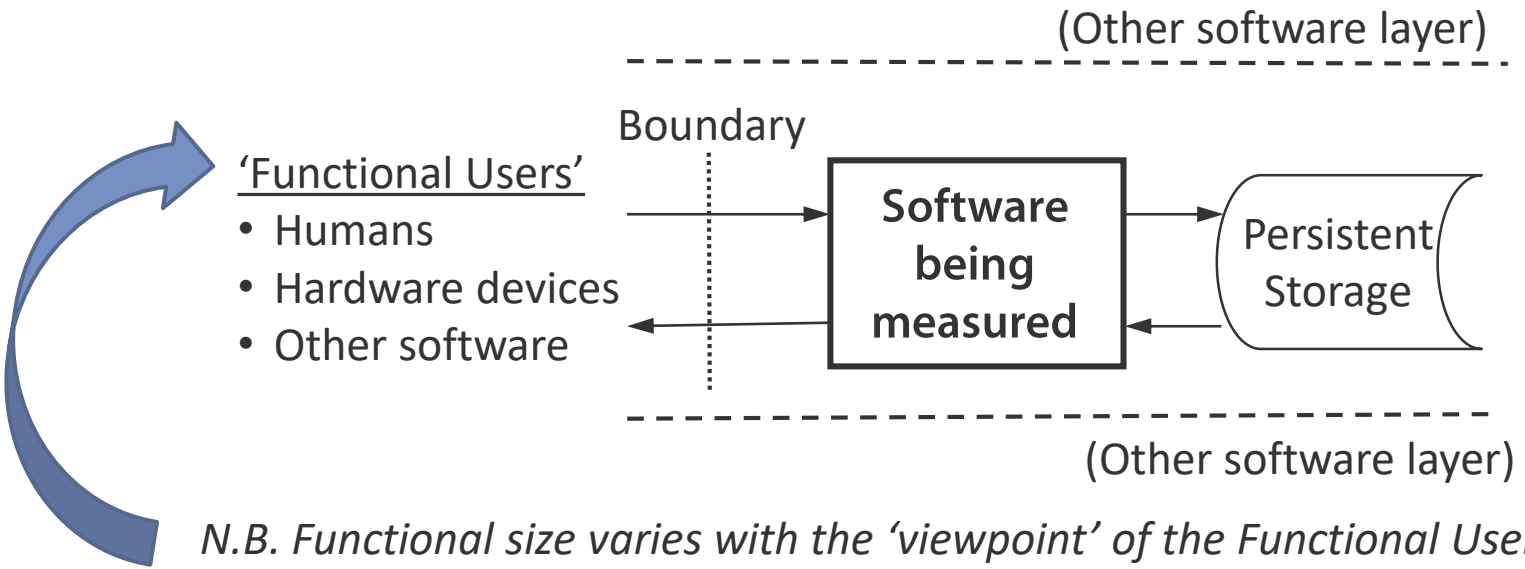
- Objectives of software size measurement: FSM methods
- The COSMIC method – key features
- Evidence that the COSMIC method achieves its goals
- Conclusions, and future measurement challenges

COSMIC method goals for sizing software Functional User Requirements

- usable for both our software measurement objectives
- based on fundamental software engineering principles
- for business, real-time and infrastructure software
- independent of technology or processes used for the software or project
- ‘open’, freely available (via www.cosmic-sizing.org)

The method design uses two models based on fundamental software engineering principles

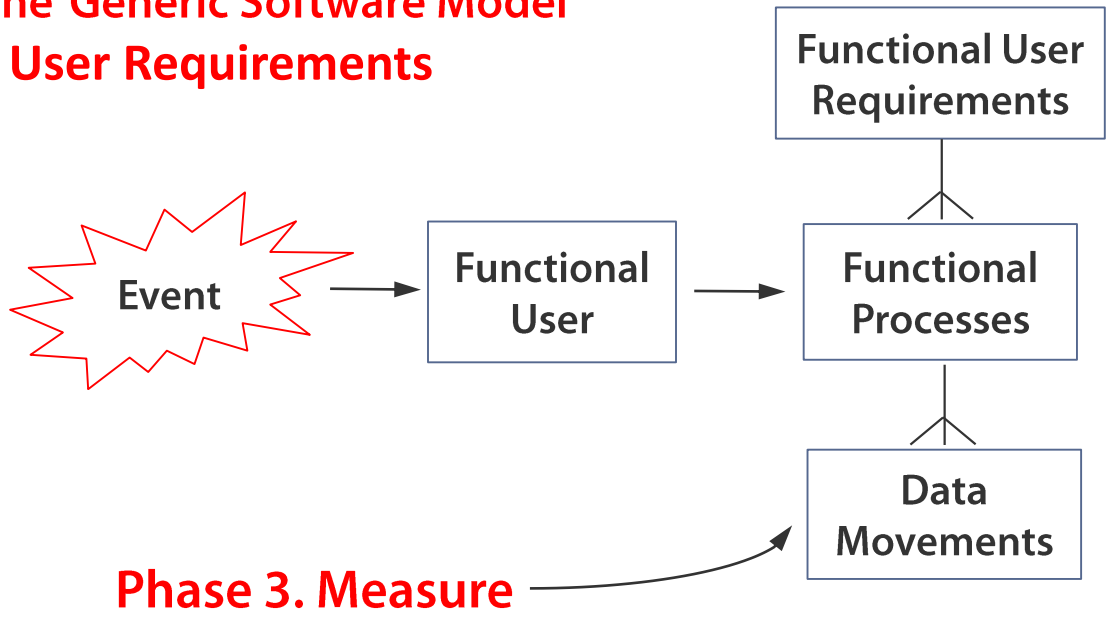
Phase 1. Define the Measurement Strategy: the 'Software Context Model'



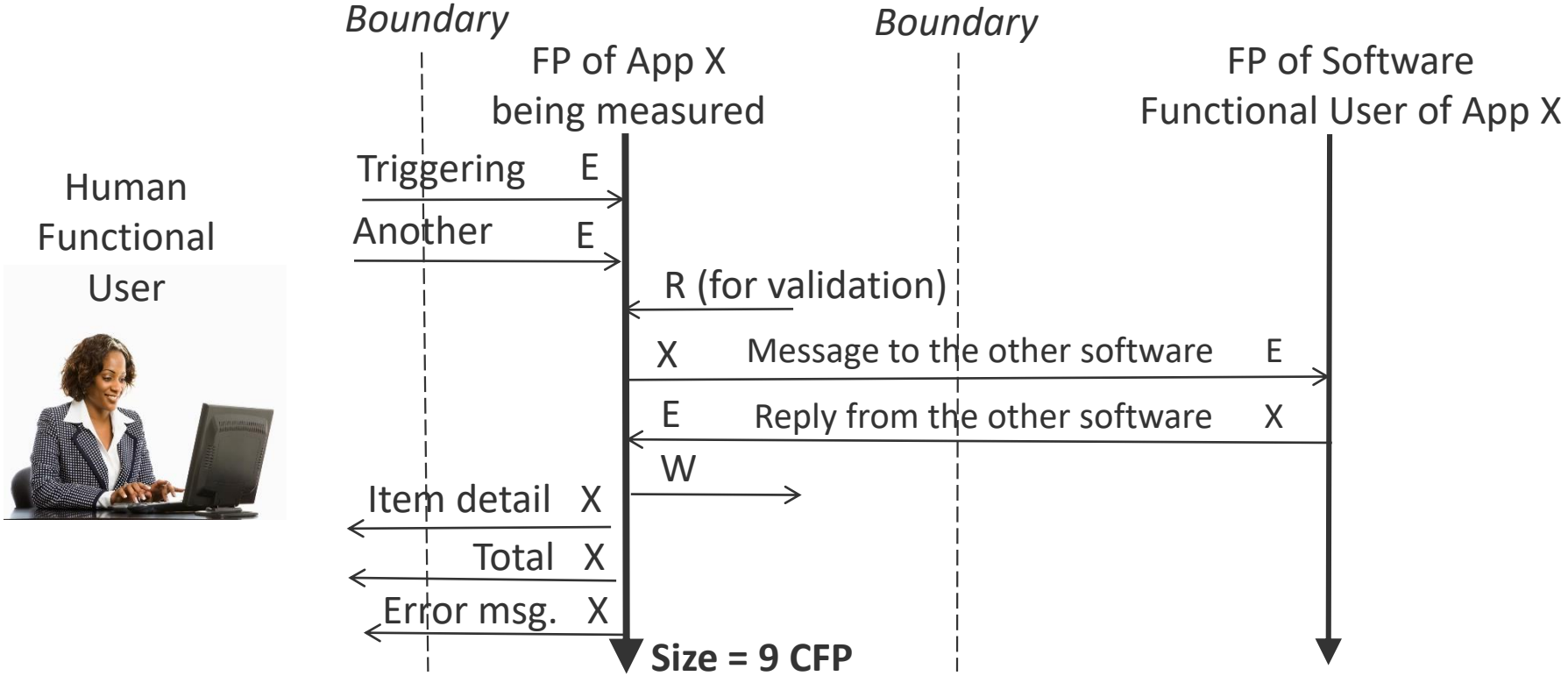
N.B. Functional size varies with the 'viewpoint' of the Functional User(s)

The 'Mapping' and 'Measurement' Phases

Phase 2. Develop the 'Generic Software Model' of the Functional User Requirements



A Message Sequence Diagram for an example functional process

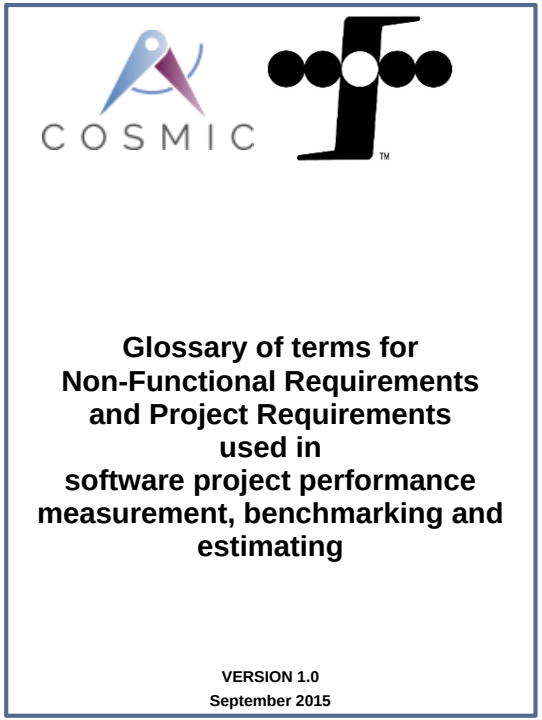


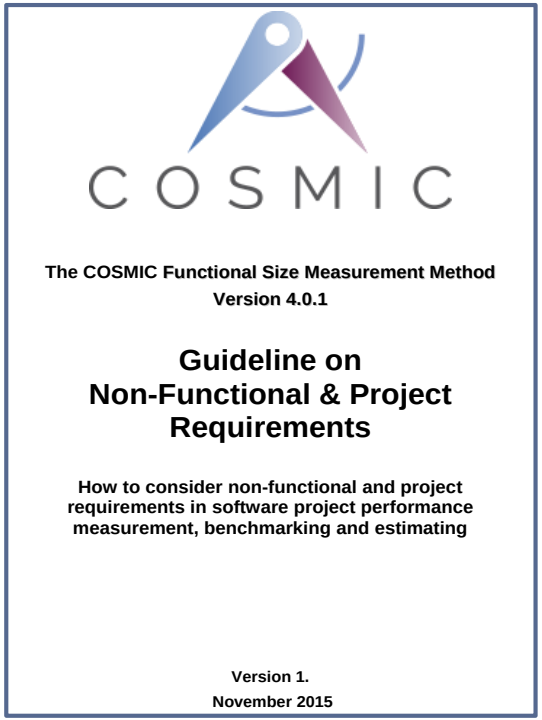
What about non-functional requirements (NFR)?

There was no good standard definition of a NFR

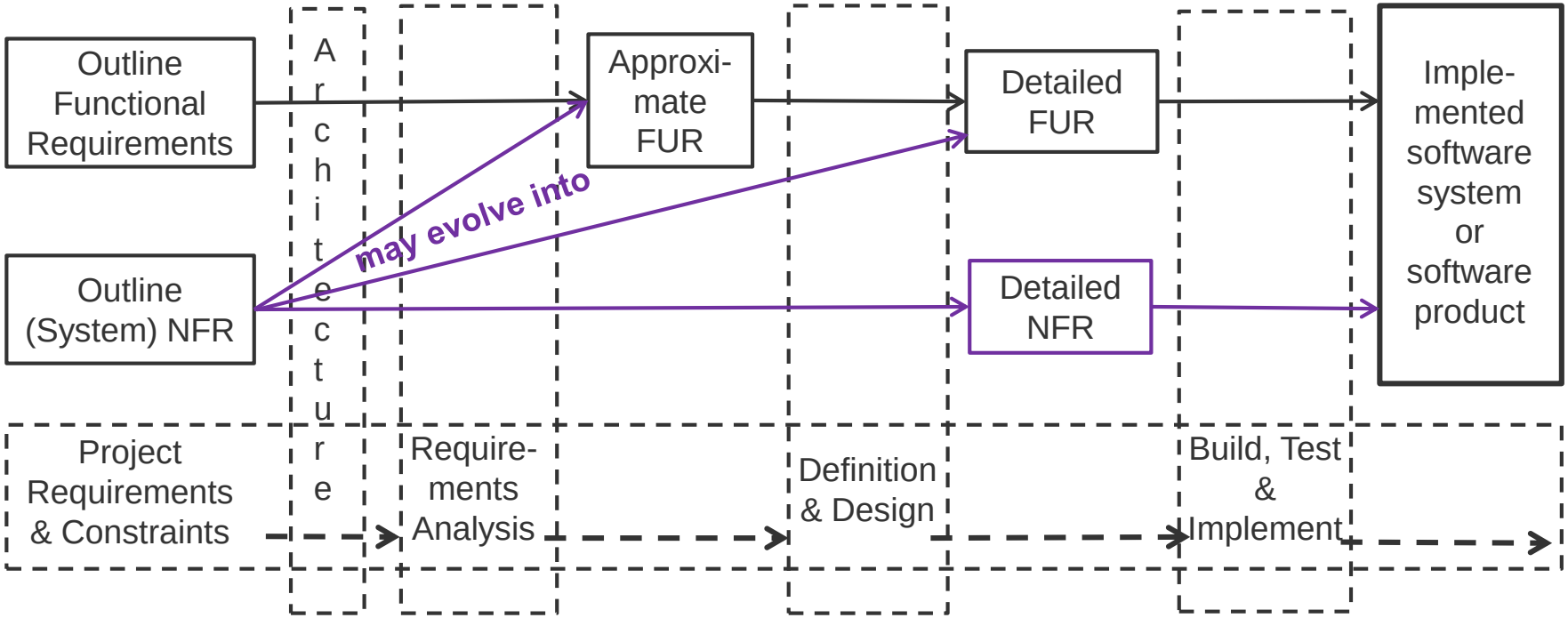
A joint COSMIC/IFPUG effort developed good definitions and a Glossary of NFR and Project Requirements

The COSMIC Guideline advises how to deal with NFR





Studies show that system NFR may evolve into software FUR, that COSMIC can measure

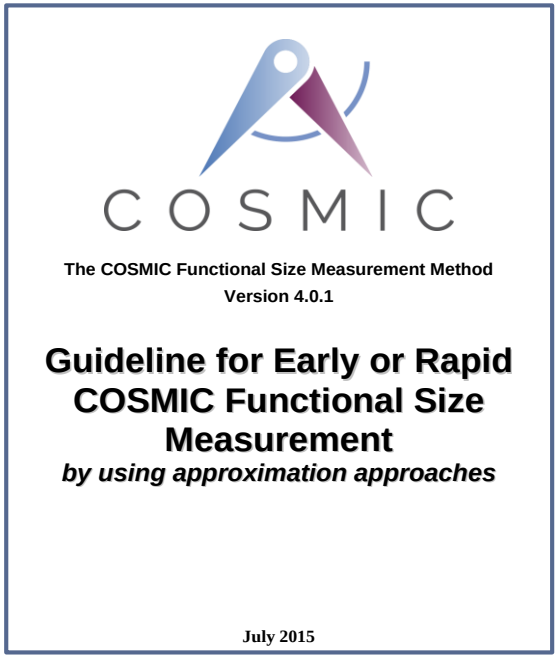


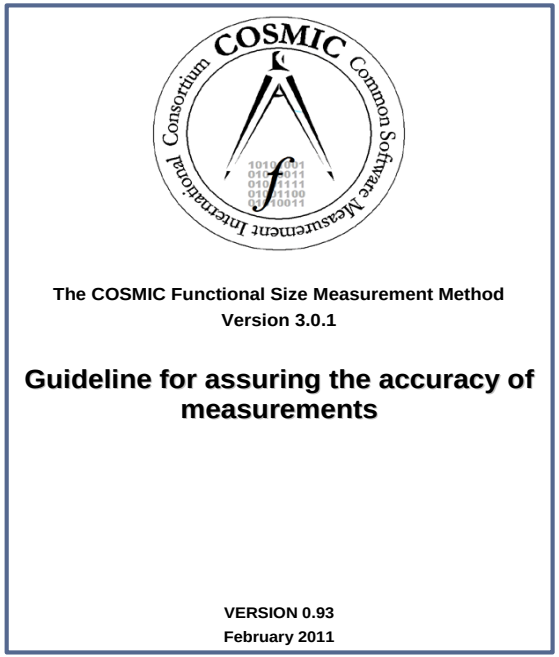
Size/Cost estimates are usually needed before the FUR have been defined in detail

So we developed:

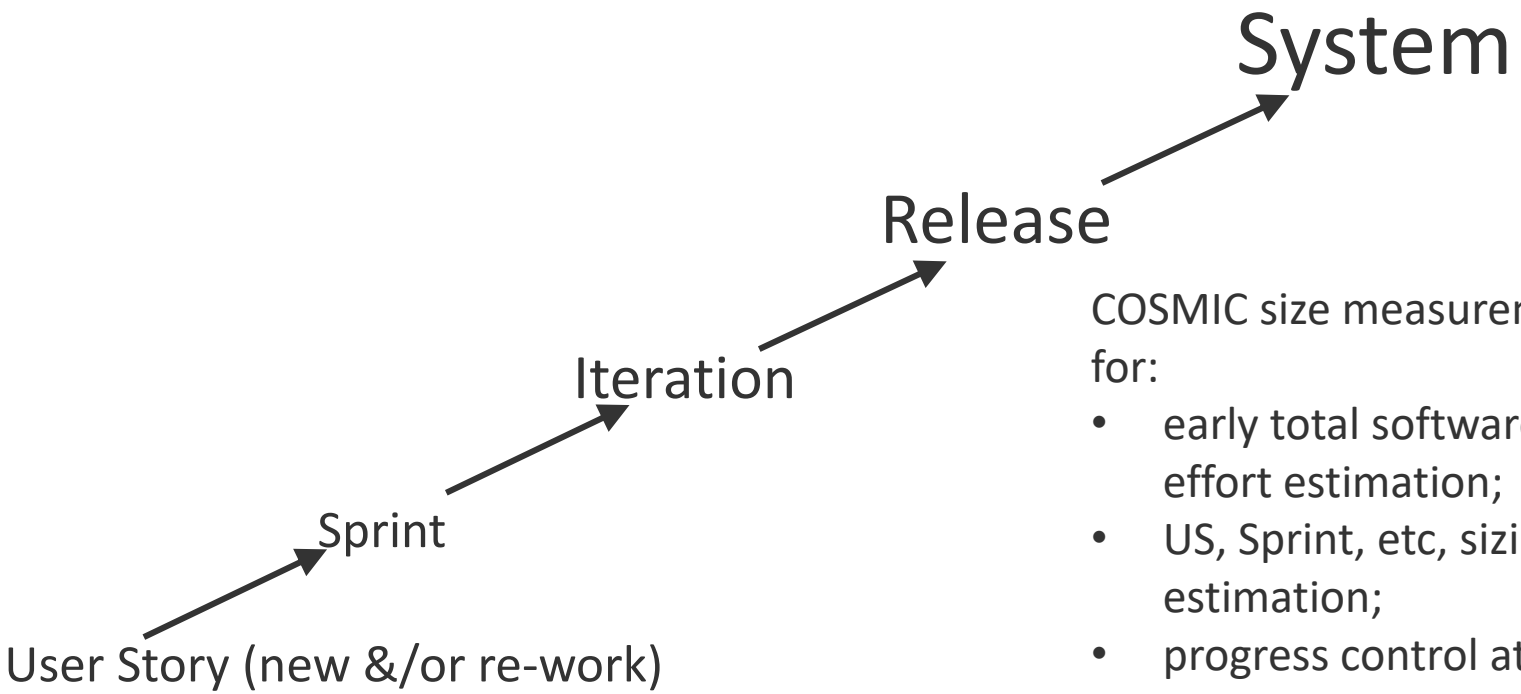
A Guideline describing a range of **Approximate Sizing methods**

A Guideline on **'Assuring the accuracy of COSMIC measurements'**





Agile: sizes can be measured at all levels and aggregated up to the system size



COSMIC size measurement is usable for:

- early total software sizing and effort estimation;
- US, Sprint, etc, sizing and estimation;
- progress control at any level.

Guidelines and case studies show how to apply the COSMIC method in various domains

Guidelines

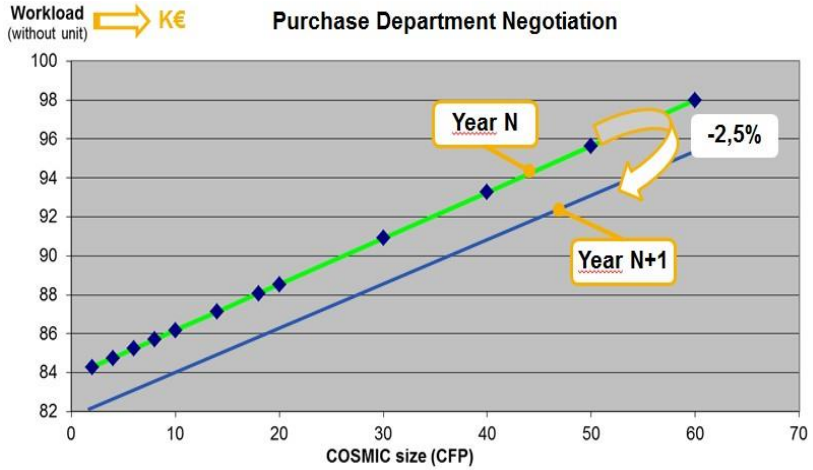
- Business applications
- Real-time software *
- Data Warehouse software
- SOA software*
- Mobile apps (in devt.)*
- Agile Developments

- (* should suffice for the Internet of Things?)

Agenda

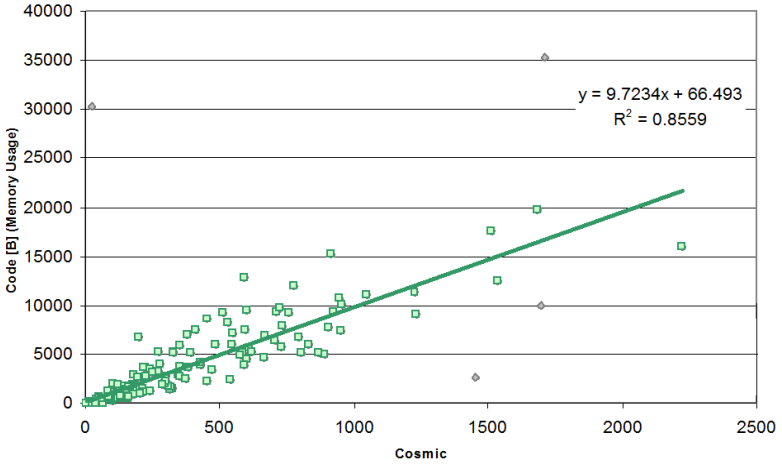
- Objectives of software size measurement: FSM methods
- The COSMIC method – key features
- Evidence that the COSMIC method achieves its goals
- Conclusions, and future measurement challenges

Case: Renault [1)] estimates sizes and then costs from its ECU specifications (automatically!)



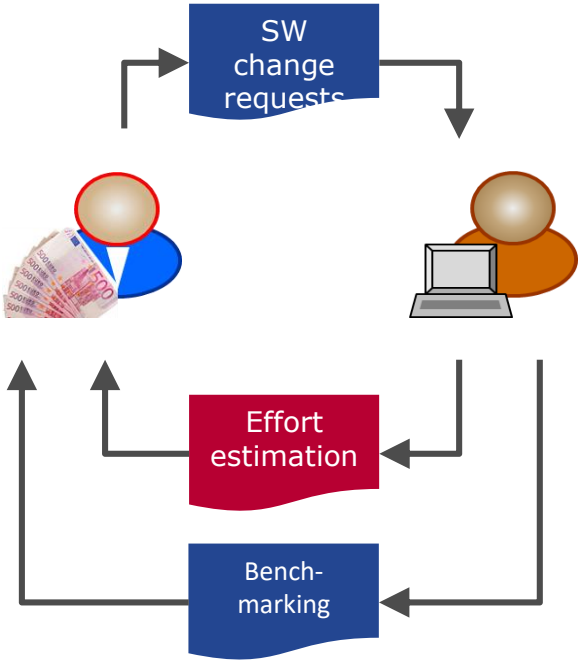
Cost vs size (CFP)

Memory size vs software size (CFP)

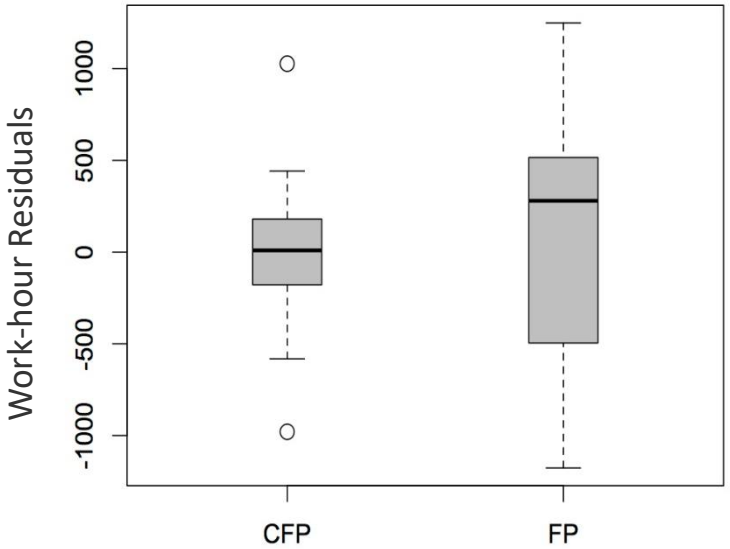


Case: Another global automotive manufacturer [2)] improved cost estimating for maintenance changes

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



Case: Italian web software supplier - effort estimation is more accurate with CFP than with FP [3)]



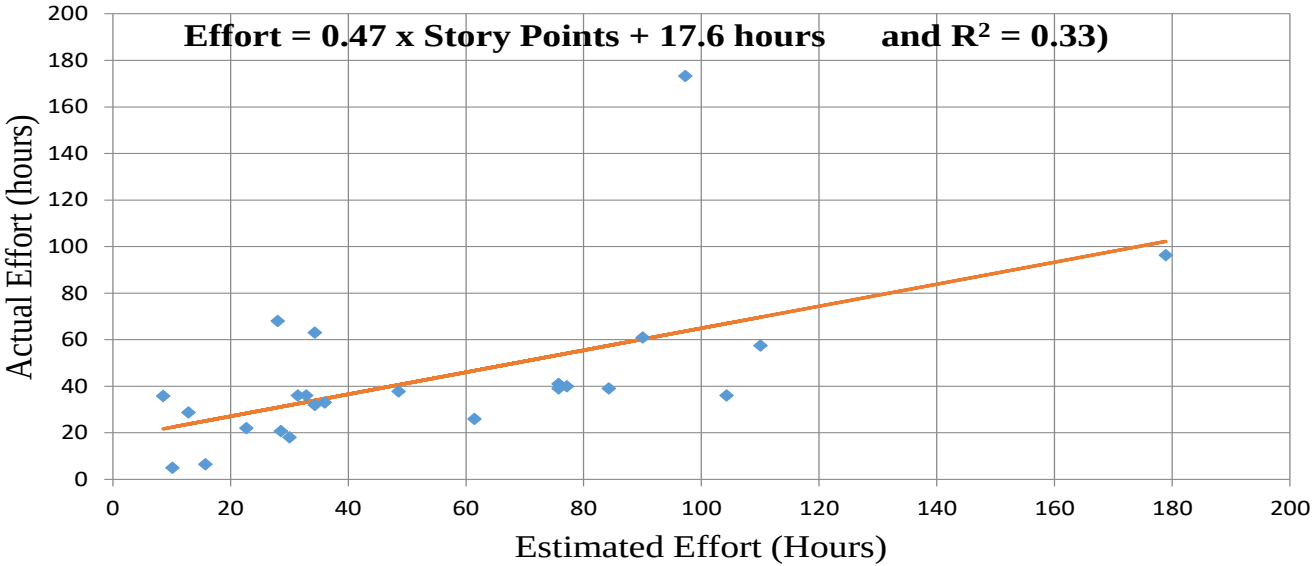
25 industrial Web applications

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

Case: A Canadian supplier of security and surveillance software systems: Agile estimating [4)]

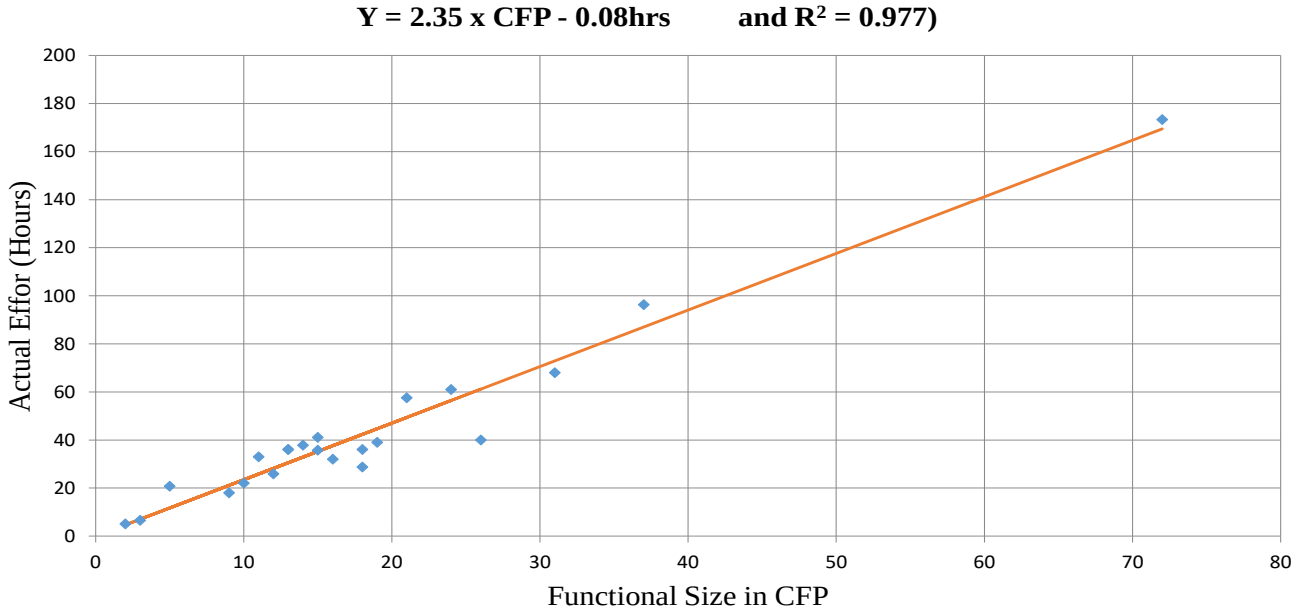
- Customer requests for new or changed function are called ‘tasks’
- Supplier uses the Scrum method; iterations last 3 – 6 weeks
- Teams used ‘planning poker’ to estimate tasks in an iteration in ‘USP’ then converted directly to effort in work-hours
- Measurements on 24 tasks from nine iterations, for which estimated and actual effort were available, were re-measured in CFP

The actual Effort vs USP size graph (24 tasks) would be poor for effort estimation



Notice the wide spread and the 17.6 hours 'overhead'

The final actual Effort vs CFP size graph is much better for estimating



CFP measurements revealed that two tasks had very low effort/CFP due to much software re-use. These were considered separately.

Case: An Australian User view of 'COSMIC for Agile' [5)]

"We have found that adopting this approach provides us with excellent predictability and comparability across projects, teams, time and technologies."

The reality of achieving predictable project performance has driven me to investigate many methods of prediction. COSMIC is the method that lets me sleep at night."

Denis Krizanovic, Aon Australia, August 2014

Agenda

- Objectives of software size measurement: FSM methods
- The COSMIC method – key features
- What design issues did we have to solve?
- Evidence that the COSMIC method achieves its goals
- Conclusions, and future measurement challenges

Conclusion: COSMIC has achieved all its design goals: a major advance in FSM

- The method, based on fundamental principles, is stable and 'future-proof'.
- CFP sizes correlate very well with effort and code size
- Very widely used:
  - Business, real-time and infrastructure software
  - 'Measurement Manual' available in 11 languages
  - 50% of known users are software houses
  - Adopted by Governments (China, Mexico, Poland)
  - > 30,000 downloads of research & conference papers
- ISO/IEC 19761 standard: GAO, NIST endorsed.

The biggest challenge for increased acceptance of FSM: automation

CFP sizes are *already* being measured:

- automatically, from requirements models in UML, Simulink, SCADE
- with manual assistance, from requirements in text and Java code

The need: automatic measurement of

- requirements and designs, in various forms, especially for:
  - User Stories
  - Early outline requirements
- programs or executing code (in various languages)

Some other challenges

- A Guideline on sizing and estimating software assembled from components:
  - New
  - Modified
  - Re-used

  New and Modified: } Existing rules

  Re-used: } Measure the API?

- An ‘Expert-level’ certification exam
- Marketing

Postscript: join me for a free seminar: 'Introduction to COSMIC FSM'

- Agenda: basic method features, approximate sizing, NFR, uses in Agile projects, estimating, etc.
- Friday May 5th, 10:30 – 16:00
- Offices of SITA, Hayes, West London

Thank you for your attention

(www.cosmic-sizing.org)

Charles Symons cr.symons@btinternet.com

References

1. Alexandre Oriou et al, "Manage the automotive embedded software development cost & productivity with the automation of a Functional Size Measurement Method (COSMIC), Alexandre Oriou et al, IWSM/Mensura Conference 2014, www.ieeexplore.org
2. Private communication
3. S. Di Martino, F. Ferruci. C. Gravion, F. Sarro, "Web Effort Estimation: Function Point Analysis vs. COSMIC", Information and Software Technology 72 (2016) 90–109
4. C. Commeyne, A. Abran, R. Djouab. "Effort Estimation with Story Points and COSMIC Function Points - An Industry Case Study", Software Measurement News, Vol 21, No. 1, 2016 *
5. Comment on Linkedin discussion, September 4th, 2014