




INTRODUCTION TO THE COSMIC METHOD OF MEASURING SOFTWARE REQUIREMENTS: USE IN AGILE PROJECTS

UK COSMIC SIG Meeting

April 2016

Charles Symons

Agenda

- 
- A blue arrow pointing to the right, highlighting the first item in the agenda.
- Background to Functional Size Measurement (FSM) methods and their uses**
 - **The COSMIC FSM Method**
 - **Application of COSMIC sizing in Agile projects**
 - **Conclusions**

Objective 1: we want to measure and control software project activities

Delivery to budget & time:

- Actual vs. Estimated Cost
- Actual vs. Estimated Duration

Project speed

- **Size** / Duration



Project productivity

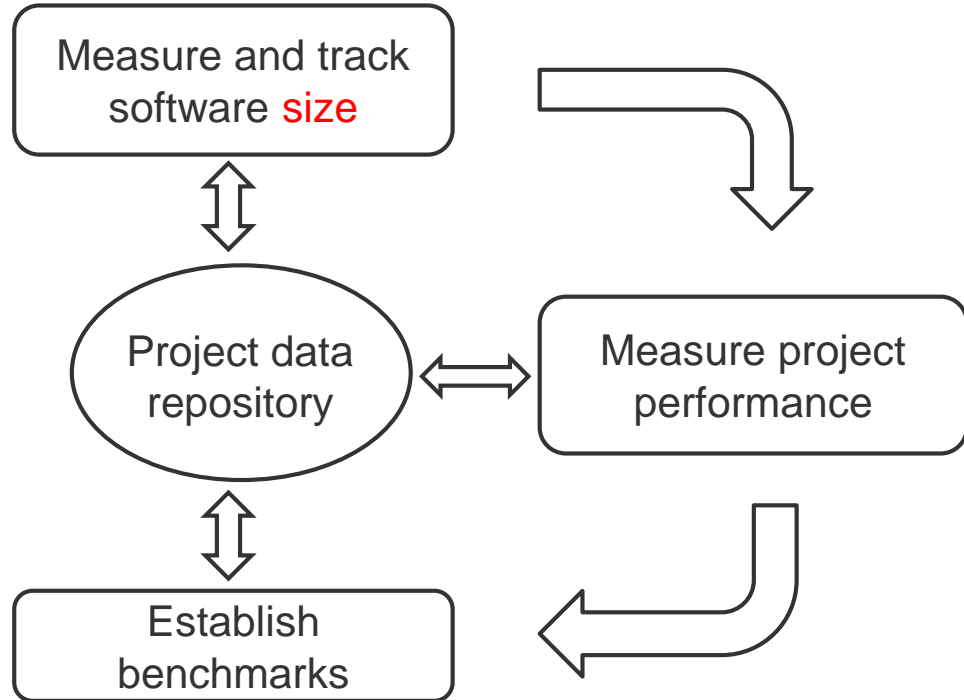
- **Size** / Effort

Product scope and quality

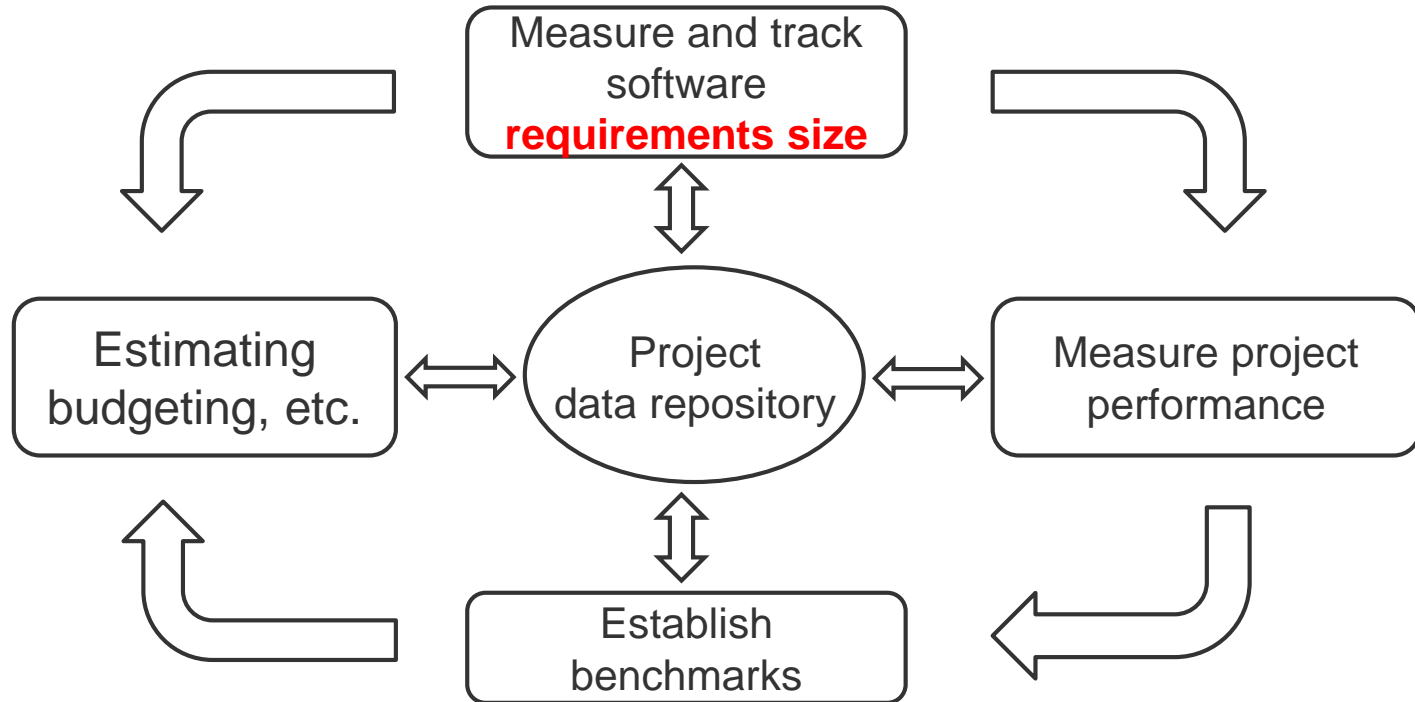
- **Size**
- Functional (e.g. business needs)
- Technical (e.g. maintainability, post-delivery defects density)

The performance of on-going **maintenance and enhancement** activities must also be considered

This can only be done by collecting data systematically



Objective 2: we want to use performance data for estimating future projects





A simple approach to effort estimation - for a project, iteration, etc

$$\text{Productivity} = \frac{\text{Software size}}{\text{Project effort}} \quad (\text{Use to establish benchmarks})$$

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

$$\text{Estimated new project effort} = \left\{ \frac{\text{Estimated software size}}{\text{Benchmark project productivity}} \right\} \times \left\{ \text{Adjustments for project-specific factors} \right\}$$



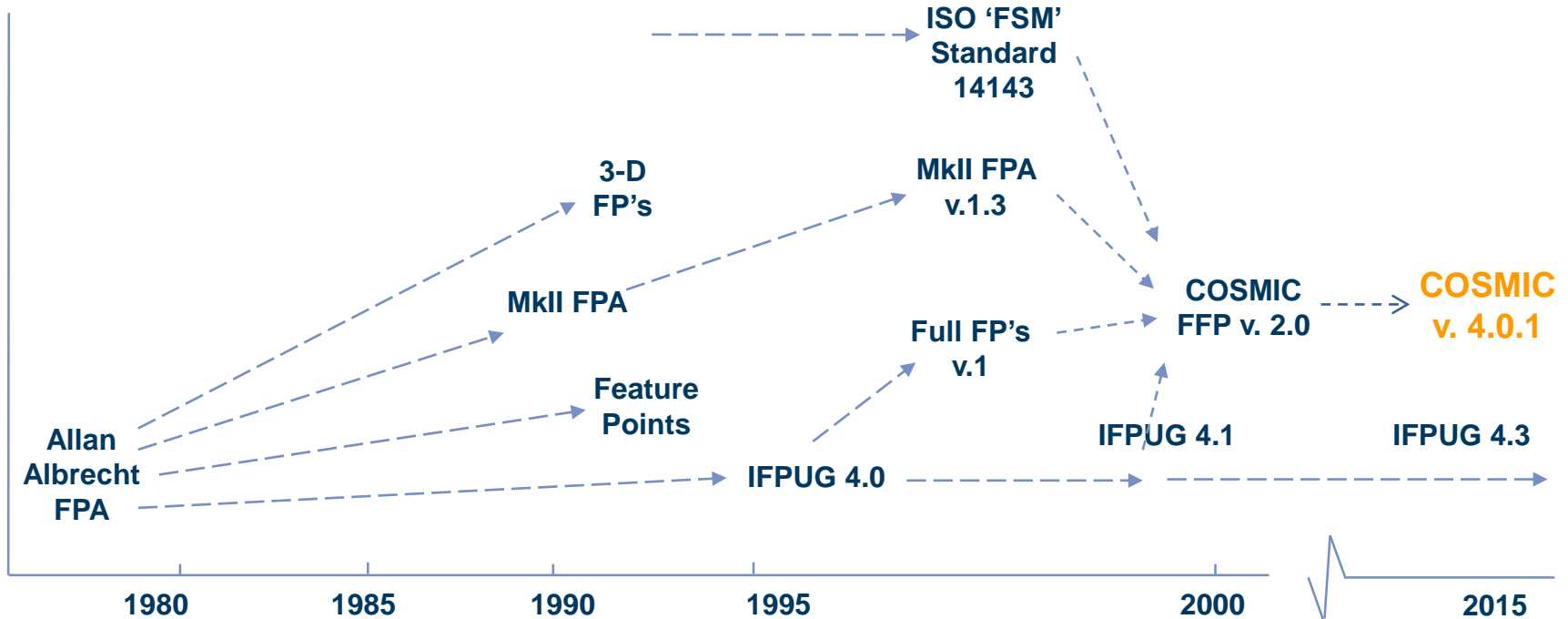
Only Functional size measurement (FSM) methods can help achieve both objectives

Sizing method options:

- | | |
|---|---|
| Counts of lines of code: | <ul style="list-style-type: none">✗ Can't estimate until software designed✗ Technology-dependent, no standards✓ Accounts for all requirements |
| Functional size: | <ul style="list-style-type: none">✓ International standard methods✓ Technology-independent<ul style="list-style-type: none">▪ What about 'Non-Functional' Requets? |
| Other sizing methods:
e.g. UCP, OOP, Story Pts. (?) etc: | <ul style="list-style-type: none">✗ No reliable standards; benchmark data possible only locally✗ What about 'Non-Functional' Requets? |



FSM Methods have been around for a long time





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

Functional User Requirements (FUR)

- what the software must do
- size can be measured by a FSM method

Non-Functional Requirements (NFR)

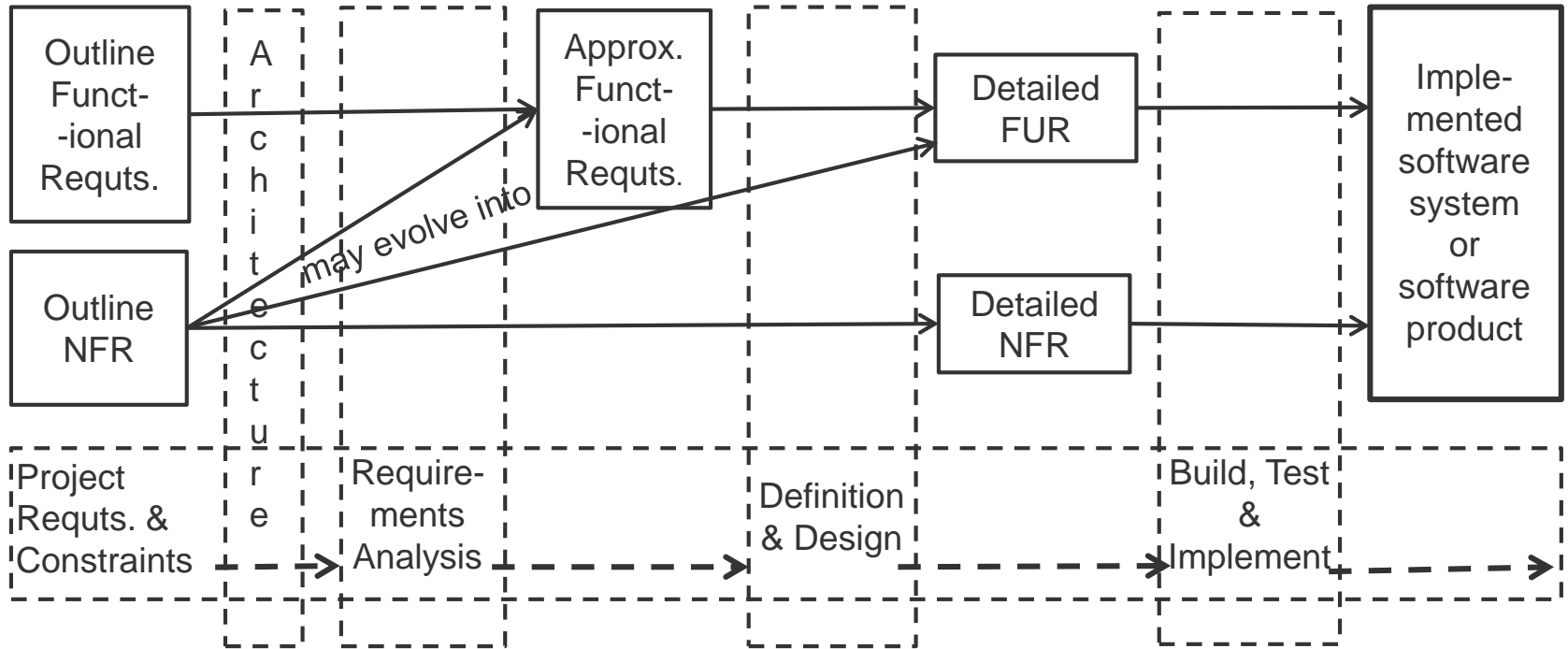
- quality, technical and environmental constraints, etc.

Project Requirements and Constraints (PRC)

- targets, processes & tools, resources, dependencies, etc



Requirements that first appear as NFR may evolve into FUR that the COSMIC method can measure

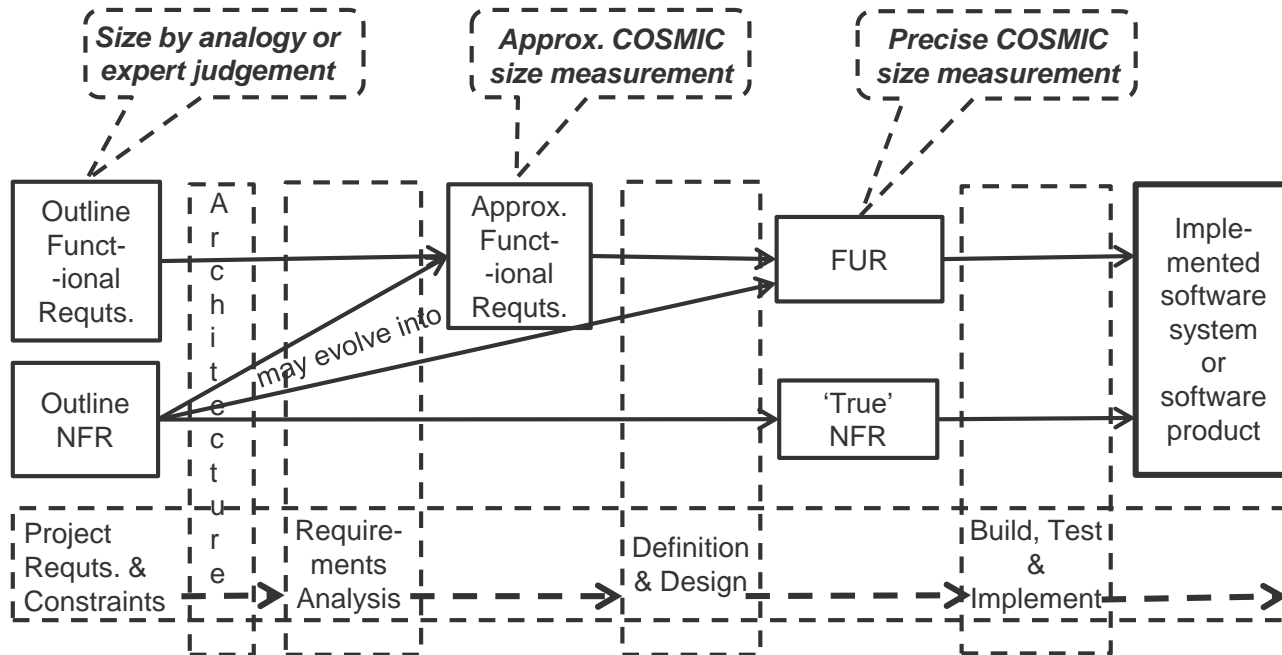


Examples of Quality NFR that evolve into FUR

Initial NFR	FUR that evolve from the initial NFR	Residual 'true' NFR
Auditability	Audit trails, special enquiries, etc.	Adequate documentation
Recoverability	Functionality to recover data & resume processing after faults	The specific recoverability target
Response time	Functionality to make data available in real-time	The response time target; fast hardware, etc.
Usability	GUI features; Help facilities	The specific usability target

.... and that can be measured

Total size estimation accuracy improves as the project progresses



Agenda

- Background to Functional Size Measurement (FSM) methods and their uses
- ➔ The COSMIC FSM Method
- Application of COSMIC sizing in Agile projects
- Conclusions

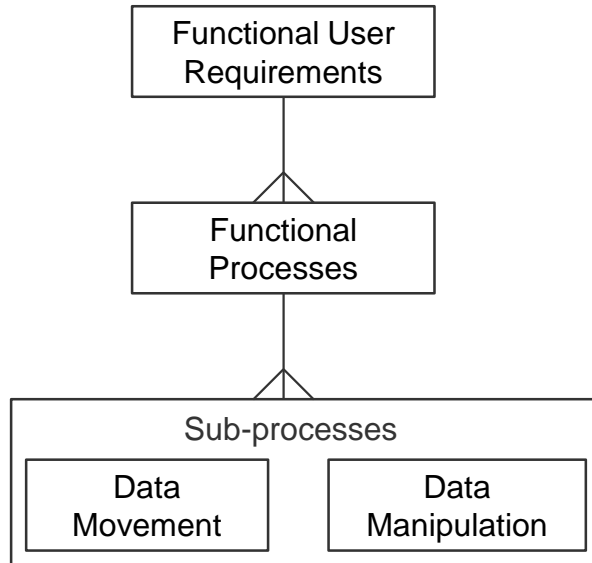


Goals of the COSMIC method for sizing software

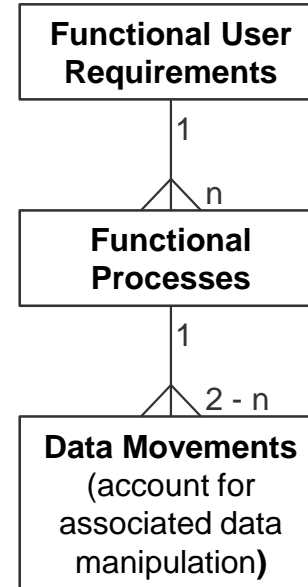
- A measure of Functional User Requirements based on fundamental software engineering principles
- Applicable to business, real-time and infrastructure software
- Completely independent of technology or processes used for the software or project
- Open, freely available, via www.cosmic-sizing.org

All software Functional User Requirements can be broken down into functional processes

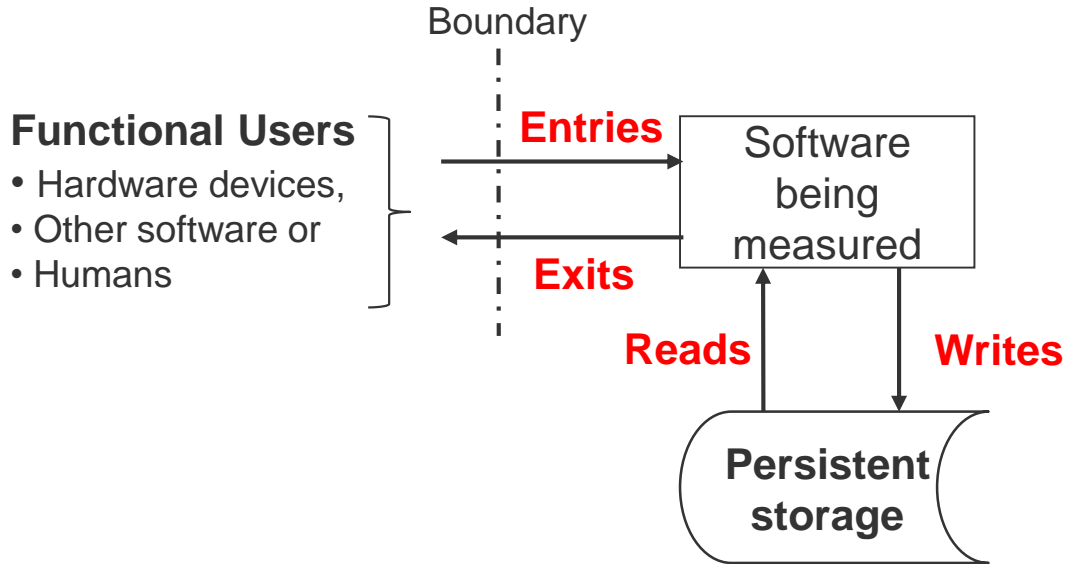
Theory:



In practice:

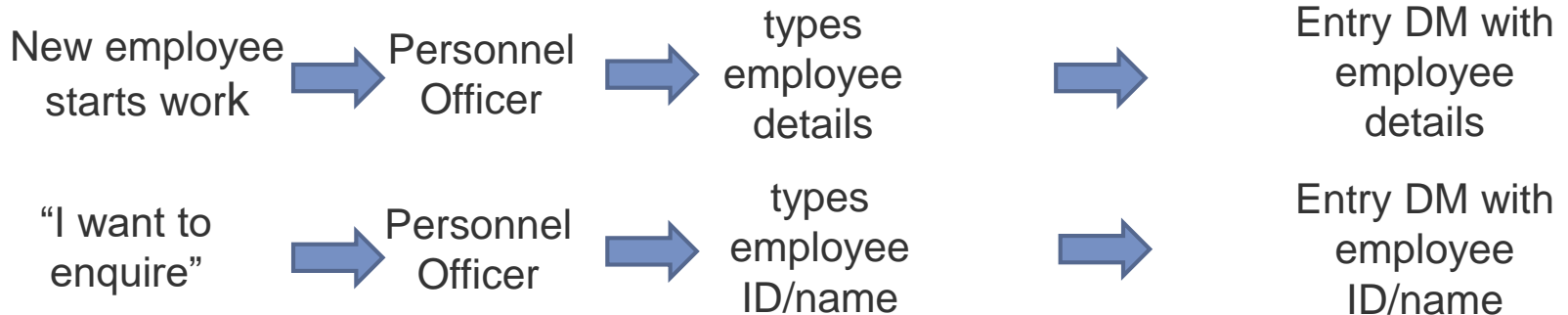
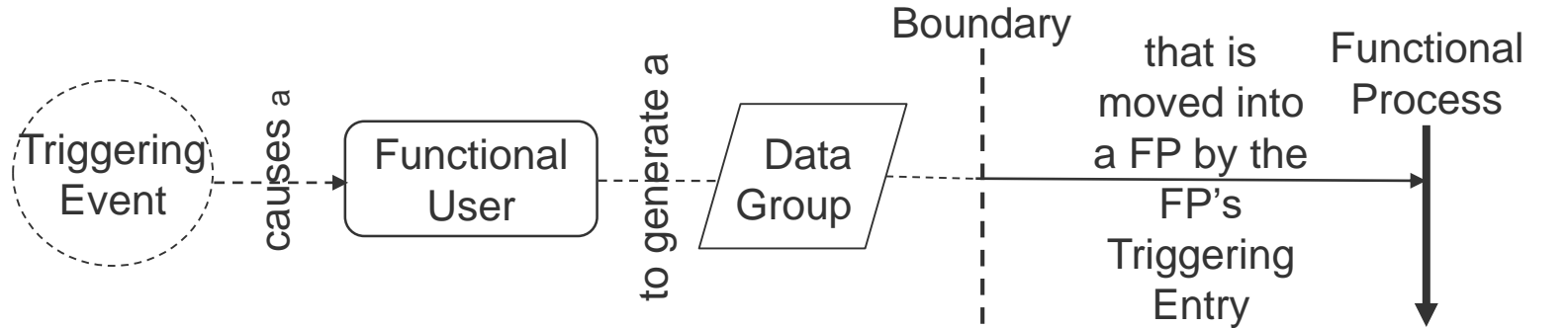


There are four types of 'Data Movement' sub-processes

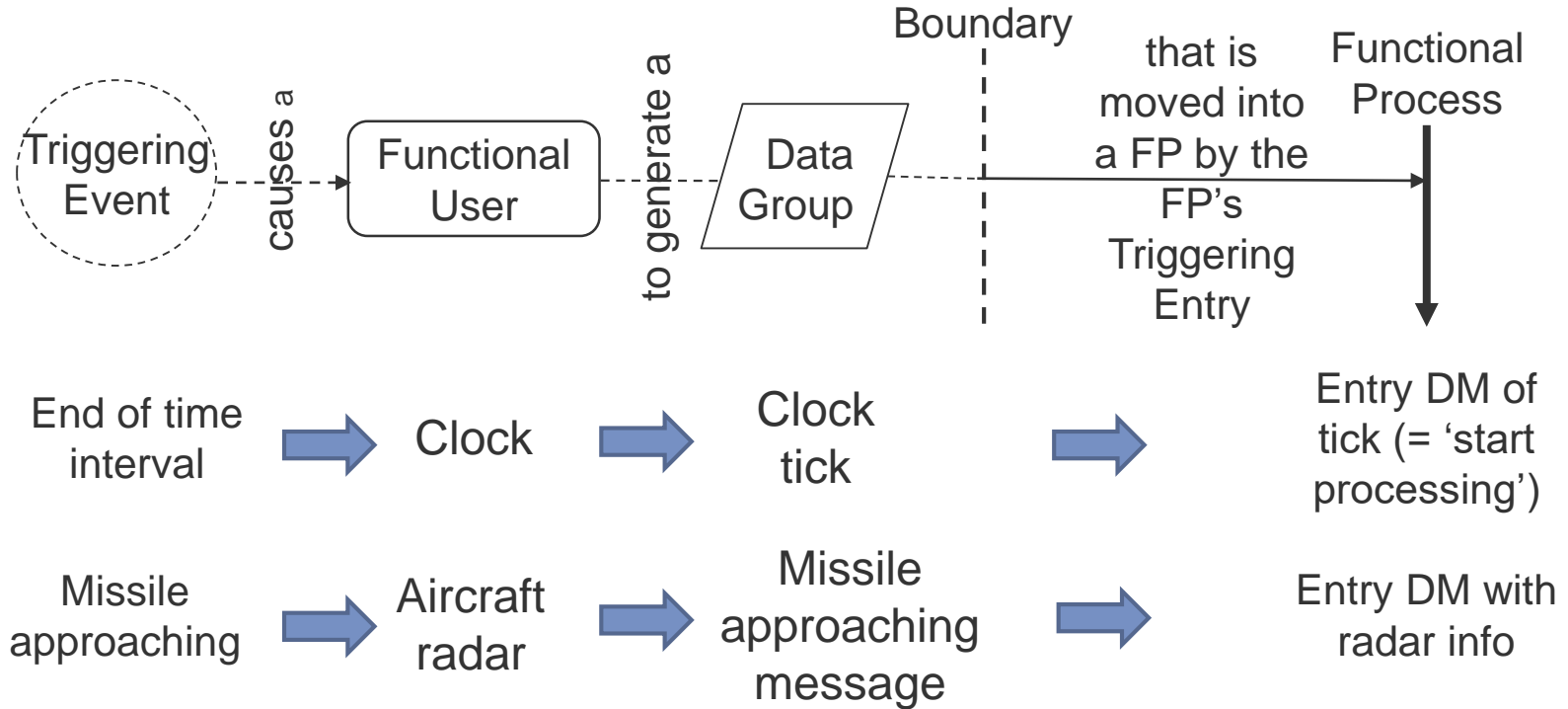


The 'Data Movement' is the unit of measure: 1 CFP (COSMIC Function Point)

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



Some real-time examples



Definition of a Functional Process (abbrev.)

- a) A set of data movements ... an elementary part of the Functional User Requirements (FUR) for the software being measured, that is unique within those FUR and that can be defined independently of any other functional process in those FUR.
- 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 FUR for all the possible responses to its triggering Entry.

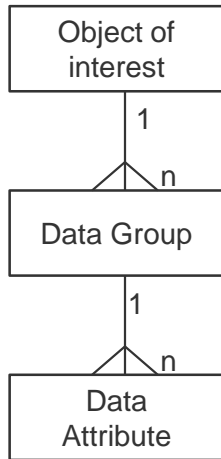
Some more definitions

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

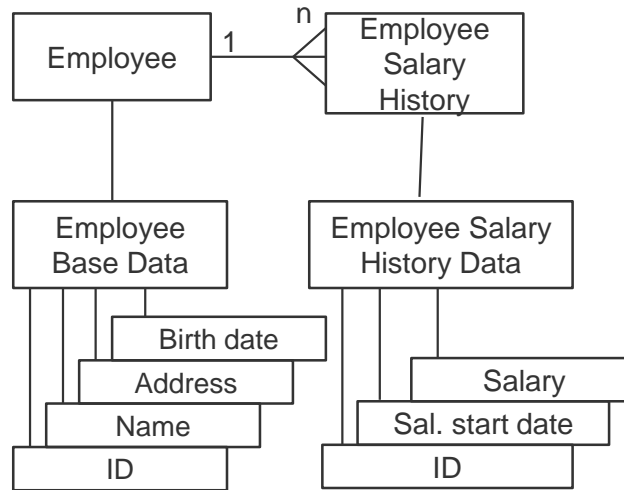
- A **data group** consists of one or more **data attributes** that describe a single **object of interest**
- An **object of interest** is any 'thing' (physical or conceptual) in the world of the **functional user**, about which the software being measured must process or store/retrieve data
- (A **data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

Examples of the data relationships

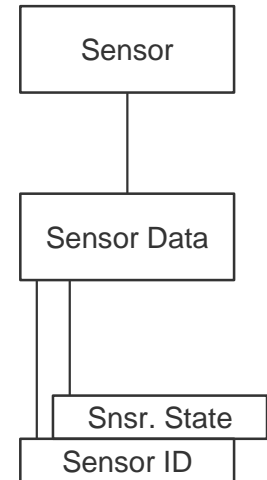
The model



Business Example



Real-time Example

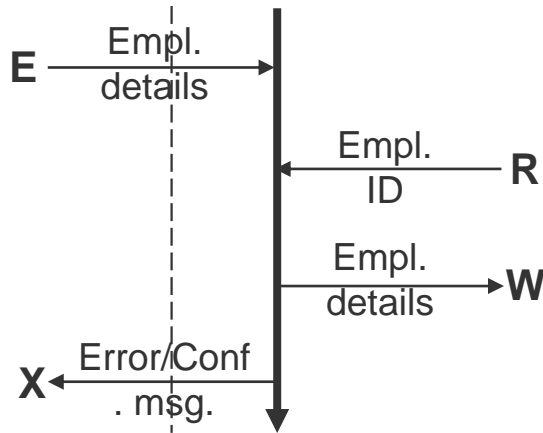




Some example business application Functional Processes

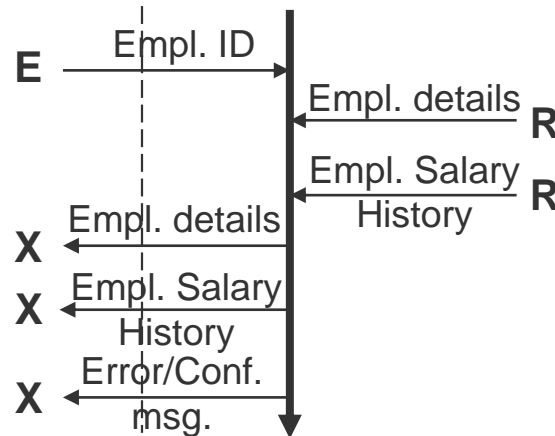
'Maintain' employee salary

Create Employee



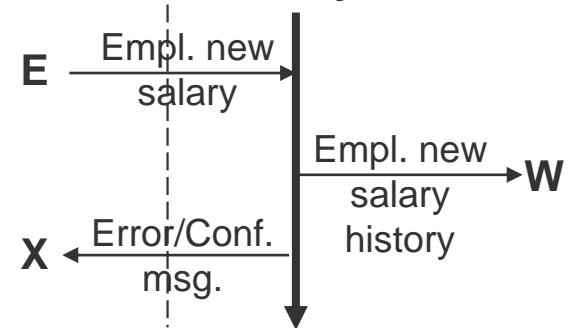
4 CFP

Enquire on current salary



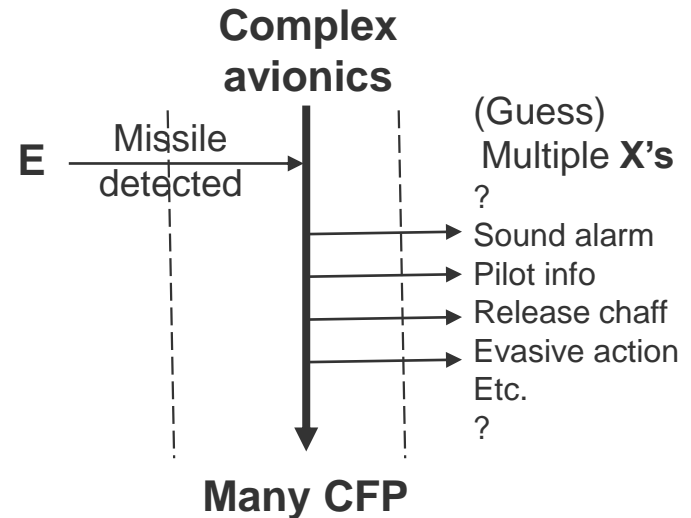
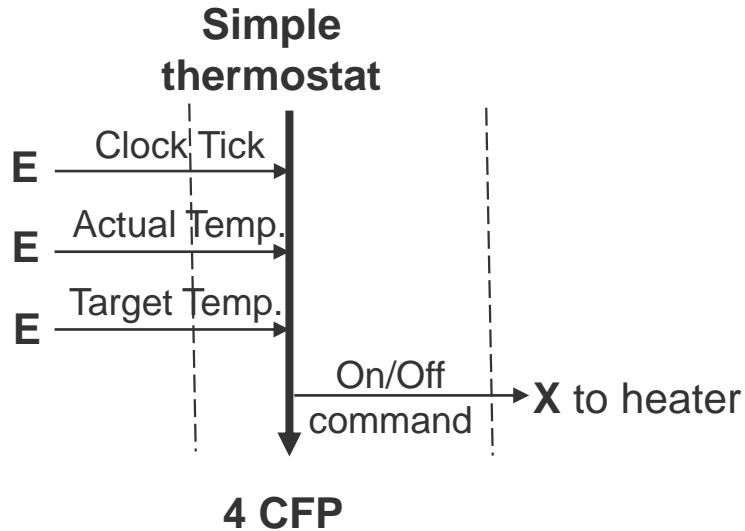
6 CFP

Update salary



3 CFP

Some example real-time Functional Processes

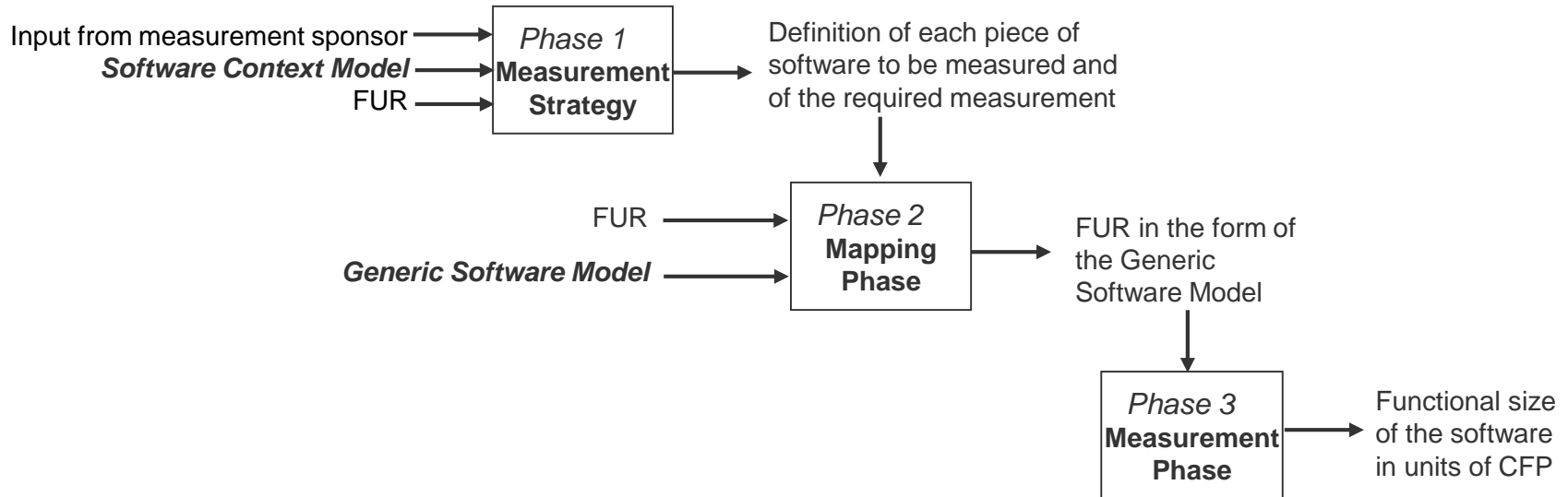


There is no upper limit to the size of a functional process

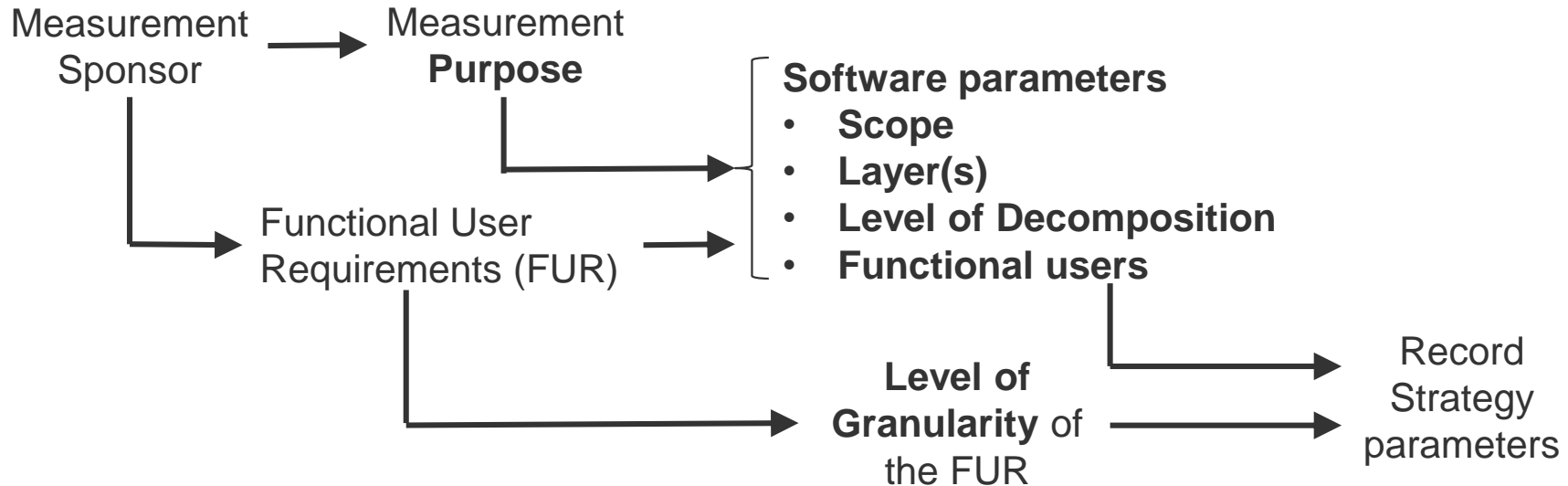
- A functional process must have at least 2 CFP
 - A triggering Entry
 - An 'outcome' – i.e. a Write or an Exit
- Largest observed functional processes?
 - In banking ~ 65 CFP
 - In avionics >100 CFP
- The smallest change to an existing functional process is 1 CFP



The COSMIC Measurement Process

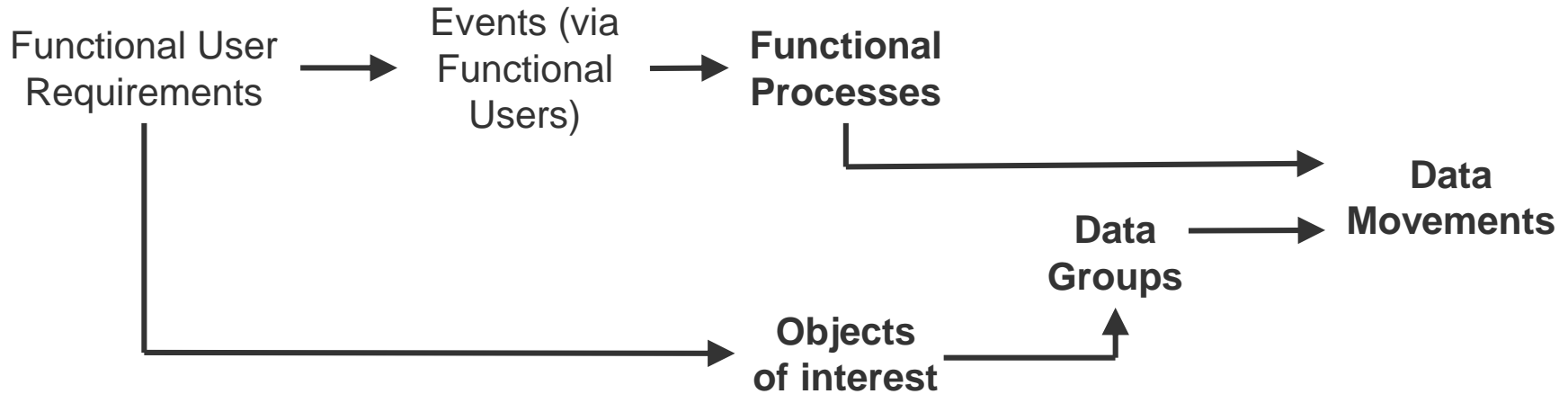


Measurement Strategy phase: define the parameters of the 'Software Context Model'



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

Mapping phase: determine the parameters of the 'Generic Software Model' from the FUR





An example output from the Mapping Phase

Acme Car Hire Functional Processes	Data Group Names										Nos. of Data Movements					
	Customer name	Customer Master Record	Customer name and address	Customer ID	Customer Summary Details	Customer Details	Customer latest Invoice	Existing 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



Measurement phase: count the data movements

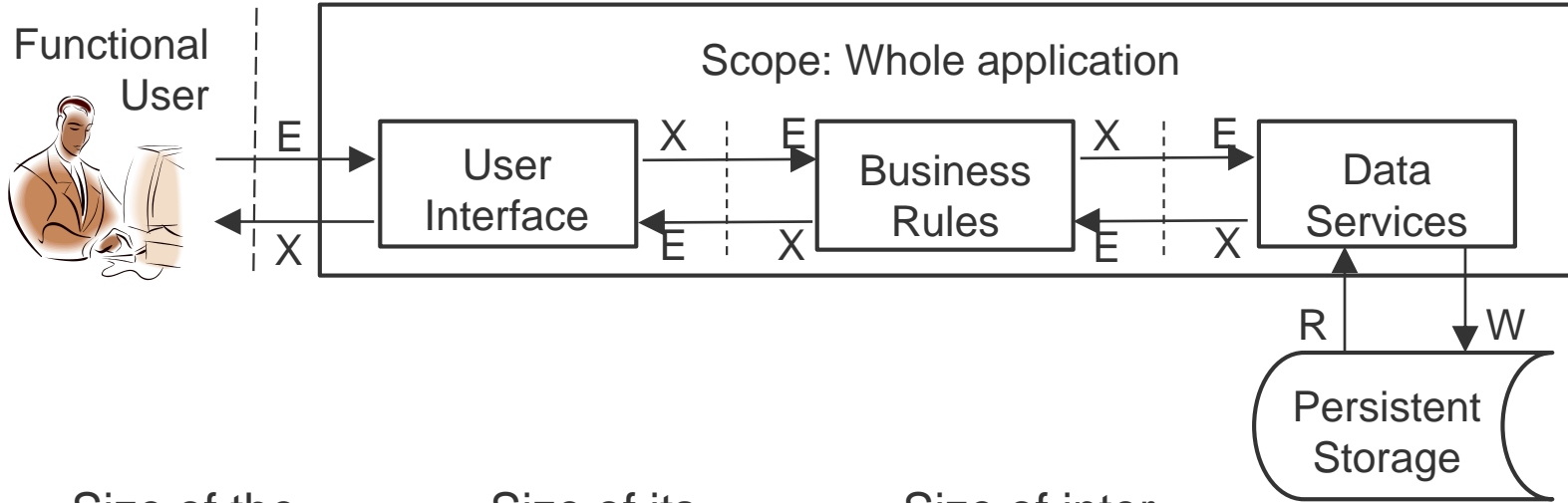
Within a defined Measurement Scope:

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

Size of software after change = Size of software before change plus Count of DM's added less Count of DM's deleted

Sizes may be aggregated if sensible, and subject to one rule



Size of the 'whole' software = Size of its components less Size of inter-component DM's



The acid test: can COSMIC-measured sizes be used for prediction?

Renault ¹⁵ 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

'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



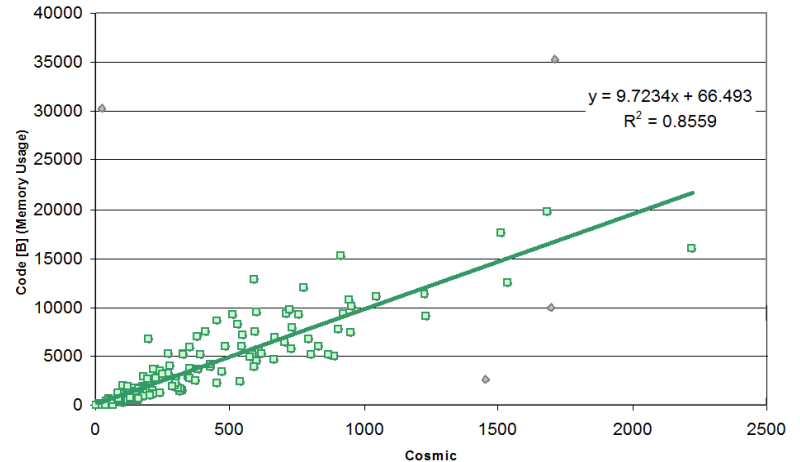
Renault achieves remarkable cost estimation accuracy from its ECU designs



Cost vs size (CFP)



Memory size vs software size (CFP)





Summary

The COSMIC method is a major advance on other FSM methods in terms of

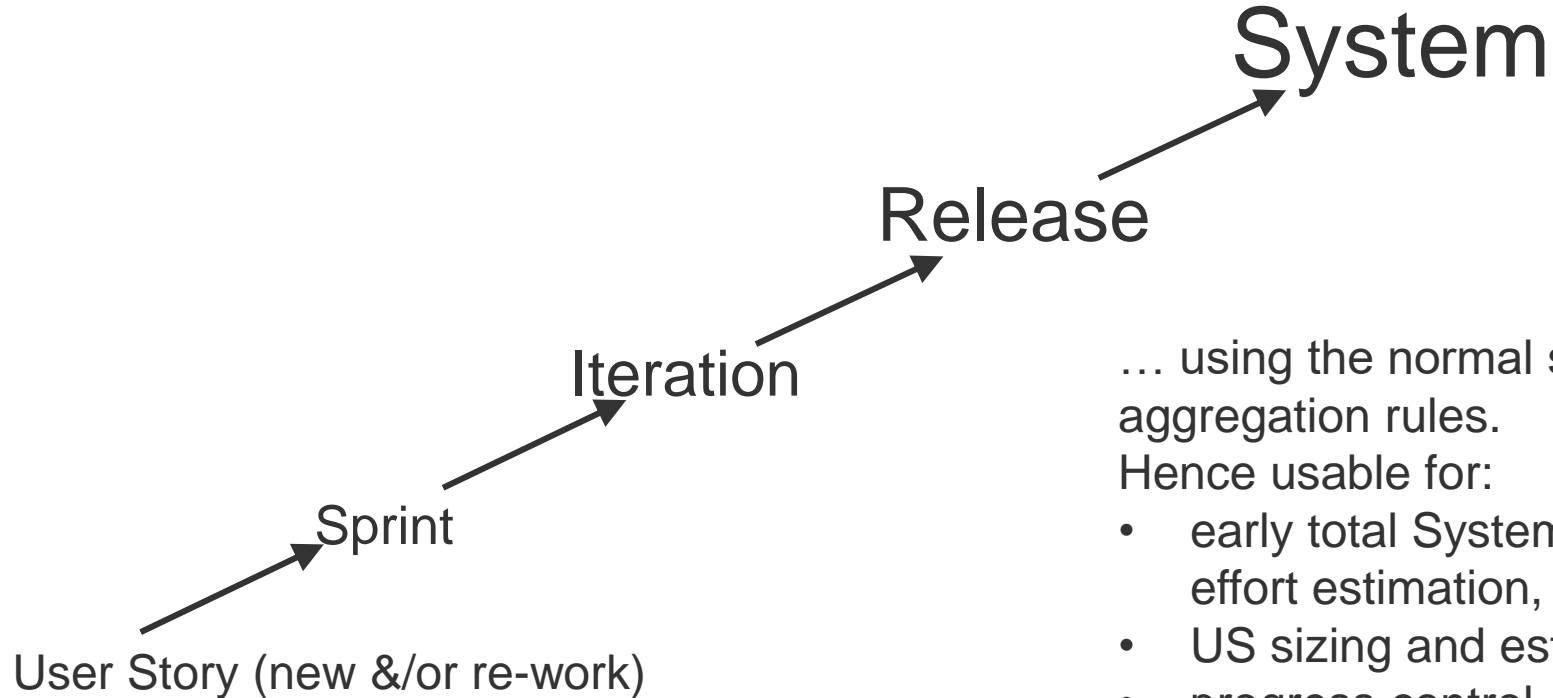
- Use of fundamental SE principles, aligned with modern development methods
- Scope of applicability
- Measuring functional sizes, totally independent of effort, that nevertheless correlate very well with effort



Agenda

- Background to Functional Size Measurement (FSM) methods and their uses
- The COSMIC FSM Method
- ➔ Application of COSMIC sizing in Agile projects
- Conclusions

COSMIC sizes can be measured at any level of aggregation ...



... using the normal size aggregation rules.

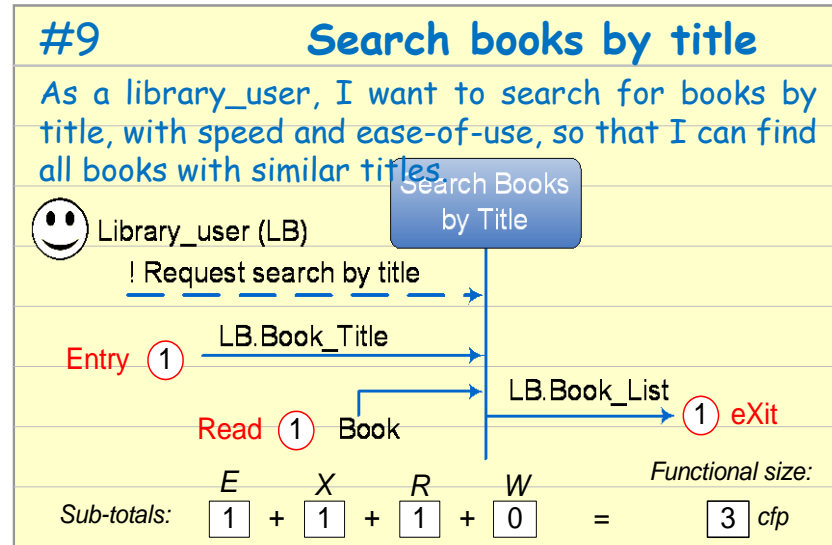
Hence usable for:

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

Each Sprint develops a number of User Stories

General form of a User Story (US)

“As a <user type>,
I want to <feature or functionality>,
so that <value or expected benefit>”.



List of tests on back:
FUR & NFR



A User Story may or may not correspond to a COSMIC functional process

Ideally:

1 x User Story describes 1 x Functional Process ('FP')

In practice:

1 x User Story may describe

- more than 1 x FP (and then later split into multiple US)
- less than 1 x FP because e.g.
 - the US describes e.g. one physical screen
 - or, early in an iteration, all the FUR are not known in detail
- anything (e.g. a non-functional requirement)



The challenge for using COSMIC in Agile projects is 'what & when to measure?' (not 'how')

Customer concern

- a) What is the size accepted as delivered?
- b) Exclude re-work due to Supplier errors

Supplier concern

- c) What is the size the Customer has asked to be delivered?
- d) Include re-work due to Customer changing the requirements

Options for dealing with this (needs discussion!)

- At what level(s) do you record sizes (sprint, iteration, etc.)?
- Measure and pay for size a) + d)?
- Measure and pay for size a) + effort for d), subject to check on size of d)

For Benchmarking

- Measure final delivered size, or size of all changes (for enhancement projects)

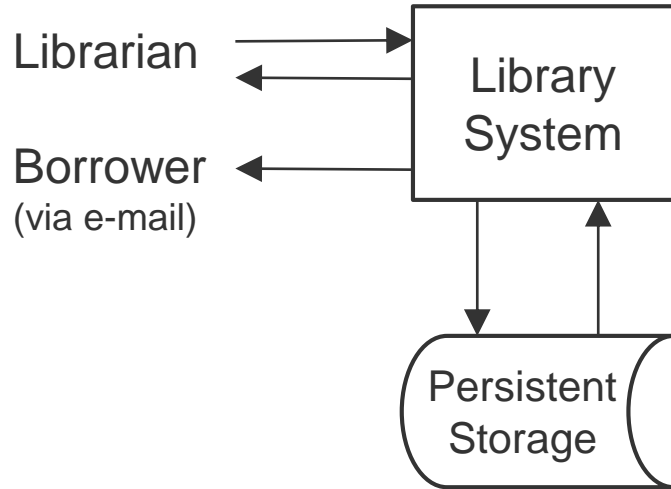
Exercise: Discuss and measure the following US

I want to

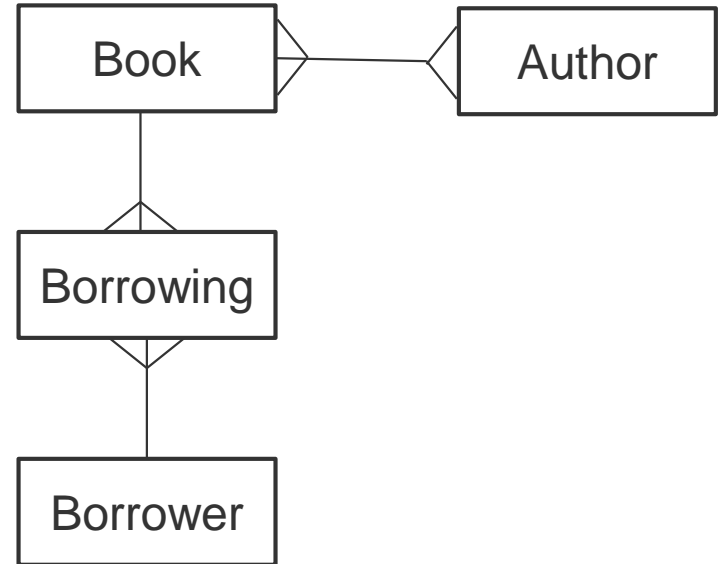
1. Create and maintain a Book Catalogue for a Library
2. Add an Author to the Library system
3. Add a Book to the Catalogue and enable a link to its Authors
4. Search for all the Books in the Catalogue by a given Author
5. Add a Borrower to the Library System
6. Enable a Librarian to record a Borrowing
7. Send an e-mail to a Borrower about an overdue Book

Exercise: What can we generally determine from the requirements?

The system 'Context Diagram'



The objects of interest

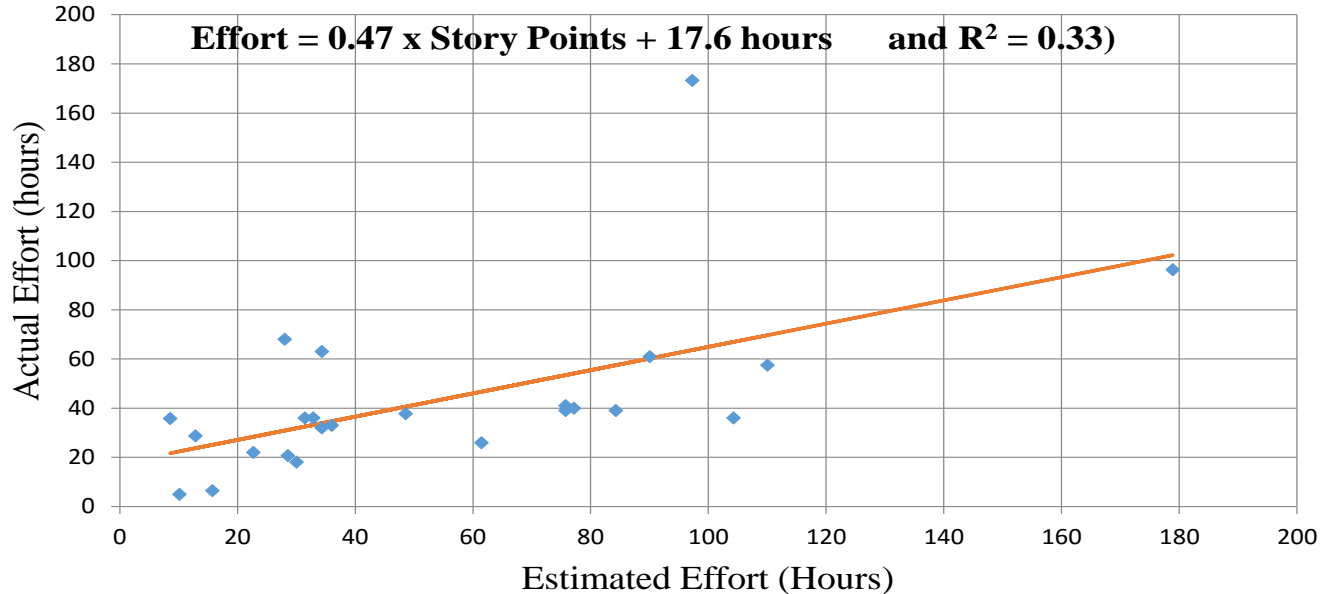




Case history: A Canadian supplier of security and surveillance software systems

- A customer request for new or changed function is called a ‘task’
- Uses Scrum method; iterations last 3 – 6 weeks
- Teams estimate tasks within each iteration in USP, and convert directly to effort in work-hours (this is not considered good Agile practice)
- Study* involved measurements on 24 tasks in nine iterations
 - Each task estimated in USP, then effort; actual effort recorded
 - Each task also measured in CFP

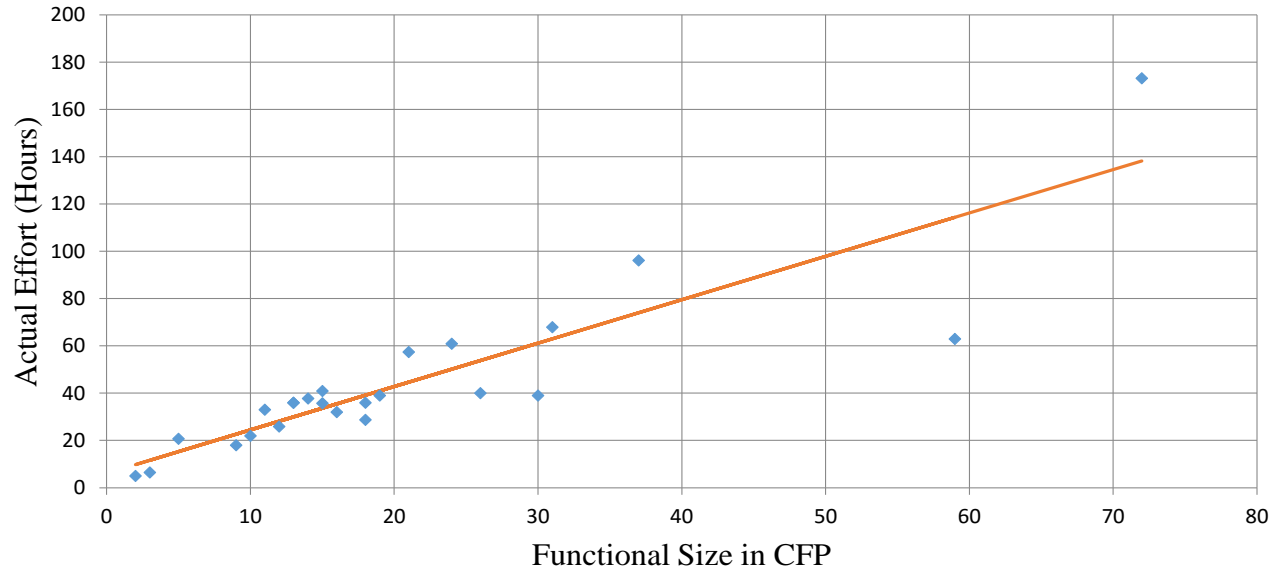
A best-fit straight line would be a poor predictor of effort from USP



Notice the wide spread and the 17.6 hours 'overhead'

The Effort vs CFP size graph (24 tasks) shows a good fit, but there are two outliers

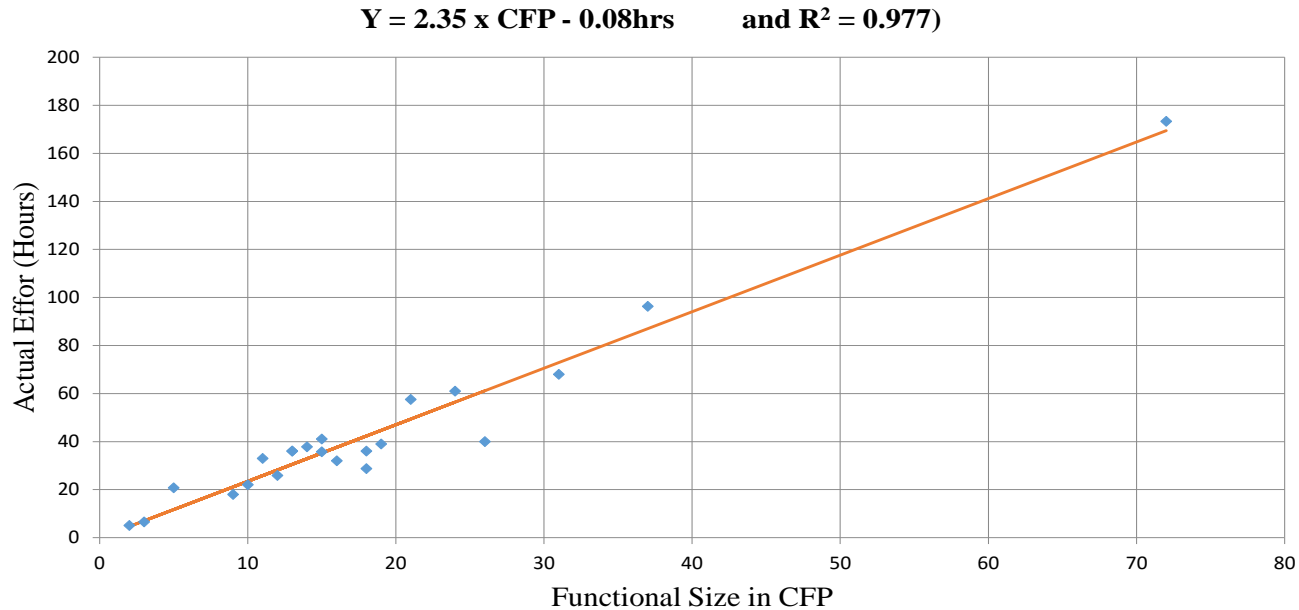
Effort = 1.84 x CFP + 6.11 hours and $R^2 = 0.782$



The two projects with low effort/CFP were found to involve significant software re-use, so were rejected as outliers



Now we have a good effort vs CFP correlation (22 tasks), usable for predicting effort





USP v CFP for Agile projects: Summary

User Story Points

- In practice, a subjective measure of relative size
- Meaningful only within a project team
- Not usable for estimating total System effort
- No guidance on how to deal with NFR

COSMIC Function Points

- An objective measure of functional size
- An ISO Standard. Sizes meaningful across projects
- Usable at all levels (US, sprint, up to whole System)
- Method advises how to deal with NFR

USP and CFP require the same effort to use



A User view of 'COSMIC for Agile'

- *“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

- Background to Functional Size Measurement (FSM) methods and their uses
- The COSMIC FSM Method
- Application of COSMIC sizing in Agile projects



Conclusions



The COSMIC method has comprehensive support

- Measurement Manual standard (11 languages)
- Guidelines (business, real-time, SOA, DWH, NFR, approximate sizing, use in Agile projects, etc.)
- Case Studies
- Forums (on LinkedIn CUG, www.cosmic-sizing.org)
- Certification
- Support and automatic measurement tools



Summary of benefits

- Free, open
- Based on fundamental SE Principles, future-proof, stable
- Very wide applicability
- Proven value for performance measurement and estimating
- ISO standard
- GAO¹, NIST² endorsed

1) 'Cost Estimating and Assessment Guide' <http://www.gao.gov/new.items/d093sp.pdf> , March 2009

2) 'A Rational Foundation for Software Metrology', National Institute for Standards & Technology, NIST IR 8101, January 2016

Thank you for your attention

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