




# INTRODUCTION TO THE COSMIC METHOD OF MEASURING SOFTWARE REQUIREMENTS

Charles Symons

- 
- The COSMIC Method design principles and measurement process**
    - **Setting the Measurement Strategy**
    - **Mapping and Measurement phases**
    - **Conclusions**

# COSMIC method design objectives

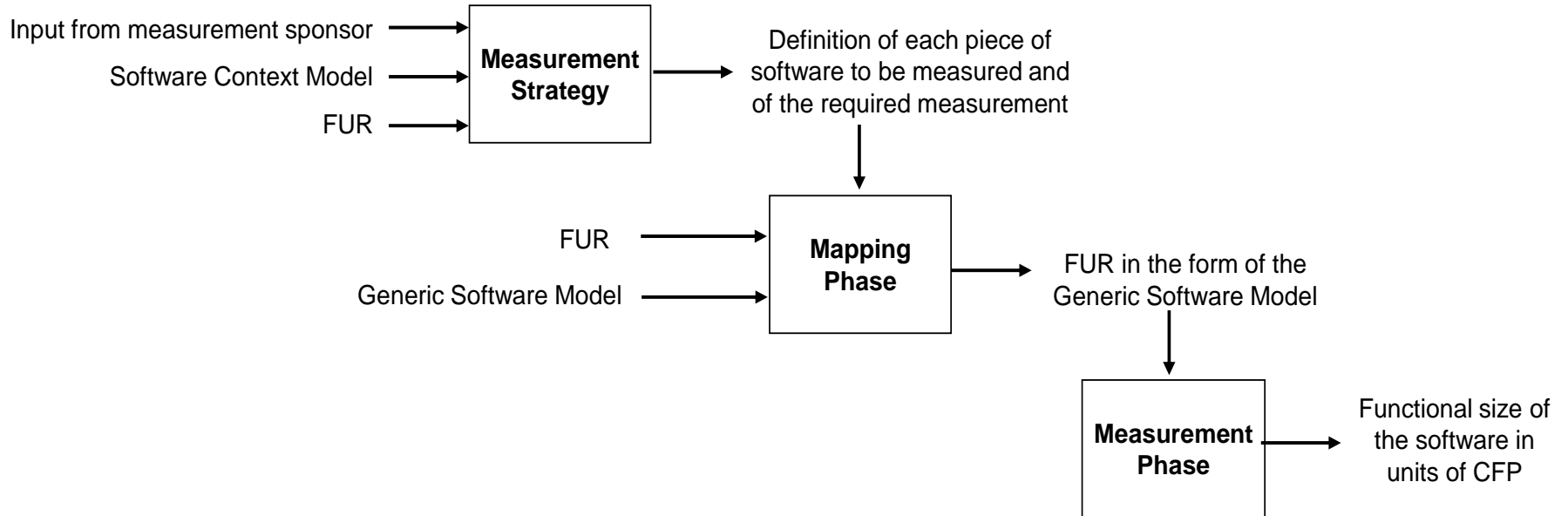
To measure a size of Functional User Requirements based on fundamental software engineering principles

- applicable to business, real-time and infrastructure software
- at any level of decomposition (from a whole software system down to e.g. a User Story or re-usable component)
- independent of the technology or processes used to develop the system, but measured sizes must correlate well with development effort
- with approximate sizing variants for use in early project estimation
- with guidance on how to deal with Non-Functional Requirements

# The COSMIC method was designed based on two sets of Principles

- The ‘Software Context Model’  
helps define the software to be measured
- The ‘Generic Software Model’  
the model to which FUR must be mapped for measurement

# The COSMIC Measurement Process



# Agenda

- The COSMIC Method design principles and measurement process
- ➔ **Setting the Measurement Strategy**
  - Mapping and Measurement phases
  - Conclusions

# The Software Context Model - essentials

- Software is bounded by hardware and typically structured into **layers**.
- The **scope** of any piece of software to be measured shall depend on the **purpose** of the measurement and shall be confined wholly within a single layer.
- The **functional users** of a piece of software to be measured shall be identified from its FUR as the senders and/or intended recipients of data to/from the software respectively.
- A precise COSMIC size measurement of a piece of software requires that its FUR are known at a **level of granularity** at which its **functional processes** and sub-processes may be identified.
- An approximate COSMIC size measurement is possible if its FUR are measured at a high level of granularity by an approximation approach and scaled to the level of granularity of the functional processes.

# The Strategy phase parameters are important because of the COSMIC method's flexibility

- Essential to define the parameters:
  - to be sure you're measuring what the Sponsor needs
  - and so that future users of the measurement understand its meaning and how it may be used
- In practice you will probably need only a few recurrent 'Measurement Patterns'

*(see 'Guideline for Measurement Strategy Patterns. Ensuring that COSMIC size measurements may be compared.')*

# First, some terminology

## ***Level of Granularity (of the Requirements)***

*Any level of expansion of **the description of the requirements for a single piece of software** such that at each increased level of expansion, the description of the functionality of the piece of software is at an increased and uniform level of detail.*

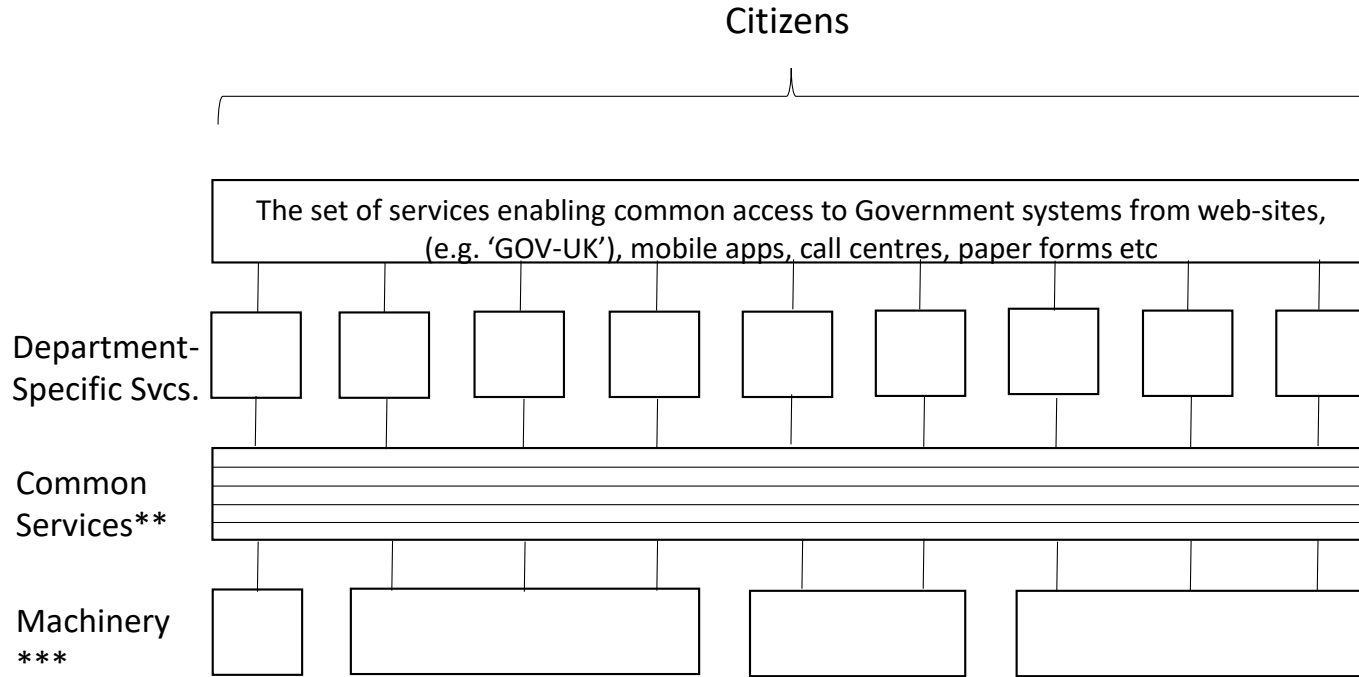
(The LoG of the available requirements will determine the possible measurement accuracy)

## ***Level of Decomposition (of the Software)***

*Any level resulting from **dividing a piece of software into components** (named 'Level 1', for example), then from dividing components into sub-components ('Level 2'), etc.*

# Defining the layers

## Example: A modern layered architecture\*



Loosely copied from 'The Gubbins of Government' ([www.markfoden.com](http://www.markfoden.com)), with permission.

\* UK Government Digital Service

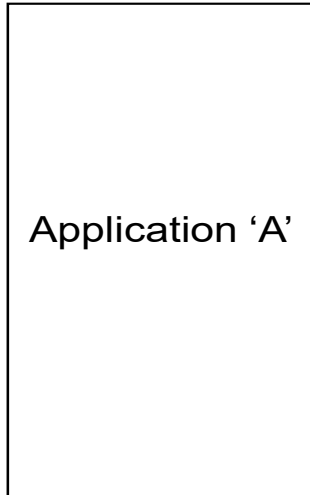
\*\* Examples: collecting and paying money, or checking identity

\*\*\* The computer hardware and data storage facilities, communication networks, etc realised in the Cloud

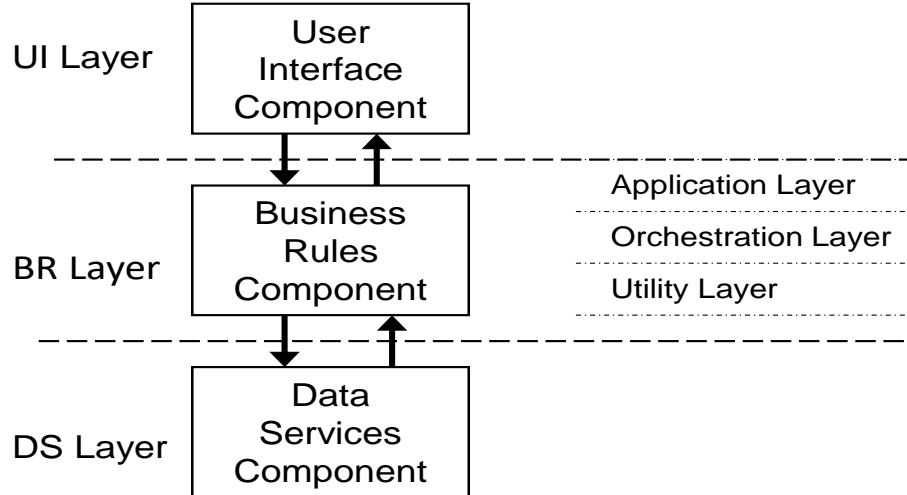
# Different architectures have different 'views' of the layers

a) View of an application 'A' as a whole

Application Layer



b) Application 'A' components in a 3-layer Architecture



c) Layers for SOA components of Business Rules

Application Layer

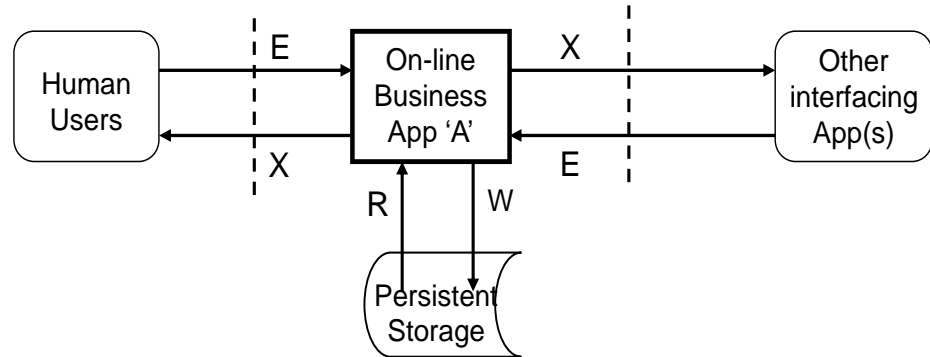
Orchestration Layer

Utility Layer

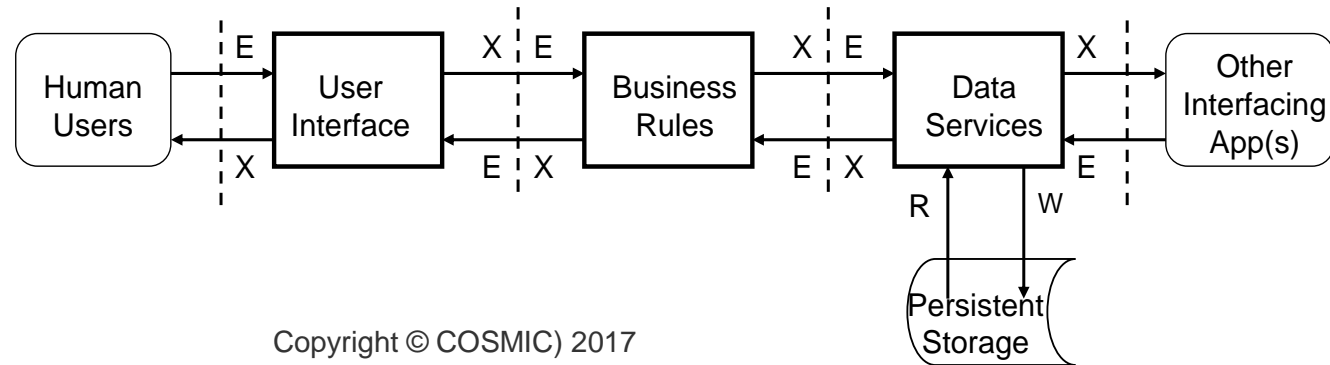
# Defining the measurement scope.

## Example: the 'whole' or separate components?

**Purpose:** measure a 'whole' online business application



**Purpose:** measure the components of a 3-tiered online business application



# Define the functional users - the 'senders or intended recipients of data'

**A Human Functional User** 'sees' a few control buttons, the display and the paper movements.

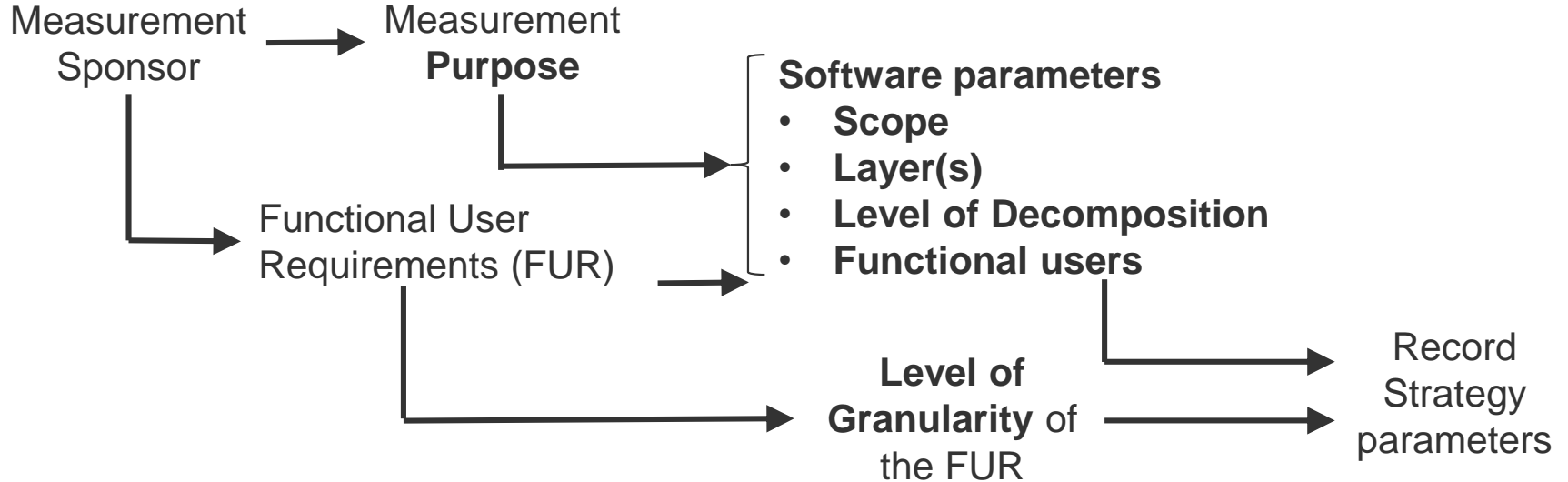


**Hardware Functional Users** are the power supply, sensors, buttons, motors, ink control, lights, paper jam detection that interact directly with the software

Different Functional Users 'see' different functional sizes.

So we must define the Functional Users - depending on the **Purpose** of the measurement

# Summary: define and record the parameters of the 'Software Context Model'



# Measurement Strategy discussion exercises

Define the measurement scope(s), layer(s), possible functional users, LoD and LoG, for the following Purposes.

“We need to estimate the effort to develop a new distributed business application with user access via the web. The app will exploit some re-usable components. We will need a program to convert data from the old system”

“We contracted on a fixed price/CFP with a supplier to develop the software to drive a new model washing machine that can be linked to the IoT for remote operation. We need to measure the CFP size delivered”

# Measurement Strategy parameters do NOT need to be set for each individual measurement

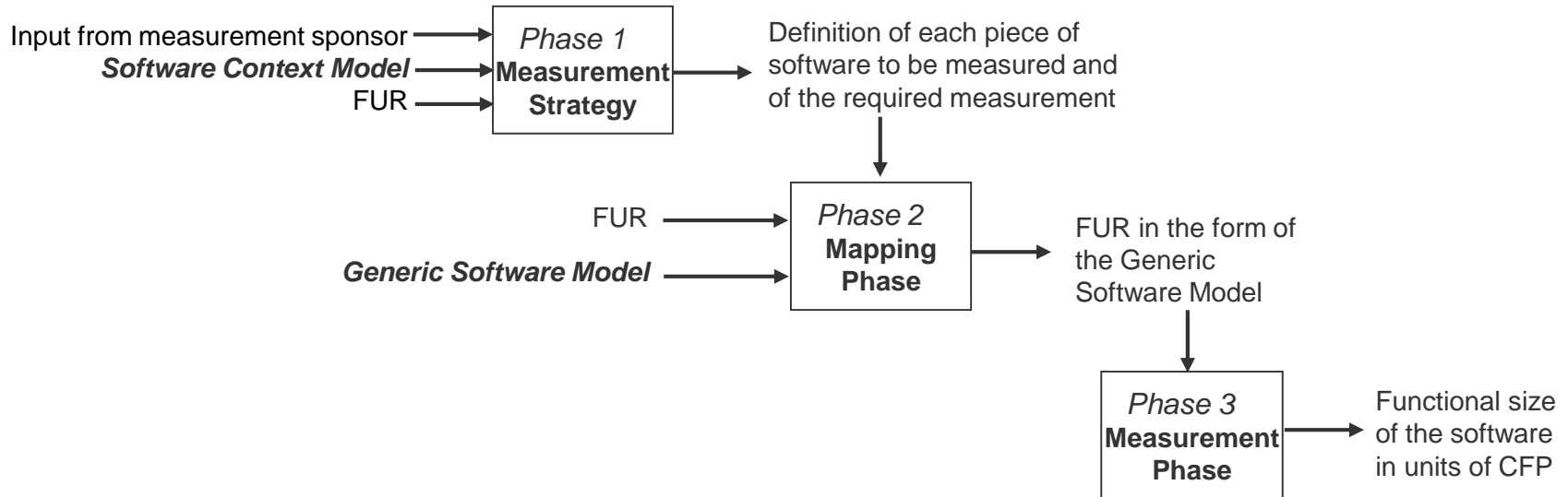
**Recommendation:** define ‘patterns’ of standard Measurement Strategy parameter sets for use in your environment

- Purpose, e.g. to measure: initial estimated size, or delivered total size, or User Story sizes, etc.
- Layers / Level of decomposition -> Scope
- Functional user types

# Agenda

- The COSMIC Method design principles and measurement process
- Setting the Measurement Strategy
- ➔ Mapping and Measurement phases
- Conclusions

# The COSMIC Measurement Process

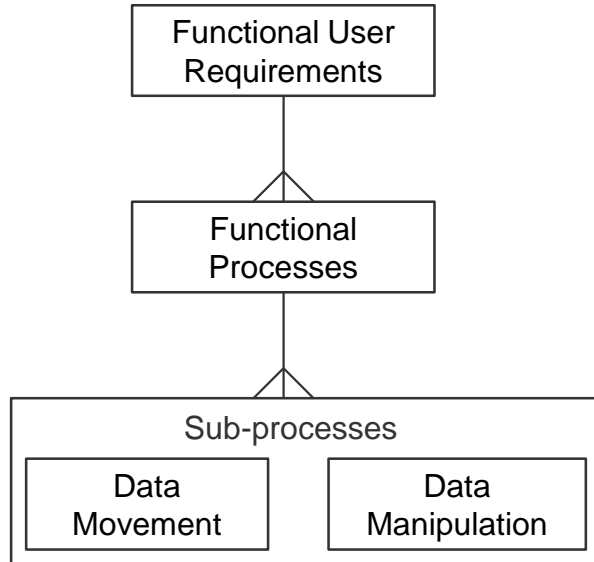


# The Generic Software Model - essentials

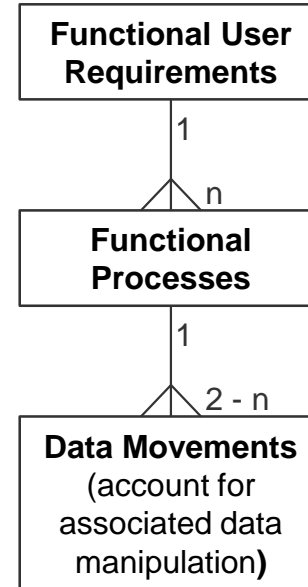
- A piece of software interacts with its functional users across a **boundary**, and with **persistent storage** within the boundary.
- The FUR of a piece of software can be mapped into unique **functional processes**.
- Each functional process consists of sub-processes, **data movements** and **data manipulations**.
- As an approximation for measurement purposes, the COSMIC method assumes that the functionality of any data manipulation is accounted for by the data movement with which it is associated.
- There are four data movement types, **Entry, Exit, Write and Read**.
- A data movement moves a single **data group**, which consists of a unique set of **data attributes** that describe a single **object of interest**.
- Each functional process is started by its **triggering Entry** data movement. The data group moved by the triggering Entry is generated by a functional user in response to a **triggering event**.
- A functional process shall include at least one Entry data movement and either a Write or an Exit data movement. There is no upper limit to the number of data movements in a functional process

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

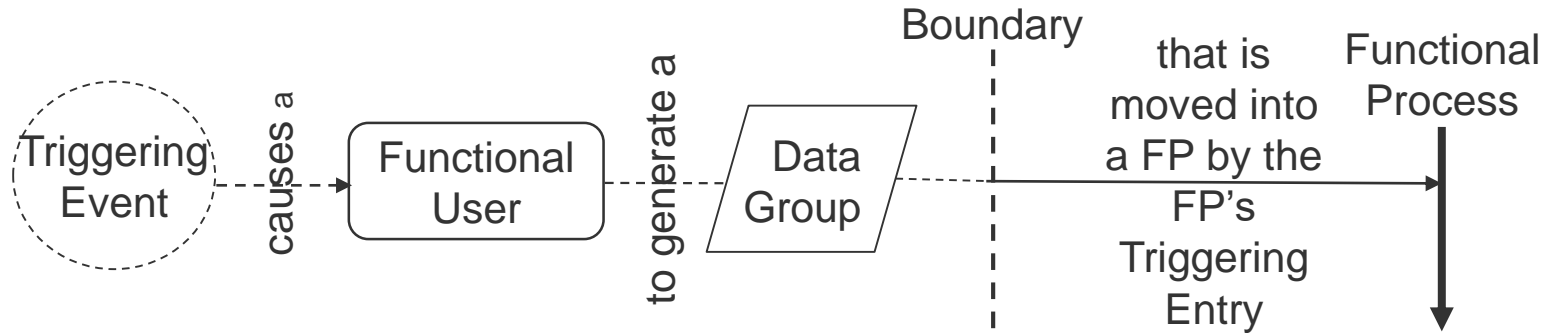
Theory:



In practice:

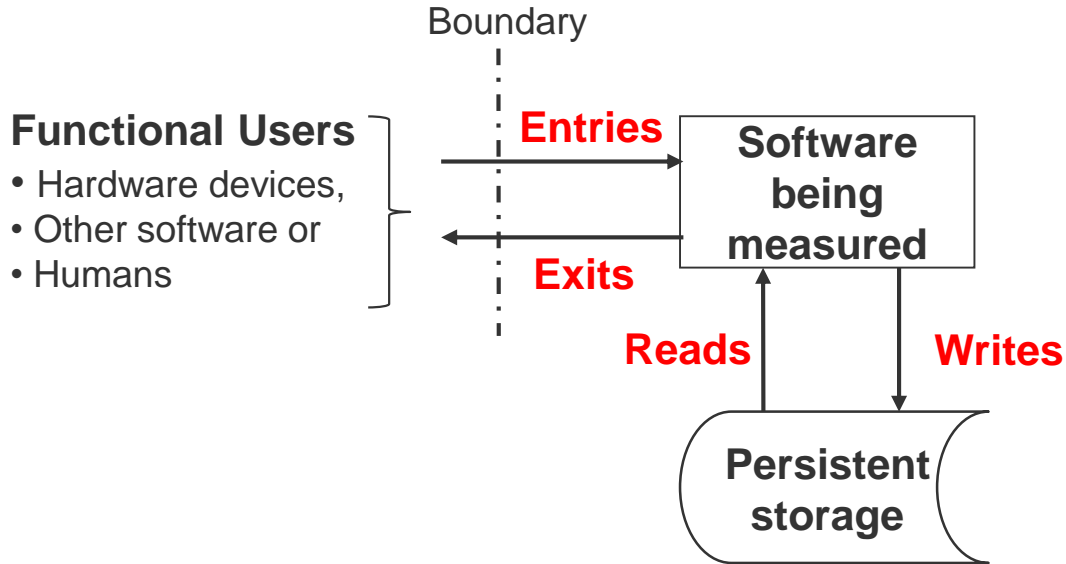


# The most important diagram in the method – the key to distinguishing functional processes



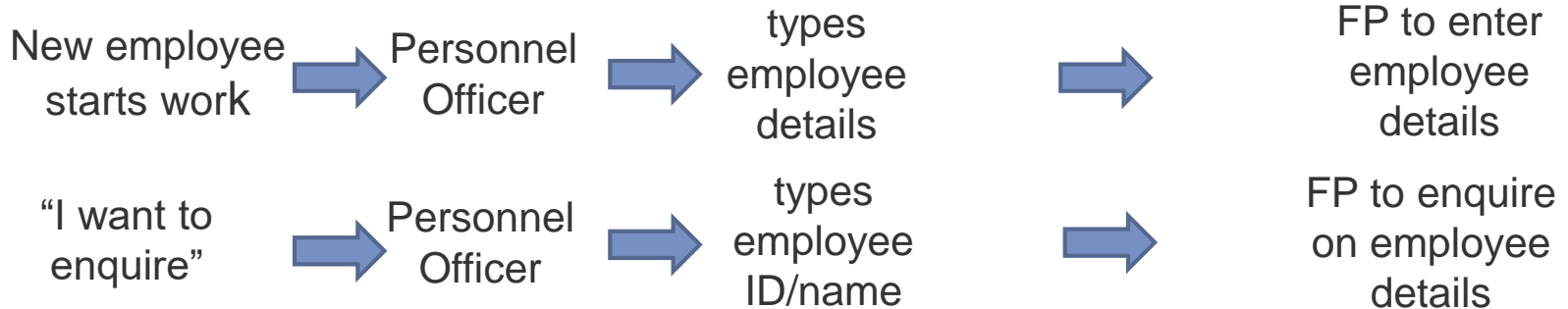
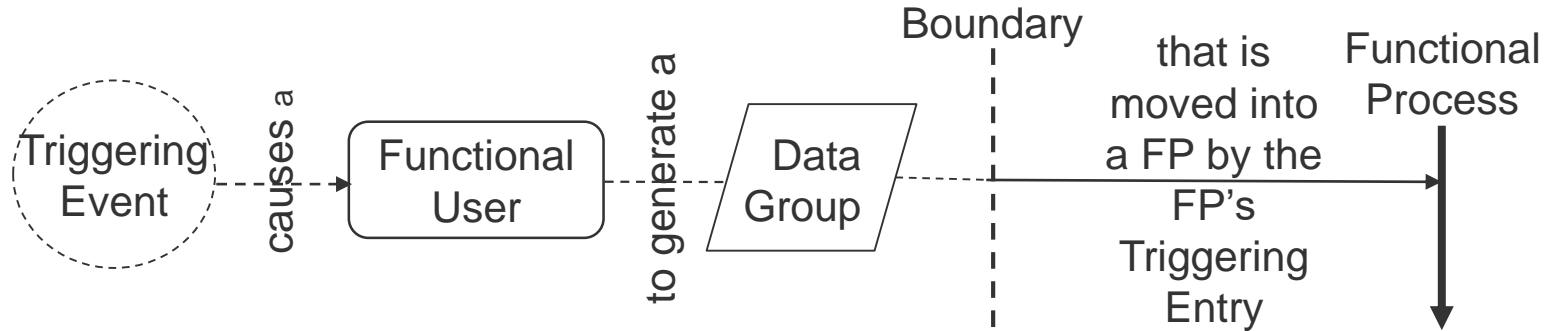
N.B. There can be many-to-many relationships at each stage along this chain, except that any one functional process may have only one triggering Entry

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

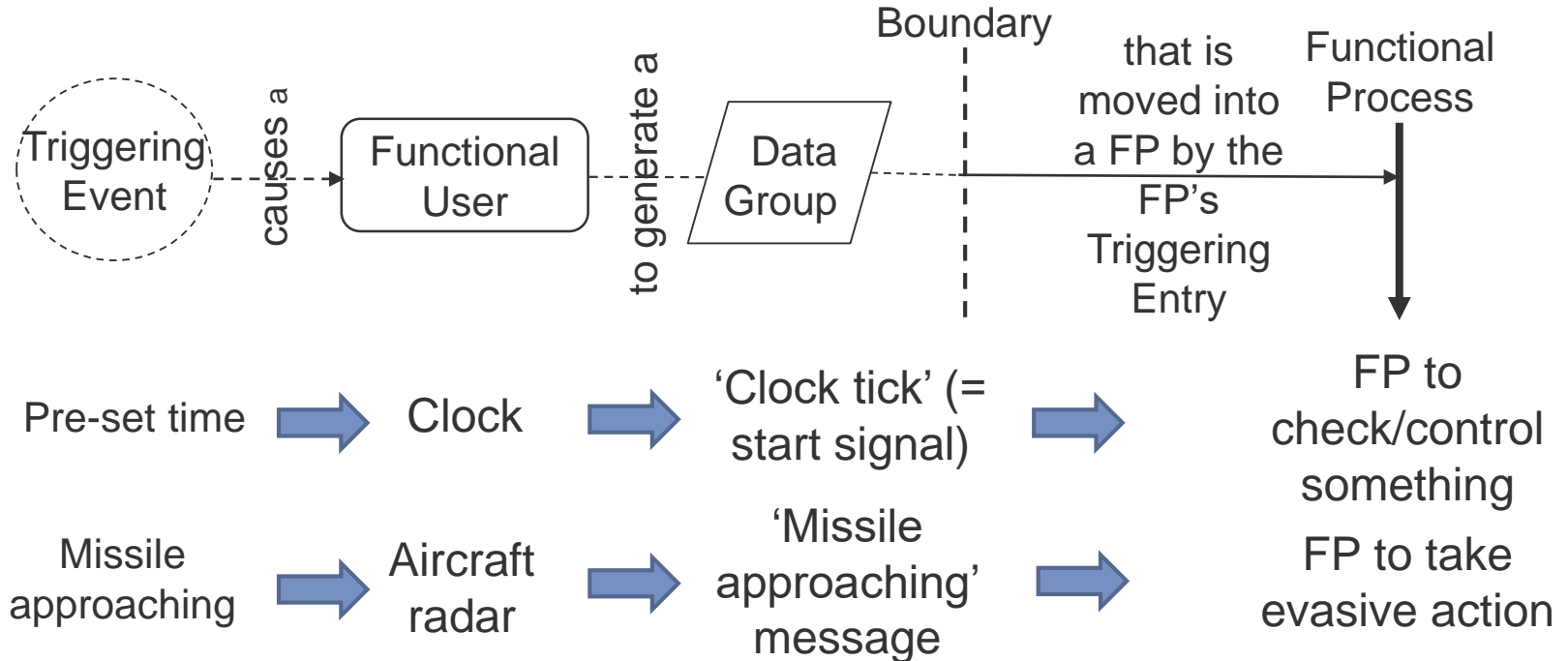


The 'Data Movement' is the unit of measure: 1 CFP (COSMIC Function Point)

# 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.

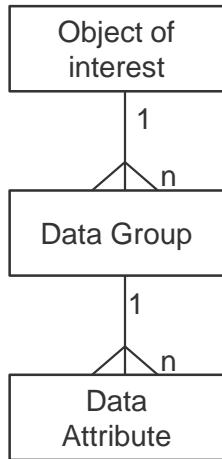
# Some more definitions

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

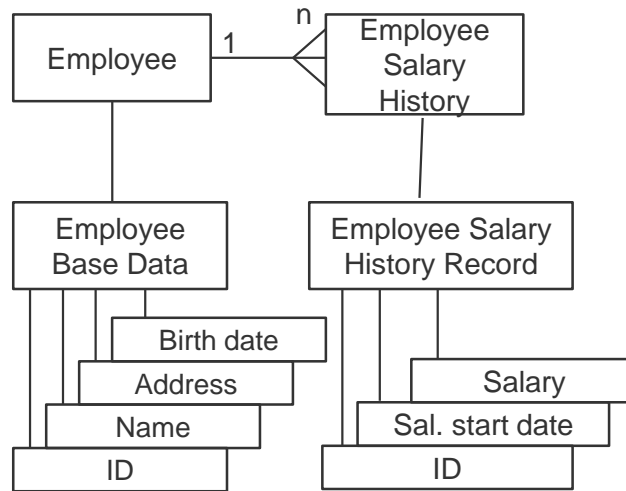
- A **data group** consists of one or more **data attributes** that all describe a single **object of interest**
- An **object of interest** is any ‘thing’ (physical or conceptual) in the world of the **functional user**, about which the software being measured must process or store/retrieve data
- (A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

# Examples of the data relationships

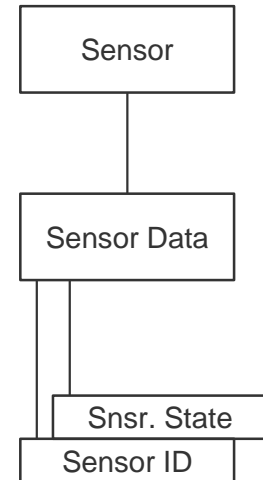
**The model**



**Business Example**

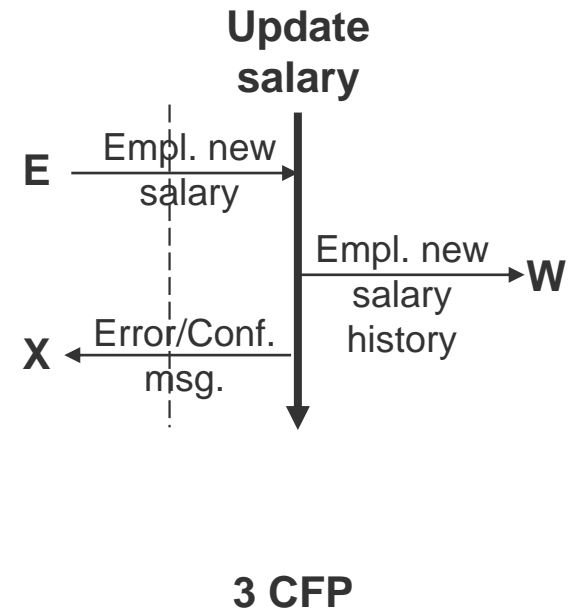
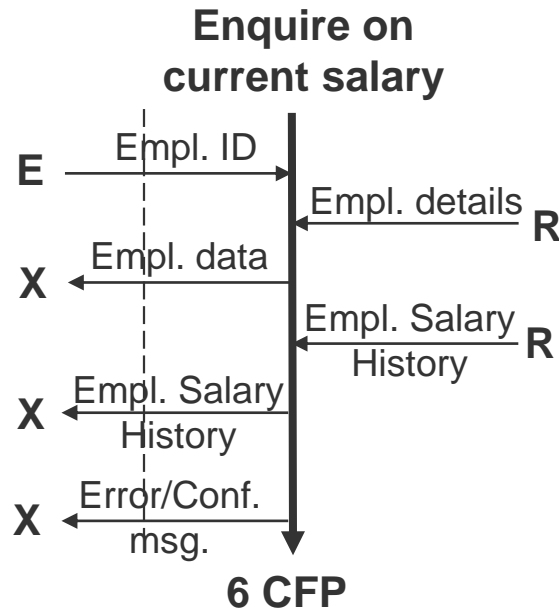
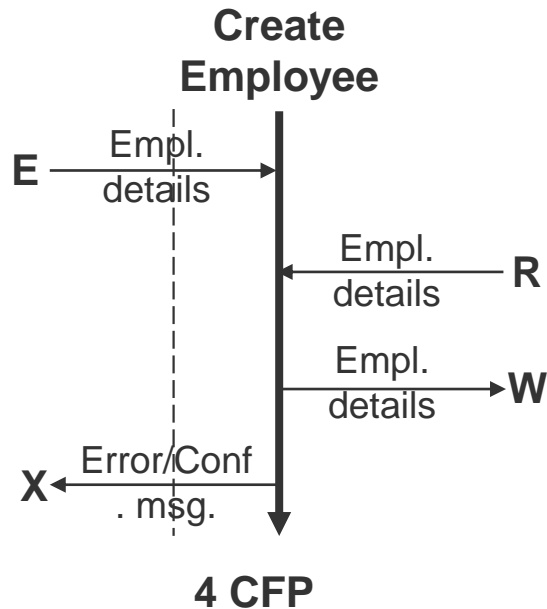


**Real-time Example**

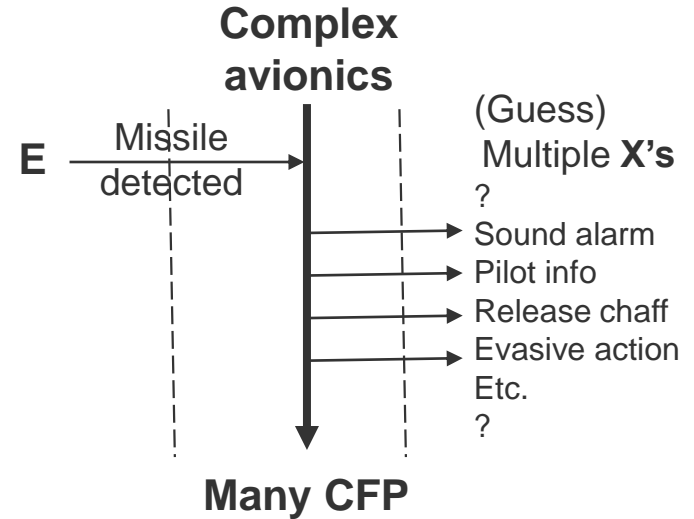
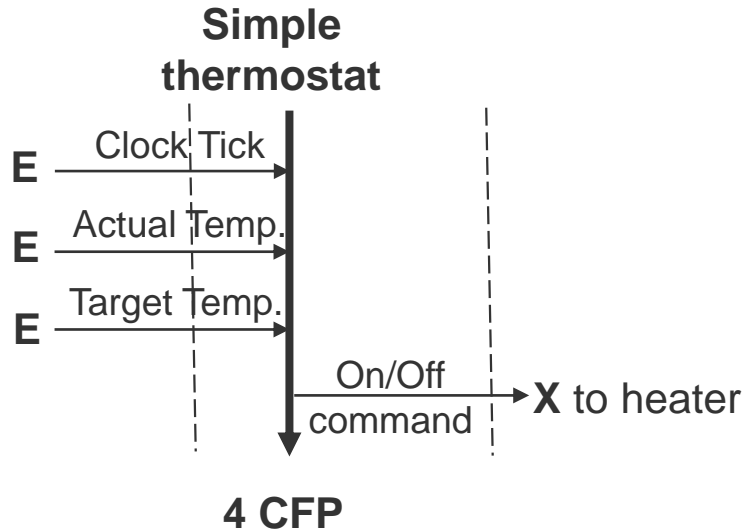


# Some example business application functional processes

## 'Maintain' employee salary



# Some example real-time functional processes



# Example FUR – discussion exercises

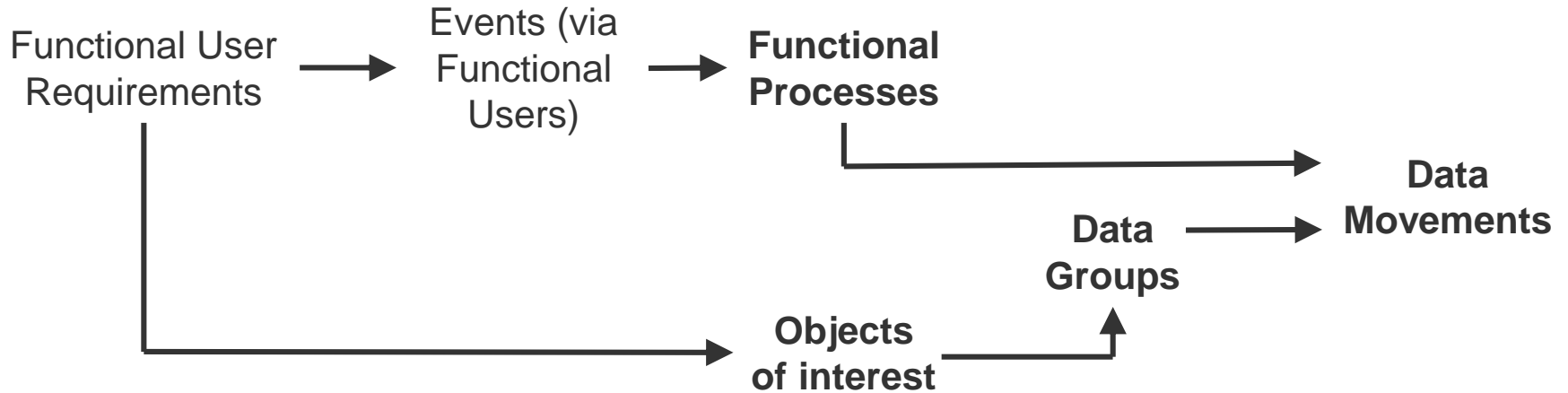
**Do the following FUR statements define a part, a whole or multiple functional processes?**

- The application must maintain data about stock levels for all our products
- Interest shall be applied annually to account balances at the relevant current rate, obtained from the ‘retail interest rate table’
- Foreign income shall be credited at annual-budget exchange rates
- The system shall check tyre pressures at one-second intervals; if any tyre pressure drops below standard, a warning light shall be illuminated
- Each work-station shall have an emergency-stop button. When pressed and held for 2 seconds the conveyor belt shall be stopped and the alarm sounded

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

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

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



# 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

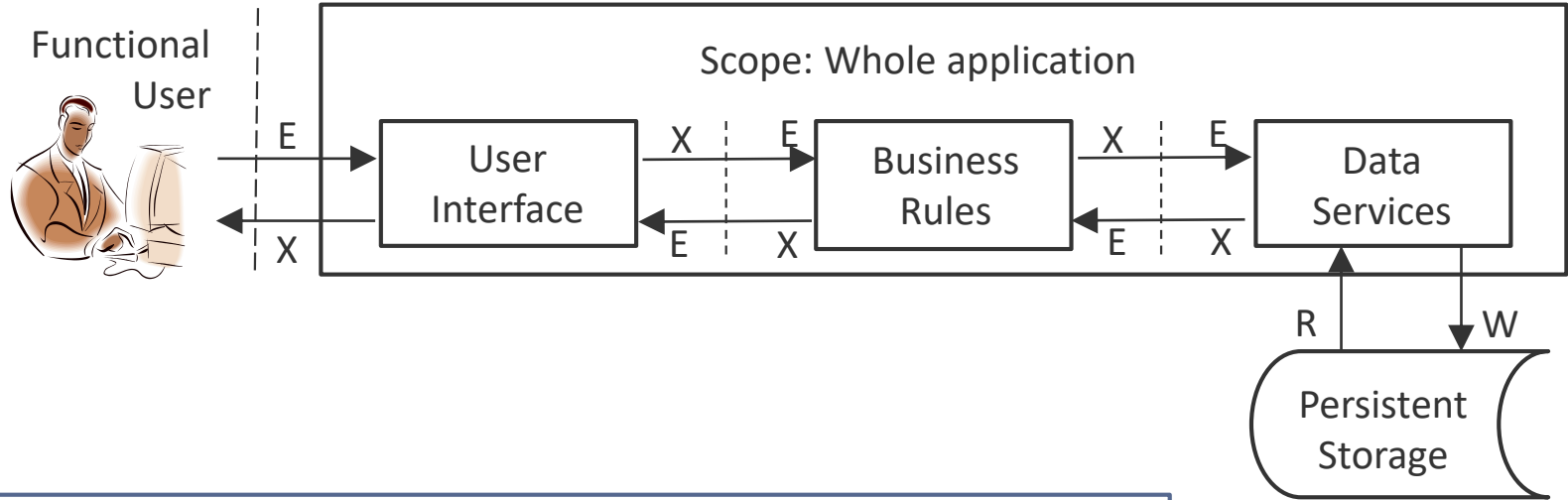
Size of a change to software = Count of DM's added plus Count of DM's modified plus Count of DM's deleted

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

# An example output from the Mapping and Measurement Phases

| Acme Car Hire Functional Process | Data Group Names |                  |                        |             |                       |                       |                 |                 |                      | Nos. of Data Movements     |           |          |          |           |       |
|----------------------------------|------------------|------------------|------------------------|-------------|-----------------------|-----------------------|-----------------|-----------------|----------------------|----------------------------|-----------|----------|----------|-----------|-------|
|                                  | Customer name    | Customer details | Selected customer data | Customer ID | Customer summary data | Customer invoice data | Booking details | Current booking | Booking invoice data | Error/confirmation message | Entries   | Exits    | Reads    | Writes    | Total |
| List customers                   | E                | R                | X                      |             |                       |                       |                 |                 |                      | X                          | 1         | 2        | 1        |           | 4     |
| View customer summary            |                  | R                |                        | E           | X                     |                       | R               | X               |                      |                            | 1         | 2        | 2        |           | 5     |
| View cust. details (Enquiry)     |                  | R, X             |                        | E           |                       |                       |                 |                 |                      |                            | 1         | 1        | 1        |           | 3     |
| View cust. details (pre Update)  |                  | R, X             |                        | E           |                       |                       |                 |                 |                      |                            | 1         | 1        | 1        |           | 3     |
| Update customer details          |                  | E, W             |                        |             |                       |                       |                 |                 | X                    |                            | 1         | 1        |          | 1         | 3     |
| Display invoice print preview    |                  | R                |                        | E           |                       | X                     | R               |                 | X                    | X                          | 1         | 3        | 2        |           | 6     |
| Add new customer                 |                  | E, W             |                        |             |                       |                       |                 |                 |                      | X                          | 1         | 1        |          | 1         | 3     |
| View customer booking details    |                  | R                |                        | E           | X                     |                       | R, X            |                 |                      | X                          | 1         | 3        | 2        |           | 6     |
| <b>Totals for Acme System:</b>   |                  |                  |                        |             |                       |                       |                 |                 |                      | <b>8</b>                   | <b>14</b> | <b>9</b> | <b>2</b> | <b>33</b> |       |

# Sizes may be aggregated, if sensible to do so, subject to one rule



$$\text{Size of the 'whole' application} = \text{Size of its components} \text{ less } \text{Size of inter-component DM's}$$

# An analogy: the surface area of a brick wall

Suppose a wall built of 'n' bricks.



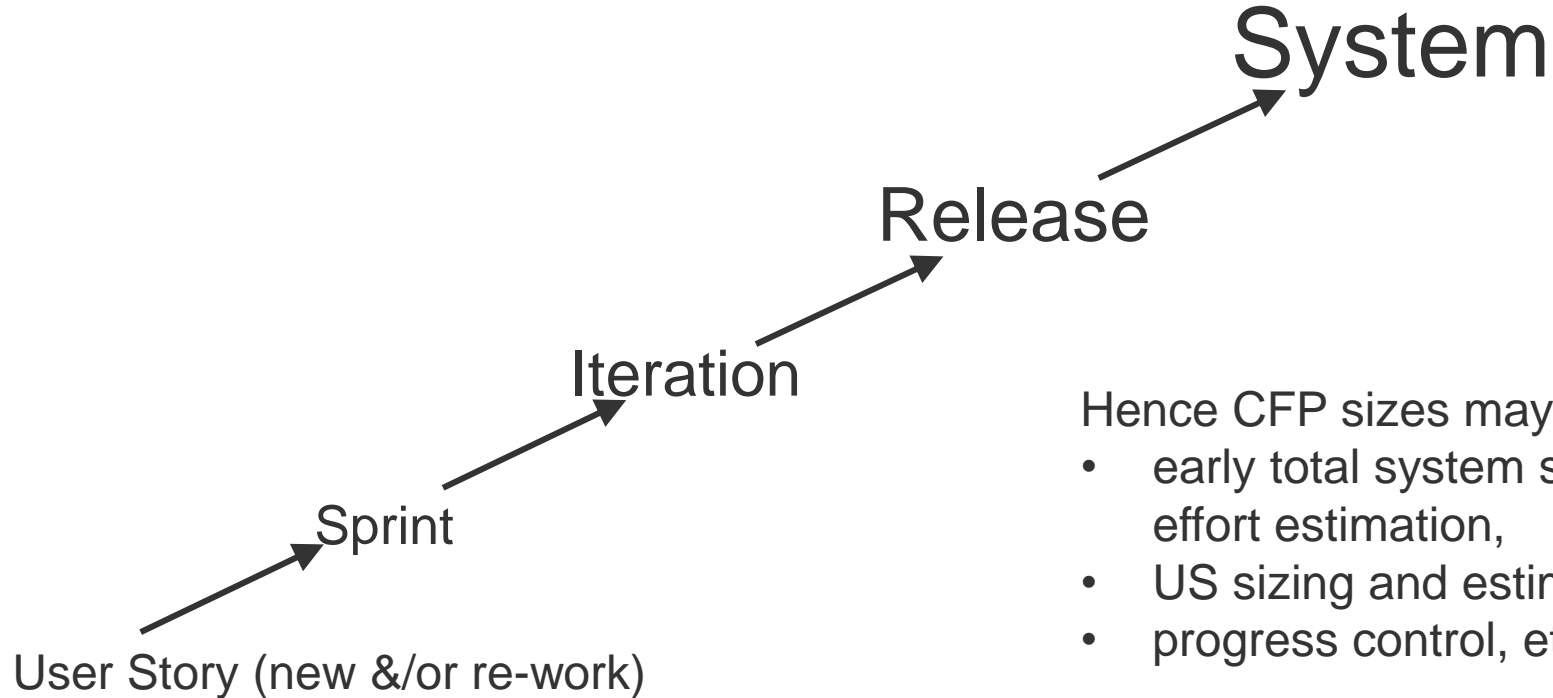
The surface area  
of each brick =  $S_B$

The surface area of the wall  
 $S_W$  is NOT =  $n \times S_B$

$S_W$  = ONLY the area of the  
outward-facing surfaces of  
the 'n' bricks




# So we can measure the size of any Agile deliverable if we follow the aggregation rules



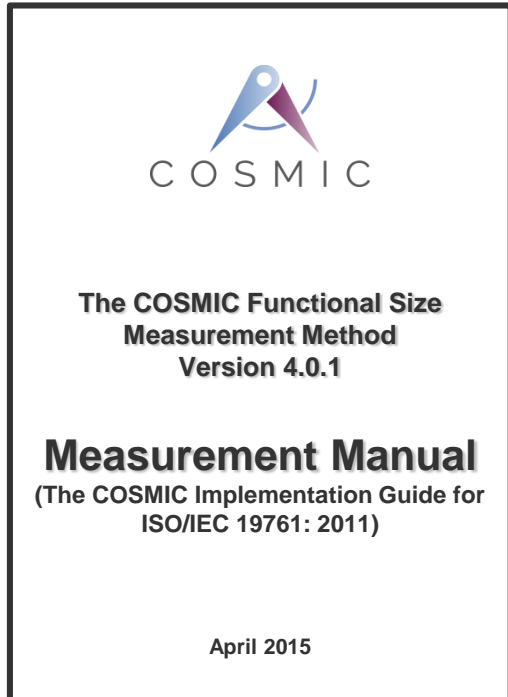
Hence CFP sizes may be used for:

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

# Agenda

- The COSMIC Method design principles and measurement process
  - Setting the Measurement Strategy
  - Mapping and Measurement phases
-  **Conclusions**

# The key elements to master the COSMIC method



+

Guidelines on:

- Early and rapid sizing
- How to deal with NFR

with  
support  
from

Domain or method-  
specific Guidelines :

- Business
- Real-time
- Data Warehouse
- SOA
- Mobile
- Use in Agile
- + many Case Studies

&amp;

Guidelines on:

- Assuring accuracy of COSMIC FSM
- Conversion from '1<sup>st</sup> Generation' sizes

# Other important support

- Forums (on LinkedIn CUG, [www.cosmic-sizing.org](http://www.cosmic-sizing.org) )
- Certification examinations
- Data capture and automatic measurement tools
- Many research reports \*
- ISBSG benchmark data

\* See International Workshop on Software Measurement papers, via IEEE Xplore, or via [www.cosmic-sizing.org](http://www.cosmic-sizing.org)