The COSMIC Functional Size Measurement Method, Version 3.0

Charles Symons

On behalf of the COSMIC Measurement Practices Committee

IWSM - Mensura 2007   Palma da Mallorca, Spain

Agenda

- Background to the COSMIC method
- Brief overview of the basic measurement principles
- The ‘unfinished business’ of v2.2
- Version 3.0 The ‘Measurement Strategy’ phase: identifying <u>which</u> measurement should be made
- Conclusions

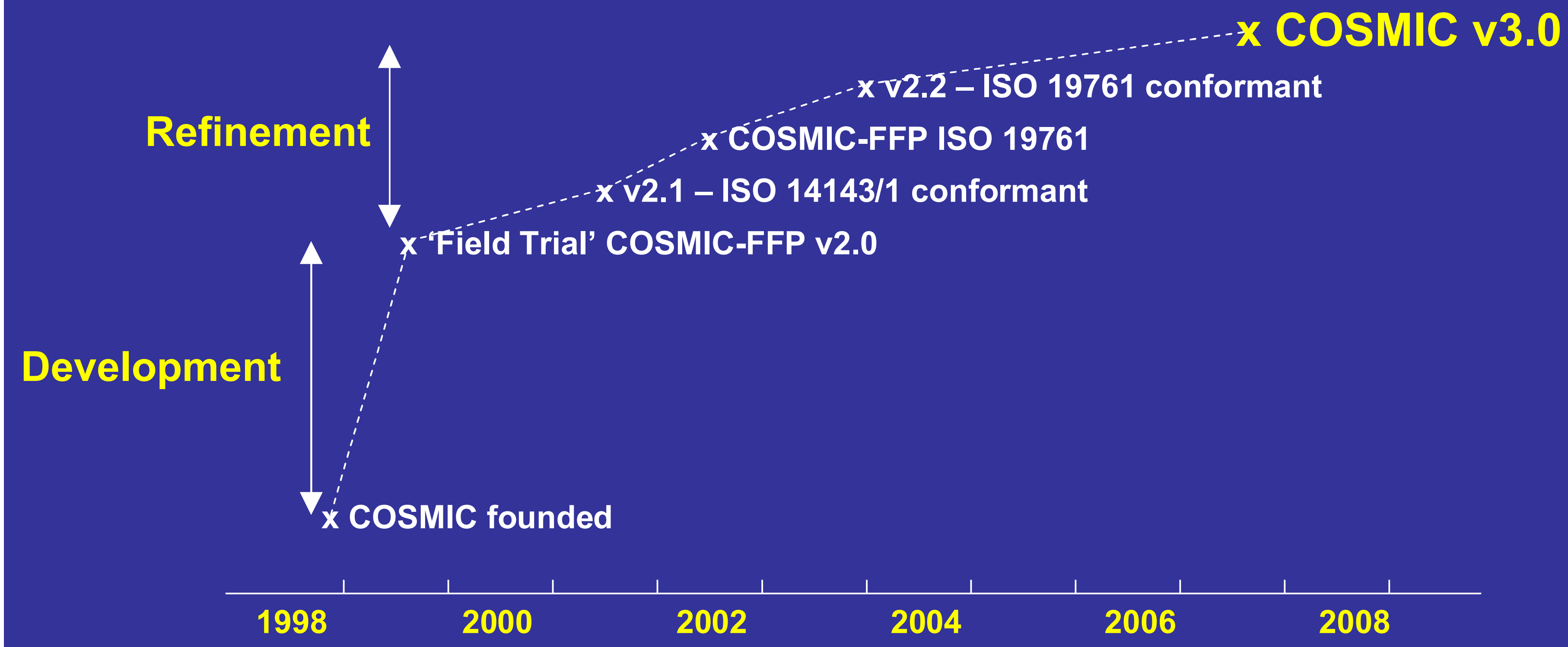
The Common Software Measurement International Consortium

Aim: To develop, test, bring to market and seek acceptance of new software sizing methods to support estimation and performance measurement

Result: The COSMIC functional size measurement method, applicable to:

- business and real-time software and hybrids of these
- software in any layer of a multi-layer software architecture
- at any level of decomposition

The COSMIC method has now existed for nearly eight years



x COSMIC v3.0

x v2.2 – ISO 19761 conformant

Refinement

x COSMIC-FFP ISO 19761

x v2.1 – ISO 14143/1 conformant

x 'Field Trial' COSMIC-FFP v2.0

Development

x COSMIC founded

The basic measurement principles have not changed since the first version (2.0)

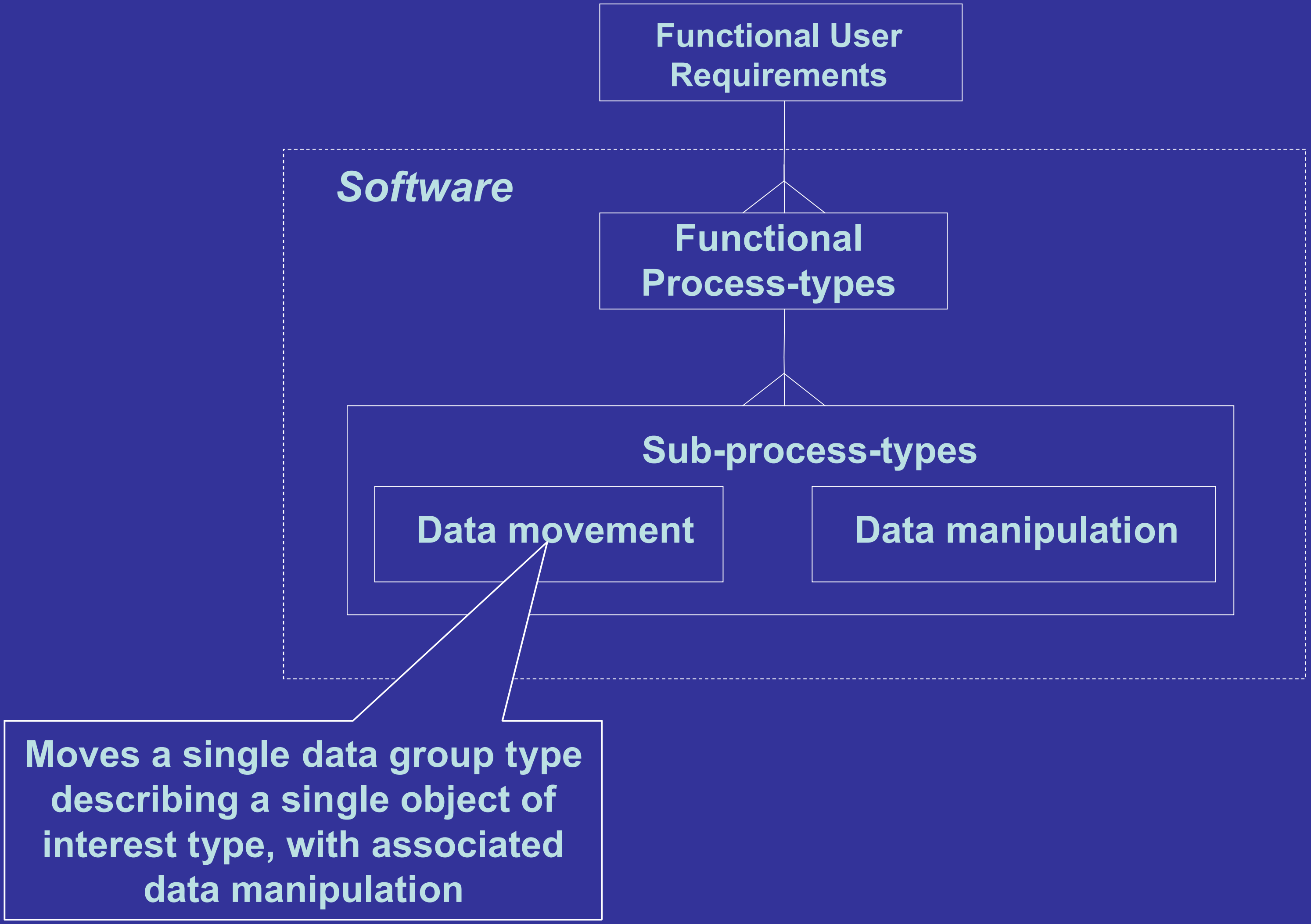
So what has changed since 1999?

- Refinement of definitions and rules
- Alignment of terminology with ISO standards and publication of ISO/IEC 19761 for COSMIC
- Name changes in v3.0
  - 'COSMIC-FFP' to the 'COSMIC' method
  - Unit of measure name from 'Cfsu' to 'CFP'
- Re-structuring of the documentation
- Separation of a 'Measurement Strategy' phase in the measurement process of v3.0
  - Clarification of 'which' size is to be measured

Agenda

- Background to the COSMIC method
- Brief overview of the basic measurement principles
- The ‘unfinished business’ of v2.2
- Version 3.0 The ‘Measurement Strategy’ phase: identifying which measurement should be made
- Conclusions

The ‘Generic Software Model’



Functional User Requirements

Software

Functional Process-types

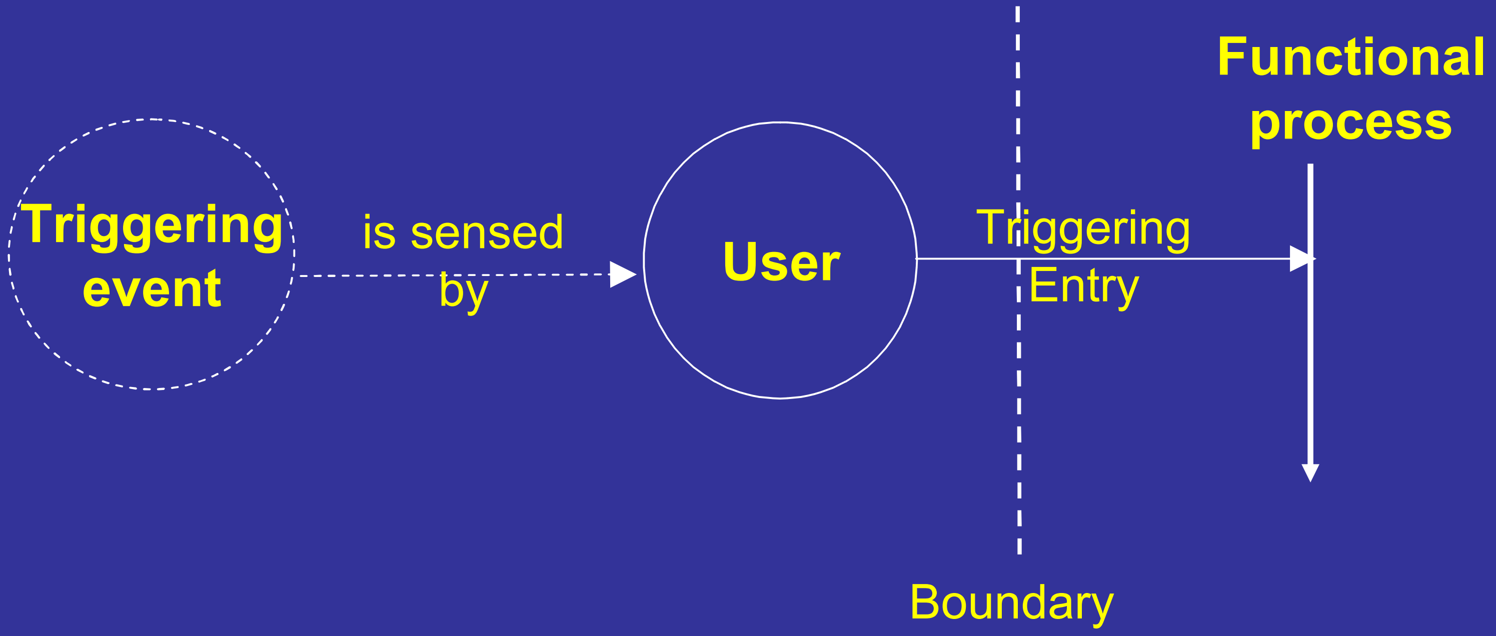
Sub-process-types

Data movement

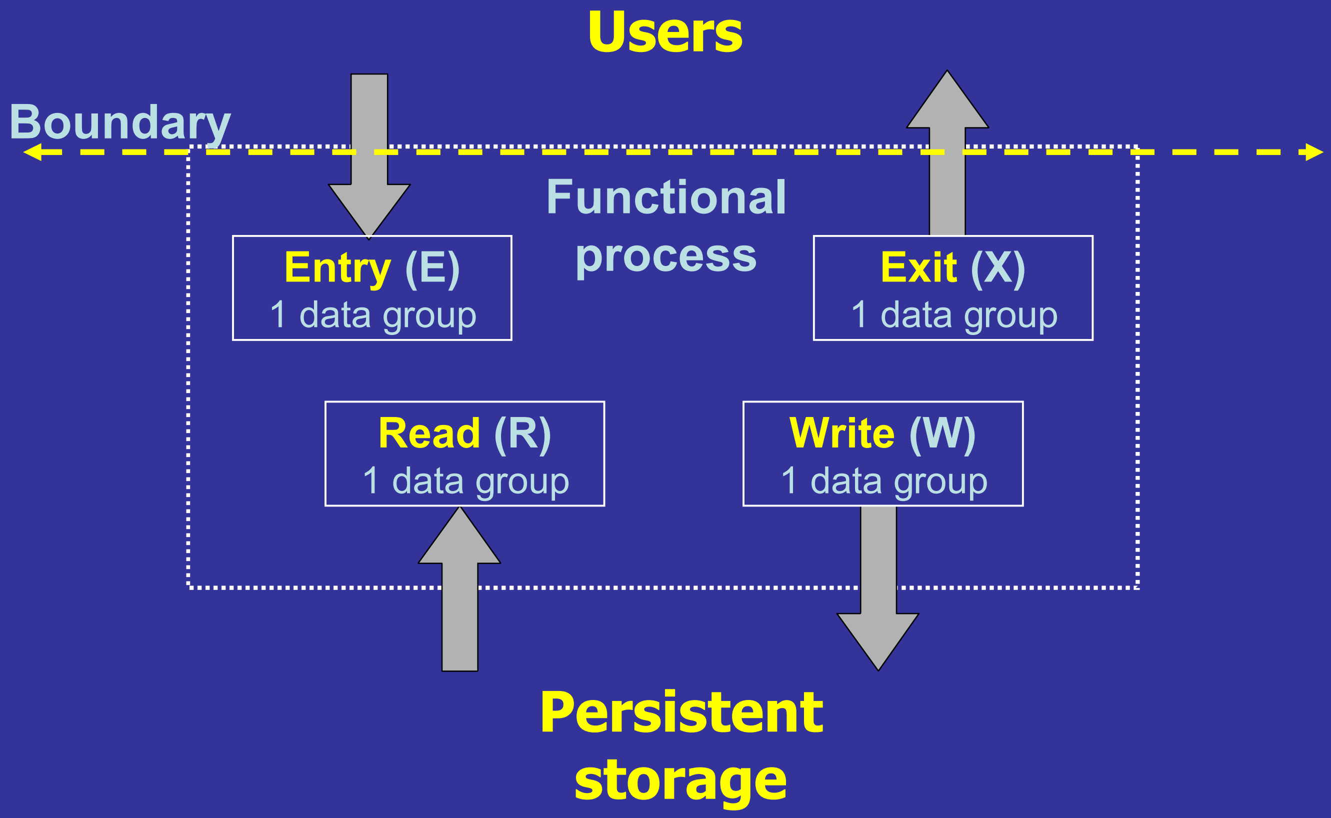
Data manipulation

Moves a single data group type describing a single object of interest type, with associated data manipulation

The ‘Generic Software Model’ (continued)



The ‘Generic Software Model’ (continued)



The Measurement Principles

The size of

- each Data Movement type in each functional process type (Entry, Exit, Write and Read) is assigned one ‘CFP’ (COSMIC Function Point’)
- a Functional Process type is the arithmetic sum of the number of its Data Movement types (no upper size limit)
- an item of software is the sum of the size of all its Functional Process types

COSMIC's 'principles-based' approach ensures the method is future-proof

All rules, guidance and examples must be derivable from a basic set of software engineering principles

The COSMIC Method:

• ISO standard (v2.1)	17 pages
• Principles (v3.0):	3 pages
• Rules (v3.0):	6 pages
• Basic documents (v3.0):	100+ pages

Contrast the IFPUG 'rules-based' method

• ISO standard	342 pages

Agenda

- Background to the COSMIC method
- Brief overview of the basic measurement principles
- The ‘unfinished business’ of v2.2
- Version 3.0 The ‘Measurement Strategy’ phase: identifying which measurement should be made
- Conclusions

Q: Who are the ‘users’ of the embedded application software of e.g. a printer/copier?

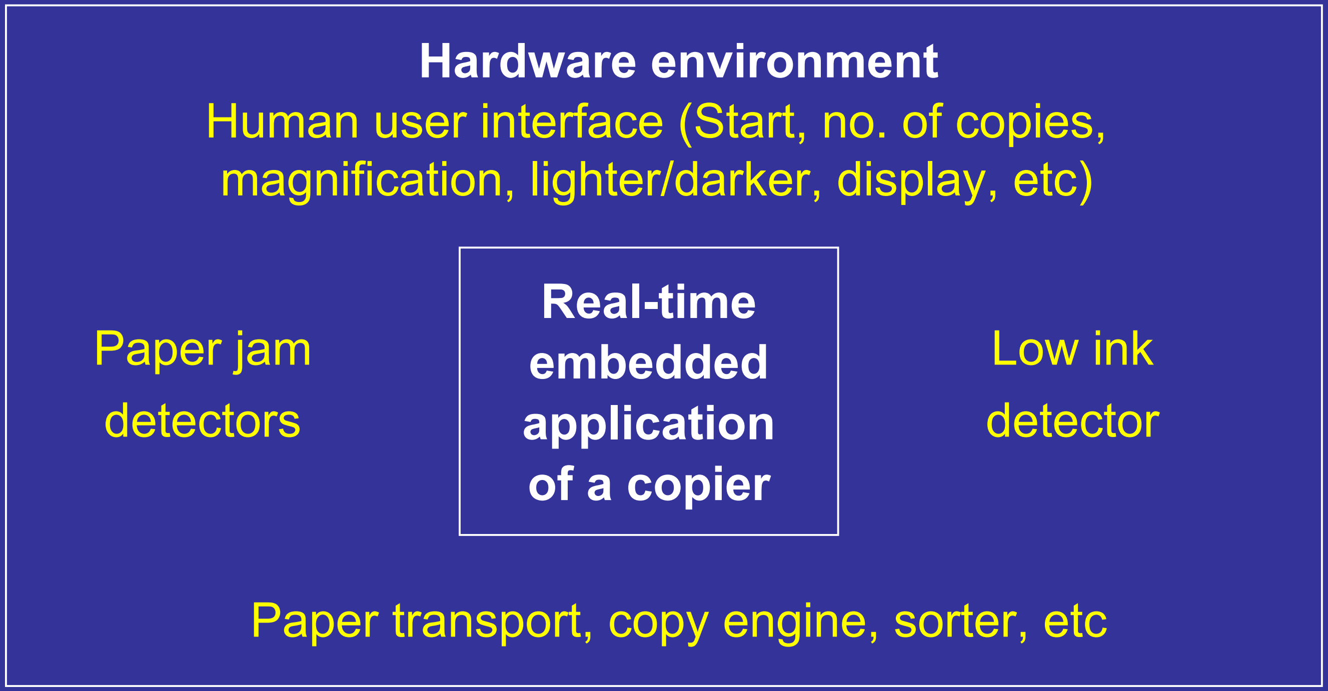
User(s) – ***‘any person or thing that interacts with the software at any time’***

So is the user

- the human operator?
- the engineered hardware devices of the copier?
- the operating system (if any)?
- peer applications, e.g. if the printer/copier is networked?

In v2.2 we introduced two 'Measurement Viewpoints' to solve this problem in real-time software sizing

The human 'End User' sees the functionality available via the human interface



Hardware environment

Human user interface (Start, no. of copies, magnification, lighter/darker, display, etc)

Paper jam detectors

Real-time embedded application of a copier

Low ink detector

Paper transport, copy engine, sorter, etc

The 'Developer' sees all the functionality provided to all the hardware devices

Now apply these two ‘Measurement Viewpoints’ to a business application

The ‘End User Measurement Viewpoint’ is clearly OK

- But what functionality is revealed in the ‘Developer Measurement Viewpoint’ for a business application?
- And what functionality is seen by the operating system as a ‘user’ of the application?
- And are these the only two Measurement Viewpoints?

??????

Summary of ‘unfinished business’

- We know that functional size varies depending on who you define as the ‘user’
- Also, functional sizing must take account of the level of decomposition of the software being sized
- And we often need to size requirements before we have all the detail needed

How do we untangle these concepts?

How do we decide which size to measure, or clarify which size has been measured?

Agenda

- Background to the COSMIC method
- Brief overview of the basic measurement principles
- The ‘unfinished business’ of v2.2
- Version 3.0 The ‘Measurement Strategy’ phase: identifying <u>which</u> measurement should be made
- Conclusions

COSMIC calls the process of defining which size to measure:- ‘Setting the Measurement Strategy’

Establish the Purpose of the measurement, which determines

- The Scope of each piece of software to be measured
- The Functional Users of the software to be measured
- The Level of Granularity (LoG) at which the measurement result is required

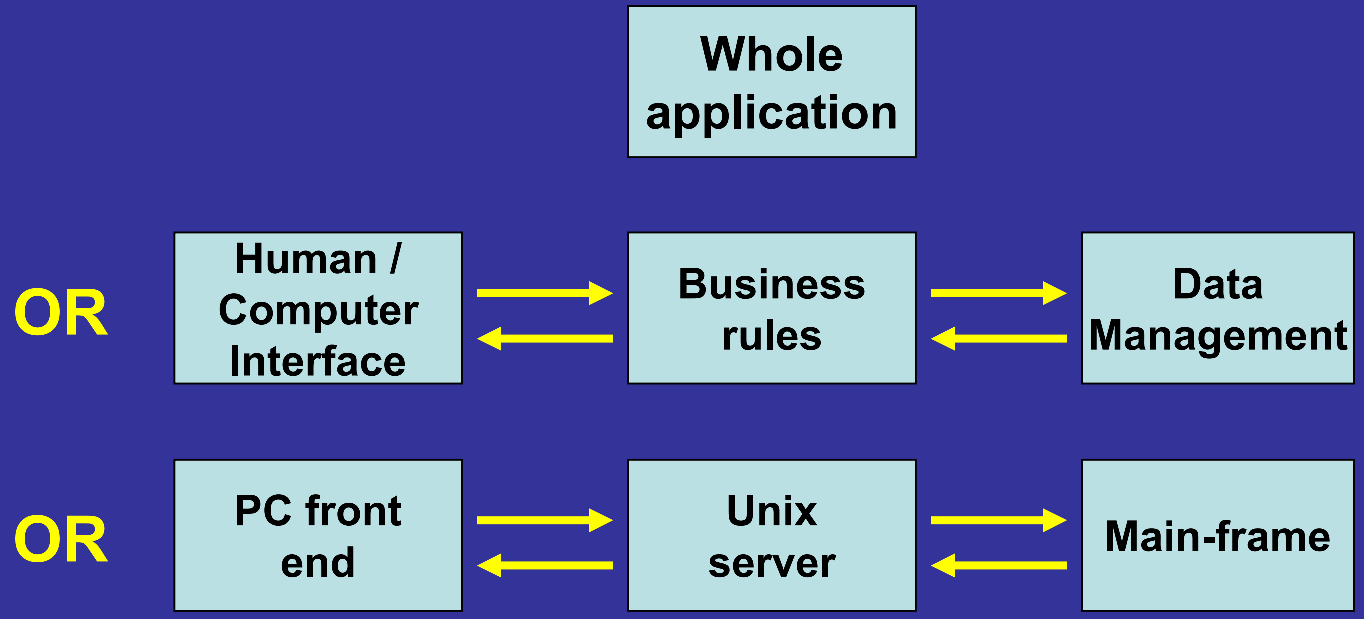
This process and the parameters are totally independent of the COSMIC Method

Determining the scope: *‘the set of FUR to be included in a specific FSM instance’*

- (Distinguish the ‘overall scope’ from the scopes of the individual pieces to be measured)
- Define the level of decomposition of the pieces to be measured
- Distinguish the types of work needed to deliver the functionality within the individual scope(s)
  - Newly developed functionality
  - Changes to existing functionality
  - Re-used functionality (existing functionality that has been re-used, unchanged

Setting standard levels of decomposition is important

Why important? Because the size of a whole piece of software can only obtained by adding up the sizes of its components, if the size contributions of inter-component communications are eliminated



But setting standard levels of decomposition is also difficult

Because widely-used terms have different meanings in different organizations

Possible standard Levels of decomposition	Single Platform	Multi-Platform
Whole application	Yes	Yes
Major component	Yes	-
Object-class	Yes	-

‘Yes’ = possible functional size measures that should not be confused

We now prefer to use the term the '**functional user**', rather then 'user'

Definition: *'a (type of) user that is a sender or intended recipient of data in the functional users requirements of the software to be measured'*

(i.e. the 'FU' in the 'FUR')

Example

Purpose: measure the functional size of the embedded application software of the copier/printer as input to estimating the development effort

Solution: measure the FUR that define the functionality provided to the engineered hardware devices and to any peer applications, as functional users

(Human operators and the OS will not appear as 'functional users' in these FUR of the application.)

Generally, the types of functional users are obvious from the FUR and the purpose of the measurement

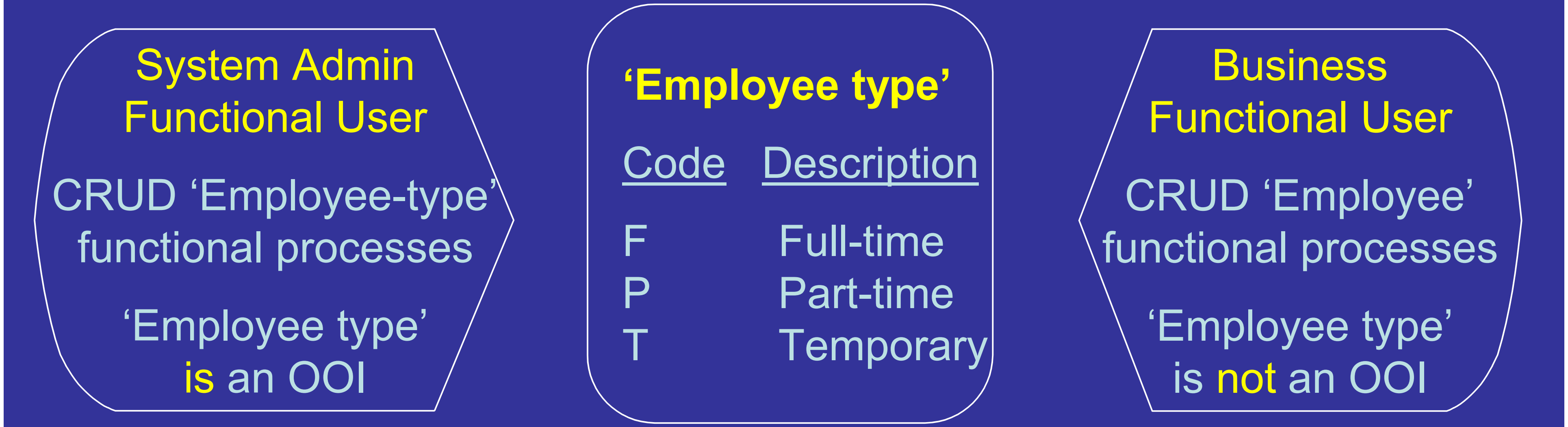
Examples: When the purpose is related to performance measurement, benchmarking or estimating

If the scope is a	The functional users will normally be:
• Business application	Humans and maybe peer applications
• Embedded real-time	Engineered hardware devices and peer apps
• Object-class	Peer object classes
• Complex software architecture component	Other peer software components of the architecture at the same level of granularity

But it ain’t necessarily so! The type of functional users should always be stated for a given measurement

With the concept of ‘functional user’ we can now improve rules for the measurement of ‘code tables’

Example code table



System Admin Functional User

CRUD ‘Employee-type’ functional processes

‘Employee type’ is an OOI

‘Employee type’

Code	Description
F	Full-time
P	Part-time
T	Temporary

Business Functional User

CRUD ‘Employee’ functional processes

‘Employee type’ is not an OOI

‘CRUD’ = Create, Read, Update and Delete; ‘OOI’ = Object of interest

Contrast the IFPUG method rule: ‘Ignore code tables’

Software can be described and measured at any ‘level of granularity’ (or ‘LoG’)

Definition:

‘Any level of expansion of the description of a single piece of software (e.g. a statement of its requirements) such that at each increased level of expansion the description reveals the software’s functionality at an increased and comparable level of detail.

We are familiar with road-maps at different LoG’s

The size of a nation’s road system appears to increase as you zoom-in to lower LoG’s

- Motorways and main roads
- Typical motorists atlas
- Typical street plans

The same is true for software, but with software we have only one ‘standard’ LoG at which we can measure – that of the functional processes

Functional sizes should always be measured at, or scaled to, the LoG of individual functional processes

This is easy when the functional users are individual humans, e.g.

Case 1: Amazon web-based ordering application

or when the functional users are individual engineered hardware devices, e.g.

Case 2: Printer/copier embedded application

If we must size the FUR before we have the detail of the functional processes, then we use an approximate sizing approach

- Al FSM Methods have approximate sizing approaches, e.g.
  - IFPUG – ‘quick and easy’
  - COSMIC – various approaches

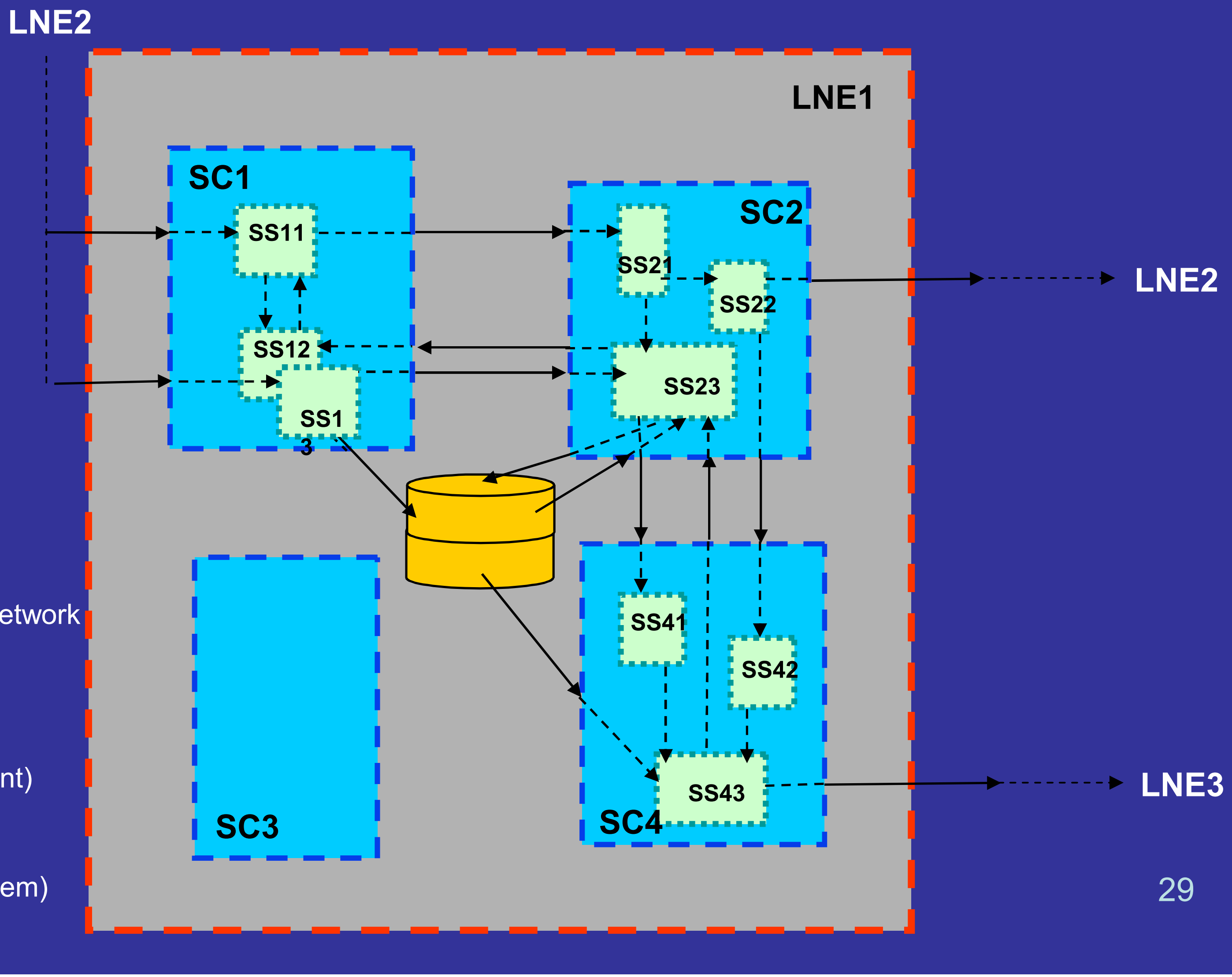
- Example scaling: we might determine that

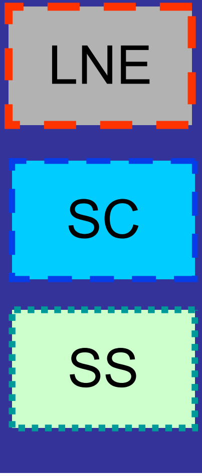
  a Use Case on average comprises 3 functional processes

  a functional process has an average size of 10 CFP

  Then 1 x Use Case = 30 CFP

Case 3. In a complex software architecture, it's not at all clear at which LoG we should stop zooming in





(Logical Network Element)

(System Component)

(Sub-System)

In this pure software architecture, functional users and processes can be recognised at any level of granularity and/or decomposition

Level of Granularity	No. of functional processes	Functional Size (CFP)
LNE	1	8
SC	4	20
SS	9	32

A standard LoG for measurements can only be defined locally

Determining the ‘right’ LoG at which to measure in complex software architectures requires great care

- Functional users and functional processes can be defined at any LoG

  (since software functional users can be decomposed to many levels, unlike individual humans or engineered devices)

- Need to define standard LoG’s locally at which measurements must be made and can be compared

Agenda

- Background to the COSMIC method
- Brief overview of the basic measurement principles
- The ‘unfinished business’ of v2.2
- Version 3.0 The ‘Measurement Strategy’ phase: identifying which measurement should be made
- Conclusions

The COSMIC method v3.0 represents a big advance for FSM in general

- The basic functional size measurement principles and rules are unchanged
- The question of which size to measure is greatly clarified
- The approach we have adopted is valid for all FSM methods

The FSM community needs to define standards to ensure functional size measurements can be compared

- Scope parameters
  - Levels of decomposition
  - Types of work
- (types of) Functional users
- Levels of granularity

…. with real benefits for more reliable performance measurement, estimating and benchmarking

A final word

- This presentation may appear to make functional sizing more difficult

  (there are hundreds of possible sizes resulting from combinations of these concepts!)

- But in any one organization, only a limited number of combinations will be necessary

Thank you for your attention

www.cosmicon.com

www.gelog.lrgl.ca/cosmic-ffp