

# Measuring the functional size of real-time software

**Co-authored by:**

A. Abran, J.-M. Desharnais, S. Oligny

UQAM - Software Engineering Management Research Laboratory,  
Centre d'Intérêt sur les Métriques (C.I.M.)

*January 1999*

# *Presenter profile*

---

- ⊙ **Serge Oligny, M.Sc.**
  - ✓ **Director - Technological innovations, UQAM-Software Engineering Management Research Laboratory**
  - ✓ **CIM - Counting Practices, Executive Committee**
  - ✓ **Formerly Corporate Manager of software development in the pulp & paper industry**
  - ✓ **13 years experience in the IT consulting market**

# Agenda...

---

- ⊙ Introduction
- ⊙ Characteristics of real-time software
- ⊙ The measurement process model
- ⊙ Measurement Procedures:
  - ⊙ Measurement boundary
  - ⊙ Measurement Scope
  - ⊙ Identifying elements to be measured
  - ⊙ Assigning points
- ⊙ Overview of field tests results
- ⊙ Conclusion

# *Introduction...*

---

- ⦿ **Functional size measurement**
- ⦿ **Origins and evolution**
- ⦿ **Characteristics of FFP**
- ⦿ **An analogy**

# ***Functional Size Measurement***

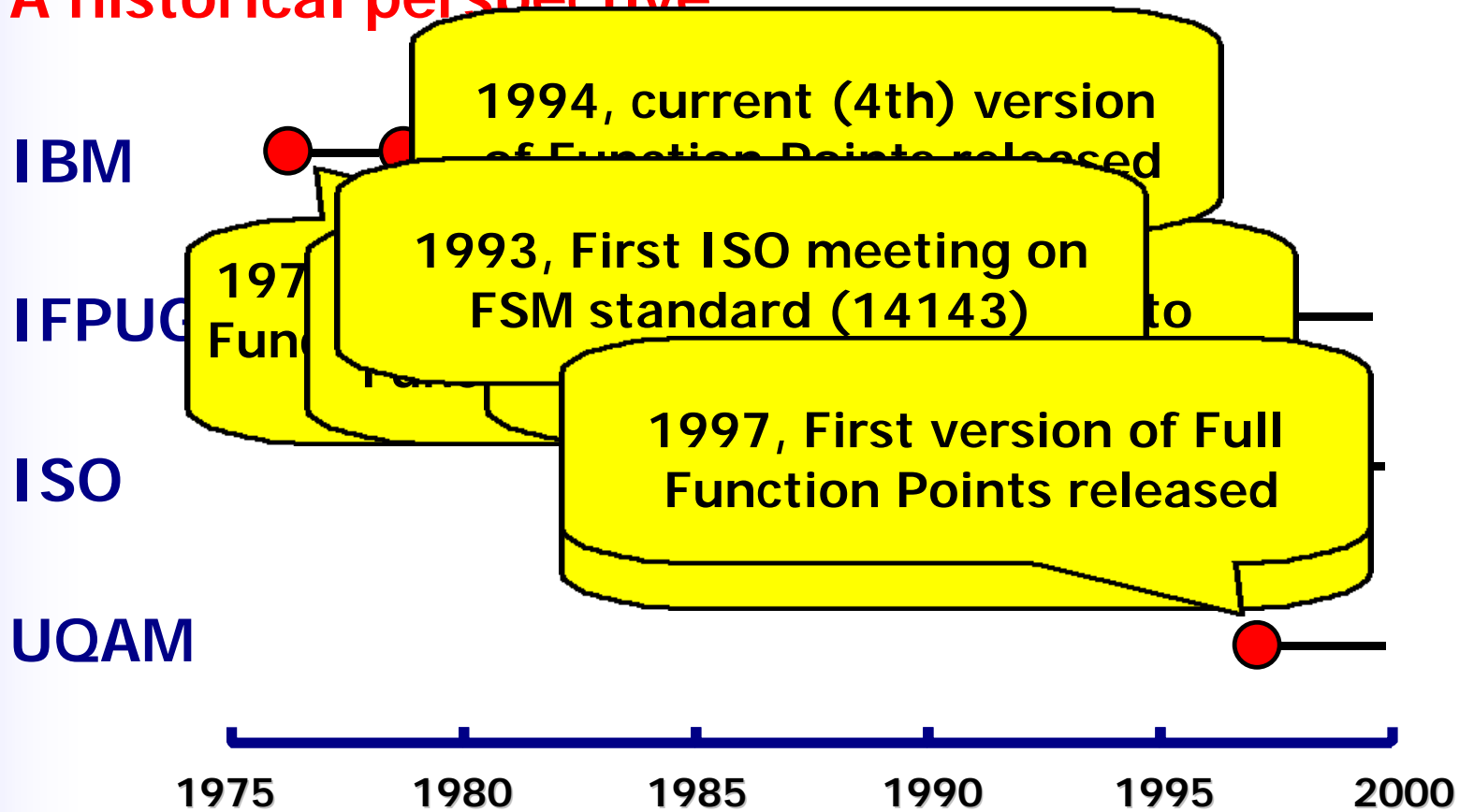
---

- ⊙ ISO/IEC/JTC1/SC7 Standard #14143 definition:

**“ Functional Size : A size of software derived by quantifying the **functional user requirements**”**

# Origins and evolution...

## A historical perspective



# *Characteristics of FFP...*

---

- FFP is a Functional Size Measure
- Focused on the '**User functional view**'
- Applied at **any time** during the software development life cycle
- Derived in terms **understood by users**
- Derived without reference to:
  - **effort**
  - **methods** used
  - **physical or technical** components .

# *An analogy...*

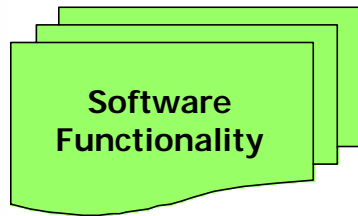
---



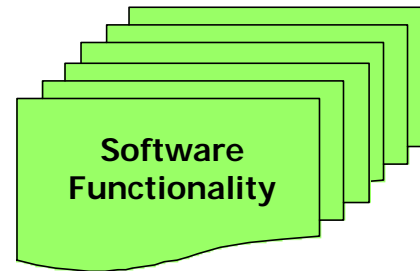
**2000 sq. ft.**



**4000 sq. ft.**



**500 FFP**



**1000 FFP**

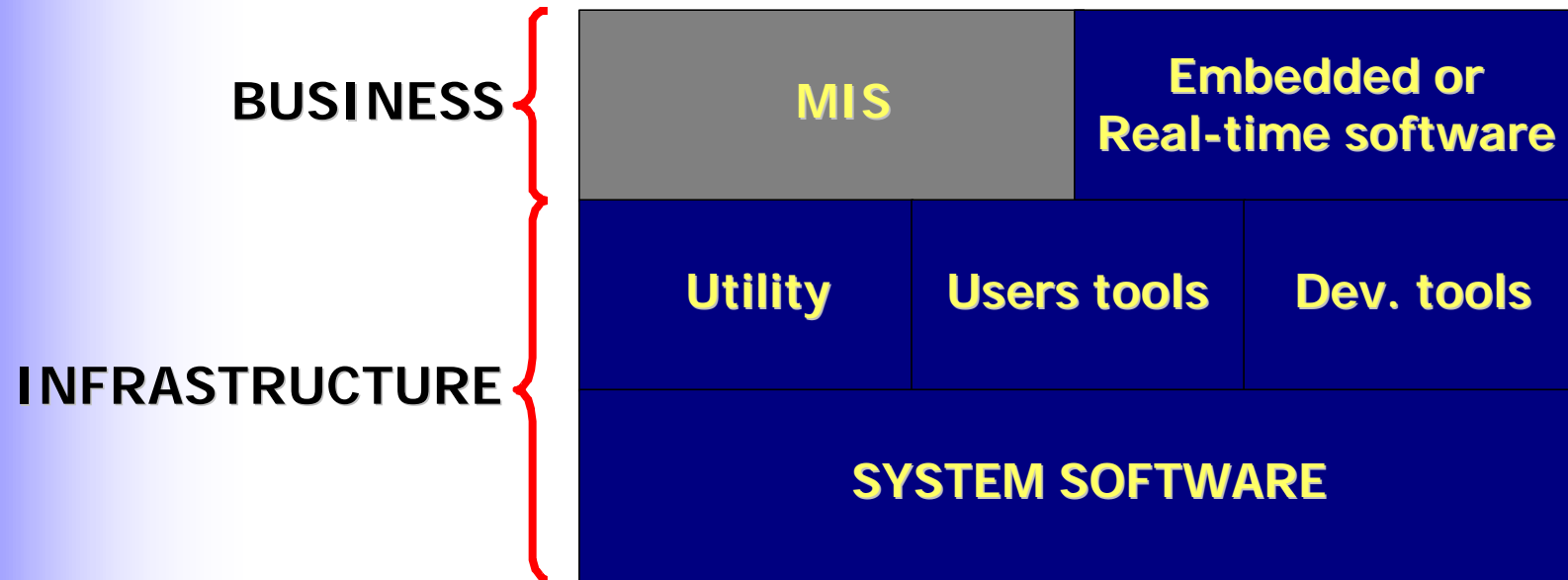


# *Characteristics of real-time software*

---

- ⦿ Different types of software
- ⦿ Real-time or embedded software
- ⦿ Limitations of IFPUG 4.0 Function Point

# Different types of software



# Real-time or embedded software

---

## ⊙ Timing

- ✓ Tight constraints on the rate of execution and on the timing of tasks
- ✓ Explicit constraints on timing
- ✓ Dedicated components to manage timing
- ✓ Correctness of the result is linked to timing

## ⊙ Interaction with

- ✓ Mechanical devices
- ✓ People
- ✓ Other applications

# Limitations of IFPUG 4.0 FP

---

Compared to MIS software...

<b>USERS</b>	People Other software Devices
<b>DATA</b>	Permanently stored (files, DB, ...) Not stored permanently (signals, ...)
<b>PROCESSES</b>	No. of sub-processes varies a lot Processes role is not easily classified as input, output or inquiry

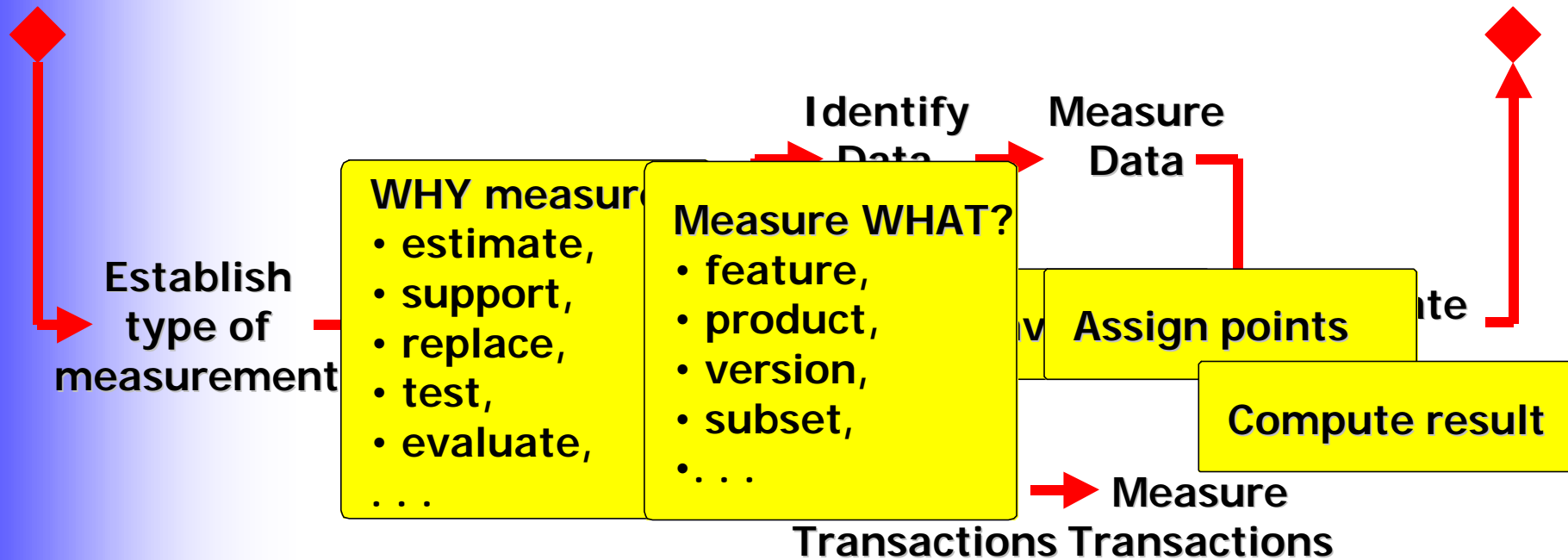
**IFPUG Function Points (4.0), do not adequately measure the functional size of real-time software**

# *The measurement process model*

---

- ◉ Overview of the measurement process
- ◉ Notes on measurement purpose...
- ◉ Notes on measurement strategy...
- ◉ Notes on documentation to be used...

# Overview of the measurement process



# **Notes on measurement purpose**

---

- ⊙ Identify the **business issue** which needs to be addressed
  - ✓ ...to estimate the size of a development project,
  - ✓ ...to determine the functionality supported by the maintenance team,
  - ✓ ...to determine the amount of functionality required to support day to day work activities of a user,
  - ✓ ...to determine replacement costs of software portfolio,
  - ✓ ...to assist in determining system testing strategies,
  - ✓ ...to assess the size of development backlog,
  - ✓ ...to determine mandatory functionality for package evaluation.

# **Notes on measurement purpose**

---

## ⊙ Determine:

- ✓ what questions need to be answered by the size measure,
- ✓ which software applications need to be sized
- ✓ what components of the software will be included or excluded



# Notes on measurement strategy

---

## ⊙ Identify:

- ✓ Which **software** is to be sized,
- ✓ **How** the sizing will be performed,
- ✓ **Who** will do the sizing,
- ✓ Who will **assist** as the application expert,
- ✓ Which Functional Size Measurement method will be used e.g. Full Function Points (FFP) Version 1.0,
- ✓ **When** and **where** will the sizing take place,
- ✓ Which **software tools**, counting forms, will be available.

# ***Notes on documentation to be used***

---

## ⊙ **Planned Applications** (New development)

- ✓ requirements specification
- ✓ logical design specification
- ✓ report layouts
- ✓ screen layouts
- ✓ logical data model

## ⊙ **Existing Applications** (Enhancements)

- ✓ all of the above plus
- ✓ user manual
- ✓ access to application online

# *Measurement Procedures*

---

- ◉ **APPLICATION BOUNDARY**
- ◉ **MEASUREMENT SCOPE**
- ◉ **IDENTIFYING ELEMENT TO BE MESURED**
- ◉ **ASSIGNING POINTS**
- ◉ **EXERCICES**

# Application boundary

---

⊙ Definition of **BOUNDARY** \* :

*'a conceptual interface between the software under study and its users'*

⊙ Definition of **USER** \* :

*'Any person that specifies Functional User Requirements and/or any person or thing (hardware, equipment, other applications) that communicates or interacts with the software at any time'*

\* ISO/IEC/JTC1/SC7 - # 14143 - Software Measurement - Definition of the concepts of functional size Measurement.

# Application boundary

---

## ⊙ Boundary is:

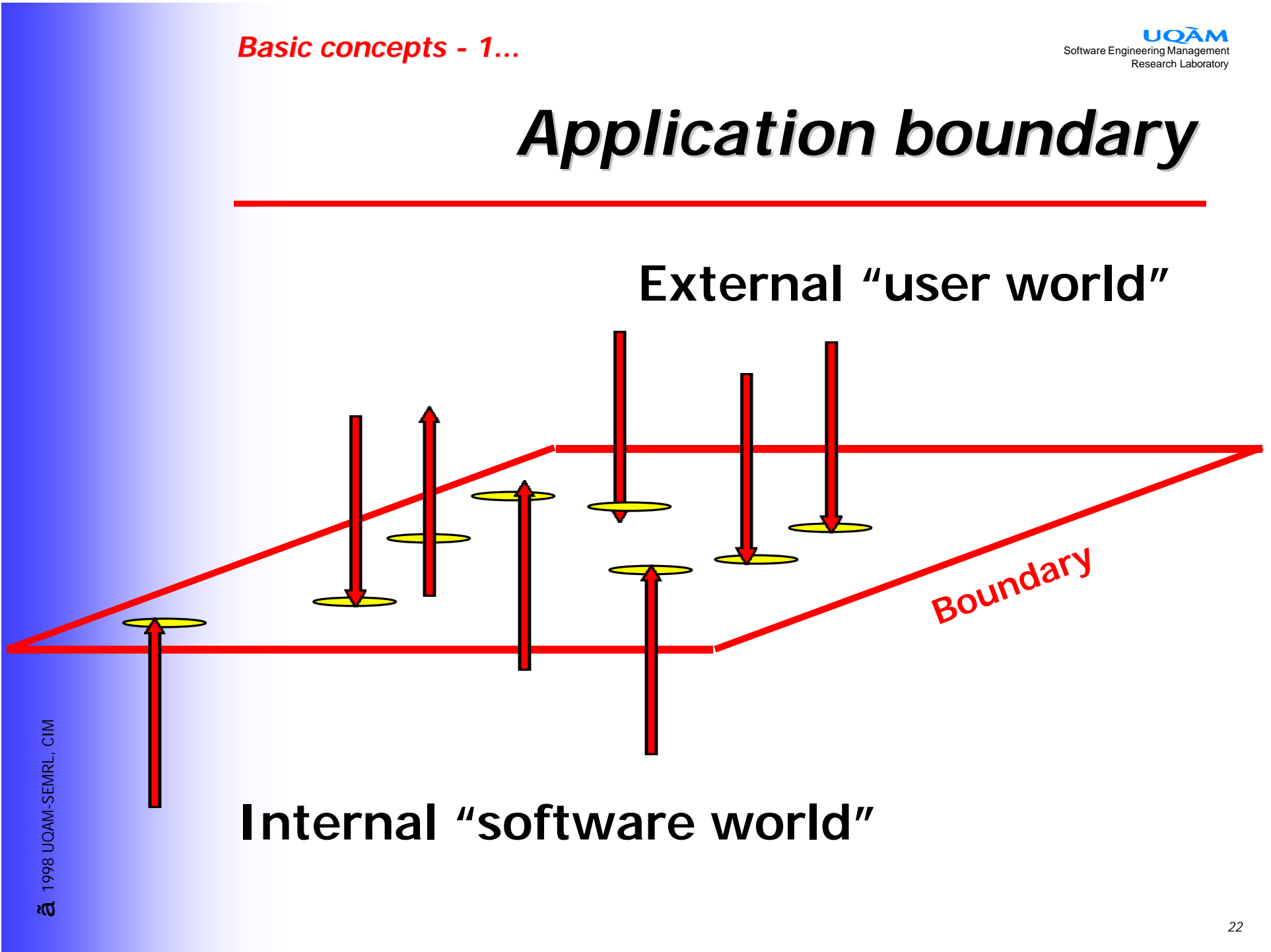
- ✓ 'membrane' through which the transactions pass into and out of the software,
- ✓ external limitation of the software,
- ✓ point where the software stops and the external user world starts.

# Application boundary

External "user world"

Boundary

Internal "software world"

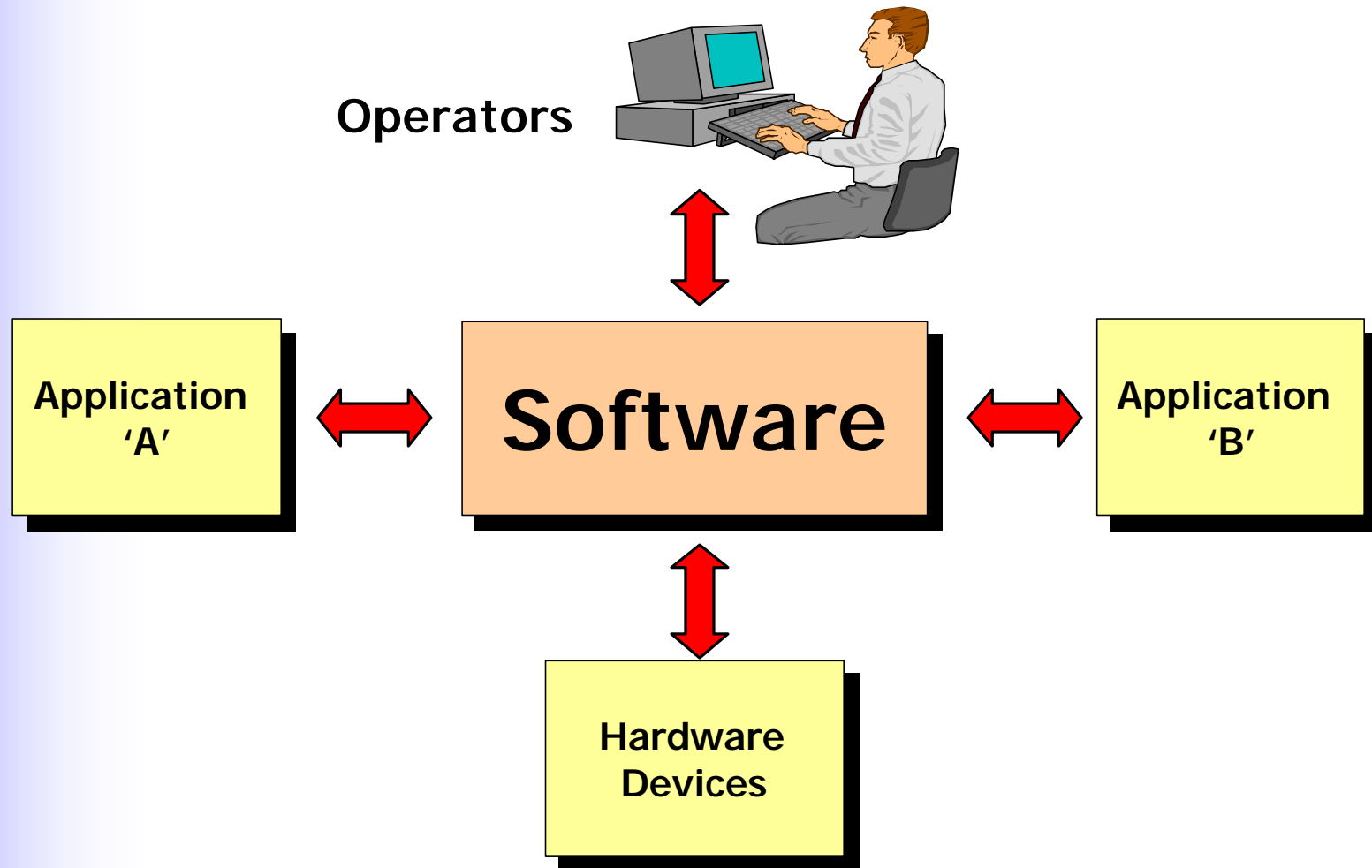


# Application boundary

---

- ⊙ Boundary may be illustrated on an application boundary diagram similar to a 'context diagram'
- ⊙ Identify all **major groups of data movements** between the boundary of this software and:
  - ✓ its human user operators,
  - ✓ and the boundaries of other applications or other hardware devices

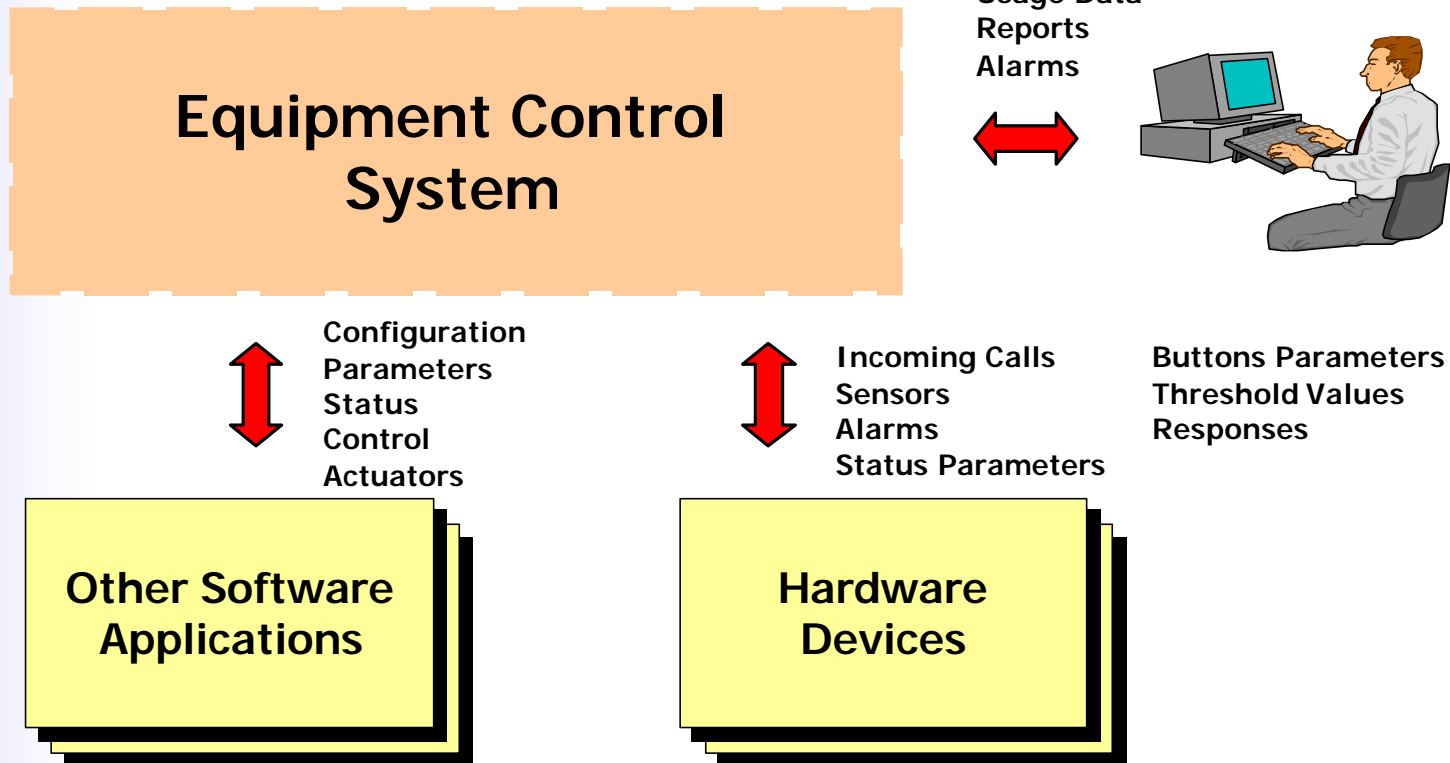
# Application boundary





# Application boundary

## Application Boundary



# Measurement scope

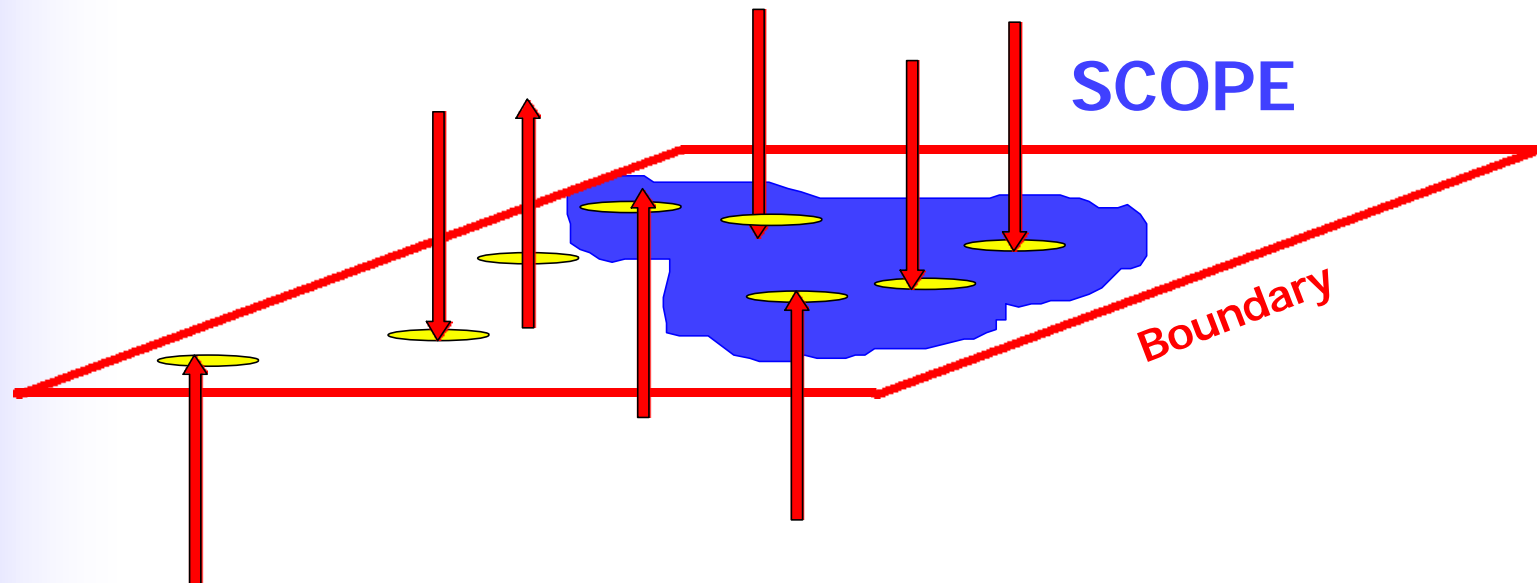
---

- ⊙ Definition of **SCOPE**:

*"The set of functional features, inside the application boundary, for which the size have to be measured"*

- ⊙ Measurement **SCOPE** is dictated by the **PURPOSE** of the measurement exercice.

# Measurement scope



**SCOPE defines a sub-set of the software to be sized**

# Exercise

- Read the Case Study document and answer those questions:
  - what is the purpose of the measurement exercise ?
  - what will be your strategy ?
  - could you draw the boundary of the application ?
  - what is the scope of the project ?

# Exercise : discussion

- What is the purpose of the measurement process?
- What will be your strategy?
- Could you draw the boundary of the application?
- What is the scope of the projects?

# *Identifying elements to be measured*

---

- ◉ Identifying data
- ◉ Identifying transactions

# *Identifying data*

---

- ⊙ **Key concepts**
- ⊙ **Identification rules**
- ⊙ **Summary**

# *Identifying data*

---

## Key concepts

- ⊙ **Data selection**

Which ones are measured ?

- ⊙ **Data occurrences**

How are they organized ?

- ⊙ **Data activity**

How are data handled by the measured application ?



# *Identifying data*

---

## Key concepts - Data selection

- ➔ If a piece of data is processed but not saved or reused, it does not live for more than one transaction. This piece of data is not permanent and it is not measured.
- ➔ If a piece of data is reused for multiple transactions, it lives for more than one transaction. This piece of data is measured.

# Identifying data

---

## Key concepts - Data occurrence

- ⊙ **Single occurrence** are groups of data which have one and only one instance of the record.
  - ✓ Example: Data related to a time clock for a specific time.
  
- ⊙ **Multiple occurrences** are groups of data which can have more than one instance of the same type of record. In real-time, multiple occurrences have the same structure than the one found in MIS System.
  - ✓ Example: Flight record (black box)

# *Identifying data*

---

## **Key concepts - Data activity**

- ⊙ **Updated Groups of data**  
e.g.: add, change, delete, populate, revise, update, assign, create ...  
A group of data may be updated by more than one application.
- ⊙ **Read only Groups of data**  
The group of data is consulted by the application being measured without being updated. The group of data may be updated by other applications.

# *Identifying data*

---

## **Identification rules**

- 1- Select all logically related groups of data that live for more than one transaction.**
  - ✓ There is no formal definition of what is a logically related group of data
  - ✓ From a normalization point of view our practice suggest that a logically related group of data could be at the second or third normal form, but not normalized more than the third normal form
  
- 2- Group data according to their structure**
  - ✓ Each multiple occurrences group is identified
  - ✓ Merge all single occurrence together into one group

# Identifying data

---

## Identification rules

### 3- Determine the nature of data activity for each identified group

- ✓ A **UCG** is a group of data updated by the application being counted.
- ✓ An **RCG** is a group of data used, but not updated, by the application being counted.
- ✓ **UCGs** and **RCGs** are:
  - groups of logically related data (multiple occurrences),
  - groups of not necessarily related data (single occurrence)
  - identified from a functional perspective and contain data that live for more than one transaction

# Identifying data

---

## Identification rules

- 4- Verify that Updated Control Group (UCG) and Read-only Control Group (RCG)

### ARE

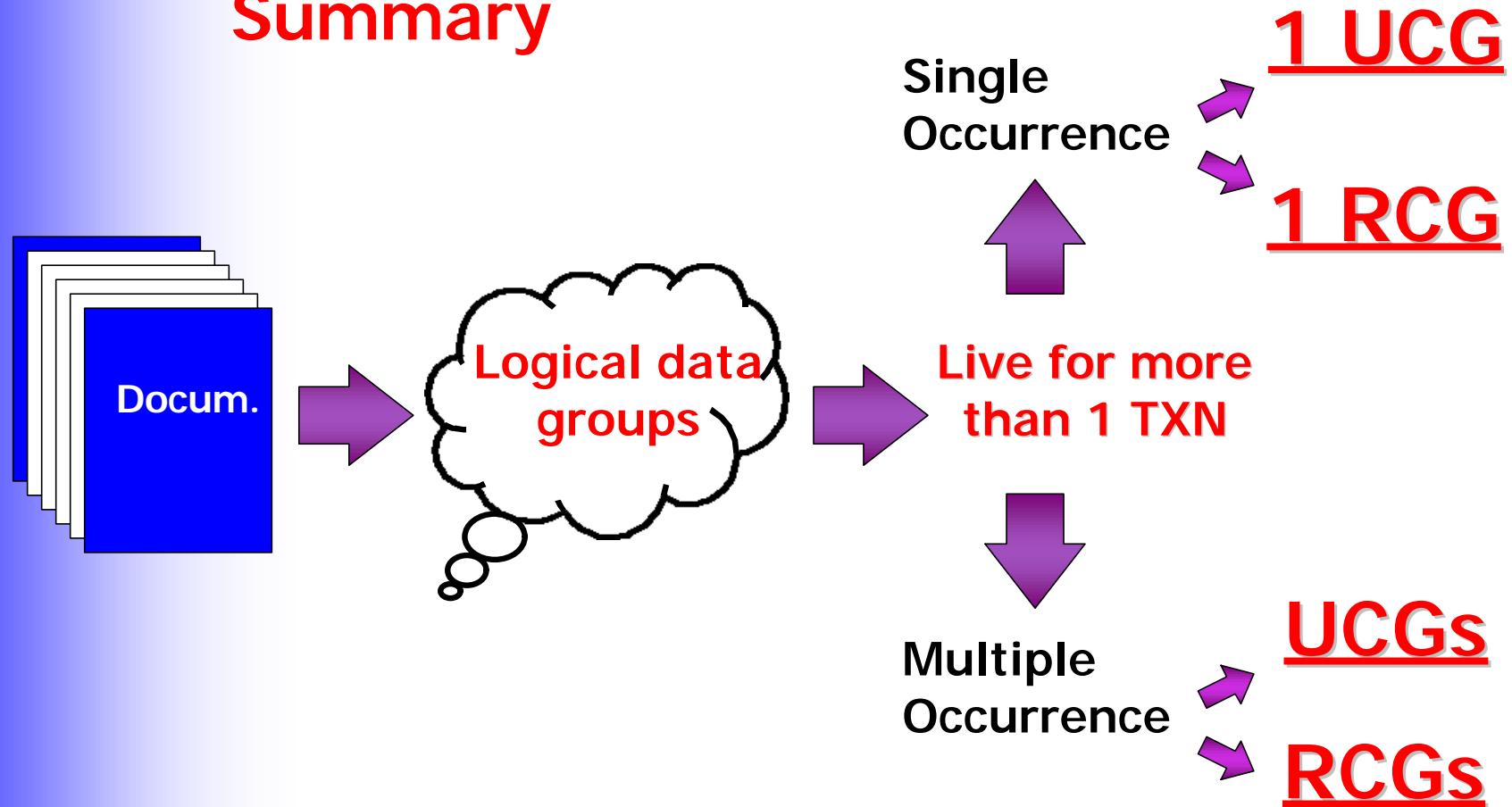
- ✓ Files maintained by the user

### BUT ARE NOT

- ✓ Sorting files
- ✓ Index files or secondary index
- ✓ Generated files sent to another application

# Identifying data

## Summary



*Identifying elements to be measured*

# *Identifying transactions*

---

- ⊙ Key concepts
- ⊙ Identification rules
- ⊙ Summary



*Identifying elements to be measured*

# *Identifying transactions*

---

## Key concepts

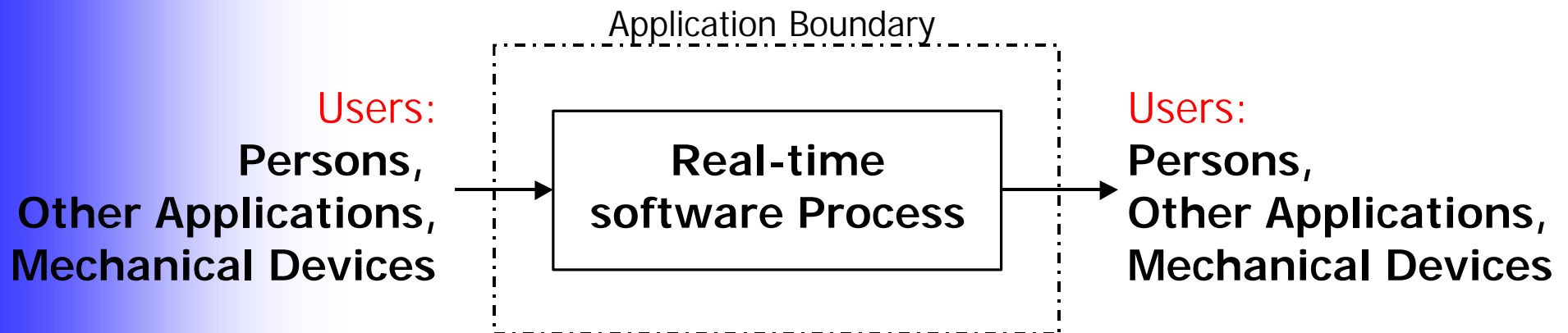
- ⊙ Process
- ⊙ Transaction
- ⊙ Trigger
- ⊙ Sub-processes

## Identifying elements to be measured

# Identifying transactions

## Key concepts - Process

“A set of operations or activities which acts on inputs to produce a result.”



# *Identifying transactions*

---

## Key concepts - Triggers

- ⊙ An event which **initiates** a process from a functional perspective,
- ⊙ An event occurring outside the application boundary,
- ⊙ The manifestation of the event is data which enters the application boundary,
- ⊙ Clocks and timing events can be triggers.

# *Identifying transactions*

---

## Key concepts - Transactions

- ⊙ A **transaction** is an instance of a process/  
sub-process,
- ⊙ A transaction includes all processing  
associated with an occurrence of an  
external trigger.

**Example:** in a watch, each tick of the timing crystal is a trigger. All processing associated with each new tick is a separate transaction.

*Identifying elements to be measured*

# *Identifying transactions*

---

## **Key concepts - Sub-processes**

The smallest processing step identifiable from a functional perspective as either an entry, exit, read or write.

## *Identifying elements to be measured*

# *Identifying transactions*

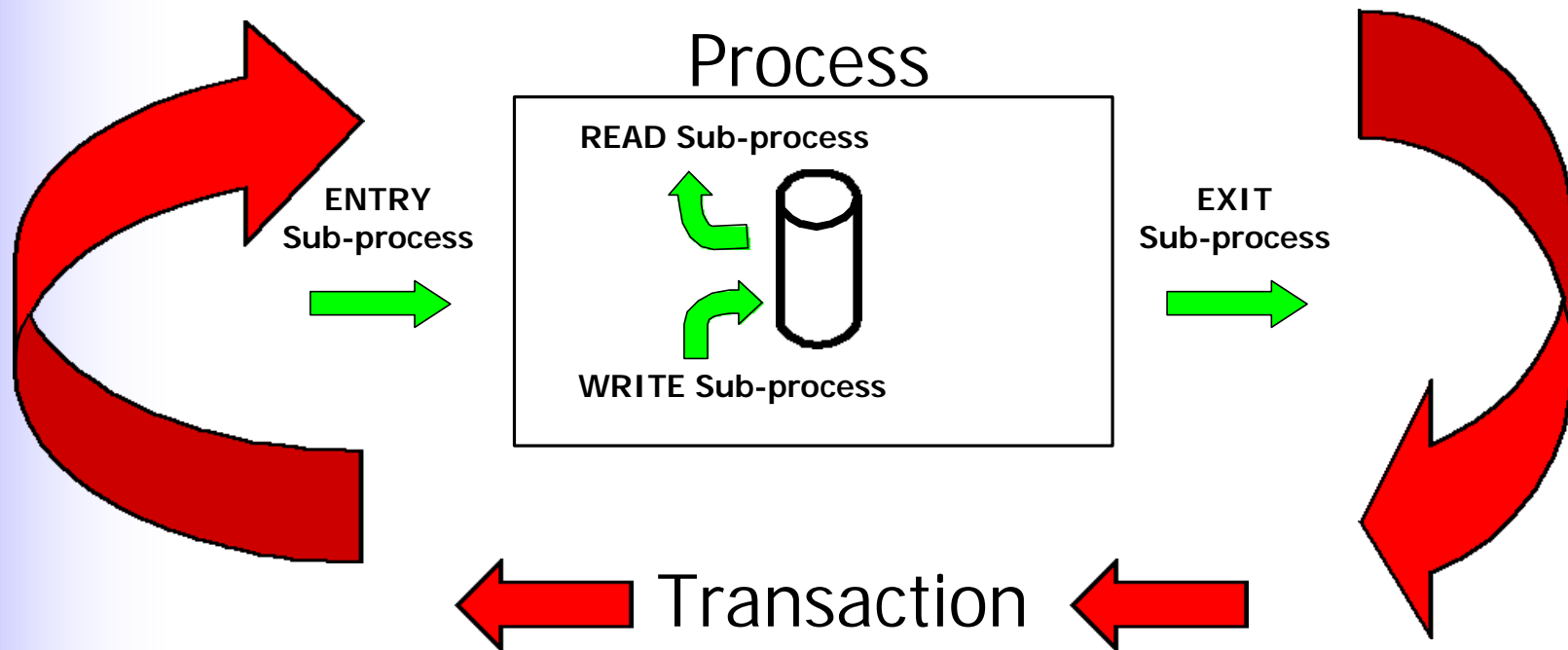
---

## **Key concepts - Sub-processes**

- ⊙ Identified from a functional perspective,
- ⊙ Single sub-processes,
- ⊙ Located at the lowest functional level of a process and acting on one group of data. If a sub-process acts on two groups of data, there are at least two sub-processes

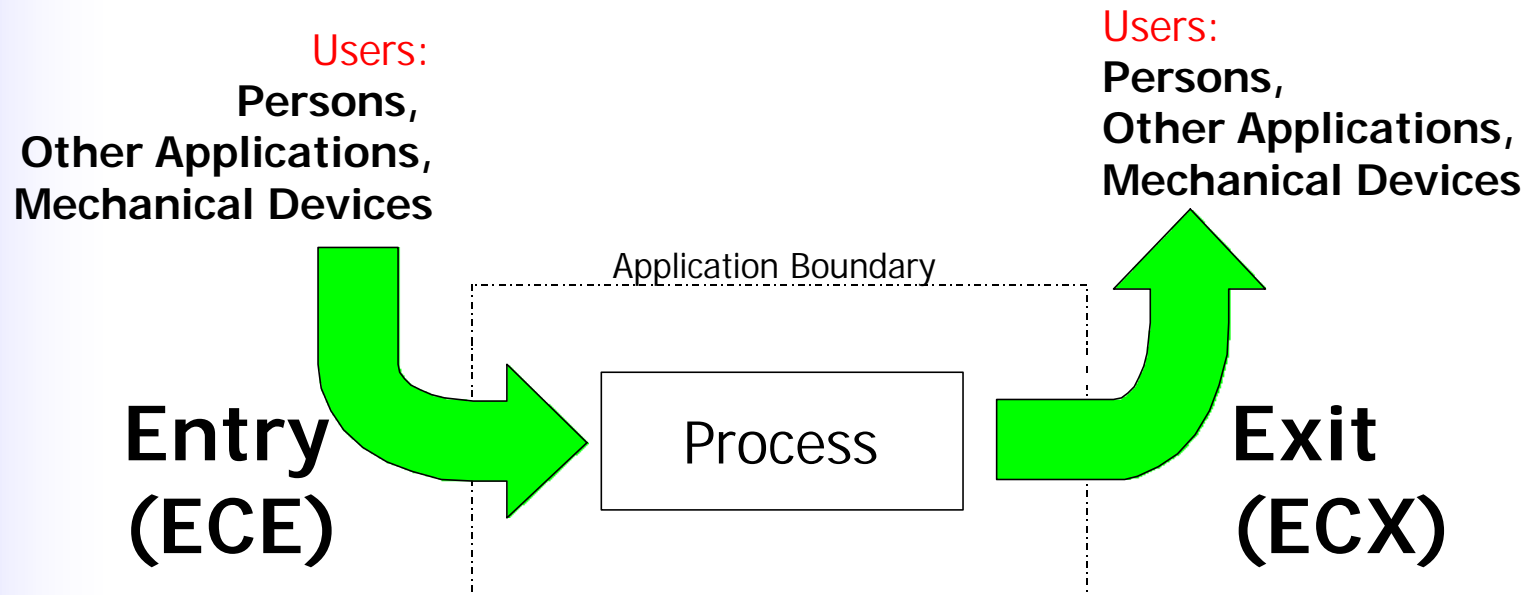
# Identifying transactions

## Key concepts - 4 types of sub-processes



# Identifying transactions

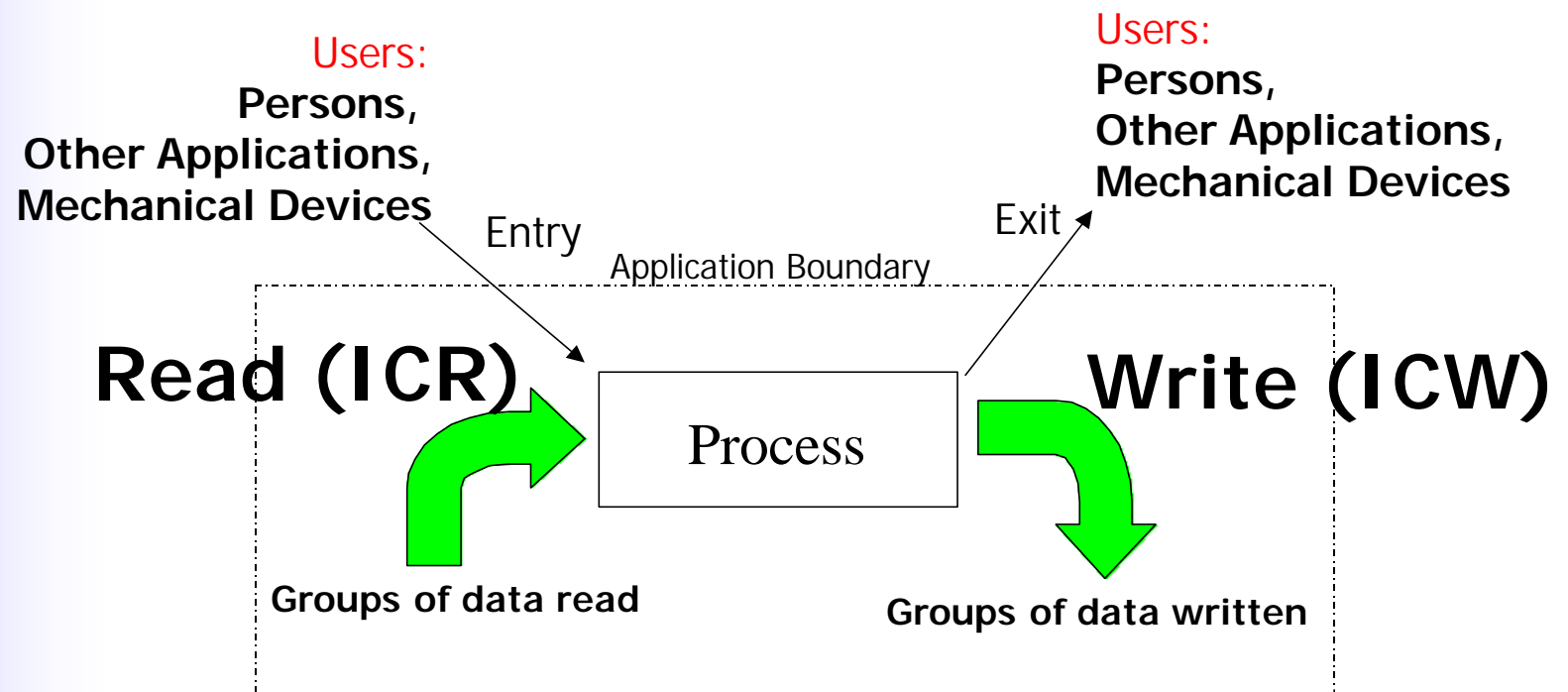
## Key concepts - 4 types of sub-processes





# Identifying transactions

## Key concepts - 4 types of sub-processes



# *Identifying transactions*

---

## Identification rules: ECE

- ⊙ The sub-process **receives** a group of data from outside the application boundary,
- ⊙ The sub-process is associated with **only one** group of data,
- ⊙ The sub-process does not **exit**, **read**, or **write** data,
- ⊙ The sub-process is unique: processing and data element types identified are different from other ECEs within the same process,
- ⊙ The primary trigger is an ECE.

# *Identifying transactions*

---

## Identification rules: ECX

- ⊙ The sub-process **sends** data external to the application's boundary.
- ⊙ The sub-process sends **only one** group of data.
- ⊙ The sub-process does not **receive, read, or write** data.
- ⊙ The sub-process is unique: processing and data element types identified are different from other ECXs of the same process.

# *Identifying transactions*

---

## Identification rules: ICR

- ⊙ The sub-process **reads** a group of data.
- ⊙ The sub-process reads **only one** group of data.
- ⊙ The sub-process does not receive, exit, or **write** data.
- ⊙ The sub-process is unique: processing and data element types identified are different from other ICRs of the same process.

# *Identifying transactions*

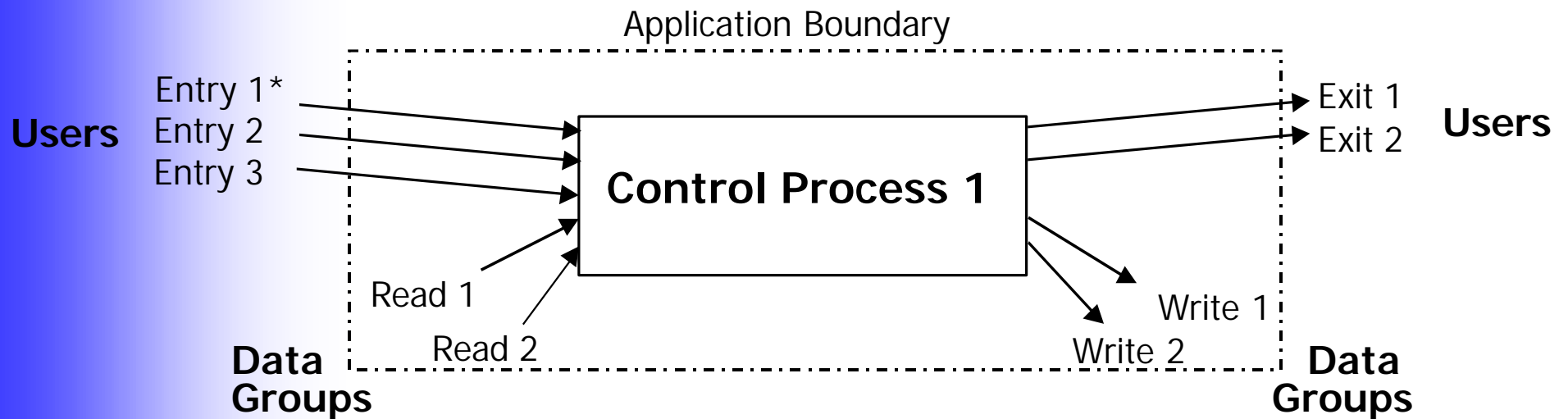
---

## Identification rules: ICW

- ⊙ The sub-process **writes** to a group of data.
- ⊙ The sub-process writes to **only one** group of data.
- ⊙ The sub-process does not **receive, exit, or read** data.
- ⊙ The sub-process is unique: processing and data element types identified are different from other ICWs of the same process.

# Identifying transactions

## Summary

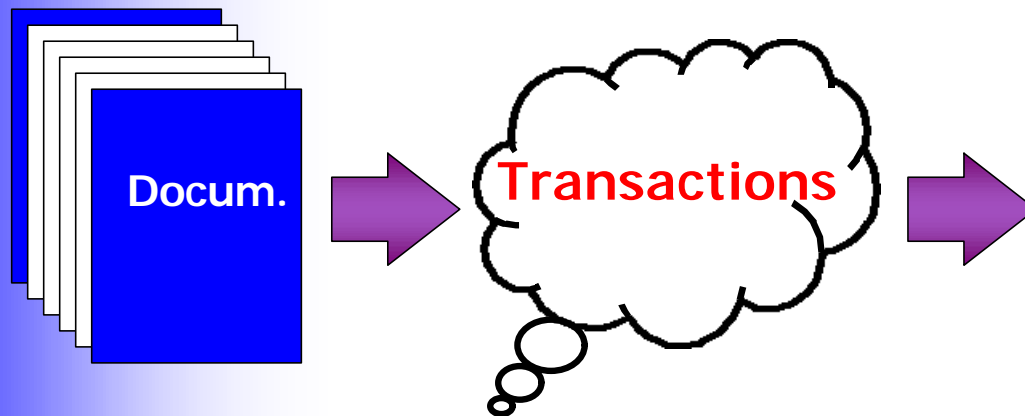


Each arrow is a sub-process.

\* Entry 1 is the trigger

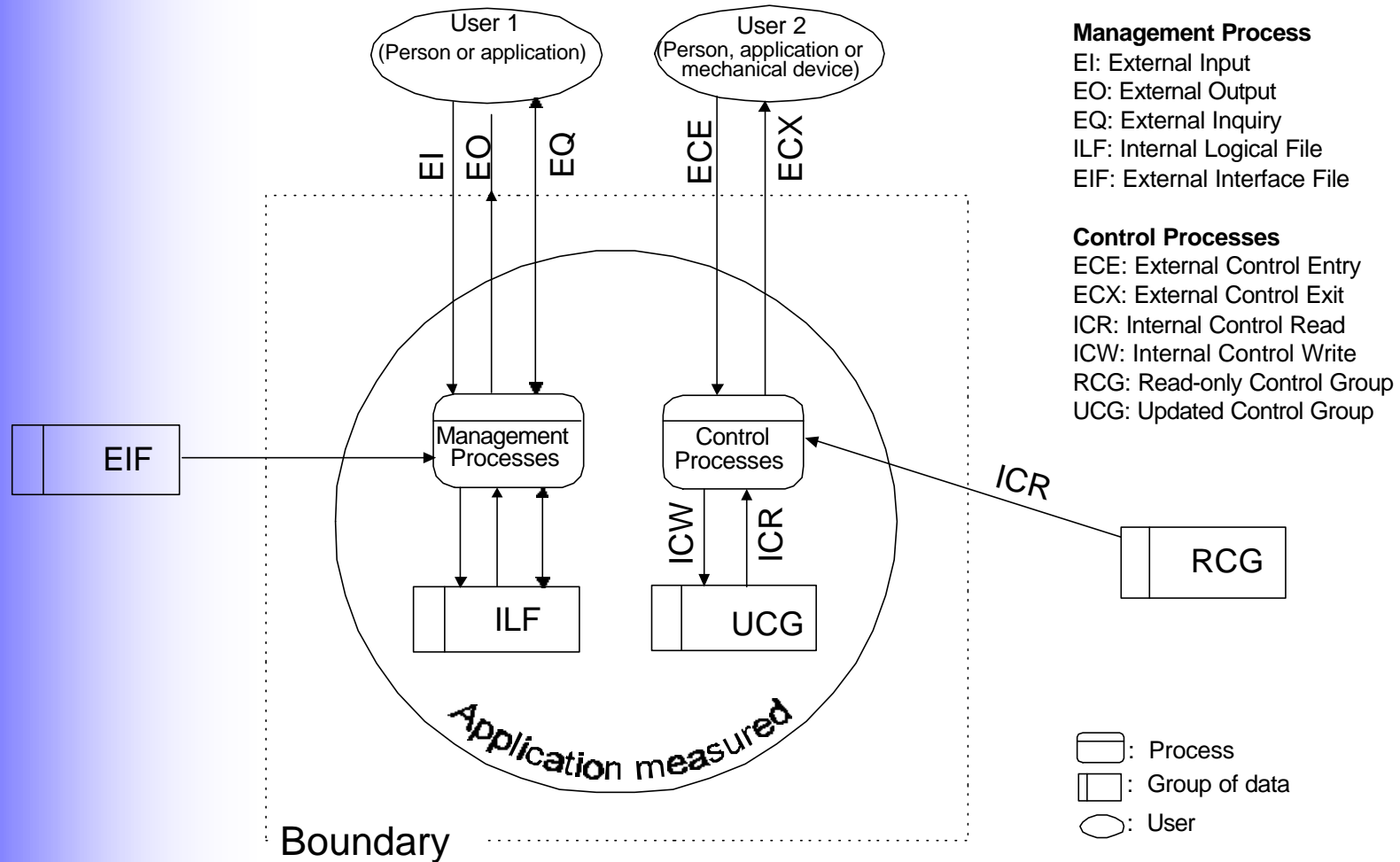
# Identifying transactions

## Summary



- **Trigger 1**
  - Control Process 1
    - Sub process 1.1
    - Sub process 1.2
    - ...
  - Control Process 2
    - Sub process 2.1
    - Sub process 2.2
- **Trigger 2**
  - Control Process 1
    - Sub process 1.1
    - ...

# Summary of Function Types





# *Assigning points to measured elements*

---

- ⊙ **Data points:**
  - ⊙ key concepts
  - ⊙ Assigning points to data
  - ⊙ Example
  - ⊙ Quick validation of data measurement
  
- ⊙ **Transaction points:**
  - ⊙ key concepts
  - ⊙ Assigning points to transactions
  - ⊙ Example
  - ⊙ Quick validation of transaction measurement

# *Data points: key concepts*

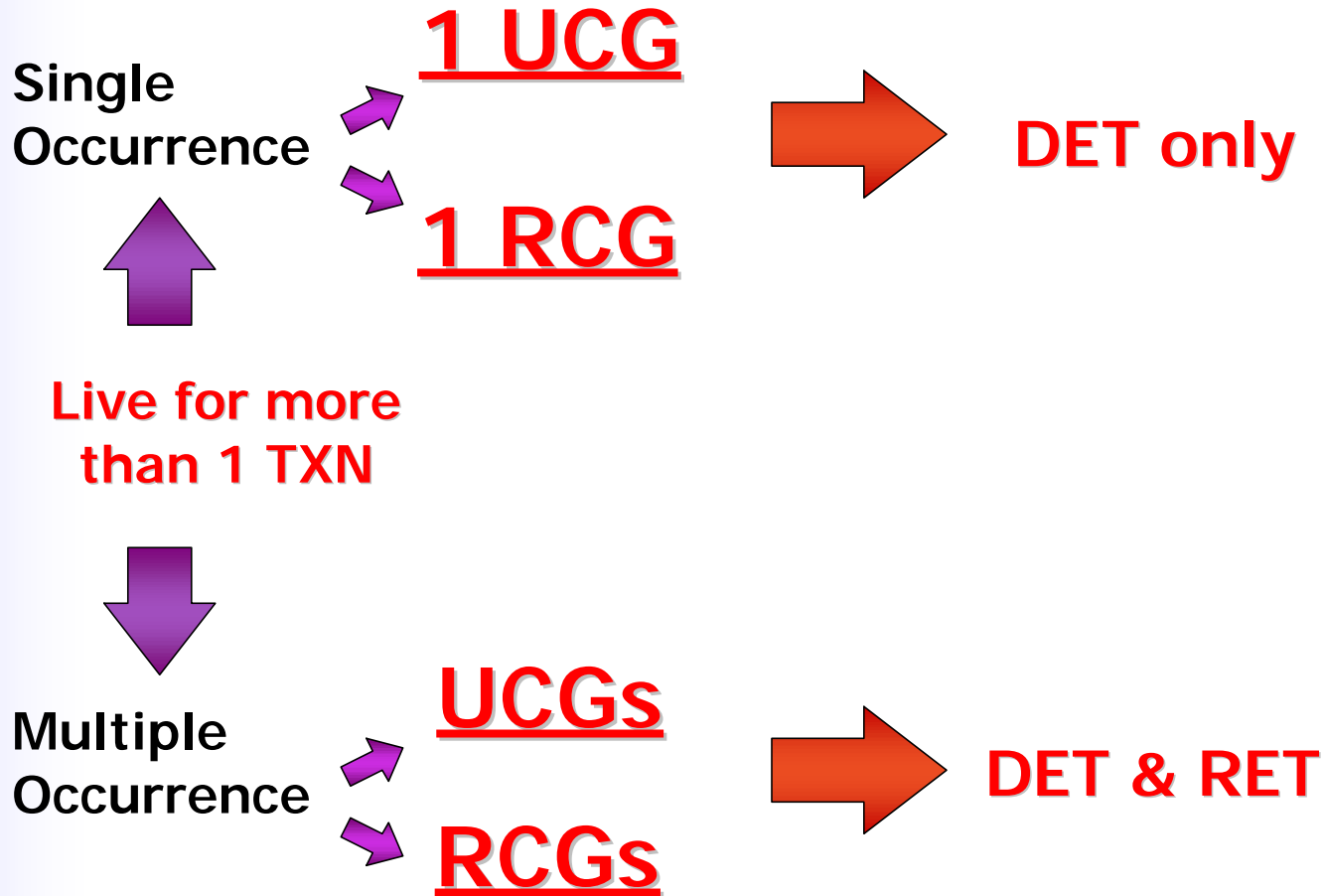
---

Points are assigned to data as a function of two characteristics:

**DET:** The number of data elements

**RET:** The number of user recognizable subgroup of data elements

# Assigning points to data



# *Assigning points to data*

---

## **Single occurrence Updated data (UCG):**

- ⊙ Point assignment is based on the number of data element types (DET)
- ⊙  $\text{Points} = (\text{number of DET} / 5) + 5$

**Note:** There is only one single occurrence UCG within an application. It comprises all the single occurrence updated values within the application being measured.

# ***Assigning points to data***

---

## **Single occurrence Read-Only Data (RCG):**

- ⊙ Point assignment is based on the number of data element types (DET)
- ⊙ Points = number of DET / 5

**Note:** There is only one single occurrence RCG within an application. It comprises all the single occurrence read-only values within the application being measured.

# Assigning points to data

---

## Multiple occurrence RCG and UCG:

DETs \ RETs	1 - 19	20 - 50	51 +
1	L	L	A
2 - 5	L	A	H
6 +	A	H	H

# *Assigning points to data*

---

## Multiple occurrence UCG and RCG:

	UCG	RCG
L = Low	7	5
A = Average	10	7
H = High	15	10

## Example - RCG mult. occ.

---

- **Temperature Data**
  - The temperature data for each cooking mode (Figure 2) is a *multiple occurrence group of data*, that is, there are more than one occurrence of the same type of record.
  - The temperature data are maintained *outside* the application boundary but *referenced* by the application to control the heater and the status indicators. This group of data could be therefore an RCG. The following table shows the evaluation of the RCG rules.
- Remember: All of the counting rules must apply (Yes) to count the group of data as a RCG.



# Example - RCG mult. occ.

---

RCG Counting Rules	Does the Rule Apply?
The group of data is either a logical related group of data <i>or</i> a single occurrence group of data	Yes. Temperature data are required to control the heater and the status indicator.
The group of data is <b>not updated</b> by the application being counted.	Yes. There is no process within the application that updates the temperature data.
The group of data is referenced by the application being counted.	Yes. The temperature data are referenced to control the heater and the status indicator.
The group of data lives for more than one transaction.	Yes. Each time the end user cooks rice, the temperature data are referenced.
The group of data has not been counted as an UCG, ILF or EIF for the application.	Yes. The rule applies because the group of data is not counted as an UCG, an ILF or EIF.



## ***Example - RCG sing. occ.***

---

- Selected cooking mode: This field keeps the cooking mode (fast, normal and gruel) selected by the end user. The default value of this field is 'normal'. If the end user does not select a mode, the rice is cooked in normal mode. Various processes of the application need to reference the selected cooking mode. Therefore, the selected cooking mode lives for more than one transaction.
- Target temperature: During cooking, the application receives the actual temperature from a sensor and update the target temperature every 30 seconds. It is referenced every 5 seconds by the process which controls the heater. Therefore, target temperature lives for more than one transaction.
- Elapsed time: During cooking, the elapsed time is continuously updated. It is used by the processes which calculates the target temperature and controls the heater. Therefore, elapsed time lives for more than one transaction.

# Example - RCG sing. occ.

DET Counting Rules	Does the Rule Apply?
Count a DET for each unique user recognizable, non recursive, field on the UCG.	Selected cooking mode Target temperature Elapsed time
Count a DET for each or piece of data in the UCG that exists because the user requires a relationship with another ILF or UCG to be maintained.	There is no data of this type.
Count physical implementation techniques as a single DET for the entire group of fields.	There are no fields of this type.

## Point assignment: RCG

Points = Integer part of (number of DET / 5)

Points = Integer part of (3 / 5) = **5**

# ***Quick validation of data measurement***

---

## **Check if:**

- ⊙ **All data live for more than one transaction**
- ⊙ **Repeated fields have been counted only once**
- ⊙ **Data updated in more than one application has been counted in each application**

# ***Transactions points: key concepts***

---

- ⊙ Points are assigned only at the level of the sub-process,
- ⊙ The functional size of a process is the sum of the points assigned to the set of its sub-processes.

Note that there is no upper limit to the size of a process.

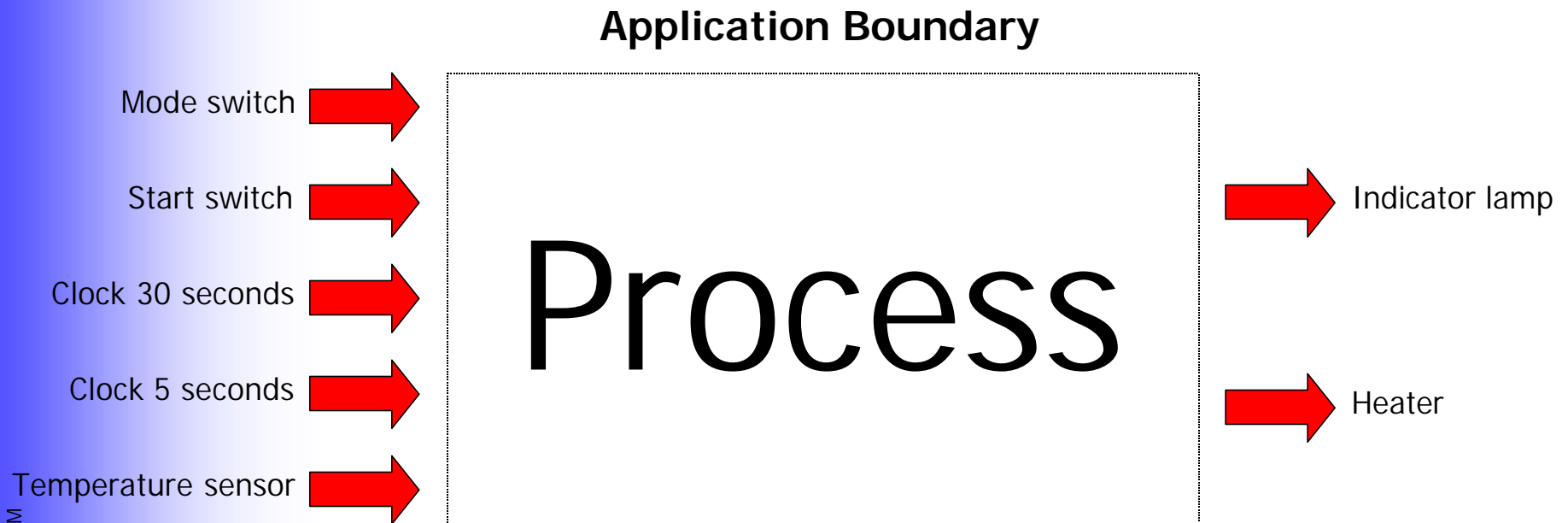
# ***Assigning points to transactions***

---

- ① Points assigned to a sub-process is a function of the number of DET manipulated by the sub-process

<b>DETs:</b>	1 to 19 DETs	20 to 50 DETs	51 + DETs
<b>Points:</b>	1	2	3

# Example





# Example

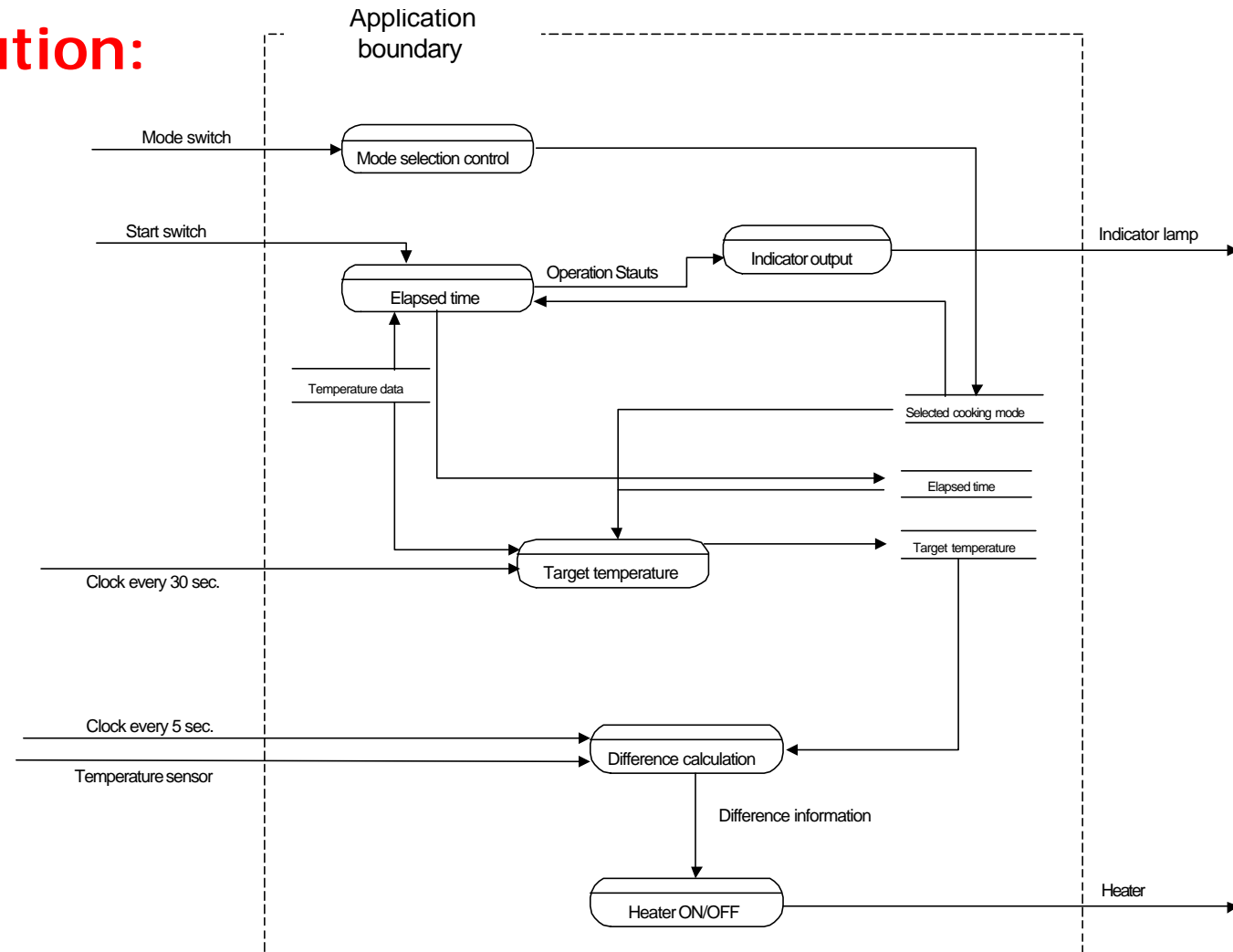
---

## Hints:

- ⊙ Start with the triggers, identify all **control processes** link to each trigger,
  
- ⊙ For each identified **processes**
  - ⊙ *Identify all **sub-processes***
  - ⊙ *Identify their transaction type*
  - ⊙ *Assign points to each **sub-process***

# Example

## Solution:



# ***Quick validation of transaction measurement***

---

- ⊙ Check that each **process** :
  - ✓ has at least one External Control Entry (**ECE**),
  - ✓ has at least one External Control eXit (**ECX**) or one Internal Control Write (**ICW**),
  - ✓ does not have duplicate sub-processes.

## Example

### Solution:

Logical file - Process/sub-process	Function Type	RET	DET	Point
<b>Control data function type</b>				
1: Temperature data (multiple occurrences)	RCG	1	4	5
2: Single occurrence UCG (single occurrence)	UCG	N/A	3	5
<b>Control transactional function type</b>				
1: Mode selection:				
1.1: Receive cooking mode	ECE	N/A	1	1
1.2: Update selected cooking mode	ICW	N/A	1	1
<b>Total points:</b>				<b>2</b>
2: Elapsed time :				
2.1: Receive start signal	ECE	N/A	1	1
2.2: Update elapsed time	ICW	N/A	1	1
2.3: Read selected cooking mode	ICR	N/A	1	1
2.4: Read cooking time	ICR	NA	1	1
2.5: Set the status indicator	ECX	N/A	1	1
<b>Total points:</b>				<b>5</b>

## Example

### Solution:

3: Target temperature:				
3.1: Clock trigger	ECE	N/A	1	1
3.2: Read selected cooking mode and elapsed time	ICR	N/A	2	1
3.3: Read 'temperature data' file	ICR	N/A	4	1
3.4: Update target temperature	ICW	N/A	1	1
<b>Total points:</b>				<b>4</b>
4: Heater control:				
4.1: Clock trigger	ECE	N/A	1	1
4.2: Receive actual temperature	ECE	N/A	1	1
4.3: Read target temperature	ICR	N/A	1	1
4.4: Set the heater ON/OFF	ECX	N/A	1	1
<b>Total points:</b>				<b>4</b>

Total FFP points = 25

# *Overview of field tests results*

---

- ⊙ Sources of data
- ⊙ First set: comparing FPA and FFP
- ⊙ Second set: relevance and usability
- ⊙ Third set: further comparisons FPA/FFP

# Available resources

---

- ⊙ **Complete documentation on the Web**
  - ✓ Concepts and definitions,
  - ✓ Counting Practice Manual,
  - ✓ Publications,
  - ✓ <http://www.lrgl.uqam.ca/ffp.html>
  
- ⊙ **Support available**
  - ✓ Case Study
  - ✓ On site custom training
  - ✓ Consulting support

# *First set*

---

- ⊙ Conducted by the research team in 1997,
- ⊙ 3 RT or embedded products measured,
- ⊙ 2 industrial partners participated,
- ⊙ **GOAL: Compare FFP with FPA (IFPUG 4.0)**



# First set

## Results...

	PRODUCT 1		PRODUCT 2		PRODUCT 3	
	TXN <sup>3</sup>	Points	TXN <sup>3</sup>	Points	TXN <sup>3</sup>	Points
<b>FPA<sup>1</sup></b>	54	256	9	38	32	123
<b>FFP<sup>2</sup></b>	753	777	40	46	468	479

Note 1: Using IFPUG 4.0 CPM, processes only

Note 2: Using FFP 1.0 CPM, processes only

Note 3: Number of processing transactions for which points are assigned

# *First set*

---

## Observations:

- ⊙ FFP results close to FPA when processes contain small number of sub processes,
- ⊙ FFP yield larger size measures when processes contain large number of sub processes,
- ⊙ Both methods require similar measurement effort

## *Second set*

---

- ⊙ Conducted without assistance from the research team in 1997,
- ⊙ Operational real-time products measured,
- ⊙ 1 industrial partner,
- ⊙ **GOAL: Evaluate FFP for relevance and usability**

## *Second set*

---

### Observations:

- ⊙ Functional coverage established at 97%, based on expected number of functions to be measured.
  
- ⊙ Concepts and procedures are:
  - ✓ Clear,
  - ✓ Easy to understand,
  - ✓ Usable without assistance of specialists

# *Third set*

---

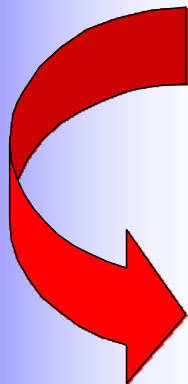
- ⊙ 4 industrial partners in North-America and Australia participated,
- ⊙ 10 software products measured:
  - ✓ 8 products related to the telecom business
  - ✓ 1 product related to power utility
  - ✓ 1 product related to the military sector
- ⊙ All products measured by the same individual (CFPS, 12 years exp. in FSM)

# Third set

**1<sup>st</sup> GOAL:** Compare IFPUG 4.0 and FFP

## RESULTS

Product	Type	FPA size	FFP size
A	Real-Time	210	794
B	Real-Time	115	183
C	Real-Time	N / A	2 604
D	Real-Time	43	318
<b>E</b>	<b>Mostly MIS</b>	<b>764</b>	<b>791</b>
F	MIS (batch)	272	676
<b>G</b>	<b>MIS</b>	<b>878</b>	<b>896</b>



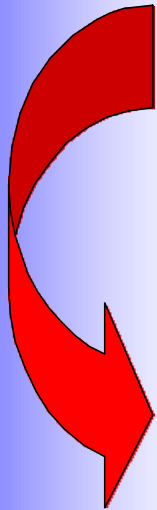
Size is similar when measuring typical MIS software products

# Third set

**1<sup>st</sup> GOAL:** Compare IFPUG 4.0 and FFP

## RESULTS

Product	Type	FPA size	FFP size
A	Real-Time	210	794
B	Real-Time	115	183
<b>C</b>	<b>Real-Time</b>	<b>N/A</b>	<b>2 604</b>
D	Real-Time	43	318
E	Mostly MIS	764	791
F	MIS (batch)	272	676
G	MIS	878	896



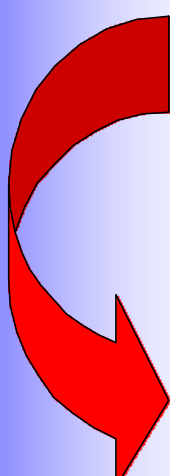
**One real-time software could only be sized with FFP**

# Third set

**1<sup>st</sup> GOAL:** Compare IFPUG 4.0 FPA and FFP

## RESULTS

Product	Type	FPA size	FFP size
<b>A</b>	<b>Real-Time</b>	<b>210</b>	<b>794</b>
<b>B</b>	<b>Real-Time</b>	<b>115</b>	<b>183</b>
C	Real-Time	N / A	2 604
<b>D</b>	<b>Real-Time</b>	<b>43</b>	<b>318</b>
E	Mostly MIS	764	791
<b>F</b>	<b>MIS (batch)</b>	<b>272</b>	<b>676</b>
G	MIS	878	896



Larger functional size for software products with numerous R-T processes (A, B and D); even for MIS with fewer direct user interactions (F).



# Third set

---

**1<sup>st</sup> GOAL:** Compare IFPUG 4.0 and FFP

What does it mean ?

	MIS product	RT product
FPA	200	200
FFP	~ 200	>> 200

Obviously, when considering RT products, FFP is measuring functionality that is not measured by IFPUG 4.0.

# Third set

## 2<sup>nd</sup> GOAL: Explore key economic ratios

### RESULTS

These 3 software products are all R-T software

Product	Size (FFP)	Effort (ph)	Duration (mth)	Unit effort (ph/FFP)	Sched. del. Rate (FFP/mth)
H	205	3 913	26	19	8
I	138	6 580	16	48	9
J	198	7 448	14	38	14

Until further data is available to allow statistically significant analysis, these should be interpreted as “order of magnitude” figures.

# Conclusion

---

- ⊙ International recognition
- ⊙ Benchmarking your results
- ⊙ The future of Full Function Points
- ⊙ Available resources
- ⊙ Final remarks
- ⊙ Acknowledgements

# *International recognition*

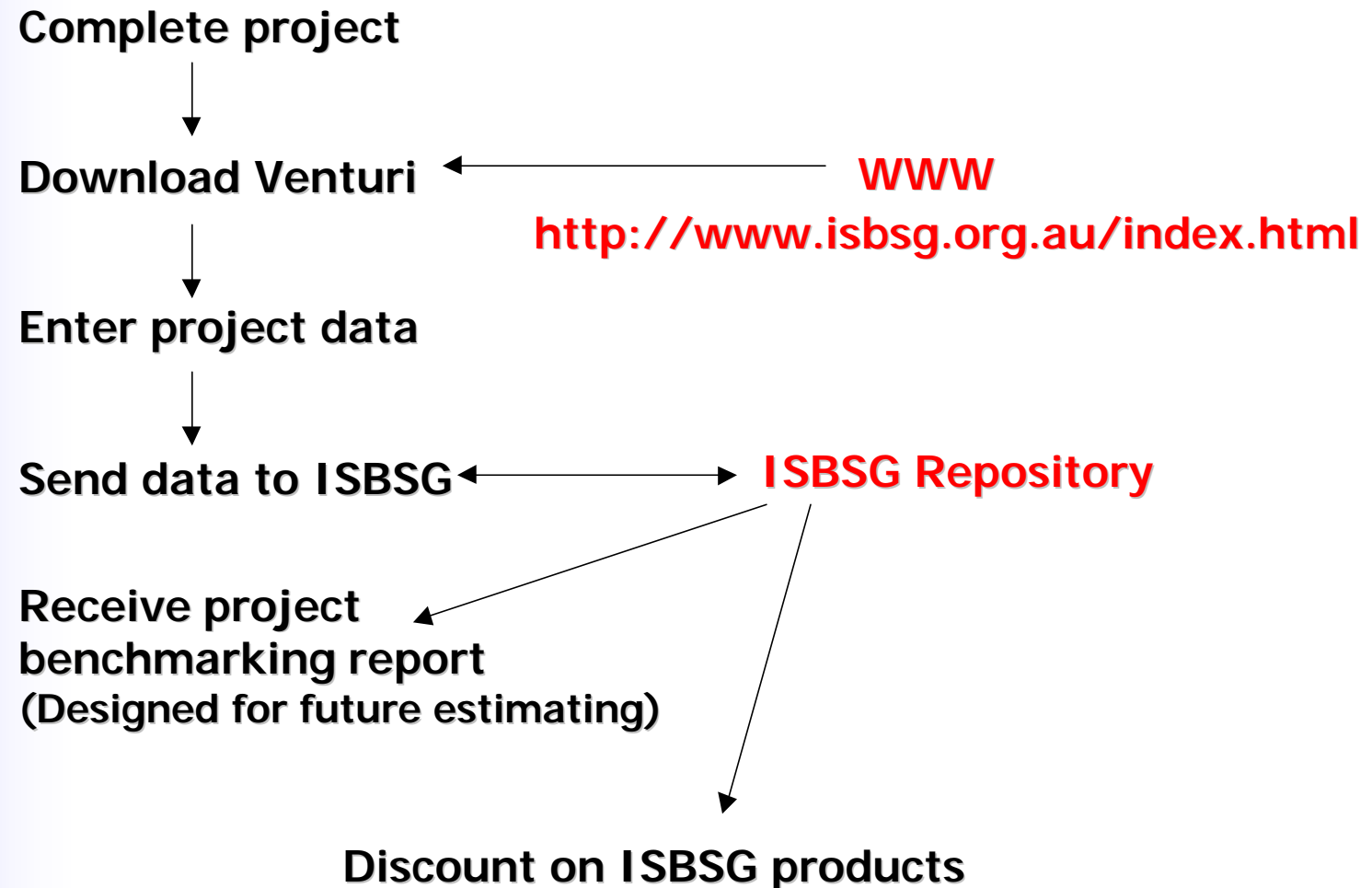
---

**In the Spring of 1998, FFP was accepted as a valid functional size measure by ISBSG\*, an international benchmarking organization.**

**ISBSG: International Software Benchmarking Standards Group**

# Benchmarking your results

---



# *The future of Full Function Points*

---

- ◉ Version 2 on its way for 1999,
- ◉ Looking for more industrial partners for field testing,
- ◉ Looking for more industrial partners for data collection,
- ◉ International Counting Practice Committee,
- ◉ ISO 14143 certification to start in 1999.

## *Final remarks...*

---

- ⊙ FFP addresses a problem identified since 1986,
- ⊙ FFP was designed for ISO compliance,
- ⊙ FFP has been designed FOR the industry, WITH the industry,
- ⊙ FFP is an open and transparent initiative, fully documented and easily available,
- ⊙ FFP is already helping organizations manage their non-MIS software.

# Sources of Funding

---

Developing FOR the industry, WITH the industry, FFP industrial partners...



Bell Canada, CANADA



Northern Telecom, Canada & USA



JECS Systems Research, JAPAN



Hydro-Québec, CANADA



# *Acknowledgements...*

---

- ① **The Software Engineering Management Research Laboratory of the Université du Québec à Montréal is supported through a partnership with Bell Canada.**
- ① **Additional funding is provided by the National Science and Engineering Research Council of Canada.**