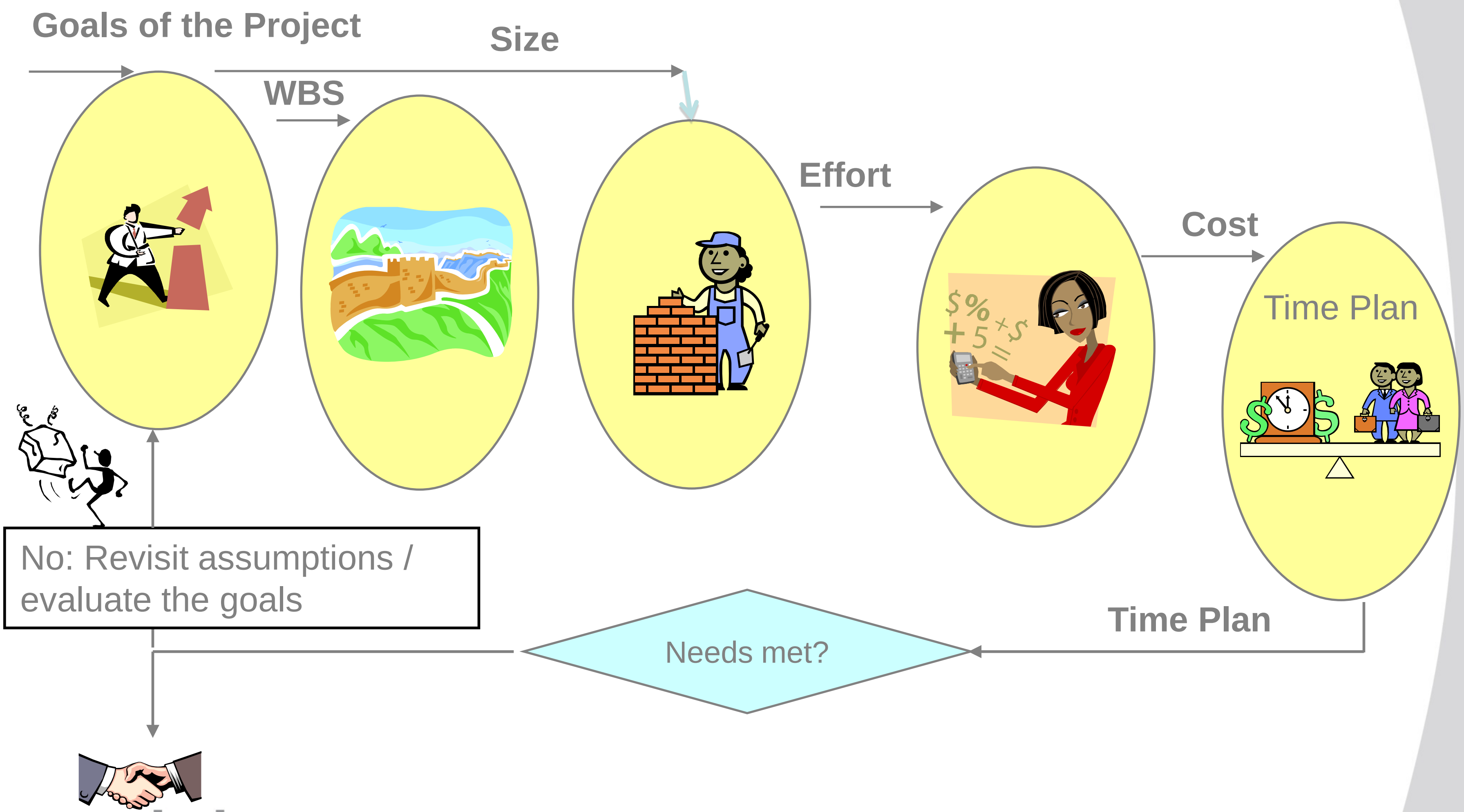
Applied Software Project Management

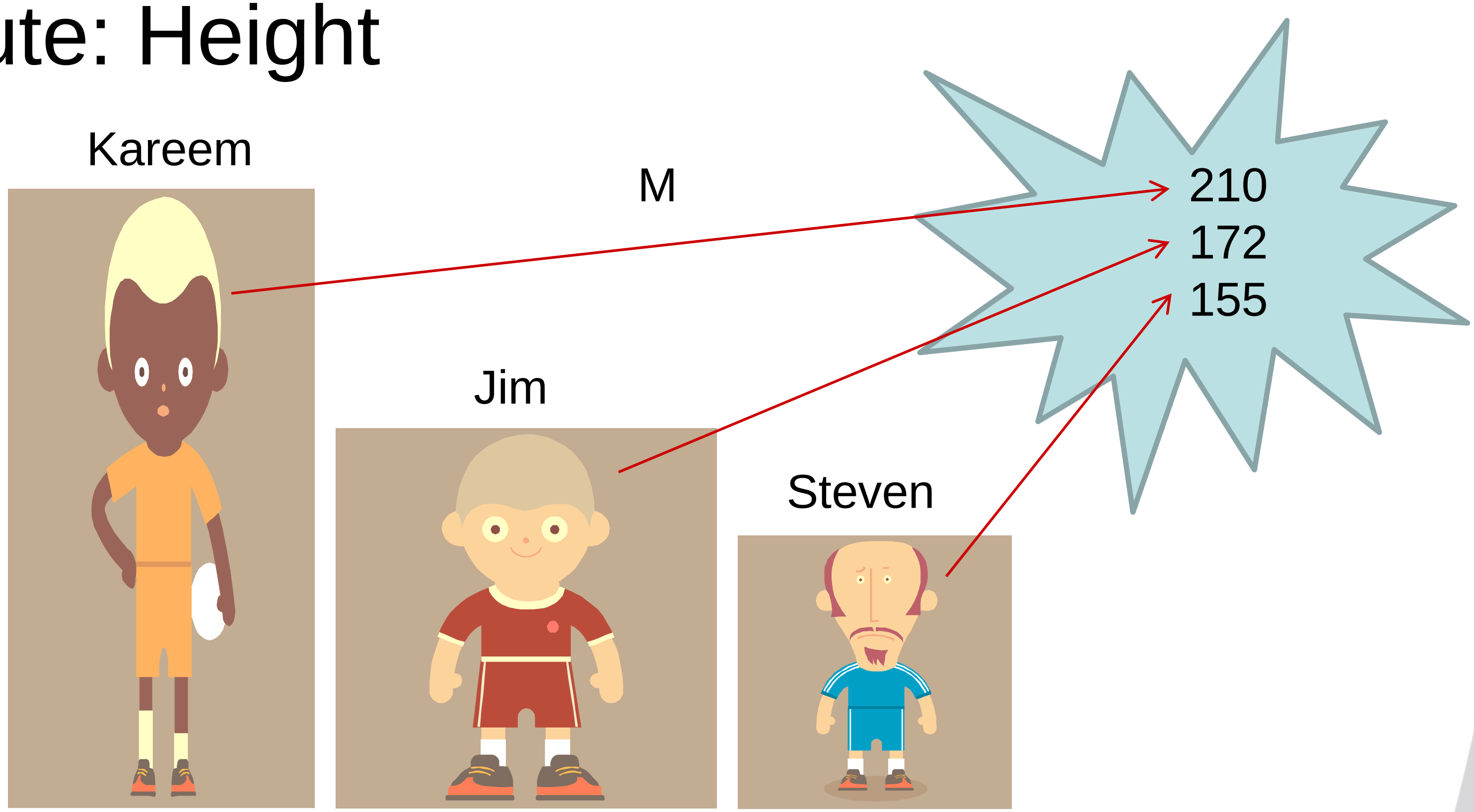
- **Software Size Measurement**
  - **Functional Size Measurement**
  - **COSMIC Function Points**
- **Software Effort Estimation**

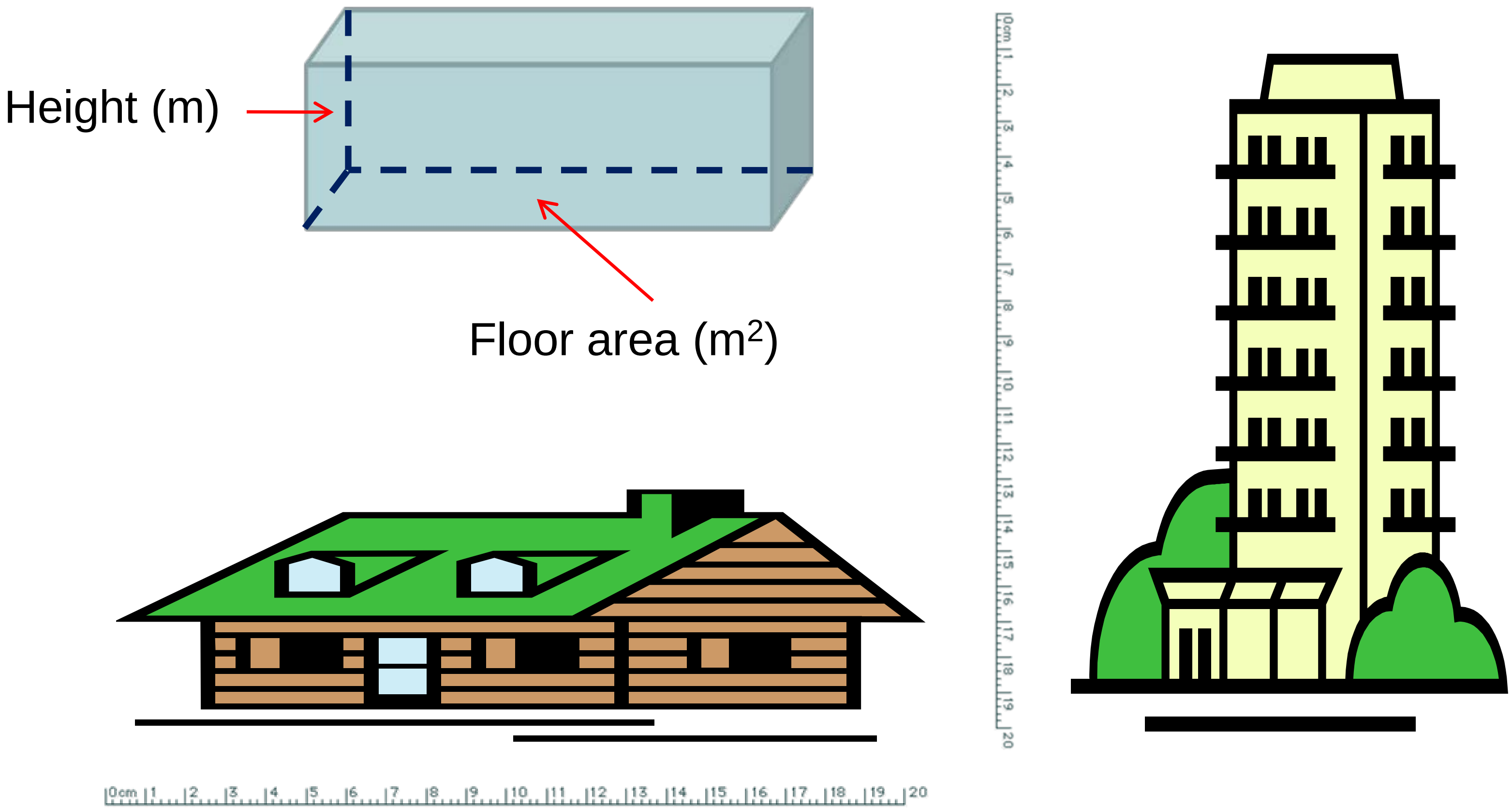
Software Project Planning Cycle



What is measurement?

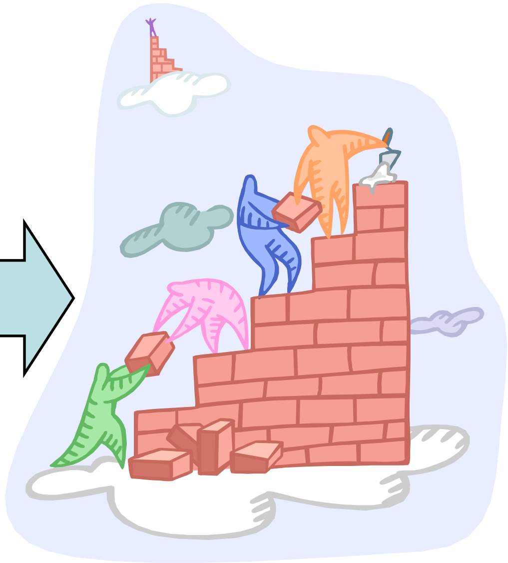
Entity: Person
Attribute: Height

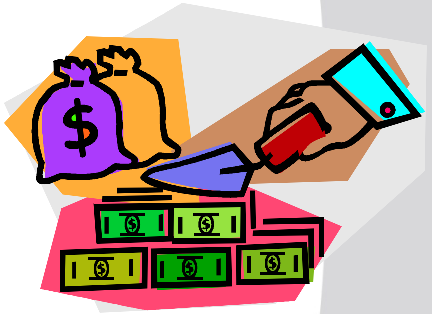




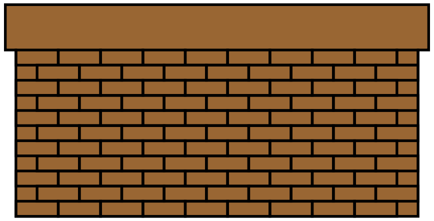
Size of an engineering product

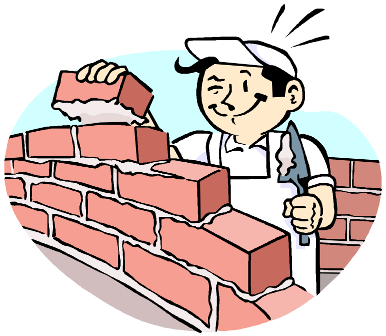
Size- Effort - Cost





10,000 km x 8 m x 6 m





20 m x 1 m x 1 m

Principles for Engineering

Projects should be completed:

- Within anticipated budget: Cost
- Within anticipated schedule: Time
- With conformance to customers' requirements: Quality

Size is the base measure!!

What is Software?

S
O
F
T
W
A
R
E

- Computer programs
- Configuration files used to set up these programs
- User documentation explaining how to use the software
- Support service
- System documentation describing the structure of the software

Software Size Aspects

- Software size was described with the following attributes in the past:
  - **Length**
    - The physical size of the product
  - **Functionality**
    - The functions supplied by the product to the user
  - **Complexity**
    - The characteristics of the underlying program that the software is solving
  - **Reuse**
    - The extent to which the software is genuinely new

Software Size

- Each product of software development can be described in terms of its size
  - SRS – number of pages or use cases
  - SDD – number of pages or classes
  - Code – number of lines (SLOC)
  - ….

Length

- There are three development products whose size would be useful to know:
  - Requirements specification
    - E.g., measuring the size of specification is a useful indicator of how long the design is likely to be, which in turn is a predictor of code length
  - Design description
  - Code

Length of Documents

- Specification and design documents:
  - Include text, graphs, mathematical diagrams and symbols
  - Their presentation depends on the particular style, method, or notation used
- It is necessary to identify atomic objects to measure the length
  - E.g., page is the most widely used atomic object in industry

View	Diagram	Atomic objects
Functional	Data-flow	Bubbles
	Data dictionary	Data elements
Data	Entity relation diagram	Entities, relations
State	State transition diagram	States, transitions

Length of Code

- Used to collect direct measures of software engineering output
- Size software products from the <u>developer's point of view</u>.
- The <u>key input</u> to many software cost, effort, productivity and quality estimations

SLOC

- Benefits
  - Simple and automatically measurable
  - Directly relates to the end product
  - Correlates with programming effort
- Criticisms
  - Vague definition
  - Language dependability
  - Developers' skill dependability
  - Not available for early planning

Length of Code: Example

```
If A>B
    then A - B
    else A + B;
```

```
If A>B
    then
       begin
              A - B
       end
    else
       begin
              A + B
       end;
```

Physical Length:
    Left: 3 LOC
    Right: 9 LOC

Logical Length:
Every logical construct is a line
    Left and right: 1 LOC

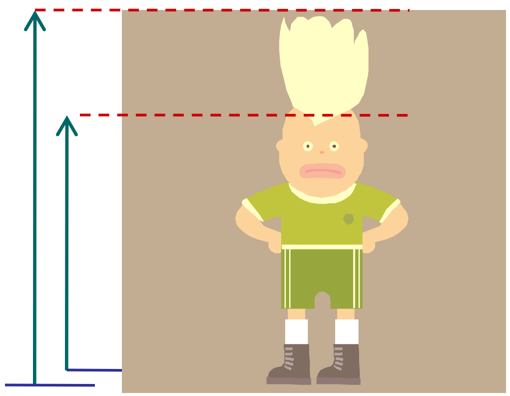
Count keywords (if, then, else, begin, …)
    Left : 5 LOC
    Right: 9 LOC

Length of Code (cont'd)

- Should specify how to <u>handle blank lines</u>, <u>comment lines</u>, <u>data declarations</u>, etc.
  - E.g., Hewlett-Packard definition of LOC
    - NCLOC: non-commented source LOC or effective LOC (ELOC)
    - CLOC: commented source LOC
    - Total length (LOC) = NCLOC + CLOC
    - The ratio CLOC/LOC measures the density of comments
- It is reported that count can be as much as <u>five times larger</u> than another, due to the difference in counting technique (Jones, 1986)

Measurement Models

- Why do we need models for measurement?
  - In most situations, an attribute may have a different intuitive meaning to different people
  - E.g., height of humans
    - Include hair height,
    - allow shoes to be worn, etc.



  - Every measure should be associated with a model of how the measure **maps the entities and attributes in the real world to the elements of a numerical system**

Example: PSP LOC Counting Template

Count Type	**Type**	**Comments**
Physical/Logical		
Statement Type	**Included**	**Comments**
Executable		
Nonexecutable:		
Declarations		
Compiler Directives		
Comments		
On own lines		
With source		
Banners		
Blank lines		
Clarifications		**Examples/Cases**
Nulls		continues, no-ops, ...
Empty statements		“;;”, lone ;’s, etc.
Generic instantiators		
Begin...end		when executable
Begin...end		when not executable
Test conditions		
Expression evaluation		when used as sub program arguments
End symbols		when terminating executable statements
End symbols		when terminating declarations or bodies
Then, else, otherwise		
Elseif		
Keywords		procedure division, interface, implementation
Labels		branch destinations when on separate lines

Summary

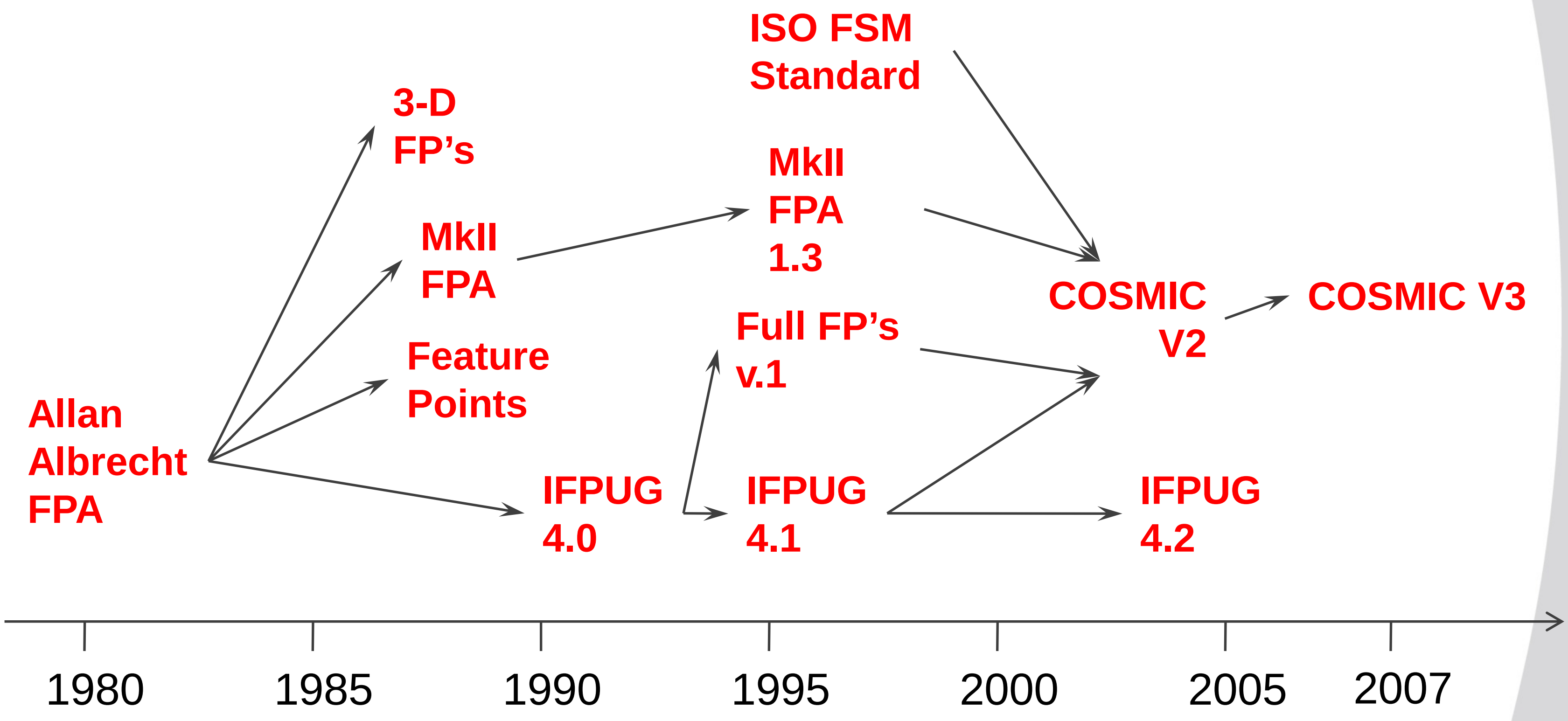
- Size is a base measure to estimate effort and cost for creating software
- Software is more than just code
- LOC is widely used to measure size, but
  - not standardized measure
  - not available early in the project to support estimation

Functional Size

- The amount of functionality a software provides as described in a functional requirements specification [Albrecht, 1983]

- Provide indirect measures of software which focus on <u>functionality and utility</u>.

History of Functional Size Measurement (FSM) Methods

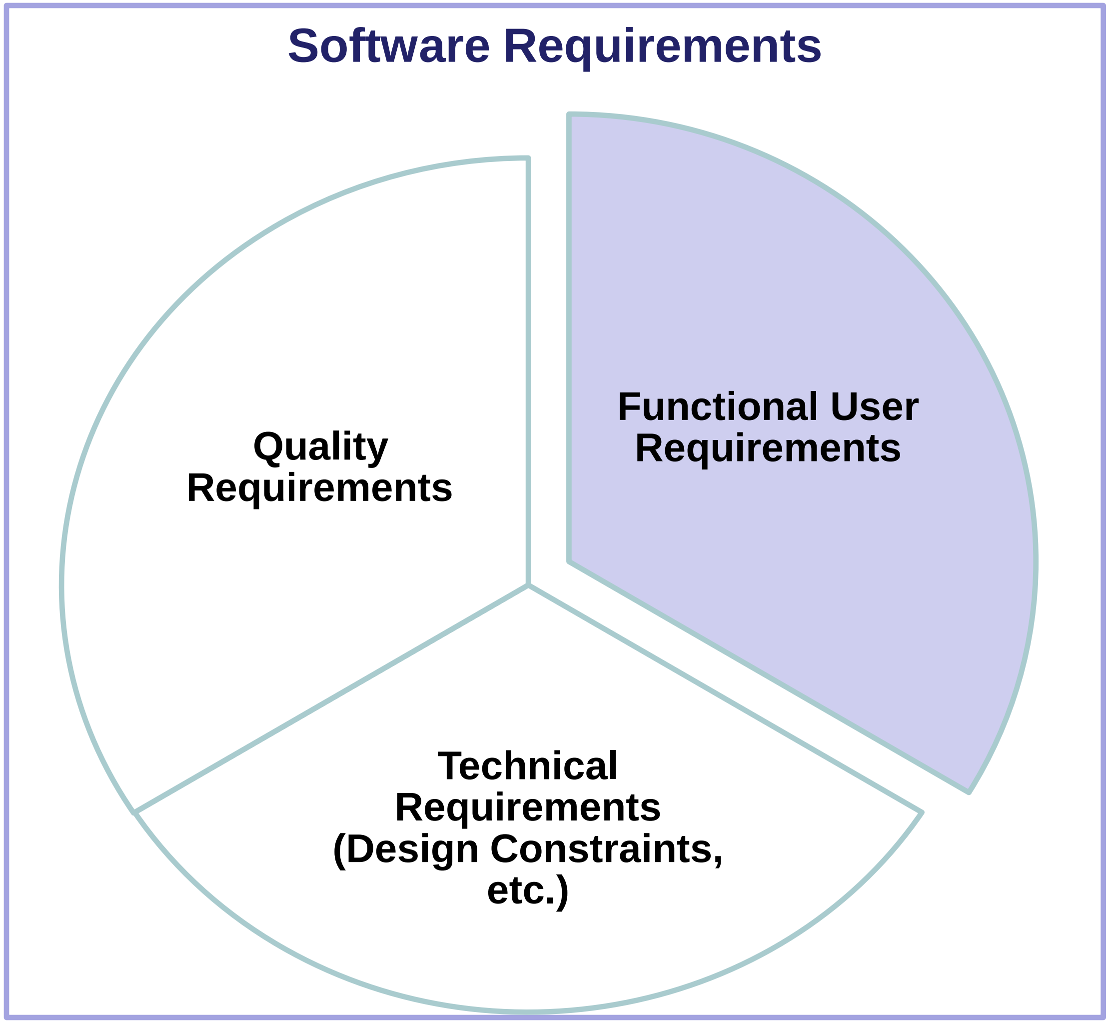


ISO 14143-1 Terminology for FSM (I)

- Functional User Requirements (FUR): A sub-set of the user requirements. The FURs represent the user practices and procedures that the software must perform to fulfill the users' needs.

- Functional Size: A size of the software derived by quantifying the FUR.

Functional Size Measures



Applicability of FSM Methods

- FUR can be extracted from software engineering artifacts BEFORE the software exists… (using UML for instance).. Inputs:
  - Requirements definition artifacts
  - Data analysis / modeling artifacts
  - Artifacts from functional decomposition of requirements
- FUR can also be extracted from software engineering artifacts AFTER the software has been constructed... Inputs:
  - Physical programs and screens
  - Software operations manuals and procedures
  - Physical data storage artifacts

IFPUG FPA (ISO/IEC 20926 : 2003)

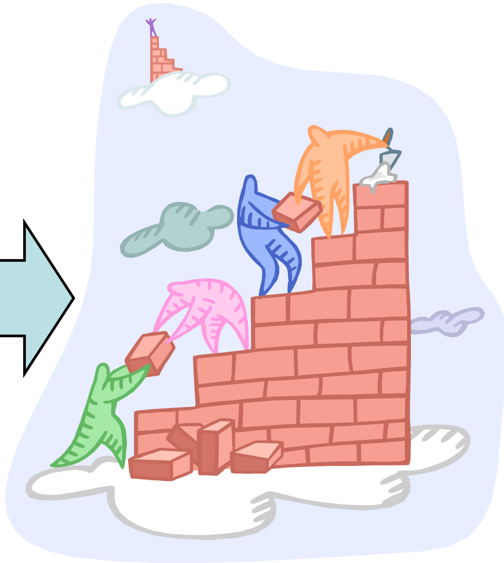
- In 1979, Albrecht developed the original Function Points method
- In 1986, an International Function Point Users' Group (IFPUG) was set up
- Since then, IFPUG has been clarifying FP counting rules and expanded the original description of Albrecht
- Designed to measure the <u>business information systems</u>

COSMIC FP (ISO/IEC 19761: 2003)

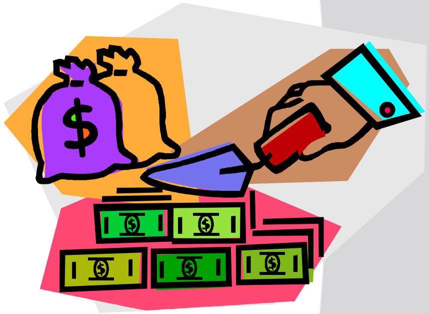
- Published by Common Software Measurement International Consortium (COSMIC) in Nov’99
- Becoming popular: Measures the functional size of software for both “business application” (or MIS or ‘data -rich’) software and “real-time” software and hybrids of these
- Last version: 2009

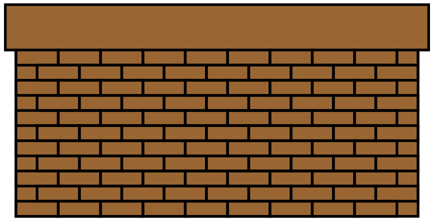
Effort Estimation

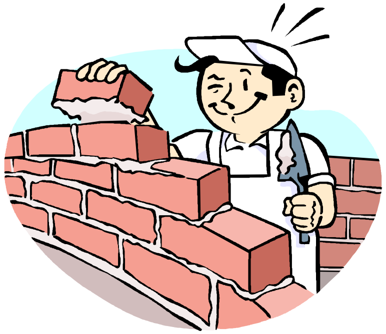
Size- Effort - Cost



10,000 km x 8 m x 6 m







20 m x 1 m x 1 m

Effort/Cost Estimation

- Software estimation is the process of predicting the amount of effort required to build a software.
- Cost estimates are needed throughout the life cycle!
  - Preliminary estimates are required for bidding for a contract, or determining whether a project is feasible (very difficult to obtain, and often the least accurate)
  - Periodic re-estimation is required to re-allocate resources when necessary

Software is…

"… different [from other engineering disciplines] in that we take on novel tasks every time. The number of times [civil engineers] make mistakes is very small. And at first you think, what's wrong with us? It's because it's like we're building the first skyscraper every time."

-- Bill Gates (1992)

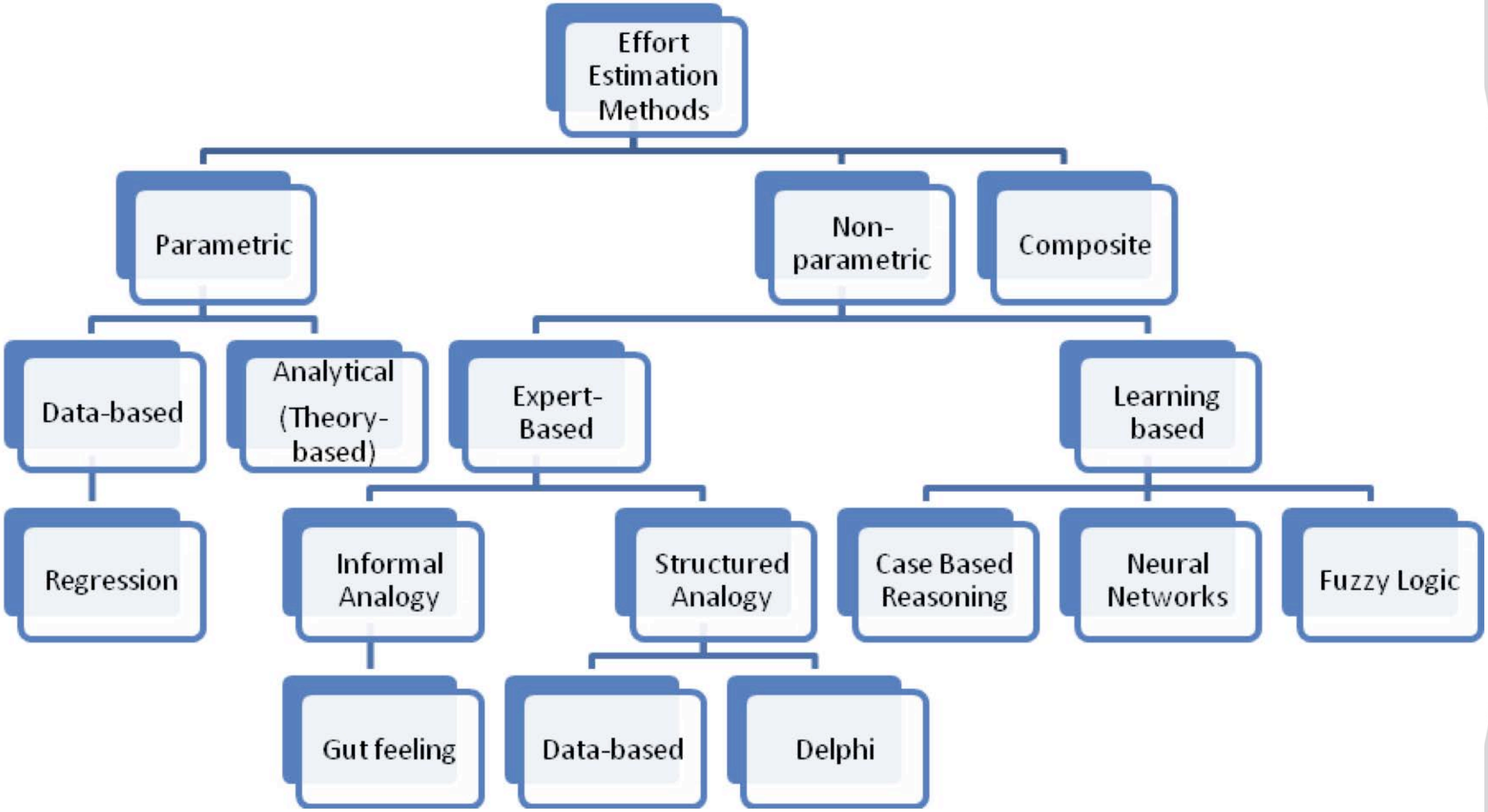
Effort Estimation Challenges

- Every new software project is likely to be different from the past ones
  - Application domain, hardware, supporting tools, techniques, staff, etc. may differ
  - Lack of the advantage of repeatability of tasks
- Often, we are solving a problem that has never been solved before (creative rather than constructive)
  - Likelihood of changes in requirements and design
- Some estimating problems are political
  - Estimates are set as targets, or estimation parameters are fit into an already-given outcome - E.g., price-to-win

A study on effort estimation accuracy (ISBSG)

- In 2005, International Software Benchmarking Standards Group (ISBSG) analyzed project duration, effort, cost and size estimates using the data from over 400 completed software projects in the ISBSG data repository.
- Among those, effort attribute is found to be estimated worst.
  - It is found that <u>less than one quarter of projects are estimated accurately</u> and on average <u>the actual effort was about double</u> the estimate.
  - About <u>60% of the projects underestimate effort by at least 10%</u>.
  - Moreover, significant errors are observed; for instance, <u>actual effort utilized has become 20 times the estimate.</u>

Effort Estimation Methods



Source: Gencel, C. and Symons, C. Re-thinking Estimating, Benchmarking and Performance Measurement, 2010.

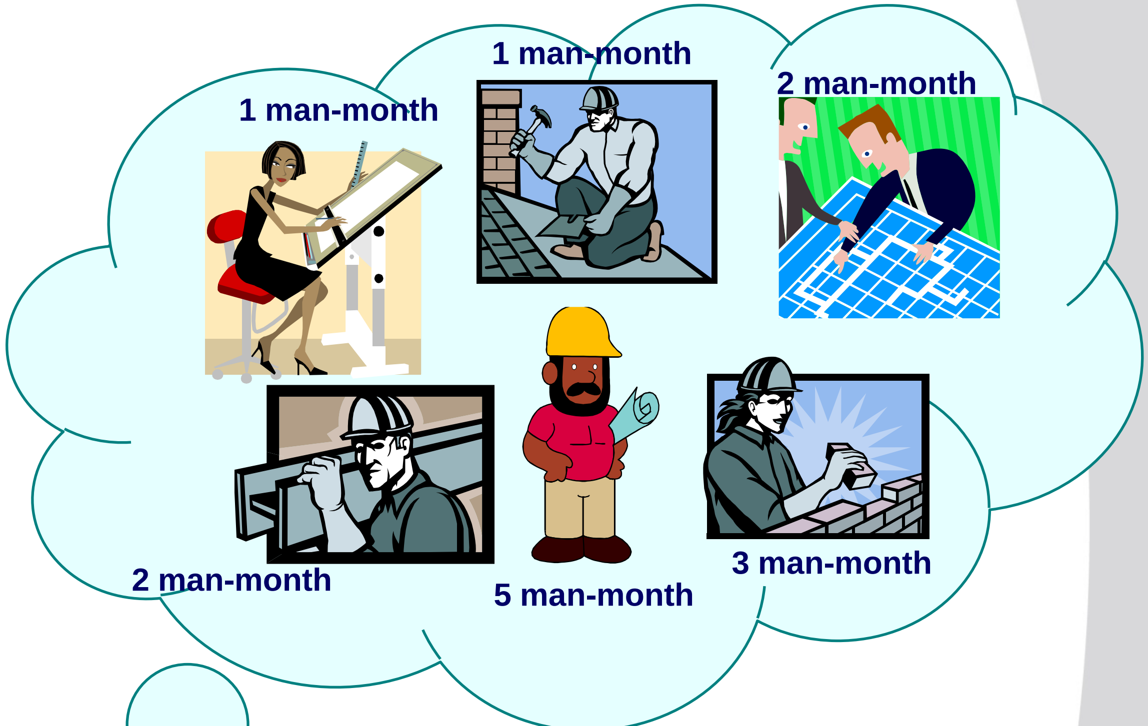
Approaches to Cost Estimation

- Bottom-up
  - Begins with the lowest-level parts of products or tasks, and provides estimates for each
  - Combines the low-level estimates into higher-level ones
- Top-down
  - Begins with the overall process or product
  - A full estimate is made, and the estimates for the component parts are calculated as relative portions of the whole

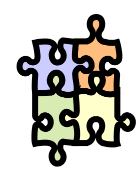
Bottom-Up Approaches

1) Decompose project into work activities (Work Breakdown Structure)

2) Estimate effort to be utilized for each activity based on past productivity values for each activity



3) Sum them up to find the total effort to be utilized for the whole project.



Productivity Based Estimation

Productivity = Software size / Project effort

$$\text{Estimated new project effort} = \frac{\text{Estimated software size}}{\text{Assumed project productivity}}$$

$$\text{Estimated new project effort} = f\left\{\frac{\text{Estimated software size}}{\text{Assumed project productivity}}, \text{Cost factors}\right\}$$

Reliable software sizing is critical for productivity measurement and for estimating effort!

Productivity in Hardware Industries

- Compares what goes into a process and what comes out
  - Increasing the inputs should increase outputs
  - Improving the process for the same inputs should increase the outputs
    - E.g., by speeding up the assembly line or adding more personnel, we expect to finish on time

Productivity in Software Engineering

- Requires to consider different perspectives:
  - The followings are not definite:
    - What constitutes the set of inputs
    - How process changes affect the relationship of input to output
  - Added personnel can affect productivity and quality, but not necessarily positively
    - If you throw more people on to a late software project, you will make it later [Brooks, 1975]
  - Productivity measurement has some problems!
    - Recording effort is difficult and not straightforward
      - E.g., full-time, part-time, hours in a working day, etc
    - Views output only in terms of length or functionality, and ignores the quality of the output

Project Related Factors Affecting Productivity

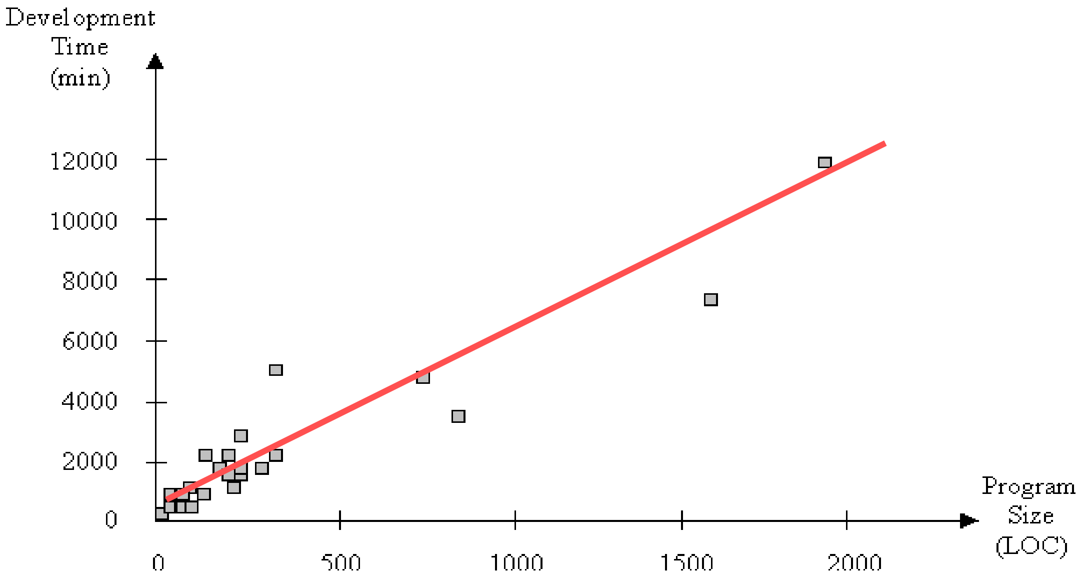
- Application Type
- Defect Density
- Team Size
- Development Type (new, re-dev., enhancement)
- Business Type
- Programming Language Type
- …

Cost Estimation Models

- Cost models
  - Provide direct estimates of effort or duration
  - Often have one primary input (i.e. size), and a number of adjustment factors (*cost drivers*)
    - Cost drivers are the characteristics of the project, process, products, or resources that are expected to influence effort or duration in some way
    - E.g., COCOMO
- Constraint models
  - Demonstrate the relationship over time between two or more parameters of effort, duration, or staffing level
    - E.g., SLIM

Regression-Based Models

- The model is derived using regression analysis on data collected from past software projects
  - The line of best fit is calculated for the data points



Regression-Based Models (cont'd)

- If the primary cost factor (size) were a perfect predictor of effort, then every point on the graph would lie on the line of best fit
  - However, there is a significant residual error in reality
    - It is necessary to identify the factors that cause variation between estimated and actual effort
    - These parameters are added to the model as cost drivers

Regression-Based Models (cont'd)

- The regression-based cost models take the form

  $E = (a\ S^b)\ F$

  - E is the effort in person months
  - a and b are empirically derived constants
  - S is the primary cost factor (typically, either LOC or FP)
  - F is the effort adjustment factor

COCOMO

- Constructive Cost Model (COCOMO)
  - Original COCOMO
    - Proposed by Boehm in 1970s
    - Based on data from a large set of projects at TRW
    - Brought economics viewpoint into software engineering
  - COCOMO 2.0
    - Original COCOMO was updated by Boehm and his colleagues (1995)
    - Updated according to the technological changes over 15 years
      - E.g., use of tools, reengineering, application generators, object-oriented approaches, etc.

Original COCOMO

- A collection of three models
  - Basic model
    - Used at the beginning of development when little is known about a project except its mode and likely size
  - Intermediate model
    - Applied after requirements are specified when more is known about staff, language, tools, etc.
    - Estimators select a rating for 15 cost drivers
  - Advanced model
    - Applied when design is complete (individual software modules have been identified)
    - Intermediate COCOMO is applied at the component level, and a phase-based model is used to build up an estimate of the project

Original COCOMO (cont’d)

- Development modes
  - Organic system
    - Involves data processing, tends to use databases and focus on transactions and data retrieval
      - E.g., a banking or accounting system
  - Embedded system
    - Contains real-time software that is an integral part of a larger, hardware-based system
      - E.g., a missile guidance system, or a water temperature sensing system
  - Semi-detached system
    - Something between organic and embedded

Original COCOMO (cont’d)

- Cost drivers
  - Rated on an ordinal scale with six points, each assigned an adjustment factor value derived for TRW environment
    - [Very low, low, normal, high, very high, extra high]
    - E.g., very high reliability has an adjustment factor value of 1.40
  - Overall effort adjustment factor is calculated by multiplying 15 independent cost driver values

Product attributes:	Required software reliability	
	Database size	
	Product complexity	
Process attributes:	Use of modern programming practices	
	Use of software tools	
	Required development schedule	
Resource attributes:	**Computer attributes**	**Personnel attributes**
	Execution time constraints	Analyst capability
	Main storage constraints	Applications experience
	Virtual machine volatility	Programming capability
	Computer turnaround time	Language experience
	Virtual machine experience	

Original COCOMO (cont'd)

$$\text{Effort} = a \times \text{Size}^b$$

Size; thousands of delivered source instructions (KDSI)
Effort; person-months (19 days/month or 152 hr/month)
a, b: depend on the development mode, determined by the type of software under construction

Original COCOMO (cont'd)

$$\text{Duration} = a \times \text{Effort}^{b}$$

Duration; months
Effort; person-months (19 days/month or 152 hr/month)
a, b: depend on the development mode, determined by the type of software under construction

COCOMO 2.0

- Useful for a much wider collection of techniques and technologies
  - Experiences with Original COCOMO showed that the model is inflexible and inaccurate for new techniques and technologies
  - http://sunset.usc.edu/research/COCOMOII/
- Incorporates models of reuse, takes into account maintenance and the change in requirements over time
- Allows to use different sizing models as we know more about the system during development
  - *Object Points, Function Points, LOC*

COCOMO 2.0 Estimation Process

Stage-1

- Project usually builds prototypes to resolve high-risk issues (user interfaces, SW-system interaction, performance, etc)
  - Little is known about the likely size of the final product
  - Size is estimated in terms of Object Points

COCOMO 2.0 Estimation Process

Stage-2

- A decision has been made to move forward with development, but the designers must explore alternative architectures and concepts of operation
  - Still, there is not enough information to support fine-grained effort and duration estimation
  - Size is estimated in terms of Function Points

COCOMO 2.0 Estimation Process

Stage-3

- Development has begun
  - Far more information is known
  - Size is estimated in terms of LOC

Summary

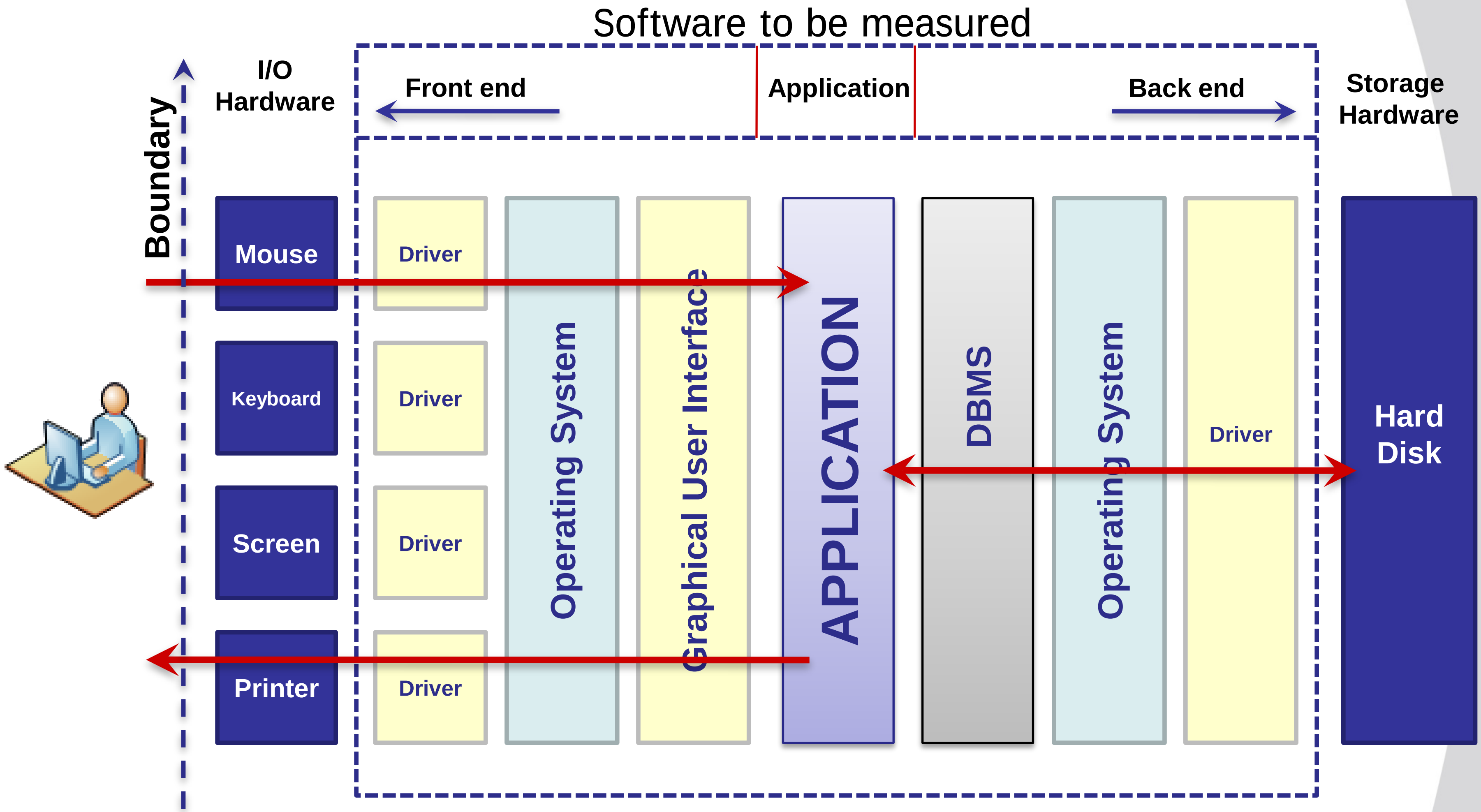
- Effort/Cost estimation is a difficult problem (depends on many factors)
- Size is the base measure and must be estimated reliably (this is hard too)
- COCOMO is an empirically developed cost estimation model that is parameterized by:
  - Size: LOC or FP (concretely IFPUG)
  - Cost drivers (15 factors)

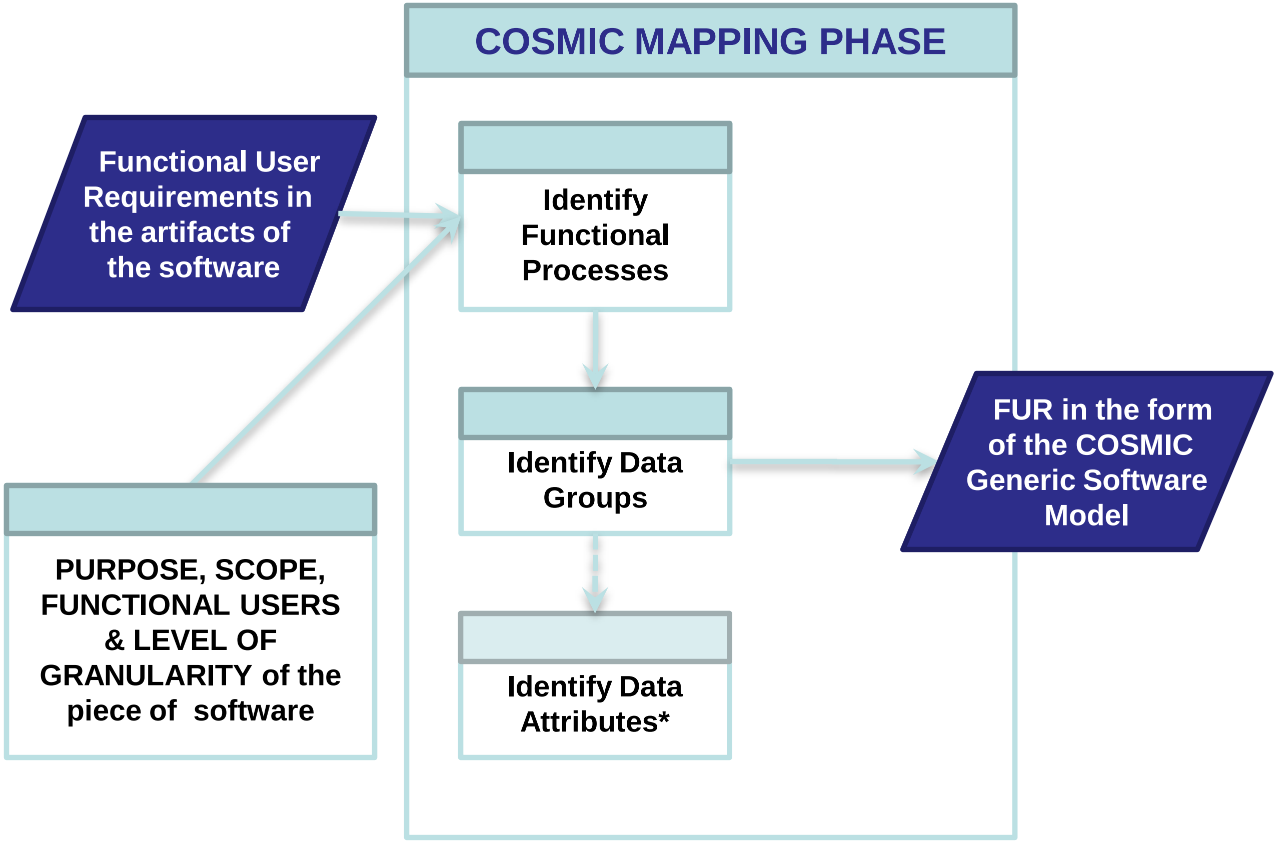
COSMIC Method (ISO/IEC 19761)

Alternatives:

- IFPUG Function Point Analysis (ISO/IEC 20926)
- Mark II Function Point Analysis (ISO/IEC 20968)
- NESMA Functional Size Measurement Method (ISO/IEC 24570)
- FİSMA Functional Size Measurement Method (ISO/IEC 29881)

How does it work?





** This step is performed only when a sub unit of measure is required*

The Boundary

- The conceptual interface between the user and the software being measured, across which Entries and Exits flow.

- Identify the functional user(s) that interact with the software being measured.

- The boundary lies between the functional users and this software.

Identifying the Functional Users

- **User**: Anything that interacts with the software being measured
- **Functional User**: A (type of) user that is a sender and/or an intended recipient of data in the Functional User Requirements of a piece of software
  - A business application
    - include humans and other peer applications with which the application interfaces
  - A real-time application
    - the functional users would normally be engineered hardware devices or other interfacing peer software

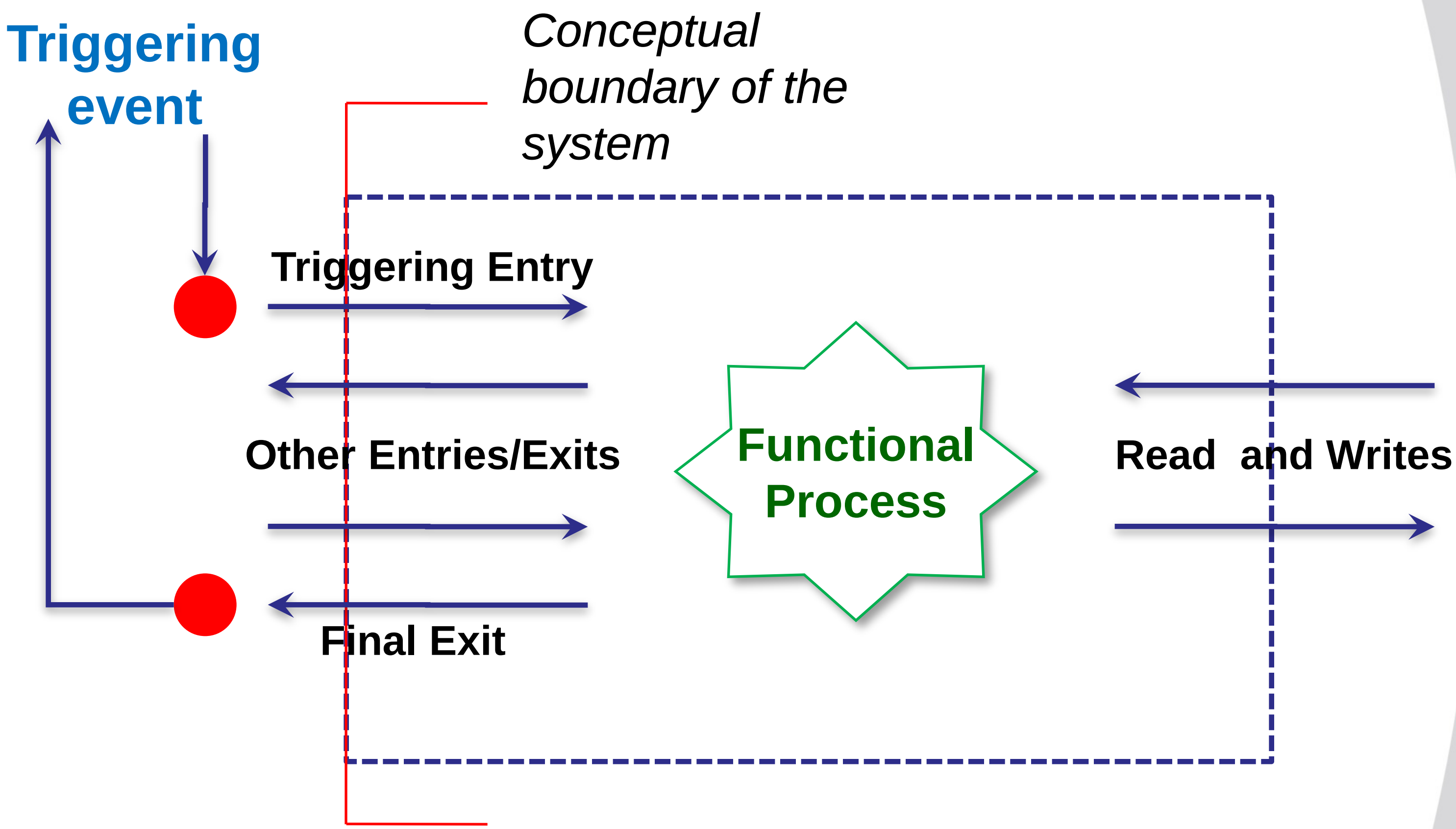
Identify Functional Processes

- **<u>Functional Process</u>:** is an elementary component of a set of FURs comprising a unique, cohesive and independently executable set of data movements
- It is **triggered** by a data movement (an Entry) from a functional user
- It is complete when it has executed all that is required to be done in response to the triggering event
- *Note: Similar to IFPUG FPA "Elementary process"*

Triggering Event

- **Triggering Event** : An event (something that happens) that causes a functional user of the piece of software to initiate ('trigger') one or more functional processes
  - occurs outside the boundary of the software being measured and initiates one or more functional process
    - In MIS application software, such an event is usually something in the real world about which the software is required to store data,
    - In real-time software, the event might be the arrival of a message, or the occurrence of a clock tick or of an interrupt
      - Clock and timing events can be triggering events!

Identifying Functional Processes




Data movement. A data movement moves attributes belonging to a single data group.

RULES – Functional process

- A functional process shall be derived from at least one identifiable Functional User Requirement within the agreed scope
- A functional process shall be performed when an identifiable triggering event occurs
- A specific event may trigger one or more functional processes that execute in parallel.
- A specific functional process may be triggered by more than one event
- Check that each functional process:
  - has at least one Entry (E)
  - has at least one Exit (X) or one Write (W)
    - At least → 1 E + 1 X/1 W
  - does not contain duplicate sub-processes

Examples (Business Applications)

- In a company,
  - an order is received (triggering event),
  - causing an employee (functional user) to enter the order data (triggering Entry conveying data about an object of interest 'order'), as the first data movement of the ‘order entry’ functional process
- An end-of-week clock tick (triggering event) causes the start of production of reports, and of the process to review expiry of time-limits in a workflow system.

Examples (real time)

- When the temperature reaches a certain value (triggering event), a sensor (functional user) is caused to send a signal (triggering Entry data movement) to switch on a warning light (functional process)
- An emergency condition detected in a nuclear power plant may trigger independent functional processes in different parts of the plant to lower the control rods, start emergency cooling, close valves, sound alarms to warn the operators, etc.
- Retraction of an aircraft's wheels may be triggered by the 'weight-off-ground' detector, or by pilot command

Example: Functional Process

Is “Withdrawal of cash from an ATM” a functional process or not?

YES

- Triggered by a unique event
- There is a data entry into application’s boundary (withdraw cash request, customer ID, cash amount)
- Operations performed inside (cash amount is accounted from customer account)
- Data exit to the user (receipt)
- When completed, it leaves the application in a consistent state

- What about “Entering the amount to be withdrawn from user interface”?
- Sending a triggering message to the money counting machine?

Identifying Objects of Interest and Data Groups

- COSMIC method is based on identifying and counting the data movements (performed in each functional process) which move a group of attributes (Data Group) of an object of interest

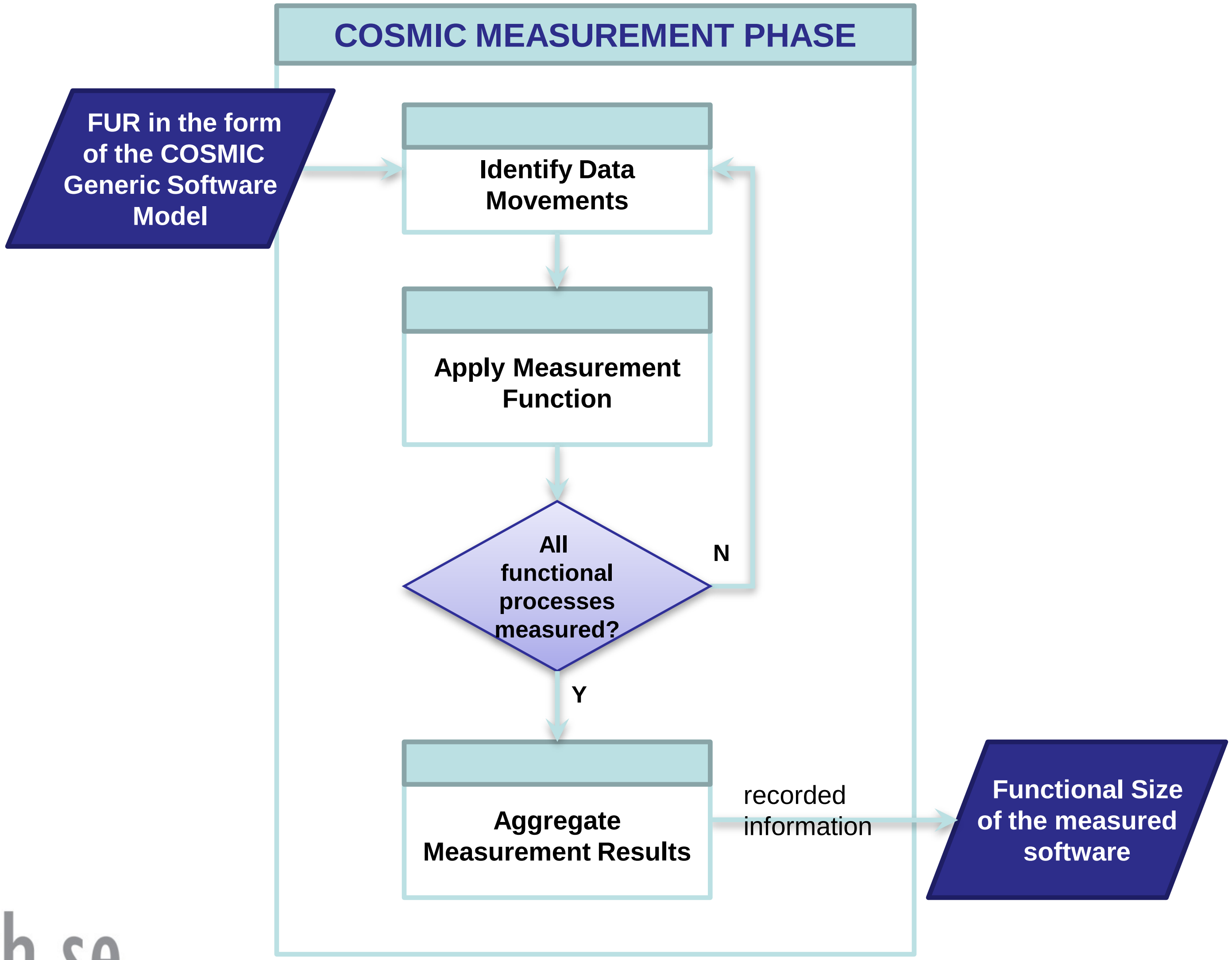
- Identified from the point of view of the Functional User Requirements.
  - Any physical thing, as well as
  - Any conceptual objects or
  - Parts of conceptual objects in the world of the user about which the software is required to process and/or store data
  - E.g. Customer, Student, Department, Course

Definition: Data Group

- A data group (type) is a:
  - **distinct, non empty, non ordered and non redundant** set of **data attributes** where each included data attribute describes a complementary aspect of the same **object of interest**
- PRINCIPLES
  - Each identified data group shall be unique and distinguishable through its unique collection of data attributes
  - Each data group shall be directly related to one object of interest in the software's FURs

Data Groups - Examples

Object of Interest	Data Attributes
MIS Systems	*Much persistent data; several attributes per object*
employee	name, address, date of birth
dependent	name, relationship
employment history	start date, department, job title
salary history	start date, basic salary
qualification	qualification, college, date
Mobile Telecom Systems	*Data is mostly transient - few attributes per object*
phone owner	PIN, mobile no.
screen	message, symbols
speaker	tone message
battery	voltage
send button	called number
‘phone book’	name, tel no.



COSMIC MEASUREMENT PHASE
FUR in the form of the COSMIC Generic Software Model
Identify Data Movements
Apply Measurement Function
All functional processes measured?
N
Y
Aggregate Measurement Results
recorded information
Functional Size of the measured software

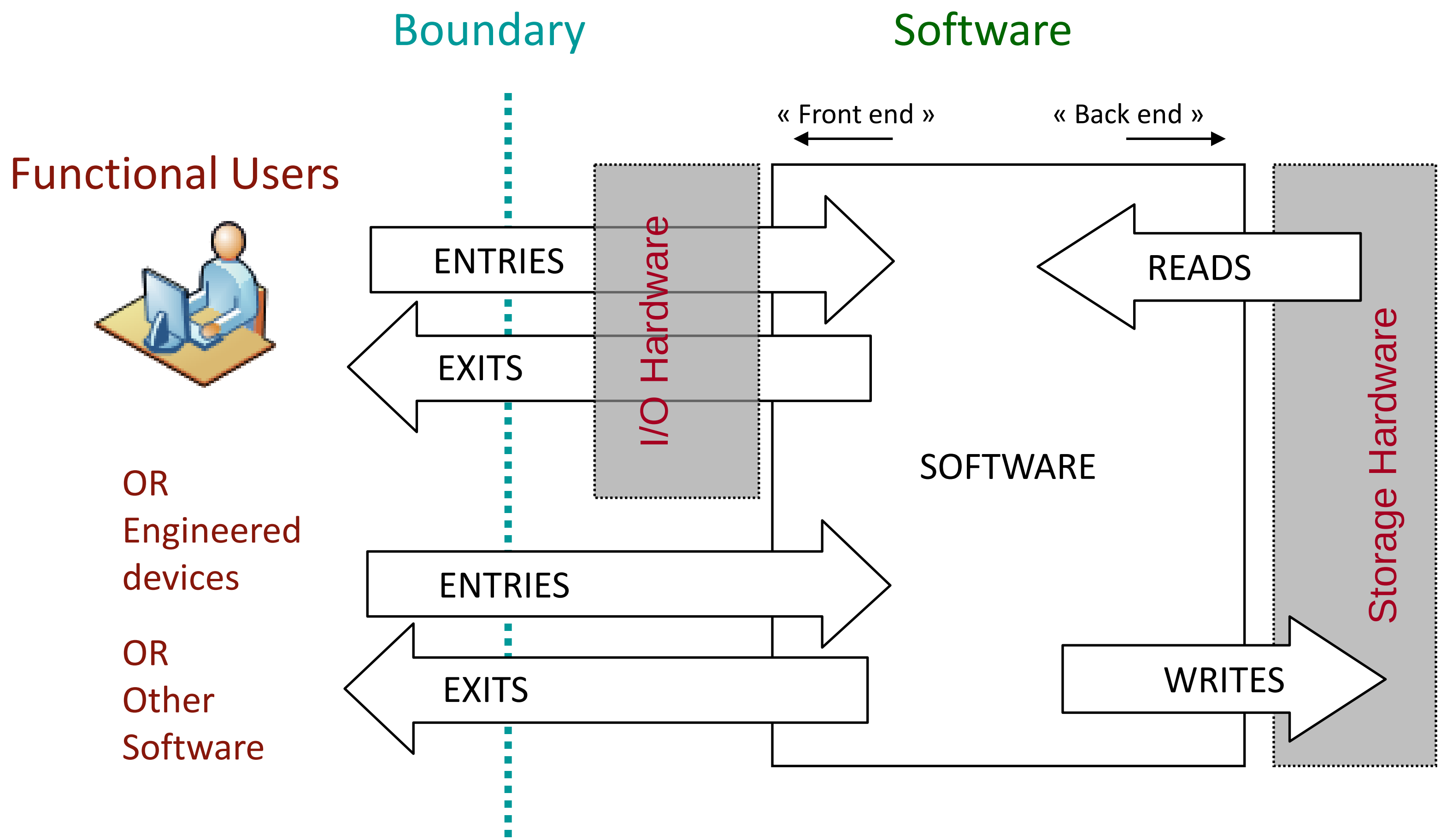
Identifying Data Movements

- A COSMIC data-movement: A base functional component which moves a single data group.

- There are four types of data movements:
  - Entry (E)
  - Exit (X)
  - Read (R)
  - Write (W)

Data Movement Types

- An **ENTRY (E)** : Moves a Data Group type from a user across the boundary into the functional process type where it is required
  - An ENTRY (E) does not update the data it moves.
- An **EXIT (X)**: moves a Data Group type from a functional process across the boundary to the user that requires it
  - An EXIT (X) does not Read (R) the data it moves
- A **READ (R)** : moves a data group type from persistent storage within reach of the functional process which requires it
- A **WRITE (W)** : moves a data group type lying inside a Functional Process to persistent storage


Boundary
Software
« Front end »
« Back end »
Functional Users
ENTRIES
EXITS
I/O Hardware
READS
SOFTWARE
Storage Hardware
OR
Engineered
devices
ENTRIES
OR
Other
Software
EXITS
WRITES

Calculate COSMIC Functional Size

- Size_{CFP} (functional process$_i$) =

  Σ size(Entries$_i$) + Σ size(Exits$_i$) + Σ size(Reads$_i$) + Σ size(Writes$_i$)

Method:	**COSMIC FFP**
Entity:	Functional User Requirements
Attribute:	Functional Size
Scale Type:	Ratio
Scale	Natural numbers
Unit of Measurement	1 COSMIC FP

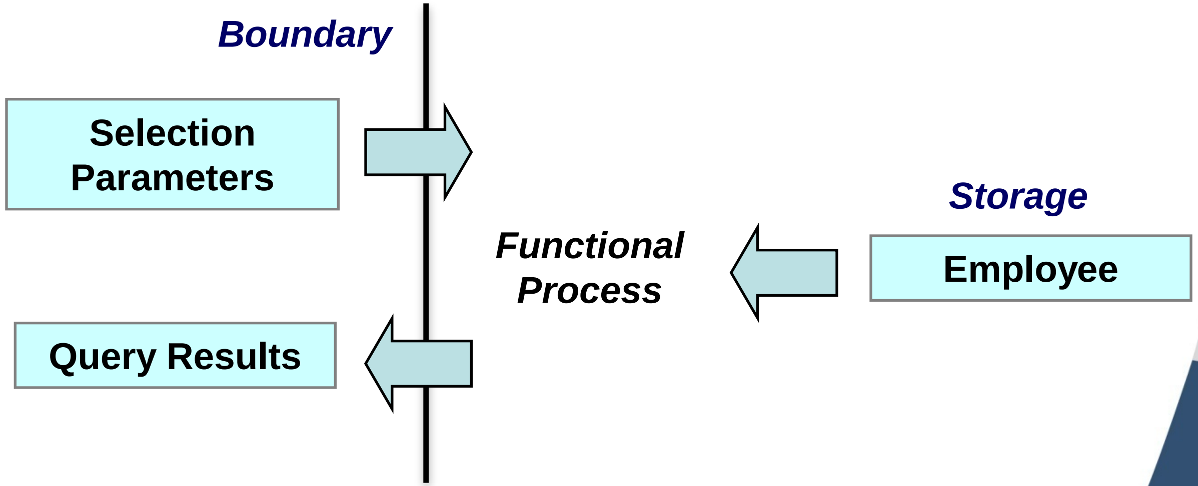
Object of Interests and Functional Processes

- **CRUDL - Create, Read, Update, Delete, List**
- It is important to identify persistent data in the applications because creation, reading, update, deletion and listing of these objects are generally separate *functional processes*

- **Persistent storage** is storage which enables a functional process to
  - store a data group beyond the life of the functional process
  - and/or from which a functional process can retrieve a data group stored by another functional process,
  - or stored by an earlier occurrence of the same functional process,
  - or stored by some other process

Example 1

- Given an ad hoc enquiry against a personnel database to extract a list of names of all employees aged over 35.
  - Does it consist of a unique and sequential data movement? Yes
  - Is it triggered by a unique event? Yes
  - Does the triggering event happen outside the boundary of the application? Yes
- The Entry moves a data group containing the selection parameters.
- The Exit moves a data group containing the single attribute ‘name’
- The ‘object of interest (or ‘thing’) is ‘all employees aged over 35’



Example 2

- Order-processing application maintaining customer details:
  - Customer (customer ID, name, address, phone, credit limit, customer_type_code, latest_order_date, etc.)
- ‘Customer’ is considered as an ‘object of interest’ for the users in the order-filling desk using the application
- For this case, the basic customer creation process includes:
  - 1 Entry (related with the customer object)
  - 1 Write (related with the customer object)
  - 1 Exit (error message)
- Total size: 3 CFP (COSMIC Function Point)

Example 3

- FUR: The user will be able to update an employee record, where the user knows the employee’s name but not the unique employee ID.

FP1: The user is invited to list all employees by name
E - Request ‘list employees’
R - Employee
X - Employee data (some attributes in list form to choose the desired employee)
X - Error Message
Size of FP1 = 4 CFP

FP2: The user chooses the particular employee from the list and displays the data that needs to be updated
E - Employee ID (= selecting the desired employee)
R - Employee data
X - Employee data (detailed form, all attributes)
X - Error Message
Size of FP2 = 4 CFP

FP3: The user updates employee info
E - Updated employee data
W - Employee data
X - Confirmation or error message
Size of FP3 = 3 CFP

Size of this FUR
= Size (FP1) + Size (FP2) + Size(FP3)
= 4Cfsu+4Cfsu+3Cfsu
= **11 CFP**