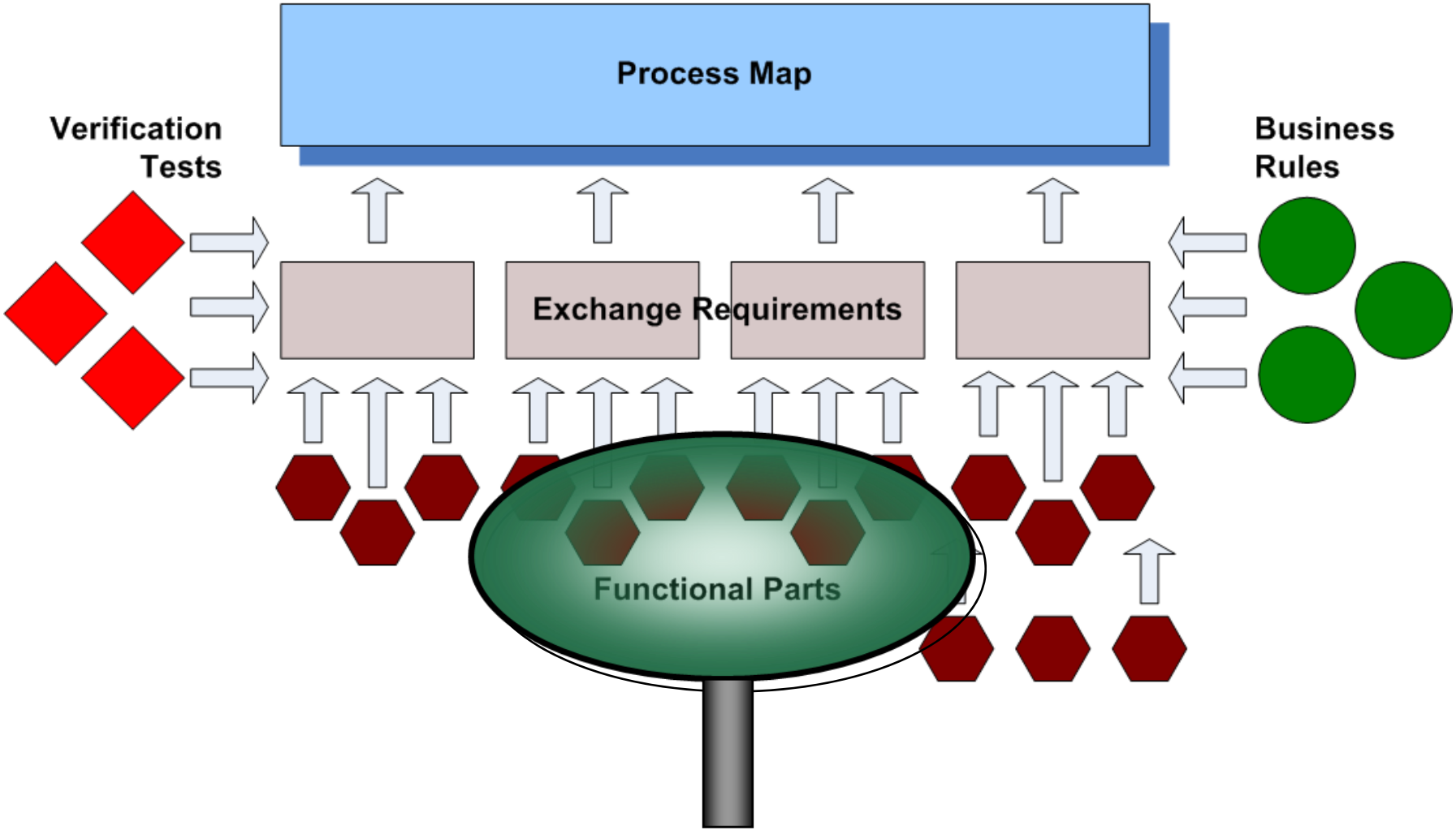
IDM
Functional Parts

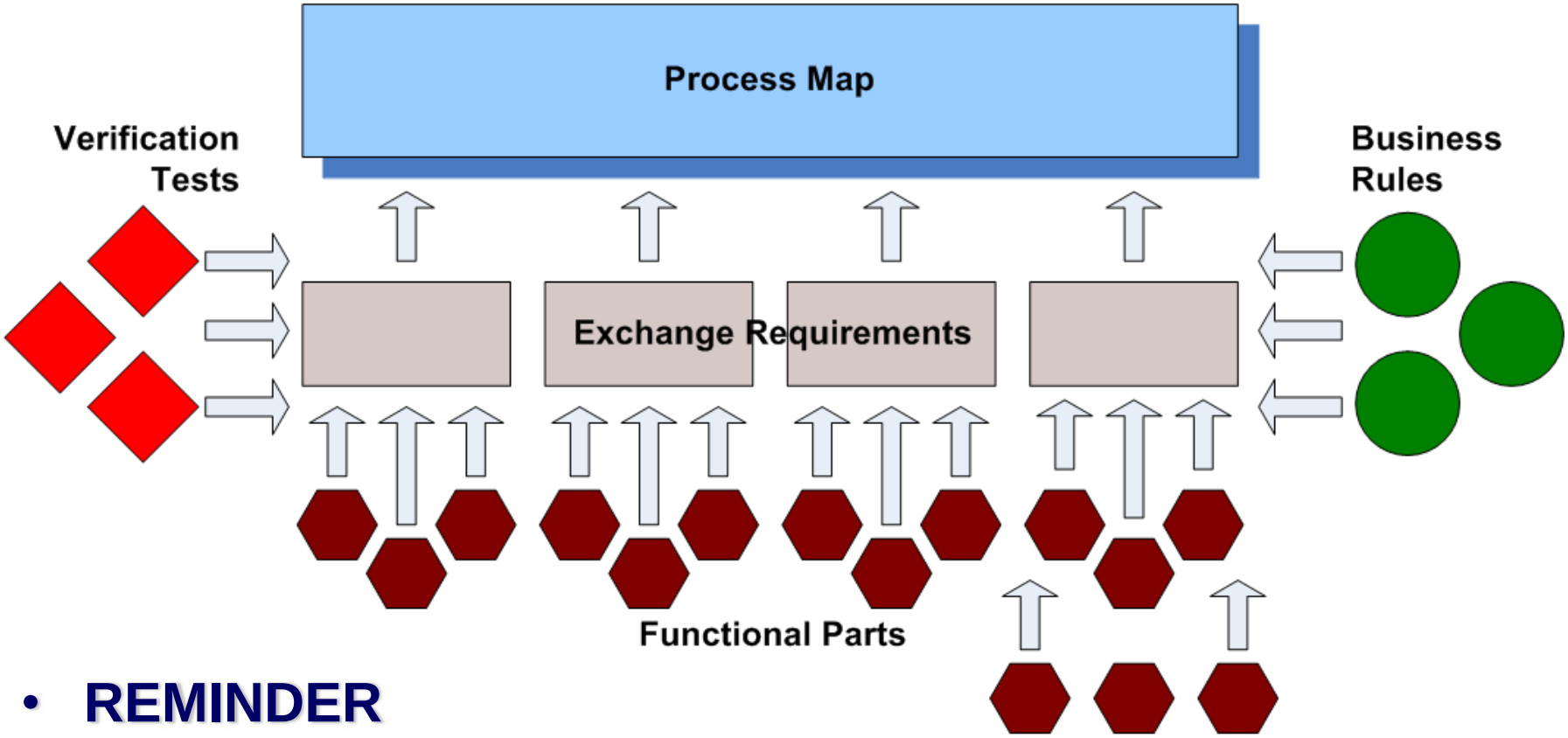
Jeffrey Wix

IDM Technical Architecture



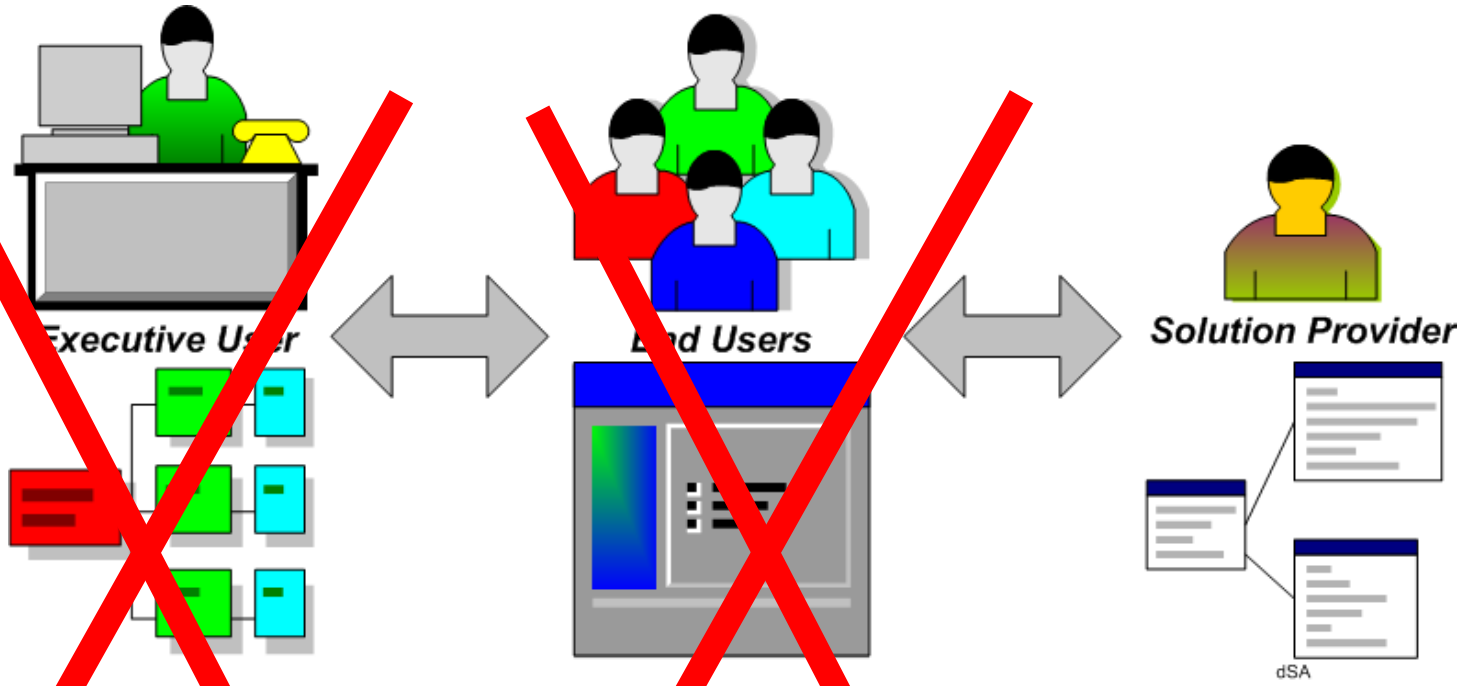
Focus on Functional Parts

IDM Technical Architecture



- **REMINDER**
- A unit of information, or a single information idea, used by solution providers to support an exchange requirement
- A functional part is a complete schema in its own right as well as being a subset of the full standard on which it is based

Users

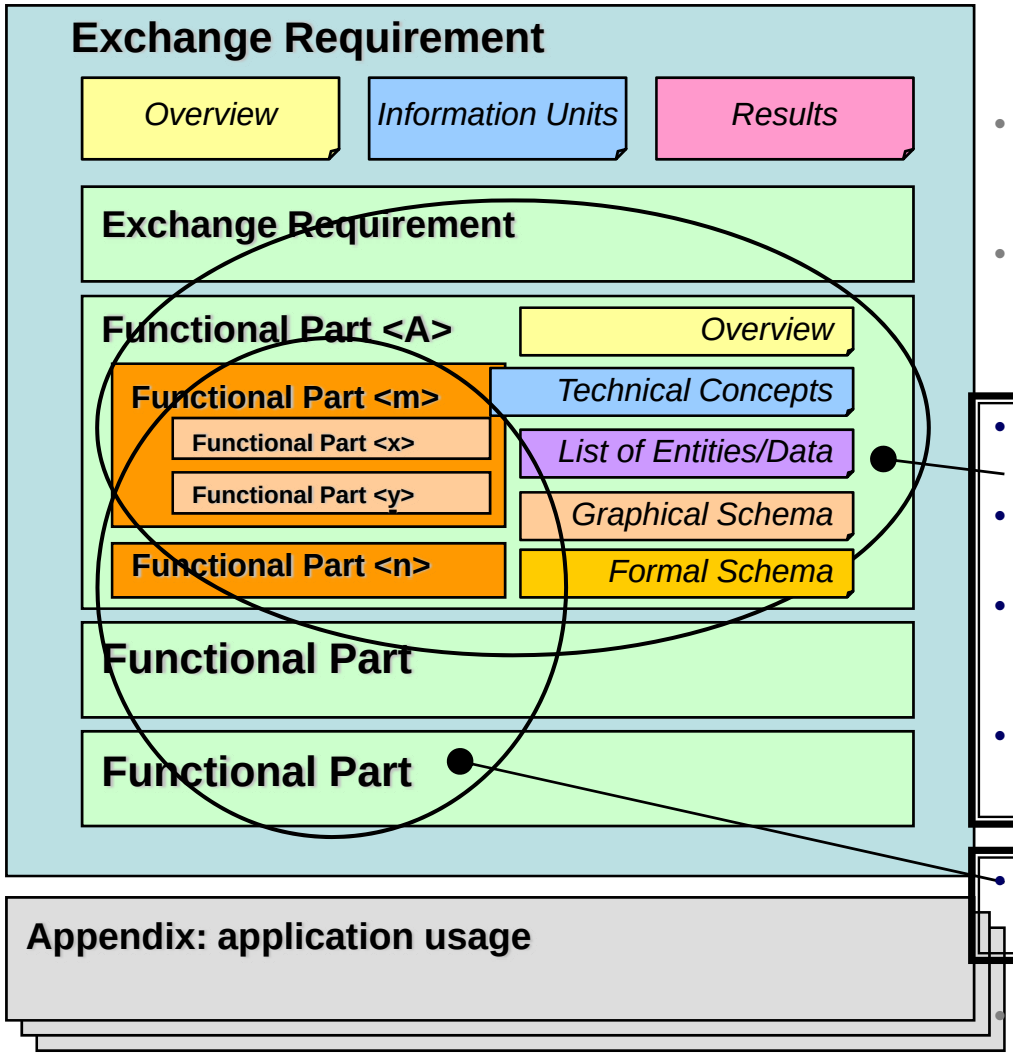


- aware of process and business impacts
- doesn't need technical detail about use of information in the process
- does not need to know about software or format

- needs to know what information to expect and how to use it in the business process
- does not need to know about software or format

- writes software + data exchange interface
- needs to know what users expect from software
- needs to know about the exchange format

IDM's Exchange Requirement definition



- Plain language description to make it easy to understand for different groups (managers, project participants, software developers
- Building professional terminology – No deep technical detail or references to data format.
- Identifies the information needing to be exchanged at a point in the business process

- One or several "groups" of information required to perform a certain process.
- Possibility to define separate exchange requirements for each "group".
- This applies to the both to the Information delivered to the process and from the process (the result).
- Identifies the functional parts to be used in a Exchange Requirement

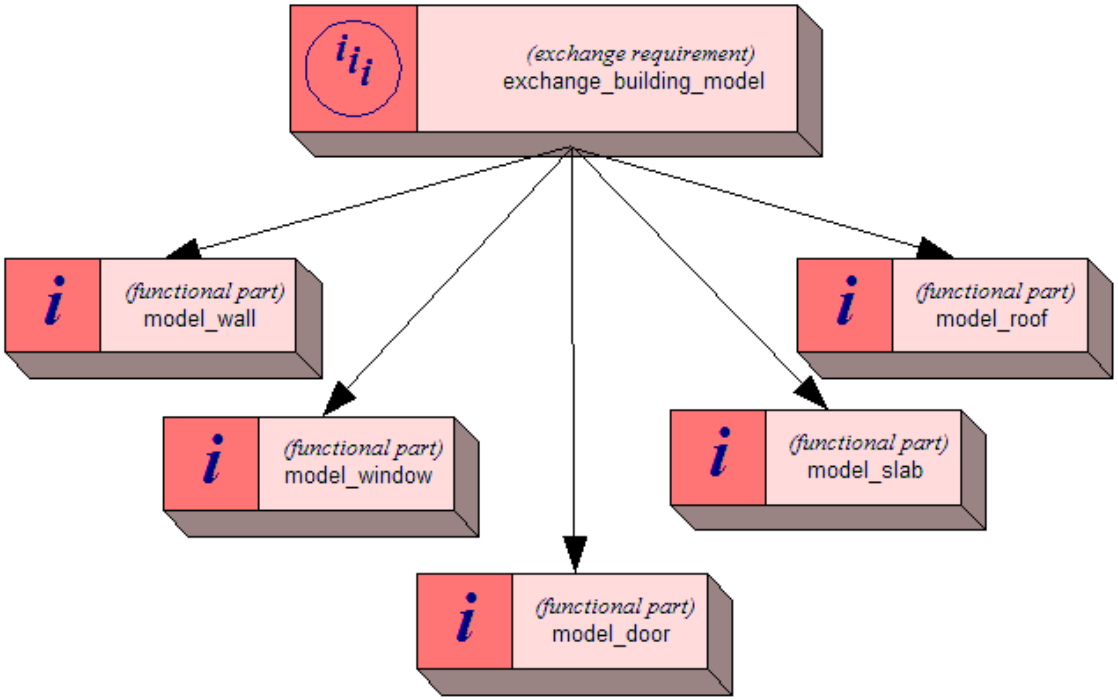
- Recursive, enabling re-use and re-combination at many levels

- Appendix contains information on how applications should be used to ensure correct IFC export and/or import

WHAT

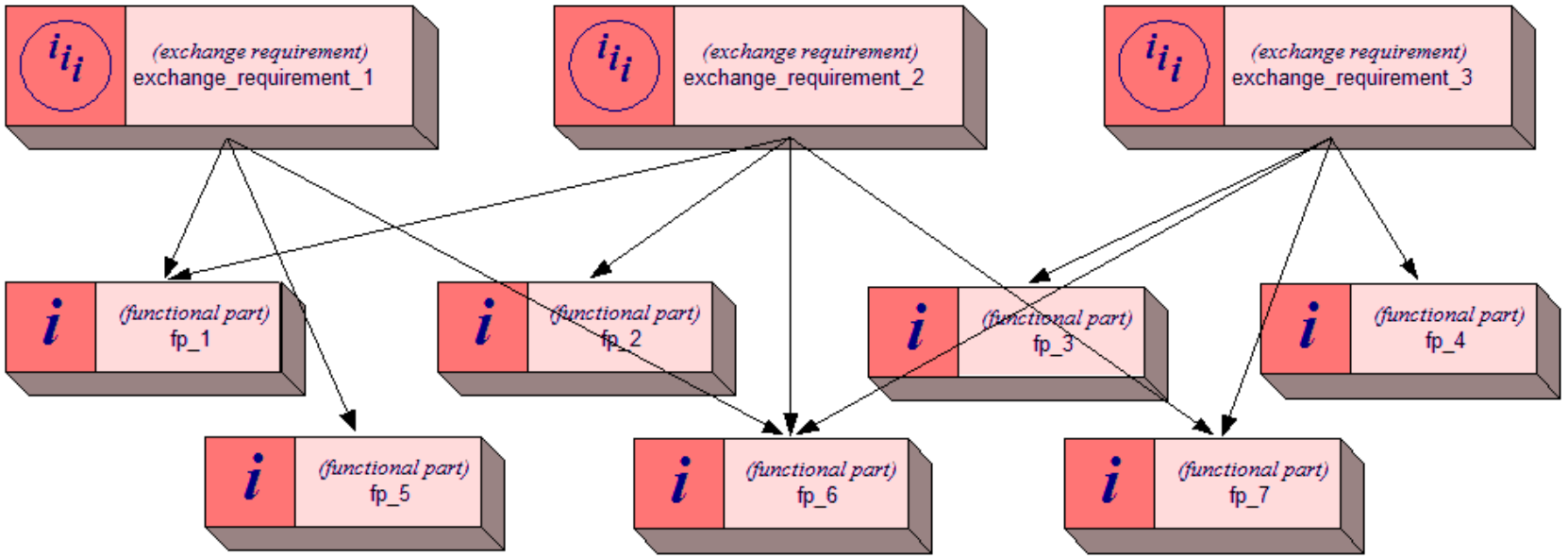
- A unit of information, or a single information idea, used by solution providers to support an exchange requirement.
- A functional part describes the information in terms of the required capabilities of the industry standard information model upon which it is based.
  - For IDM as presently established, the functional parts are based on versions of the IFC model.
- A functional part is fully described as an information model in its own right as well as being a subset of the information model on which it is based.
  - i.e. a functional part is a complete information exchange specification

Functional Part - Basics



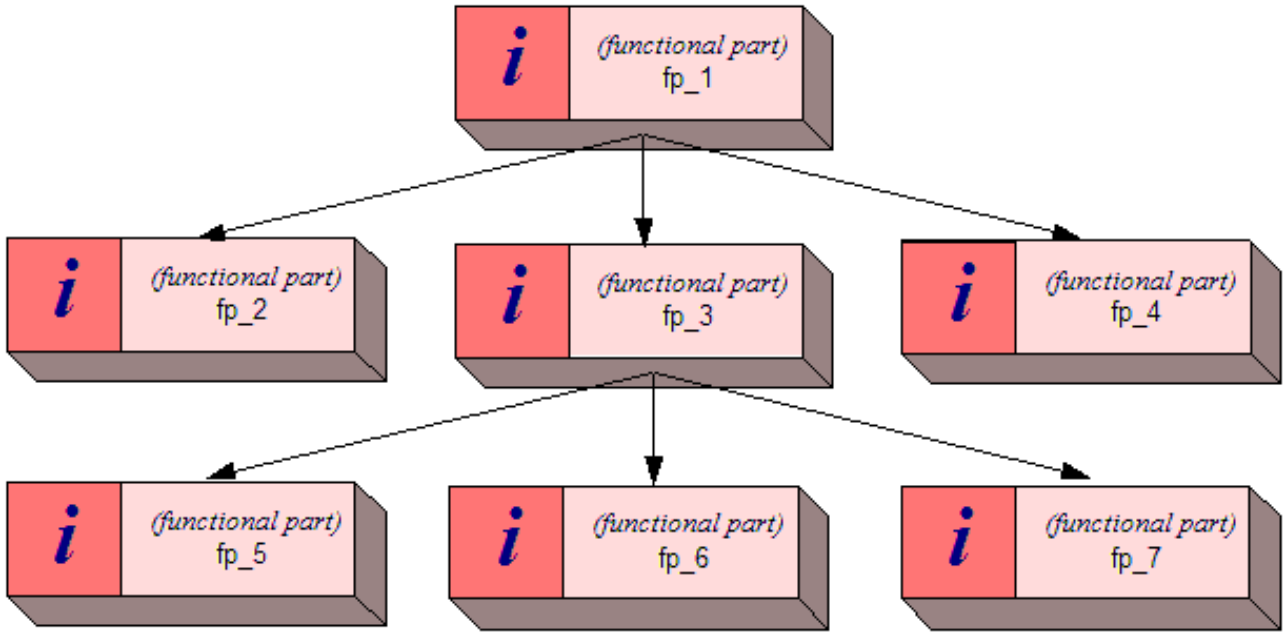
- A functional part describes the actions that are carried out within a business process to provide the resulting output information.

Functional Part - Reuse



- A functional part may be used by many exchange requirements
  - geometric shape representations
  - relationships
  - general concepts (cost, materials etc.)

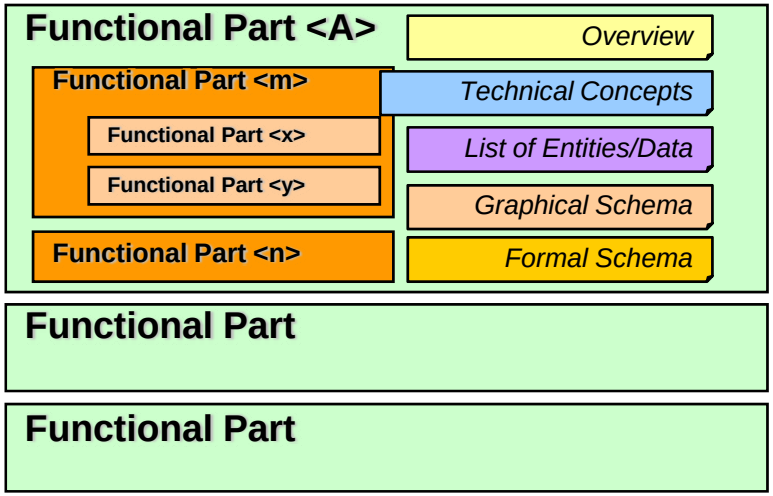
Functional Part - Decomposition



- Functional parts can be broken down into other functional parts

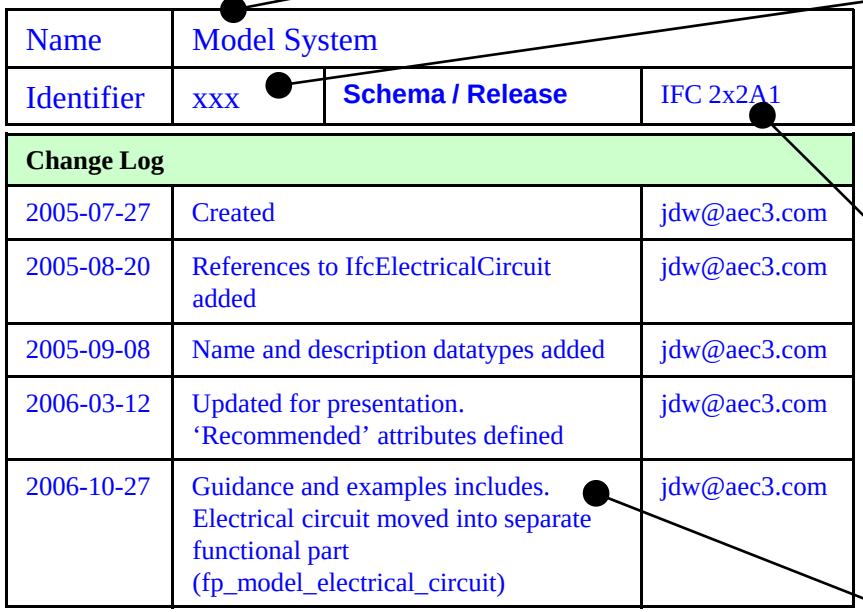
Functional Part Development

- A functional part contains the following sections:
  - header section
    - administrative information
  - overview section
    - short description of purpose
  - results section
    - outcome of the functional part
  - technical concepts section
    - detailed technical description using the selected standard
  - lists section
    - entities, datatypes, functions and property sets used



  - schema section
    - presentation of the schema
  - examples section
    - examples of how the functional part is used
  - software guidance
    - how software applications support the functional part

Header Section



Name	Model System		
Identifier	xxx	**Schema / Release**	IFC 2x2A1

Change Log		
2005-07-27	Created	jdw@aec3.com
2005-08-20	References to IfcElectricalCircuit added	jdw@aec3.com
2005-09-08	Name and description datatypes added	jdw@aec3.com
2006-03-12	Updated for presentation. 'Recommended' attributes defined	jdw@aec3.com
2006-10-27	Guidance and examples includes. Electrical circuit moved into separate functional part (fp_model_electrical_circuit)	jdw@aec3.com

- name or title of the functional part
- conform to the IDM naming rules
- unique identifier (not used)
- will provide index of all published functional parts
- schema name and release
- a functional part is applicable for a particular release of a schema
- creation of and changes made to the functional part.
- include date, person identifier and a description of the changes made.

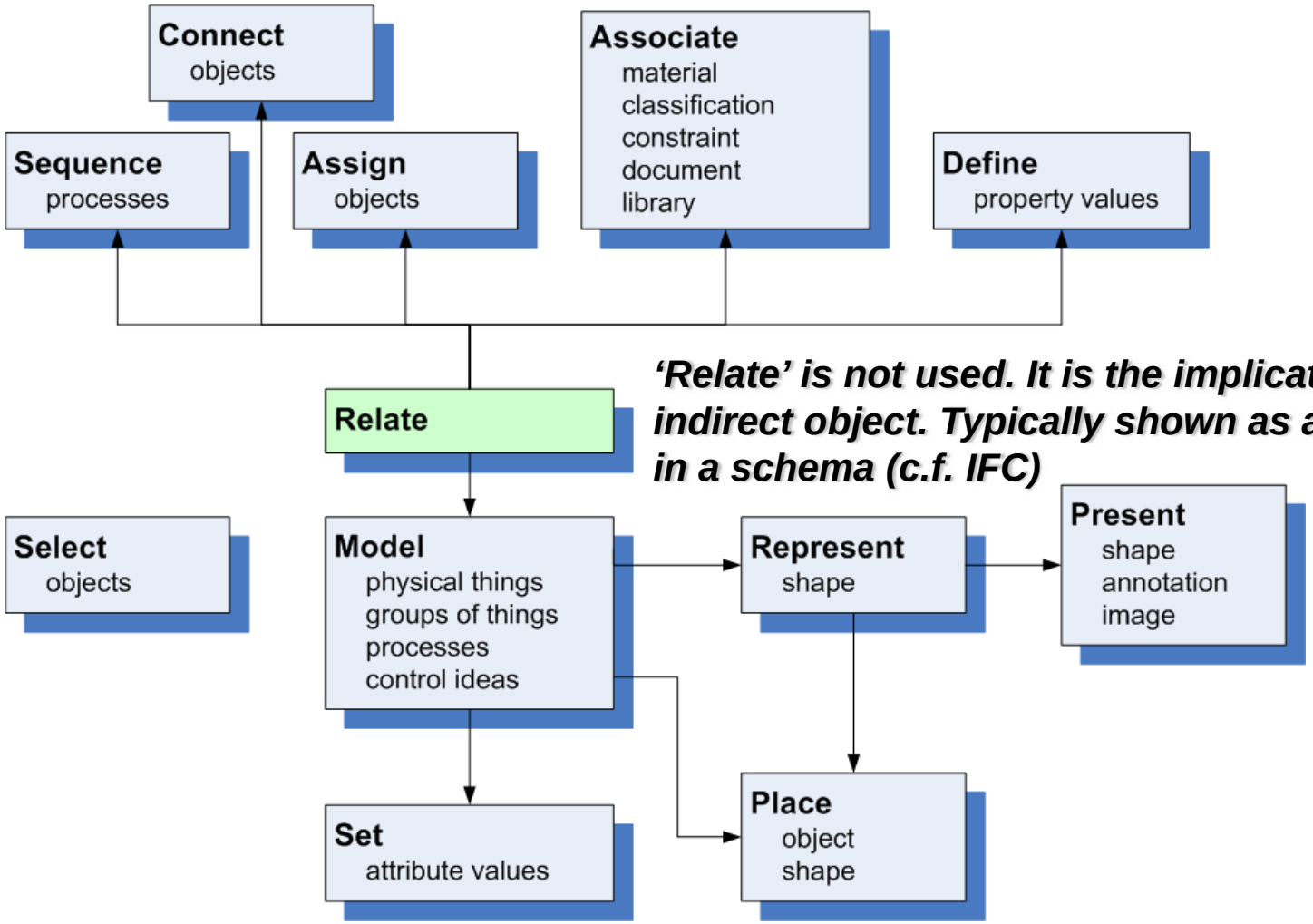
Naming Rules

- Why do we have naming rules?
  - Consistency in naming
  - Grammatical rules
  - Progressive development of a ‘scripting’ language
- A functional part name has three parts
  - the prefix ‘fp’ identifies that this component is a functional part
  - an action (or activity) required expressed as a verb.
  - an object that receives the action expressed as a noun or noun phrase).
    - may be a direct object as in ‘model wall’
    - may be an implied indirect object as in ‘associate material’
      - which means associate {to wall} {the} material

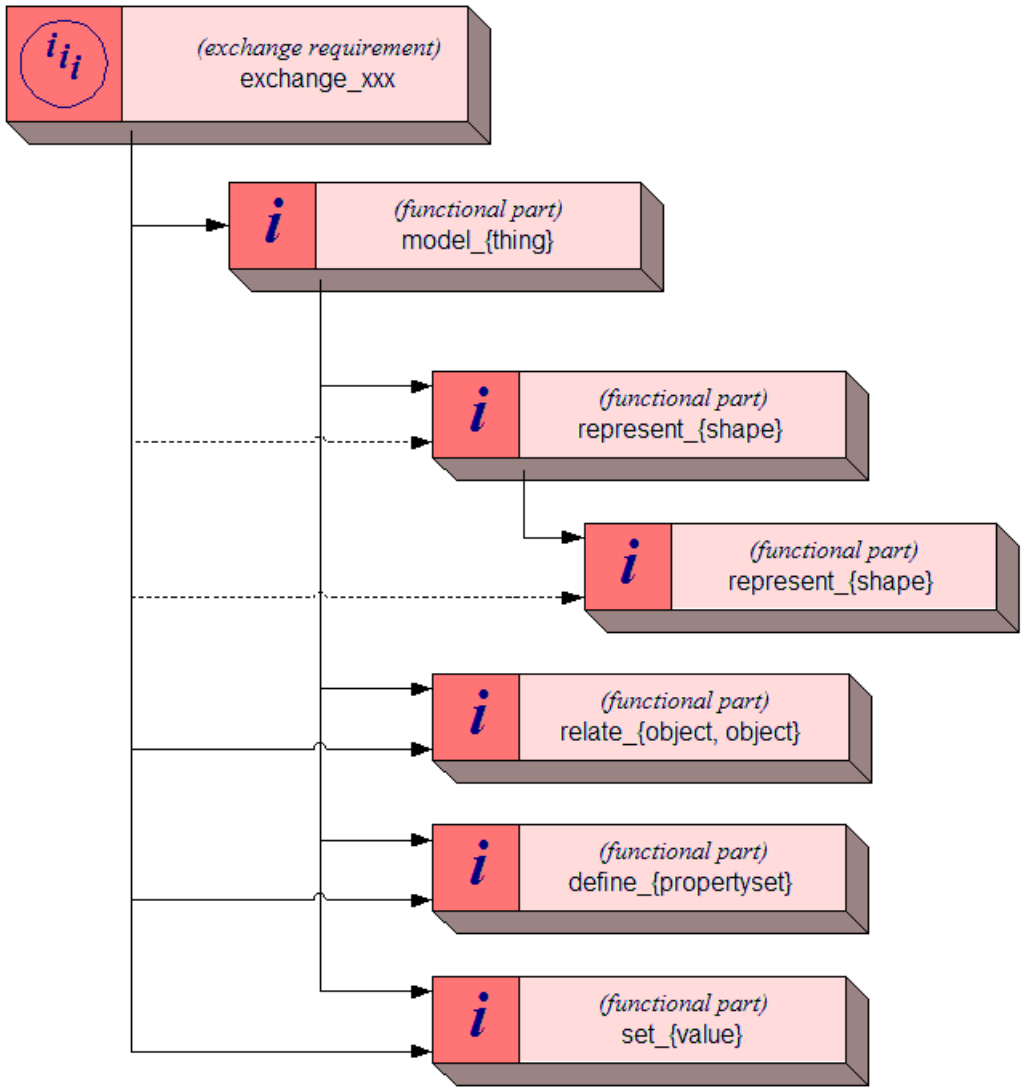
Allowed actions

- Naming rule allow various actions to be applied.
- These are from an allowed range that includes:
  - Assign: an object as an attribute of another
  - Associate: an externally defined idea
  - Connect: how objects come together
  - Define: provision of a set of properties
  - Model: key ideas that are expressed
  - Present: an image, including annotation elements
  - Represent: a form of geometric representation
  - Select: alternative selections that may be available
  - Sequence: the time relationship between processes
  - Set: values that may be set for a purpose

Naming Rules Diagram



Naming Tree


(exchange requirement)
exchange_xxx
(functional part)
model_{thing}
(functional part)
represent_{shape}
(functional part)
represent_{shape}
(functional part)
relate_{object, object}
(functional part)
define_{propertyset}
(functional part)
set_{value}

Overview Section

Provides the information concerning systems where a system is a grouping of elements. In the context of a distribution system, all of the elements that are grouped to form the system should be subtypes of IfcDistributionElement.

As well as enabling the description of complete systems, this functional part enables the grouping of elements into subsystems.

For an electrical system, the subtype IfcElectricalCircuit is used. This provides a simple approach to validate that only electrical items are connected to the electrical system.

The information provided about a system includes:
- Specification of a name and description for the system if required
- The elements that are grouped together to form the system.

Note that the shape representation of a system is derived from the shape representation of the elements that are grouped together within it. The system itself has no 'own' shape representation.

Further information about the system may be provided through locally defined property sets. There are no predefined property sets for systems.

- states aims and content in (fairly) non-technical form
- whilst a functional part is intended for solution providers, a 'user' should still be aware of the content
- first part is an 'excerpt'
  - abbreviated description of the functional part for web sites, Wiki site
- remainder extends the discussion and makes clear the intended content and purpose
- Mau use terminology that is from the standard, schema etc.

Results

- a simple statement of the outcome of a functional part

Specification of the system and the elements that make up the system

Technical Concept

Description of information required and specification of values to be set

Name of entity/attribute or property set/property to be used

Mandatory (actual/ recommended) or Optional to use

Description	Entity/Pset/Functional Part	MAN	REC	OPT
Specify the system occurrence in which elements will participate *A system opening is directly specified as an occurrence.*	IfcSystem OR IfcElectricalCircuit	✓		
Set the global unique identifier	IfcSystem.GlobalId → IfcGloballyUniqueId OR IfcElectricalCircuit.GlobalId → IfcGloballyUniqueId	✓		
Assert the owner history of the system	IfcSystem.OwnerHistory → fp_apply_owner_history OR IfcElectricalCircuit.OwnerHistory → fp_apply_owner_history	✓		
Specify the name of the system. *Although this is an optional attribute within IFC, it must be asserted for system.*	IfcSystem.Name → IfcLabel OR IfcElectricalCircuit.Name → IfcLabel		✓	
Specify a description for the system *Whilst the description does not add value to the semantics of the system, it can provide significant information for later project stages.*	IfcSystem.Description → IfcText OR IfcElectricalCircuit.Description → IfcText			✓

Note use of IFC schema terminology

Convention for attributes/properties

- The convention defined within the IDM for the expression of object/attribute/datatype and property set/property/datatype is as follows:
  - Object.Attribute → Datatype
  - OR
  - PropertySet.Property → Datatype

- For a property set, where a defined datatype is used, the form is:
  - PropertySet.Property → Defined type :: Datatype

Lists

- The list section shows the IFC components used
- Provides a quick ‘view’ of the schema
  - Entities
    - The subjects (or classes of item) of current interest
  - Datatypes (defined, enumeration and select)
    - Named types of data that may be used within the functional part such as labels, text descriptions, identifiers OR;
    - Enumerated values from which a selection should be made OR;
    - Alternative selections of route through the schema
  - Function
    - Processing capabilities used to validate data
  - Property sets
    - Those property sets that are relevant to the current functional part
  - Functional parts
    - Other functional parts whose services are used

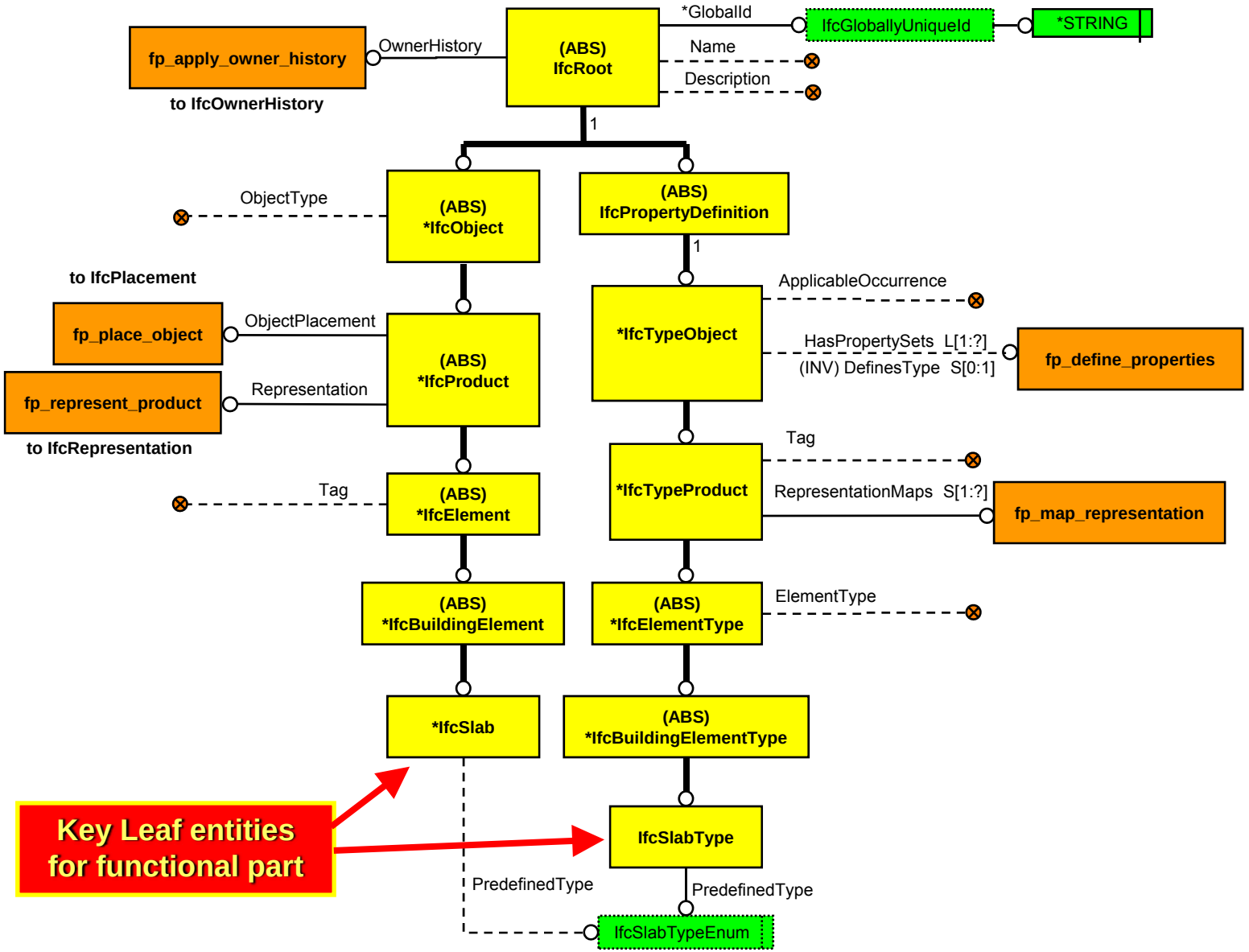
Lists Example

- *IFC Entities Required*
  - IfcElectricalCircuit
  - IfcElement
  - IfcGroup
  - IfcObject
  - IfcProduct
  - IfcRoot
  - IfcSystem
- *IFC Datatypes Required*
  - IfcGloballyUniqueId
  - IfcLabel
  - IfcText
- *IFC Functions Required*
  - -
- *IFC Property Sets Required*
  - Pset_SystemCommon
- *IDM Functional Parts Required*
  - fp_assigns_to_group
  - fp_apply_owner_history
  - fp_define_by_properties
  - fp_place_object
  - fp_represent_product
  - fp_services_building

Schema Section

- Presents the functional part as a schema
- There may be more than one schema presented
- With IFC, two schemas are routinely presented and three are easily possible
  - EXPRESS-G
    - The name of the graphical notation often used
  - EXPRESS
    - The name of the data definition language which is the ‘master’ form of the IFC standard.
    - EXPRESS is standardized as ISO 10303 part 11:2004
  - ifcXML
    - An XML form of the schema as an alternative to EXPRESS
    - Automatically converted from EXPRESS using a softwrae tool compliant with the rules of ISO 10303 part 28 (XML binding)

Schema (Graphical)



Schema (Data Definition Language)

```
  ABSTRACT SUPERTYPE
  SUBTYPE OF(IfcTypeProduct);
    ElementType : OPTIONAL IfcLabel;
END_ENTITY;
ENTITY IfcBuildingElementType
  ABSTRACT SUPERTYPE
  SUBTYPE OF(IfcElementType);
END_ENTITY;
ENTITY IfcSlabType
  SUBTYPE OF(IfcBuildingElementType);
    PredefinedType : IfcSlabTypeEnum;
END_ENTITY;
ENTITY fp_map_representation;
END_ENTITY;
ENTITY fp_apply_owner_history;
END_ENTITY;
ENTITY IfcElement
  ABSTRACT SUPERTYPE
  SUBTYPE OF(IfcProduct);
    Tag : OPTIONAL IfcIdentifier;
END_ENTITY;
ENTITY IfcBuildingElement
  ABSTRACT SUPERTYPE
  SUBTYPE OF(IfcElement);
END_ENTITY;
ENTITY IfcSlab
  SUBTYPE OF(IfcBuildingElement);
    PredefinedType : OPTIONAL IfcSlabTypeEnum;
  WHERE
    WR2 : (PredefinedType <> IfcSlabTypeEnum.USERDEFINED) OR
          ((PredefinedType = IfcSlabTypeEnum.USERDEFINED) AND EXISTS(SELF\IfcObject.ObjectType));
END_ENTITY;
```

Included functional parts
(schema imported)

Examples Section

- provide guidance to implementers
- may be used by them for preliminary testing to ensure that they are returning the correct results from their solutions.
- examples may include:
  - a description of the example scenario
  - samples of IFC files according to the schema of the functional part to show how the results of using the functional part might appear in practice
  - sample instance diagrams providing an easier visual reference

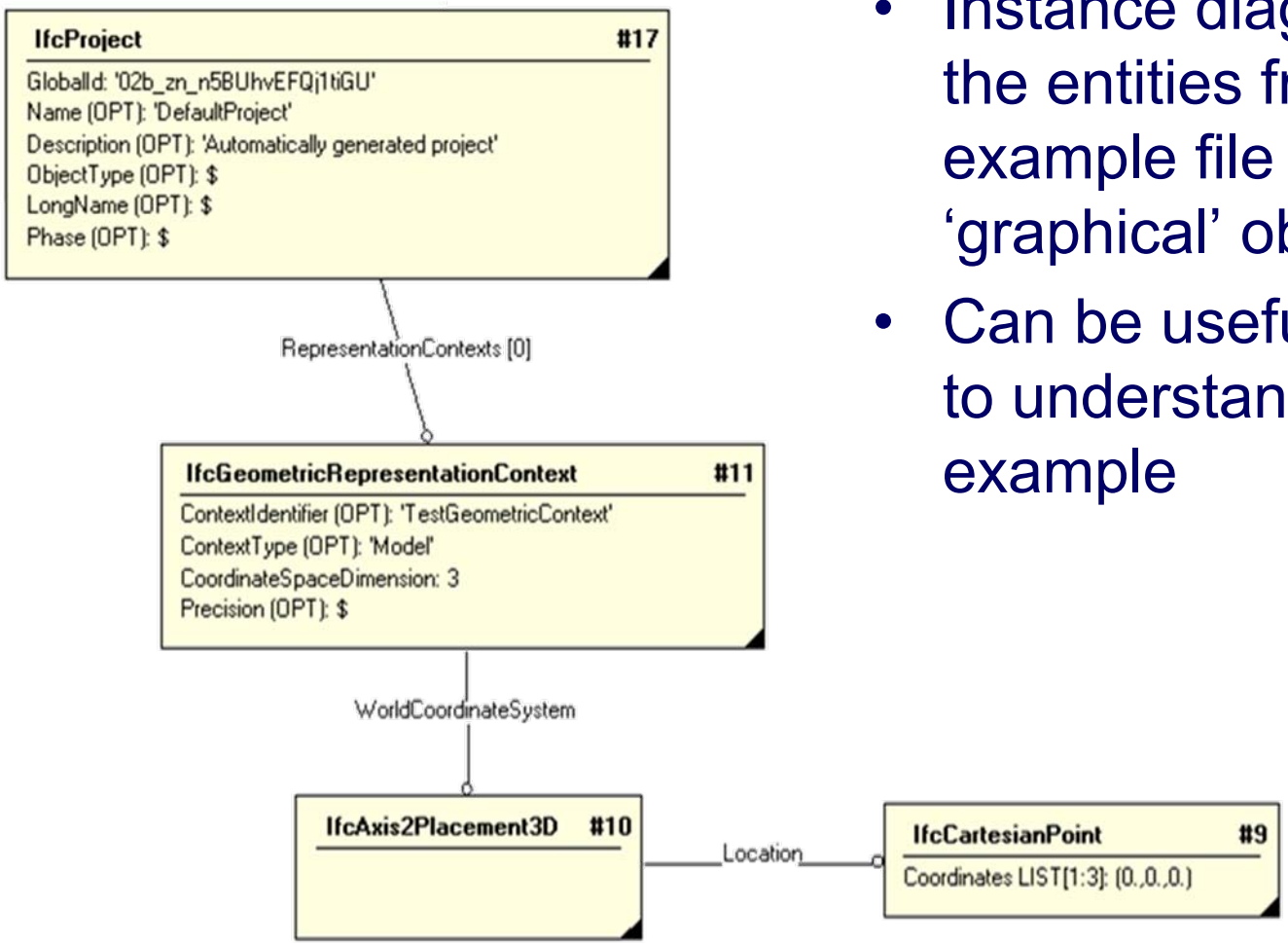
Example 1: Definition of a single geometric representation context, being the basic model context. It only contains the absolute mandatory information.

Note: It is an IFC2x2 example which still has the WorldCoordinateSystem given, even if it defaults to Location=[0.,0.,0.] an z-axis=[0.,0.,10.] and x-axis =[1.,0.,0.].

```
/* Definition of the project */
#17=IFCPROJECT('02b_zn_n5BUhvEFQj1tiGU',#16,'DefaultProject','Automatically generated
project',$,$,$,(#11),#6);

/* Definition of the geometric representation context */
#11=IFCGEOMETRICREPRESENTATIONCONTEXT('TestGeometricContext','Model',3,$,#10,$);
#10=IFCAXIS2PLACEMENT3D(#9,$,$);
#9=IFCCARTESIANPOINT((0.,0.,0.));
```

Examples – Instance Diagram



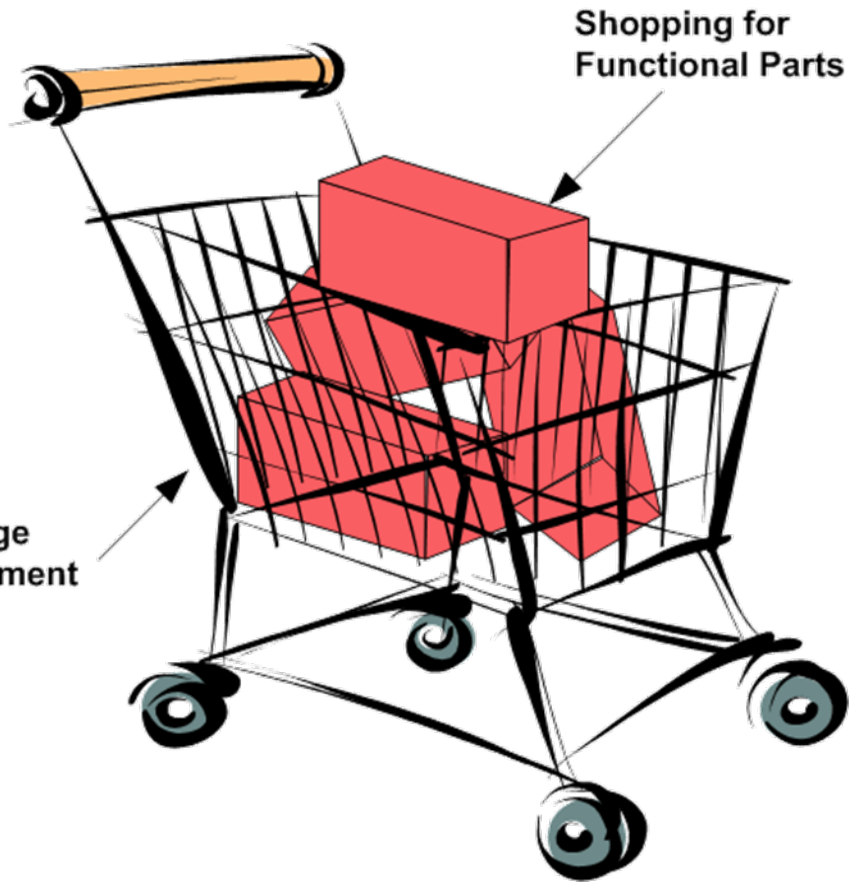
- Instance diagrams show the entities from the example file as ‘graphical’ objects
- Can be useful in adding to understanding of the example

Software used is ‘GraphicalInstance’ by Eurostep

Compiling Exchange Requirements

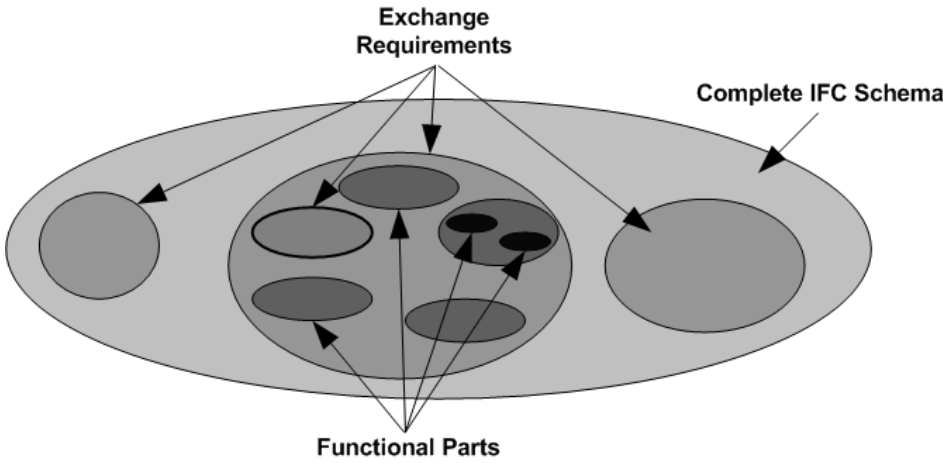
- A fundamental premise of IDM is that an exchange requirement can be generated by compiling a set of functional parts.
- Simply, it can be seen as ‘shopping’ for a basket of functional parts.
- More theoretically, it is the sum or integral of the functional parts

$$ER = \sum FP$$



Overview of Parts

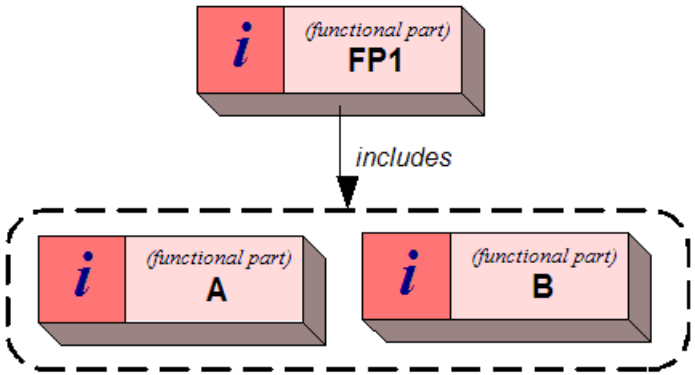
- This illustration shows the relationship between the IFC schema, exchange requirements and functional parts.
- Specifically, it shows that an exchange requirement is strictly specified by its constituent functional parts



Adding Functional Parts

- Functional part FP1 includes functional parts A and B.
- It also includes fp1 which is fully defined within the definition of FP1
- The schema for FP1 is obtained by adding together the schema of all the included functional parts.

  FP1 = fp1 + A + B
- That is, add all of the entities in each schema into the total schema.
- By doing this, the same entity may be included in the total more than once.
- This is not allowed.
  - Each entity may only appear once in the total schema.



Adding Entities in Functional Parts

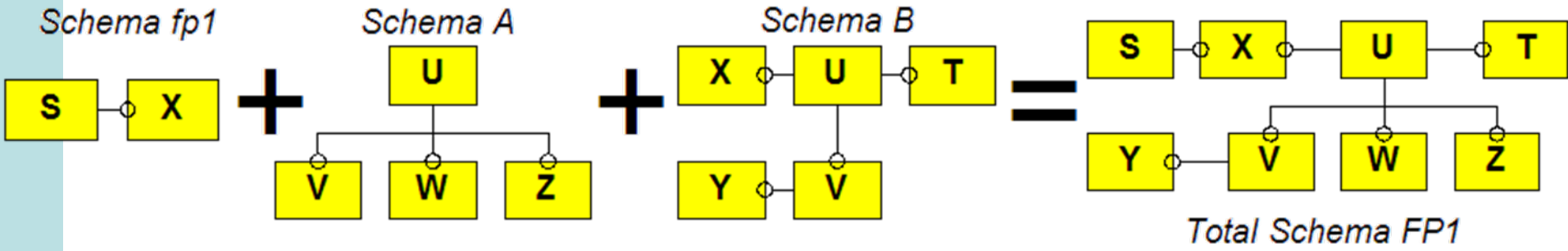
- Therefore, the process of adding schemas together also has to resolve the issue of overlapping entities such that each required entity appears only once. That is:

  if fp1 shows the set of entities S and X (shown as fp1 = SET {S, X})

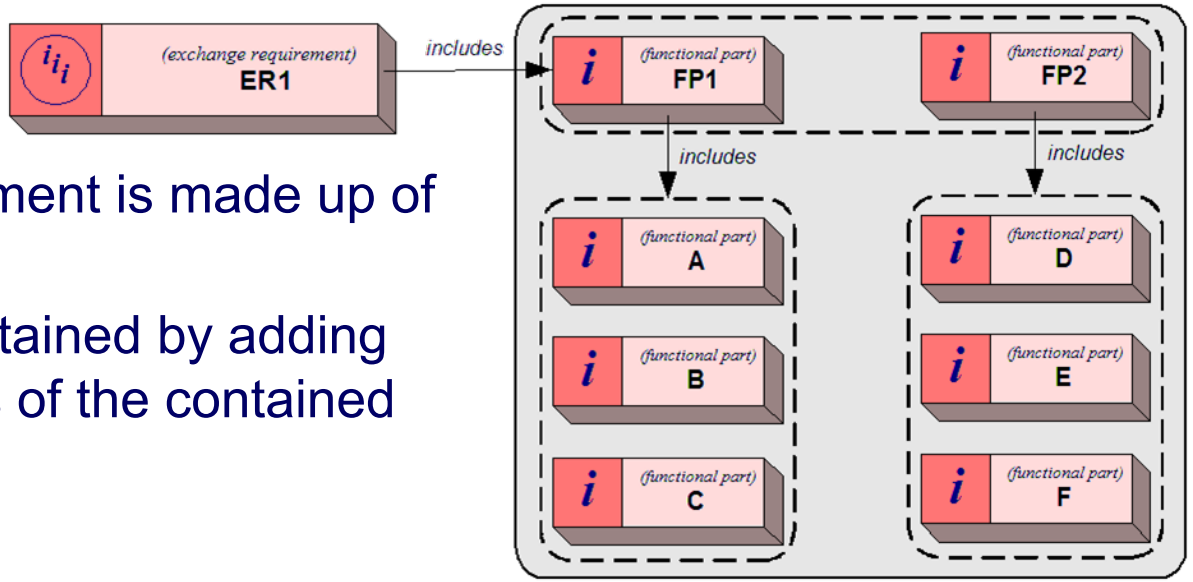
  and A = SET {U, V, W, Z}

  and B = SET {T, U, V, X, Y}

  then FP1 = SET {S, T, U, V, W, X, Y, Z}



Delivering an Exchange Requirement

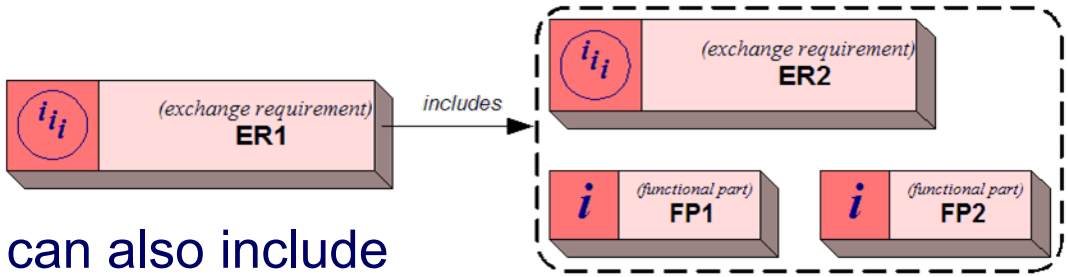


- An exchange requirement is made up of functional parts, so …
- it's schema is also obtained by adding together the schemas of the contained functional parts.
- ER1 = FP1 + FP2
- If FP1 = fp1 + A + B + C

  and
- FP2 = fp2 + D + E + F

  then it follows that:
- ER1 = fp1 + A + B + C + fp2 + D + E + F

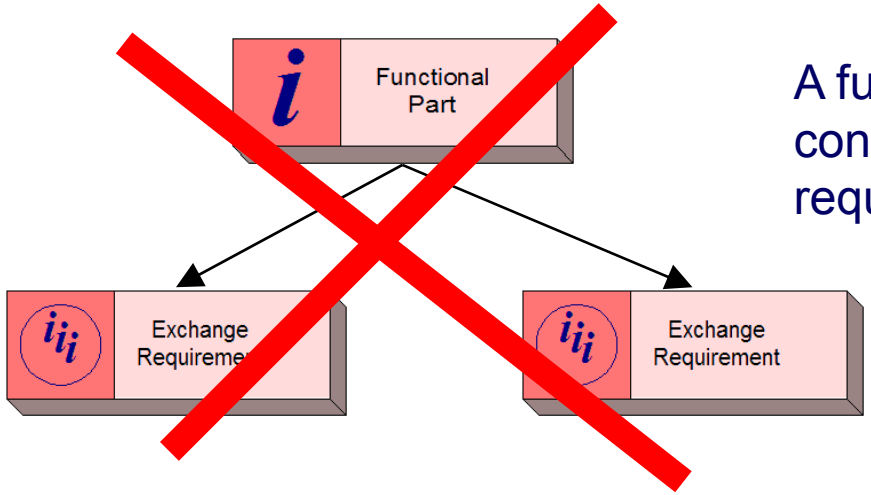
Including Other Exchange Requirements



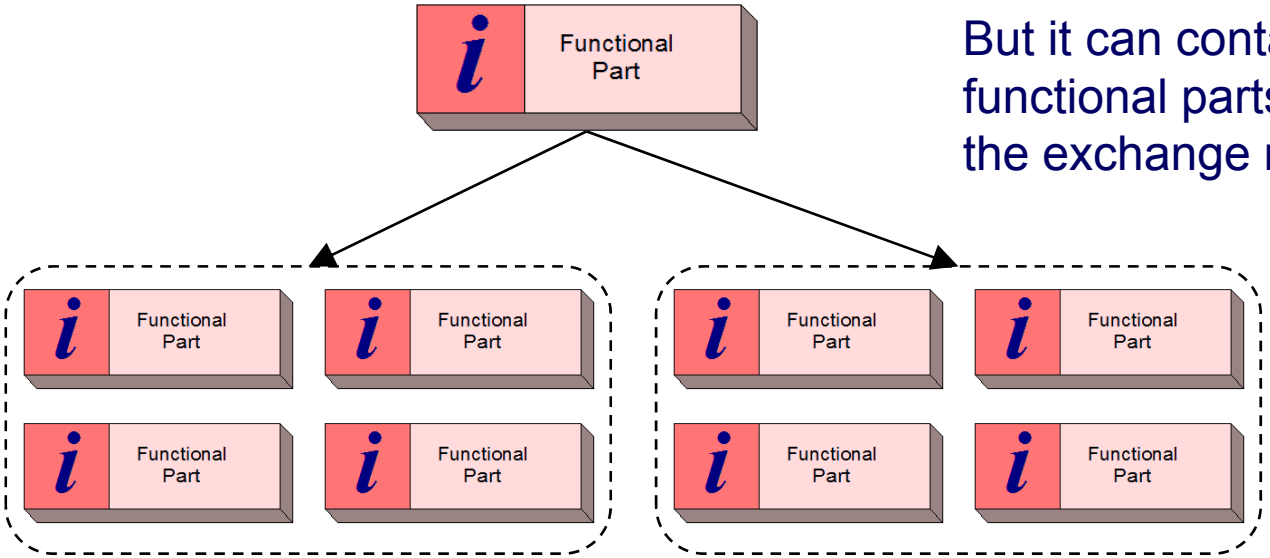
- An exchange requirement can also include other exchange requirements.
- This facility can be used effectively to reduce the effort needed in describing the information requirements. That is:

  ER2 = ER1 + FP1 + FP2
- This is no different to the addition of functional parts as …
- an exchange requirement ultimately reduces to the functional parts from which it is built.
- Therefore, a reference to an included exchange requirement is only a reference to a set of functional parts that are used to build it.

Resolving Exchange Requirements



A functional part cannot contain exchange requirements



But it can contain the functional parts that define the exchange requirement