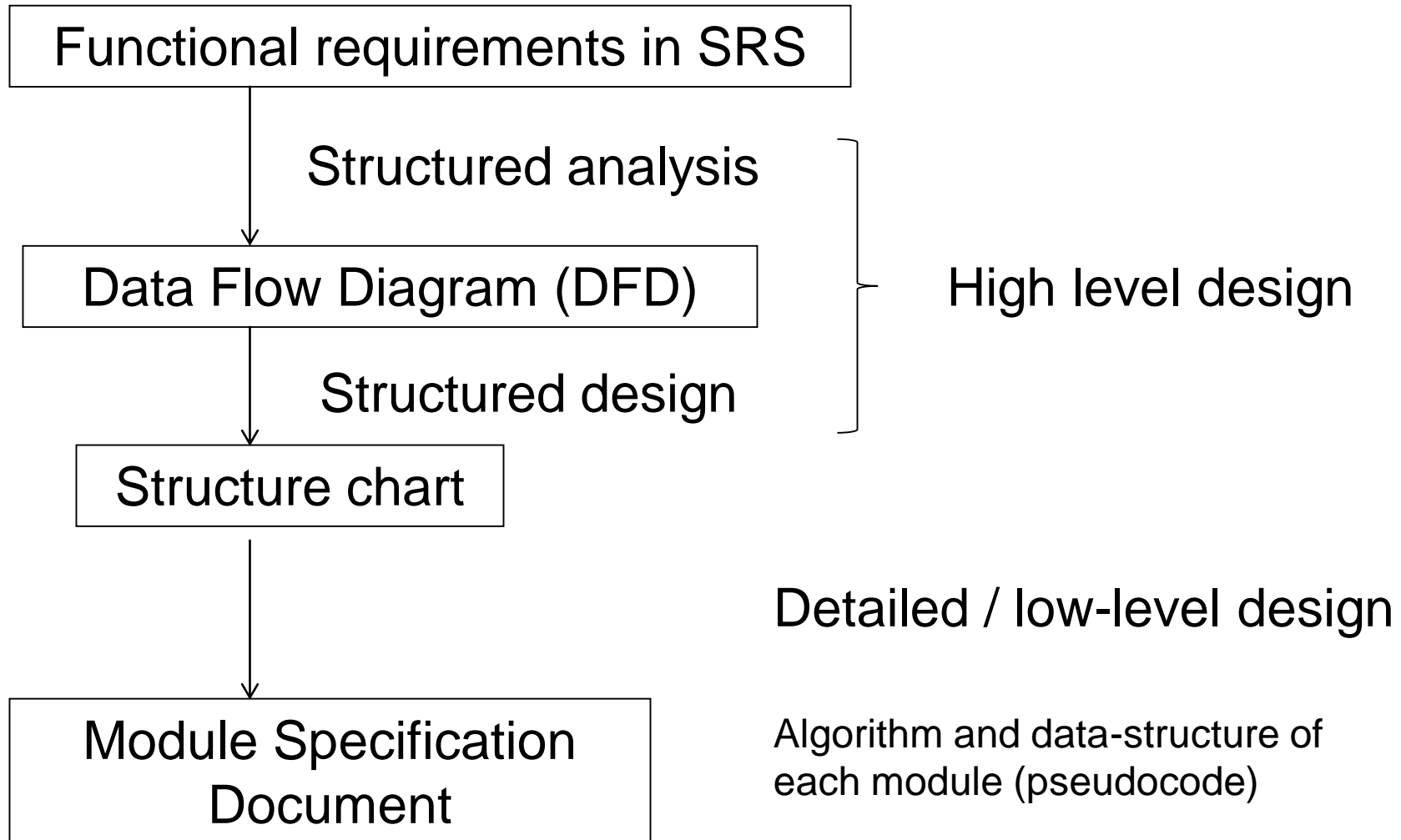


Software Design

Function Oriented Design

Function-oriented design: summary



SA/SD methodology

- One of several function-oriented design methodologies, used for high-level design
 - **Structured analysis (SA)**
 - Top-down decomposition of high-level functions into detailed functions
 - Simultaneous decomposition of high-level data
 - Transform textual problem description into a graphic model (DFD)
 - **Structured design (SD)**
 - Map detailed functions represented in DFD to modules
 - Module structure is formalized (software architecture)
-

SA vs SD

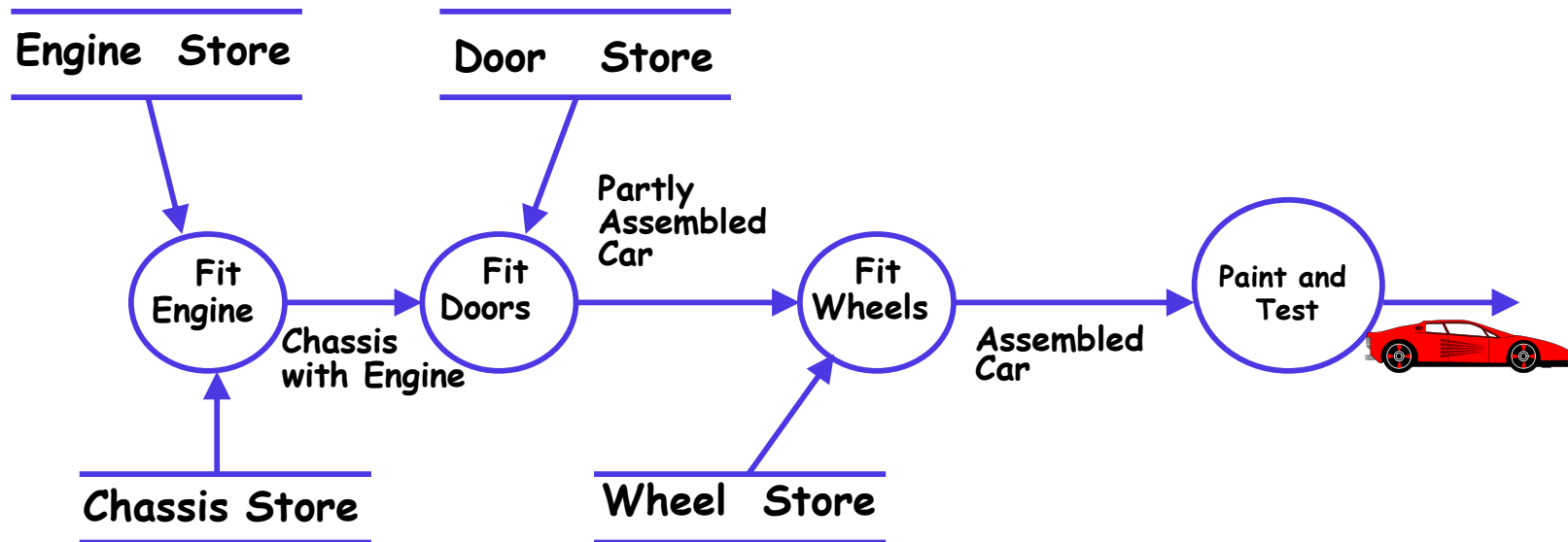
- Purpose of structured analysis
 - ❑ Capture the detailed structure of the system as the user / customer views it
 - ❑ Use customer terminology for functions & data
 - ❑ Can be reviewed by customer, knowledge of computer not required to understand DFD

- Purpose of structured design
 - ❑ Arrive at a form that is suitable for implementation in some programming language
 - ❑ To be used in subsequent implementation phase

Structured Design

Data Flow Diagram (DFD)

- Graphical representation of data flow in a system
 - Hierarchical model
 - Also called bubble chart
 - Each function considered as a processing station which consumes some input, produces some output



DFD symbols

- External entity (also called source / sink)

- Real physical entities who input data to the system or consume data produced by system

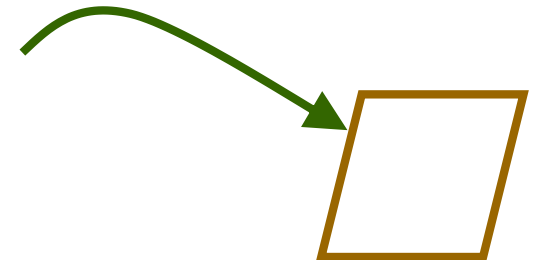
Librarian

- Function

- Symbol also called process / bubble / transform
- Bubbles annotated with name of function
- Name should be a verb

**search-
book**

- Output produced by the system



DFD symbols

■ Data flow symbol

- Data flow in the direction of the arrow
- Annotated with names of data they carry

book-name



■ Data store

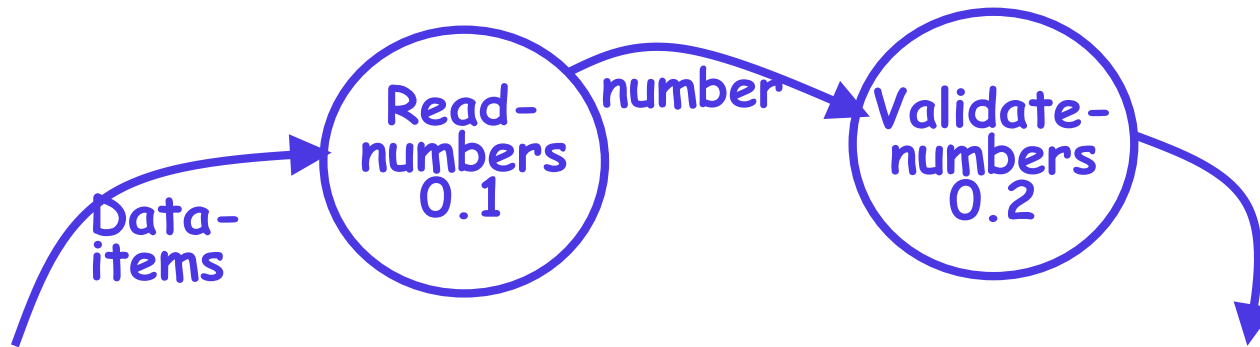
- Represents a logical file (data structure in memory, or physical file in disk or database)
- An arrow into or out of a data store
 - Direction shows whether data is read from / written to data store
 - Implicitly represents the entire data of the data store, need not be annotated with any data name

book-details

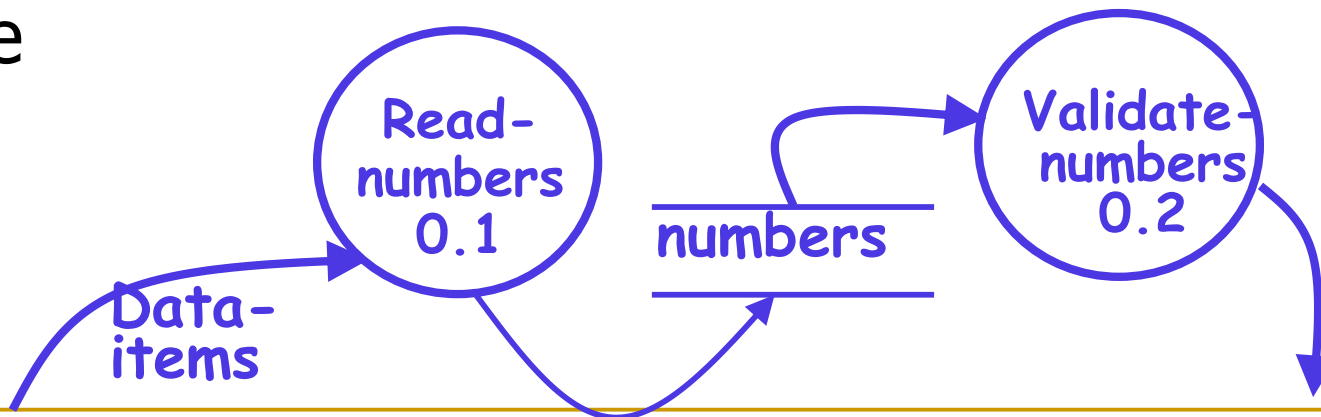


Synchronous vs asynchronous

- Synchronous: two bubbles directly connected by data flow arrow

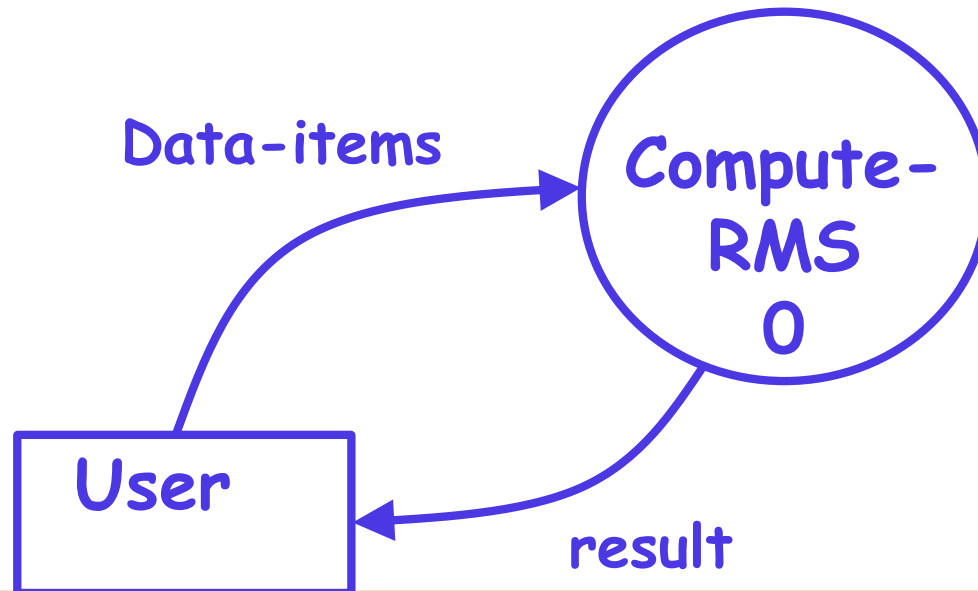


- Asynchronous: two bubbles connected via a data store



Performing structured analysis

- Initially represent system at the most abstract level
 - Called the **context diagram** or **Level 0 DFD**
 - Entire system represented as a single bubble (labeled)
 - Show data input to system, output data generated by system, external entities



Performing structured analysis

- Level 1 DFD

- For each high-level function in the SRS ...
- Represent each function as a bubble
- Represent data input to / output from each function

- Higher level DFDs

- Each high-level function separately decomposed into sub-functions
 - DFDs represent the sub-functions, data input to / output from each sub-function
-

Decomposition of a bubble

- Also called factoring or exploding a bubble
 - Each bubble usually decomposed into 3 to 7 bubbles
 - Decomposition of a bubble should be carried on until a level at which each function can be described by a simple algorithm
 - Numbering of bubbles
 - Single bubble in context diagram numbered 0
 - When bubble numbered x is decomposed, its children bubbles are numbered $x.1$, $x.2$, $x.3$, ...
-

Data Dictionary

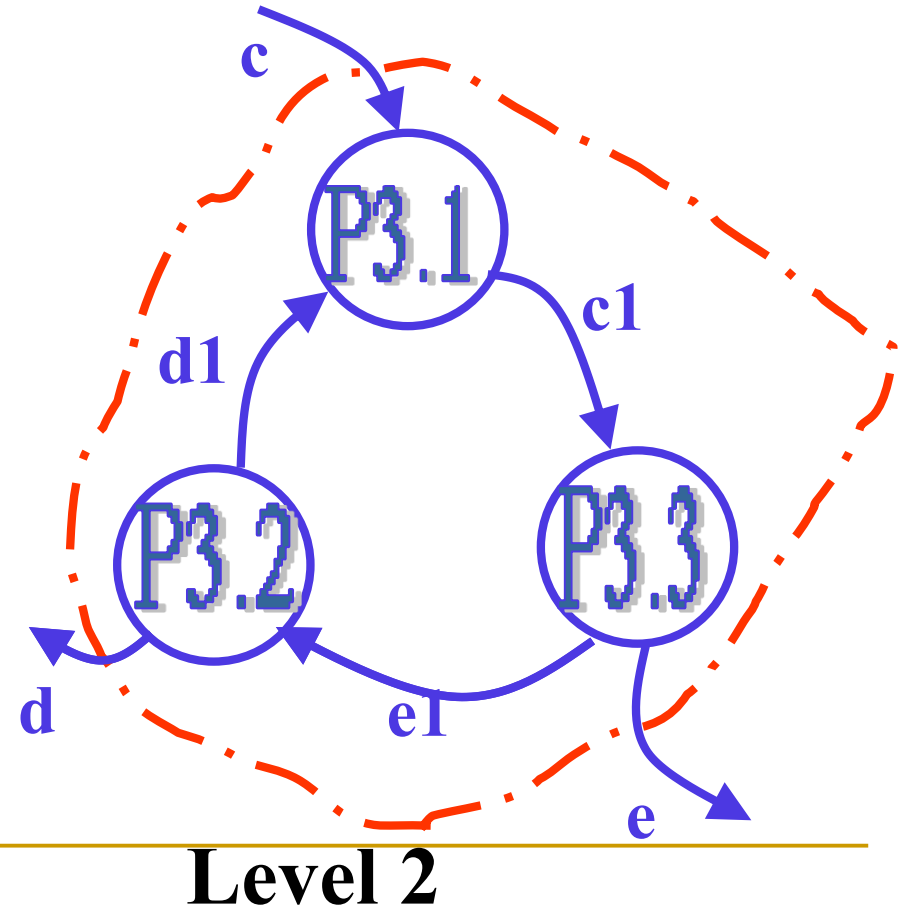
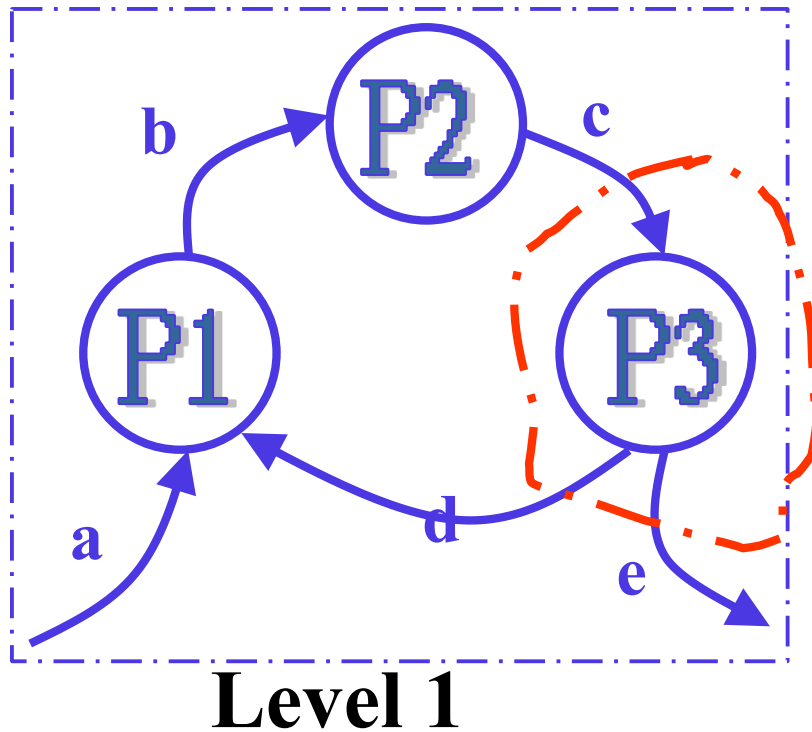
- A DFD is always accompanied by a data dictionary
- Data dictionary lists all data items appearing in DFD
 - Definition of all composite data items in terms of their component data items
 - All data names along with the purpose of the data items
 - Example of a data dictionary entry:
 $\text{grossPay} = \text{regularPay} + \text{overtimePay}$
- Importance of Data Dictionary
 - Provides everyone involved in project with standard terminology for all data (consistent vocabulary)

Data definition in data dictionary

- Composite data are defined in terms of primitive data items using following operators
 - + denotes composition of data items
 - [, , ,] represents selection, i.e. any one of the data items listed inside the square bracket can occur
 - Contents inside the brackets () represent optional data e.g. $a+(b)$ represents either a or $a+b$ occurs
 - { } represents iterative data definition, e.g. {name}5
 - {name}* represents zero or more instances of name data
 - = represents equivalence
- Anything between * ... * is a comment

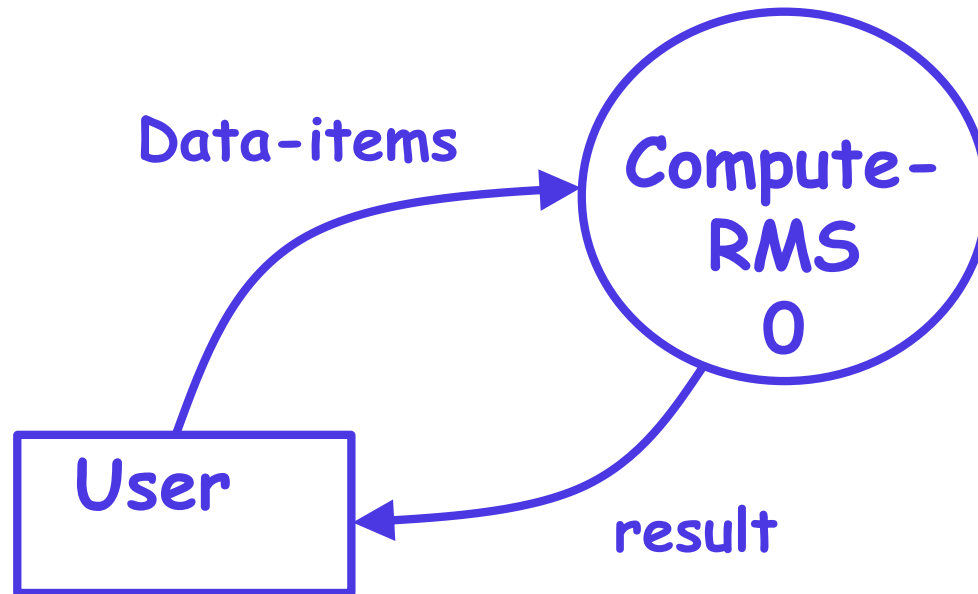
Balancing a DFD

- Data flow into or out of a bubble at level j DFD must match the data flows at level $j+1$ DFD

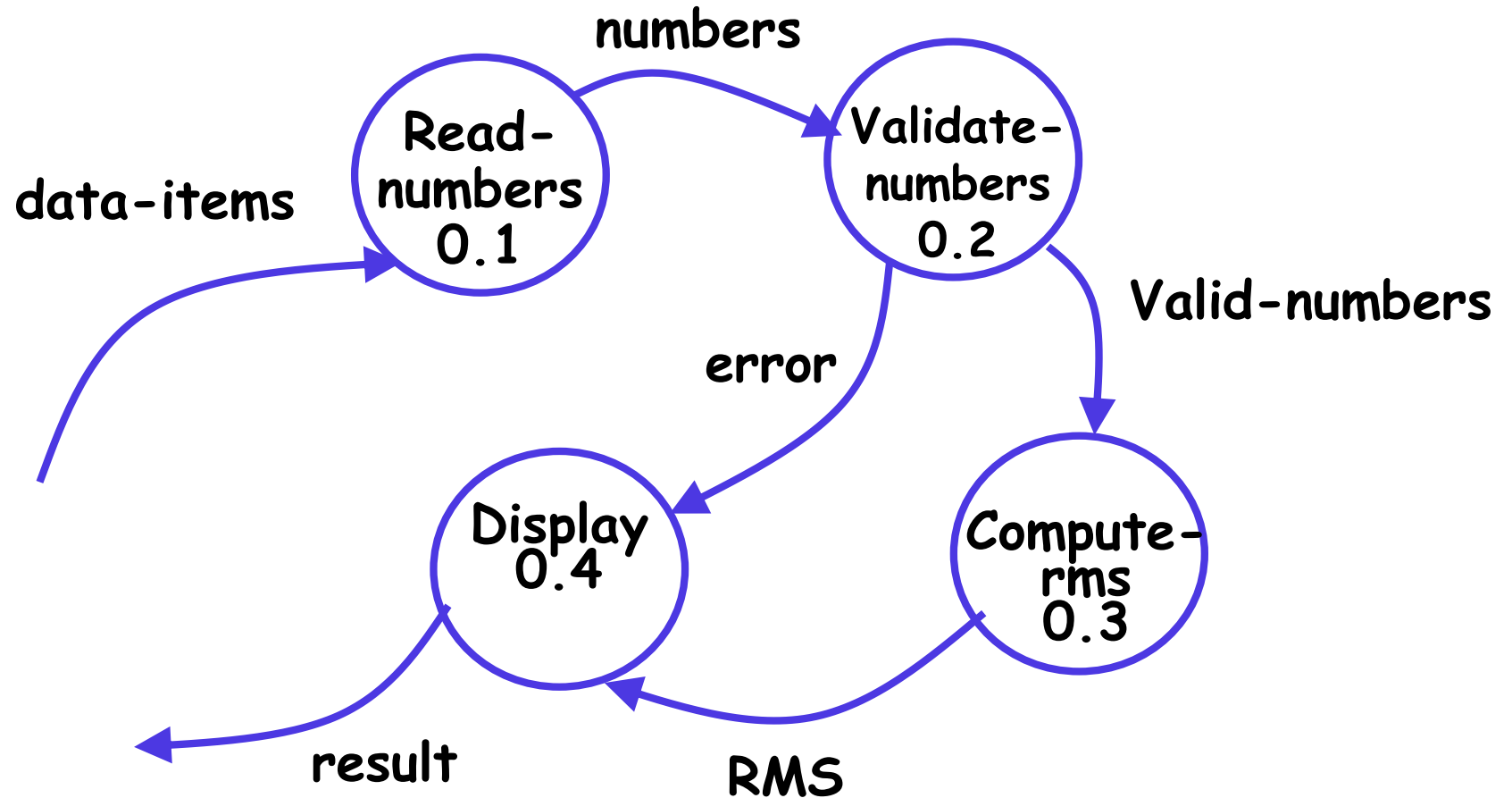


Example: RMS calculating software

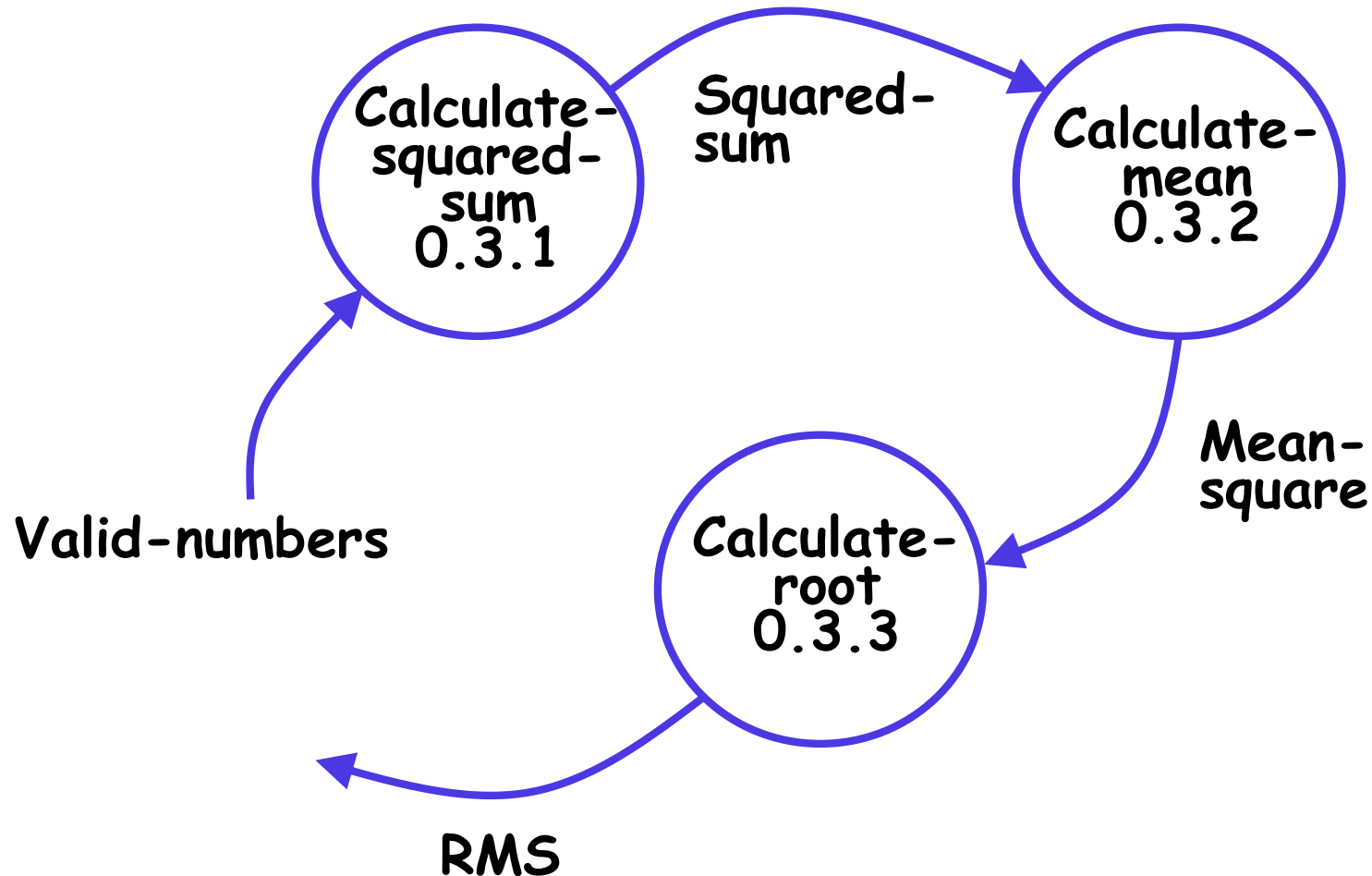
- RMS calculating software
 - ❑ Reads three integers in the range of -1000 and +1000
 - ❑ Finds out the root mean square (rms) of input numbers
 - ❑ Display the result



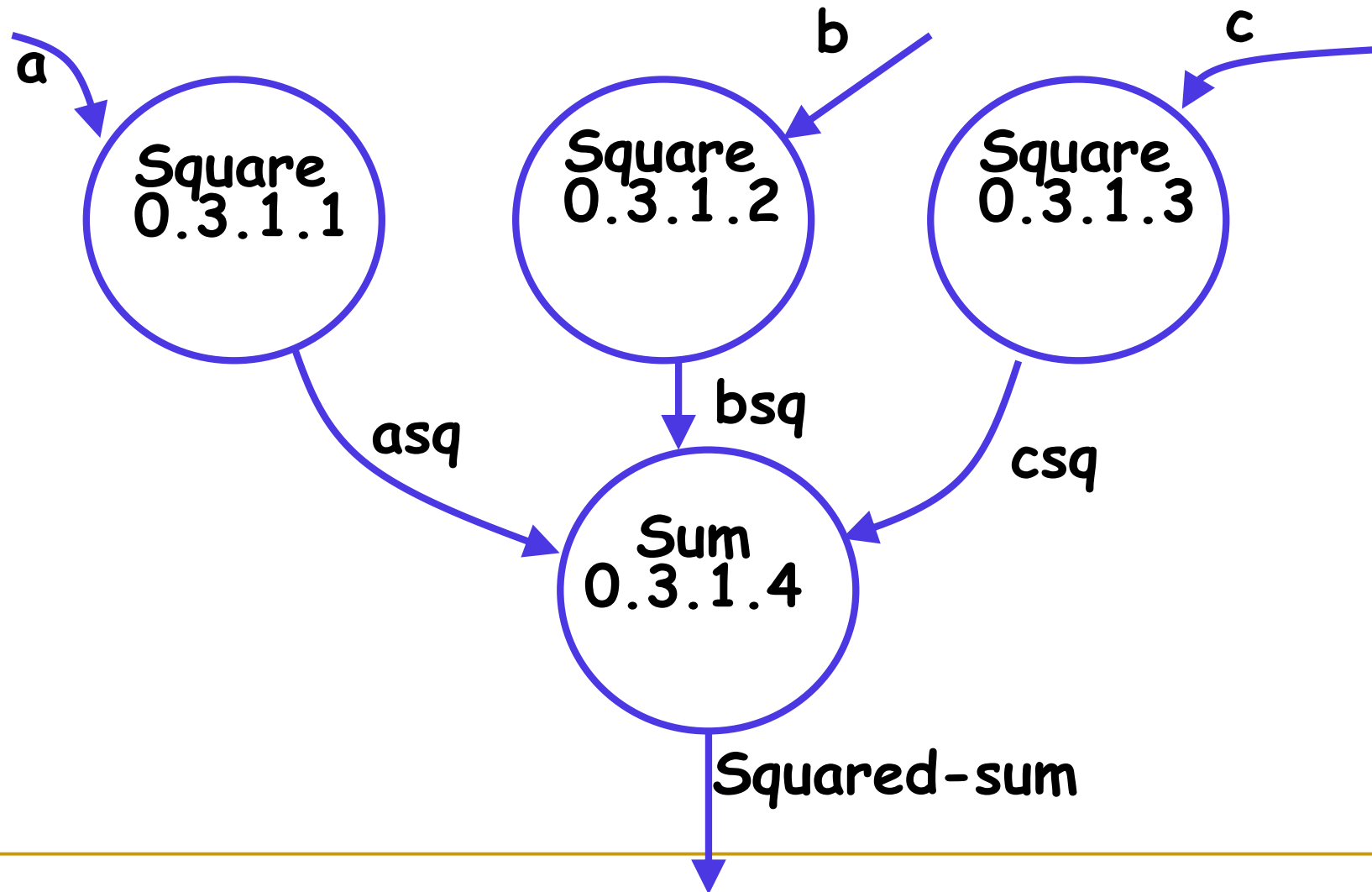
RMS software: Level 1 DFD



RMS software: Level 2 DFD



RMS software: Level 3 DFD



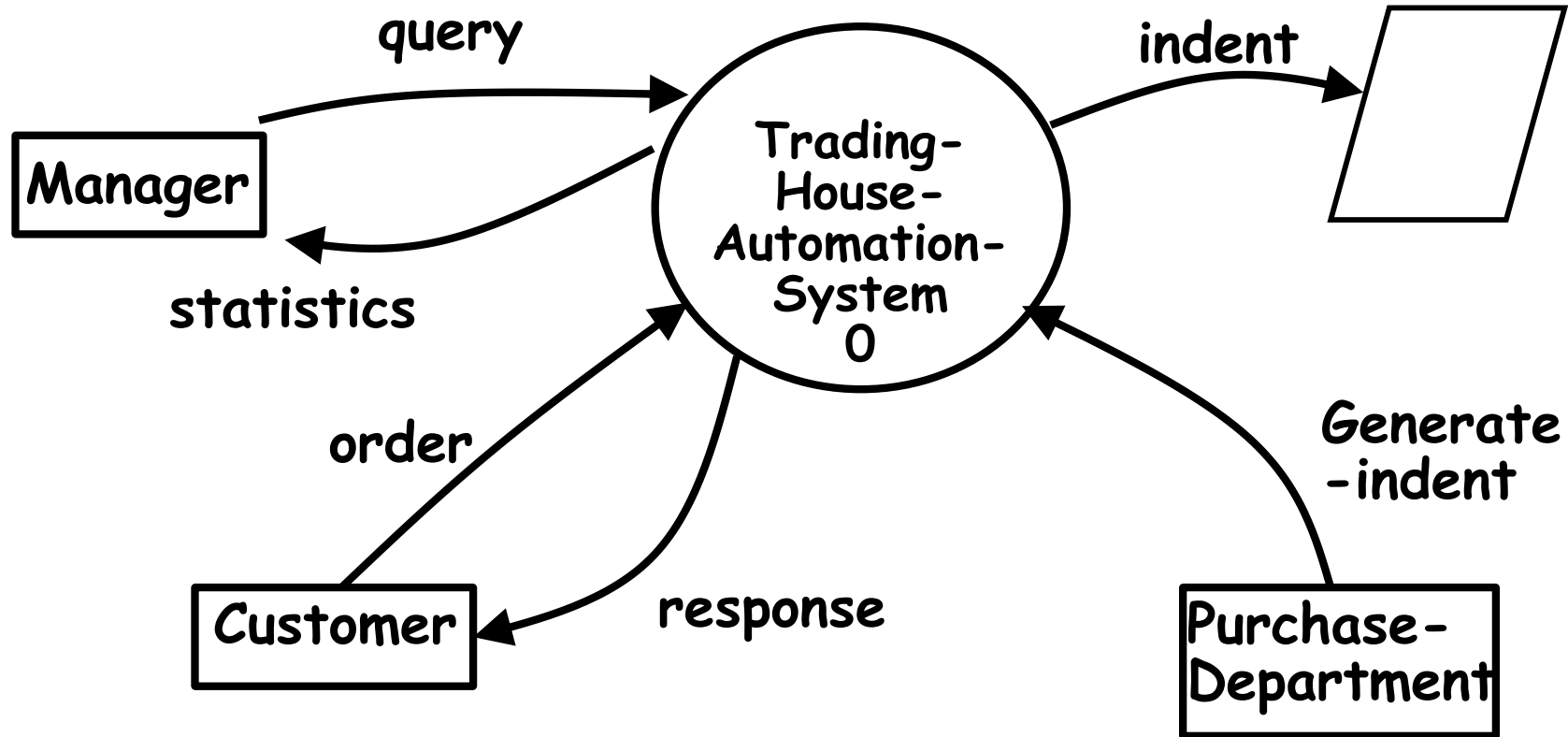
RMS software: Data Dictionary

- data-items = {integer}3
- numbers = data-items
- valid-numbers = numbers
- a: integer * input number *
- b: integer * input number *
- c: integer * input number *
- asq: integer
- bsq: integer
- csq: integer
- squared-sum: integer
- RMS: float * root mean square value*
- error: string * error message*
- Result = [RMS,error]

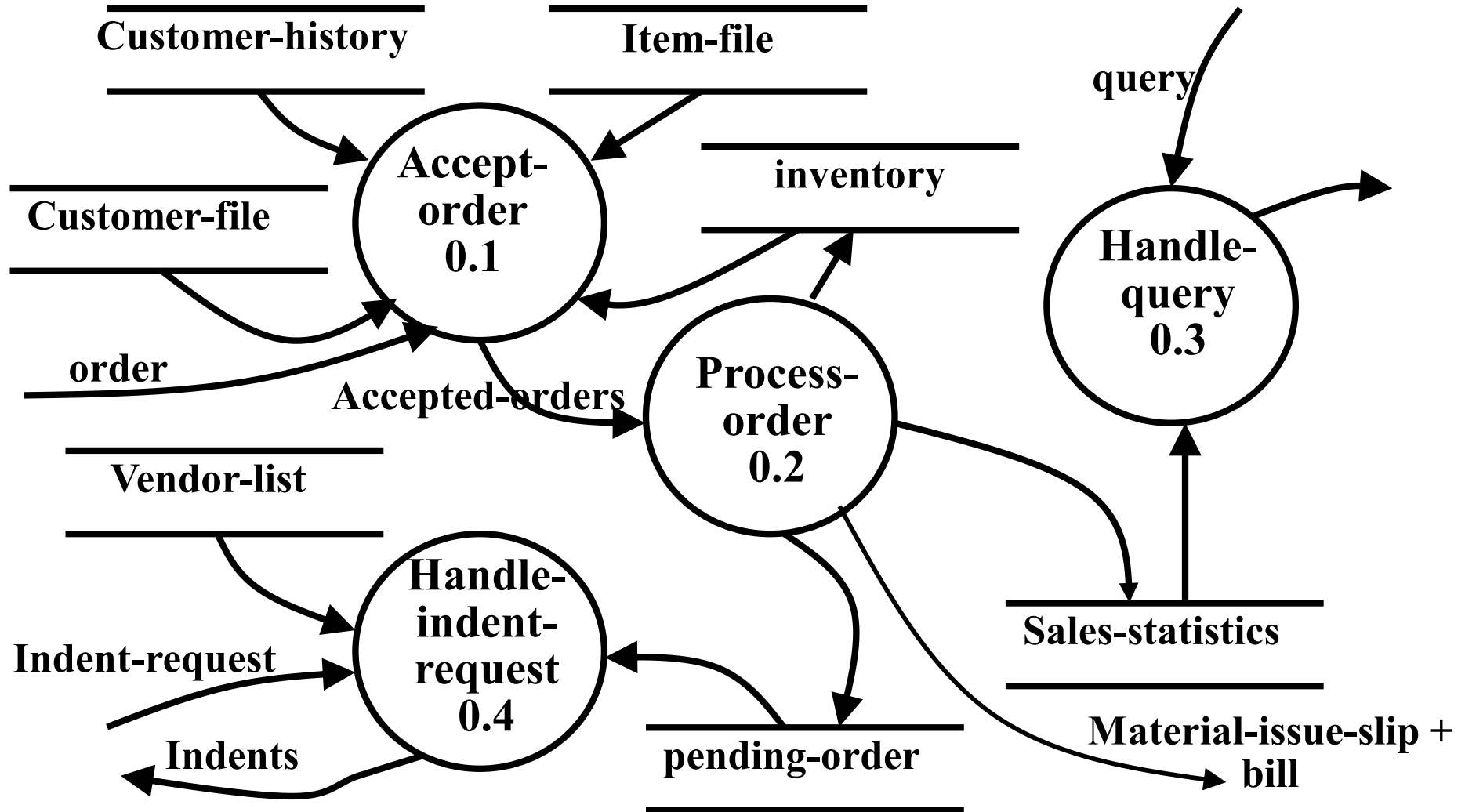
Example: Trading house automation software

- To automate book keeping activities associated with the business of a trading house
 - ❑ Customers place orders for various items
 - ❑ Before accept order, check if customer is credit-worthy
 - ❑ If order accept, customer issued a bill & material issue (to be produced at storehouse to get the items)
 - ❑ If ordered items not available, store details in a pending-orders file
 - ❑ Purchase department periodically generates an indent to know what needs to be bought (pending orders)
 - ❑ TAS software should also answer managerial queries
-

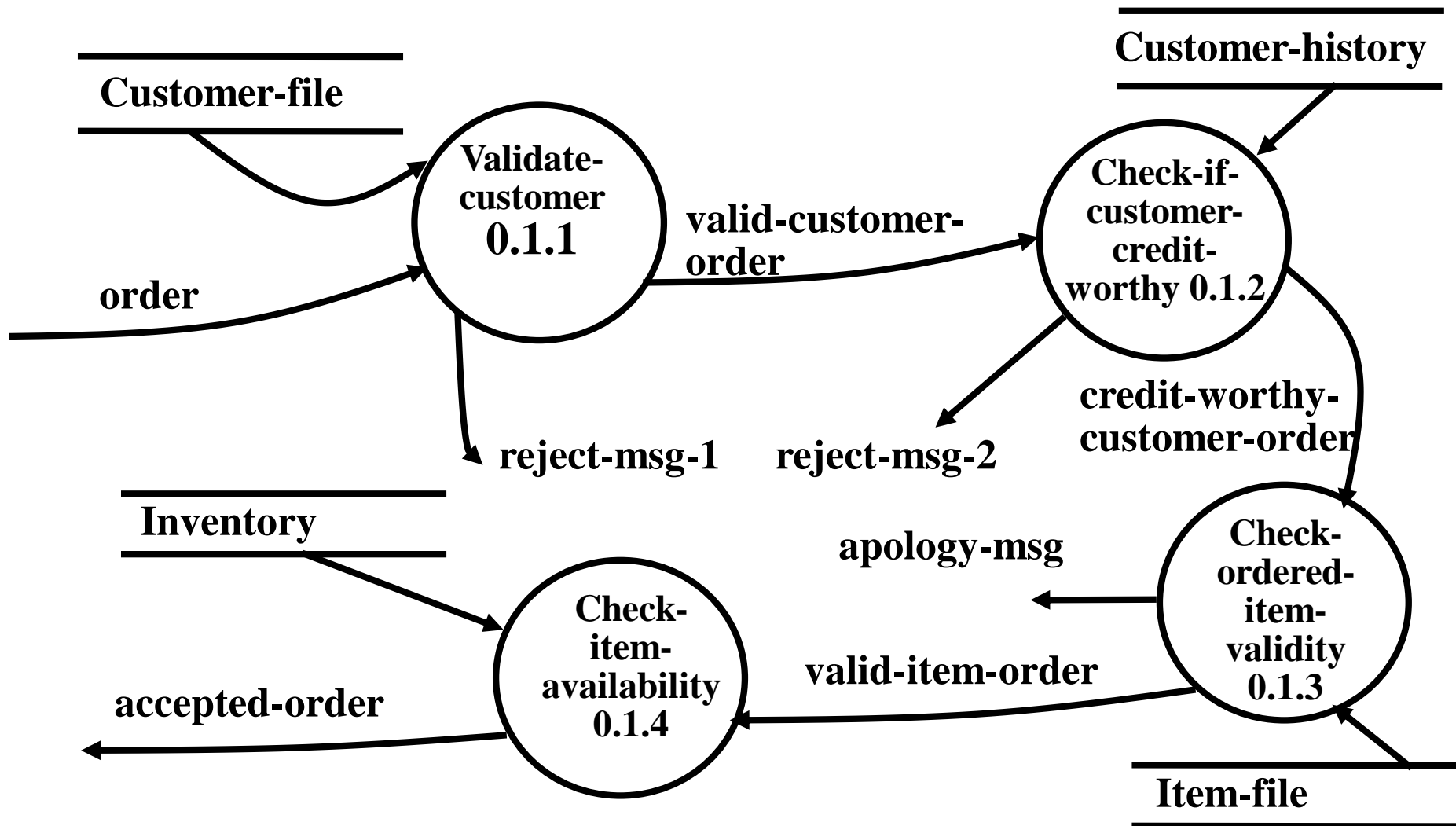
TAS software: Context Diagram



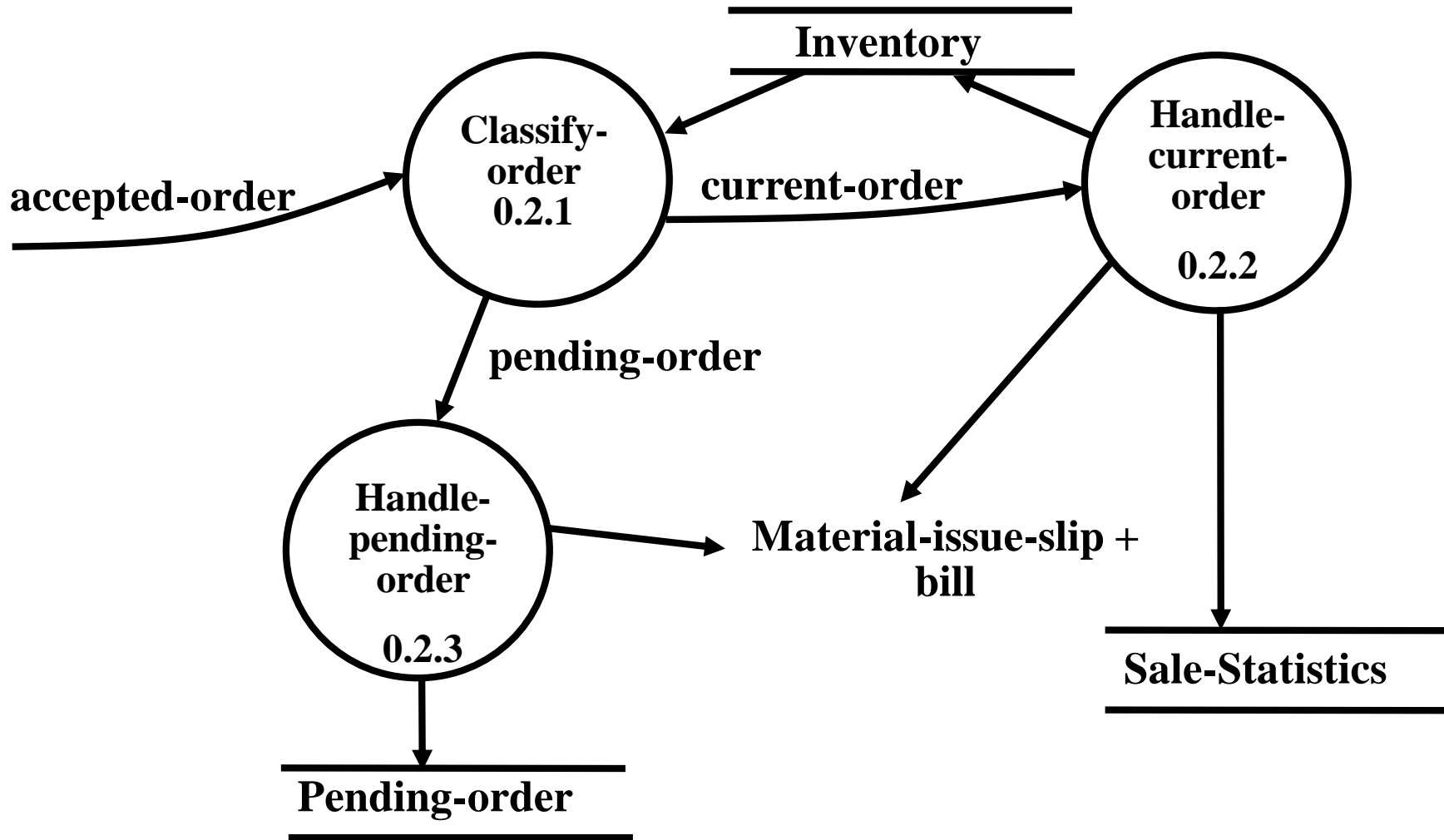
TAS software: Level 1 DFD



TAS : Level 2 DFD : Accept Order



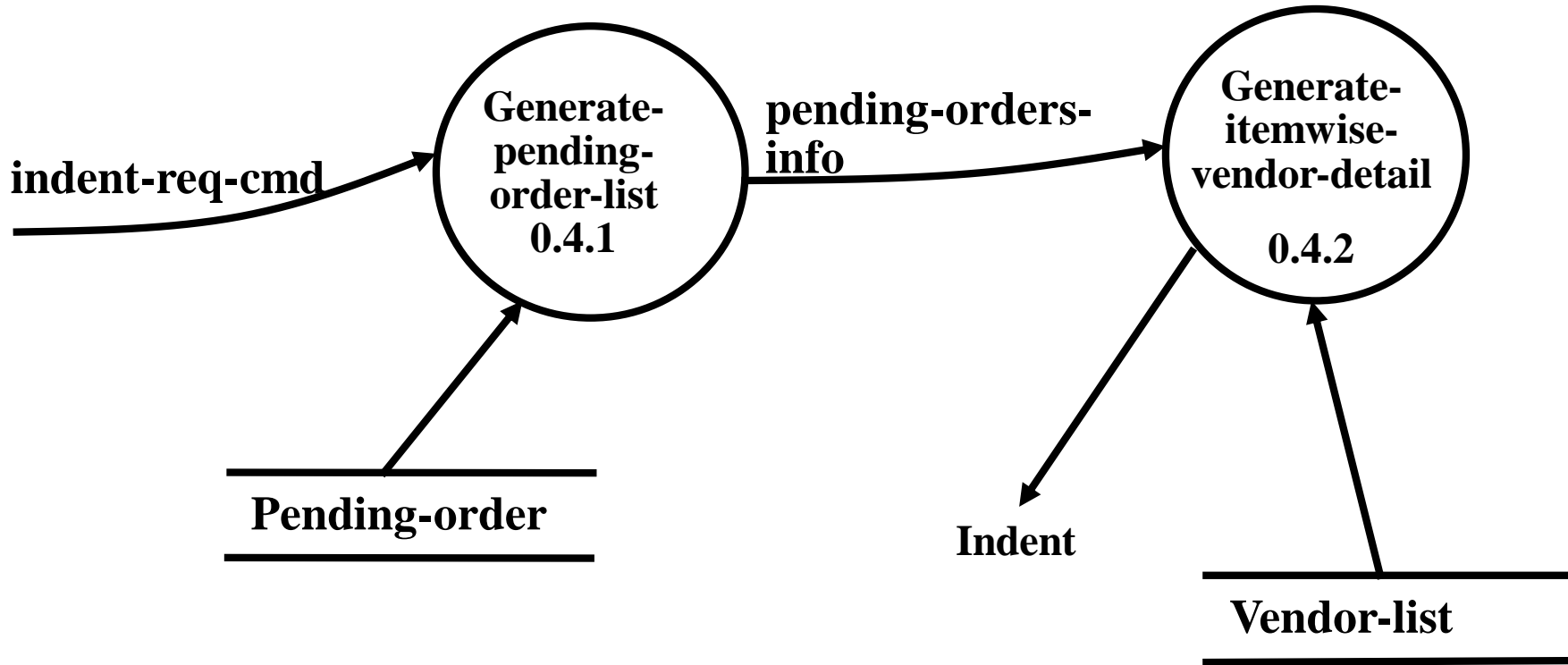
TAS : Level 2 DFD : Process Order



TAS : Level 2 DFD : Handle Query

- Skipping level 2 DFD for Handle Query

TAS : Level 2 DFD : Indent Request



More about DFD

- DFDs help create
 - **Function model** – analyst is performing an implicit functional decomposition
 - **Data model** – deciding what data items need to be exchanged, refinements of data, etc
 - External entities appear only in context diagram, not in higher level DFDs
-

More about DFDs

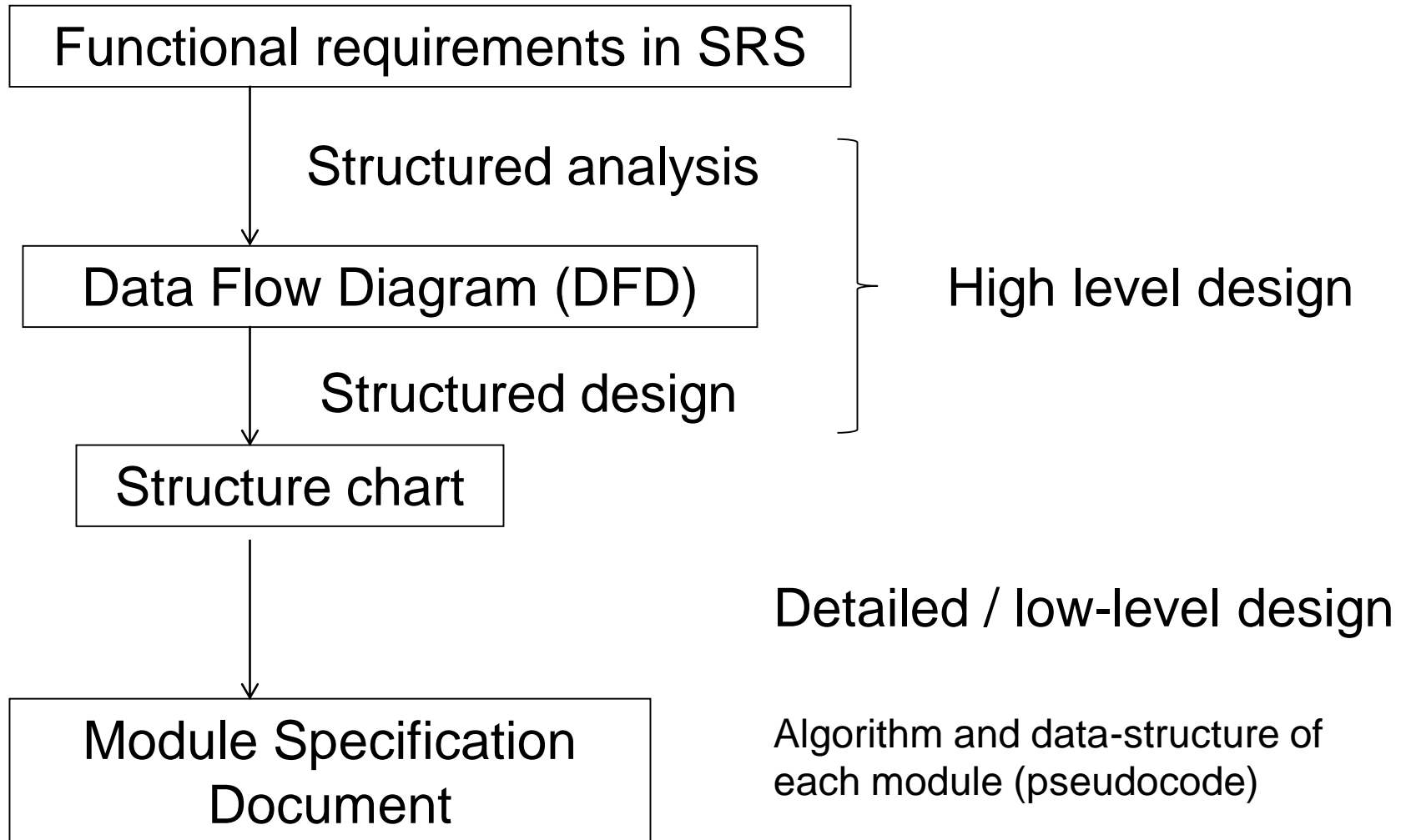
- DFD does not represent control information
 - When or in what order different functions are invoked
 - Conditions under which different functions are invoked
 - If bubble A invokes either bubble B or bubble C depending on some conditions
 - Represent the data that flows from bubble A to bubble B and from bubble A to C, ...
 - not the conditions depending on which a process is invoked
-

Shortcomings of DFD model

- Ample scope to be imprecise
 - May not specify action for wrong / missing input
 - Control information not represented
 - Synchronization aspects not specified
 - Subjective way of decomposing functions
 - Many decompositions possible for same system
 - No objective way of determining which is better
 - Level to which a function needs to be decomposed is subjective
-

Structured Design

Function-oriented design: summary



Structured design

■ Goal

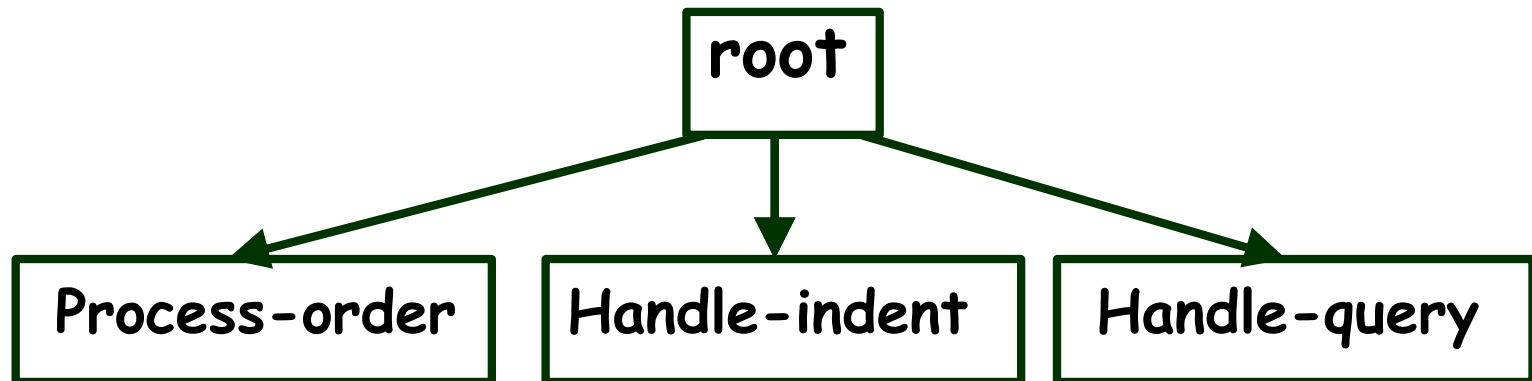
- ❑ Transform results of SA (e.g. DFD) into software architecture (represented by Structure Chart)

■ Structure Chart specifies

- ❑ Various modules making up the system (module structure)
- ❑ Module dependency (i.e. which module calls which other modules)
- ❑ Parameters passed among different modules
- ❑ Procedural aspects (e.g. how a particular functionality is achieved) are not represented

Structure chart symbols

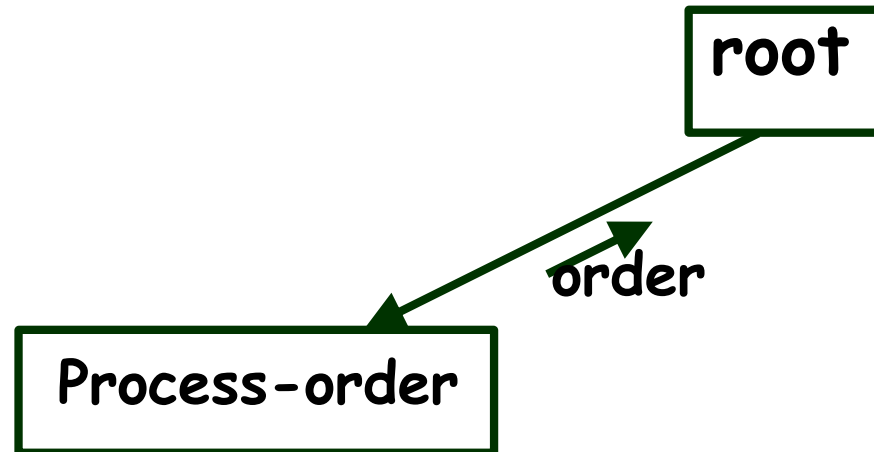
- Module: rectangle, annotated with name of module
- Arrow between two modules
 - During execution, **control** passed from one module to the other in the direction of the arrow



Structure chart symbols

- Data flow arrows

- Data passing from one module to another in the direction of the arrow



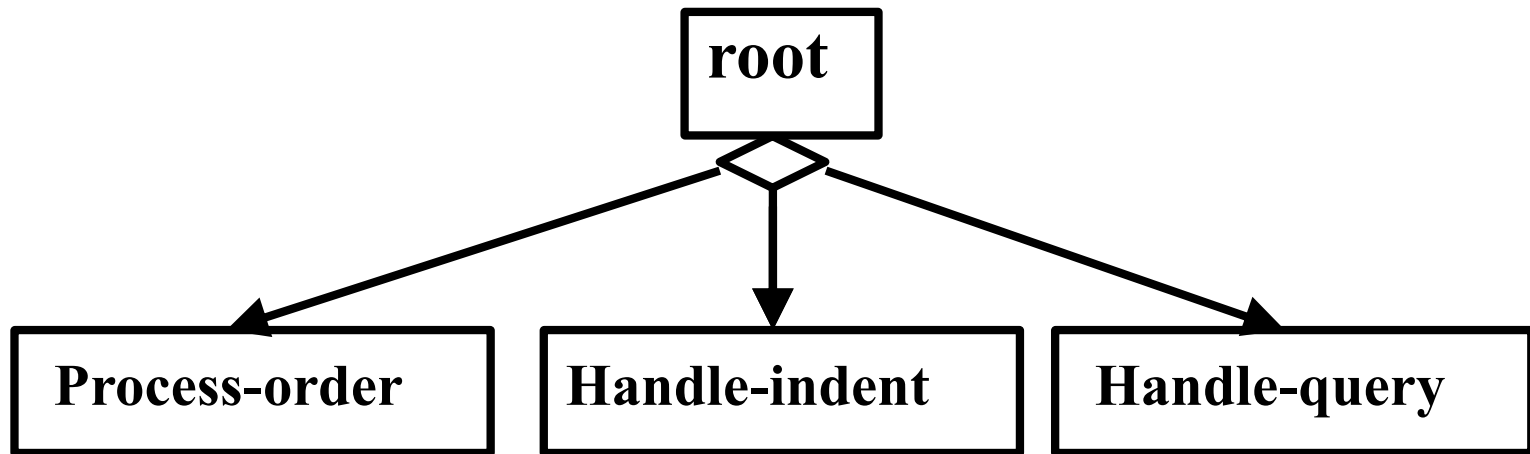
- Library modules

- Represents frequently used modules (i.e. A module that is called by several modules)



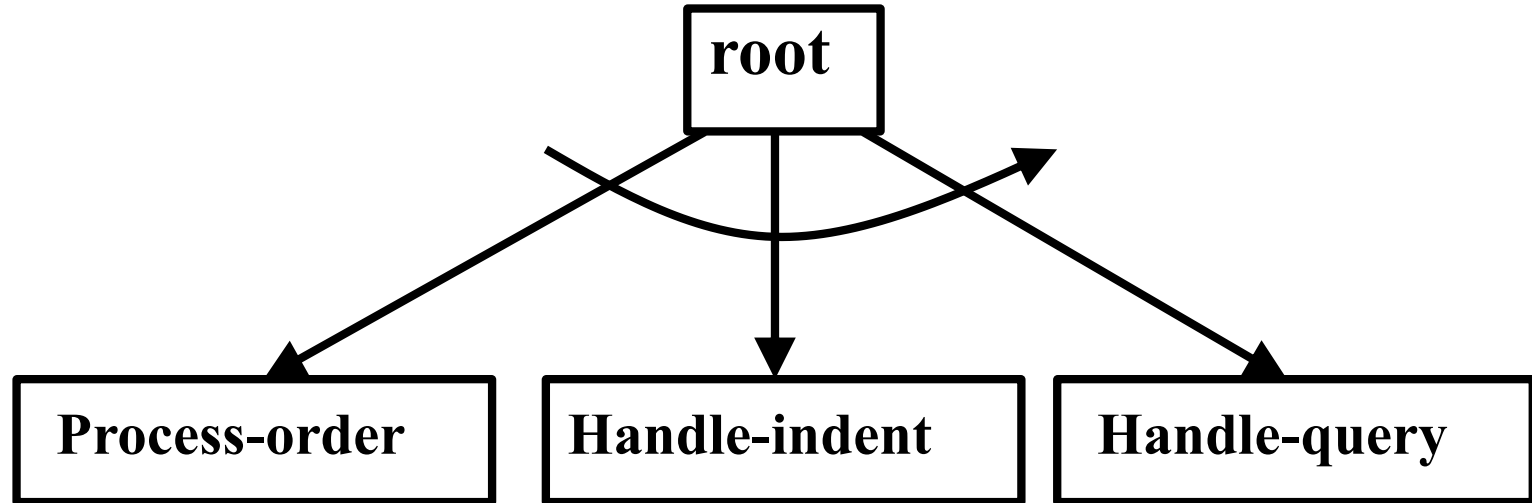
Structure chart symbols

- Selection: diamond symbol
- One of the several modules connected to the diamond symbol is invoked depending on some **condition**



Structure chart symbols

- Repetition: A loop around control flow arrows denotes that the concerned modules are invoked repeatedly



Rules for structure chart

- Modules arranged in layers or levels
 - Only one module at the top level – root module
 - At most one control relationship between two modules
 - Principle of abstraction: lower-level modules should not need to call higher-level modules
 - Shortcomings of structure chart
 - Flow of control within modules is not represented
 - Does not specify order in which different modules invoked
-

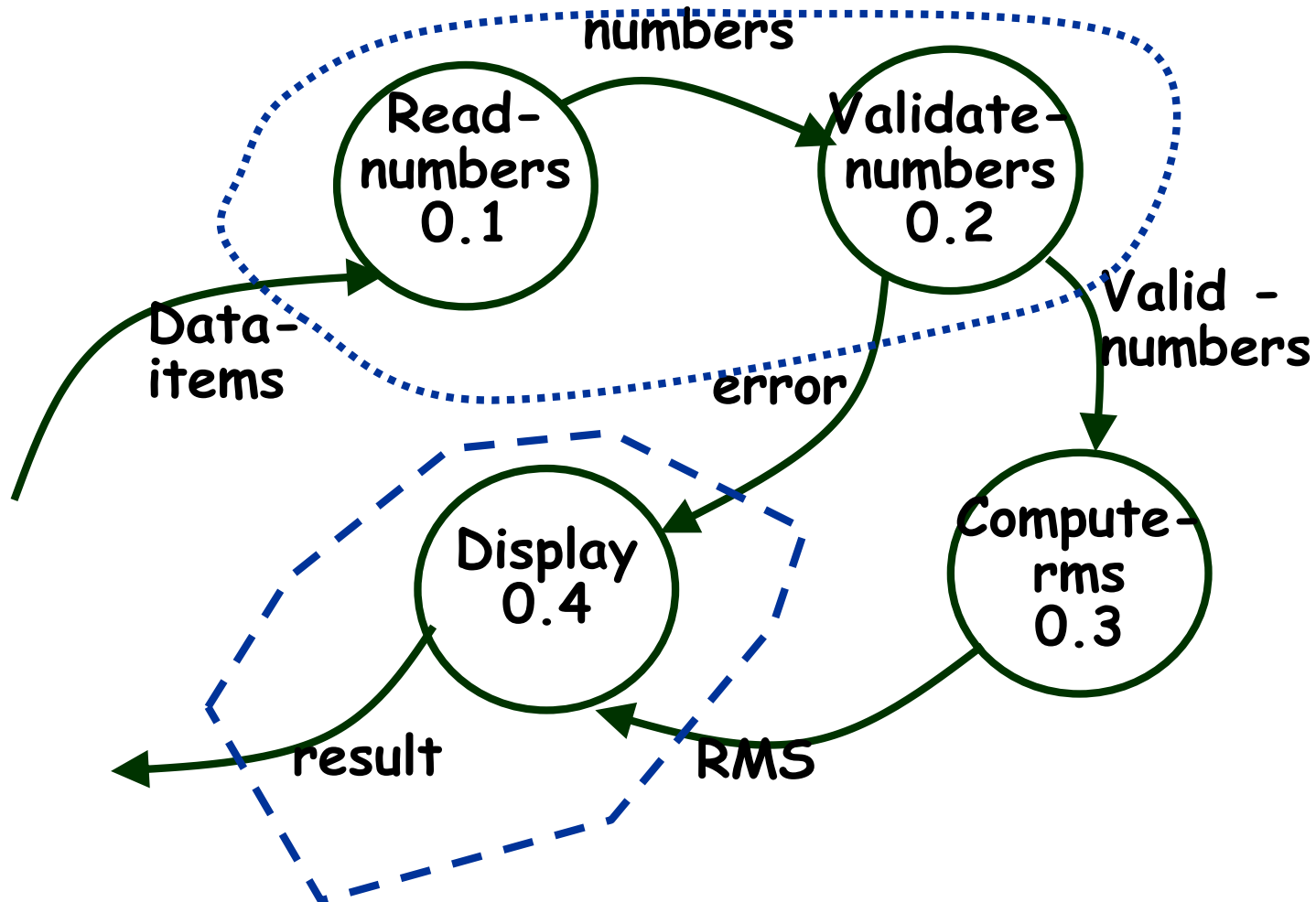
Transforming DFD to structure chart

- Two strategies to transform DFD to structure chart
 - Transform analysis
 - Transaction analysis
 - Transform analysis (for transform-centered system)
 - Similar processing steps for every data item
 - E.g. input, process and output bubbles
 - Transaction analysis (for transaction-driven systems)
 - One of several possible paths through the DFD is traversed depending upon the input data value
-

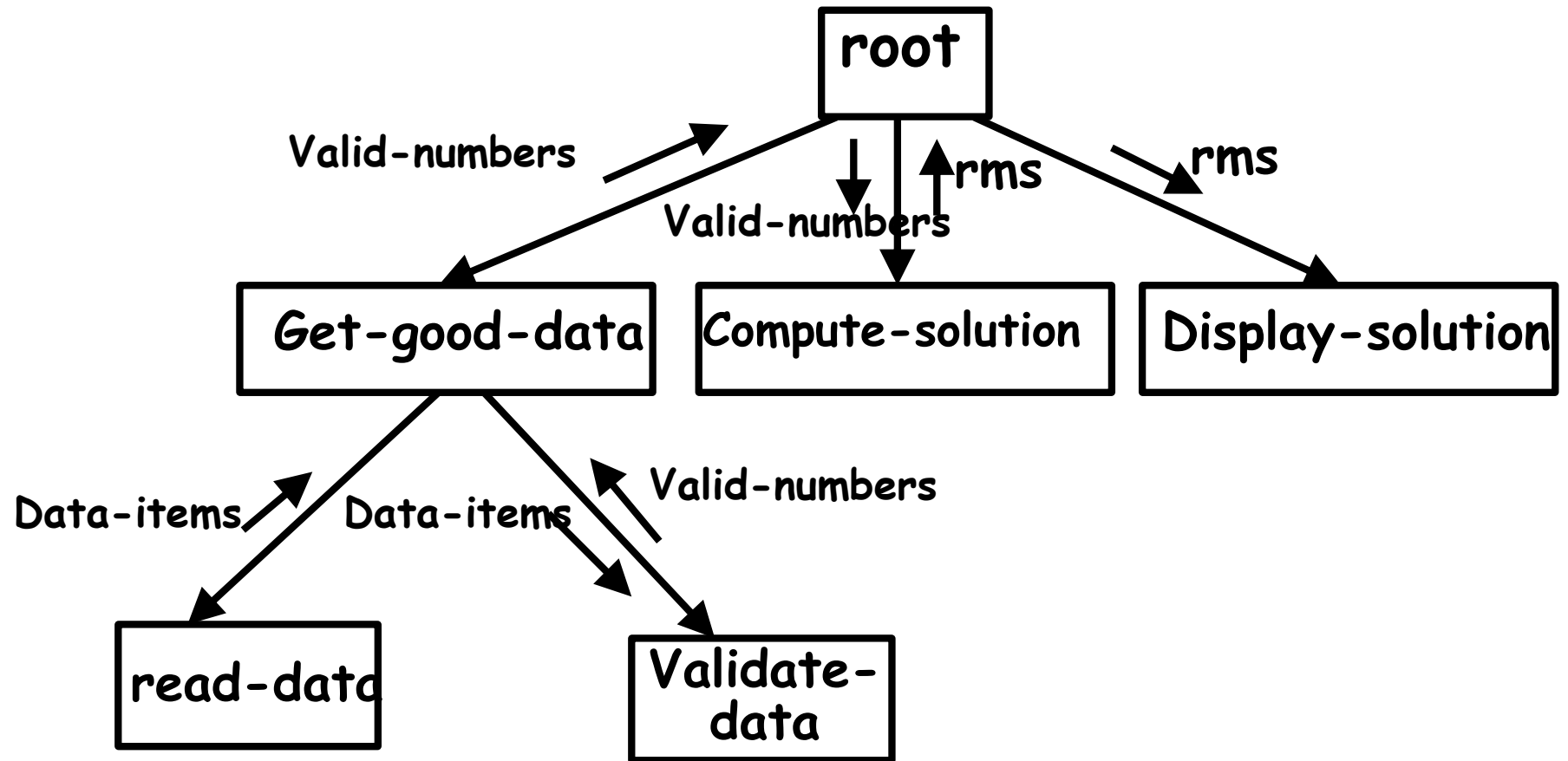
Transform analysis

- Divide DFD into three portions – input, logical processing, output
 - Each input portion called an afferent branch
 - Each output portion called an efferent branch
 - Possible to have more than one afferent / efferent branch in a DFD
 - Derive structure chart
 - Draw one functional component for each efferent and afferent branch, one for the central transform
 - Add sub-modules required by each high-level module (also called factoring)
-

Level 1 DFD for RMS calculator ...



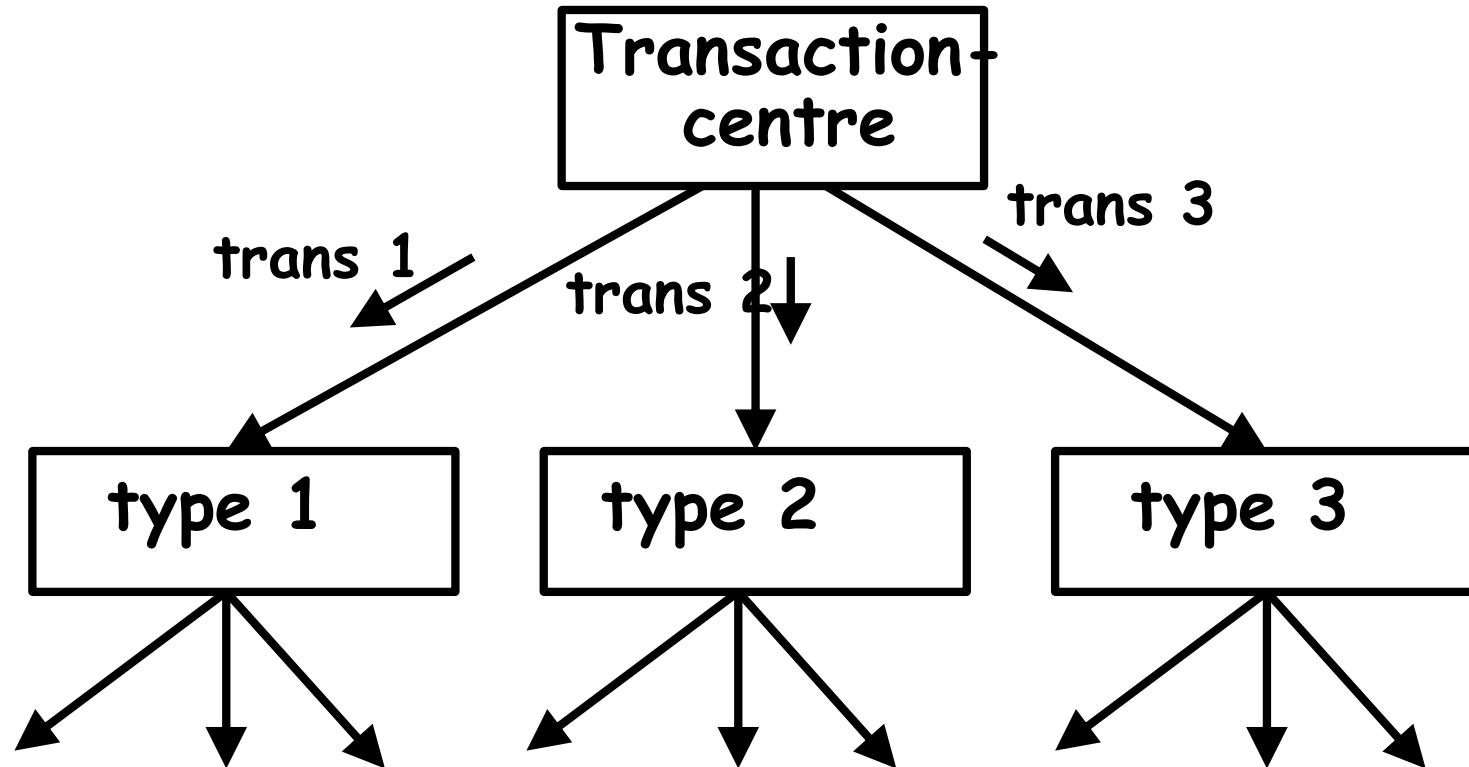
... converted to Structure Chart



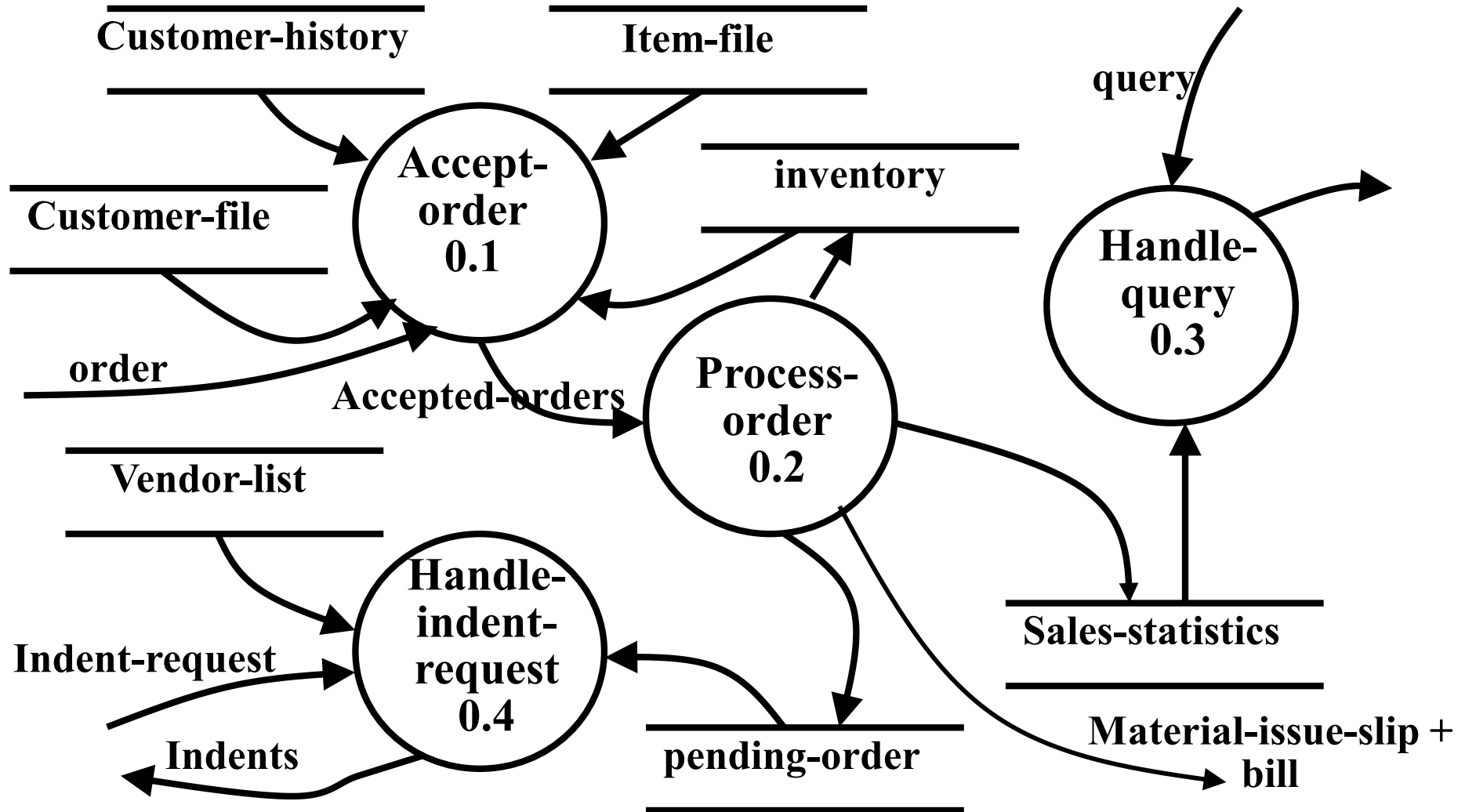
Transaction analysis

- Transaction-driven system
 - E.g. selected menu options might trigger different functions
 - Transaction analysis divides system into
 - Several transaction modules
 - One transaction-centre module
-

Transaction analysis



TAS software: Level 1 DFD



... converted to Structure Chart

