The Data Flow Diagram

DS 2004

S.S.A.D.M.

- S.S.A.D.M. - Structured Systems Analysis and Design Method
- Uses different techniques to model a system
  - Data Flow Diagrams
  - Entity Relational Model (Logical Data Stores)
  - Normalisation

What is a Data Flow Diagram?

- Known as DFDs
- A way to model a real world situation
- They are the interface between the real world activities and an understanding of how this can be converted into a computer system.

Why do we use DFDs?

- It is a way of taking the physical view and converting it into a logical view.
- The physical view - all documents involved
- The logical view - the data they contain
- Their main purpose is to communicate with the user, the analyst's understanding of the scope of the required system

Levelling

- DFDs are expanded or decomposed into levels.
- Separating each process into sub processes
- Uncovers more and more detail

Conventions

Balancing

Process at lower level should have identical data flows if they flow out of a process

Modelling Data Stores

Only use DATA STORES used within this process on the diagram

Numbering

1 - 1.1 - 1.1.1
1.2 - 1.2.1

Labels

Should carry as much meaning as possible

Decomposition and Abstraction

- **Decomposition - Divide and subdivide into manageable size problems**
- **Abstraction - Concentrate on the important issues and ignore the irrelevant**

The Elements

The four main elements of DFDs notation

- **Data Flows, with a label to indicate what data is flowing**
- **Processes, that handle the data**
- **Data stores, within the system (diary, filing cabinet or computer file)**
- **External/Outside entities/Terminator, outside sources of data**

The Data Flow Diagram

- Looks at the system from point of view of a single piece of data.
  - Not reiterative -- no loops shown.
  - As a result, we cannot program directly from a DFD.

The Data Flow Diagram

- Four symbols:
  - Terminator/external entities
  - data store
  - process bubble
  - data flow

Symbols

- Terminator/External Entities
  - Person or organization that lies outside the system and that is a net originator or receiver of data.

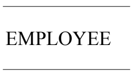
EMPLOYEE

- Key - outside the area of our concern and control.

Symbols

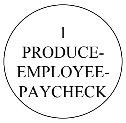
- ***Source*** (originator of data) or ***sink*** (receiver of data).
- Prime sources on the left side of the DFD, prime sinks to right.
- Name inside box.
- Also called an external entity.

Symbols



- Data store (file)
  - Same as the data store in the data dictionary.
  - Could be a computer file, card file, file cabinet, etc.
  - Note that EMPLOYEES here is the data store that contains the employee information, while EMPLOYEE (the terminator) is the actual person.
  - Size: about 1 inch by 1/2 inch.

Symbols



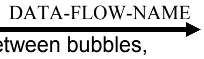
- Process (bubble, transform)
  - An activity, task, function, etc.
  - Shows work being done against the data.
    - Transforms incoming data into outgoing data.
    - Changes status (logical) or content, format, or medium (physical).

Symbols

- Each bubble has a unique number and name.
  - The name must be an active verb followed by object clause:
    - EDIT-CUSTOMER-PAYMENT
    - WRITE-PAYMENT-REPORT
  - If no active verb, it's not a process!

Symbols

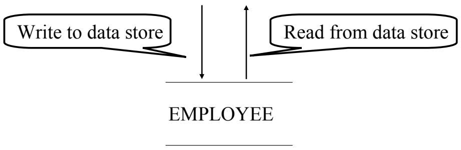
- Data flow



  - The data interface between bubbles, terminators, and data stores.
  - Should be a packet of logically related data items.
    - good--CUSTOMER-PAYMENT-TRANSACTION
    - bad--MISCELLANEOUS-STUFF
  - No excess data passed around.
    - ***Tramp data*** is not acceptable.
    - Data flows should be lean and mean.

Symbols

- Arrows show direction of data movement.
- Into and out of a data store...



- The access to a data store (request or key) is not shown, only the net result.

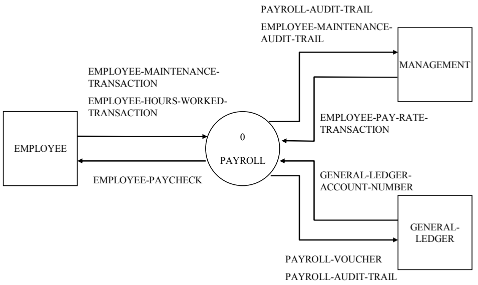
Symbols

- Naming
  - Unique, descriptive.
  - Data dictionary naming conventions (because all of these names need to be in the DD, too).
  - No loops, so never GET-NEXT-CUST.
  - No flags.
  - Avoid vague names like -INFO, -DATA.
    - Can usually (but not always) be more specific.
    - Real test--can you write a DD entry?

Context Level DFD

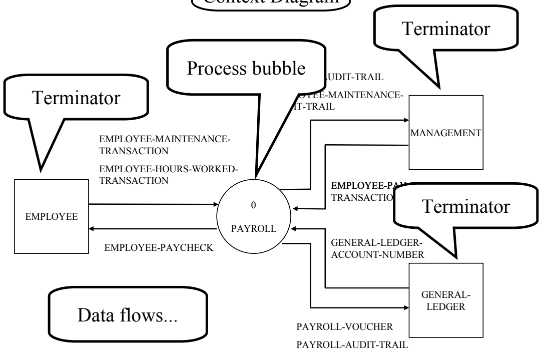
- Upper-most level, most abstract view of system.
- The "outside" view of the system.
- Shows a single process bubble, the net inputs and outputs of entire system, and the terminators with which they communicate.
- Purpose is to delineate the domain (scope) of the system.
- Sometimes called level 0 diagram

Context Diagram


PAYROLL-AUDIT-TRAIL
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
MANAGEMENT
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-HOURS-WORKED-TRANSACTION
EMPLOYEE
0
PAYROLL
EMPLOYEE-PAY-RATE-TRANSACTION
EMPLOYEE-PAYCHECK
GENERAL-LEDGER-ACCOUNT-NUMBER
GENERAL-LEDGER
PAYROLL-VOUCHER
PAYROLL-AUDIT-TRAIL

Clearly labeled "Context Diagram"

Context Diagram


Terminator

Process bubble

Terminator

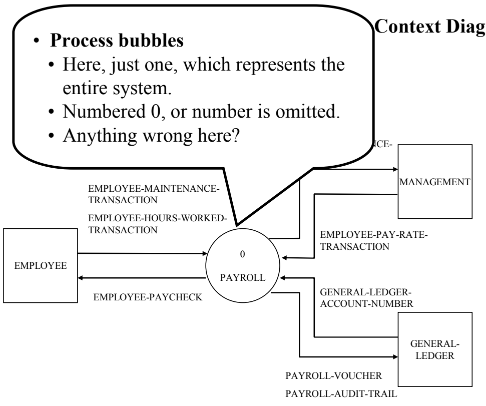
Terminator

Data flows...

Context Level DFD

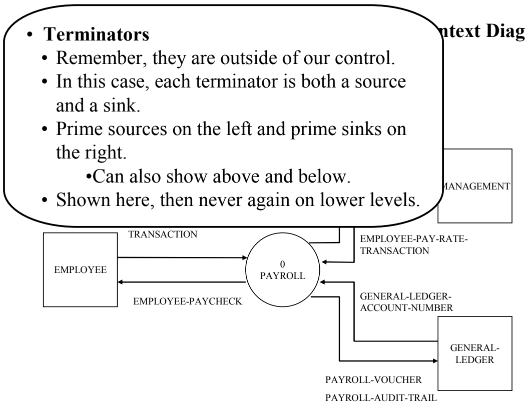
- We will go through each of the four components (bubble, data flow, data store, terminator) with each diagram.

Context Diagram


- **Process bubbles**
  - Here, just one, which represents the entire system.
  - Numbered 0, or number is omitted.
  - Anything wrong here?

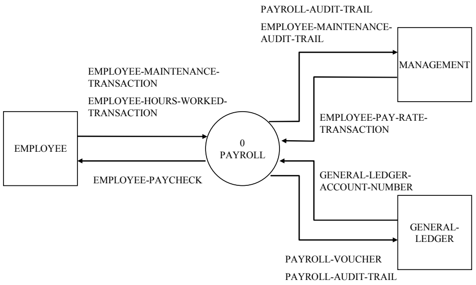
CE-
MANAGEMENT
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-HOURS-WORKED-TRANSACTION
EMPLOYEE
0
PAYROLL
EMPLOYEE-PAY-RATE-TRANSACTION
EMPLOYEE-PAYCHECK
GENERAL-LEDGER-ACCOUNT-NUMBER
GENERAL-LEDGER
PAYROLL-VOUCHER
PAYROLL-AUDIT-TRAIL

ntext Diagram


- **Terminators**
  - Remember, they are outside of our control.
  - In this case, each terminator is both a source and a sink.
  - Prime sources on the left and prime sinks on the right.
    - Can also show above and below.
  - Shown here, then never again on lower levels.

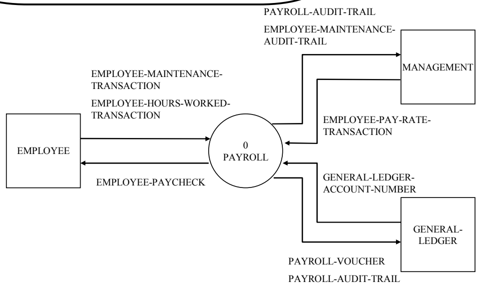
- **Data stores**
  - Internal to our system, so none on this level ever.

Context Diagram


PAYROLL-AUDIT-TRAIL
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
MANAGEMENT
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-HOURS-WORKED-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
EMPLOYEE
0
PAYROLL
EMPLOYEE-PAYCHECK
GENERAL-LEDGER-ACCOUNT-NUMBER
GENERAL-LEDGER
PAYROLL-VOUCHER
PAYROLL-AUDIT-TRAIL

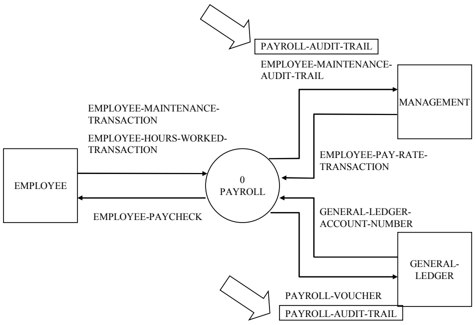
- **Data flows**
  - Long, descriptive, singular names.
  - A "packet" of logically related data.

Context Diagram


PAYROLL-AUDIT-TRAIL
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
MANAGEMENT
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-HOURS-WORKED-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
EMPLOYEE
0
PAYROLL
EMPLOYEE-PAYCHECK
GENERAL-LEDGER-ACCOUNT-NUMBER
GENERAL-LEDGER
PAYROLL-VOUCHER
PAYROLL-AUDIT-TRAIL

- **Anything wrong?**

Context Diagram


PAYROLL-AUDIT-TRAIL
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
MANAGEMENT
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-HOURS-WORKED-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
EMPLOYEE
0
PAYROLL
EMPLOYEE-PAYCHECK
GENERAL-LEDGER-ACCOUNT-NUMBER
GENERAL-LEDGER
PAYROLL-VOUCHER
PAYROLL-AUDIT-TRAIL

Context Level DFD

- Duplicate data flow names acceptable if two or more *identical* copies of the same item going to two or more destinations.
  - To show how the system relates to the world, we must show each copy.
  - On level below, treat as a single data flow.
    - Whether one or multiple copies is irrelevant except to outside world; we process the same regardless.

Leveling

- If a system is too large to be shown on a single diagram (aren't they all!), break into subsystems and sub-subsystems.
- Called *leveling* or top-down *partitioning*.
- Each partitioning (breaking up) of a bubble to a lower level is done to show more detail.
  - Called an *explosion* in engineering terminology.

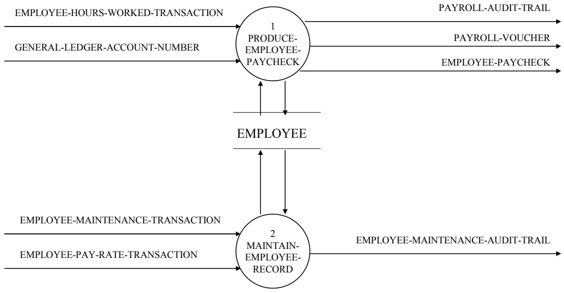
Leveling

- Parent/child relationship
  - A parent *bubble* can have a child *diagram*.
- How do we decide upon partitioning boundaries?
  - Use the same techniques as when partitioning programs into subroutines.

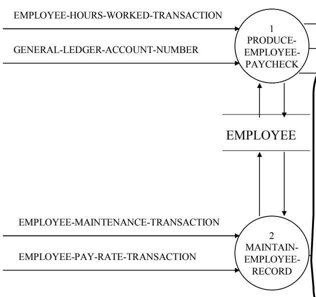
Overview/Level 1 Diagram

- Child of the single bubble on the Context Diagram.
- Shows ***major*** functions, ***major*** data stores and ***major*** data flows.

Overview / Level 1 Diagram


EMPLOYEE-HOURS-WORKED-TRANSACTION
GENERAL-LEDGER-ACCOUNT-NUMBER
1 PRODUCE-EMPLOYEE-PAYCHECK
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
EMPLOYEE-PAYCHECK
EMPLOYEE
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
2 MAINTAIN-EMPLOYEE-RECORD
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL

Overview Diagram


EMPLOYEE-HOURS-WORKED-TRANSACTION
GENERAL-LEDGER-ACCOUNT-NUMBER
1 PRODUCE-EMPLOYEE-PAYCHECK
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
EMPLOYEE
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
2 MAINTAIN-EMPLOYEE-RECORD

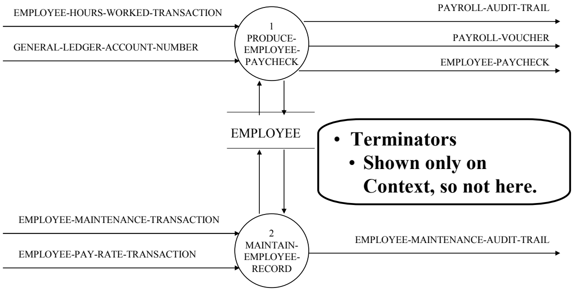
- **Process bubbles**
  - **Here, two major functions (bubbles).**
  - **May have up to seven bubbles on a diagram.**
  - **What about 1 bubble?**
  - **Numbered 1, 2, 3, etc.**
  - **Names have an active verb & object clause.**
  - **Avoid vague verbs like PROCESS.**

Overview Diagram

- Partition the Overview Diagram based on:
  - Different major functions.
    - Don't put trivial functions (like EDIT, FORMAT, WRITE, etc.) on Overview.
  - Different major inputs.
  - Different time frames.
  - Different equipment.

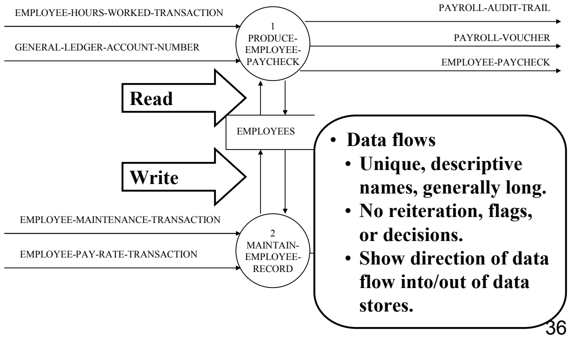
  - Note: know all four of these criteria for tests.

Overview Diagram


EMPLOYEE-HOURS-WORKED-TRANSACTION
GENERAL-LEDGER-ACCOUNT-NUMBER
1 PRODUCE-EMPLOYEE-PAYCHECK
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
EMPLOYEE-PAYCHECK
EMPLOYEE
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
2 MAINTAIN-EMPLOYEE-RECORD
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL

- **Terminators**
  - **Shown only on Context, so not here.**

Overview Diagram


EMPLOYEE-HOURS-WORKED-TRANSACTION
GENERAL-LEDGER-ACCOUNT-NUMBER
1 PRODUCE-EMPLOYEE-PAYCHECK
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
EMPLOYEE-PAYCHECK
Read
EMPLOYEES
Write
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
2 MAINTAIN-EMPLOYEE-RECORD

- **Data flows**
  - **Unique, descriptive names, generally long.**
  - **No reiteration, flags, or decisions.**
  - **Show direction of data flow into/out of data stores.**

Overview Diagram

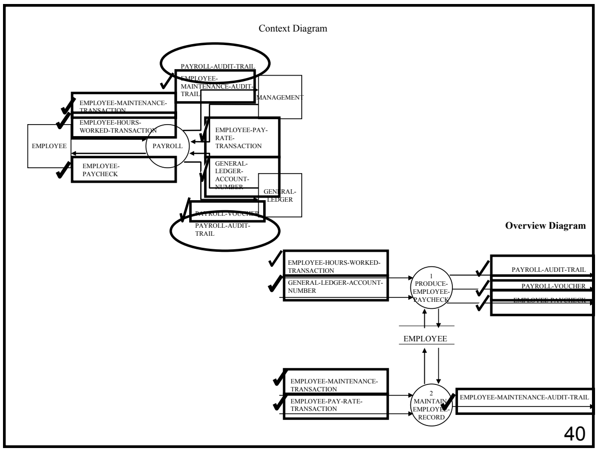
- No labels on data flows into and out of data stores when using the entire record.
  - *Always* need to use the entire record on a write, so writes are never labeled.
  - On reads, if using just one or two fields, then label as such.

Overview Diagram

- Placement of data flows
  - Try to move left to right, top to bottom if possible.
  - Inputs and outputs to edge of page.
  - Avoid line crossings by rearranging.

Balancing

- A child diagram is balanced with a parent bubble if there are the same net inputs and outputs to the entire child diagram that there are to the parent bubble.
- Balancing is the foundation for the entire DFD system.
- Let's check the balancing between the Context Diagram and the Overview Diagram...


Context Diagram
PAYROLL-AUDIT-TRAIL
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
MANAGEMENT
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-HOURS-WORKED-TRANSACTION
EMPLOYEE
PAYROLL
EMPLOYEE-PAY-RATE-TRANSACTION
EMPLOYEE-PAYCHECK
GENERAL-LEDGER-ACCOUNT-NUMBER
GENERAL-LEDGER
PAYROLL-VOUCHER
PAYROLL-AUDIT-TRAIL
Overview Diagram
EMPLOYEE-HOURS-WORKED-TRANSACTION
GENERAL-LEDGER-ACCOUNT-NUMBER
1 PRODUCE-EMPLOYEE-PAYCHECK
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
EMPLOYEE-PAYCHECK
EMPLOYEE
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
2 MAINTAIN-EMPLOYEE-RECORD
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL

Balancing

- 1st exception to balancing rule: multiple copies of same data flow don't violate balancing; they are logically the same data.
  - On context, there were two PAYROLL-AUDIT-TRAILs.
  - On lower level, treat logically and show just one copy.

Data Stores

- Data stores
  - Tricky rules governing where and when we create and show files.

Data Stores

- At what level do we show an *existing* file?
  - Show it for the first time at the highest level at which it is used by two or more bubbles.
  - Then show all references to it.
  - From then on, show it where it only when accessed.
  - 2nd exception to the balancing rule: data stores that are shown at lower levels but not on the higher levels.

Data Stores

- Never show a data store on the context diagram.
- What if used by only one bubble in entire system?
  - Show at the very lowest level only.

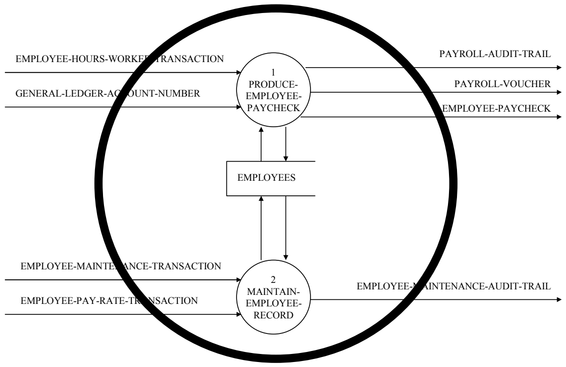
Data Stores

- When should you *create* a data store from scratch?
  - When data must be delayed for some period of time.
    - Example: collect transactions all day, then apply at night.
  - When data must be used in a different order.
    - Example: Data validation input files.
- A data store may be only interface between two or more bubbles.

Summary of the Overview Diagram

- If we draw a big circle around the Overview Diagram, bisecting the inputs and outputs, then collapse the circle...

Overview Diagram



EMPLOYEE-HOURS-WORKED-TRANSACTION

GENERAL-LEDGER-ACCOUNT-NUMBER

1 PRODUCE-EMPLOYEE-PAYCHECK

PAYROLL-AUDIT-TRAIL

PAYROLL-VOUCHER

EMPLOYEE-PAYCHECK

EMPLOYEES

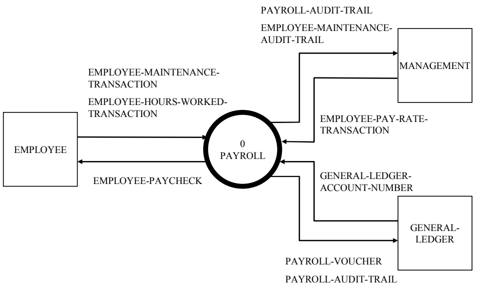
EMPLOYEE-MAINTENANCE-TRANSACTION

EMPLOYEE-PAY-RATE-TRANSACTION

2 MAINTAIN-EMPLOYEE-RECORD

EMPLOYEE-MAINTENANCE-AUDIT-TRAIL

Context Diagram



PAYROLL-AUDIT-TRAIL

EMPLOYEE-MAINTENANCE-AUDIT-TRAIL

MANAGEMENT

EMPLOYEE-MAINTENANCE-TRANSACTION

EMPLOYEE-HOURS-WORKED-TRANSACTION

EMPLOYEE

0 PAYROLL

EMPLOYEE-PAY-RATE-TRANSACTION

EMPLOYEE-PAYCHECK

GENERAL-LEDGER-ACCOUNT-NUMBER

GENERAL-LEDGER

PAYROLL-VOUCHER

PAYROLL-AUDIT-TRAIL

Diagram 1

- Child of bubble 1 on Overview.
- Diagram numbering: bubble 1 explodes to Diagram 1.

Diagram 1

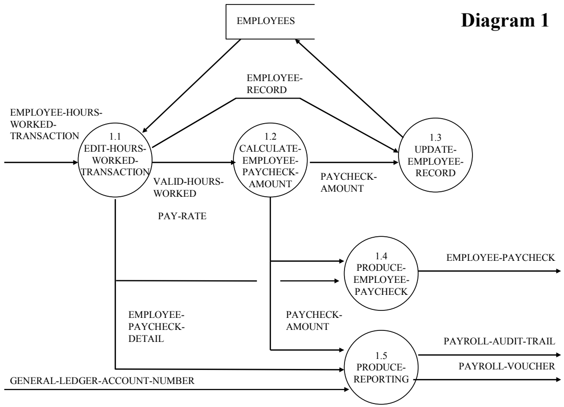
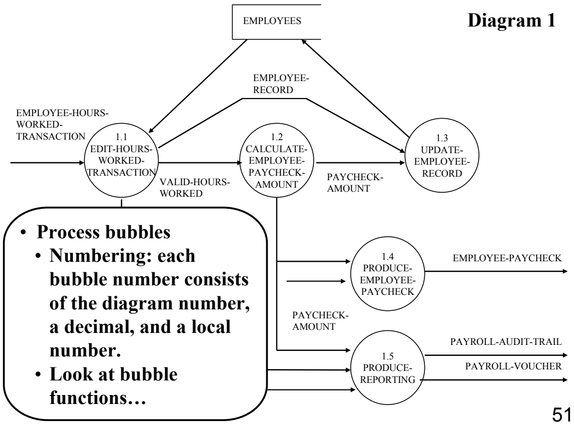

EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAYCHECK-AMOUNT
PAY-RATE
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
PAYROLL-AUDIT-TRAIL
1.5 PRODUCE-REPORTING
PAYROLL-VOUCHER
GENERAL-LEDGER-ACCOUNT-NUMBER

Diagram 1


EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAYCHECK-AMOUNT
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
PAYCHECK-AMOUNT
PAYROLL-AUDIT-TRAIL
1.5 PRODUCE-REPORTING
PAYROLL-VOUCHER

- **Process bubbles**
  - **Numbering: each bubble number consists of the diagram number, a decimal, and a local number.**
  - **Look at bubble functions…**

Diagram 1

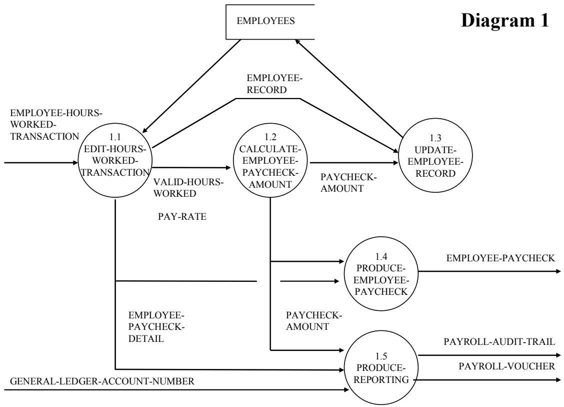
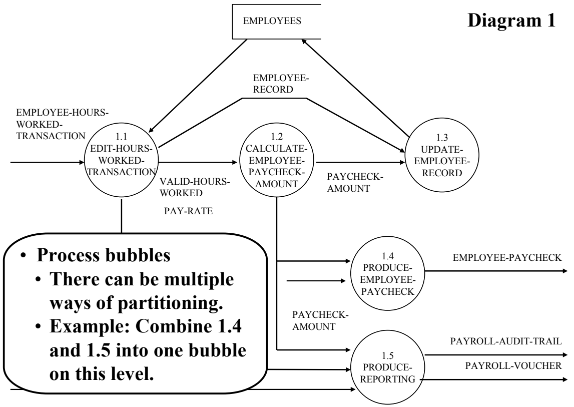
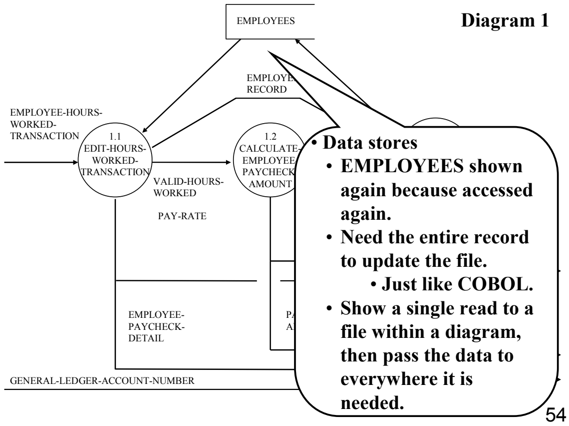

EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAYCHECK-AMOUNT
PAY-RATE
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
PAYROLL-AUDIT-TRAIL
1.5 PRODUCE-REPORTING
PAYROLL-VOUCHER
GENERAL-LEDGER-ACCOUNT-NUMBER

Diagram 1


EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAYCHECK-AMOUNT
PAY-RATE
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
PAYCHECK-AMOUNT
PAYROLL-AUDIT-TRAIL
1.5 PRODUCE-REPORTING
PAYROLL-VOUCHER

- **Process bubbles**
  - **There can be multiple ways of partitioning.**
  - **Example: Combine 1.4 and 1.5 into one bubble on this level.**

Diagram 1


EMPLOYEES
EMPLOYE RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE PAYCHECK AMOUNT
VALID-HOURS-WORKED
PAY-RATE
EMPLOYEE-PAYCHECK-DETAIL
GENERAL-LEDGER-ACCOUNT-NUMBER

- **Data stores**
  - **EMPLOYEES shown again because accessed again.**
  - **Need the entire record to update the file.**
    - **Just like COBOL.**
  - **Show a single read to a file within a diagram, then pass the data to everywhere it is needed.**

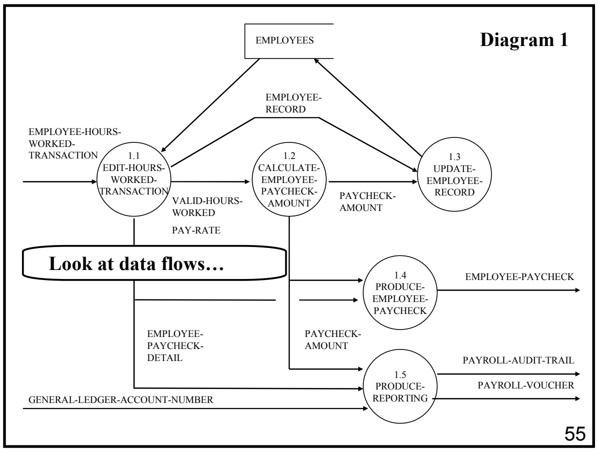


Diagram 1

Look at data flows…

EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAY-RATE
PAYCHECK-AMOUNT
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
1.5 PRODUCE-REPORTING
GENERAL-LEDGER-ACCOUNT-NUMBER

55

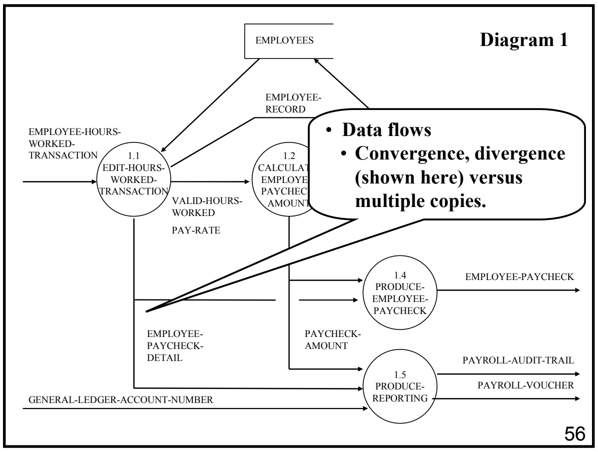


Diagram 1

- Data flows
  - Convergence, divergence (shown here) versus multiple copies.

56

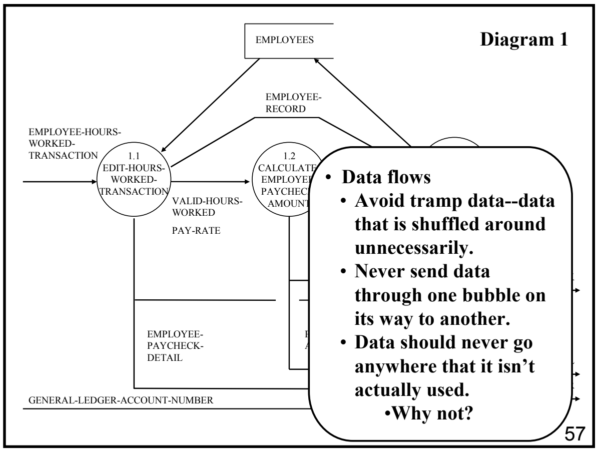


Diagram 1

- Data flows
  - Avoid tramp data--data that is shuffled around unnecessarily.
  - Never send data through one bubble on its way to another.
  - Data should never go anywhere that it isn't actually used.
    - Why not?

EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE EMPLOYEE PAYCHEC AMOUNT
VALID-HOURS-WORKED
PAY-RATE
EMPLOYEE-PAYCHECK-DETAIL
GENERAL-LEDGER-ACCOUNT-NUMBER

57

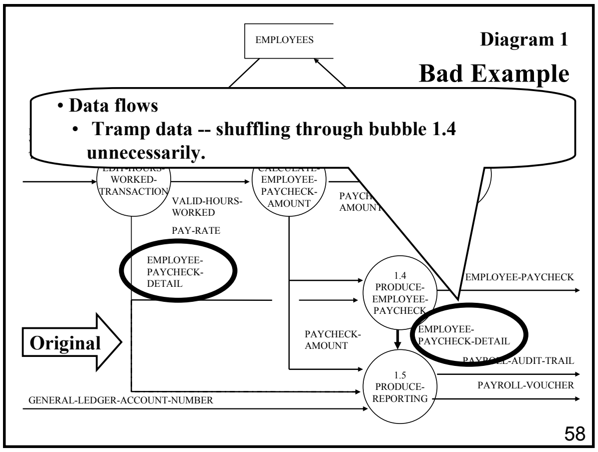


Diagram 1

Bad Example

- Data flows
  - Tramp data -- shuffling through bubble 1.4 unnecessarily.

58

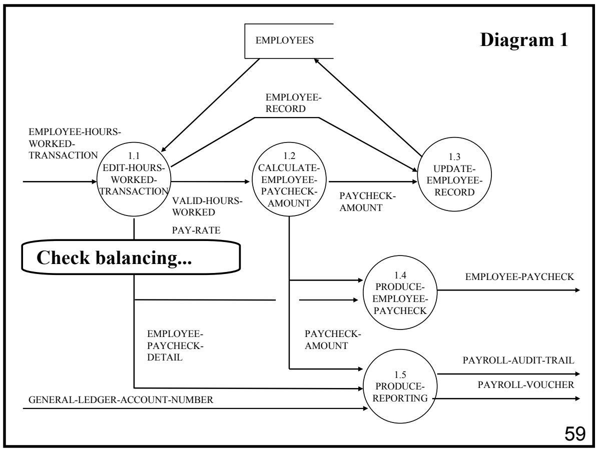
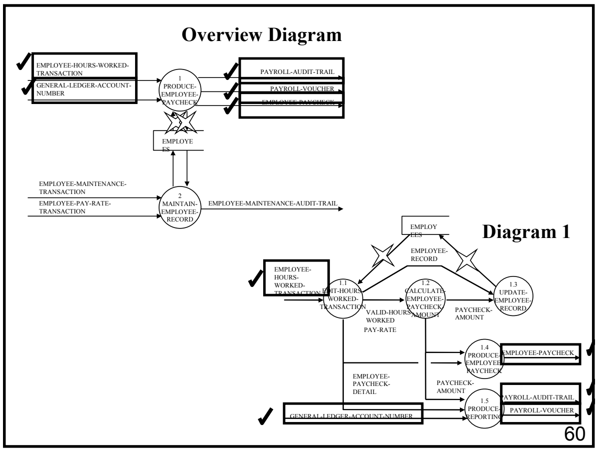


Diagram 1

Check balancing...

EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAY-RATE
PAYCHECK-AMOUNT
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
1.5 PRODUCE-REPORTING
GENERAL-LEDGER-ACCOUNT-NUMBER

59



Overview Diagram

Diagram 1

✓ EMPLOYEE-HOURS-WORKED-TRANSACTION
✓ GENERAL-LEDGER-ACCOUNT-NUMBER
1 PRODUCE-EMPLOYEE-PAYCHECK
✓ PAYROLL-AUDIT-TRAIL
✓ PAYROLL-VOUCHER
✓ EMPLOYEE-PAYCHECK
EMPLOYEES
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
2 MAINTAIN-EMPLOYEE-RECORD
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
EMPLOYEES
EMPLOYEE-RECORD
✓ EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAY-RATE
PAYCHECK-AMOUNT
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK ✓
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
1.5 PRODUCE-REPORTING
PAYROLL-AUDIT-TRAIL ✓
PAYROLL-VOUCHER ✓
✓ GENERAL-LEDGER-ACCOUNT-NUMBER

60

Diagram 1

- Data flows
  - An edit transforms data, so the name must change to reflect that.
  - Name by the last transformation.

Diagram 1.1

- Child of bubble 1.1 on Diagram 1.

Diagram 1.1

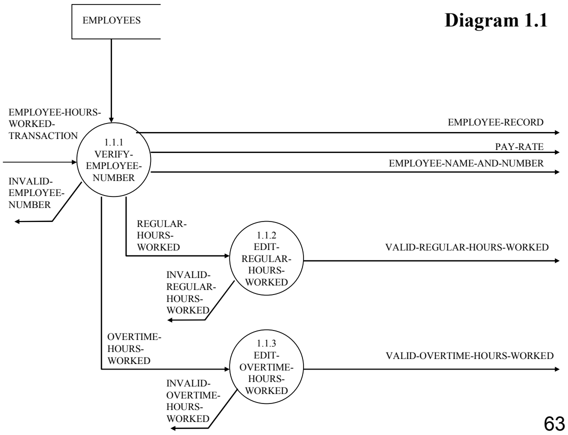

EMPLOYEES
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1.1 VERIFY-EMPLOYEE-NUMBER
EMPLOYEE-RECORD
PAY-RATE
EMPLOYEE-NAME-AND-NUMBER
INVALID-EMPLOYEE-NUMBER
REGULAR-HOURS-WORKED
1.1.2 EDIT-REGULAR-HOURS-WORKED
VALID-REGULAR-HOURS-WORKED
INVALID-REGULAR-HOURS-WORKED
OVERTIME-HOURS-WORKED
1.1.3 EDIT-OVERTIME-HOURS-WORKED
VALID-OVERTIME-HOURS-WORKED
INVALID-OVERTIME-HOURS-WORKED

Diagram 1.1

- ***Functional primitive***
  - A process that is not further decomposed into a lower level diagram. It performs only a single task.
    - Bubbles 1.1.2, 1.1.2, and 1.1.3 are functional primitives.
  - Choosing what should be a functional primitive is rather arbitrary and uses circular reasoning.
    - How do you know when you have a functional primitive? When you stop partitioning.
    - When do you stop partitioning? When you have a functional primitive.

Diagram 1.1

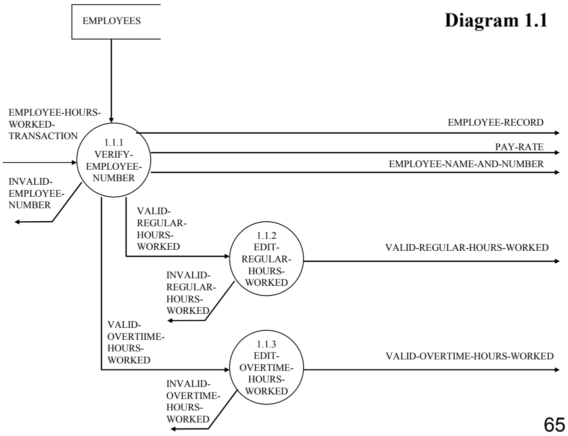

EMPLOYEES
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1.1 VERIFY-EMPLOYEE-NUMBER
EMPLOYEE-RECORD
PAY-RATE
EMPLOYEE-NAME-AND-NUMBER
INVALID-EMPLOYEE-NUMBER
VALID-REGULAR-HOURS-WORKED
1.1.2 EDIT-REGULAR-HOURS-WORKED
VALID-REGULAR-HOURS-WORKED
INVALID-REGULAR-HOURS-WORKED
VALID-OVERTIIME-HOURS-WORKED
1.1.3 EDIT-OVERTIME-HOURS-WORKED
VALID-OVERTIME-HOURS-WORKED
INVALID-OVERTIME-HOURS-WORKED

Diagram 1.1

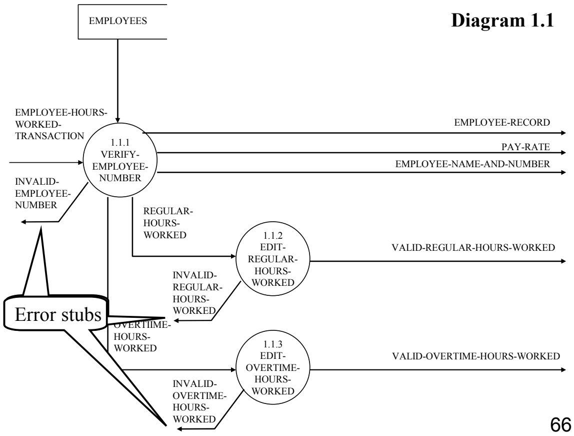

EMPLOYEES
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1.1 VERIFY-EMPLOYEE-NUMBER
EMPLOYEE-RECORD
PAY-RATE
EMPLOYEE-NAME-AND-NUMBER
INVALID-EMPLOYEE-NUMBER
REGULAR-HOURS-WORKED
1.1.2 EDIT-REGULAR-HOURS-WORKED
VALID-REGULAR-HOURS-WORKED
INVALID-REGULAR-HOURS-WORKED
Error stubs
OVERTIIME-HOURS-WORKED
1.1.3 EDIT-OVERTIME-HOURS-WORKED
VALID-OVERTIME-HOURS-WORKED
INVALID-OVERTIME-HOURS-WORKED

Diagram 1.1

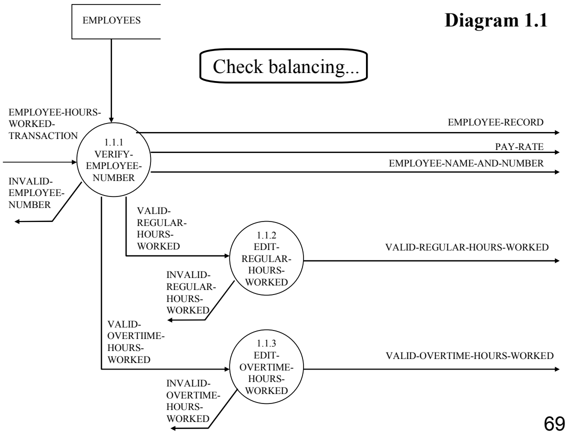
- Error stub--a note that an error condition must be handled, with no details on how to handle.
- Used only for *trivial errors*, errors that haven't yet made it into a file so they don't need undoing.

Diagram 1.1

- Error stubs shown only on functional primitives.
  - Don't want to clutter higher level diagrams with such trivial details.
- Name the error stub by the field in error.
- 3rd balancing exception, since they are shown on lower levels but not on the higher ones.

Diagram 1.1

Check balancing...


EMPLOYEES
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1.1 VERIFY-EMPLOYEE-NUMBER
EMPLOYEE-RECORD
PAY-RATE
EMPLOYEE-NAME-AND-NUMBER
INVALID-EMPLOYEE-NUMBER
VALID-REGULAR-HOURS-WORKED
1.1.2 EDIT-REGULAR-HOURS-WORKED
VALID-REGULAR-HOURS-WORKED
INVALID-REGULAR-HOURS-WORKED
VALID-OVERTIIME-HOURS-WORKED
1.1.3 EDIT-OVERTIME-HOURS-WORKED
VALID-OVERTIME-HOURS-WORKED
INVALID-OVERTIME-HOURS-WORKED

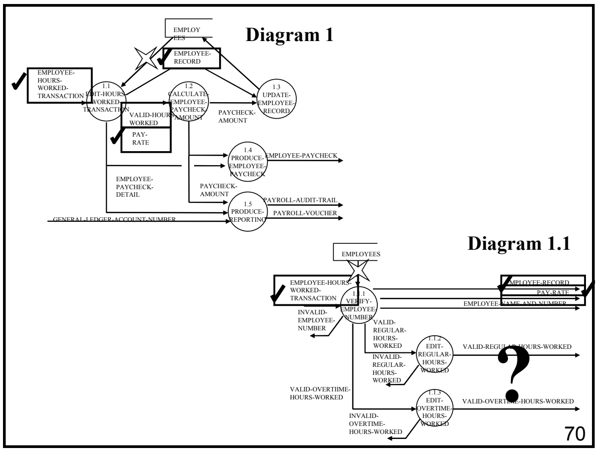


Diagram 1

Diagram 1.1

EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAY-RATE
PAYCHECK-AMOUNT
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
1.5 PRODUCE-REPORTING
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
GENERAL-LEDGER-ACCOUNT-NUMBER
EMPLOYEES
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1.1 VERIFY-EMPLOYEE-NUMBER
EMPLOYEE-RECORD
PAY-RATE
EMPLOYEE-NAME-AND-NUMBER
INVALID-EMPLOYEE-NUMBER
VALID-REGULAR-HOURS-WORKED
1.1.2 EDIT-REGULAR-HOURS-WORKED
VALID-REGULAR-HOURS-WORKED
INVALID-REGULAR-HOURS-WORKED
VALID-OVERTIIME-HOURS-WORKED
1.1.3 EDIT-OVERTIME-HOURS-WORKED
VALID-OVERTIME-HOURS-WORKED
INVALID-OVERTIME-HOURS-WORKED
?

Diagram 1.1

- VALID-HOURS-WORKED doesn't match...
- *Parallel decomposition*--one arrow on parent may become several arrows on the child diagram.

Diagram 1.1

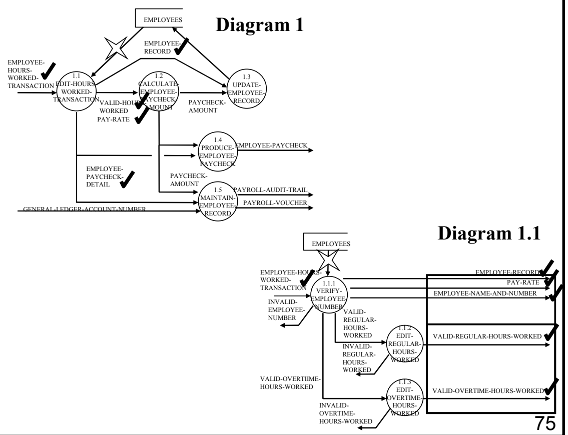
- The group data flow is broken into its *pieces* or *choices*.
  - Example: PAYMENT-TRANSACTION is broken into its ***pieces*** of CUSTOMER-NUMBER and PAYMENT-AMOUNT, each going a different direction.
  - Example: UPDATE-TRANSACTION is broken into its ***choices*** of ADD-TRANSACTION, ALTER-TRANSACTION, DELETE-TRANSACTION, each one going a different direction.

Diagram 1.1

- The multiple arrows on the child are equivalent to the single data flow on the parent.
- Disadvantage--Makes the diagram harder to read. Any alternatives?
- Evaluate each situation and use only when necessary.
- 4th balancing exception.

Diagram 1.1

- So here, VALID-HOURS-WORKED-TRANSACTION breaks down into its pieces of VALID-REGULAR-HOURS-WORKED and VALID-OVERTIME-HOURS-WORKED.


EMPLOYEES
Diagram 1
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
VALID-HOURS-WORKED
PAY-RATE
PAYCHECK-AMOUNT
1.4 PRODUCE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
1.5 MAINTAIN-EMPLOYEE-RECORD
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
GENERAL-LEDGER-ACCOUNT-NUMBER
Diagram 1.1
EMPLOYEES
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1.1 VERIFY-EMPLOYEE-NUMBER
INVALID-EMPLOYEE-NUMBER
EMPLOYEE-RECORD
PAY-RATE
EMPLOYEE-NAME-AND-NUMBER
VALID-REGULAR-HOURS-WORKED
1.1.2 EDIT-REGULAR-HOURS-WORKED
INVALID-REGULAR-HOURS-WORKED
VALID-REGULAR-HOURS-WORKED
VALID-OVERTIIME-HOURS-WORKED
1.1.3 EDIT-OVERTIME-HOURS-WORKED
INVALID-OVERTIME-HOURS-WORKED
VALID-OVERTIME-HOURS-WORKED

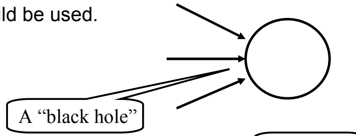
Summary of Balancing Exceptions

- Multiple copies of same item.
- Data stores not shown on higher levels.
- Error stubs.
- Parallel decomposition.

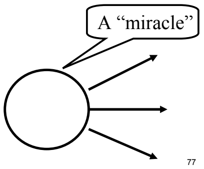
- Note: Convergence and divergence are *not* balancing exceptions, because they are internal to the diagram.

Data Conservation

- Data that goes into a bubble should be used.

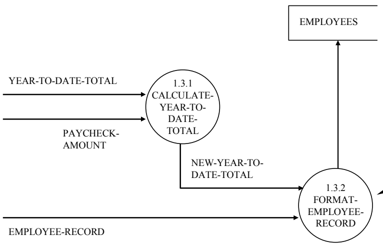


- Data can't be created from scratch.
  - Exception: current date and time.



Remaining Diagrams...

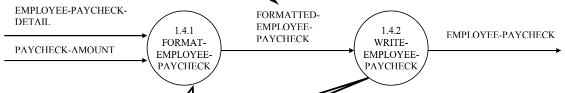
Diagram 1.3



Always need a FORMAT when writing to files.

Diagram 1.4

Secondary alias for EMPLOYEE-PAYCHECK


EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
1.4.1 FORMAT-EMPLOYEE-PAYCHECK
FORMATTED-EMPLOYEE-PAYCHECK
1.4.2 WRITE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK

Always need a FORMAT *and* a WRITE when writing to a printer.

Diagram 1.4

- Avoid diagrams with only two bubbles.
  - Haven’t really partitioned much.
  - Makes set of DFDs bulkier and harder to read.
- How would we avoid here?

Diagram 1.4

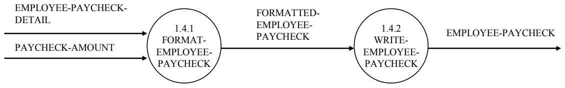

EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
1.4.1 FORMAT-EMPLOYEE-PAYCHECK
FORMATTED-EMPLOYEE-PAYCHECK
1.4.2 WRITE-EMPLOYEE-PAYCHECK
EMPLOYEE-PAYCHECK

Diagram 1

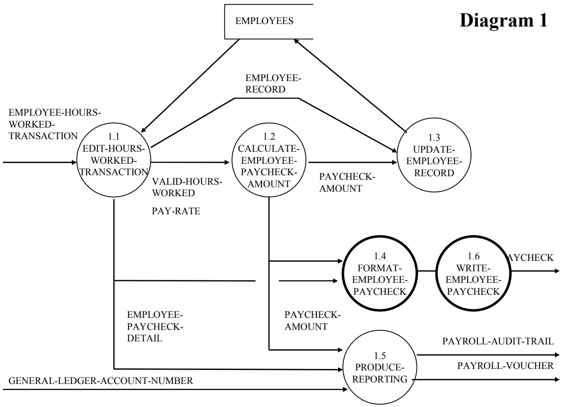

EMPLOYEES
EMPLOYEE-RECORD
EMPLOYEE-HOURS-WORKED-TRANSACTION
1.1 EDIT-HOURS-WORKED-TRANSACTION
VALID-HOURS-WORKED
PAY-RATE
1.2 CALCULATE-EMPLOYEE-PAYCHECK-AMOUNT
PAYCHECK-AMOUNT
1.3 UPDATE-EMPLOYEE-RECORD
1.4 FORMAT-EMPLOYEE-PAYCHECK
1.6 WRITE-EMPLOYEE-PAYCHECK
AYCHECK
EMPLOYEE-PAYCHECK-DETAIL
PAYCHECK-AMOUNT
1.5 PRODUCE-REPORTING
PAYROLL-AUDIT-TRAIL
PAYROLL-VOUCHER
GENERAL-LEDGER-ACCOUNT-NUMBER

Diagram 1.5

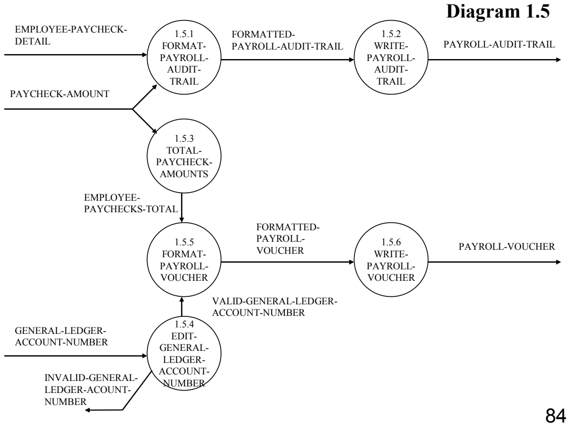
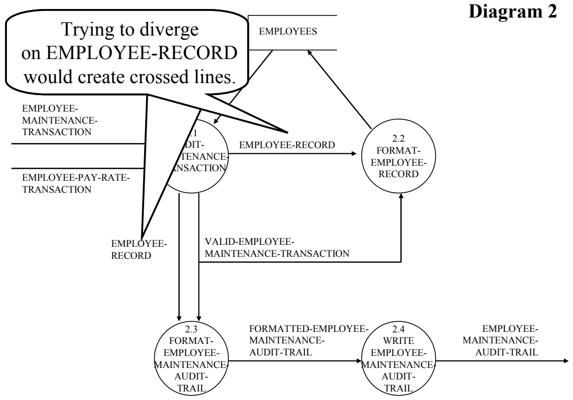
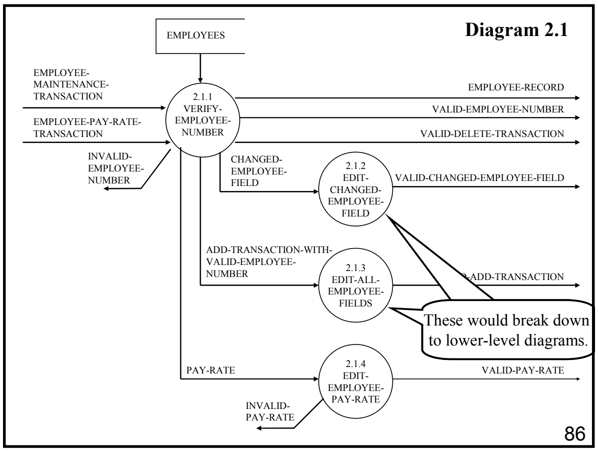

EMPLOYEE-PAYCHECK-DETAIL
1.5.1 FORMAT-PAYROLL-AUDIT-TRAIL
FORMATTED-PAYROLL-AUDIT-TRAIL
1.5.2 WRITE-PAYROLL-AUDIT-TRAIL
PAYROLL-AUDIT-TRAIL
PAYCHECK-AMOUNT
1.5.3 TOTAL-PAYCHECK-AMOUNTS
EMPLOYEE-PAYCHECKS-TOTAL
1.5.5 FORMAT-PAYROLL-VOUCHER
FORMATTED-PAYROLL-VOUCHER
1.5.6 WRITE-PAYROLL-VOUCHER
PAYROLL-VOUCHER
VALID-GENERAL-LEDGER-ACCOUNT-NUMBER
GENERAL-LEDGER-ACCOUNT-NUMBER
1.5.4 EDIT-GENERAL-LEDGER-ACCOUNT-NUMBER
INVALID-GENERAL-LEDGER-ACOUNT-NUMBER

Diagram 2


EMPLOYEES
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
.1 EDIT-…TENANCE-…NSACTION
EMPLOYEE-RECORD
2.2 FORMAT-EMPLOYEE-RECORD
EMPLOYEE-RECORD
VALID-EMPLOYEE-MAINTENANCE-TRANSACTION
2.3 FORMAT-EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
FORMATTED-EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
2.4 WRITE EMPLOYEE-MAINTENANCE-AUDIT-TRAIL
EMPLOYEE-MAINTENANCE-AUDIT-TRAIL

Trying to diverge on EMPLOYEE-RECORD would create crossed lines.

Diagram 2.1


EMPLOYEES
EMPLOYEE-MAINTENANCE-TRANSACTION
EMPLOYEE-PAY-RATE-TRANSACTION
2.1.1 VERIFY-EMPLOYEE-NUMBER
EMPLOYEE-RECORD
VALID-EMPLOYEE-NUMBER
VALID-DELETE-TRANSACTION
INVALID-EMPLOYEE-NUMBER
CHANGED-EMPLOYEE-FIELD
2.1.2 EDIT-CHANGED-EMPLOYEE-FIELD
VALID-CHANGED-EMPLOYEE-FIELD
ADD-TRANSACTION-WITH-VALID-EMPLOYEE-NUMBER
2.1.3 EDIT-ALL-EMPLOYEE-FIELDS
…-ADD-TRANSACTION
PAY-RATE
2.1.4 EDIT-EMPLOYEE-PAY-RATE
VALID-PAY-RATE
INVALID-PAY-RATE

These would break down to lower-level diagrams.

How do we know when to stop exploding?

- Partition to tiny.
- Each bubble documented by 1/2 page or less (usually).
- Each bubble performs a single, indivisible function.

Clues that we haven't partitioned far enough

- A process difficult to name.
- A single process has many inputs and/or many outputs.

Creating a DFD

- Identify terminators and their data flows, and use them to create a Context Diagram.
- Repeat until system completely partitioned to functional primitives:
  - Do first draft of a single diagram, with processes and data flows.
  - Do several more drafts of the diagram.
  - Draw last version neatly.

Creating a DFD

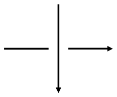
- Redraw all diagrams for clarity, incorporating any changes.
- Walk through the diagrams with the project team. Return to prior steps if problems are encountered.
- Walk through with the users. Return to prior steps if problems are encountered.

Miscellaneous

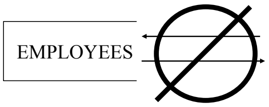
- Show data movement, not physical movement of goods.
- Use a CASE tool, a graphics package, or, if hand-drawn, use pencil, not ink.

Miscellaneous

- Minimize crossed lines. When necessary, show as follows:

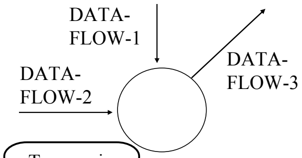


- *Don't* connect data flows to right side of a data store:

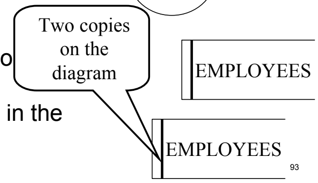


Miscellaneous

- Always write horizontally:



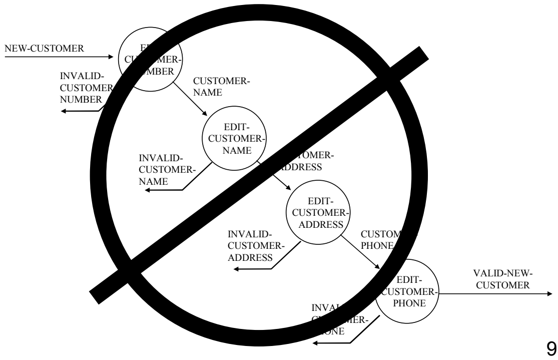
- Indicate multiple co[pies] of a data store by placing extra lines in the symbol:



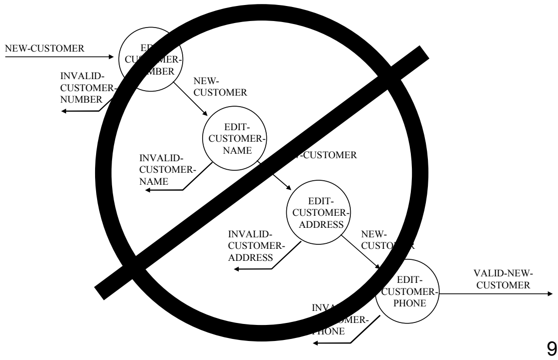
Editing Patterns

- Different ways to edit potentially dirty data from the outside world...

Editing Patterns

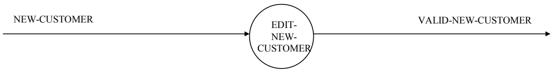


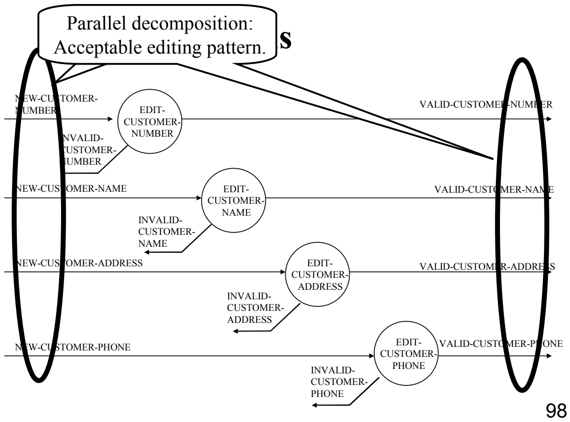
Editing Patterns

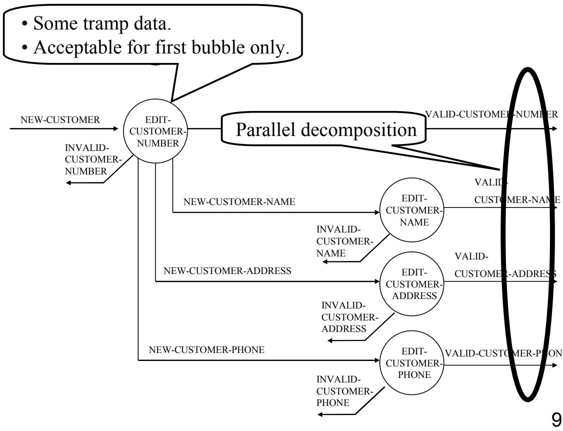


Editing Patterns

Acceptable editing pattern:
Showing upper level bubble...




Parallel decomposition:
Acceptable editing pattern.


- Some tramp data.
- Acceptable for first bubble only.

Parallel decomposition

Possible Signs of Errors

- The diagram is entwined, choked with data flows.
- Some place cries out for a flag.
- Flows or processes don't lend themselves to good names.
- There is a wide discrepancy in leveling.

Possible Signs of Errors

- The diagram shows:
  - data composition, access methods to data (data dictionary).
  - decisions, loops, insides of bubbles (process descriptions).
- The diagram makes you uneasy.