

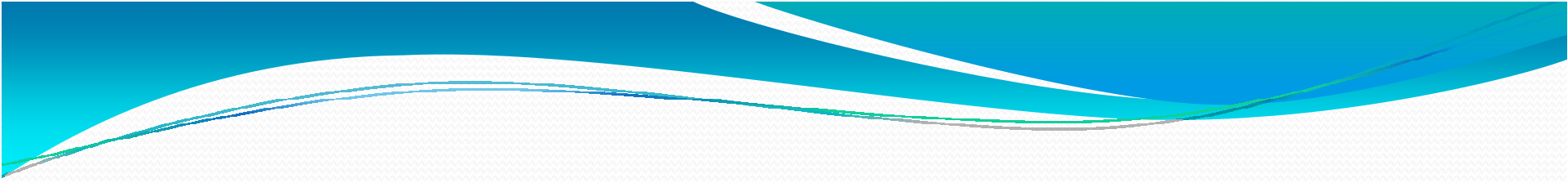
Data Flow Diagrams

Daryle Niedermayer, I.S.P., ITCP, PMP



Imagine...

- You bake a batch of cookies. They smell great; you take them out of the oven and then realize that you forgot to add sugar.
- You're inviting the guys over to watch the big game on your 52" screen. Everyone shows up and you realize you forgot to pick up the beer.
- You and your friend wait for the instructor to hand back your project plans. Yours never comes; you look at each other as you both say at the same time, "I thought you were going to hand in our project!"

- 
- Your partner asks you to pick up Tylenol on your way home after work. You stop at the store, grab a cart, pick up a few other things you remember you need and drive home. Only then do you realize that the one thing you forgot was the milk.



Missed Steps

- Sometimes these events are just embarrassing.
- Sometimes they are worse:
 - The project plan was worth 50% of your mark
 - The cookies were for your child's school bake sale
 - Your boss was going to come to watch the game and you're in line for a promotion
 - The Tylenol is for your sick child who is now going to cry through the night.



In System Design, Missed Steps...

- Can be very expensive.



How do we prevent mistakes?

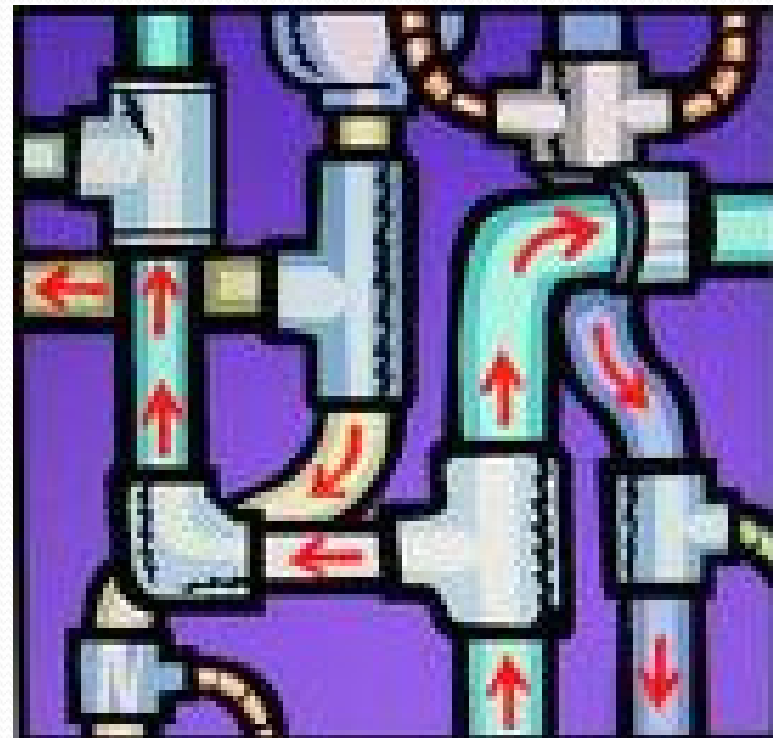
Data Flow Models



Data Flow Models

- **Data Flow Models** are a tool to map functional requirements into actual implementation details.
- As the name implies, an Information Management System normally:
 - receives input or data;
 - processes it;
 - and then either stores the results or passes it along to another system as output data.
- By looking at the flows of data, we can think of an Information System like a plumbing system

- As long as all inflows go somewhere, we won't have **water** on the floor!
- As long as all outflows go somewhere we won't have **other stuff** on the floor!
- So we just need to keep a list of our inflows and outflows.





How to Draw a Data Flow Diagram

- We document our Data Flow Models using Data Flow Diagrams
- There are two standards for DFDs:
 - Gane & Sarson
 - Yourdon & Coad
- Microsoft Visio uses Gane and Sarson; we will too.



4 Types of Objects in a DFD

- **Processes:** Processes transform incoming data into outgoing data
- **Datastores:** Datastores are repositories of data in the system such as files or databases
- **Dataflows:** Dataflows are the “pipelines” through which packets of information flow. They have labels and direction.
- **External Entities:** External entities are objects outside the system. They can be people (actors like customers, or managers) or other data systems or companies.



1. A Process

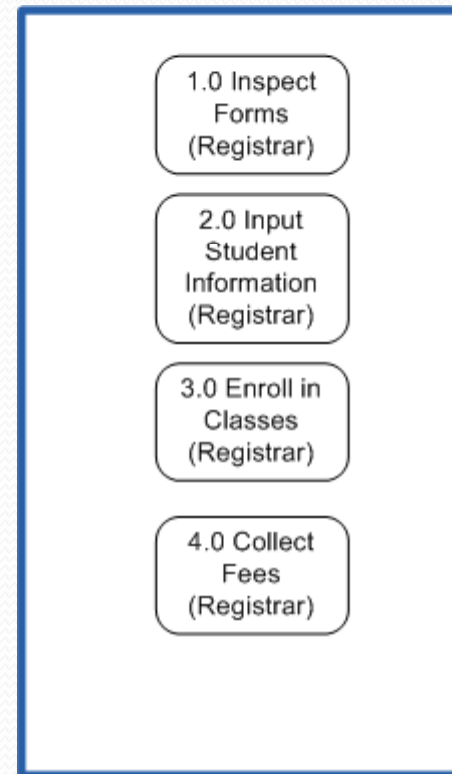
D2. A
Datastore

A Dataflow

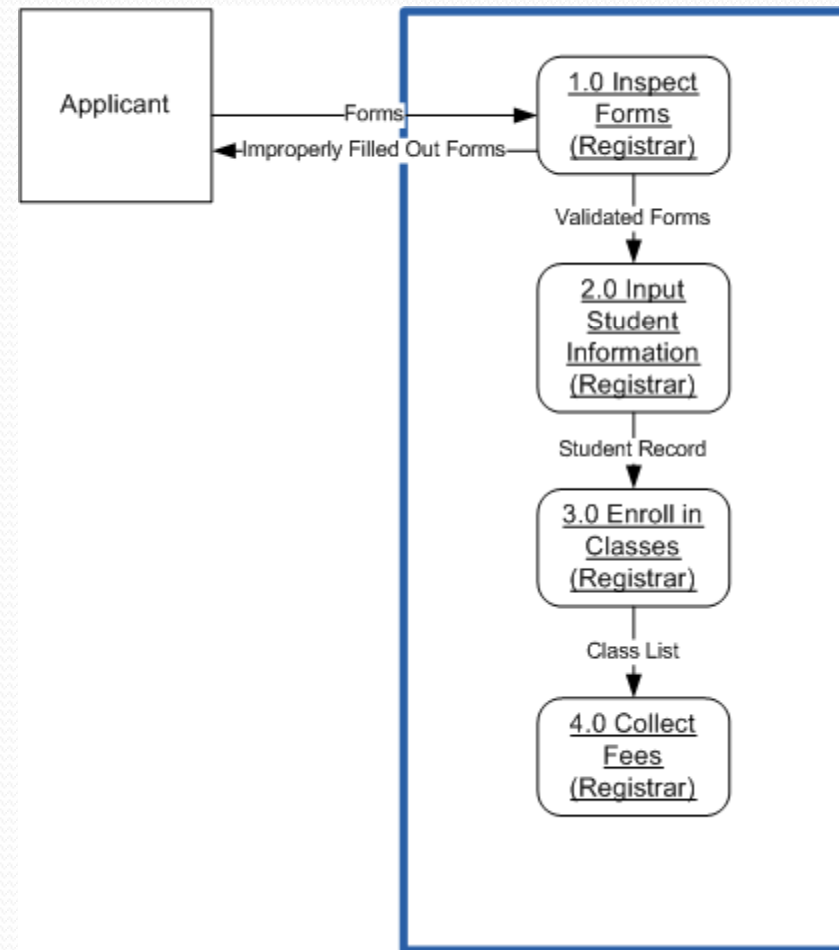
An External
Entity

How to Get Started with a DFD

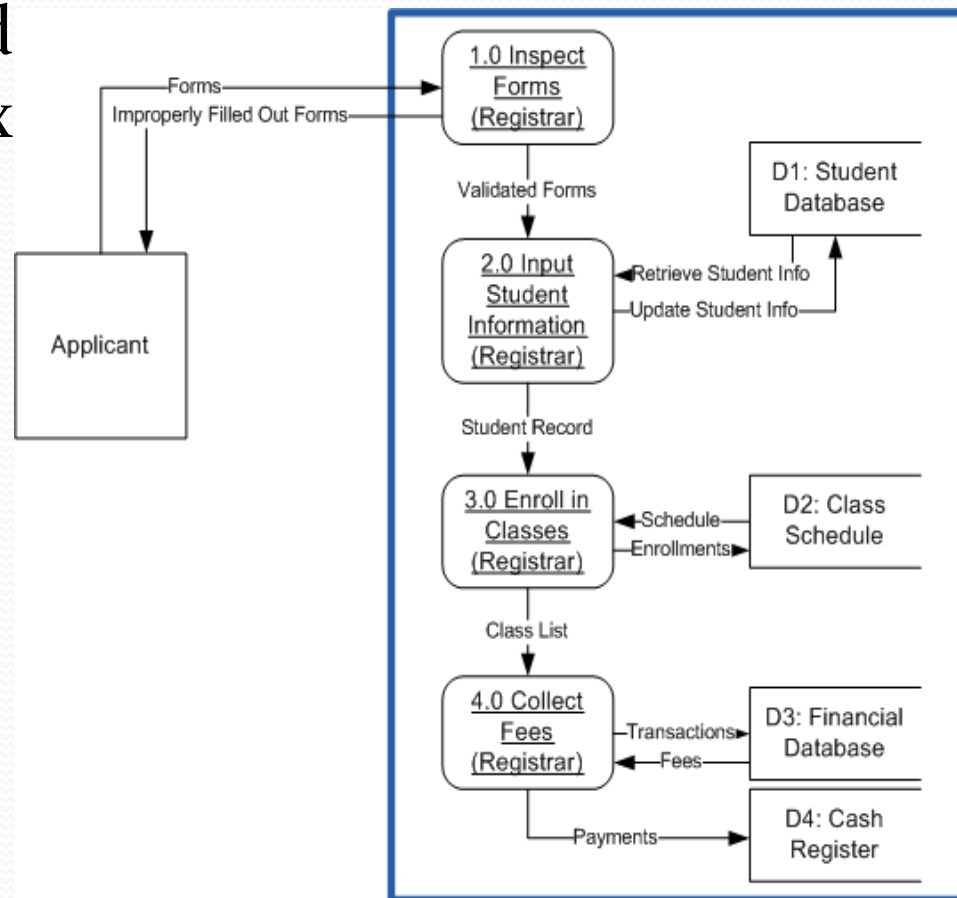
- Draw an empty box. This will represent the new system.
- Inside of this box, draw a process icon for each function the new system will contain. Number it with an whole number.
- Each Use Case Diagrams should be a process.



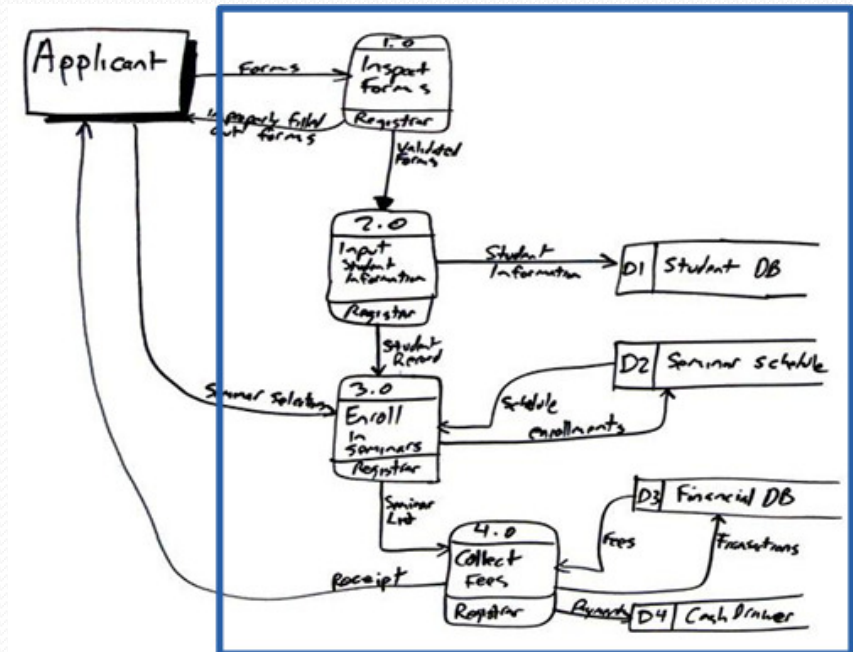
- Add in the External Entities
- Add in the Dataflows between processes and entities.



- If data is not consumed immediately by the next process, or outputted, then it needs to be put into a Datastore.
- Interviews with the client should answer this.
- Congratulations, you have a Level 0 DFD




- DFDs don't need to be pretty.
- You can develop them in a JAD session on a white board.





DFD Rules

- There are some rules to follow in creating proper DFDs:
 1. All processes must have at least one inflow and one outflow. If it doesn't have an inflow and an outflow it isn't a process (remember the plumbing example).
 2. All processes must modify data flowing through them in some way. If they don't they aren't a process
 3. Each datastore must be linked with at least one process. If it doesn't it is outside your system and you don't need to worry about it.

- 
4. Each external entity must be involved with at least one process. If it doesn't it isn't a concern to your system
 5. A dataflow must be attached to at least one process. Again, remember the plumbing example.
 6. Each datastore must be linked with at least one process. If it doesn't it is outside your system and you don't need to worry about it.
 7. Datastores have no intelligence. They can't process or manage data; they only store it. If you want to do anything with the data in a datastore, you need a process.



I Have my Level 0 DFD, Now What?

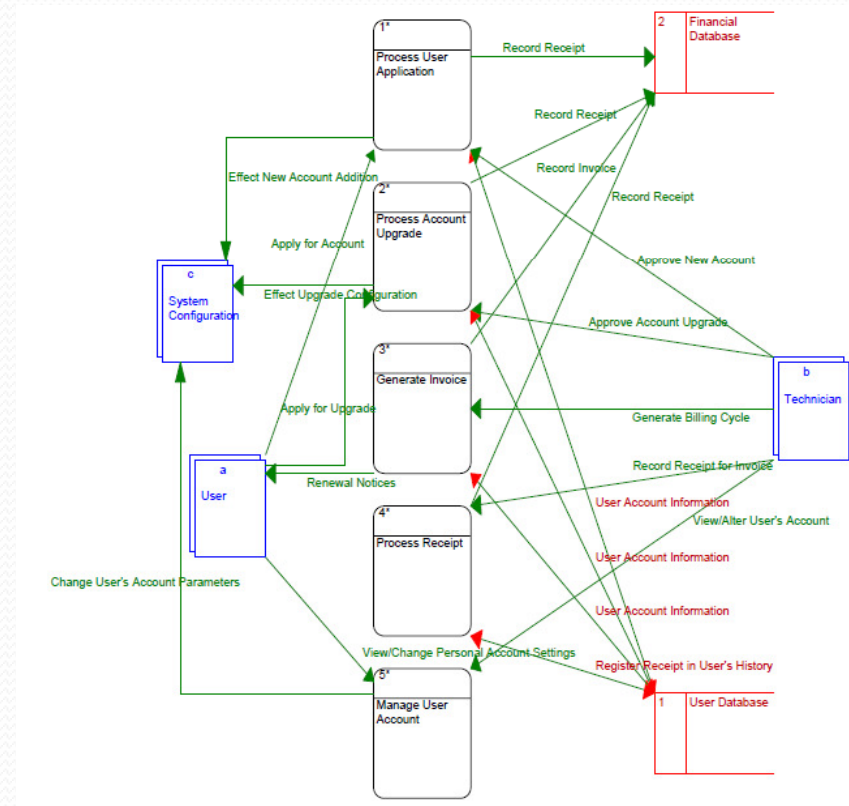
- A high level DFD still doesn't tell us how to build a new system.
- Once we have a DFD of our entire system, we use a process of “Successive Elaboration” to decompose it into smaller concepts. Sometimes we call this **granularity**.
 - A beach is just a beach until you get out your magnifying glass. Then you have granularity.



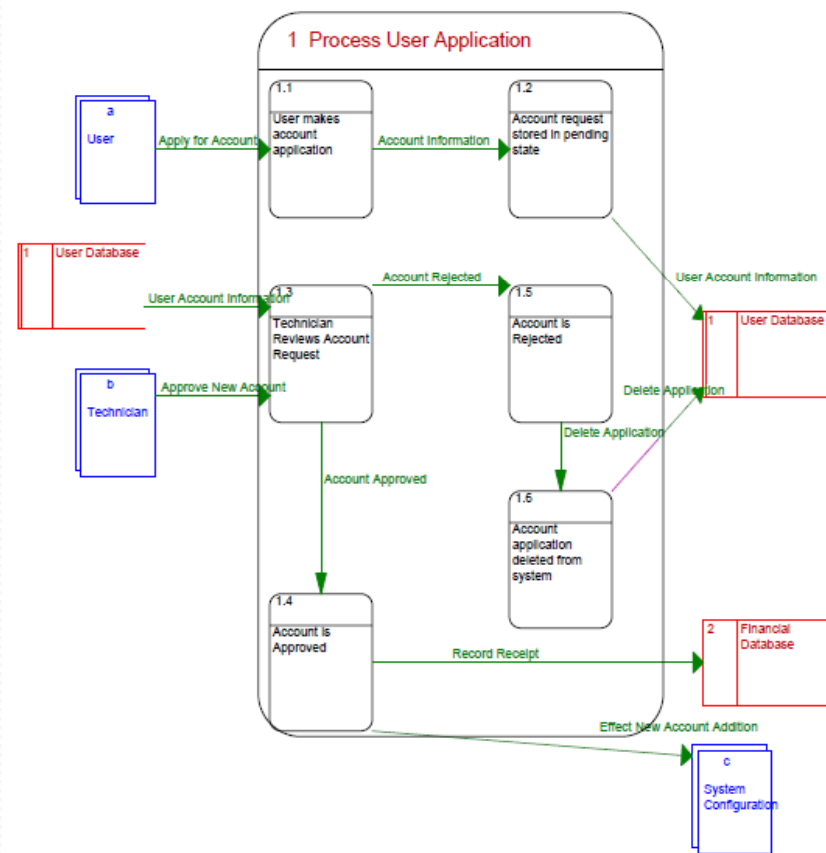
Let's Get Granular

- We can take each process and “zoom in” on it until we get to a level where we can understand how to build the individual processes and pieces.
- DFD's support this decomposition through the numbering schema.

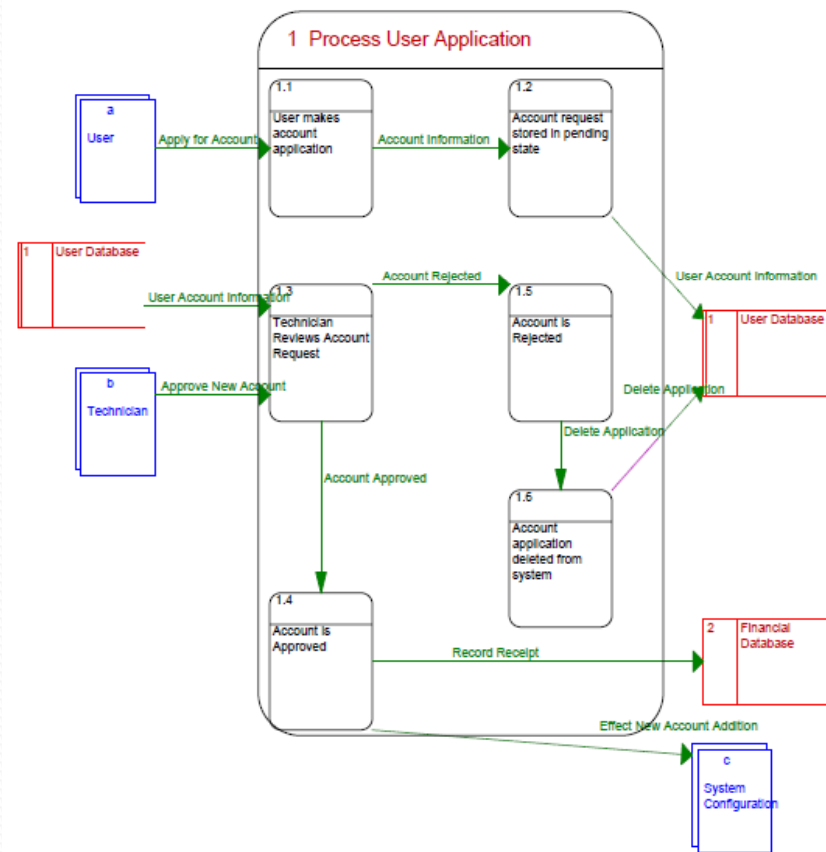
- Here is the top level DFD from the User Management System example.
- Each process has a unique whole number identifier.
- All other rules of DFDs are followed.



- Process 1: Process User Application is further decomposed.
- Note that all data flows with external entities and datastores in the higher level are maintained in this diagram.
- Nothing is lost.



- All subprocesses within Process 1 are numbered 1.x to denote their parent process.
- If this is still not enough detail, we can decompose any of these processes again, numbering them as 1.x.y
- All DFD rules must still apply.





The Laws of Data

- **Law of Inertia:** Data at rest, stays at rest until moved by a process.
- **Law of Conservation of Data:** Processes cannot consume or create data. They can only change data.



Validating DFDs

- **Syntax Validation:** Many System Design packages will create DFDs and ensure that all data flows are maintained through decomposition. (It will warn you if a pipe isn't connected to anything, or if a process doesn't process anything). They make sure that the DFD is “well-formed.”
- **Semantic Validation:** Only your users and client can tell you if the DFDs you've created match their business practice. It's important to take your DFDs back to your client, have a **walk-through** and ensure that they are accurate.



Subtle Questions to Ask:


- Does a process have all the input it needs to change the data?
- Do the lowest level DFDs maintain the dataflows documented in the highest level DFDs?
- Is the same terminology used through the model? Is “Customer Order” used at one level and “Sales Order” at another? Do terms mean the same thing within the organization? (eg. Does order date mean the same thing to the sales department as the shipping department?)


Summary




Summary

- Data Flow Diagrams (DFDs) are an important method create System Design without omitting any requirements (or forgetting any ingredients)
- DFDs map Functional Requirements to System Process Models
- DFDs show how data flows through a system. If a dataflow doesn't have a connection at both ends, data can't flow (or like bad plumbing, it ends up in a mess on the floor).

- 
- DFDs contain representations for **processes**, **datastores**, **external entities**, and **data flows**.
 - Well formed DFDs have the following rules:
 1. All processes must have at least one inflow and one outflow. If it doesn't have an inflow and an outflow it isn't a process (remember the plumbing example).
 2. All processes must modify data flowing through them in some way. If they don't they aren't a process
 3. Each datastore must be linked with at least one process. If it doesn't it is outside your system and you don't need to worry about it.

- 
4. Each external entity must be involved with at least one process. If it doesn't it isn't a concern to your system
 5. A dataflow must be attached to at least one process. Again, remember the plumbing example.
 6. Each datastore must be linked with at least one process. If it doesn't it is outside your system and you don't need to worry about it.
 7. Datastores have no intelligence. They can't process or manage data; they only store it. If you want to do anything with the data in a Datastore, you need a process.

- 
- We can successively decompose a Level 0 (or High Level DFD) into lower and more **granular** levels.
 - When we have enough detail that we can understand how to build the processes, we can stop.
 - We must validate our DFDs to make sure that:
 - They are **syntactically correct** and well formed;
 - They are **semantically correct** and reflect the business model and practice we are automating;
 - Once validated, we are able to start designing our process model.



References

- [http://www.smartdraw.com/resources/tutorials/Drawi
ng-Nested-DFDs](http://www.smartdraw.com/resources/tutorials/Drawi
ng-Nested-DFDs)
- [http://www.agilemodeling.com/artifacts/dataFlowDia
gram.htm](http://www.agilemodeling.com/artifacts/dataFlowDia
gram.htm)