TSSM™
Creating VCE Success

Software Development Teach Yourself Series

Topic 4: Analysis Tools Unit 3

A: Level 14, 474 Flinders Street Melbourne VIC 3000
T: 1300 134 518 **W:** tssm.com.au **E:** info@tssm.com.au

Contents

CASE STUDY

All Teach Yourself Series in this package will refer to the following case study.

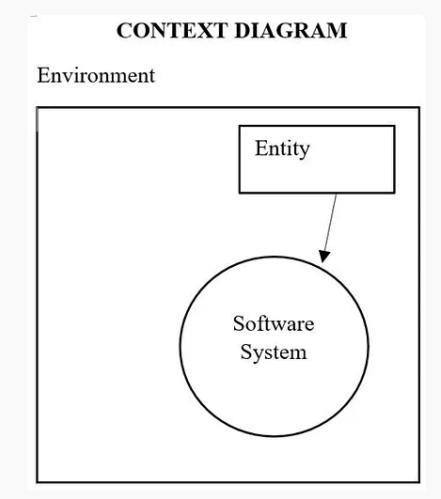
Tariq Mulner is the manager of a school canteen. He manages how many lunches are going to be prepared each day. It is difficult to tell how many lunches will be sold, so he would like a software solution that students can use to order lunches. This application would provide him with a complete list of orders.
Most of the students have smart phones, so Tariq is suggesting the solution is a phone app that can read in the lunch order, and send it to his device so he can print out the order list.

Photo by Tirachard Kumtanom from Pexels used with permission

Analysis Tools

In the Software Development Study Design, we focus on the following analysis tools:
Context Diagrams
Use Case Diagrams
Data Flow Diagrams

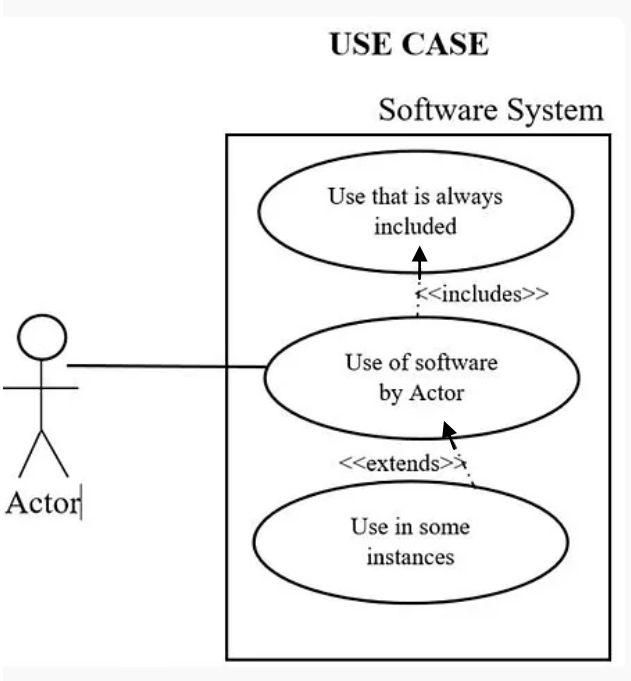
Context Diagrams



To initially analyse the problem your software is going to solve it is best to start with a context diagram. The diagram is contained in a rectangle which represents the environment where the software will be used. For example; an office, at home, in a school, in a vehicle etc. The Circle represented the whole software solution and the user is now called an Entity. An entity is a person or thing (it can also be an outside database or computer system).

Context diagrams can be very useful when multiple users and outside organisations are connecting to the system you are building.

Use Case Diagram



Use Case Diagrams display the way a solution will be used. Rather than writing a complex algorithm to solve the problem, the first stage is to identify the system in terms of its users and what each "CASE" will be. Each CASE is a how a user interacts with the solution.

You create a USE CASE in 6 STEPS:

1. Create the System boundary
2. Identify the actors (the users' roles)
3. Identify each CASE the user will interact with
4. Create associations between Actors and CASES
5. Add extra CASEs where there are more than one step required (includes)
6. Add extra CASEs where there may be an extra step involved on occasion (extends)

Arrows in Use Case

We use a direct line between the Actor and the Use Case, but if there are <include> steps the line must be dashed and the arrow head pointing to the extra step. If there is a Case where an extra step is required on occasion the arrow much point from the <extend> to the main case. See the example above.

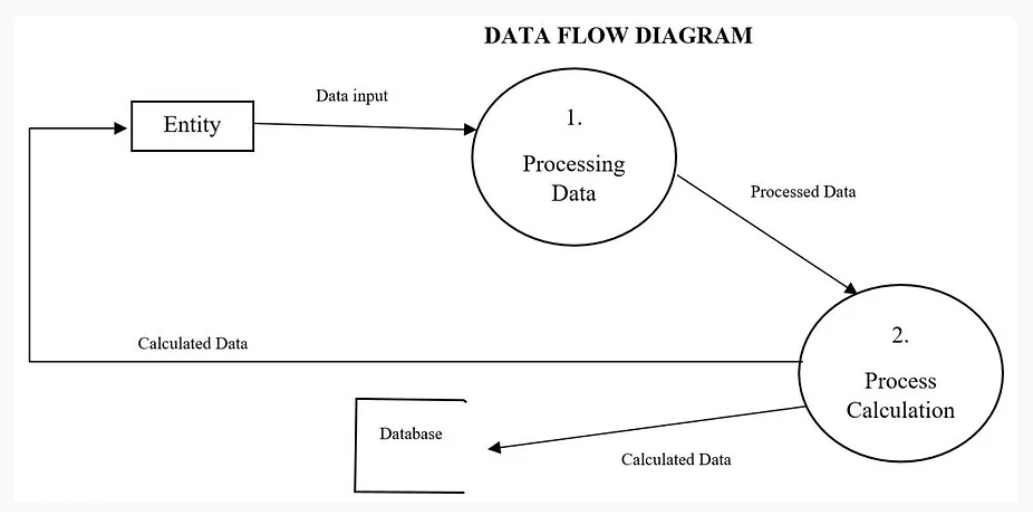
Data Flow Diagrams

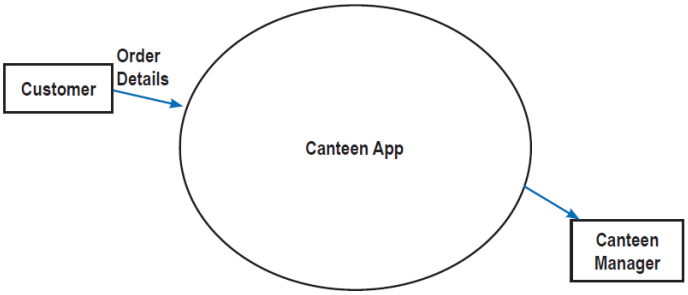
It is important to know the structure of DFDs. First you must begin with a Context Diagram which is the highest level of DFD. It displays the system as a whole and how data flows in and out of the system to and from entities.

The Next Layer DFD takes the Context Diagram and adds up to 8 key processes that represent the solution. Please take care to use the all the rules below:

A DFD contains: Processes, Entities, Data Flows and Data Stores.

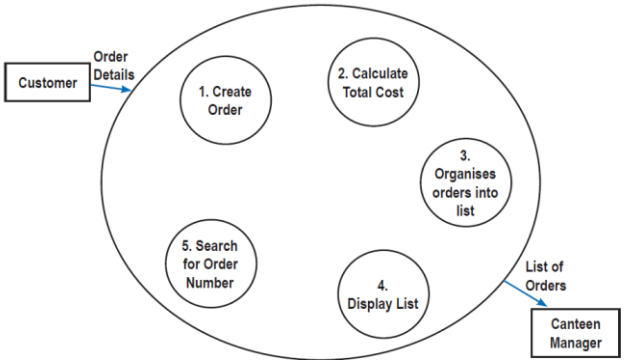
1. Reading the diagram should start at the top left and move to the bottom right.
2. Data can't flow entity to entity
3. Data from a Data Store must go through a process before going to another Data Store
4. A process must have data in and data out. Don't leave them hanging.
5. Data coming out of a process must be different to the data entering it.
6. Rejected data can be sent out of the diagram
7. Keep the DFD to 8 processes
8. Use good processing verbs in your processes: Calculate, edit, delete, sort, add.



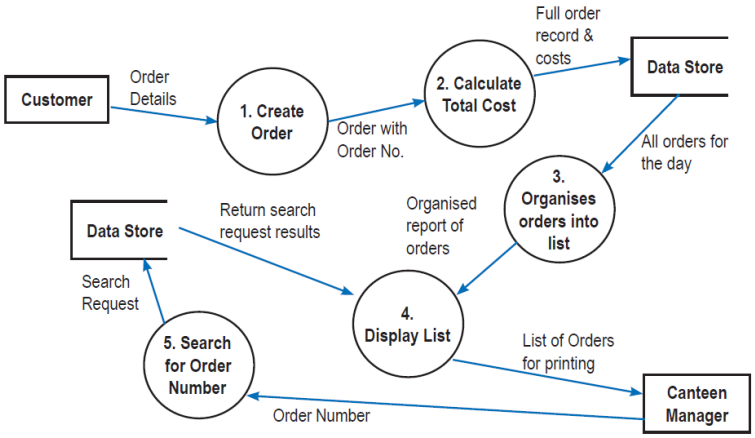


A Context Diagram is a low level Data Flow Diagram. Below is a Context Diagram for the Canteen App:

To convert the Context to a Data Flow Diagram, simply put more detail into the system circle with each process.



Finally, the data flow and storage is added.



Requirements

Once the problem has been analysed, a list of requirements need to be produced. There are two types of requirements:

- Functional – what the software will do. They identify the input, the processes and the output.
- Non-Functional – How the software will function.

Functional Requirements for the Canteen App will include:

- Reads in StudentID and Password
- Checks if login details are correct
- Reads in the user's lunch item selection
- Reads in the lunch item quantity
- Sends the order to the database
- Calculates the total number of items ordered
- Allows for the full list to displayed/printed.

Non-Functional Requirements

- Usability – is it easy to use
- Reliable – always correctly processes orders
- Robust – responds to user errors

Constraints

All project have constraints in what can be implemented in the development of the software. There are five main constraints within software development:

- Economic Constraints – limited finances mean there is often a budget to stick to. Most project have a deadline so time is limited.
- Technical Constraints – these might have something to do with systems that are already in place, such as emails servers, WiFi or just the hardware that is available for the system to run.
- Social Constraints – in some contexts, people are not willing to take up new technology. Some have limited experience with using apps and require design features to assist them.
- Legal Constraints – there is legislation that limits what developers can do with data especially sensitive data such as personal or financial details.
- Usability Constraints – the ease of use of the solution will limit its complexity. How useful the app will be is also a constraint on what can be included.

Scope

It is important for all projects to clearly identify the scope of what is included in the software build. The scope outlines:

- What is included in the project
- What is NOT included in the project.

With the Canteen App for Tariq, there is an opportunity for other functions to be built in – such as a stocktake element. Often while a project is in progress, a client may start asking for extra functions that were not agreed upon in the first instance. This is called scope-creep and can be avoided by clearly stating what the prototype will and will not do.

The Scope of the Canteen App will:

- Read in Orders
- Calculate the number of each item
- Produce a list of items and the quantity required
- Search for a student's order by their ID.

The Scope of the Canteen App will not include:

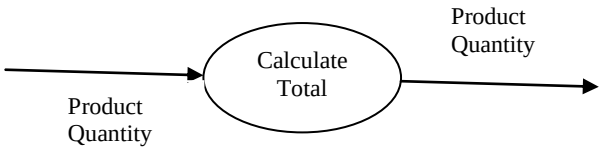
- A calculation of ingredients necessary to fulfill the orders
- A stocktake tracking system of all items in the canteen
- An ordering feature for ordering new items from suppliers.

Review Questions

1.What does an Actor represent?

A. An entity in a DFD
B. An entity in a Use Case
C. An entity in a Context Diagram

2. What is wrong with the data flow on the right?



A. Product Quantity is not data
B. Calculate is not a process
C. The data is unchanged after the process

3. What does a circle represent in a Use Case?

A. A Use Case
B. A Process
C. An Entity

4. Why do we create Use Case diagrams?

A. To identify the way it will be used
B. To identify the processes
C. To identify the data

5. Why do we create Data Flow diagrams?

A. To identify the way it will be used
B. To identify the processes
C. To identify the data structures

Applied Questions

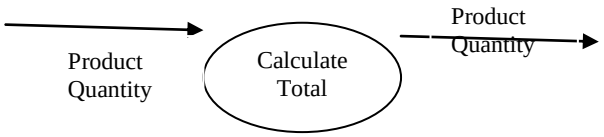
Tariq has asked you to produce an app. The Analysis is available in TOPIC 1. You have 3 weeks to develop the app to hand over. Ideally Tariq should have the solution for 2 weeks before evaluation begins.

6. Create a Use Case diagram for the Canteen App.

Solutions to Review Questions

1. B. An entity in a Use Case

2. C. The data is unchanged after the process



3. A. A Use Case

4. A. To identify the way it will be used

5. B. To identify the processes

Applied Questions

Tariq has asked you to produce an app. The Analysis is available in TOPIC 1. You have 3 weeks to develop the app to hand over. Ideally Tariq should have the solution for 2 weeks before evaluation begins.

6.