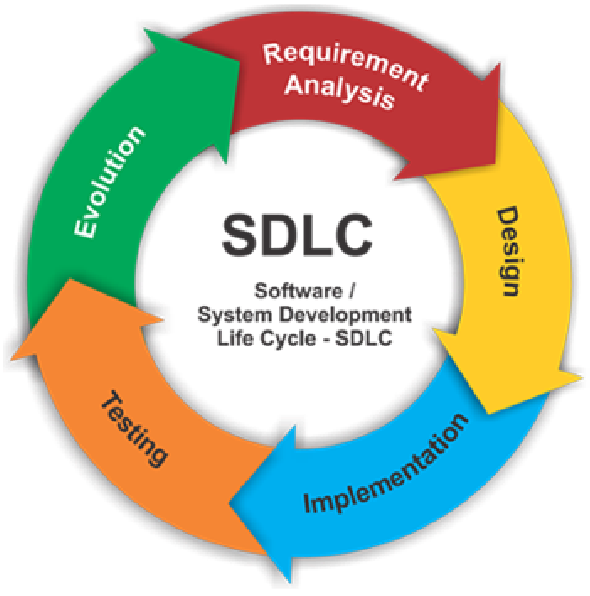
Unit 1: Software development life cycle



This unit considers the equivalent of the old MYP Design cycle. At university level for a Computer Science course, the design cycle is known as the SDLC: System development life cycle (some books substitute "system" with "software").

The major sections of the SDLC are:

* Requirements analysis
* Design
* Implementation
* Testing
* Evolution

The SDLC itself is not a specific part of the current IB syllabus, however unit 1 is completely built around the idea of the SDLC so it makes sense to structure this unit in that way.

1. Requirements analysis.

The aim of the requirements analysis is simple: Figure out what the project needs to accomplish.

There are typically 4 elements to this:

- Project scoping
- Stakeholder consultation
- Project research
- Requirements planning

1.1 Project scoping

A scope document allows all parties to agree in writing as to what is and is not included within the project. It effectively becomes the contract and establishes common expectations for client and creator.

Happy customer = good word of mouth.
Disgruntled customer (feels ripped off or that you didn't deliver on what was promised) = bad for business.
Burnt out producer (feeling taken advantage of, frustrated at changing requirements) = Not healthy!

Scoping should be SMART

- Specific
- Measurable
- Agreed upon
- Realistic
- Time bound

Ensure your scopes identify time, financial and resource constraints!

It is inevitable that a project scope will change over time. When that occurs, beware the challenges of scope creep vs scope discovery:

- Changes to the scope that make it more clear / less vague = Scope discovery
- Changes that make the scope less clear / more vague = Scope creep

1.2 Stakeholder consultation

Who is relevant? Employers, employees, customers…?
Who will use the system?
Who will depend on the system (even if they don't use it directly)?
Who will provide information the system depends on? (even if they don't enter it directly)
Who is paying for the system? (why?)
Possible strategies for obtaining requirements from stakeholders

Surveys
Interviews
Direct observations
Document collection
Failure to involve all relevant stakeholders may lead to software that is not suitable for its intended use! (The manager doesn't necessarily always know what the clerical staff do!)

Effective collaboration and communication between all parties: client, developer, end users.

Consultation should occur continually throughout the lifecycle to identify problems early.

Be aware of privacy issues – being able to get honest, frank information from a stakeholder without fear of retribution (eg: a staff member who might have valuable insight into how some part of the system doesn't work the way management thinks it does) – create an environment where you can extract those valuable nuggets

Exercise

Split off into pairs - one taking the role of customer, one taking the role of developer. The customer comes up with an app/software project they want the developer to "design". The developer must ask questions to ascertain:

- A scope document: Explicitly list "in scope" and "out of scope" items
- Identify stakeholders
- Requirement specification: Functional and non-functional items

Reverse roles when ready. Save these project outlines as we will use them later.

1.3 Project research

- Examine the **current system** that yours will replace – strengths, weaknesses, idiosyncrasies.
- Examine **competing products** – strengths, weaknesses, idiosyncrasies.
- Examine **your capabilities** – what are you capable of producing? If you need additional expertise, what can you afford to recruit?
- Examine the **literature** (journals, online forums) – how are other people addressing the problem?

When researching your project, don't forget to account for international factors.

- Does your program only have to work for a homogeneous group of people in the same location? Or might you have to deal with different time zones, different languages, different conventions in date formats? Things can get very complex very quickly as these videos demonstrate.
- The Problem with Time & Timezones - Computerphile
  https://www.youtube.com/watch?v=-5wpm-gesOY
- Internationalis(z)ing Code - Computerphile
  https://www.youtube.com/watch?v=0j74jcxSunY

1.4 Requirements planning

Turn your project scope into a detailed set of requirements that you can give to your engineering team. Your requirements are generally split into functional and non-functional requirements.

Functional: What will the program actually do? For example:

- Store hours worked per employee per day
- Store hourly rates for employees by category of employment
- Store daily timesheets for up to 5 years history
- Calculate income tax obligations per employee at the end of each month

Non-functional: Doesn't affect what the program will actually do, but will impact on creating it anyway. For example:

- The system shall run on Android / iOS / the web.
- The system shall be compatible with Microsoft SQL / Google Firebase
- The system shall share data with (insert other existing product here)
- The system shall be operational in 3 months
- The system shall store its data within Switzerland (due to privacy laws)

2. Design

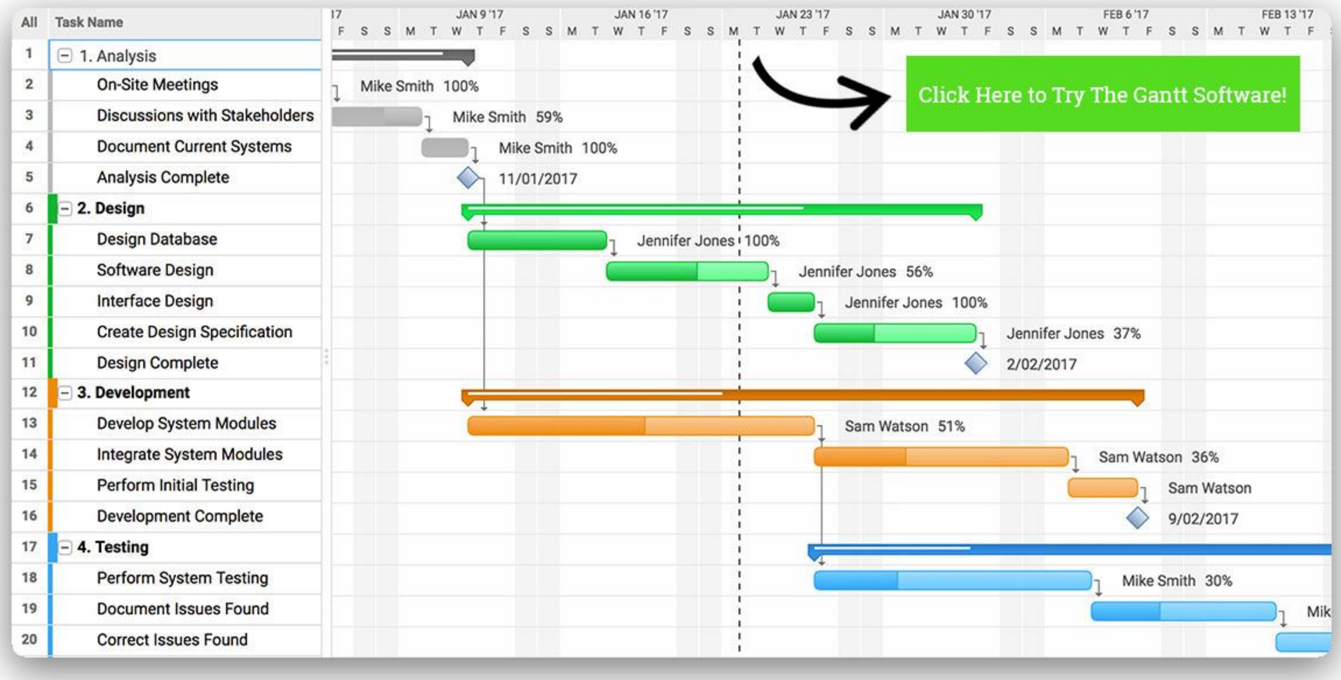
The aim of the Design phase: Create a plan on paper before hacking away at code

There are typically the following elements to this:

- Organise your time constraints
- Diagrammatic representation
- Prototyping
- Client sign off

There are a range of diagrams that may be useful when designing a new software project. We will learn a few key ones now...

Gantt charts



All	Task Name
1	1. Analysis
2	On-Site Meetings
3	Discussions with Stakeholders
4	Document Current Systems
5	Analysis Complete
6	2. Design
7	Design Database
8	Software Design
9	Interface Design
10	Create Design Specification
11	Design Complete
12	3. Development
13	Develop System Modules
14	Integrate System Modules
15	Perform Initial Testing
16	Development Complete
17	4. Testing
18	Perform System Testing
19	Document Issues Found
20	Correct Issues Found

A gantt chart is a type of bar chart that illustrates the timeline of a project schedule. Each task is assigned a row, and the horizontal axis is time scale.

Within the Gantt chart it is important to identify tasks that are prerequisites for other tasks so as to find the critical path and other potential bottlenecks.

A good introduction into Gantt charts is here:

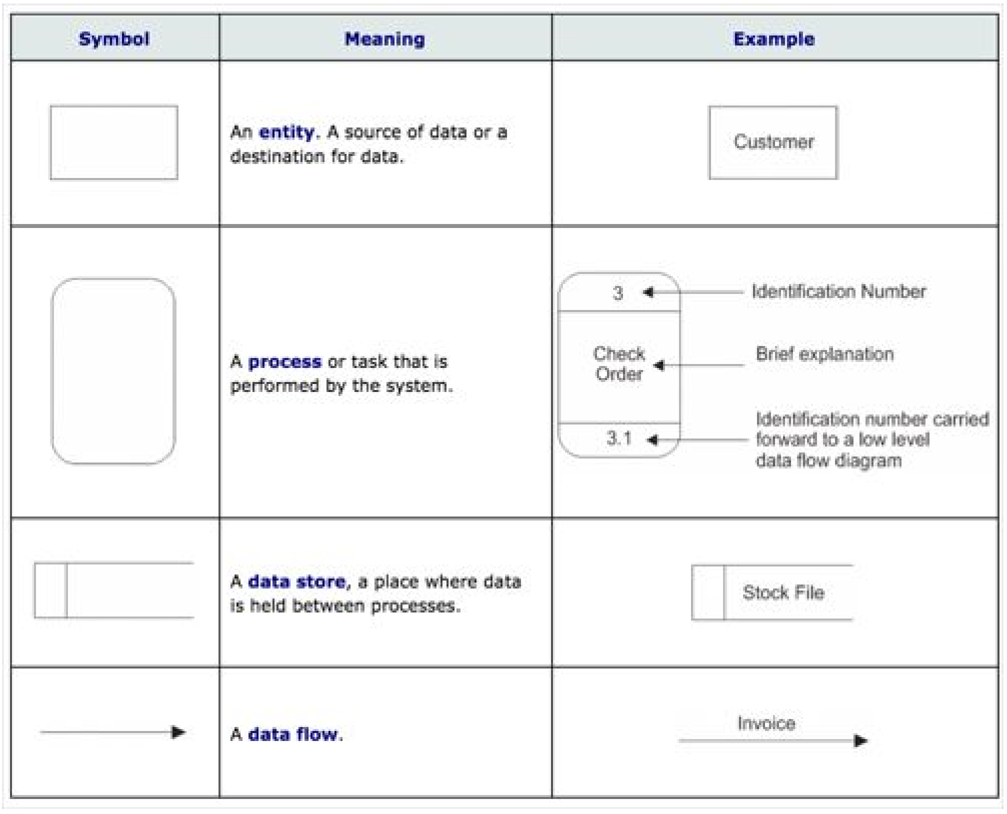
- https://www.youtube.com/watch?v=fB0wsdmV3Sw (5:21)

Data Flow diagrams

Data Flow diagrams (DFDs) come in two flavours:

DFD context diagram – shows the “context” (environment?) your system is part of, particularly external entities it has a relationship with
DFD level 0 diagram – shows inside the main / top level process of the system: the core functional parts and the flow of data between them.

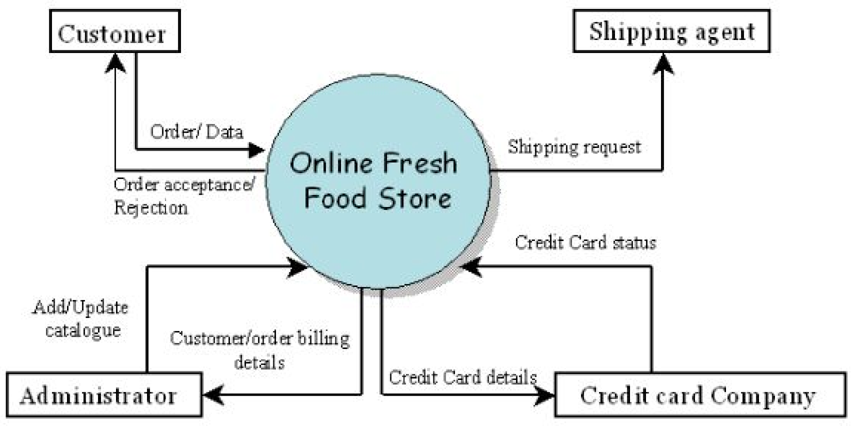
Data Flow diagrams: Symbols



Symbol	Meaning	Example
▭	An **entity**. A source of data or a destination for data.	Customer
▢	A **process** or task that is performed by the system.	3 ← Identification Number Check Order ← Brief explanation 3.1 ← Identification number carried forward to a low level data flow diagram
⊏	A **data store**, a place where data is held between processes.	Stock File
→	A **data flow**.	Invoice →

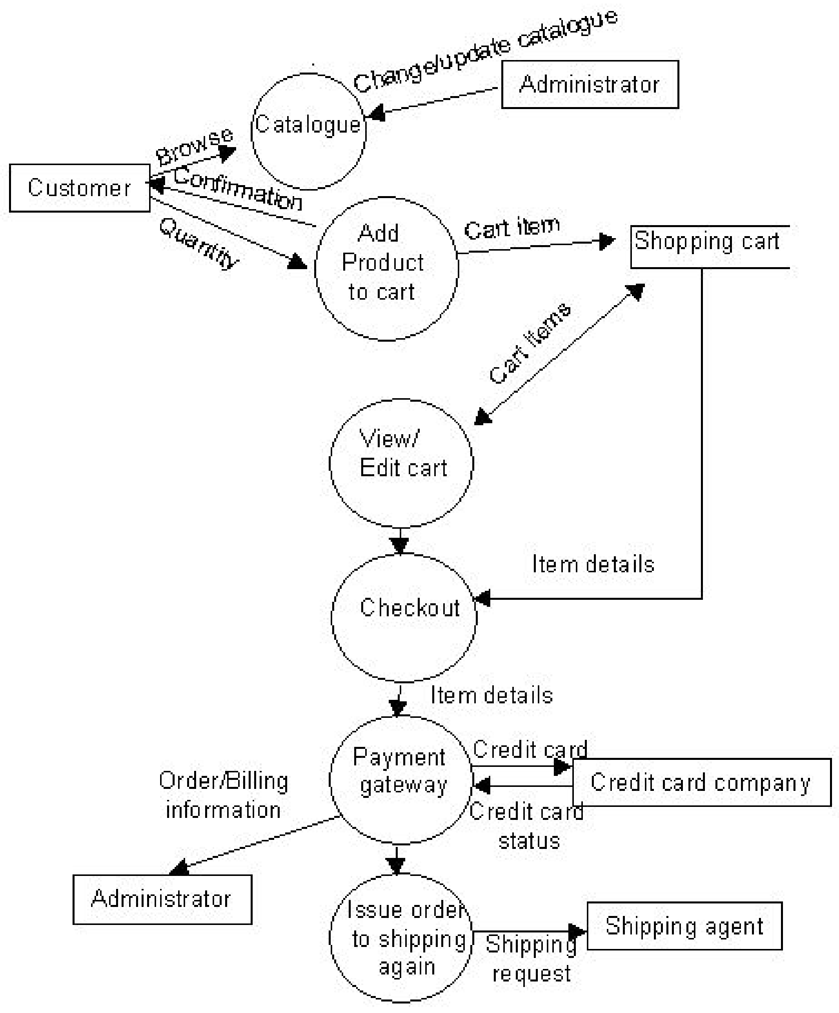
Data Flow diagrams: Context diagrams

Example: Online fresh food store[1]



[1] Examples from http://editorialzone.blogspot.com/2010/09/data-flow-diagrams-of-online-shopping.html

Data Flow diagrams: Level 0 diagrams

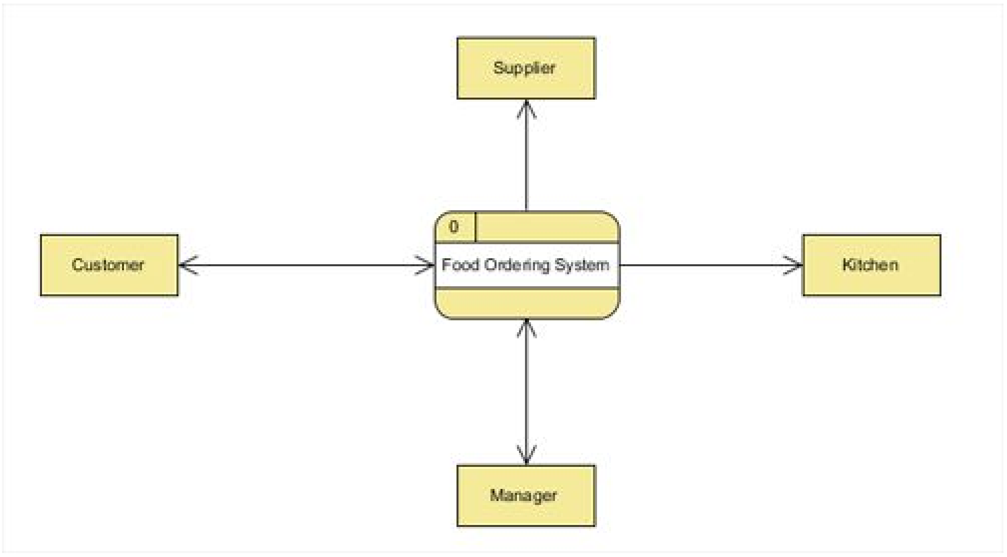


Data Flow diagrams: Guidelines

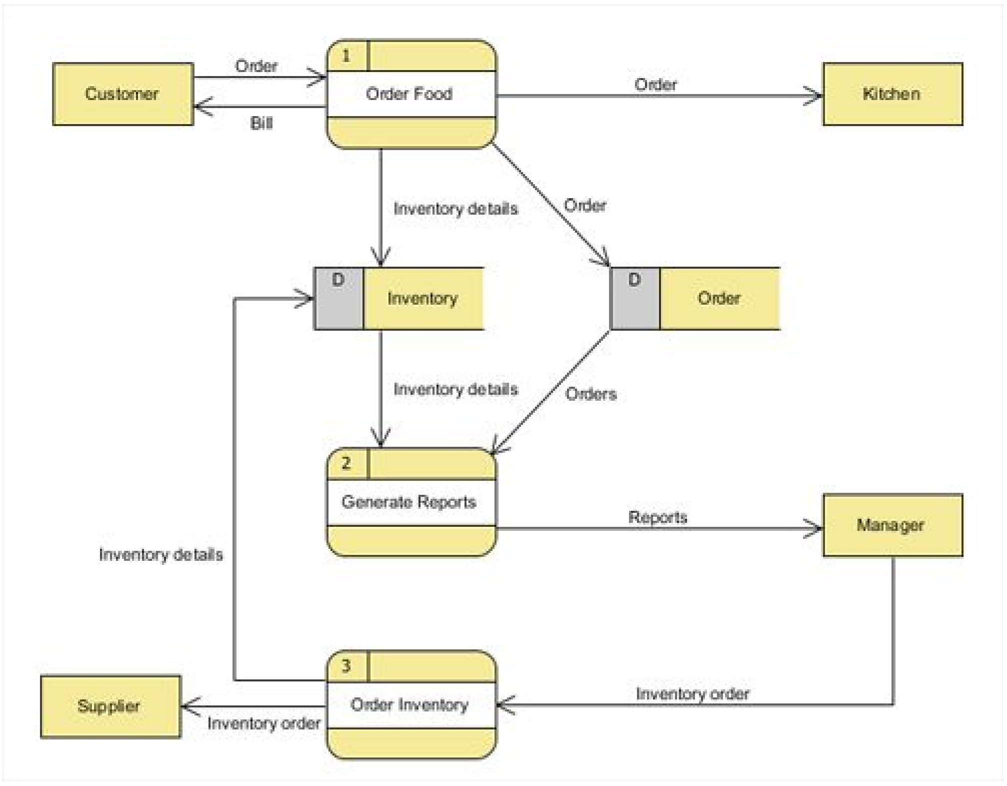
- The context diagram is intended to show the external environment (context) that your system is situated within. The level 0 diagram expands on the context to show the top-level internal processes within the system.
- The context diagram and level 0 diagram both identify the external entities the system interacts with. The set of external entities identified in each diagram must match.
- On the context diagrams arrows indicate whether the flow of information is incoming, outgoing, or two-way for interaction between the system and the entity. They do not need to be labelled.
- On the level 0 diagrams, arrows should be labeled to indicate the information being transferred. Information to/from external entities should never connect directly to a data store, it must travel via a process.

Restaurant ordering system

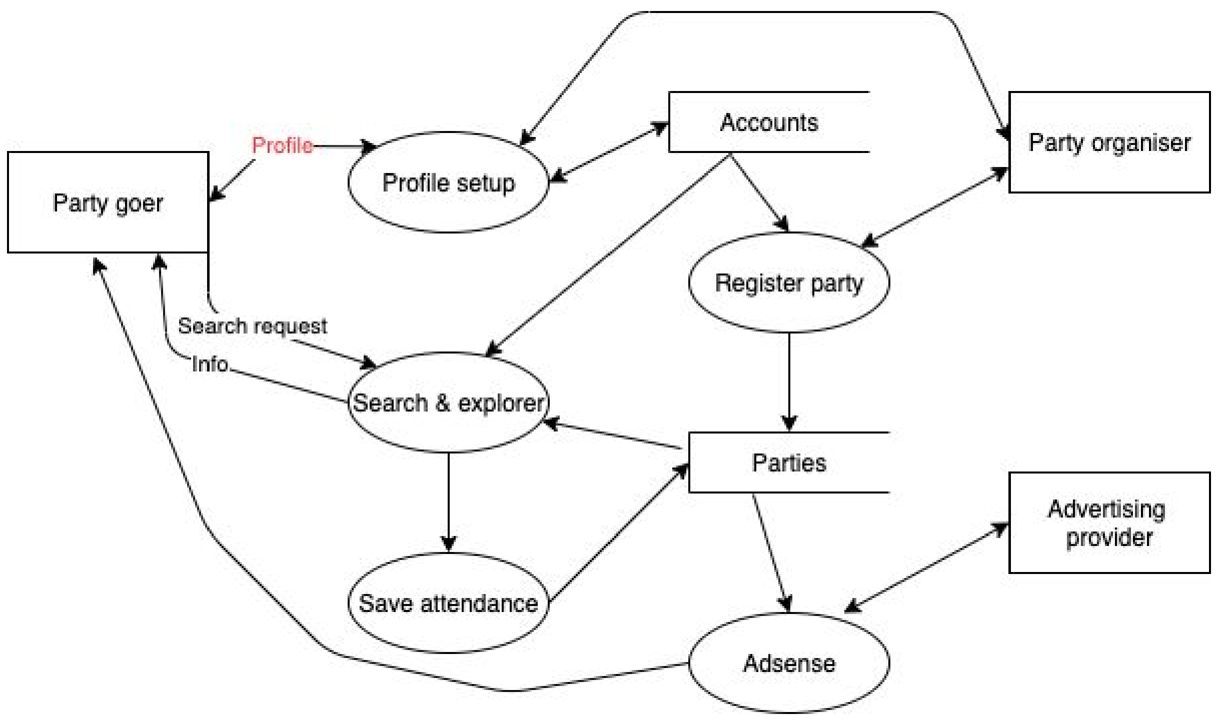
Context



Level 0



Party finder example



Exercise

Pick one of the options the class came up with earlier, and create a context and level 0 DFD for it.

System flow diagrams

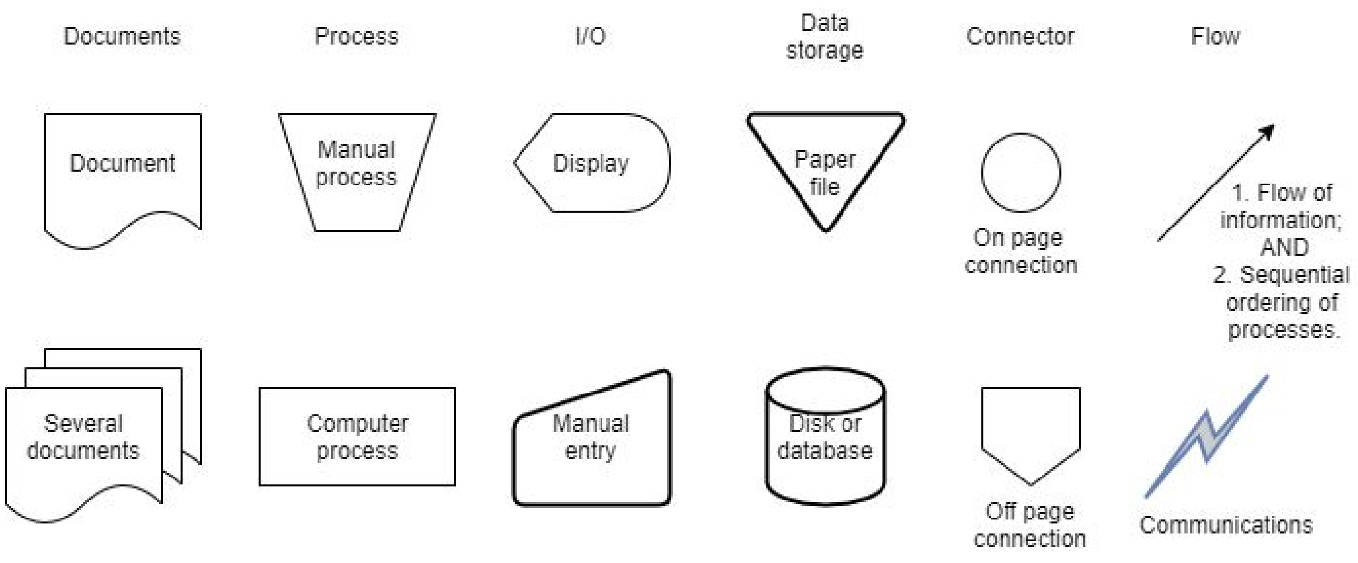
Purpose of the diagram: To capture the flow of documents through a system and to show who is responsible for performing the different activities within the system.

This includes physical processes that may occur outside any particular piece of software. It provides a “system” view rather than an application view. Manual as well as computer processes are represented.

Whereas the logical flow diagram can be used to design the decision making logic of an algorithm, the system flowchart might be used when observing and documenting how users interact with an existing system.

Nomenclature note: Researching for this diagram online: Searching for “system flowchart” comes up with very little. Searching for a process flowchart produces more helpful resources.

Symbols:



- Document - To reprepresent documents being passed around the system. Can be used to denote electronic but typically used to denote paper documents (a printed receipt or a voucher would examples).
- The arrow creates a dependency. One can’t start until the other has ended.
- Data storage paper file - Think a paper file or filing cabinet
- Data storage disk - An electronic file on the computer
- Computer process - As it sounds. When deciding if you have to create a process bubble, keep to the general rule that a computer will always require a process to store or receive information.

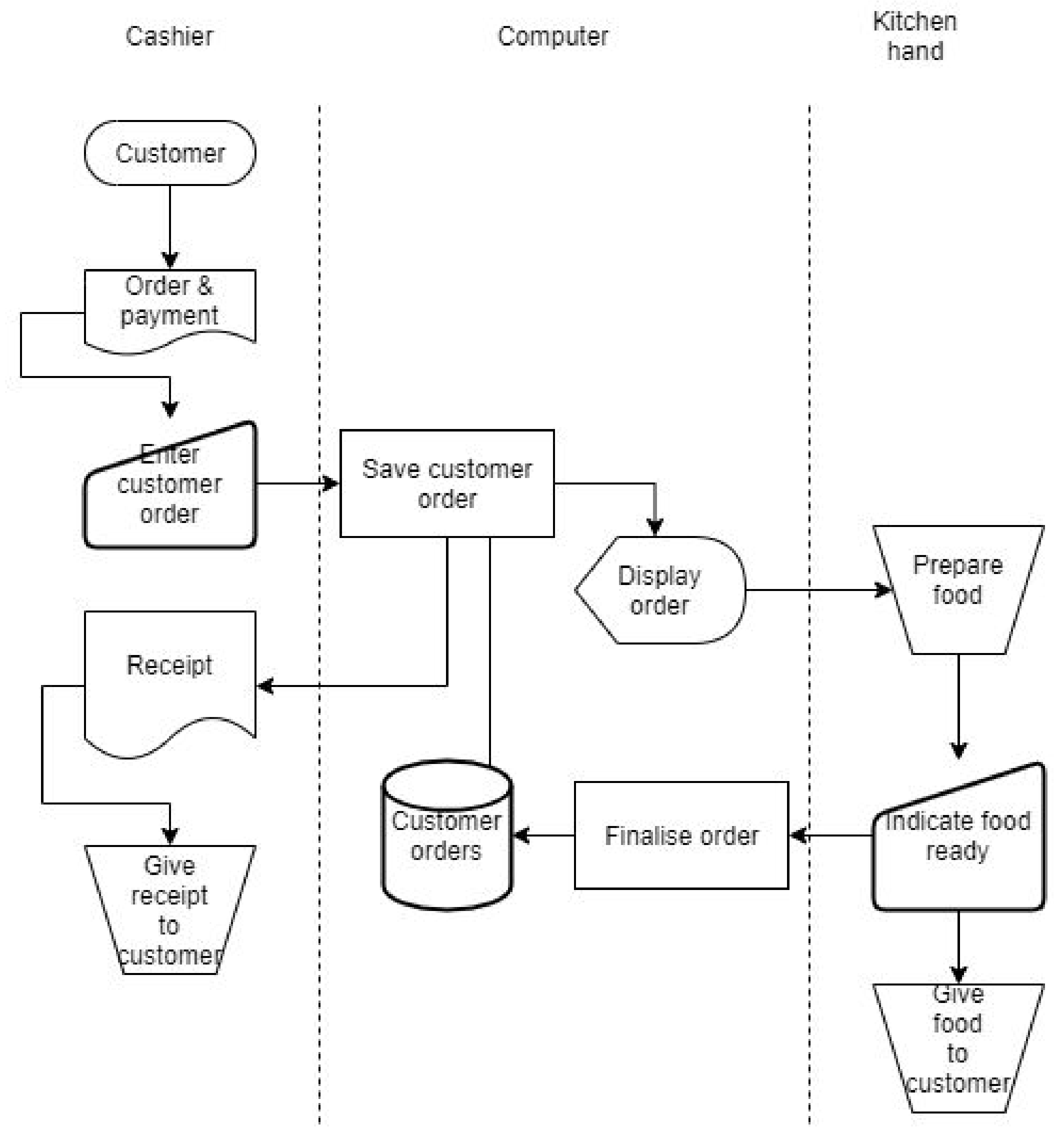
Entities

Some versions of this diagram create columns for the individual entities. Whatever entities you have identified in your DFD should be listed. The IB mark schemes have not required this. My examples below include it anyway.

Example 1 - Consider "customer at a cafe"....

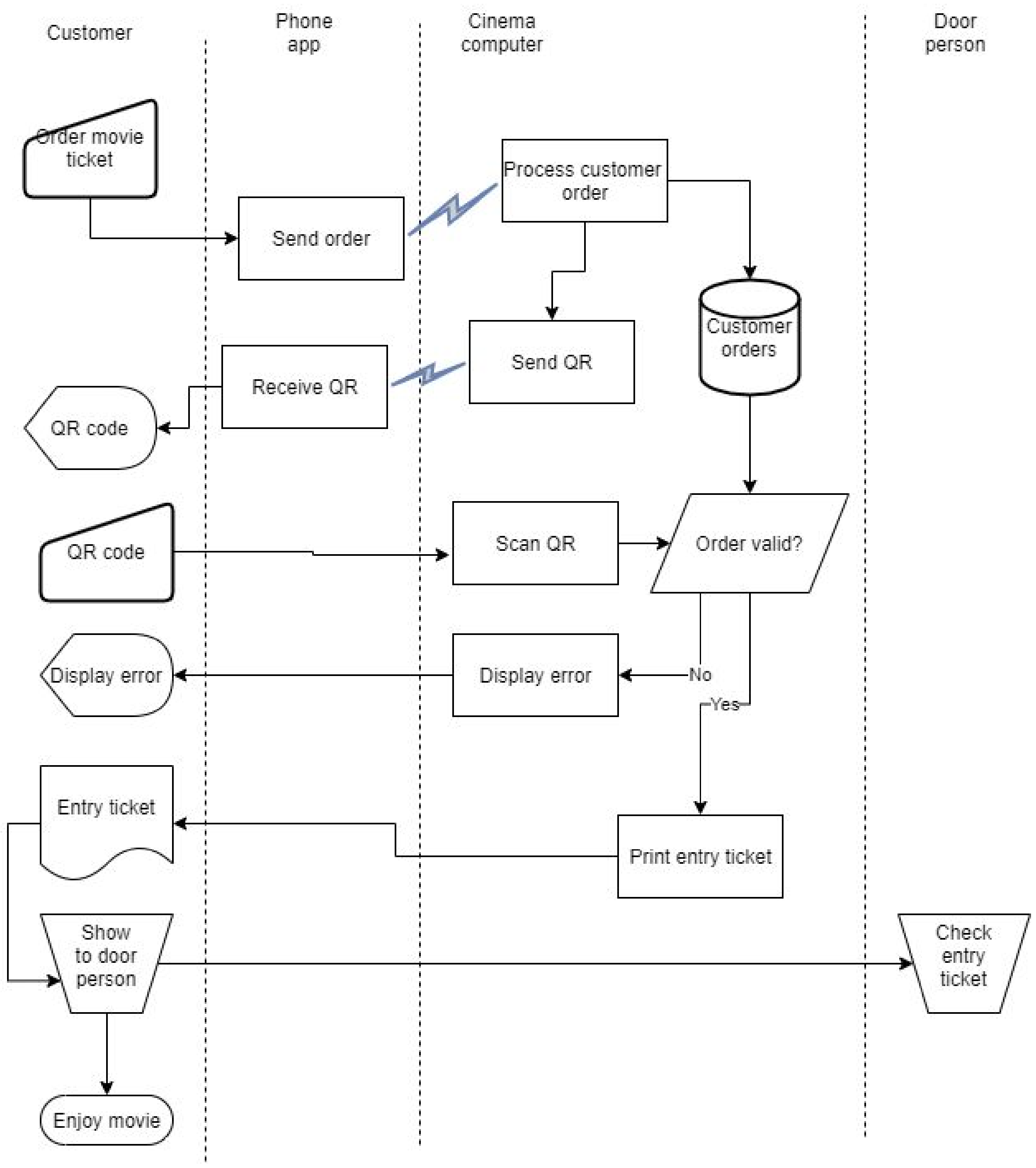
- Customer gives order to sales person
- Sales person enters the order into a computer. That is saved to a computer file, displayed on a screen for a kitchen hand to make the order, and a receipt is printed that the cashier gives the customer.
- The kitchen hand prepares the food, presses an "order finished" button and gives it to the customer.
- The order finished button records the order as finalised

.



Example 2 - Consider “Going to the movies”...

- A customer uses a phone app to order movie tickets online. The cinema online server receives the order and stores in a database and sends back a ticket QR code for display on the phone screen.
- On arrival at the cinema, customer shows the QR ticket code to a computer
- The system checks if the order is valid, and then prints a cinema ticket, otherwise it gives the customer an error screen.
- The customer shows the printed cinema ticket to the door person to gain entry.



Question 1 - Student submits work to teacher for marking….

- A student creates a Google Doc containing an assessment
- Student uploads assessment to Google Classroom. Google Classroom stores that assessment and notifies the teacher it is there.
- Teacher reviews work and enters a grade. Google Classroom stores that grade and notifies the student their work has been graded.
- Student views grade.

Question 2 - Borrower borrows a book from a library….

- Borrower takes a book they wish to borrow to the counter, along with their borrower card.
- The librarian scans the borrower card on the computer, which looks up the Borrower's file to check the borrower is still in good standing (late fees paid etc).
- If approved, the librarian then scans the books and the computer stores the loan.
- A loan receipt is printed and handed to the Borrower to help remind them of the due date for return.

Past paper question 1 - November 2014

- A customer buys an item in a small local shop and pays with a credit card. The sales transaction data is input into a computer at the point of sale. Prices are downloaded every morning from a central computer at the company headquarters. The credit card is verified with the card authorisation centre and then the receipt is printed. (5 marks)

Past paper question 2 - November 2016

- A bookshop has a computer at each point of sale, and also a central computer. When a customer buys a book in the book shop, the salesperson at the point of sale uses a scanning device to input a barcode from the book. The barcode is sent to the central computer where the barcode of each book and the corresponding price are held in a database on a disk. When the price is found, it is sent to the point of sale computer where all necessary calculations are performed, details of the transaction are stored on a local disk and a receipt is printed. (5 marks)

Past paper question 3 - November 2018

- The subsea oil and gas exploration and production unit of a company relies on thousands of kilometers of pipeline which are monitored by a computer control system which can detect leaks. The process of detecting leaks is carried out by sensors which are continuously monitoring changes in the flow and pressure of liquids in the pipes. The data is processed on a computer in the office. If any of the sensor values are outside of the acceptable parameters stored on a disk in the office, the following error routines are performed:
    - An alarm is sounded on the computer in the office
    - An email message is sent to the managers in the Head Office.
    - A member of staff can also configure the system. (5 marks)

Further resources

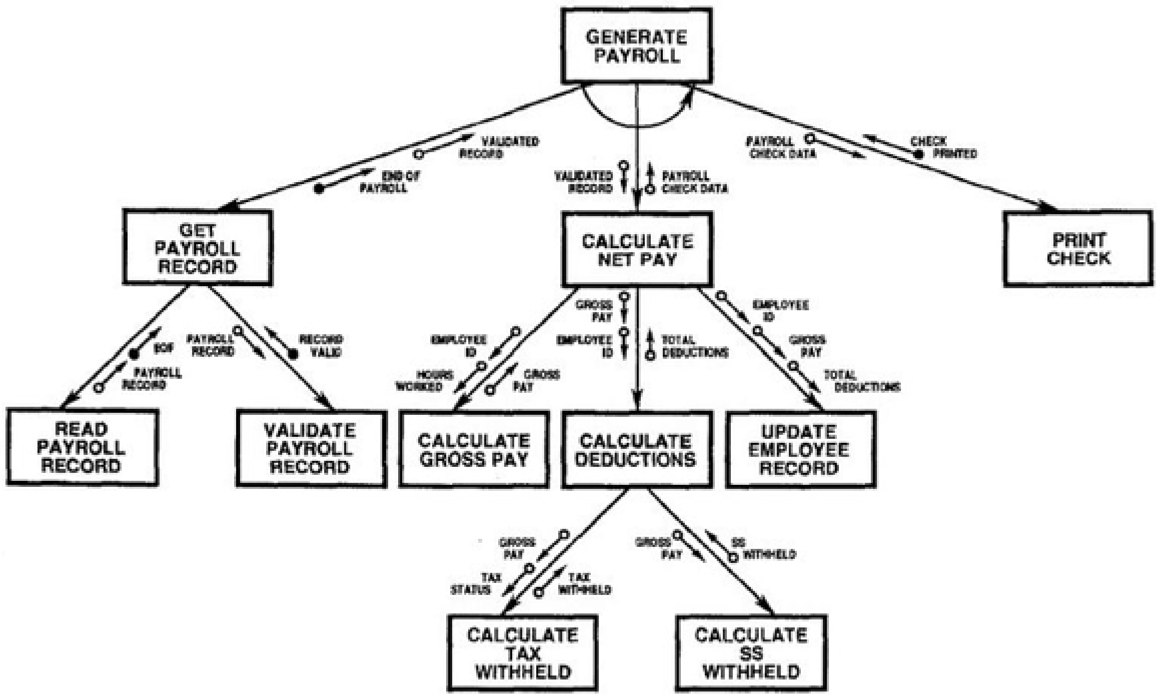
- System Documentation - Part VI: Creating the Flowchart (24:53)
  https://www.youtube.com/watch?v=YTcah66oYJU
- System Flowchart Symbols (5:25) by Sharon Walkey
  https://www.youtube.com/watch?v=AaiQehBm2Y4
- System flow charts (8:31) by Comp Franklin
  https://www.youtube.com/watch?v=QF-ml1QVQYs
- Basic FLOWCHARTING for auditors - documenting SYSTEMS OF INTERNAL CONTROL (15:02)
  https://www.youtube.com/watch?v=2De4hO3lu14

Structure charts

Provides a top-down approach to explain how the different parts of the program are put together.

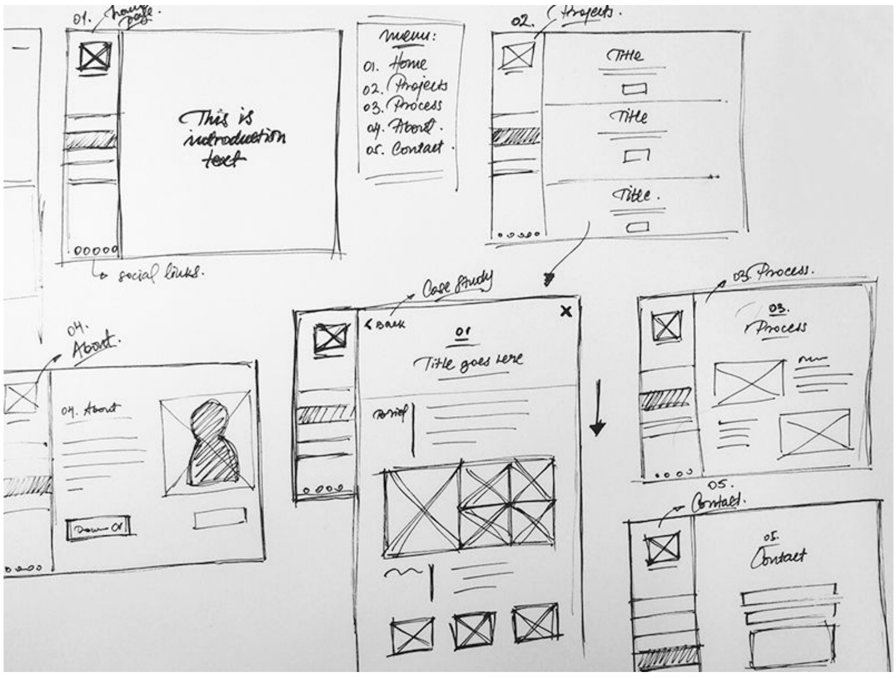
Simplistically:

Programs are made up of modules
Modules are made up of subroutines and functions
Subroutines & functions are made up of algorithms
Algorithms are made up of lines of code
Lines of code are made up of statements and data

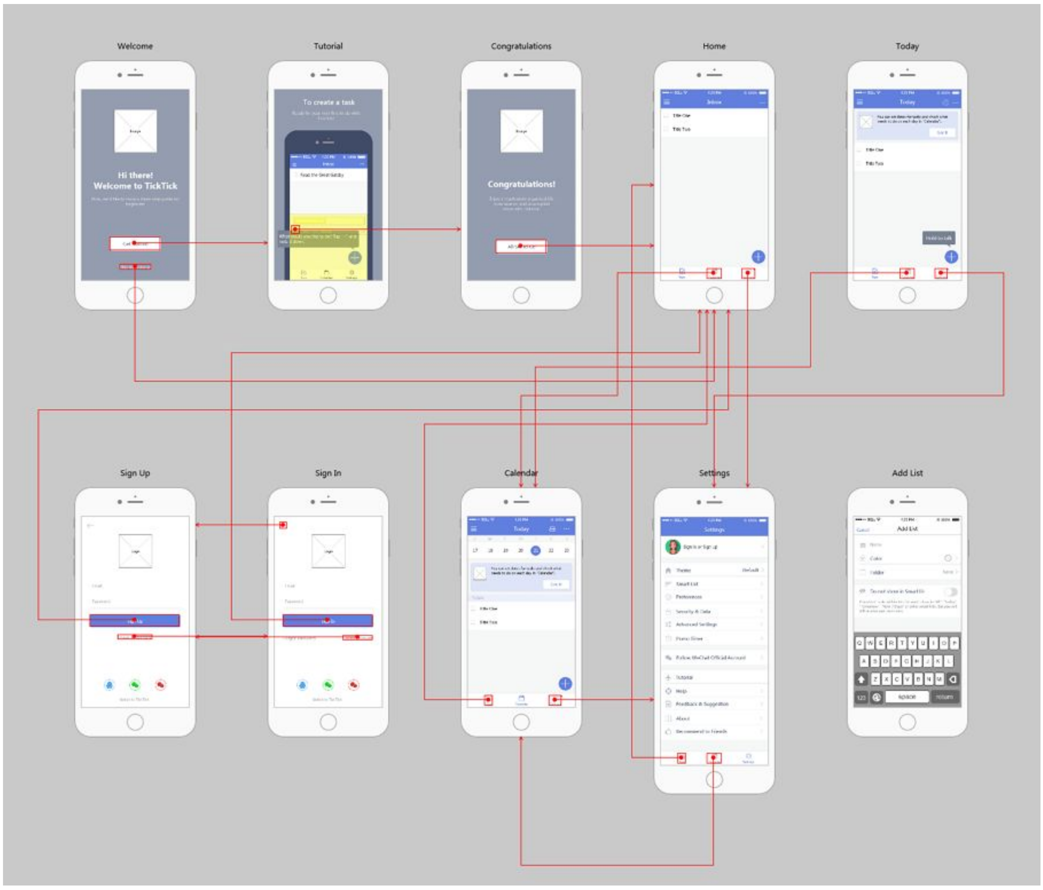


Wireframing & prototyping

- Wireframe: outlines the arrangement of textual and media elements. Wireframing tools allow both parties—the designers and the client—discuss and comment on a sketch.



- Prototype: shows your website or product in action. It is clickable.



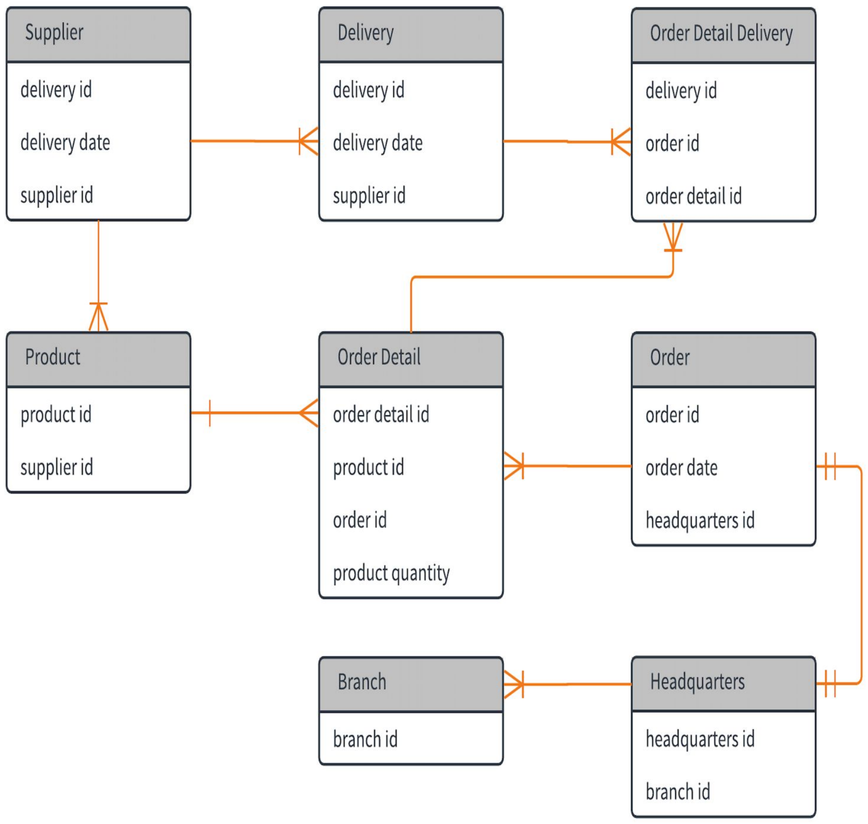
Search for “free wireframe tools” and you will find a bunch of useful tools you can use to help you with this.

Some possibilities:

- https://www.adobe.com/products/xd (free plan available?)
- https://mockflow.com/ (limited free plan, maximum 3 pages without upgrade)
- https://moqups.com/ (limited free plan, 1 project (limited to 300 objects) and 5MB of storage)
- https://www.lucidchart.com/ (limited free plan available)
- https://balsamiq.com/wireframes/ (30 day free trial)

Entity relationship diagrams

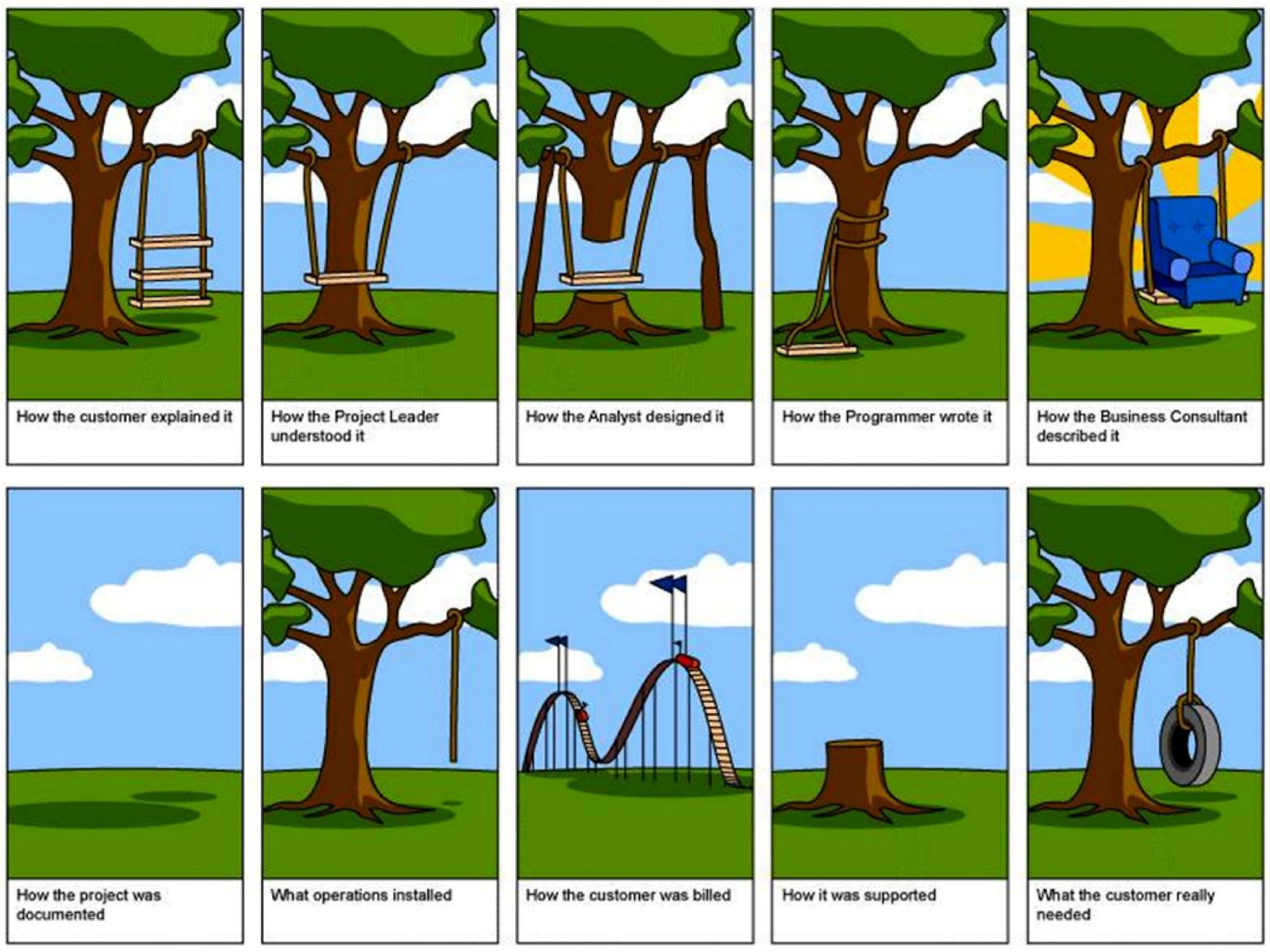
An entity relationship diagram is used for documenting a relational database system. It is not a required diagram of the course, though it would be highly recommended for your Internal Assessment if you are going to create a database as part of your solution. In that instance, I will give you a quick lesson regarding how to construct the diagram. Until then, feel free to move on, nothing to see here ;-)



Client sign-off

If you expect them to pay your bill, make sure they've agreed to everything... in writing.

You had the signed and agreed scope, now have them agree to the design you've built as the proposed incarnation of that project before you start serious development!


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

3. Implementation

The aim of the implementation phase: Build it compliant to the design

Some issues that need to be considered in this phase:

- Choice of technology stack
- Migration issues
- Deployment issues
- Implementation issues

3.1 Technology stack

Will your software be available for purchase as:

- Perpetual license? (buy once, use forever)
- Rent? (SaaS… software as a service is becoming popular. Eg: Adobe Creative Suite, Microsoft Office 365)

A lot of major products are moving to a SaaS model (eg: Microsoft Office 365, Evernote, Adobe Suite). Sometimes you aren't even installing anything on your computer because you are using the "software" through their web portal. Eg: Google Docs.

Modern tools available give wider options for how your new technology might be implemented. Specifically a huge development that is comparatively recent is the emergence of cloud computing.

Amazon Web Services, Google Cloud, and Microsoft Azure are the three big global players in this space at the moment, though there are others.

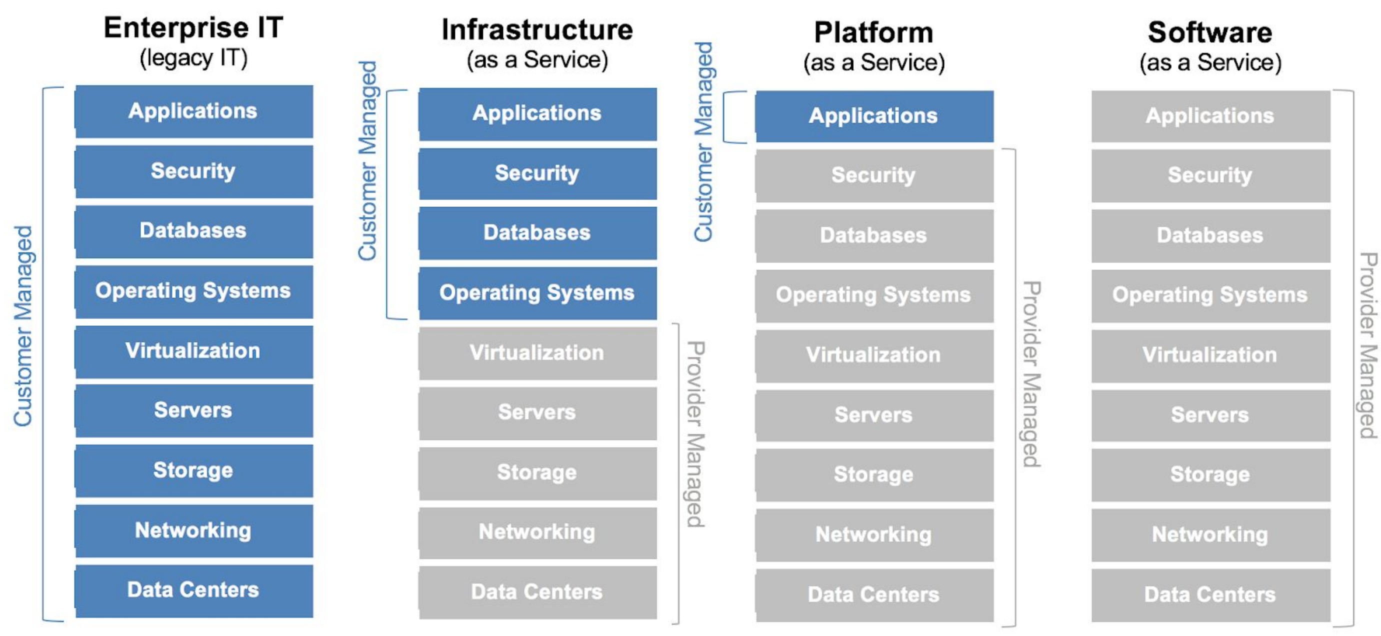
What technologies will you use to build your project and why?

Common concerns/questions about using the cloud reside around you lacking ultimate control over where your data resides, who has access to it. Trust in the cloud provider is necessary for this approach to work.

Elements to evaluate if weighing a cloud-based project:

- Security?
- Encryption?
- Privacy laws (restrictions on where your data may be)?
- Backups?
- Cost?
- Reliability?

Degrees of cloud based computing.



3.2 Migration issues

Issues regarding the maintaining compatibility of your customers data as they migrate to your new system.

- Migrating from legacy systems, or through business mergers etc
- International issues: languages, dates, currency, time zones
- System issues: data types and limits. Example: the Y2K issue and Year 2038 problem
- Converting file formats
- Validating the data you convert from one system to the next (did it all copy across? Accurately?)

3.3 Deployment issues

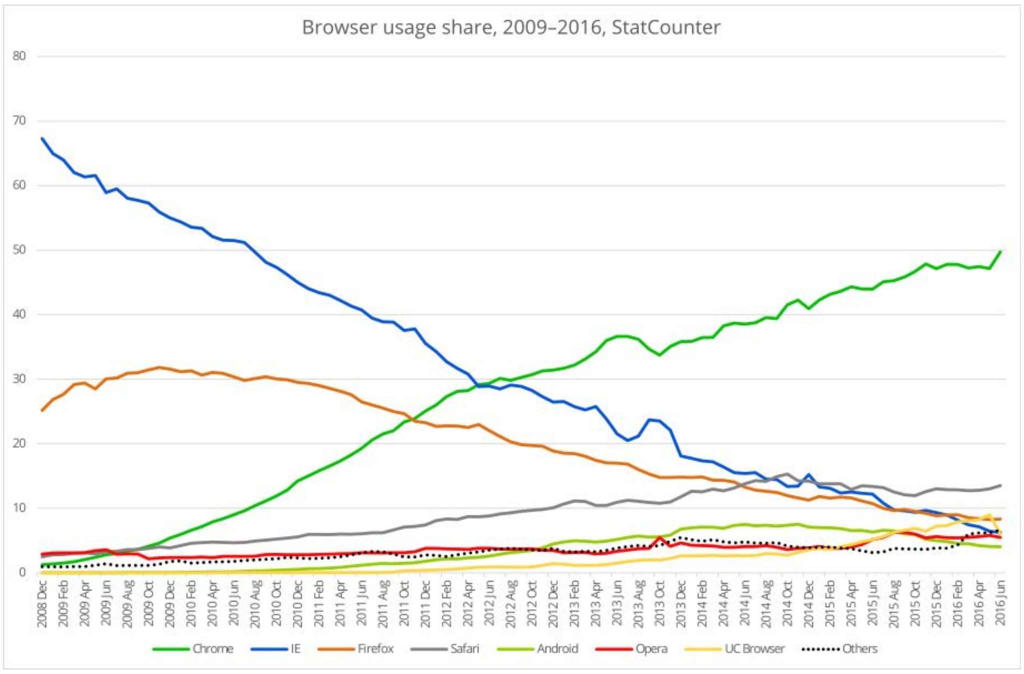
How are you going to deploy your tool?

- App store?
- Website download and install?
- Distribute CD/DVD/USB disks? (do people still do that?)

How will you manage updates/patches to your tool?

- Online, self downloading and updating (bonus: customer never has to think about it. Downside: think of the fuss over Windows 10 "auto" updating)
- Beware of annoying people by using up their internet quotas, or slowing down their link when they are wanting to do something else.
- How can you charge for an update they get automatically? Subscription model which is the way most major vendors are going (eg: Office 365).
- Download and manually install at the customers leisure.
- Distribute disks??? Really? Still?
- If your product is web based, you simply update YOUR systems, and all the customers get the new experience immediately (whether they want it or not – aka facebook's downfall)

The lesson of Google Chrome in relation to auto-updates done "right". There are a multitude of reasons why Google Chrome burst onto the browser market to rapidly dominate market share. See Full Circle Design if you are interested in some well articulated reasons. The aspect that is of relevance here is automatic updates. No longer does the user have to be prompted (nagged) to download updates, go through an install screen, possibly reboot their computer etc; Google Chrome keeps itself up to date for you behind the scenes. This meant new features, bug fixes, security patches, are all rolled out automatically.



3.4 Implementation issues

What implementation / conversion method will you use?

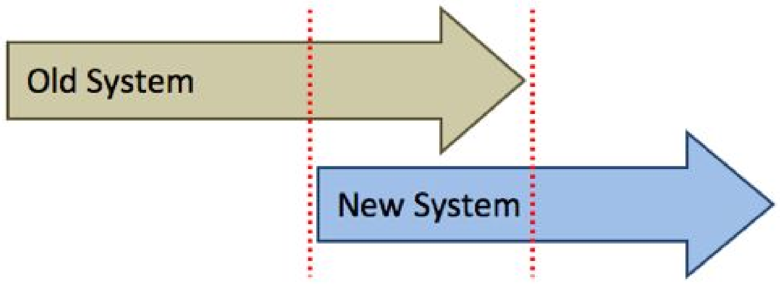
- Parallel running - new system runs simultaneously with the old for a given period of time
- Phased conversion - new system is implemented one stage at a time
- Pilot running - new system is tried out at a test site before launching it company-wide
- Direct change over - entire system is replaced in an instant

What can you see as the plus/minus/interesting of each?

Parallel running

The new system is started, but the old system is kept running in parallel (side-by-side) for a while. All of the data that is input into the old system, is also input into the new one.

Eventually, the old system will be stopped, but only when the new system has been proven to work.



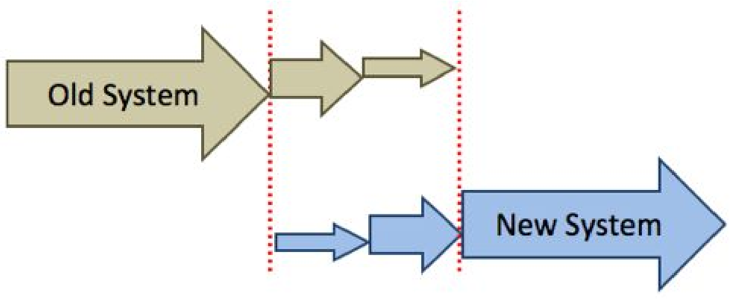
Advantages...

- If anything goes wrong with the new system, the old system will act as a back-up.
- The outputs from the old and new systems can be compared to check that the new system is running correctly

Disadvantages...

- Entering data into two systems, and running two systems together, takes a lot of extra time and effort

Phased conversion



The new system is introduced in phases (stages, or steps), gradually replacing parts of the old system until eventually, the new system has taken over.

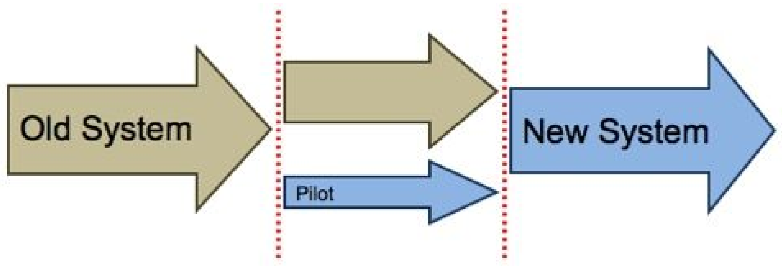
Advantages...

- Allows users to gradually get used to the new system
- Staff training can be done in stages

Disadvantages...

- If a part of the new system fails, there is no back-up system, so data can be lost

Pilot running



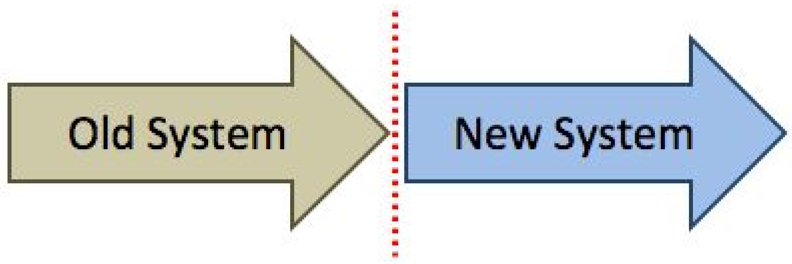
Advantages...

- All features of the new system can be fully trialled
- If something goes wrong with the new system, only a small part of the organisation is affected
- The staff who were part of the pilot scheme can help train other staff.

Disadvantages...

- For the office / department doing the pilot, there is no back-up system if things go wrong

Direct changeover



The old system is stopped completely, and the new system is started. All of the data that used to be input into the old system, now goes into the new one.

Advantages...

- Takes the minimal time and effort
- The new system is up and running immediately

Disadvantages...

- If the new system fails, there is no back-up system, so data can be lost

Case study

https://erichmusick.com/writings/technology/1992-london-ambulance-cad-failure.html

4. Testing

Elements to be considered within the testing phase:

Testing the software
Testing the user documentation
Testing the user training
Testing the backup regime and restoration procedure

Testing the software

Debugging
Automated testing
Beta testing
User acceptance

User documentation

The quality of your user documentation will affect the rate of implementation, or even overall success, of the new system.

What to provide?

- Big fat textbook style “user manuals”? … pretty old school. Some people don’t like reading off a screen. Updating the textbook with changes will be difficult though.
- Help files
- Online support (online chat to support, online community forums)
- Videos

What is best for YOUR users in YOUR project? No one-size-fits-all approach. Along with the documents, there might be a need for actual “training” too…

- Self instruction
- Formal classes
- Online course
- 1:1 coaching

Backups

Data can be lost through a variety of ways. How will your system protect from:

- Malicious activity (crackers, malware, insider espionage)
- Natural disasters (fire, flood)
- Human idiocy (dropping the computer, spilling drink, losing it, leaving laptop in a hot car)
- Time (all disks deteriorate over time)

The amount of effort/expense dedicated to data loss prevention should be judged by the answer to: How would the worst case scenario affect my business?

Eg: hotels/airlines losing all record of reservations? Banks losing record of all account balances? loss of medical records?!

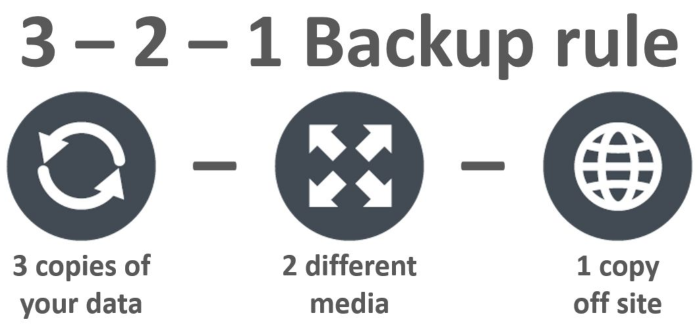
A data loss prevention regime should include a mix of: failover systems, redundancy, removable media, offsite/online storage

Strategies:

- Hot spares
- Cold spares
- Backups

3-2-1 rule of backups:

- At least 3 copies
- At least 2 different forms
- At least 1 copy offsite



Other backup related considerations

- Automated vs manual backups
- Full vs differential backups

Does the cloud count as a backup?

- Debatable question?
- Always live and available is a positive and a negative?
- Quick and easy restoration vs vulnerable to ransomware attacks
- Simply syncing your files somewhere does not help in many scenarios such as:
    - Data corruption
    - Malicious software
    - Deleting files by mistake

5. Evolution

So, your project is up and running. What’s left to do? A post implementation review!

Gather stakeholder feedback on the whole experience (design, migration, training) not just the functionality of the final product.

What improvements or enhancements do they suggest?

Start the cycle all over again with your identified changes!

Past paper questions for review