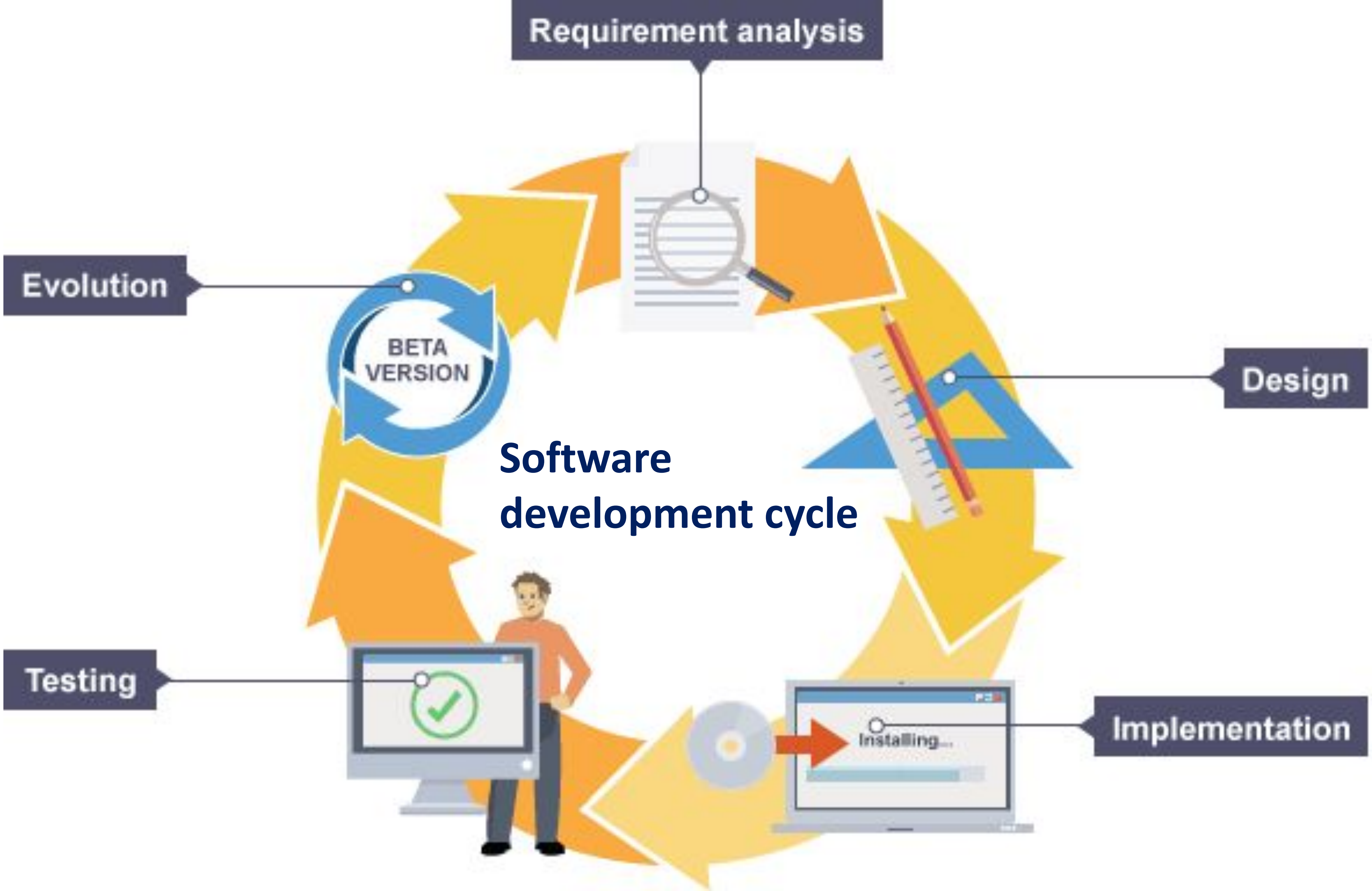
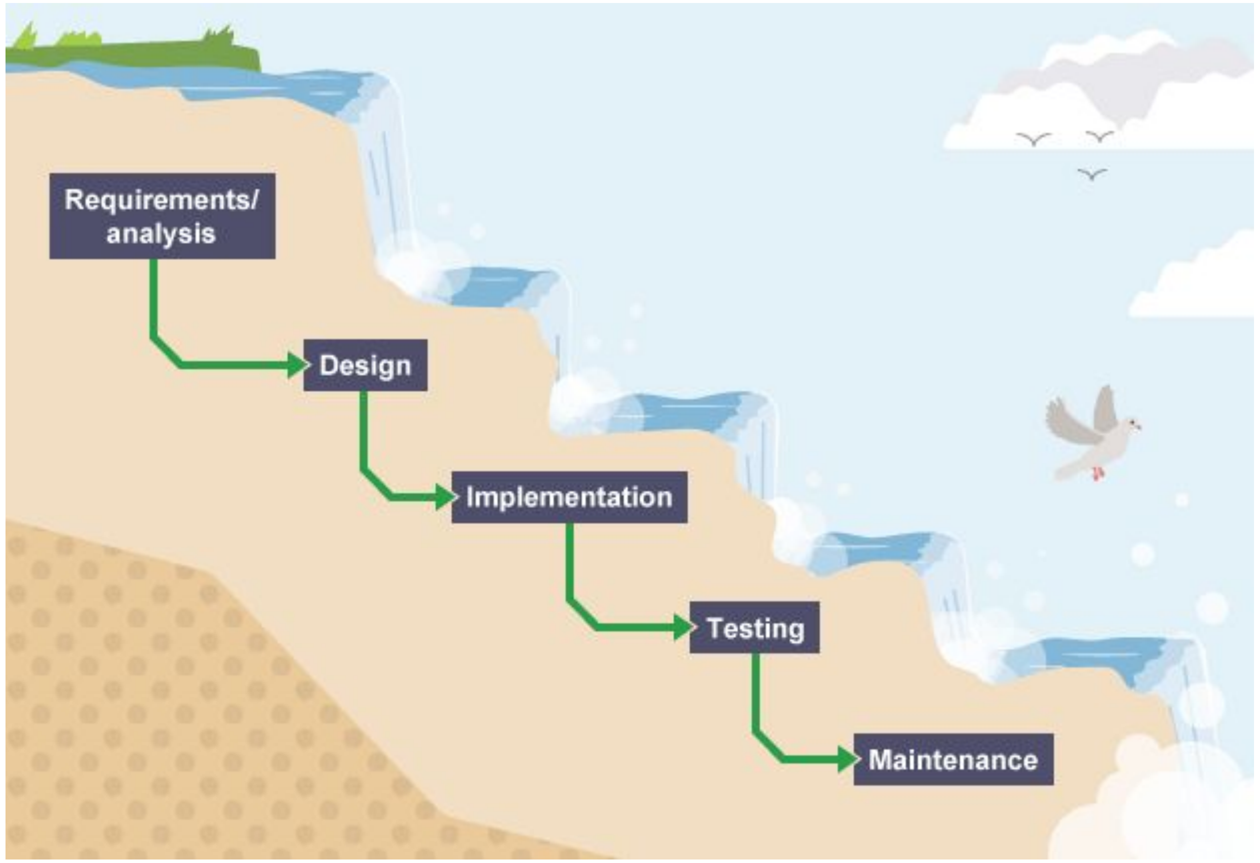
Software development cycle



- **Requirements** - also known as the **analysis stage**. This is the first step, when the team decide what the software needs to do. The main point is to think about what the **user** will want from the program. At this stage, it might be a good idea to ask other people what they want from the software. Who is going to use it? What information do they need to **input**? What information or data does it need to **output**?
- **Design** - the team work out the details of the program by breaking it down into smaller chunks. This includes thinking about the visual appearance and the programming behind the software. The team will use **pseudocode** and diagrams to work out how the program should go.

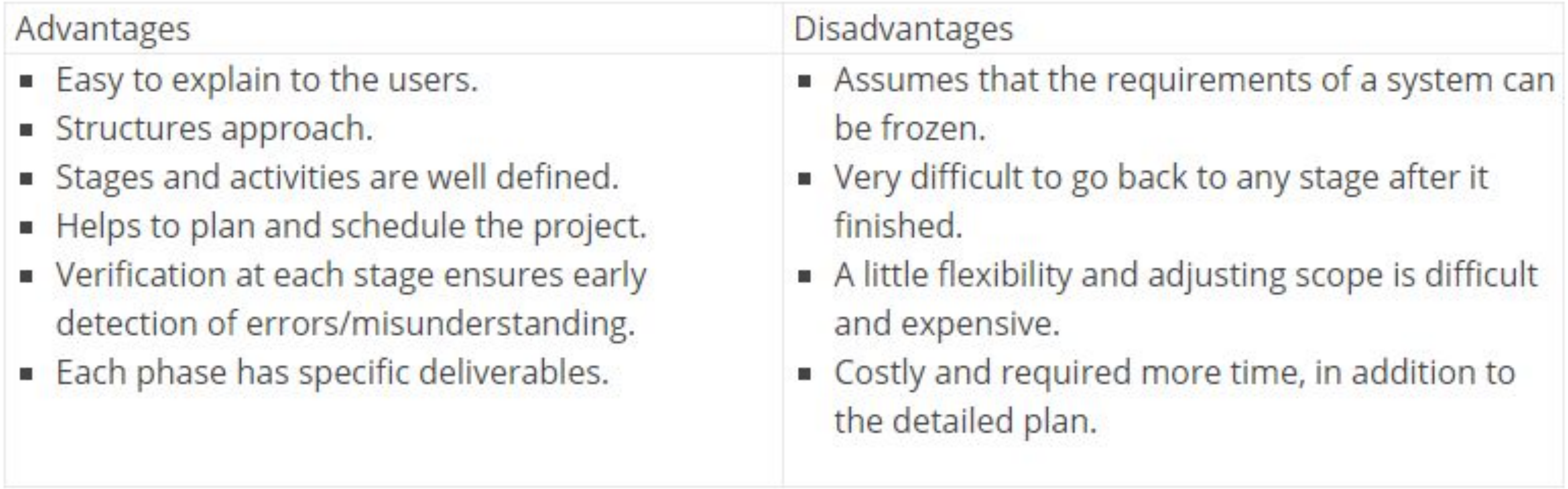
- **Implementation** - the program code is written. Good pseudocode allows the implementation stage to be relatively easy. The code is normally written in a **high-level language**.
- **Testing** - this involves testing the program under various conditions to make sure it is going to work. You need to think about what devices it could be used on and what might cause the program to crash.
- **Evolution** - the software is ready to be launched, but after it has been launched you will need to think about how the software evolves. Software needs to be maintained to ensure it works on new systems. **Smartphone apps** are constantly being maintained to make sure they work on the latest smartphones and computers.

Waterfall Model

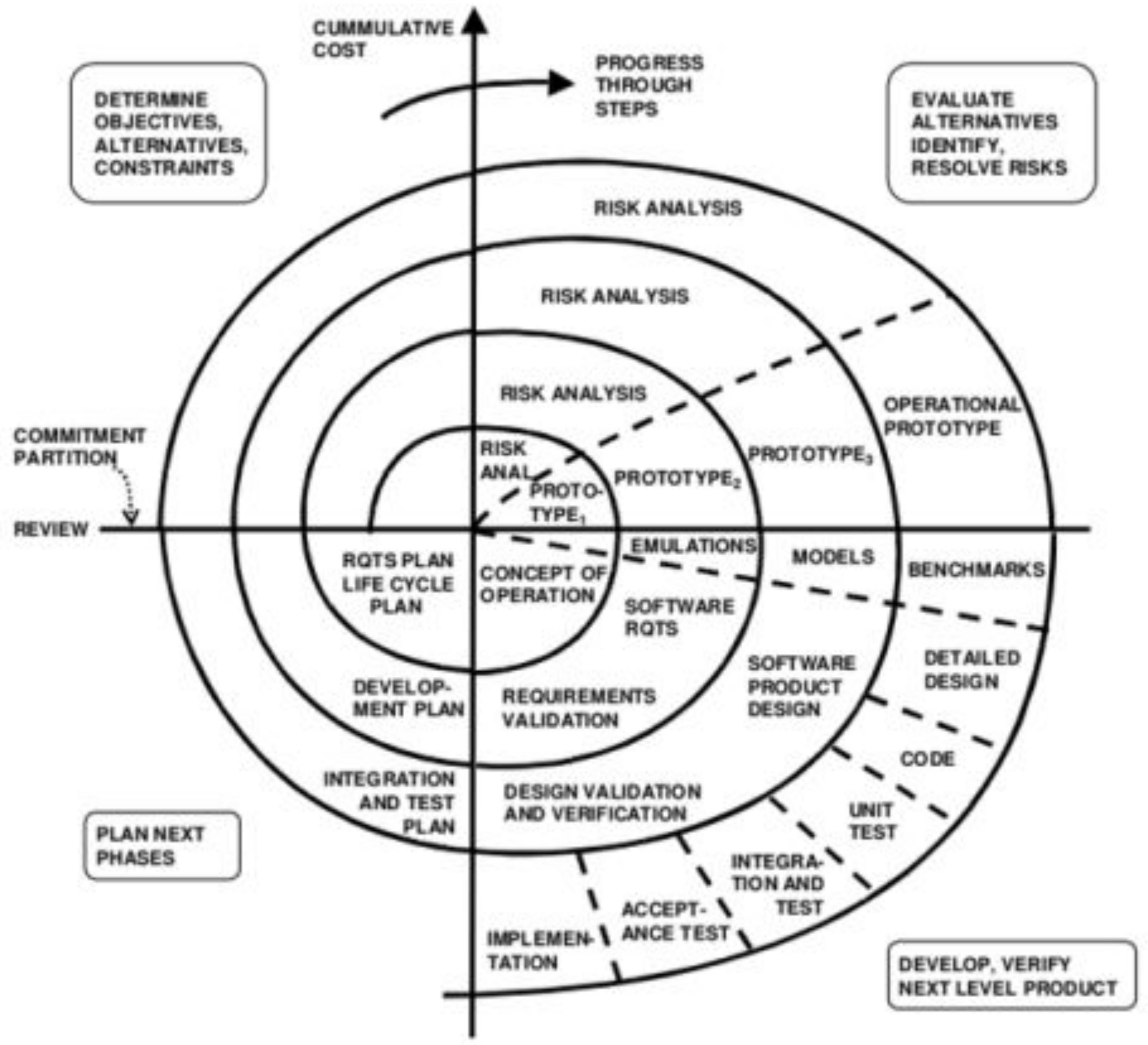


- This is a more **linear** version of the cycle. Each phase must be complete before you move onto the next. It is easy to follow this process and easy to manage. However, it is not a very flexible model. In theory, you should not need to return to a previous phase after it is completed. A final piece of software is only produced at the very end of the process.

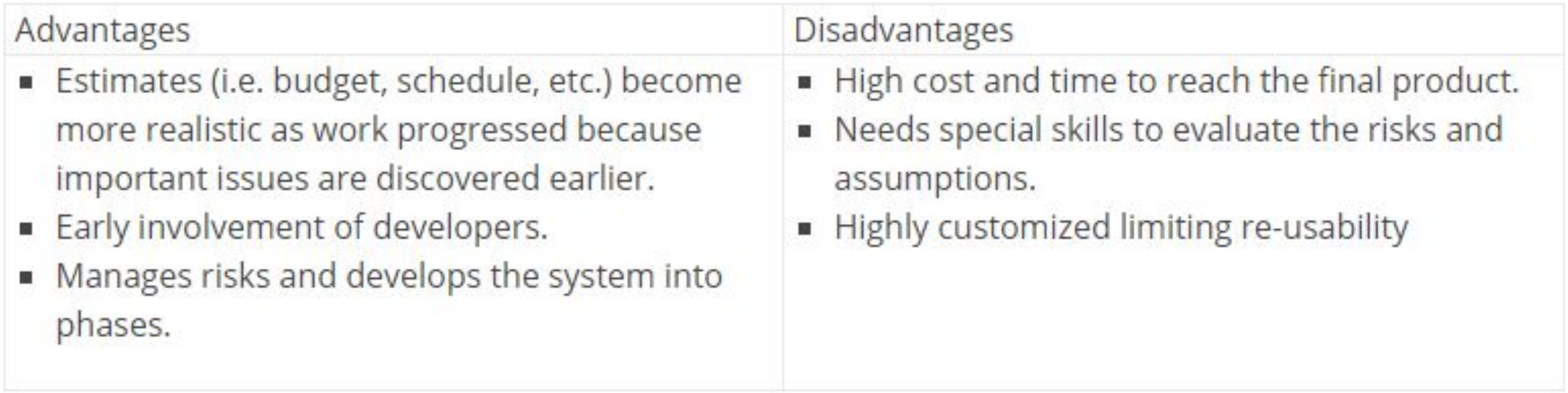
Advantages and Disadvantages



Advantages	Disadvantages
▪ Easy to explain to the users.	▪ Assumes that the requirements of a system can be frozen.
▪ Structures approach.	▪ Very difficult to go back to any stage after it finished.
▪ Stages and activities are well defined.	▪ A little flexibility and adjusting scope is difficult and expensive.
▪ Helps to plan and schedule the project.	▪ Costly and required more time, in addition to the detailed plan.
▪ Verification at each stage ensures early detection of errors/misunderstanding.	
▪ Each phase has specific deliverables.	



- The **spiral model** is a risk-driven process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.



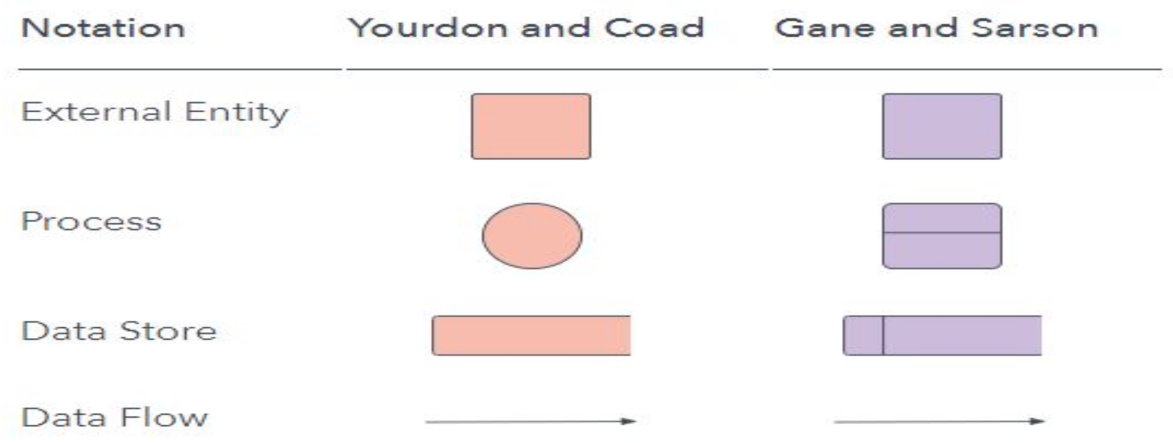
Advantages	Disadvantages
▪ Estimates (i.e. budget, schedule, etc.) become more realistic as work progressed because important issues are discovered earlier. ▪ Early involvement of developers. ▪ Manages risks and develops the system into phases.	▪ High cost and time to reach the final product. ▪ Needs special skills to evaluate the risks and assumptions. ▪ Highly customized limiting re-usability

DFD

- A data flow diagram (DFD) maps out the flow of information for any process or system. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences.

Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

- **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system.
- **Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as "Submit payment."
- **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."
- **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."


Notation
Yourdon and Coad
Gane and Sarson
External Entity
Process
Data Store
Data Flow