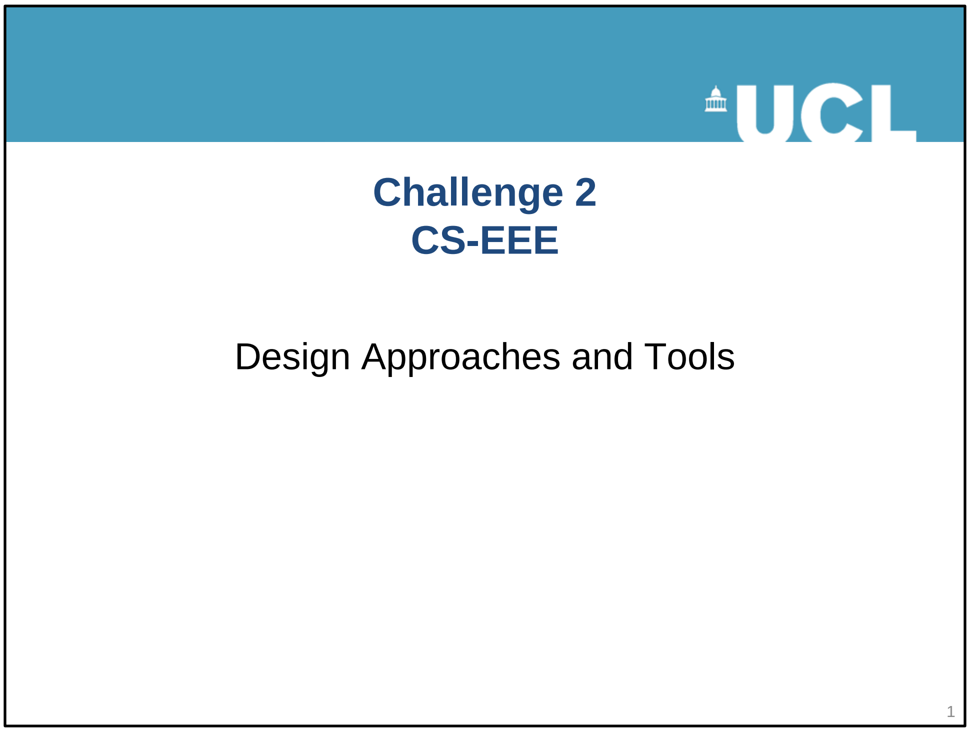
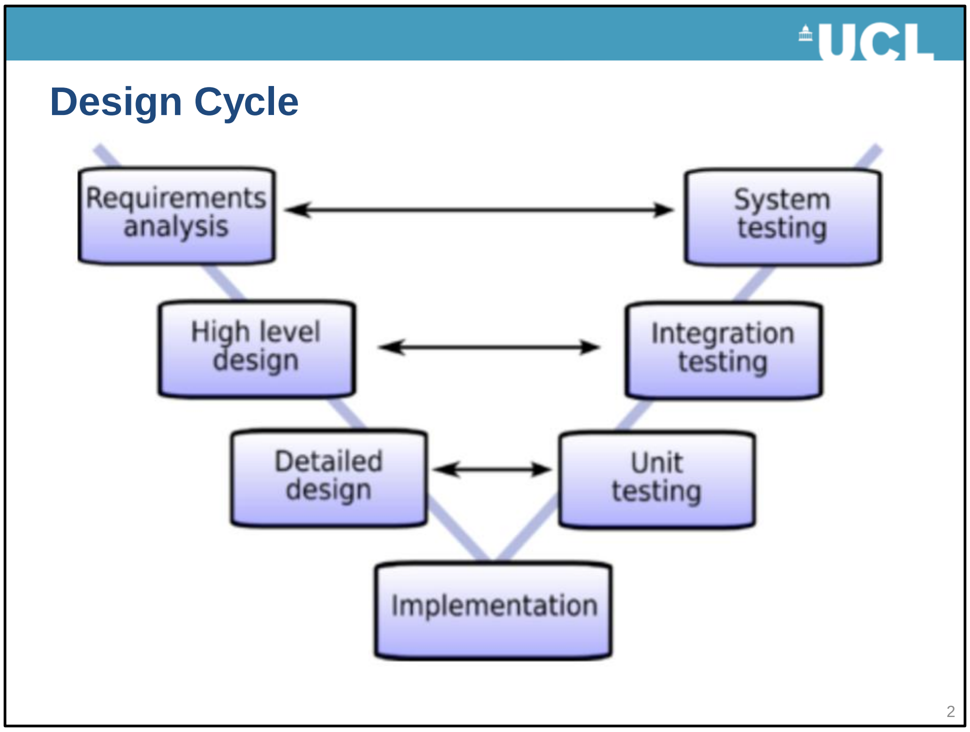


Challenge 2
CS-EEE

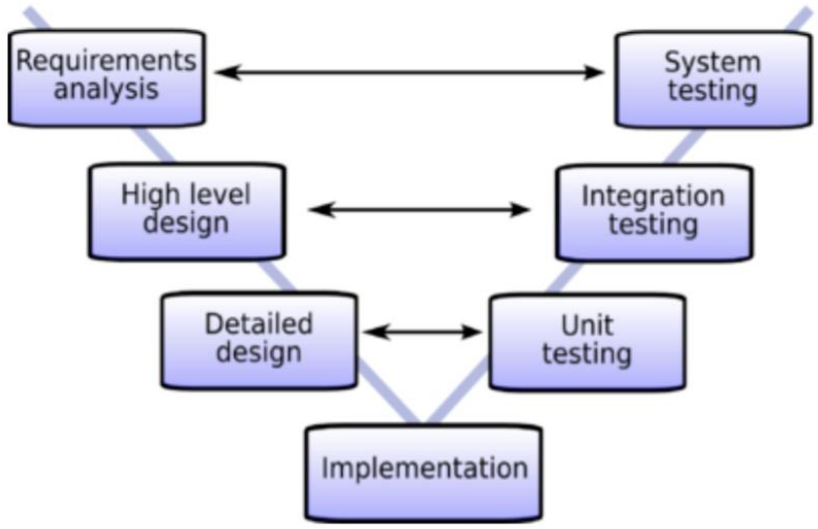
Design Approaches and Tools

1

Over the next slides, we will explore the design approaches and tools, which will help us model our bioreactor monitoring and control system.



Design Cycle



2

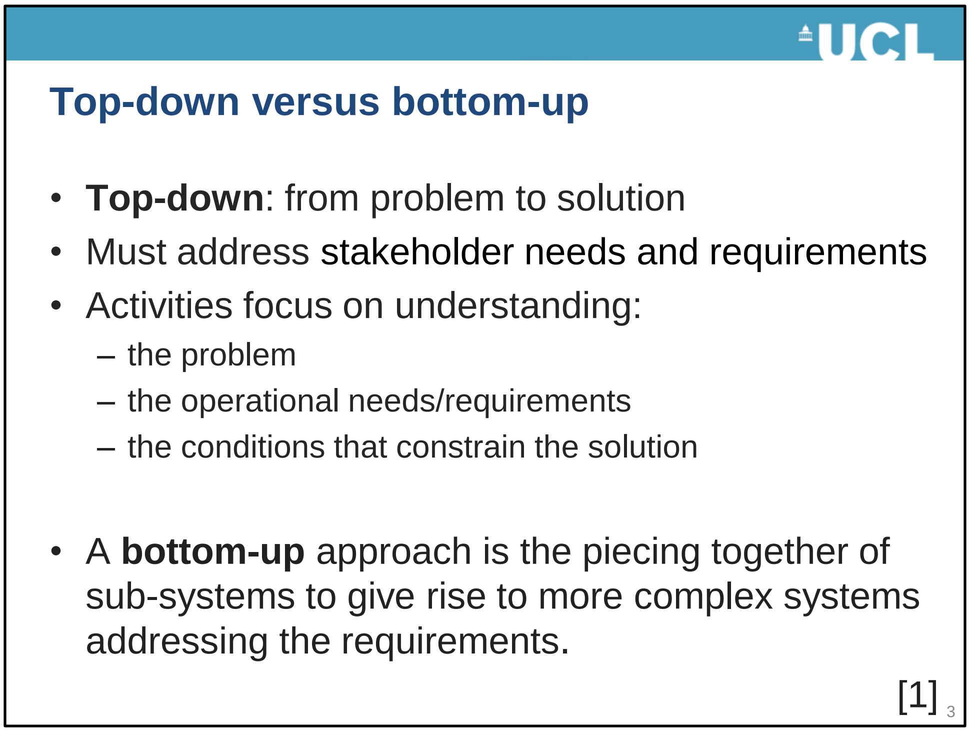
First, let us look at the typical design cycle of an engineering project. The slide presents quite a high-level, abstract view of this cycle, yet, it is generic enough to be applied to virtually any engineering system. The left-hand side lists the design stages while the right-hand side lists the testing stages. Together, these stages enable us to take a design from concept all the way through to implementation.
Starting at the top of the left-hand side, the first and most important stage of any engineering project is to identify requirements. These requirements are typically referred to as the technical specification, which define both the user and system requirements. It is the list of requirements that is used to define the subsequent objectives and tasks of the project. For example, stating that our motor is spinning, is merely a functionality but does not meet a particular specification. A specification would specify specific details about this functionality, for example, our motor has to spin within the range of 500-1500 revolutions per minute (or RPM) with an error margin of +/- 20 RPM. Furthermore, we might specify other technical aspects of the motor, for example, that is should be a direct current (DC) instead of an alternating current (AC) motor and that it should operate from a 6V power supply unit. Such an analysis of all elements of a system, is referred to as a Requirements analysis.
Once we have analysed our requirements, our next step is to develop a high-level design of our system. This high-level design provides the 'big picture' of our system showing how all the elements come together but without delving into the details. This is the aim of the next stage in the design being the detailed design. During the detailed design of a system, we typically breakdown our system in multiple,

subsystems which are more manageable, in terms of implementing and testing. It is during the detailed design process that we also identify specific functionalities for each element of the system and show how these feed into the overall system. This design cycle on the left-hand side follows a top-down approach starting with an analysis of the requirements all the way through to implementing separate blocks and elements of the system.

The testing indicated on the right-hand side follows a bottom-up approach. Specifically, once we have implemented each separate unit in our detailed design, we test it in isolation to prevent errors that might be present in other parts of the system from clouding our judgement regarding this particular unit. Unit testing also allows us to refine and optimise multiple units of the system in parallel, thereby improving efficiency and reliability. Once we have completed testing for all units, the next stage upwards involves integrating these units and carrying out integration testing. This integration testing is aimed at identifying issues or problems that might have occurred during the integration of the different units. A significant part of the integration process involves interfacing the different units between them and hence issues or problems around interfacing, are usually revealed during the integration testing phase.

Finally, once our complete, end-to-end system is operational, we can test the functionality and performance of the overall system against the requirements that we had set originally, that is to say, against the user and system specification that were defined at the start of the project.

As is evident from the slide, while the design and testing phases are carried out in a top-down and bottom-up approach, respectively, when viewed in sequence, in practice, there is horizontal interaction between each design and testing phase, with each design-testing pair in itself constituting a continuous, repetitive loop, within the overall design cycle.

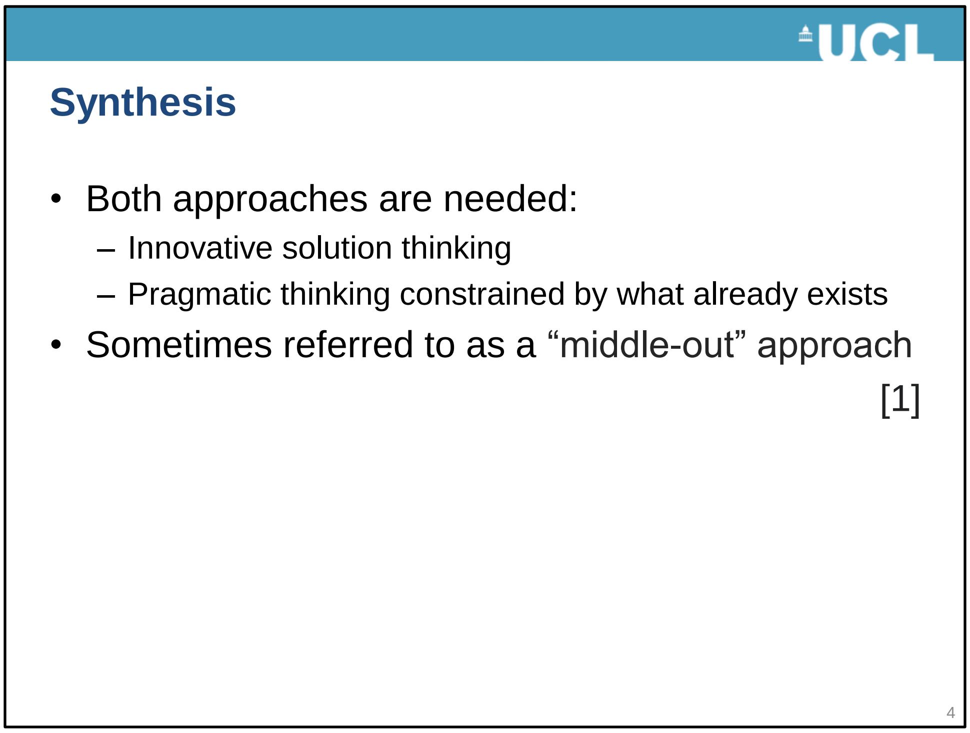

Top-down versus bottom-up

- **Top-down**: from problem to solution
- Must address stakeholder needs and requirements
- Activities focus on understanding:
  - the problem
  - the operational needs/requirements
  - the conditions that constrain the solution

- A **bottom-up** approach is the piecing together of sub-systems to give rise to more complex systems addressing the requirements.

[1] 3

Another, more generic view of the top-down and bottom-up approaches, is outlined in this slide. Specifically, the top-down approach starts with the stakeholders needs and requirements in order to understand the problem and propose a solution accordingly that fit these needs and requirements.
The bottom-up approach on the other hand focuses on the design, development and testing of individual subsystems, which when pieced together, form a more complex system, which in turn, should address the stakeholders' needs and requirements.

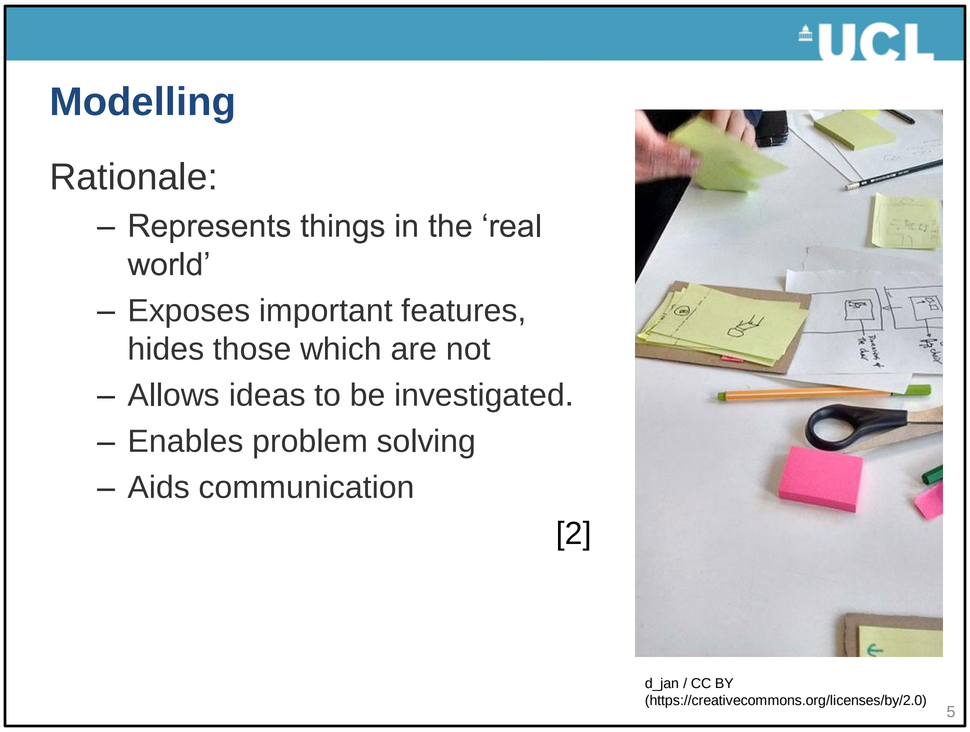

Synthesis

- Both approaches are needed:
  - Innovative solution thinking
  - Pragmatic thinking constrained by what already exists
- Sometimes referred to as a “middle-out” approach

[1]

4

In practice, both top-down and bottom-up approaches are employed in an engineering design. This approach is therefore referred to as the hybrid, or “middle-out” approach.

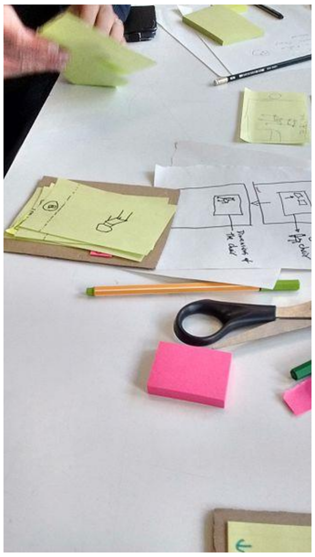


Modelling

Rationale:

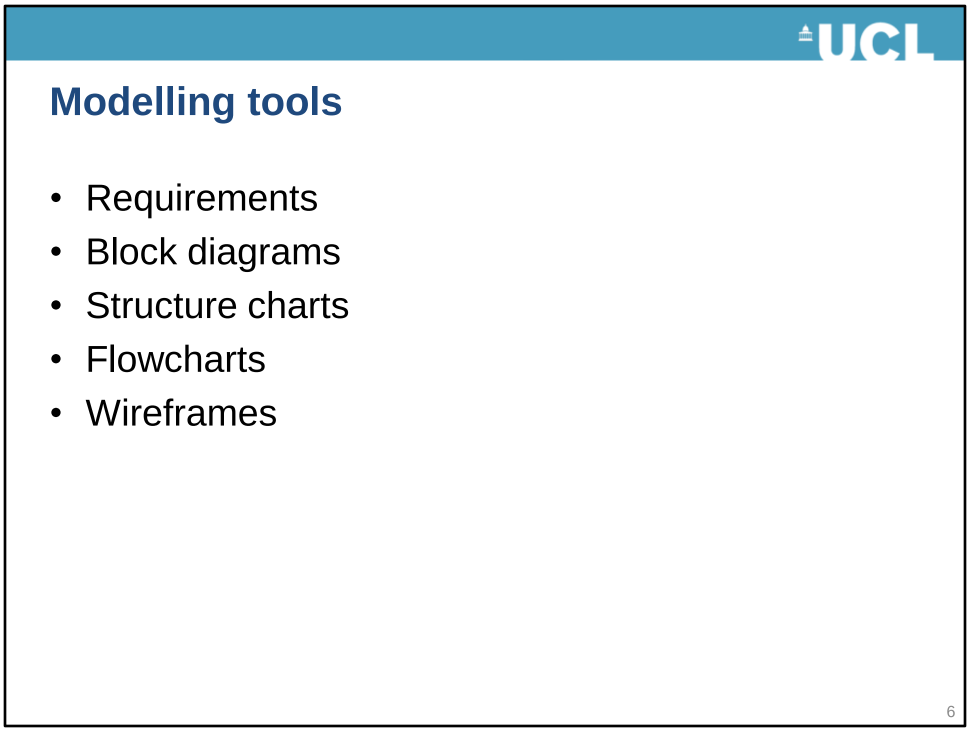
- Represents things in the 'real world'
- Exposes important features, hides those which are not
- Allows ideas to be investigated.
- Enables problem solving
- Aids communication

[2]



d_jan / CC BY (https://creativecommons.org/licenses/by/2.0)

Before embarking on the design and implementation of a system, we first need to model it. Think of this modelling stage as the initial brainstorming, which allows ideas to be explored, important features to be revealed, as well as to provide the connection between our system and the real world. As can be seen from the image in the slide, this modelling approach in its simplest, yet most powerful form, starts with pen and paper.

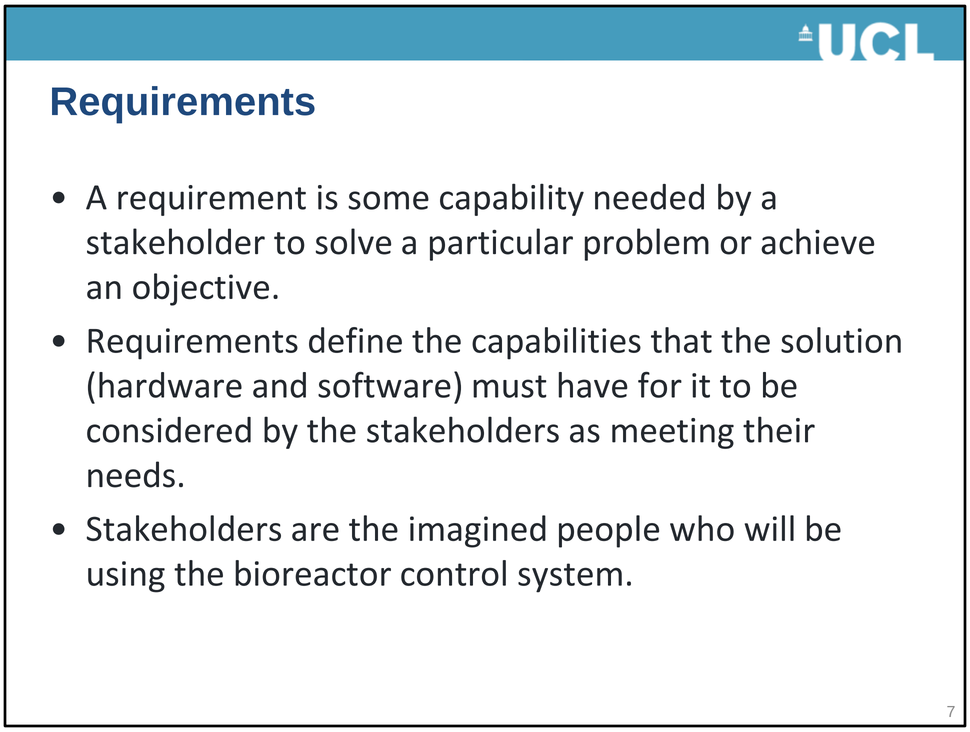


Modelling tools

- Requirements
- Block diagrams
- Structure charts
- Flowcharts
- Wireframes

6

Let us now move on to explore the specific tools that can help us model a system. The slide lists six key categories of tools, which we will look at in further detail over the coming slides. These six tools include requirements, block diagrams, structure charts, flowcharts, wireframes and schematic or circuit diagrams.

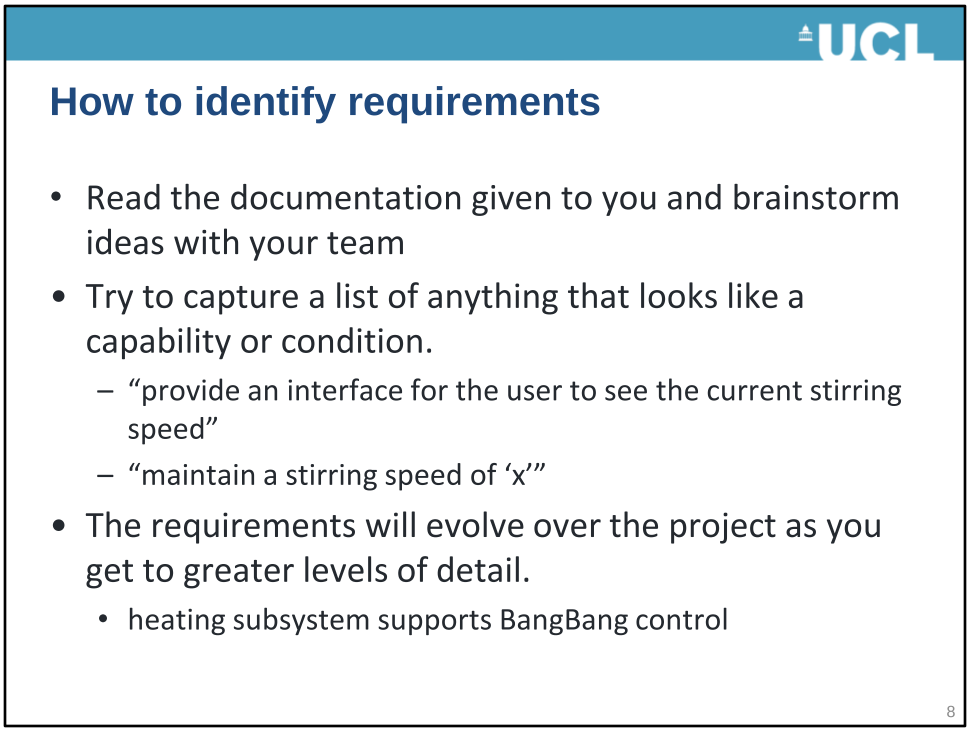


Requirements

- A requirement is some capability needed by a stakeholder to solve a particular problem or achieve an objective.
- Requirements define the capabilities that the solution (hardware and software) must have for it to be considered by the stakeholders as meeting their needs.
- Stakeholders are the imagined people who will be using the bioreactor control system.

7

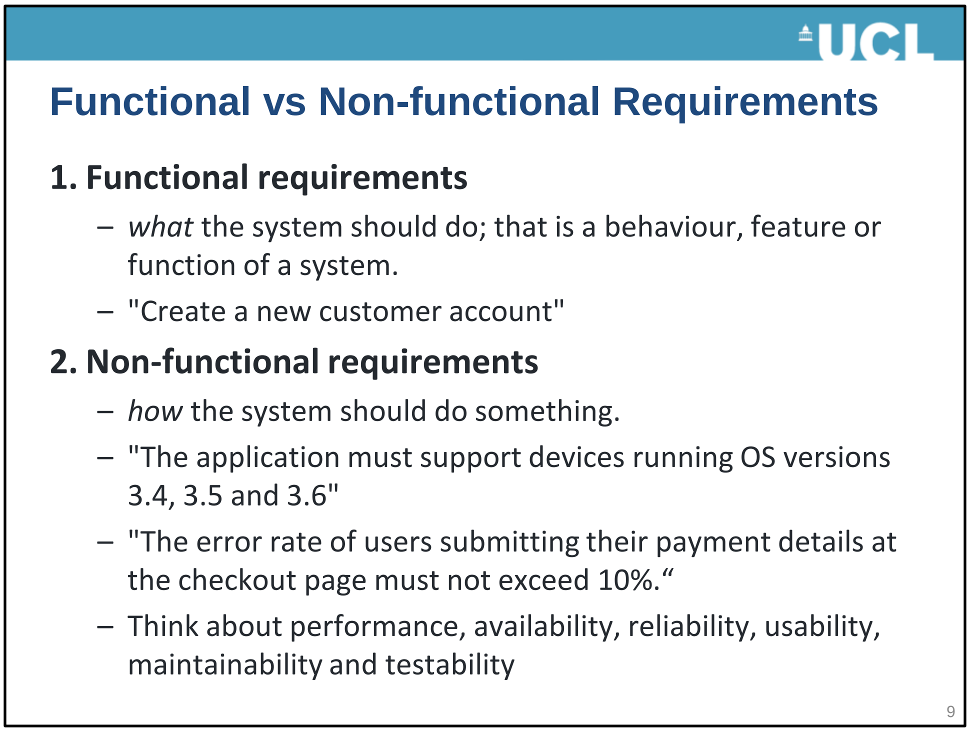
- Get used to developing and maintaining a list of requirements that can evolve in the early stages of your project.
- Think carefully about what the stakeholders have asked for. Don’t invent requirements.
- In the context of this project, the stakeholders are the imagined people who will be using the bioreactor control system.



How to identify requirements

- Read the documentation given to you and brainstorm ideas with your team
- Try to capture a list of anything that looks like a capability or condition.
  - “provide an interface for the user to see the current stirring speed”
  - “maintain a stirring speed of ‘x’”
- The requirements will evolve over the project as you get to greater levels of detail.
  - heating subsystem supports BangBang control

8



Functional vs Non-functional Requirements

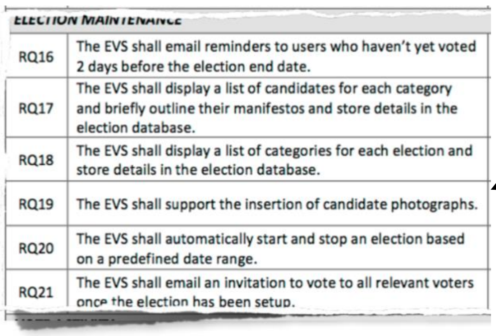
1. Functional requirements

- *what* the system should do; that is a behaviour, feature or function of a system.
- "Create a new customer account"

2. Non-functional requirements

- *how* the system should do something.
- "The application must support devices running OS versions 3.4, 3.5 and 3.6"
- "The error rate of users submitting their payment details at the checkout page must not exceed 10%."
- Think about performance, availability, reliability, usability, maintainability and testability

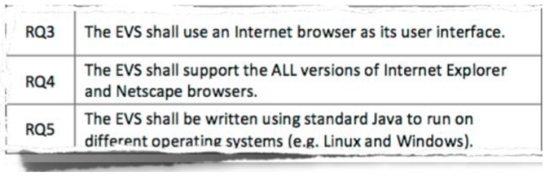
Recording requirements – 2 tables



ELECTION MAINTENANCE

RQ16	The EVS shall email reminders to users who haven't yet voted 2 days before the election end date.
RQ17	The EVS shall display a list of candidates for each category and briefly outline their manifestos and store details in the election database.
RQ18	The EVS shall display a list of categories for each election and store details in the election database.
RQ19	The EVS shall support the insertion of candidate photographs.
RQ20	The EVS shall automatically start and stop an election based on a predefined date range.
RQ21	The EVS shall email an invitation to vote to all relevant voters once the election has been setup.

Functional

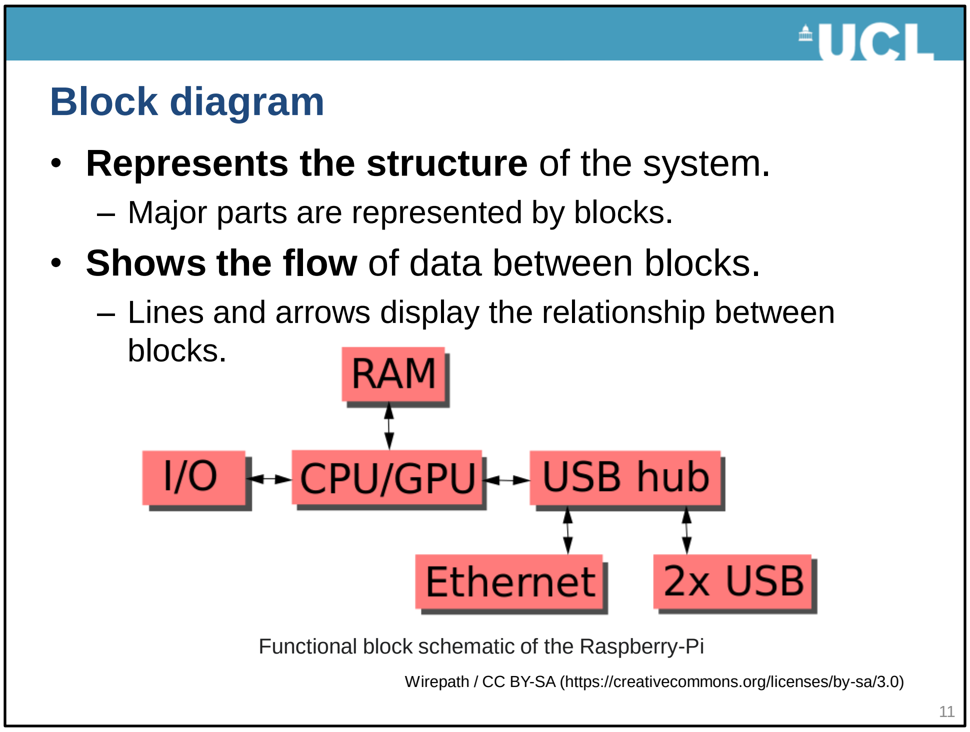


RQ3	The EVS shall use an Internet browser as its user interface.
RQ4	The EVS shall support the ALL versions of Internet Explorer and Netscape browsers.
RQ5	The EVS shall be written using standard Java to run on different operating systems (e.g. Linux and Windows).

Non Functional

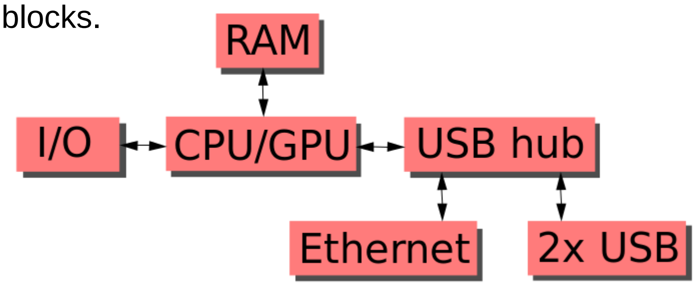
10

A simple table for each type of function is good enough.
Requirements should be numbered. Tables should be kept up-to-date throughout the project.



Block diagram

- **Represents the structure** of the system.
  - Major parts are represented by blocks.
- **Shows the flow** of data between blocks.
  - Lines and arrows display the relationship between blocks.

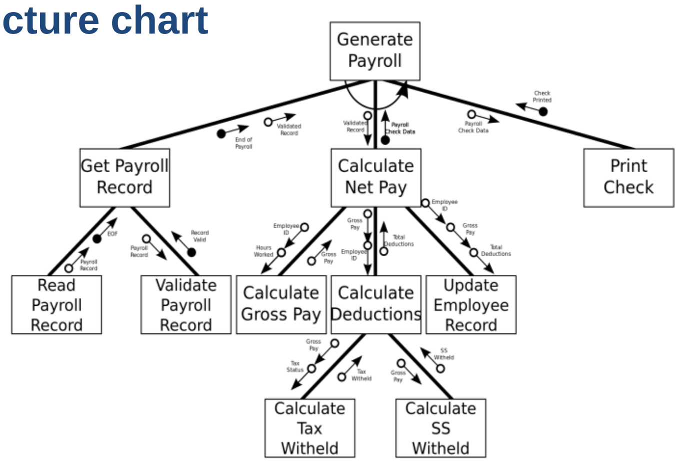


Functional block schematic of the Raspberry-Pi

11

A block diagram is a drawing illustration whose major parts are represented by blocks. The lines and arrows display the relationship between blocks and show whether the transfer of information is unidirectional or bidirectional. Block diagrams serve to describe the architecture of a system along with its workflows and processes at a high, abstract level. This is the reason block diagrams are typically referred to as black boxes since they do not reveal the inner workings of the system itself.

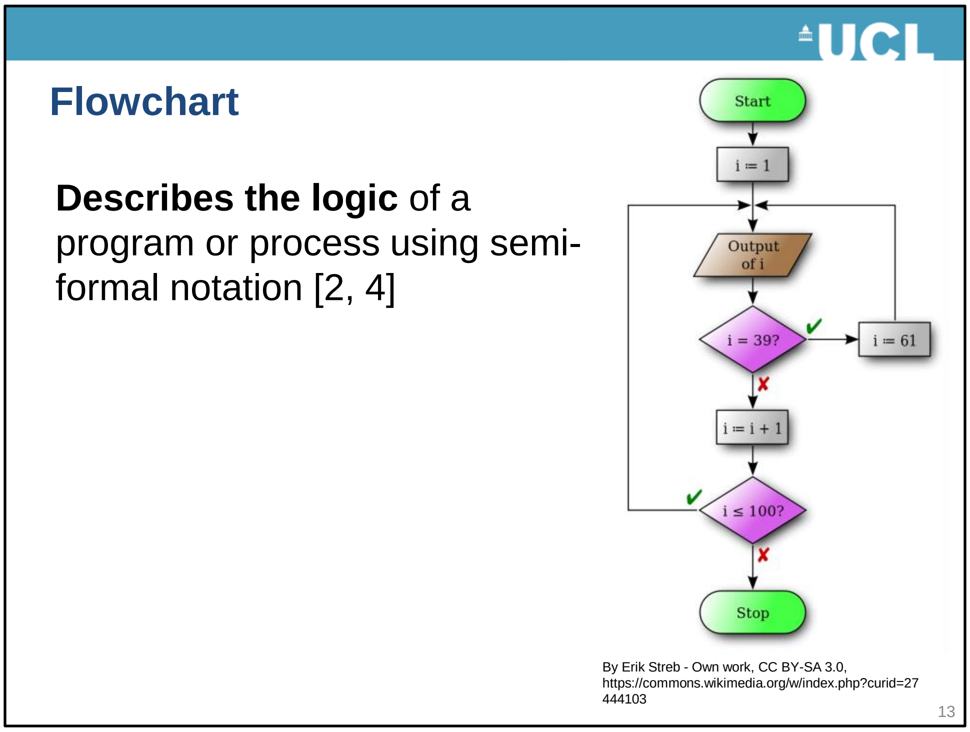
Structure chart



By Structured_Chart_Example.jpg: Sandia National Laboratories derivative work: Pluke (talk) - Structured_Chart_Example.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=16283788

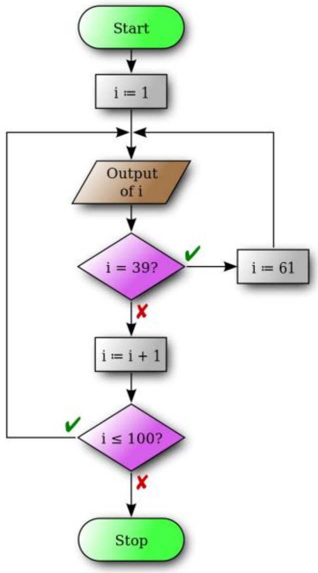
- **Breaks down a system** into its lowest manageable parts [3]

A structure chart uses blocks and arrows like a block diagram but encourages a top-down design approach. Each block is a module which represents a process or task in the system. Two types of arrows are drawn. The arrows with the empty circles represent the flow of data between modules. The arrows with the filled in circles represent the flow of control between modules.



Flowchart

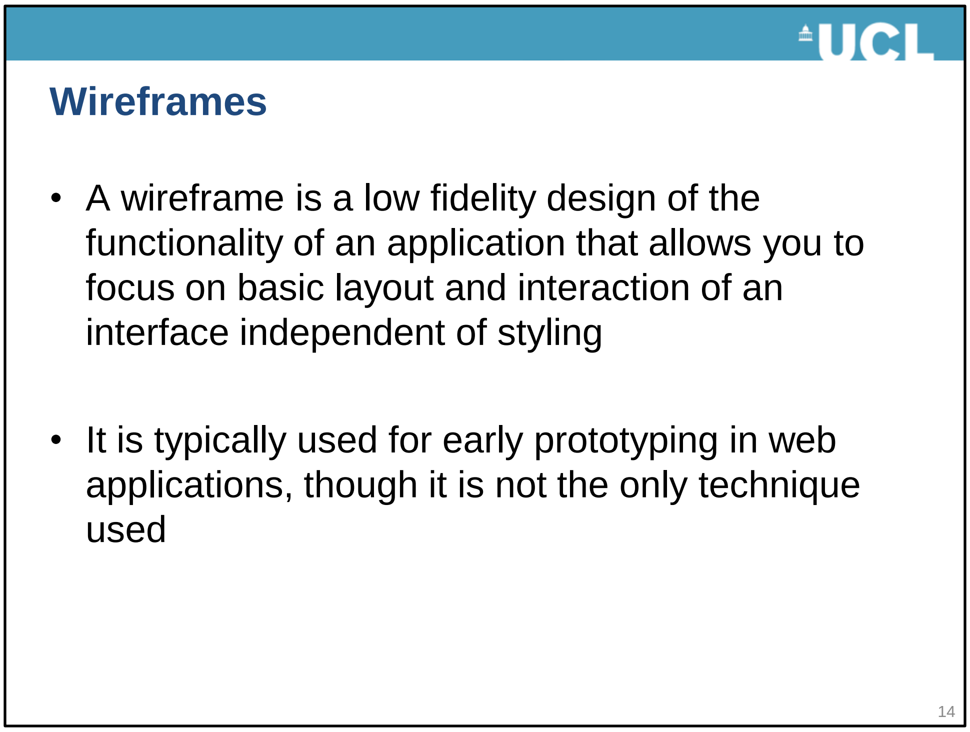
Describes the logic of a program or process using semi-formal notation [2, 4]



By Erik Streb - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=27444103

13

A flowchart also uses blocks, lines and arrows. Contrary to block diagrams and structure charts, however, a flowchart describes the actual logic of a system, program or process. That is to say, a flowchart, represents the actual sequence of steps and decisions required to carry out a process. As seen in the image in this slide, the oval represents the start or end of a process, the rectangle represents a specific subprocess within the process, the parallelogram represents input or output, while the diamond indicates a decision.



Wireframes

- A wireframe is a low fidelity design of the functionality of an application that allows you to focus on basic layout and interaction of an interface independent of styling
- It is typically used for early prototyping in web applications, though it is not the only technique used

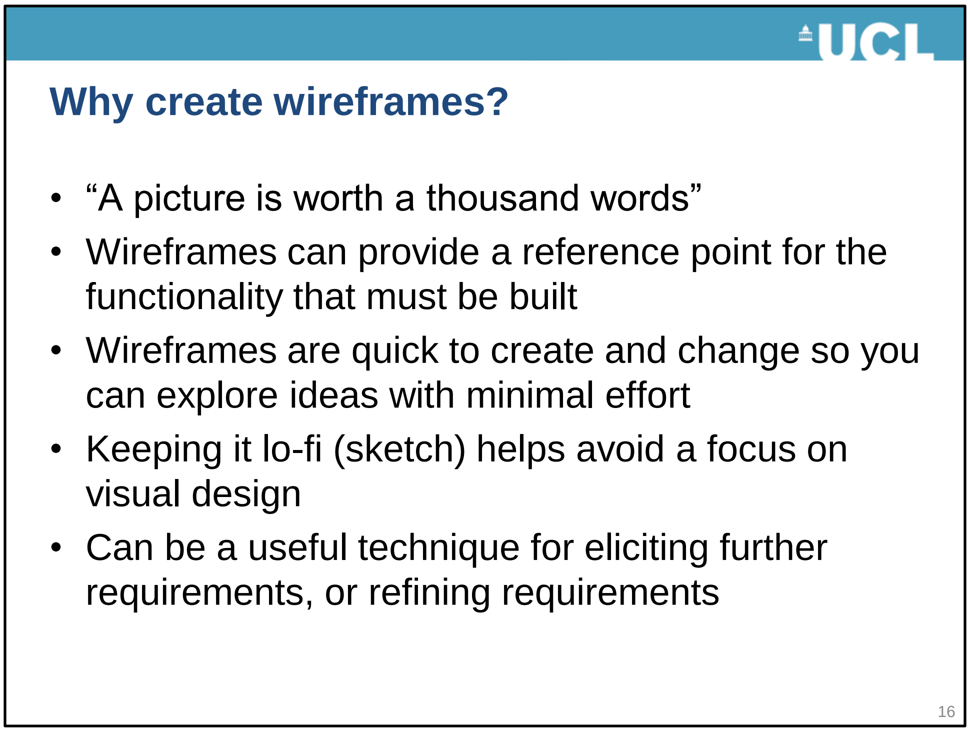
14

Equivalent of an architect’s floor plan in building houses

Characteristics of a wireframe

- Pencil/paper sketch
- Created before styling or coding
- Provides an outline of the structure and layout
- Presents the main information and application flows

"Mobile First" by Sauce Babilonia is licensed under CC BY-NC 2.0

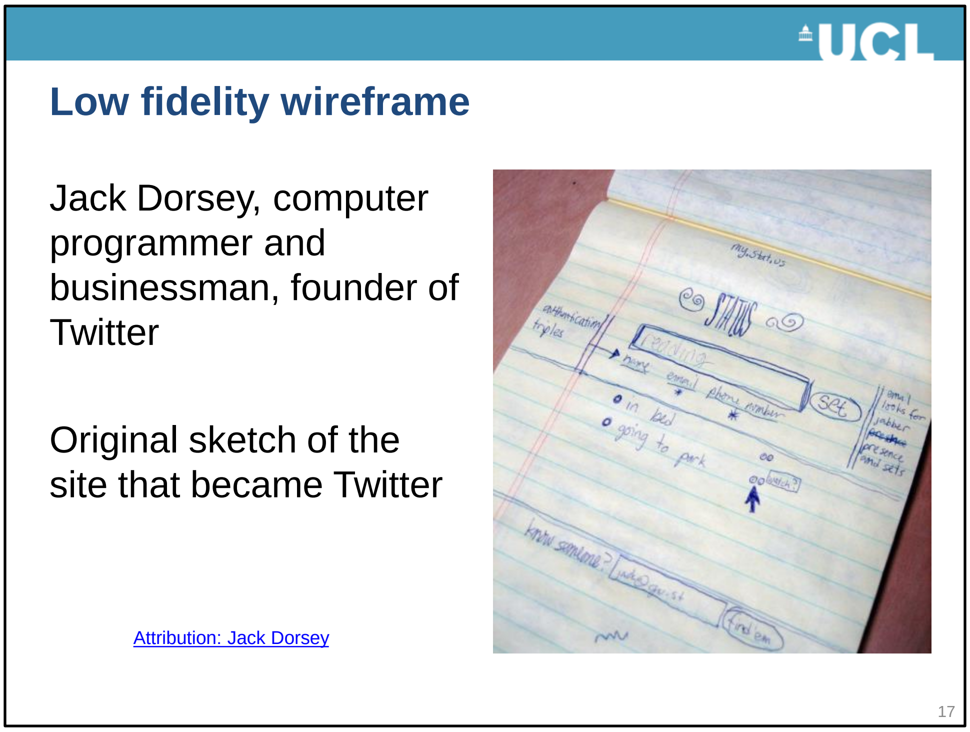

UCL

Why create wireframes?

- “A picture is worth a thousand words”
- Wireframes can provide a reference point for the functionality that must be built
- Wireframes are quick to create and change so you can explore ideas with minimal effort
- Keeping it lo-fi (sketch) helps avoid a focus on visual design
- Can be a useful technique for eliciting further requirements, or refining requirements

16

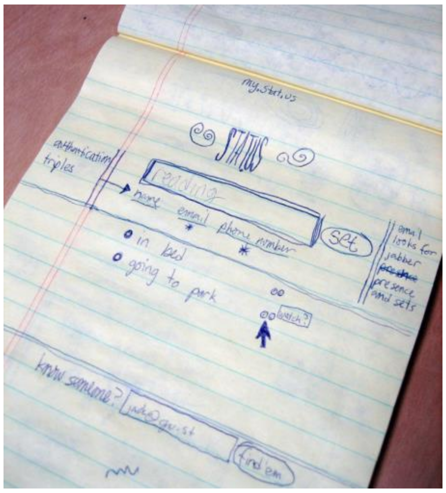
“A picture is worth a thousand words” - wireframes clearly communicate design decisions to stakeholders and teammates and, by looking at a wireframe, they should have a good idea of what screens an app will have



Low fidelity wireframe

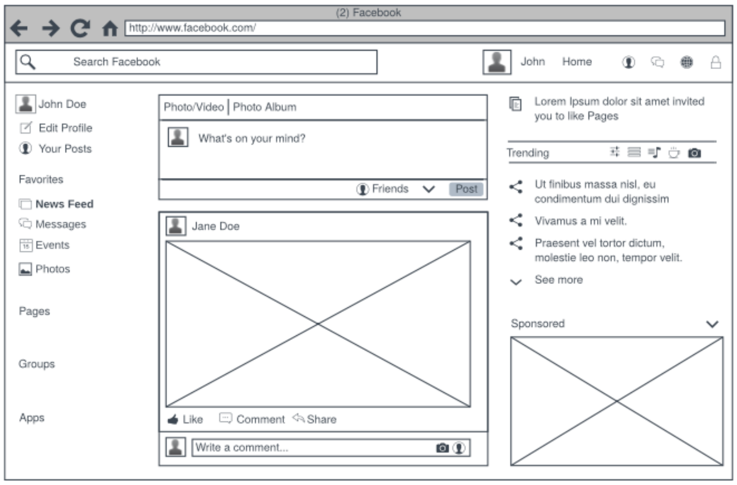
Jack Dorsey, computer programmer and businessman, founder of Twitter

Original sketch of the site that became Twitter

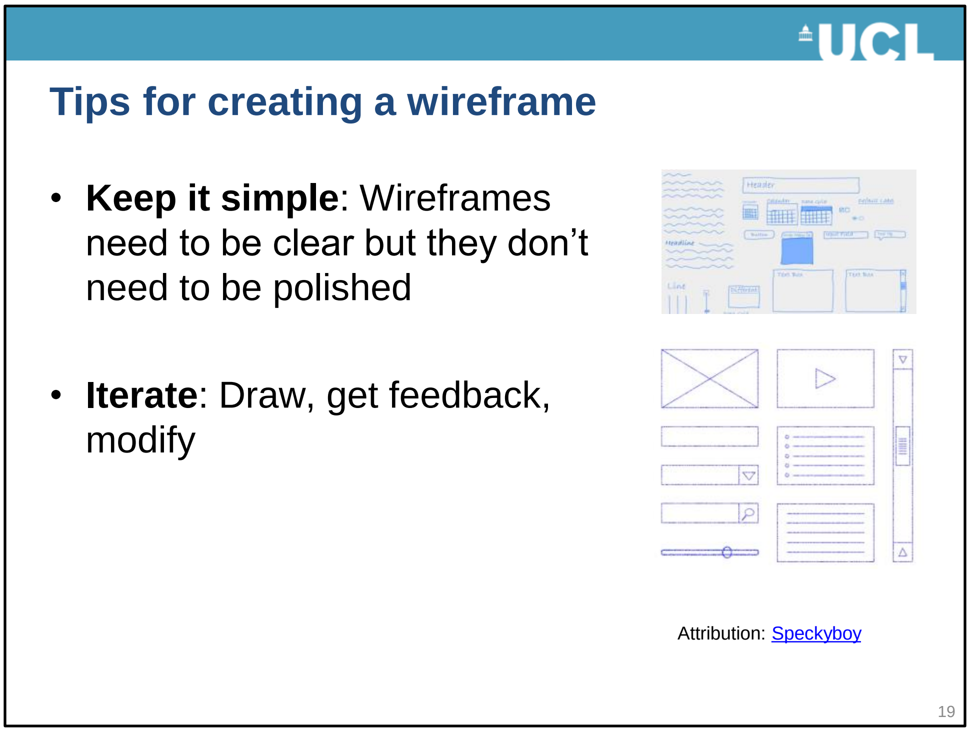


Attribution: Jack Dorsey

Digital wireframes

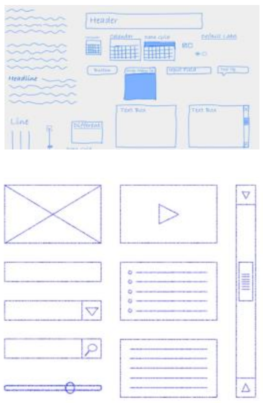


Use tools such as Lucidchart to create wireframes

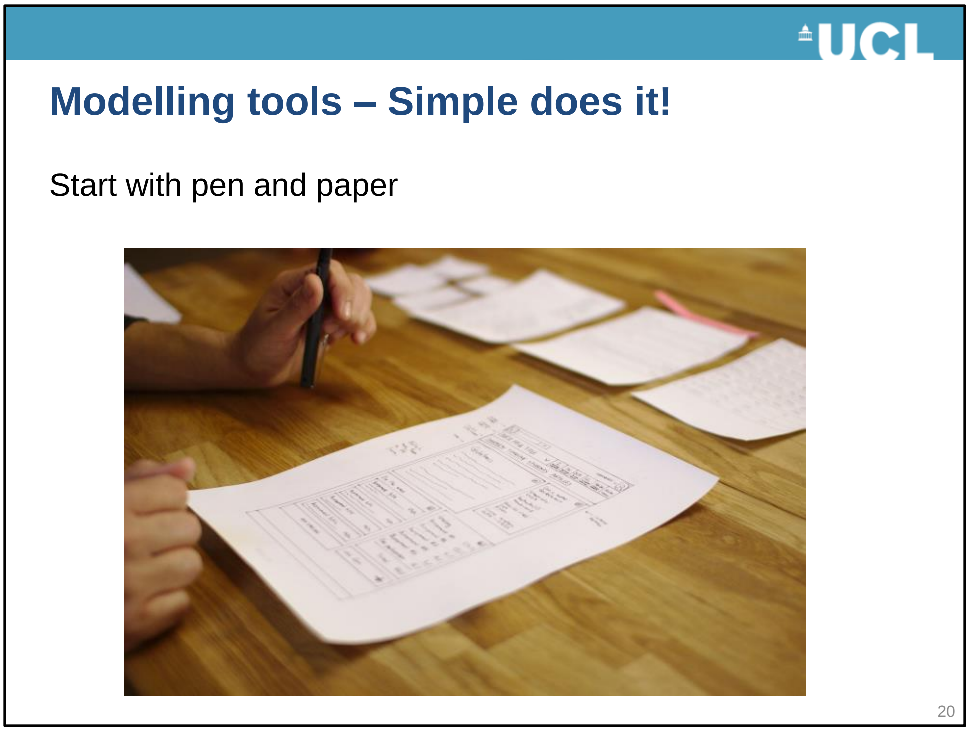


Tips for creating a wireframe

- **Keep it simple**: Wireframes need to be clear but they don't need to be polished
- **Iterate**: Draw, get feedback, modify



Attribution: Speckyboy

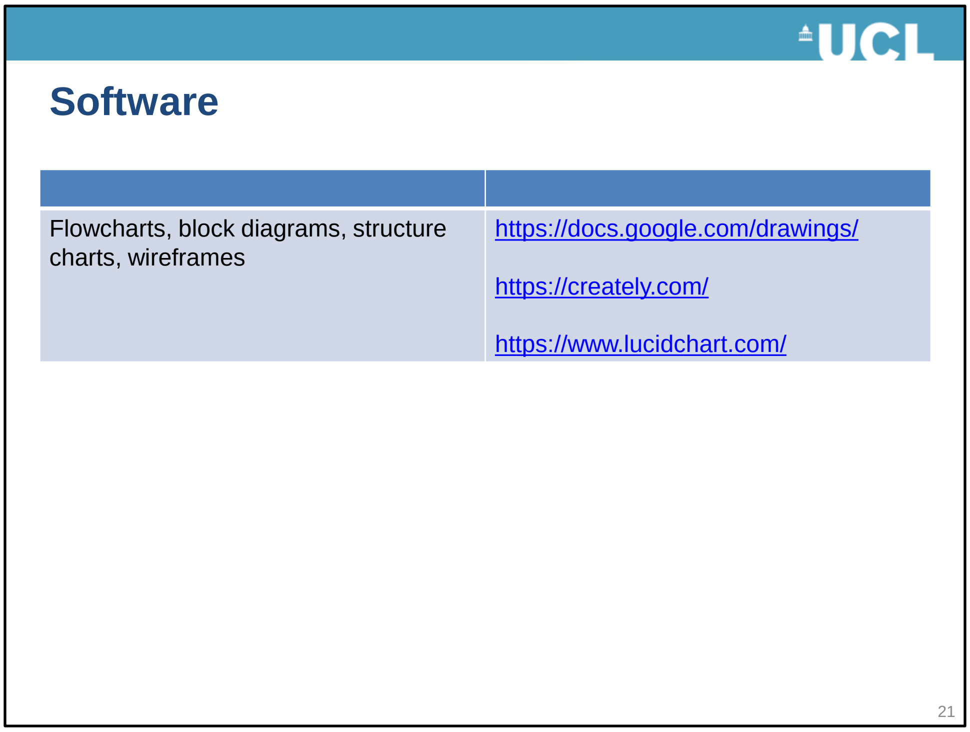


Modelling tools – Simple does it!

Start with pen and paper

20

Computer-aided design tools, or CAD for short, are useful for creating professional diagrams and facilitate collaboration with our peers. However, the easiest and most effective approach to start modelling a system is by using pen and paper. Pen and paper allows us to sketch our ideas in different ways, over and over again, using an unparallel level of flexibility and free-style approach that cannot be offered by any CAD tool.

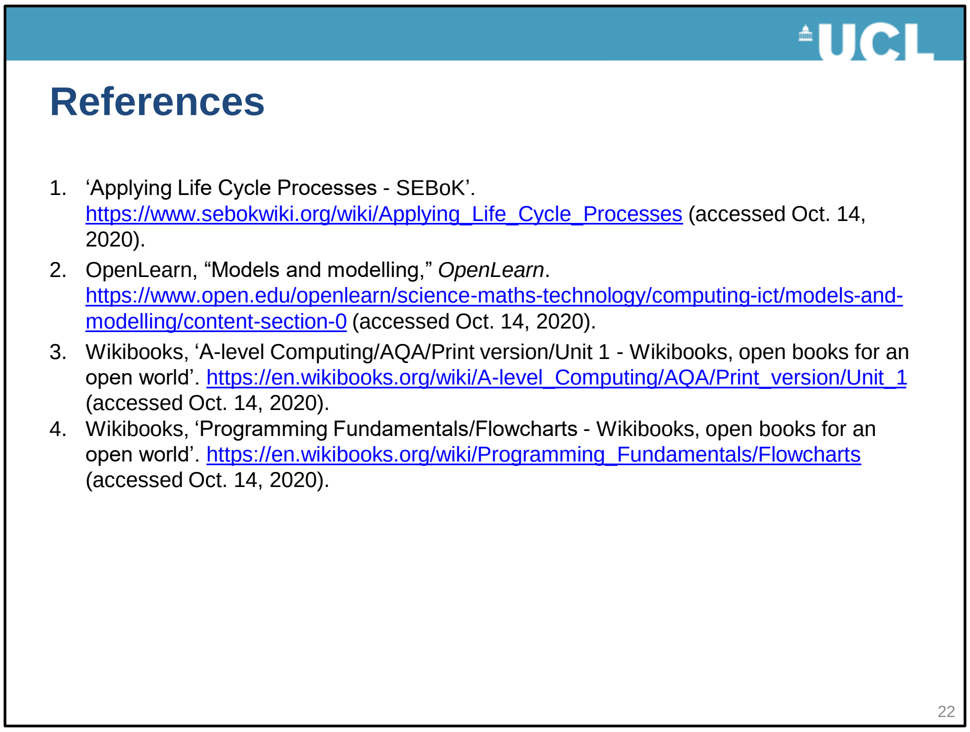


Software

Flowcharts, block diagrams, structure charts, wireframes	https://docs.google.com/drawings/ https://creately.com/ https://www.lucidchart.com/

21

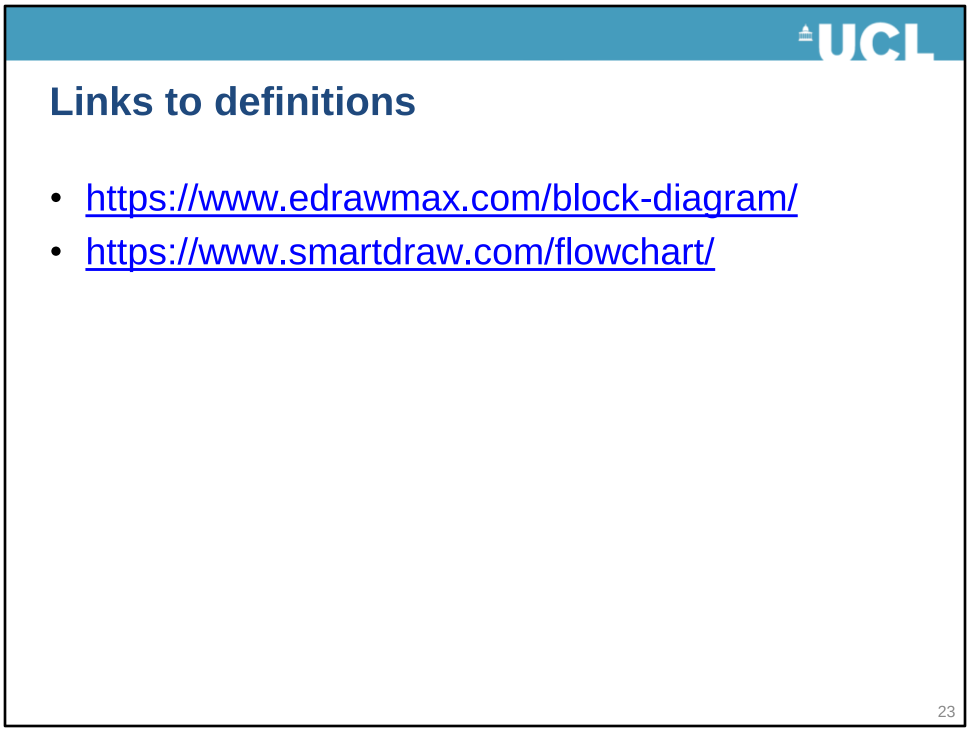
Here is a list of CAD tools that you can use to help you model your systems. At the time of writing, these tools were offered free of charge.



References

1. ‘Applying Life Cycle Processes - SEBoK’. https://www.sebokwiki.org/wiki/Applying_Life_Cycle_Processes (accessed Oct. 14, 2020).
2. OpenLearn, “Models and modelling,” *OpenLearn*. https://www.open.edu/openlearn/science-maths-technology/computing-ict/models-and-modelling/content-section-0 (accessed Oct. 14, 2020).
3. Wikibooks, ‘A-level Computing/AQA/Print version/Unit 1 - Wikibooks, open books for an open world’. https://en.wikibooks.org/wiki/A-level_Computing/AQA/Print_version/Unit_1 (accessed Oct. 14, 2020).
4. Wikibooks, ‘Programming Fundamentals/Flowcharts - Wikibooks, open books for an open world’. https://en.wikibooks.org/wiki/Programming_Fundamentals/Flowcharts (accessed Oct. 14, 2020).

Here is a list of references that were consulted for this presentation.



Links to definitions

- https://www.edrawmax.com/block-diagram/
- https://www.smartdraw.com/flowchart/

23

And here are links to definitions for each of the modelling tools that were presented in this slide deck.

This concludes this recording. Thank you for watching and good luck!