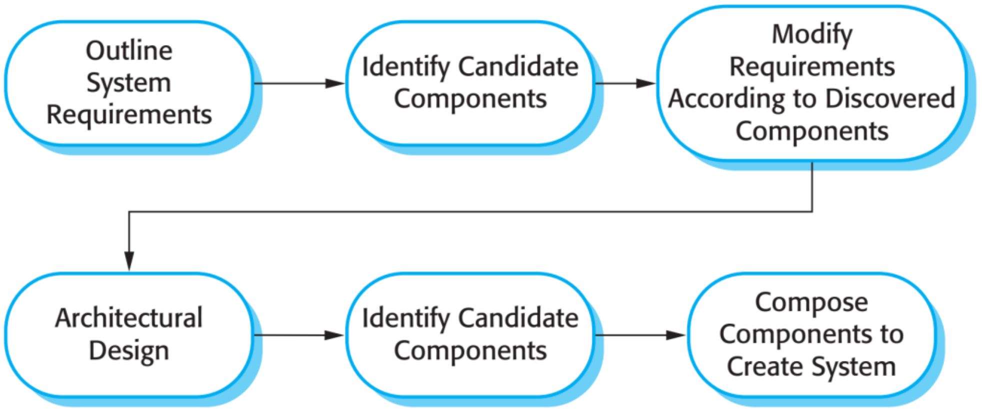
Workshop on Software Reuse and Component Based Software Engineering

In the previous workshop, you have created UML diagrams to model your system. This gave you an idea of what the system will look like in abstract form.

In this workshop, you will need to go back to the **components diagram** and **class diagram** you have created and develop them further with software *reuse* and *CBSE* in mind.

Development with reuse

The following diagram shows the typical workflow of a Component Based Software System with Reuse, i.e. that uses off-the-shelf components. This can include commercial components or open-source ones, or a combination of both.


Outline System Requirements
Identify Candidate Components
Modify Requirements According to Discovered Components
Architectural Design
Identify Candidate Components
Compose Components to Create System

With the work you have done in the previous weeks, you have already outlined your system requirements and identified the components in abstract form. Now it is time to ground the components by identifying available parts (libraries, services, softwares, etc.) that you may actually integrate into your system.

1. Based on the components diagram, search the internet available solutions that implement each specific components. You can search on GitHub.com for open source solutions, but also look for commercial ones. Moreover, you can use cloud services and/or programs/libraries to run on your system.
2. For each component required, list all the off-the shelf softwares you have found and identify their **'requires interface' and 'provides interface'**. This is usually specified in their API.
3. For each component select the "best" solution based on the functionalities required and the compatibility with the other components in your system; but take into account considerations such as how many users the software have, if it is still maintained or not, their license, and generally favour open-source solutions. NOTE: it is fine if one or more components do not have any suitable candidate solutions, you will deal with it later.
4. Now that you have a list of potential components, look back at your **components diagram**. Do you need to modify the architecture of your system to accommodate the interface of your components? If yes, do it!
5. Document all the process in the previous steps in a Word document named 'components.docx' and save it in your repository for the assignment. Include the solutions you have found, which you have

selected and why, and an updated version of your system components diagram (in case you have modified it).

Software Design

From the previous section, you now probably have some components that are left without a solution for implementation.

1. For each of these, design their **class diagram** by specifying the objects (as in OOP objects to implement). If a class diagram for that component already exists from last week, then you can start from that.
2. Update the class diagrams to make sure they interface with the off-the-shelf components and with one another.
3. Now consider the off-the-shelf components; can they all be integrated seamlessly within the system, or do they need some "glue code"? Do any need some "wrapper code" instead?
4. Update the **class diagrams** and **components diagrams** to integrate your glue and wrapper components as appropriate.
5. Like before, document all these steps in the same word document as before and also include any modifications to the diagrams, with the appropriate justification for each modifications.