Where did I start to move from Analysis to design?

Where did I go from what to do to how to do it or which object does what?

I have decided that Interfaces, interface controls, event handlers Messages, parameters, and output controls are part of the objects to use for describing interactions in the application/system

So when I am writing uses case stories I should include them freely, they help to draw the interaction diagrams of the story

With interactions down, we can establish the architecture of the application. All these are design activities

Details of processing data and messages to output are obtained during the requirement gathering and analysis stage, but during the design stage you assign them as responsibilities to the classes that are being generated.

The summary of the processing activities can be abstracted into high level methods which can be subdivided into smaller methods, (which may even be found in other classes).

So there is actually a stage of detailed study in analysis, it falls short of design, in that it describes the process of the existing system rather than synthesize the processes of the proposed system howbeit in details.

Analysis tries to understand the existing system, how come there is a detailed study, if it is not analysis? I come to realize that analysis tries to get the big picture; therefore it focuses on why things are being done, rather than what is being done and who is doing it. Focusing on the reason for doing things helps us to understand what the system is trying to achieve which also means we wish to understand the problem the system is trying to solve. You can actually get a sufficient understanding of the big picture during preliminary study, and that is why you are able to come up with a problem definition and proposal, but the detailed study is going to be used during design.

Detailed study is going to be used during design, since we will need to know the details of how information is processed during program design. You move from analysis to design when you start using the information you gathered from the existing system to synthesize the new system. You write stories for the use cases develop interactions and establish the architecture. You then go into elaboration (details) in order to get into program design. Use data structures, detailed description of processes and elaboration to formulate algorithms. Use decision trees to simplify complex branching

There are other forms of design. The details obtained from the analysis process can be used to design databases; you can know why the system needs to keep information, therefore determine the information that it needs to keep (relevant information). Based on the interactions and the system requirement you can determine the data entities, and the relationships between them

You can do user experience design from the use cases and interactions, and considering the user. Can user find what they need, do users need to go through one or 2 steps to accomplish a task. What kind of data should it be that the user is viewing while working on a given task? Which task should lead to other tasks/ interfaces? (Horse before cart)

You can do GUI design based on UX design. Fonts (face and size), colour combination, padding, margins and colourful pictures/backgrounds

**Program Design**

Find some automation solution for some coding techniques like String encoding, dot operator, string addition (recursively), etc.

In program design, I find algorithms more useful for describing sequence (you do not have to waste a box each operation, and the flow is quite natural. For branches I find flowcharts more explicit. I combine algorithm with flow chart by combining several sequence of steps into a single box and using decision symbol to go from operation to operation.

Iterations can be treated as a single combination of sequence and branches repeated over and over. Focus on processing a single iteration, and then what needs to change in the next iteration. While using data to drive program design, you are better of using algorithms to describe what happens to each part of the data structure before converting to actual programs

Algorithms will lend themselves to elaboration, and that is how you go from concepts to actual code

**Test Driven programs**

The trick is to get into the code at some point having all the data that will normally be available, and getting out with the result/output redirected to areas of no side effect. For most other programming, the key is to test functions (one entry point one exit point). For php you do get some weird entry points with request to pages. The idea remains the same, open the entry point from a test project, having created the data required, redirect the output, just before entry (so you do not tamper with the actual code) and restore the redirection after the code has executed

You will also need some ways of constructing the input, and validating the output, all which should be part of the test kit. A good idea is to be able to generate the data from live application, thus eliminating the possibility of encoding error. If the data gets its error from the marshalling of live data, then you know that the error is from the program rather than your encoding process.