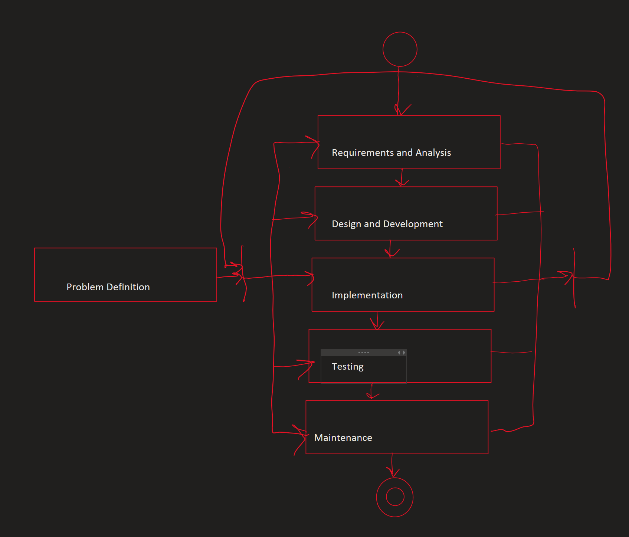
15-4 Draw a UML activity diagram describing the dependency between activities for a life

cycle in which requirements, design, implementation, test, and maintenance occur

concurrently. (This is called an evolutionary life cycle.)



15-6 In project management, a relationship between two tasks is usually interpreted as a

precedence relationship; that is, one task must complete before the next one is initiated.

In a software life cycle, a relationship between two activities is a dependency; that is,

one activity uses a work product from another activity. Discuss this different. Provide

an example in the context of the V-model.

A precedence relationship specifically refers to the order of tasks in an projects schedule. Whereas a dependency relationship represents a larger scope of relationships. Dependencies are the foundation of precedence in logical activities within a project.

As for an example in the context of the V-Model, which represents a methodology used for software development that is an extension of the waterfall method. The idea behind the method is that it introduces testing as part of the development process from the start. Suppose we have two activities in the V-model: "Requirements Gathering" and "System Design." The Requirements Gathering activity involves gathering and documenting the client's requirements for the software. The System Design activity involves creating a detailed design specification based on the gathered requirements. In this example, there is a clear dependency between the two activities. The System Design activity depends on the completion of the Requirements Gathering activity because it needs the documented requirements as an input to create the design specification. The work product or output of the Requirements Gathering activity, i.e., the requirements documentation, serves as the basis for the System Design activity.

16-2 You are improving a legacy system. You need to customize a software life cycle for

your project. Which activities would you need? In which order?

I would say that, in general, this is highly dependent on what is being updated. That is, what is being updated and how in depth this update is. Now, with that said, one small change can break everything that is currently in production, so it is important to have all the pieces currently in place before even thinking about making live changes. That being said, the FIRST thing should be information gathering. Getting all the information involved in a current process/tool should be the very first step, before even thinking about making changes. Even if it is a complete rework. Moving on from there, should be the planning stage of the new/updated tool. If possible, I would identify any existing code/architecture that can be reused as is or with minimal reworking in the new system as the next step. Plenty of steps that would need to happen after this, but what has been mentioned already, if done thoroughly, could easily take months depending on the scope of the legacy system. I would say there should be a reassessment of priorities once more information is gathered and nothing should be set in stone so early in a project.

16-4 The heuristics we outlined in Section 16.5.4 indicate that the need for models is higher

in distributed organizations. Open-source projects are highly distributed projects that

follow an entity-based life cycle and typically do not have requirements or system

design documents. Provide examples of how modeling knowledge is made explicit and

transferred among participants in such cases.

There are several ways that the knowledge can be passed along to different people working on an open-source project. In a lot of cases there is a ReadMe file that is added along to any such project that includes the most pertinent information. Other examples would include self-documentation for any variable names or adding in comments to the different parts of the project. I would say another good way to help with documentation/modeling would be to separate into different files as much as possible so there can be an easy trail to follow instead of scrolling through thousands (or millions) of lines of code.