PDE PROVIDER

Professional Development for the Expert

Course Title

Scope Planning

Instructor

Alan Fata, DBA

Credit	Questions
2 PDU	20

Professional Development for the Expert, Inc.
Baabda, Lebanon, (571) 619-5550, info@pdeprovider.com

Adaptation Statement

9. Scope Planning

You always want to know exactly what work has to be done **before** you start it. You have a collection of team members, and you need to know exactly what they're going to do to meet the project's objectives. The scope planning process is the very first thing you do to manage your scope. Project scope planning is concerned with the definition of all the work needed to successfully meet the project objectives. The whole idea here is that when you start the project, you need to have a clear picture of all the work that needs to happen on your project, and as the project progresses, you need to keep that scope up to date and written down in the project's scope management plan.

Defining the Scope

You already have a head start on refining the project's objectives in quantifiable terms, but now you need to plan further and write down all the intermediate and final deliverables that you and your team will produce over the course of the project. Deliverables include everything that you and your team produce for the project (i.e., anything that your project will deliver). The deliverables for your project include all of the products or services that you and your team are performing for the client, customer, or sponsor. They include every intermediate document, plan, schedule, budget, blueprint, and anything else that will be made along the way, including all of the project management documents you put together. Project deliverables are tangible outcomes, measurable results, or specific items that must be produced to consider either the project or the project phase completed. Intermediate deliverables, like the objectives, must be specific and verifiable.

All deliverables must be described in a sufficient level of detail so that they can be differentiated from related deliverables. For example:

- A twin engine plane versus a single engine plane
- A red marker versus a green marker
- A daily report versus a weekly report
- A departmental solution versus an enterprise solution

One of the project manager's primary functions is to accurately document the deliverables of the project and then manage the project so that they are produced according to the agreed-on criteria. Deliverables are the output of each development phase, described in a quantifiable way.

Project Requirements

After all the deliverables are identified, the project manager needs to document all the requirements of the project. Requirements describe the characteristics of the final deliverable, whether it is a product or a service. They describe the required functionality that the final deliverable must have or specific conditions the final deliverable must meet in order to satisfy the objectives of the project. A requirement is an objective that must be met. The project's requirements, defined in the scope plan, describe what a project is supposed to accomplish and how the project is supposed to be created and implemented. Requirements answer the following questions regarding the **as-is** and **to-be** states of the business: who, what, where, when, how much, and how does a business process work?

Requirements may include attributes like dimensions, ease of use, colour, specific ingredients, and so on. If we go back to the example of the company producing holiday eggnog, one of the major deliverables is the cartons that hold the eggnog. The requirements for that deliverable may include carton design, photographs that will appear on the carton, colour choices, etc.

Requirements specify what the final project deliverable should look like and what it should do. Requirements must be measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. They can be divided into six basic categories: functional, non-functional, technical, business, user, and regulatory requirements.

Functional Requirements

Functional requirements describe the characteristics of the final deliverable in ordinary non-technical language. They should be understandable to the customers, and the customers should play a direct role in their development. Functional requirements are what you want the deliverable to do.

Vehicle Example

If you were buying vehicles for a business, your functional requirement might be: "The vehicles should be able to take up to a one ton load from a warehouse to a shop."

Computer System Example

For a computer system you may define what the system is to do: "The system should store all details of a customer's order."

The important point to note is that **what** is wanted is specified and **not how** it will be delivered.

Non-Functional Requirements

Non-functional requirements specify criteria that can be used to judge the final product or service that your project delivers. They are restrictions or constraints to be placed on the deliverable and how to build it. Their purpose is to restrict the number of solutions that will meet a set of requirements. Using the vehicle example, the functional requirement is for a vehicle to take a load from a warehouse to a shop. Without any constraints, the solutions being offered might result in anything from a small to a large truck. Non-functional requirements can be split into two types: performance and development.

To restrict the types of solutions, you might include these performance constraints:

- The purchased trucks should be American-made trucks due to government incentives.
- The load area must be covered.
- The load area must have a height of at least 10 feet.

Similarly, for the computer system example, you might specify values for the generic types of performance constraints:

- The response time for information is displayed on the screen for the user.
- The number of hours a system should be available.
- The number of records a system should be able to hold.
- The capacity for growth of the system should be built in.
- The length of time a record should be held for auditing purposes.

For the customer records example, the constraints might be:

- The system should be available from 9 a.m. to 5 p.m.Monday to Friday.
- The system should be able to hold 100,000 customer records initially.
- The system should be able to add 10,000 records a year for 10 years.
- A record should be fully available on the system for at least seven years.

One important point with these examples is that they restrict the number of solution options that are offered to you by the developer. In addition to the performance constraints, you may include some development constraints.

There are three general types of non-functional development constraints:

- **Time**: When a deliverable should be delivered
- **Resource**: How much money is available to develop the deliverable
- **Quality**: Any standards that are used to develop the deliverable, development methods, etc.

Technical Requirements

Technical requirements emerge from the functional requirements to answer the questions: how will the problem be solved this time and will it be solved technologically and/or procedurally? They specify how the system needs to be designed and implemented to provide required functionality and fulfill required operational characteristics.

For example, in a software project, the functional requirements may stipulate that a database system will be developed to allow access to financial data through a remote terminal. The corresponding technical requirements would spell out the required data elements, the language in which the database management system will be written (due to existing knowledge in-house), the hardware on which the system will run (due to existing infrastructure), telecommunication protocols that should be used, and so forth.

Business Requirements

Business requirements are the needs of the sponsoring organization, always from a management perspective. Business requirements are statements of the business rationale for the project. They are usually expressed in broad outcomes, satisfying the business needs, rather than specific functions the system must perform. These requirements grow out of the vision for the product that, in turn, is driven by mission (or business) goals and objectives.

User Requirements

User requirements describe what the users need to do with the system or product. The focus is on the user experience with the system under all scenarios. These requirements are the input for the next development phases: user-interface design and system test cases design.

Regulatory requirements

Regulatory requirements can be internal or external and are usually **non-negotiable**. They are the restrictions, licenses, and laws applicable to a product or business that are imposed by the government.

An Example of Requirements

Automated teller machines (ATMs) can be used to illustrate a wide range of requirements (Figure 9.1). What are some of the physical features of these machines, and what kinds of functions do they perform for the bank's customers? Why did banks put these systems in place? What are the high-level business requirements?

Figure 9.1 Automated Teller Machine.

The following represents one possible example of each type of requirement as they would be applied to a bank's external ATM.

- ATM functional requirement: The system will enable the user to select whether or not to produce a hard-copy transaction receipt before completing a transaction.
- ATM non-functional requirement: All displays will be in white, 14-point Arial text on black background.
- ATM technical requirement: The ATM system will connect seamlessly to the existing customer's database.
- ATM user requirement: The system will complete a standard withdrawal from a personal account, from login to cash, in less than two minutes.

- ATM business requirement: By providing superior service to our retail customers, Monumental Bank's ATM network will allow us to increase associated service fee revenue by 10% annually on an ongoing basis.
- ATM regulatory requirement: All ATMs will connect to standard utility power sources within their civic jurisdiction, and be supplied with an uninterrupted power source approved by the company.

The effective specification of requirements is one of the most challenging undertakings project managers face. Inadequately specified requirements will guarantee poor project results.

Documenting requirements is much more than just the process of writing down the requirements as the user sees them; it should cover not only what decisions have been made, but why they have been made, as well. Understanding the reasoning that was used to arrive at a decision is critical in avoiding repetition. For example, the fact that a particular feature has been excluded, because it is simply not feasible, needs to be recorded. If it is not, then the project risks wasted work and repetition, when a stakeholder requests the feature be reinstated during development or testing.

Once the requirements are documented, have the stakeholders sign off on their requirements as a confirmation of what they desire.

While the project manager is responsible for making certain the requirements are documented, it does not mean that the project manager performs this task. The project manager enlists the help of all the stakeholders (business analysts, requirement analysts, business process owners, customers and other team members) to conduct the discussions, brain-storming, and interviews, and to document and sign off the requirements. The project manager is responsible only for enabling the process and facilitating it. If the project manager feels that the quality of the document is questionable, his or her duty is to stop the development process.

The project manager reviews the requirements, incorporates them into the project documentation library, and uses them as an input for the project plan.

Software Requirements Fundamentals

This section refers to requirements of "software" because it is concerned with problems to be addressed by software. A software requirement is a property that must be exhibited by software developed or adapted to solve a particular problem. The problem may be to automate part of a task of someone who will use the software, to support the business processes of the organization that has commissioned the software, to correct shortcomings of existing software, to control a device, etc. The functioning of users, business processes, and devices is typically complex. Therefore, the requirements on particular software are typically a complex combination of requirements from different people at different levels of an organization and from the environment in which the software will operate.

An essential property of all software requirements is that they be verifiable. It may be difficult or costly to verify certain software requirements. For example, verification of the throughput requirement on a call centre may necessitate the development of simulation software. Both the software requirements and software quality personnel must ensure that the requirements can be verified within the available resource constraints.

Requirements have other attributes in addition to the behavioural properties that they express. Common examples include a priority rating to enable trade-offs in the face of finite resources and a status value to enable project progress to be monitored. Typically, software requirements are uniquely identified so that they can be monitored over the entire software life cycle.

Measuring Requirements

As a practical matter, it is typically useful to have some concept of the volume of the requirements for a particular software product. This number is useful in evaluating the size of a change in requirements, in estimating the cost of a development or maintenance task, or simply in using it as the denominator in other measurements (see Table 9.1).

Table 9.1: Measuring Requirements

Property	**Measure**
Speed	• Processed transactions/second • User/Event response time • Screen refresh time
Size	• K Bytes • Number of RAM chips
Ease of use	• Training time • Number of help frames
Reliability	• Mean time to failure • Probability of unavailability • Rate of failure occurence • Availability
Robustness	• Time to restart after failure • Percentage of events causing failure • Probability of data corruption on failure
Portability	• Percentage of target dependent statements • Number of target systems

Scope Inputs

The project manager gathers initial project facts from the project charter. In addition, background information on the stakeholder's workplace, existing business model and rules, etc. assist in creating the vision of the final product/service, and consequently, the project scope (see Figure 9.2).

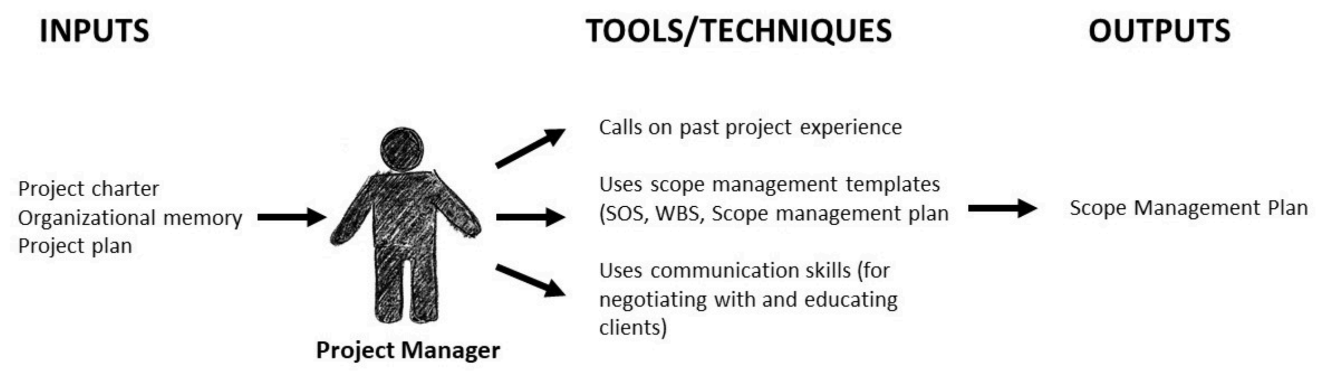

INPUTS
TOOLS/TECHNIQUES
OUTPUTS
Project charter
Organizational memory
Project plan
Project Manager
Calls on past project experience
Uses scope management templates (SOS, WBS, Scope management plan
Uses communication skills (for negotiating with and educating clients)
Scope Management Plan

Figure 9.2: Scope input-output. *[Image description]*

Techniques

Certainly being a seasoned project manager broadens the repertoire of one's scope planning techniques. An experienced project manager can draw on past experiences with like projects to determine the work that is realistically doable, given time and cost constraints, for a current project. Communication and negotiation skills are a "must-have" as well. Project managers need to educate stakeholders about the project impacts of some requirements. Adding complexity to a project may require more staff, time, and/or money. It may also have an impact on project quality. Some aspects of the project may be unfeasible – stakeholders need to know this so they can adjust their vision or prepare for future challenges.

Gathering requirements is part of scope definition, and it can be done using one or more of following techniques:

- Interviews
- Focus groups
- Facilitated groups such as JAD (joint application development)
- Group creativity techniques: brainstorming, nominal groups, delphi, mind map, affinity diagnostics
- Prototyping
- Observation
- Questions and surveys
- Group decision-making techniques: unanimity, majority, plurality, dictatorship

Requirements Traceability Matrix

The requirements traceability matrix is a table that links requirements to their origin and traces them throughout the project life cycle. The implementation of a requirements traceability matrix helps ensure that each requirement adds business value by linking it to the business and project objectives. It provides a means to track requirements throughout the project life cycle, helping to ensure that requirements approved in the requirements documentation are delivered at the end of the project. Finally, it provides a structure for managing changes to the product scope. This process includes, but is not limited to, tracking:

- Requirements to business needs, opportunities, goals, and objectives
- Requirements to project objectives
- Requirements to project scope/work breakdown structure deliverables
- Requirements to product design
- Requirements to product development
- Requirements to test strategy and test scenarios
- High-level requirements to more detailed requirements

Attributes associated with each requirement can be recorded in the requirements traceability matrix. These attributes help to define key information about the requirement. Typical attributes used in the requirements traceability matrix may include a unique identifier, a textual description of the requirement, the rationale for inclusion, owner, source, priority, version, current status (such as active, cancelled, deferred, added, approved), and date completed. Additional attributes to ensure that the requirement has met stakeholders' satisfaction may include stability, complexity, and acceptance criteria.

Matrix Fields

These are suggestions only and will vary based on organizational and project requirements.

- A unique identification number containing the general category of the requirement (e.g., SYSADM) and a number assigned in ascending order (e.g., 1.0, 1.1, 1.2)
- Requirement statement
- Requirement source (conference, configuration control board, task assignment, etc.)
- Software requirements specification/functional requirements document paragraph number containing the requirement
- Design specification paragraph number containing the requirement
- Program module containing the requirement
- Test specification containing the requirement test
- Test case number(s) where requirement is to be tested (optional)
- Verification of successful testing of requirements
- Modification field (If a requirement was changed, eliminated, or replaced, indicate disposition and authority for modification.)
- Remarks

Work Breakdown Structure

Now that we have the deliverables and requirements well defined, the process of breaking down the work of the project via a work breakdown structure (WBS) begins. The WBS defines the scope of the project and breaks the work down into components that can be scheduled, estimated, and easily monitored and controlled. The idea behind the WBS is simple: you subdivide a complicated task into smaller tasks, until you reach a level that cannot be further subdivided. Anyone familiar with the arrangements of folders and files in a computer memory or who has researched their ancestral family tree should be familiar with this idea. You stop breaking down the work when you reach a low enough level to perform an estimate of the desired accuracy. At that point, it is usually easier to estimate how long the small task will take and how much it will cost to perform than it would have been to estimate these factors at the higher levels. Each descending level of the WBS represents an increased level of detailed definition of the project work.

WBS describes the products or services to be delivered by the project and how they are decomposed and related. It is a deliverable-oriented decomposition of a project into smaller components. It defines and groups a project's discrete work elements in a way that helps organize and define the total work scope of the project.

A WBS also provides the necessary framework for detailed cost estimating and control, along with providing guidance for schedule development and control.

Overview

WBS is a hierarchical decomposition of the project into phases, deliverables, and work packages. It is a tree structure, which shows a subdivision of effort required to achieve an objective (e.g., a program, project, and contract). In a project or contract, the WBS is developed by starting with the end objective and successively subdividing it into manageable components in terms of size, duration, and responsibility (e.g., systems, subsystems, components, tasks, subtasks, and work packages), which include all steps necessary to achieve the objective.

The WBS creation involves:

- Listing all the project outputs (deliverables and other direct results)
- Identifying all the activities required to deliver the outputs
- Subdividing these activities into subactivities and tasks
- Identifying the deliverable and milestone(s) of each task
- Identifying the time usage of all the resources (personnel and material) required to complete each task

The purpose of developing a WBS is to:

- Allow easier management of each component
- Allow accurate estimation of time, cost, and resource requirements
- Allow easier assignment of human resources
- Allow easier assignment of responsibility for activities

Example of a WBS

If I want to clean a room, I might begin by picking up clothes, toys, and other things that have been dropped on the floor. I could use a vacuum cleaner to get dirt out of the carpet. I might take down the curtains and take them to the cleaners, and then dust the furniture. All of these tasks are subtasks performed to clean the room. As for vacuuming the room, I might have to get the vacuum cleaner out of the closet, connect the hose, empty the bag, and put the machine back in the closet. These are smaller tasks to be performed in accomplishing the subtask called vacuuming. Figure 9.3 shows how this might be portrayed in WBS format.

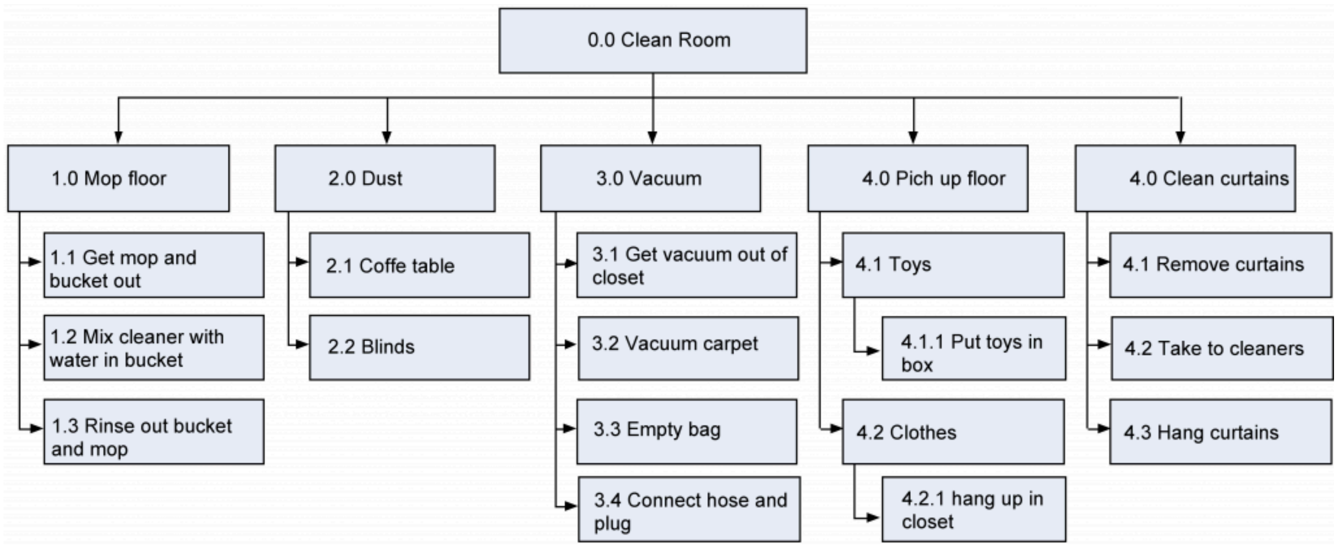
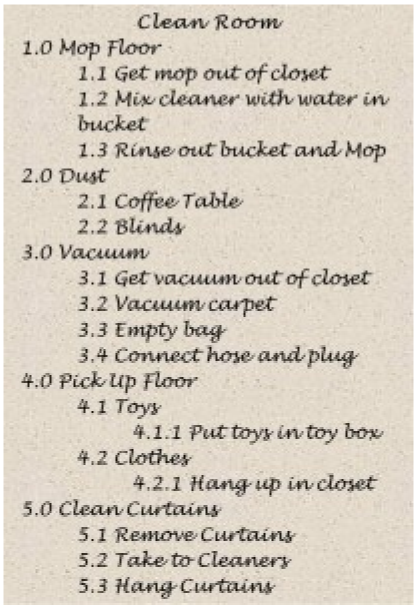


Figure 9.3: A WBS for cleaning a room. [Image description]

It is very important to note that we do not worry about the sequence in which the work is performed or any dependencies between the tasks when we do a WBS. That will be worked out when we develop the schedule. For example, under 3.0 Vacuum, it would be obvious that 3.3 Vacuum carpet would be performed after 3.4 Connect hose and plug! However, you will probably find yourself thinking sequentially, as it seems to be human nature to do so. The main idea of creating a WBS is to capture all of the tasks, irrespective of their order. So if you find yourself and other members of your team thinking sequentially, don't be too concerned, but don't get hung up on trying to diagram the sequence or you will slow down the process of task identification. A WBS can be structured any way it makes sense to you and your project. In practice, the chart structure is used quite often but it can be composed in outline form as well (Figure 9.4).



Clean Room

- 1.0 Mop Floor
    - 1.1 Get mop out of closet
    - 1.2 Mix cleaner with water in bucket
    - 1.3 Rinse out bucket and Mop
- 2.0 Dust
    - 2.1 Coffee Table
    - 2.2 Blinds
- 3.0 Vacuum
    - 3.1 Get vacuum out of closet
    - 3.2 Vacuum carpet
    - 3.3 Empty bag
    - 3.4 Connect hose and plug
- 4.0 Pick Up Floor
    - 4.1 Toys
        - 4.1.1 Put toys in toy box
    - 4.2 Clothes
        - 4.2.1 Hang up in closet
- 5.0 Clean Curtains
    - 5.1 Remove Curtains
    - 5.2 Take to Cleaners
    - 5.3 Hang Curtains

Figure 9.4: Clean Room in an outline view.

You'll notice that each element at each level of the WBS in both figures is assigned a unique identifier. This unique identifier is typically a number, and it's used to sum and track costs, schedules, and resources associated with WBS elements. These numbers are usually associated with the corporation's chart of accounts, which is used to track costs by category. Collectively, these numeric identifiers are known as the code of accounts.

There are also many ways you can organize the WBS. For example, it can be organized by either deliverable or phase. The major deliverables of the project are used as the first level in the WBS. For example, if you are doing a multimedia project the deliverables might include producing a book, CD, and a DVD (Figure 9.5).

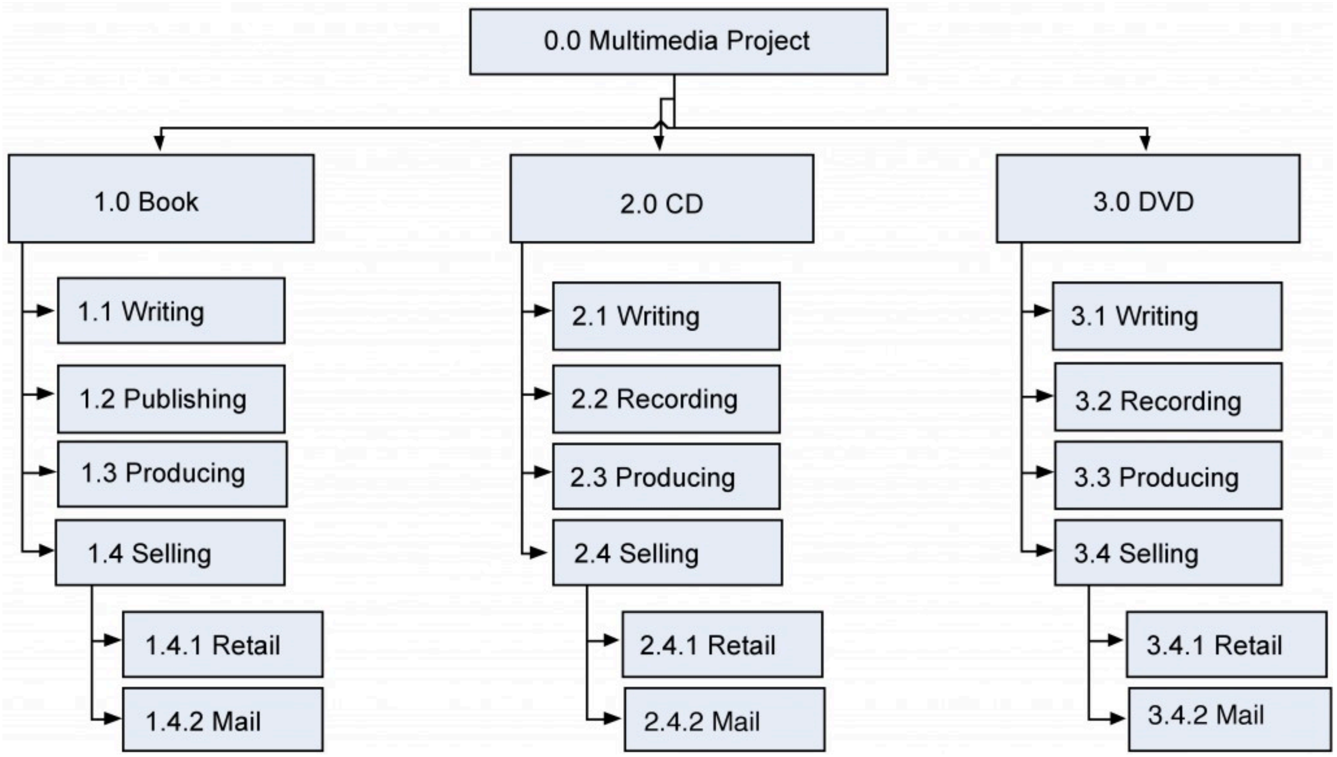


Figure 9.5: A WBS for a multimedia project

Many projects are structured or organized by project phases (Figure 9.6). Each phase would represent the first level of the WBS and their deliverables would be the next level and so on.

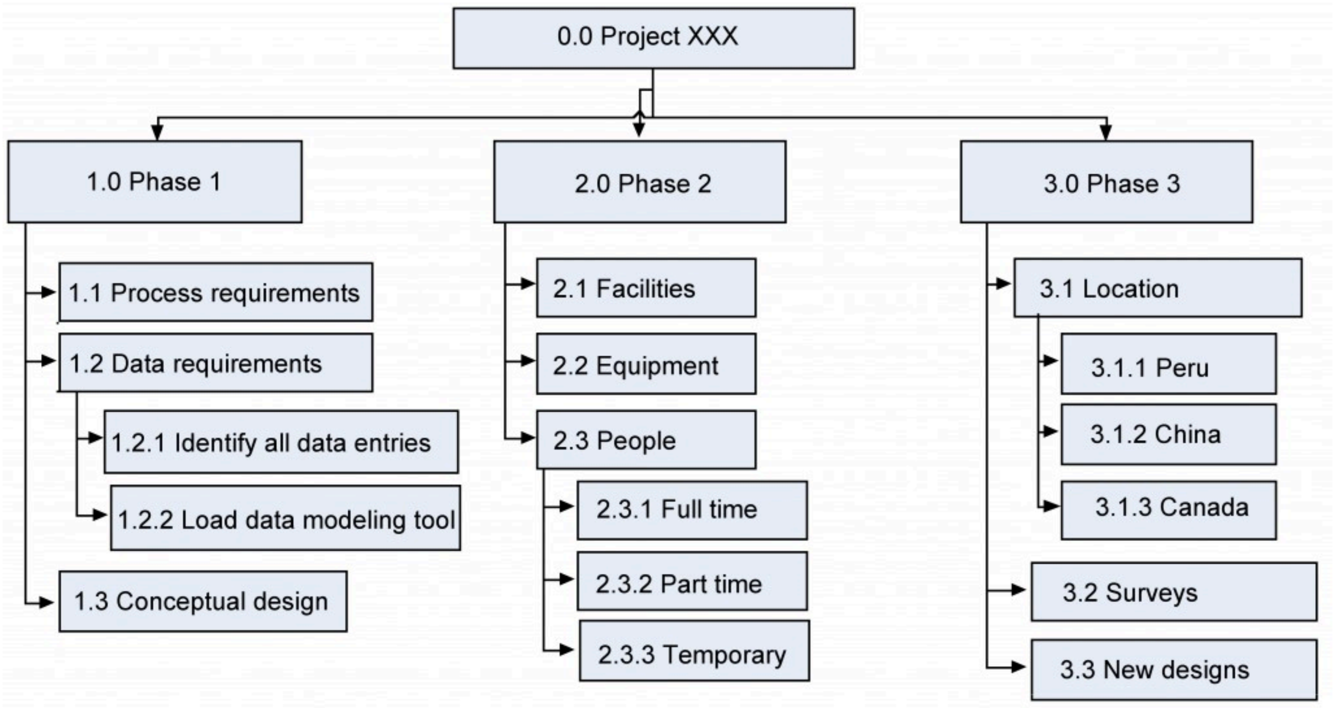


Figure 9.6: WBS Project Phases

The project manager is free to determine the number of levels in the WBS based on the complexity of the project. You need to include enough levels to accurately estimate project time and costs but not so many levels that are difficult to distinguish between components. Regardless of the number of levels in a WBS, the lowest level is called a work package.

Work packages are the components that can be easily assigned to one person or a team of people, with clear accountability and responsibility for completing the assignment. The work-package level is where time estimates, cost estimates, and resource estimates are determined.

100 Percent Rule

The 100 percent rule is the most important criterion in developing and evaluating the WBS. The rule states that each decomposed level (child) must represent 100 percent of the work applicable to the next higher (parent) element. In other words, if each level of the WBS follows the 100 percent rule down to the activities, then we are confident that 100 percent of the activities will have been identified when we develop the project schedule. When we create the budget for our project, 100 percent of the costs or resources required will be identified.

Scope Statement

Scope statements may take many forms depending on the type of project being implemented and the nature of the organization. The scope statement details the project deliverables and describes the major objectives. The objectives should include measurable success criteria for the project.

A scope statement captures, in very broad terms, the product of the project: for example, "development of a software-based system to capture and track orders for software." A scope statement should also include the list of users using the product, as well as the features in the resulting product.

As a baseline scope statements should contain:

- The project name
- The project charter
- The project owner, sponsors, and stakeholders
- The problem statement
- The project goals and objectives
- The project requirements
- The project deliverables
- The project non-goals (what is out of scope)
- Milestones
- Cost estimates

In more project-oriented organizations, the scope statement may also contain these and other sections:

- Project scope management plan
- Approved change requests

- Project assumptions and risks
- Project acceptance criteria

Image Descriptions

Figure 9.2 image description: A project manager develops a Scope Management Plan by taking the project charter, organizational memory, and the project plan and applying the following techniques and tools:

- Calls on past project experience
- Uses scope management templates (SOS, WBS, Scope Management Plan)
- Uses Communication skills (for negotiating with and educating clients)

[Return to Figure 9.2]

Figure 9.3 image description:
0.0 Clean Room

- 1.0 Mop floor.
  - 1.1 Get mop and bucket out.
  - 1.2 Mix cleaner with water in bucket.
  - 1.3 Rinse out bucket and mop.
- 2.0 Dust
  - 2.1 Coffee table
  - 2.2 Blinds
- 3.0 Vacuum
  - 3.1 Get vacuum out of closet
  - 3.2 Vacuum carpet
  - 3.3 Empty bag
  - 3.4 Connect hose and plug
- 4.0 Pick up floor
  - 4.1 Toys
    - 4.1.1 Put toys in box
  - 4.2 Clothes
    - 4.2.1 Hang up in closet
- 5.0 Clean curtains
  - 5.1 Remove curtains
  - 5.2 Take to cleaners
  - 5.3 Hang curtains

[Return to Figure 9.3]