alodar systems™

Integration Tip

WHITE PAPER

How to Define Tasks

Everyone knows the value of planning work before performing it. Planning divides a big job down in to smaller, more understandable pieces. A plan gives you a ruler, a way to measure how well a project team is doing against goals while the project is on-going. Surprises are avoided and corrective action can be taken cost-effectively.

Tools such as Microsoft Project help capture the plan on paper, and no project of any real size goes forward without using them. Yet, despite the power and ease of use of such tools, many projects still overrun their budgets, require rework, or are abandoned altogether.

Merely using a sophisticated computer based tool is no guarantee of project success. In fact, there is a tendency to use them to over plan. If having weekly milestones is a good thing, then, by golly, daily ones have got to be even better. Workers can find that a significant portion of their time is spent in updating status and explaining why they are late on a myriad of tiny steps. Despite the use of summary rollup features, the overall sense of how a project is going is lost; visibility and understanding are decreased rather than enhanced.

And almost inevitably there is a complete restructuring of the project, its schedule or deliverables. The replanning effort becomes a project unto itself, consuming valuable resources while the actual work that needs to get done proceeds uncontrolled and unmonitored.

There is no algorithm for automatically dividing up work and defining tasks, but there are three hallmarks of good task definition that should be heeded: Concrete Deliverables, Unambiguous Responsibility, and Sensible Timespans.

Concrete Deliverables

A task is not a task unless it accomplishes something that can be independently measured. Without such a measurement you can never really be sure that the task is done, or while it is being performed how far along it is.

It is easy to fall into the trap of merely dividing a project into developmental states without thinking what the products of each of those phases are. Traditionally this would include such things as Requirements Analysis, Design, Implementation, and Test. Each of these phases quite naturally translates to high-level tasks in a planning tool.

But for effective task planning, you must think through the measurable end result of the phase. For Requirements Analysis, for example, it would be the Specification Document. You are not really done with Requirements Analysis until it is signed off. The primary focus of management review should be on this deliverable rather than events that happen to occur along the way.

Of course each task can be divided into subtasks, the completion of which will constitute the completion of the task itself. But each of these must have deliverables as well. Subtasks for Design, for example, would include the Loading and Timing Analysis, the User Interface Mockups, and so on.

Avoid inclusion in the plan of level-of-effort tasks. For example, on a large project there might be a role for a system guru, someone who keeps the network running and trouble shoots crashes. But defining a task for this role, although useful for budgetary purposes, contributes nothing to understanding of how the project is progressing; it only adds clutter to the plan.

Unambiguous Responsibility

Even though every task at the lowest level has definite deliverables, there still can be confusion if more than one person is assigned to it. At the lowest level of task there should be one and only one person with the responsibility for its completion.

Although it is fairly easily to ensure such a one-to-one assignment of person to task in the Development phases of a project, what about the Requirements Analysis and Design? What about Testing? Are not such activities best done with a team getting together and hashing things out?

There are two techniques that can be used for such situations. First, the format of the deliverables of these phases can be defined beforehand. One person can have the responsibility of being the final editor or volume captain of the deliverable. Sections of the volume can be assigned to particular individuals.

Secondly, the definition of subtasks can be deferred until it becomes clear what they are. A two-week design effort for example can be divided into two subtasks -- preliminary and final design. One person has the responsibility to document the results of the preliminary design as informal notes. Then a subtask for final design for each piece is added to the plan with an individual assigned to each.

Sensible Timespans

Even if tasks are clearly defined in terms of deliverables and are unambiguously assigned, a project can still suffer from overplanning. Nothing is really gained by having the duration of tasks significantly shorter than the interval between management review. If status is reviewed weekly, then few of the tasks should have duration of less than one week. If a developer is coding five small modules each of which should take a day, then the task should be to code five small modules, not have five tasks of one-day duration.

And of course having tasks that are significantly longer than the review period can lead to management uncertainty as to exactly how far along the project is. The old adage that "the first 90% of the effort takes 90% of the time and the last 10% takes the other 90%" all too often seems to be true.

What then is a reasonable task length? A rule of thumb is that a good task is roughly the same length as the review interval. This gives time to assess how well things are progressing, gives sufficient warning that something is not going well, and does not clutter the plan with so many tasks that the sense of the project is lost and replanning becomes a monumental task.

Hallmarks of Good Task Definition

The next time you generate a project plan, try checking each of the tasks against these three hallmarks -- Concrete Deliverables, Unambiguous Assignment, and Sensible Timespans. The plan will be better if you do.