# Planning and executing time-bounded projects using statistically planned incremental deliveries (SPID)




## *Abstract*

Statistically Planned Incremental Deliveries (SPID) is a practical approach to planning and executing projects that need to meet hard deadlines. SPID's objective is to guarantee that at least a minimum functionality is delivered by a required date.



## *Keywords*

Project planning, incremental deliveries, critical chain, risk management, project management and control.


## *Introduction*

A time-bound project is a project that is constrained by hard deadlines. Hard deadlines are those in which the date of delivery is as important as the delivery itself. If the project delivers after the deadline, the delivery loses much of its value. Meeting the deadline is a critical success factor for the project. Examples of hard deadlines are exhibition dates, government regulations, a competitor's announcement and the customer's own business plans.

SPID is not a one-stop solution for all software development problems. It focuses on how to best organize a project to guarantee that even under the most adverse circumstances a working product, with an agreed functionality, could be delivered by a required date.

SPID combines ideas from statistics, critical chain planning[1,2] and technical performance monitoring[3] into a practical method for planning and executing time-bound projects.

Figure 1 illustrates SPID's project model. The *Increment Planning* task uses statistical techniques to brake the project scope down into a series of *Development Increments* in such a way that it is almost certaint that all requirements allocated to the first increment will be implemented on time; that there is a fair chance to implement those allocated to the second increment and so on. *System Engineering* encompasses requirements, value and trade-off analysis from a user perspective. *System Architecting* is responsible for the general form of the solution, interface definitions and the analysis of requirements dependencies. All three activities take place concurrently as there is a need to balance what needs to be done from the user perspective with what could be done from a technical perspective. Each *Development Increment* is a self-contained mini-project. SPID does not impose any particular approach beneath this level, so development could be organized according to a waterfall or an iterative life cycle as deemed appropriate. All increments, but the last, are isolated from the project delivery date by a buffer whose purpose is to absorb any overrun in their execution.

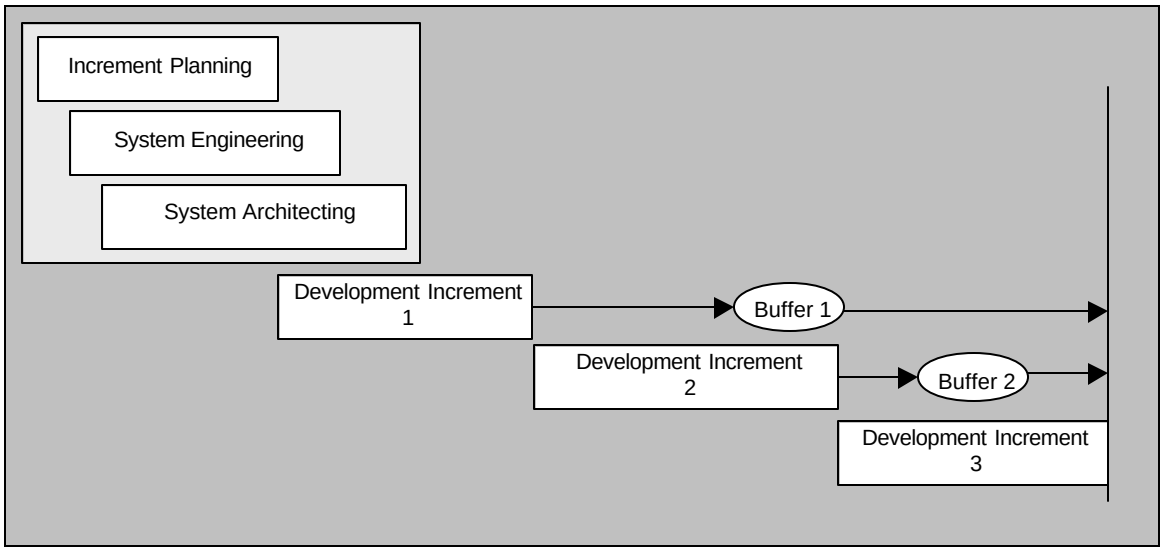


**Figure 1.** SPID project Model

During execution, work progress is tracked using non-linear models which more closely resemble the way people work than a simple extrapolation of last week's results. As shown by Figure 2, the models are used to forecast the activities' completion dates and to take corrective actions. Work in one increment does not start until the previous one is completed. This prevents people from wasting time developing things that might never be finished. To do otherwise, implies either increasing headcount or multitasking the developers. Both practices have a negative impact on the development time as soon as the communication overhead among members of the team starts to surpass the productivity gained by introducing an additional person.

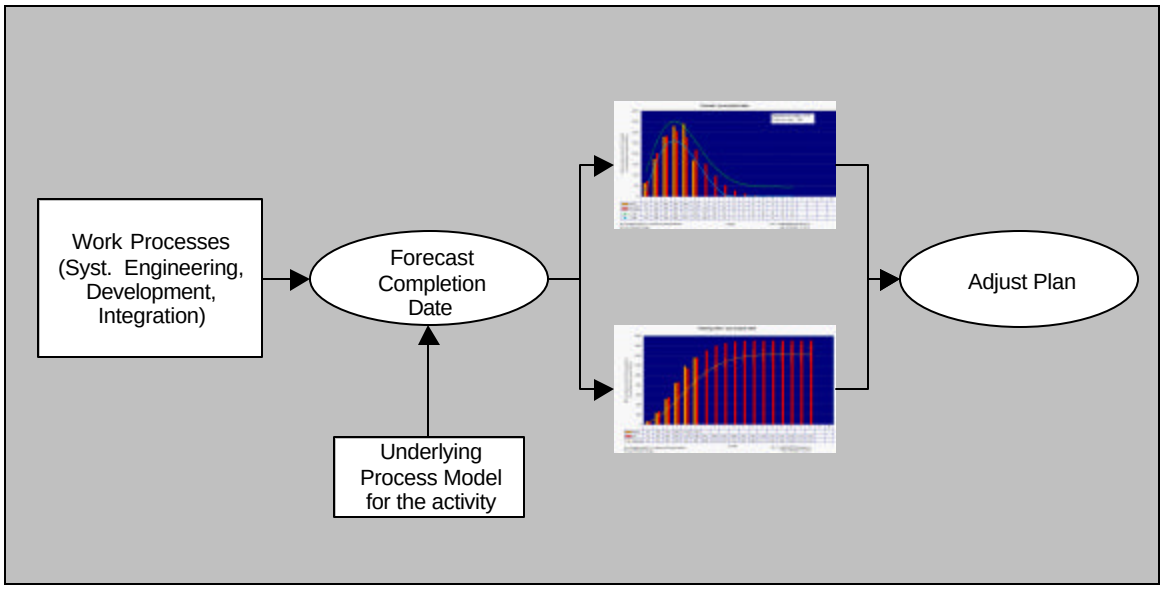


**Figure 2**. SPID execution model

### *Task statistics*

Uncertainty is the reason project management is needed. The estimates on which project schedules and resource allocations are based are never single numbers; whether spoken or not, there are many assumptions behind each of them. Some of these assumptions concern the complexity of the tasks, others our ability to carry them out. Some of them, if true, will contribute to an early completion of a task, others will add to the execution time. Intuitively we could see, that for a task to finish at the earliest possible time, all the "favorable" assumptions must be true and all the "inauspicious" ones false. The probability of this happening is very low. The same argument could be made about the latest possible date. The most likely date corresponds then to a situation in which the most probable "good" assumptions are true and the most probable "bad" ones are false. Numerically, the situation can be expressed by a triangular probability distribution such as the one shown by Figure 3.

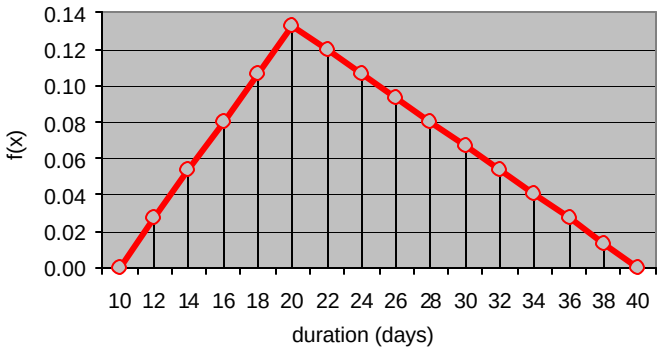


**Figure 3**. If all the favorable assumptions are true and all the gloomy are false, the task will be completed in 10 days, this is the Earliest Completion Date. The Most Likely duration is 20 days. If everything that can go wrong, short of abandoning the task, goes wrong the task could be completed in 40 days. This is the Latest Completion Date.

Since the actual probability distribution function for the duration of the task is unknown, the choice of a simple triangular distribution is a sensible one[4]. Its right skewedness captures the fact that while there is a limited number of things that can be done to shorten the duration of a task, the number of things that can go wrong is virtually unlimited.

From the project management point of view, more important than the probability of finishing on a specific date, is the probability of completing the task on or before a certain date. This probability, called the *on-time probability of the task*, can be derived from the cumulative distribution shown in Figure 4.

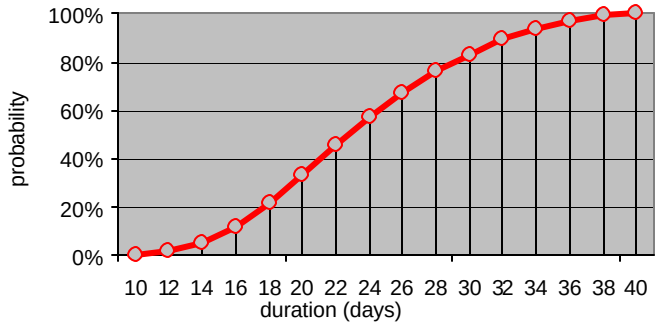


**Figure 4**. Cumulative probabilities. The Most Likely completion date has an on-time probability of less than 40%. The Expected completion date is of around 23 days. If we want to be 75% sure of completing the task on time we would have to schedule 27 days.

In general, the larger the number of assumptions behind the estimated time duration, the larger the spread between the earliest and the latest completion dates. The effect of such an uncertainty results in very different on-time probabilities, as shown by Figure 5.

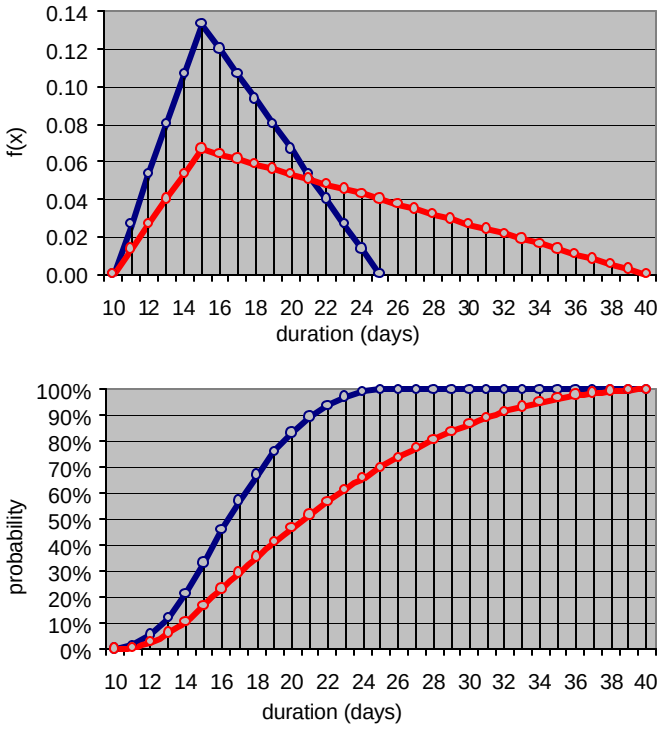


**Figure 5**. Two tasks with the same Earliest and Most Likely, but different Latest Completion dates have different levels of risk. The Expected completion dates for the less risky task is 17 days, while for the other is 23 days. By the same token, the on-time probability of the Most likely date is around 37% in the first case and under 20% in the second.

### *From tasks to projects*

A common approach used to assess uncertainty in projects, is to calculate the expected duration of the project as the sum of the expected duration of the tasks along the critical path, with an standard deviation equal to the square root of the sum of the squares of the standard deviation of the same tasks, and then to use a normal distribution to calculate the on-time probabilty for the project. This approach, is based on the *central limit theorem* which states that the distribution of the sum of independent random variables approaches a normal distribution as the number of variables (tasks) grows larger.

The assumption of independent tasks durations, as required by the central limit theorem, although one of the most common assumptions in project management, is one of the most dangerous a project manager can make. In practical terms, this assumption expresses the belief that the lateness of some tasks is compensated by the early completion of others and in the end everything balances out. This may be a valid assumption in the construction industry and when dealing with events such as rain, but not in a software development project where an underestimation of the system's complexity will affect the duration of most tasks in the same direction. Thus, if there is an underlying cause that could shift the duration of several tasks in the same direction the tasks are not independent but correlated. The practical consequence of dealing with correlated task durations is an increase in the project's standard deviation which translates into higher risks.

As an example of the effect of correlated task durations on the project risk consider the simple project of Figure 6 which has an expected duration of 112 days. The probability of being late by more than 7 days is around 22% under the assumption of independent task durations and 32% when the durations are assumed to be correlated; a 45% increase in a project with only four tasks!

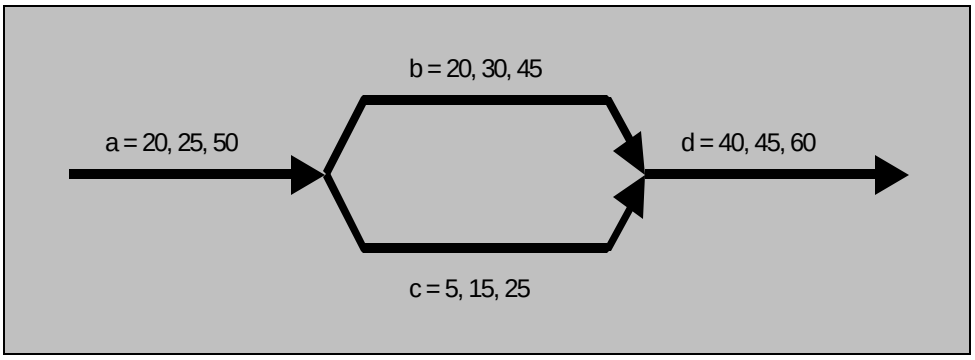


**Figure 6 -** A simple project. The first number indicates the minimum task duration, the second the most likely and the third the maximum.

A simpler approach, described in the boxed note at the end of the article, is to use simulation to calculate the project on-time probabilities.

### *Measuring Progress Using Rate of Changes*

When measuring the progress of a task in terms of its main output, i.e requirements defined, LOC, errors found, pages of documentation written, etc, it is possible to observe that the rate of growth of the output is not constant throughout its life and that it more closely resembles the shape of Figure 7. This "S" pattern[5,6,7,8], typical of many intellectual activities could be explained by the existence of a number of actions and thought processes at the beginning and end of the task which, although value adding and enabling, do not contribute directly to the quantity being measured. Examples of such actions and thought processes are: learning, team formation and re-examination of work already done, which despite never being explicitly included in the planning of a task, are nonetheless very real and time consuming. It is also possible to argue that the marginal productivity of one hour of effort tends to dimish towards the end of the task as it becomes necessary to deal with a myriad of details, or that there are less errors to be found as is the case during testing. Whatever the true reasons for this effect are, it is so common and noticeable that has a name of its own: "the 90% complete syndrome".

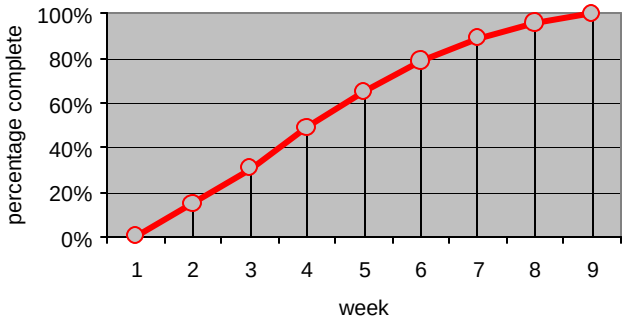


**Figure 7.** The "S" curve. Production does not grow at a constant rate. At the peak of productivity, between weeks 3 and 5, the percentage complete soars 20% in just one week. Towards the end of the task it takes three more times to go from 80 to 100% complete.

The forecasting method does not require the total output of the task to be known in advance. Looking at Figure 7 it is easy to see, that the completion of a task coincides with a leveling-off of the percentage complete curve, so one could choose an arbitrary level of the production forecasted, let's say 95% or 97%, and derive a completion date from it.

Despite the non-linearity of the production process, it is common practice to extrapolate completion dates from the rates of progress observed during the half life of the task using a straight line. The practical consequence of this, is the announcement of optimistic completion dates which are never met. Figure 8 shows the error incurred by using a linear forecast instead of the “S” curve paradigm.

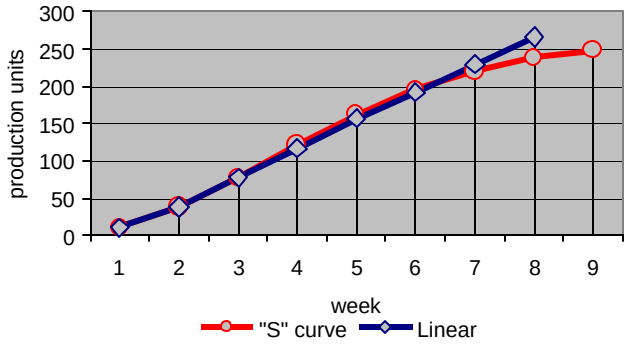


**Figure 8.** Assuming that the task output is 250 units of production (Requirements, FP, Errors detected, etc) a linear projection would forecast its completion by week 7.5 while the “S” curve will put it at week 9. Assuming the task duration was originally estimated to be 7 weeks, according to the linear projection it will be completed almost on time, but according to the “S” curve it will be 2 weeks late.

## *Project Planning*

Once the feasibility of the project has been established, the first step in SPID is to define the duration of the development tasks in terms of its Best, Most Likely and Worst case scenarios as functions of the increment’s scope. Second, the content of the increment is adjusted so it will have a high probability, i.e. 95%, of being completed in the allotted time. Third, the tasks are re-scheduled using the duration that corresponds to a 50% on-time probability, allocating the difference between the high and the lower confidence dates to a buffer. The next increment is then planned using the length of the buffer as the time allotted. Figure 9 illustrates the overall process and the boxed note at the end of the paper, the probability calculations.

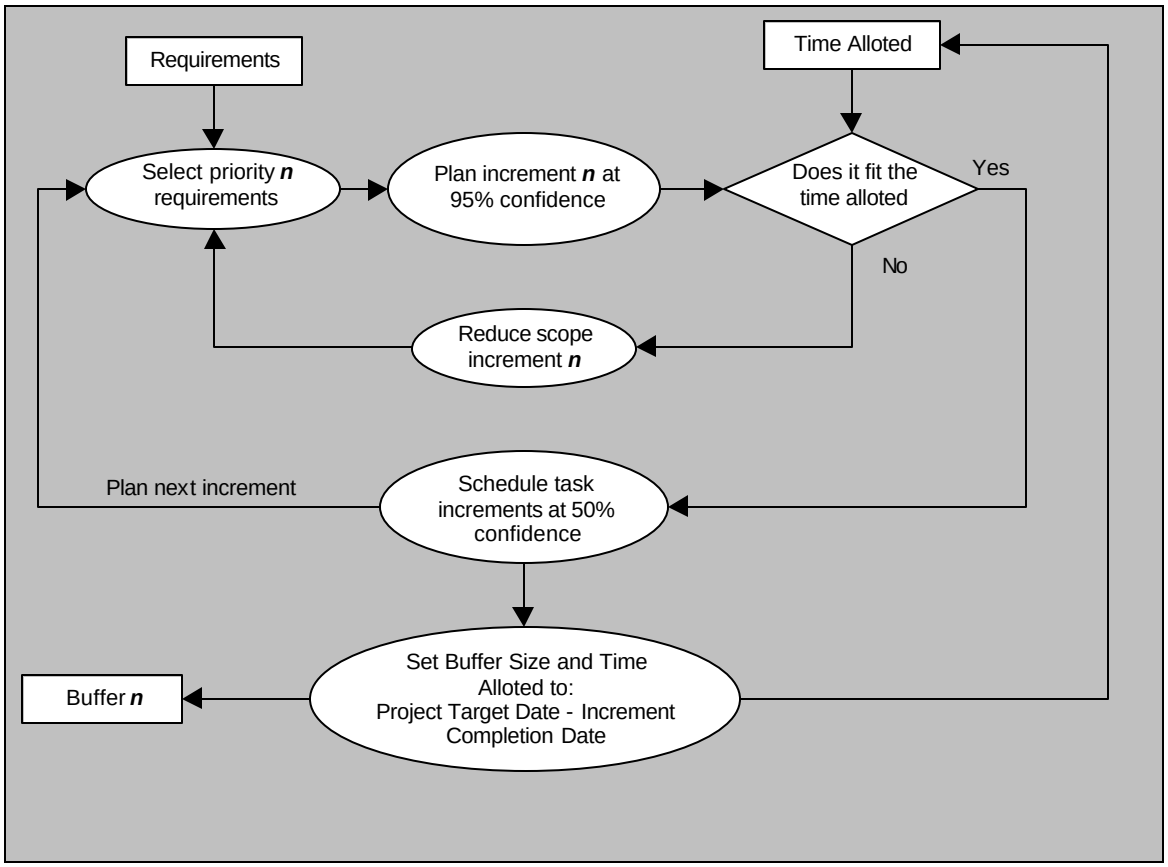


**Figure 9 -** SPID planning process

Table 1 shows the the approximate[1] probabilities of delivering the content of each increment when planned according to SPID. Compare this to a conventional plan, in which every requirement has the same probability, let's say 50% irrespective of its importance to the user.

**Table 1** - Success Probabiliies

| Increment | Calculation | On-time probability |
|---|---|---|
| 1 | As planned | 95% |
| 1 + 2 | 0.50 * 0.95 | ˜ 47.5% |
| 1 + 2 + 3 | 0.50 * 0.50 * 0.50 | ˜ 12.5% |

[1] These calculations assume that the time it takes to develop each increment are independent from one another. As was said earlier this is seldom the case, nontheless these numbers offer a reasonable approximation.

### *Project Control*

In a time-bound project there is very little room for recovery, so once a problem manifests itself, it is almost too late. Controlling a project under these circumstances requires a mechanism that:

1. Identifies the early the signs of a delay;
2. Minimizes false alarms;
3. Minimizes disturbances to ongoing work;
4. Provides a clear definition of what will be delivered and by when.

While the first three properties are important to the people working and managing the project, the fourth is of utmost importance to the customer who depends on the project's deliverables to execute his own business plan.

The early identification of a delay is achieved by updating the buffers, not with the actuals but with the estimates at completion (EAC) of the individual tasks. The estimates are computed by fitting a Rayleigh curve to the progress reported, and then projecting it into the future.

False alarms and disturbances to on-going work are prevented by the use of buffers which isolate workers from overreactions to small variations, by absorbing up to a 25% variance before sending a signal.

Figure 10 describes the SPID control approach. Depending on the specific task being monitored, the units in which the work performed is measured will be Requirements Defined, LOC produced per week, number of errors detected, etc.

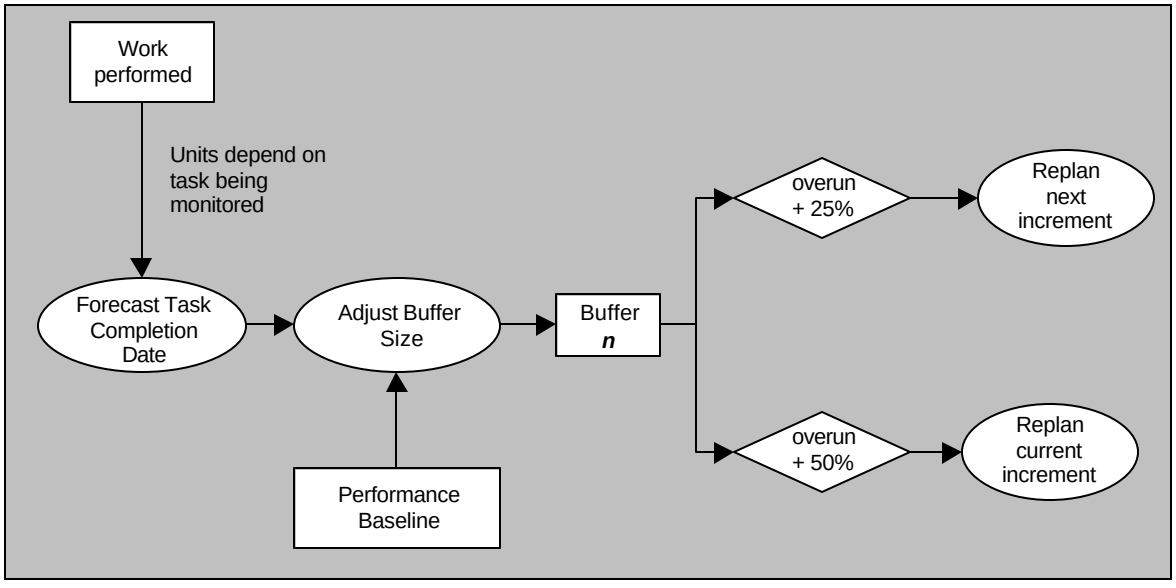


**Figure 10 -** SPID control process

### *Rewards, recognition and price incentives*

How can all project stakeholders be sure that the best effort will be applied towards implementing all requirements and that people will not just get by implementing those in the first increment? The answer could be found in the reward and recognition system.

Whether employee's rewards or price incentives in contracts, the incremental model provides a clear criteria by which performance can be evaluated and rewarded. The delivery of the first increment has no reward associated with it: everybody is just doing their job; subsequent increments result in increased recognition of the extra effort put into the task.

The On-time probabilities shown in Table 1 can be used to calculate the expected value of the reward. This calculation is important because a large amount, with a very small probability will result in a low expected value and could be perceived as a lottery by the employees, thus failing to act as motivator.

As an example, a $5,000 reward for "Increment 2" has an expected value of $2,375. The same amount applied to "Increment 3" has an expected value of $625. Clearly, the motivational value of the reward is not the same in both cases.

## *Summary*

When faced with uncertainty there are two things that can be done: one is to pretend it does not exist, the other is to acknowledge it and devise the means to deal with it. SPID recognizes that in any development project there are hundreds of things that can go right and thousands that can go wrong and makes them an intrinsic part of the planning and control processes while putting emphasis on delivery precision.

The basic premise in SPID is that businesses are better off when they know what realistically could be expected than when they are promised the moon but no assurances are given with respect as to when they could get it. As software and business development grow more entangled, being more predictable in our development projects becomes an obvious need: advertising campaigns could be planned and launched in the confidence that a product will be available, government regulations could be met on time and windows of opportunity respected.

## *Acknowledgements*

Thanks to Tamara Keating, Ericsson Research Canada; Alain Abran, Université du Québec à Montréal; and Raul Martinez, RMyA, for their comments and insight.

## *References*

1. Critical Chain, E. Goldratt, The North River Press, 1997
2. Project Management in the Fast Lane, R. Newbold, St. Lucie Press, 1998
3. Technical Performance Measurement, Earned Value and Risk Management: An Integrated Diagnostic Tool for Program Management, N. Pisano, http://www.acq.osd.mil/pm/paperpres/nickp/nickpaso.htm
4. Practical Risk Assessment for Project Management, S. Grey, John Willey & Sons, 1995
5. A Model for Software Development Effort and Cost Estimation, K. Pillai and S. Nair, IEEE Transactions on Software Engineering, Vol. 23, No.8, 1997
6. Measures for Excellence – Reliable Software On Time, Within Budget, Prentice-Hall, 1992

7. A Learning Model for Forecasting the Future of Information Technology, B. Gaines & M. Shaw, http://spuds.cpsc.ucalgary.ca/articles/BRETAM/FCS_IT/FCS_IT1.html
8. Technological Forecasting for Decision Making, J. Martino, McGraw-Hill, 1993
9. The Use of Reliability Growth Models in Project Management, E. Miranda, 9th International Simposium in Software Reliability, IEEE, 1998
10. On Predicting Software Related Performance of Large-Scale Systems, J. Gaffney, CMG XV, San Francisco 1984

## Calculating Project Probabilities

The most common way of calculating project probabilities is using the PERT approach:

1. For each activity *i*, produce best, most likely and worst case estimates.
2. Compute the mean, $d_i$ and standard deviation, $s_i$ of each task using the following formulas[2]:

$$d_i = \frac{Best + MostLikely + Worst}{3}$$

$$s_i^2 = \frac{Best(Best - MostLikely) + Worst(Worst - Best) + MostLikely(MostLikely - Worst)}{18}$$

3. Determine the critical path based on the $d_i$ , $i = 1,2,\ldots,n$
4. Once the critical activities are identified, sum their means and variances to find the mean and standard deviation of the project length using the following formulas:

$$ProjectLength = \sum_i^p d_i$$

$$ProjectStdDev = \sqrt{s_1^2 + s_2^2 + \ldots + s_p^2}$$

5. Calculate the probability of finishing before a given date *T* using the formula below and a table of normal probabilities:

$$P(t \le T) = P\left( t \le \frac{T - ProjectLength}{ProjectStdDev} \right)$$

The approach described above works well as long as the duration of the tasks is independent, however as mentioned before this is hardly the case in most software development projects. The calculation of the variance of a project on the presence of correlated variables is a complicated process, which requires the calculation of the task's covariances, so a better approach is to use a method called Monte Carlo simulation. A single simulation run would consist of the following steps:

1. Generate a random value for the duration of each activity, using their distribution and a random number. If the tasks are correlated use the same random number if not, use a different one.
2. Determine the project critical path
3. Record the results

After executing these step a sufficiently large number of times, the probability of finishing the project within T days can be estimated using the following expression:

$$P(t \le T) = \frac{\text{Number of Times Project Finished in Less Than or Equal to T Weeks}}{\text{Total Number of Runs}}$$

---

[2] These formulas assume a triangular distribution. Other approximations, like the traditional PERT calculations based on the beta distribution, could be also used.

At Ericsson a home grown tool called MinimumTime (see Figure 2), implements all necessary calculations. More sophisticated packages could be obtained from specialized vendors such as Primavera, Palisade or Cristal Ball.