Software Construction, Renovation,
& Maintenance

Cosine Of Progress

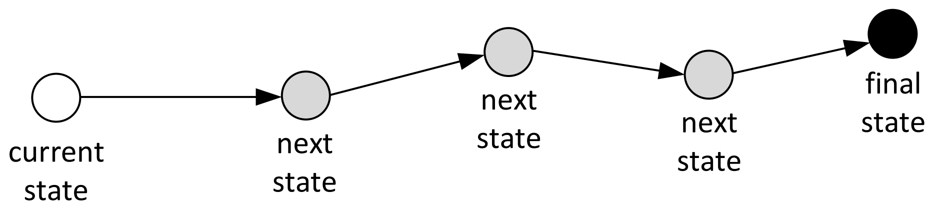
Jim Ladd

August 5, 2024

Most current software development processes follow an incremental and iterative model. At a high level, the major steps include:

1. The software exists in the current state
2. The sponsors and stakeholders visualize the next state
3. The team plans for a state transition
4. The plan is executed within a time-boxed period
5. The progress is reviewed with the users, sponsors, etc.
6. This state becomes the current state
7. The process is repeated until the final state is reached.

A basic diagram of this progression is shown below:



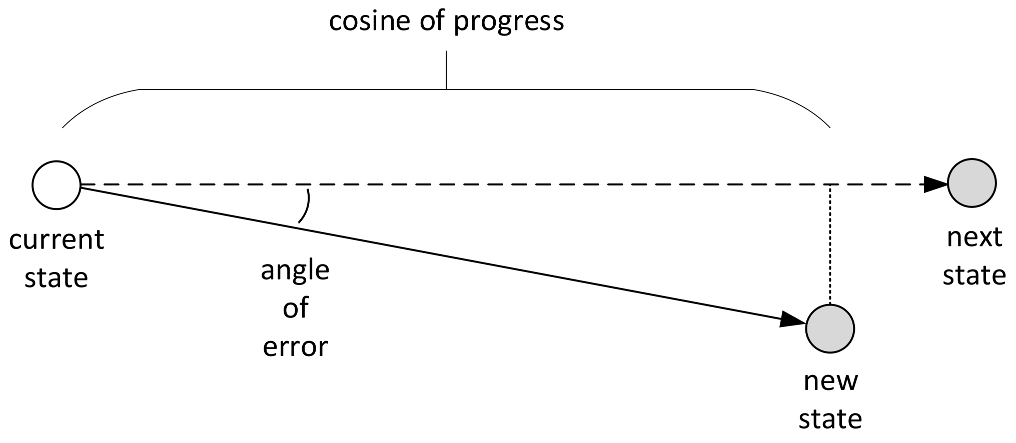
When this process is clicking and everything is running on all cylinders, it's an exhilarating experience. Developers get their required dose of gratification, users take ownership into "their" system, sponsors achieve critical business goals, and there's an abundance of kudos for managers to claim for themselves *(just kidding…sort of)*.

When things don't click, the process can be very frustrating and inefficient. In my experience, the top reason a team doesn't achieve a high cadence with this process is the lack of clear and shared visualization of the next state. There can be several different types of causes for this condition; 1) the sponsors or stakeholders don't have time to devote to the project, 2) they assumed the vision was sufficiently captured and communicated, or 3) they simply don't have the information and data to formulate the next step yet.

In the remainder of this article, two concepts are presented to help navigating forward where the next state of the software is unclear.

Cosine of Progress

When the vision forward is blurred or is even a total void, one technique that I use is to have the development team create a synthetic state and drive towards it. To help explain this concept, I use the following diagram where the new state represents the synthetic or internally generated state.



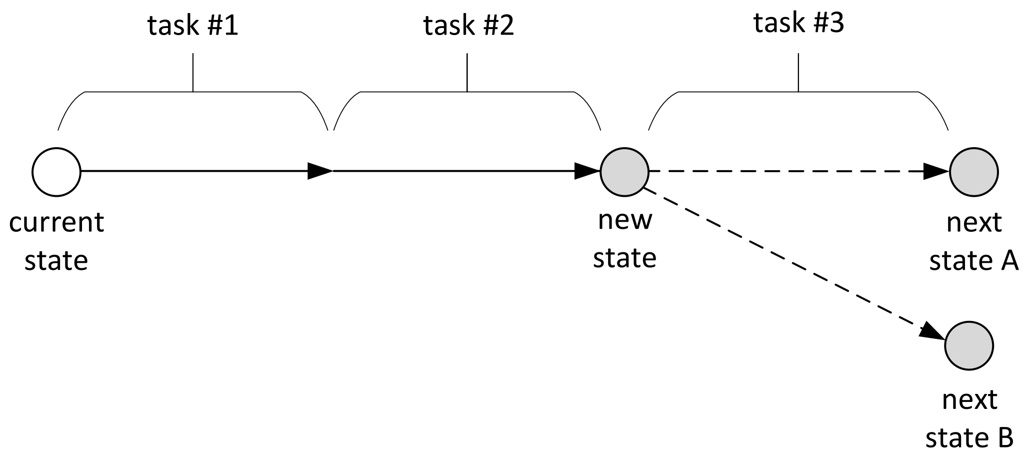
At the end of the period when the artifacts are reviewed by the interested parties, the difference between the new state and the elusive next state can be determined. Rarely do the two states perfectly align, and the delta can be represented by the angle of error. The advantage of this approach is the cosine of progress as shown. This represents the amount of progress toward the next state that was achieved by moving forward.

While some have claimed that this approach is high-risk, my argument is that taking action, any action, is better than doing ***NOTHING***. If the synthetic state of the software is even close to the desired next state, significant progress will be achieved, and the momentum of the project preserved.

Hurry Up and ~~Wait~~…Move On

There are times when faced with a blurry vision that the Cosine of Progress technique isn't the most effective approach. There are scenarios where the multiple possible next states exist and are drastically different. The state transition may also have impacts on systems outside of the team's realm.

In these cases, the tasks required for the transition are identified. Those tasks that are needed regardless of the next state are completed. For example, in the diagram below, task #1 and #2 can be completed ASAP. Task #3 must wait for the next state to be decided. Once the tasks that can be completed are finished, the developers can move on to other work.



As a side note, once the new state is reached and the transition is truly stalled, the developer(s) can escalate the condition and then move on to other work. I've been repeatedly surprised by how quickly, even in lethargic organizations, the issue is resolved.

Software development is not easy, but it should be straightforward. During a project's lifecycle, there will be several challenges and obstacles emerge that can derail the budget and schedule. The key to long-term success in this industry is to find ways to keep moving forward.