INCISIVE VERIFICATION ARTICLE
JUNE 2005

GOOD AND BAD VERIFICATION PLANNING

STEVE BROWN, CADENCE

DO YOU HAVE A PLAN?

Project management is all about planning and execution. But if everyone properly plans their verification project, why do quality problems and schedule slips persist? It really comes down to the adage, "Begin with the end in mind." A good plan contains detailed goals using measurable metrics, along with optimal resource usage and realistic schedule estimates.

Managers and their teams engage in planning, yet the results of that planning process typically do not address the common problems that cause schedule slips, resource productivity issues, or product quality. Most verification plans focus solely on task performance rather than defining the verification problem, independent of its solution. This almost guarantees gaps leading to bug escapes, schedule delays to fix those bugs, or significant resource strains and inefficiencies. With a bit of improvement in planning, we can do much better.

WHY NOT GOOD VERIFICATION PLANNING?

Most teams don't do complete verification planning for two interdependent reasons: they leap to verification environment development before design because they are stretched extremely thin. This can lead to an unchanged or inflexible plan that becomes irrelevant because there is no easy way to maintain the relationship with the actual project. In other words, the verification plan isn't really a plan; it's merely a set of incomplete discussion notes that atrophy as the project moves forward, as the team begins working and learns new things about what it must accomplish.

The solution is to make the verification plan an executable part of the verification process itself. A verification plan becomes executable when it is read by a verification process automation tool, which organizes and generates reports about project status and becomes the basis for analyzing data to determine next steps. The plan's value is maximized, serving as the genesis and touchstone of the verification process throughout the life of the project. This directly increases the return on investment in developing and maintaining the verification plan by automatically using it to measure verification completeness. Thus, when changes in the project are necessary, those changes are introduced, tracked, and measured through the updated, executable verification plan. This makes the plan a continuously valuable part of the verification project, from specification to closure.

Better planning, and using the executable plan to measure project completion, leads to higher quality, increased schedule predictability, and improved resource productivity. With a better plan, it's easier for the team to produce the desired results. Using the plan throughout the project helps surface issues earlier, which is a key to staying on schedule. Tracking progress against the plan metrics enables everyone on the team to self-manage and stay focused, thus improving productivity.

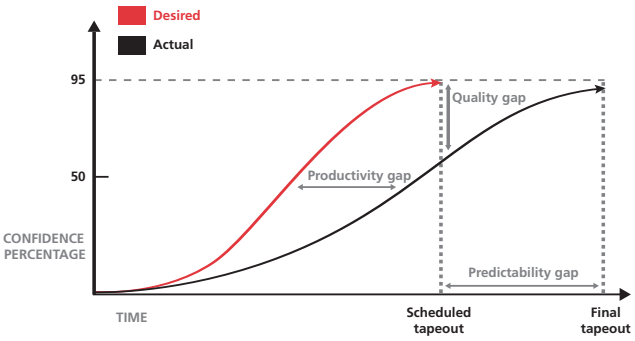

Figure 1: The verification productivity gap

WHAT IS GOOD PLANNING?

Verification planning needs to change from a focus on *how* to a focus on *what*. By starting with the important goals of *what* needs to be verified, the team can ensure the plan is complete and balanced. The plan in this case is more than an engineering specification of *how* the verification is to be accomplished. It is a unanimous statement about what the project must accomplish to be successful. It identifies success criteria in a prioritized fashion — for example, these features *must* work; these features *should* work in the first release; these features are *nice to have* working in the first release, and so on — and the details on how the success criteria will be measured, indicating the project has reached closure.

BASICS OF VERIFICATION PLANNING

From a high-level process perspective, verification planning is actually quite straightforward. The basic steps of verification planning are:

1. Analyze the device specifications
2. Scope the verification objectives
3. Identify the feature set of the design
4. Design detailed coverage models
5. Select aggregate metrics to track progress
6. Use historical metrics to estimate schedules

Team leaders tend to follow this process in general. However, they invariably miss at least one critical aspect of each step. By cutting corners to save time, they gamble and risk their project. And, more often than not, they lose the bet.

ANALYZE THE DEVICE SPECIFICATIONS

All projects begin with some (or several forms of) specifications. Marketing contributes the customer requirements. Management defines the resources, schedule, and build versus buy constraints of the project. Systems engineers, hardware and software engineers, and verification teams each provide verification implementation specifications that will guide the project.

The biggest risk to the project is not anticipating the moving target. Some teams attempt to capture all their requirements and then move through the project sequentially, sometimes called the waterfall method. In reality, the process is iterative. The risk comes from not fully anticipating the iterative impact nor managing the frequency and burden of those iterations.

SCOPE THE VERIFICATION OBJECTIVES

The most common failure at the scoping stage of the project is not requiring a comprehensive verification discussion process with everyone involved. Scoping and documenting the verification objectives is the only way to discern whether the specifications were well conceived and understood. It is terribly inefficient, and unpredictable, to plan details into a verification project after it is underway. Management must allocate a sufficient amount of time upfront to get the verification goals accurate and complete. Propagating requirements changes are another matter and will be discussed in the next section.

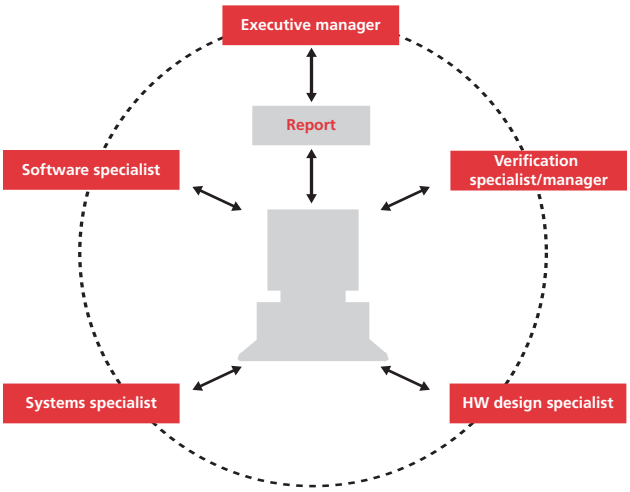

Figure 2: Team verification planning

All stakeholders must participate in scoping verification objectives: management, marketing, systems designers, hardware designers, software designers, and the verification team. It is only through team dialog that the full project requirements emerge. Marketing requirements are never complete because they only address market-critical factors. Management resource and schedule constraints are usually aggressive, by definition, and must be relaxed as the project is fully scoped. Systems engineers bring the technical perspective closest to the marketing requirements. Hardware and software engineers provide valuable insight on the implications of implementation. And, the verification team choreographs the discussion, continuously challenging the team to find and fill the gaps and

identify the unverifiable choices: a key design-for-verification objective. If any of these points of view are missed, the gaps and risks will remain in the project, inevitably leading to quality concerns, cost over-runs, or schedule slips.

IDENTIFY THE FEATURE SET OF THE DESIGN

Good measurement techniques have been an elusive part of verification. Metrics are the means by which verification progress is measured. Historically, engineers have designed tests, implemented checkers, and used code coverage. Test lists are obsolete as a metric because they have become too numerous to specify or implement. Although directed tests are sometimes easier to write, they simply don't scale to the size of today's systems on chip. Checkers aren't appropriate as metrics because they detect erroneous system behavior rather than record that good behavior has been observed. And, code coverage is only loosely correlated with behavior because the device context of each line of code is not recorded.

Leading electronics teams have identified functional coverage as the most accurate verification metric. Using a coverage-driven verification (CDV) methodology, a team is able to measure how much real verification they've completed, as opposed to how many (mostly redundant) simulation cycles they've executed. The specification language of functional coverage is designed to match the requirements specification captured during the scoping process. The assertion language in a coverage specification can also be used to capture implementation assumptions, part of assertion-based verification (ABV).

Total coverage is the only means to have a complete picture of the project's verification status. Functional coverage correlates directly to the features. Assertion coverage relates to functional coverage and to the implementation integrity. Hardware/software code coverage tells us how well we have exercised the design.

DESIGN DETAILED COVERAGE MODELS

The scoping process results in a list of the features to be verified. The specification must describe those features so that they can be measured. Coverage is used to define the metrics of verification, derived from design features. There are two types of metrics that result from scoping: explicit specification metrics and explicit implementation metrics. Explicit specification metrics are chosen by the engineer from the specification. Explicit implementation metrics are chosen by the engineer from the RTL implementation, once the RTL is available.

The typical failure at this stage of planning is lack of review of the coverage models. Not only do they need to be complete enough to represent the full list of features to be verified, but they also need to be succinct enough that the tools and the team can actually accomplish the task within the desired timeframe. Judgment is crucial: it's a lot better to have verified 100% of the most critical functionality than to have critical functionality lost in a coverage model that is too detailed.

For a more complete treatment of this topic, refer to *Functional Verification Coverage Measurement and Analysis* by Andrew Piziali (Springer, ISBN 1-4020-8025-5).

SELECT AGGREGATE METRICS TO TRACK PROGRESS

Management is all about planning and executing. Tracking progress is the only way to know your team is executing. The management adage, "You get the behavior that you measure," is apropos. The challenge is selecting the few metrics to track that are representative of progress and will surface problems and risks.

Typically, teams will define milestones for a project to delineate progress. To maximize the effectiveness of the team, these milestones should be planned such that they pull forward development of the most time-critical or high-risk areas of the project. But verifying a particular feature is no longer sufficient.

Increasingly, teams define milestones that expose possible system integration issues. They deliver the right subset of block functionality to enable early system verification. This requires a fine-grained definition of features that must be working at each milestone. By using coverage to define those features, and by carefully defining and tracking milestones that collect a system subset of features, the project milestone becomes important for much more than simply marking a box in the schedule as complete. It truly reduces quality and schedule risk in the project.

Other manually tracked milestones include first working simulations, as well as generation, checking, and coverage model completion. Also known as implementation tracking, anticipating and achieving these two milestones often befuddle teams first adopting coverage-driven verification. An executable verification plan can specify the conditions for each, improving the predictability of these critical early stages of a project.

USE HISTORICAL METRICS TO ESTIMATE SCHEDULES

Creating engineering schedules is an art. It reflects the team's experience with project planning and recalling how much time they spent on similar tasks in the past. Any act of engineering that jumps to the answer is an art. As more metrics are used in functional verification, the art of schedule estimation is becoming more an act of engineering, where the process is visible.

The challenge is that few teams have captured metrics in the past. Nor have they invested in creating a model whereby a schedule can be quantified and estimated. Some of our more sophisticated customers are tracking additional metrics, beyond coverage, to improve their schedule-estimating accuracy.

The metrics they're tracking measure time for implementing, debugging, and integrating each level and abstraction of IP in the design and verification environment, with data points for varying sizes of IP. They also estimate the impact of developing new IP versus re-use, the impact of engineering experience and skill, and the tradeoff between verification process automation and manual verification management. They look at hardware and software, as well as block-, chip-, and system-level scopes.

Without actual historical metrics, the team can estimate these metrics. Today's schedules are estimates of the tasks, rather than of the metrics themselves. By breaking the estimates down to one more level of detail, the team discovers assumptions in their reasoning. They also create detail against which each individual can track their own work.

They can then construct a model or equation that uses these historical metrics and the current project estimates to generate detailed project schedules. These estimates and the model itself are then debated for their accuracy and a project schedule commitment can be made from the schedule estimate. With the right kind of automation software, the overall project, task, and detailed metrics can be compared against these estimates and captured for the benefit of the next project.

CHANGING REQUIREMENTS

Every engineer knows that requirements always change. Delaying planning does not alter this reality. Many electronics products now serve consumer markets that are more closely tied to the season than to the completion of a project. Managers have long sought to control the change process to make projects more predictable. The big question is, "How late can we introduce a change and release a quality product on schedule?"

BENEFITS OF AN EXECUTABLE VERIFICATION PLAN

The historical verification plan is a crucial part of the automation mechanism for driving and managing verification from specification to closure. The components of an automated verification mechanism are:

- Executable verification plan
- Coverage and assertions to measure verification progress
- Checkers and assertions to detect and report specification violations
- Stimulus generation

Several benefits arise from better verification planning and using an executable verification plan:

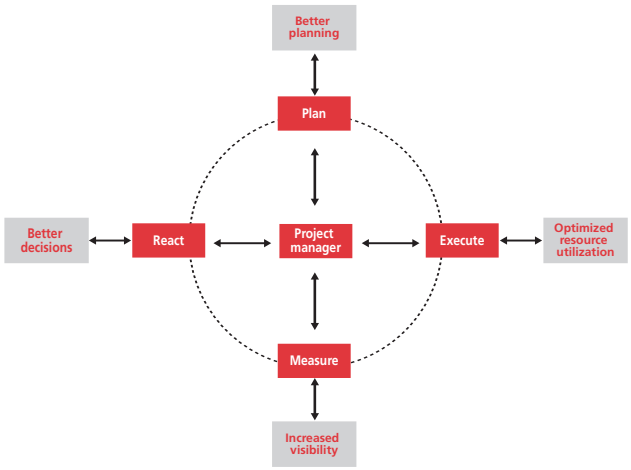

Figure 3: An executable plan drives the verification process

IMPROVED PRODUCT QUALITY

It is easy to see that good planning = good quality. Every customer we've helped to improve verification planning shares a story of the bug they discovered following this process. Even for projects well under way, the value of rigorous verification planning is clear. In some cases, the biggest value is creating a common process and nomenclature for planning.

SCHEDULE PREDICTABILITY

Improved schedule predictability derives from more accurate work estimates and earlier visibility to deviations from the plan. Detailed milestones help surface crucial problems earlier, giving the team an opportunity to work through the challenges in the normal course of the project.

TEAM PRODUCTIVITY

An executable verification plan helps the team communicate more efficiently as well as focus on critical issues more easily. Standard reporting increases team and management effectiveness, keeping resources focused on the most crucial tasks.

CONCLUSIONS

With this picture, "good" verification planning can be defined as using the input of all stakeholders to capture and review the verification plan; using historical metrics and a formulaic model to estimate schedules and drive the whole verification project, from specification to closure; and using an executable verification plan.