Bringing order out of schedule chaos

Angela Tuffley, *RedBay Consulting Pty Ltd*
Adrian Pitman, *Australian Defence Materiel Organisation*
Elizabeth K. Clark, *Software Metrics Incorporated*
Bradford K. Clark, *Software Metrics Incorporated*

Abstract

A recent Gartner survey found that irrespective of project size the single most common reason that projects were considered a failure, was because they were substantially late. Two-thirds of the Gartner survey respondents consider the challenges of bringing projects in on time, on budget and with the agreed functionality as the primary causes of project failure. This paper focuses on what is almost always the primary concern of project stakeholders; to bring projects in on time.

Schedule slippage is a symptom of any number of problems or causes occurring on a project. Identifying root causes of schedule slippage is not always easy but is necessary if schedule slippage is to be remedied and managed. During a project's life cycle, project managers are flooded with a wealth of information on the progress and status of their projects and the challenge is to bring order to chaos so effective action can be taken to address problems.

A perfectly constructed schedule cannot accurately reflect the work to be done if staffing issues are not handled, estimations are inaccurate, requirements are churning, rework and technical debt is omitted, stakeholders and subcontractors are problematic or third party vendor products (COTS) don't provide the required functionality. And even if all these areas are working well, a well-constructed schedule is useless unless it is properly used to execute the project.

This paper will discuss root causes that can drive schedule slippage which, as supported by the Gartner survey, is a reality for many development programs. These root causes, which have been experienced and observed by the authors, are arranged into the Root Cause Analysis of Schedule Slippage Model (RCASS).

RCASS is used as the basis of the Schedule Compliance Risk Assessment Methodology (SCRAM), a framework for identifying and communicating the root causes of schedule slippage and recommendations for going forward to Program and Executive-level management. SCRAM, along with the RCASS model, includes an assessment of the schedule preparation, probability distribution of forecast completion dates and a "health check of the documented schedule". SCRAM can be used at the commencement of a program to validate a proposed schedule and identify potential risks; during program execution as a "health check"; or as a diagnostic tool to identify root causes when schedule slippage occurs.

The development of SCRAM is funded by the Australian Defence Materiel Organisation, (DMO) and is an on-going collaborative effort between DMO, Software Metrics Inc. and RedBay Consulting.

Keywords

Schedule Slippage, Project Risk, Forecasting

Introduction

In 2012 Gartner surveyed its users and concluded that "while large IT projects are more likely to fail than small projects, around half of all project failures, irrespective of project size, were put down to functionality issues and substantial delays."[1]. Two-thirds of the Gartner survey

respondents consider the challenges of bringing projects in on time, on budget and with the agreed functionality as causes of project failure.

Schedule slippage is a symptom of any number of problems or causes occurring on a project;

Optimistic, unrealistic estimates	Conflicting views among stakeholders
Evolving or unstable requirements	Poor subcontractor performance
Use of immature technology	Dependencies not realized and/or often not scheduled
Poor monitoring of changing workloads	Poor quality work leading to unanticipated or unplanned rework
Incurring Technical Debt with no plans to repay	Inadequate staffing
Lack of adequate planning and preparation fo System Integration	Artificially imposed deadlines
Poorly constructed schedules	Lack of Technical Progression
Poor management communication	Lower than estimated productivity

Table 1 provides some examples.

Optimistic, unrealistic estimates	Conflicting views among stakeholders
Evolving or unstable requirements	Poor subcontractor performance
Use of immature technology	Dependencies not realized and/or often not scheduled
Poor monitoring of changing workloads	Poor quality work leading to unanticipated or unplanned rework
Incurring Technical Debt with no plans to repay	Inadequate staffing
Lack of adequate planning and preparation fo System Integration	Artificially imposed deadlines
Poorly constructed schedules	Lack of Technical Progression
Poor management communication	Lower than estimated productivity

Table 1 - Problems and Causes of Schedule Slippage

Trying to identify root causes of schedule slippage is not always easy but is necessary if schedule slippage is to be remedied and managed.

Schedule Chaos

During a project's life cycle, project managers are flooded with a wealth of information on the progress and status of the projects and the challenge is to bring order to chaos so effective action can be taken to address the problems. The following describes some examples of the chaos experienced or observed by the authors on a variety of projects.

- Underestimate the amount of software code to be written and the amount of documentation to be developed and reviewed.
- Identical estimates in four different areas of software development.
- One developer on a program described their stakeholders as being like a "100-headed hydra: nobody could say "yes" and anyone could say "no."'
- Key stakeholders not talking to each other even though they were in the same facility.

- The misinterpretation of a communication standard discovered late in development added an additional 3000 message format requirements implied by that one standard.
- In a large ERP project there were two system specifications; the acquisition agency had one specification with the customer and a different specification under contract with the developer causing considerable problems for final system acceptance and meeting the go-live date.
- A subcontractor that claimed highly mature development processes; however a visit to the subcontractor site revealed that developers were sidestepping processes in order to make deadlines incurring Technical Debt (defects).
- A subcontractor that was eight time zones away severely restricting coordination and virtual meetings that impacted schedule performance.
- Underperformance of pre-existing products, i.e. the legacy systems or COTS products do not work as advertised.
- Underestimating the amount of code that must be written or modified in using a legacy product. One program reviewed planned to only modify 10% of a legacy system but by the end of the development phase, 50% of the system had been modified to satisfy requirements.
- A COTS product required an unplanned “technology refresh” as it was already years late resulting in further pressure on the schedule.
- Scheduling major acceptance testing for the customer during the December / January period of reduced activity even though the contract specifically excluded this period for customer allocated tasks.
- No staff access to the schedule due to the project undergoing a schedule tool transition for approx. 2 years.
- Rework often underestimated, not planned or prioritised for correction.
- No contingency built into the schedule for any rework.
- After the clarification of a requirement an additional seven software releases that were not planned or scheduled.
- Technical Debt is incurred through the suspension of process (e.g. stop peer reviews to meet deadlines) and other process short-cuts.
- Projects not able to fill all the positions for their FTE allocation.
- To recover schedule, schedule staff for 12 hour shifts though only indicated in the master schedule as 8 hour shifts resulting in staff “burn out” and resignations at a crucial time.
- The “star” software developer who understands the most about how the software system worked. Even though he worked long hours, he was a bottleneck. He was so busy, he did not have time to respond to problems, train others or update design documentation.
- A parent company sacked 119 associated workers as the project they were on had completed. As a result, the remaining workers went on a “Go Slow” to make sure they still had a pay packet.

- During the System Integration and Test phase, progress stalls as tests become blocked whilst issues with the system integration and test are resolved.
- Lack of adequate planning, grooming and qualification testing of the integration environment prior to conducting formal testing resulting in the allocation of schedule to System Integration being eroded by activities to address and correct the issues.
- The critical stakeholder (aka the customer) added one condition for acceptance that removed three months from the development schedule; resulting in concern that the project would slip schedule and not complete on time.
- On a large development program, the prime and sub-contractor schedules were not aligned.
- Thirteen (13) subordinate schedules with no effective integrated master schedule to provide an overall understanding of the completion date of the project.
- The critical path went subterranean only appearing occasionally like a duck diving. In this case, the project had intentionally constructed the schedule so that virtually no tasks were on the critical path to be able reporting that the project was not slipping.
- 80% through schedule and cost yet only 50% of the product has been developed. Status of technical progress is based on “guess work” rather than using feasible evidence provided by developers and independently validated.
- Inadequate system integration and test facilities in terms of capacity and/or fidelity, e.g. simulators, emulators, and live environments.
- The Configuration Change Management System could not keep pace with the software defect notification and resolution process resulting in slowing down software releases to systems integration.
- Assumption that developers know what they are doing and will perform at a high rate of productivity.
- During a re-plan the estimates were based on twice the historic productivity with no basis for improvement.

Order out of Chaos

The issues described in the previous section are not new or exhaustive and many books could be written on the bad experiences in projects. But the challenge is to bring some order to the chaos so the root cause to schedule slippage can be addressed and identified risks mitigated.

A typical behavior seen in programs that slip schedule is early milestones or deadlines are missed, new requirements are added, productivity is lower than estimated but schedule milestones do not change. Activities later in the development cycle then get their durations squeezed. A common remedy is to add more people late in the program to increase production. This typically slows down progress due to lack of familiarisation and training and increases communication overhead among development teams.

A perfectly constructed schedule cannot accurately reflect the work to be done if staffing issues are not handled, estimations are inaccurate, requirements are churning, rework and technical debt is omitted, stakeholders and subcontractors are problematic or third party vendor products (COTS) don’t provide the required functionality. And even if all these areas are working well, a well-constructed schedule is useless unless it is properly used to execute the project.

To gain order, the massive amounts of information on a project can be organised into 10 information categories as shown in Figure 1. The categories are then arranged to show their relationships to form the Root Cause Analysis of Schedule Slippage (RCASS) model, as shown in Figure 2. These categories and relationships are adapted from McGarry [2] and Boehm [3] and are used as part of a Schedule Risk Compliance Assessment Methodology (SCRAM) to identify issues and quantify the risk to schedule slippage.

In the RCASS model shown in Figure 2, the forward direction of an arrow indicates that there is an effect of issues in one category upon another. All arrows eventually lead to the bottom of the figure and to the categories that are of main concern: Program Schedule & Duration and Project Execution. By uncovering issues in each category, it is possible to identify risks and problems to schedule compliance and the causes of delays.

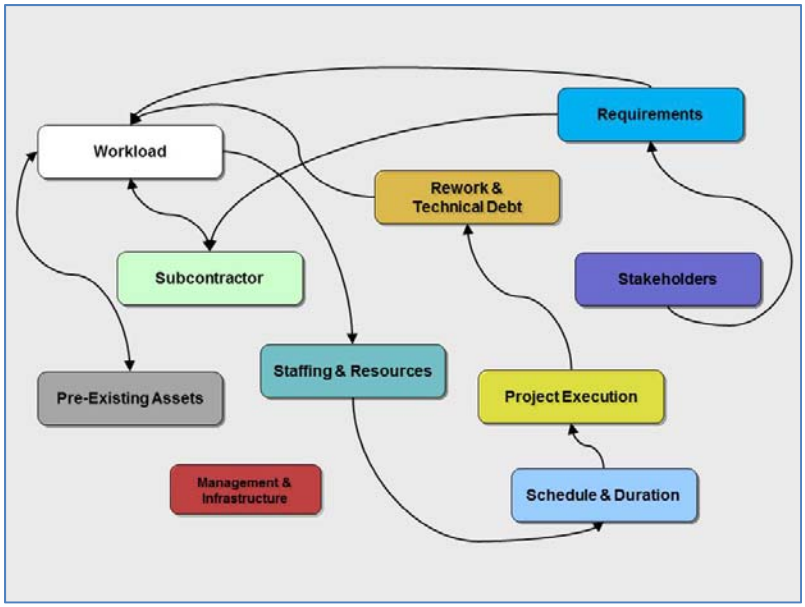

Figure 1 - Categories of Information

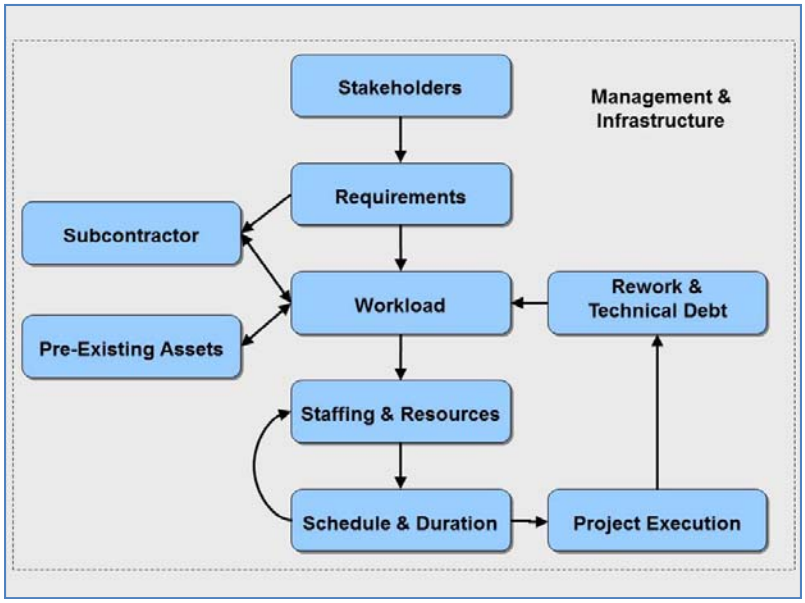

Figure 2 – RCASS Model

The following sections briefly describe each RCASS category and present some sample questions addressed by a SCRAM team. During a SCRAM assessment, the answers to these questions help to identify root causes of schedule slippage.

Stakeholders

Description: Issues in this category represent project turbulence and entropy caused by difficulties in synchronising the project's stakeholders.

Questions: Who are the stakeholders? How do they interact on requirements clarification, technical problems, and tradeoff analysis? Are one or more stakeholders imposing unrealistic constraints on implementation solutions or acceptance testing?

Requirements

Description: Issues in this category represent the understanding and stability of the functional and non-functional requirements, performance requirements, system constraints, standards, etc. used to define and bound what is to be developed.

Questions: Are all of the requirements defined and understood? Have the requirements been agreed to? Are there (Regulatory and Technical) standards that have to be implemented? Is there a mapping of requirements to development builds and production components? Are there technical performance requirements that are being tracked? Are the interfaces to other systems well understood?

In Figure 1, the arrow from requirements to subcontractors represents the handing off of program requirements to subcontractors so as to reduce the workload for the prime contractor. The arrow to Workload means that requirements are the basis of workload estimation and that workload increases with volatility or poorly defined requirements. Programs are often plagued with the IKIWISI (I'll Know It When I See It) approach to requirements definition and sign off which creates unplanned rework.

Subcontractor

Description: Issues in this category represent the subcontractor products or services that will be delivered as a part of the overall system. In Figure 1, the arrow from Subcontractor to Workload reflects additional work to correct poor quality products or handle late deliveries. Late products will cause other system components to be delayed having a ripple effect on workload and delivery schedules.

Questions: Are there subcontractors involved? When are their deliverables needed? How is subcontracted work coordinated, integrated and accepted? Are subordinate schedules aligned and integrated in a integrated Master Schedule? Are system interfaces well enough defined for the subcontractor to deliver a product that works within the system?

Pre-Existing Assets

Description: Issues in this category represent products developed independently of the project that will be used in the final product, i.e. an asset that reduces the amount of new work that has to be done on a project. In Figure 1, the arrow from assets to workload shows that incorrect assumptions about functional assets may impact the amount of work to be done.

Questions: What COTS, MOTS, NDI, or GFE[1] products are being used on the program? Are they providing the required functionality and are they meeting hardware constraints? Are

[1] COTS: Commercial Of The Shelf; MOTS: Modified Of The Shelf; NDI: Non-Developed Item (previously existing); GFE: Government Furnished Equipment

there legacy products being used and were they developed locally? Is the current product architecture defined and stable enough to evaluate and accept other pre-existing products? Do existing interface definitions accurately describe the actual product interface? What level of assurance accompanies the product? How will unused function or features be managed?

Workload

Description: Issues in this category represent the quantity of work to be done and provide a basis for estimating effort/staffing and duration. Issues with requirements, subcontractor products, functional assets, and rework may negatively impact this category.

Questions: Is the scope of work well understood? Is the workload changing for any reason, e.g. changing requirements, unstable platform or unplanned rework? Is workload being transferred to a later build? Workload is different depending on the development life cycle phase. Has the amount of work to be done been quantified, e.g. number of requirements, hardware and software configuration items or test procedures to be developed?

Staffing and Resources

Description: Issues in this category represent the availability, capability and experience of the staff necessary to do the work as well as the availability and capacity of other resources, such as test and integration labs. The arrow in Figure 1 points from staffing and resource to schedule because issues in this category may negatively impact the amount of time needed (schedule) to do the 'actual' work.

Questions: Are the right people (with the right experience) working on the program and are there enough people to do the work? Is the work force stable or is there turnover? Are the key personnel qualified to lead their area of work? Programs often suffer staffing issues related to high turnover, especially among experienced staff; bringing more people onto the program late making things worse.

Schedule and Duration

Description: This is a category of primary interest that is impacted by issues in the other categories. Issues in this category represent the task sequencing and calendar time needed to execute the workload by available staff and other resources (e.g. test labs).

Questions: What is the current schedule with respect to milestones, builds and phases? What are the dependencies, when are they due and are they linked into the schedule? What was the basis of estimates used to construct timelines, e.g. were analogous projects or parametric models used to estimate duration? Is there any contingency built into the schedule or is it success oriented? What is the "health" of the current schedule?

Project Execution

Description: Issues in this category stem from problems in communicating the schedule and monitoring and controlling the execution of the project in accordance with the project schedule. As shown in Figure 1, the capability to execute a project schedule is impacted by the feasibility and "health" of the schedule itself as well as by the effectiveness with which the scheduled tasks are executed. In relation to the latter issue of effectiveness, experience from multiple SCRAM assessments has highlighted the need to focus on Technical Progression and System Integration.

Questions: When was the schedule base-lined? Is it being used as a communication, monitoring and control tool? Is there an integrated master schedule? How is progress being tracked? Does actual productivity match the estimated or planned productivity? Does everyone on the project have access to the schedule (at an appropriate level of detail)? Are

System Integration and Formal Test phases conducted as discrete activities with specific objective entry and exit criteria? Is the system under development Technical Progression based on objective evidence of a maturing system and is the level of maturity commensurate with the resources and scheduled consumed?

Rework and Technical Debt

Description: Issues in this category represent additional work caused by the discovery of defects in the product and/or associated artefacts, as well as work that is deferred for short-term expediency (Technical Debt) and their resolution. Causes include rushing into development before requirements are fully understood, skipping inspections and verification testing due to lack of time, and deploying a product before the operating environment is ready. Technical Debt is often accrued with no plans to repay the debt until perhaps too late. The arrow in Figure 1 shows the disrupting impact that rework and technical debt has on workload.

Questions: Has the likely amount of rework been estimated and planned? Are the compounding consequences of incurring intentional Technical Debt identified and understood?

Management and Infrastructure

Description: This category impacts all of the above information categories. Issues in this category reflect the factors that impact the efficiency and effectiveness of getting work done, e.g. work environments and processes, use of management and technical software tools, management practices, etc. Efficiency is negatively impacted by a lack of tools, lack of facilities and burdensome security requirements. Effectiveness is negatively impacted by poor management practices such as in the areas of quality assurance, configuration management and process improvement.

Questions: Have the capacity requirements for the development system infrastructure (e.g. integration labs, network bandwidths etc.) been explicitly estimated based on an analysis of historical productivity and system under development operational performance needs? Is an active process improvement program in place that is driven by best practice assessment (e.g. CMMI)? Is the configuration management/change control system cycle time suitable to support development performance? Does the quality management system adequately support the program?

SCRAM Methodology.

SCRAM provides a methodology for conducting an independent review of risk to program schedule. Structured around the RCASS model described above, SCRAM has been used to find the root causes of schedule slippage and recommend improvements on programs that have experienced multiple or protracted schedule overruns. Moreover, SCRAM has proven extremely valuable in communicating schedule status and root causes of slippage to senior executives. Several recent SCRAM assessments found that schedule slippage was, in part, due to factors outside of the program's control. Once aware of these factors, executive management was able to bring about changes to facilitate resolution.

Figure 3 shows a high level overview of the SCRAM Assessment Process.

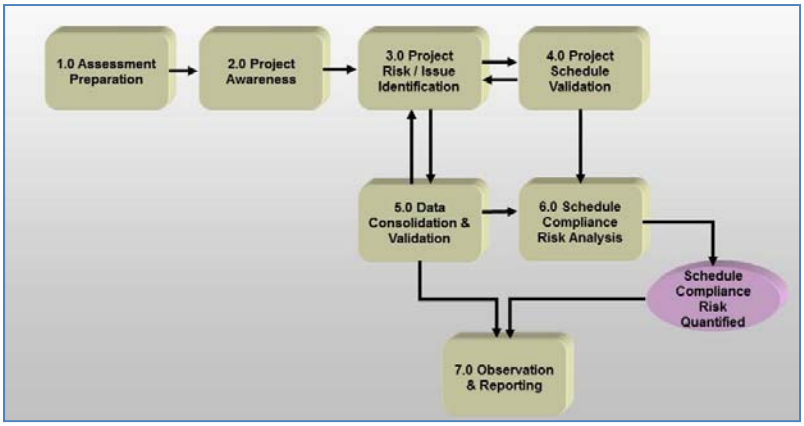

Figure 3 – SCRAM Assessment Process Overview

SCRAM reviews produces three types of outputs:

1. Identification and quantification of Schedule Compliance Risks (this includes identification of significant schedule drivers, root causes of existing slippage, risks to schedule and the potential impact on Program objectives)
2. The "health" of the current program and schedule
3. Recommendations for going forward

In quantifying the risk to schedule compliance, two specific methods are used; Schedule Risk Assessment and Parametric Modeling, discussed later in this paper.

A SCRAM assessment is conducted by a small team of highly experienced system and software engineering subject matter experts along with a schedule specialist. There are seven key principles for this review methodology:

- **Minimal Disruption:** Program information is collected one person at a time in an interview that usually lasts no more than one hour.
- **Rapid turn-around:** For major programs a SCRAM team typically spends one week on-site gathering information and data. A second week is spent consolidating, corroborating, analysing and modeling the data culminating with an executive presentation on the results. The RCASS model is used to structure the presentation to show the interrelationships (causes and effects). Finally, a written report is provided by the end of the fourth week.
- **Independence**: Review team members are organisationally independent of the program under review.
- **Non-Advocate**: All significant issues and concerns are considered and reported regardless of source or origin. The review does not favor the stakeholder, customer, end-user, acquisition office, or developer.
- **Non-Attribution**: None of the information obtained on an assessment is attributed to any individual. The focus is on identifying and mitigating risks to schedule.
- **Corroboration of Evidence:** The findings and observations of the review that are reported are based on at least two independent sources of corroboration.
- **Openness and Transparency:** For the Monte Carlo analysis or software parametric analysis component of a SCRAM review, the developer is invited to assist in resolving data anomalies, witness the analysis process and challenge model results. This transparency builds cooperation, trust and confidence in the schedule forecast. However the SCRAM Team is the final arbiter.

Interviews are conducted with key personnel, both customer and developer, and questions are structured around RCASS categories. Interview responses are tagged to the relevant RCASS category. The review includes the examination of development plans, management and product artifacts, risk databases and the schedule health check discussed earlier.

Schedule Risk Assessment and Parametric Modeling

Schedule Risk Assessment

During a SCRAM Review a schedule health check is performed to evaluate the quality of a schedule to determine it's suitability for running a Monte Carlo simulation. The health check examines the construction and logic of the schedule under review and includes an analysis of the schedule work breakdown structure, logic, dependencies, constraints and schedule float. Tasks are allocated three-point estimates based on the assessed level of risk. Risks and problems identified from each of the RCASS categories discussed above provide input into these probability estimates. The three-point estimate (pessimistic, optimistic, most likely) can be applied with either a generic risk multiplier (derived from past experience) across all like tasks or a risk factor based on a task-by-task risk assessment. A Monte Carlo analysis is then performed on the critical path and near critical path tasks and work packages in the schedule; an example of the output of this type of analysis is shown in Figure 4.

The result of the Monte Carlo analysis is a distribution showing the percentage probability of achievement for any planned delivery date. If the planned program delivery is on the left side of the program completion distribution curve, there is cause for concern, depending on the degree of risk the stakeholders are prepared to accept. Projects should use the results of the analysis to develop mitigation plans to ensure that the risks don't become reality.

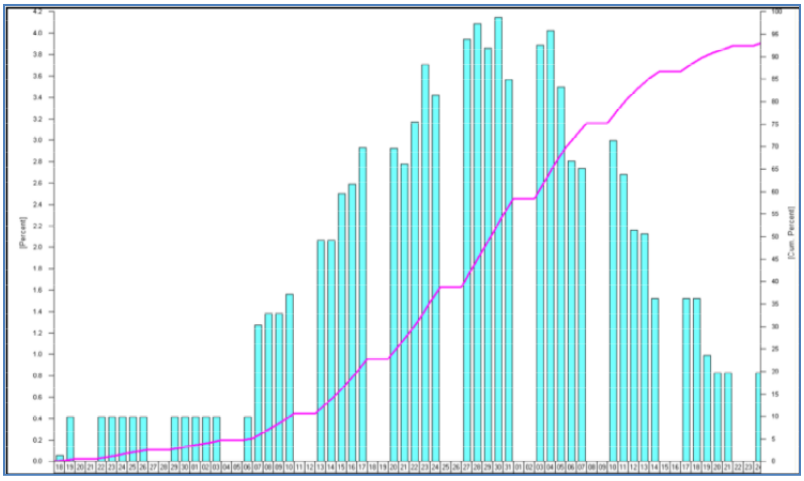

Figure 4 - Monte Carlo Schedule Analysis

Another consideration of a SCRAM schedule health check is the allocation of schedule contingency. Some contingency is recommended for the inevitable rework. It is important to have some schedule contingency distributed throughout the schedule accompanying the higher risk tasks instead of a cumulative buffer at the end of the schedule before delivery or held as management reserve. This will allow some slippage to occur during development without disrupting subsequent successor task(s) scheduling.

Software Parametric Modeling

SCRAM can be applied at any point during the system engineering or project lifecycle. For the software development elements of a program, a schedule forecast tool is used to assess existing schedule estimates. SCRAM includes this forecasting activity because software is a

common schedule driver for complex systems and software durations are almost always optimistic. While SCRAM is not dedicated to a specific forecasting tool, the preference is to use a tool that uses objective software metric data 'actuals' that reflect the development organisation's current performance or productivity.

The inputs to the model are size (usually estimated source lines of code and actual code complete to date), major milestones planned and completed, staffing planned and actual, and defects discovered.

Figure 5 below shows a typical output from a modeling tool.

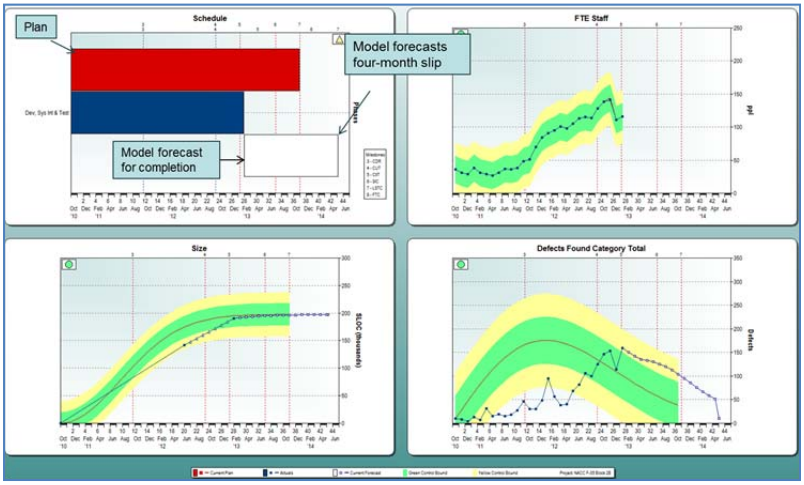

Figure 5 - Parametric Modeling

Elements of the SCRAM Product Suite

In addition to the RCASS Model described in this paper, further elements of the SCRAM Product Suite include:

- SCRAM Process Reference / Assessment Model (PR/AM) (Relates processes and best practices to the relevant RCASS category) conformant with ISO/IEC 15504 [4]
- SCRAM PR/AM Model and Assessor Training Courses
- SCRAM Assessor Guidebook

The PR/AM is available for download from www.scramsite.org. Additional details about SCRAM can also be found at this website

SCRAM Application

There are three potential areas of SCRAM application:

***Pro-Active SCRAM or P-SCRAM*:** Conducted at or immediately prior to or shortly after Contract (e.g. at Integrated Baseline Review) to ensure the systemic issues covered by SCRAM are avoided.

***Monitor SCRAM or M-SCRAM*:** Conducted at regular intervals to monitor all categories for status and new risks, i.e. provide program health checks to support appropriate gate or progress reviews.

***Diagnostic SCRAM or D-SCRAM*:** Conducted on challenged programs or programs of concern. The methodology is used to assess the likelihood of schedule compliance and identify root causes of schedule slippage. Recommendations are made to remediate or mitigate the issues and risks respectively.

References

[1] Lars Mieritz, “Survey Shows Why Projects Fail”, Gartner Group, 2012.
[2] John McGarry, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Fred Hall, “Practical Software Measurement: Objective Information for Decision Makers,” Addison-Wesley, 2001.
[3] Barry Boehm, “Section 2: Risk Management Practices: The Six Basic Steps,” from Software Risk Management, IEEE Computer Society Press, 1989.
[4] International Organization for Standardization; ISO/IEC 15504.2:2003 – Information Technology Process Assessment – Part 2: Performing an assessment