Using Earned Value Management (EVM) in Spiral Development

Lisa Brownsword
Jim Smith

June 2005

Integration of Software-Intensive Systems Initiative

Technical Note
CMU/SEI-2005-TN-016

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Contents

List of Figures

List of Tables

Abstract

Earned Value Management (EVM) helps managers to plan, monitor, and control the development and evolution of custom developed software-intensive systems. EVM traditionally assumes a waterfall development model. However, to meet the demands for today's complex, dynamic systems, certain trends have emerged. First, projects no longer develop all components of a system as custom components. Instead, projects use pre-existing, off-the-shelf packages, components, or entire systems, potentially along with custom components. A second trend is a realization that often requirements are not known in detail at the start of a project and must evolve efficiently in response to changing needs and technology. A further trend (often in response to the first two trends) is the move to other development models, such as spiral or iterative development processes. Spiral development processes can better support 1) the required discovery of what users want and 2) negotiation to reconcile what engineers can quickly and reasonably assemble from pre-existing and custom components.

While projects have applied EVM to spiral development projects, the results have not been uniformly satisfying. This report explores the fundamental challenges in using EVM with spiral development processes and proposes adaptations to some EVM principles to render it more suitable for today's software-intensive systems.

1 Introduction

The increasing complexity and capabilities of today's software-intensive systems have brought forth several trends in how projects approach the development and evolution of systems. First, projects are leveraging pre-existing components and systems as part of the delivered solution. Sources of these components include commercial off-the-shelf (COTS) products or packages, open source, free ware, reuse libraries, and legacy components or systems. A second trend is the realization that often all requirements cannot be defined in sufficient detail at the start of a project. Rather, requirements must be discovered and negotiated as part of forming a cost-effective and feasible solution. A further trend—often emerging in response to the first two trends—is the move to spiral development approaches.

Earned value management (EVM) is a recognized project management approach that integrates the technical, schedule, and cost parameters of a project [Wilkins 99]. The concept of EVM has existed since the 1960s, and has seen proven use in projects ranging from very large, complex systems to small-scale development efforts [Abba 97]. When properly applied, EVM provides project managers with key information to answer the fundamental question, "How much progress have I made against my original plan?" The validity of the plan, and the means to objectively measure against that plan, are paramount to the success of EVM on any project.

In working with projects that are applying current development trends such as spiral development to build today's complex software-intensive system, we find that project managers and oversight executives are struggling to use EVM. While EVM has been used on spiral development projects, the results have not been uniformly satisfying. This report explores several critical challenges in using EVM with spiral development and proposes adaptations to some of the principles of EVM.

1.1 Intended Audience

This report is intended for project managers, program managers, and oversight executives in development, maintenance, or acquirer organizations who need to create, use, or monitor integrated baselines using earned value. Policy makers will also find the information particularly relevant.

1.2 Using This Report

Section 2 of this report provides an overview of EVM. It summarizes the purpose, key concepts, and important mechanisms of EVM. This information is intended as background, not as a tutorial on the subject. Section 3 notes development and management issues

associated with spiral development. Again, the information is not intended as a tutorial on the subject. Section 4 proposes interpretations and adaptations of EVM for spiral development and Section 5 provides concluding remarks.

2 Earned Value Management

2.1 Overview of EVM Concepts

EVM projects are managed through the establishment of a *performance management baseline* that represents the work to be performed along with the needed resources and schedule. The fundamental requirement for using EVM on a project is to plan **all** work prior to beginning development [Alexander 98]. Project progress is measured as *earned value* against the performance management baseline.

EVM focuses a project manager on answering five essential questions:

1. What is the value of the work planned? – Budgeted Cost for Work Scheduled (BCWS)
2. What is the value of the work accomplished? – Budgeted Cost for Work Performed (BCWP)
3. How much did the work cost? – Actual Cost of Work Performed (ACWP)
4. What was the total budget? – Budget at Completion (BAC)
5. What do we now expect the total job to cost? – Estimate at Completion (EAC)

In an EVM project, work required to complete the project is arranged into a tree structure that successively subdivides pieces of work—a *work breakdown structure* (WBS)—where the "leaves" of the tree are individual work packages and planning packages. Near-term work is divided into *work packages* that are manageably sized pieces of work that can be planned in detail, covering technical content, budget, and schedule. Far-term work is divided into *planning packages* that have little detail. Over the course of the project, planning packages are refined into work packages with the necessary content and detail. Work and planning packages are assigned start and end dates and arranged across the project time line. Thus, a WBS identifies all significant work and provides a framework to assign responsibilities, schedule, and budget. A skeletal WBS is shown in Figure 1. Typically the WBS is structured by the subsystems and components that will form the system solution. Development or maintenance tasks, such as requirements, design, coding, or testing, are aggregated for each component. Thus, the WBS is based on the system solution structure and can be characterized as a *product-oriented* WBS.

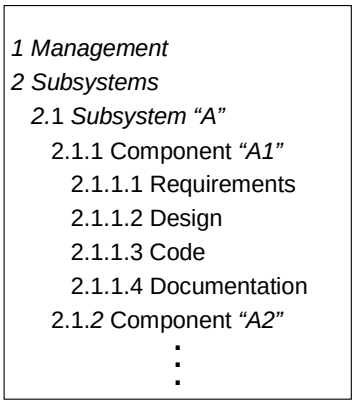

Figure 1: Fragment of Traditional WBS

The performance measurement baseline (BCWS) is formed by summing the value of all work packages and planning packages over time, as shown in Figure 2. The budget at completion (BAC) is the estimated or planned cost at the completion date. The BAC plus a management reserve forms the total allocated budget (TAB) available to a project. The management reserve is not designated for a specific task but is available to respond to unknown events.

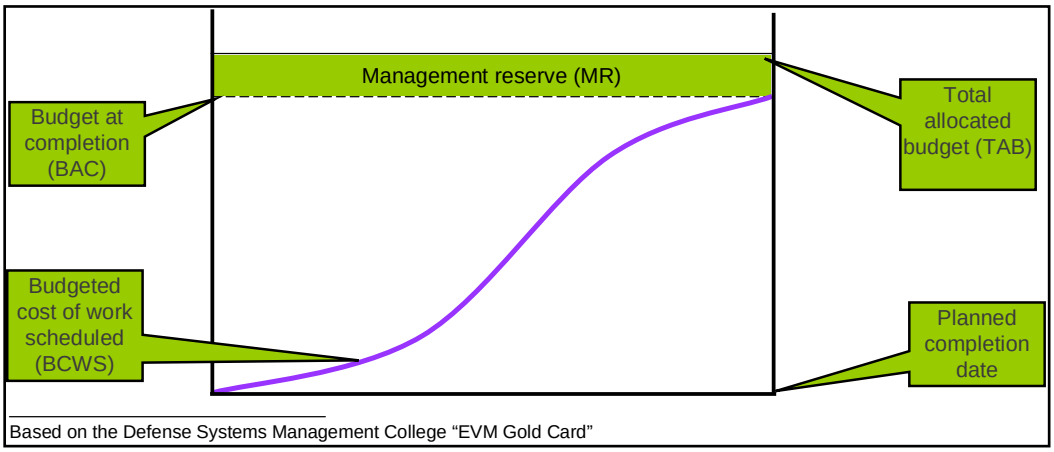

Figure 2: Performance Baseline

Progress against the plan is determined by comparing the earned value to the baseline at any point in time. Numerous value methods exist to determine the earned value of a work package, such as

- weighted milestones, interim milestones, or percent complete for discrete tasks with an end product or result
- apportioned effort for tasks such as quality control or peer reviews
- level of effort (LOE) for non-discrete tasks such as coordination that have no specified end product or result

The baseline is thus the plan against which the actual performance (BCWP) and cost (ACWP) of the project are compared. A negative cost variance results when the ACWP is greater than the BCWP at a given point in time, and indicates that a project is overrunning its development budget. Correspondingly, an unfavorable schedule variance results when actual costs exceed the budget for completed work. Figure 3 depicts these relationships.

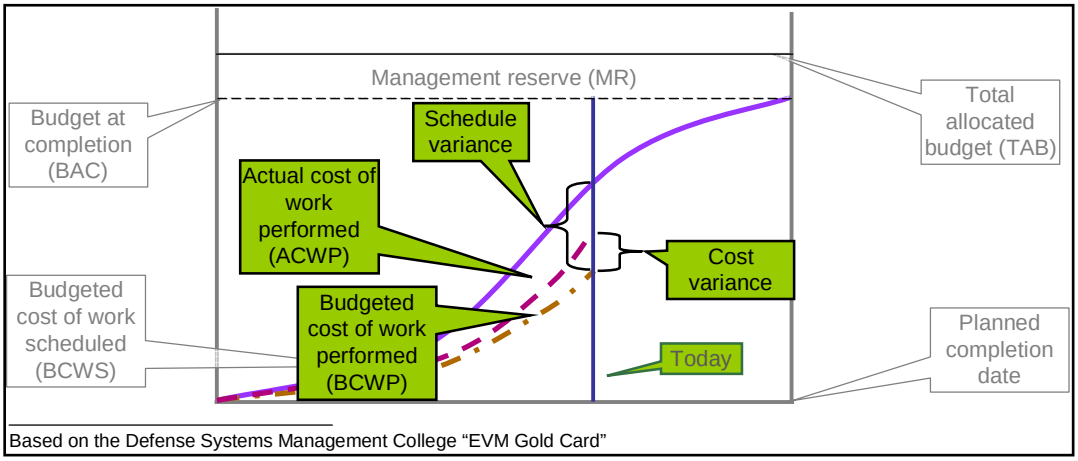

Figure 3: Relationships of Actual to Plan

2.2 Implications of EVM

For traditional EVM to work effectively, the following must be true:

- Time phasing of the work packages must be accurate.
- The relation of work packages to the life-cycle approach for a project must be appropriate.
- The performance baseline must be objective and verifiable.
- Earned value (BCWP) must accurately represent progress. For example, use of an arbitrary weighted milestone method where 50% of the value is earned when a work package begins, and the remainder earned when the work package is completed, is usually only appropriate for work packages of two months or less.
- Inappropriate—and excessive—use of level of effort should be avoided. For example, level of effort (LOE) is not appropriate for calculating the earned value of software development work packages.

EVM was designed for, and has been used primarily with, projects following a waterfall development process. The conventional WBS and earned value methods used by many projects reflect a number of essential assumptions of a waterfall development model. These assumptions include the following: 1) requirements can be defined prior to the start of a project (or very soon thereafter); 2) there are no high-risk elements of the requirements or the solution; and 3) with good requirements management, the requirements will not change significantly [Boehm 00a]. It is also assumed that the high-level structure or architecture for

the system can be defined prior to the start of the project. Further, the activities necessary to define, build, field, and evolve the system can also be well defined—with much of their definition in the early project planning phases of a project. These assumptions are summarized in Table 1.

Table 1: Essential Assumptions of Waterfall Development Model

Waterfall Development Model Assumptions
Requirements are knowable in advance of development.
Requirements have no unresolved high-risk implications.
Nature of requirements will not change very much during development or maintenance.
Requirements are compatible with all key system stakeholders' expectations.
The "right" architecture for implementing the requirements is knowable in advance.
There is sufficient calendar time to proceed sequentially.

3 Spiral Development

3.1 Overview of Spiral Development Concepts

The notion of spiral development emerged in the mid-1980s in response to a number of fundamental problems with the conventional use of the waterfall development model. Spiral development was first codified in the literature by Boehm and has been used in various forms throughout the world on software-intensive projects of all sizes and complexity for all domains [Boehm 88].

The spiral development model is focused on these essential problems:

- late identification and resolution of critical risks (including requirement and design flaws)
- incorrect or inflexible statements of user needs and requirements (often leading to requirements churn)
- delays in integration and testing (as modules do not readily integrate)
- late discovery (often not until the system is in full operation) of poor system performance, poor quality, low end-user satisfaction, or unmet business needs
- no options, or limited options, for early deployment
- difficult-to-maintain systems
- significant unplanned work (including rework)

Boehm describes spiral development [Boehm 00b] as a risk-driven process model generator for guiding multi-stakeholder concurrent engineering of software-intensive systems. He characterizes its distinguishing features as

1. a cyclic approach that incrementally grows the definition and implementation of a system
2. a set of anchor point milestones that ensures continued stakeholder commitment and feasibility of the incremental definitions and implementations

These features are graphically shown in Figure 4 where successive spirals are used to grow the system definition from an idea to fielded products over time. The evolving definition typically occurs within the context of phases, which are shown in the figure using the Boehm and the Rational Unified Process (RUP) [Boehm 96, Kruchten 04] designated phases of inception, elaboration, construction, and transition. Anchor point milestones, labeled as Life Cycle Objectives, Life Cycle Architecture, and Initial Operational Capability, establish and

manage completion criteria for progressing from one phase to the next. Ongoing commitment of all affected stakeholder groups is a key criterion for each anchor point.

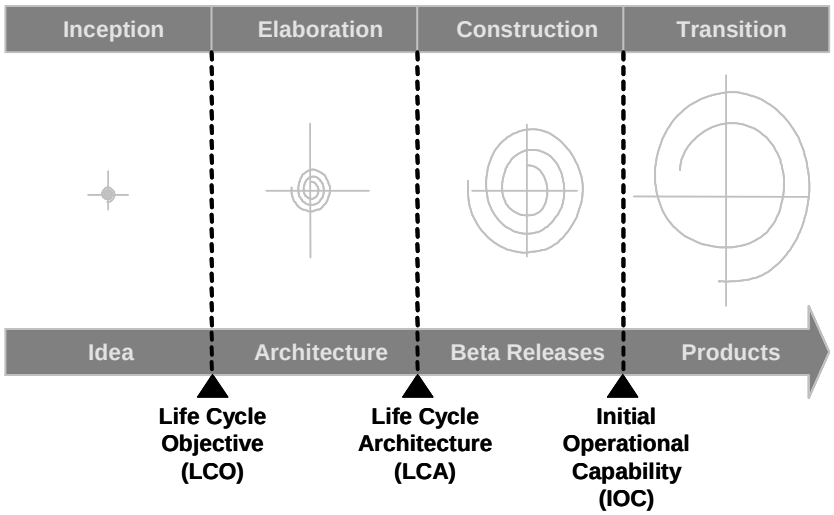

Figure 4: Incrementally Growing a System (adapted from [Royce 98])

The activities within the inception phase focus on discovery and negotiation among stakeholders to form a feasible scope for the project and culminate with the Life Cycle Objective (LCO) anchor point milestone. The elaboration phase activities are oriented toward forming and validating a feasible architecture for the solution, which concludes with the Life Cycle Architecture (LCA) anchor point milestone. The construction phase focuses on creating beta releases of the system and demonstrating readiness for the user community with the Initial Operational Capability (IOC) anchor point milestone. The activities of the transition phase focus on creating and moving production and maintenance releases to the user community.

A pass through the four phases forms a development cycle, resulting in a generation of the system and its capabilities. Each successive pass through the four phases produces another generation—with new or modified capabilities. In this way, a system is incrementally and iteratively defined and implemented in a systematic and managed approach.

Over the past decade, as individuals (including Dr. Boehm) and organizations have expanded upon the original concept, variations in development model structure, operation, and terminology have arisen. At a spiral development workshop in 2000, Boehm offered a list of six essential elements—or invariants—that **all** must be present for a process to be regarded as following the spiral development model [Boehm 00a]:

1. Key artifacts are developed concurrently (operational concept, requirements, design, code, plans).
2. To proceed, each spiral must cover all of these: objectives, constraints, alternatives, risks, review, and commitment.

3. Level of effort is driven by risk considerations.
4. Degree of detail is driven by risk considerations.
5. Use of anchor point milestones (Life Cycle Objectives, Life Cycle Architecture, and Initial Operational Capability) serve as commitment and progress checkpoints.
6. Emphasis is on system and life-cycle activities and artifacts.

In summary, spiral development approaches share a number of key characteristics. The system is defined, refined, and developed in multiple, risk-driven spirals (also referred to as *iterations* in some spiral processes, such as the Rational Unified Process). Anchor points bound phases. Several spirals are typically used to achieve the criteria for an anchor point milestone. The nature and impact of risks (technical, management, operational, and business) drive the process: the number of spirals within each phase, the level of effort in each spiral, and the level of detail within each spiral. High-priority risks that often cause major perturbations in requirements, architecture, design, and operational use are identified and confronted early in the project.

Each spiral includes all the management, engineering, and support activities in varying proportions depending on the risks. Each spiral expands the knowledge about the emerging system and produces some type of an executable representation of the evolving system. This helps stakeholders to reason effectively about tradeoffs and to make decisions regarding what is needed and what can be reasonably built, fielded, and maintained within the constraints on the system.

3.2 Implications of Spiral Development

Basically, a spiral development approach focuses on practices to ensure that a project solves the **right** problem and builds the **right** system—from the perspective of all affected stakeholders. Spiral development approaches are well suited for today's systems when the general "goal" of the system is known, but the exact definition of the system is not.

Use of spiral development makes a number of crucial assumptions (also summarized in Table 2):

- System goals and very high-level requirements are known prior to development—but requirements are discovered and refined during the development process. Premature decisions are consciously avoided.
- Risks are continually discovered throughout the development cycles; these risks drive the development process and are used to determine "how much" engineering and artifacts are needed.
- Requirements, and the stakeholders' understanding of them, will change throughout the development—and for very good, legitimate reasons. The changing nature of requirements is acknowledged with processes in place to manage the changes.

- All affected stakeholders must remain committed throughout the development and evolution of the system. Mechanisms, such as executable representations (e.g., prototypes, early releases) and anchor point milestones, are used to gain and validate stakeholder agreements.
- The "right" architecture is incrementally formed and validated through executable representations of the architecture. Thus the architecture is typically not known at the start of a project.
- Time (schedule) is treated as an independent variable in engineering and management decisions. Understanding of program cost and schedule is continually refined as a greater understanding of needs and feasible solutions is determined. As a result, plans are continually refined.

Table 2: Essential Assumptions of Spiral Development Model

Spiral Development Model Assumptions
Requirements are discovered and refined during development.
Risks are continually discovered and high-priority risks drive the development process.
Requirements will continue to change during development or maintenance.
Actual requirements fulfilled and key system stakeholders' expectations will need continual negotiation.
The "right" architecture for implementing the requirements is not known in advance.
Plans (including cost and schedule) are continually refined as a better understanding of the requirements and feasible solutions is gained.

4 Analysis and Recommendations

In this section we first summarize several key differences between the waterfall and spiral development models and how those difference give rise to problems when EVM is applied "as-is" to a spiral development project. We then provide alternatives for the work breakdown structure and earned value measures to address the noted deficiencies.

4.1 Comparison of Spiral and Waterfall Development Models

Earned value management is well suited for projects using a waterfall development model for its life-cycle processes. To better understand the issues involved in applying EVM in projects using a spiral development model, we first compare the differences between waterfall and spiral development models. The left side of Table 3 lists a number of the key assumptions that underlie the waterfall development model (same elements as summarized in Table 1). For each assumption of the waterfall model, we provide an accompanying characterization from the perspective of the spiral development model in the right column (same elements as summarized in Table 2).

Table 3: Waterfall and Spiral Models Compared

Waterfall Development Model [Boehm 00b]	***Spiral Development Model***
Requirements are knowable in advance of development.	Requirements are discovered and refined during development.
Requirements have no unresolved high-risk implications.	Risks are continually discovered and high-priority risks drive the development process.
Nature of requirements will not change very much during development or maintenance.	Requirements will continue to change during development or maintenance.
Requirements are compatible with all key system stakeholders' expectations.	Actual requirements fulfilled and key system stakeholders' expectations will need continual negotiation.
The "right" architecture for implementing the requirements is knowable in advance.	The "right" architecture for implementing the requirements is not known in advance.
There is sufficient calendar time to proceed sequentially.	Plans (including cost and schedule) are continually refined as a better understanding of the requirements and feasible solutions is gained.

The following demonstrates how these differences in assumptions between the two development models can affect the use of EVM. First, consider requirements management in a waterfall development project. Using the conventional approach of a product-oriented WBS, work packages are created for a requirements definition task in each system, subsystem, and lower level component that makes up the developed solution. The earned

value of these work packages is based on the portion of the total requirements that have been finalized or allocated, which is often a straight percentage. This earned value measure is then, in turn, tied to various engineering and management review exit criteria (e.g., 80% of the requirements definition completed prior to preliminary design review). Given the assumption in a waterfall development model that the requirements can be unambiguously defined prior to the start of development, that there are no unforeseen risks, and that the architecture will remain constant—in other words, that the project has a stable program management baseline—then this approach results in a reasonable definition of earned value.

On the other hand, if a project is using a spiral development model, then it is inappropriate to define all of the requirements "up front." Attempts to use the same approach for determining the earned value of the requirements management WBS elements as in the waterfall development model (i.e., percentage of requirements defined and/or allocated) results in endless cycling between high earned value as requirements are defined and low earned value as it is discovered that the requirements thus defined are incorrect, inadequately understood, or not agreed-upon by all of the relevant stakeholders.

To be meaningful for spiral development, earned value must be based on a WBS that reflects its inherent evolutionary and discovery nature, particularly in the earlier stages of a project. The next section discusses the characteristics of such a WBS and makes some recommendations about an alternative approach.

4.2 Work Breakdown Structures

While the majority of the published experience in using EVM has focused primarily on projects using a conventional waterfall development model, there has been some research and use of EVM with spiral development. Walker Royce proposed a WBS that is centered on the elements of the spiral process—a *process-oriented* WBS—that forms a sound foundation for spiral development [Royce 98]. The Royce WBS is organized as follows with a partial outline in Figure 5 to illustrate the approach:

- Level 1 indicates a major discipline, usually allocated to a single team such as management, requirements definition, analysis and design, or implementation.
- Level 2 indicates a phase of the life cycle, such as inception, elaboration, construction, or transition.
- Level 3 indicates key tasks that produce principal artifacts within a given phase and discipline, such as use cases in the inception phase for the requirements discipline.

1. Management
2. Environment
3. Requirements
    3.1. Inception phase requirements development
        3.1.1. Vision specification
        3.1.2. Use Case modeling
    3.2. Elaboration phase requirements baselining
        3.2.1. Vision baselining
        3.2.2. Use case model baselining
    3.3. Construction phase requirements maintenance
    3.4. Transition phase requirements maintenance
4. Design
    4.1. Inception phase requirements development
    4.2. Elaboration phase requirements baselining
    4.3. Construction phase requirements maintenance
    4.4. Transition phase requirements maintenance

    . . .

Figure 5: Royce WBS for Spiral Development (partial outline)

The Royce WBS addresses a key issue with the conventional product-oriented WBS. The structure of the Royce WBS is independent of the eventual architecture of the system. This feature is crucial for projects using a spiral development since the architecture is typically not known at a sufficient level of granularity to organize, plan, and track the necessary work for a project. Thus the Royce WBS provides a sound foundation. The following section outlines several improvements to that base.

4.3 Alternative Work Breakdown Structure

While the Royce WBS approach infers that the elements in the WBS support the attainment of phase anchor point milestones, it does not **directly** link the objectives and exit criteria for a phase with the WBS elements. In our experience, it is often difficult for project managers and other staff to reason about the appropriateness of WBS tasks toward meeting the project's goals. Our proposed structure seeks to rectify this situation in two ways. First, we propose to change the structure of the WBS such that Level 1 is the phase with an explicit statement of the primary focus of the phase objective and of how to achieve the corresponding anchor point milestone. Level 2 would then indicate the development disciplines that would be carried out during the phase. Secondly, we propose the addition of explicit WBS elements at Level 3 that capture the essential tasks required to achieve the phase exit criteria. A partial outline of our revised WBS for spiral development projects is summarized in Table 4. For the purposes of this report, the table provides partial details of the revised WBS outline for requirements.

Table 4: Revised Phase-Oriented WBS for Spiral Development

1. Inception phase [Focus = demonstrate feasible scope] 1.1. Business modeling 1.2. Requirements 1.2.1. Vision specification 1.2.2. Critical requirements modeled (e.g., use cases) 1.2.3. Critical non-functional quality attributes (e.g., reliability) characterized with stakeholder consensus 1.3. Analysis and design 1.4. Implementation 1.5. Test 1.6. Deployment 1.7. Configuration and change management 1.8. Project management 1.9. Environment
2. Elaboration phase [Focus = demonstrate valid architecture] 2.1. Business modeling 2.2. Requirements 2.2.1. Significant use cases modeled 2.2.2. Significant non-functional quality attributes characterized with stakeholder consensus 2.3. Analysis and design 2.4. Implementation 2.5. Test 2.6. Deployment 2.7. Configuration and change management 2.8. Project management 2.9. Environment
3. Construction phase [Focus = demonstrate initial production release] 3.1. Business modeling 3.2. Requirements 3.2.1. Remaining use cases modeled 3.2.2. Monitor attainment of non-functional quality attributes 3.3. Analysis and design 3.4. Implementation 3.5. Test 3.6. Deployment 3.7. Configuration and change management 3.8. Project management 3.9. Environment
4. Transition phase [Focus = demonstrate full deployment releases] 4.1. Business modeling 4.2. Requirements 4.2.1. Update use cases as components change 4.2.2. Monitor attainment of non-functional quality attributes 4.3. Analysis and design 4.4. Implementation 4.5. Test 4.6. Deployment 4.7. Configuration and change management 4.8. Project management 4.9. Environment

The first item to note in Table 4 is the revised Level 1 and its identifier. In contrast to the Royce WBS where Level 1 indicated a major discipline, such as requirements definition, our revised WBS uses Level 1 to organize all tasks associated with a given phase and achieving the corresponding anchor point milestone. We also recommend stating explicitly the primary

focus of the phase in the identifier label. For example, for the inception phase the primary goal is the formation of the scope for the project that is demonstrated to be feasible. Fundamentally, this is the purpose of the Life Cycle Objective anchor point milestone. We would thus use **Inception phase [Focus = demonstrate feasible scope]** to capture this information. The intent is to provide constant and visible awareness of what all sub-elements should be focused on achieving. While on the surface this may seem to be a trivial modification, our experience indicates that it is all too easy for project managers to lose sight of project priorities.

Returning to the structure of our revised WBS, the Level 2 elements then represent the typical system development disciplines (similar to the Royce WBS), such as project management or requirements definition, that collectively are needed to achieve the objectives for a given phase. The Level 2 elements would then each organize the lower level activities and tasks that are associated with that discipline in support of achieving the phase goal.

A further refinement of the Royce WBS is to reflect more of the anchor point milestone criteria into the Level 3 elements themselves. For example, to define a project scope that is feasible in the inception phase, project staff must identify and understand the architectural and implementation implications of the most critical functional and non-functional requirements—but not necessarily all of the requirements (future spirals in the elaboration phase will explore less critical requirements). Identifying the critical requirements (and gaining consensus on their priority) is an essential risk mitigation approach and is the basis of a spiral development approach. The Royce WBS elements provide limited indication of which requirements should be addressed in the inception phase. Our revised WBS makes this more explicit by refining the Level 3 element to model the critical requirements.

4.4 Earned Value Determination

The previous section described an alternative WBS that we assert provides a more rational basis for planning and managing a spiral development effort. Just as this WBS ties the activities to the phase objectives and exit criteria, the definition of earned value must be linked to progress towards attaining those goals. This section will introduce residual risk as a way to measure progress, and illustrate its use in constructing a risk-oriented baseline and measuring earned value.

As shown in Figure 6, the emphasis during the inception and elaboration phases is on reducing the unknowns associated with the scope for delivered capability, achievable requirements, and feasibility. Armour notes that the focus of the development effort is not on the software product itself but, rather, on knowledge acquisition and “ignorance reduction” [Armour 00]. Essential to this focus is understanding and agreement upon the priorities for resolving these unknowns; not all risks are equally important. Boehm uses the term “value-based software engineering” to describe this process of incorporating value considerations into all aspects of software development [Boehm 03].

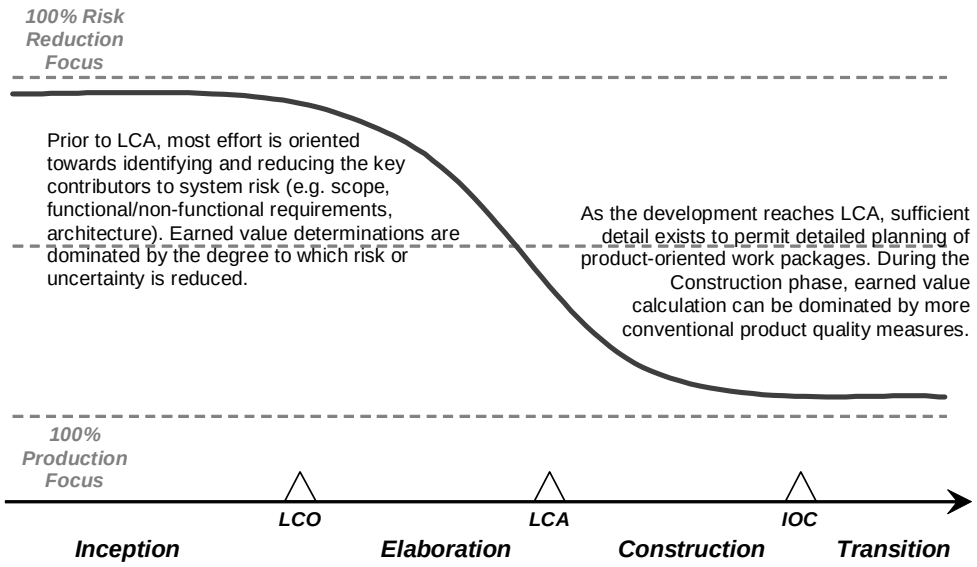

Figure 6: Shift in Focus of Spiral Development Activities

Successful completion of the inception phase occurs with the Life Cycle Objective (LCO) milestone. Recall that the inception phase is characterized by formation of a feasible scope for the project that promotes stakeholder definition and negotiations to identify, characterize, and prioritize key system functionality, quality attributes, and risks. The degree to which there is consensus on these goals can be used to represent progress towards LCO. Thus, lack of consensus represents a possible definition of "residual risk." It is in these early phases that conventional earned value measures—and hence, baseline definitions—fail to adequately reflect progress.

We assert that during the construction and transition phases, conventional WBS and baseline definitions, and associated earned value measures, provide reasonable insight into development progress. This is precisely because the reductions in uncertainty and, therefore, risk during the inception and elaboration phases make it possible to define a **useful** product-oriented baseline. Therefore, we will not dwell on the construction and transition phases in this report—there are innumerable textbooks, courses, and experts available to help a program in this area. Instead, the next sections will describe how the use of a risk-oriented baseline and earned value determined by residual risk reduction can provide a more meaningful representation of actual progress during the inception and elaboration phases.

4.5 Risk-Oriented Baseline

To measure progress, there must be something to measure progress against: a baseline. As previously discussed, a conventional EVM baseline is determined by summing the value (i.e., cost) of the WBS elements over time. By analogy, a risk-oriented baseline should reflect the cumulative residual risk reduction of the WBS elements over time. However, instead of

simply imputing the value of a work package as its cost, relating progress within individual WBS elements to the entire program requires that **total** development risk be allocable across the WBS elements. Moreover, the risk must be allocated in some fashion that reflects the relative contributions of individual activities to the total development risk. This is an inexact science, but techniques such as Boehm's Wideband Delphi technique provide a structured, consensus-driven process for making reasoned judgments in the absence of quantifiable data [Boehm 81].

Once the total development risk has been allocated down to the WBS element level, a "normalized" value for each task within an element may be defined as its BCWS (i.e., cost) adjusted by a factor that represents that element's contribution to the total development risk:

$$\left|BCWS_i\right| = \left(BCWS_i \cdot eRisk_i\right) \cdot \frac{\sum_{j=1}^{n} BCWS_j}{\sum_{k=1}^{n} \left(BCWS_k \cdot eRisk_k\right)} \quad (1)$$

In Equation 1, the term $\sum_{j=1}^{n} BCWS_j$ is the sum of the costs for all WBS elements. The term $\sum_{k=1}^{n} \left(BCWS_k \cdot eRisk_k\right)$ is the sum of the WBS element costs multiplied by the portion of the total development risks allocated to each element (the "risk-weighted" costs); the sum of the element costs divided by the sum of the risk-weighted costs results in a weighted normalization factor. Finally, the normalized cost for a particular WBS element is simply the element's cost multiplied by its allocated risk, multiplied by the weighted normalization factor. This can best be illustrated through a simple example:

> A project has three tasks, and the budgeted costs are
>
> $Task_1 = 100$
>
> $Task_2 = 200$
>
> $Task_3 = 200$
>
> The total development risks are allocated thus:
>
> > $Task_1$ represents 40% of the total development risk (based on the criticality of the results of $Task_1$ to the overall development success).
> >
> > $Task_2$ is allocated 20% of the total development risk.
> >
> > $Task_3$ is allocated 40% of the total development risk.
>
> The sum of the WBS element costs is 500; the sum of the risk-weighted costs is 160. Therefore, the normalized costs for the three tasks are
>
> $|Task_1| = (100 \cdot 0.4) \cdot (500 \div 160) = 125$

$|Task_2| = (200 \cdot 0.2) \cdot (500 \div 160) = 125$

$|Task_3| = (200 \cdot 0.4) \cdot (500 \div 160) = 250$

Note that the sum of the risk-normalized costs equals the sum of the budgeted costs. As shown by this simple example, $Task_1$ has the same risk-normalized cost as $Task_2$, despite the difference in their budgeted costs: this reflects the "true" value of the contributions of each individual task towards the complete development.

Next, the earned value of each work package is determined over time based on the reduction in residual risk obtained throughout the performance of the activity, as shown:

$$EV_T = \left|BCWS_i\right| \times \left(\frac{1 - rRisk_T}{1}\right),\ rRisk_T \xrightarrow{\lim} \delta \tag{2}$$

In Equation 2, the term $rRisk_T$ represents the remaining—or residual—risk through the accomplishment of the task to time *T*, and ranges from 1 before the task begins (when there has been no risk reduction) to some acceptable threshold value (δ) at the completion of the task (when all the allocated risk has been eliminated). So, at time *T*, the risk-normalized earned value for a task (EV_T) equals its risk-normalized cost ($|BCWS_i|$) multiplied by the percentage of residual risk reduction.

To complete the baseline, the normalized work package earned values are summed over time:

$$BCWS_T = \sum_{j=0}^{T} \sum_{i=1}^{n} EV_{ij} \tag{3}$$

In other words, Equation 3 states that the risk-normalized budgeted cost at time *T* (i.e., the risk-normalized baseline) is the sum of the risk-normalized earned values for all WBS elements from the beginning of the project, through time *T*.

4.6 Measuring Risk-Oriented Earned Value

The previous section discussed how risk-oriented earned values are used to determine a risk-normalized baseline. This section will discuss how to define and measure earned value in a manner consistent with the notion of risk reduction as a measure of progress.

As previously discussed, measurements of earned value during the inception and elaboration phases must reflect progress in some way that is meaningful during those phases. Since much of the emphasis during these phases is on the reduction of risk (through achieving consensus on project scope, etc.), a natural measure of earned value is the degree to which residual risk is reduced through the performance of an activity. There are several possible ways to measure this risk reduction. For example, the degree to which consensus has been achieved can be a meaningful measure in some contexts. Similarly, as the project moves from one iteration to

the next, the degree of change in the project's agreed-upon scope may be a significant indicator of residual risk. The key is in understanding what it is you are trying to accomplish at a given point in the project, and then defining measures that are meaningful for your context.

To illustrate these concepts

> Suppose there is a task in WBS element 1.2.2 (vision specification) that is projected to last three weeks and cost $50,000 to complete. Assume that the risk allocated to this element is 0.3 (i.e., 30% of the total development risk), and the total development cost is $500,000. Considering the allocation of risk to the other tasks (which is beyond the scope of this report), the normalized BCWS for this task is calculated as $55,600. The task baseline is shown in Figure 7.

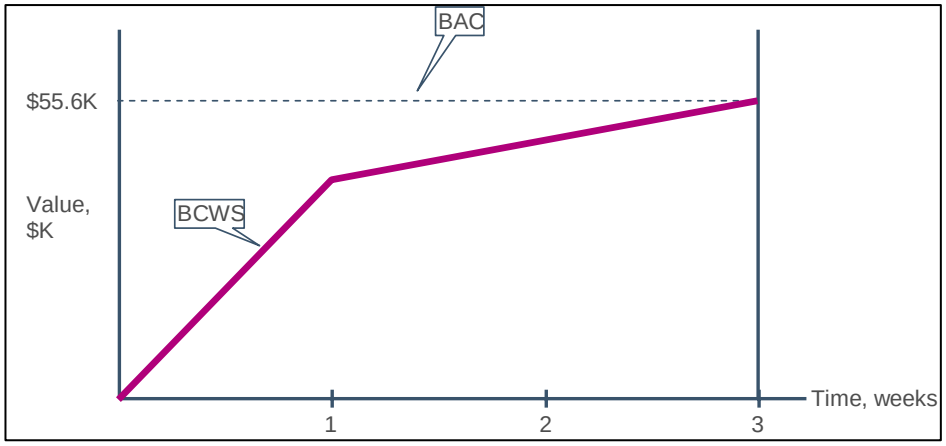

Figure 7: Sample Task Baseline

> Assume, furthermore, that *residual risk* is defined as the degree to which consensus regarding the vision is not achieved among the stakeholders. Here *vision* refers to the high-level requirements (functional and non-functional) and constraints such as cost, schedule, etc. These values are shown in Table 5.

Table 5: Sample Definition of Residual Risk

Residual Risk	***Definition***	***Estimated Time to Achieve***
100%	Task initiation, ¬∃ consensus	0
30%	∃ consensus • 1 feasible solution	1 week
0%	∃ consensus • > 1 feasible solution	3 weeks

After one week, it is determined that the stakeholders have only made 50% progress towards a consensus on one feasible solution, but that $21,000 has been spent. Thus, the earned value is $19,250, against a baseline of $38,500. This would be a clear signal that this task was seriously behind schedule after only one week. Since $21,000 was spent to accomplish $19,250 worth of work, the task is also slightly over budget (by $1,750). This is apparent in Figure 8.

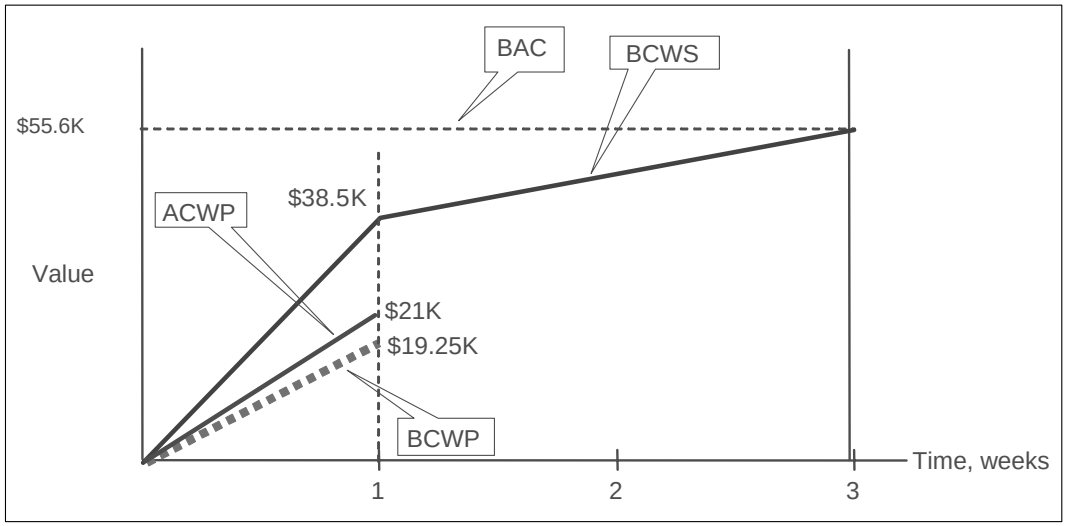

Figure 8: Sample Task Actual Versus Baseline

By way of contrast, the more conventional EVM approach (using LOE as the baseline) would still lead to the conclusion that the task is slightly over budget, but the earned value would equal the baseline, so there would not be any apparent schedule slip.

5 Summary

While EVM has proven to be an effective tool for managing software development projects, it is dependent on an objective, accurate program management baseline. The baseline, in turn, depends on a WBS that can be understood. We have shown how structuring the WBS around the system architecture, as is typically done with conventional waterfall projects, makes it much more difficult to change the plan as the system architecture changes. This renders the WBS unusable as a baseline against which to measure progress for spiral developments. The Royce WBS provides a sound foundation to address this weakness. However, the Royce WBS is often insufficient for enabling projects to identify and track progress against the important tasks and activities necessary to accomplish the goals and objectives typified in the anchor point milestones. The alternative WBS proposed in this report addresses these weaknesses. Coupled with a new interpretation of earned value, this approach can potentially lead to a more incisive answer to the program manager's age-old question: "How much progress have I made against my original plan?" At the same time, other factors, such as technical performance metrics, provide additional insight into the progress (or lack thereof) in a development effort, and will also require reinterpreting for spiral development.

Bibliography

URLs are valid as of the publication date of this document.

[Abba 97] Abba, W. "Earned Value Management: Reconciling Government and Commercial Practices." *Program Manager 26* (Jan-Feb 1997): 58-69. http://www.dau.mil/pubs/pm/pmpdf97/abba.pdf

[Alexander 98] Alexander, S. *Earned Value Management Systems (EVMS): Basic Concepts*. Presented at Project Management Institute, Washington, DC Chapter, 1998. Available from http://gaenet.com/vis1on/support/faq.htm

[Armour 00] Armour, P. "The Five Orders of Ignorance." *Communications of the ACM, 43,* 10 (October 2000): 17-20.

[Boehm 81] Boehm, B. *Software Engineering Economics*, Upper Saddle River, NJ: Prentice Hall, 1981.

[Boehm 88] Boehm, B. "A Spiral Model of Software Development and Enhancement." *Computer* (May 1988): 61-72.

[Boehm 96] Boehm, B. "Anchoring the Software Process." *IEEE Software 13,* 4 (July 1996): 73-82.

[Boehm 00a] Boehm, B. "Spiral Development and Evolutionary Acquisition: Where Are We Today?" *SEI-CSE Spiral Development and Evolutionary Acquisition*. Washington, DC, September 2000. http://www.sei.cmu.edu/cbs/spiral2000/september/Boehm/index.htm

[Boehm 00b] Boehm, B. *Spiral Development: Experiences, Principles, and Refinements* (CMU/SEI-00-SR-008 ADA382590). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/00.reports /00sr008.html

[Boehm 03] Boehm, B. "Value-Based Software Engineering." *Software Engineering Notes, 28,* 2 (March 2003) pp. 33-41.

[DoD 05a] Department of Defense. *Earned Value Management Frequently Asked Questions*. http://www.acq.osd.mil/pm/faqs/faq.htm (2005).

[DoD 05b] Department of Defense. *Glossary of Earned Value Management Terms*. http://www.acq.osd.mil/pm/faqs/glossary.htm (2005).

[DSMC 00] Defense Systems Management College. *Earned Value Management Gold Card*. http://www.acq.osd.mil/pm/faqs/evm_gold_card.pdf (2000).

[Kruchten 04] Kruchten, P. *The Rational Unified Process: An Introduction*, 3rd ed. New York, NY: Addison-Wesley Object Technology Series, March 2004.

[Royce 98] Royce, Walker. *Software Project Management: A Unified Framework*. Reading, MA: Addison-Wesley Object Technology Series, 1998.

[Royce 70] Royce, Winston. "Managing the Development of Large Scale Software Systems" 1-9. *Proceedings of IEEE WESCON 1970*. Los Angeles, CA, Aug 25-28, 1970. Los Alamitos, CA: IEEE Computer Society Press, 1970. Reprinted in *Proceedings of the Ninth International Conference on Software Engineering*. New York, NY: ACM Press, 1989, 328-338.

[USAF 00] United States Air Force Science Advisory Board. *Ensuring Successful Implementation of Commercial Items in Air Force Systems* (SAB-TR-99-03). http://www.ichnet.org/April2000AFSAB%20Report.pdf (April 2000).

[Wilkins 99] Wilkins, T. "Earned Value, Clear and Simple" April 1999. http://www.acq.osd.mil/pm/old/Old%20Papers/Papers%20-%20Govt/paperpres/wilkins_art.pdf

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	June 2005	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Using Earned Value Management (EVM) in Spiral Development	F19628-00-C-0003

6. AUTHOR(S)

Lisa Brownsword, Jim Smith

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2005-TN-016

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Earned Value Management (EVM) helps managers to plan, monitor, and control the development and evolution of custom developed software-intensive systems. EVM assumes a waterfall development model. However, to meet the demands for today's complex, dynamic systems, certain trends have emerged. First, projects no longer develop all components of a system as custom components. Instead, projects use pre-existing, off-the-shelf packages, components, or entire systems, potentially with custom components. A second trend is a realization that often requirements are not known in detail at the start of a project and must evolve efficiently in response to changing needs and technology. A further trend (often in response to the first two trends) is the move to other development models, such as spiral or iterative development processes. Spiral development processes can better support 1) the required discovery of what users want and 2) negotiation to reconcile what engineers can quickly and reasonably assemble from pre-existing and custom components.

While projects have applied EVM to spiral development projects, the results have not been uniformly satisfying. This report explores the fundamental challenges in using EVM with spiral development processes and proposes adaptations to some EVM principles to render it more suitable for today's software-intensive systems.

14. SUBJECT TERMS	15. NUMBER OF PAGES
spiral development, waterfall development model, work breakdown structure, WBS	35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL