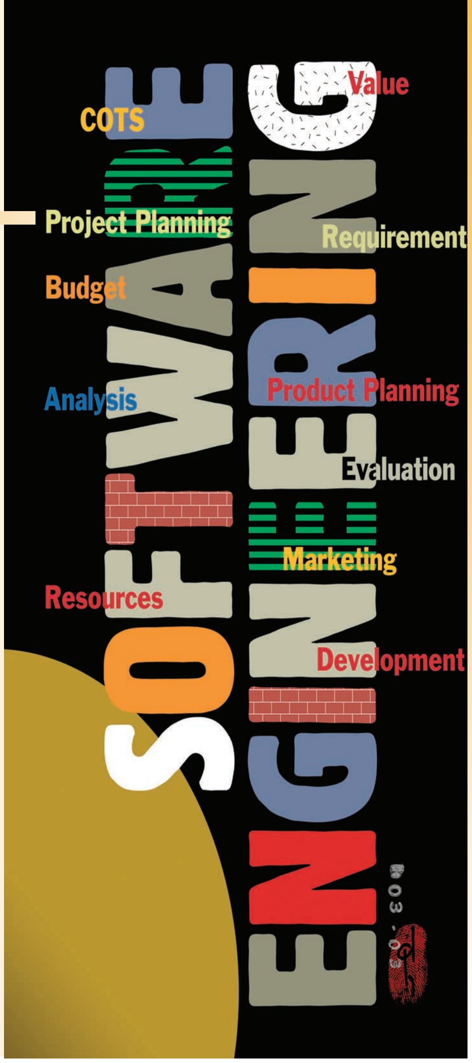
COVER FEATURE

Value-Based Software Engineering: A Case Study



The value-based approach to software development integrates value considerations into current and emerging software engineering principles and practices, while developing an overall framework in which these techniques compatibly reinforce each other.

Barry Boehm
Li Guo Huang
University of Southern California

The increasing pace of change in the information technology field makes feedback control essential for organizations to sense, evaluate, and adapt to changing value propositions in their competitive marketplace. Traditional project feedback control mechanisms, such as earned-value systems, can effectively control the development efficiency of relatively stable projects in well-established value situations. But these mechanisms are insensitive to the project's return-on-investment factors, and they can lead to wasteful misuse of an organization's scarce resources.

Neglecting to monitor and control the value that software adds was not too risky when business values changed slowly and software made minor contributions to them. However, such neglect is increasingly risky in today's world of software-driven product lines and tornado-driven changes in the information technology marketplace—and will only become more so. Many IT projects have faithfully delivered products on or near their planned budgets and schedules, only to find that the business need for the product had largely disappeared, or that competitors had already captured its planned market niche.

A particular anomaly in the monitoring and control area, the Earned Value Management System,[1] provides a useful technique for monitoring and controlling the cost, schedule, and progress of a complex project. But it has absolutely nothing to say about the stakeholder value of the system the project is developing. It serves a purpose, but needs to be incorporated into feedback control systems that focus on the real stakeholder value being earned.

The value-based software engineering agenda, described in the "Accounting for Value in Software Engineering" sidebar, seeks to integrate value considerations into current and emerging software engineering principles and practices, while developing an overall framework in which they compatibly reinforce each other. One area that VBSE addresses—value-based planning and control—includes principles and practices for extending traditional cost, schedule, and product planning and control techniques that also manage the value delivered to stakeholders.

USING EARNED VALUE

Performing feedback control of a large software project becomes difficult because hundreds of tasks progress concurrently during development. Some tasks will be ahead on budget and schedule, others behind. Current earned-value systems let managers of large projects achieve better control of such complex situations.

Most software cost and schedule estimation models include breakdowns of a project's budget and schedule by software component, development

Accounting for Value in Software Engineering

Developers perform much of current software engineering practice and research in a value-neutral setting, in which they tend to do the following:

- treat every requirement, use case, object, and defect as equally important;
- express and practice methods as largely logical activities involving mappings and transformations;
- use earned-value systems to track project cost and schedule, but not stakeholder or business value;
- practice a separation of concerns that confines software engineers' responsibility to turning software requirements into verified code; and
- set goals for improving productivity or correctness independent of stakeholder value considerations.

Given that today's software has a major influence on system costs, schedule, and value, and that software decisions intertwine inextricably with system-level decisions, we can no longer afford to follow this approach.

Further, value-neutral software engineering principles and practices cannot deal with most sources of software project failure. Major studies such as the Standish Group's CHAOS report (www.standishgroup.com) reveal that value-oriented shortfalls cause most software project failures. These shortfalls include lack of user input, incomplete or changing requirements, lack of resources, unrealistic expectations, unclear objectives, and unrealistic time frames.

Value-neutral methods also provide an insufficient basis for an engineering discipline. Most concerns expressed about the adequacy of software engineering focus on the shortfalls in its underlying science. But it is difficult for a value-neutral approach to provide guidance for satisfying the engineering definition of making its products useful to people, as this involves dealing with different people's utility functions or value propositions.

A value-based software engineering agenda

Progress has been made over the years to integrate some value-oriented perspectives into software engineering. This includes approaches such as participatory design, user engineering, cost estimation, software economics, software investment analysis, and software engineering ethics. However, these approaches generally have been treated as add-on band-aids to baseline software engineering principles and practices. The value-based software engineering agenda[1] includes the following elements:

- *Requirements engineering*. Developing principles and practices for identifying a system's success-critical stakeholders, eliciting their value propositions with respect to the system, and reconciling these value propositions into a mutually satisfactory set of objectives for the system.
- *Architecting*. Reconciling system objectives with achievable architectural solutions.
- *Design and development*. Developing techniques to ensure that the software's design and development inherit the system's objectives and value considerations.
- *Verification and validation*. Ensuring that a software solution satisfies its value objectives, and organizing V&V tasks to operate as an investment activity.
- *Planning and control*. Extending traditional cost, schedule, and product planning and control techniques to include the value delivered to stakeholders.
- *Risk management*. Developing principles and practices for identifying, analyzing, prioritizing, and mitigating risk.
- *Quality management*. Prioritizing desired quality factors with respect to stakeholders' value propositions.
- *People management*. Stakeholder teambuilding and expectations management, managing the project's accommodation of all stakeholders' value propositions throughout the life cycle, and integrating ethical considerations into daily project practice.
- *Principles and practices*. These include COTS-based systems, rapid development, agile methods, high-dependability systems, systems of systems, and ethics.

This agenda has been actively pursued by the Economics-Driven Software Engineering Research (EDSER) community (www.edser.org).

Reference

1. B. Boehm, *Value-Based Software Engineering*, to be published in *ACM Software Eng. Notes*, Mar. 2003.

phase, and task activity. A development team can use these models to set up an earned-value system in which it can consider the estimated cost for each task as the value earned for the project when the task completes.

The earned-value system works as follows.

1. The project team develops a set of tasks necessary for the project's completion, and associated budgets and schedules for each task.
2. Developers assign each task an earned value (EV) for its completion, usually its task budget.
3. As a project proceeds, the project team reviews three primary quantities at selected times *T*: the budgeted cost of work scheduled (BCWS), which equals the sum of the earned values for all tasks scheduled to be completed by time *T*; the budgeted cost of work performed (BCWP), or project-level earned value, which equals the sum of the earned values for all tasks actually completed by time *T*; and the actual cost of the project through time *T*.
4. If the BCWP equals or exceeds the BCWS, the project is on or ahead of schedule.
5. If the BCWP equals or exceeds the project cost, the project is on or ahead of budget.
6. If the BCWP is significantly less than the BCWS, the project cost at time *T*, or both, the project

is significantly overrunning its schedule, budget, or both, and corrective action must be performed.

Figure 1 summarizes the six earned-value feedback process steps.

Earned-value system example

Figure 2 provides an example of how the earned-value system can help assess the likely cost to complete a software project and how to track its progress. For simplicity, we assume that the project starts with four sequential tasks: prototypes, analyses, plans, and specs.

First, we assign an earned value to each task. We estimate that the completion of prototypes will take two months and \$15,000, the analyses one month and \$10,000, the plans one month and another \$10,000, and the specs one month and \$15,000. This yields a cumulative earned value of \$50,000 after successfully finishing these four tasks.

Table 1 represents the findings from a review performed at the end of the fourth month to assess the project's status and actual cost. At this point, the project was scheduled to complete the first three tasks—prototypes, analyses, and plans—so the cumulative BCWS ($EV_{scheduled}$) equals \$35,000. However, only the prototypes and analyses are finished, which yields a BCWP ($EV_{performed}$) of only \$25,000.

In this case, since the BCWP of \$25,000 is less than the BCWS of \$35,000, the project is behind schedule. On the other hand, the actual cost is \$14,000 to finish the prototypes and \$6,000 to complete the analyses. Thus, the cumulative actual cost of work performed is \$20,000, which indicates that the actual cost is below the budget for the first two tasks.

In terms of the earned-value feedback process in Figure 1, the BCWP is greater than cost, so no corrective budgeting action is required. However, given that the BCWP is less than the BCWS, the project is behind schedule and needs corrective scheduling action. This fix might involve slipping the schedule or, if that is infeasible, rescoping the project to fit within the available schedule. Techniques such as the *schedule as independent variable* process[2] can accommodate such corrective action.

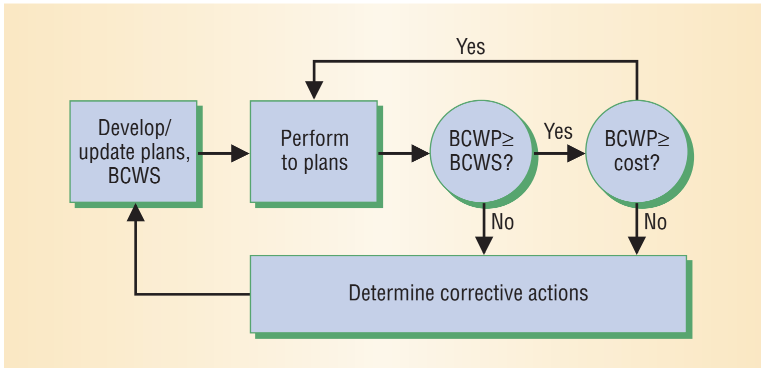


Figure 1. Earned-value feedback process. The process first determines if the budgeted cost of work performed (BCWP) is greater than or equal to the budgeted cost of work scheduled (BCWS). It next determines if BCWP is greater than or equal to cost. If both hold true, development proceeds; if either proves false, the project team determines corrective actions.

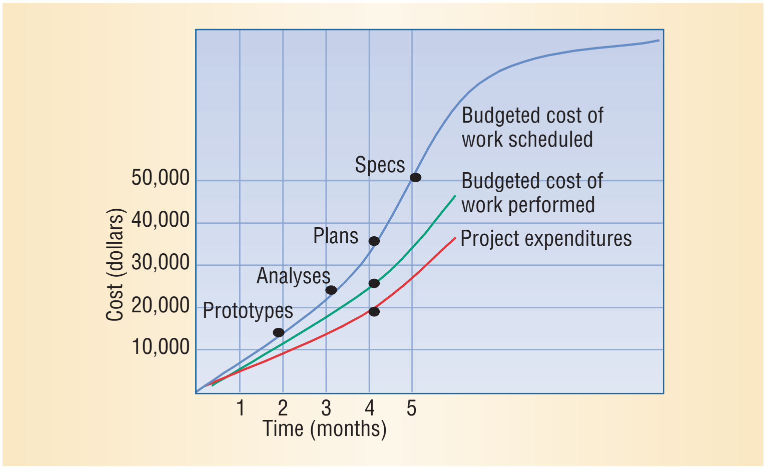


Figure 2. Earned-value system. This simple example starts with four sequential tasks: prototypes, analyses, plans, and specs. The project team assigns earned values to the tasks by estimating a completion time and cost for each one.

REAL EARNED-VALUE FEEDBACK CONTROL

The earned-value management process generally performs well when tracking how closely the project is meeting its original plan. However, it be-

Table 1. Review of earned value and actual cost at the end of the fourth month.

Project tasks	Budgeted cost of work scheduled (earned value$_{scheduled}$), in dollars	Budgeted cost of work performed (earned value$_{performed}$), in dollars	Actual cost, in dollars
Prototypes	\$15,000	\$15,000	\$14,000
Analyses	10,000	10,000	6,000
Plans	10,000	0	0
Specs	0	0	0
Total	BCWS = 35,000	BCWP = 25,000	Cost = 20,000

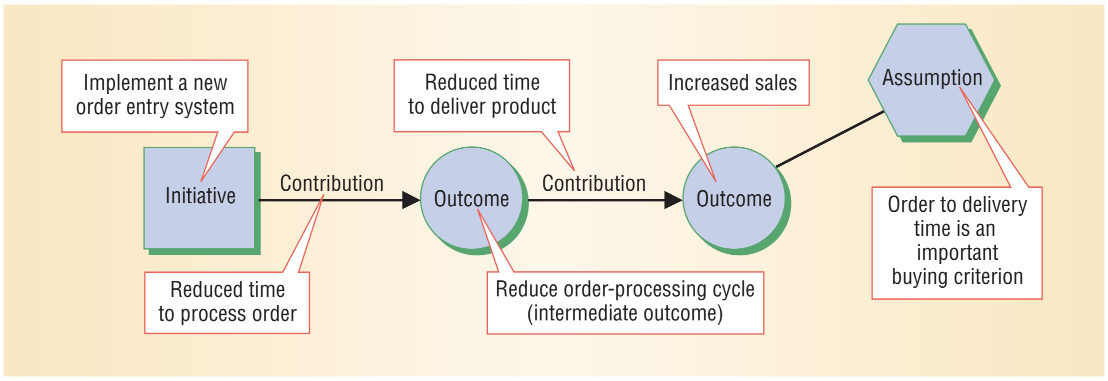

Implement a new order entry system
Initiative
Contribution
Reduced time to process order
Reduced time to deliver product
Outcome
Reduce order-processing cycle (intermediate outcome)
Contribution
Increased sales
Outcome
Assumption
Order to delivery time is an important buying criterion

Figure 3. Benefits-realization approach results chain. The chain links to assumptions, which condition the realization of outcomes.

comes difficult to administer if the project's plan changes rapidly. More significantly, it neglects the actual value the project is earning for the organization. A project can be tremendously successful with respect to cost-oriented earned value, but an absolute disaster in terms of actual organizational value earned.

This frequently happens when the resulting product has flaws with respect to user acceptability, operational cost-effectiveness, or timely market entry. Thus, it is preferable to have techniques that support monitoring and control of the actual value the project earns and the resulting return on investment: ROI = (benefits – costs)/costs, adjusted for inflation effects.

Earned-value monitoring and control

To start, we can use the project's business case to monitor the actual business value of its promised capabilities. This involves a continuing update of the business case to reflect changes in business model assumptions, market conditions, organizational priorities, and progress with respect to enabling initiatives.

In *Making the Software Business Case,*[3] Donald J. Reifer provides information and guidelines for software business case analysis. Monitoring the value of undelivered capabilities is difficult, however. Thus, this approach works best on a project organized to produce relatively frequent increments of delivered capability.

To formulate the business case, we must determine and reconcile the value propositions of the project's success-critical stakeholders. This makes it necessary to identify these stakeholders and their roles in realizing the project's benefits.

An excellent technique for doing this is the DMR Consulting Group's *benefits realization approach*.[4] Figure 3 shows an example of its centerpiece, the results chain. This chain establishes a framework linking *initiatives* that consume resources, such as implementing a new sales-order-entry system, to *contributions*—not delivered systems, but their effects on existing operations—and *outcomes*, which can lead either to further contributions or to added value, such as increased sales.

The results chain links to assumptions, which condition the realization of outcomes. Thus, in Figure 3, if order-to-delivery time turns out to be an unimportant buying criterion for the product being sold, the reduced time to deliver the product will not result in increased sales.

Software project members can use the results chain as a framework for working with their clients to identify additional nonsoftware initiatives that may be needed to realize the potential benefits of the software or IT system initiative. Using this framework can also help identify additional success-critical stakeholders who need to be represented so that they will buy into the shared vision.

For example, the initiative to implement a new order-entry system to reduce the time required to process must convince the salespeople that using the new system features will be good for their careers. For example, if the order-entry system is so efficiency-optimized it doesn't track sales credits, the salespeople will fight using it. The salespeople also must be trained to use the system effectively.

Further, the reduced order-processing cycle will decrease the product delivery time only if additional initiatives are pursued to coordinate the order-entry system with the order-fulfillment system. Some classic cases where this didn't happen include the late delivery of Hershey's Halloween candy and delayed Christmas-toy shipments for Toys R Us.

Such additional initiatives must be added to the results chain. Besides increasing its realism, this also identifies additional success-critical stakeholders—salespeople and order-fulfillment people—who must be involved in the system definition and development process. The expanded results chain involves these stakeholders not just in a stovepipe software project to satisfy some requirements, but in a program of related software and nonsoftware initiatives focused on value-producing end results.

Once the stakeholders agree on the initiatives, they can elaborate them into project plans, requirements, architectures, budgets, and schedules. The resulting

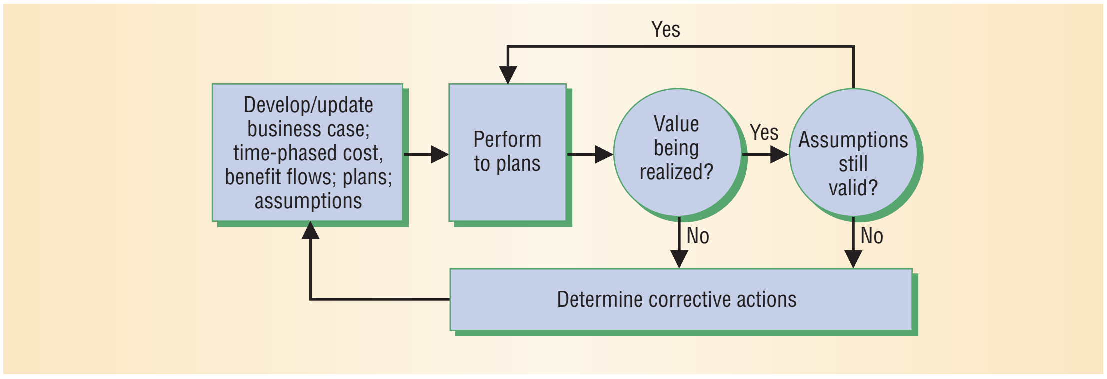

Yes
Develop/update business case; time-phased cost, benefit flows; plans; assumptions
Perform to plans
Value being realized?
Yes
Assumptions still valid?
No
No
Determine corrective actions

Figure 4. Value-realization feedback process. As the project performs to plan, the actual or projected achievement of the cost and benefit flows and the assumptions' realism may become invalid, at which point the project team will need to determine and apply corrective actions.

budgets establish the cost inputs to the business case. A benefits analysis then determines the corresponding benefits for the business case. The quantitative part of this analysis includes the effects on the organization's profit-and-loss bottom line, either in terms of additional profit streams or cost savings.

The benefits analysis should also include a qualitative evaluation of benefits such as customer satisfaction, corporate reputation, and supply-chain controllability. For a public service organization, the benefits analysis should determine quantitative proxies for benefits wherever possible, such as improved health-and-safety records, reduced service delays and complaint rates, or high service-recipient satisfaction ratings. Such evaluations are often worth tracking for commercial organizations, as they provide key competitive discriminators.[5]

Figure 4 shows the resulting value-realization feedback process. The results chain, business case, and program plans set the baseline in terms of expected time-phased costs, benefit flows, returns on investment, and underlying assumptions. As the projects and program perform to plans, the actual or projected achievement of the cost and benefit flows and the assumptions' realism may become invalid, at which point the project team will need to determine and apply corrective actions by changing plans or initiatives, making associated changes in expected cost and benefit flows.

ORDER-PROCESSING EXAMPLE

Sierra Mountainbikes—a fictitious company representative of two companies with less successful projects—has an outstanding reputation for high quality in their specialty area, mountain bicycle manufacturing. However, an operations review confirmed its retailers' extreme dissatisfaction with Sierra's order-processing systems. Retailers were encountering significant problems with delivery delays and mistakes; poor synchronization between order entry, confirmation, and fulfillment; and disorganized responses to problem situations. The resulting crisis-management mode of operation added to an already costly and inefficient order-processing system.

To address these problems, Sierra entered into a strategic partnership with eServices Inc. for joint development of a new order-processing and fulfillment system. The partnership will integrate the new system with an upgrade of Sierra's financial, production, and human-resource-management information systems. The strategic partnership is organized around both the system's results chain and business case, so that both parties share in the responsibilities and rewards of realizing the system's benefits. Thus, both parties share a motivation to understand and accommodate each other's value propositions or win conditions and to use value-based feedback control to manage the program of initiatives.

Benefits-realization results chain

Figure 5 shows the program's overall results chain. Compared to Figure 3, it adds new initiatives for order-entry processes, outreach, and training, and for order-fulfillment functions. Besides involving its in-house stakeholders in those initiatives, Sierra has obtained commitments from its three leading distributors to participate in defining and beta testing the new system, and in evaluating such expected outcomes as increased customer satisfaction and ease of use. Planning how to monitor these aspects of benefits realized led Sierra to recognize the distributors as additional success-critical stakeholders to be involved in the system's definition and development.

Figure 5 is at about the right detail level for a system of this complexity, being sufficiently detailed to record the overall program vision and structure, and to identify the success-critical stakeholders. It also provides a sufficient framework for developing the new system's business case, and for feedback control of whether the initiatives are realizing the expected benefits. As most of its critical assumptions are program-wide, these are recorded in Figure 5 as a general list of assumptions to be monitored, rather than additional hexagons in the figure.

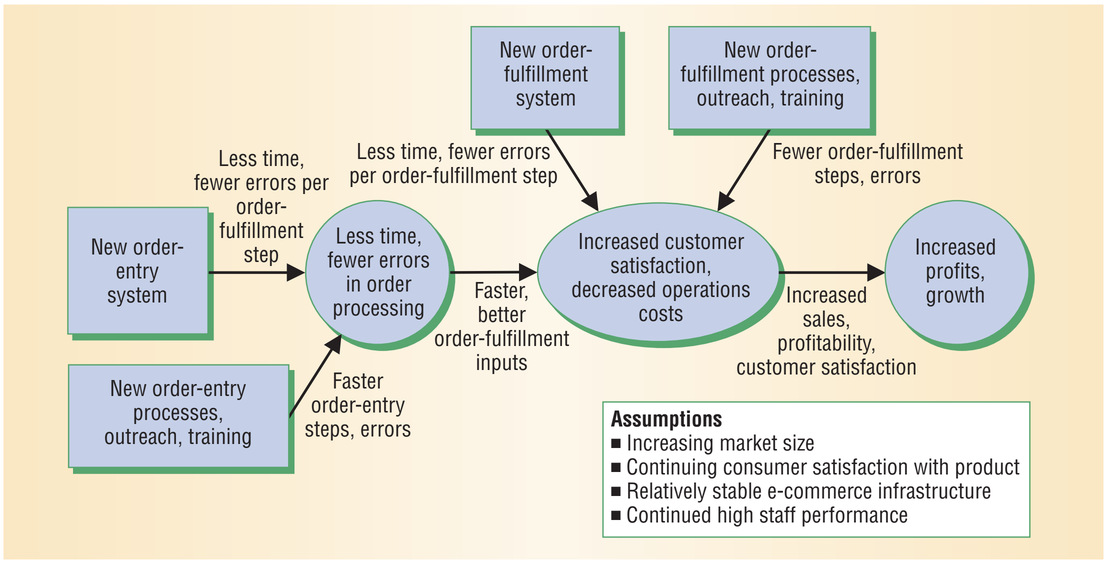

New order-fulfillment system
New order-fulfillment processes, outreach, training
Less time, fewer errors per order-fulfillment step
Less time, fewer errors per order-fulfillment step
Fewer order-fulfillment steps, errors
New order-entry system
Less time, fewer errors in order processing
Increased customer satisfaction, decreased operations costs
Increased profits, growth
Faster, better order-fulfillment inputs
Increased sales, profitability, customer satisfaction
New order-entry processes, outreach, training
Faster order-entry steps, errors
Assumptions
■ Increasing market size
■ Continuing consumer satisfaction with product
■ Relatively stable e-commerce infrastructure
■ Continued high staff performance

Figure 5. Expanded order-processing system-results chain. Compared to the version shown in Figure 3, this version adds new initiatives for order-entry processes, outreach, and training, as well as order-fulfillment functions.

Table 2. Order-processing system schedules and budgets.

Milestone	Due date	Budget ($K)	Cumulative budget ($K)
Inception readiness	1 Jan. 2004	0	0
Life cycle objectives	31 Jan. 2004	120	120
Life cycle architecture	31 Mar. 2004	280	400
Core capability drivethrough	31 July 2004	650	1,050
Initial operational capability: software	30 Sept. 2004	350	1,400
Initial operational capability: hardware	30 Sept. 2004	2,100	3,500
Developed initial operational capability	31 Dec. 2004	500	4,000
Responsive initial operational capability	31 Mar. 2005	500	4,500
Full operational capability, core capability drivethrough	31 July 2005	700	5,200
Full operational capability, beta	30 Sept. 2005	400	5,600
Full operational capability, deployed	31 Dec. 2005	400	6,000
Annual operations and maintenance		3,800	
Annual operations and maintenance, old system		7,600	

Costs, benefits, and ROI

Table 2 shows the overall development budgets and schedules for the order-processing system. The project uses an evolutionary spiral approach[6] involving overall strategic plans for an initial operational capability (IOC) and full operational capability (FOC). Prioritized feature sets initialize the content of the IOC and FOC, with an architecture supporting growth to FOC and ease of dropping or adding features as their nature or priorities change.

Additional key milestones include the life cycle objectives (LCO) and life cycle architecture (LCA) milestones used in the USC model-based system architecting and software engineering (Mbase) approach, the Rational unified process (RUP),[7] and the core capability drivethrough (CCD) milestone in the schedule as independent variable specialization of Mbase and RUP.[2]

The Cocomo II cost estimation model[8] also uses these milestones to estimate the budget, effort, schedule, and staffing level required to meet them. Table 2 shows the milestone due dates, budgets, and cumulative budgets derived for the order-fulfillment project using the Cocomo II estimates, adjusted to cover the related business processes, outreach, and training initiatives listed under software, and the new purchased computer hardware and COTS infrastructure software, listed under hardware in the 30 Sept. 2004 IOC milestone. The project team uses these values to monitor and control project progress with traditional earned-value techniques, and to calculate the costs incurred to determine the program's return on investment.

The IOC will start development on 1 Jan. 2004. It will be installed for beta testing with the three key distributors on 30 Sept. 2004, and cut over as a replacement for most of the old system on 31 Dec. 2004, at a cumulative investment cost of $4 million.

Table 3. Order-processing system: Current versus new system.

Projections		Current system			New system		
Date	Market size ($M)	Market share %	Sales	Profits	Market share %	Sales	Profits
31 Dec. 2003	360	20	72	7	20	72	7
31 Dec. 2004	400	20	80	8	20	80	8
31 Dec. 2005	440	20	88	9	22	97	10
31 Dec. 2006	480	20	96	10	25	120	13
31 Dec. 2007	520	20	104	11	28	146	16
31 Dec. 2008	560	20	112	12	30	168	19

Table 4. New system: Expected benefits and business case.

Time	Expected improvements						Overall customer satisfaction		
Target date	Cost savings	Change in profits	Cumulative change in profits	Cumulative cost	Return on investment	Late delivery (percent)	Customer satisfaction	In-transit visibility	Ease of use
31 Dec. 2003	0	0	0	0	0	12.4	1.7	1.0	1.8
31 Dec. 2004	0	0	0	4.0	–1	11.4	3.0	2.5	3.0
31 Dec. 2005	2.2	3.2	3.2	6.0	–.47	7.0	4.0	3.5	4.0
31 Dec. 2006	3.2	6.2	9.4	6.5	.45	4.0	4.3	4.0	4.3
31 Dec. 2007	4.0	9.0	18.4	7.0	1.63	3.0	4.5	4.3	4.5
31 Dec. 2008	4.4	11.4	29.8	7.5	2.97	2.5	4.6	4.6	4.6

An incremental release of the IOC responding to the most cost-effective fixes and enhancements will occur on 31 Mar. 2005. Concurrently, work will start on the FOC enhancements, which the three key distributors will also beta test, then cut over as a full replacement for the old system on 31 Dec. 2005, at a cumulative cost of $6 million. Thereafter, six-month increments and annual new releases will be installed at an annual investment level of $500,000. Table 3 shows the corresponding expected benefits for the current and new systems.

As Table 3 shows, estimates indicate that Sierra's current market share and profit margins will stay roughly constant over the 2004 through 2008 period, with annual profits growing from $7 million to $12 million, if the new program is not executed. This is a generous estimate, as the problems with the current system would increase with added sales volume, leading to decreased market share and profitability. With the new system, profits due to increased sales grow to $19 million by 2008, a gain of $7 million.

Table 4 shows the expected benefits to be reaped with the new system, annually, for 2004 through 2008, with return on investment calculated as ROI = (benefits – costs)/costs. For simplicity, the table shows the costs and benefits in 2004 dollars to avoid discounted cash-flow calculations, and does not compound the 10 percent annual growth rate in estimated market size, both for simplicity and conservatism.

In Table 4, the columns from "Cost savings" through "Return on investment" show the expected improvements in market share and profit margins—both from increased sales and decreased operational costs—achievable with the new system and the resulting ROI relative to continuing with the current system. These numbers show that the expected increase in market share—from 20 percent to 30 percent by 2008—and profit margins produce a 45 percent ROI by the end of the second year of new-system operation in 2006: ROI = (benefits – costs)/costs = (9.4 – 6.5)/6.5 = 0.45. The expected ROI by the end of 2008 is thus 297 percent.

Table 4's final four columns show expected improvement in overall customer satisfaction from 2004 through 2008, with the last three categories rated on a scale from 0 to 5. It also shows three of this measure's critical components: percentage of late deliveries, ease of use, and in-transit visibility. The project team identified the latter capability as both important to distributors—if they know what is happening with a delayed shipment, they can improvise workarounds—and one that some of Sierra's competitors already provide. The team expected Sierra's 2004 through 2008 improvements with the new system to improve the company's 0 to 5 satisfaction rate on in-transit visibility from a low 1.0 to a high 4.6 and to increase its overall customer satisfaction rate for order processing from 1.7 to 4.6.

Value-based monitoring and control

The expected benefits and business case in Tables 3 and 4, although valuable as a means of justifying the program of initiatives, also provide a means

Milestone	Schedule	Cost ($K)	Op-Cost Savings %	Market Share ($M)	Annual Sales ($M)	Annual Profits	CumΔ	ROI Profits %	Late Deliv.	Cust. Sat.	ITV	Ease of Use	Risks/Opportunities
Life Cycle Architecture	3/31/04 3/31/04	*400* 427		*20* 20	*72* 72	*7.0* 7.0			*12.4* 12.4	*1.7* 1.7	*1.0* 1.0	*1.8* 1.8	Increased COTS ITV risk. Fallback identified.
Core Capa-bility Demo (CCD)	7/31/04 7/20/04	*1050* 1096								2.4*	1.0*	2.7*	Using COTS ITV fallback. New HW competitor; renegotiating HW
Software Init. Op. Capability (IOC)	9/30/04 9/30/04	*1400* 153								2.7*	1.4*	2.8*	
Hardware IOC	9/30/04 10/11/04	*3500* 3432											$200K savings from renegotiated HW
Deployed IOC	12/31/04 12/20/04	*4000* 4041		*20* 22	*80* 88	*8.0* 8.6	*0.0* 0.6	*−1.0* −.85	*11.4* 10.8	*3.0* 2.8	*2.5* 1.6	*3.0* 3.2	New COTS ITV source identified, being prototyped
Responsive IOC	3/31/05 3/30/05	*4500* 4604	*300* 324						*9.0* 7.4	*3.5* 3.3	*3.0* 1.6	*3.5* 3.8	
Full Op. Cap'y CCD	7/31/05 7/28/05	*5200* 5328	*1000* 946							3.5*	2.5*	3.8*	New COTS ITV source initially integrated
Full Op. Cap'y Beta	9/30/05 9/30/05	*5600* 5689	*1700* 1851							3.8*	3.1*	4.1*	
Full Op. Cap'y Deployed Release 2.1	12/31/05 12/20/05 6/30/06	*6000* 5977 6250	*2200* 2483	22 24	106 115	*12.2* 13.5	*3.2* 5.1	*−.47* −.15	*7.0* 4.8	*4.0* 4.1	*3.5* 3.3	*4.0* 4.2	

Figure 6. Value-based expected-versus-actual-outcome tracking-capability matrix. Among other variables, the matrix tracks return on investment (ROI) and in-transit visibility (ITV) for various commercial off-the-shelf (COTS) products. Numbers with an asterisk indicate ratings based on early trial use.

of tracking actual progress in realizing the benefits and applying corrective action wherever

- the expected benefits are not being realized,
- the assumptions in the results chain shown in Figure 5 are becoming invalid, or
- new opportunities may surface with a higher payoff than the program being executed.

Figure 6 shows a sample value-based monitoring capability for the order-processing system: a simple, straightforward approach easily implemented as a spreadsheet program. Some complications, such as present-value discounting of future cash flows, are not included here, but they can easily be added in a spreadsheet program.

Most of the cells shown here compress two cells of information into a pair of numbers. For each pair, the number in italic shows the expected value of the given elements at the given time, as expressed in the expected benefits and business case in Tables 3 and 4. The number in roman type shows the actual cost or benefit realized at the given time. Cells that have only a single number with an asterisk show interim ratings provided by the early-user distributors, based on trial use. They provide additional feedback on whether the project is on track in meeting its qualitative goals.

For example, let's look at the expected versus actual outcomes at the end of the program's first year. In this hypothetical example, the project team expected the deployed IOC to be finished by 31 Dec. 2004, but actually finished it early, on 20 Dec. 2004. The cost overrun would be about one percent: $4,041,000 actual versus $4,000,000 expected. A $200,000 savings in hardware and COTS software acquisition would keep the overrun from being serious.

On the other hand, customers' knowledge that Sierra had undertaken development of an improved order-processing system helped the company land some new purchase contracts that actually raised its market share to 22 percent in the first year. Thus, Sierra could record a $600,000 profit increase by the end of 2004, instead of the zero profit increase expected. The company also could repay $5.1 million of the $6 million investment by the end of 2005. The new system decreased the late-delivery rate to 10.8 percent versus the expected 11.4 percent, and increased the system's ease-of-use rating to 3.2, versus the expected 3.0 by the end of 2004.

However, the new system's in-transit visibility rating increased to only 1.6 instead of the expected 2.5, pulling down the overall customer satisfaction rating to 2.8 versus the 3.0 expected. We can follow the feedback control trail of this shortfall by looking at the Risks/Opportunities column. Using the Life Cycle Architecture milestone, which requires a risk management plan to either resolve or cover all major risks, the project identified a risk that TrackCorp, the primary COTS vendor offering an in-transit visibility package, would back off from its marketing promise to convert its ITV package to run on Sierra's selected Unix platform in early 2004. The project's fallback plan was to develop an interim in-house ITV capability and search for alternative Unix-based COTS ITV sources.

We can project other numbers by extrapolating on this hypothetical example. By the 20 July 2004 Core Capability Demo milestone, the project team estimates that TrackCorp could not deliver a Unix ITV package in 2004, if at all, and decides to use a fallback strategy. The in-house ITV capability would slowly increase the distributor's ITV rating, but the project team would achieve significant progress on improving the distributor's ITV rating only when the search for alternative Unix COTS ITV packages turns up a viable new source at the end of 2004. By the end of 2005, that rating would increase to 3.3 versus the expected 3.5, and the overall customer satisfaction rating would increase to 4.1, versus the expected 4.0.

Thus, we can see that the Mbase/RUP life cycle milestones, the DMR benefits realization approach and results chains, the business case analysis, and the value-based outcome tracking capability shown in Table 5, provide value-based monitoring and control information that organizations can use to proactively anticipate shortfalls and initiate corrective action to recover from them. Organizations also can use these tools to proactively pursue opportunities to improve on a planned outcome, as when the emergence of a new hardware competitor provides the opportunity to save on hardware costs and compensate for a software cost overrun.

Several challenges lie ahead for value-based monitoring and control. These include integrating advanced financial instruments such as real options into the planning and control of software-intensive systems.[9] To support both agility and discipline in controlling software projects, developers must learn to integrate the explicit information in plan-driven feedback control methods with the tacit information used in agile methods.[10, 11]

At the project and organization levels, integrating feedback control of software-intensive projects, programs, portfolios, and organizations through techniques such as the Experience Factory,[12] Balanced Scorecards,[13] the Benefits Realization Approach,[4] and the CeBASE Method[14] will become vital. Likewise, integrating value-based methods into the full range of disciplines involved in software engineering—requirements, design, development, test, COTS integration, planning, and control—will be crucial. ■

Acknowledgements

This article—based on research supported by the National Science Foundation, the Department of Defense Software Intensive Systems Directorate, and the USC Center for Software Engineering's affiliates—draws heavily on the work of the EDSER community and its leaders, particularly Hakan Erdogmus, Warren Harrison, Donald Reifer, Mary Shaw, and Kevin Sullivan.

References

1. US Air Force Systems Command, "Cost-Schedule Management of Non-Major Contracts," *AFSCP 173-3*, 1978.
2. B. Boehm et al., "Using the Spiral Model and MBASE to Generate New Acquisition Process Models: SAIV, CAIV, and SCQAIV," *CrossTalk*, Jan. 2002, pp. 20-25.
3. D. Reifer, *Making the Software Business Case*, Addison-Wesley, 2002.
4. J. Thorp, *The Information Paradox*, McGraw Hill, 1998.
5. W.C. Kim and R. Mauborgne, "Charting Your Company's Future," *Harvard Business Review*, June 2002, pp. 77-83.
6. B. Boehm and W. Hansen, "Understanding the Spiral Model as a Tool for Evolutionary Acquisition," *CrossTalk*, May 2001, pp. 4-11.
7. P. Kruchten, *The Rational Unified Process*, 2nd ed., Addison-Wesley, 2001.
8. B. Boehm et al., *Software Cost Estimation with Cocomo II*, Prentice Hall, 2000.
9. K. Sullivan et al., "The Structure and Value of Modularity in Software Design," *Proc. European Software Eng. Conf.* and *ACM SIGSOFT Symp. Foundations Software Eng.*, ACM Press, 2001, pp. 99-108.
10. J. Highsmith, *Agile Software Development Ecosystems*, Addison-Wesley, 2002.
11. B. Boehm and R. Turner, *Balancing Agile and Plan-Driven Methods: A Guide for the Perplexed*, Addison-Wesley, 2003.
12. V. Basili, G. Caldeira, and H.D. Rombach, "The Experience Factory," in *Encyclopedia of Software Engineering*, J. Marciniak, ed., Wiley, 1994.
13. R. Kaplan and D. Norton, *The Balanced Scorecard: Translating Strategy into Action*, Harvard Business School Press, 1996.
14. B. Boehm et al., "Achieving CMMI Level 5 Improvements with MBASE and the CeBASE Method," *CrossTalk*, May 2002, pp. 9-16.

***Barry Boehm** is a professor in the Computer Science Department at the University of Southern California. Contact him at Boehm@sunset.usc.edu.*

***Li Guo Huang** is a PhD candidate in the Computer Science Department at the University of Southern California. Contact him at liguohua@sunset.usc.edu.*