Lecture Slides for Managing and Leading Software Projects

Chapter 8: Measuring and Controlling Work Processes

developed by

Richard E. (Dick) Fairley, Ph.D.

to accompany the text

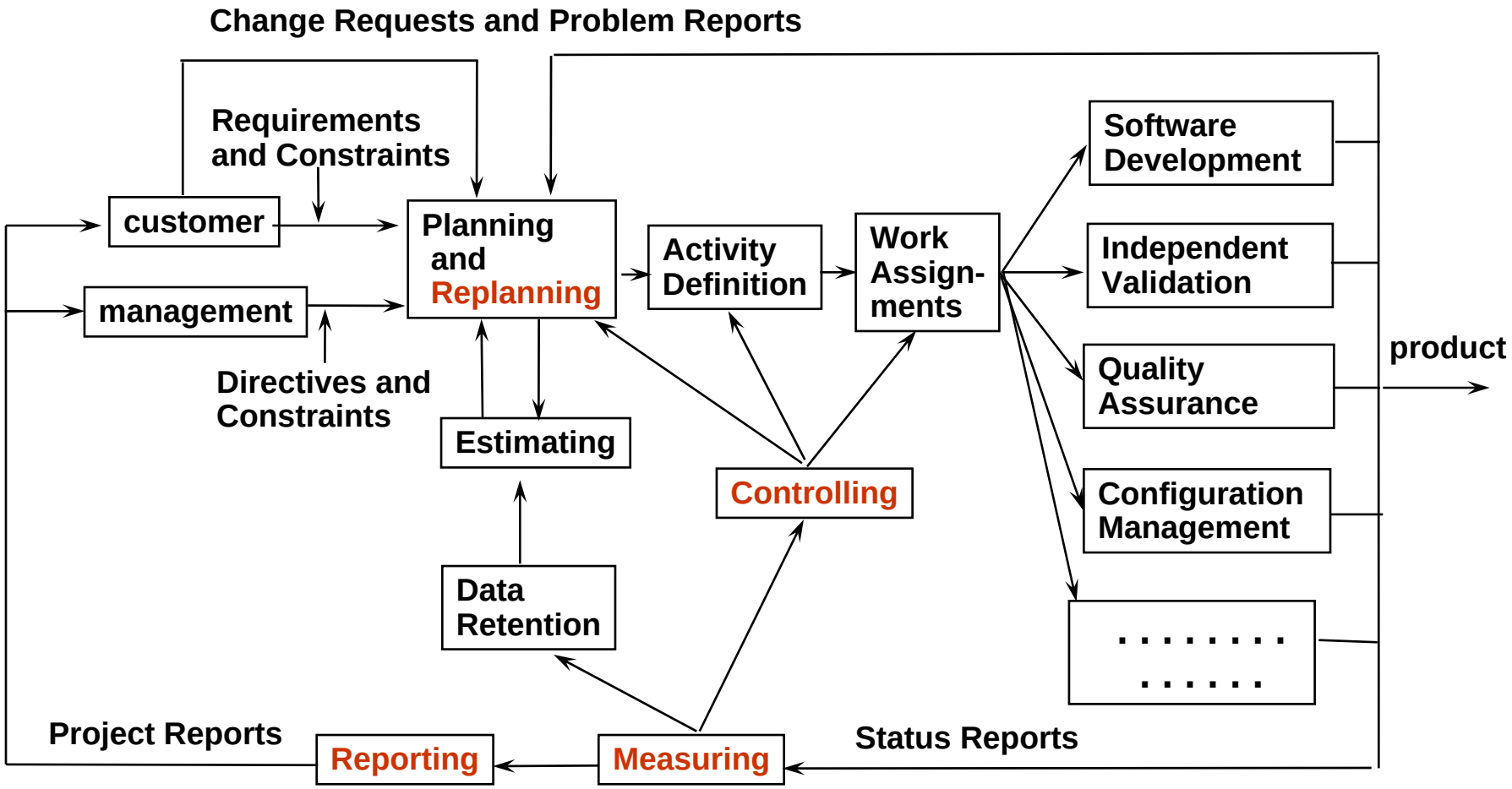
Managing and Leading Software Projects

published by Wiley, 2009

FOUR TYPES OF PROJECT MANAGEMENT ACTIVITIES

- Plan and Estimate
  - tasks, schedule, budget, resources
- Measure and Control
  - schedule, budget, resources, progress
  - work products (quantity & quality)
- Communicate and Coordinate
  - help people do their work activities
  - represent the project to others
- Manage Risk
  - Identify and confront risk factors

A Workflow Model for Measuring and Controlling Software Projects



Chapter 8 Topics

- Measuring and Analyzing Effort
- Measuring and Analyzing Rework Effort
- Tracking Effort, Schedule, and Cost
- Binary Tracking
- Estimating Future Status
- Earned Value Reporting
- Project Control Panel®

Additional Sources of Information (1)

- An introduction to measures, measurement, and control is presented in Chapter 7, Sections 7.1 and 7.2.
  - Those sections should be read as background material for this chapter.

Additional Sources of Information (1)

- CMMI-DEV-v1.2, ISO/IEEE Standard 12207, IEEE Standard 1058, and the PMBOK Guide of PMI all provide guidance for measurement and control of software projects. See Appendix 7A in Chapter 7 of the textbook.
- Terms used in this chapter and throughout this text are defined in Appendix A to the text. Presentation slides for this chapter and other supporting material are available at the URL listed in the Preface.

Objectives for Chapter 8

- After reading this chapter and completing the exercises you should understand how to:
  - measure and analyze original effort, evolutionary rework, and avoidable rework
  - use work packages to track effort, schedule, and work products
  - use binary tracking to avoid the 90% complete syndrome, and to thus accurately determine the status of effort, schedule, and work products, and to estimate effort and schedule to complete a project
  - use earned value reporting, based on binary tracking, to provide succinct and accurate reports of effort, schedule, and work progress
  - use earned value techniques to forecast estimated actual cost and estimated completion date for software projects

Why Measure Process Attributes?

- There are several reasons to measure various attributes of your work processes:
  - to provide frequent indicators of progress,
  - to provide early warning of problems,
  - to permit analysis of trends for your project,
  - to allow estimates of the final cost and completion date of your project, and
  - to build a data repository of project histories for your organization.

What Should be Measured and Controlled?

- It is difficult to imagine a software project for which some level of measurement and control over each of the following attributes is not important to assure a successful outcome:
- Chapter 7:
  - product features: requirements implemented and demonstrated to work
  - quality attributes of the product: defects, reliability, availability, response time, throughput, and others as specified
- Chapter 8:
  - effort: amount of work expended for various work activities
  - schedule: achievement of objectively measured milestones
  - cost: expenditures for various kinds of resources, including effort
  - progress: work products completed, accepted, and baselined
- Chapter 9:
  - risk: status of risk factors and mitigation activities

Measuring and Controlling

- Measuring is concerned with
  1) collecting,
  2) validating, and
  3) analyzing project status information
- Controlling is concerned with applying corrective action when actual status does not conform to planned status
  - status of the work products
    - quantity and quality of work products
  - status of the development process
    - schedule, budget, resources, progress
  - status of risk factors and mitigation strategies

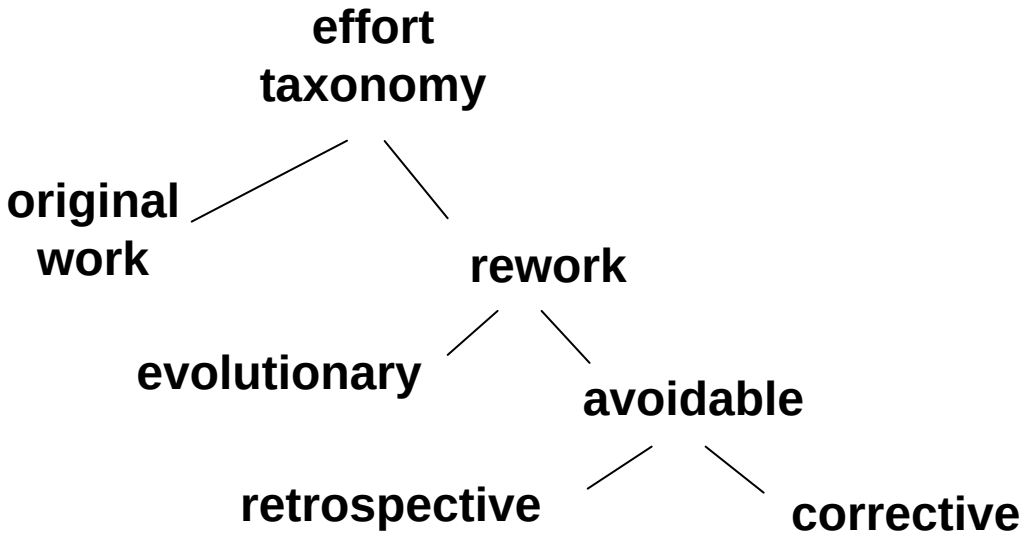
Corrective Action

- Options for corrective action include:
  - extending the schedule,
  - adding more resources,
  - using superior resources,
  - improving various elements of the development process,
  - and/or de-scoping the product requirements.
- Resources that might be improved, added, or replaced include:
  - people (being mindful of Brooks' Law when adding people),
  - software components (e.g., re-engineering a software component to improve performance),
  - hardware components (e.g., more memory, a faster processor), and
  - software tools (e.g., a language processor or testing tool).

Bad Options

- You should never use the following techniques to “get a project back on track:”
  - excessive levels and durations of overtime
  - reduction or elimination of planned verification and validation activities
  - reduction or elimination of planned user documentation, training aids, and so forth
  - reduction or elimination of any planned activity that would reduce the features or quality attributes of the system to be delivered without the customer’s consent

A Taxonomy of Project Effort



Tracking of Original Work

- Methods used to plan projects can be used to track effort, cost, and schedule for work completed:
  - WBS
  - work packages
  - critical path networks
  - milestone charts
  - activity Gantt charts
  - resource Gantt charts

see Chapter 5

Work Package Tracking

Work packages can be used to track:

- Planned vs actual schedule
- Planned vs actual personnel by skill level, number, and time period
- Other resources by type, number, and time period
- Planned vs actual cost
- Planned vs actual work products
- Risks and problems

at the task level and by roll-up

A Work Package Tracking Example

Activity : 3.2.2.1 DESIGN_COMM_SUBSYSTEM

Activity description:	Specify internal architecture of the COMM subsystem
Duration:	planned: 5 weeks;
	actual:_____
Personnel:	planned: 2 senior telecom designers;
	actual:_____
Skills:	planned: Designers must know the X25 protocol
	actual: _____
Tools:	planned: One Sun workstation running Statemate
	actual:_____
Travel:	planned: 3 day Design Review in San Diego for 2 people
	actual:_____
Predecessor tasks:	planned: 3.2.1 - Develop system architecture
	actual:_____
Successor tasks:	planned: 3.3.2.2 - Implement COMM
	actual:_____
Work Products:	planned: Architectural specification for COMM subsystem
	Test plan for COMM
	actual:_____
ETC	

Iterative Rework*

Table 1. An iterative rework taxonomy.

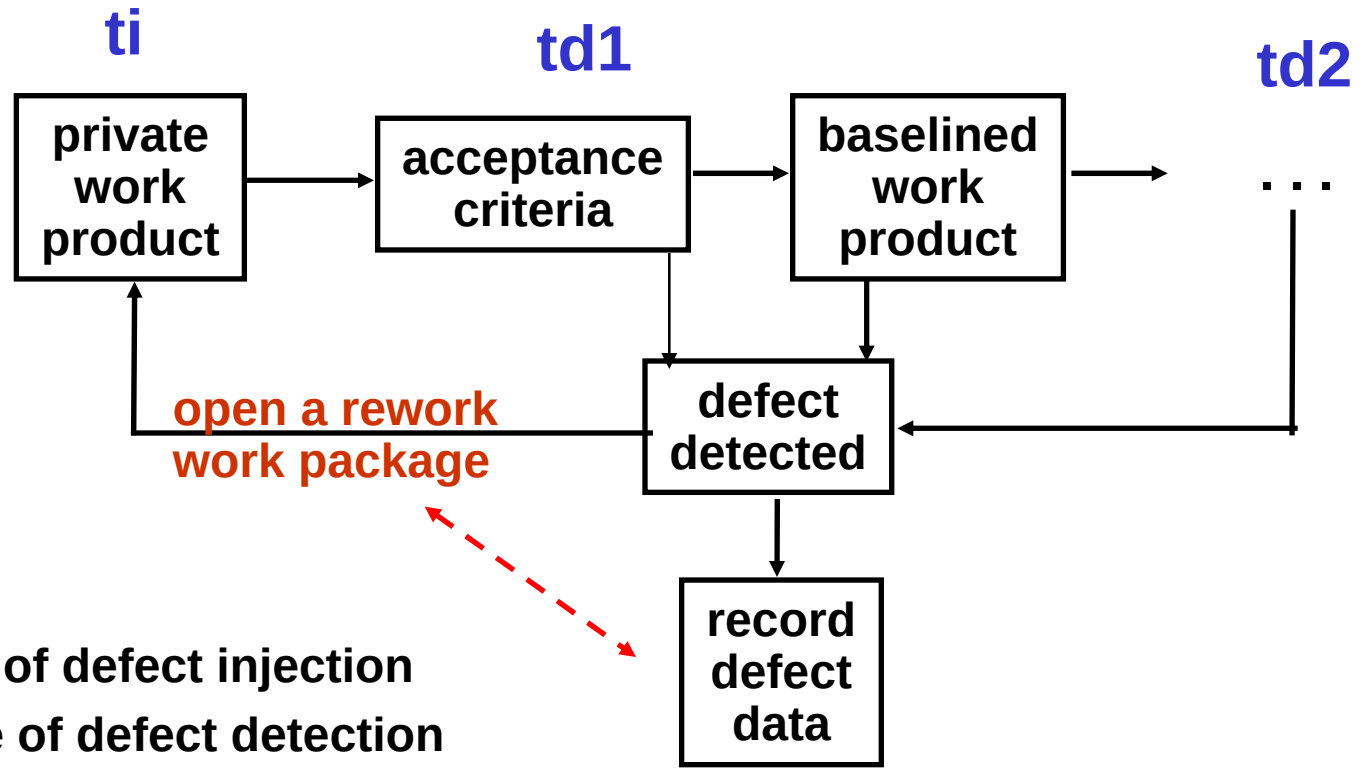
Type of rework	Characteristics	Good, bad, or ugly?
Evolutionary	Work performed on a previous version of an evolving software product or system to enhance and add value to it	Good—if it adds value without violating a cost or schedule constraint Bad—if it violates a cost or schedule constraint Ugly—if it smacks of "gold plating"
Avoidable Retrospective	Work performed on a previous version of an evolving software product or system that devlopers should have performed previously	Good—small amounts are inevitable; better now than later Bad—if it occurs routinely Ugly—if excessive, it indicates a need to revise work processes
Avoidable Corrective	Work performed to fix defects in the current and and previous versions of an evolving software product or system	Good—if total rework is within control limits Bad—if it results in patterns of special-cause effects Ugly—if it results in an out-of-control development process

* from the paper "Iterative Rework: The Good, Bad, and the Ugly" by R. Fairley and M. Willshire, *IEEE Computer*, September, 2005

Tracking Rework

- Work products generated by original work are placed under version control
  - upon satisfaction of their acceptance criteria, as documented in work packages
  - original work products (the first versions) are changed in response to a change request or a problem report
- Evolutionary rework is tracked by tracking time and effort devoted to change requests
- Avoidable rework is tracked by tracking time and effort devoted to problem reports

A Defect Tracking Model



ti: time of defect injection

td_j: time of defect detection

defect lifetime = ti - tdj

An Example of Defect Tracking

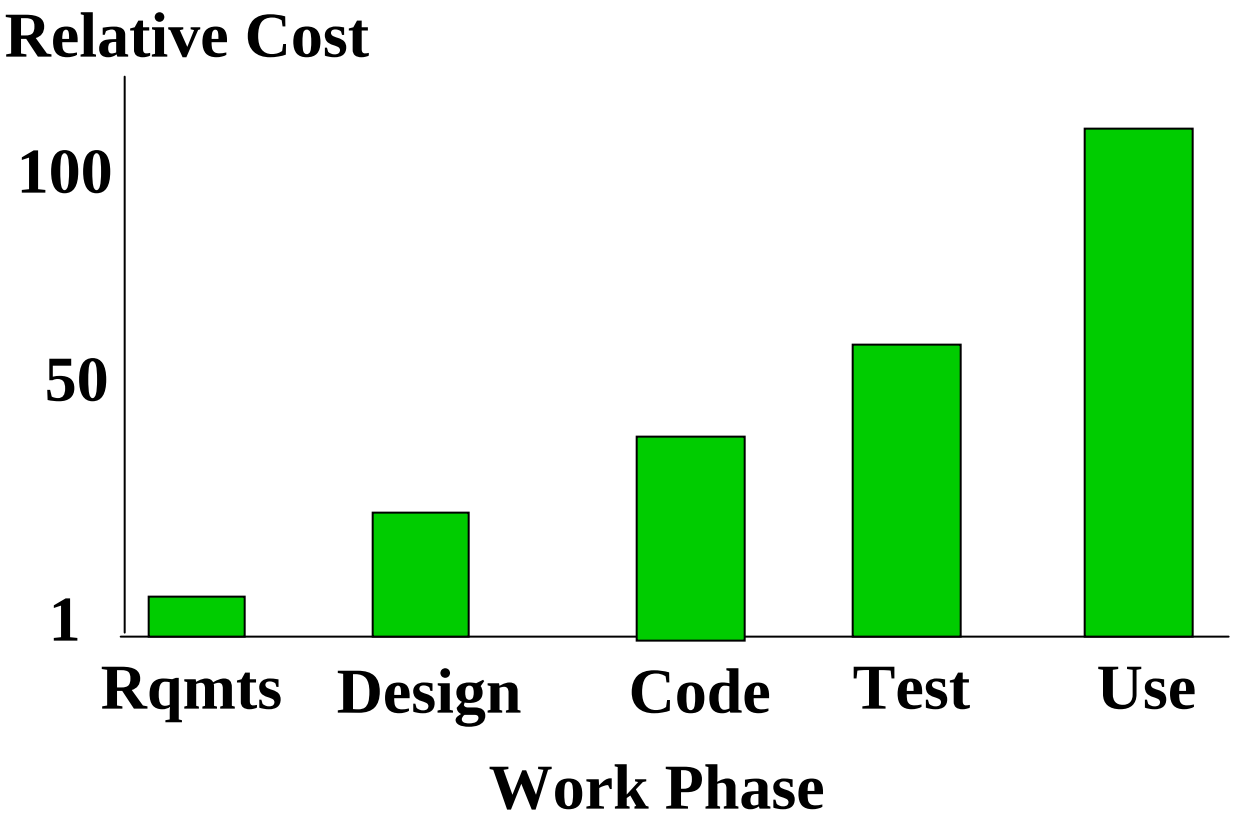
defect kind: \ phase when defects found:	Rqmts	Design	Imple.	Verif.	Valid.	Ops	Totals
Rqmts	50	25	13	6	3	3	100
Design		60	30	15	8	7	120
Imple.			80	40	20	10	150
Verif.				6	3	0	9
Valid.					7	0	7
Totals:	50	85	123	67	41	20	386

Corrective Rework for the Example

defect kind: \ work hours to correct defects:	Rqmts	Design	Imple.	Verif.	Valid.	Ops	Totals
Rqmts	50	100	130	200	250	300	1030
Design		60	90	150	225	200	725
Imple.			80	220	200	150	650
Verif.				6	10	0	16
Valid.					7	0	7
Totals:	50	160	300	576	692	650	2428

note that defects found later require exponentially more effort to fix than defects found earlier

Relative Cost to Fix a Software Defect



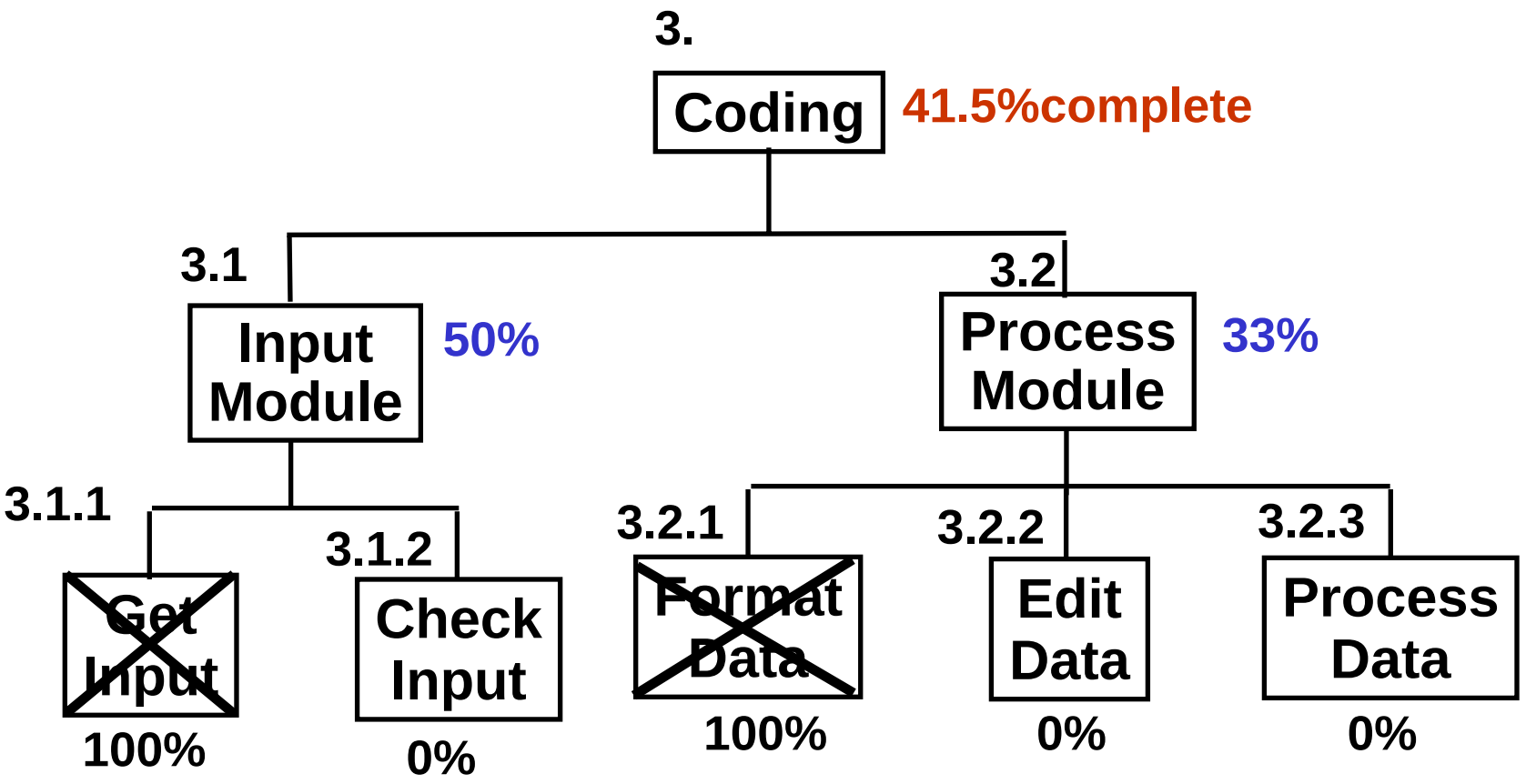
Binary Tracking of Work Packages, Change Requests, and Problem Reports

- Binary tracking requires that progress on a work package, change requests, and problem reports be counted as:

  0% complete until the associated work products pass their acceptance criteria

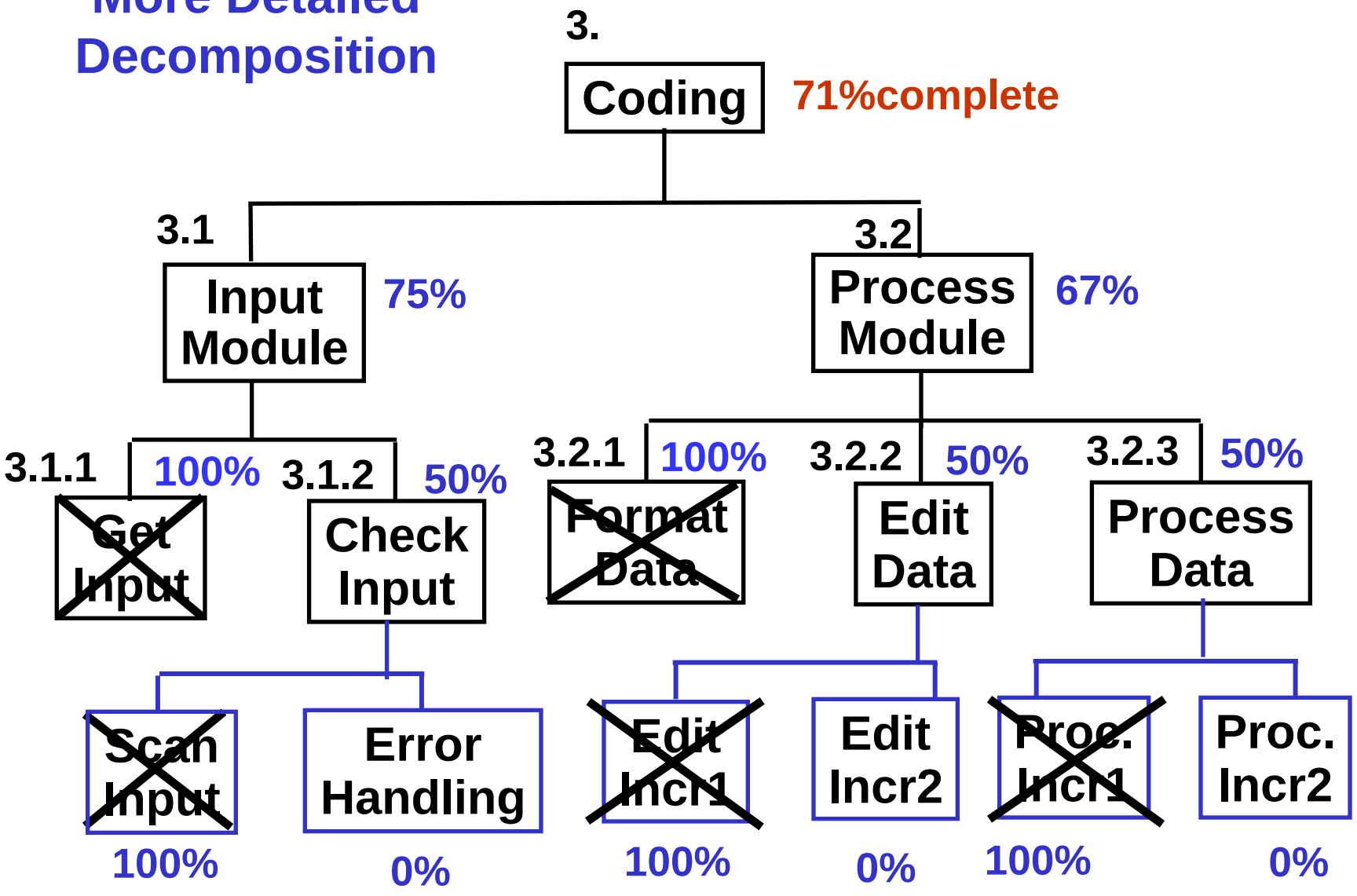
  100% complete when the work products pass their acceptance criteria

Binary Tracking of Work Packages



assuming all work packages require equal effort;
if not, they should be weighted by relative effort

More Detailed Decomposition



An Observation

- The tracking examples report progress as:
  1. 41.5% complete

     and
  2. 71% complete
- Finer levels of detail provide increased accuracy of measurement
  - at the risk of micro-management

  Hence, the 40 staff-hour rule-of-thumb

  which is a good compromise between accuracy of measurement and micro-management

The 40 Staff-Hour Rule of Thumb

- Original work should be decomposed to 40 staff-hour work packages for purposes of measurement and control
  - software developers may decompose their individual units of work
    - e.g., daily builds of their software

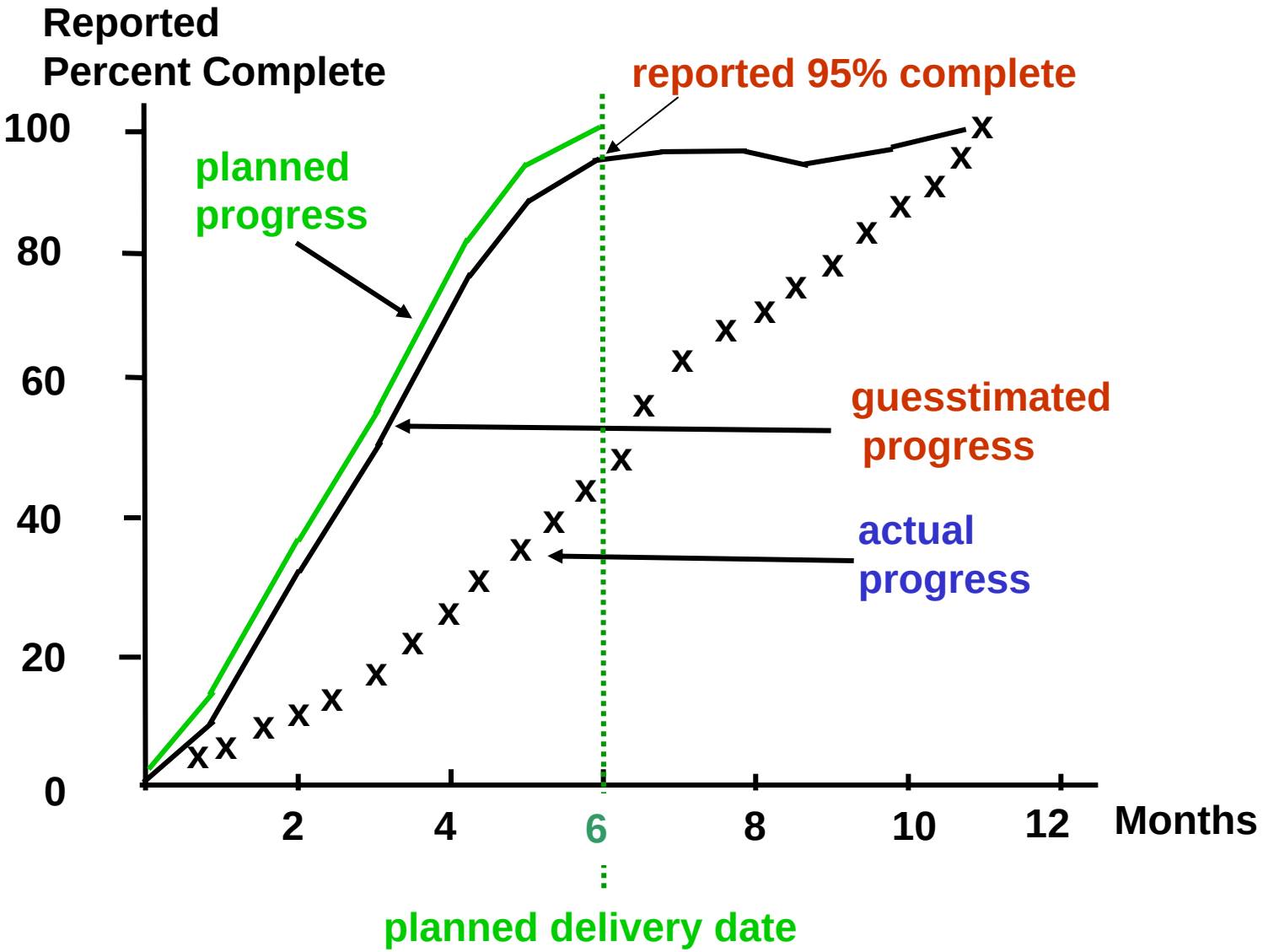
BINARY TRACKING
AND THE 95% COMPLETE SYNDROME

Binary tracking of work packages, with objective acceptance criteria for the work products, *is the only technique known to us* that can provide accurate status information for software projects

- and prevent the 95% complete syndrome -

The 95% complete syndrome:
The product is reported to be 95% complete as the scheduled completion date approaches; it remains that way for a long time

THE 95% COMPLETE SYNDROME



Estimating Cost and Schedule to Complete A Software Project Using Binary Tracking

An example:

- Assume a 20,000 LOC system (estimated), with development metrics:

270	of 300	requirements designed: 90%
750	of 1000	modules reviewed: 75%
500	of 1000	modules through CUT: 50%
200	of 1000	modules integrated: 20%
43	of 300	requirements tested: 14%

CUT: Code and Unit Test

- These numbers are obtained using binary tracking of work packages

An Example (2)

- Also assume our typical distribution of effort is*:
  - Arch. Design: 17 %
  - Detailed Design: 26 %
  - Code & Unit Test: 35 %
  - Integration Test: 10 %
  - Acceptance Test: 12 %
- Percent complete is therefore:

  90(.17)+75(.26)+50(.35)+20(.10)+14(.12)

  = 56% complete

* these percentages include typical amounts of rework

An Example (3)

- Project is 56% complete; 44% remains
- Effort to date is 75 staff-months
- Estimated effort to complete is therefore:

  (44 / 56) * 75 = 60 staff-months

Estimating Schedule to Complete

- We have used 75 staff-months of effort and completed 7 months of the project
- Staffing level:

  75 SM / 7 MO ~ 11 persons
- Remaining estimated effort: 60 SM
- Schedule to complete:

  60 / 11 = 5.5 months: 7 + 5.5 = 12.5
- Adding 4 people to the project:

  60 SM: 15 people for 4 months

  schedule: 7+4 => ≥ 11 MO

  Q: why ≥ 11 MO?

but not 30 people for 2 months

A Caution

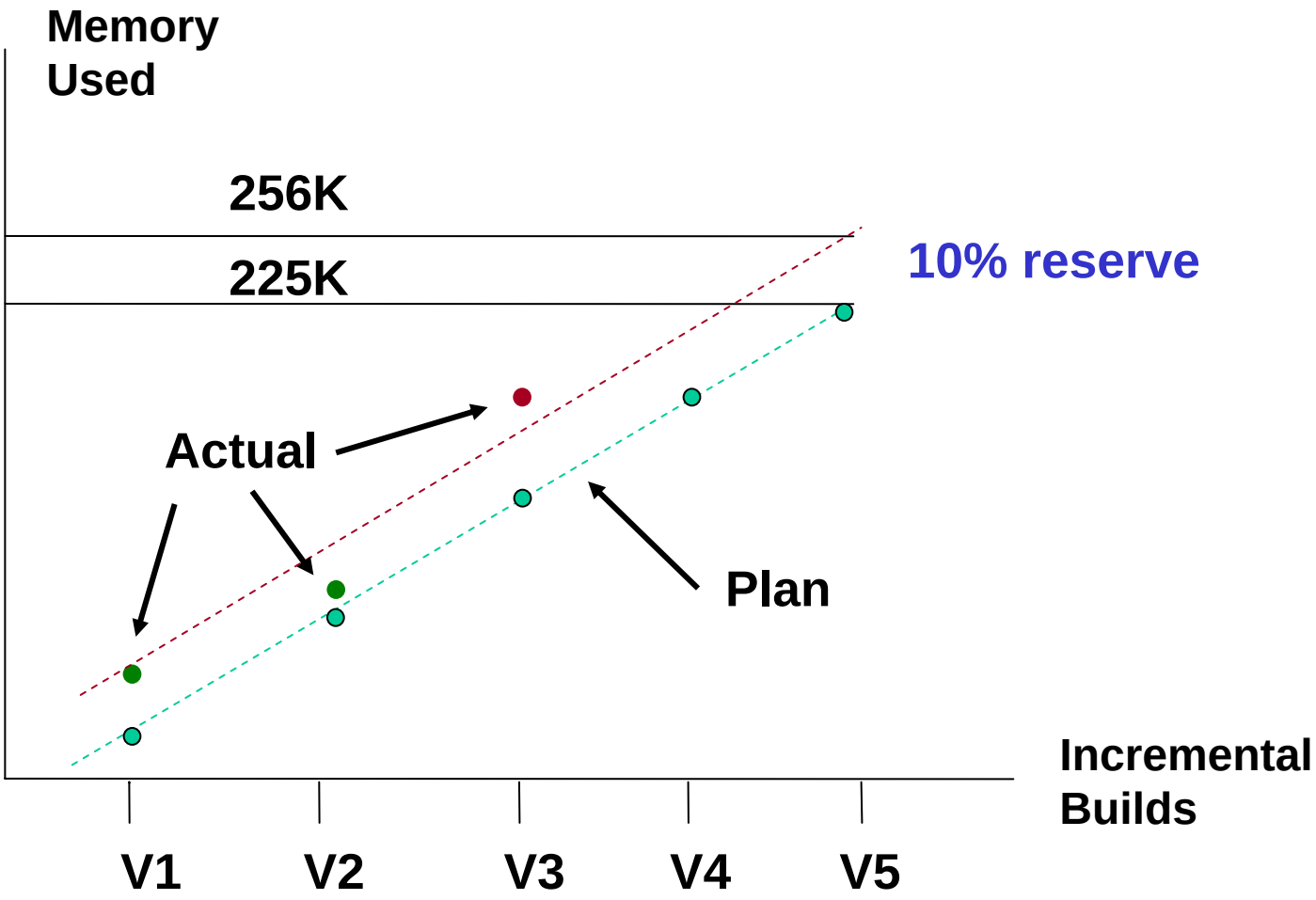
- Caution: this model assumes the remaining work to be done is at the same level of difficulty as the work completed
- Each identified work package should have an associated effort factor, ranked on a scale of 1 to 10 (using an ordinal measurement scale)
- Estimates to complete should be based on work activities weighted by the effort factors

Tracking Iterative Development

- Iterative demonstrations of implemented capabilities may be the only way* to demonstrate progress of implementing product features such as:
  - functionality
  - performance
  - quality attributes
  - memory usage
  - interfaces

* because software does not have physical properties that can be modeled, analyzed, and projected

An Iterative Example: Tracking Memory Usage



Binary Tracking and Earned Value Reporting

The earned value procedure:

1. Determine resources to be tracked

   e.g., dollars, person-months, machine cycles, memory space

2. Allocate the resource budget to individual elements of the work breakdown structure

3. When the work package for an element of the WBS is completed, the allocated budget for that element is "earned back"

   "work package is completed" means the associated work products have satisfied their acceptance criteria using binary tracking

Earned Value Reporting

4. Compare the amount "earned back"
   (i.e., the allocated amount]
   to the actual amount
5. Determine the status of total allocations and expenditures to date:
   - If cumulative (actual expenditures > earned value)
     then the project is over budget
   - If cumulative (budget-to-date > earned value)
     then the project is behind schedule
     - and vice versa

Earned Value Terminology (see Table 8.9, Chapter 8)

- BCWP: Budgeted Cost of Work Performed
- ACWP: Actual Cost of Work Performed
- BCWS: Budgeted Cost of Work Scheduled

- Example: a project has completed $5000 of budgeted work; it should have completed $6000 of budgeted work; the project has spent $ 7000 to complete the work
- Q: what is the BCWP?
  - what is the ACWP?
  - what is the BCWS?

BCWP is the “earned value”

Earned Value Terminology (2)

- Budget Variance = ACWP – BCWP
  = actual cost - budgeted cost
- Schedule Variance = BCWS – BCWP
  = planned progress - actual progress

where:

ACWP: Actual Cost of Work Performed

BCWS: Budgeted Cost of Work Scheduled

BCWP: Budgeted Cost of Work Performed

BCWP is the “earned value” that is earned back when a work package is completed

performed = completed using binary tracking

Earned Value Terminology (3)

CPI = ACWP / BCWP

SPI = BCWS / BCWP

CPI is the Cost Performance Index

SPI is the Schedule Performance Index

where:

ACWP: Actual Cost of Work Performed

BCWS: Budgeted Cost of Work Scheduled

BCWP: Budgeted Cost of Work Performed

Earned Value Terminology (Table 8.9, Chapter 8)

Term	Definition	Explanation
BCWP	Budgeted Cost of Work Performed	the cumulative Earned Value for all tasks completed to date
ACWP	Actual Cost of Work Performed	actual cost of all tasks completed to date
BCWS	Budgeted Cost of Work Scheduled	planned cost of all tasks scheduled for completion to date
BAC	Budget Actual Cost	planned cost of the total project
SCD	Scheduled Completion Date	planned completion date of the project
EAC*	Estimated Actual Cost	estimated actual cost of the project based on Earned Value progress to date
ECD*	Estimated Completion Date	estimated completion date based on Earned Value progress to date
CV	Cost Variance	CV = ACWP – BCWP
SV	Schedule Variance	SV = BCWS – BCWP
CPI	Cost Performance Index	CPI = ACWP / BCWP
SPI	Schedule Performance Index	SPI = BCWS / BCWP
CVC	Cost Variance at Completion	CVC = BAC - EAC
SVC	Schedule Variance at Completion	SVC = SCD - ECD
	* where	EAC = BAC * CPI and ECD = SCD * SPI

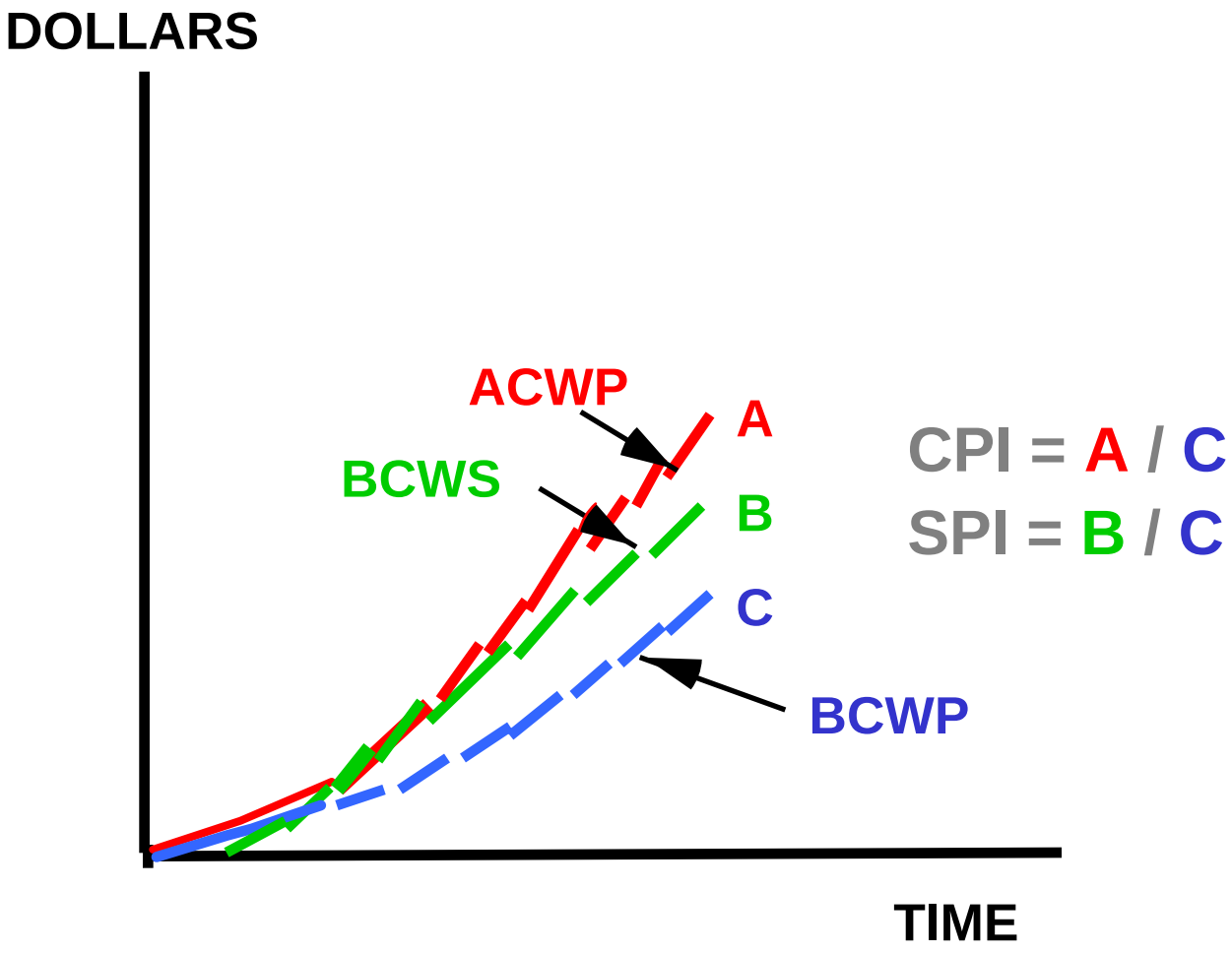
An Example (1)

- BCWP = $5000
- BCWS = $6000
- ACWP = $7000
- CV = $7000 - $5000 = $2000 cost overrun
- SV = $6000 - $5000 = $1000 behind schedule
- CPI = ACWP / BCWP = 7000/5000 = 1.4
- SPI = BCWS / BVWP = 6000/5000 = 1.2

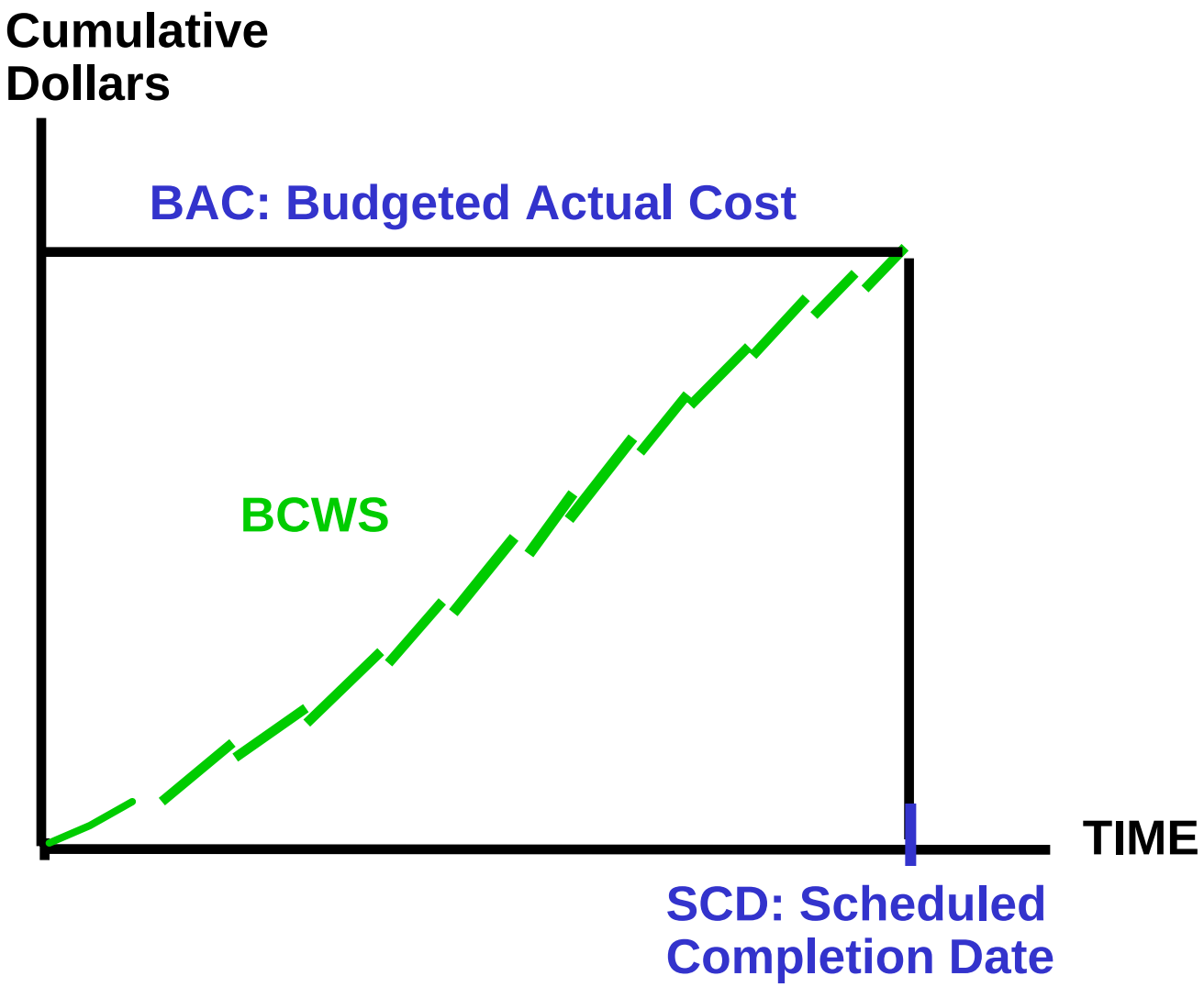
An Example (2)

- BAC = $1,000,000
- EAC = BAC x CPI = 1000000 x 1.4 = $1,400,000
- SCD = 10 months
- ECD = SCD x SPI = 10 x 1.2 = 12 months

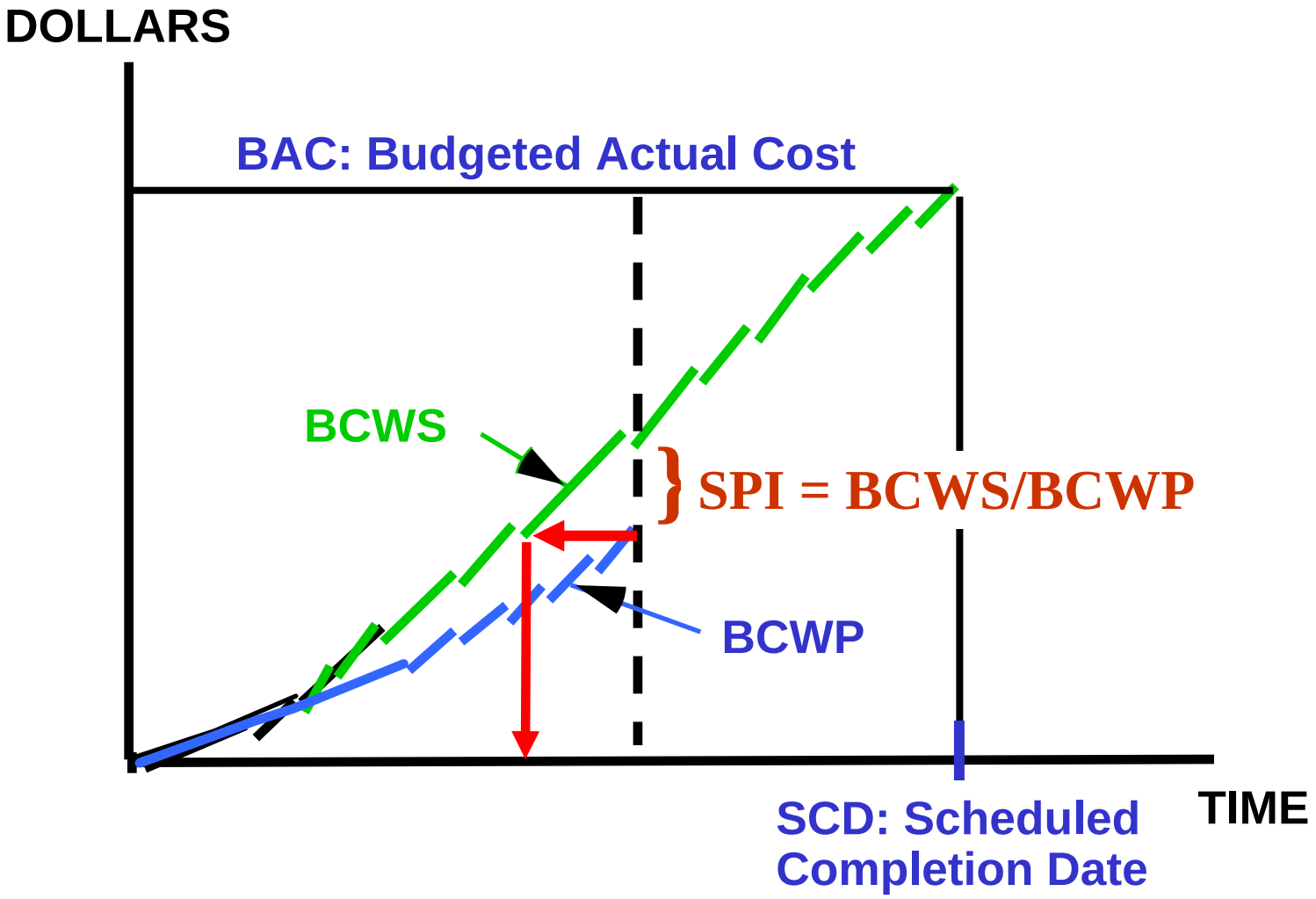
COST/SCHEDULE/PROGRESS TRACKING



THE EARNED VALUE PLAN



EARNED VALUE TRACKING OF SCHEDULE DELAY



Projecting Estimated Completion Date (ECD) and Estimated Actual Cost (EAC)

By proportional adjustment:

- Estimated Completion Date: ECD = SCD * SPI

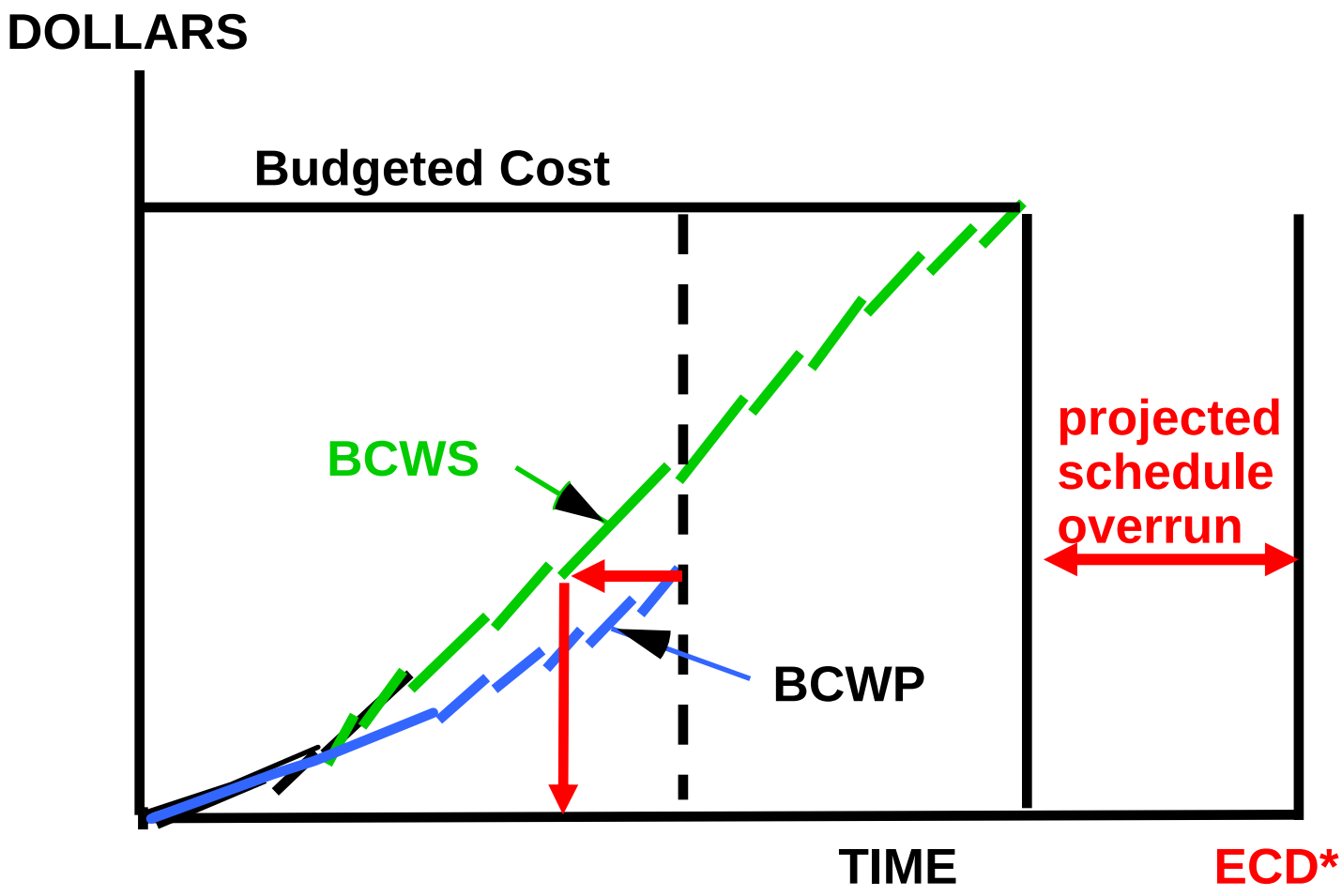
  ECD = SCD * [BCWS / BCWP]

  - where ECD: Estimated Completion Date
  - and SCD = Scheduled Completion Date
- Estimated Actual Cost: EAC = BAC * CPI

  EAC = BAC * [ACWP / BCWP]

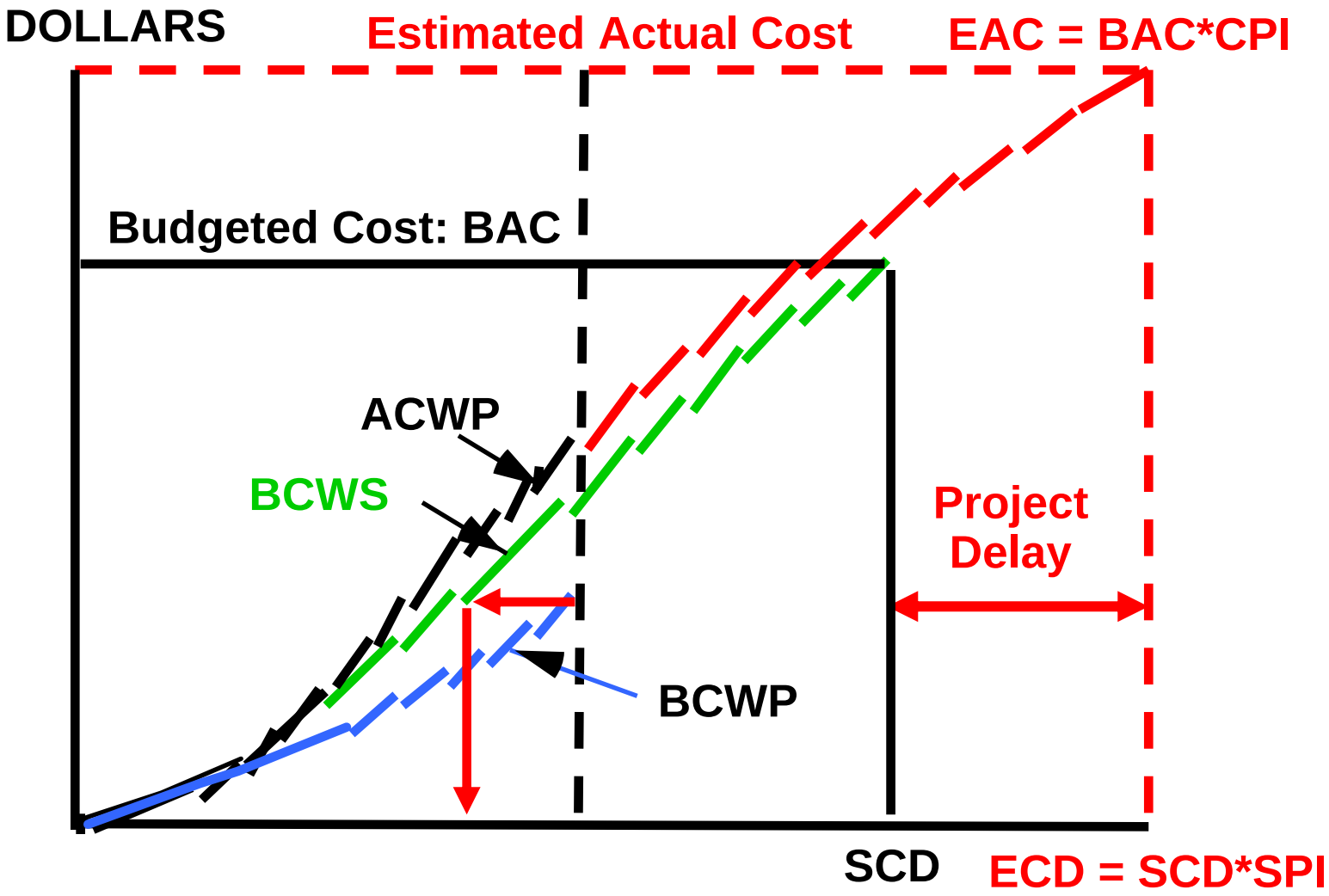
  - where EAC: Estimated Actual Cost
  - and BAC = Budgeted Actual Cost

Earned Value Tracking of Schedule Delay



*** Estimated Completion Date**

Earned Value Tracking of Cost and Schedule to Complete a Project



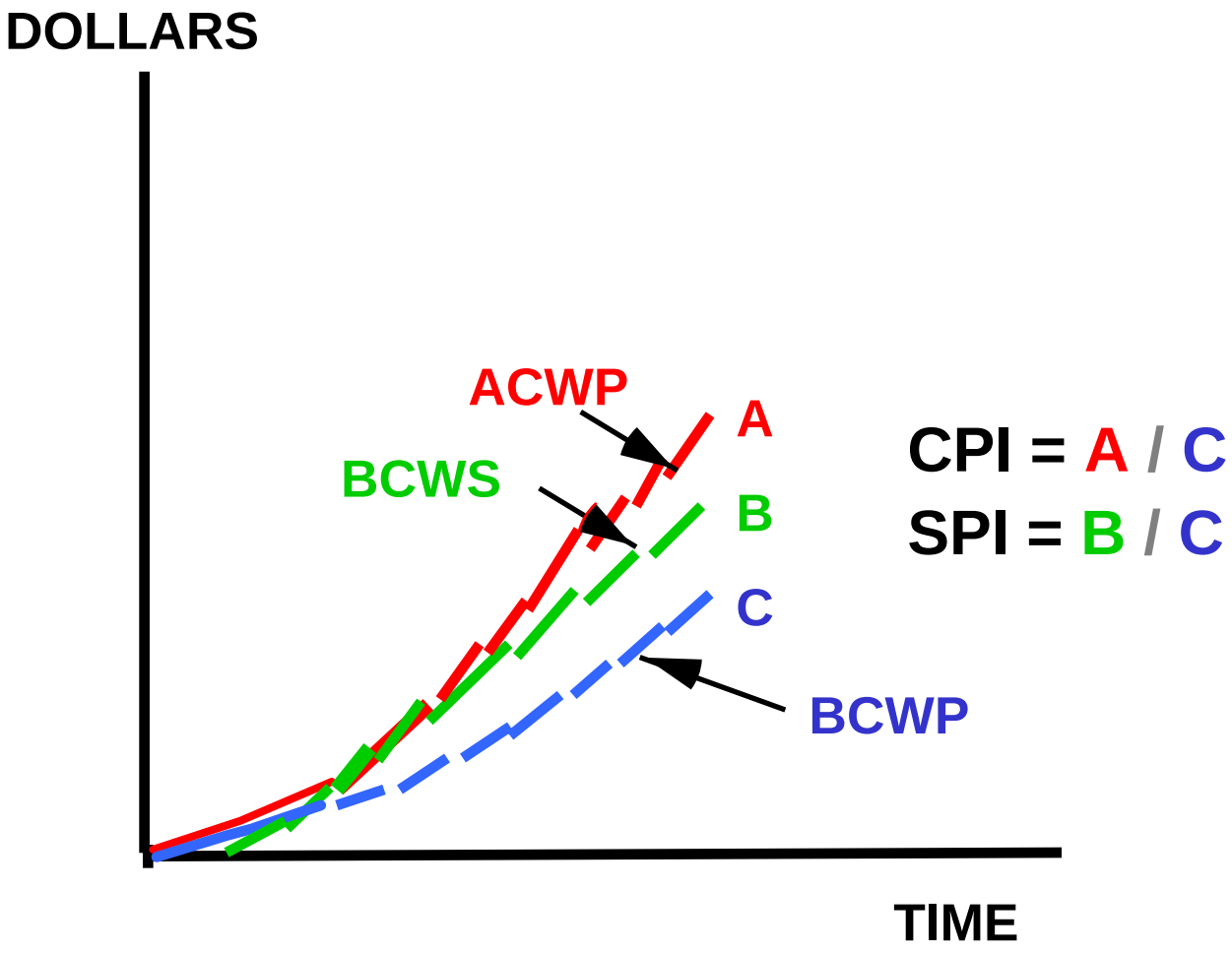
Earned Value Example 1

- A project is at the end of month 3 of a 12 month schedule with a budgeted cost of $200K
- Current status is:
  - BCWP: $40K
  - ACWP: $50K
  - BCWS: $60K
- Then:
  - CPI = ACWP/BCWP = 50/40 = 1.25
  - SPI = BCWS/BCWP = 60/40 = 1.5
- And:
  - ECD = SCD * SPI = 12*1.5 = 18 months
  - EAC = BAC * CPI = 200 * 1.25 = $250K

Earned Value Example 2

- A project is at the end of month 3 of a 12 month schedule with a budgeted cost $200K
- Current status is:
  - BCWP: $70K
  - ACWP: $60K
  - BCWS: $50K
- Then:
  - CPI = ACWP/BCWP = 60/70 = 0.86
  - SPI = BCWS/BCWP = 50/70 = 0.72
- And:
  - ECD = SCD * SPI = 12*0.72 = 8.6 months
  - EAC = BAC * CPI = 200 * 0.86 = $172K

Cost – Schedule – Progress Tracking



Earned Value Variations

orientation: / condition:	A B C	A C B	B A C	C A B	B C A	C B A
cost overrun	x	x	x			
cost savings				x	x	x
schedule slip	x		x		x	
schedule advance		x		x		x

A: ACWP

B: BCWS

C: BCWP*

* BCWP is the "earned value"

column 1 is the orientation on the previous slide

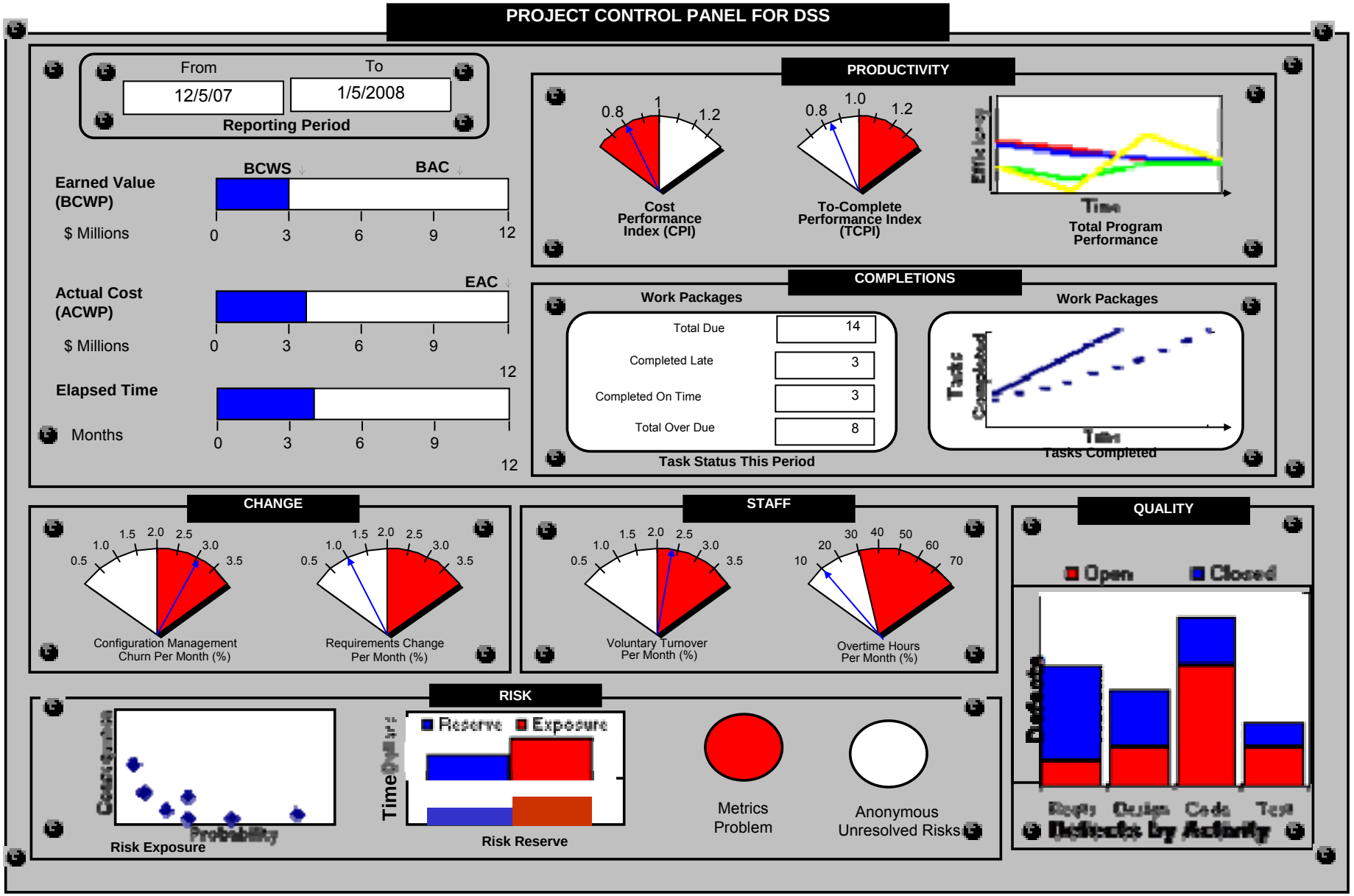
Necessary Conditions for Accurate Earned Value Reporting

- Individual work packages, change requests, and problem reports must be clearly defined and tracked
- Binary reporting of task completions must be used
- Actual time and effort must be reported for each work package, change request, and problem report
- A common format must be used to enter and report status data

Obtaining Accurate Data for Time and Effort

- Four techniques:
  1. templates provided by the version control system or “to do” task lists
  2. templates attached to computer-based task lists
  3. non-threatening, manual collection
     - on a daily basis
  4. time cards *filled out daily*
- Time and effort are reported using the WBS numbers of the associated work packages
- Information can be collected and processed on spreadsheets
  - and displayed on project control panels

A Project Control Panel



A Systematic Approach to Tracking Software Projects (1)

- The following techniques can be used together to provide a systematic approach to tracking software projects:
  1. WBS and work packages
     - tasks ~ 40 work-hours
     - rolling wave planning
  2. Tracking of rework by kind
     - evolutionary, retrospective, corrective
  3. Binary tracking
     - accurate measurement of progress
  4. Iterative development
     - frequent demos of progress

A Systematic Approach to Tracking Software Projects (2)

5. Earned value reporting
   - planned vs actual effort, cost, schedule, work completed
   - accurately updated forecasts for final cost and completion date
6. Risk registers, Top-N risk reporting, and risk confrontation
   - based on continuous risk management
   - presented in Chapter 9

The Main Points of Chapter 8 (1)

- The purposes of process measurement are:
  - to provide frequent indications of progress,
  - to provide early warning of problems,
  - to permit analysis of trends in your project,
  - to allow estimates of the final cost and completion date of your project, and
  - to build a data repository of project histories for your organization
- The primary dimensions of work to be measured and controlled are effort, schedule, and cost for each of the various work processes
- Measurement of effort, schedule, and cost must be related to tracking of work products produced using binary tracking
- The amount of effort, time, and money you invest in measurement and control is determined by considerations of risk:
  - What is the potential impact of not doing enough?
  - What is the potential impact of doing too much?

The Main Points of Chapter 8 (2)

- Possibilities for corrective action, when actual values of project attributes are not as planned or expected, include:
  - extending the schedule,
  - adding more resources,
  - using superior resources,
  - improving various elements of the development process, and/or
  - de-scoping the product requirements.
- Possibilities for corrective action that should never be used include:
  - excessive amounts and durations of overtime;
  - reduction or elimination of planned verification and validation activities;
  - reduction of planned user documentation, training aids, and so forth; and
  - reduction, without agreement of the customer, of any planned activity that would reduce the specified features or quality attributes of the system or product to be delivered.

The Main Points of Chapter 8 (3)

- Rolling wave planning by team leaders and project managers, with detailed plans for the coming month in the range of one to two staff-weeks per task, provides sufficient granularity for accurate tracking of progress
- Binary reporting of work packages is the only technique known to us that avoids the 95% complete syndrome of software projects
- Earned value reporting based on binary tracking of completed work packages provides concise reports of actual versus planned cost, schedule, and work completed
- Reporting of time spent on tasks at intervals of 2 to 4 hours each day by each individual is sufficiently accurate for most software projects
- Productivity and quality data should be reported at the level of teams and projects but never at the level of individual contributors

The Main Points of Chapter 8 (4)

- The following techniques, when used together, can provide accurate status information and accurate forecasts for software projects:
  - rolling wave elaboration of work plans documented in work packages, change requests, and defect reports;
  - iterative development with frequent demonstrations of progress;
  - baseline control of work products;
  - tracking and analysis of rework by kind (evolutionary, retrospective, corrective);
  - binary tracking of work packages, change requests, and defect reports; and
  - earned value reporting.
- Summary displays, such as the one provided by a Control Panel, can provide a succinct status reports for software projects