CS1530, LECTURE 10

SOFTWARE PROJECT MANAGEMENT

DIDN'T WE ALREADY LEARN METHODOLOGIES?

- Project management != software development methodology
- Management is…
  - A meta-level above a specific methodology
  - Includes choosing which methodology

GOALS OF PROJECT MANAGEMENT

- Ensure that the end results satisfy customer needs
- Project and product requirements and quality attributes are met
- Target milestones met and artifacts produced in a timely manner
- Ensuring that team members are operating effectively, and acting as a gatekeeper to people being on/off the team
- Ensure that tools and resources (including personnel) are being used effectively

PROJECT MANAGEMENT PROCESS - POMA

- Planning
- Organizing
- Monitoring
- Adjusting

MANAGEMENT IS NOT A WATERFALL!

- All of these phases interact with each other
- When organizing, you may realize and issue with the planning and go back and fix it
- When adjusting, you may have to re-do the work you did in organizing
- Et cetera…

PLANNING PHASE

- **“Plans are useless, but planning is everything.”**
  -Dwight Eisenhower, Supreme Allied Commander of the Allied Expeditionary Force and 34th President of the United States

- **“Everyone has a plan until they get punched in the face.”**
  -noted software engineer Mike Tyson

- Planning does not mean that you will determine everything that you need to do during development of the project

- Planning does mean that you start to think about what can go wrong

PROJECT PLANNING

- Ensure that requirements are accurately understood and specified
- Estimate work effort, schedule, needed resources
- Establish measurable goals for the project
- Determine project resource allocations
- Identify and analyze project risks

RISK MANAGEMENT

- Software development is risky (remember the CHAOS report)
- We cannot ELIMINATE risk but we can MANAGE it
- Three components:
  - Risk identification - What could go wrong?
  - Risk prioritization - Which risks should we be most concerned with? Likelihood of occuring * cost if it does occur
  - Risk mitigation - How can we avoid or minimize these risks?

ORGANIZING PHASE

- **Start determining how to execute your high-level plan**
- Determine specific tasks and schedules and how to track them
- Acquire necessary personnel for the team
- Establish how your team will track and measure progress/goals
- Establish mechanism to determine and track risk mitigation efforts

MONITORING PHASE

- Occurs during the software development process
- Ensures that the project is still on track, via -
  - Collection of project information (informal or formal)
  - Analysis and evaluation of the colleced data
  - Presentation and communication of that information

ADJUSTING PHASE

- Based on information obtained in the Monitoring phase, how do we modify our development in order to meet our goals?
- This may mean adjusting the Iron Triangle of Project Management-
  - Cost /Resources
  - Schedule
  - Project scope / quality

SOFTWARE ESTIMATION

- One of the most difficult things to do correctly in software engineering
- General formula
  - Units of effort = $a + b(\text{size})^c + \text{ACCUM}(\text{factors})$
- Probably too generic to be of much actual use

SOFTWARE ESTIMATION TECHNIQUES

- COCOMO and COCOMO II are two relatively heavyweight software estimation techniques
  - Determine mode of project based on eight parameters
  - Estimate size of project in KLOC (1000's of lines of code)
  - Calculate 15 cost drivers and their potential impact on the product
  - Insert all of these estimated values into one of three equations, based on the mode, e.g.
    - Organic: Effort = [3.2 * (size)1.05] * PROD(cost drivers)

FUNCTION POINT ESTIMATION

- Developed because using KLOC as a measurement (central to COCOMO) has many issues
- Idea of FPE - determine number of:
  - External inputs
  - External outputs
  - External inquiries
  - External logical files
  - External interface files
- For each component, determine weight based on complexity: simple, average, or complex

FUNCTION POINT ESTIMATION

- UFP (unadjusted function point) = multiply FPs by FP weights
- Then consider 14 technical factors on a scale of 0 - 5, calculate TCF (technical complexity factor) via following formula:
  - TCF = 0.65 + (0.01 * ∑(individual complexity factors))
- FP = UFP * TCF
- Effort = (X FP) / (N FP/Person-month)

LORENZ-KIDD OO ESTIMATION

- A very simple method of estimating effort for object-oriented systems
- … and the version the book covers is actually simplified from the original Lorenz-Kidd 1994 method
- Very lightweight compared to COCOMO, COCOMO II, or FPE
- Lots of assumptions and prior experience necessary
- But estimates are often pretty far-off no matter what we do

LORENZ-KIDD OO ESTIMATION

- Estimate number of class in the problem domain (i.e. that would relate specifically to solving the problem)
- Determine type of user interface (No UI, Simple Text, GUI, Complex GUI), which gives you a multiplier (2.0, 2.25, 2.5, 3.0)
- Multiply classes by weight to get additional number of classes.
- Add additional classes to original estimate of classes
- Based on previous experience, determine average developer productivity (between 15-20 days/class according to Lorenz-Kidd)
- Seems like a long time per class! But includes testing, design, defect fixes during development, etc.

LORENZ-KIDD EXAMPLE 1

- Making the main Rent-A-Cat backend API
- Estimate 100 classes for problem domain
- No UI (as it is just an interface), weight = 2.0
- Developer productivity = 15 days/class
- ((100 * 2.0) + 100) * 15 = 4500 developer-hours

LORENZ-KIDD EXAMPLE 2

- Rent-A-Cat Front-End
- Estimate 20 classes in problem domain
- Complex UI, weighting = 3.0
- Productivity = 20 person-days/class
- ((20 * 3.0) + 20) * 20 = 1600 developer-hours

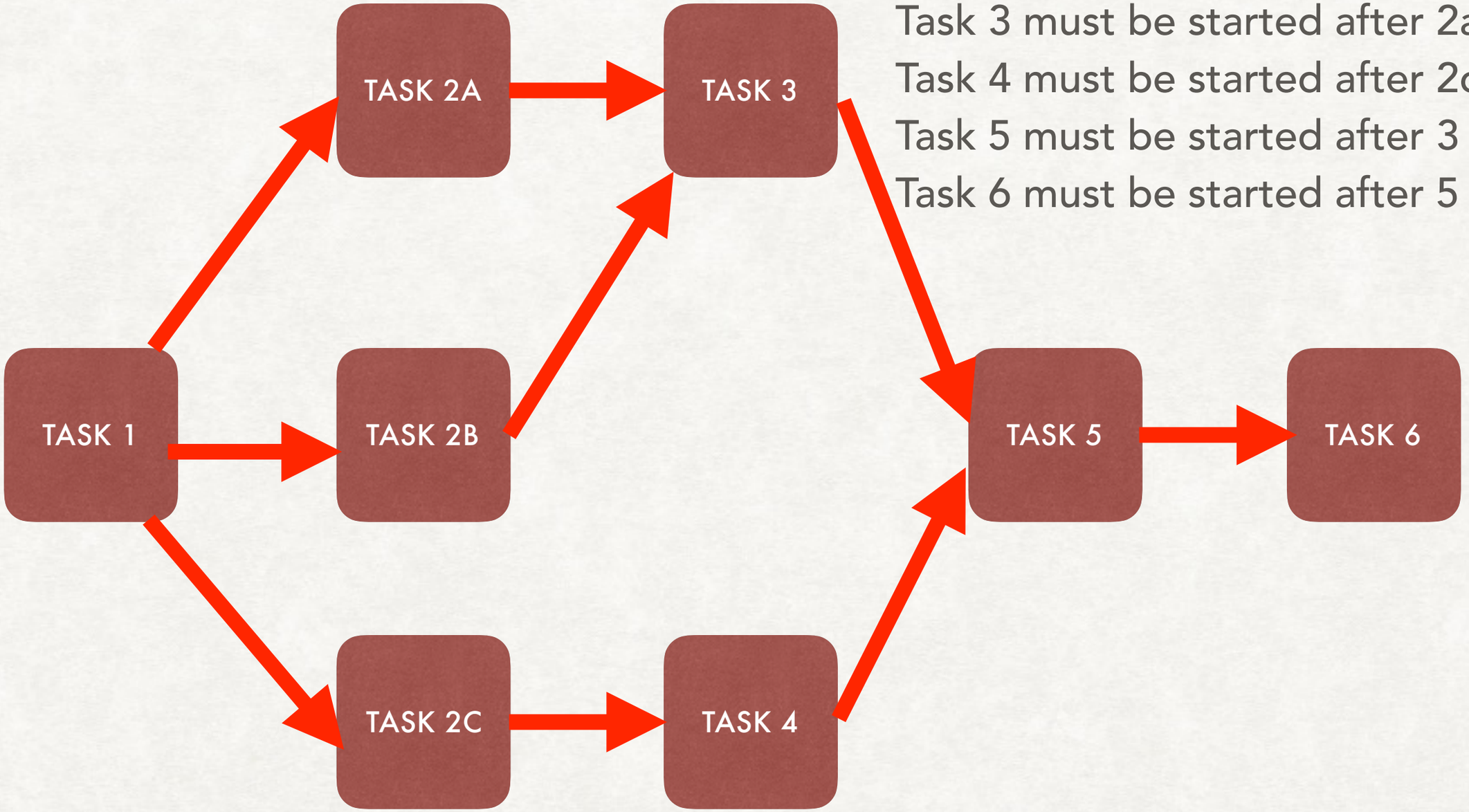
WORK BREAKDOWN STRUCTURE

WORK BREAKDOWN STRUCTURE

- “Breaks down” the work that is to be done
- Provides a basis for planning activities, scheduling staff, etc.
- Tasks are ordered and dependencies determined
- Provide estimates for length of each task
- Can then lay out a timeline which takes into account dependencies, parallelism, etc.

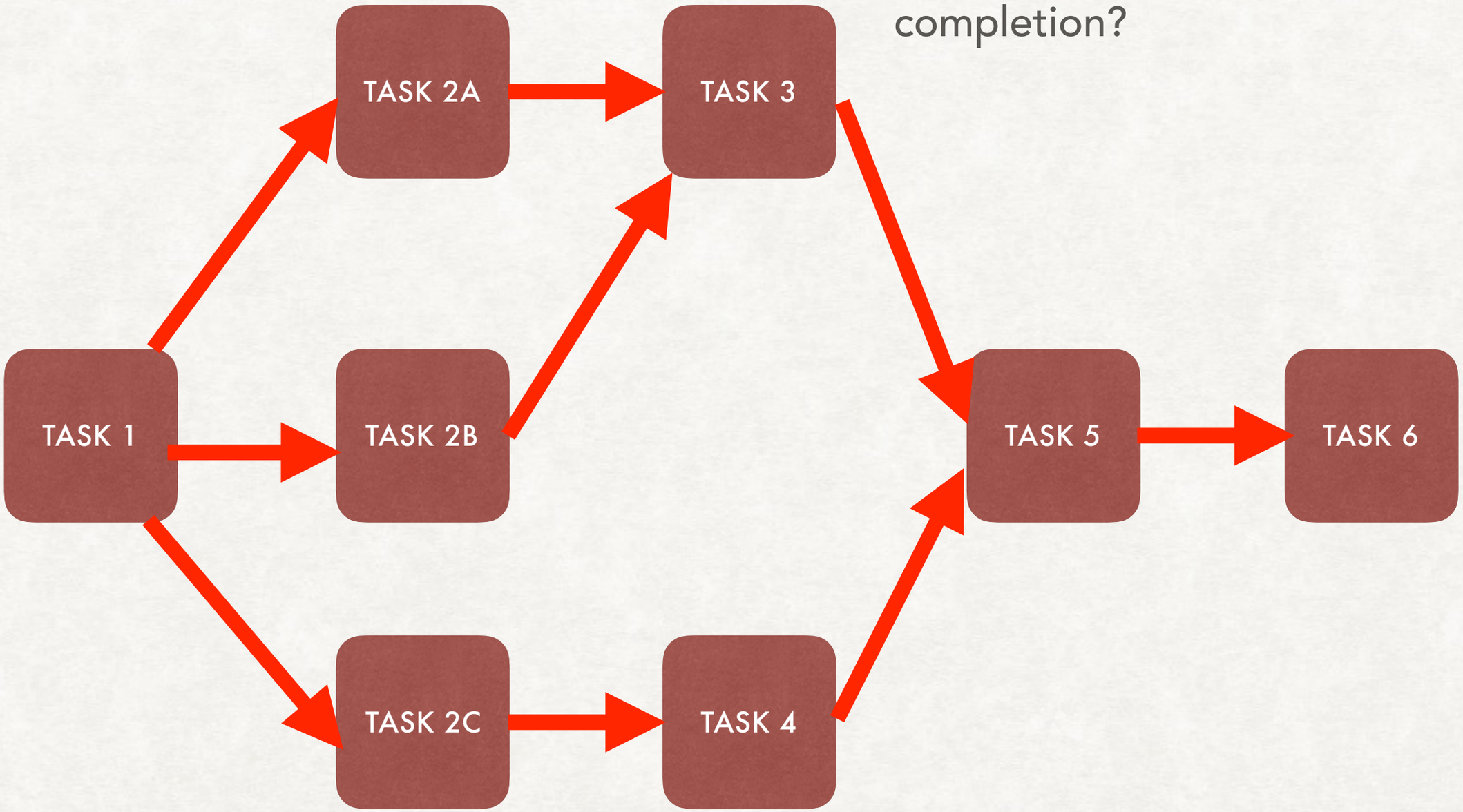
WORK BREAKDOWN STRUCTURE - TASK NETWORK

Task 1 must be done before 2a,b,c
Tasks 2a,b,c can be done in parallel
Task 3 must be started after 2a,b
Task 4 must be started after 2c
Task 5 must be started after 3 and 4
Task 6 must be started after 5



WORK BREAKDOWN STRUCTURE - SCHEDULE ESTIMATION

Assume each task takes 1 day
How do we determine time until completion?



WBS - INITIAL SCHEDULE ESTIMATE

Task	Person
1	X
2a	X
2b	Y
2c	Z
3	X
4	Y
5	Z
6	Z

Shortest path = 5 days

Note: Assumes a perfect world! Enough resources, engineers, no illnesses, no unforeseen delays, etc. Also assumes that a task absolutely cannot be started before its predecessor is complete - not always true.

“

HOFSTADTER’S LAW:
IT ALWAYS TAKES LONGER THAN YOU EXPECT, EVEN WHEN YOU TAKE INTO ACCOUNT HOFSTADTER'S LAW.

— Douglas Hofstadter, Gödel, Escher, Bach: An Eternal Golden Braid

”

MONITORING PROGRESS W/ EARNED VALUE

- Earned Value: Comes from the US government and government contracting world
- Core idea: compare how much effort has been expended against how much effort was planned to have been expended
- Part of “Monitoring”

EARNED VALUE IN A NUTSHELL

- Create WBS and initial schedule estimate
- Compare actual completion date of tasks to original estimated completion dates of tasks
- Express this as a percentage
- Extrapolate this across the rest of the project to tell you what percentage of the project is complete and adjust (next phase of POMA!) estimats

EARNED VALUE IN A NUTSHELL EXAMPLE

- Assume 3 tasks, A, B, and C, each of which should take 10 days and are in sequence (i.e., first A is completed, then B, then C) for a total of 30 days of effort
- After 20 days, only A is completed. We had estimated A and B would be completed by now
- At 2/3rds of the way through the project, only 1/3 of the work as originally estimated is done
- Earned value = 33% (1/3)
- Note our reliance on assumption that original estimates were good

ACTUAL EARNED VALUE CALCULATION - TERMS

- **BCW (budgeted cost of work)** - Estimated effort for each task
- **BCWS (budgeted cost of work scheduled)** - Sum of estimated efforts of all tasks that were schedule to be completed on a given date
- **BAC (budget at completion)** - Estimate of total project effort (sum of all BCWs)
- **BCWP (budgeted cost of work performed)** - Sum of estimated efforts of all tasks actually completed on a given date
- **ACWP (actual cost of work performed)** - Sum of actual efforts that have been completed on a given date

EXAMPLE - 4 TASKS (SEQUENTIAL)

- Project start date: 2 OCT 2017, one person
- Task A - Estimated effort (BCW) = 5 person-days
- Task B - Estimated effort (BCW) = 2 person-days
- Task C - Estimated effort (BCW) = 3 person-days
- Task D - Estimated effort (BCW) = 4 person-days
- BAC = 5 + 2 + 3 + 4 = 14 person-days
- Estimated completion date: 16 OCT 2017 (working weekends)

EARNED VALUE: 9 OCT

- BCWS = (Task A) 5 + (Task B) 2 = 7 person-days
- Should hope that BCWP >= BCWS, but only Task A is complete
- BCWP = (Task A) 5 = 5 person-days
- ACWP = 7 person-days
- The work that we budgeted to take 5 person-days instead took 7
- EV = (BCWP) 5 / (BAC) 14 = 0.357 = 36% of earned value
- However, 50% of budget/schedule used up!

EARNED VALUE DERIVED METRICS

- SV (Schedule variance) = BCWP - BCWS
  - Budgeted cost of work performed - Budgeted cost of work scheduled; what is the variance in our schedule from work supposed to have been done at this date versus how much has actually been done?
- CV (Cost variance) = BCWP - ACWP
  - Budgeted cost of work performed - actual cost of work performed; what is the variance in how much effort the task was estimated to take versus how much it actually took?