



SWINBURNE
UNIVERSITY OF
TECHNOLOGY

SWE30010

Development Project 2: Design, Planning and Management

Lecture 5

Tracking and Monitoring



Commonwealth of Australia
Copyright Act 1968

Notice for paragraph 135ZXA (a) of the *Copyright Act 1968*

Warning

This material has been reproduced and communicated to you by or on behalf of Swinburne University of Technology under Part VB of the *Copyright Act 1968* (the *Act*).

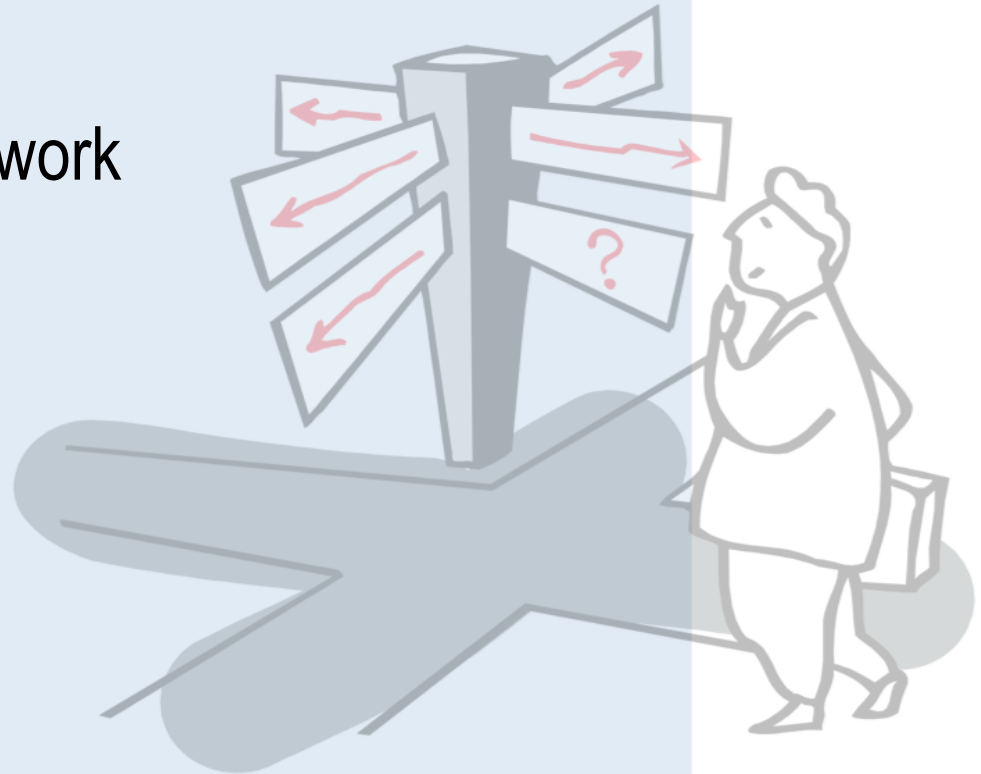
The material in this communication may be subject to copyright under the *Act*. Any further reproduction or communication of this material by you may be the subject of copyright protection under the *Act*.

Do not remove this notice.

Roadmap



- Overview, General Framework
- Tracking (and how often)
- Monitoring



Principal References



- Bob Hughes and Mike Cotterell, *Software Project Management* (4th Edition), Wiley, 2006, Chapter 9.
- Kent Beck, Martin Fowler, *Planning Extreme Programming*, Addison-Wesley, 2001, Chapters 19 and 21.
- Pankaj Jalote, *Software Project Management in Practice*, Addison-Wesley, 2002, Chapter 11.
- Ian Sommerville, *Software Engineering* (8th Edition), Addison-Wesley, 2007, Chapter 28.

Driving in the dark...



How would you feel about driving in the dark,
without a dashboard?

Why Tracking and Monitoring?



- If we know, during the progress of a project, that something is not “quite right”, there is hope that we can do something about it!

☞ *If we do not know in advance, we can only count the costs at the end...*

- Tracking and Monitoring are *risk mitigation strategies*.





What can go wrong in a project?

- Inadequate functionality of a product
 - Related to SRS
- Poor quality of a product
 - Related to quality management
- Late delivery of the product
 - Related to scheduling and scoping
- Exceeding the budget
 - Related to staffing and scheduling
- You name it!!!





Example : Basic Philosophy in XP

1. Smell a Problem*
2. Devise a Measurement
3. Display the Measurement
4. Take Corrective Action
5. If the problem does not go away, Goto 2



Source: Kent Beck, Martin Fowler, *Planning Extreme Programming*, Addison-Wesley, 2001

* Sometimes it may just be a “whiff” of a smell; other times, it may be an obvious deviation from clearly identified expected progress

It's reasonable to apply this philosophy in other process models too!

Planning, Tracking and Monitoring



- Planning: *know where we want to go*
(Melbourne, Australia)



- Tracking: *know where we are (now)*
(Melbourne – Florida!)



- Monitoring: *determine whether we are*
where we want to be





Planning, Tracking and Monitoring

- **Planning:** *know where we want to go*
 - Task decomposition, allocating resources to tasks etc.
 - Scheduling, milestones etc.
- **Tracking:** *know where we are (now)*
 - Finding out what is happening
 - Collect data about current state of project
- **Monitoring:** *determine whether we are where we want to be*
 - Comparing the current status with the targets/initial plans
 - Need a plan, a schedule, **and collected data during the execution**
 - To exercise control over the project
 - To ensure the targets are met
 - Might require contingency plans to address any issues! (part of Risk Management)

Planning, Tracking and Monitoring (cont.)



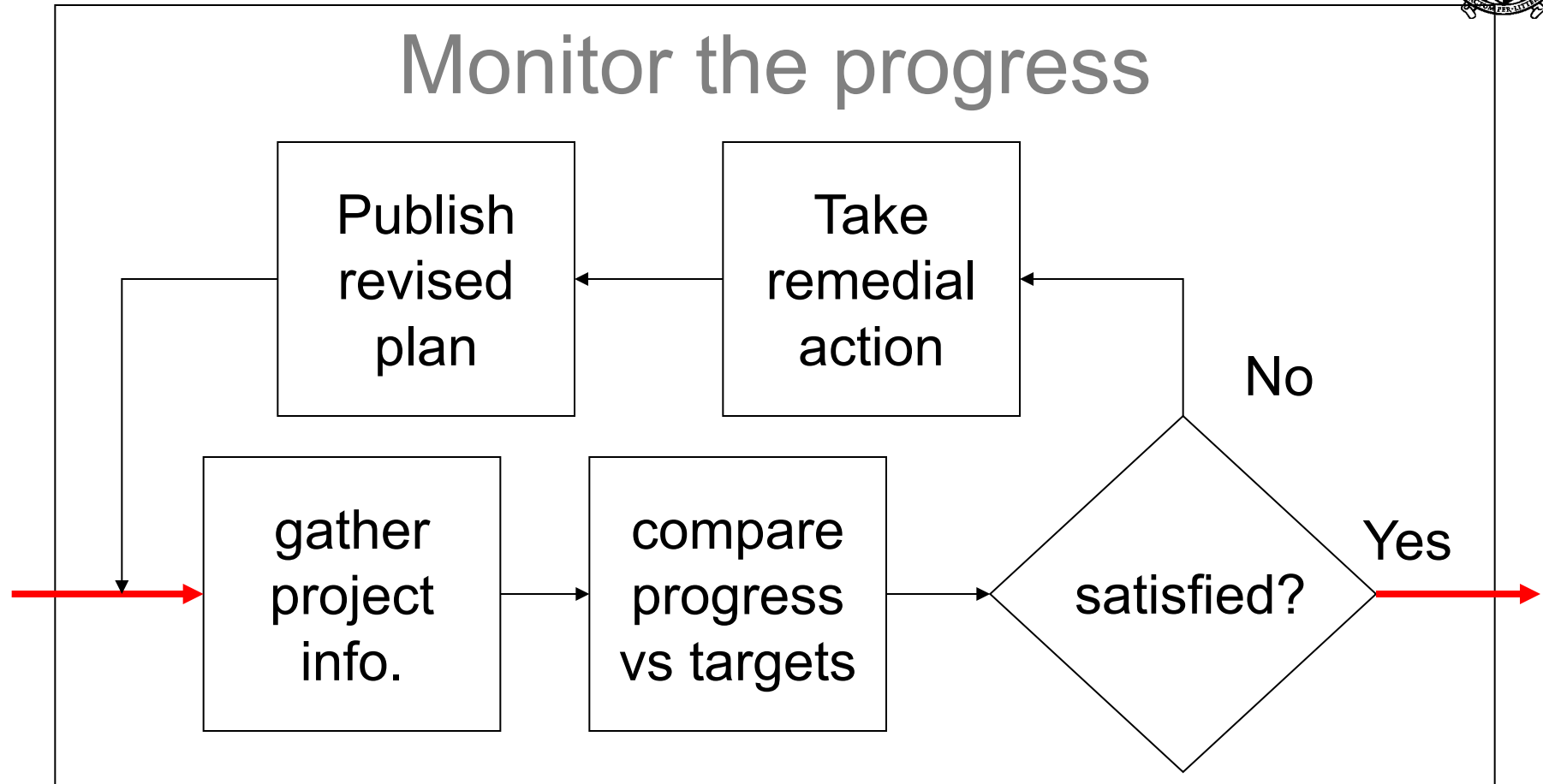
- Are in essence “meta-activities”
 - ☐ no *direct benefit* for project progress,
 - ☞ but needed to ensure project progress!
- Disrupt from “real” project work
 - ☐ Take time!
 - ☐ have to be scheduled carefully
 - ☐ Should not distract project workers from their real purpose
 - ☐ need to collect *important* information/measures
 - ☞ *should never become the end, always the mean!*
- KISS: **K**ep **I**t **S**imple and **S**hort !



Importance of Tracking and Monitoring

- Helps ensure (but *not* guarantee) that the project
 - ☐ Can be delivered on time and within budget
 - ☞ Or ensure that adjustments are recognised and agreed as early as possible
 - ☞ Avoid “surprises” at the end!
 - ☐ Meets the predefined quality
 - ☐ Meets clients needs
 - ☐ ...
- ☞ *But: it cannot address fundamental problems of a project - it makes these problems **very visible!***

Tracking and Monitoring: Framework



Example : Basic Philosophy in XP

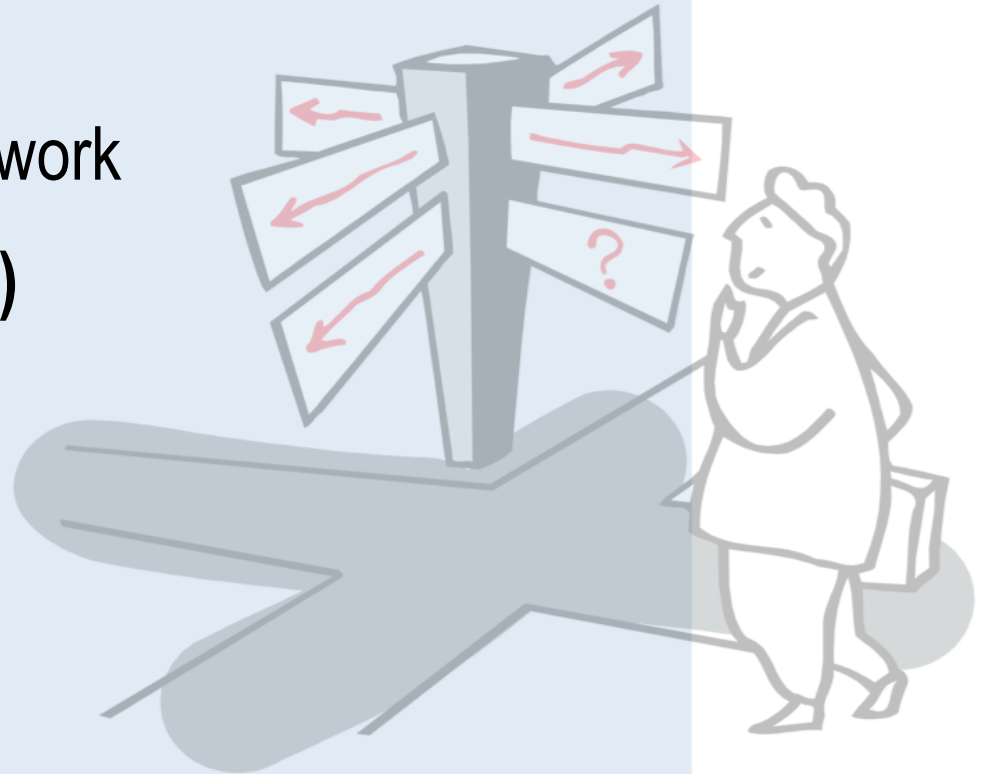
1. Smell a Problem*
2. Devise a Measurement
3. Display the Measurement
4. Take corrective Action
5. If the problem does not go away, Goto 2



Roadmap



- Overview, General Framework
- **Tracking (and how often)**
- Monitoring





Tracking Current State of Project

This involves the following steps

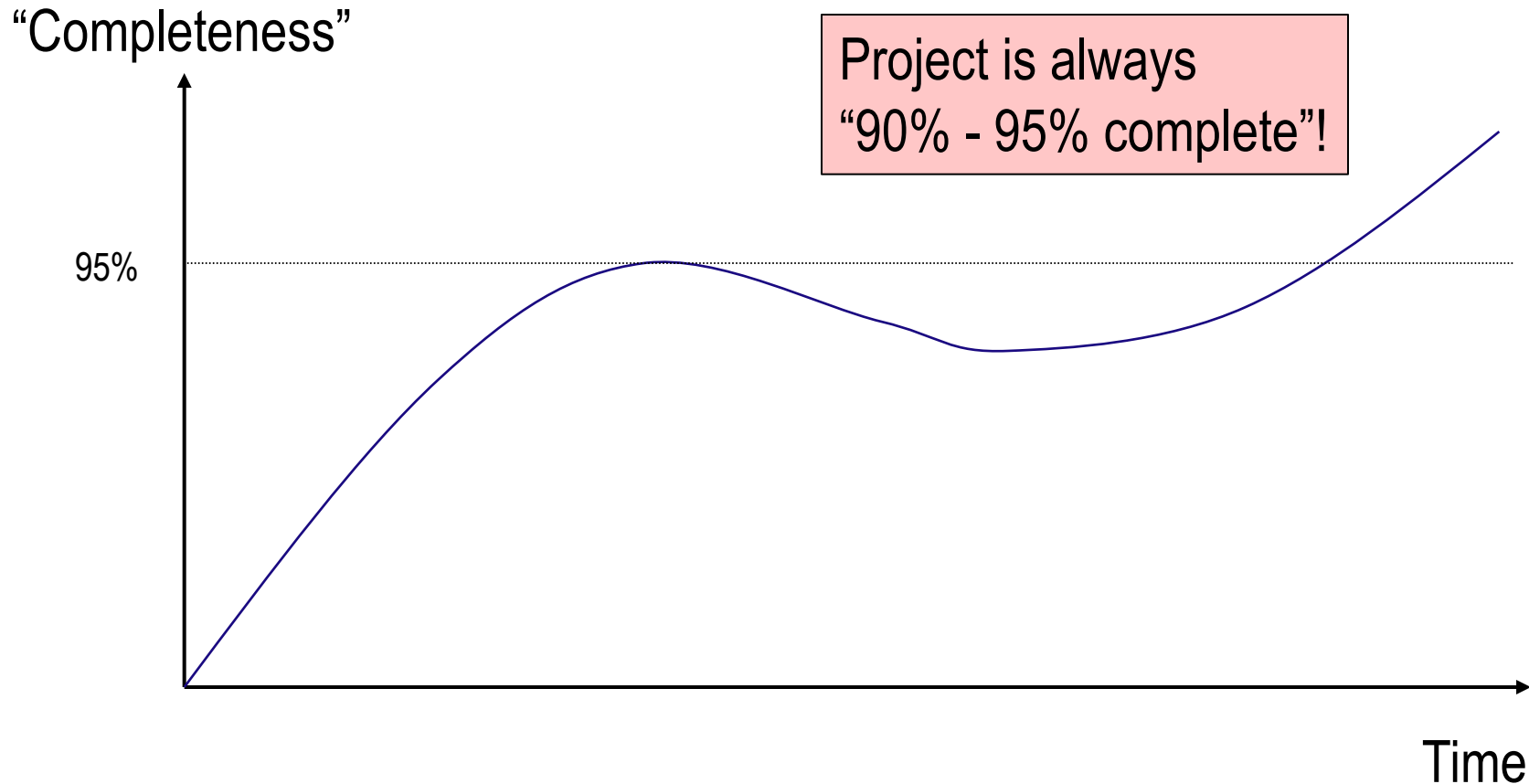
■ Setting *check points* {in advance}

- ☐ Good candidates: Milestones and deliverables
- ☐ eXtreme Programming: 1 - 2 times per week
- ☐ Scrum: (informally) at daily Scrum meetings

■ Collecting data

- ☐ Collect data *relevant* to the current progress of the project
 - ☐ Might be worth/necessary tracking risks probabilities/impacts!
- ☐ Make sure *relevant units* are used to collect data!
 - ☐ “*Percent complete*” is a bad unit to measure progress!

Percent Complete - the CS Paradox





Check Points – Different Strategies

■ Based on **regular time intervals**

- ☐ Can be weekly or monthly or quarterly (dangerous!)
- ☐ Depend on what to check and how to check it

■ Based on a **particular event**

- ☐ At the end of each activity
- ☐ In the “middle” of an activity that falls in a critical path
- ☐ In the middle of an activity that produces a high quality work product which are important to the success of the project

■ Based on **work product**

- ☐ Milestones and deliverables

■ Should be set **before** the project plan is published

- ☐ Make sure everyone knows when and what the check points are!

☞ *Checkpoints should be **risk driven**!*



Example : XP - Iteration Progress Check

Dedicated project role (i.e. *Tracker*) asks two questions about each task signed up by a developer:

- *How many **ideal days** have you worked on this task?*
 - ☐ Not directly relevant for monitoring, but useful for calibrating “Yesterday’s Weather”
- *How many **more ideal days** do you need before completion?*
- ☞ Note the unit (**ideal days** vs. percentage complete)!
- ☞ “Yesterday’s weather” is best estimator



Issues with Tracking

- Data collected must be *honest* and *accurate*

- ☞ Data should never be used to assess performance of teams!
 - ☞ Otherwise, you will be told what you “want” to hear...

- 3-Point time tracking:

- ☐ Planned vs. Unplanned vs. Ongoing work

- State of progress not always easy to determine

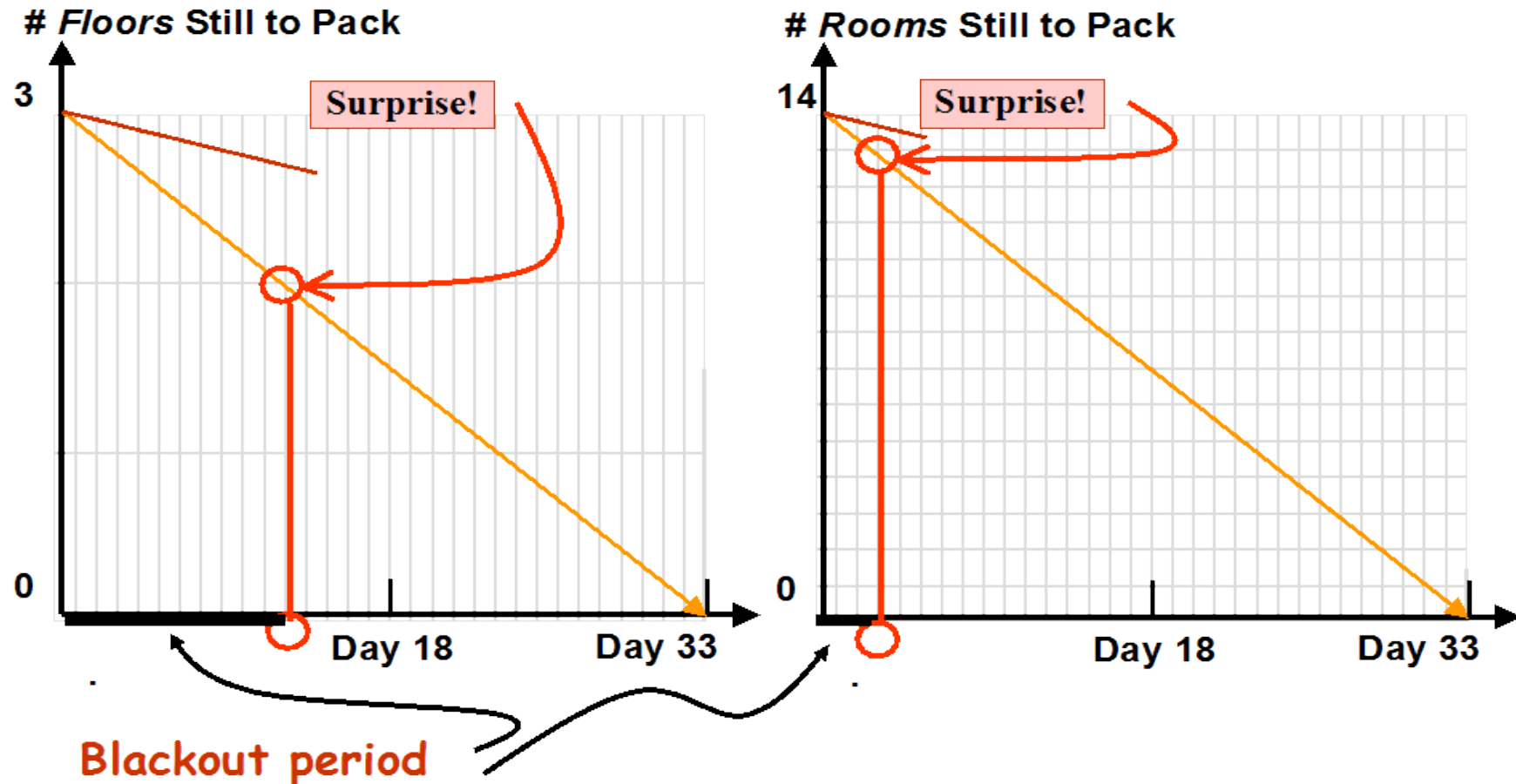
- ☐ When is a task “really” complete?
 - ☐ Who makes this decision (developer, client, project manager, team, ...)?
 - ☐ This is a BIG ISSUE



Issues with Tracking (cont.)

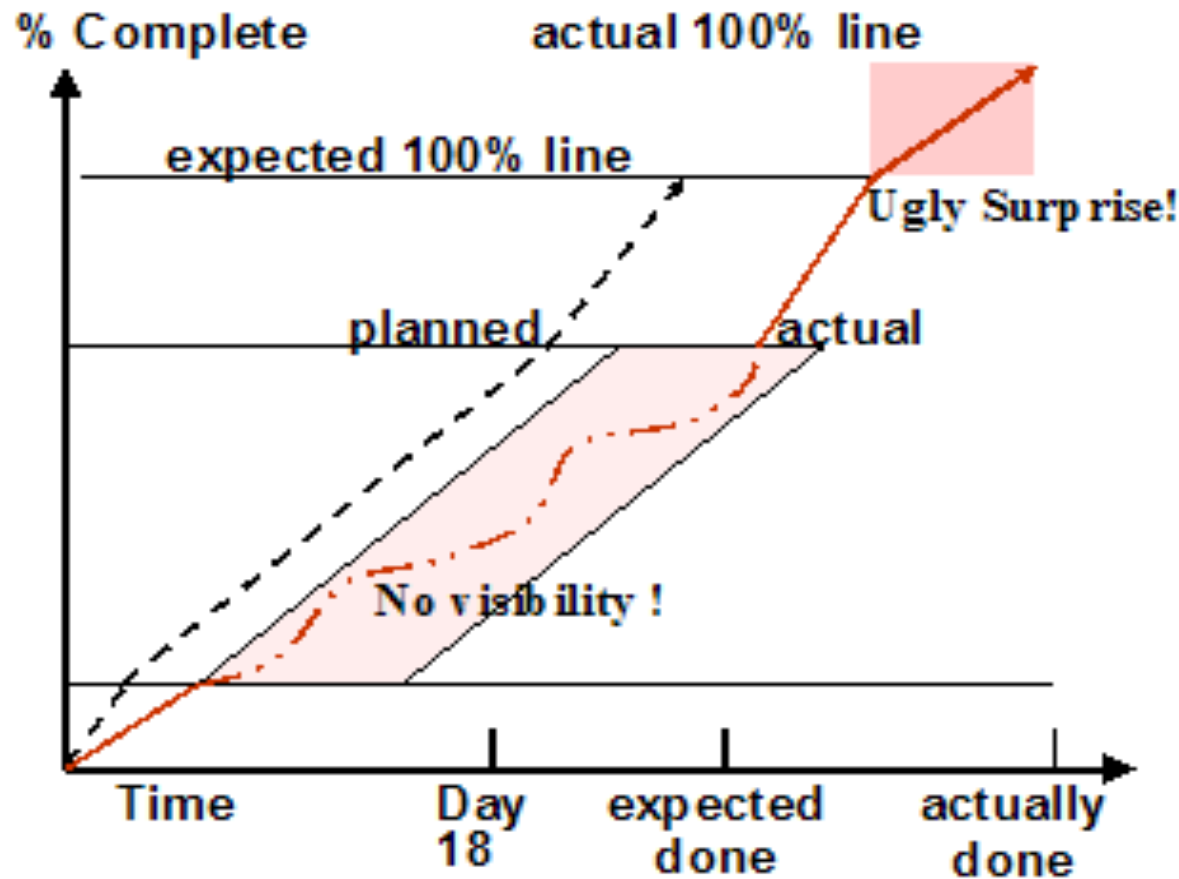
- Choose metrics that reflect the *actual state of progress*
- Estimates might be wrong
 - ☐ report progress based on *adjusted* estimates
- Units of measurement do matter
 - ☐ Choose units that cannot expand (“days” not as good as “hours”!)

Short vs. Long Time Intervals



Source: Alistair Cockburn, *Crystal Clear*, Addison-Wesley, 2004

Lack of Intermediate Check Points

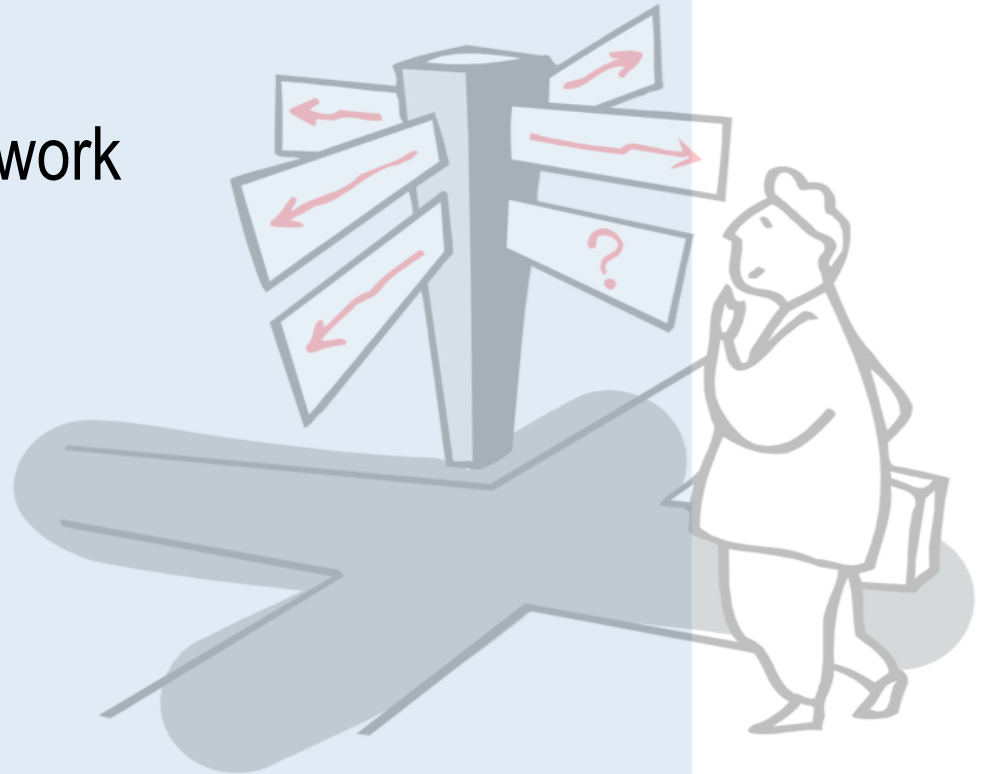


Source: Alistair Cockburn, *Crystal Clear*, Addison-Wesley, 2004

Roadmap



- Overview, General Framework
- Tracking (and how often)
- **Monitoring**

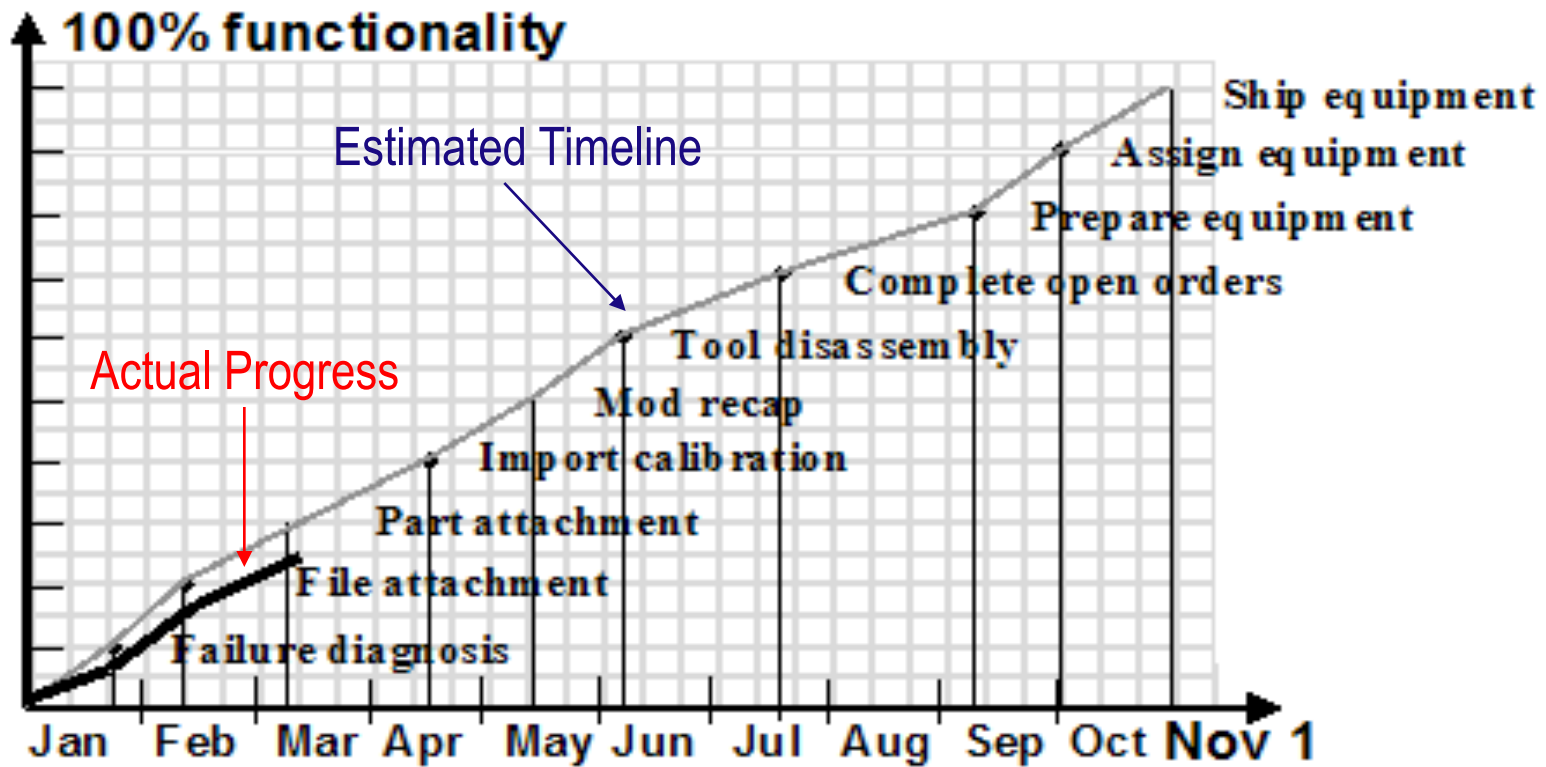




Monitoring Progress

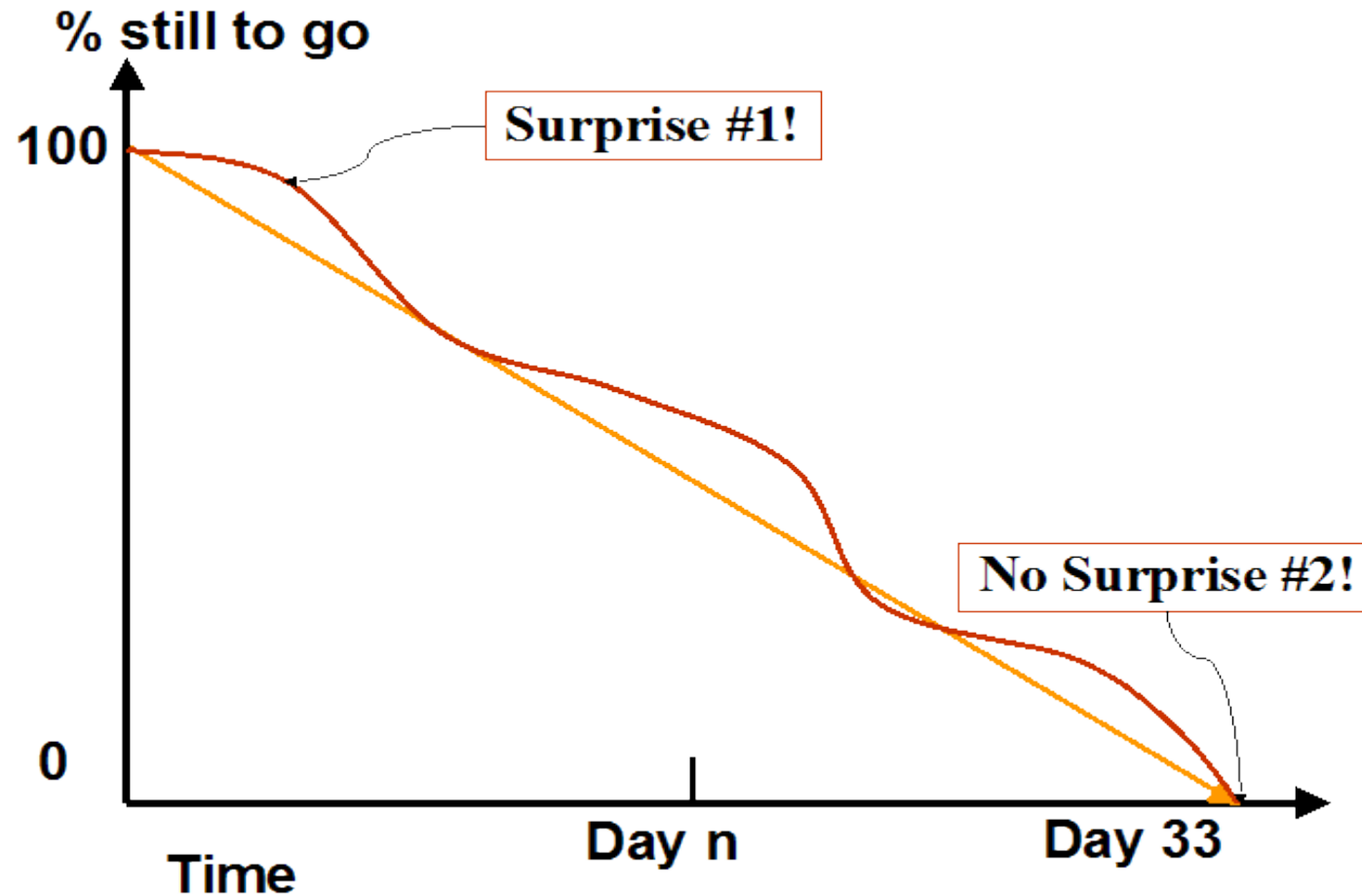
- Need a plan, a schedule, and collected progress data
 - The “fuzzier” the plan/schedule, the more difficult to assess progress!
- Need tools for *visualizing* progress
 - Graphs are (often) easier to understand than tables
 - “One picture can say more than a thousand words”
- Need to monitor both time and cost
 - Cost in general easier to monitor than time.
- *But what kind of visualizations help?*

Burn-up Charts



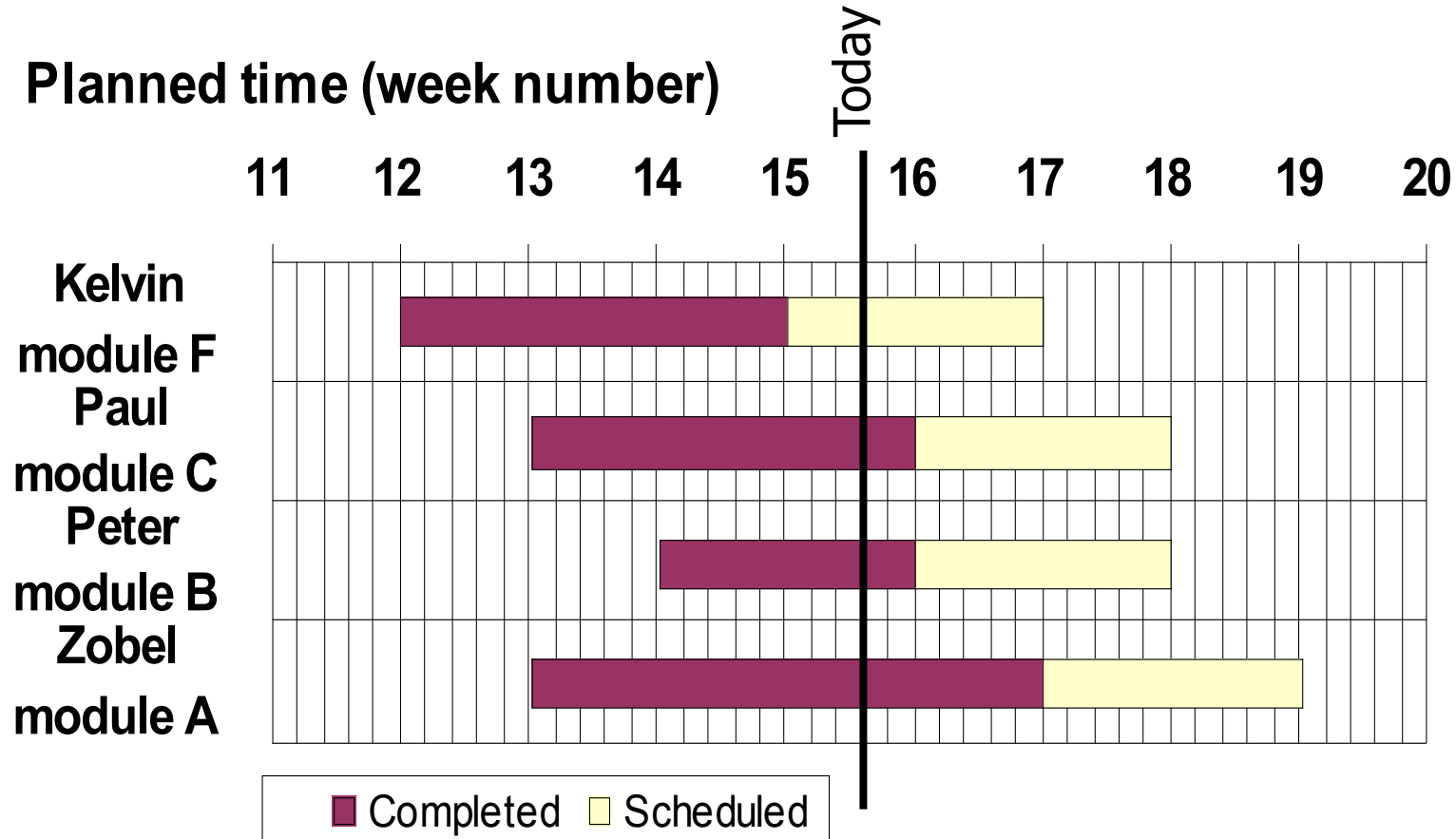
Source: Alistair Cockburn, *Crystal Clear*, Addison-Wesley, 2004

Burn-down Charts



Source: Alistair Cockburn, *Crystal Clear*, Addison-Wesley, 2004

Gantt Charts... and why they are dangerous!





Measures to Visualize

In essence, anything that is of interest for project team!

- Tasks completed vs. tasks still outstanding
- (Real) time spent vs. (ideal) time still needed to complete task
- “Size” of functionality completed vs. estimated size
- Risk assessments
- Number of bugs still to be addressed
- Amount of money from budget already spent
- ...



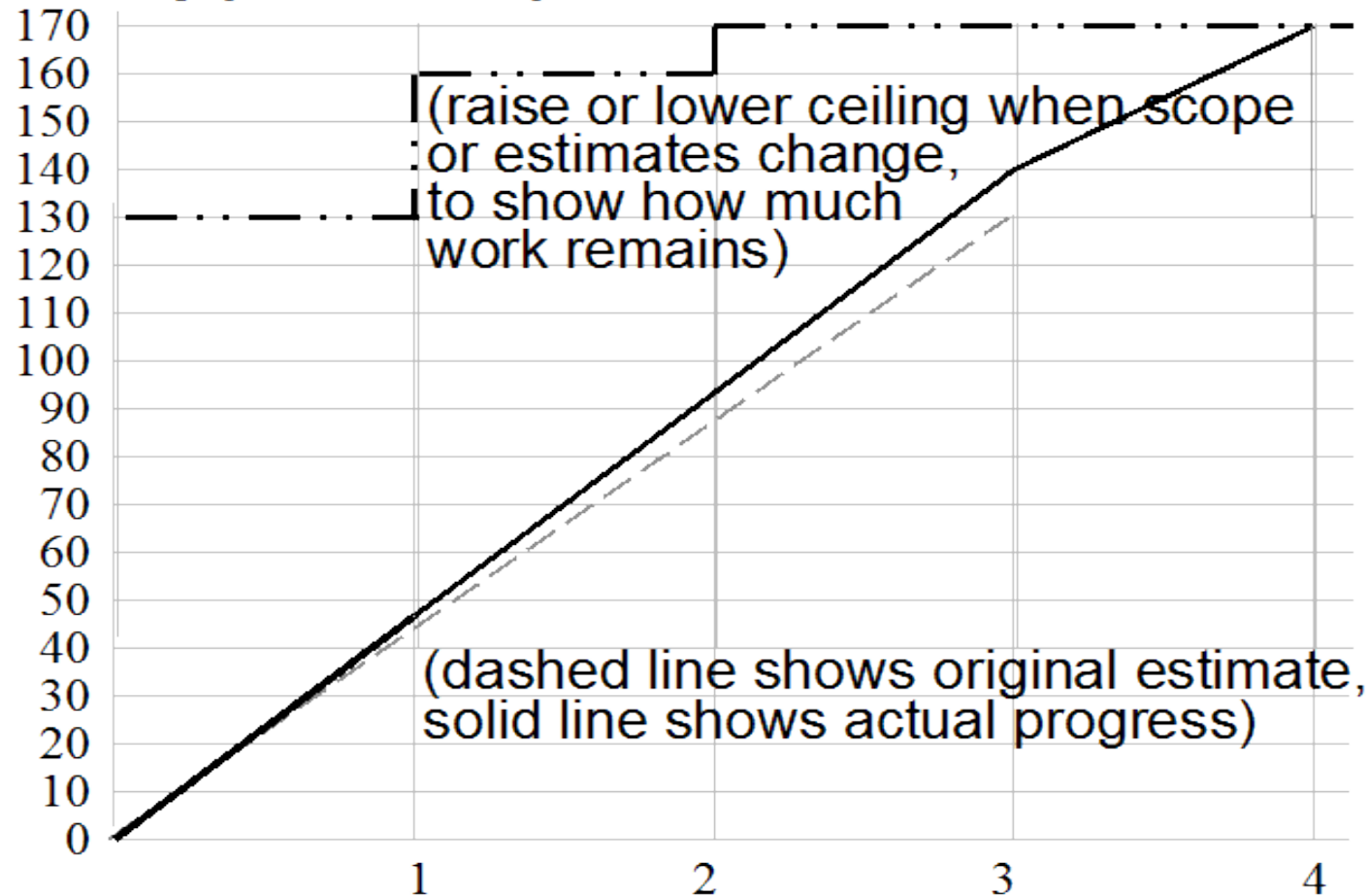
Simple “Visualization” to Track XP Progress

- Get a big pin-board into the project room, place *visible to everybody*
 - preferably next to the coffee machine, fridge etc.
- Create four clearly visible areas on pin-board:
 1. User stories *signed up* (but not yet started)
 2. User stories *currently being worked upon*
 3. User stories *completed*
 4. User stories *accepted* by the customer (or the customer representative)
- All story cards of user stories to be completed in current iteration are initially pinned to Area 1.
- Only once user story is fully implemented (and accepted by on-site customer), move corresponding card to Area 4.
- Simple, but effective way to visualize project progress
- ☞ *Note: does not work on a web-page (pull vs. push-flow)*

Do not Hesitate to Adjust Graphs!



story points completed





Hierarchical Traffic-light Method

For assessing progress:

- Identify the key (first-level) tasks that are to be completed
- Break them into smaller, more manageable tasks
- Assess each (lower-level) task by
 - ☐ Green as ‘on target’
 - ☐ Amber as ‘not on target but recoverable’
 - ☐ Red as ‘not on target and recoverable only with difficulty’
- Assess the first-level tasks based on the assessments of their lower-level tasks.
- Assess the overall progress based on all the assessments
- ☞ Note: a similar approach can be used to monitor quality.

Iceberg List



Iceberg List (cont.)



- Ordered list of (all) features that need to be completed for the project
 - Includes brief description, relative business value/priority
 - Estimated development time (ideal days)
 - Time already used for development (calendar days)
 - May also include dependencies between tasks
- Based on estimates, draw a visible line to separate tasks of current iteration (“above water”) from future iterations (“below water”).
- If changes are made to list, adjust separation line accordingly
- Use color coding (Green, Amber, Red) to highlight status of tasks.
- ☞ *Can be used both as a planning and reporting tool!*

Reporting



Progress Report:

- Indicate the work done by the personnel and the time spent on the work
- Indicate how much work still needs to be done
 - ☞ *Use suitable estimates here!*
- Optional items
 - ☐ Likelihood of failing to complete the task by the scheduled date
 - ☐ Estimated time of completion

Risk Report:

- Indicate the **likelihood** of meeting the scheduled target date
 - ☐ Instead of asking the estimated completion date
- Risk Reporting Technique
 - ☐ The traffic-light method



Important to Consider

- Report the *true* state of the project, not wishful thinking!
- Make graphs *visible* to development team
 - Webpages are not necessarily a good idea!
 - Pinboard with task-cards work quite well
- Display graphs/reports only as long as they are useful
 - once out of date, discard them
- Monitoring as a “*meta-activity*” that assists development
 - It is *not* a primary activity and should not “hinder” development.
- Not to be used as a tool to assess individual’s performance!
- If problems are detected
 - Identify root cause, not only its effect.

“Exercise for the Reader”



- What would you do to track and monitor *quality* in a software project?
- What kind of metrics/measures are useful? How can you best visualize them?