# Project Management Metrics

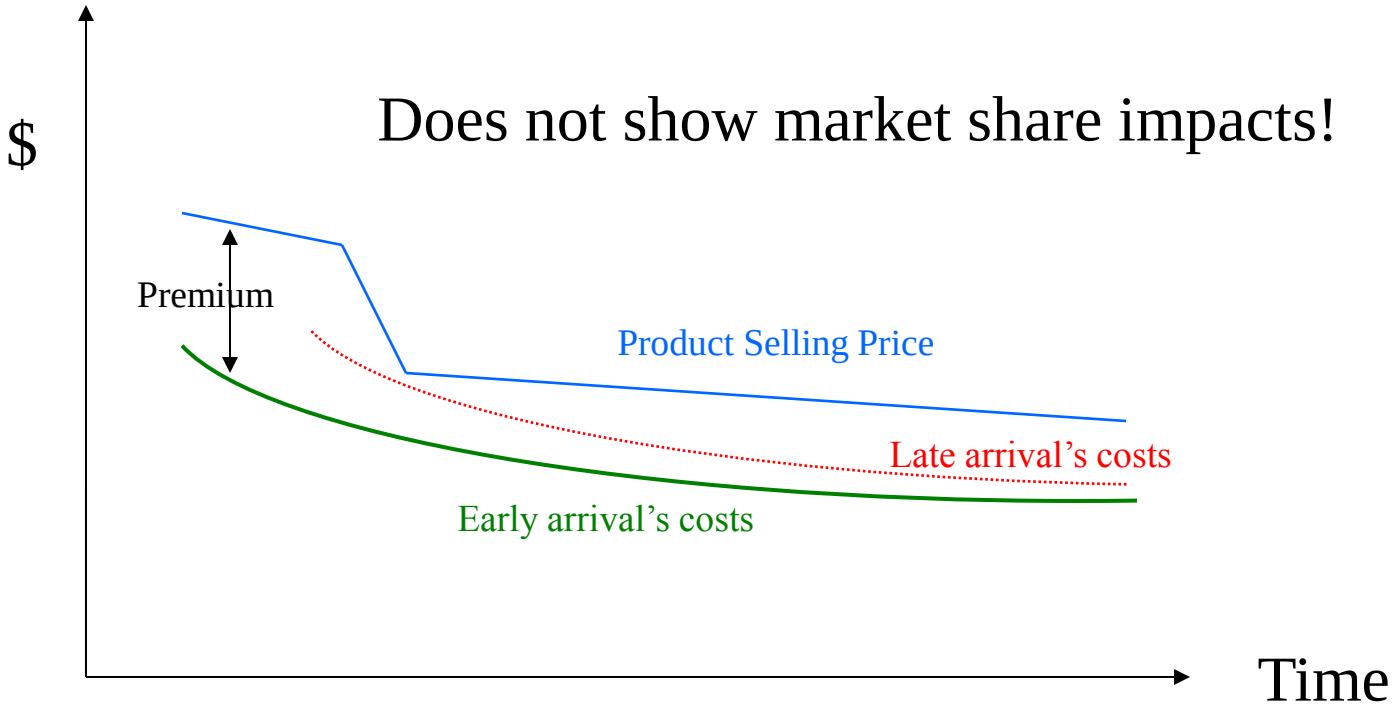
# Project Management Metrics

- Cycletime
- Productivity
- Staffing
- Requirements volatility
- Reuse metrics
- Activity progress measurement
- Estimation accuracy

# Cycletime

- Time from requirements to release (one cycle)
- Constant pressure in the corporate world to improve cycletime:
  - Improves time-to-market
    - Getting to market ahead of competition has big impact on market share, profits
  - Correlates heavily with cost
  - Reduces gap between market survey and actual release to market
- Also important for custom solutions
  - Getting deliverable earlier to customer saves them money (increases value of deliverable – shorter “time to money”)

# Impact of Time-to-Market




Products of early and late arrival both mature over time, reducing costs, but early arrival has higher maturity at any given time.

# Practices for Cycletime Reduction

- Incremental development (→ agile development)
  - Quicker release cycle makes it easier to get new features into product quickly
  - Break up 12-month cycle into 4 cycles of 4 months each! (yes, that makes sense!)
- Use of tools and technologies that improve productivity
- More concurrent engineering (increases coordination needs)
- Planning and risk management to avoid holdups
  - Rightsizing teams to minimize development cycletime
- Avoid building from scratch: use existing libraries and products where possible
  - Invest in developing libraries and domain architectures
- Streamlining development through checklists, templates, workflow, etc.

# Measuring Cycletime

- Basically simple: project start date, end date
- Project cycletime vs. development cycletime:
  - Development time: requirements-to-release
  - May expend a lot of time before requirements phase
    - Project concept, inception, etc.
- Issue: what about holdups "beyond one's control"?
  - May have a concept of "stopping cycletime clock"
  - Shows the need for proper operational definitions
  - Note the possibility of superior practices that avoid holdups
    - Measurements & metrics can impact which practices are encouraged!

# Cycletime Metrics

- Challenging to create metric for cycletime
  - Are projects really "comparable"?
    - Different features, different complexity
    - Customers may or may not be willing to pay for speed
  - Avoid encouraging "bad practices" such as unreasonably small increments
    - Release must provide "significant value" to customer
- "Bucket" concept
  - Group together "broadly similar" projects and measure
- Hard to get enough projects for statistical significance
- More important to compare with competitor cycletimes
- Focus on constant improvement

# Productivity

- Objective: Measure effectiveness of organizational practices in getting work done
  - Measuring individual productivity is not good:
    - Extremely prone to abuses, creates pressures
    - Impacts teaming, co-operation: “credit-grabbing”
    - Hard to balance with quality
    - Counter-productive in the longer term
- Metric: size of deliverable / effort expended
  - Size of deliverable ≠ volume of work (KLOC)
    - Credit for effective reuse, choosing good platforms, etc.

# Productivity Metrics

- Function-points/staff-month
  - Better than KLOC / staff-month
    - Avoids problems related to “density of code”
- Challenges in productivity comparisons:
  - Accounting for complexity (compare only with same domain)
    - But still, not all function points are created equal!
  - Assigning proper value for tools / technology / platform usage
    - “Size of deliverable” gives too much credit (what about added cost?)
    - “Actual work done” gives too little
  - Impact of other factors
    - Requirements volatility, staff profile, nature of work (fresh / legacy), tough product quality requirements, development infrastructure, time overheads …
- Interpret with extreme caution!
  - Minefield – Easy to overemphasize because it is so “bottom line”

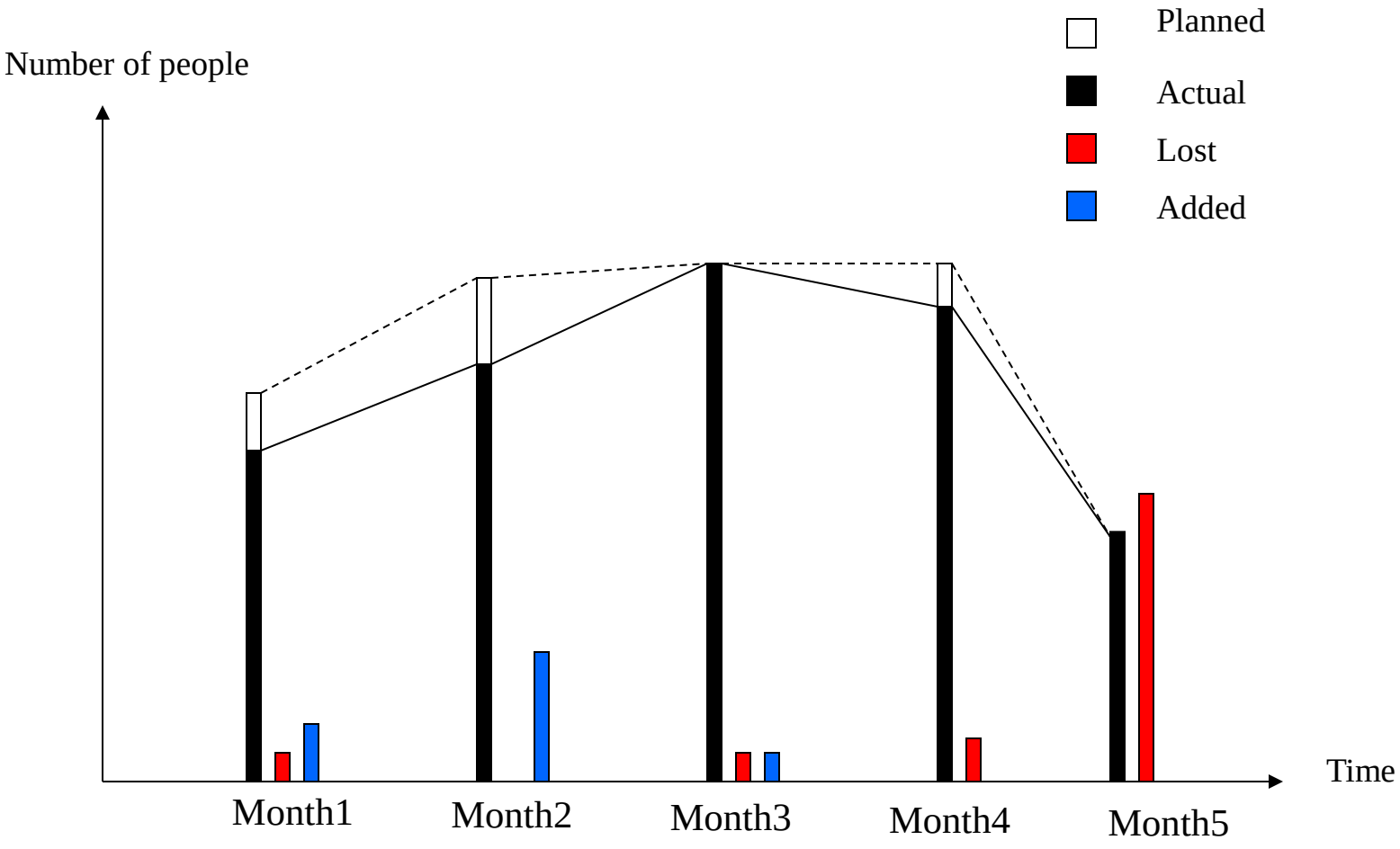
# Using Productivity Numbers

- Trend information may add value
  - Indicate whether there is constant improvement of practices
- Comparison with competitors or industry average
  - OK measure of overall effectiveness
  - Beware of differences in measurements, reporting
- Useful to evaluate technologies and practices
- Excellent complementary metric
  - Improvements in other numbers should mostly show up in productivity for example, COQ/COPQ balancing, fault injection

# Staffing

- Curves showing planned & actual staffing for each month:
  - Gaps would indicate potential schedule impacts
  - Significant increases in planned staffing must be accompanied by training/induction plans
- May include turnover rates:
  - People moving out, people added
  - High turnover will impact productivity, schedule
- Limitation: shows raw numbers, not skill level
- Metrics:
  - % staffing (actual/planned)
  - % turnover

# Staffing Chart



# Requirements Volatility

- Month-by-month percentage change in requirements
  - Based on either use cases or numbered requirements
  - Includes added/deleted/changed requirements
- High requirements volatility impacts schedule, fault injection, productivity
  - Can use "control line" e.g. 10% → requirements change more than this triggers risk mitigation (impact analysis / replanning)
- If using tools to manage requirements, relatively easy to generate requirements volatility metrics
- Limitation: does not show severity/impact of changes

# Reuse Metrics

- Percentage of reused code
  - Hard to define how much to count as reused code:
    - "Scavenged code" (cut-paste) is least valuable
    - Libraries better – should we give full credit for each use?
    - Using COTS (commercial-off-the-shelf) software better, for example, don't write your own OS or GUI framework – how do you count this?
    - Domain engineering – creating standard product architectures and avoiding developing a fresh from-scratch best – should we give full credit for each use?
- Common practice:
  - Measure libraries and/or scavenged code
  - Can add notes about use of COTS and/or domain architectures and components
- Note that the end goal is productivity, not reuse

# Progress

- Objective: Measure progress against plan
  - Avoid situation where lateness is realized just prior to release
- Practices:
  - Define milestones 2-3 weeks apart
  - Measure planned vs. actual completion dates
  - If two weeks or more behind schedule, replan
    - Re-negotiate fresh delivery dates with customer
- Metric:
  - Chart of planned and actual completion dates
  - Percentage slippage: (actual – planned ) / planned completion time

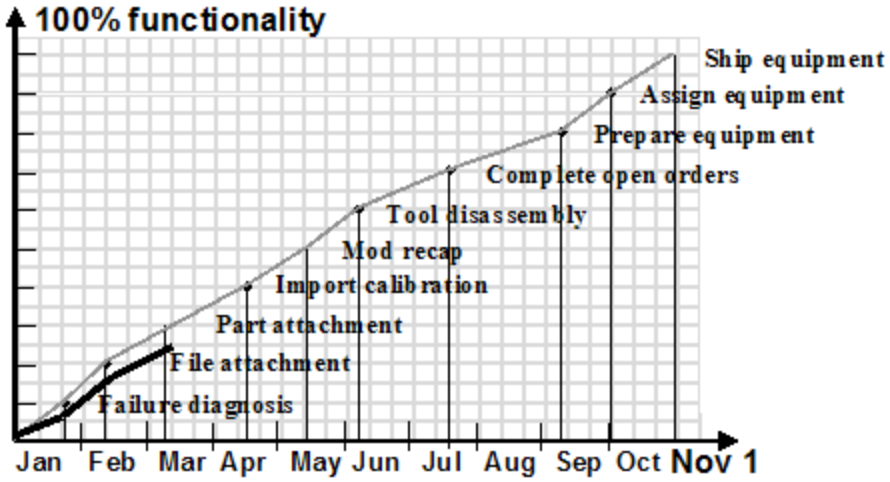
# Progress: Milestone chart

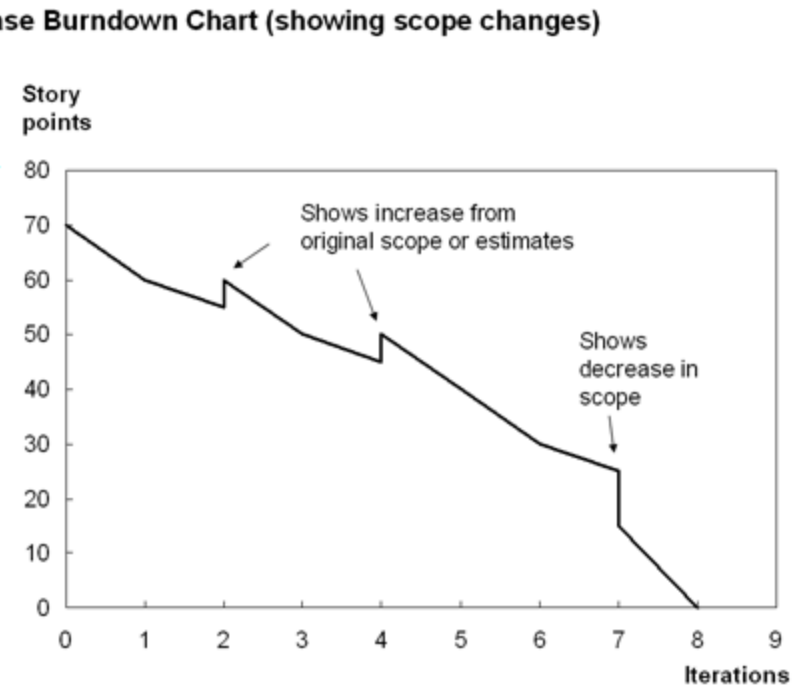
| Milestone | Planned | Actual |
|---|---|---|
| Initial requirements | 14-Mar | 13-Mar |
| Prototype | 4-Apr | 6-Apr |
| Requirements baselined | 12-Apr | 12-Apr |
| Initial design | 23-Apr | 28-Apr |
| V1 code complete | 8-May | 24-May (replan) |
| Integration done | ~~12-May~~ 28-May | 29-May |
| Release 1 | ~~1-June~~ 12-June | |

# Progress: Earned Value Charts

- A superior way to measure progress
  - Focuses on value delivered instead of effort spent
- For each activity, define an "earned value" – some number of points
  - Assign more earned value if more effort needed
- Track actual earned value:
  - Total points earned for all *completed* activities
  - Irrespective of actual effort expended
  - May add another curve that shows actual effort expended
- Plot planned vs. actual earned value against time
  - Shows % completion of project very clearly

# Burn-Up and Burn-Down Charts



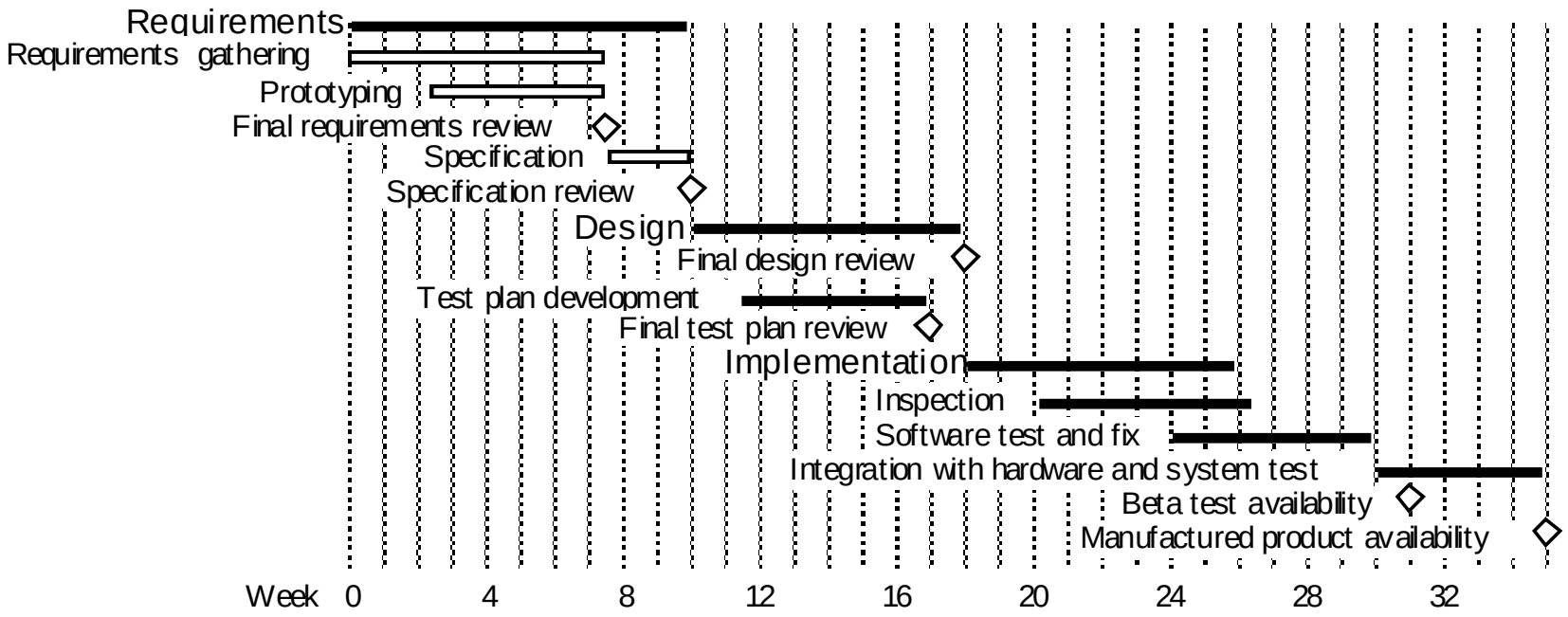




Burn-Up (Earned Value)

Release Burn-Down

Alistair Cockburn (http://alistair.cockburn.us/Earned-value+and+burn+charts)

# Gantt Charts




From Lethbridge & Laganiere, "Object-oriented software engineering"

# Estimation Accuracy

- (Actual effort – Estimated effort ) / Estimated effort
- Typically around 20% (that is, 20% underestimate) for "good" organizations
  - Note that you expect to not estimate 20% of the required work
  - Note that this often doesn't translate to 20% slippage – either replanning or overtime work
  - If maturity is low, maybe 50% - 100% or even more
- Can track estimation accuracy for initial estimates as well as for "final" estimates
  - Initial estimates may be prior to understanding requirements or identifying technical risks
  - Correlate with requirements volatility to get better picture
- Limitation: "Work expands to fill time available"
  - Hard to detect overestimates

# Summary

- Can track a variety of metrics that reflect various project management concerns
- Used to detect likelihood of various problems:
  - Slippage, productivity loss, need for training
- Correlate multiple curves to assess health of project
  - Typically all these curves on one big chart – Management Dashboard
- Each metric vulnerable to abuse
  - Need to be careful how we use them!