



# *lecture 20*

## *capacity constraint*

csc302h  
winter 2014



## *recap from last time*

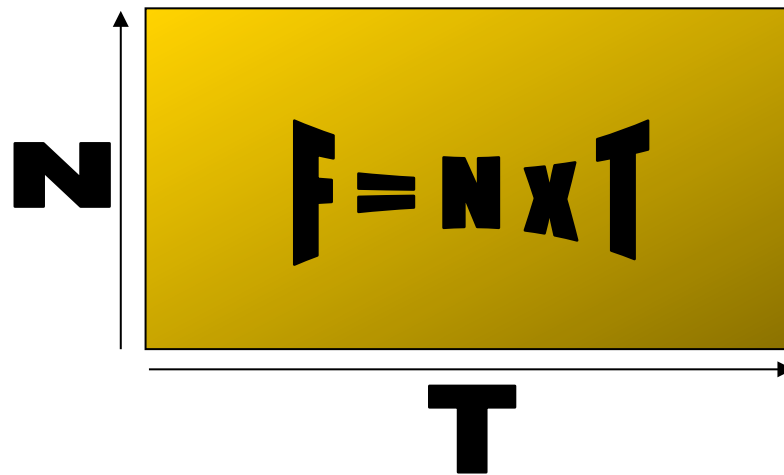
- requirements (or features =  $F$ )
  - prioritized potential requirements from wish list
    - can even do full cost / benefit analysis
  - estimate a size for each (planning poker) in ideal days (ECDs)
- calculate available resources ( $N$ )
  - pick a value for  $T$  (workdays until release date, end of sprint, horizon, ...)
  - for each developer, determine availability, vacation, and then multiply by  $w$

## *recap from last time (2)*

$$F \leq N \times T$$

- plan must respect the capacity constraint
- keep plan up to date with most current estimates at all times
- dealing with overflow
  - move dates
  - cut features
  - combination
  - adding developers is rarely helps the current plan

- at the end of a release (sprint, horizon, ...) the following relationship must hold:

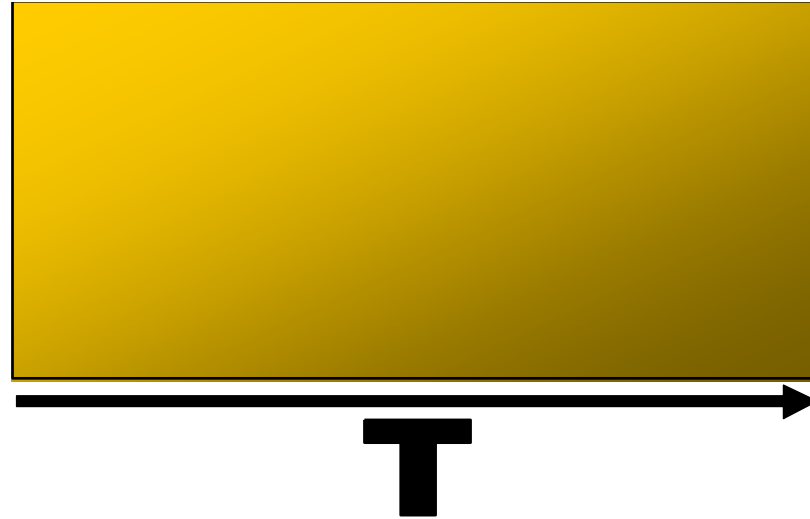


A diagram illustrating the equation  $F = N \times T$ . The equation is written in large, bold, black letters inside a yellow rectangular box. To the left of the box is a vertical double-headed arrow labeled with a bold black 'N'. Below the box is a horizontal double-headed arrow labeled with a bold black 'T'.

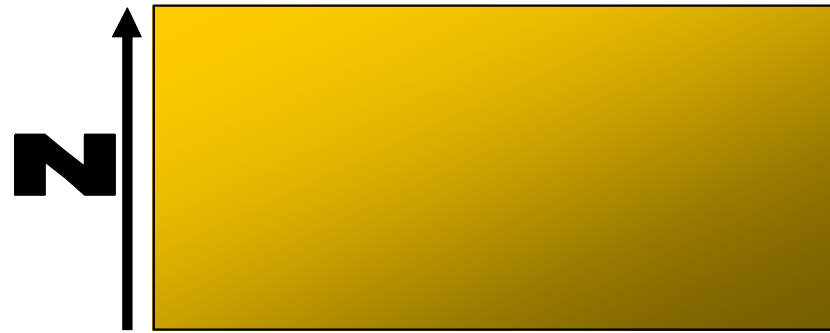
- today we will more rigorously define this
  - to know what we are trying to estimate
  - how to compare actuals to estimates (post-mortem)



## *number of work days: $T$*



- number of full working days for development, subtracting
  - weekends
  - statutory holidays
  - “company days”
  - anything we know ahead of time



- the average number of dedicated developers per work day during the  $T$ -day period
  - subtract vacation
  - multiply individual work factors:  $w_i$
  - also defines daily burn-down rate



## *work time vs. dedicated time*

- work time
  - defined as 8 hours per work day
- dedicated time
  - uninterrupted hour equivalents
  - time dedicated to adding new features to release or (unit testing)
- uninterrupted time
  - 8 hours with 1 hour of constant interruptions is not 7 hours...more like 4

- maintenance on previous (still supported) releases
- leadership duties
- meetings
- training
- unexpected days off (ex. illness)
- sales/marketing support
- loss of “zone time” due to interruptions
- work on other projects (availability)





## *measuring N (post-facto)*

$$N = \frac{\sum_{i=1}^n h_i}{8 \cdot T}$$

- assume all developers understand what dedicated, uninterrupted hours are
- $n$  is the number of developers
- $h_i$  is the total number of hours logged by developer  $i$  on all features in the release (best read from time tracking system)



## *attributing N*

$$t_i = d_i - v_i \quad w_i = \frac{h_i}{8 \cdot t_i} \quad N = \frac{\sum_{i=1}^n t_i \cdot w_i}{T}$$

- $d_i$  is the number of days developer  $i$  is available during the development phase (sprint, horizon, ...)
- $v_i$  is the number of vacation days developer  $i$  took during the development phase

substitute to get back to:

$$N = \frac{\sum_{i=1}^n h_i}{8 \cdot T}$$



$$T = 39$$

$$d_{bob} = 35$$

$$v_{bob} = 5$$

$$t_{bob} = d_{bob} - v_{bob} = 35 - 5 = 30$$

$$h_{bob} = 120$$

$$w_{bob} = \frac{h_{bob}}{8 \cdot t_{bob}} = \frac{120}{8 \cdot 30} = 0.5$$

- Bob called in sick 2 days
  - accounted for in  $h_{bob}$
- Bob took an afternoon off, but worked on the weekend to make up for it
  - Accounted for in  $h_{bob}$



# Features

$$F = \sum_{k=1}^K f_k$$

- $f_k$  is the number of dedicated days (dedicated hours  $\div$  8) it took to develop the  $k^{\text{th}}$  feature



- need a time tracking system capable of tracking  $h_{i,k,d}$ :
  - dedicated (uninterrupted) hours spent
    - by the  $i^{\text{th}}$  developer (out of all  $n$  developers)
    - on the  $d^{\text{th}}$  day
    - working on the  $k^{\text{th}}$  feature
- each such “quantum” would appear on either side of  $F = N \times T$  constraining them to be equal
- see section 5.10 of the Penny book for proof