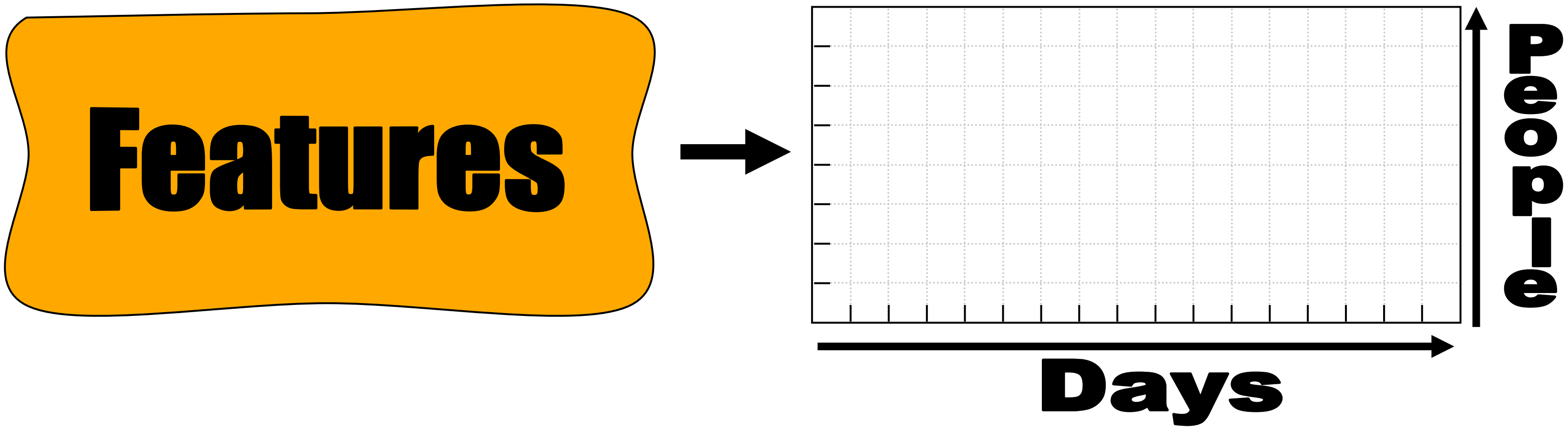
capacity constraint

capacity constraint

- fundamental constraint governing all planning activity
- geometric analogy:



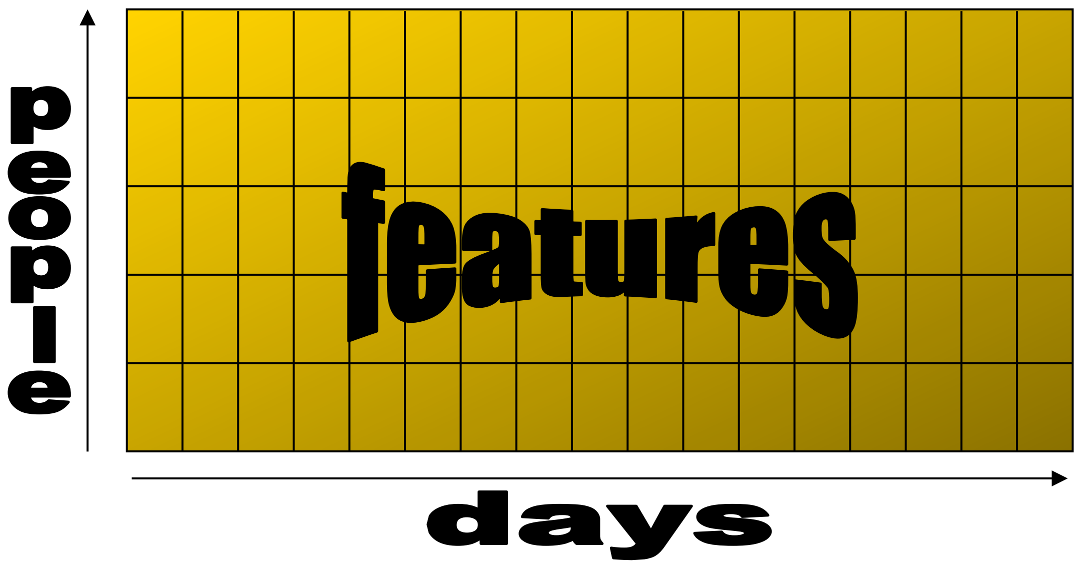
requirement

capacity

capacity constraint (2)

- fundamental constraint governing all planning activity

it’s all gotta fit!



simple release plan

Dates:	Coding phase:	Jul.1—Oct.1
	Beta availability:	Nov.1
	General availability:	Dec.1
Capacity:		*days available*
	Fred	**31** ecd
	Lorna	**33** ecd
	…	…
	Bill	**21** ecd
	total	***317 ecd***
Requirement:		*days required*
	AR report	**14** ecd
	Dialog re-design	**22** ecd
	…	…
	Thread support	**87** ecd
	total	***317 ecd***
Status:	*Capacity:*	**317 effective coder-days**
	Requirement:	**317 effective coder-days**
	Delta:	**0 effective coder days**

release planning

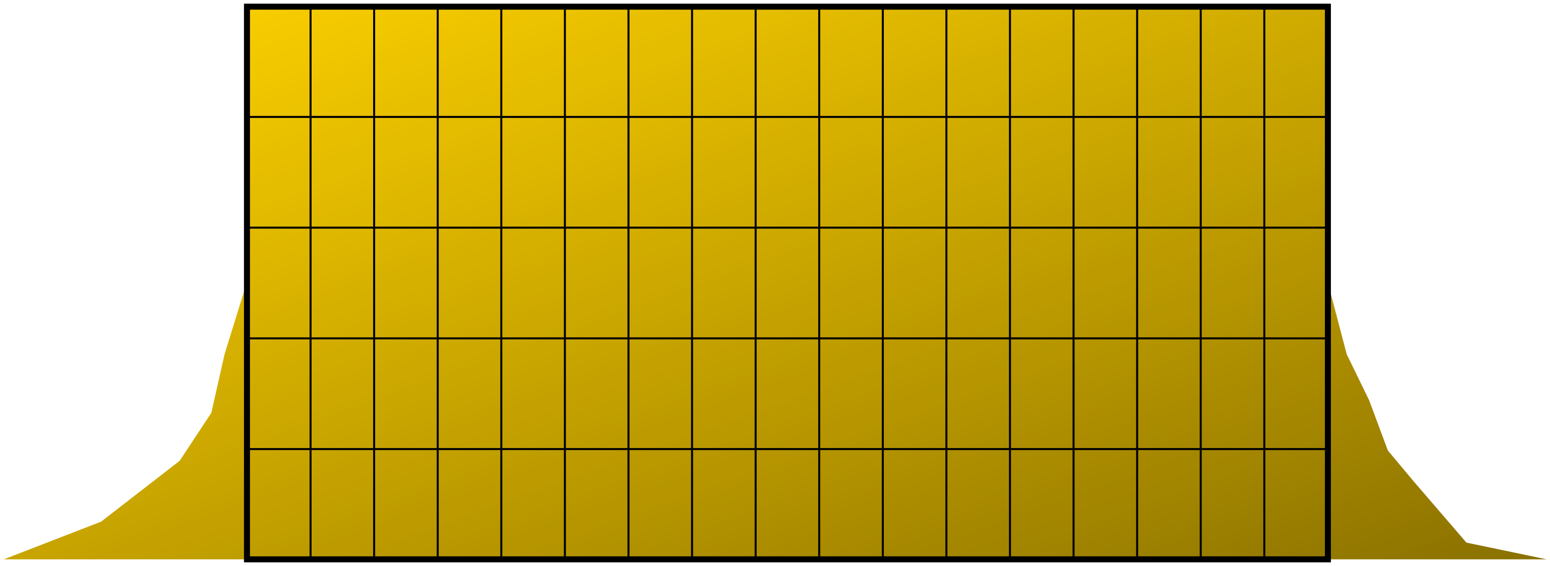
- what to build: F
- by when to build it: T $F \leq N \times T$
- using how many people: N

- need to build an initial plan that respects the capacity constraint
- need to continuously update the plan to maintain its adherence to the capacity constraint.

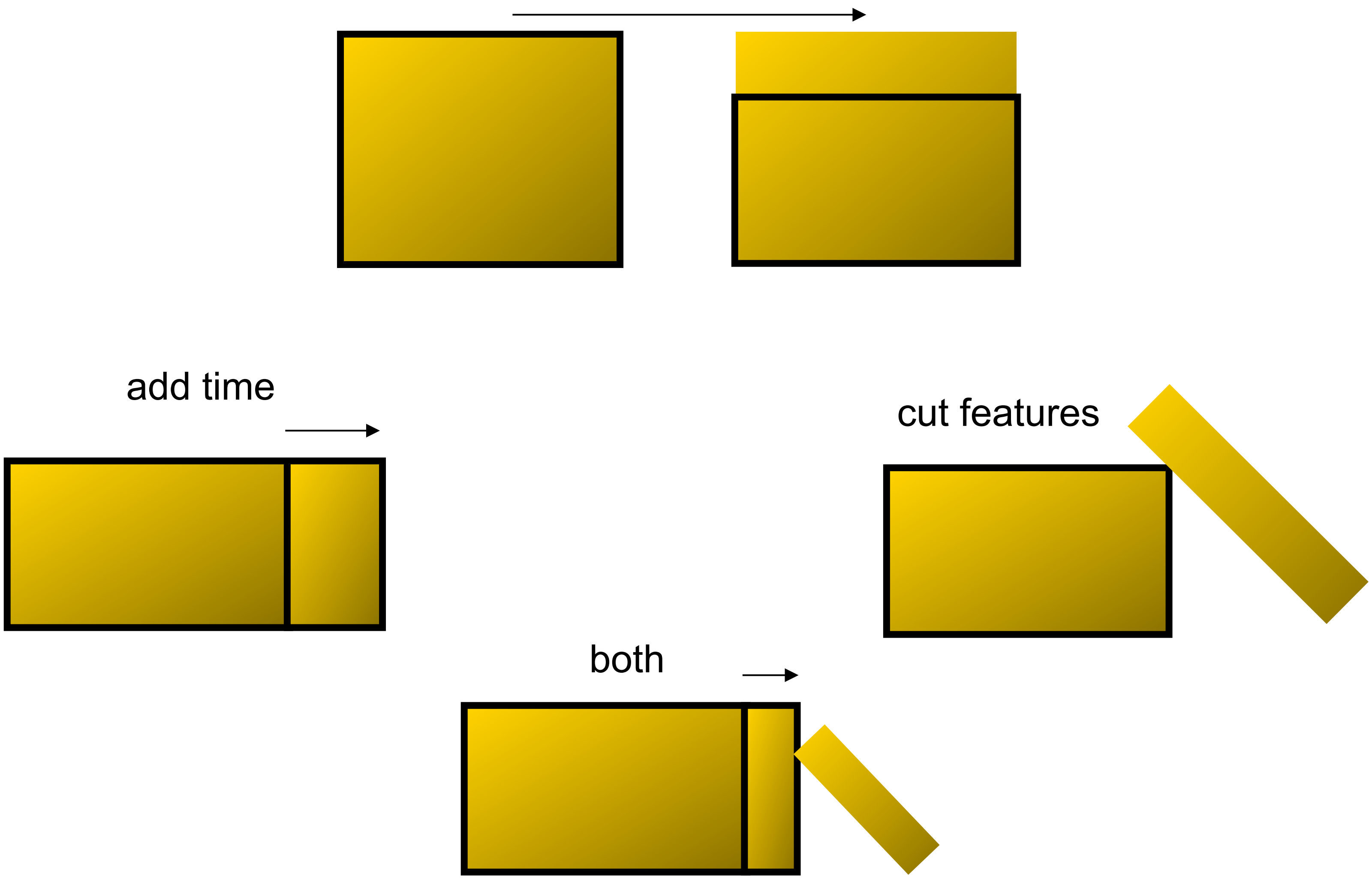
most common problem

- comes from either:
  - not knowing
  - knowing but hoping for the best (Yourdon's *Death March*)

(can happen initially, or as we go)



dealing with issues



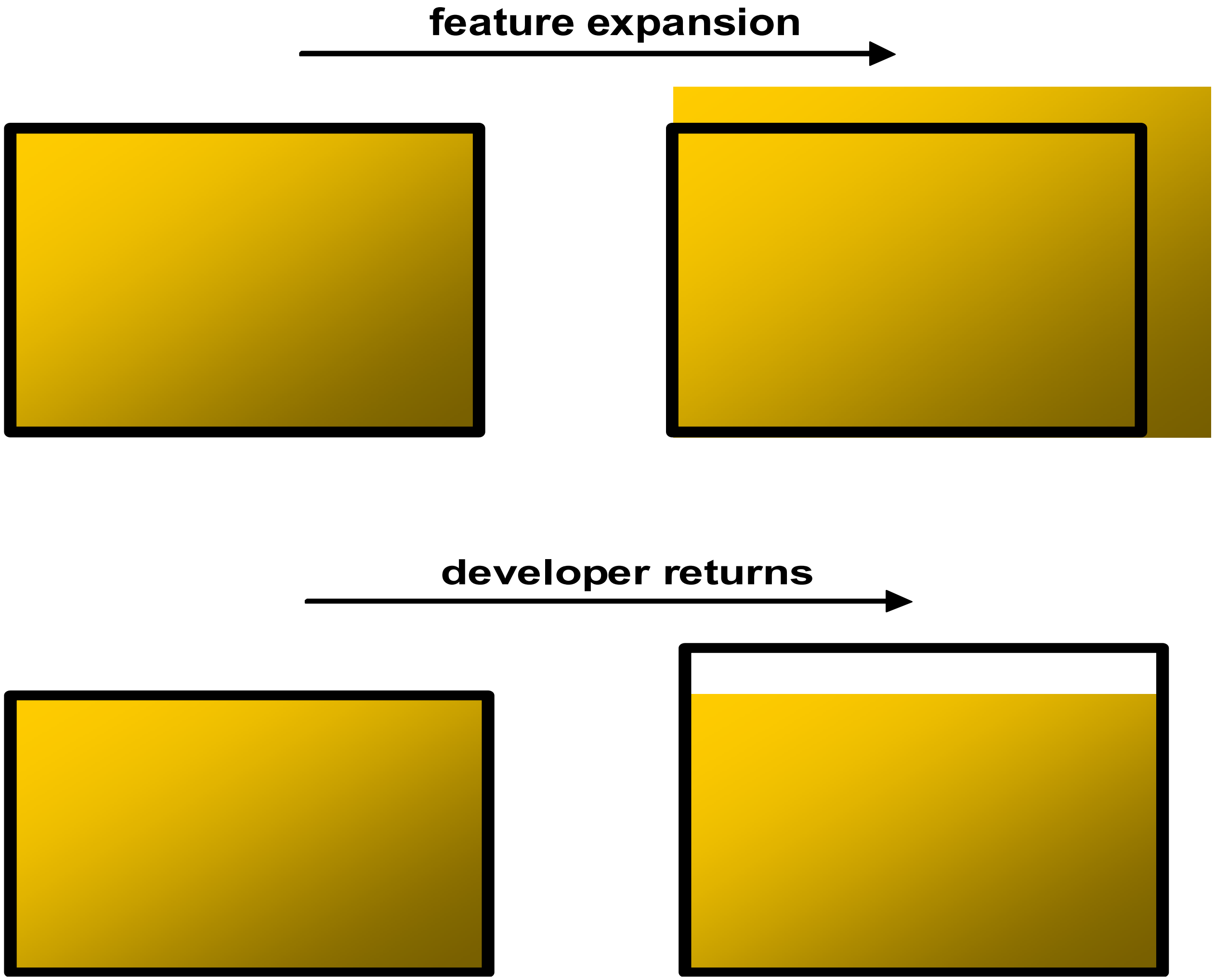
developer leaves the team

add time

cut features

both

other issues

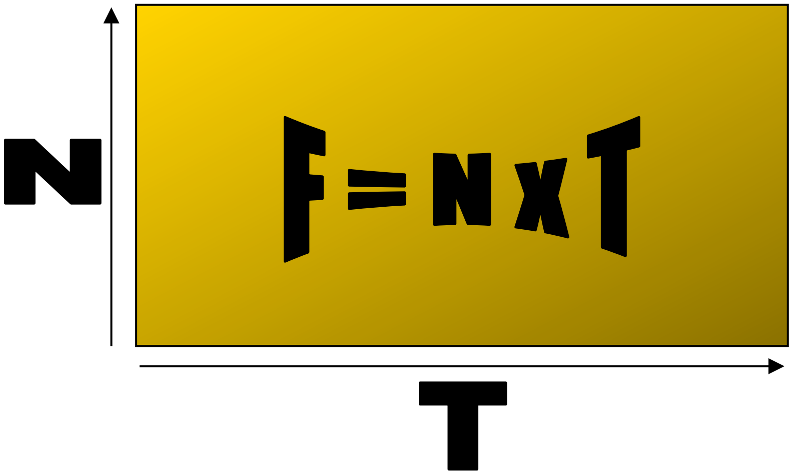


organizational issues

- management must appreciate that software development carries with it certain inherent risks
- the business of a software organization is to manage and adapt as possibilities continuously become reality
- ranting and raving is unproductive
- with good data, good managers will make good decisions

quantitative capacity constraint

- post-facto, the following relationship ***must*** hold:
  (but, it requires careful definition)

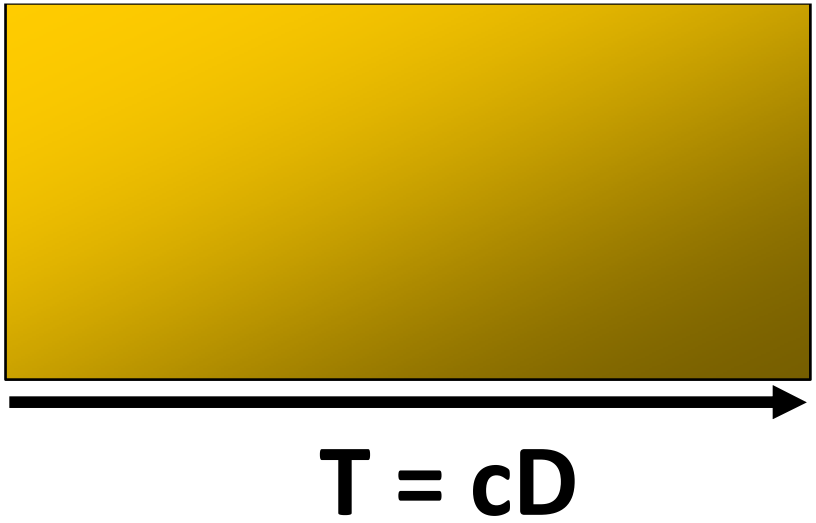


we define carefully so that we know what it is we are trying to estimate, and how to compare actuals against estimates for post-mortem

T: number of workdays

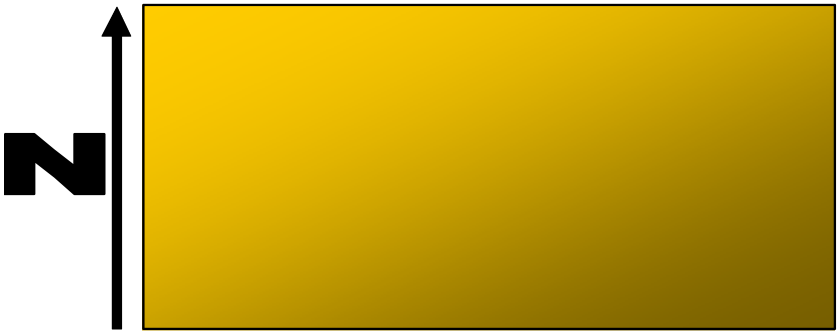
- the number of full-equivalent working days from fork to dcut.
- subtracts
  - weekends
  - statutory holidays
  - “company days”
- subtracts anything we know in advance that nobody is expected to work.

T = cD: for SaaS



- D = full working days in planning horizon
- c = factor to convert to predominantly coding days

N: developer power



- the average number of dedicated developers per workday working during the T-day period.
- dedicated developer?

work time vs. dedicated time

- work time or body time
  - defined as 8 hours per workday
    - excludes weekends, stat. holidays, vacation entitlement.
    - e.g., 9-to-6 with 1 hour for lunch.
- dedicated time
  - uninterrupted hour equivalents.
  - time dedicated to adding new features to the release.
- uninterrupted time
  - 4 hrs with 30 min. of constant interruptions
    - not 3.5 hrs of dedicated uninterrupted time – more like 2
  - 2 hrs with NO interruptions at all

dedicated "losses"

- maintenance (tracking down and fixing defects) on previous releases
- other simultaneous projects
- team-leader duties (& helping others)
- meetings
- training
- unexpected, non-made-up days off (e.g., sick days)
- sales/marketing support
- loss of flow due to interruptions

measuring N

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

- assume each developer understands the concept of a dedicated uninterrupted hour.
- get each of the ***n*** developers to record how many dedicated uninterrupted hours they spent in total during the coding phase.
- $\boldsymbol{h_i}$ is what's in the time tracking system for the ***i******th*** developer.

attributing N

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

- ***d***

Substitute to get back to:

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

example

$$T = 39$$

$$d_{bob} = 35$$

$$v_{bob} = 5$$

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

$$h_{bob} = 120$$

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

- Bob called in sick for 2 days: accounted for in ***h***
- Bob took an afternoon off, but worked on the weekend to make up for it: accounted for in ***h***

features

Features

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

$\boldsymbol{f_k}$ = dedicated hours / 8 it took to code the $\boldsymbol{k}^{th}$ feature

post-mortem

- imagine a time-tracking system that tracks:
  - $h_{i,k,d}$ = dedicated (uninterrupted) hours spent
    - by the ***i****th* developer
    - on the ***d****th* day
    - doing coding work on the ***k****th* feature
- each such quantum would appear on both sides of F= N x T constraining them to be equal.
- see section 5.10 in book for proof.