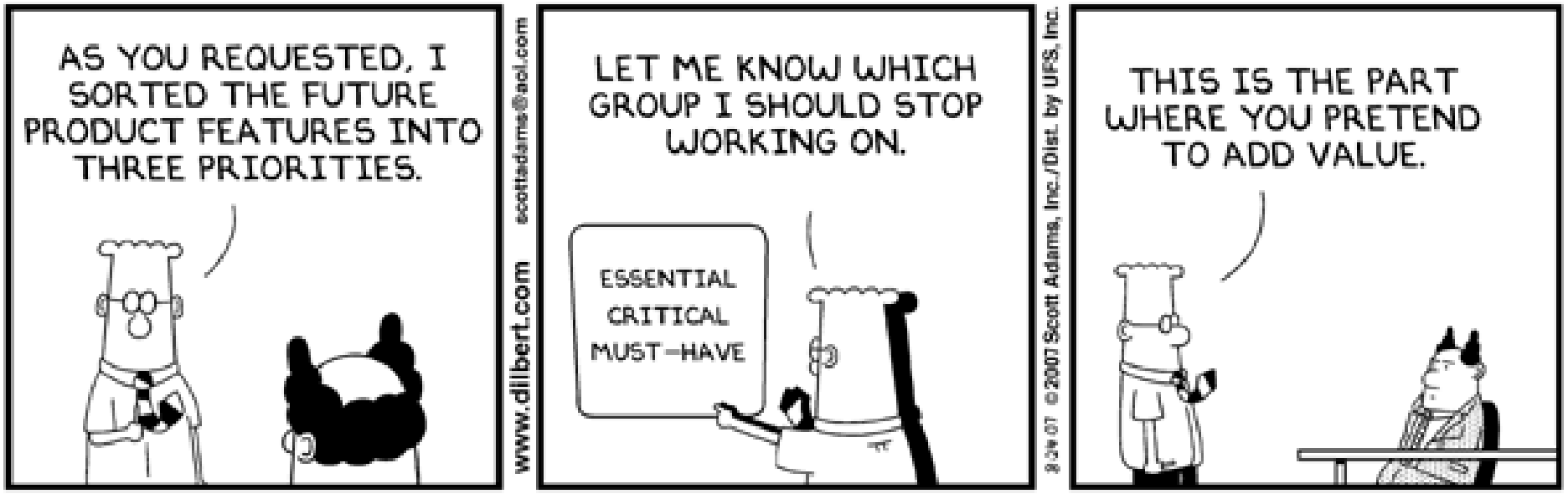


Release Planning & Buffered MoSCoW Rules

Dr. Eduardo Miranda
Institute for Software Research

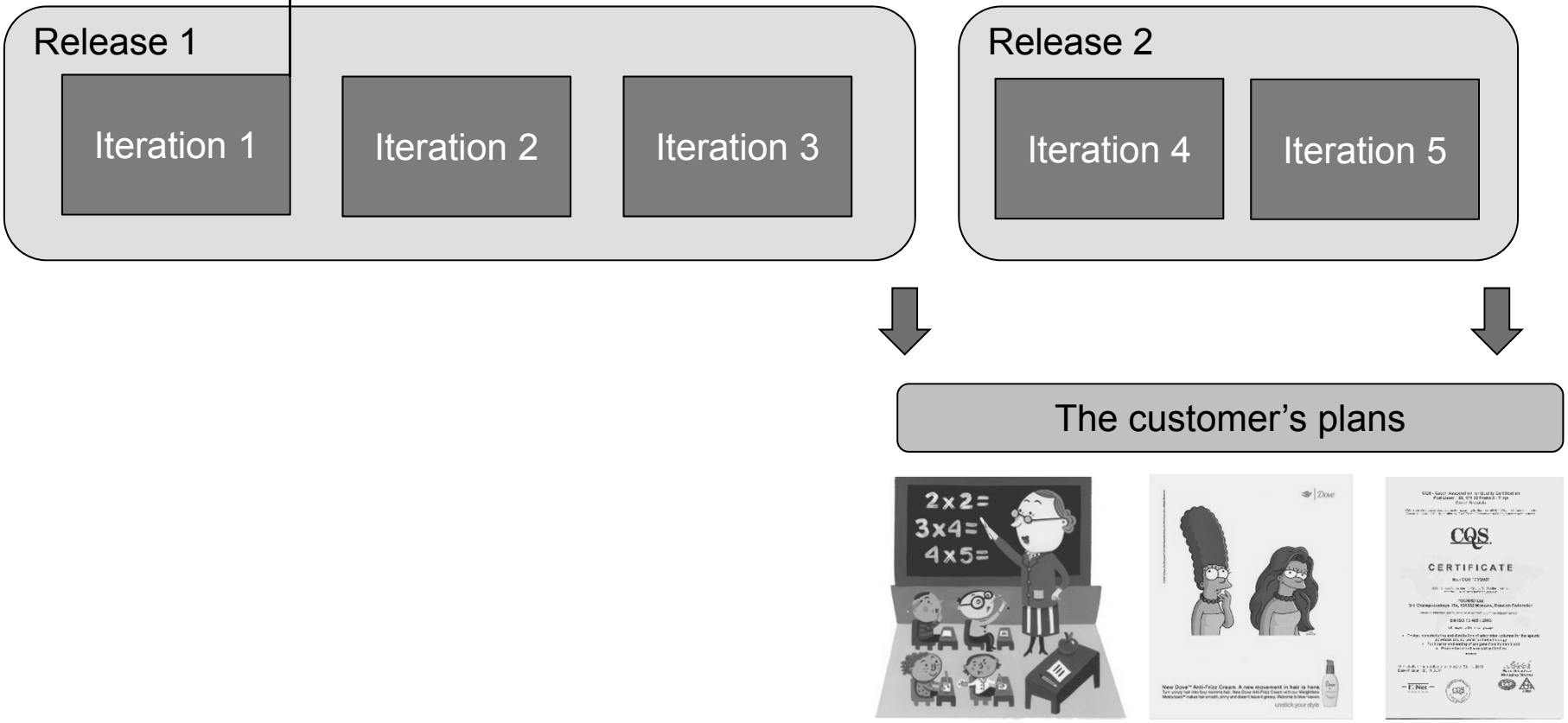
ASSE 2013 - 14th Argentine Symposium on Software Engineering /
42 JAIIO (Argentine Conference on Informatics)
September 16th, 2013, Cordoba, Argentina

Carnegie Mellon University

Gracias

Iterations vs. releases

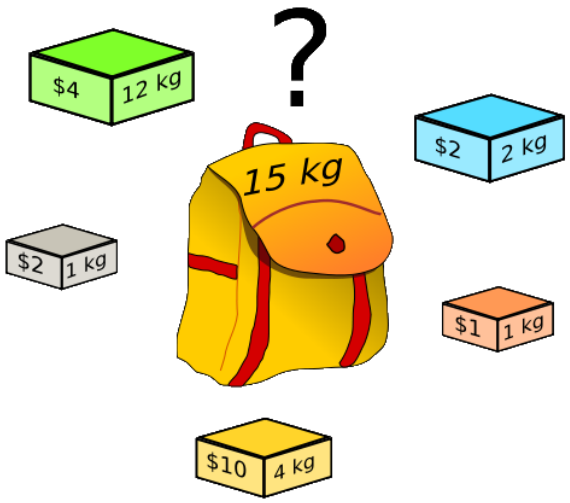
- Pacing
- Activity planning
- Feedback
- Sense of accomplishment



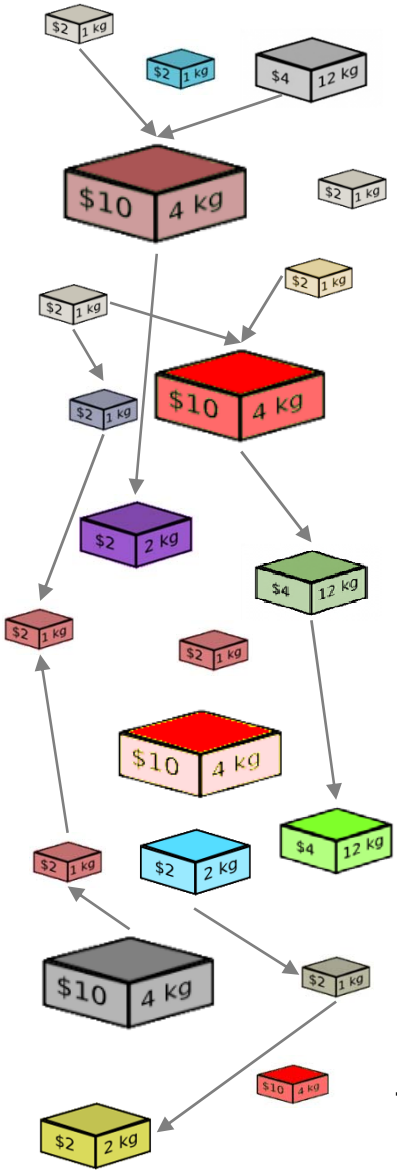
The release planning problem

- Release planning is the process of deciding when and with what features a working version of a software product will be made available to its customers
- The process must take into consideration business objectives, technical and functional dependencies and resource constrains

Release planning and the knapsack problem



The knapsack problem



The release planning problem

Release planning methods (1)

- Must Have, Should Have, Could Have, Won't have (MoSCoW) Method; D. Clegg 1994 & DSDM Consortium
  - Unstated customer preference & delivery certainty
  - Assumes three releases within a predefined time box
  - Manual method, qualitative
- Statistically Planned Incremental Delivery (SPID); E. Miranda, 2002
  - Unstated customer preference & delivery certainty
  - Assumes a defined number of releases within a predefined time box
  - Manual. Uses subjective probabilities and statistical concepts to guarantee the delivery of the most important features consistent with the basic assessments
- Evolve, D. Greer & G. Ruhe, 2003
  - Maximizes an objective function defined in terms of benefits & penalties
  - Inputs: Features' value, criticality and required effort; stakeholders weight, effort constraints for release
  - Undefined number of releases. Genetic algorithm makes recommendation of *k* best solutions for the immediate release, customer choose and iterates

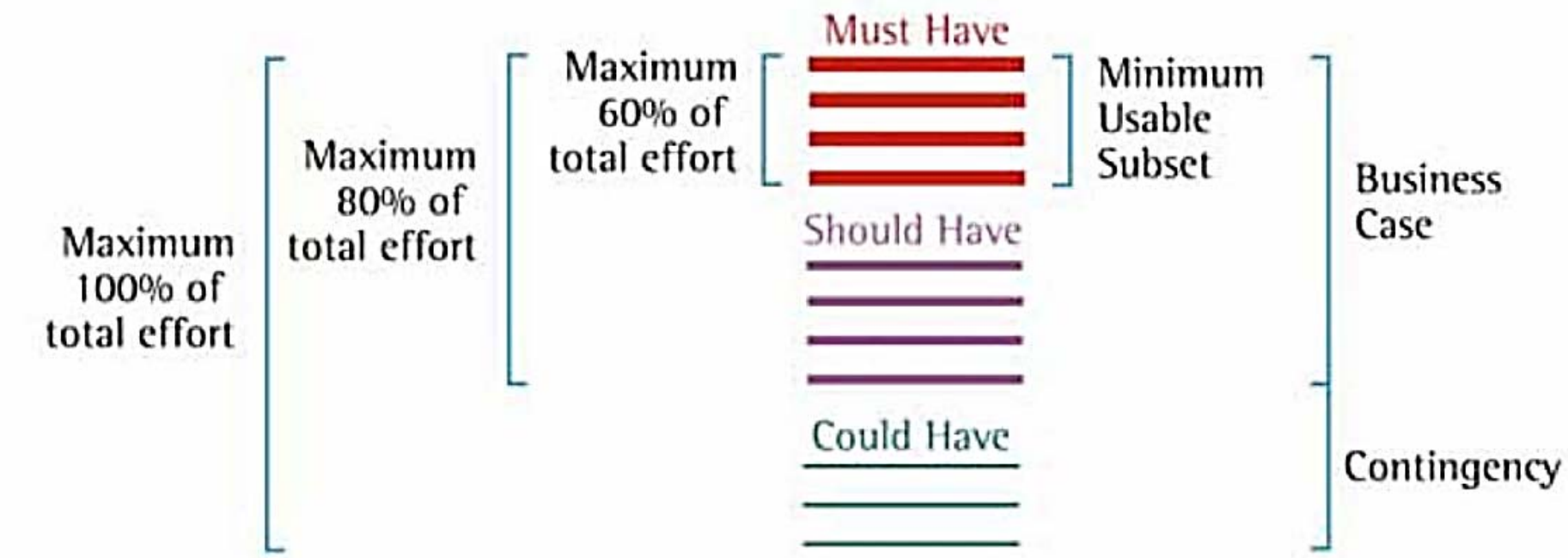
Release planning methods (2)

- Incremental Funding Method (IFM); M. Denne & J. Cleland Huang; 2004
  - Software is divided into Minimal Marketable Features (MMFs) and Architectural Elements (AE)
  - Maximizes the "Weighted Sequence Adjusted Net Present Value", a combination of cash flow and NPV, of MMFs. Indefinite number of release periods. Does not address resource constraints. The basic method assumes one MMF per period. Users wanting to do develop more than one MMF per period must accommodate them manually
  - Can be implemented using spreadsheets, the authors reference one GA implementation
- Buffered MoSCoW rules; E. Miranda; 2011
  - Unstated customer preference & delivery certainty
  - Assumes three releases
  - A simpler version of the SPID method suitable for agile teams do not wanting to deal with subjective probabilities and more complicated quantitative techniques

Prioritizing

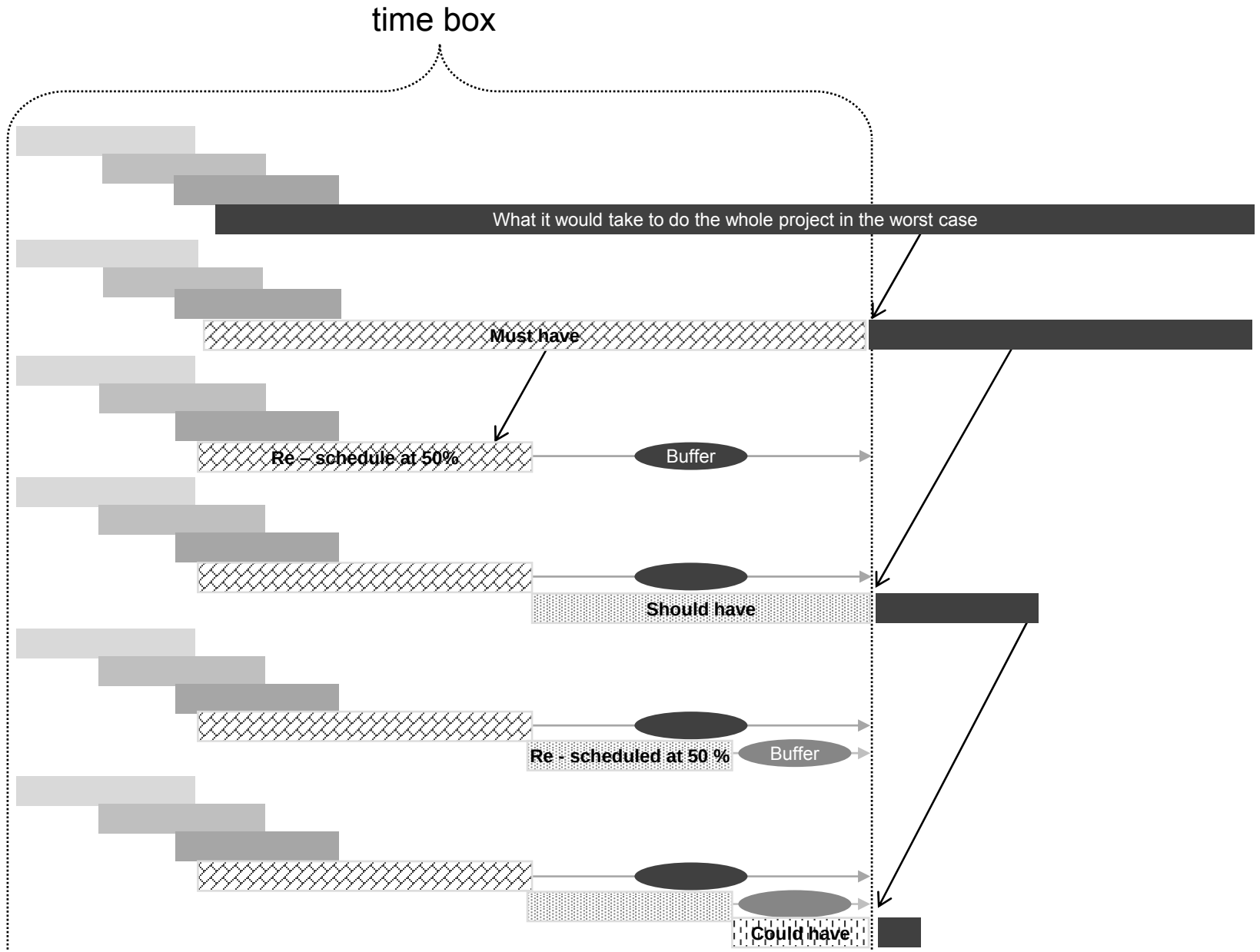
MoSCoW Rules	DSDM	Buffered MoSCoW Rules
Must have	Fundamental to project success. Without any of these the project should be considered a failure	Those features that the project, short of a calamity, would be able to deliver within the defined time box
Should have	Important but not vital. May be painful to leave them out but the solution still viable	Those features that have a fair chance of being delivered within the defined time box
Could have	Nice to have. Will enhance a solution but will not undermine basic functionality	Those features that the project could deliver within the defined time box if everything went as expected, i.e. if there were no hiccups in the development of features assigned higher priority
Wont' have	There is not enough budget to develop them	

DSDM buffering scheme

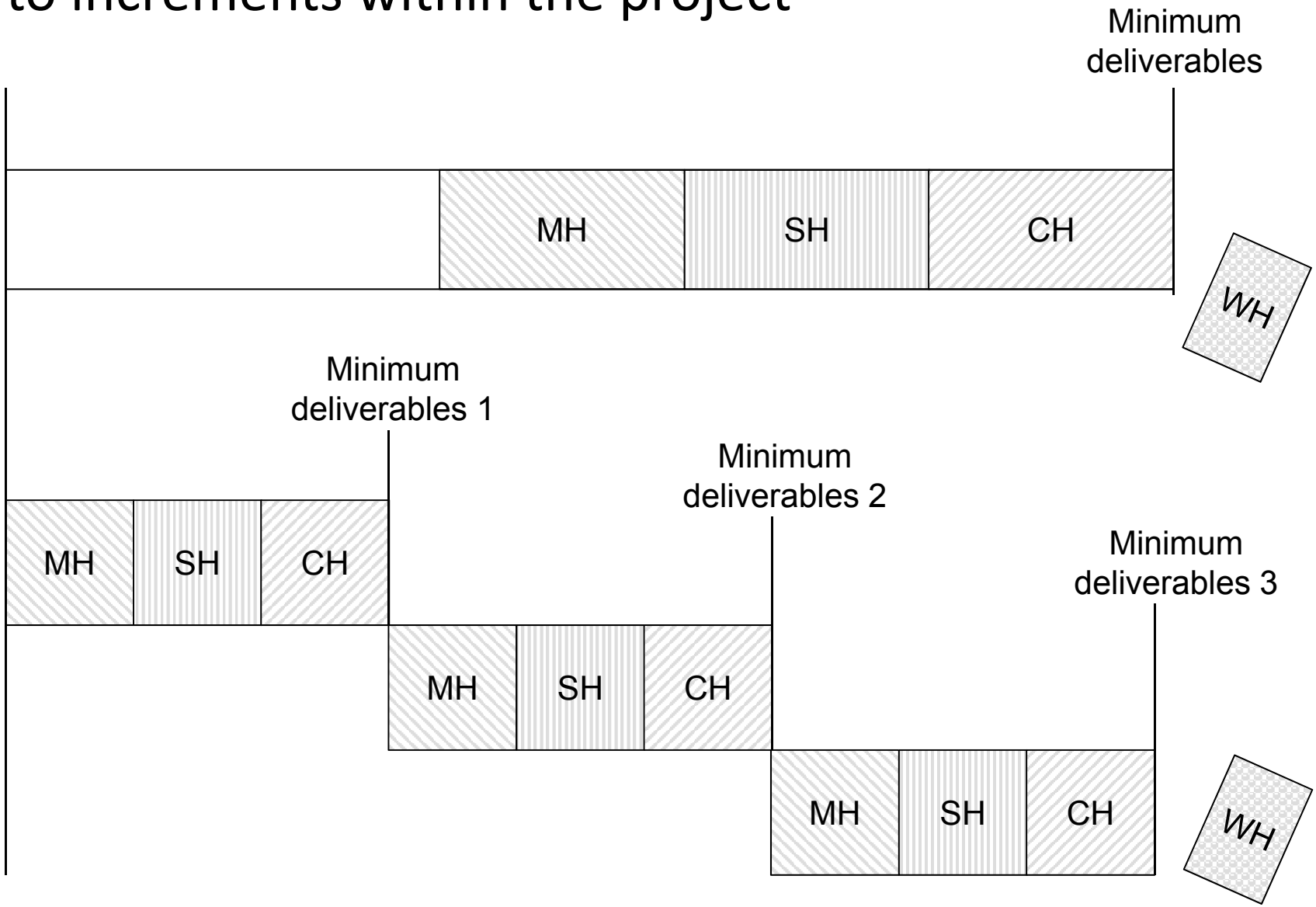


DSDM Atern Handbook, DSDM Consortium, Accessed 9/2/2013

Buffered MoSCoW Rules



Buffered MoSCoW rules can be applied to the project or to increments within the project



Numerical example (1)

Feature	Nominal Estimate	Worst Case Estimate	Dependencies
A	20	40	B, C
B	7	9	
C	20	30	
D	5	7	E
F	5	6	
G	20	40	
H	10	20	J, K
I	15	30	
J	12	15	
K	8	10	
E	6	7	
L	10	18	

- Total budget (time box) = 180hrs
- Project management, analysis & design = 60hrs
- Development budget = 120hrs
- Find the:
  - “Must Have” set
  - “Should Have” set
  - “Could Have” set
  - “Won’t Have” set

Numerical example (2): Must have

Feature	Nominal Estimate	Worst Case Estimate	Dependencies
A	20	40	B, C
B	7	9	
C	20	30	
D	5	7	E
F	5	6	
G	20	40	
H	10	20	J, K
I	15	30	
J	12	15	
K	8	10	
E	6	7	
L	10	18	

1. Select, according to its business value, as many features as you can fit into the development budget (120hrs) using the Worst Case Estimates. Assume that the preferred order is: F, D, A, G, K, E, L, J, H, I, B, C

1) Select F	120	-	6	= 114
2) Select D, E	114	-	14	= 100
3) Select A, B, C	100	-	79	= 21
4) Select G	21	-	40	= -19
5) Select K	21	-	10	= 11

2. Schedule the MUST HAVE increment using the Nominal Estimates

F + D + E + A + B + C + K = 5 + 5 + 6 + 20 + 7 + 20 + 8 = 71	120	-	71	= 49

Numerical example (3): Should have and could have

Feature	Nominal Estimate	Worst Case Estimate	Dependencies
A	20	40	B, C
B	7	9	
C	20	30	
D	5	7	E
F	5	6	
G	20	40	
H	10	20	J, K
I	15	30	
J	12	15	
K	8	10	
E	6	7	
L	10	18	

3. Select as much as you can fit in the remaining budget using the worst case estimates . Assume that the preferred order is: F, D, A, G, K, E, L, J, H, I, B, C

1) Select G 49 - 40 = 9

4. Schedule the SHOULD HAVE increment using the nominal estimates

G 49 - 20 = 29

5. Select as much as you can fit in the remaining budget using the worst case estimates

2) Select L 29 - 18 = 11

6. Schedule the COULD HAVE increment using the nominal estimates

L 29 - 10 = 19

Numerical example (4): Won't have

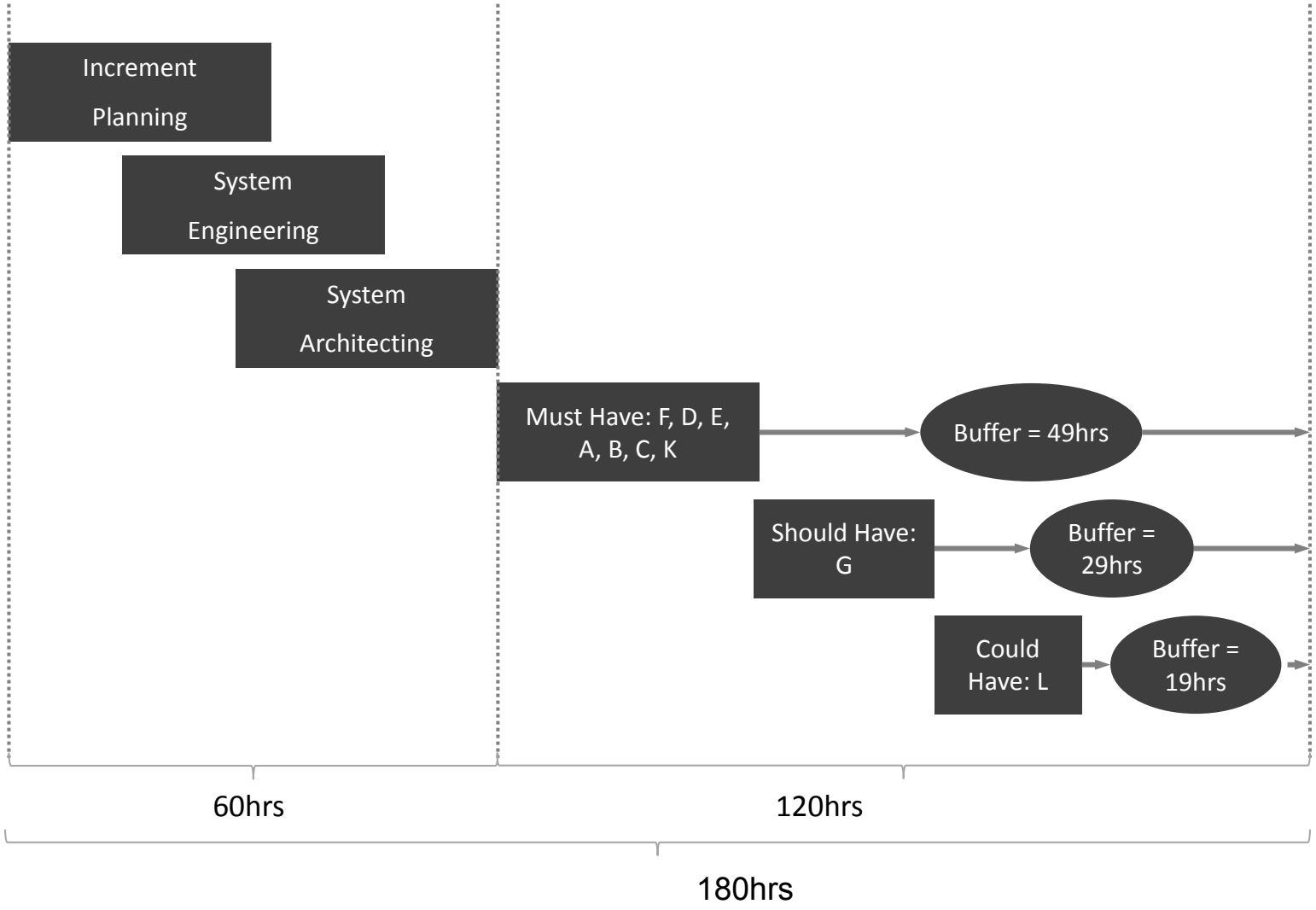
Feature	Nominal Estimate	Worst Case Estimate	Dependencies
A	20	40	B, C
B	7	9	
C	20	30	
D	5	7	E
F	5	6	
G	20	40	
H	10	20	J, K
I	15	30	
J	12	15	
K	8	10	
E	6	7	
L	10	18	

7. Features H, I and J do not fit into the remaining budget and will be scheduled as part of a new project at a later date

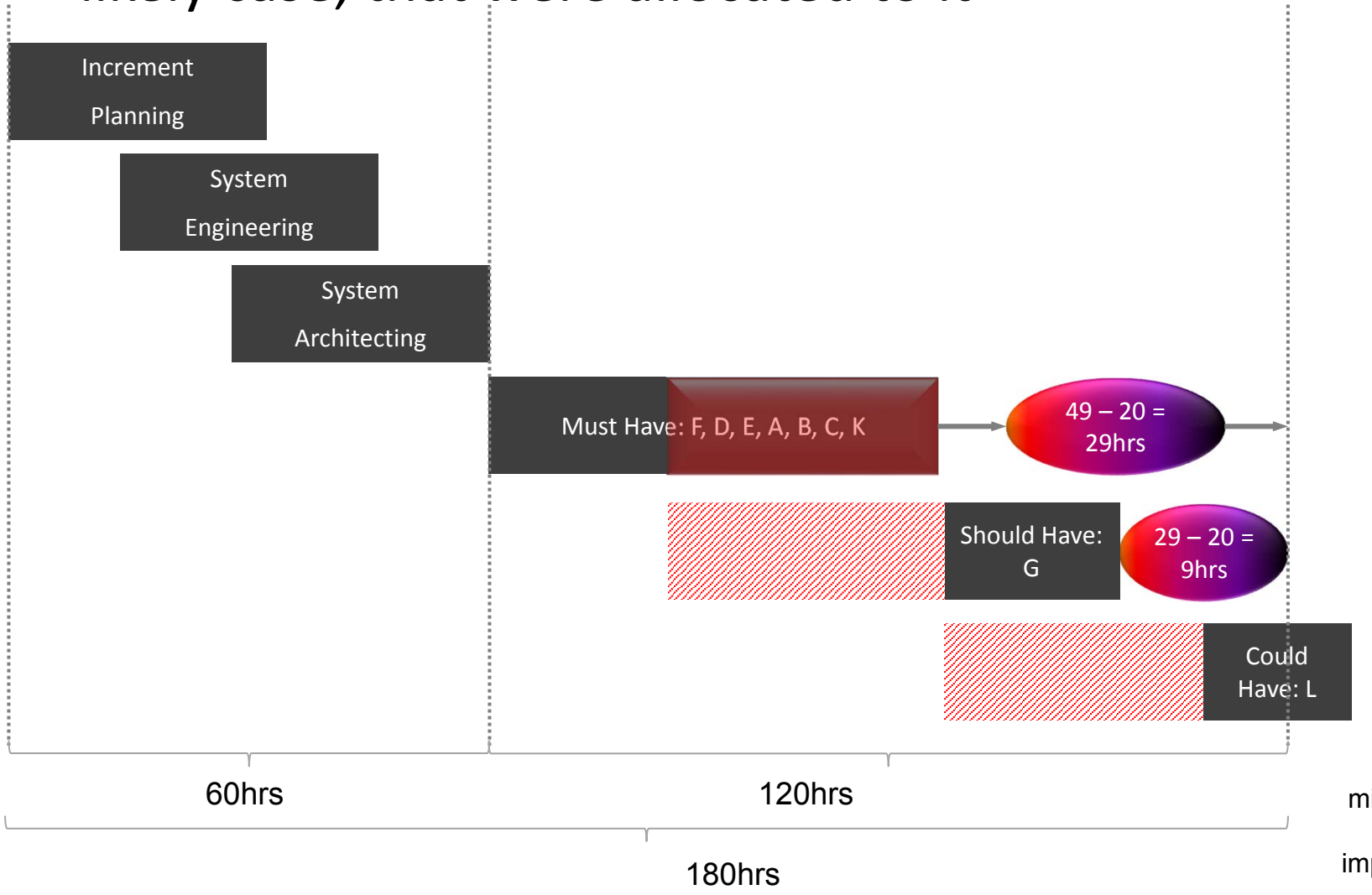
8. The final priorities are:

- Must have: F, D, E, A, B, C, K
- Should have: G
- Could have: L
- Won't have: H, I, J

Anatomy of the project planned according to buffered MoSCoW rules



Imagine that during execution feature “A” requires 40 hours, its worst case, instead of the 20, its most likely case, that were allocated to it



Feature L might or might not be implemented at this time

FAQ

- What it is the extent of the guarantee?
- How do you handle infrastructure work, changes and defects?
- Contracts & Employee incentives
  - Why will developers do something more than the “must have”?
  - Why will a client accept less than “everything”?
- What do you loose by using the worst case scenarios to simplify the problem?

What it is the extent of the guarantee?

- Results at the project level are consistent with assertions made at the lower levels
- If the assertions are wrong, the results will be consistently wrong

Handling infrastructure work, defects and changes in time boxed projects

- Technical work (aka infrastructure, technical stories, libraries, common code)
  - Incorporate these as tasks that consume available resources but are not prioritized, i.e. they spread along the duration of the project or they are done up-front
  - Breakdown the technical work into that strictly necessary to support the features on each of the releases and consider each of the parts as a feature on which all the others depend. As one depend on the other
- Changes
  - Accepted at the expense of some other feature(s). Estimate must include any rework needed
- Defects
  - Critical and major defects will be fixed
  - Minor defects will be postponed to the end or deferred to a follow on project

Incentives & contracts

- Associate employee rewards with release completion, suppress overtime and provide larger bonuses after successful deployment
- Contracts must include price incentives/penalties contingent on deliveries
- Supplier must take in consideration the probabilities of successful deliveries to price the contract

Contracting example

- Time box = 4,000 hrs.; Hourly cost = 100$, Project cost = 400,000$; Target gross margin = 30%; Project price = 520,000$; Must have = 2,000 hrs. Should have = 1,000 hrs. Could have = 1,000 hrs.
- Assume probability of delivering "must have" ≈ 1 by definition. Probability of delivering "should have" ≈ 50% (prob. of delivering every thing in the "must have" in nominal time). Probability of delivering "could have" ≈ 25% (prob. of delivering everything on the "must have" and "could have" releases in nominal time)

Fixed price with incentives

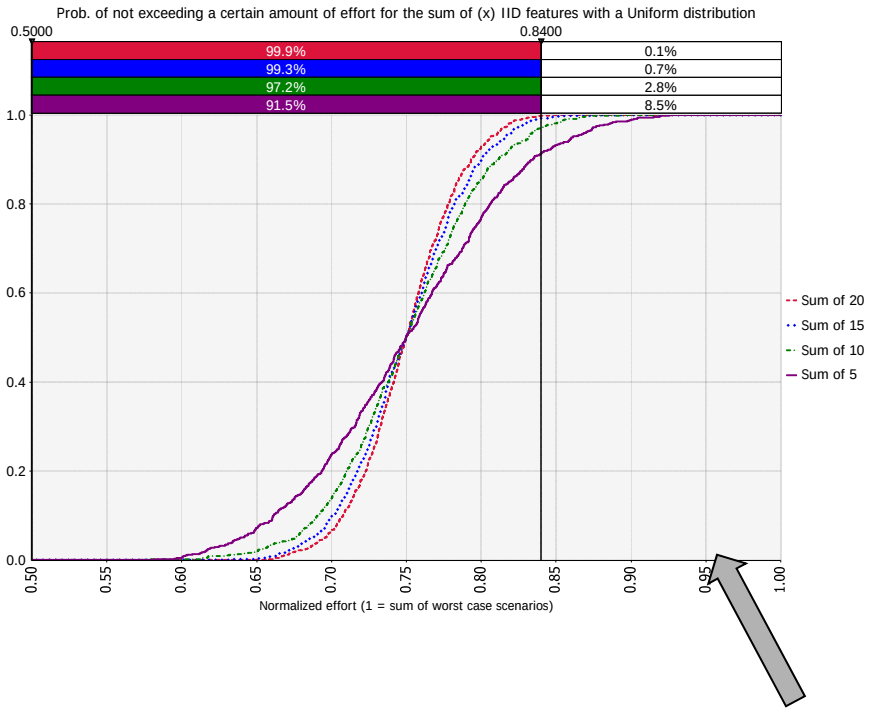
Project delivers	Fixed price (K$)	Delivery incentive (K$)	Client pays (K$)	Gross margin
Must have	400	0	400	0
Should have		100	500	25%
Could have		150	550	38%

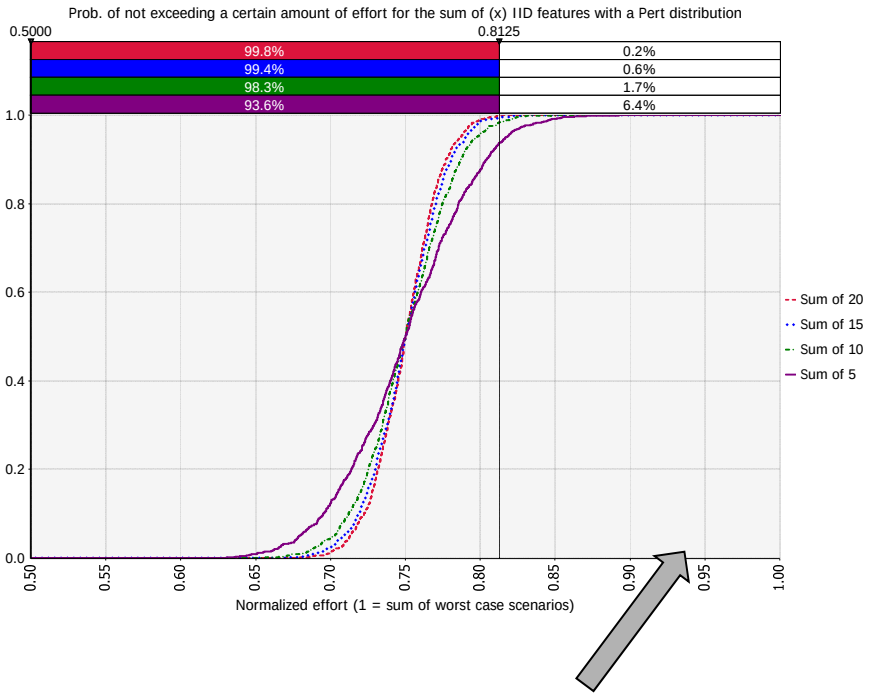
Fixed price with penalties

Project delivers	Fixed price (K$)	Delivery penalty (K$)	Client pays (K$)	Gross margin
Must have	550	150	400	0
Should have		50	500	25%
Could have		0	550	38%

Employee incentives

What do you loose by using the worst case scenarios? (1)

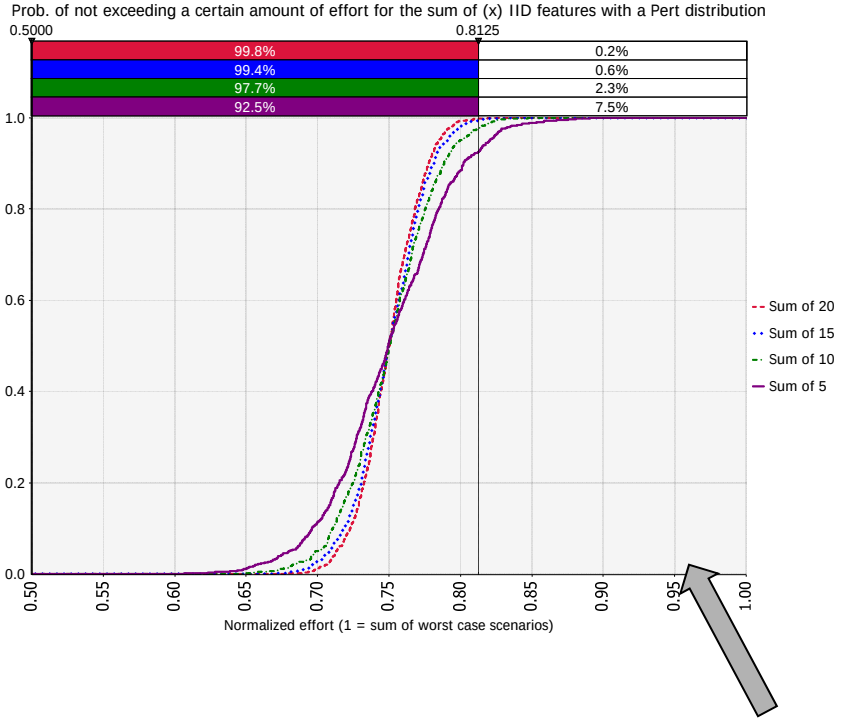




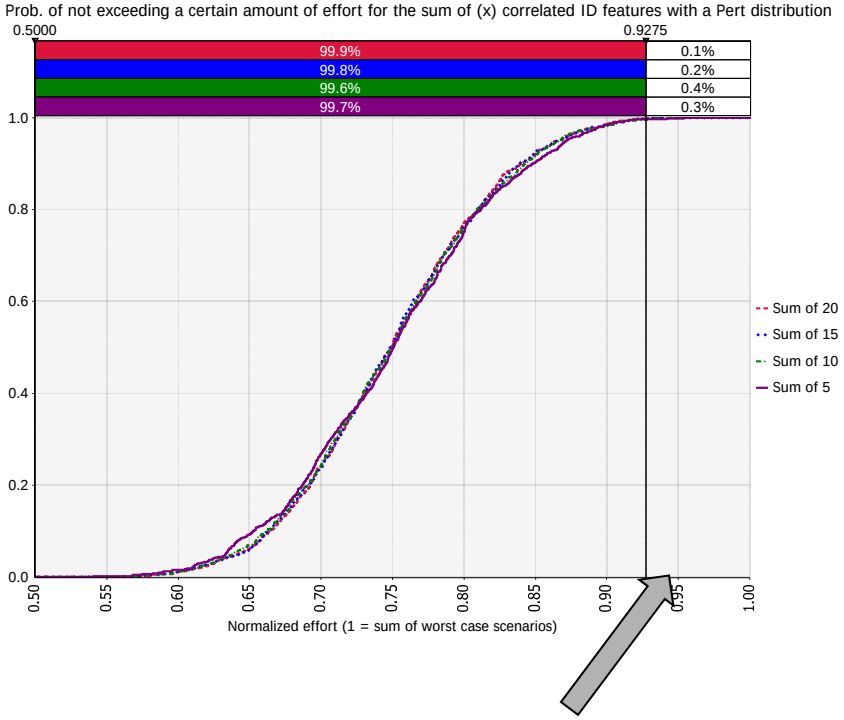
You could have accepted a couple of extra features in the “must have” or in the “should have” releases

What do you loose by using the worst case scenarios? (2)

Independent effort distribution



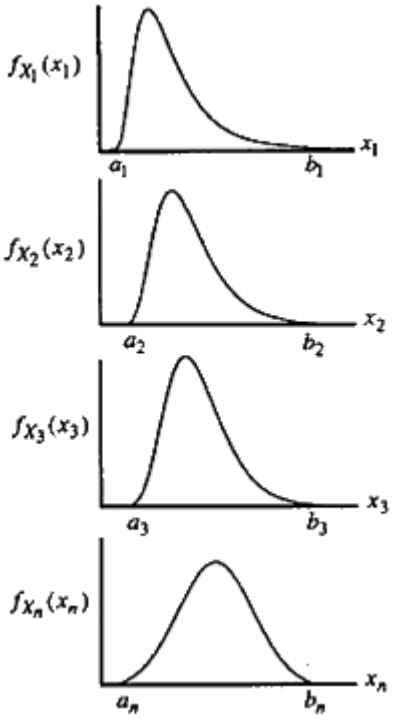
Correlated effort distribution

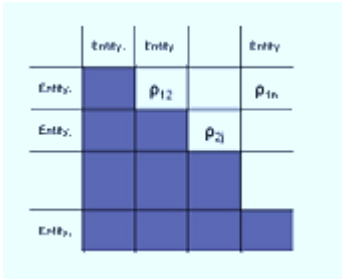


If there is an underlying common cause, e.g. we underestimated the complexity of all features, or the environment is highly unstable, the effort required by each feature will tend to shift in concert with the effort required by the others (the variables are correlated) and the total will get closer to the sum of the worst cases

A more flexible approach: The SPID Process

1) Individual distributions and correlations (Estimator analysis)





2.x) Stepwise aggregation for each release

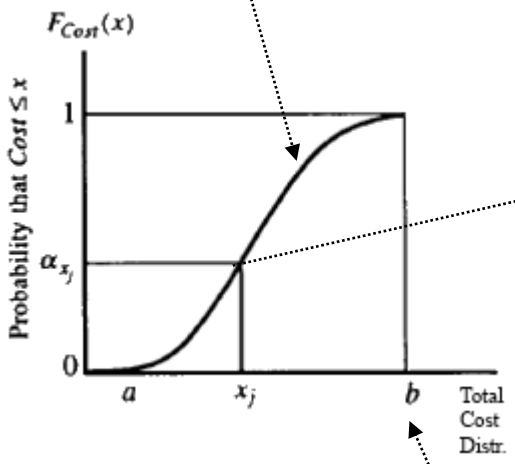
1) $ReleaseCostDistribution = f_{X_1}(x_1)$

2) $ReleaseCostDistribution = f_{X_1}(x_1) \oplus f_{X_2}(x_2)$

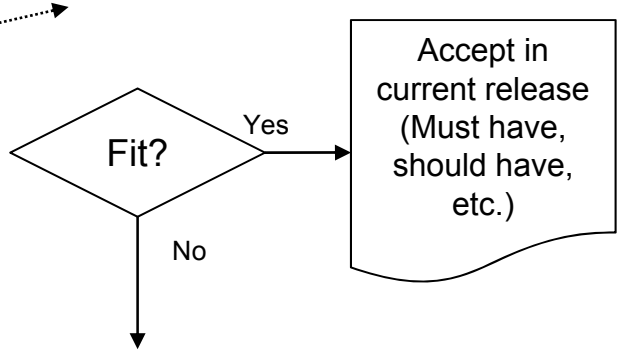
3) $ReleaseCostDistribution = f_{X_1}(x_1) \oplus f_{X_2}(x_2) \oplus f_{X_3}(x_3)$

$\vdots$

$n)\ ReleaseCostDistribution = f_{X_1}(x_1) \oplus f_{X_2}(x_2) \oplus f_{X_3}(x_3) \oplus \cdots \oplus f_{X_n}(x_n)$



3.x) Decision making



4) Repeat with new available budget for next release

Closed form solution

$$\mu_{Release} = \sum_{i=1}^{n} \mu_i$$

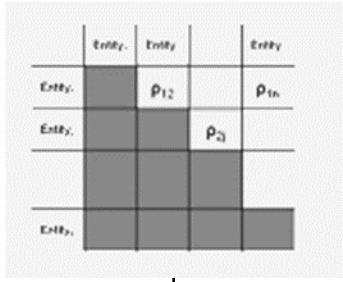
$$\mu_{Release} + 3\sigma_{Release} \leq AvailableBudget^{*}$$

$$\sigma_{Release} = \sqrt{\sum_{i=1}^{n} \sigma_i^2 + 2\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \rho_{i,j}\sigma_i\sigma_j}$$

Cost distributions

Feature	Depends on	Min	Most Likely	Max	PDF	μ	σ	Comments
A								
B								
C								
E								
F								
D								
G								

Correlation matrix



σ

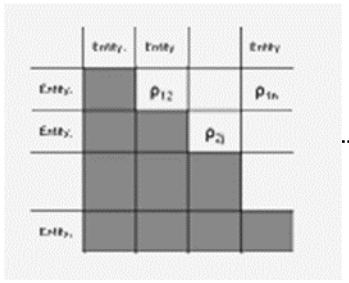
ρ

* One sided Chebyshev inequality, The Economic Analysis of Industrial Projects, L. Bussey, Prentice-Hall series in Industrial and System Engineering, 1978

Calculating the total cost distribution using Monte Carlo simulation

Inputs

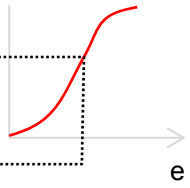
Feature	Depends on	Min	Most Likely	Max	PDF	Comments
A						
B						
C						
E						
F						
D						
G						



Monte Carlo simulation

Compute the CDF for each entity

Generate a random number [0,1]



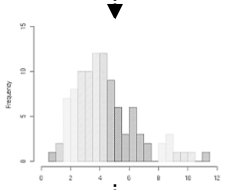
Compute the cost of entity for the run

No

All entities completed ?

Add computed costs for all the entities

Store computed cost

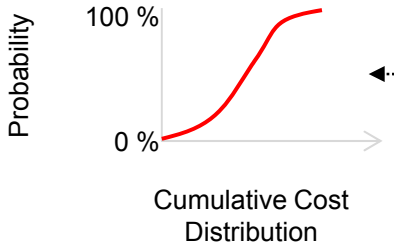


No

All iterations completed ?

Compute cost distribution

Outputs



Summary

- We have presented a simple prioritization procedure that can be applied to the ranking of requirements at the release as well as the project level. The procedure does not only captures customer preferences, but by constraining the number of features in the “must have” set as a function of the uncertainty of the underlying estimates, is able to offer project sponsors a high degree of reassurance in regards to the delivered of an agreed level of software functionality by the end of the time box
- To be workable for the supplier and the employer, the contract between the parts must incorporate the notion that an agreed partial delivery is an acceptable, although not preferred, outcome
- Similarly, employees must be engaged into the process to prevent free rides
- The method simplicity is not free. It comes at the expense of the claims we can make about the likelihood of delivering a given functionality and a conservative buffer. Users seeking to make more definitive probabilistic statements or optimize the buffer size should consider the use of a more sophisticated approach such as SPID (described in Planning and Executing Time Bound Projects, E. Miranda, 2002)

Questions?