ETSN15 - Requirements Engineering

Lecture 6: Release Planning

This lecture helps you prepare for the second part of Lab 2 on release planning (esp. [RP], see papers behind moddle wall)

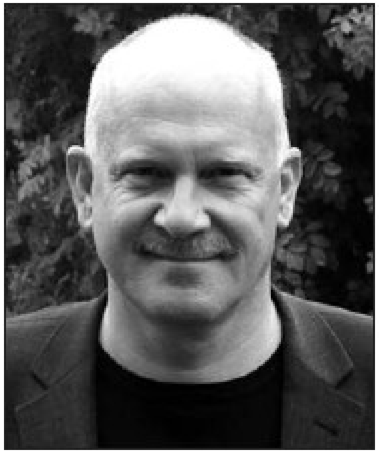
(Lecture 8 helps you prepare for the first part of Lab 2: QR)

Björn Regnell
http://www.cs.lth.se/krav

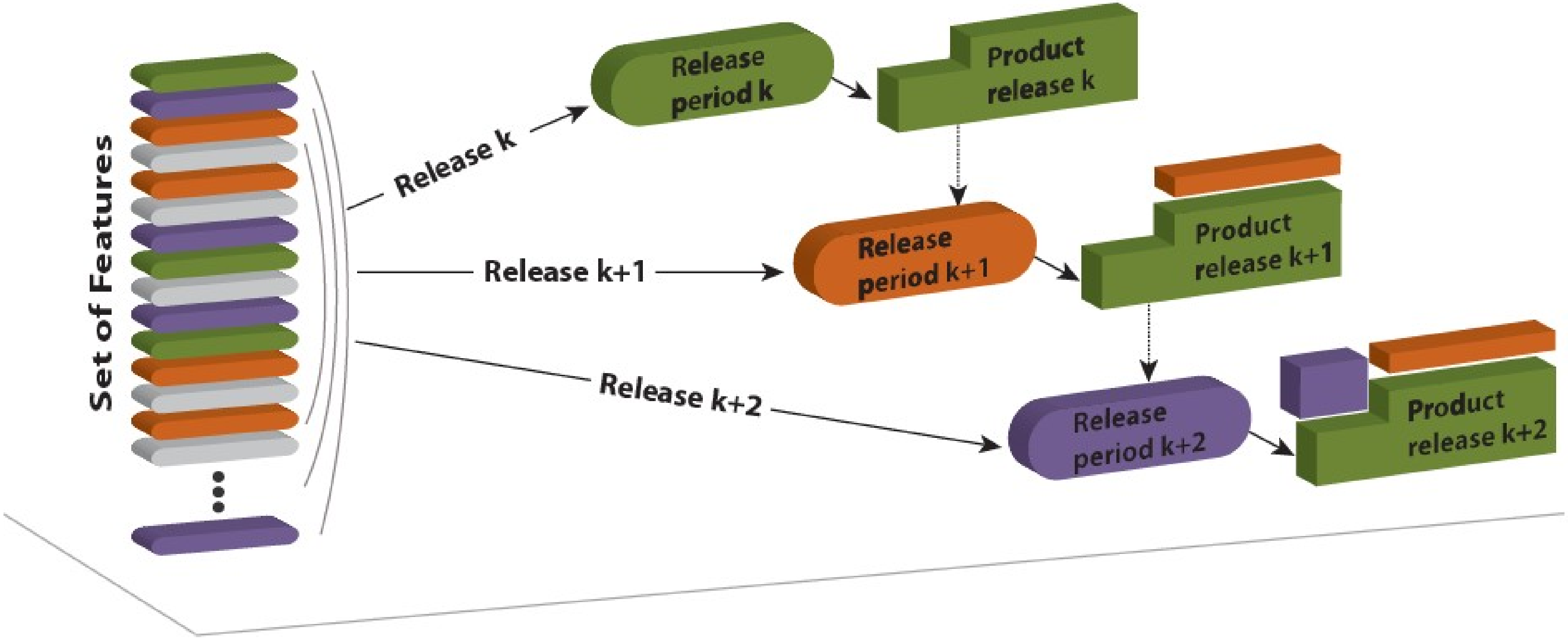
Release Planning

Paper [RP] in compendium

- “The art and science of software release planning”
  Ruhe, G., & Saliu, M. O,
  IEEE software, 22(6), 47-53. 2005



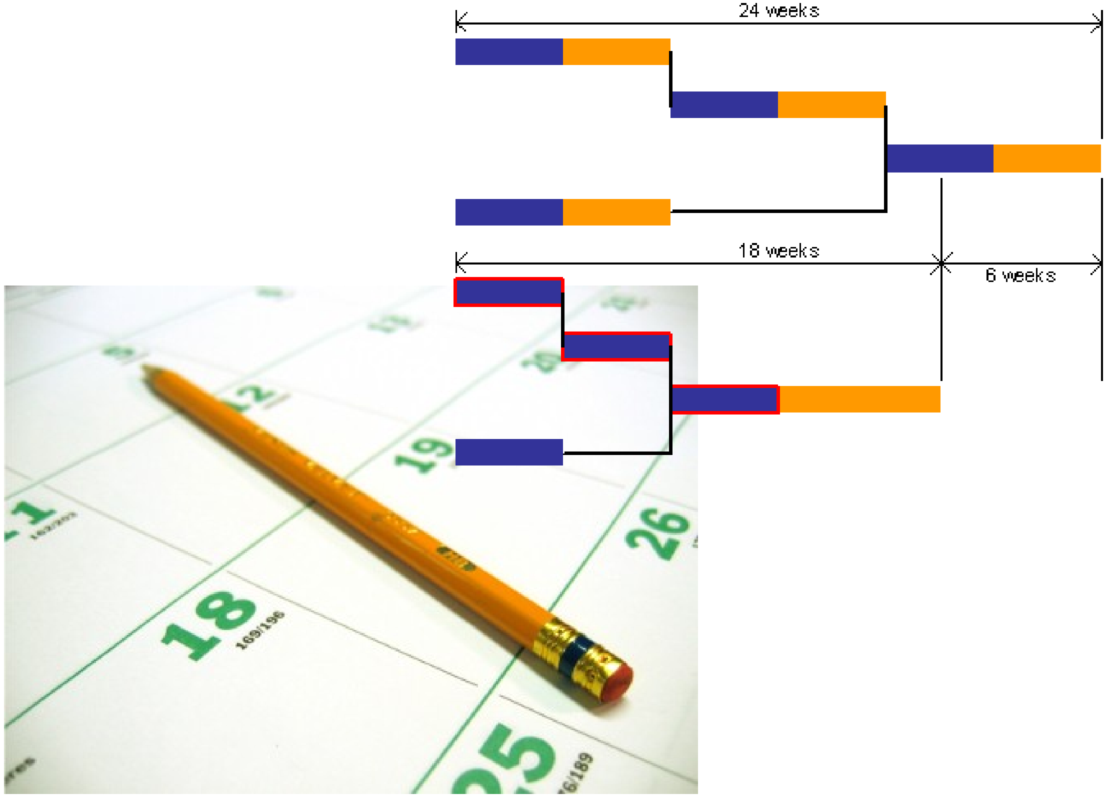
What is Release Planning?



[RP]

Release Planning involves...

- ...prioritization + scheduling under various constraints, e.g., resource and precedence constraints



Time
Money
Quality

[RP]

Example planning parameters

- Requirements priorities (from prioritization)
- Available resources
- Delivery time
- Requirements interdependencies
  - Precedence, Coupling, Excludes
- System architecture
- Dependencies to the code base

[RP]

What is a good release plan?

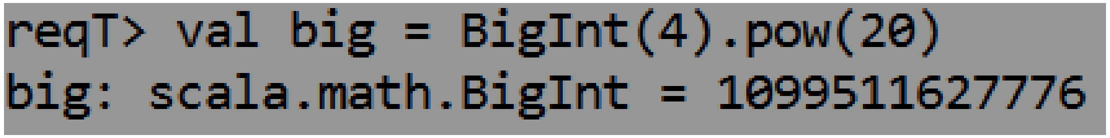
- A good release plan should...
  - provides maximum (?) business value by
    - offering the “best” possible blend of features
    - in the “right” sequence of releases
  - satisfy the most important stakeholders
  - be feasible with available resources, and
  - reflect existing dependencies between features

- Release planning is similar to the **NP-complete** Knapsack problem:
  https://en.wikipedia.org/wiki/Knapsack_problem

Baseline:
Release Planning - on the fly

- Informal process
- Rationale behind decisions not always clear
- Constraints regarding e.g., resources and stakeholders not systematically taken into account
- Already in case of 20 features and 3 releases

$4^{20} > 1.000.000.000.000 = 10^{12}$ possibilities



```
reqT> val big = BigInt(4).pow(20)
big: scala.math.BigInt = 1099511627776
```

[RP]

Investigate with reqT why greedy is not good

https://github.com/reqT/reqT/blob/3.0.x/doc/lab2/greedy.sc

```
val m = Model(
  Feature("a") has (Benefit(90), Cost(100)),
  Feature("b") has (Benefit(85), Cost(90)),
  Feature("c") has (Benefit(80), Cost(25)),
  Feature("d") has (Benefit(75), Cost(23)),
  Feature("e") has (Benefit(70), Cost(22)),
  Feature("f") has (Benefit(65), Cost(20)),
  Feature("g") has (Benefit(60), Cost(10)),
  Feature("h") has (Benefit(55), Cost(30)),
  Feature("i") has (Benefit(50), Cost(30)),
  Feature("j") has (Benefit(45), Cost(30)),
  Release("r1") has Capacity(100),
  Release("r2") has Capacity(90))
```

```
def plan(input: Model,
      pickNext: (Model,Release)=>Option[Feature]): Model = {
  var result = input
  releases(input).foreach { r =>
    var next = pickNext(result, r)
    while (next.isDefined) {
      result = allocate(result, next.get, r)
      next = pickNext(result, r)
    }
  }
  result
}

plan(m, random)
plan(m, greedy)
```

```
def features(m: Model): Vector[Feature] = m.collect{case f: Feature => f}.distinct
def releases(m: Model): Vector[Release] = m.collect{case r: Release => r}.distinct
def allocate(m: Model, f: Feature, r: Release): Model = m + (r has f)
def isAllocated(m: Model, f: Feature): Boolean = releases(m).exists(r => (m/r).contains(f))
def allocatedCost(m: Model, r: Release): Int = (m/r).entities.collect{case f => m/f/Cost}.sum
def isRoom(m: Model, f: Feature, r: Release) = m/r/Capacity >= allocatedCost(m,r) + m/f/Cost
def featuresInGreedyOrder(m: Model) = features(m).sortBy(f => m/f/Benefit).reverse

def random(m: Model, r: Release): Option[Feature] = scala.util.Random.shuffle(features(m)).
  filter(f => !isAllocated(m,f) && isRoom(m,f,r)).headOption

def greedy(m: Model, r: Release): Option[Feature] =
  featuresInGreedyOrder(m).find(f => !isAllocated(m,f) && isRoom(m,f,r))
```

Optimal vs. Greedy

```
val optimal = Model(
  Feature("a") has (Benefit(90), Cost(100)),
  Feature("b") has (Benefit(85), Cost(90)),
  Feature("c") has (Benefit(80), Cost(25)),
  Feature("d") has (Benefit(75), Cost(23)),
  Feature("e") has (Benefit(70), Cost(22)),
  Feature("f") has (Benefit(65), Cost(20)),
  Feature("g") has (Benefit(60), Cost(10)),
  Feature("h") has (Benefit(55), Cost(30)),
  Feature("i") has (Benefit(50), Cost(30)),
  Feature("j") has (Benefit(45), Cost(30)),
  Release("r1") has (Capacity(100),
    Feature("c"), Feature("d"), Feature("e"), Feature("f"), Feature("g")),
  Release("r2") has (Capacity(90),
    Feature("h"), Feature("i"), Feature("j")))
```

```
def sumAllocatedBenefit(m: Model): Int =
  releases(m).map(r => (m/r).collect{case f: Feature => m/f/Benefit}.sum).sum

val beneftitOptimal = sumAllocatedBenefit(optimal)
val benefitGreedy   = sumAllocatedBenefit(plan(m,greedy))
val ratio = benefitGreedy.toDouble / beneftitOptimal
```

How to estimate benefit and cost?

- Use prioritsation techniques [PRIO]
- Implemented in reqT and used in lab1:
  ordinal-scale comparisons
  ratio-scale $100-method

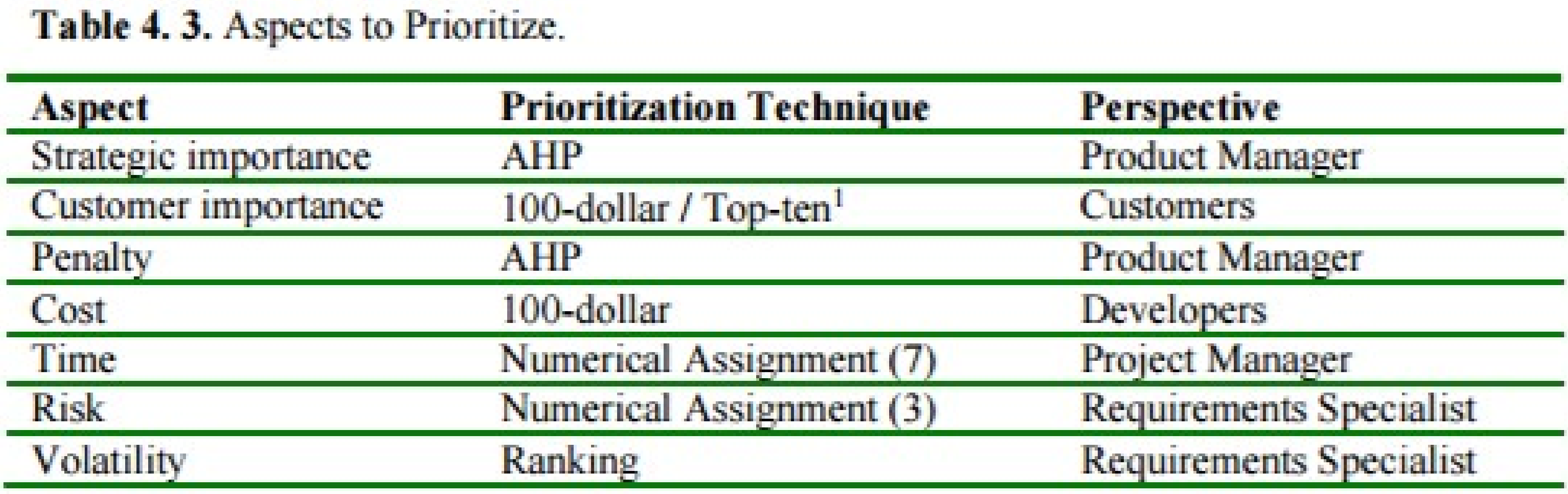


Table 4. 3. Aspects to Prioritize.

Aspect	**Prioritization Technique**	**Perspective**
Strategic importance	AHP	Product Manager
Customer importance	100-dollar / Top-ten[1]	Customers
Penalty	AHP	Product Manager
Cost	100-dollar	Developers
Time	Numerical Assignment (7)	Project Manager
Risk	Numerical Assignment (3)	Requirements Specialist
Volatility	Ranking	Requirements Specialist

Example from [PRIO]

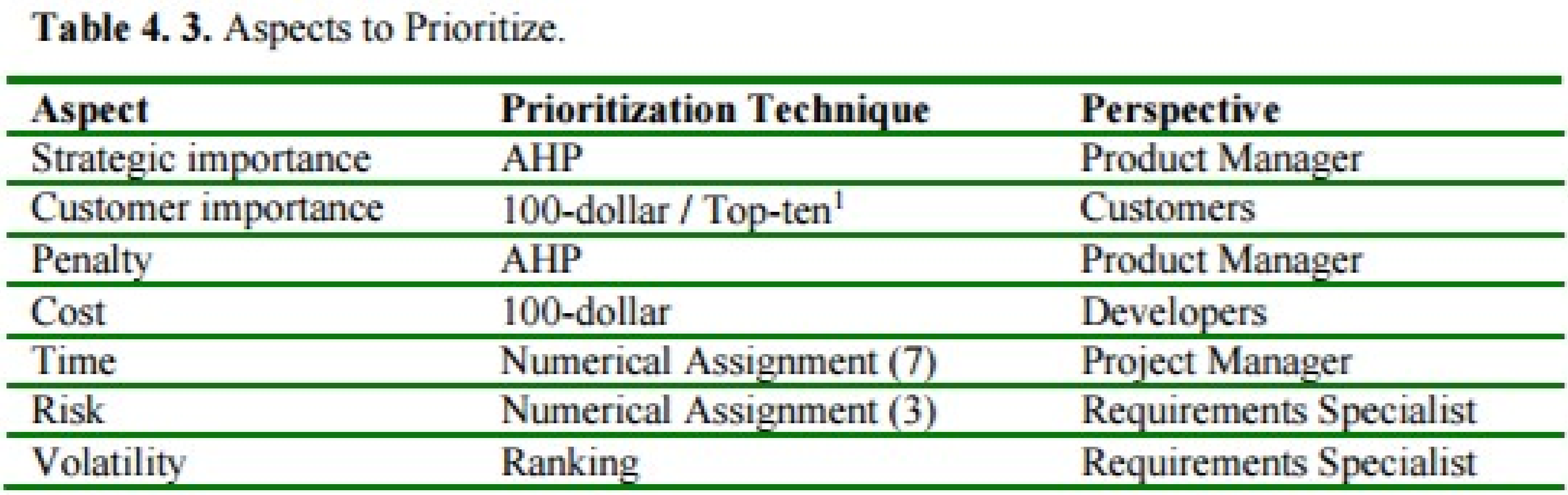


Table 4. 3. Aspects to Prioritize.

Aspect	**Prioritization Technique**	**Perspective**
Strategic importance	AHP	Product Manager
Customer importance	100-dollar / Top-ten[1]	Customers
Penalty	AHP	Product Manager
Cost	100-dollar	Developers
Time	Numerical Assignment (7)	Project Manager
Risk	Numerical Assignment (3)	Requirements Specialist
Volatility	Ranking	Requirements Specialist

Example from [RP]

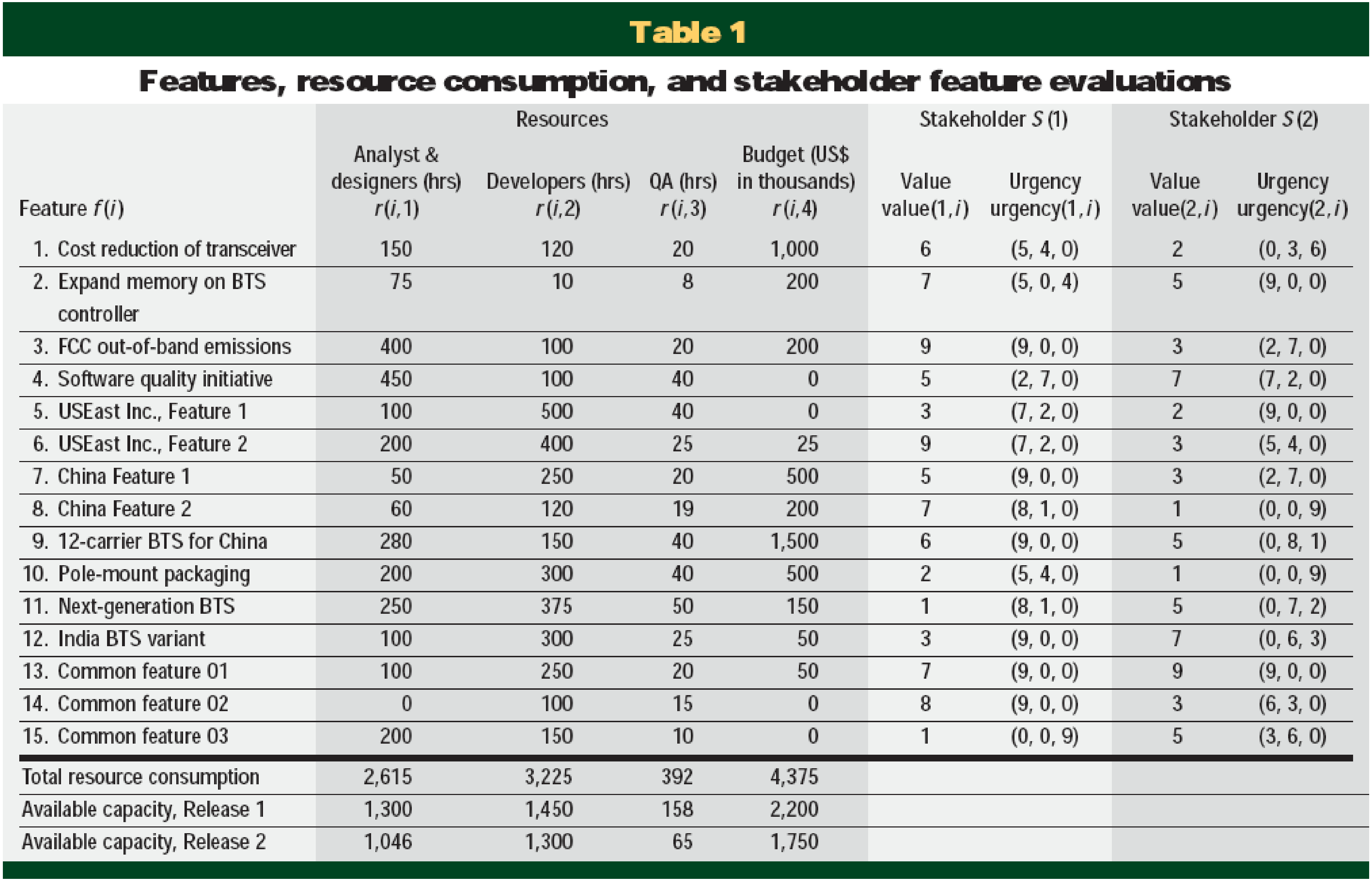


Table 1

Features, resource consumption, and stakeholder feature evaluations

	Resources				Stakeholder S (1)		Stakeholder S (2)	
Feature $f(i)$	Analyst & designers (hrs) $r(i,1)$	Developers (hrs) $r(i,2)$	QA (hrs) $r(i,3)$	Budget (US$ in thousands) $r(i,4)$	Value value(1,i)	Urgency urgency(1,i)	Value value(2,i)	Urgency urgency(2,i)
1. Cost reduction of transceiver	150	120	20	1,000	6	(5, 4, 0)	2	(0, 3, 6)
2. Expand memory on BTS controller	75	10	8	200	7	(5, 0, 4)	5	(9, 0, 0)
3. FCC out-of-band emissions	400	100	20	200	9	(9, 0, 0)	3	(2, 7, 0)
4. Software quality initiative	450	100	40	0	5	(2, 7, 0)	7	(7, 2, 0)
5. USEast Inc., Feature 1	100	500	40	0	3	(7, 2, 0)	2	(9, 0, 0)
6. USEast Inc., Feature 2	200	400	25	25	9	(7, 2, 0)	3	(5, 4, 0)
7. China Feature 1	50	250	20	500	5	(9, 0, 0)	3	(2, 7, 0)
8. China Feature 2	60	120	19	200	7	(8, 1, 0)	1	(0, 0, 9)
9. 12-carrier BTS for China	280	150	40	1,500	6	(9, 0, 0)	5	(0, 8, 1)
10. Pole-mount packaging	200	300	40	500	2	(5, 4, 0)	1	(0, 0, 9)
11. Next-generation BTS	250	375	50	150	1	(8, 1, 0)	5	(0, 7, 2)
12. India BTS variant	100	300	25	50	3	(9, 0, 0)	7	(0, 6, 3)
13. Common feature 01	100	250	20	50	7	(9, 0, 0)	9	(9, 0, 0)
14. Common feature 02	0	100	15	0	8	(9, 0, 0)	3	(6, 3, 0)
15. Common feature 03	200	150	10	0	1	(0, 0, 9)	5	(3, 6, 0)
Total resource consumption	2,615	3,225	392	4,375				
Available capacity, Release 1	1,300	1,450	158	2,200				
Available capacity, Release 2	1,046	1,300	65	1,750				

Example from [RP]

WAS: weighted average satisfaction of stakeholder priorities

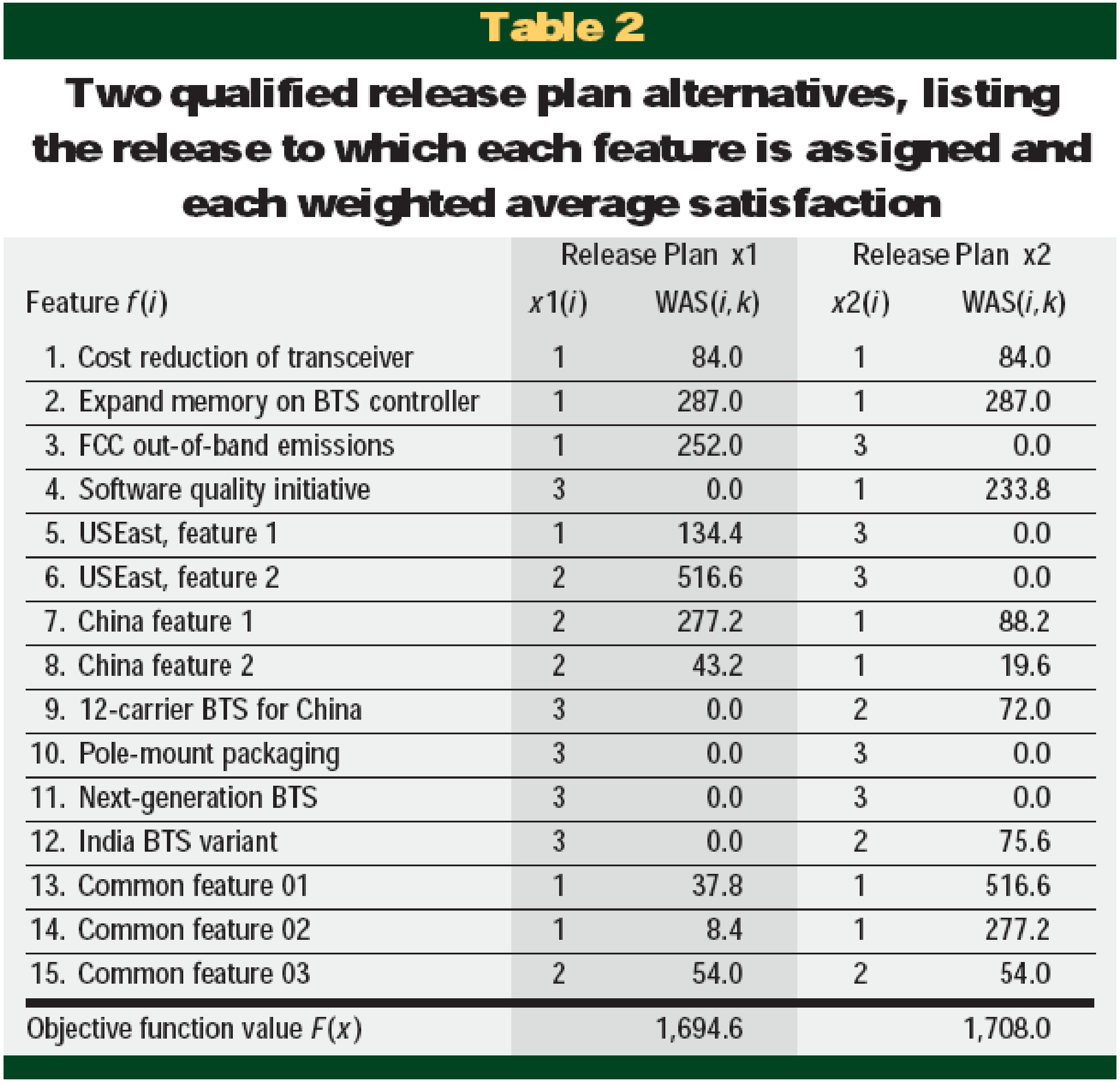


Table 2

Two qualified release plan alternatives, listing the release to which each feature is assigned and each weighted average satisfaction

	Release Plan x1		Release Plan x2	
Feature *f*(*i*)	*x*1(*i*)	WAS(*i*, *k*)	*x*2(*i*)	WAS(*i*, *k*)
1. Cost reduction of transceiver	1	84.0	1	84.0
2. Expand memory on BTS controller	1	287.0	1	287.0
3. FCC out-of-band emissions	1	252.0	3	0.0
4. Software quality initiative	3	0.0	1	233.8
5. USEast, feature 1	1	134.4	3	0.0
6. USEast, feature 2	2	516.6	3	0.0
7. China feature 1	2	277.2	1	88.2
8. China feature 2	2	43.2	1	19.6
9. 12-carrier BTS for China	3	0.0	2	72.0
10. Pole-mount packaging	3	0.0	3	0.0
11. Next-generation BTS	3	0.0	3	0.0
12. India BTS variant	3	0.0	2	75.6
13. Common feature 01	1	37.8	1	516.6
14. Common feature 02	1	8.4	1	277.2
15. Common feature 03	2	54.0	2	54.0
Objective function value *F*(*x*)		1,694.6		1,708.0

Example from [RP]

WAS:
weighted
average
satisfaction
of stakeholder
priorities

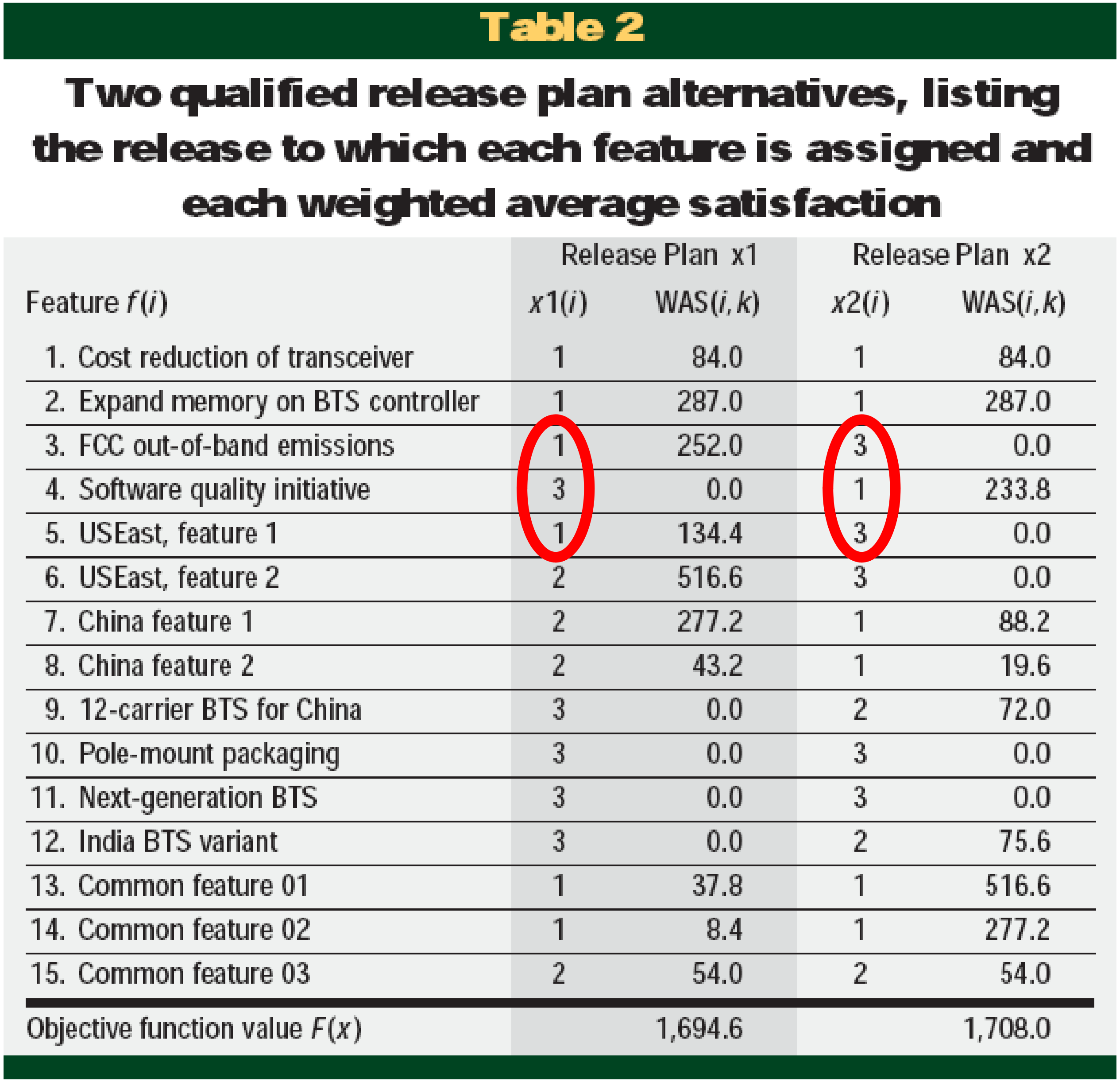


Table 2

Two qualified release plan alternatives, listing the release to which each feature is assigned and each weighted average satisfaction

	Release Plan x1		Release Plan x2	
Feature *f*(*i*)	*x*1(*i*)	WAS(*i*, *k*)	*x*2(*i*)	WAS(*i*, *k*)
1. Cost reduction of transceiver	1	84.0	1	84.0
2. Expand memory on BTS controller	1	287.0	1	287.0
3. FCC out-of-band emissions	1	252.0	3	0.0
4. Software quality initiative	3	0.0	1	233.8
5. USEast, feature 1	1	134.4	3	0.0
6. USEast, feature 2	2	516.6	3	0.0
7. China feature 1	2	277.2	1	88.2
8. China feature 2	2	43.2	1	19.6
9. 12-carrier BTS for China	3	0.0	2	72.0
10. Pole-mount packaging	3	0.0	3	0.0
11. Next-generation BTS	3	0.0	3	0.0
12. India BTS variant	3	0.0	2	75.6
13. Common feature 01	1	37.8	1	516.6
14. Common feature 02	1	8.4	1	277.2
15. Common feature 03	2	54.0	2	54.0
Objective function value *F*(*x*)		1,694.6		1,708.0

“qualified RP” =
covers at least 95% of the
objective function’s
maximum value

TODO!

- Lab2: Quality Requiremenst (Lecture 8) and Release Planning. Mandatory Preparations: http://cs.lth.se/krav/labs/lab2/
  - Read [PRMAN], [RP] and [QUPER], [Lau:6]
  - Bring written representations of: 3 QR, 3 Features, 2 Stakeholders from your project.
- Lecture 7 – Validation, inspections, Agile RE
  - This lecture covers research papers etc and if you attend you will **save much effort** when you study for the exam and when you plan your project work.
- Lecture 8 – Quality Requirements:
  - Let’s try “efterläsning”: You watch this video http://cs.lth.se/krav/quality-requirements/ **BEFORE** the lecture and at the lecture we will actively discuss QR in your projects.
- Exercise 5 – Validation
  - Practical work that you must do in your project anyway