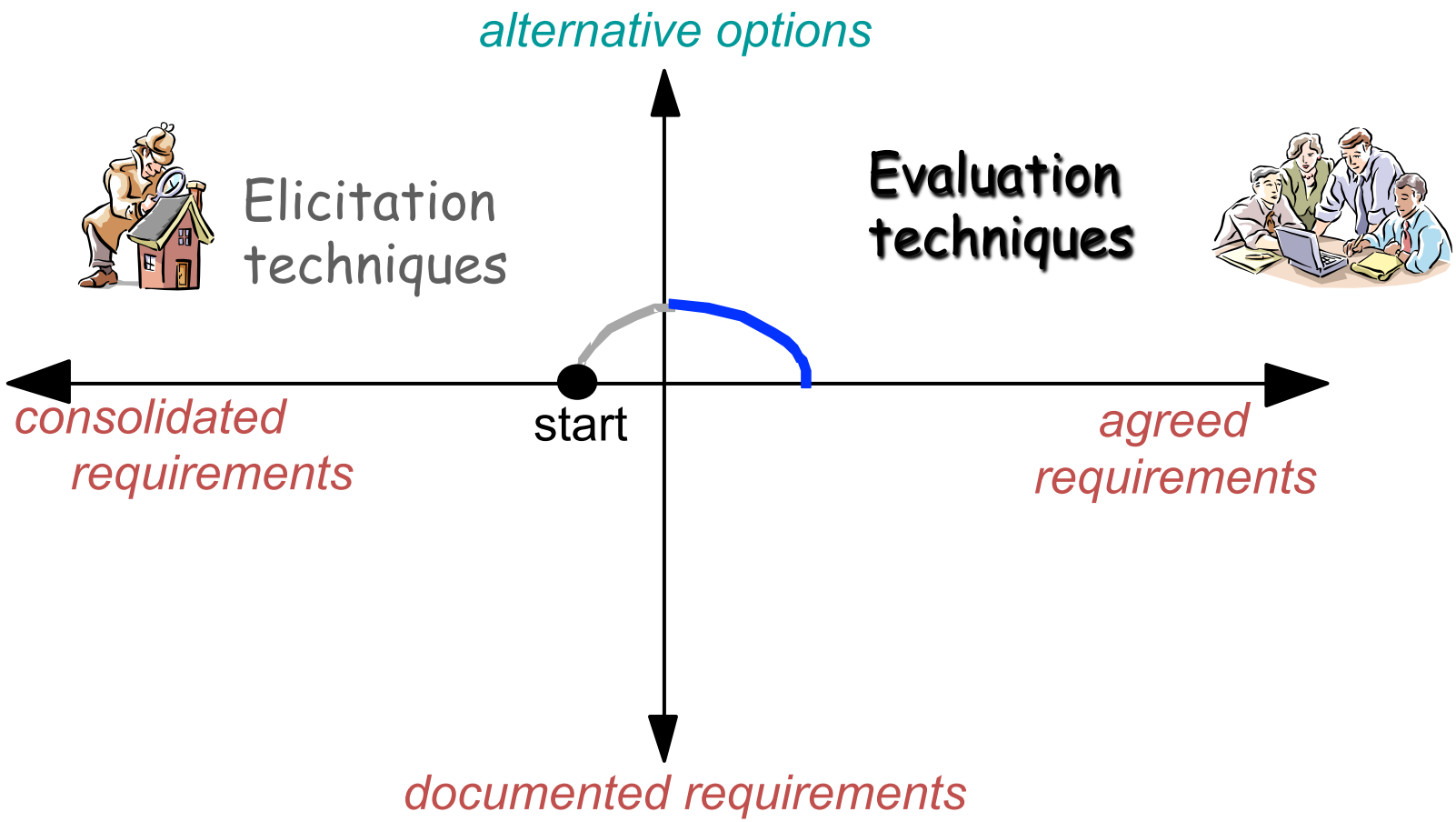
Requirements Engineering
- Requirements Evaluation -

Slides adapted from official slides of the book “Requirements Engineering”, A. van Lamsweerde


alternative options
Elicitation techniques
Evaluation techniques
consolidated requirements
start
agreed requirements
documented requirements

Requirements evaluation: outline

- Inconsistency management
  - Types of inconsistency
  - Handling inconsistencies
  - Managing conflicts: a systematic process
- Evaluating alternative options for decision making
- Requirements prioritization

Inconsistency management

Inconsistency = violation of consistency rule among items

inter-viewpoints

each stakeholder has its own focus & concerns

intra-viewpoint

conflicting quality reqs

Inconsistencies must be detected and resolved

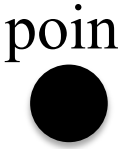
- **not too soon**: to allow further elicitation within viewpoint
- **not too late**: to allow software development

Types of inconsistency

Terminology clash: same concept named differently in different statements

e.g. library management: "borrower" *vs.* "patron"

Designation clash: same name for different concepts in different statements



e.g. "user" for "library user" *vs.* "library software user"

Structure clash: same concept structured differently in different statements

e.g. "latest return date" as time point (e.g. Fri 5pm) *vs.* time interval (e.g. Friday)

Strong conflict: statements not satisfiable together (*S*, *not S*)

e.g. "participant constraints may not be disclosed to anyone else" *vs.* "the meeting initiator should know participant constraints"

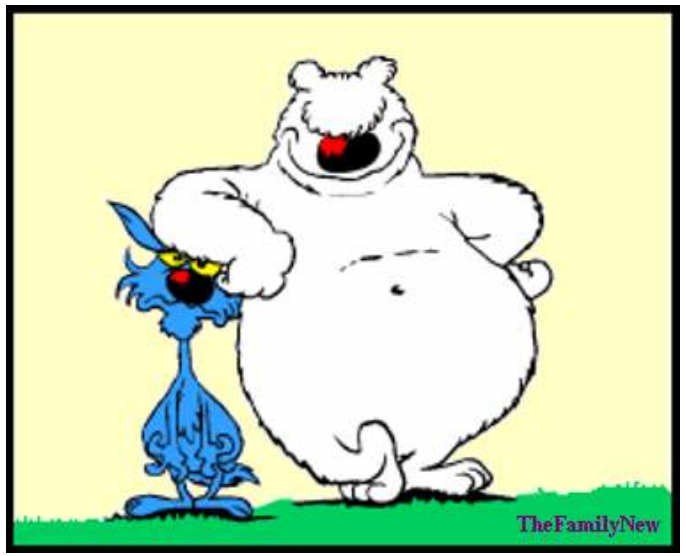
Weak conflict (divergence): statements not satisfiable together under some **boundary condition**

i.e. strongly conflicting if *B* holds

e.g. (staff's viewpoint) "patrons shall return borrowed copies within *X* weeks"

vs. (patron's viewpoint) "patrons shall keep borrowed copies as long as needed"

B: "a patron needing a borrowed copy more than *X* weeks"



Handling clashes in terminology, designation, structure

- through agreed **glossary of terms** to stick to
  - For some terms, if needed: accepted synonym(s)
  - To be built during elicitation phase

SAMPLE GLOSSARY OF TERMS

…

Card with a credit function: see credit card.

Card with a debit function: see debit card.

Cash card: a card which has only a cash function. See also card with a cash function.

Cash dispenser: an electromechanical device that permits authorised users to withdraw banknotes, typically using machine-readable plastic cards. See also automated teller machine.

Cash settlement agent: the entity whose assets or liabilities are used to settle the payment obligations arising from funds transfer systems or from securities transfers within a central securities depository (CSD). Commercial banks and central banks are typical cash settlement agents.

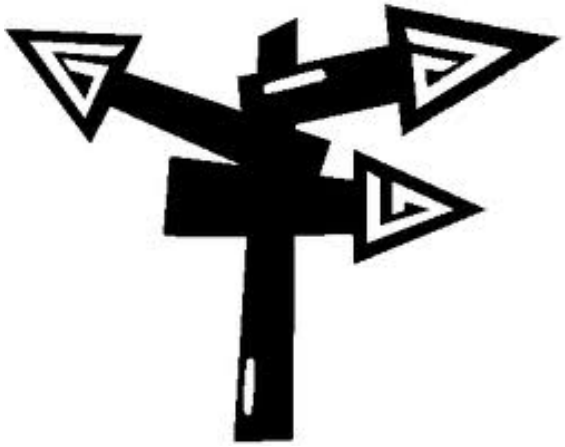
CCBM: see correspondent central banking model.

Handling inconsistencies

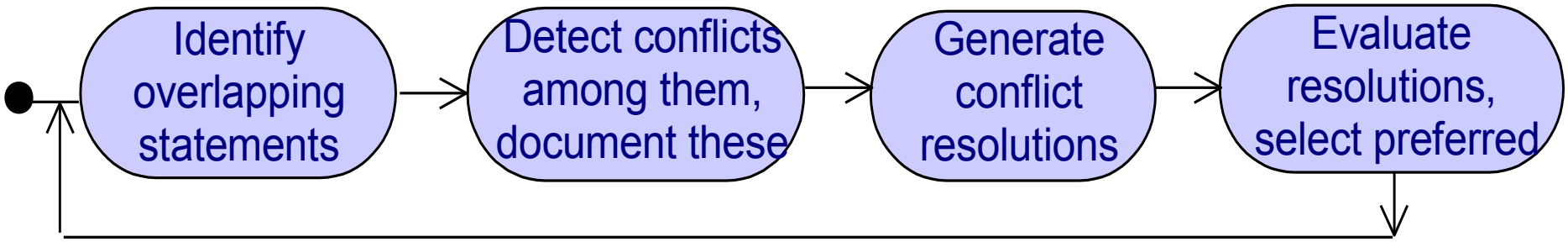
Weak, strong conflicts: more difficult, deeper causes...

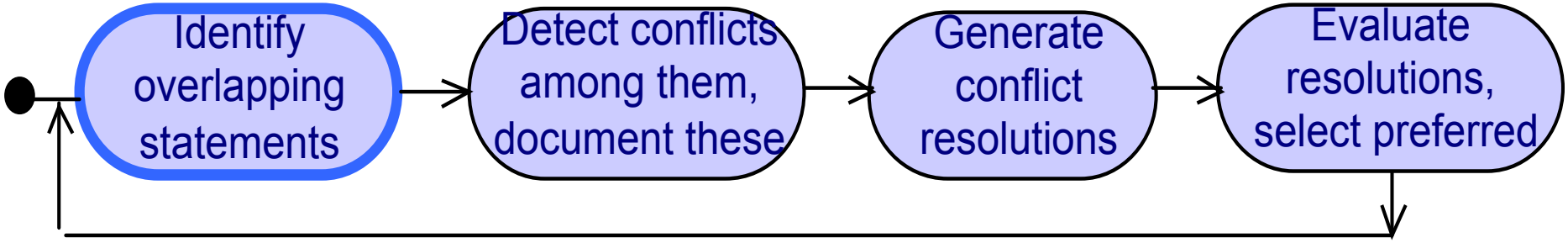
- Conflicting personal objectives of stakeholders => to be handled at root level and propagated to requirements level

- Inherent to some non-functional concerns (performance *vs.* safety, confidentiality *vs.* awareness, ...) => exploration of preferred tradeoffs

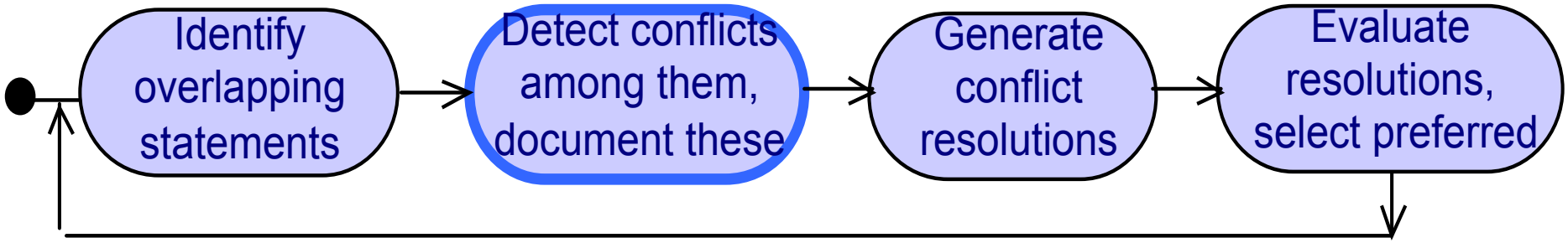


Managing conflicts: a systematic process





- **Overlap** = reference to common terms or phenomena
  - precondition for conflicting statements
  - e.g. gathering meeting constraints, determining schedules



- **Conflict detection**
  - informally
  - using heuristics on conflicting req categories
    - "Check *information* req & *confidentiality* req on related objects"
    - "Check reqs on *decreasing* & *increasing* related quantities"
  - formally

Detected conflicts should be documented

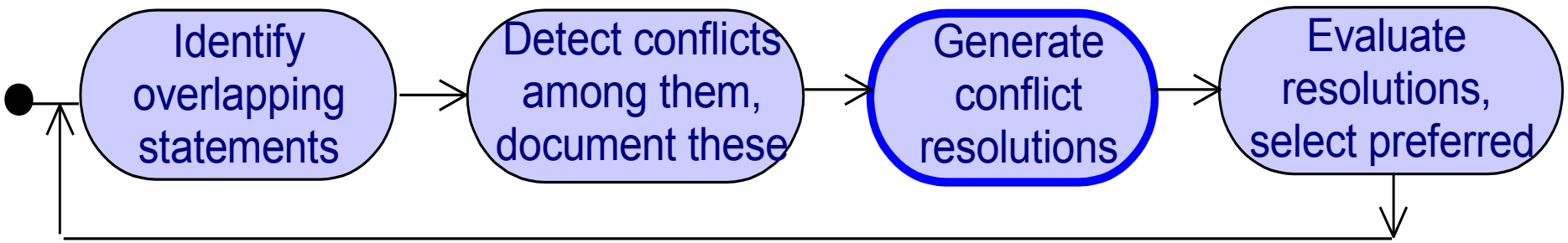
- For later resolution, and for impact analysis
- RE tools support tracing of conflicts
- **Interaction matrix**:

Statement	S1	S2	S3	S4	Total
S1	0	1000	1	1	1002
S2	1000	0	0	0	1000
S3	1	0	0	1	2
S4	1	0	1	0	2
Total	1002	1000	2	2	2006

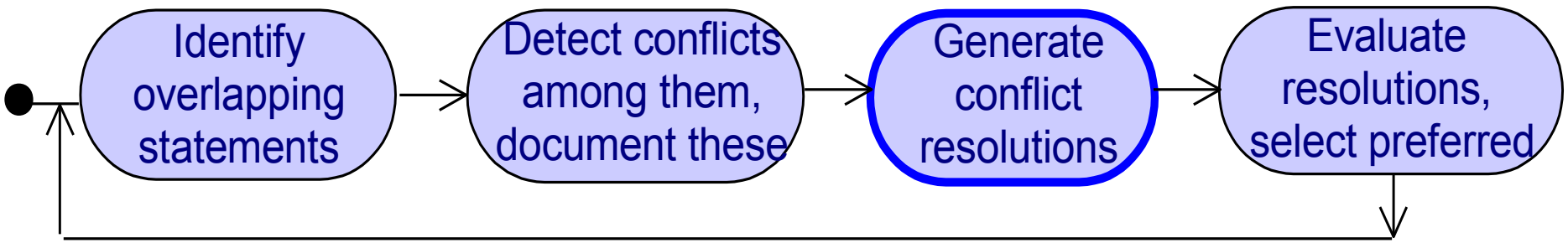
1 = conflict
0 = no overlap
1000 = overlap, but not conflict

#Conflicts(S1) = remainderOf(1002 **div** 1000)

#nonConflictingOverlaps(S1) = quotientOf(1002 **div** 1000)



- For optimal resolution, better to
  - **explore** multiple candidate resolutions *first*,
  - **compare**, select/agree on most preferred *next*
- To generate candidate resolutions, use?



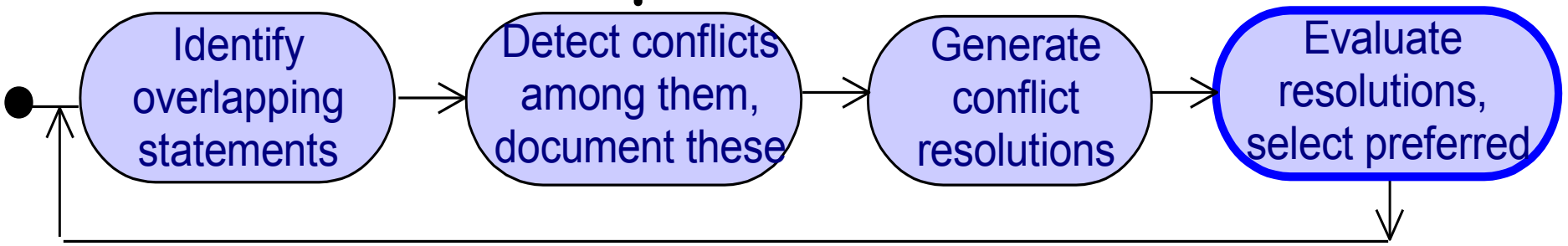
- For optimal resolution, better to
  - **explore** multiple candidate resolutions *first*,
  - **compare**, select/agree on most preferred *next*
- To generate candidate resolutions, use
  - **elicitation** techniques (interviews, group sessions)
  - Apply **resolution tactics** (see next slide)

Conflict resolution tactics

- **Avoid** boundary condition
  - e.g. "Keep copies of highly needed books unborrowable"
- **Restore** conflicting statements
  - e.g. "Copy returned within X weeks *and then* borrowed again"
- **Weaken** conflicting statements
  - e.g. "Copy returned within X weeks *unless* explicit permission"
- **Drop** lower-priority statements
- **Specialize** conflict source or target
  - e.g. "Book loan status known *by staff users only*"

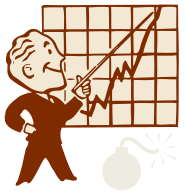
Transform conflicting statements or involved objects, or introduce new requirements

Managing conflicts: a systematic process



- Evaluation criteria for preferred resolution:
  - contribution to critical non-functional requirements
  - contribution to resolution of *other* conflicts & risks
  - Application of risk analysis principles

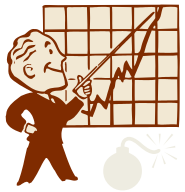
Requirements Prioritization



Requirements prioritization

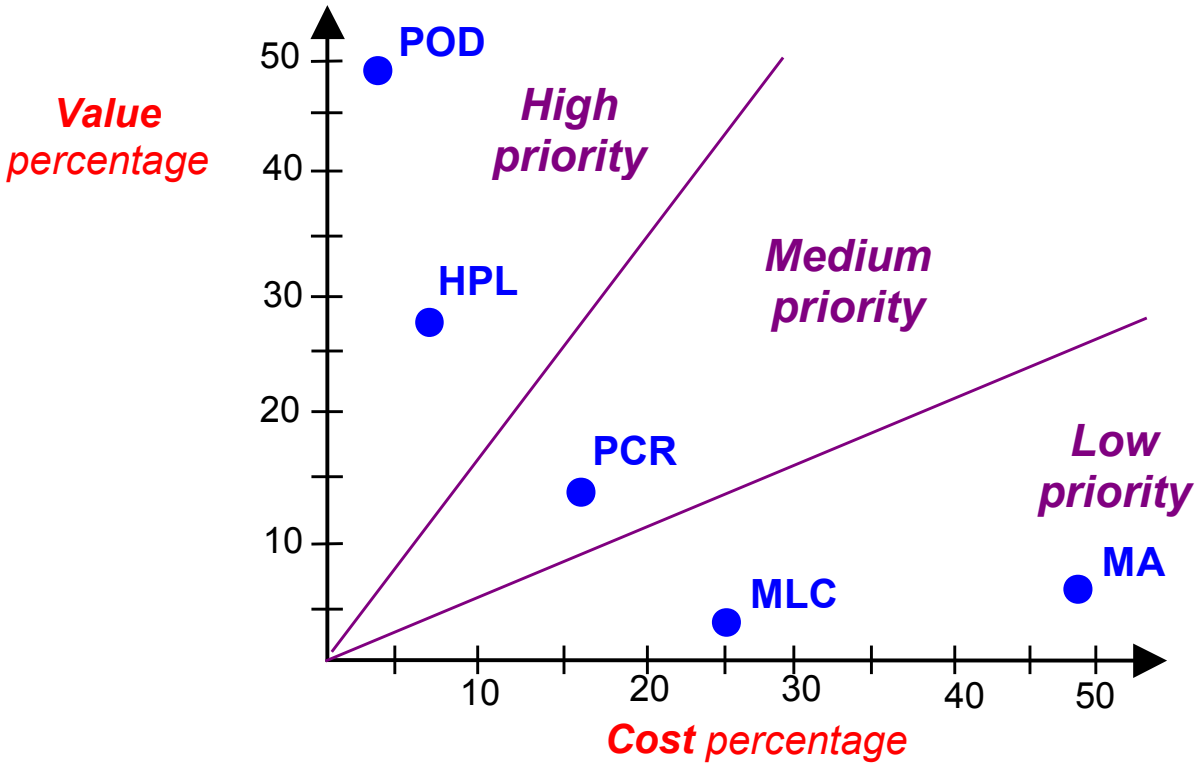
- Elicited & evaluated reqs must be assigned priorities
  - conflict resolution
  - resource limitations (budget, personnel, schedules)
  - **incremental development**
  - replanning due to unexpected problems

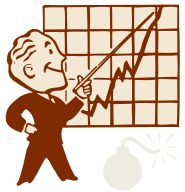
- Some principles for effective req prioritization
  (1) by ordered levels of equal priority, in small number
  (2) relative levels ("higher than", ...)
  (3) comparable reqs: same granularity, same abstraction level
  (4) reqs not mutually dependent (one can be kept, another dropped)
  (5) agreed by key players



Value-cost prioritization

- Systematic technique, meets principles (1) - (3)
- Three steps:
  1. Estimate relative contribution of each req to project's **value**
  2. Estimate relative contribution of each req to project's **cost**
  3. Plot the **value-cost diagram**



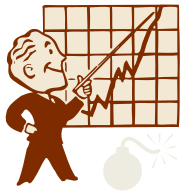


Estimating relative contributions of requirements to project value & cost

- AHP technique from Decision Theory

  ("Analytic Hierarchy Process", [Saati, 1980])

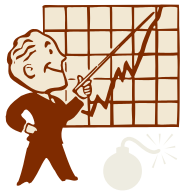
- Determines in what proportion each req R_1, ..., R_N contributes to criterion *Crit*
- Applied twice: *Crit* = value, *Crit* = cost
- Two steps:
  1. Build **comparison matrix**:
     estimates how R_i's contribution to *Crit* compares to R_j's
  2. Determine how *Crit* distributes among all R_i



AHP, Step 1: Compare requirements pairwise

- Scale for comparing R_i's contribution to *Crit* to R_j's:
  - **1**: contributes equally
  - **3**: contributes slightly more
  - **5**: contributes strongly more
  - **7**: contributes very strongly more
  - **9**: contributes extremely more
- In comparison matrix, $R_{ji} = 1 / R_{ij}$ $(1 \le i, j \le N)$

Crit: **value**	Produce optimal date	Handle preferred locations	Parameterize conflict resolution strategy	Multi-lingual communication	Meeting assistant
Produce optimal date	1	3	5	9	7
Handle preferred locations	1/3	1	3	7	7
Parameterize conflict resolution strategy	1/5	1/3	1	5	3
Multi-lingual communication	1/9	1/7	1/5	1	1/3
Meeting assistant	1/7	1/7	1/3	3	1

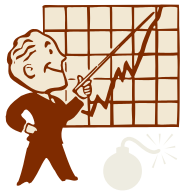


AHP, Step 2: Evaluate how the criterion distributes among all requirements

- Criterion distribution = eigenvalues of comparison matrix
  - 2.a Normalize columns: $R'_{ij} := R_{ij} / \sum_i R_{ij}$
  - 2.b Average accross lines: $Contrib(R_i, Crit) = \sum_j R'_{ij} / N$

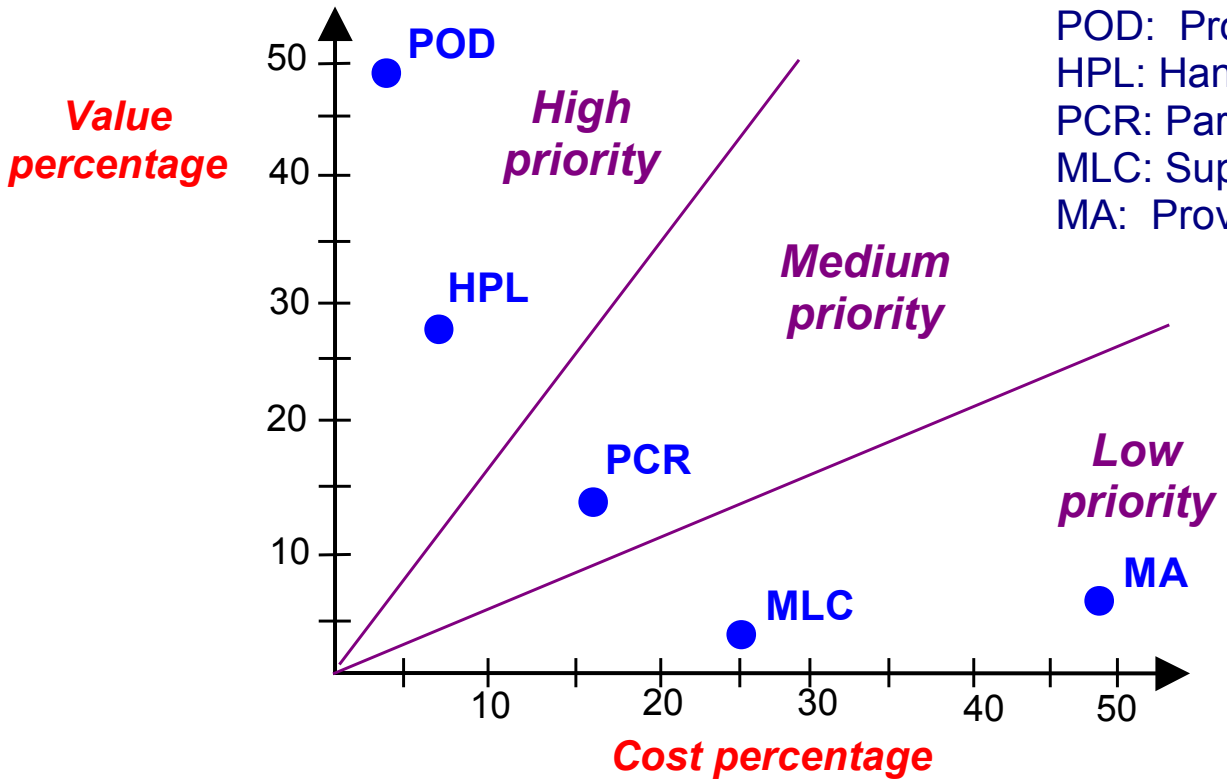
	Produce optim. date	Handle preferred locations	Param. conflict resolution strategy	Multi-lingual communication	Meeting assistant	**Relative value**
Produce optimal date	0.56	0.65	0.52	0.36	0.38	**0.49**
Handle preferred locations	0.19	0.22	0.31	0.28	0.38	**0.28**
Parameterize conflict resolution strategy	0.11	0.07	0.10	0.20	0.16	**0.13**
Multi-lingual communication	0.06	0.03	0.02	0.04	0.02	**0.03**
Meeting assistant	0.08	0.03	0.03	0.12	0.05	**0.07**

- AHP has rules for ensuring consistent estimates & ratios



Plotting contributions on value-cost diagram

- Replay Steps 1 & 2 of AHP with *Crit* = **cost**
- Visualize value/cost contributions on diagram partitioned in selected priority levels



POD: Produce Optimal meeting Dates
HPL: Handle Preferred Locations
PCR: Parameterize Conflict Resolution
MLC: Support Multi-Lingual Communication
MA: Provide Meeting Assistant