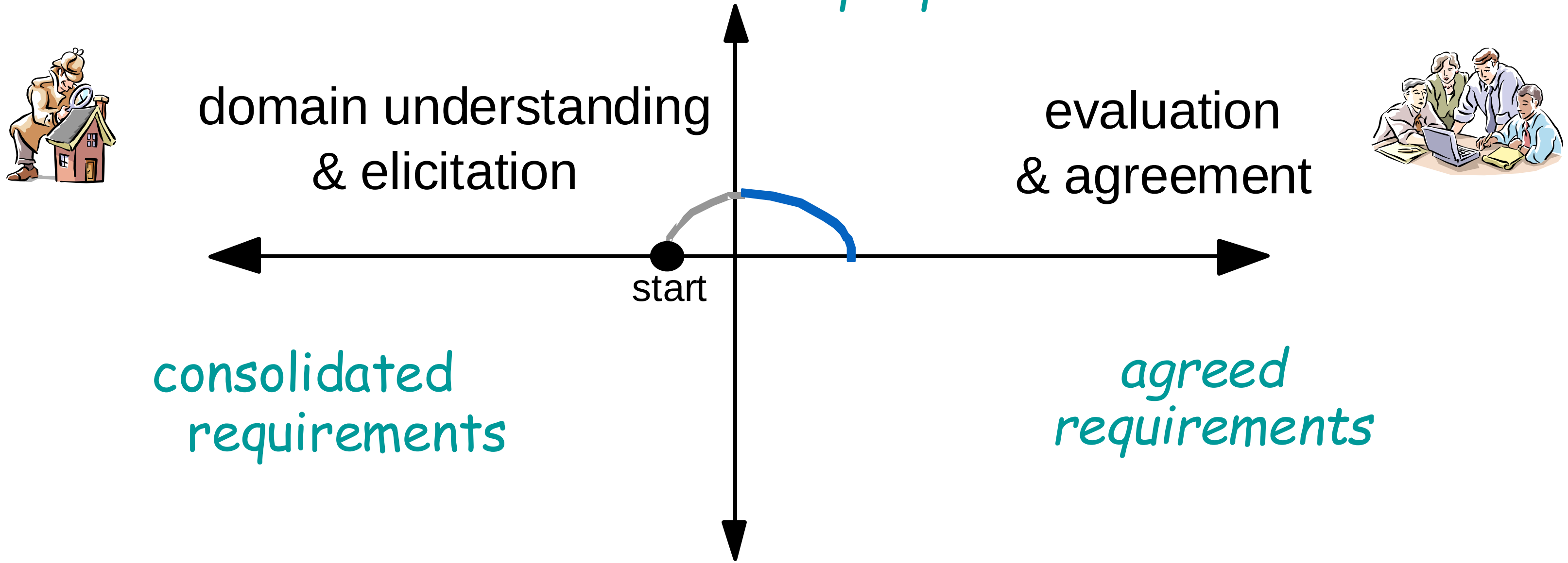
Requirements Evaluation

Mariano Ceccato

mariano.ceccato@univr.it

RE products and processes



alternative proposals

domain understanding
& elicitation

evaluation
& agreement

start

consolidated requirements

agreed requirements

documented requirements

Negotiation-based decision making:

- Identification & resolution of **inconsistencies**
  - conflicting stakeholder viewpoints, non-functional reqs, ...
  - to reach agreement
- Identification, assessment & resolution of system **risks**
  - critical objectives not met, e.g. safety hazards, security threats, development risks, ...
  - to get new reqs for more robust system-to-be
- Comparison of **alternative options**, selection of preferred ones
  - different ways of: meeting same objective, assigning responsibilities, resolving conflicts & risks
- Requirements **prioritization**
  - to resolve conflicts, address cost/schedule constraints, support incremental development

Outline

- Inconsistency management
    - Types of inconsistency
    - Handling inconsistencies
    - Managing conflicts: a systematic process
- Risk analysis
    - Types of risk
    - Risk management
    - Risk documentation
    - DDP: quantitative risk management for RE
- Evaluating alternative options for decision making
- Requirements prioritization

Inconsistency management

- Inconsistency = violation of consistency rule among RD items
- Inconsistencies are highly frequent in RE
  - **inter-viewpoints**: each stakeholder has its own focus & concerns (e.g. domain experts vs. marketing dept)
  - **intra-viewpoint**: conflicting quality reqs (e.g. security vs. usability)
- Inconsistencies must be detected and resolved:
  - not too soon: to allow further elicitation within viewpoint
  - not too late: to allow software development
    (anything may be developed from inconsistent specs)

Types of inconsistency in RE

- **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)

Types of inconsistency in RE

- **Strong conflict**: statements not satisfiable together
  - i.e. logically inconsistent: *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: *potential* conflict
  - MUCH more frequent in RE

  e.g. (staff's viewpoint)

  "patrons shall return borrowed copies within *2* weeks"

  vs. (patron's viewpoint)

  "patrons shall keep borrowed copies as long as needed"

  B: "a patron needing a borrowed copy more than *2* weeks"

Handling inconsistencies

- 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
- Weak, strong conflicts: more difficult, deeper causes
  - Often rooted in underlying 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
  - Example: spiral, negotiation-based reconciliation of *win* conditions

Managing conflicts: a systematic process

Identify overlapping statements

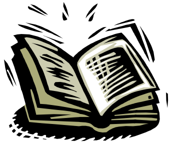
Detect conflicts among them, document these

Generate conflict resolutions

Evaluate resolutions, select preferred

- **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"
  - using conflict patterns
  - formally (theorem proving techniques)

Detected conflicts should be documented



- For later resolution, for impact analysis

  statement in multiple conflicts, most conflicting statements, ...

- Using documentation tools, query tools along *Conflict* links recorded in requirements database

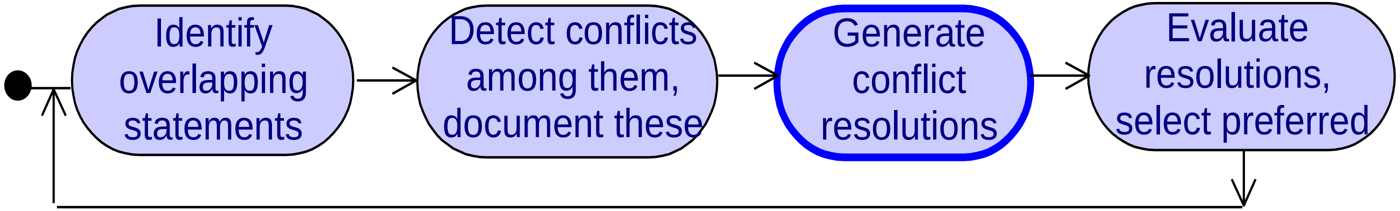
- Or in **interaction matrix**:

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**

S_{ij} = 1: conflict
0: no overlap
1000: no conflict

#Conflicts(S1) = remainderOf (1002 div 1000)
#nonConflictingOverlaps(S1) = quotientOf (1002 div 1000)

Managing conflicts: a systematic process


Identify overlapping statements
Detect conflicts among them, document these
Generate conflict resolutions
Evaluate resolutions, select preferred

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

- To generate candidate resolutions, use:
  - elicitation techniques

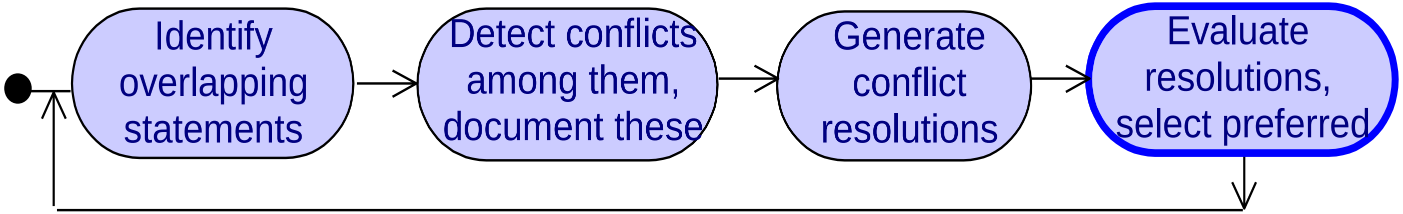
    (interviews, group sessions)
  - resolution tactics

Conflict resolution tactics

- **Avoid** boundary condition
  - e.g. "Keep copies of highly needed books unborrowable"
- **Restore** conflicting statements
  - e.g. "Copy returned within 2 weeks *and then* borrowed again"
- **Weaken** conflicting statements
  - e.g. "Copy returned within 2 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


Identify overlapping statements
Detect conflicts among them, document these
Generate conflict resolutions
Evaluate resolutions, select preferred

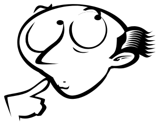
- Evaluation criteria for preferred resolution:
  - contribution to critical non-functional requirements
  - contribution to resolution of *other* conflicts & risks

- See later “Evaluating alternative options”

Outline

- Inconsistency management
  - Types of inconsistency
  - Handling inconsistencies
  - Managing conflicts: a systematic process
- **Risk analysis**
  - **Types of risk**
  - **Risk management**
  - **Risk documentation**
  - **DDP: quantitative risk management for RE**
- Evaluating alternative options for decision making
- Requirements prioritization

What is a risk ?



- Uncertain factor whose occurrence may result in **loss of satisfaction** of a corresponding **objective**

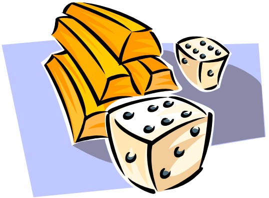
  e.g. a passenger forcing doors opening while train moving
  a meeting participant not checking email regularly

- A risk has:
  - a **likelihood** of occurrence,
  - one or more undesirable **consequences**

  e.g. passengers falling out of train moving with doors open

- Each risk consequence has:
  - a **likelihood** of occurrence if the risk occurs
    - (not to be confused with risk likelihood)
  - a **severity**: degree of loss of satisfaction of objective

Types of RE risk



- **Product-related** risks: negative impact on functional or non-functional objectives of the system

  → failure to deliver services or quality of service

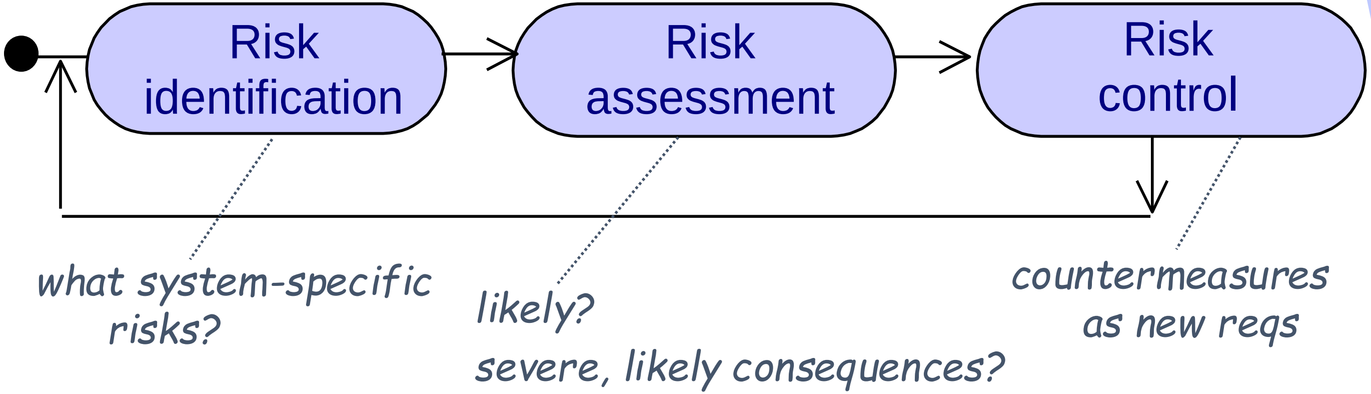
  e.g. security threats, safety hazards

- **Process-related** risks: negative impact on development objectives

  → delayed delivery, cost overruns, ...

  e.g. personnel turnover

RE risk management


Risk identification
Risk assessment
Risk control
what system-specific risks?
likely?
severe, likely consequences?
countermeasures as new reqs

- Risk management is iterative
  - countermeasures may introduce new risks
- Poor risk management is a major cause of software failure
  - natural inclination to conceive over-ideal systems (nothing can go wrong)
  - unrecognized, underestimated risks → incomplete, inadequate reqs

Risk identification: risk checklists

- Instantiation of risk categories to project specifics
  - associated with corresponding req categories
- Product-related risks: req unsatisfaction in functional or quality req categories
  - info inaccuracy, unavailability, unusability, poor response time, poor peak throughput, ...

  e.g. inaccurate estimates of train speed, positions
- Process-related risks: top 10 risks
  - req volatility, personnel shortfalls, dependencies on external sources, unrealistic schedules/budgets, ...
  - poor risk management

  e.g. unexperienced developer team for train system

Risk identification: component inspection

- For product-related risks
- Review each component of the system-to-be: human, device, software component ...
  - can it fail?
  - how?
  - why?
  - what are possible consequences?

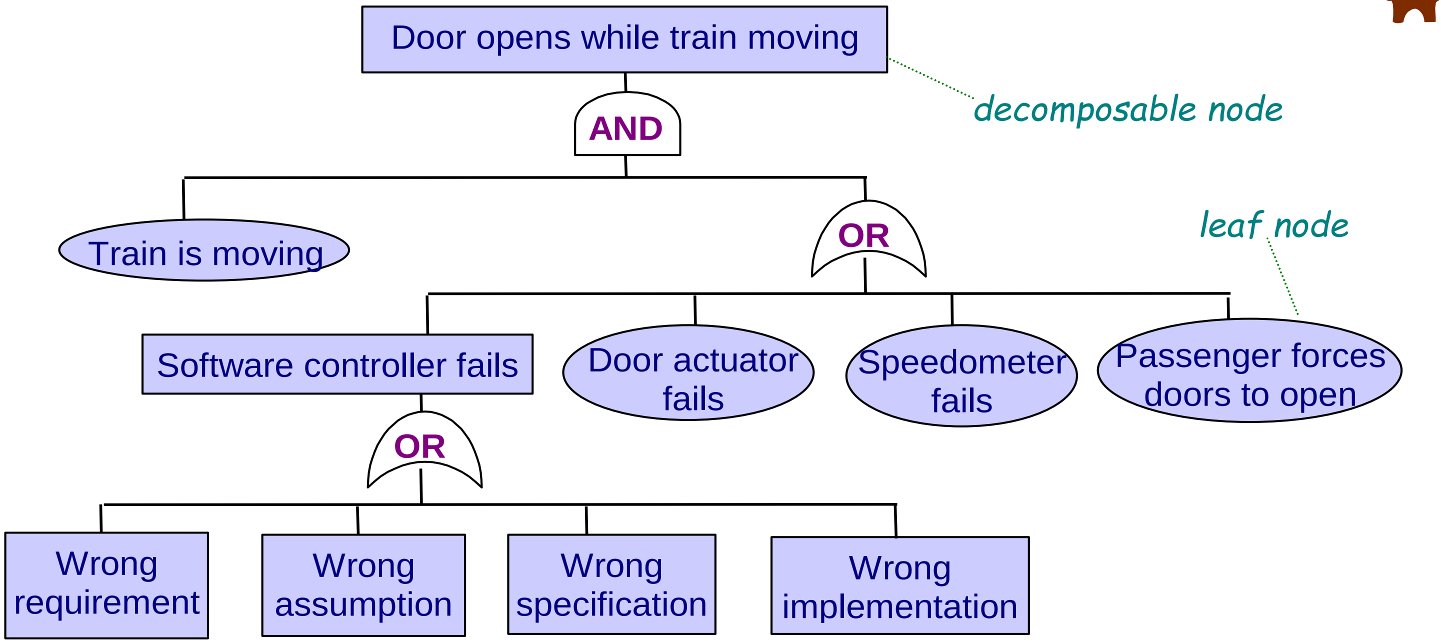
  e.g. on-board train controller, station computer, tracking system, communication infrastructure, ...
- Finer-grained components → more accurate analysis

  e.g. acceleration controller, doors controller, track sensors, ...

Risk identification: risk trees

- Tree organization for causal linking of failures, causes, consequences
  - similar to *fault trees* in safety, *threat trees* in security
- **Failure node** = independent failure event or condition
  - decomposable into finer-grained nodes
- **AND/OR links**: causal links through logical nodes:
  - **AND-node**: child nodes must all occur for parent node to occur as consequence
  - **OR-node**: only one child node needs to occur

Risk tree: example



Building risk trees: heuristic identification of failure nodes

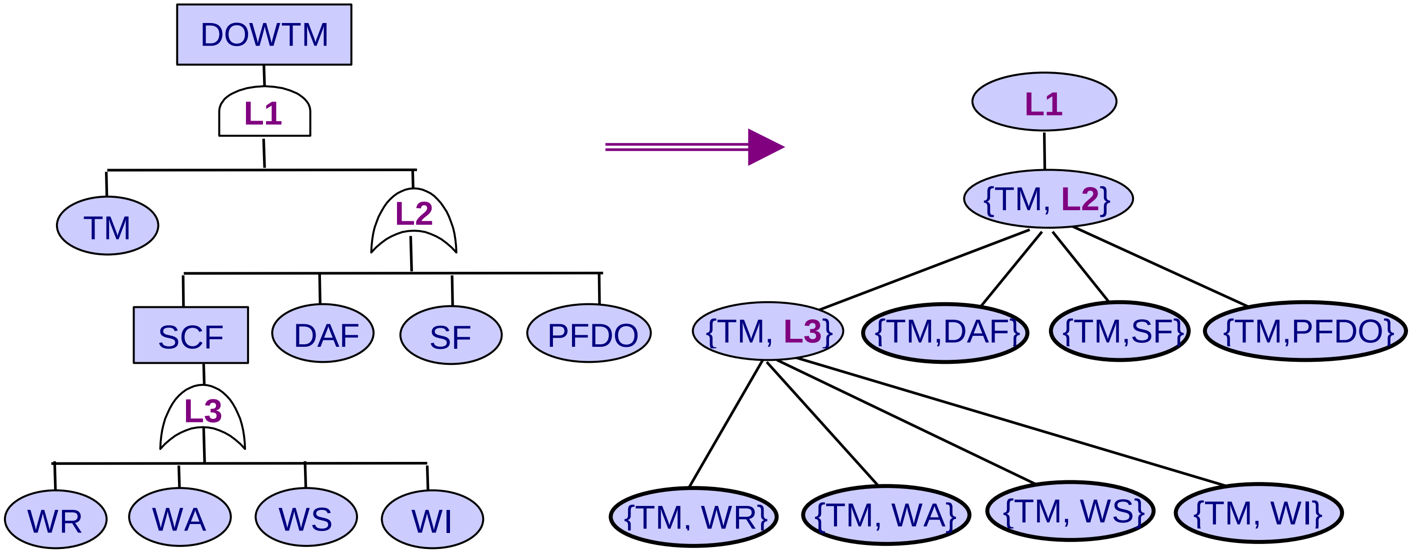
- Checklists, component failure
- **Guidewords** = keyword-based patterns of failure
  - NO: “something is missing”
  - MORE: “there are more things than expected”
  - LESS: “there are fewer things than expected”
  - BEFORE: “something occurs earlier than expected”
  - AFTER: “something occurs later than expected”
- But ... problems frequently due to *combinations* of basic failure events/conditions ...

Analyzing failure combinations: cut set of a risk tree

- **Cut set** of risk tree RT: set of minimal AND-*combinations* of RT's leaf nodes sufficient for causing RT's root node
  - **Cut-set tree** of RT: set of its leaf nodes = RT's cut set
- Derivation of cut-set tree CST of RT:
  - CST's top node := RT's top logical node
  - **If** current CST node is **OR**-node:
    expand it with RT's corresponding alternative child nodes

    If current CST node is **AND**-node:
    expand it in single aggregation of RT's conjoined child nodes
  - Termination when CST's child nodes are all aggregations of leaf nodes from RT

Cut-set tree derivation: example



Cut set = {{TM, WR}, {TM, WA}, {TM, WS}, {TM, WI}, {TM, DAF}, {TM, SF}, {TM, PFDO}}

all combinations of bad circumstances for root risk to occur

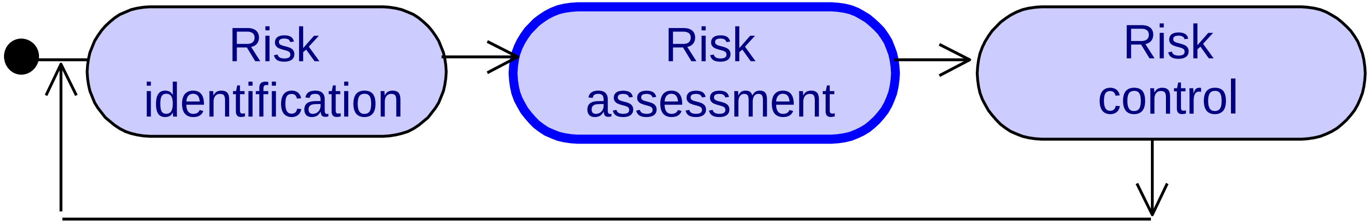
Risk identification: using elicitation techniques

- **Scenarios** to point out failures from WHAT IF questions
    - interactions not occurring
    - interactions occurring too late
    - unexpected interactions (e.g. under wrong conditions), ...

- **Knowledge reuse**: typical risks from similar systems

- **Group sessions** focused on identification of project-specific risks

Risk assessment

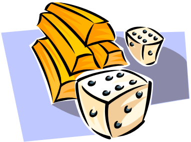


- **Goal**: assess likelihood of risks + severity, likelihood of consequences, to control high-priority risks
- **Qualitative** assessment: use qualitative estimates (levels)
  - for **likelihood**: {very likely, likely, possible, unlikely, ...}
  - for **severity**: {catastrophic, severe, high, moderate, ...}

→ risk *likelihood-consequence* table for each risk

→ risk comparison/prioritization on severity levels

Qualitative risk assessment table: example



Risk: “Doors open while train moving”

	Risk likelihood		
Consequences	**Likely**	**Possible**	**Unlikely**
Loss of life	*Catastrophic*	*Catastrophic*	*Severe*
Serious injuries	*Catastrophic*	*Severe*	*High*
Train car damaged	*High*	*Moderate*	*Low*
#passengers decreased	*High*	*High*	*Low*
Bad airport reputation	*Moderate*	*Low*	*Low*

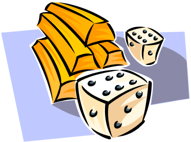
Likelihood level

Severity level

☺ Easy to use

☹ Limited conclusions: coarse-grained, subjective estimates
likelihood of consequences not considered

Risk assessment



- **Quantitative** assessment: use numerical estimates
  - for likelihoods: {0, 0.1, 0.2, ..., 0.9, 1.0} *probability values*

    or {0-0.3, 0.3-0.5, 0.5-0.7, 0.7-1.0} *probability intervals*
  - for severity: scale from 1 to 10

→ **Risk exposure** for risk *r* with independent consequences *c*:

$$\text{Exposure}(r) = \sum_c \text{Likelihood}(c) \times \text{Severity}(c)$$

→ Risk comparison/prioritization based on exposures

(with risks weighted by their likelihood)

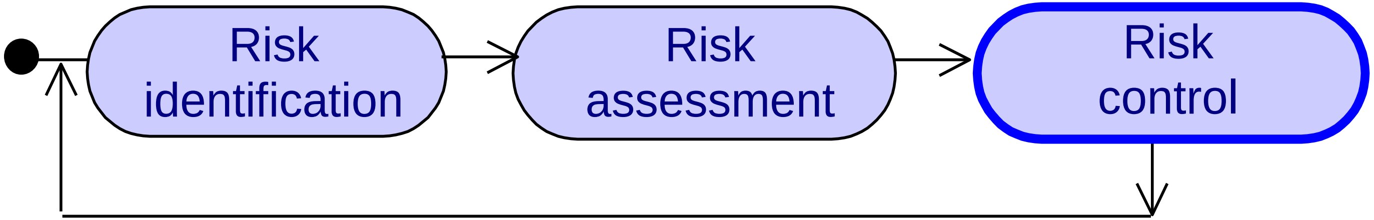
☺ Finer-grained than qualitative assessment

☹ Sill subjective estimates: not grounded on system phenomena

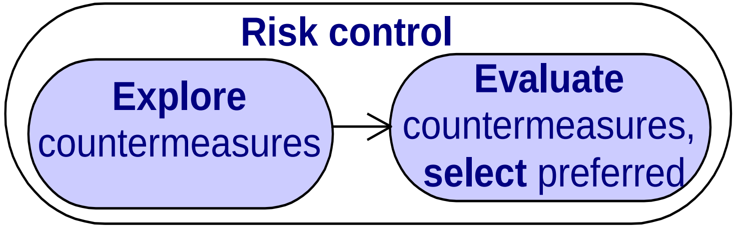
→ to be elicited from domain experts

or data collection from accumulated experiments

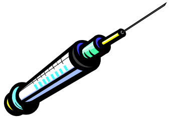
Risk control



- **Goal**: Reduce high-exposure risks through countermeasures
  - yields new or adapted requirements
  - should be cost-effective
- Cf. conflict management:



Exploring countermeasures

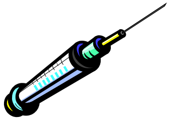


- Using elicitation techniques
  - interviews, group sessions
- Reusing known countermeasures

  e.g. generic countermeasures to top 10 risks

  - simulation ❼poor performance
  - prototyping, task analysis ❼poor usability
  - use of cost models ❼unrealistic budgets/schedules
- Using risk reduction tactics

Risk reduction tactics



- **Reduce risk likelihood**: new reqs to ensure significant decrease

  e.g. “Prompts for driver reaction regularly generated by software”

- **Avoid risk**: new reqs to ensure risk may never occur

  e.g. “Doors may be opened by software-controlled actuators only”

- **Reduce consequence likelihood**: new reqs to ensure significant decrease of consequence likelihood

  e.g. “Alarm generated in case of door opening while train moving”

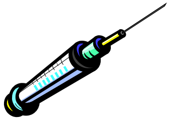
- **Avoid risk consequence**: new reqs to ensure consequence may never occur

  e.g. “No collision in case of inaccurate speed/position estimates”

- **Mitigate risk consequence**: new reqs to reduce severity of consequence(s)

  e.g. “Waiting passengers informed of train delays”

Selecting preferred countermeasures



- Evaluation criteria for preferred countermeasure:
  - contribution to critical non-functional requirements
  - contribution to resolution of *other* risks
  - cost-effectiveness

- Cost-effectiveness is measured by **risk-reduction leverage**:

$$RRL_{r,cm} = \frac{EXP_r - EXP_{r|cm}}{Cost_{r,cm}}$$

Exp$_r$: exposure of risk *r*
Exp$_{r|cm}$: new exposure of *r* if countermeasure *cm* is selected

❼Select countermeasures with highest RRLs

- refinable through cumulative countermeasures & RRLs

Risks should be documented

- To record/explain **why** these countermeasure reqs, to support system evolution
- For each identified risk:
    - conditions/events for occurrence
    - estimated likelihood
    - possible causes & consequences
    - estimated likelihood & severity of each consequence
    - identified countermeasures + risk-reduction leverages
    - selected countermeasures

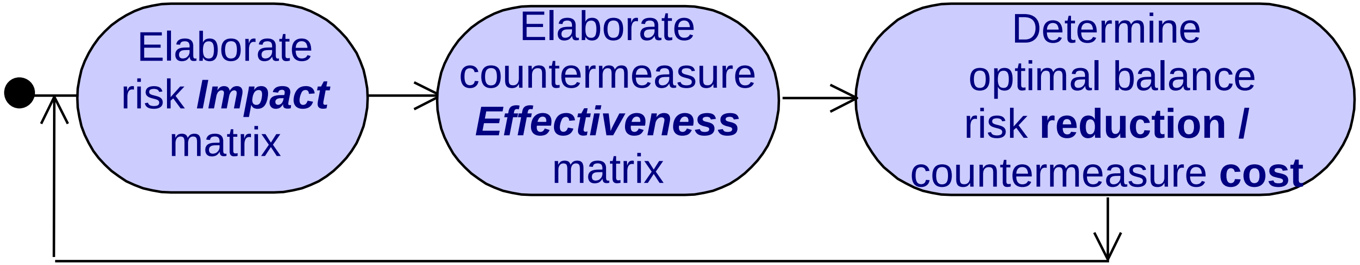
    ≅ annotated risk tree

Requirements evaluation: outline

- Inconsistency management
  - Types of inconsistency
  - Handling inconsistencies
  - Managing conflicts: a systematic process
- Risk analysis
  - Types of risk
  - Risk management
  - Risk documentation
  - **DDP: quantitative risk management for RE**
- Evaluating alternative options for decision making
- Requirements prioritization

DDP: quantitative risk management for RE

- DDP = Defect Detection Prevention
- Technique & tool developed at NASA
- Quantitative support for *Identify-Assess-Control* cycles
- Three steps:


Elaborate risk ***Impact*** matrix
Elaborate countermeasure ***Effectiveness*** matrix
Determine optimal balance risk **reduction /** countermeasure **cost**

Step 1: Elaborate the *Impact* matrix

- Build a **risk-consequence table** with domain experts for:
  - prioritizing risks by critical impact on all objectives
  - highlighting the most risk-driving objectives

- For each objective *obj*, risk *r:*

  $\text{Impact}(r, obj) = [0,1]$ estimated <u>loss of satisfaction</u> of *obj* by *r*

  0 (no loss) ❼1 (total loss)

- Last line, for each risk *r*:

  $$\text{Criticality}(r) = \text{Likelihood}(r) \times \sum_{obj} (\text{Impact}(r, obj) \times \text{Weight}(obj))$$

- Last column, for each objective *obj*:

  $$\text{Loss}(obj) = \text{Weight}(obj) \times \sum_{r} (\text{Impact}(r, obj) \times \text{Likelihood}(r))$$

Impact matrix: example for library system

	Risks				
Objectives	Late returns (likelihood: 0.7)	Stolen copies (likelihood: 0.3)	Lost copies (likelihood: 0.1)	Long loan by staff (likelihood: 0.5)	**Loss obj.**
Regular availability of book copies (weight: 0.4)	0.30	0.60	0.60	0.20	**0.22**
Comprehensive library coverage (weight: 0.3)	0	0.20	0.20	0	**0.02**
Staff load reduced (weight: 0.1)	0.30	0.50	0.40	0.10	**0.04**
Operational costs decreased (weight: 0.2)	0.10	0.30	0.30	0.10	**0.05**
Risk criticality	**0.12**	**0.12**	**0.04**	**0.06**	

$\text{Criticality}(r) = \text{Likelihood}(r) \times \sum_{obj} (\text{Impact}(r, obj) \times \text{Weight}(obj))$

$\text{Loss}(obj) = \text{Weight}(obj) \times \sum_{r} (\text{Impact}(r, obj) \times \text{Likelihood}(r))$

Step 2: Elaborate the *Effectiveness* matrix

- Build a **risk-countermeasure table** with domain experts for:
  - estimating risk reduction by alternative countermeasures
  - highlighting most globally effective countermeasures

- For each countermeasure *cm*, weighted risk *r:*

$\text{Reduction}(cm, r) = [0,1]$ estimated reduction of *r* if *cm* applied

0 (no reduction) ❼1 (risk elimination)

- Last line, for each risk *r*:

$$\text{combinedReduction}(r) = 1 - \prod_{cm} (1 - \text{Reduction}(cm, r))$$

- Last column, for each countermeasure *cm*:

$$\text{overallEffect}(cm) = \sum_{r} (\text{Reduction}(cm, r) \times \text{Criticality}(r))$$

Effectiveness matrix: example for library system

	Weighted risks				
Countermeasures	Late returns (likelihood: 0.7)	Stolen copies (likelihood: 0.3)	Lost copies (likelihood: 0.1)	Long loan by staff (likelihood: 0.5)	**Overall effect of countermeasure**
Email reminder sent	0.70	0	0.10	0.60	**0.12**
Fine subtracted from registration deposit	0.80	0	0.60	0	**0.12**
Borrower unregistration + insertion on black list	0.90	0.20	0.80	0	**0.16**
Anti-theft device	0	1	0	0	**0.12**
Combined risk reduction	**0.99**	**1**	**0.93**	**0.60**	

$$\text{combinedReduction}\,(r) = 1 - \Pi_{cm}\,(1 - \text{Reduction}\,(cm, r))$$

$$\text{overallEffect}\,(cm) = \textstyle\sum_r\,(\text{Reduction}\,(cm, r) \times \text{Criticality}\,(r))$$

Step 3: Determine optimal balance risk reduction vs. countermeasure cost

- Cost of each countermeasure *cm* to be estimated with domain experts
- DDP can then visualize ...
  - risk balance charts: residual impact of each risk on all objectives if *cm* is selected
  - optimal combinations of countermeasures for risk balance under cost constraints
    - simulated annealing search for near-optimal solutions
    - optimality criterion can be set by user

      e.g. "maximize satisfaction of objectives under this cost threshold"

      "minimize cost above this satisfaction threshold"

Requirements evaluation: outline

- Inconsistency management
    - Types of inconsistency
    - Handling inconsistencies
    - Managing conflicts: a systematic process
- Risk analysis
    - Types of risk
    - Risk management
    - Risk documentation
    - DDP: quantitative risk management for RE
- **Evaluating alternative options for decision making**
- Requirements prioritization

Evaluating alternative options for decision making

- The RE process raises multiple alternative options of different types
  - alternative ways of satisfying a system objective
  - alternative assignments of responsibilities among system components
  - alternative resolutions of a conflict
  - alternative countermeasures to reduce a risk
- Preferred alternatives must be negotiated, selected:
  - agree on evaluation criteria (e.g. contribution to NFRs)
  - compare options according to criteria
  - select best option
- *Qualitative* or *quantitative* reasoning techniques for this

Qualitative reasoning for evaluating options

- Goal: determine qualitative contribution of each option to important non-functional requirements (NFRs):

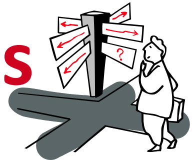
  very positively (++), positively (+), negatively (-), very negatively (--)

- Example: meeting scheduling

	Non-functional requirements		
Options	Fast response	Reliable response	Minimal inconvenience
Get constraints by email	-	+	-
Get constraints from e-agenda	+ +	- -	+ +

- Qualitative labels "+", "-" on higher-level NFRs are obtained by bottom-up propagation from lower-level reqs in goal-subgoal refinement/conflict graph
- Given "+", "-" contributions of each option to lowest-level reqs, option with best contribution to critical high-level NFRs is taken

Quantitative reasoning for evaluating options



- Build a **weighted matrix** for:
  - estimating score of each option on each evaluation criterion (weighted by relative importance)
  - selecting option with highest overall score on all criteria
- For each option *opt*, criterion *crit:*

  Score (*opt*, *crit*) = [0,1] estimated score percentage of *opt* on *crit*

  0 --> 1, Y/100 means "*crit* satisfied in Y% of cases"
- Last line, for each option *opt*:

  totalScore (*opt*) = $\sum_{crit}$(Score (*opt*, *crit*) × Weight (*crit*))

Evaluation criteria (NFRs)	**Significance weighting**	**Option scores**	
		Get constraints by email	Get constraints from e-agenda
Fast response	*0.30*	*0.50*	*0.90*
Reliable response	*0.60*	*0.90*	*0.30*
Minimal inconvenience	*0.10*	*0.50*	*1.00*
TOTAL	**1.00**	**0.74**	**0.55**

Requirements evaluation: outline

- Inconsistency management
    - Types of inconsistency
    - Handling inconsistencies
    - Managing conflicts: a systematic process
- Risk analysis
    - Types of risk
    - Risk management
    - Risk documentation
    - DDP: quantitative risk management for RE
- Evaluating alternative options for decision making
- **Requirements prioritization**

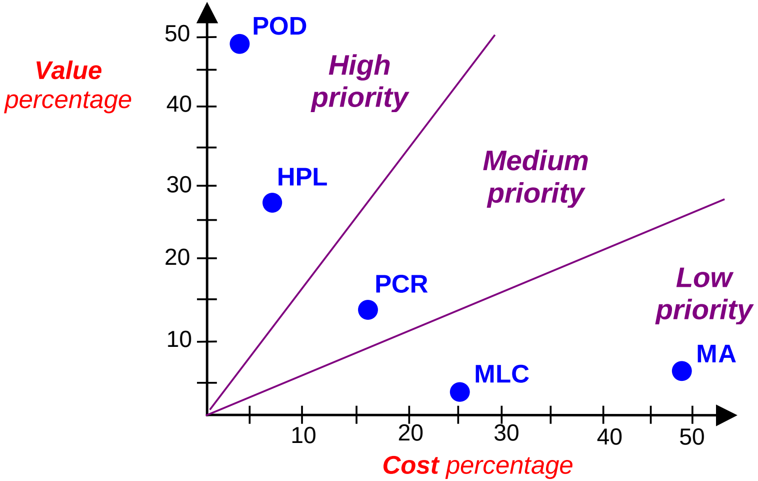
Requirements prioritization

- 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) qualitative & 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
- Too early ranking at elicitation time might be subjective

  ❼risk of inadequate, inconsistent results

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 contributions on **value-cost diagram**: shows what req fits what priority level according to value-cost tradeoff



Estimating relative contributions of requirements to project value & cost

- AHP technique from Decision Theory: Analytic Hierarchy Process
- 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, $R_{ji} = 1 / R_{ij}$ $(1 \leq i, j \leq N)$

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

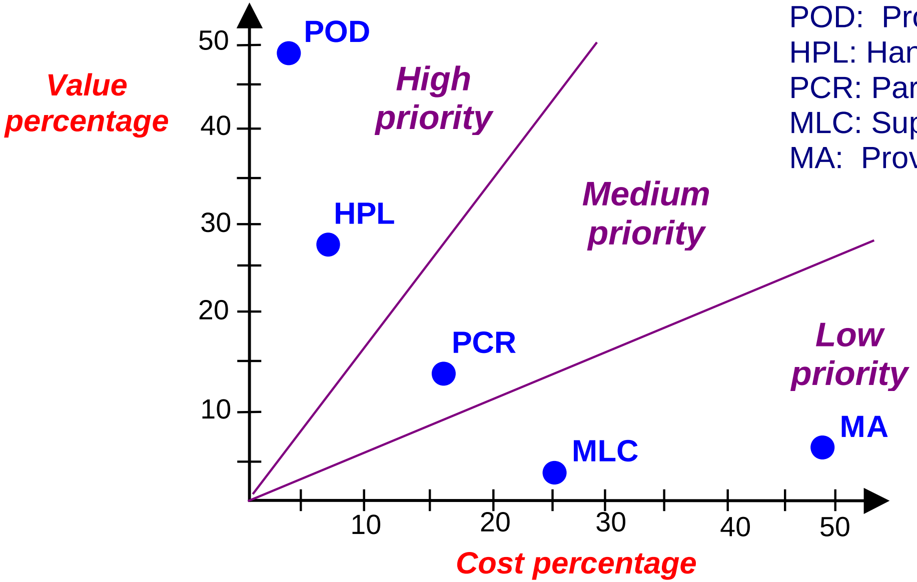
2.a Normalize columns: $R'_{ij} := R_{ij} / \sum_i R_{ij}$

2.b Average across lines: $\mathrm{Contrib}(R_i, \mathit{Crit}) = \sum_j R'_{ij} / N$

	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**

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

Requirements evaluation: summary

- Inconsistencies are frequent during req acquisition
  - For clashes in terminology, designation, structure: a glossary of terms is best
  - For weak, strong conflicts: variety of techniques & heuristics to support cycles “identify overlaps, detect conflicts, generate resolutions, select preferred”
- Product-/process-related risks must be carefully analyzed
  - Loss of satisfaction of system/development objectives
  - Variety of techniques for risk identification, incl. risk trees & their cut set
  - Likelihood of risks & consequences + severity need be assessed, qualitatively or quantitatively, with domain experts
  - Heuristics for exploring countermeasures, selecting cost-effective ones
  - DDP: an integrated quantitative approach for RE risk management

Requirements evaluation: summary

- Alternative options need be evaluated for selecting preferred, agreed ones
  - Different types, incl. resolutions of conflicts & risks
  - Qualitative or quantitative reasoning for this (weighted matrices)
- Requirements must be prioritized
  - Due to resource limitations, incremental development
  - Constraints for effective prioritization
  - AHP-based value-cost prioritization: a systematic technique