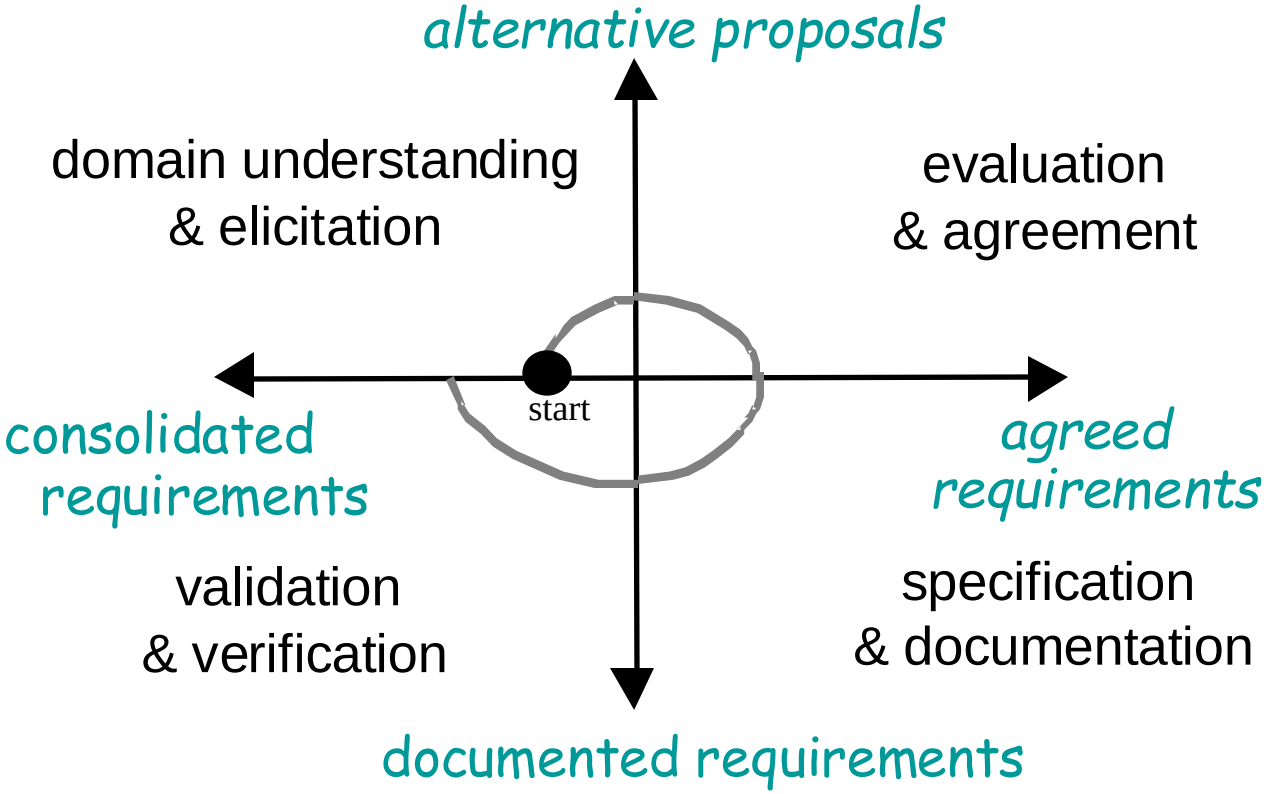
Requirements Quality Assurance

Mariano Ceccato

mariano.ceccato@univr.it

The RE process



Quality assurance at RE time

- Errors & flaws in requirements documents ...
  - the most expensive, numerous, persistent, dangerous
  - different types: omission, contradiction, inadequacy, ambiguity, unmeasurability, noise, overspecification, poor structuring, opacity

- QA task 1: **find** as many of these as possible in the RD by ...
  - **validation**: adequacy of RD items wrt real needs ?
  - **verification**: completeness, consistency of RD items ?
  - **checks** for other target qualities
    - non-ambiguity, measurability, feasibility, good structure, ...?
- QA task 2: **report** defects, **analyze** their causes, fix them

Quality assurance at RE time

- Various products resulting from this phase
  - consolidated RD
  - acceptance test data, prototype
  - development plan
  - project contract

- Complements the requirements evaluation phase:
  - later, more accurate analysis
  - along selected options
  - on detailed specifications

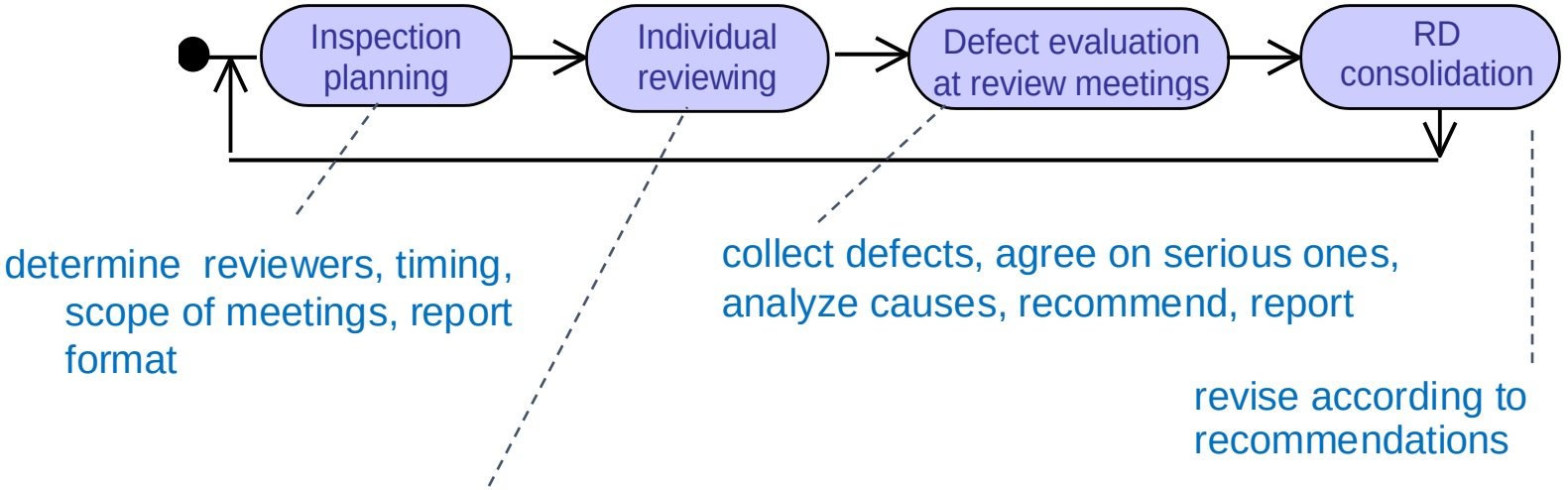
Outline

- Requirements inspections and reviews
  - The requirements inspection process
  - Inspection guidelines
  - Requirements inspection checklists
- Queries on a specification database
- Requirements validation by specification animation

Requirements inspections and reviews

- Simple idea:
  - ask selected people to inspect the RD individually
  - meet to agree on list of problems to be fixed
- Various forms
  - **walkthroughs**: internal inspection by project members
  - **more structured**: external reviewers, well-prepared meetings, inspection reports, etc.
- Fairly common for source code (cf. Fagan inspections)
- Empirical studies showed it effective for requirements too

A structured inspection-based QA process



- **Free mode**: no directive on *where* to find *what*
- **Checklist-based**: specific issues, defect types, RD parts
- **Process-based**: specific role for each reviewer, specific procedure, defect type, focus, analysis technique -> **more effective**

Inspection guidelines

- Inspection report should be...
  - informative, accurate, constructive
    - (no opinion, no offending comment)
  - standard structure with room for free comments, lightweight
    - may guide individual reviewing
- Inspectors should be ...
  - independent from RD authors
  - representative of all stakeholders, different background
- Inspection should come not too soon, not too late
  - shorter, repeated meetings are more effective
- More focus required on critical parts
  - the more defects in some part, the more scrutiny required

Inspection checklists

- Aim: guide defect search towards specific issues
- **Defect-driven** checklists
  - list of generic questions structured by defect type
- **Quality-specific** checklists
  - specialize defect-driven questions to specifc NFR categories
    - safety, security, performance, usability, ...
  - sometimes based on guidewords
  - often focussed on omissions
- **Domain-specific** checklists
  - further specialization to domain concepts & operations
  - for increased guidance in defect search
- **Language-based** checklists
  - specialize defect-driven checklists to spec language constructs
  - support automation
    - richer spec languages => more sophisticated checks

Defect-driven checklists: a sample

- (Omissions?)
  - Is this precisely defined somewhere ?
  - Is there any hidden assumption required for this ?
  - Is the rationale for this made explicit somewhere ?
- (Contradictions?)
  - Is this consistent with stated objectives or constraints?
- (Inadequacies?)
  - Does this formulate what stakeholders really want?
- (Ambiguities?)
  - Can this be interpreted differently ?
  - Any other statement using this term with a different meaning?
- (Unmeasurability?)
  - Is there a fit criterion for this quality requirement?
  - Is this stated in a way discriminating from alternative options?

Defect-driven checklists: a sample

- (Noises?)
  - Is this relevant to system objectives or constraints?
  - Does the negation of this make any sense?
  - Any other statement using this under different terms?
- (Overspecification?)
  - Does this entail premature design choices - any sensible alternative?
- (Unfeasibility?)
  - Is this implementable within reasonable time/budget?
- (Unintelligibility?)
  - Is this comprehensible by anyone needing to use it?
- (Remorse ?)
  - Has this concept been used already before this definition?

Defect-driven checklists: a sample

- (Poor structuring?)
  - Is the structuring rule for those sections apparent, and natural ?
  - Any item related to this and described elsewhere?
  - Is this covering unrelated requirements?
  - Is this mixing requirements and assumptions altogether?
  - Is this mixing requirements and domain properties?
  - Any unnecessary mixing of functional & non-functional concerns?
- (Opacity?)
  - Any inderdependent items whose dependencies are not made visible?
- (Poor modifiability?)
  - Would any change to this require propagation throughout major portions of the RD?

Quality-specific checklists: a sample

- Any unspecified response to out-of-range input values?
- Any unspecified response to receiving expected input values too early; too late; or never?
- Is the OR of input conditions on this operation a tautology?
- Any recovery operation in case of input saturation?
- Any output producible faster than it can be consumed?
- Any input from sensors unused by the software controller?
- Does every state sequence from a hazardous state lead to a low-risk state?
- Is there a data consistency check before a decision is made from these data?

Language-specific checklists: a sample for statement templates

- Is this statement identifier used consistently throughout the RD?
- Does the indicated type for this statement correctly refer to a requirement, assumption, domain property, or definition?
- Is this fit criterion defined in terms of measurable quantities & measurement protocols?
- Is this rationale consistent with the stated system objectives?
- Is this priority consistent with the priority of other requirements this requirement contributes to?

Checking decision tables

N input conditions

Train receives outdated acceleration command	T	T	T	T	F	F	F
Train enters station block at speed ≥ *X* mph	T	T	F	F	T	F	F
Preceding train is closer than *Y* yards	T	F	T	F	F	T	F
Full braking activated	X		X				
Alarm generated to station computer	X	X	X	X			

number of case combinations < 2^N => missing cases

number of case combinations > 2^N => redundant cases

Language-specific checklists: intra- and inter-diagram checks

- Any output flowing out this DFD operation not flowing out any downstream suboperation in the DFD refining this operation?
- Any relationship in ER diagram with inadequate multiplicities?
- Any input (output) of this DFD operation not declared in ER diagram?
- Is this event trace in ET diagram covered by a path in SM diagram?
- Any non-final state in this SM diagram with no outgoing transition?
- Any dynamic attribute/relationship defining this state in SM diagram not declared in ER diagram?

- **Can be automated by tools**

Requirements inspections and reviews: strengths & limitations

☺ Experienced to be even more effective than code inspection

- process-based mode with mix of defect-based, quality-specific, domain-specific, language-specific checklists

☺ Wide applicability

- any defect type, any spec format

☹ Burden & cost of inspection process

- size of inspection material
- required time/cost for external inspectors, review meetings

☹ No guarantee of not missing important defects

Requirements quality assurance: outline

- Requirements inspections and reviews
  - The requirements inspection process
  - Inspection guidelines
  - Requirements inspection checklists
- Queries on a specification database
- Requirements validation by specification animation
- Formal checking
  - Language checks
  - Dedicated consistency and completeness checks
  - Model checking
  - Theorem proving

Queries on a specification database

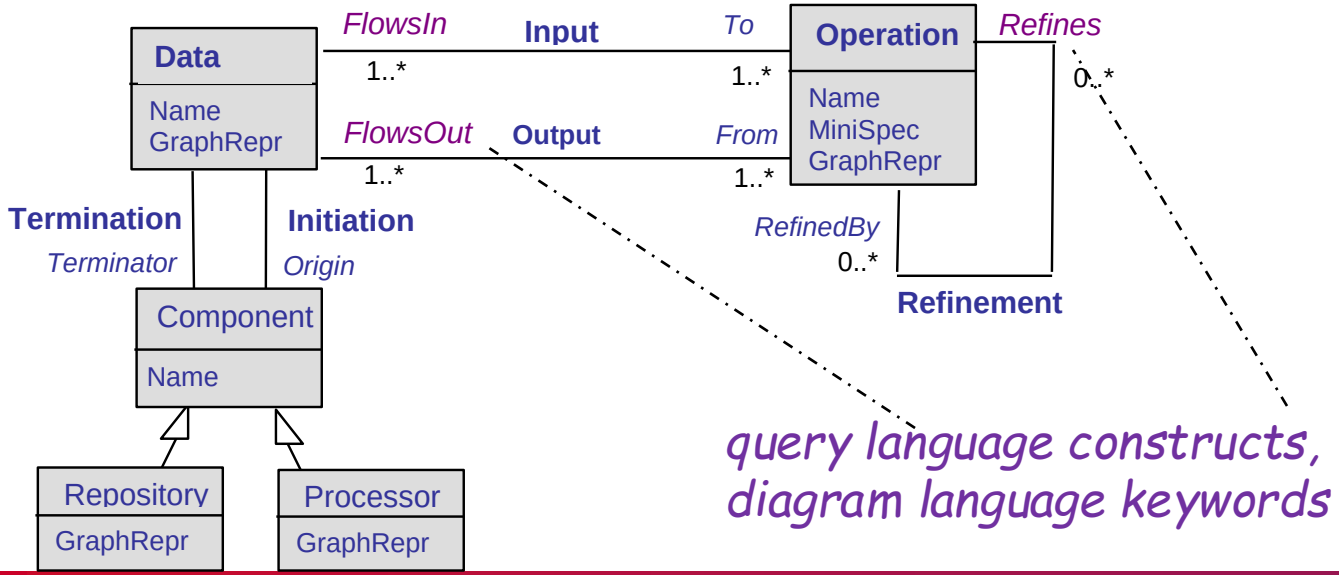
- For diagrammatic specs
- Specs maintained in requirements database
  - logical schema derived from structure of diagram language
  - DB engine can be generated from meta-spec of diagram language
- Queries capture checks for structural consistency intra- or inter-diagrams (automate checklist questions)
  - e.g. Any output flowing out of this DFD operation and not flowing out of any of the refining sub-operations ?

ER query language, constructs = diagram keywords

```
set out-data  =  Data
      which FlowsOut Operation with Operation.Name = 'myOperation'
            and which not FlowsOut ref-ops
      where set ref-ops = Operation
            which Refines Operation with Operation.Name = 'myOperation'
```

Queries on a specification database

- Diagram-specific query engine can be generated from ER meta-spec of diagram language
- Press-button mode commonly used by CASE tools
  - precooked queries for violation of standard consistency rules



Requirements validation by specification animation

- Aim: check requirements adequacy wrt actual needs
- Approach 1: show concrete interaction scenarios in action
  - 👍 Enactment tools can be used on ET diagram
  - 👎 Scenario coverage problem
- Approach 2: use spec animation tool
  - 1. Extract or generate executable model from specs
  - 2. **Simulate** system behavior from this model
    - submit stimulus events to mimic environment behavior
    - watch model's response
  - 3. **Visualize** the simulation while running
  - 4. Get user's feedback
- Model-based (unlike prototyping)
- Many animators available for variety of spec languages
  - SCR, LTSA, Statemate, Jaza, Octopus, etc

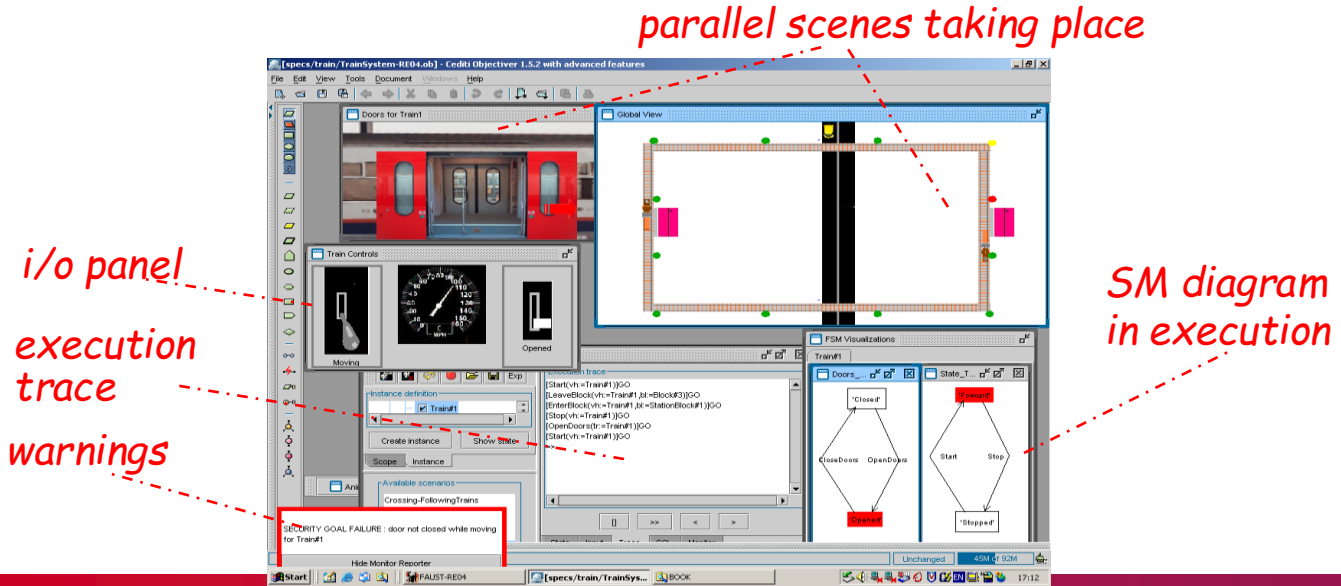
Simulating the model

- 1. Define animation scope
  - what should be shown for adequacy checking?
- 2. Produce an executable model
  - e.g. compile specs into parallel state machines
- 3. Instantiate model to selected component instances
  - e.g. 2 trains, 3 platforms?
- 4. Run the **NextState** simulation function on selected instances in response to environment events

```
NextState (E, CS):        % E: event set,  SC: state configuration %
      T =  Triggered (E);      % set of transitions triggered by events in E %
      T' =  Enabled (T, CS) ∩ Permitted (T, CS) ∩ Consistent (T);
      if T' ≠ ∅ then
            E' = EvGenerated (T');      % events generated by T' %
            CS' = newConfig (T', CS);
            NextState (E', CS')
```

Visualizing the simulation

- To submit environment events and watch model's reaction while simulation running:
  - **textual format**: input commands, execution traces
  - **diagrammatic**: input event selection, tokens progressing along model diagram
  - **domain scenes**: input panels, animated scene in the environment



Requirements animation: strengths & limitations

- ☺ Best way to check adequacy wrt *real* needs, *actual* environment
- ☺ Supports stakeholder involvement
  - WYSIWYC: What You See Is What You Check
- ☺ Extendable to animate counterexamples generated by other tools (e.g., model checkers)
- ☺ Animation scenarios can be kept for later replay
  - usable as acceptance test data
- ☹ Coverage?
  - requires careful design of scenarios likely to exibit problems
  - no guarantee of not missing important inadequacies
- ☹ No free lunch: formal spec needed

Requirements quality assurance: summary

- Major concern in view of error diversity, frequency, criticality, cost

- Requirements inspections and reviews
  - any defect type, any spec format
  - more effective if well-planned process & specific checklists
  - limited tool support; less likely to reveal subtle errors

- Queries on a specification database
  - for structural inconsistencies & omissions in semi-formal specs
  - cheap; easy-to-use tools; surface errors

- Requirements validation by specification animation
  - for adequacy & completeness checks
  - formal, executable model required
  - possible stakeholder involvement; subtle errors; but partial