Formal tools for managing inconsistency and change in RE

Aditya K. Ghose
Decision Systems Lab
SITACS
University of Wollongong
Australia
aditya@uow.edu.au
www.uow.edu.au/~aditya
Goals

- Dealing with inconsistent specifications
  - Caused by multiple sets of stakeholders (saying contradictory things)
  - Caused by intrinsic contradictions within stakeholder viewpoints
  - Caused by the (occasionally) competing pulls of functional and non-functional requirements.

- Managing requirements evolution
  - Caused by changing problem domains
  - Caused by stakeholders changing their minds

- Dealing with non-functional requirements
  - Inconsistencies may arise within sets of functional requirements and between functional and non-functional requirements
The ideal specification

- Must tolerate inconsistencies
- Must support a domain-independent facility to make explicit the trade-offs involved when requirements must be discarded to make a specification consistent.
- Must support the distinction between essential and tentative requirements.
- Must make explicit the connection between a requirement and the conditions/assumptions/justifications that its satisfaction is contingent on.
- Must include some abstractions of system behaviour to enable analysis of boundary conditions
The ideal evolution management tool

- Must ensure that evolution steps involve *minimal change* to specifications.
- Must “discard” requirements only after rigorous *trade-off analysis* (which would weigh the cost of discarding the requirement against the cost of ignoring the change request)
- Must never really “discard” requirements, but must retain them in a background store in anticipation of future reuse.
- Must support a *deferred commitment strategy*, i.e., it must delay as much as possible any commitment to an outcome of the change process (since premature choices might turn out to be poor ones given new information)
Requirements specifications: I

\[ \langle D, E_{FR}, T_{FR}, E_{NFR}, T_{NFR} \rangle \] where:

- \( D \) is the domain theory and consists of:
  - A set of domain invariants \( D_{inv} \)
  - A set of domain trajectories \( D_{traj} \)
- A set \( E_{FR} \) of essential functional requirements.
- A set \( T_{FR} \) of tentative functional requirements.
- A set \( E_{NFR} \) of essential non-functional requirements.
- A set \( T_{NFR} \) of tentative non-functional requirements.
- \( E_{FR} \cup E_{NFR} \cup D_{inv} \cup t_i \) is consistent for each trajectory \( t_i \) contained in \( D_{traj} \).
Requirements specifications: II

Language: Any language that comes with a well-defined notion of consistency is applicable (our examples use the formal assertion layer of KAOS).

Consistency: Need not be logical consistency. Weaker notions are applicable. We use two notions of consistency:

- Meta-level: A weak notion that requires all essential requirements to be consistent with domain invariants and domain trajectories for a specification to be deemed consistent.

- Object-level: Our examples use logical consistency in generating maximal consistent subsets (but we could weaken this).
Requirements specifications: III

Set of domain trajectories $D_{traj}$:

- Ideally an oracle capable of generating all domain behaviours
- A pragmatic approximation involves recording critical states and critical trajectories (sequences of critical states)
- Each critical state is a (possibly partial) description of a domain state
- Ideally, all boundary conditions should be captured in the explicitly identified critical states (but we do not have the means to guarantee that this can be done)
- A range of other pragmatic approximations are possible
**Requirements specifications: IV**

$E_{FR}, T_{FR}, E_{NFR}$ and $T_{NFR}$ are sets of *justified requirements*, each of the form $\alpha : \beta$, where:

- $\alpha$ is a (functional or non-functional) goal
- $\beta$ might include the *rationale* for the corresponding goal or *assumptions* about the domain (or other goals) that the satisfaction of the requirement is contingent on

Justified requirements make explicit interactions for which material implication is too strong.
Object-level consistency

Let \( R = \{ \alpha_1 : \beta_1, \ldots, \alpha_n : \beta_n \} \).
Let \( S = \langle (D_{inv}, D_{traj}), E_{FR}, T_{FR}, E_{NFR}, T_{NFR} \rangle \).
\( R \) is \( r \)-consistent with respect to \( S \) iff
\( \alpha_1 \cup \ldots \cup \alpha_n \cup \beta_1 \cup \ldots \cup \beta_n \cup D_{inv} \cup t_i \) is satisfiable for every trajectory \( t_i \in D_{traj} \).
Inconsistency resolution

Two approaches:

- Identifying *minimal sources of inconsistency*. Useful when we need tool support to highlight areas which need “repair”. Note: Each source must be repaired to restore consistency.

- Identifying *maximal consistent subsets*. Useful when we are willing to “discard” requirements (as opposed to repair them) to restore consistency. Note: We may be willing to settle for the first maximal consistent subset that the tool suggests.

- The two notions are inter-definable.
**Min-conflict sets**

A min-conflict set $I$ of a specification $S = \langle (D_{inv}, D_{traj}), E_{FR}, T_{FR}, E_{NFR}, T_{NFR} \rangle$ is any set satisfying the following properties:

- $I \subseteq T_{FR} \cup T_{NFR}$
- $I$ is r-inconsistent relative to $S$
- Any $I' \subset I$ is r-consistent relative to $S$
r-maximal sets

An r-maximal set $M$ for a specification $S = \langle (D_{inv}, D_{tra}), E_{FR}, T_{FR}, E_{NFR}, T_{NFR} \rangle$ is any set satisfying the following properties:

1. $M \subseteq E_{FR} \cup T_{FR} \cup E_{NFR} \cup T_{NFR}$
2. $(E_{FR} \cup E_{NFR}) \subseteq M$
3. $M$ is r-consistent relative to $S$.
4. For every $M'$ such that $M \subset M' \subseteq E_{FR} \cup E_{NFR} \cup T_{FR} \cup T_{NFR}$, $M'$ is r-inconsistent relative to $S$. 
Outcome choice

Multiple r-maximal subsets may exist for a given specification. We apply an outcome choice function to select one of these - only necessary when a (object-level) consistent outcome is required.

\[ c_o : 2^\mathcal{M} \rightarrow \mathcal{M} \]

where \( \mathcal{M} \) is the class of r-maximal sets.
May be determined by priorities amongst individual goals, organizational precedences amongst stakeholders, or other criteria.
Remarks on inconsistency handling

- Integrity constraints (such as \( \neg \alpha \)) can be handled by adding \( \emptyset : \neg \alpha \) as an essential justified requirement
- Most categories of inconsistencies listed in (van Lamsweerde, Darimont, Leiter 1998) are subsumed
- *Goal weakening* and *temporal relaxation* strategies can be incorporated
- Maximality w.r.t. set inclusion can be replaced by maximality w.r.t. set cardinality
- Decision-theoretic extensions possible
A requirements evolution operator

\[ E : \mathcal{R} \times \mathcal{RC} \times \mathcal{OC} \times \mathcal{OP} \times \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{R} \times \mathcal{M} \] where:

- \( \mathcal{R} \) is the class of specifications
- \( \mathcal{M} \) is the class of r-maximal sets
- \( \mathcal{OP} = \{ FR\text{-}addition, FR\text{-}removal, NFR\text{-}addition, NFR\text{-}removal \} \)
- \( \mathcal{L} \) is the first-order language in which requirements and their justifications are represented
- \( \mathcal{RC} \) is the class of revision choice functions
- \( \mathcal{OC} \) is the class of outcome choice functions
Revision choice

In revising specifications, weak (meta-level) consistency must be maintained: essential requirements in the revised specification must be consistent with domain trajectories.
A revision choice function \( c_o \) selects amongst multiple possible (meta-level) consistent outcomes.

\[ c_r : 2^{\mathcal{R}\mathcal{E}} \rightarrow \mathcal{R}\mathcal{E} \]

where \( \mathcal{R}\mathcal{E} \) is the class of (partial) specifications of the form \((EF, EN)\)
where \( EF \) is a set of justified functional requirements and \( EN \) is a set of justified non-functional requirements.
\( c_o \) implements trade-off analysis.
Evolution: I

- Identify all possible maximally (meta-level) consistent outcome specifications by identifying maximal subsets of essential requirements that are consistent with:
  - The essential requirement $f : j$ if the operation is *addition*. If the operation succeeds (determined via trade-off analysis), $f : j$ becomes an essential requirement in the revised specification.
  - The essential requirement $\emptyset : \neg f$ is the operations is *removal*. If the operation succeeds (determined via trade-off analysis), $\emptyset : \neg f$ becomes an essential requirement in the revised specification.
- Apply the revision choice function to select one of these outcomes.
Evolution: II

- The outcome selected by the revision choice function becomes the new set of essential requirements in the revised specification.
- Elements of the prior set of essential requirements that do not belong to the new set are demoted to the status of (but retained as) tentative requirements.
- For operations that do not succeed, the input \( f : j \) or \( \emptyset : \neg f \) is retained as a tentative requirement.
Evolution: Properties

- Trade-off analysis: not every change request is implemented.
- Lazy evaluation: Do not generate (computationally expensive) \(r\)-maximal subsets for each change step. Do generate (computationally less expensive) weakly (meta-level) consistent outcome specifications at each change step.
- Deferred commitment: Apply an outcome choice function to generate an (object-level) consistent specification only at a late stage (when no significant additional change requests are expected).
- Choice functions (revision and outcome) are context-sensitive and can vary with time.
The REFORM system

- Implemented in Java (initially with the OTTER theorem prover)
- Tested on a medium-scale case-study
- Practitioner-accessible interfaces are a challenge
- Algorithm that interleaves the application of user choice of preferred r-maximal subsets with the computation of r-maximal subsets currently being implemented
- A variety of heuristics (based on relevance, partitions and language restrictions) currently being implemented