Exploiting Decision to Requirements Traceability

John Fitch
SAIC
john.a.fitch@saic.com
Overview

Context
- 20+ years of Systems Thinking/Engineering
- Current work at TARDEC
  - Science and Technology (S&T) programs for Army ground vehicles
  - Technology uncertainty -> decision and requirements volatility
  - Traceability has high value, long-term payoff
  - Scalable methods required

Gaps
- Observations from the tip of the spear

Key Concepts
- True and useful
- Process and tool implications

Practical Techniques
- 5 ways to get started
Gaps = Opportunities

Rationale vs. relationships
- Requirements derivation = rich relationships, not text

Batch traceability + memory loss = lost requirements + low ROI from Requirements Management

Problem domain – solution space entanglement
- Definition of a decision
  - Fundamental question/issue that demands an answer/solution
- Myth of solution-independent decomposition
- Misuse of models

Missing relationships among SE knowledge
Ambiguity tolerated in the SE information model
Information Model for Science & Technology (S&T)

**TODAY’S FOCUS:**
Decision-Requirement Links

---

**Issues**
Issues may be linked to objects in any module

**Requirements**
- Sources
- VoC
- System

**SOW**
Plan Links

**WBS**
Decision-Plan Links

**Decisions**
Decision-Plan Links
Decision-Architecture Links

**Architecture**

**Vee-model**
Test Links

**Tests**
Requirements Links

---

UNCLASSIFIED
Decision to Requirements Traceability

Problem Domain

Criterion
Criterion
Criterion
Criterion
Criterion

Solution Space

Decision

Alternative

Derived Reqt
Derived Reqt
Derived Reqt

Criterion
Criterion
Criterion
Criterion
Criterion

Decision

Alternative

Derived Reqt
Derived Reqt
Derived Reqt

Criterion
Criterion
Criterion
Criterion
Criterion

Decision

Alternative

Derived Reqt
Derived Reqt
Derived Reqt

Criterion
Criterion
Criterion
Criterion
Criterion

Decision

Alternative

Derived Reqt
Derived Reqt
Derived Reqt

Operational
System
Subsystem

R
T
D

UNCLASSIFIED
Decision to Plan Traceability

Requirements Analysis & Design

Top N decisions (Thinking Breakdown Structure) frame the WBS

Implementation

WBS = incremental realization of the solution design (alternatives)

Integration & Test

WBS = incremental verification of the solution design

Requirements elicitation or modeling, simulation and prototyping to inform the decision

Changes to the plan to protect or optimize alternatives

Realize the alternative: build, integrate, test, deploy

Information Model for a Decision

UNCLASSIFIED
Decision-centric Information Model

Enables continuous traceability and seamless software tools

UNCLASSIFIED
Process and tool implications

Stable and proven information model
  – Fuzziness wrung out by 2+ software implementations
  – Comprehensive, domain-independent

Focus on thinking content (linked objects)
  – Not its packaging in document or view containers

High level of process and tool integration possible
  – Enterprise decision model behind all knowledge-driven processes

Preserves decision context for all knowledge

Maximize use of knowledge patterns
  – Decisions + related criteria/requirements, plan, models, etc.

Decisions create all dependencies/interfaces
  – Need methods & tools to capture these at the point of decision
Practical Techniques for Leveraging Decision-Requirements Traceability

Requirements Derivation

Requirements Analysis – Reverse Engineering

Functional Decomposition – Requirements Allocation

Traceability Matrices

Change Management
Requirements Derivation

No decision analysis is complete until:

The inherent consequences of the “committed” alternative have been captured as derived requirements
  – Based on the decision-maker’s insights/perspective of the alternative’s:
    • Structure
    • Behavior
    • Footprint
    • Interfaces
    • Life Cycle

These “raw” derived requirements are captured within the formal requirements structure
  – Copied/linked with explicitly traceability back to their decision source
  – Refined, decomposed and accepted by the owner of the requirements “branch” in which they reside

With similar traceability maintained for risk mitigation and opportunity growth actions that are committed for implementation
Requirements Derivation

Decisions create requirements!

- Requirements
  - Business
  - Customer/User
  - System/Product
  - Component
  - Process
  - Project

- Performance Models & Metrics
- Technologies & Solution Architectures
- Derived Requirements
- Risks & Opportunities
- Risk & Opportunity Actions

Decision
Alternatives
Requirements Analysis – Reverse Engineering

To improve the quality of a system requirements baseline:
– Exploit the fact that all requirements are derived requirements

Reverse engineer upstream decisions (Decision Blitz)
– Your customers’, users’ and System of System’s decisions define your problem
– Map source documents to a proven decision pattern
  • Concept of operations, capability descriptions, use cases, DoDAF views
  • Build explicit model of the customer’s problem domain (decisions) and committed alternatives
  • ASK: “If X is the answer (solution), what was the question (decision)?”
  • Very efficient, convergent knowledge acquisition process

Trace system requirements to these decisions
– 100% trace possible, but not necessary (focus on toughest constraints)
– Highlights gold-plating and tunnel vision (prematurely imposed designs)
– Highlights unknowns that need to be known

Re-open upstream decisions
– Flex the trade space to give the customer what they want, not just what they asked for
Every requirement is a derived requirement, i.e. an inherent consequence associated with a “committed” alternative in another decision.
Functional Decomposition – Requirements Allocation

Decisions drive functional decomposition

- Choose Function X Technology/Method
- Next layer of functions created at the point of decision
- Use caution with model-based decomposition
  - Model = representation of the structure/behavior of an alternative
  - Avoid tunnel vision – first valid model is seldom the best alternative

Decisions drive requirements allocation

- Solution architecture decision “creates” components and interfaces
- Decision analysis is incomplete unless functional and performance allocation is evaluated for each design
- After decision is ratified, complete the allocation trace
Traceability Matrices

Problem Domain

- Requirement
- Criterion
- Criterion
- Criterion

Solution Space

- ALTERNATIVE
- ALTERNATIVE
- ALTERNATIVE

- RISK
- Mitigation
- Requirement

- OPPORTUNITY
- Growth
- Requirement

- Derived Req
- Requirement

- Derived Req
- Requirement

- Derived Req
- Requirement

Capture the decision in the middle

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Mitigation X to Risk Y in Alternative B</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>Growth Action R to Opportunity S in Alternative B</td>
<td>Requirement</td>
</tr>
<tr>
<td>Requirement</td>
<td>Alternative B – structure, behavior, footprint, interfaces, or life cycle</td>
<td>Requirement</td>
</tr>
</tbody>
</table>

The actual derivation trace is a rich many-to-one-to-many transformation that balances multiple criteria and leads to the selection of an alternative. A one-to-one requirement trace is an approximation at best.
A change to requirements may affect related evaluation criteria

A change to criteria may affect the scoring of alternatives

Scoring changes may overturn a previous decision so that a new alternative is selected for implementation

The ripple effect may spread to other decisions

A new “committed” alternative invalidates previous derived requirements and creates new ones
Business Case for Decision to Requirements Traceability

Improved requirements quality
- Completeness, consistency, feasibility
- Less design rework

Continuous traceability
- Decision-makers know their constraints as early as possible

Improved understanding of customer’s needs
- Opportunity to offer higher level solutions

Innovation
- Avoid tunnel vision and imposed solutions
- Optimize solution architecture, functional decomposition

Faster impact/change analysis
- Explicit trace localizes the impact of a change
- Decision logic preserved; more efficient if revisited

Scalable
- Focus traceability on the critical decisions and constraints (Pareto)