Requirements and Rationale Capture:
Current research and promising approaches

USC -Focused Workshop
Automating Software Engineering Knowledge Delivery

John Salasin
February 11, 1999

Briefing Topics

• Research Thrusts
  – Capture
  – Analysis
  – Bridging the gap to architecture
  – Custom explanation

• New (?) areas
  – Event semantics
  – Active, evolvable ontologies
  – View Integration
Capture

- **Central concerns** — information about relationships / interdependencies
  - among development goals, functional requirements, and approaches to meeting goals and requirements (EMMA)
  - conflicts, design obligations, or known design patterns (FAMILIAR)
  - stakeholder values and architectural attributes / models (WinWin)
  - by "issue" or "scenario" schematic scenarios to elicit and formulate requirements.

Analysis and Transformation

- transform mission-oriented goals into specifications of the desired behaviors of architectural features (MORALE)
- assess the impact of new requirements on an existing system's architecture (MORALE)
- generate sub-goal structure from top-level feature lists (EMMA)
- detect and notify users of obligations raised by design decisions, (e.g., alternative distributed processing architectures), perform 'What If' analyses varying design features and goals (FAMILIAR)
- compute tradeoffs between designs based on success factors, domain requirements, life cycle process and property models (cost, schedule, performance, reliability, interoperability). (WinWin)
**Bridging the gap to architecture**

- automated reasoning about each component's functional role within a system, and the interdependencies of these roles (FAMILIAR)
- evaluate a design's compliance with domain-specific rules (e.g., platform-specific constraints), assess the interoperability of components in a proposed design, and identify component interface mismatches and recommend adaptation approaches (FAMILIAR)
- assessing architecture quality attributes with respect to win conditions (WinWin) or features (FAMILIAR / EMMA)

**Custom Explanation**

- Generate software (or design, or plan, or...) explanations on demand rather than storing shelves of documents
  - Focus explanations based on model of task
  - Automated planning of multimedia presentations
  - Automated extraction of info from legacy docs
Event-Based Semantics

- Semantics describes "Meaning"
  - But -- the thing itself is its meaning
    - Useful semantics provide a simpler or more understandable (than the thing itself) description of meaning.
- Types of semantics
  - Data -- Name of the thing; Name + Attributes (Data dictionary); Data dictionary + Relations (ERA); Extended Relations (Ontologies)
  - Functional Semantics -- "Informal" narrative; causal structures; Extensional / Intensional definitions;
    Operational, Axiomatic, or Denotational semantics
  - Event Semantics -- Event types; Partially ordered sets of events; Timed sets of events (for hard real-time)

We need better ways of specifying and analyzing systems in terms of event semantics

For distributed, dynamic, mobile systems, all we have to describe the system are events. We need tools for:
- Specifying legal and illegal sequences of events (constraints on event sets)
- Modeling systems with respect to events
- Monitoring, filtering, and analyzing event sequences in real systems
- Assessing architectural conformance based on events
Active, Evolvable Ontologies

**Background -- Clifford Alexander example of ontology:** “Tree on a hill” implies (assumes):
- “Tree” has trunk (above ground) and roots (below ground)
- “On” means root immersed in, trunk above
- “Hill” is raised soil
- etc.

“Active Ontologies” would include, as relationships, methods that the thing can do or that can be done to the thing?
- Tree can grow at rate of Z%/year, shed M bushels of leaves (some trees), photosynthesize P liters of CO2, ...
- Tree can create X board-feet of lumber, be blown over by wind of Z force, ...

**Active ontologies must be extensible as they are used**

**View integration**

**Two main approaches co-exist**
- “Issue” Based (IBIS, GIBIS, WinWin)
  - Pros and cons wrt specific design issues
  - Decision theoretic
- “Scenario” Based (Use-cases, SCR, SAAM)
  - (Leads to) Finite State Machine representation
  - Analyzable / Executable (prototyping approach)

**Combining methods allows:**
- Switching from scenario- to issue-based as needed to resolve specific alternatives
- Switching from issue- to scenario-based to assure consistency, completeness and to prototype