Providing Support for Dynamism in ACME

Dave Wile
USC/ISI/CSE
wile@isi.edu
http://www.isi.edu/software-sciences/wile/home-page.html

ACME:
An Architecture Interchange Language

New Ideas
Community Consensus Architecture Interchange Language
Integration of independently-developed architecture analysis tool sets
Idiosyncracies conveyed via annotated properties
Architecture Infrastructure Shared via the Web

Impact
No duplication of architecture-related tools among EDCS Community
Shared Architecture and Generation Cluster community vision
Architecture-based prototyping support for the commercial sector

Schedule
ACME Language Specification
Web-based analyzers and tool activation
Dynamic Architectures
ACME-based performance analysis
Style-centered support
FY97 FY98 FY99
Architecture Dynamism

- Variety of ways architecture can change:
  - Instantiation of nearly complete template
  - Logical refinement of components
  - Additions to skeleton or reference architecture
  - Implementation details
  - Self-modifying architecture
- The last is what is usually meant by “dynamism” in architecture, but the concerns are the same in all

What are these Concerns?

- Stem from analyzability
  - Want to guarantee that certain properties hold or do not hold
  - These depend on the architecture, the topological structure
- Examples:
  - End-to-end throughput depends on interfering traffic
  - Security depends on knowing all connections through which secure information could leak
  - Accountability depends on intercepting all sources of information needing accounting
Closed World Assumptions

- Allow finer reasoning
  - structural closure is ideal
  - closure of means for changing/constructing may be just as good (induction needed to prove properties maintained)

- Alternatives:
  - model the dynamic architecture and check constraints on behavior dynamically
  - must detect before a system becomes unsafe

ACME is Stable

- I am about to describe my research into an aspect of architecture of concern to many in the community, dynamism.
- Do not mistake the speculative nature of the research as instability in the language.
- Existing syntax and tools will be supported as long as there are users.
- We are extending Acme to include dynamism.
- Moreover, we are specifying Acme's semantics precisely to understand the dynamic effects
What is ACME?

- ACME, the language = “ACME”
  - Formal communication mechanism
  - Among people, primarily
  - Constraints in predicate calculus or temporal logic
  - **Declarative** (theory-based)
- ACME semantic base = “Base ACME”
  - Structural model maintained in an object base
  - Logical properties for constraint reasoning
  - “Ideal” for mechanical manipulation and **interchange**
  - **Imperative** (model-based)

(Imperative) Base ACME

- Element instances: identify elements by qualified name
- Topological Relationships
  - Component-of(comp, sys)
  - Connector-of(conn, sys)
  - Port-of(port, comp)
  - Role-of(role, conn)
  - Attached(r,p)
  - Parent-of(x,p)
- Properties and Constraints
- Types
Operationally

- Operations
  - Create, classify, destroy, and declassify elements
  - Make and break relationships
  - Nested transactions
- Multiple “Hosts” -- logical platforms for
  - Analyzers
  - Simulators
  - Scripting

Operational ACME:
Potential Hosts

- C++ and Java (current ACME Lib facility)
- Common Lisp (current Semantics, ACME PowerPoint Editor, Analyzer facilities)
- Corba and COM
  - Haskell scripting language available to manipulate COM representation
- Future platforms
(Declarative) ACME

- Planned-for extensions, e.g.
  - UML as a state model (Irvine, Betty Cheng)
  - Rapide’s event formalism (stakeholder organized)
- More dramatic proposals
  - Parameterized types (e.g. n-ports)
  - Dynamic topology (Strawman proposal)
  - Design rules
  - UML (as ADL) to ACME conversion
  - UML/OCL as potential ACME constraint language

Declarative Dynamic ACME

- Semantic implications:
  - open elements, elements whose parts are not entirely specified
  - optional and multiple elements, whose presence is not necessarily guaranteed
- Constraints represented by specifications are conditionally existentially quantified
- Operational ACME mirrors in transactions
Acme Semantics

- Properties and predicates are either:
  - Assumed to hold -- the axioms
  - Derivative from those assumed to hold -- the theorems
- Implicitly, an Acme specification requires that topological assumptions be verified to hold in the artifact the specification purports to describe
- Assumed predicates must also be verified there
- Derived predicates are logical consequences

Example Open Component

- open component A-D
  = \{\text{port in} = \{\text{assume property}\ I = r\}\}\);
- Component A-D and port A-D.in can be identified in the artifact:
  - identifiable(A-D) and identifiable(A-D.in)
- They are instances of types corresponding to component and port in the implementation:
  - component(A-D) and port(A-D.in)
- Port A-D.in is a port of A-D:
  - port-of(A-D.in, A-D)
- A-D.in's properties are satisfied in the artifact:
  - I(A-D.in) = r
Same Component Closed

- open component A-D
  = \{ \text{port in} = \{ \text{property I = r} \} \};

- semantics:
  identifiable(A-D) and identifiable(A-D.in)
  and component(A-D) and port(A-D.in)
  and port-of(A-D.in, A-D)
  and for all p :
    port-of(p, A-D) => p = A-D.in

More Constrained Openness

- Allow optional parts in specification (ports, components, connectors, roles)
- Semantics is a predicate conditioned on identifying the optional parts
- Some closure statements are still possible
- Done with "multiplicity" notation of mathematics adopted by UML
Example Optional Port

- `component A-D`
  = {port in;
     0..1 port scale=property P=q}

- semantics:
  identifiable(A-D) and identifiable(A-D.in)
  and component(A-D) and port(A-D.in)
  and port-of(A-D.in, A-D)
  and (identifiable(A-D.scale)
       => P(A-D.scale)=q)
  and for all p:(port-of(p,A-D) =>
                 p=A-D.in or p=A-D.scale)

Optional and Multiple Parts

- `open component A-D`
  = [ port in;
       open 0..1 port scale;
       0.. port out]

- Informally:
  • `A-D` may have more ports
  • If port scale is present, it may be defined further
  • Port out is multiple and may be referred to as out[i],
    but it need not appear
  • Ports in and out are "closed"
  • Essentially, these provide idioms for refined openness
Dynamic ACME

- Possibile uses:
  - Design rules a la Armani
  - Law-governed systems a la Naftaly Minsky
  - Flea-enforced multiple stage dynamic constraints
  - Single state constraint checking a la Common Lisp ACME Analyzer
  - Adapt Wright dynamism (~ constraints on script)

Pedagogical Use for Semantics

- Characterize assumptions and expected derivations of all raw Acme specifications
- Characterize different ADLs as properties attached to Acme, to make translations perspicuous (well, as perspicuous as predicate calculus gets…)
- Characterize different architectural styles
Real Use for Semantics

- Specify styles using dynamism constructs
- Instantiate style types in a specification
- Derive semantics automatically:
  - assumptions and proof obligations
- Pass off semantics to a theorem prover
  - Theorem prover may prove some things inconsistent; must rewrite specification
  - Theorem prover may be able to derive some derivatives
  - Take residue and check dynamically in a simulation keeping track of the abstract architectural structure (using a Base Acme facility)

Future Tests

- Looking for participants in more convincing alpha tests of these notions
- Have potential link up with J Moore using his Common Lisp theorem prover
- Can also link up with Swamy’s Flea system for dynamic constraint checking
Web-based ACME Tools

- www.cs.cmu.edu/~acme
- Webifier: produces web-viewable description of ACME
- Layout tool that packages graph layout algorithms that you can experiment with (used by webifier)

ACME Tools available over Web

- ACME Lib (parsing, unparsing, object-model representation)
  - C++
  - Java
- Common Lisp ACME suite
  - Popart-based parsing, unparsing, transformation, analysis (abstract syntax representation)
  - ACME <-> ACME Semantics (relational representation)
  - ACME topological connectedness analyzer:
    - Pure ACME
    - Pipe-Filter style
unix-based ACME Tools

- Performance analyzer given specific style and properties on connectors and components (queueing theory in an architectural package)
- Language converters for:
  - Unicon <-> ACME
  - Aesop <-> ACME
  - Wright > ACME
  - ACME > Rapide

Registered EDCS Users
- Keysoft Inc. Secure Fault-Tolerant Architectures. Franklin Webber, contact
- Georgia Tech MORALE project. Bob Waters, contact. Software Architecture Analysis. Pascal Schuchhardt and Colin Potts, contacts. (possibly the same as the MORALE project).
- C2 project at UCI claims to have integrated ACME into C2. Richard Taylor, contact.
- ACME is the mechanism used for integrating the Wright, Unicon, and Aesop environments produced at Carnegie Mellon University. David Garfin and Mary S haw, contacts.
Acme Common Lisp / Dynamic Language EDCS connections:

- Rick Brenner, Draper Labs: investigating Common Lisp ACME for reconfiguring missile software.
- John Paterson and Paul Hudak at Yale in a Haskellized version of Acme for scripting studies.

EDCS Integrator Connections

- Lockheed-Martin ADAM project. Dick Creps and Paul Kogut, contacts.
- Synquiry Technologies' FAMILIAR project for capturing design rationale. "We would like to use ACME to represent the device information that we use to represent system rationale, and integrate it with other tools in EDCS." Dean Allemang, contact.
Industrial registered users:


- Hyundai Information Technology. Unknown project. Nam Kyeongsoon, contact

Academic/Research Laboratory registered users:

- SEI Architecture group. Acme is used as an intermediate representation for an architecture recovery and reengineering project. Rick Kazman, contact.

- SEI. Architectures for aircraft simulation project (project name?) Larry Howard, contact.

- SEI. Translating Meta-H descriptions to Acme and Rapide. Mario Barbacci, contact.

- Keio University, Japan. Harada Lab. of Computer Science.
PC-based ACME Tools

- ACME PowerPoint editor translating into ACME Semantics and responding to analyzer (Bob Balzer)
- ACME graphical editor in C++ (no analysis) producing ACME text (CMU student)