Semantic Interoperability Scripting and Measurements

Ke-Thia Yao, In-Young Ko, Robert Neches, Robert MacGregor
University of Southern California
Information Sciences Institute
Vision of component-based software design is to assemble software applications from a database of existing software components.

Paradigm explicitly partitions software design task:
- Component developers that build the component database
- Application developers that assemble the software
- Administrators that maintain the software

Partitioning of tasks increases demand on the exchange of design information among the parties.
Component Software Development Process

How can I build a correct application with these components?

Component Database
- by function, by I/O data

Architectural Styles
- centralized, distributed
- synchronous, asynchronous

How do I make a new component interoperable?

How do I best configure this application for my local environment?

System Architect/Administrator
Component Software Development Support

Software development tools should explicitly support the automated exchange and reasoning of design information.

Current tools such as Interface Description Languages (IDLs) help to a point:

- Assume exact matching on data types and behaviors
  - Plug-compatible substitutions: yes
  - Adaptation of closely related components: no
- Capture interface and method specifications
  - Syntax level composition: yes
  - Qualitative considerations in composition: no
- Examples: implementation effort, performance, semantics
Approach

1. Provide better component/system metadata
   - Help designers express what they have created
   - Help other designers understand what they’re working with

2. Provide metadata-level scripting mechanism
   - Help designers assemble software applications

3. Provide software gauges
   - Help application developers make component selections
   - Help component developers create new components and maintain the component database
   - Help system architects/administrators make application adaptations
What Do We Mean By Metadata And Scripts

- Abstraction: semantic description of software components
- Scripting: metadata-level description of software application
- Instantiation: instantiation and execution of system-level scripts
Testbed Application: GeoWorlds Component Framework

GeoWorlds
- Geographic Information Systems plus Web processing
- Retrieve, analyze, visualize and organize documents
- Component-based system used at US Pacific Command HQ

Useful testbed
- Actively-used framework for adding, executing components
- Heavy emphasis on runtime application composition
- Large number of components, which stress tests
  - Adding new components
  - Building/changing component applications

Information Gathering

Web

Document Collection

Information Organization

Information Spaces

Document Analysis

Information Visualization
Metadata Level Modeling

Metadata framework:

- **Characteristics:**
  - Lightweight, Multi-form ontology
  - Subsumption-based inferencing
  - Organized by hierarchies
    - Data: document collections e.g., content and structure
    - Services: classes of operations, e.g., Analytic, visual, information source, data converter, input

**Goals:**

- Extensible metadata that is computational efficient
- Human understandable and easy to maintain
- Enables comparison of components wrt:
  - Functionality, I/O data

---

**Diagram:**

- Content
  - Data Type
    - object
    - syntax
  - Organization Structure
    - typeOf
    - comment
    - uri

- Collection Set
  - Collection
    - CONTENT
    - Document Collection
      - content
      - structure
  - Data Element

- Service schemas
  - Service
    - Literal
    - subClassOf
    - inputData
    - outputData
    - serviceRequest

- Document Collection schemas
  - Document
    - Literal
    - typeOf
    - syntax
    - content

- Service instances
  - Service Instance
    - Literal
    - typeOf
Define software application by defining data-flow among components
Ensure generation of semantically valid scripts
Automatically insert data converters
Instantiate, execute metadata-level scripts
Script Example: China Disaster Analysis

- Retrieve a Web document collection about “China disasters”
- Get a set of place names from the map and classify the documents based on them
- Classify documents based on the disaster types mentioned
- Document clusters for ‘China disaster’
  - QUAKE
  - FIRE
  - FLOODING
  - 1997 World Disaster Reduction
  - USAID Provides Disaster Assistance
  - NATIONAL INTERAGENCY COORDINATION
  - China Floods Exacerbated By More Rain
- Cross-product between place names and the disaster type categories
- China disasters
  - FIRE
  - QUAKE
  - FLOOD
  - ANHUI
  - ANSHAN
  - BEIJING
  - CHENGDU
  - CHONGQING
  - DUSHANBE
  - FUJIAN
  - Catastrophic Flood Disaster
- Plot the document clusters on the map to figure out the major flooding areas
- Get a set of place names from the map and classify the documents based on them
Software Gauges

- **Interoperability gauges** to find next components in the script generation process
- **Compatibility gauges** to adapt scripts by finding candidate replacement components
- **Insertion gauges** to judge
  - Interoperability: the level of effort required to insert new components into the repository
  - Utility: the uniqueness of new components within the repository
Measures Used by Interoperability Gauges: Subsumption and Graph Distance

- Which services handle the output of the current service?
  - The input of the service must subsume the output of the current service wrt content and structure

- How well do they match?
  - Prefer more specific matches by using graph distance
Nested Interoperability Gauges

**Categorization gauge**
- Finds all services that can be connected to the current selected services
- Categorize services based on their functionality

**Quality gauge**
- Displays degree of interoperability

**Problem identification gauge**
- Identifies sources of interoperability problems

**GeoWorlds example:**
- Suggest *map viewer* to display geographical data
- If frequency available, rank *frequency list viewer* over *generic list viewer*, and over *category viewer*
- Suggest specialized services
  - Suggest *SOM clustering* after *Noun Phrasrer*
Measures Used by Compatibility Gauges: Context-dependent Semantics, Not Strict Semantics

- Which services can accept the predecessor’s output and conform to the successor’s input parameter?
  - Script context permit less stringent constraints on replacement services
  - Automatic insertion of data converter services

Script fragment around Service $S_i$:

Replacing Service $S_i$ with compatible services:

- Strict Replacement
- Context-dependent Replacement
Nested Compatibility Gauges

**Ranking gauge**
- Finds all replacement services satisfying the context-dependence semantics
- Ranks replacement services by distance from the original service

**Quality gauge**
- Displays degree of compatibility

**Problem identification gauge**
- Identifies sources of compatibility problems

**GeoWorlds example:**
- *Graph-based clustering* requires the output of *Noun Phraser*
  - *Keyword Extractor* is not strictly compatible with *Noun Phraser*
  - It is context-dependent compatible, because *Graph-based clustering* can accept its output
Measures Used by Insertion Gauges: Node and Graph Connectivity

- How interoperable is the new service with the existing services?
  - Service connectivity graph: each node represents a service, arc from node A to B indicate B accepts A’s output
  - Measure interoperability using number of connected components and branching factor

- What does the new service contribute? Unique functionality versus redundancy?
  - Use compatibility measure to compare against existing services
Nested Insertion Gauges

Insertion overview Gauge
- Compares new services against existing services for interoperability, uniqueness/substitutability

Connectivity Gauge
- Show how new service fits in with existing services

Veridian’s Terrain Reasoner
- Low interoperability, high uniqueness, low graph and node connectivity

BBN’s AQE
- High interoperability, low uniqueness, good graph and node connectivity
Future Work: Architectural-level Gauges

- Improve performance by applying architectural level transformations
- Transformations must be *semantically invariant* wrt to script
  - Physical component redeployment
  - Connector change
- Transformations can occur during runtime
- Guided by IntelliGauge TIE runtime gauges,
  - bottlenecks and faults

6-Dec-01
Example: GeoTopics Application
Built using GeoWorlds Framework

- Monitors daily news articles from leading English language papers
- Identifies hot topics, hot places, and how the two relate
- Useful testbed
  - GeoTopics executes daily provide probing opportunities
  - Components executed remotely on distributed servers
    - Monitor server performance
  - Need to access remote information sources
    - Monitor network congestion/interuptions (throws Java exceptions)

www.isi.edu/geoworlds/geotopics
Summary

We have developed

- **Metadata-level description of software components**
  - Provides behavior and input/output description at more abstract level
  - Enables reasoning about their interoperability

- **Metadata-level and system-level scripting mechanism**
  - Ensure generation of semantically valid applications
  - Provides script instantiation and execution

- **Software gauges**
  - Interoperability gauge: to help building component applications
  - Compatibility gauge: to help making changes to applications
  - Insertion gauge: to help adding new components