Georeferencing with Java: Executable Metadata

Russell K. Rew
UCAR / Unidata
13 January, 1998
What is Metadata?

*Data about data*

- Machine-understandable information
- Describes data: units, domain, accuracy, time, ...
- *Georeferencing metadata* specifies data locations
- Currently applications deal with many kinds of georeferencing: lat/lon, polar stereographic, “thinned grids”, Lambert conformal, rotated, semi-staggered, Arakawa, Gaussian, orthographic, ...
- Complexity of metadata makes applications harder to develop and maintain
Conventional Approach

Applications support multiple forms of metadata

App A

App B

App C

Data 1

Data 2

Data 3
An Improvement

Applications use uniform metadata interface, implemented in intermediate software layer

App A  App B  App C

Data 1  Data 2  Data 3

Interoperability Library
A Different Approach

Applications use uniform metadata interface, implemented appropriately for each form of metadata

App A

App B

App C

Data 1

Data 2

Data 3
Executable Metadata

Makes data responsible for data-specific implementations

- Applications use simple, abstract interface to metadata
- Each kind of data implements interface appropriately
- Interface implementation is a Java class, stored as bytes
- Java classes are platform-independent
- Metadata (or reference to metadata) is part of data
Goal: Evaluate the Idea

Explore issues: Practicality, Benefits, Drawbacks

- Questions: security, efficiency, compactness
- Design and implement a pure Java prototype
- Discover unanticipated problems
- Get experience with one of Java's advanced features: class loaders
- Evaluate executable metadata as a step towards a framework for geospatial information that is portable, general, accurate, secure, compact, efficient, and extensible
**Why Java?**

*Fortran, C, or C++ not adequate*

- Portable executables
- Ubiquitous infrastructure
- Support for Interfaces
- Security built in, not bolted on
- Compact representation of classes
- Usable by non-Java applications through Java Native Interface
The Prototype

“The Devil is in the Details”

- Define uniform interface: 2-D georeferencing, transforming between grid indices and lat/lon space
- Write Java application that uses interface: plot domain of selected variables on world map
- Implement several data-specific concrete implementations of interface: lat/lon, staggered, Lambert-conformal grids
- Store implementations in data files: Java .class files converted to netCDF byte arrays
- Read file and activate bytes into a live Java class: named variable converted at run-time to Java class by NetcdfClassLoader
Implementation Issues

No Show Stoppers

- Security: applets cannot use custom class loaders
- Java namespaces: loaded classes use different namespace from application classes, need to share common interfaces
- Performance not much of an issue, adequate in prototype
- Classes surprisingly compact, e.g. 2000 bytes for two-way Lambert-conformal transformation with interpolation
- 100% Java, no portability problems discovered
Issues Avoided in Prototype
That’s why it’s only a prototype

- Tools to help data providers package metadata with datasets
- More general interface to multidimensional spatiotemporal data
- Custom security manager, installed by class loader
- Versioning support for evolving interfaces, archives
Evaluation: Benefits

Bringing dead bits to life

- Simpler applications interfaces and code (magic of polymorphism)
- Powerful metadata: can interpolate, transform, sample, ...
- Adaptable to change: old applications will work with new datasets
- Compact representations for complex georeferencing
- Abstract interfaces provide common concepts for data users and providers
Evaluation: Limitations

*Inherits Java's limitations*

- Applications must have access to a Java Virtual Machine
- Browsing or searching with executable metadata is difficult
- Security concerns
- Longevity in archives
- Data providers not code providers
Other Uses

*Makes data more self-describing*

- Interpolation algorithms
- Derivatives
- Irregular domains
- Compression
- Calibration
- Error estimates
- ...

**Other Uses**

*Makes data more self-describing*
For More Information

- Prototype Java source code: http://www.unidata.ucar.edu/staff/russ/MapGeoGrid
- See paper in Proceedings
- Visit Unidata booth (#321)
- Email: russ@unidata.ucar.edu