Scalable Light-Weight Integrated Utility Engine (SLIQUE)

Michael Molignano, Timothy Navien and Steven Shidlovsky

14 October 2009
Group 39

WPI Advisor:
Professor George Heineman

MIT LL Advisors:
Mr. Scot DeDeo
Dr. David Holl
Motivation

• Need to analyze live and recorded domain-specific mission data
• Produce real-time displays & quick-look/analysis products
Background

• Lincoln Laboratory Visual Interface and Scalable Transport API (LLVISTA)
  – Designed to support domain specific mission data
  – Expandable and flexible
  – 4 Tier architecture

• Constraints
  – Requires extensive knowledge of Java and Groovy scripting engine
  – Requires understanding of underlying framework
• SLIQUE
  – Developed a visual design tool
  – Create and execute mission data processing pipelines
    – Uses drag & drop interface with LLVISTA components and connections
    – Provides in-line validation
    – Auto-discovery of components to grow with LLVISTA
  – Reduces required knowledge of Java and LLVISTA
System in Use

Example Mission Data Processing Pipeline Using SLIQUE
System in Use

Rendered Configuration from SLIQUE
Methodology

• Software engineering principles
  – Unified Modeling Language
  – Iterative development
  – Source Code Control
  – Documentation
  – Code reviews
  – Feature/Code freeze dates
  – Validation of code through testing

• Conformance reporting of LLVISTA components
Design Decisions

- **Model-View-Controller**
  - Allows modeling pipeline internally
  - Allows for graphical connection to display model
  - Easy rendering of model through groovy
  - Supports testing of model and controllers
  - Allows checking conformance of LLVISTA components
Challenges

• Restricted development environment
• Java Swing development
• Merging LLVISTA details
• LLVISTA component standardization
• Groovy engine difficulties
Evaluation

- Code coverage through JUnit testing
- Code review
  - Singular and Team
- User testing from multiple group 39 members
  - Incorporated Feedback throughout Project
- Comparison of manual vs SLIQUE-generated scripts
Results

Script Using Current Method

```plaintext
ad = AdvertisementUtilities.createStandaloneAdvertisement("trash", "OM", "truth", "esc");
// build the event messages
firstTimeMessage = ["type" => "trash", "event_type" => "new_track", "track" => "OM", "coordinate_format" => "3d", object_name => "OM_truth", "mark_name" => "Black triangle up"];
lastTimeMessage = ["type" => "trash", "event_type" => "lost_track", "track" => "OM"];
def cmSource = new ASCIISatementStreamSource("OM truth_pile_name", ad, 5, 0);
cmSource.setFirstTimeEventMessages([firstTimeMessage]);
cmSource.setLastTimeEventMessages([lastTimeMessage]);
cmSource.interpolator = new LinearInterpolator();
cmSource.interpolator.interpolateMark = 1;
cmSource.interpolator.startInterpolateMark = true;

//*** Truth Source for Interceptor 1 ***
ad = AdvertisementUtilities.createStandaloneAdvertisement("Interceptor", "Scepti", "truth", "esc");
// build the event messages
firstTimeMessage = ["type" => "Interceptor", "event_type" => "new_interceptor", "interceptor" => "Scepti", "coordinate_format" => "3d", object_name => "Scepti_1", mark_name => "Black triangle down"];
lastTimeMessage = ["type" => "Interceptor", "event_type" => "lost_interceptor", "interceptor" => "Scepti_1"];;
def sceptSource = new ASCIISatementStreamSource("Scepti_1_pile_name", ad, 9, 0);
sceptSource.setFirstTimeEventMessages([firstTimeMessage]);
sceptSource.setLastTimeEventMessages([lastTimeMessage]);

//*** Truth Source for Interceptor 2 ***
ad = AdvertisementUtilities.createStandaloneAdvertisement("Interceptor", "Scepti2", "truth", "esc");
// build the event messages
firstTimeMessage = ["type" => "Interceptor", "event_type" => "new_interceptor", "interceptor" => "Scepti2", "coordinate_format" => "3d", object_name => "Scepti2_1", mark_name => "Black square down"];
lastTimeMessage = ["type" => "Interceptor", "event_type" => "lost_interceptor", "interceptor" => "Scepti2_1"];;
def scept2Source = new ASCIISatementStreamSource("Scepti2_1_pile_name", ad, 9, 0);
scept2Source.setFirstTimeEventMessages([firstTimeMessage]);
scept2Source.setLastTimeEventMessages([lastTimeMessage]);

//*** Skip Measurements in TBM ***
ad = AdvertisementUtilities.createStandaloneAdvertisement("Ship", "TBM", "Measurement", "esc");
// build the event messages
firstTimeMessage = ["type" => "Mark", "event_type" => "new_object_data", "sensor" => "null", "track" => "TBM"];
```

• Enables use of LLVISTA by more people
Future Work

• Future Improvements
  – Improve interaction with non-standardized components
  – On-the-fly components
  – Advanced GUI layout with additional GUI components
  – Per-component documentation
Acknowledgments

We would like to thank the following for their guidance and assistance throughout the project:

Mr. Scot DeDeo
Professor George Heineman
Dr. David Holl
Mr. Joe Georger
Mr. Dave Oostdyk
Professor Ted Clancy
Mr. Vivek Varshney
Group 39
Questions?