Visual Modelling of Complex Business Processes with Trees, Overlays and Distortion-based Displays

Lei Li, John Hosking and John Grundy

Departments of Computer Science and Electrical and Electronic Engineering
University of Auckland, New Zealand

{L.Li, john, john-g}@cs.auckland.ac.nz
Outline

• Motivating example and requirements
• Problems with existing approaches
• Introduction to EML
  – Base notation
  – Overlay layers
• MaramaEML support tool
• Evaluation
• Future work
Motivating example: University enrolment

• Dynamic collaborations between:
  – Student, Enrolment Office, Academic Departments, Finance Office and StudyLink (student loan agency)

• The main functional requirements are:
  – Students search course database and apply for enrolment;
  – If approved, they may apply for a loan from StudyLink
  – Enrolment Office checks applcn with academic Department staff and informs student of result
  – Dept staff check applcn and approve or reject
  – If approved Finance Office tracks fee payment
    • notifies Enrolment Office and Department of changes.
  – If student applies for a loan, Finance Office supplies student info to StudyLink.
  – StudyLink examines student info & approves or declines loan
Partial BPMN model

- Scalability issues
  - Cobweb and labyrinth problems or
  - Massive hidden dependency problems with drill downs
Requirements for a “good” BP VL

- Easy to understand by both business and technology participants
- Can efficiently model distributed complex systems and their collaborations
- Provides multi-level abstractions to assist different process specifications
- Addresses the problem of over-complex diagrams
- Can be integrated effectively with other modelling technologies
- Supports automatic generation from visual models to industry standard code e.g. BPEL scripts
Existing approaches

- **UML, Petri nets**
  - difficult for business end users to understand
- **WTD, T-Web DSLs**
  - limited set of abstractions, not general enough
- **ARIS, TOVE**
  - too technically focussed, need for programming knowledge
- **BPMN, BioOpera, FormChart, Zenflow**
  - cobweb and labyrinth problems, multi-view mitigations create hidden dependency problems
Our approach

- Use a service tree to provide diagram *spine*
  - Familiar abstraction for target end users
- Use a variety of elision and fisheye view approaches to manage scalability of the tree
  - Many well understood techniques to draw from
- Use elidable overlays on the tree to represent processes (and triggers + exceptions)
  - Our previous work suggest this provides good scalability while mitigating hidden dependencies
Tree elision
Process overlays
MaramaEML
Distortion-based view for scalability
Code generation
Implementation

• Used our Marama meta-tool to develop MaramaEML
  – Marama used to specify the EML domain-specific visual language notation and meta-model
  – Generated Eclipse-based editors from these to realise the basic support environment.
  – Tree layout, overlays and distortion-based displays are all implemented as complex visual event handlers (Java).
  – Integration with BPMN, code generation of BPEL, and LSTA engine integration are implemented as event-driven, model-level data updates (Java).
Evaluation

- Versus Requirements
  - All met
- Cognitive dimensions
  - Strong emphasis on:
    - Closeness of mapping
    - Hidden dependency mitigation
- Task-based end user evaluation
  - Small scale
  - Good support for EML over BPMN for both pen and paper and computer based modelling
  - Some criticism of environment
    - Speed of response for fisheye view
    - Lack of traceability support
- Large end user evaluation
  - Approx 30 users
Large Evaln Results Summary

General Quality Feedback

User Performance --- Task Completion

EML and MaramaEML General Usability Rate
Large Evaln Result Summary

Participants were divided into two groups to answer different questionnaires (General Usability and Cognitive Dimensions Walkthrough)

• Very positive results for EML modelling ability and tree-overlay methodology
• Good comments on software tool support: easy to use, provides efficient modelling, inspection and code generation functions, etc.
• Very good performance feedback on Visibility enhancement, Viscosity maintenance, Diffuseness simplification, Hard Mental Operation reduction, Consistency awareness, Hidden dependency mitigation and Closeness of mapping.

• Trade offs for Premature Commitment, Abstraction Gradient and Secondary Notation support
• Strong demand for adding UML view into framework
• An achromatopsia participant became totally lost in the overlay integration view
• Lack of F1 help function in system
• Speed improvements needed when modelling large tree structure
Next Steps

• Integration with some of our lower level tools
  – MaramaMTE software architecture specification and performance modelling
  – ViTABaL-WS web services specification

• Use as an exemplar in developing a better approach to model integration
  – Have had success with integrating our high level visual mapping tools into Marama
  – Want to extend to an even higher level paradigm for model integration
References