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

CSI Centre for Software Innovation

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

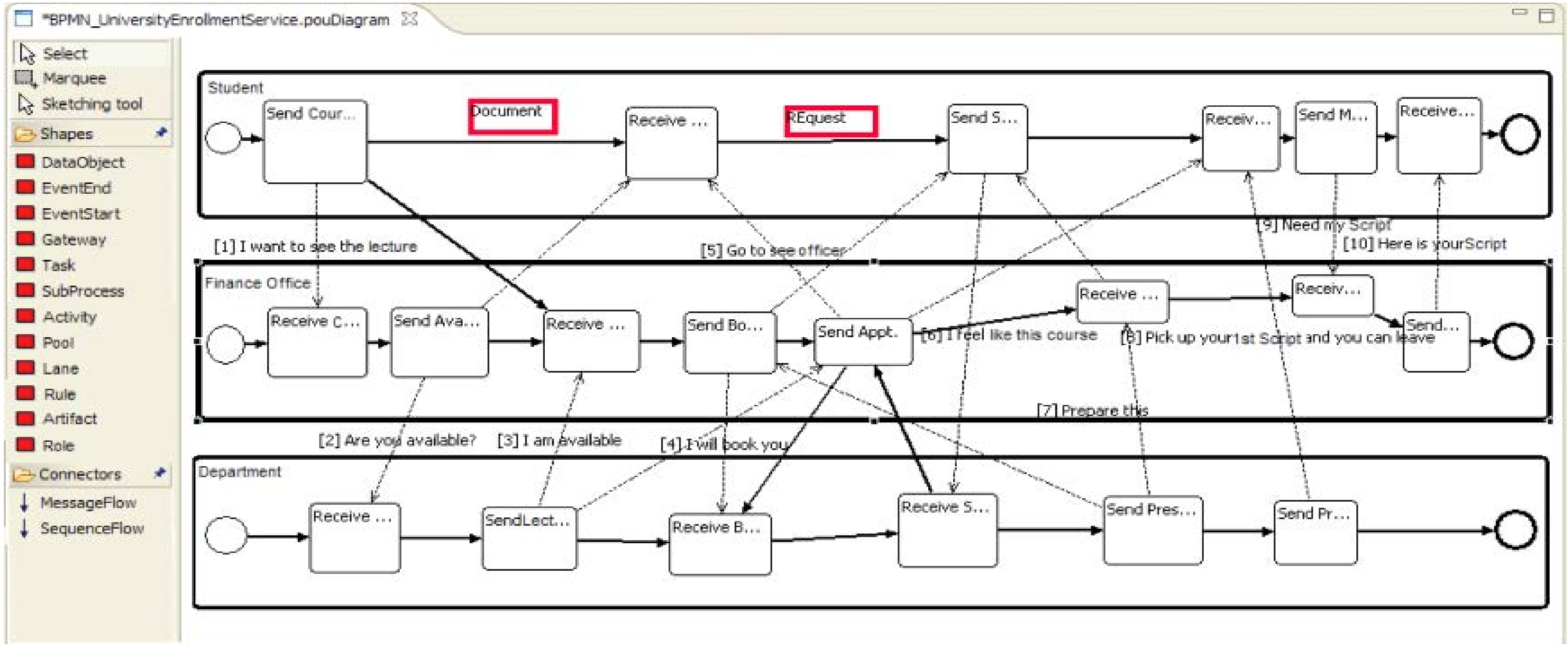
Outline

- 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



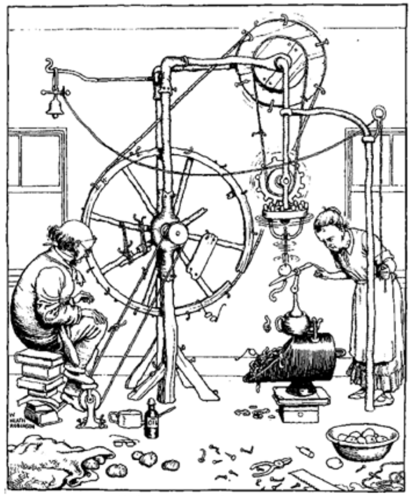
- Scalability issues
  - Cobweb and labyrinth problems or
  - Massive hidden dependency problems with drill downs

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

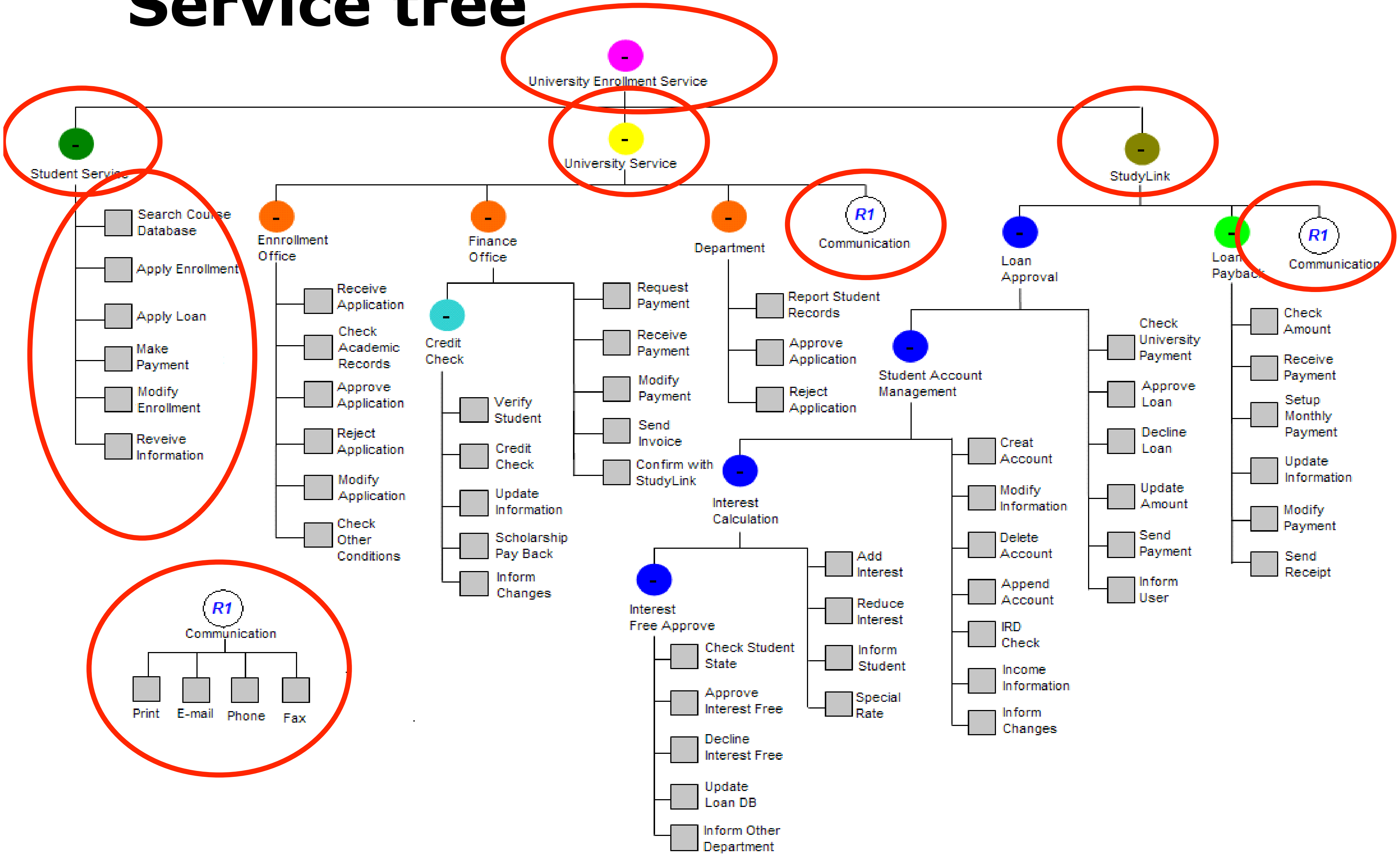


The Professor's invention for peeling potatoes.

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

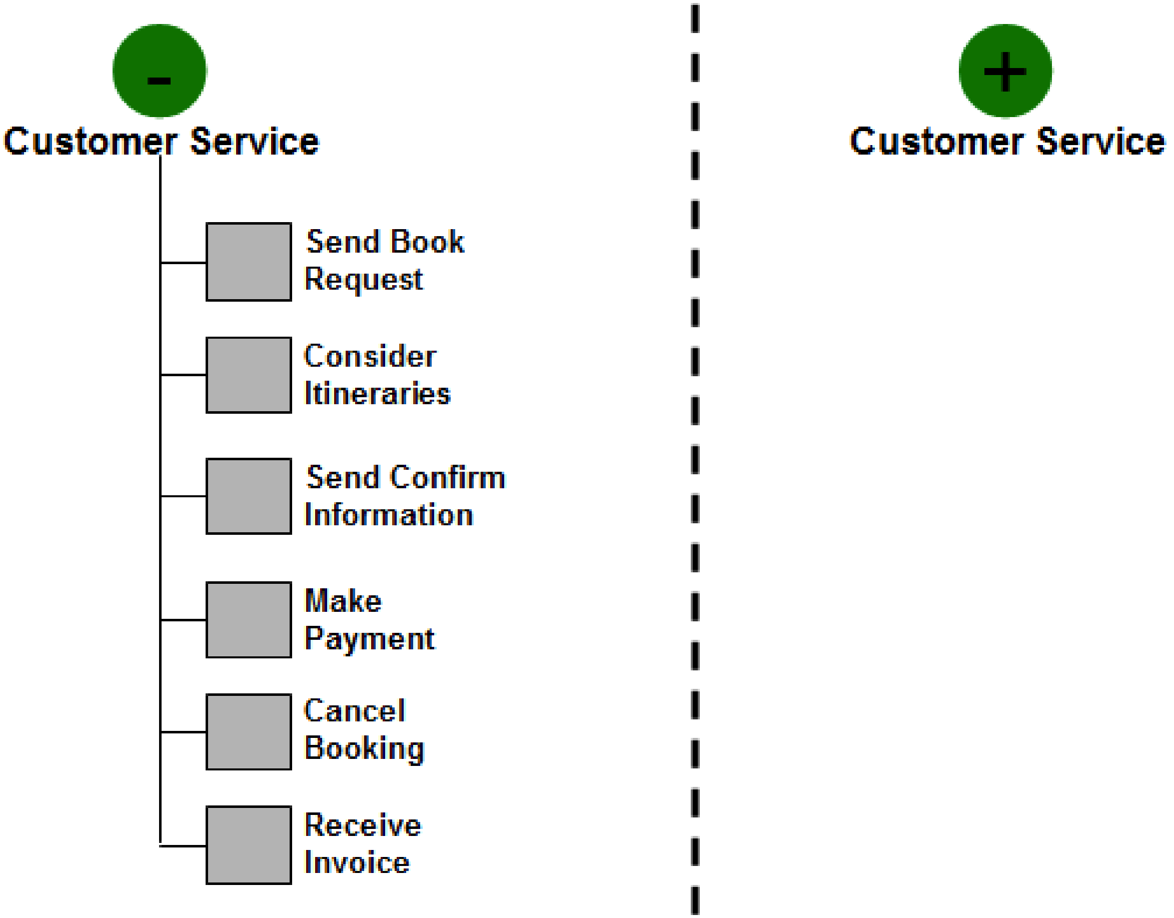
Service tree



- University Enrollment Service
  - Student Service
    - Search Course Database
    - Apply Enrollment
    - Apply Loan
    - Make Payment
    - Modify Enrollment
    - Reveive Information
  - University Service
    - Ennrollment Office
      - Receive Application
      - Check Academic Records
      - Approve Application
      - Reject Application
      - Modify Application
      - Check Other Conditions
    - Finance Office
      - Credit Check
        - Verify Student
        - Credit Check
        - Update Information
        - Scholarship Pay Back
        - Inform Changes
      - Request Payment
      - Receive Payment
      - Modify Payment
      - Send Invoice
      - Confirm with StudyLink
    - Department
      - Report Student Records
      - Approve Application
      - Reject Application
    - R1 Communication
  - StudyLink
    - Loan Approval
      - Student Account Management
        - Interest Calculation
          - Interest Free Approve
            - Check Student State
            - Approve Interest Free
            - Decline Interest Free
            - Update Loan DB
            - Inform Other Department
          - Add Interest
          - Reduce Interest
          - Inform Student
          - Special Rate
        - Creat Account
        - Modify Information
        - Delete Account
        - Append Account
        - IRD Check
        - Income Information
        - Inform Changes
      - Check University Payment
      - Approve Loan
      - Decline Loan
      - Update Amount
      - Send Payment
      - Inform User
    - Loan Payback
      - Check Amount
      - Receive Payment
      - Setup Monthly Payment
      - Update Information
      - Modify Payment
      - Send Receipt
    - R1 Communication

- R1 Communication
  - Print
  - E-mail
  - Phone
  - Fax

Tree elision


-

Customer Service

Send Book Request

Consider Itineraries

Send Confirm Information

Make Payment

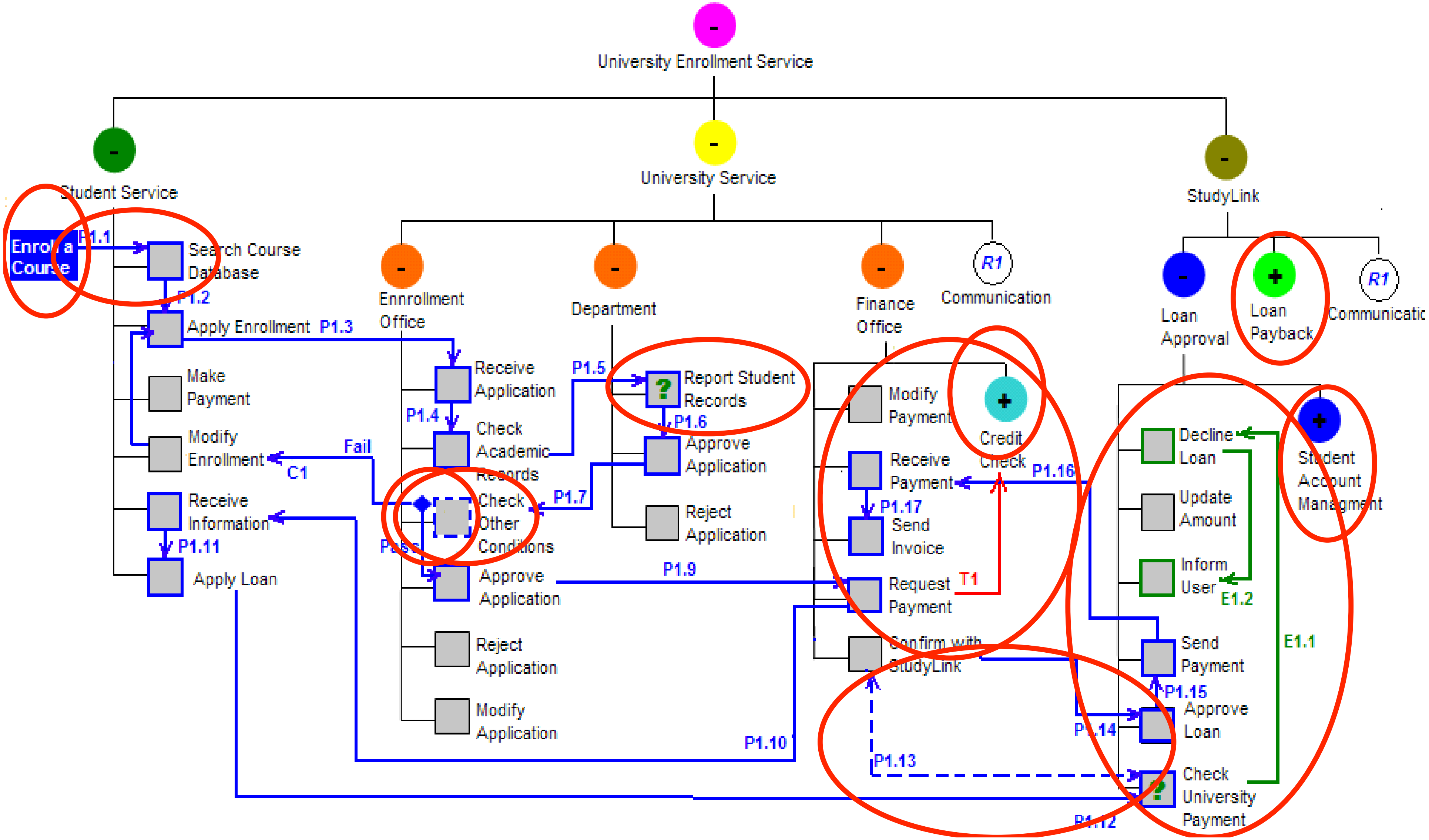
Cancel Booking

Receive Invoice

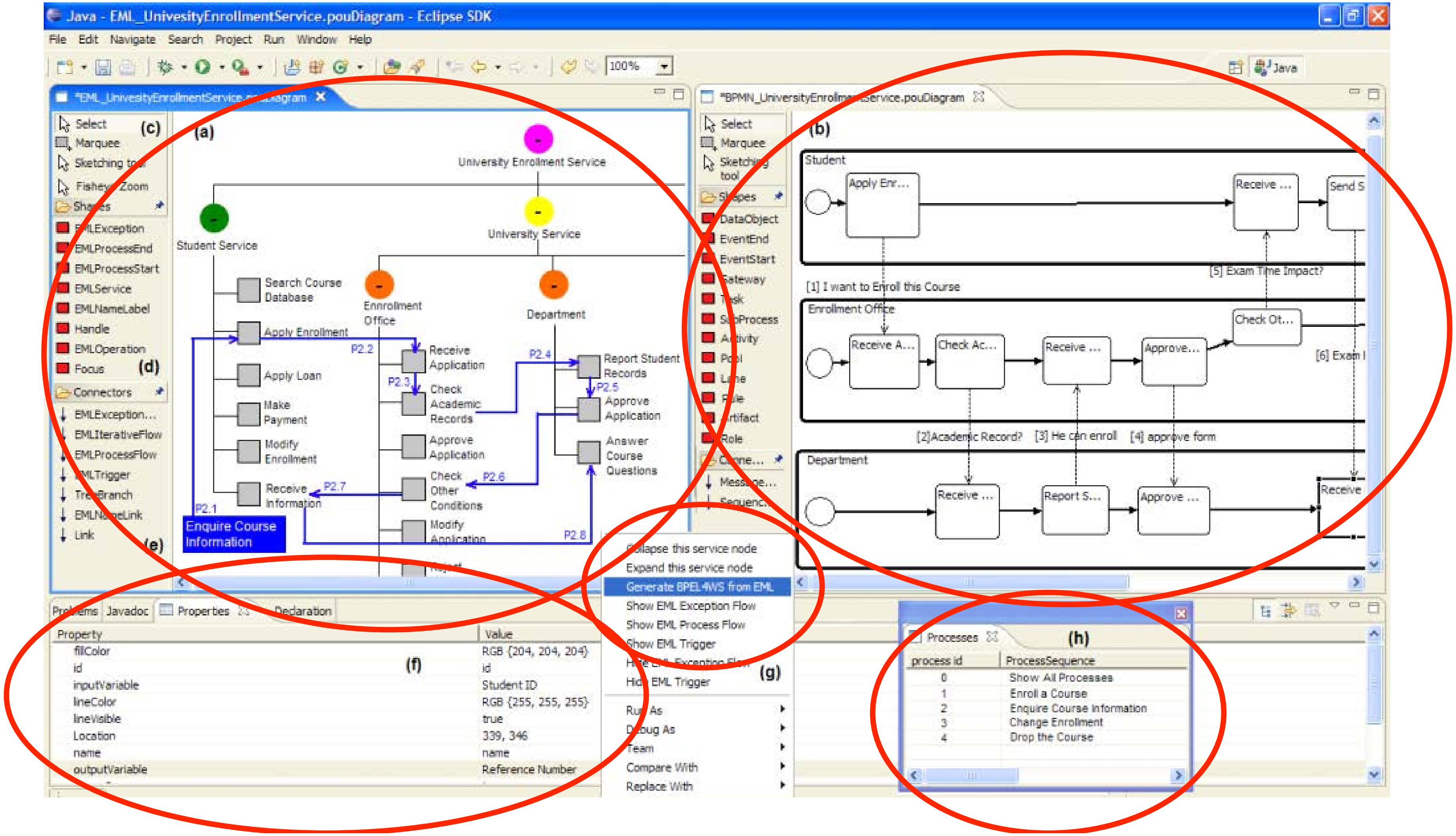
+

Customer Service

Process overlays



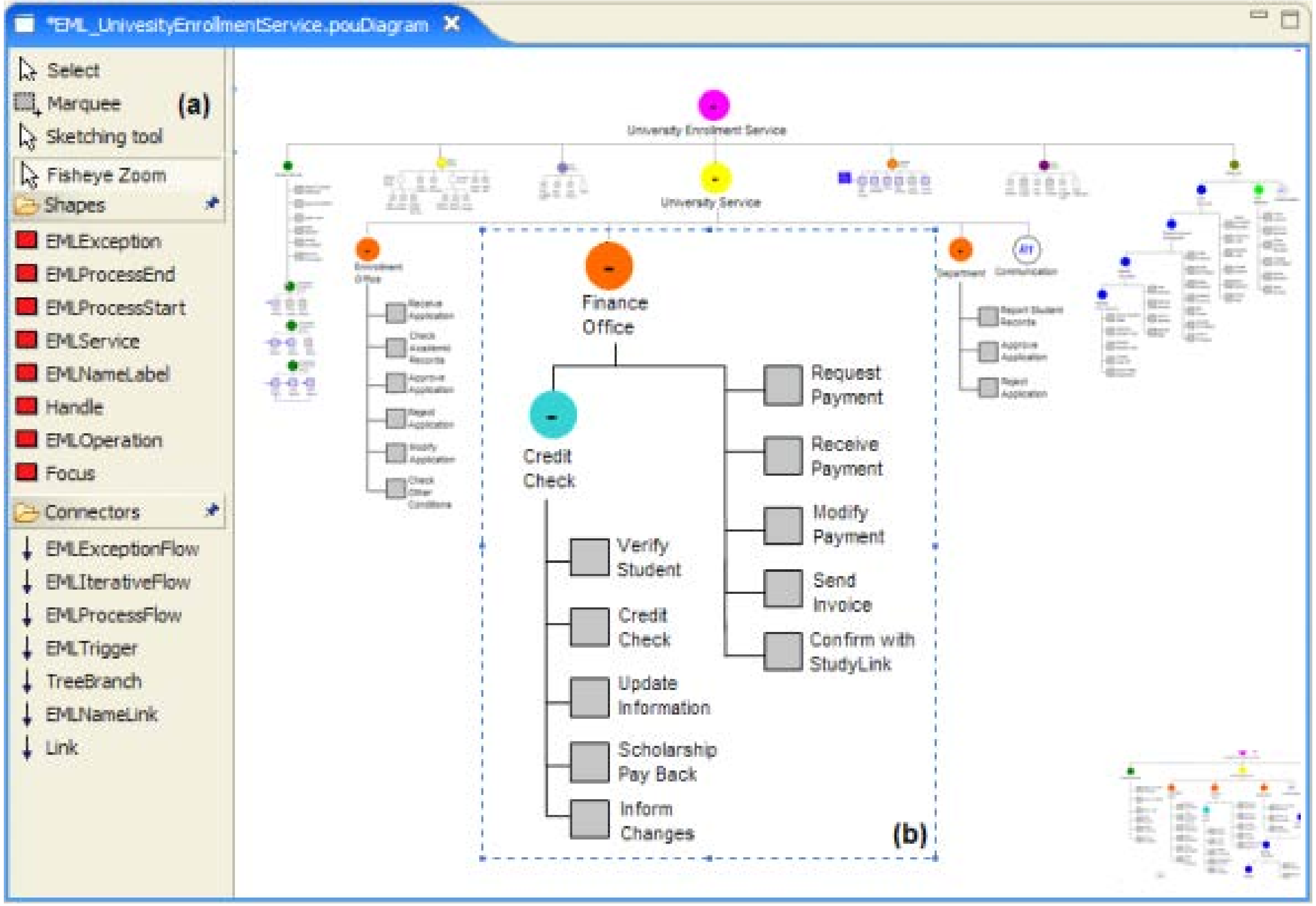
MaramaEML



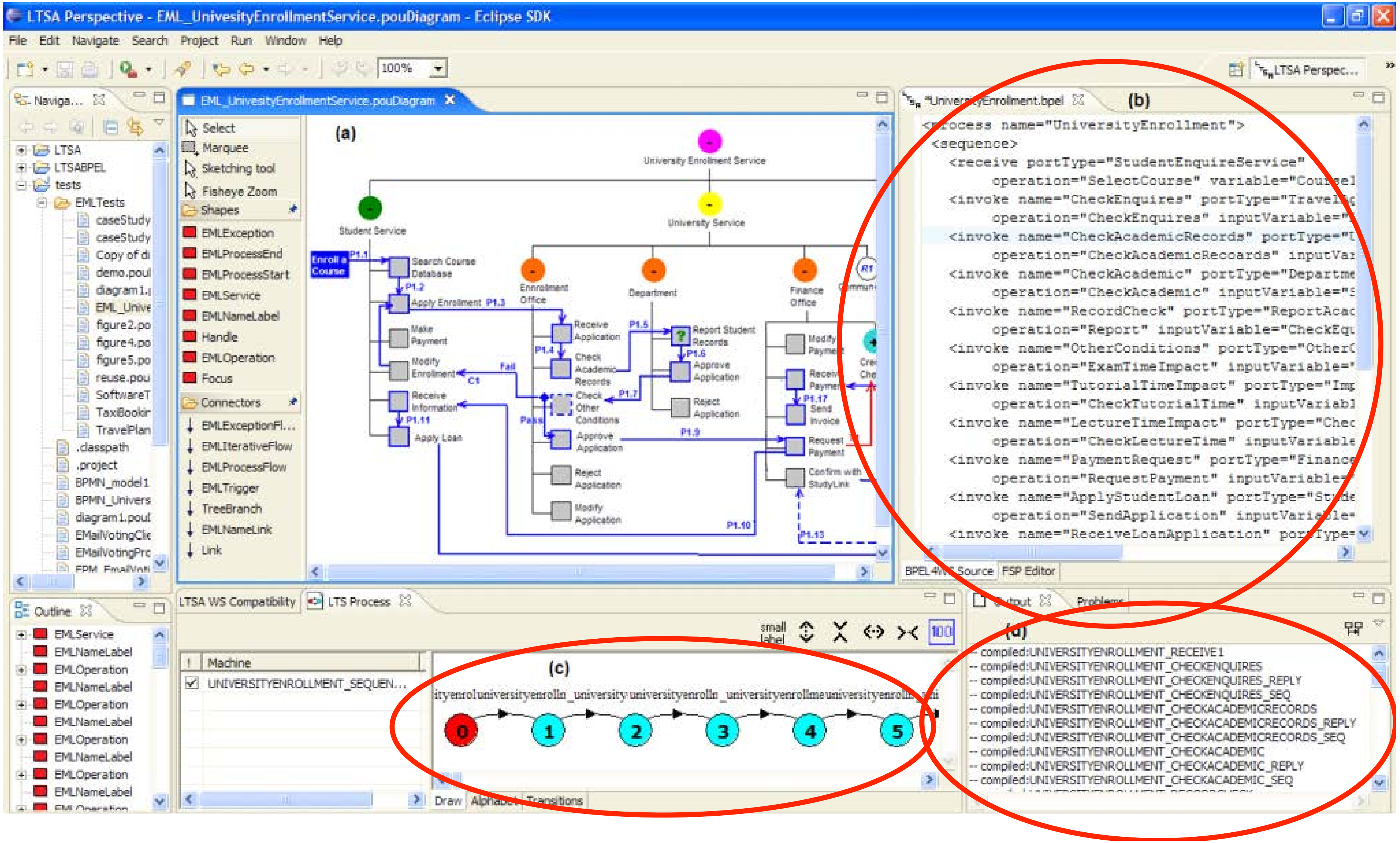
Property	Value
fillColor	RGB {204, 204, 204}
id	id
inputVariable	Student ID
lineColor	RGB {255, 255, 255}
lineVisible	true
Location	339, 346
name	name
outputVariable	Reference Number

process id	ProcessSequence
0	Show All Processes
1	Enroll a Course
2	Enquire Course Information
3	Change Enrollment
4	Drop the Course

Distortion-based view for scalability



Code generation

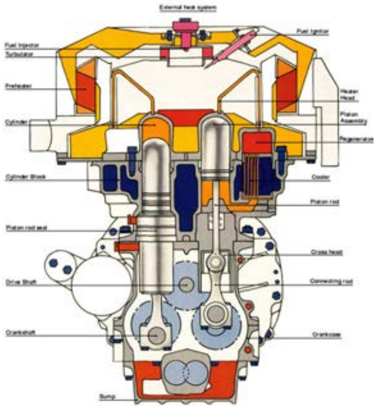


```
<process name="UniversityEnrollment">
 <sequence>
   <receive portType="StudentEnquireService"
        operation="SelectCourse" variable="CourseSel
   <invoke name="CheckEnquires" portType="TravelAg
        operation="CheckEnquires" inputVariable="
   <invoke name="CheckAcademicRecords" portType="U
        operation="CheckAcademicRecoards" inputVar
   <invoke name="CheckAcademic" portType="Departme
        operation="CheckAcademic" inputVariable="S
   <invoke name="RecordCheck" portType="ReportAcac
        operation="Report" inputVariable="CheckEqu
   <invoke name="OtherConditions" portType="OtherC
        operation="ExamTimeImpact" inputVariable="
   <invoke name="TutorialTimeImpact" portType="Imp
        operation="CheckTutorialTime" inputVariabl
   <invoke name="LectureTimeImpact" portType="Chec
        operation="CheckLectureTime" inputVariable
   <invoke name="PaymentRequest" portType="Finance
        operation="RequestPayment" inputVariable=
   <invoke name="ApplyStudentLoan" portType="Stude
        operation="SendApplication" inputVariable=
   <invoke name="ReceiveLoanApplication" portType=
```

```
-- compiled:UNIVERSITYENROLLMENT_RECEIVE1
-- compiled:UNIVERSITYENROLLMENT_CHECKENQUIRES
-- compiled:UNIVERSITYENROLLMENT_CHECKENQUIRES_REPLY
-- compiled:UNIVERSITYENROLLMENT_CHECKENQUIRES_SEQ
-- compiled:UNIVERSITYENROLLMENT_CHECKACADEMICRECORDS
-- compiled:UNIVERSITYENROLLMENT_CHECKACADEMICRECORDS_REPLY
-- compiled:UNIVERSITYENROLLMENT_CHECKACADEMICRECORDS_SEQ
-- compiled:UNIVERSITYENROLLMENT_CHECKACADEMIC
-- compiled:UNIVERSITYENROLLMENT_CHECKACADEMIC_REPLY
-- compiled:UNIVERSITYENROLLMENT_CHECKACADEMIC_SEQ
```

Implementation

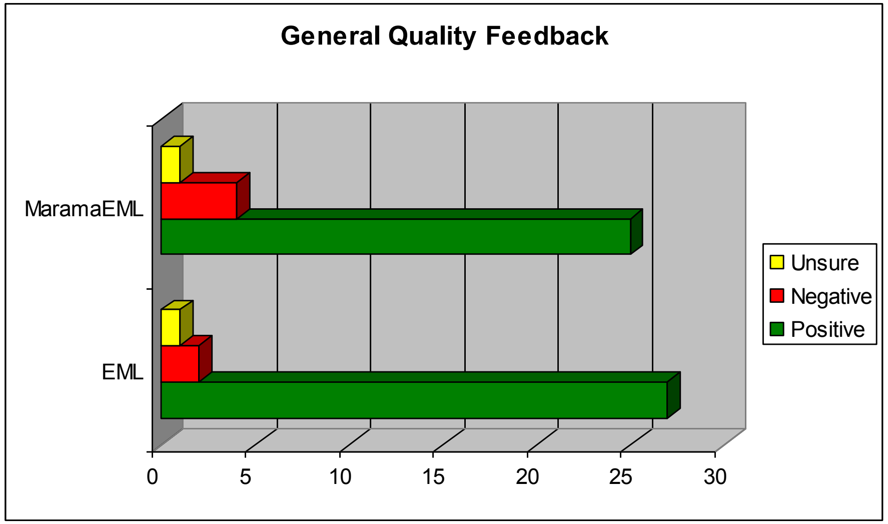
- Used our Marama meta-tool to develop MaramaEML
  - Marama used to specify the EML domain-specific visual language notation and meta-model
  - Generated Eclipse-based editors from these to realise the basic support environment.
  - Tree layout, overlays and distortion-based displays are all implemented 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).

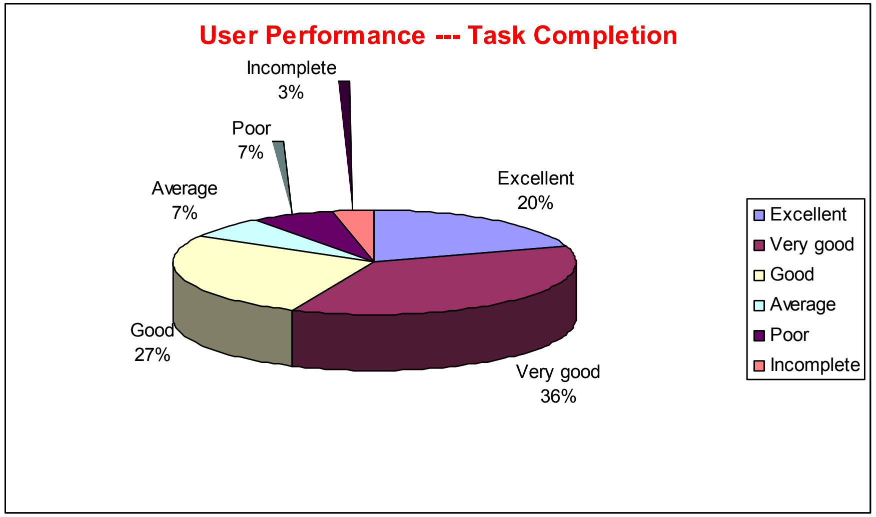


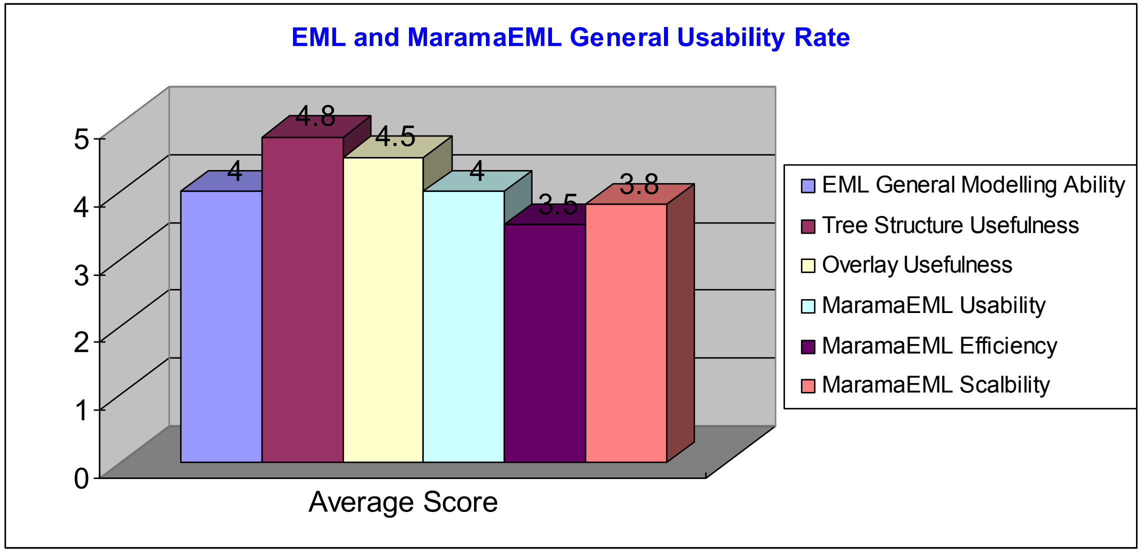
Evaluation

- 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

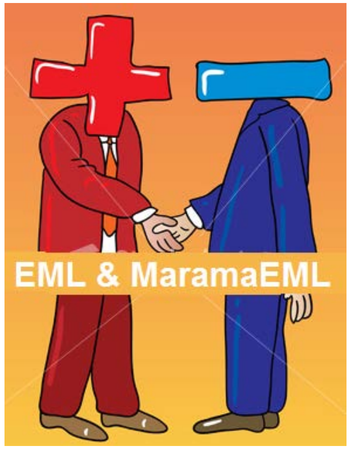

User Performance --- Task Completion


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)

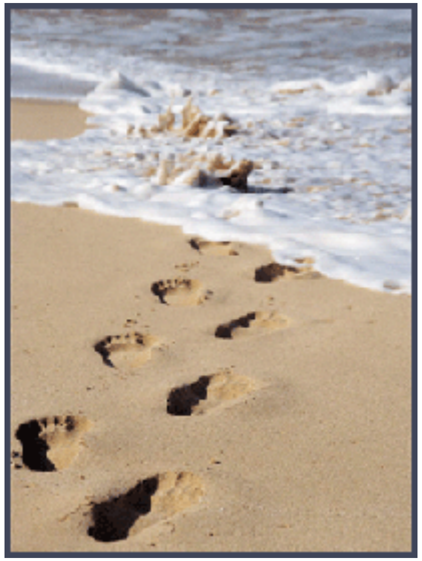
- Very positive results for EML modelling ability and tree-overlay methodology
- Good comments on software tool support: easy to use, provides efficient modelling, inspection and code generation functions, etc.
- Very good performance feedback on Visibility enhancement, Viscosity maintenance, Diffuseness 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

- Li, L, Grundy, J.C., Hosking, J.G. A visual language and environment for enterprise system modelling and automation, Journal of Visual Languages and Computing, vol. 25, no. 4, April 2014, Elsevier, pp. 253-277.
- Grundy, J.C., Hosking, J.G., Li, N., Li, L., Ali, N.M., Huh, J. Generating Domain-Specific Visual Language Tools from Abstract Visual Specifications, IEEE Transactions on Software Engineering, vol. 39, no. 4, April 2013, pp. 487 - 515.
- Li, K, Grundy, J.C., Hosking, J.G. Li, L. Visualising Event-based Information Models: Issues and Experiences, In Proceedings of Visual Analytics in Software Engineering, Workshop at 2009 IEEE/ACM Automated Software Engineering Conference, Auckland, New Zealand, 16 Nov 2009.
- Li, L., Hosking, J.G. and Grundy, J.C. MaramaEML: An Integrated Multi-View Business Process Modelling Environment with Tree-Overlays, Zoomable Interfaces and Code Generation, Demo session, In Proceedings of the 2008 IEEE/ACM International Conference on Automated Software Engineering, L'Aquilla, Italy, 15-19 September 2008, IEEE CS Press.
- Li, L. Hosking, J.G. and Grundy, J.C. Visual Modelling of Complex Business Processes with Trees, Overlays and Distortion-Based Displays, In Proceedings of the 2007 IEEE Symposium on Visual Languages and Human-Centric Computing, USA, Sept 23-27 2007, IEEE CS Press.
- Li, L., Grundy, J.C. and Hosking, J.G. EML: A tree overlay-based visual language for business process modelling, In Proceedings of the 2007 International Conference on Enterprise Information Systems, Portugal, 13-17 June 2007.
- Grundy, J.C., Hosking, J.G., Li, L. And Liu, N. Performance engineering of service compositions, ICSE 2006 Workshop on Service-oriented Software Engineering, Shanghai, May 2006.
- Grundy, J.C. Visual specification and monitoring of software agents in decentralised process-centred environments, International Journal on Software Engineering and Knowledge Engineering, Vol. 9, No. 4., August 1999, World Scientific Publishing Company, pp. 425-444.
- Grundy, J.C., Hosking, J.G., Mugridge, W.B., Apperley, M.D. A decentralised architecture for software process modelling and enactment, IEEE Internet Computing: Special Issue on Software Engineering via the Internet, Vol. 2, No. 5, September/October 1998, IEEE CS Press, pp. 53-62.
- Grundy, J.C. and Hosking, J.G. Serendipity: integrated environment support for process modelling, enactment and work coordination, Automated Software Engineering: Special Issue on Process Technology, Vol. 5, No. 1, January 1998, Kluwer Academic Publishers, pp. 27-60.