Model-driven software engineering: Engineering or not?

John Grundy
Professor of Software Engineering
Dean, Software & Electrical Engineering
Director, Centre Computing & Engineering
Software Systems

SWINBURNE UNIVERSITY OF TECHNOLOGY

Swinburne
think forward

- What characterises “Engineering”?
- What is model-driven software engineering?
- Some Examples
- Is MDE really “Engineering”??
- Future directions

What is “Engineering”?

- Wikipedia:

  “Engineering is the discipline, art and profession of acquiring and applying technical, scientific, and mathematical knowledge to design and implement materials, structures, machines, devices, systems, and processes that safely realize a desired objective or invention.”

- ABET:

  “… creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation and safety to life and property”

- Problem needing solving
- Requirements & Specifications
- Designs – usually multiple options; function & aesthetics & safety & …
- Mathematical models
- Scientific theories/principles underlying models
- Analysis of models to predict outcomes of different options
- Computer-aided design, manufacturing, analysis, processes
- Repeatable processes, project management principles
- Sharing of best practices, professionalism, ethical behaviors, …

- No “physical” models to ground, constrain, inform
- Artefacts highly changeable through engineering lifecycle [Note: just because CAN change, doesn’t mean SHOULD!]
- What are appropriate models/modelling languages?
- Where does “design” end and “construction” begin?
- Are our solutions/processes repeatable?
- Can we evaluate results – before and/or after construction?
- Can (and do) we capture “best practice”?
- Widespread practices of professionalism and ethics?

- Wikipedia ☺:

  MDE is a software development methodology which focuses on creating models, or abstractions, more close to some particular domain concepts rather than computing (or algorithmic) concepts. It is meant to increase productivity by maximizing compatibility between systems, simplifying the process of design, and promoting communication between individuals and teams working on the system.

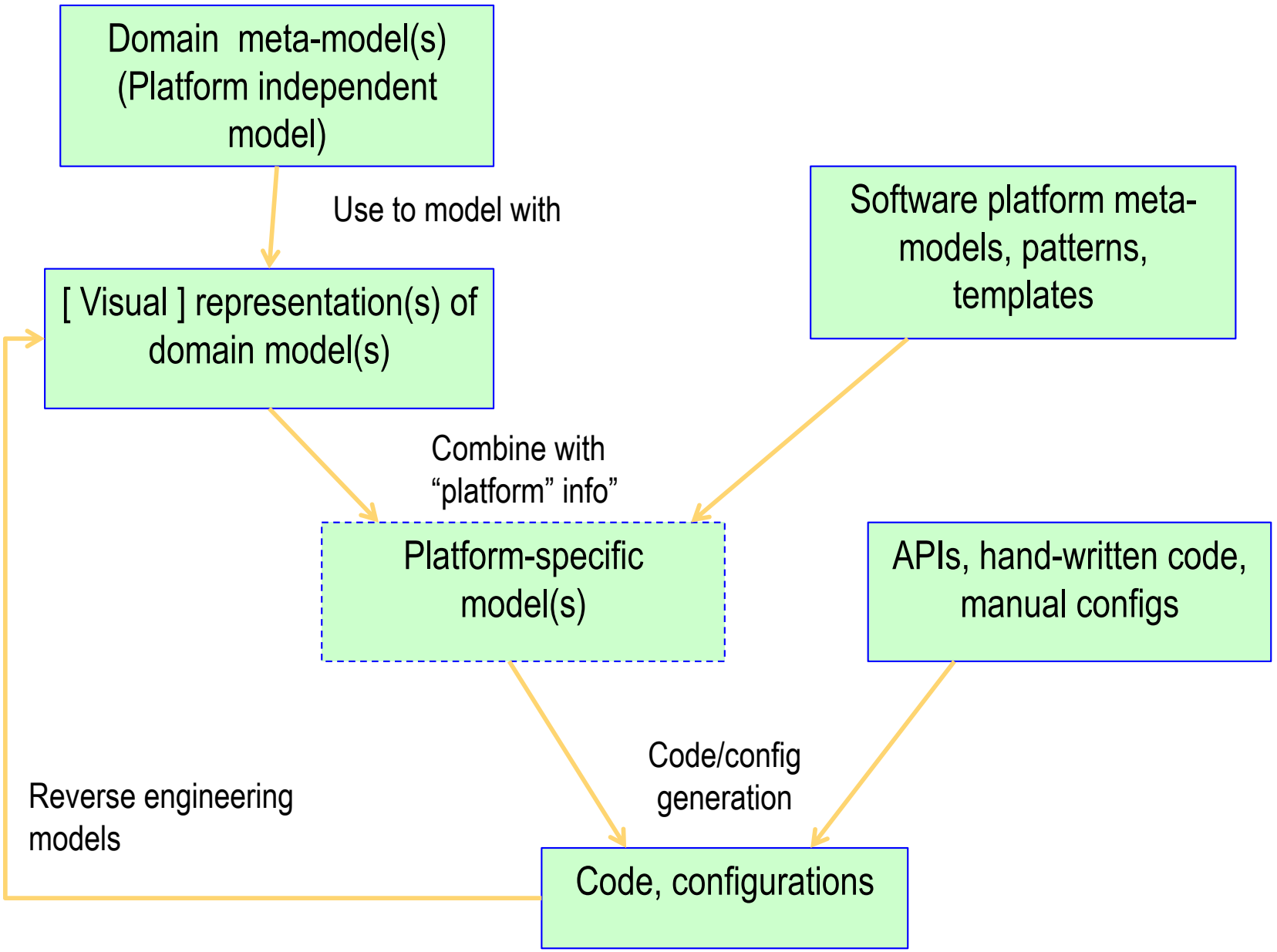
- Doug Schmidt, COMPUTER Guest Intro on MDE:

  Model-driven engineering technologies offer a promising approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively.

- Programming languages (3GLs) too low-level to describe many abstractions in software engineering
- SE models too disconnected from 3GLs (program code) e.g. traditional analysis & design languages
- BUT: such models CAN be used to "construct" software directly
- Need high-level modelling languages to better express requirements, architectures, designs, tests etc BUT that can be directly turned into/related to code constructs
- Need to provide ways to build, reason with models, translate models to(/from) code
- BUT: working directly with code (3GLs) still very useful!!!

Overview of MDE concept

Swinburne



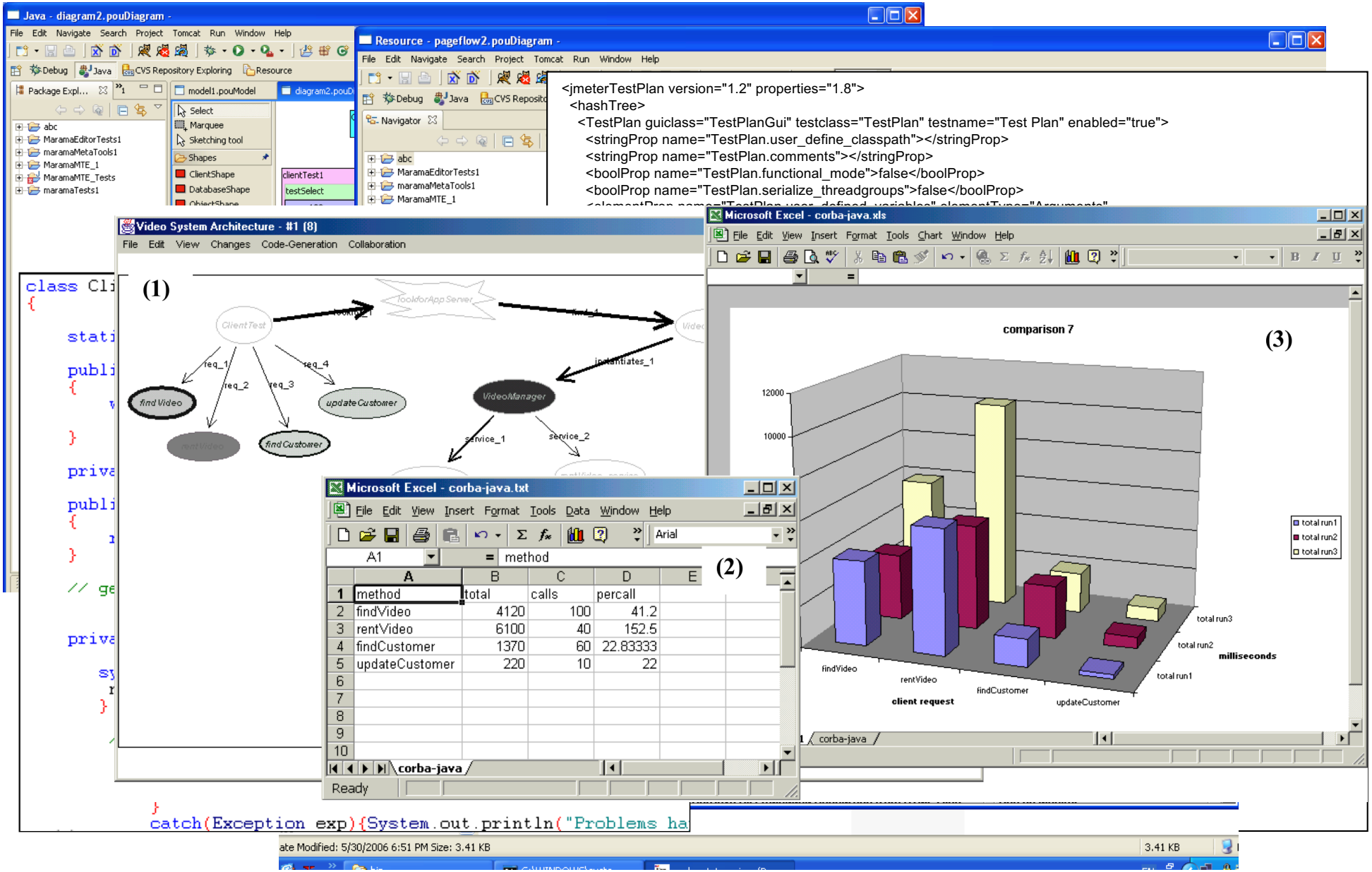
- Domain meta-model(s), models
- Visualisation(s) of domain models – textual and graphical
- Mapping between models
- Editing tools for models
- Transformation support i.e. model->model, model -> code
- Visualisation support e.g. code/data -> model
- Reasoning support e.g. analysis of models – completeness, correctness, consistency
- Model management support e.g. version control, diffing/merging, etc

- MaramaMTE (Performance Engineering)
- Form-based Mapper (Complex Data Mapping)
- MaramaVCPML (Personal Care Plan App generator)
- MaramaEML (Enterprise Modelling Language)
- MaramaAI (Requirements capture)
- VikiBuilder (Visual Wiki generation)

- Performance test generator from high-level architectural models
- Domain models = architecture, usage models
- Platform models = web domain, multi-tier domain, J2EE, .NET architectures and APIs
- Code, configs = Java, C#, JMeter, MS ACT, Selenium
- Visualisation = performance data on architecture data
- With CSIRO and several small-to-medium companies

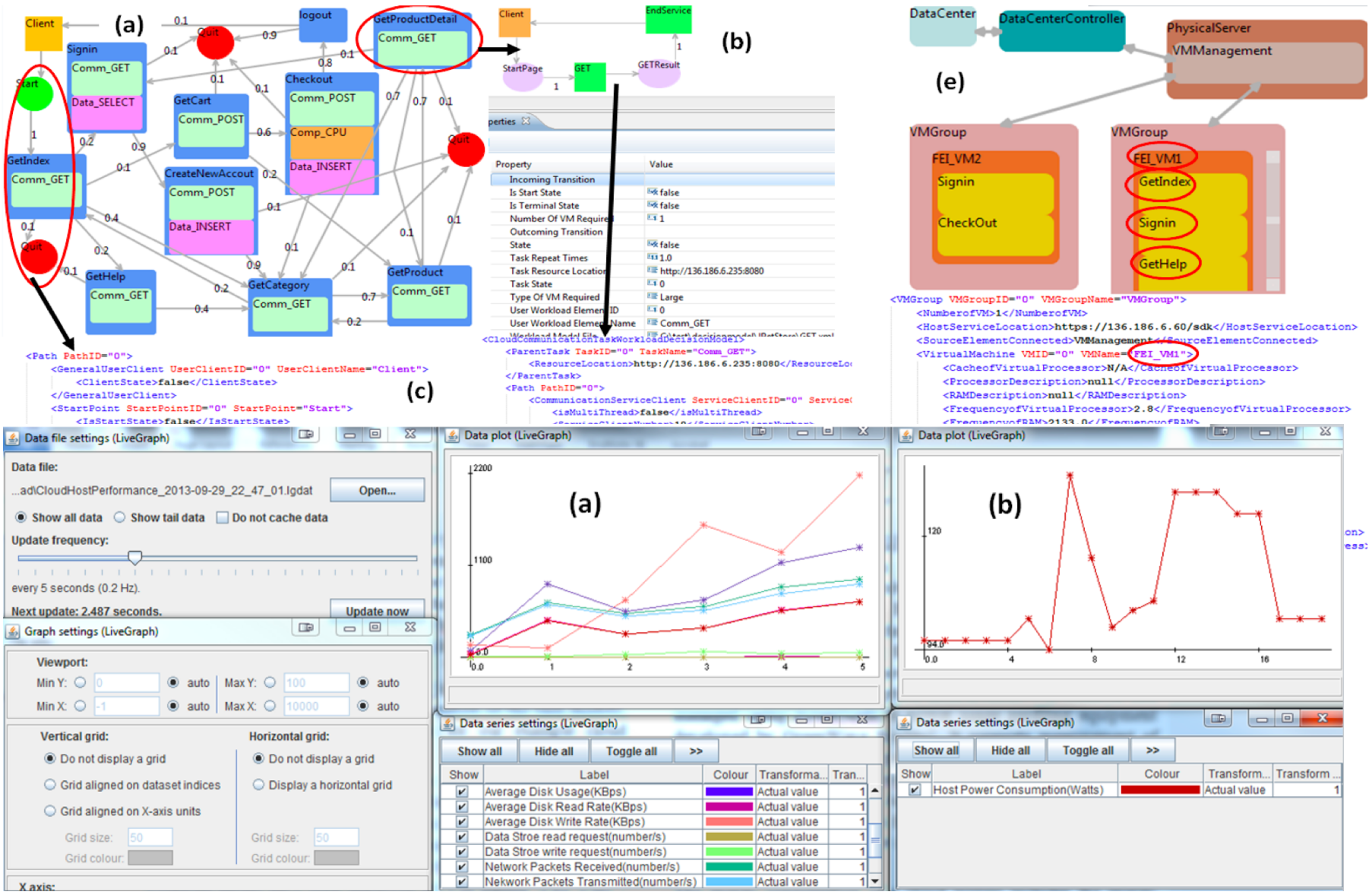
Example

Swinburne



MTE – The Next Generation: “StressCloud”

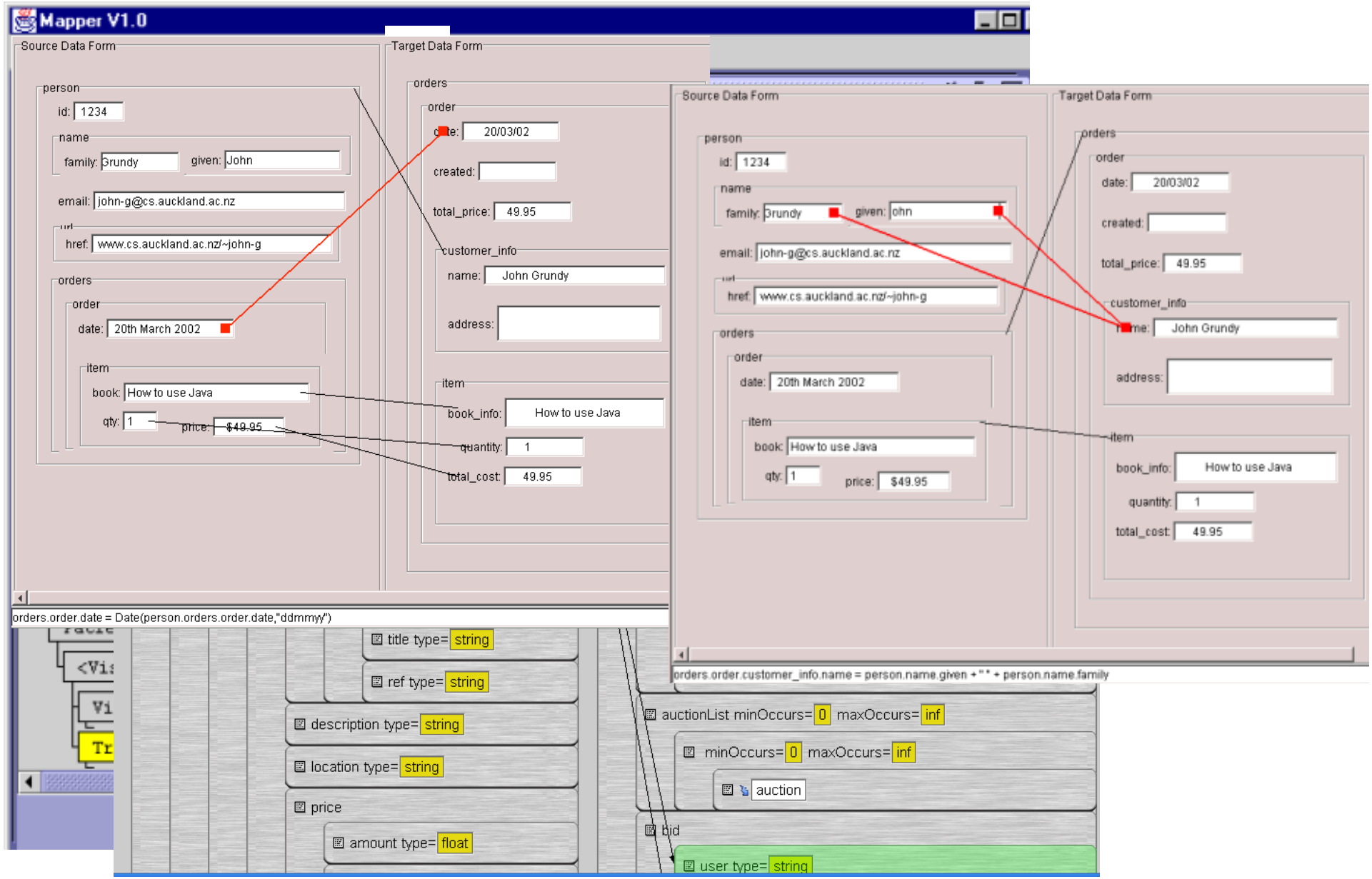
Swinburne



- Scenario: complex XML or EDI message format; want to translate into a different format; many to process
- Traditionally: write QVT/ATL/XSLT/code to do
- Alternative: model transformation visually and generate these transformation implementations
- Meta-model = source/target and mappings
- Visual models might include forms, trees, concrete data visualisations
- MDE = generate XSLT, ATL, Code (C++, Java),…
- Done various with Orion Health Ltd, XSOL Ltd, NICTA

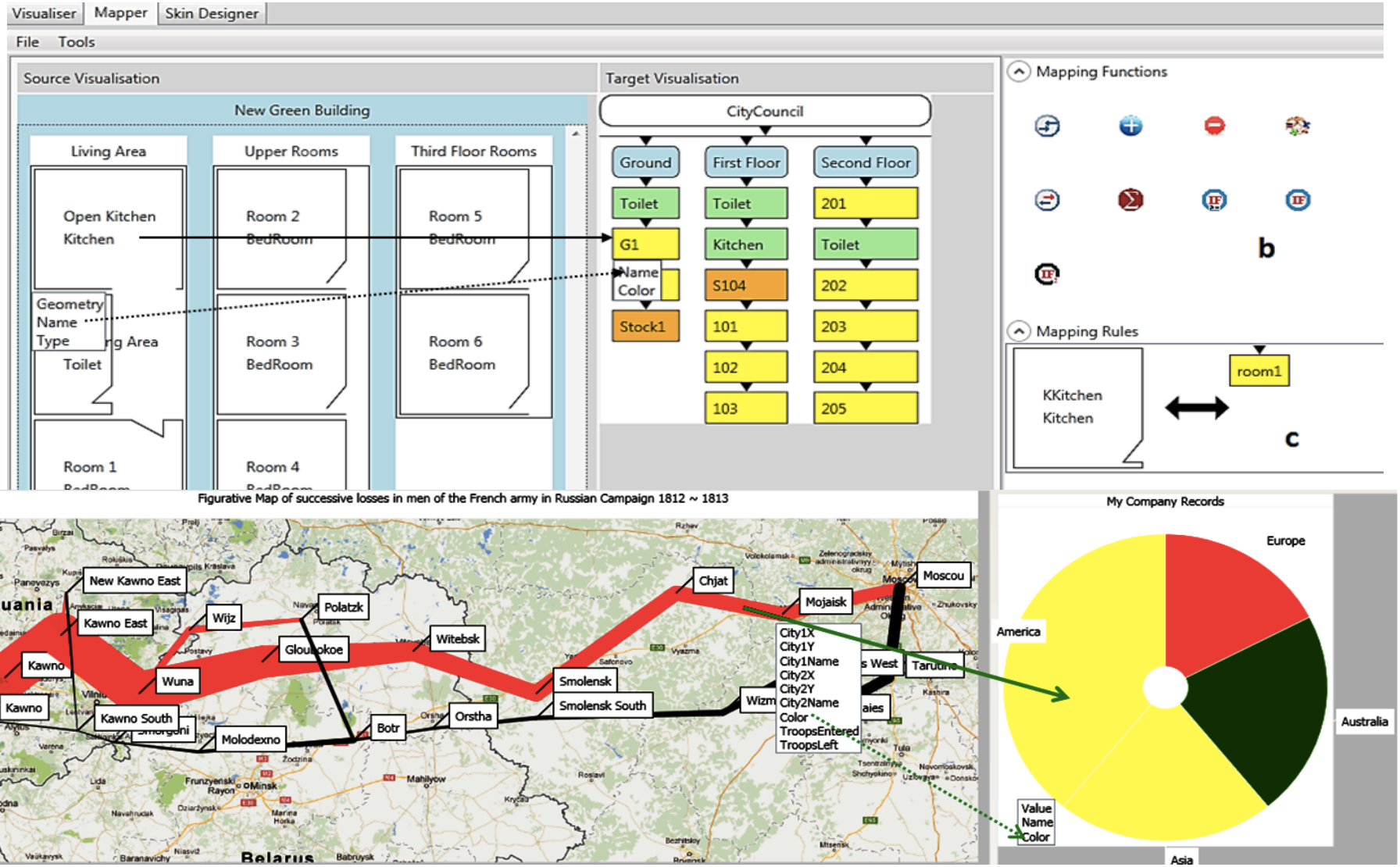
Examples

Swinburne

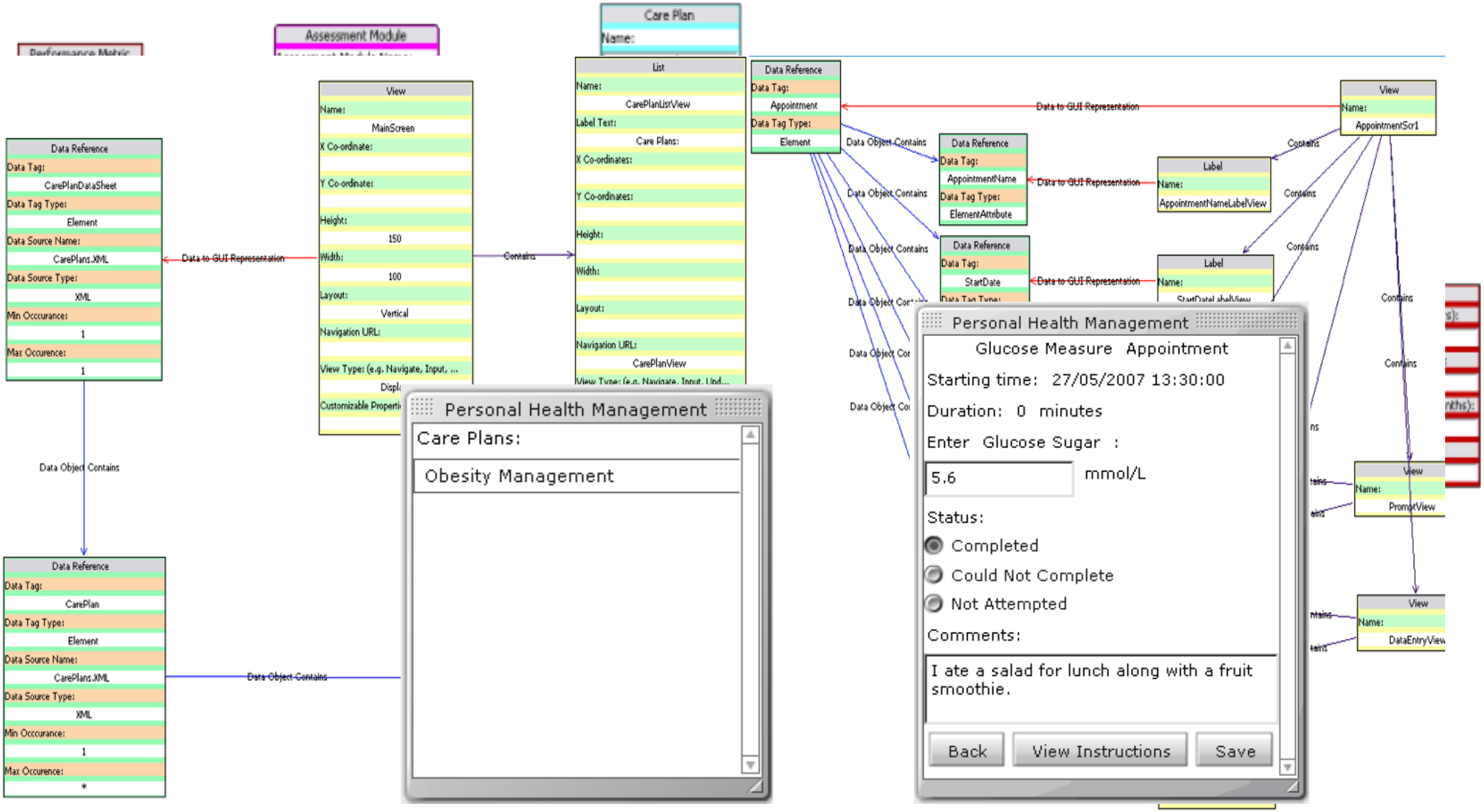


The Next Generation #2: CONVErT

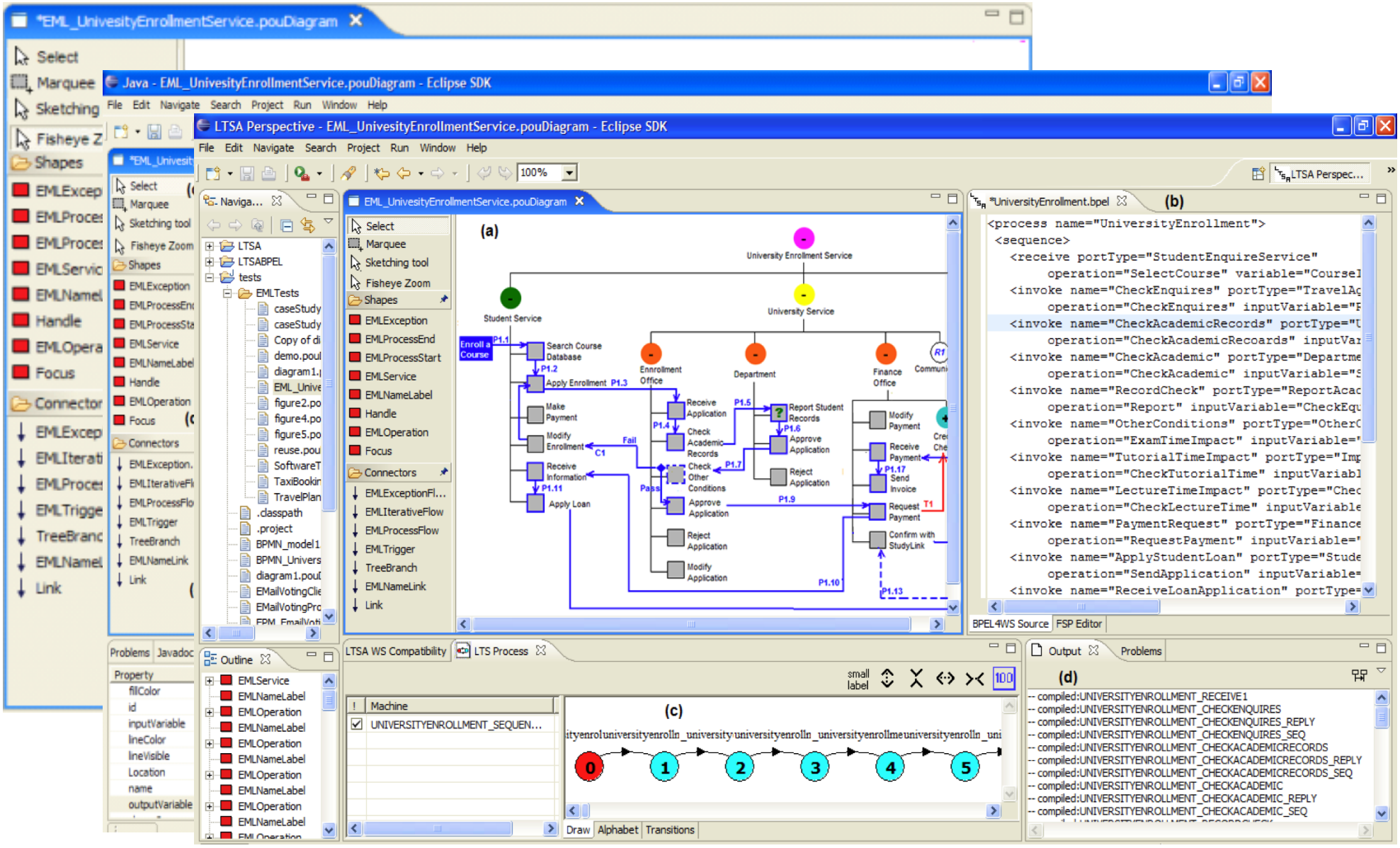
Swinburne



- Mobile phone-based personal health care planning applications
- Two meta-models with associated DVSLs: Visual Health Care Planning Language, Visual Care Application Model
- Model generic care plan with a visual DSVL tool
- Configure generic care plan for individual
- Model mobile app UI for individual from tailored care plan with a visual DSVL tool
- Generate Flash, Windows Mobile, iPhone app code


Assessment Module
Care Plan
Name:
Data Reference
Data Tag:
CarePlanDataSheet
Data Tag Type:
Element
Data Source Name:
CarePlans.XML
Data Source Type:
XML
Min Occurance:
1
Max Occurence:
1
Data to GUI Representation
View
Name:
MainScreen
X Co-ordinate:
Y Co-ordinate:
Height:
150
Width:
100
Layout:
Vertical
Navigation URL:
View Type: (e.g. Navigate, Input, ...
Contains
List
Name:
CarePlanListView
Label Text:
Care Plans:
X Co-ordinates:
Y Co-ordinates:
Height:
Width:
Layout:
Navigation URL:
CarePlanView
Data Object Contains
Data Reference
Data Tag:
CarePlan
Data Tag Type:
Element
Data Source Name:
CarePlans.XML
Data Source Type:
XML
Min Occurance:
1
Max Occurence:
*
Personal Health Management
Care Plans:
Obesity Management
Data Reference
Data Tag:
Appointment
Data Tag Type:
Element
Data Reference
Data Tag:
AppointmentName
Data Tag Type:
ElementAttribute
Label
Name:
AppointmentNameLabelView
Data Reference
Data Tag:
StartDate
Label
Name:
View
Name:
AppointmentScr1
View
Name:
PromptView
View
Name:
DataEntryView
Personal Health Management
Glucose Measure Appointment
Starting time: 27/05/2007 13:30:00
Duration: 0 minutes
Enter Glucose Sugar :
5.6
mmol/L
Status:
Completed
Could Not Complete
Not Attempted
Comments:
I ate a salad for lunch along with a fruit smoothie.
Back
View Instructions
Save

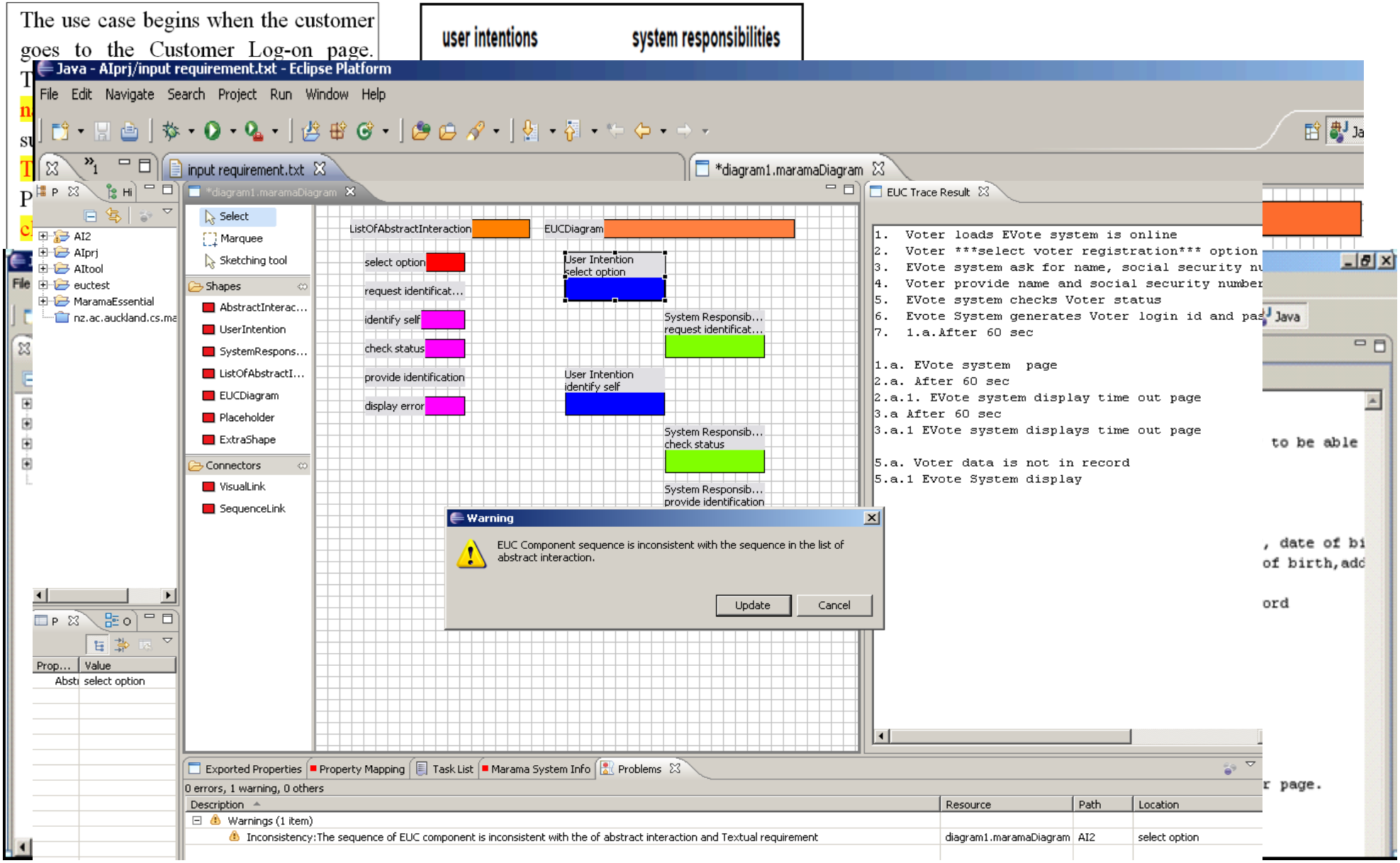
- Enterprise modelling tool
- Integrated domain meta-model synthesized from several existing & new models (BPML, EML, ViTABaL-WS, …)
- Multiple views with different DSVLs
- Platform meta-model & “code” = BPEL – executable process modelling language (a DSL)
- Tool support for large scale diagram management
- Tool support for model checking for integrated LTSA tool


*EML_UnivesityEnrollmentService.pouDiagram
Java - EML_UnivesityEnrollmentService.pouDiagram - Eclipse SDK
LTSA Perspective - EML_UnivesityEnrollmentService.pouDiagram - Eclipse SDK
File Edit Navigate Search Project Run Window Help
(a)
(b)
(c)
(d)
University Enrollment Service
University Service
Student Service
Enroll a Course
Search Course Database
Apply Enrollment
Make Payment
Modify Enrollment
Receive Information
Apply Loan
Enrollment Office
Receive Application
Check Academic Records
Check Other Conditions
Approve Application
Reject Application
Modify Application
Department
Report Student Records
Approve Application
Reject Application
Finance Office
Modify Payment
Receive Payment
Send Invoice
Request Payment
Confirm with StudyLink
Fail
Pass
<process name="UniversityEnrollment">
<sequence>
<receive portType="StudentEnquireService"
<invoke name="CheckEnquires" portType="TravelAg
<invoke name="CheckAcademicRecords" portType="U
<invoke name="CheckAcademic" portType="Departme
<invoke name="RecordCheck" portType="ReportAcac
<invoke name="OtherConditions" portType="OtherC
<invoke name="TutorialTimeImpact" portType="Imp
<invoke name="LectureTimeImpact" portType="Chec
<invoke name="PaymentRequest" portType="Finance
<invoke name="ApplyStudentLoan" portType="Stude
<invoke name="ReceiveLoanApplication" portType=
BPEL4WS Source
FSP Editor
LTSA WS Compatibility
LTS Process
Machine
UNIVERSITYENROLLMENT_SEQUEN...
Draw
Alphabet
Transitions
Output
Problems
-- 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

- Requirements capture and analysis tool
- Textual requirements -> essential interactions -> Essential Use Cases -> UIDs, OOAs
- Domain meta-models include natural language (!), EUCs, UML etc
- Textual and visual representations of domain models
- Transformation of text to/from EUC-based DSVL models
- Analysis of consistency between models, completeness/correctness of models via EUC pattern library

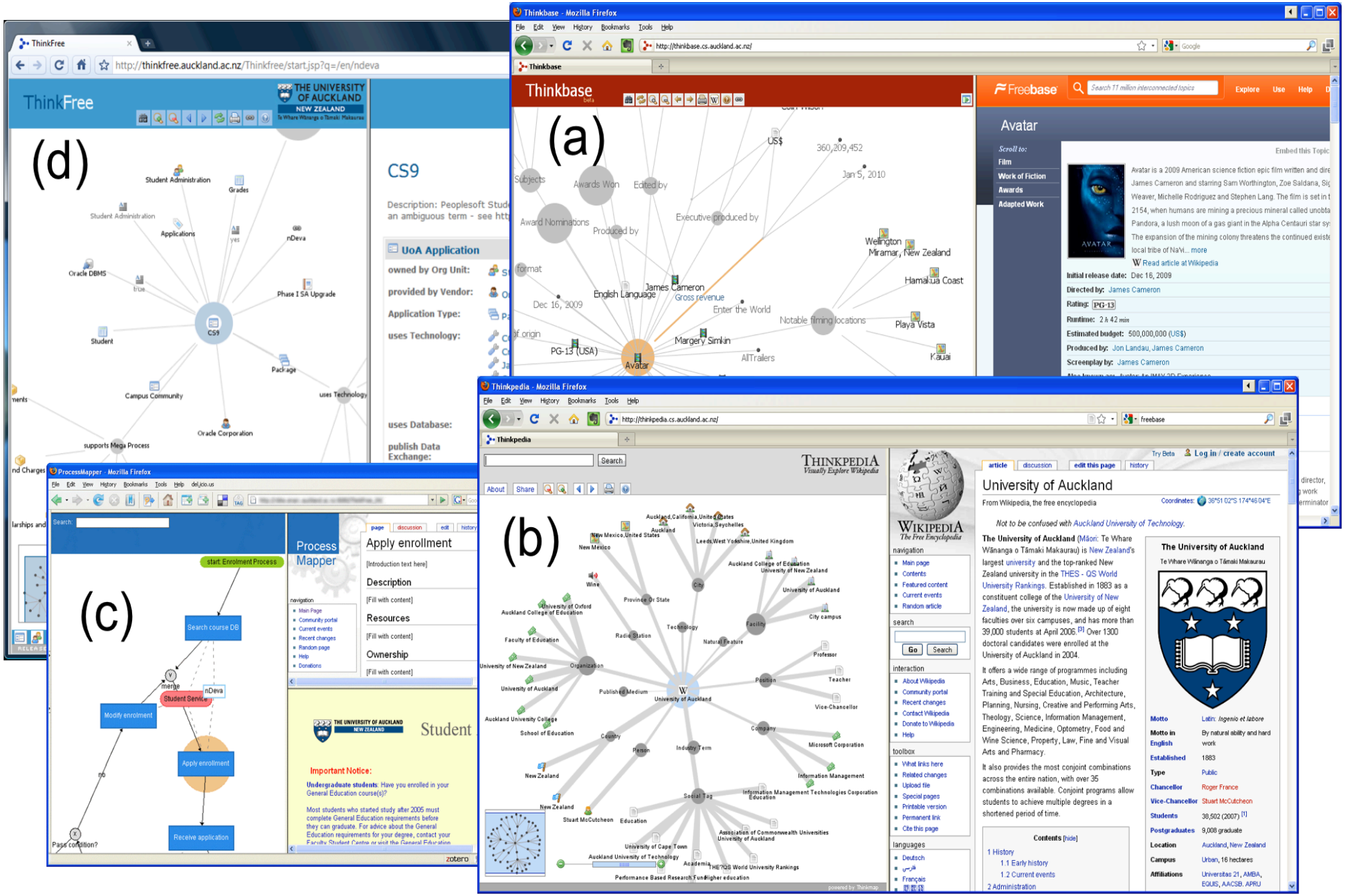
Examples

Swinburne



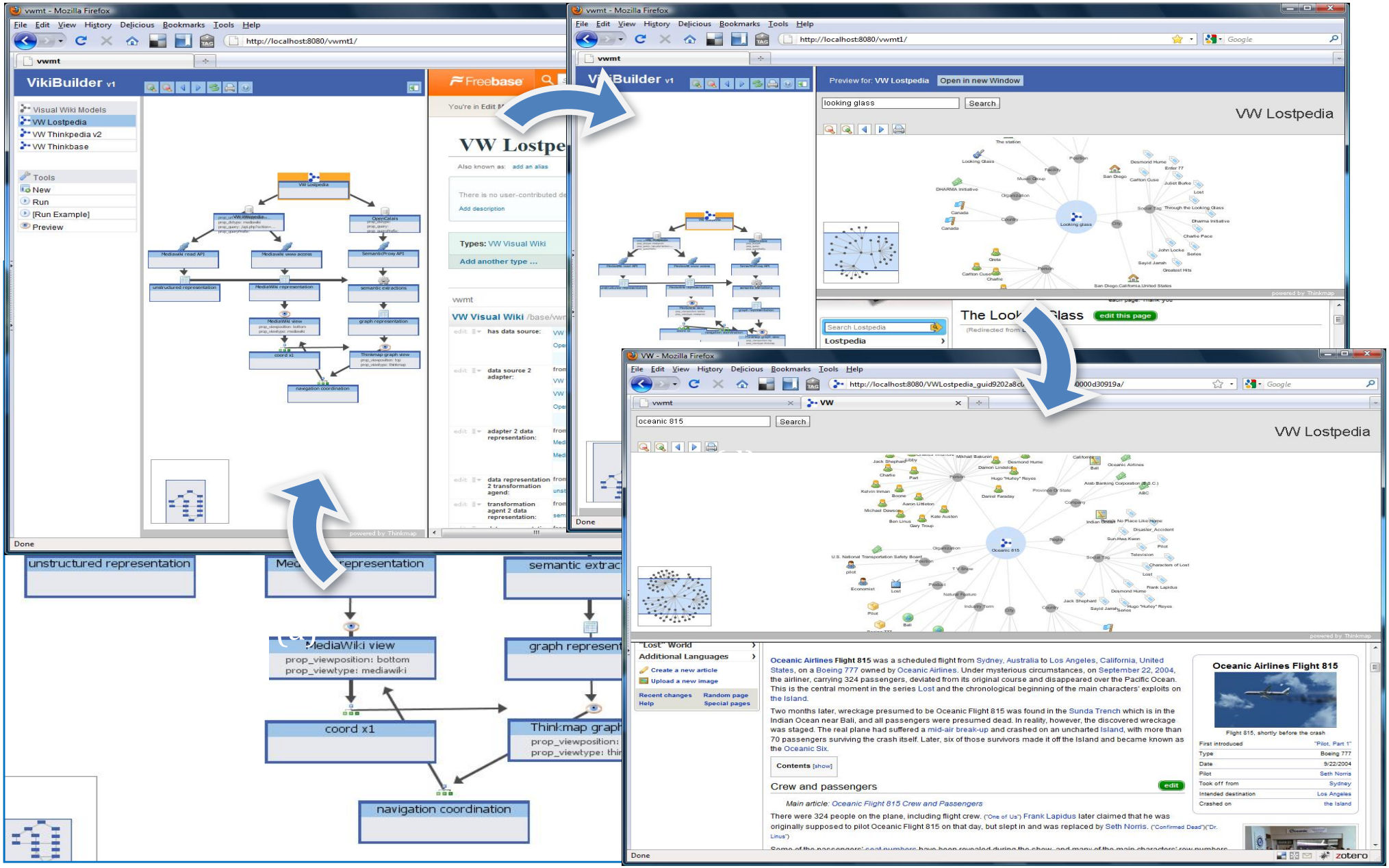
Visual Wikis (“Vickis”) & VickiBuilder

Swinburne



Example

Swinburne



- Higher level models to work with – easier to create, understand, modify, do analysis on than 3GL code
- Can refine models, ultimately to code/configurations
- Repetitive stuff (“construction”) can be largely (sometimes completely) automated

- Models get complex, need good tools to work with them
- Hard to agree on standards esp. in new areas
- Model and tool evolution gets very tricky
- Trade-off between flexibility (code) and productivity (MDE)…

Is MDE “Engineering”?

- Using meta-models, model representations
- Can check (some aspects of) models in tools
- Repeatable processes
- Productivity, quality improvements
- “Construction” becomes push-a-button

- Best practices in MDE?
- Dynamic MDE (change while running)?
- Proactive MDE (change while running in anticipation of problems…)??

Future work

- Improve design of meta-models including specify complex constraints
- Improve design and evaluation of DSVLs
  - Physics of Notations Tool – new PhD project
- Improve specification of transforms e.g. by using CONVErT ☺
- Improve tool support for DSVLs, transforms, “design critics”
  - Horus – our next-generation Web-based DSVL tool
- Support other things via MDE e.g. security, HPC, cloud platforms
  - Horus HPC, MDSE @ Runtime, TOSSMA, SMURF
- Proactive adaptation of complex systems via MDE
  - My 2016 DP application... ☺
- Assess MDE on range of real-world problems
- Identify when “best practice” to use, when not to use…

Thank yous

Swinburne

- John Hosking – most of these projects
- Jun Huh, Karen Li – Marama meta-tools
- Rainbow Cai, Feifei Chen – MaramaMTE, StressCloud
- Abizer Khambati – VCPML etc
- Michael Li – Form-based Mapper
- Massila Kalmalrudin – MaramaAI
- Christian Hirsch – VisualWikis, VickiBuilder
- Iman Avazpour – CONVErT
- Mohamed Almorsy – Horus, Horus HPC, MDSE @ R, …
- Many other contributors over many years…

In case your are interested…

Swinburne

[Marama] Grundy, J.C., Hosking, J.G., Li, N., Huh, J., Ali, M., Li, L. Generating Domain-Specific Visual Language Tools from Abstract Visual Specifications, IEEE Transactions on Software Engineering, 2013

[CONVErT] Avazpour, I., Grundy, J.C., Vu, H. Generating Reusable Visual Notations using Model Transformation, to appear in International Journal of Software Engineering and Knowledge Engineering, 2015.

[VickiBuilder] Hirsch, C., Hosking, J.G. and Grundy, J.C. VikiBuilder: end-user specification and generation of Visual Wikis, In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010), Antwerp, Belgium, 20-24 Sept 2010, IEEE CS Press.

[MaramaAI] Kalmalrudin, M., Grundy, J.C. and Hosking, J.G. Improving Requirements Quality using Essential Use Case Interaction Patterns, In Proceedings of the 2011 International Conference on Software Engineering (ICSE2011), Honolulu, Hawaii, USA, May 21-28 2011, ACM Press.

[Orion Mapper] Grundy, J.C., Mugridge, W.B., Hosking, J.G. and Kendal, P. Generating EDI Message Translations from Visual Specifications, In Proceedings of the 16th International Conference on Automated Software Engineering, San Diego, 26-29 Nov 2001, IEEE CS Press, pp. 35-42

[Form-based Mapper] Li, Y., Grundy, J.C., Amor, R. and Hosking, J.G. A data mapping specification environment using a concrete business form-based metaphor, In Proceedings of the 2002 International Conference on Human-Centric Computing, IEEE CS Press

[Visual Care Plan Modeller] Khambati, A., Warren, J., Grundy, J., and Hosking, J. Care Planning Systems for Consumer Engagement in Chronic Disease Management, Electronic Journal of Health Informatics, vol 4, no. 1, 2009

[MaramaEML] 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.

[MaramaMTE] Cai, Y., Grundy, J.C. and Hosking, J.G. Synthesizing Client Load Models for Performance Engineering via Web Crawling, In Proceedings of the 2007 IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), Atlanta, Nov 5-9 2007, IEEE CS Press

Thanks!

Ministry of Business, Innovation & Employment

FRST SER SPPI
FRST NERF DS Tools

DP110101340
DP120102653
DP140102185

SWINBURNE UNIVERSITY OF TECHNOLOGY

Malaysian Ministry of Education

Build IT
Building research capability