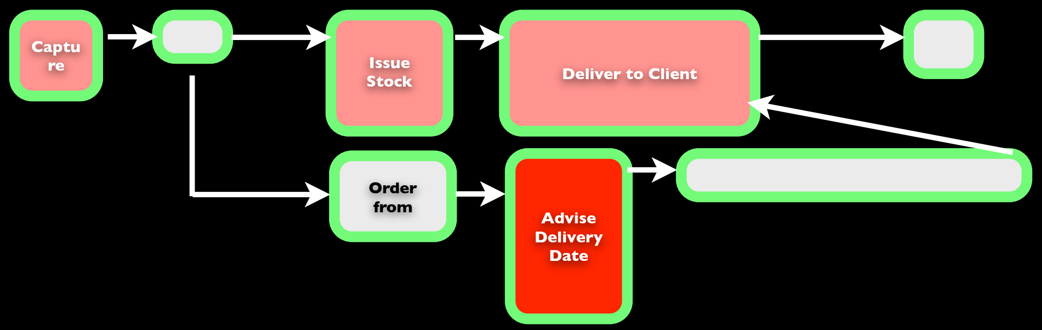
An Advanced Meta-meta Model for Visual Language Design and Tooling

Targeting a Property Graph Implementation



Graham McLeod

Contribution to Models at Work
Hendon, UK
November 2022

Duisburg-Essen University

inspired.org

Outline

Research Context

Requirements

Contributory Products, Works and Ideas

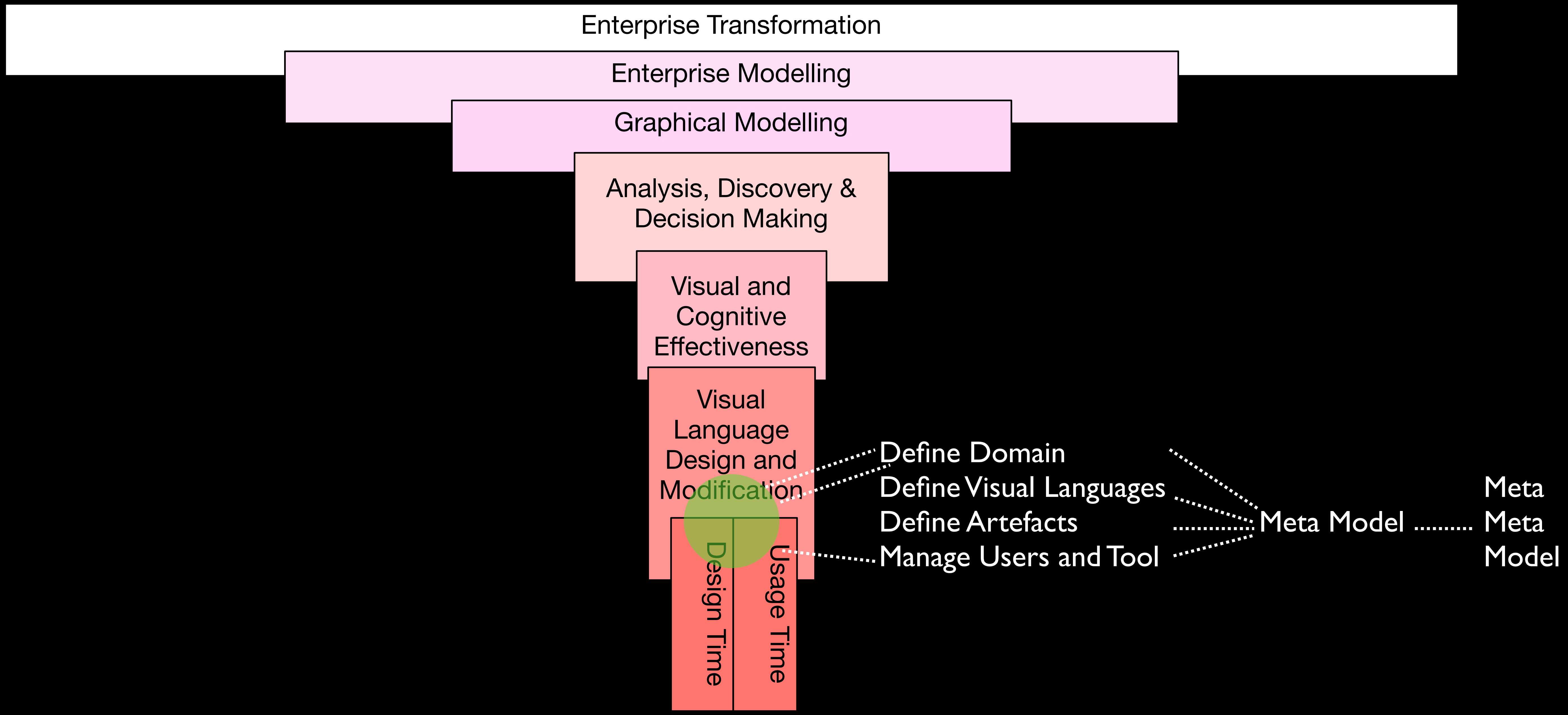
Design Choices

Resultant Model

Status and Reflection

References

Situating Research



Goals

G1: Support rich meta modelling / ontology definition which allows fully expressing concepts of the domain accurately

G2: Support definition of notations which are appropriate to the domain concept, the stakeholders who will work with them and the purpose of the modelling

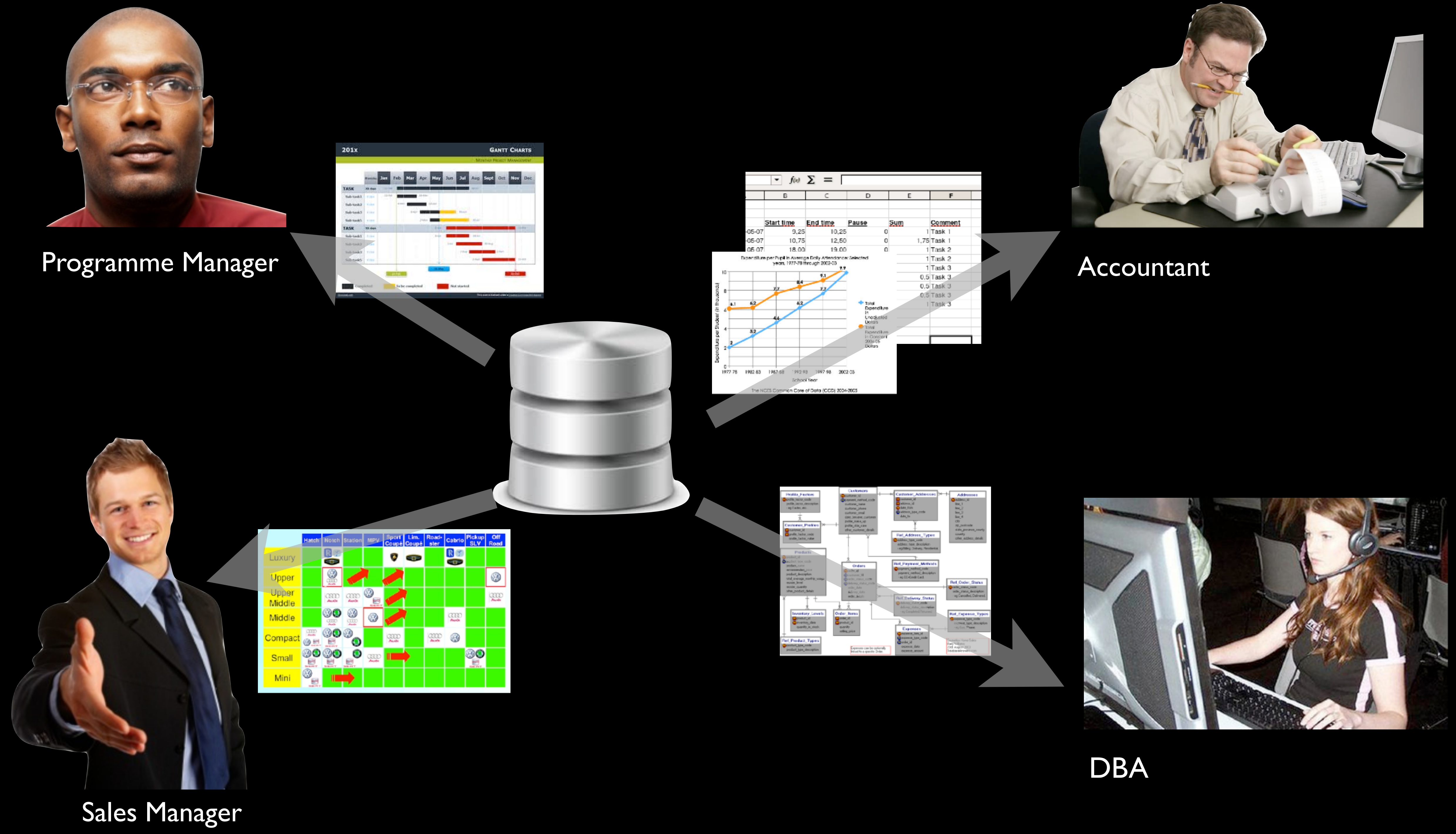
G3: Support rapid/iterative evolution of the meta model and visual language to continually improve effectiveness

G4: Support run time tailoring of models, meta model, visual language and tool user interface to support unique requirements ("moldable tools", in the spirit of: [Chiş et al. 2015])

G5: Permit multiple representations for the same semantic models, addressing the needs of different stakeholders and catering for multiple visual languages / methods

G6: As far as possible, achieve an economical implementation of the above capabilities to facilitate implementation of supporting tooling at reasonable effort / cost.

Stakeholders and Concerns



What is their orientation?

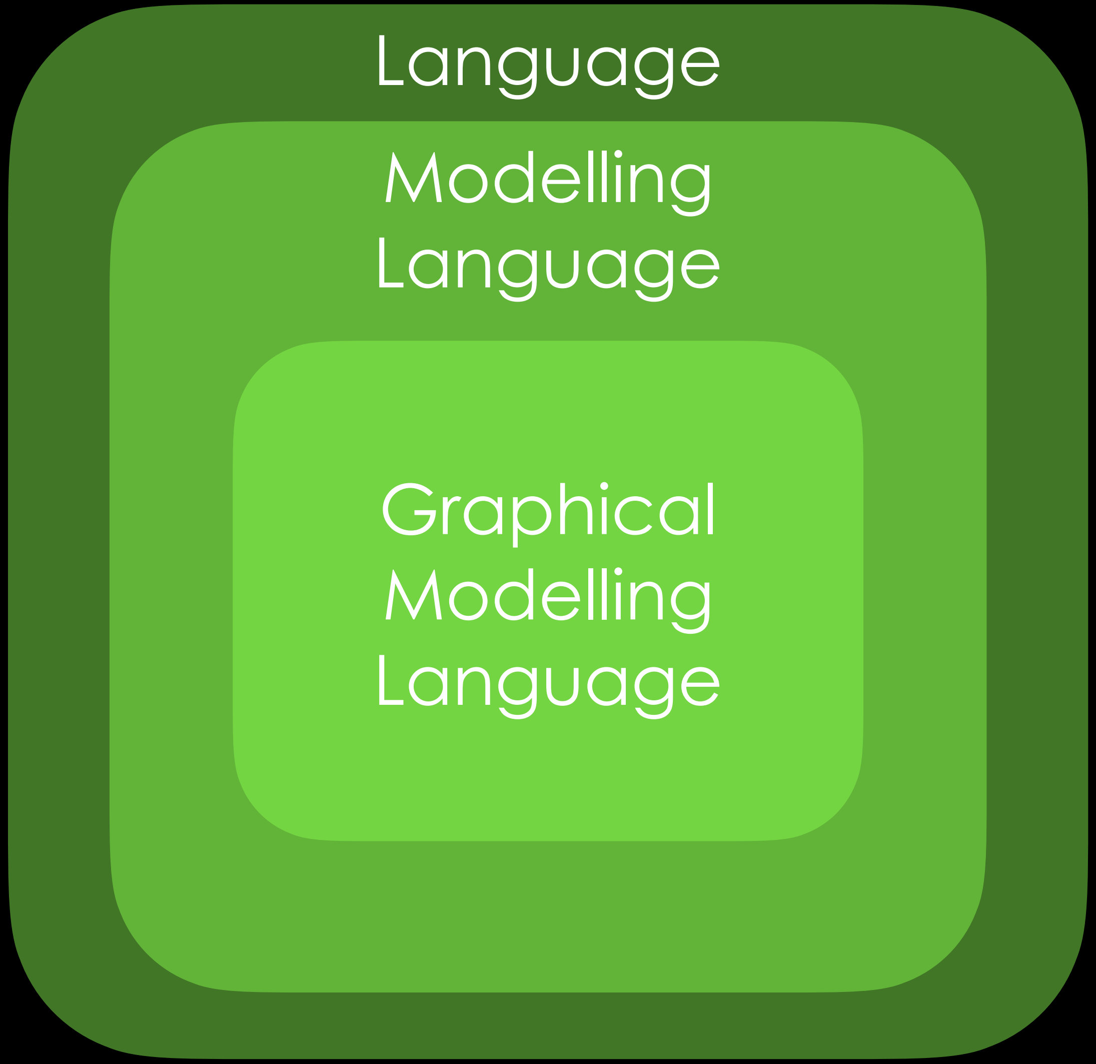
What are they familiar with?

What is their level of literacy wrt models / notation format?

What are their concerns?

What models and representation will be effective and efficient?

Language

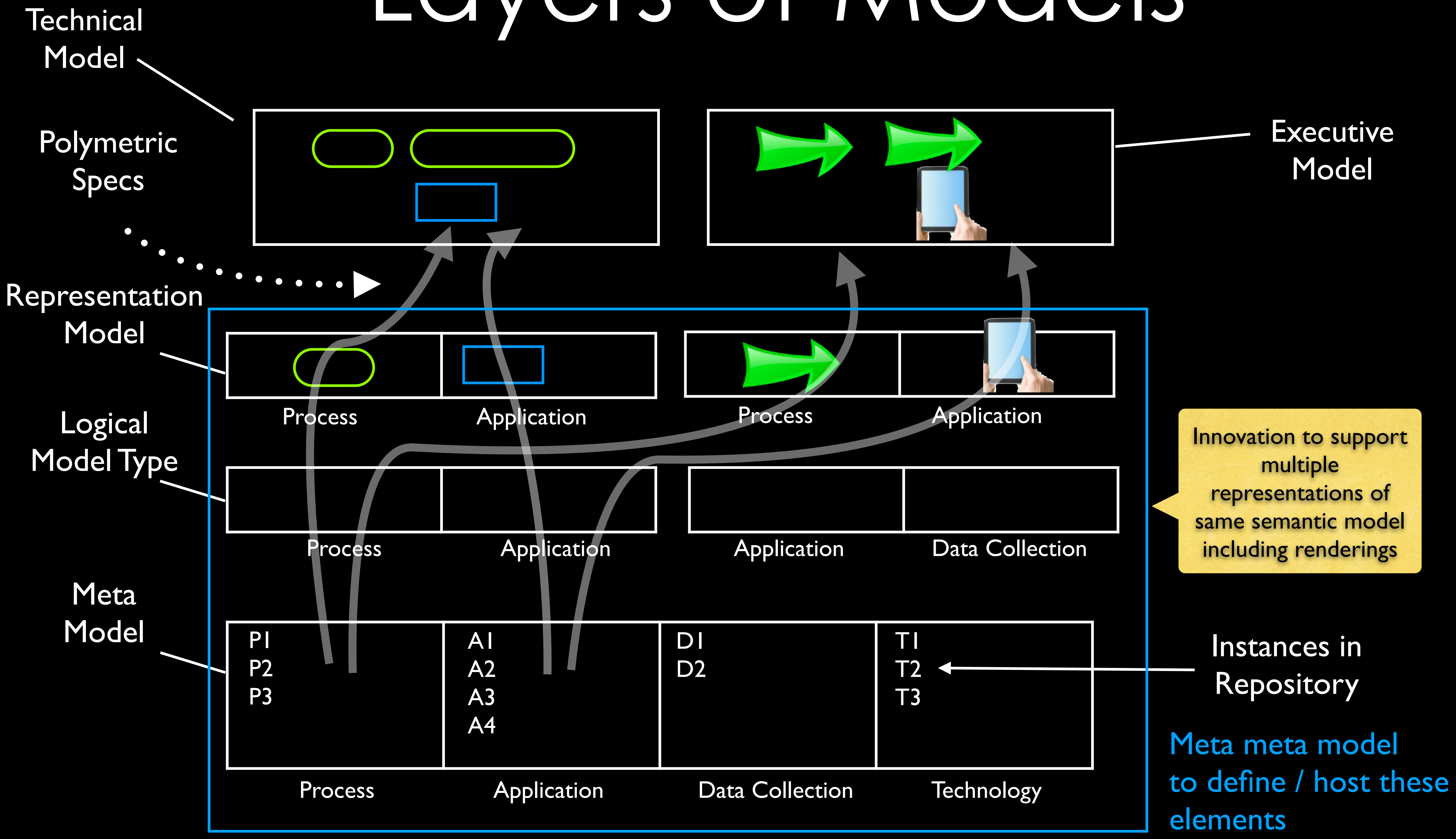


A way to communicate between parties

A way to communicate precisely between parties using an agreed vocabulary and grammar

A way to communicate precisely between parties using visual symbols, connectors, containers and their arrangement following an agreed notation, representation and (potentially) layout

Layers of Models



Requirements

R1: Multi-level modelling must be accommodated to support advanced modelling and to economise on tool size by allowing similar tools to be used on multiple levels as well as support the runtime extension of the domain model/ontology. See [Lara et al. 2014] for a discussion of why and how to use multi-level approach

R2: High level of abstraction to achieve efficiency in model type definitions and agility in changing them when required. This implies a declarative versus procedural solution and prefers configuration over code. [Däcker and Williams 1997, Hartmann and Both 2009] provide evidence for power of abstraction in software and modelling

R3: Support for n-ary relationships and relationship properties. These are necessary to support some types of modelling e.g. [Chen 1976]

R4: Cater for rich data types, provided by the implementation environment, tool classes developed in the implementation language and structures created by users themselves through model definition. We have found this invaluable in our earlier work in the imple- mentation of the EVA Toolset [Inspired.org 2022]

R5: Allow extension of the meta model and notation at run time

R6: Support a rich variety of diagram types and notations as well as facilitate other types of output (e.g. lists, reports, composed docu- ments, matrices, graphs, visualisations)

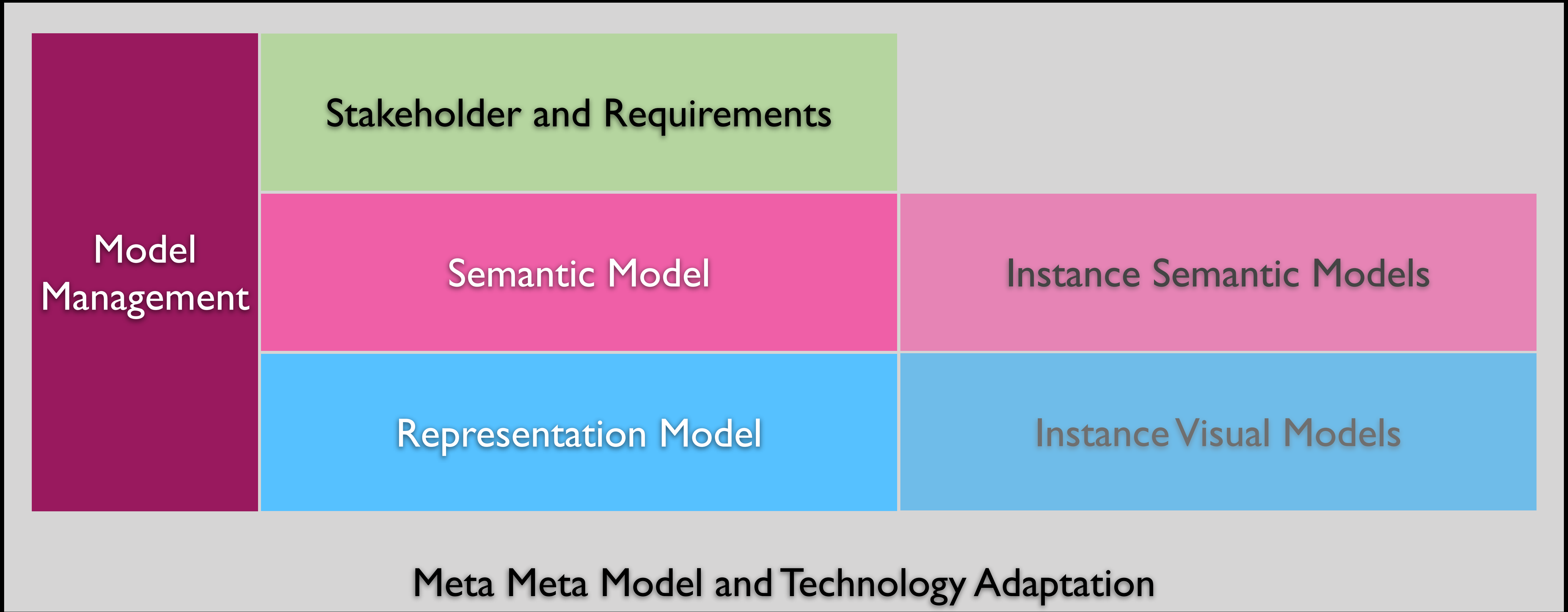
R7: Support modelling the sequence and grouping required of items

R8: Support definition of validation, constraints, derivation through configurable rules/methods

R9: Provide for documentation of modelling language and evolution/ versioning

R10: Support management of collections of things for retrieval in queries, reports and tooling

Meta Model Components



Influences and Inspirations

Enterprise Value Architect - inspired.org

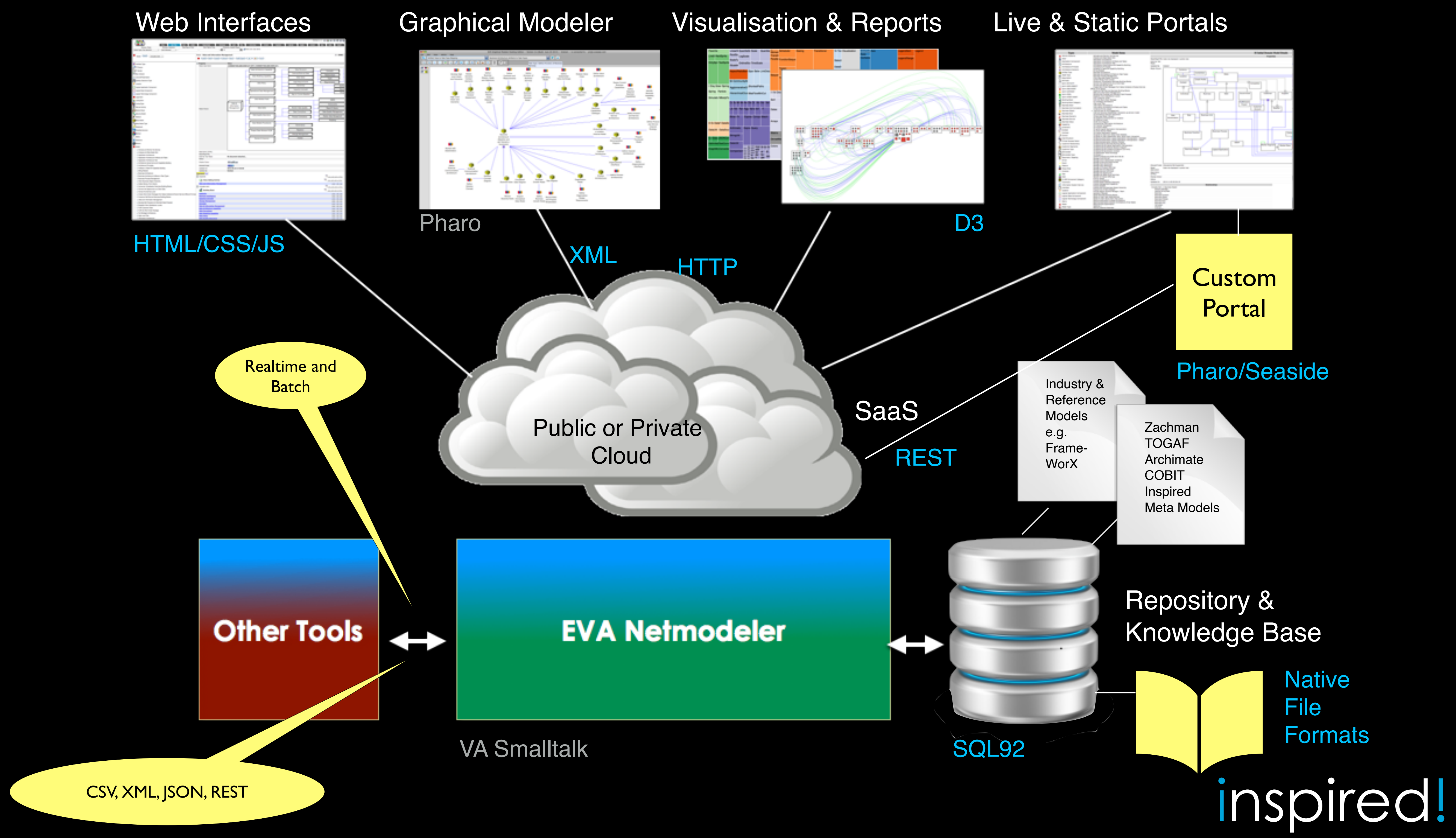
MetaEdit+ - MetaEdit

XModeler - Tony Clark

Memo Meta Modelling Language - Ulrich Frank

Semantic Technologies, RDF, Triple Stores and Graph Databases

Enterprise Value Architect



MetaEdit+ GOPPR

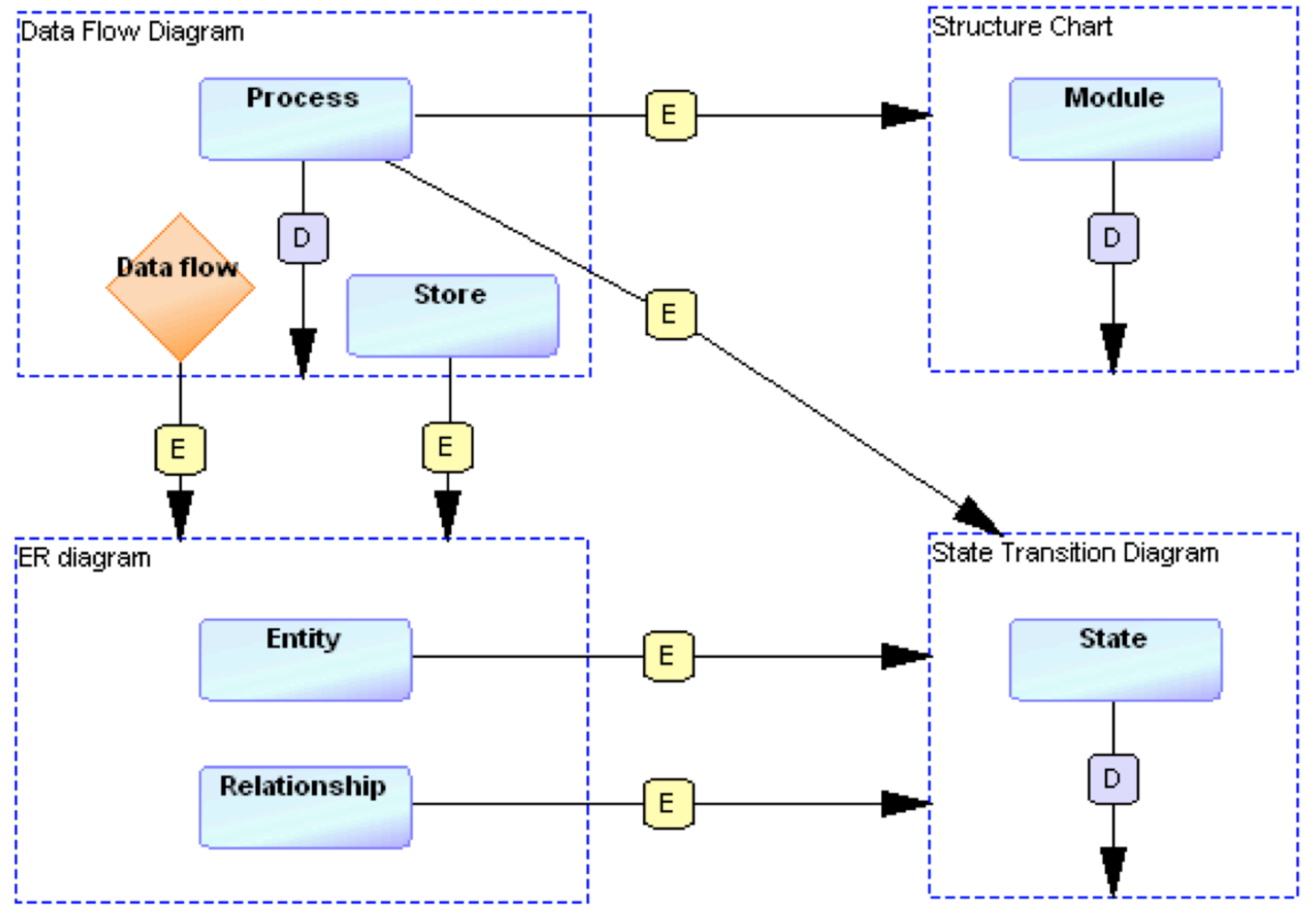
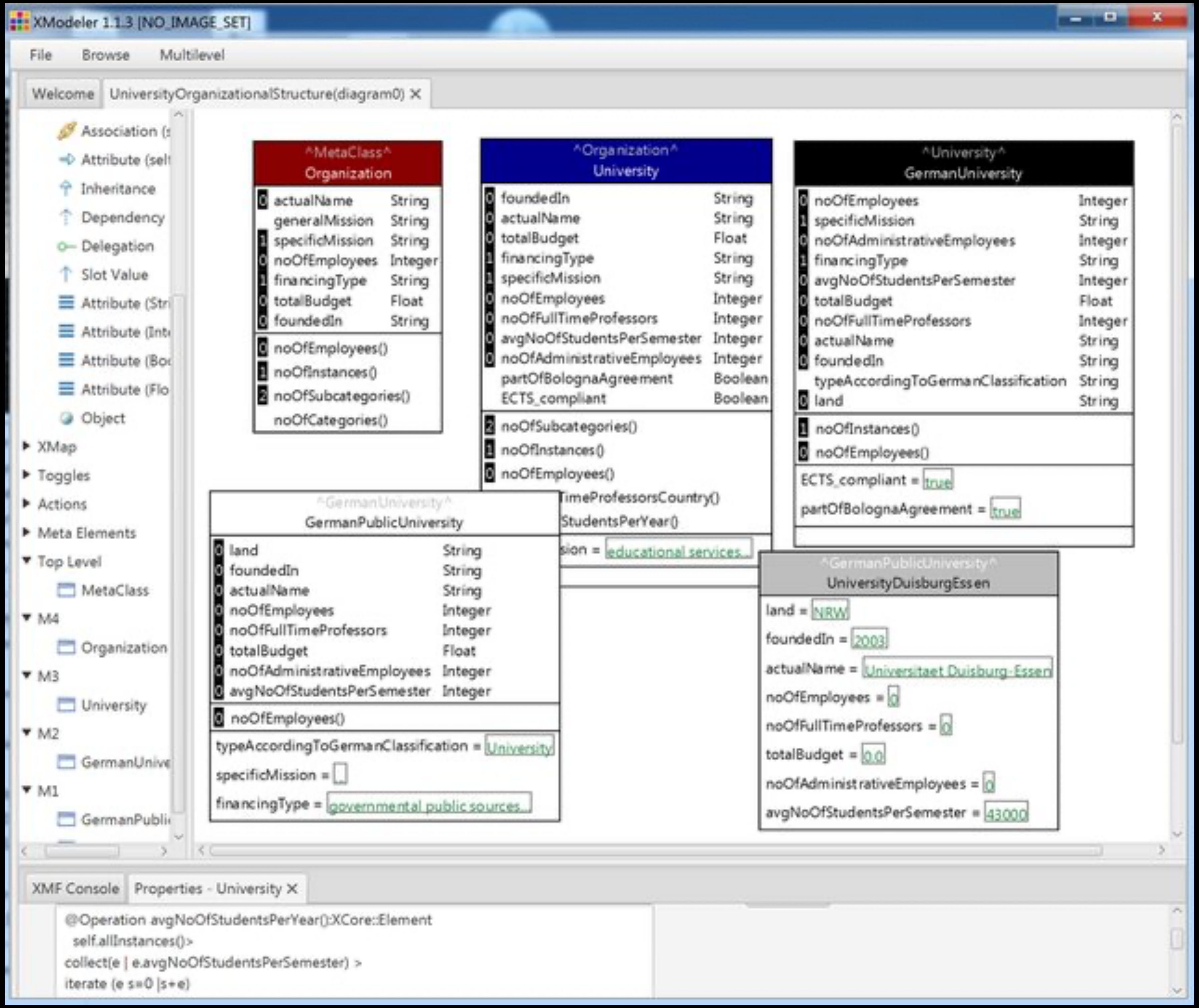


Figure 2-1. Metamodel for multiple graph types

Graph, Object, Property, Port, Role, Relationship

A graphical DSL for implementing DSLs

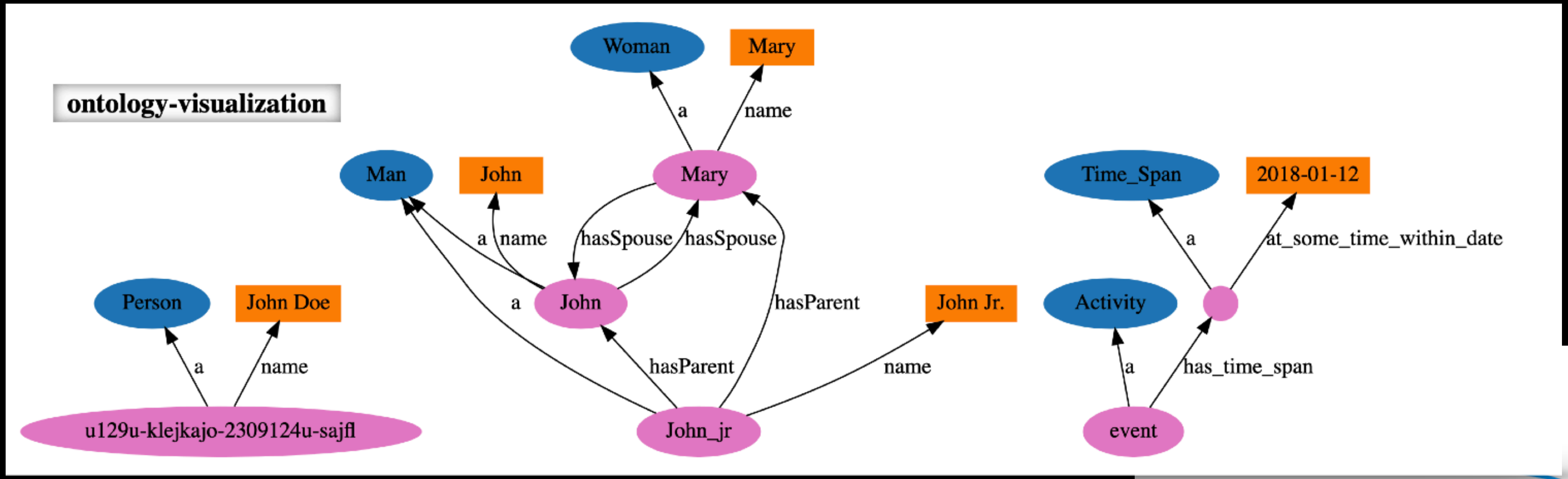
Multi-Level Model in XModeler



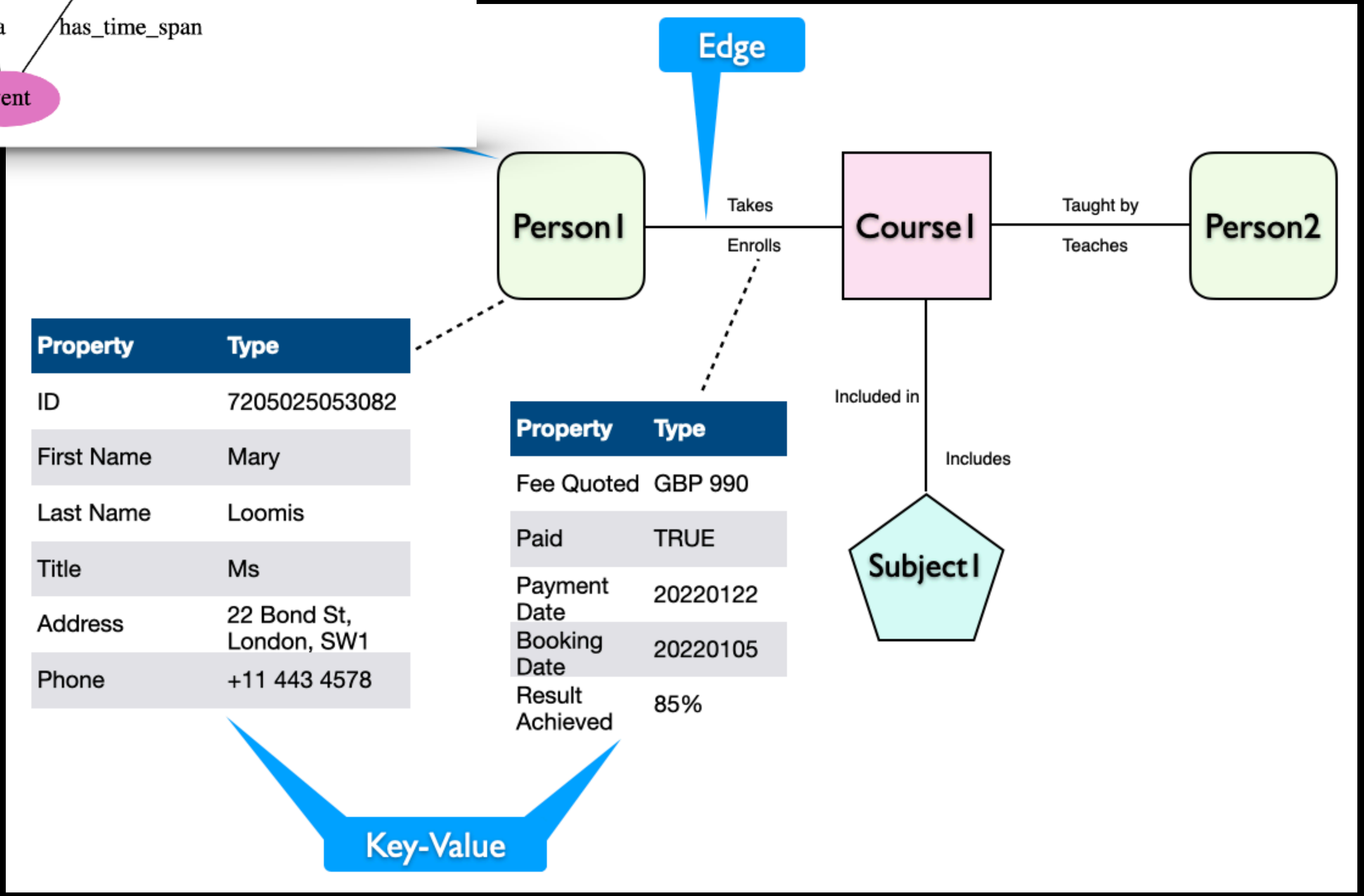
Also used to implement the MEMO Meta Model from Frank

Software and Systems Modeling
Published by Springer Nature
Online ISSN: 1619-1374

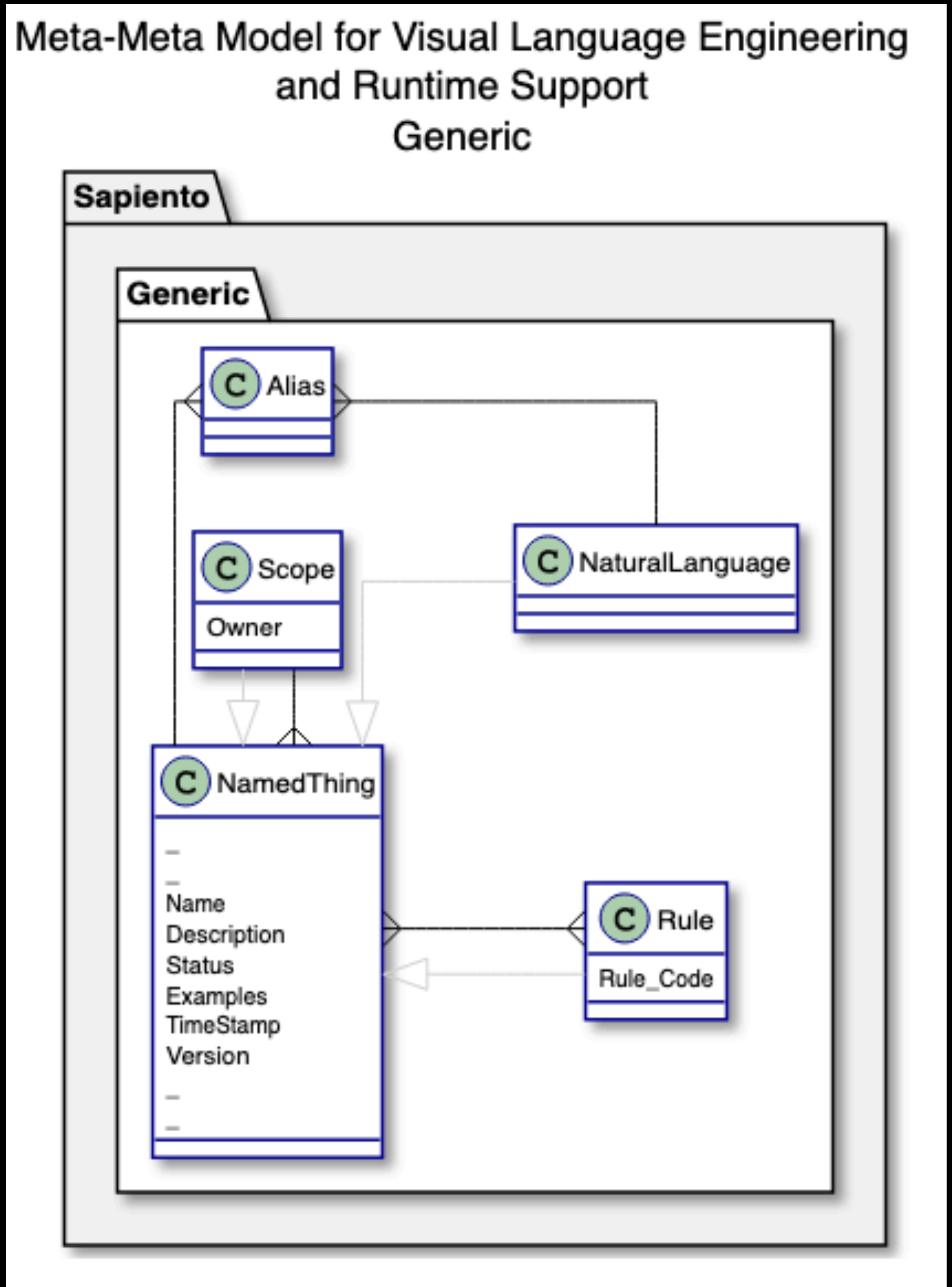
RDF vs Property Graph



Source: [w3.ORG](w3.ORG)



Generic Fragment



This caters for generic requirements across the meta meta model

NamedThing provides standard way of managing identity of anything which must have a unique id

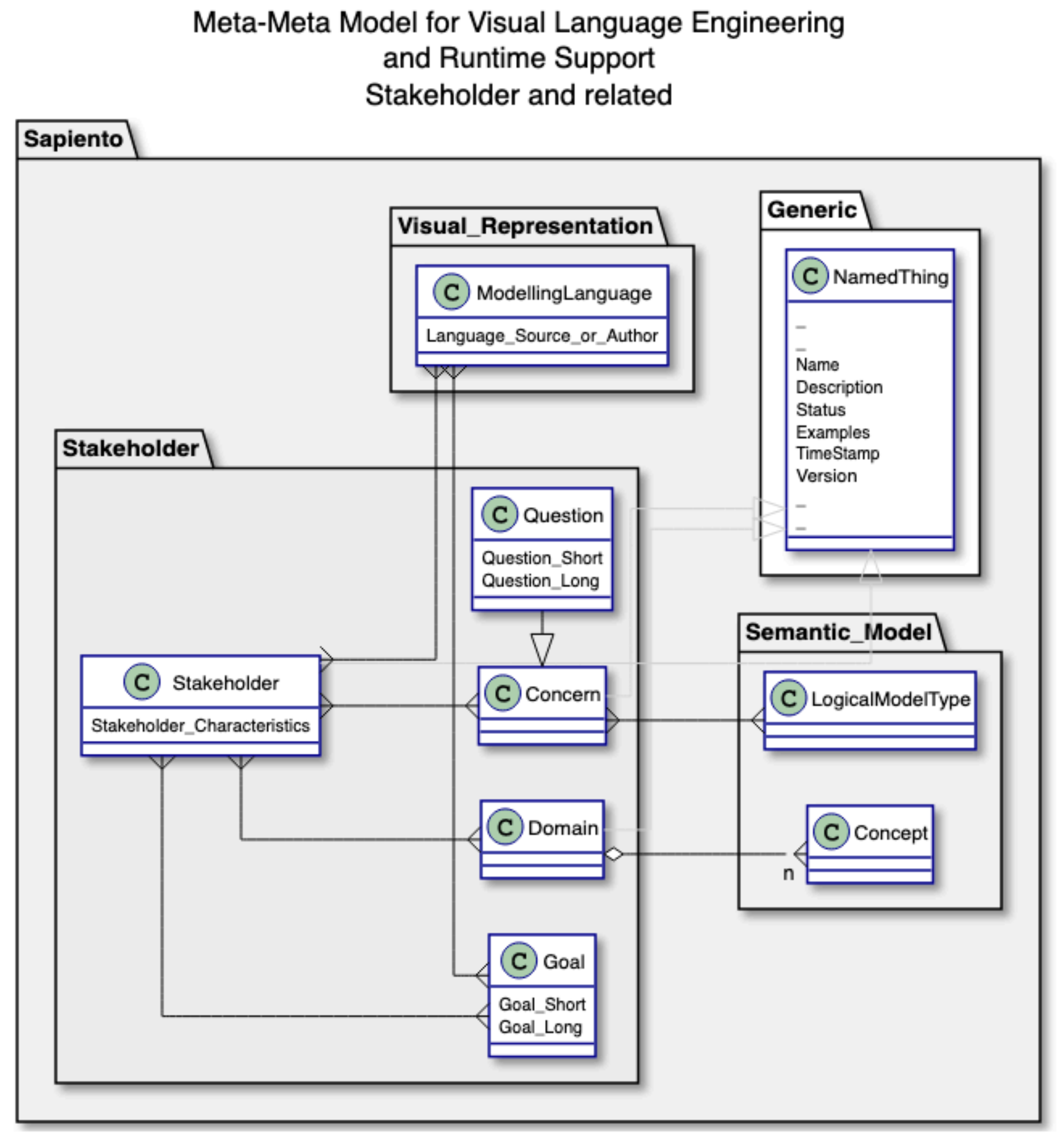
Alias caters for multiple names (e.g. Human language nouns; Technical vs Business Expert Terminology) for the same thing

Natural Language identifies the Human language used

Scope is used to prevent name clashes for data, information and models from different sources as well as to provide a packaging mechanism

Rule provides a generic mechanism for dynamic behaviour

Stakeholder Fragment



This relates Stakeholders to Concerns, Domains, Goals and Visual Languages that are appropriate

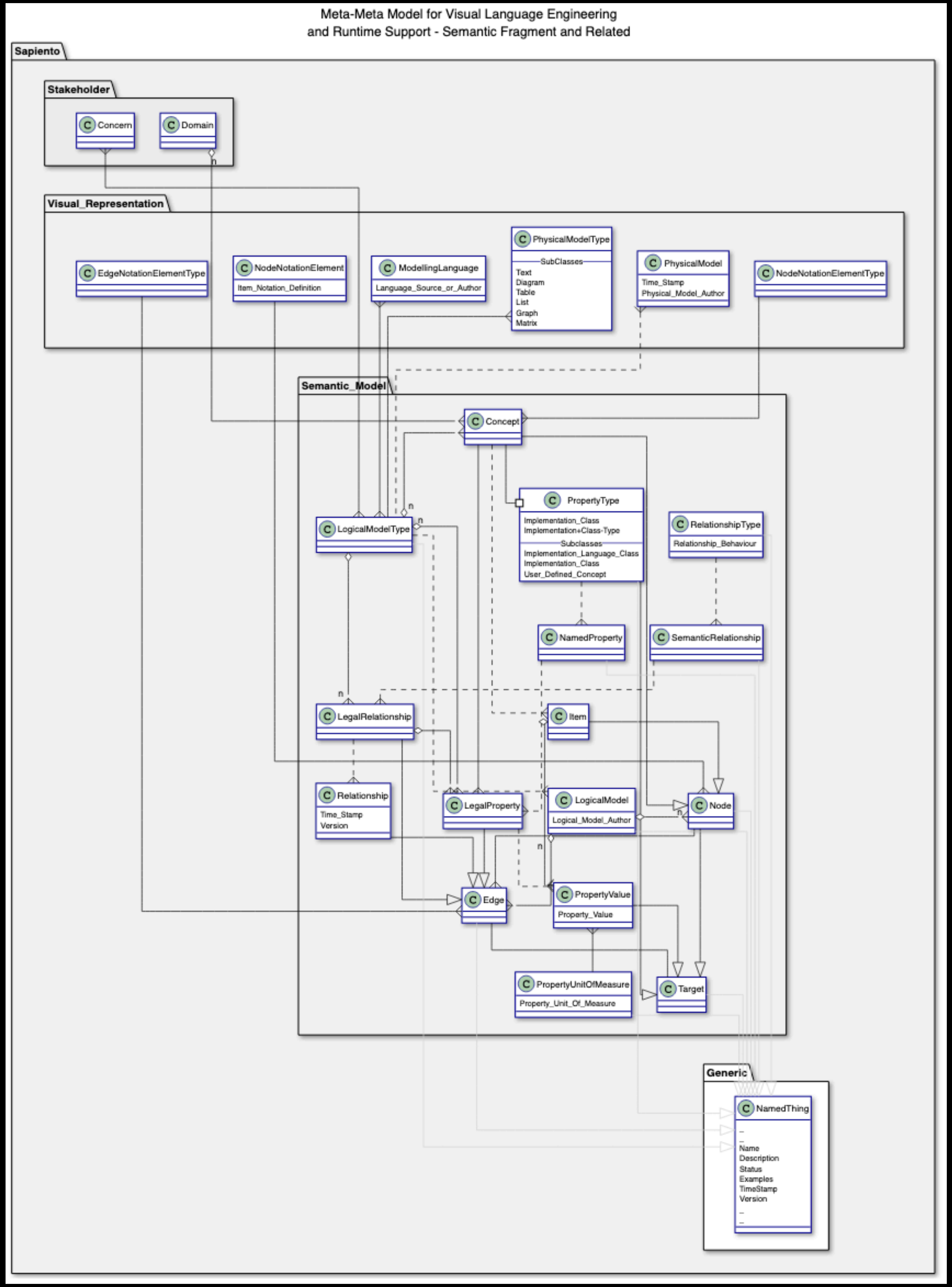
Stakeholder is a person or role which we hope to serve with relevant models and representations

Concern captures their focus areas that require relevant models and information. They relate to LogicalModelTypes in the Semantic fragment that provide support to address them

Domains relate to areas of expertise or industry and connect us to Concepts in the Semantic fragment

Goals connect Modelling Languages to Stakeholders

Semantic Fragment



This holds the domain meta model and instance models. It is concerned with *meaning*, not representation

Concept is any concept of relevance to a stakeholder (or the tool)

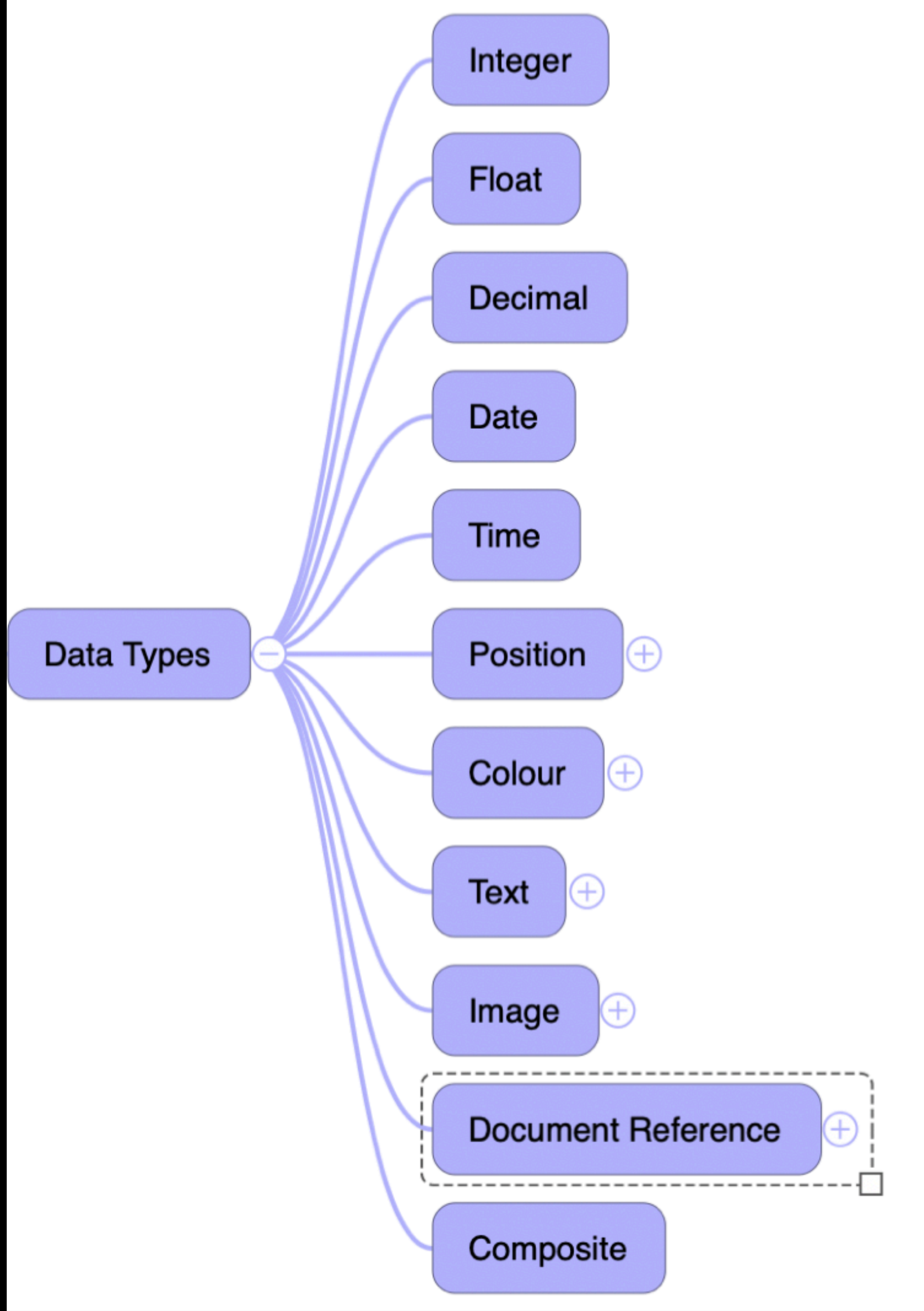
Concepts are defined by legal properties and relationships. A concept can also act as a complex property type

LogicalModelType groups concepts relevant to concerns or kind of model, independently of representation

LogicalModel groups items relevant to an instance model independent of representation

We will elaborate on Properties, Relationships, Nodes and Edges etc. in following slides

Rich Data Types



We have found these very powerful in the current EVA Netmodeler tooling

Essentially we implement a base environment (Smalltalk Class) with a predefined protocol and associated interface widgets for composing user interfaces

Defaults and Typing of Properties

Every concept defined has an associated default instance

This should be initialised with validly typed default property values (and relationships)

New instances are created with default values and relationships. These persist until altered by user inputs, imports or system functions

The valid value can specify a literal or a type

‘Unknown’ should be a valid value for all property types

Typed Relationships

Relationships are semantic and bi-directional

They are defined independently of concepts/types

They are used as legal relationships in meta models

They are typed, with the types implying behaviour

They can represent domain semantics *or* modelling semantics

e.g. Domain: Employee *works in* Department; Person *speaks* Language

Modelling: Employee *is role of* Person
Photograph *property of* Person
Profession *taxonomy for* Person

They can span layers: John *instance of* Employee
University *is a kind of* Tertiary Education Provider

Clabjects?

Well, sort of, at the Graph implementation level

We allow intermingling of definitions and instances with relationships between them and between each other

Concept — Concept [Defining Domain]

Instance — Instance [Defining Model]

Concept — Instance [Defining Multi-Level]

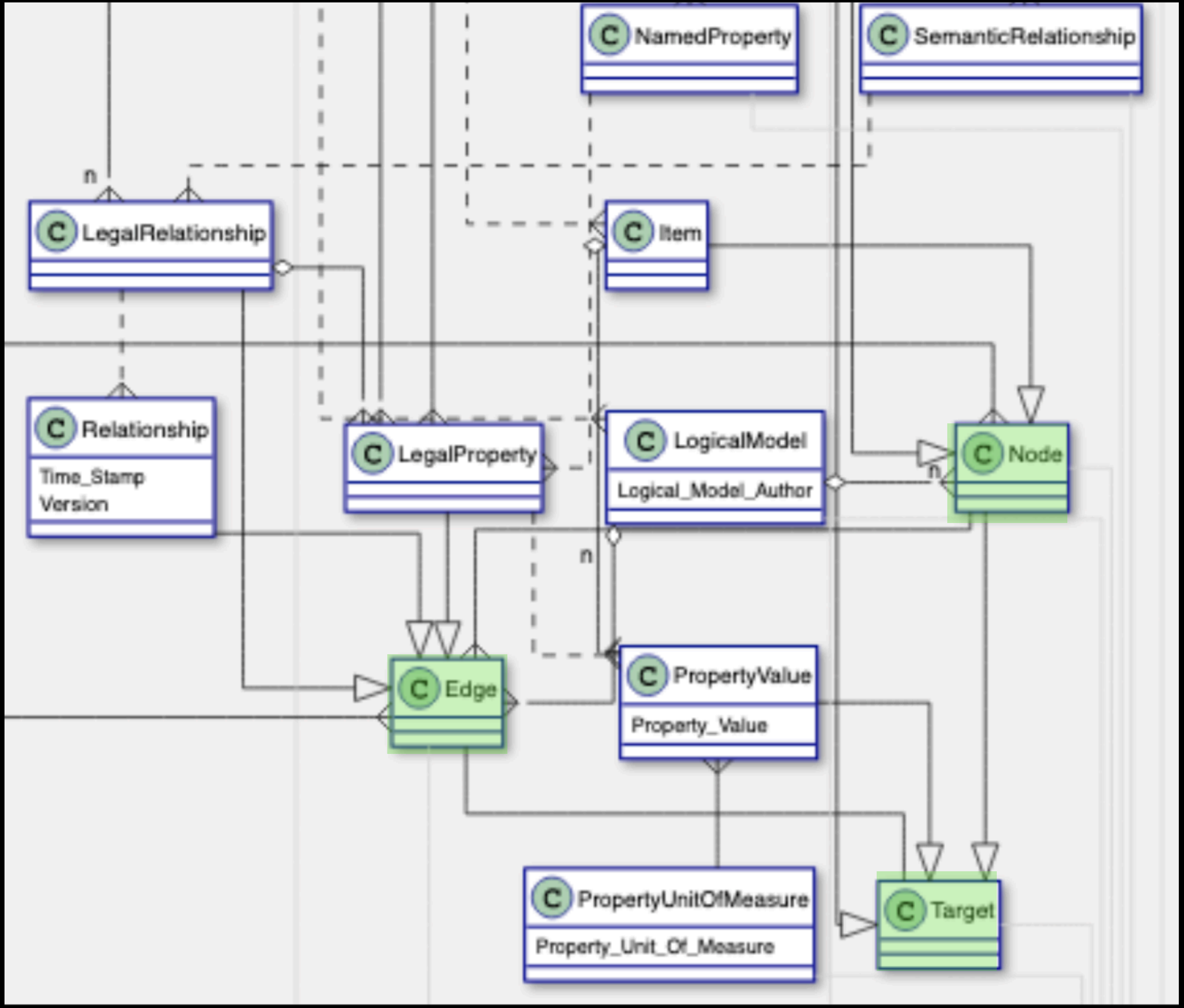
We also allow an object to act as both a definition and an instance. In the latter role, it will have properties and values. In the former it will be treated as definitional and appear in navigation tools etc. Objects could appear as concepts in one layer of modelling, but instances in another.

Car Range	
RangeName	String
DateOfRelease	Date
FuelType	Enumerated

Car Model	
ModelName	String
EngineCapacityCC	Integer
PowerKW	Decimal
TransmissionType	Enumerated

Car	
RegistrationNo	String
DateSold	Date
Owner	Party
Colour	Color

The Graph Mapping



Items in Models and Concepts in Meta Models are specialisations of Node

Nodes have Edges which can represent a Relationship, LegalRelationship or a LegalProperty

An Edge points to a Target which can be a Node or a PropertyValue

Behaviour

Any named object can have associated behaviour via the Rule concept

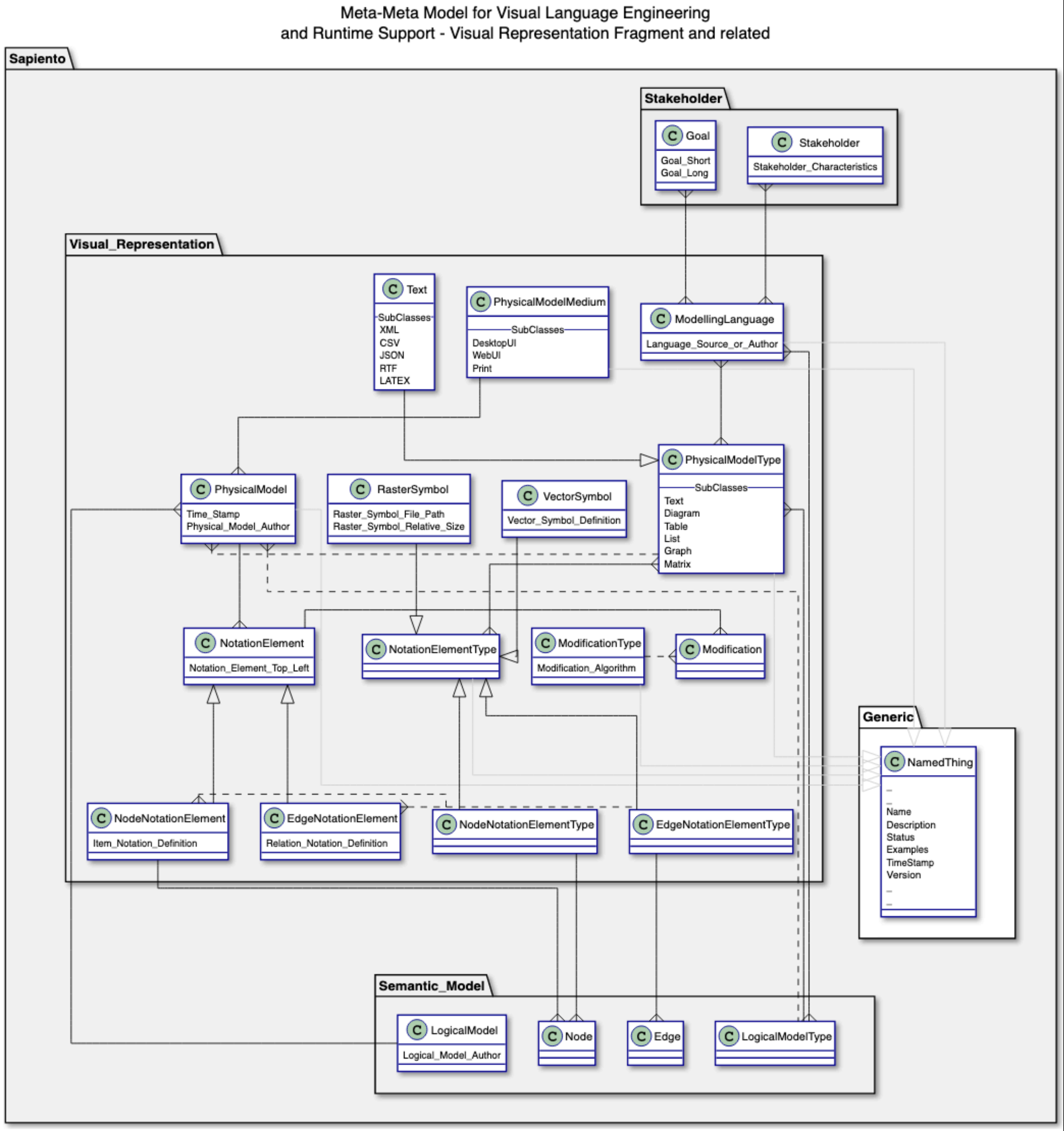
This is exploited to support:

- Constraints

- Validation

- Computation

- Derivation



Visual Representation Fragment

Maps semantic information to various representations

Modelling Language related to one or more PhysicalModelTypes

PhysicalModelTypes describe medium, format, notation, syntax

Notations can include structured text, vector symbols, raster symbols

It is intended that the model can cater for graphical models, generated visualisations, documents, import and export formats in text, potentially also UI

PhysicalModel is a container for ModelElements

Modification provides for polymathy

Status and Reflections

We have chosen a persistence environment based upon Property Graph technology, viz DGraph

This supports GraphQL (natively) as well as JSON and RDF, is very scalable and has good tools. It is open source, but supported with subscription. Cloud hosted environment available

We continue to use Smalltalk (VAST & Pharo) as development language

We have built proof on concept implementations and results are encouraging. Learnings have informed the model presented

This is first version with multi-level modelling and we hope to implement the "Multi Bicycle Challenge" as a demonstration of capability

We conclude that graph technology is very suitable for implementing meta models and supporting modelling tools and repositories

Smalltalk provides a rich and very late bound environment suitable to our needs. Easy translation is available to JSON/ STON

We can share tooling across meta modelling and instance modelling

We will upgrade our visual modelling tools to work with the new models, but this is awaiting resources

Communication or collaboration is welcome

References

Bertin J. (1983) Semiology of graphics. University of Wisconsin press

Besta M., Peter E., Gerstenberger R., Fischer M., Podstawski M., Barthels C., Alonso G., Hoefler T. (2019) Demystifying graph databases: Anal- ysis and taxonomy of data organization, system designs, and graph queries. In: arXiv preprint arXiv:1910.09017

BIAN.org (2022) BIAN Banking Service Land- scape 10 https://bian.org/servicelandscape-10-0- 0/views/view_51974.html Last Access: 2022-05- 02

Chen P. P.-S. (1976) The entity-relationship model—toward a unified view of data. In: ACM transactions on database systems (TODS) 1(1), pp. 9–36

Chiş A., Nierstrasz O., Gırba T. (2015) Towards moldable development tools. In: Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 25–26

Clark T. (2020) A Meta-Circular Basis for Model- Based Language Engineering.. In: The Journal of Object Technology 19(3), 3:1

Clark T., Willans J. (2014) Software language engineering with XMF and XModeler. In: Com- putational Linguistics: Concepts, Methodologies, Tools, and Applications. IGI Global, pp. 866–896

Däcker B. O., Williams M. C. (1997) Break- through in software design productivity through the use of declarative programming. In: International journal of production economics 52(1-2), pp. 227–231

Eclipse Foundation. https://www.eclipse.org. Last Access: Accessed: 2022Q3

Fernandes D., Bernardino J. (2018) Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and Ori- entDB.. In: Data, pp. 373–380

Frank U. (2002) Multi-perspective enterprise mod- eling (memo) conceptual framework and model- ing languages. In: Proceedings of the 35th Annual Hawaii International Conference on System Sci- ences. IEEE, pp. 1258–1267

Frank U. (2011) The MEMO meta modelling language (MML) and language architecture.. ICB- research report

Frank U. (2014) Multilevel modeling. In: Business & Information Systems Engineering 6(6), pp. 319– 337

Goldberg A., Robson D. (1983) Smalltalk-80: the language and its implementation. Addison-Wesley Longman Publishing Co., Inc.

Hartmann U., von Both P. (2009) A declarative approach to cross-domain model analysis. In: Man- aging It in Construction/Managing Construction for Tomorrow 26, pp. 45–51

Inspired.org (2022) Enterprise Value Architect https://www.inspired.org/eva-home Last Access: 2022-05-02

Kelly S., Tolvanen J.-P. (2021) Collaborative mod- elling and metamodelling with MetaEdit+. In: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Sys- tems Companion (MODELS-C). IEEE, pp. 27– 34

References

Lanza M. (2003) Object-Oriented Reverse Engi- neering Coarse-grained, Fine-grained, and Evolu- tionary Software Visualization. In:

Lara J. D., Guerra E., Cuadrado J. S. (2014) When and how to use multilevel modelling. In: ACM Transactions on Software Engineering and Methodology (TOSEM) 24(2), pp. 1–46

Martin J., Odell J. J. (1997) Object-oriented meth- ods (UML ed.,) a foundation. Prentice-Hall, Inc.

McGuinness D. L., Fikes R., Hendler J., Stein L. A. (2002) DAML+ OIL: an ontology language for the Semantic Web. In: IEEE Intelligent Systems 17(5), pp. 72–80

McLeod G. (2001) PAMELA: A Proto-pattern for Rapidly Delivered, Runtime Extensible Systems. In: Evaluation of Modeling Methods for Systems Analysis and Design (EMMSAD) 1

Moody D. (2009) The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. In: IEEE Trans- actions on software engineering 35(6), pp. 756– 779

Motik B., Patel-Schneider P. F., Parsia B., Bock C., Fokoue A., Haase P., Hoekstra R., Horrocks I., Ruttenberg A., Sattler U., et al. (2009) OWL 2 web ontology language: Structural specification and functional-style syntax. In: W3C recommendation 27(65), p. 159

Open_Source (2022) Drawing UML with Plan- tUML PlantUML Language Reference Guide https://plantuml.com/guide Last Access: 2022- 05-02

Puhlmann F. (2019) BPMN 2.0 Wimmelbild Edi- tion http://frapu.de/pdf/BPMN20-Wimmelbild.pdf Last Access: 2022-05-02

Rayner M., Hockey B. A., Chatzichrisafis N., Far- rell K. (2005) OMG Unified Modeling Language Specification. In: Version 1.3, © 1999 Object Management Group, Inc

von Rosing M., White S., Cummins F., de Man H. (2015) Business Process Model and Notation- BPMN.

Solomon C., Harvey B., Kahn K., Lieberman H., Miller M. L., Minsky M., Papert A., Silverman B. (2020) History of logo. In: Proceedings of the ACM on Programming Languages 4(HOPL), pp. 1–66

Various (2019) Archimate 3.1 Specification. The Open Group Series. Van Haren Publishing https: //books.google.co.za/books?id=kibNywEACAAJ

W3C (2022) Resource Description Framework (RDF) https://www.w3.org/RDF/ Last Access: 2022-05-02

Ward P. T. (1986) The transformation schema: An extension of the data flow diagram to represent control and timing. In: IEEE Transactions on Software Engineering (2), pp. 198–210

Ware C. (2010) Visual thinking for design. Else- vier

Researcher: Graham McLeod

- graham@inspired.org
- www.inspired.org (Blog there too)
- LinkedIn: Graham McLeod
- Skype: grahammcleod

Academia: Undergraduate BSc Comp Sc at Unisa, B Comm Hons (IS) at Univ. Cape Town, PhD (in progress) Univ Duisburg-Essen.

Faculty of Information Systems at UCT for 12 years (1991 - 2003)

Active in industry from 1975 - present. Roles of developer, designer, analyst, project manager, product manager, instructor/lecturer, consultant, architect, entrepreneur, general manager, director, chairman, business owner. Currently Chief Architect / Owner of inspired.org

Major interests: Modelling, Meta Modelling, Methods Engineering, Enterprise and Software Architecture, Business Modelling, Strategy, Tools, Dynamic Languages (Smalltalk)

http://www.travelstart.co.za/blog/wp-content/uploads/2013/11/Greg-Lumley.jpg

Based in Cape Town, South Africa