

Architecture and Modelling

Stuart Kent

University of Kent @ Canterbury, UK

www.cs.ukc.ac.uk

sjhk@ukc.ac.uk

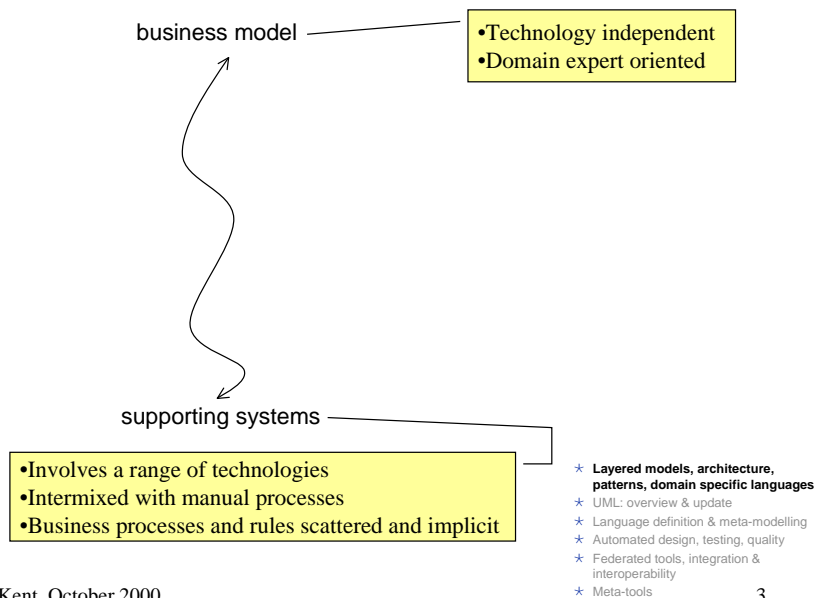
Outline

- ★ Layered models, architecture, patterns, domain specific languages
- ★ UML: overview & update
- ★ Language definition & meta-modelling
- ★ Automated design, testing, quality
- ★ Federated tools, integration & interoperability
- ★ Meta-tools

© Stuart Kent, October 2000

2

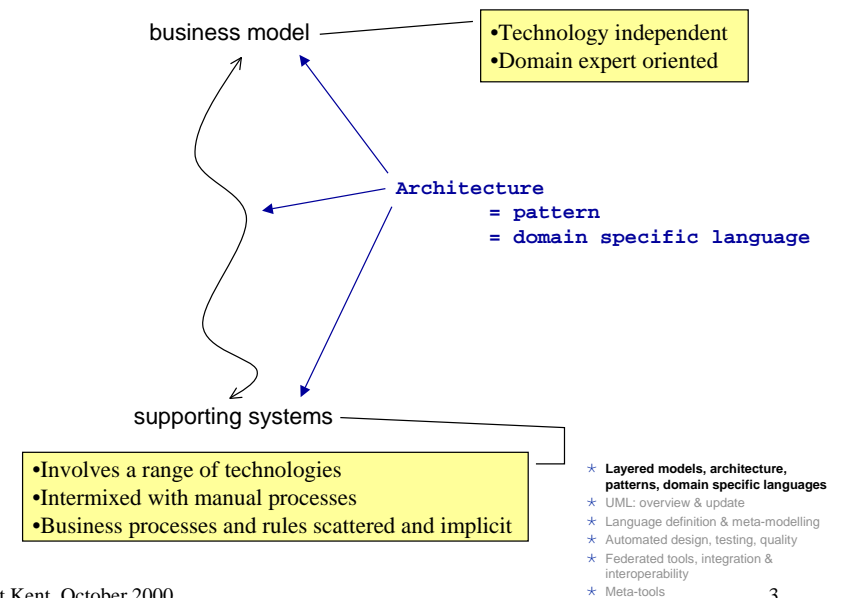
Layered Modelling



© Stuart Kent, October 2000

3

Layered Modelling

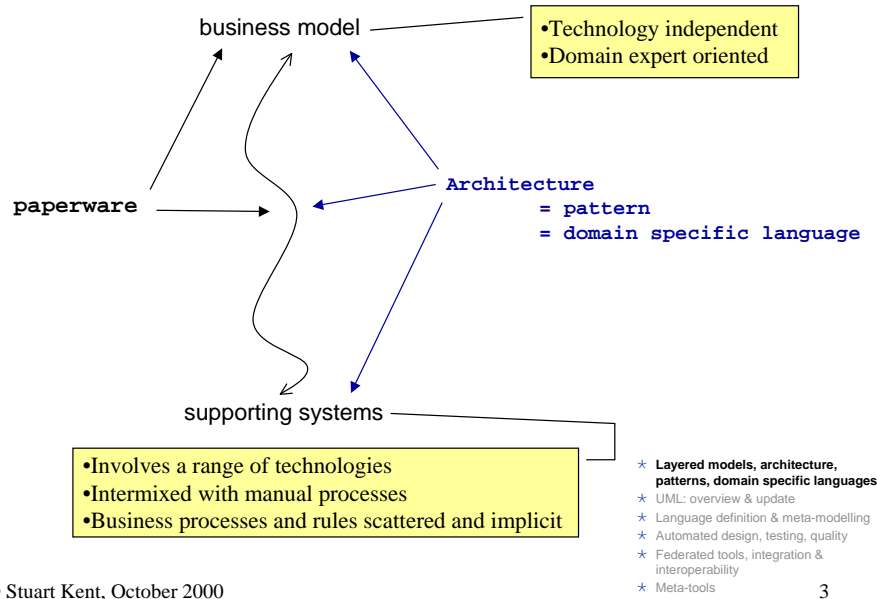


© Stuart Kent, October 2000

3

current practice

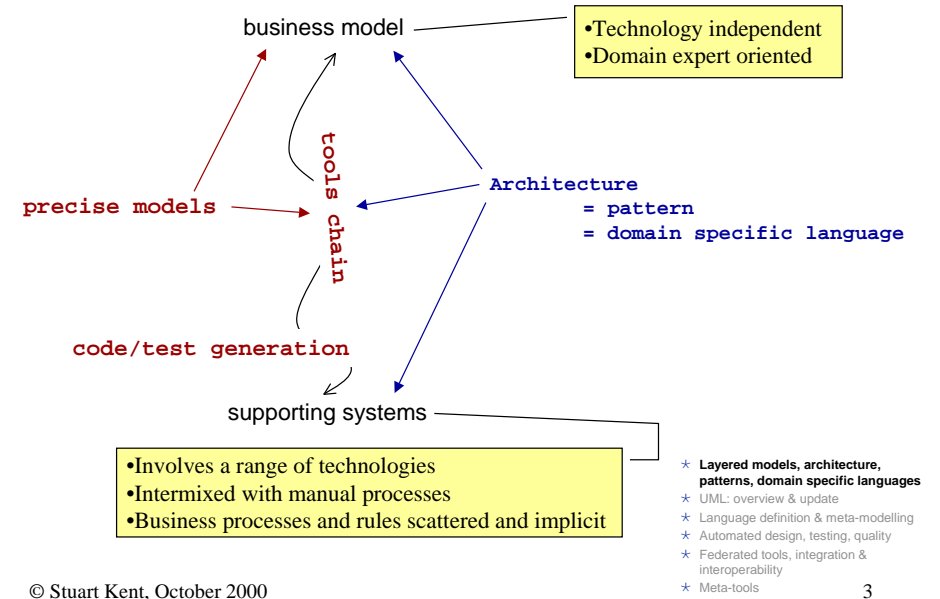
Layered Modelling



© Stuart Kent, October 2000

future practice

Layered Modelling



© Stuart Kent, October 2000

A little history

- ★ UML began with Rational and the 3 amigos
- ★ OMG put out a request for proposals for a standard object modeling language
- ★ A number of consortiums put in initial submissions
- ★ Initial submissions were combined into the eventual final submission, under one consortium
 - ⊙ exception: the OPEN group
- ★ UML was accepted by and **passed over to** the OMG
- ★ Revision of UML is now in the hands of the OMG
- ★ Possibility of ISO standardisation

- ★ Layered models, architecture, patterns, domain specific languages
- ★ UML: overview & update
- ★ Language definition & meta-modelling
- ★ Automated design, testing, quality
- ★ Federated tools, integration & interoperability
- ★ Meta-tools

4

© Stuart Kent, October 2000

UML & other OMG standards

- ★ MOF
 - ⊙ Language for describing other languages
 - ⊙ Meta-models in MOF used to generate IDL and XMI
- ★ CORBA (IDL)
 - ⊙ Generate IDL from meta-model in MOF
 - ⊙ IDL represents set of interfaces to a repository supporting that meta-model
 - ⊙ UML could be one such meta-model
 - ⊙ Generate IDL from (certain) UML models
- ★ XMI (XML interchange)
 - ⊙ Dictates how UML and MOF models are interchanged in XML
 - ⊙ Based on MOF technology

© Stuart Kent, October 2000

5

Notations: Business Modelling

- ★ Activity diagrams (Business Processes)
- ★ Use cases
- ★ Class diagrams & constraints (OCL)
- ★ State diagrams (BP's as objects)
- ★ Pre/post conditions
- ★ Policies (as patterns?)
- ★ Packages & composition

- ★ Also see e.g. Ericksson & Penker book

Notations: Software Design

- ★ Class diagrams (with all the OOP add-ons)
- ★ Interaction diagrams (sequence + collaboration)
- ★ State diagrams (sometimes)
- ★ Package diagrams (visual version of Java imports)

- ★ Ok for designs mapped to traditional (!) OOPs
- ★ But what about e-commerce systems? Legacy integration? Component-based development? Etc.

Notations: Deployment

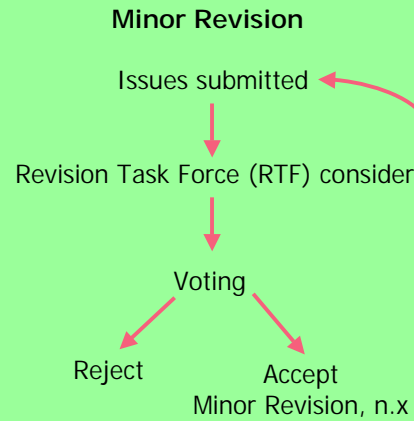
- ★ Deployment diagrams
 - ⊙ Hardware nodes and their connections
- ★ Component diagrams
 - ⊙ (Immobile) Software components, where they live, their interfaces and who they talk to

- ★ These diagrams are no better than the informal sketches that systems architects have always drawn
- ★ And they have about as much meaning – and *agreement* on meaning

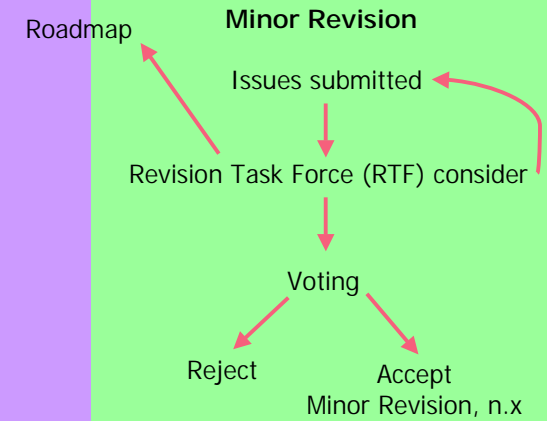
Note

- ★ No notations for mapping/translating between different models
- ★ Most CASE tools do not cover full spectrum (they do not implement UML 1.3)
- ★ Notations have agreed syntax, but not agreed meaning
- ★ Organisations/teams subset and specialise the notations they use
 - ⊙ To agree a common meaning ?
- ★ Essentially not one language but a family of languages

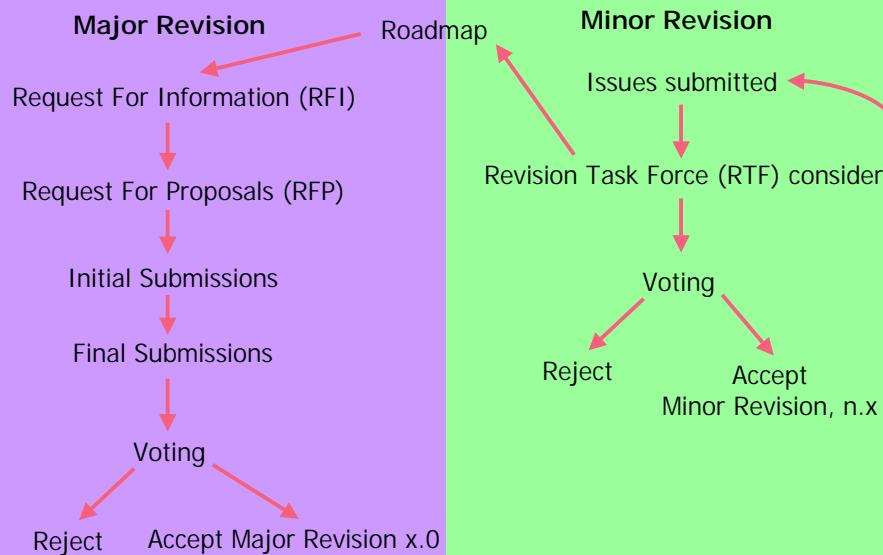
OMG processes



OMG processes



OMG processes



UML related RFP's

www.omg.org/techprocess/meetings/schedule/

- ★ Action Semantics
- ★ Various Profiles
- ★ UML 2.0
 - ⊙ 4 RFP's
 - ⊙ RFP's just issued; awaiting initial submissions

UML Profile RFP's

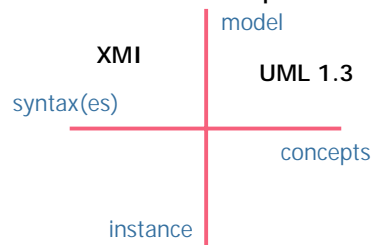
- ★ EDOC – Enterprise Distributed Object Computing
 - ⊙ Working on merging 2 initial submissions
 - ⊙ Component architecture, viewpoints & whole lifecycle (wow!)
 - ⊙ Closely tied to EAI (intersecting teams)
- ★ Textual language for EDOC profile
 - ⊙ Awaiting initial submissions
- ★ CORBA profile
 - ⊙ Use UML to write IDL (in a nutshell)
 - ⊙ Revised submission received
- ★ Scheduling profile
 - ⊙ Real-time profile for UML (in a nutshell)
 - ⊙ Awaiting initial submissions
- ★ Event Based Architecture in Enterprise Application Integration (EAI) profile
 - ⊙ Working on a single joint revised submission
 - ⊙ Closely tied to EDOC

UML 2.0

- ★ 4 RFP's
 - ⊙ **Infrastructure**
 - ⊙ Superstructure
 - ⊙ Object Constraint Language (OCL)
 - ⊙ Diagram interchange (planned)
- ★ UML 2.0 Working Group
 - www.celigent.com/omg/adptf/wgs/uml2wg.htm

Infrastructure

- ★ Alignment with MOF
- ★ Family of languages
 - ⊙ Notation mix
 - ⊙ Profiles
 - ⊙ Language Evolution
 - ⊙ Patterns
- ★ Comprehensive definition + separation of concerns



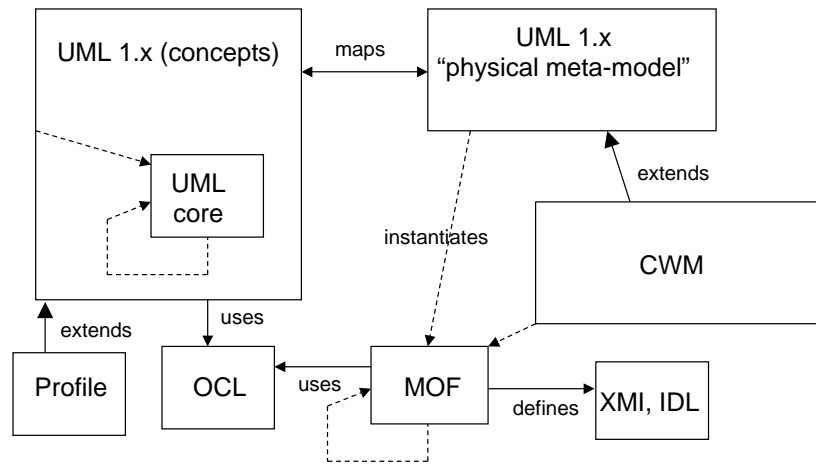
- ★ The precise UML group & IBM have done a feasibility study on this. See next slides & www.puml.org.

Approaches to meta-modelling

- ★ MOF – www.omg.org
- ★ MetaEdit+ – www.metacase.com
- ★ Dome – www.htc.honeywell.com/dome/
- ★ XML – www.w3c.org
- ★ MMF – www.puml.org

- ★ Layered models, architecture, patterns, domain specific languages
- ★ UML: overview & update
- ★ **Language definition & meta-modelling**
- ★ Automated design, testing, quality
- ★ Federated tools, integration & interoperability
- ★ Meta-tools

Current architecture of UML / MOF

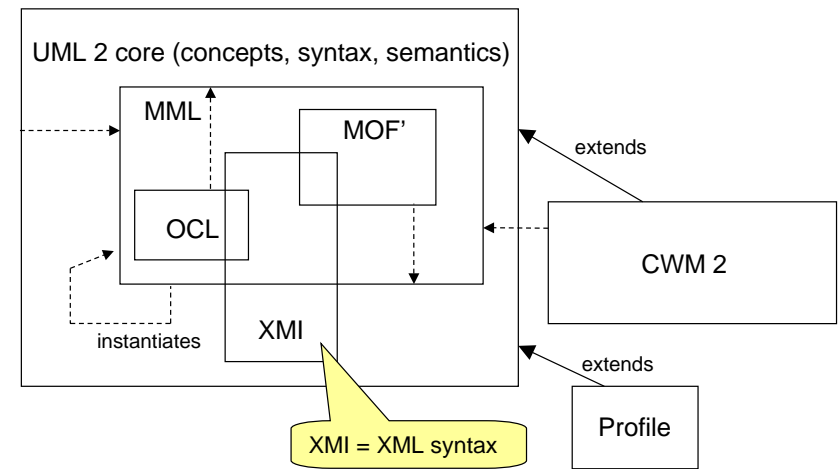


adapted from a slide by Steve Cook

© Stuart Kent, October 2000

16

A possible new architecture for UML / MOF



adapted from a slide by Steve Cook

© Stuart Kent, October 2000

17

Automated design, testing, quality

★ Quality

- ⊙ tests, inspections, proofs
- ⊙ validation... does it support the business well
- ⊙ good enough software... how important is quality
- ⊙ code generation (but how do you validate the model)

★ eXtreme programming (XP)

- ⊙ forget paperwork... focus only on stuff that executes
- ⊙ (automated) testing
- ⊙ refactoring – architecture free, or at least emergent & fluid architecture
- ⊙ responsive to change

? How to test (automatically) heterogeneous systems

? How to test business models and mappings, if these become more than paperwork

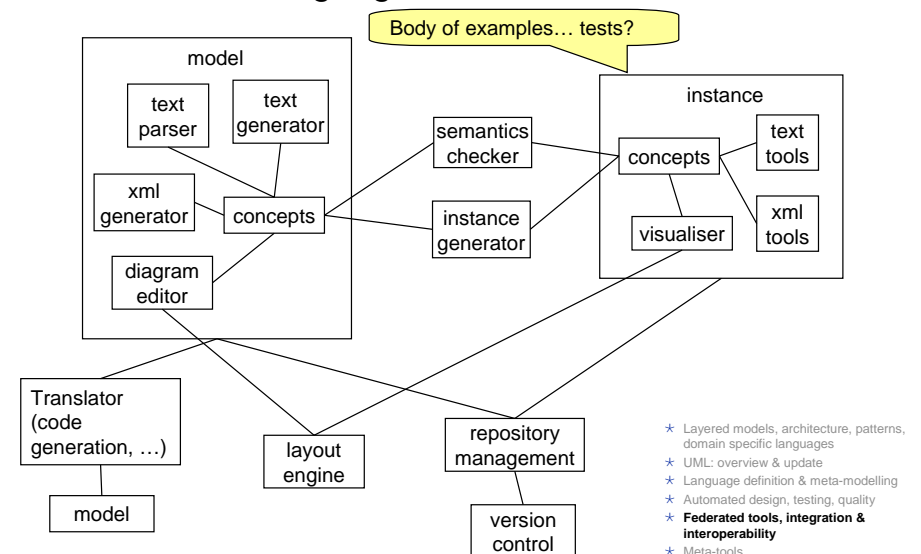
- ★ Layered models, architecture, patterns, domain specific languages
- ★ UML: overview & update
- ★ Language definition & meta-modelling
- ★ **Automated design, testing, quality**
- ★ Federated tools, integration & interoperability
- ★ Meta-tools

© Stuart Kent, October 2000

18

Federated tools: integration, interoperability

Architecture of language... architecture of tools:



© Stuart Kent, October 2000

19

Meta-tools

- ★ Don't live in a world of fixed languages
 - ⊙ different domains use different modelling patterns (architectures)
 - ⊙ i.e. domain specific languages
- ★ But, if modelling is to be used in anger we need tools
 - ⊙ Tools need to be language specific
 - ⊙ Time to market of tools is preventing (imho) the development and application of domain specific languages
- ★ Meta-tools generate tools from language definitions
 - ⊙ through interpretation or compilation
 - ⊙ still a long way to go
 - ⊙ E.g. MetaEdit+
- ★ Meta-tools would admit *opportunistic invention* of domain specific languages

- ★ Layered models, architecture, patterns, domain specific languages
- ★ UML: overview & update
- ★ Language definition & meta-modelling
- ★ Automated design, testing, quality
- ★ Federated tools, integration & interoperability
- ★ **Meta-tools**

20

© Stuart Kent, October 2000

Summary

- ★ Architecture is in all layers (viewpoints) and mappings between layers
- ★ Status quo is paperware in all but the lowest levels
- ★ Paperware is unsustainable in a fast changing world
 - ⊙ too time consuming to maintain
- ★ Choice: XP or turn paperware into tooled models
 - ⊙ XP ok for homogenous systems, but heterogeneous?
- ★ UML 1.3. Is only fit for paperware... UML 2.0. may make a difference
- ★ Different domains & levels observe different patterns of modelling... domain specific languages (DSLs)
- ★ Building DSLs requires
 - ⊙ strong meta-modelling basis and meta-tools
 - ⊙ federated tools architecture
 - ⊙ sophisticated checking & interpretation abilities
 - ⊙ we're not there yet !!

© Stuart Kent, October 2000

21

What now?

- ★ Look at books such as [1], which shows how to specify and build systems based on flexible component-oriented architectures
 - ⊙ Switching to a component-based architecture now might be flexible enough to accommodate change for the next few years
 - ⊙ Still paperware at the modelling level
- ★ Decide if you really need paperware
 - ⊙ Adopt XP principles
 - ⊙ Homogenous environment?
 - ⊙ (Implicit) Architecture continually evolving?
- ★ Take a lightweight approach to modelling (now) and wait for UML 2.0 & tools to settle down.

[1] Daniels & Cheesman, "A simple process for specifying component-based software", Addison Wesley.

© Stuart Kent, October 2000

22