CBSE and MDE: Fitting the Pieces Together

Kung-Kiu Lau

School of Computer Science
The University of Manchester
United Kingdom

kung-kiu.lau@manchester.ac.uk

Keynote, ModComp 2016, 4 October 2016, Saint-Malo, France

Overview

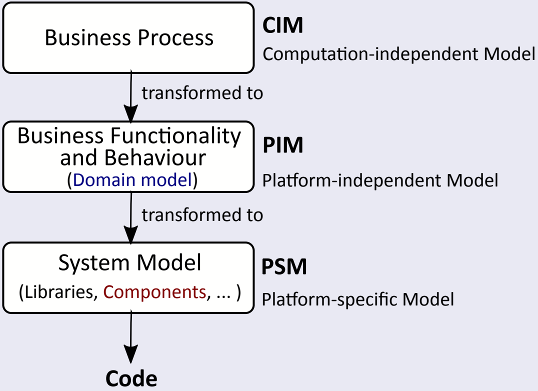
Structure of Talk

- MDA/MDE/CBSE: Terminology, essential elements and links
- Our work in CBSE
- Our use of MDE
- Observations/questions on MDE

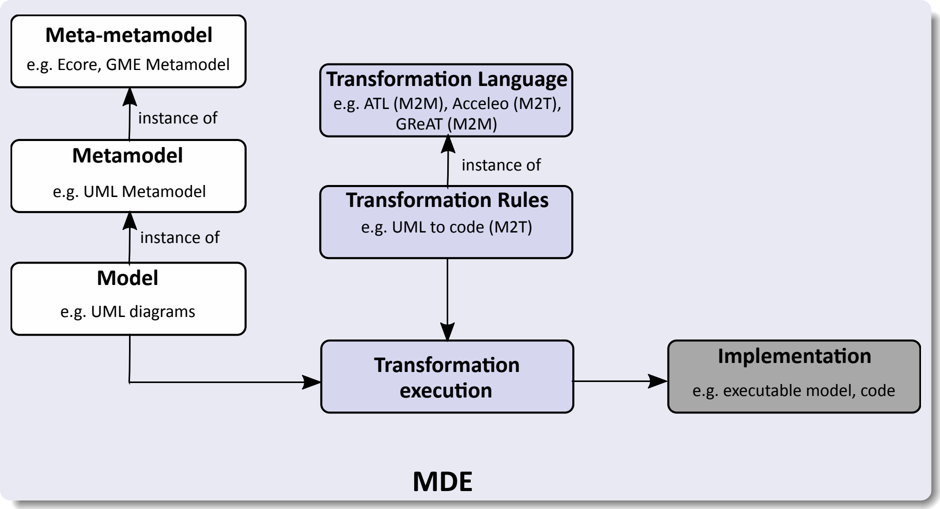
Acknowledgement: Joint work with Simone di Cola and Cuong Tran

MDA vs CBSE

MDA	CBSE
process-centric top-down correct-by-transformation	product-centric bottom-up correct-by-composition

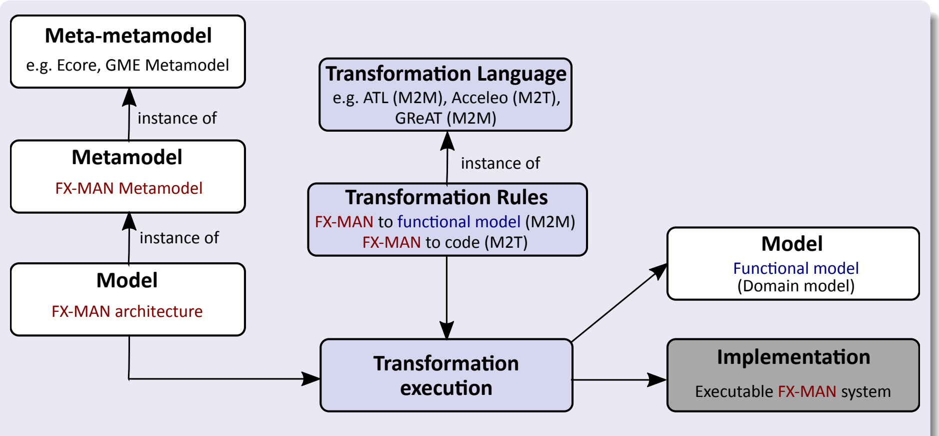

Business Process
CIM
Computation-independent Model
transformed to
Business Functionality and Behaviour
(Domain model)
PIM
Platform-independent Model
transformed to
System Model
(Libraries, Components, ...)
PSM
Platform-specific Model
Code

MDE



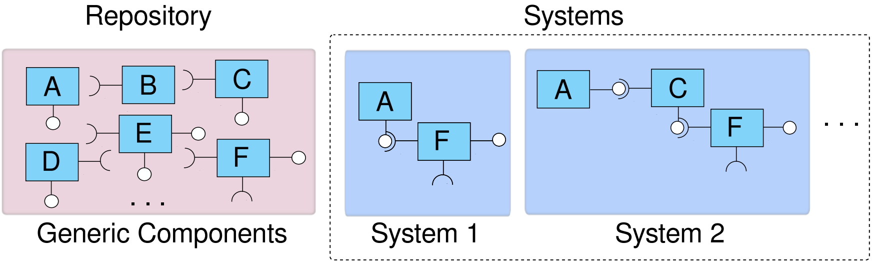
Meta-metamodel
e.g. Ecore, GME Metamodel
instance of
Metamodel
e.g. UML Metamodel
instance of
Model
e.g. UML diagrams
Transformation Language
e.g. ATL (M2M), Acceleo (M2T), GReAT (M2M)
instance of
Transformation Rules
e.g. UML to code (M2T)
Transformation execution
Implementation
e.g. executable model, code
MDE

MDE + CBSE: Our Approach



MDE + CBSE with Component Model (FX-MAN)

CBSE: General Picture



- Repository = Pre-existing components (in a domain)
- Repository components reused in many systems (in the domain)
- System = Composition of components
- Composition ⟶ Reuse
- ‘Bottom-up’

Components and Composition

	Unit of Composition	Composition Mechanism				
		Containment	Extension	Connection	Coordination	
Programming View	Function	Function nesting		Higher-order function Function call		
	Procedure	Procedure nesting		Procedure call		
	Class	Class nesting Object composition Object aggregation	Multiple inheritance	Object delegation		
	Mixin		Mixin inheritance			
	Mixin/Class		Mixin-class inheritance			
	Trait		Trait composition	Trait composition		
	Trait/Class		Trait-class composition	Trait-class composition		
	Subject		Subject composition			
	Feature		Feature composition			
	Aspect/Class		Weaving			
	Module	Module nesting		Module connection		Construction View
CBD View	Architectural unit			Port connection		
	Fragment box		Invasive composition	Invasive composition		
	Process			Channels	Data coordination	
	Web service				Orchestration (Control coordination)	
	Encapsulated component				Exogenous composition (Control coordination)	

K.-K. Lau and T. Rana. A Taxonomy of Software Composition Mechanisms. In *Proc. 36th Euromicro Conference on Software Engineering and Advanced Applications*, pages 102-110, IEEE, 2010.

Software Component Models

A software component model defines:

- components
- composition mechanisms

CBSE with a component model is model-driven by definition:

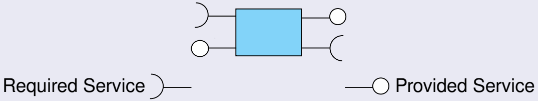
- model for components
- model for composition
- + model-driven implementation of components
- + model-driven implementation of composition

K.-K. Lau and Z. Wang. Software Component Models. *IEEE Transactions on Software Engineering* 33(10):709-724, October 2007.

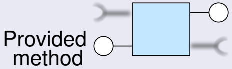
K.-K. Lau, Z. Wang, S. Di Cola, C. Tran and V. Christou. Software Component Models: Past, Present and Future. Tutorial at COMPARCH 2014 Conference, 30 June 2014, Lille, France.

Types of Components

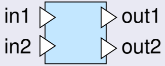
A Generic Component



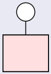
An Object



An Architectural Unit



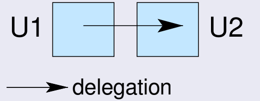
An Encapsulated Component



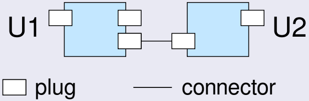
Components	Provided services	Required services	Composition mechanism
Objects	Methods	——	Method call
Architectural units	Out-ports	In-ports	Port connection
Encapsulated components	Methods	*None*	Exogenous composition

Types of Composition Mechanisms

Connection: Method Call & Port Connection

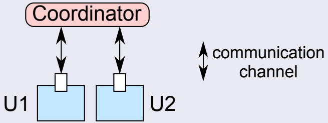


(a) Direct message passing



(b) Indirect message passing

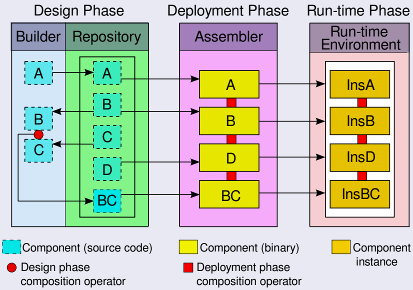
Coordination: Exogenous Composition



Idealised Component Life cycle

Composition in Component Design Phase and Component Deployment Phase

Idealised Component Life Cycle



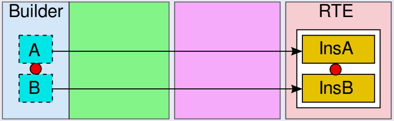
K.-K. Lau and Z. Wang. Software Component Models. *IEEE Transactions on Software Engineering* 33(10):709-724, 2007.

Traditional CBSE Desiderata

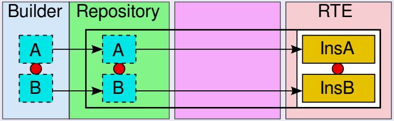
Desideratum	Design Phase	Deployment Phase
Components should pre-exist	Deposit components in repository	Retrieve components from repository
Components should be produced independently	Use builder	——
Components should be deployed independently	——	Use assembler
It should be possible to copy and instantiate components	Copies possible	Copies and instances possible
It should be possible to build composites	Composition possible	Composition possible
It should be possible to store composites	Use repository	——

M. Broy, A. Deimel, J. Henn, K. Koskimies, F. Plasil, G. Pomberger, W. Pree, M. Stal and C. Szyperski. What characterizes a software component? *Software — Concepts and Tools* 19:49-56, 1998.

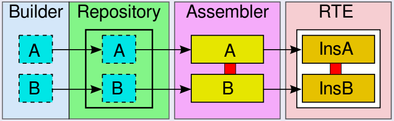
Taxonomy of Component Models



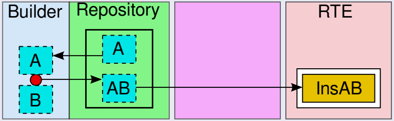
Category 1: Design without Repository
(Acme–like ADLs, UML2.0, PECOS)



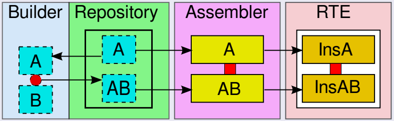
Category 2: Design with Deposit–only Repository
(EJB, OSGi, Fractal, COM, .NET, CCM)



Category 3: Deployment with Repository
(JavaBeans, Web Services)



Category 4: Design with Repository
(Koala, SOFA, KobrA, SCA, Palladio, ProCom)



Category 5: Design and Deploy with Repository
(X-MAN)

K.-K. Lau, Z. Wang, S. Di Cola, C. Tran and V. Christou. Software Component Models: Past, Present and Future. Tutorial at COMPARCH 2014 Conference, 30 June 2014, Lille, France.

CBSE Desiderata: Present & Future

Present

Taxonomy of component models shows:

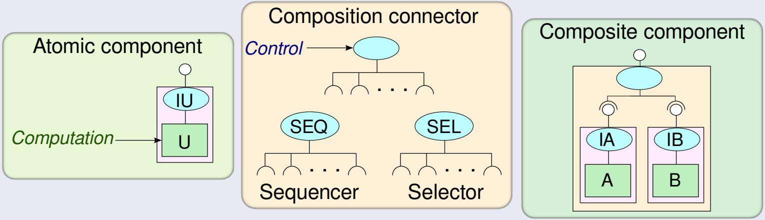
- Current component models do not fully meet the traditional CBSE desiderata

Future

- CBSE faces new challenges:
  - increased scale
  - increased complexity
  - assurance of safety of large complex systems
- Future component models have to meet these new desiderata

X-MAN Component Model

Components & Composition



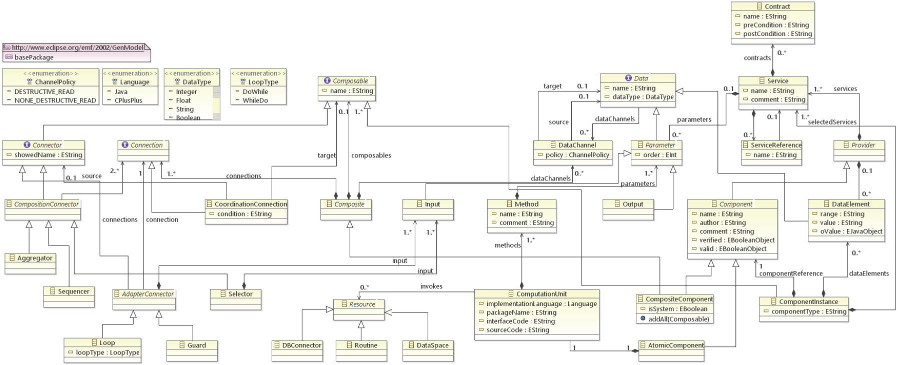
- Hierarchical (algebraic) composition ⟹ scale and complexity
- Compositional verification ⟹ large-scale verification (of safety)

K.-K. Lau, P. Velasco Elizondo and Z. Wang. Exogenous Connectors for Software Components. In *Proc. 8th International SIGSOFT Symposium on Component-based Software Engineering*. LNCS 3489:90-106, Springer-Verlag, 2005.

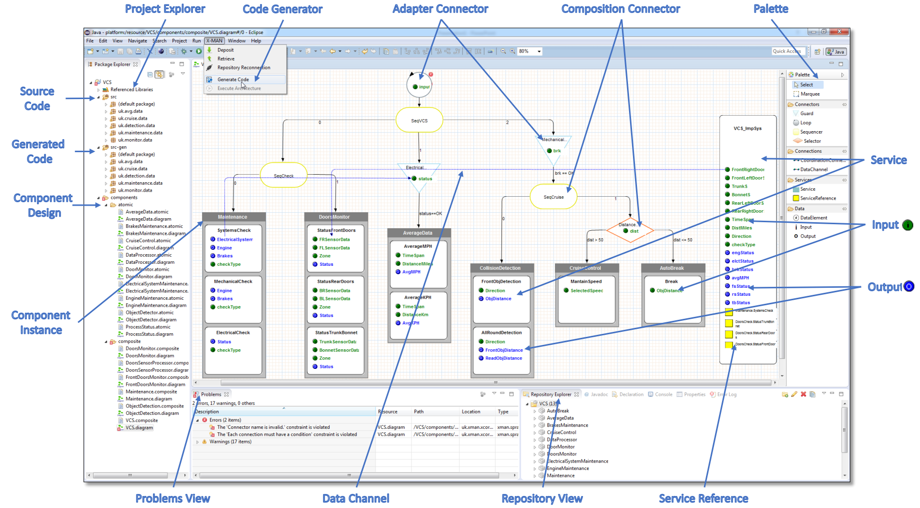
K.-K. Lau, M. Ornaghi and Z. Wang. A Software Component Model and its Preliminary Formalisation. In *Proc. 4th International Symposium on Formal Methods for Components and Objects*, LNCS 4111:1-21, Springer-Verlag, 2006.

N. He, D. Kroening, T. Wahl, K.-K. Lau, F. Taweel, P. Rümmer and S. Sharma. Component-based Design and Verification in X-MAN. In *Proc. Embedded Real Time Software and Systems*, 2012.

X-MAN Meta-model



MDE Tool for X-MAN



Project Explorer

Code Generator

Adapter Connector

Composition Connector

Palette

Source Code

Generated Code

Component Design

Component Instance

Service

Input

Output

Problems View

Data Channel

Repository View

Service Reference

Problems

2 errors, 17 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
The 'Connector name is invalid.' constraint is violated	VCS.diagram	/VCS/components/...	uk.xman.xcor...	xman.spra
The 'Each connection must have a condition' constraint is violated	VCS.diagram	/VCS/components/...	uk.xman.xcor...	xman.spra
Warnings (17 items)				

K.-K. Lau and C.M. Tran. X-MAN: An MDE Tool for Component-based System Development. In *Proc. 38th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 158-165, IEEE, 2012.

S. Di Cola, K.-K. Lau and C. Tran. A Graphical Tool for Model-Driven Development Using Components and Services. In *Proc. 41st EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 181-182, IEEE, 2015.

X-MAN Example

Vehicle Control System (VCS)

A VCS is a real-time, on-board system for supervising a vehicle.
It manages several routine services and tasks, including:

- statistical data calculation

  e.g. of fuel consumption and of average speed

- observation or monitoring of the vehicle's internal state

  e.g. maintenance status

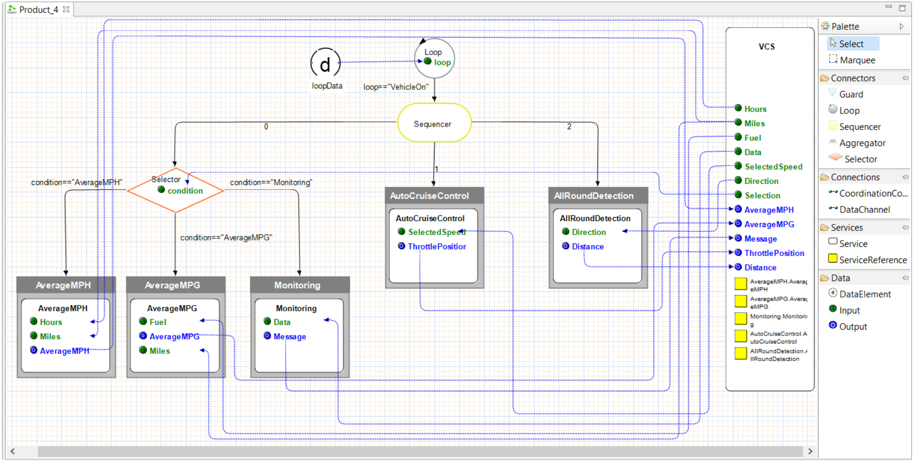
- cruise control

  i.e. automatically controlling the vehicle's speed in such a way that a steady (cruise) speed can be set (by the driver) and then maintained by taking over control of the throttle whenever necessary

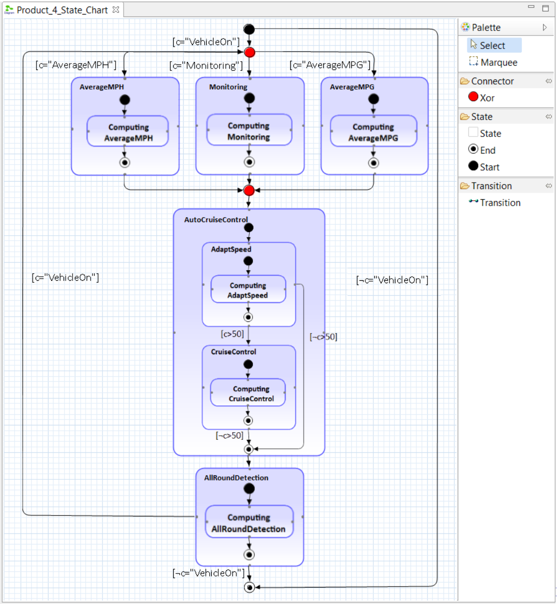
- collision detection

  to ensure safety and enable automatic driving (while cruising)

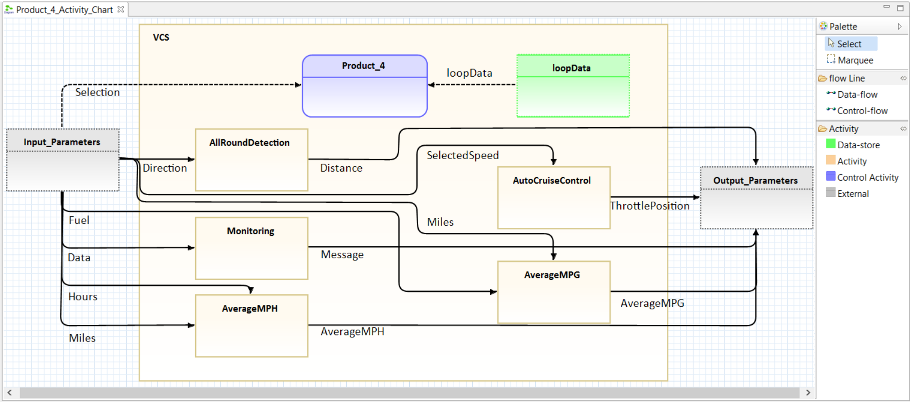
VCS System in X-MAN



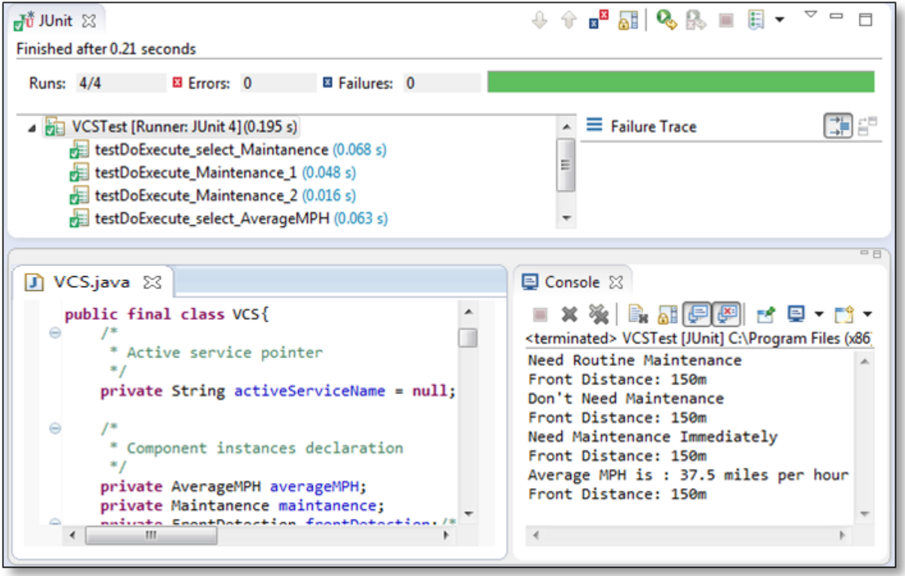
VCS Functional Model: State Chart



VCS Functional Model: Activity Chart



VCS Testing



Finished after 0.21 seconds

Runs: 4/4 Errors: 0 Failures: 0

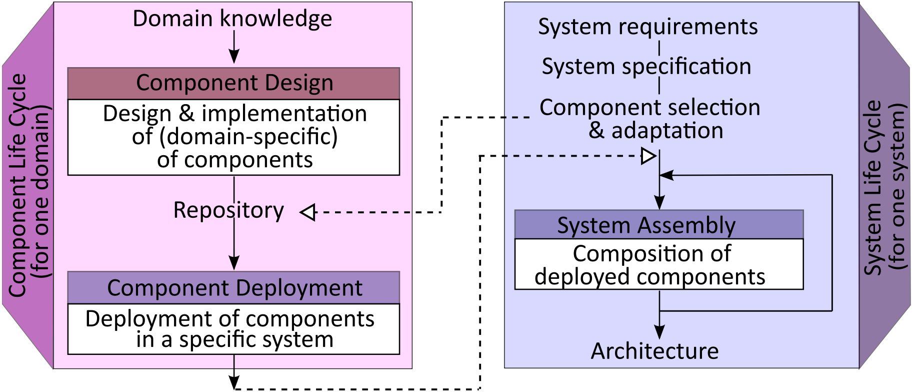
- VCSTest [Runner: JUnit 4] (0.195 s)
  - testDoExecute_select_Maintanence (0.068 s)
  - testDoExecute_Maintenance_1 (0.048 s)
  - testDoExecute_Maintenance_2 (0.016 s)
  - testDoExecute_select_AverageMPH (0.063 s)

```java
public final class VCS{
    /*
     * Active service pointer
     */
    private String activeServiceName = null;

    /*
     * Component instances declaration
     */
    private AverageMPH averageMPH;
    private Maintanence maintanence;
```

```
<terminated> VCSTest [JUnit] C:\Program Files (x86
Need Routine Maintenance
Front Distance: 150m
Don't Need Maintenance
Front Distance: 150m
Need Maintenance Immediately
Front Distance: 150m
Average MPH is : 37.5 miles per hour
Front Distance: 150m
```

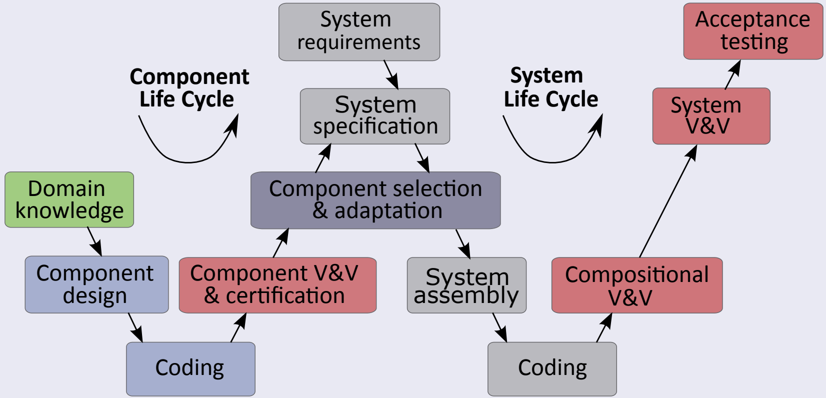
CBSE Life Cycle in a Domain


Component Life Cycle (for one domain)
Domain knowledge
Component Design
Design & implementation of (domain-specific) of components
Repository
Component Deployment
Deployment of components in a specific system
System requirements
System specification
Component selection & adaptation
System Assembly
Composition of deployed components
Architecture
System Life Cycle (for one system)

- Context for CBSE is a domain (of multiple systems)
- Separate life cycles for components and systems

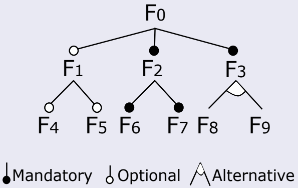
Compositional V & V

The W Model

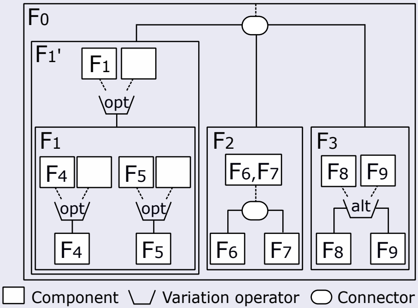


K.-K. Lau, F. Taweel and C. Tran. The W Model for Component-based Software Development. In *Proc. 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 47-50, IEEE, 2011.

Product Families in a Domain



Feature model



Product family architecture

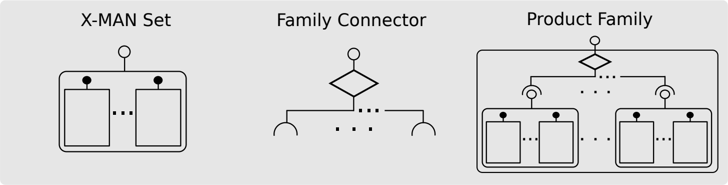
Domain Model = Feature Model + Functional Model (Behaviour)

Domain Engineering = Domain Knowledge ⟹ Domain Model ⟹ Product Family Architecture (Reference Architecture)

K.-K. Lau and S. Di Cola.(Reference) Architecture = Components + Composition (+ Variation Points)? In *Proc. 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures*, pages 1-4, ACM, 2015.

FX-MAN Component Model

Family Composition

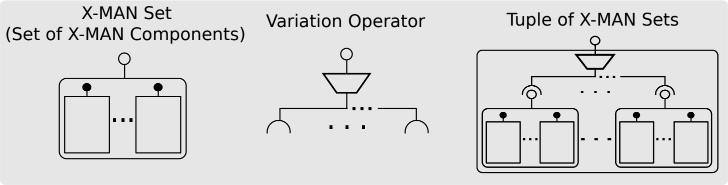


X-MAN Set

Family Connector

Product Family

Variation Generation



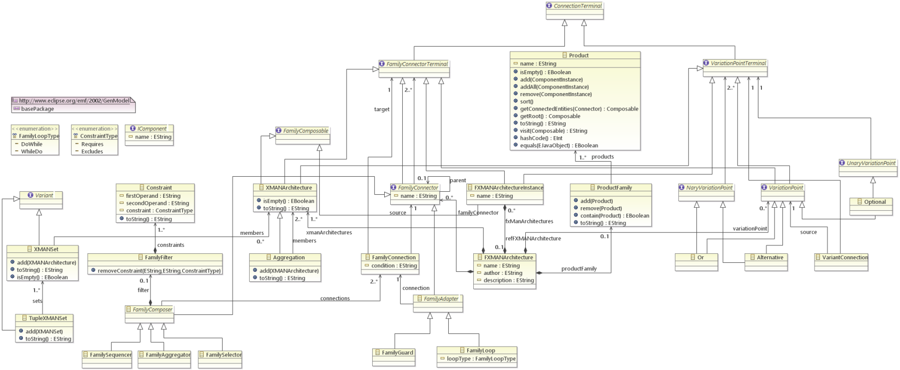
X-MAN Set
(Set of X-MAN Components)

Variation Operator

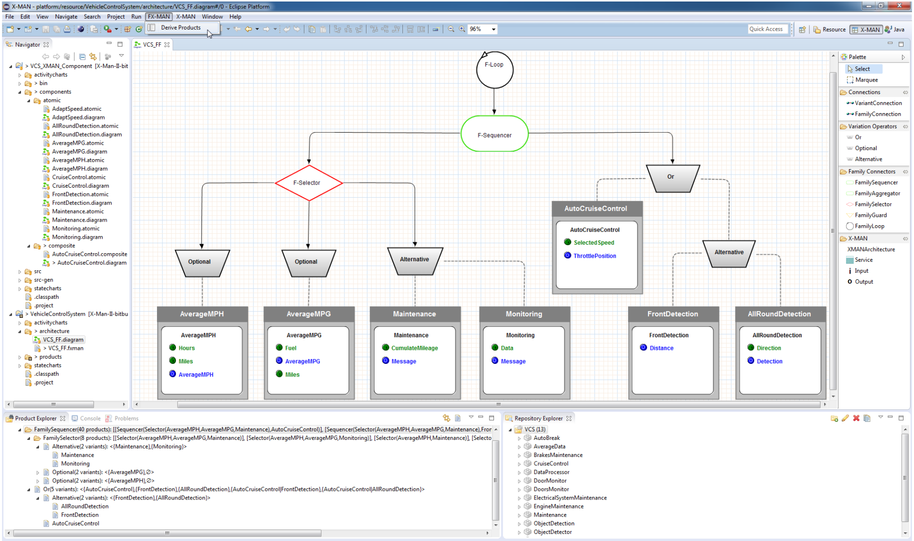
Tuple of X-MAN Sets

S. Di Cola, C. Tran, K.-K. Lau, C. Qian and M. Schulze. A Component Model for Defining Software Product Families with Explicit Variation Points. In *Proc. 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering*, pages 79-84, ACM, 2016.

FX-MAN Meta-model



MDE Tool for FX-MAN



```
> VCS_XMAN_Component [X-Man-II-bit
  activitycharts
  > bin
  > components
    atomic
      AdaptSpeed.atomic
      AdaptSpeed.diagram
      AllRoundDetection.atomic
      AllRoundDetection.diagram
      AverageMPG.atomic
      AverageMPG.diagram
      AverageMPH.atomic
      AverageMPH.diagram
      CruiseControl.atomic
      CruiseControl.diagram
      FrontDetection.atomic
      FrontDetection.diagram
      Maintenance.atomic
      Maintenance.diagram
      Monitoring.atomic
      Monitoring.diagram
    > composite
      AutoCruiseControl.composite
      > AutoCruiseControl.diagram
  src
  src-gen
  statecharts
  .classpath
  .project
> VehicleControlSystem [X-Man-II-bitbu
  activitycharts
  > architecture
    VCS_FF.diagram
    > VCS_FF.fxman
  > products
  statecharts
  .classpath
  .project
```

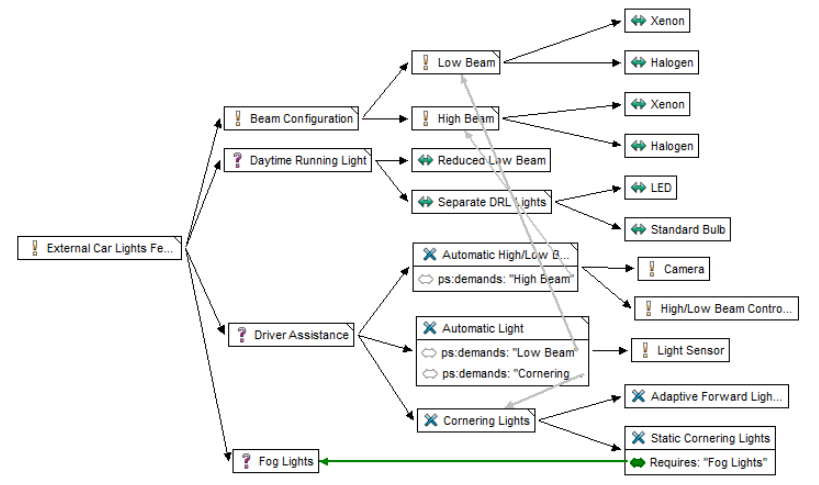
```
FamilySequencer(40 products): [[Sequencer(Selector(AverageMPH,AverageMPG,Maintenance),AutoCruiseControl)], [Sequencer(Selector(AverageMPH,AverageMPG,Maintenance),Fror
  FamilySelector(8 products): [[Selector(AverageMPH,AverageMPG,Maintenance)], [Selector(AverageMPH,AverageMPG,Monitoring)], [Selector(AverageMPH,Maintenance)], [Selecto
    Alternative(2 variants): <{Maintenance},{Monitoring}>
      Maintenance
      Monitoring
    Optional(2 variants): <{AverageMPG},∅>
    Optional(2 variants): <{AverageMPH},∅>
  Or(5 variants): <{AutoCruiseControl},{FrontDetection},{AllRoundDetection},{AutoCruiseControl|FrontDetection},{AutoCruiseControl|AllRoundDetection}>
    Alternative(2 variants): <{FrontDetection},{AllRoundDetection}>
      AllRoundDetection
      FrontDetection
    AutoCruiseControl
```

```
VCS (13)
  AutoBreak
  AverageData
  BrakesMaintenance
  CruiseControl
  DataProcessor
  DoorMonitor
  DoorsMonitor
  ElectricalSystemMaintenance
  EngineMaintenance
  Maintenance
  ObjectDetection
  ObjectDetector
```

S. Di Cola, K.-K. Lau, C. Tran and C. Qian. An MDE Tool for Defining Software Product Families with Explicit Variation Points. In *Proc. 19th International Conference on Software Product Line*, pages 355-360, ACM, 2015.

FX-MAN Example: Family of ECL Products

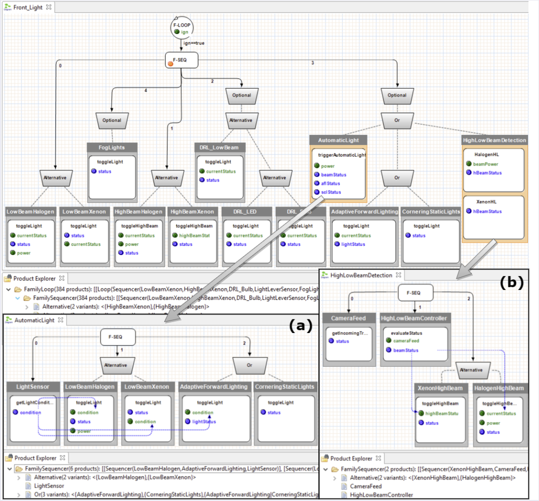
Feature Model



386 product variants (28688 without constraints)

FX-MAN Example: Family of ECL Products

Product Family Architecture



Product Explorer

- FamilyLoop(384 products): [[Loop(Sequencer(LowBeamXenon,HighBe Xenon,DRL_Bulb,LightLeverSensor,FogLigh
  - FamilySequencer(384 products): [[Sequencer(LowBeamXenon ghBeamXenon,DRL_Bulb,LightLeverSensor,FogL
    - Alternative(2 variants): <{HighBeamXenon},{HighBea alogen}>

Product Explorer

- FamilySequencer(6 products): [[Sequencer(LowBeamHalogen,AdaptiveForwardLighting,LightSensor)], [Sequencer(Lo
  - Alternative(2 variants): <{LowBeamHalogen},{LowBeamXenon}>
  - LightSensor
  - Or(3 variants): <{AdaptiveForwardLighting},{CorneringStaticLights},{AdaptiveForwardLighting|CorneringStaticLig

Product Explorer

- FamilySequencer(2 products): [[Sequencer(XenonHighBeam,CameraFeed,
  - Alternative(2 variants): <{XenonHighBeam},{HalogenHighBeam}>
  - CameraFeed
  - HighLowBeamController

386 product variants (28688 without constraints)

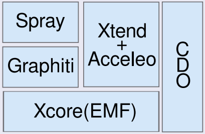
CBSE + MDE in Our Approach: Summary

Models and Transformations

Component Model	Model	M2M	Model	M2T	Implementation
X-MAN	Functional Model	←	Component	→	Code
X-MAN	Functional Model	←	Component Composition	→	Code
FX-MAN	Product Family	Variation operator ←	Product Family		
FX-MAN	Functional Model	←	Family Composition		
FX-MAN	Functional Model	←	Product Family Architecture		

Functional Model = State Charts + Activity Charts
Product Family = Set of X-MAN Architectures

Technology Stack



Closing

What We Have Done

- We use models everywhere
- We use MDE for tool development
- We have not focused on platforms, or associated M2M transformations

Tool Downloads

`http://www.click2go.umip.com/i/software/x_man.html`

Feedback most welcome!

Closing

Some Observations/Questions on MDE Technology

- More modelling elements?
  - composition (not just association and containment)
  - components (units that are more compositional than classes)
  - behaviour (e.g. control, coordination)
- Higher-level abstractions?
  - not just classes
  - less coupled to OO technology
  - more hierarchical modelling (more than referencing)
  - model transformations may be challenging