





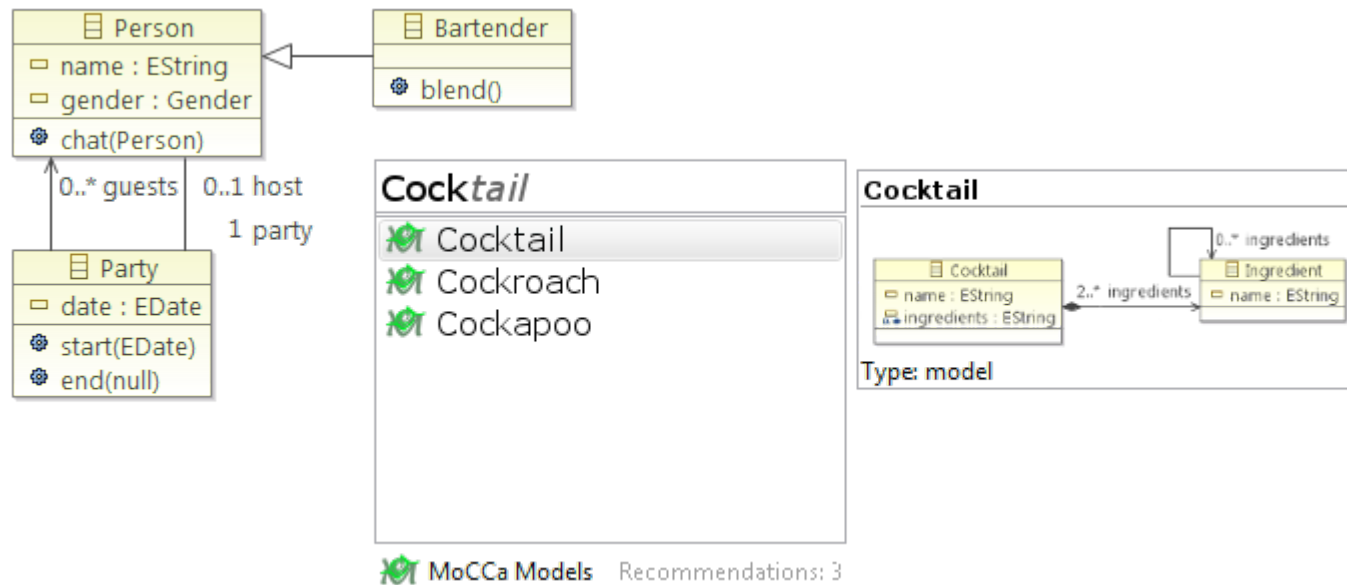
# Proactive Quality Guidance for Model Evolution in Model Libraries

Andreas Ganser, Horst Lichter, Alexander Roth, and Bernhard Rumpe

-  Setting the Scene
-  If You Take One Thing
-  The Details
-  Some References

# Setting the Scene ...

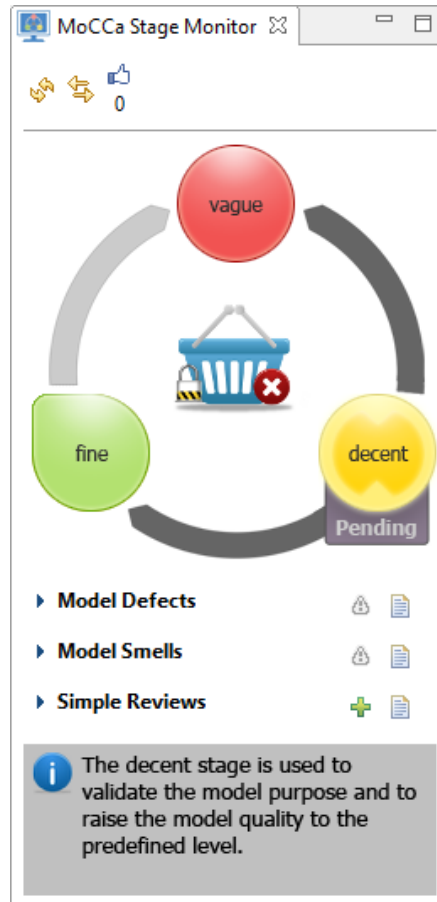
## Model Recommenders and Model Libraries



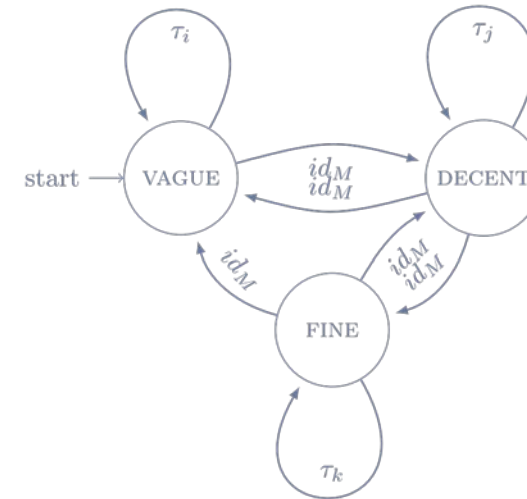
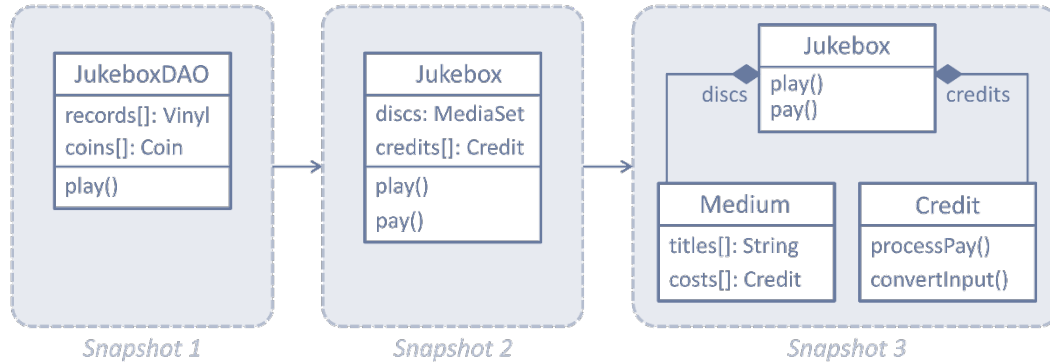
Do models change here?

# If You Take One Thing ...

Models Evolve in Model Libraries and Need Guidance



## How do they evolve?



## Evolving Models

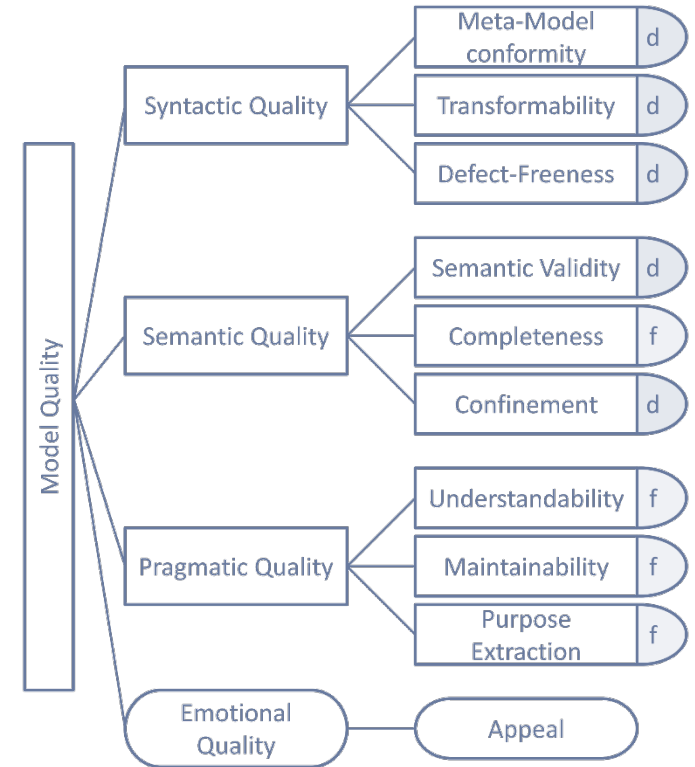
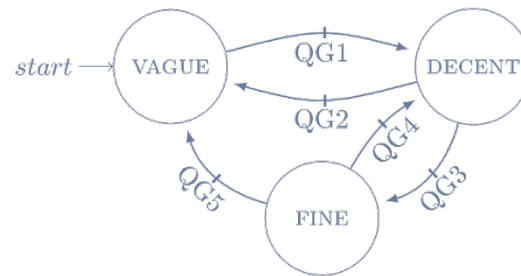
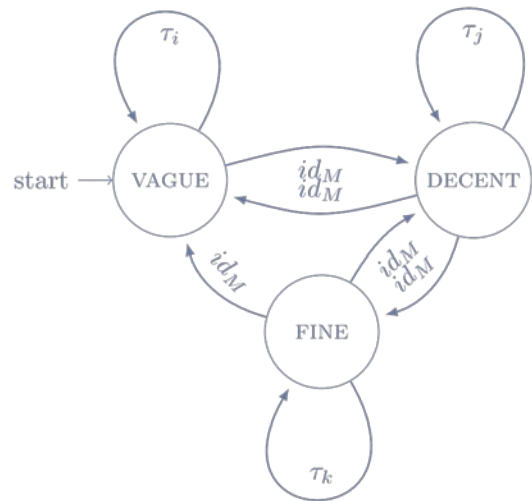
- Put model under monitoring
- Review model and set quality gates
- Resolve model issues and enhance it

- Focus: evolution workflow support

## Evolution Stages

- Vague
- Decent
- Fine

Goal: reusable, recommendable models



## Foundation for Proactiveness

- Strong Attributes
  - Defects
  - Syntax checker & metrics
  - Checker
- Medium Attributes
  - Smells
  - Metrics & reviews
  - Thresholds
- Weak Attributes
  - Hunches
  - Reviews & judgement
  - Thinking hats



## Foundations for Guidance

- Defect
  - Dangling references
  - Missing names

➔ Not well formed
- Smell
  - Too many classes
  - Good class

➔ Not well extracted
- Hunch
  - Design contradicts content
  - Design is awkward

➔ Not well designed



How to enable this?

# Metrics and Simple Reviews

## Existing Metric Suites

- Use what's there ...
- C&K Suite, Frankel, Genero, Martin, Ramirez, ...
- Link to quality model

## Simple Reviews

- “Real” reviews too complex
- Simplified reviews (streamlining)
- Idea:
  - Six Thinking Hats become Five Review Hats
  - Yellow Hat (Good Points Judgment)
  - Black Hat (Bad Points Judgment)
  - White Hat (Information)
  - Green Hat (Creativity)
  - Red Hat (Emotions)



Model Defect Metrics		
Name	Description	Reference
AttributeNameOvr	The class defines a property of the same name as an inherited attribute. During code generation, this name inadvertently hides the attribute of the parent class. Consider changing the name of the attribute in the child class.	A. Ramirez, P. Vanoverbeke, A. August, K. Gohara, J. Bertram, L. Talar, M. van der Wal, "AgileM User Manual", 2005.

Model Smell Metrics		
Name	Description	Reference
Long Method	The longer the method the harder it is to see what it's doing.	Fowler, Martin: Refactoring - Improving the Design of Existing Code. Addison-Wesley, 1999
Long Parameter List	Don't pass in everything the method needs; pass in enough so that the method can get to everything it needs.	Fowler, Martin: Refactoring - Improving the Design of Existing Code. Addison-Wesley, 1999
Duplicated Attributes/Methods/Classes	Duplicated attributes/methods/classes are bad	Fowler, Martin: Refactoring - Improving the Design of Existing Code. Addison-Wesley, 1999
Large Class	A class that is trying to do too much can usually be identified by looking at how many instance variables it has. When a class has too many instance variables, duplicated code is not to be believed.	Fowler, Martin: Refactoring - Improving the Design of Existing Code. Addison-Wesley, 1999
Inappropriate Intimacy	Two classes are overly intertwined.	Fowler, Martin: Refactoring - Improving the Design of Existing Code. Addison-Wesley, 1999
Number of Attributes	If the amount of attributes reaches a certain level	M. Linton, J. Kidd, Object-oriented Software Metrics, Prentice Hall, 1994.
Number of Aggregations	The total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship)	M. Genero, Non-Redundant Metrics for UML Class Diagram Structural Complexity, Advanced Information Systems Engineering, Vol. 3161, (2003), 127-142
Number of Dependencies	The total number of dependency relationships.	Chamberlain S.L., Kemerer C.F., A metrics suite for object-oriented design, Software Engineering, IEEE Transactions, Vol. 20, (1994), 878-893

The screenshot displays the MoCCa Evolution software interface. The main window shows a UML class diagram for a "Lecture Registration System". The diagram includes classes: Lecture, Student, Person, Lecturer, ResearchAssistant, Natural Person, ContactInfo, and Capability. Relationships are shown with arrows and multiplicities (e.g., 1 to 1.2, 1 to 1.\*).

On the right side, the "MoCCa Stage Monitor" panel is visible. It features a circular diagram with three stages: "vague" (red), "fine" (green), and "decent" (yellow). A shopping cart icon with a red 'X' is positioned in the center of the circle. Below the diagram, there are sections for "Model Defects", "Model Smells", and "Simple Reviews". An information icon (i) is present, with a text box stating: "The decent stage is used to validate the model purpose and to raise the model quality to the predefined level."

At the bottom of the interface, there is a "MoCCa Graph UI Search" section with a search input field and radio buttons for search types: "infix" (selected), "whole words", "Lucene", and "First Exact Then Infix Search".



# Some References



What else is going on ...?

## The HERMES Project

# HERMES

HARVEST



EVOLVE



REUSE



MODELS EASILY AND SEAMLESSLY

## References

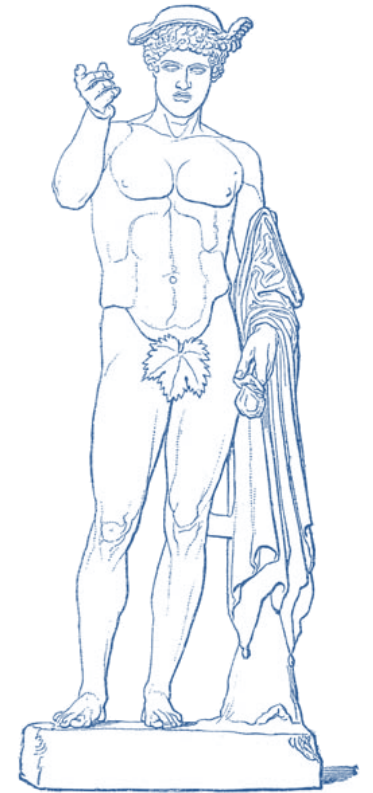
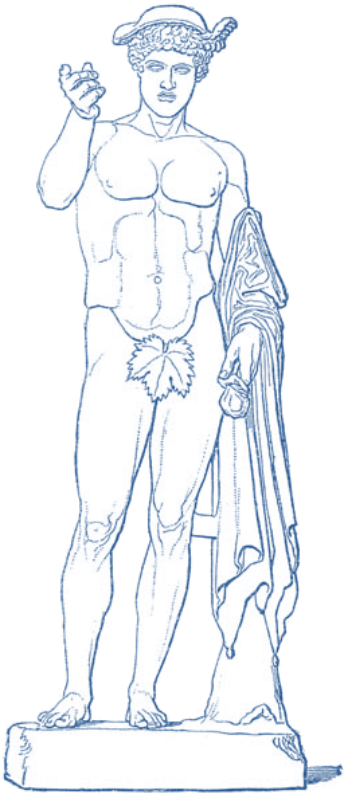
A. Ganser, H. Lichter, Engineering Model Recommender Foundations – From Class Completion to Model Recommendations, (Modelsward 2013, Spain)

A. Ganser, T. N. Viet, H. Lichter, Multi Back-Ends for a Model Library Abstraction Layer, (ICCSA 2013, Vietnam)

A. Dyck, A. Ganser, H. Lichter, Enabling Model Recommenders for Command-Enabled Editors, (MoDELS MDEBE 2013, US)

*and more to come on*

Model Recommender UI Survey,  
Framework Internals, Contexts / Scanners



**Thanks for your attention**

... any questions?