http://www.isis.vanderbilt.edu

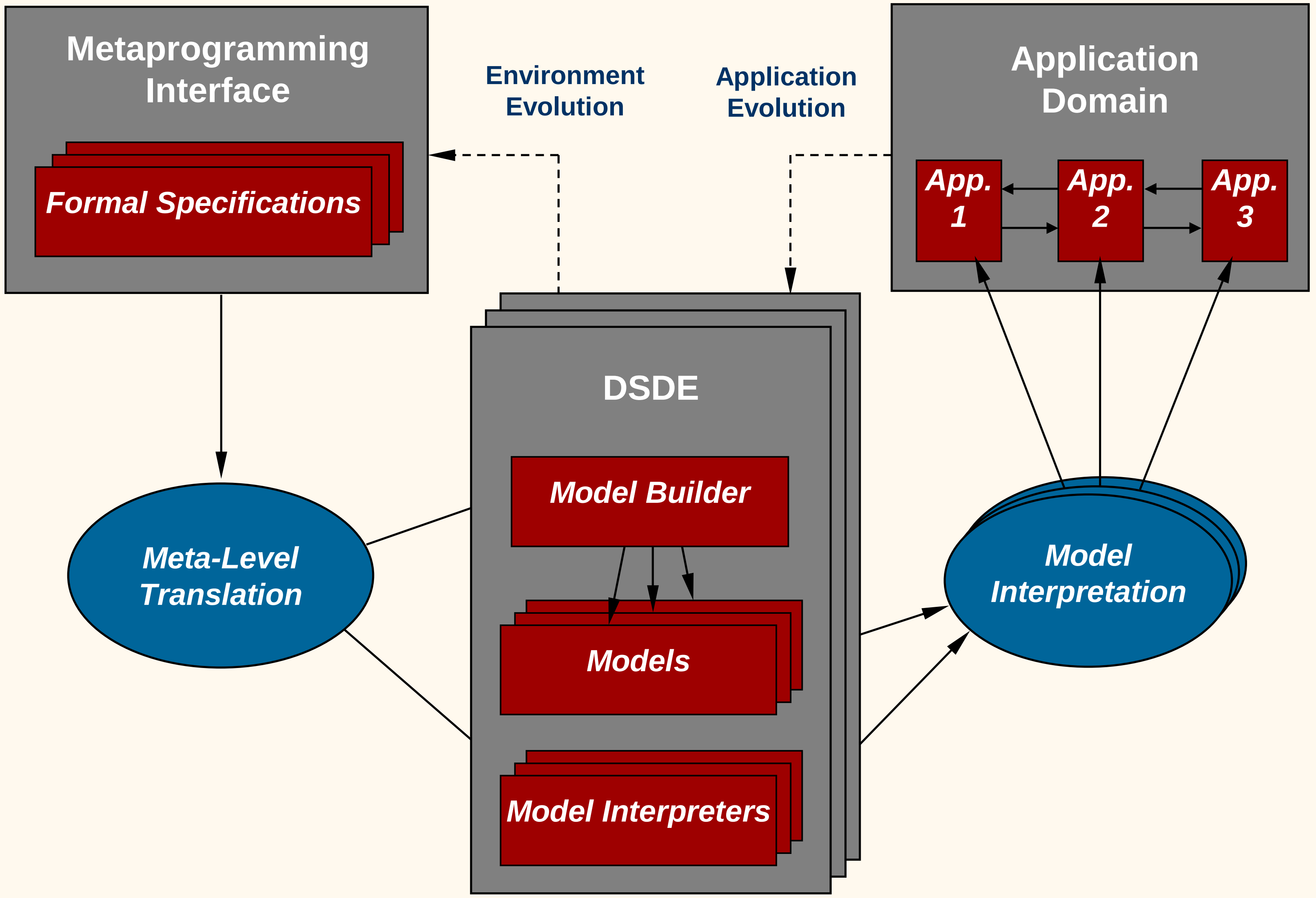
Composable Metamodeling Environment

Akos Ledeczi

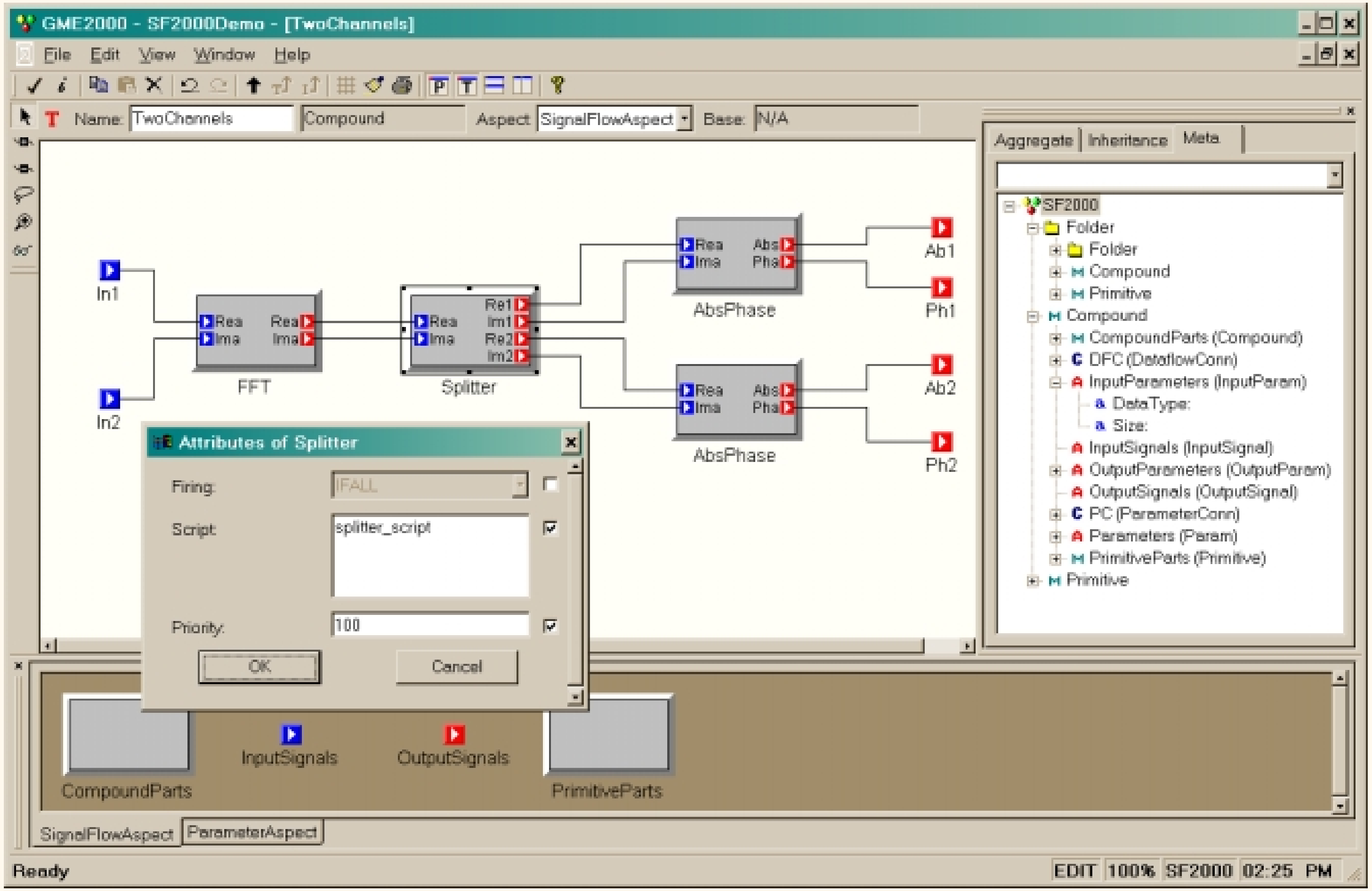
Institute for Software Integrated Systems

Vanderbilt University

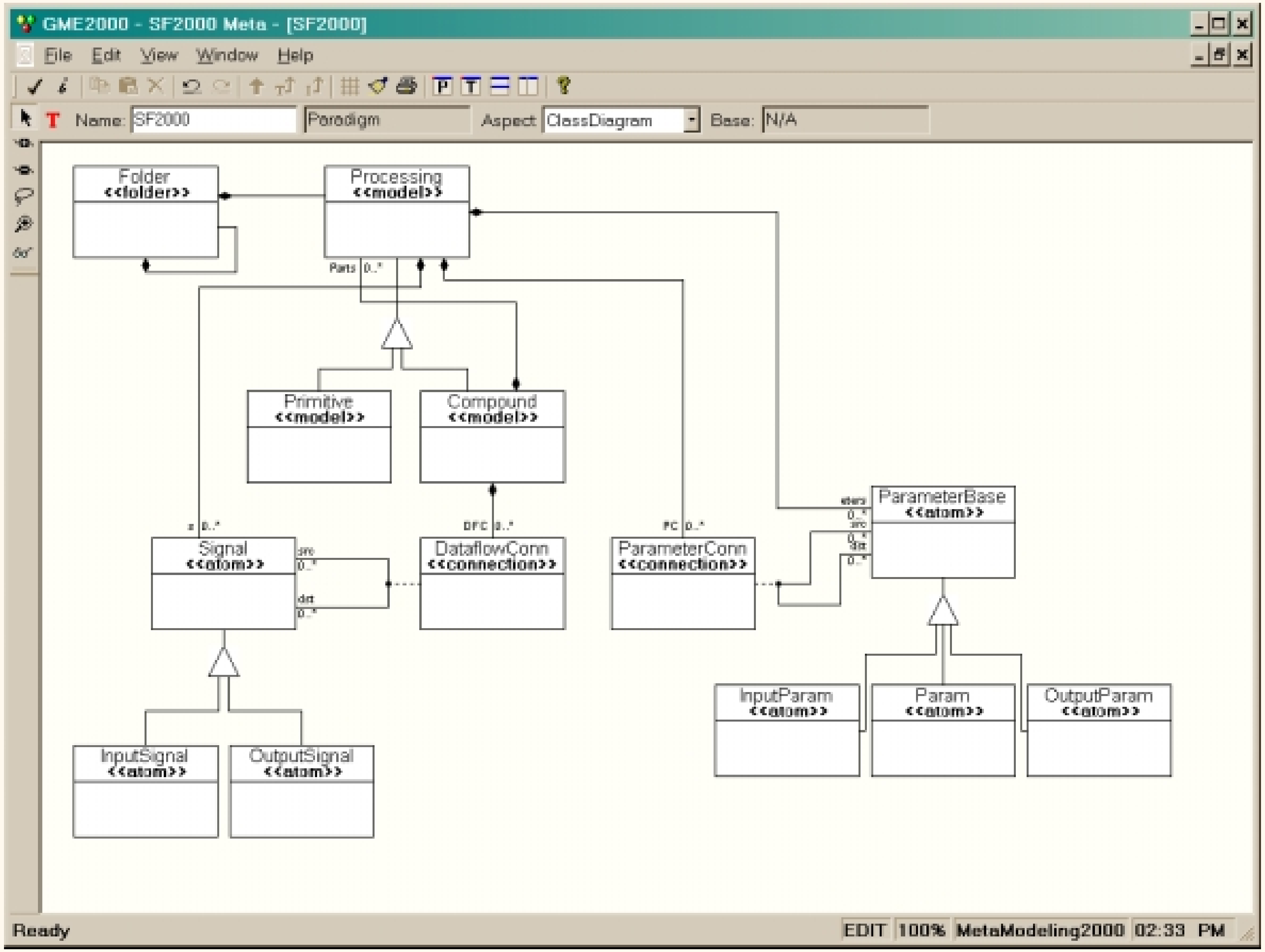
Model Integrated Computing



Signal Flow Models



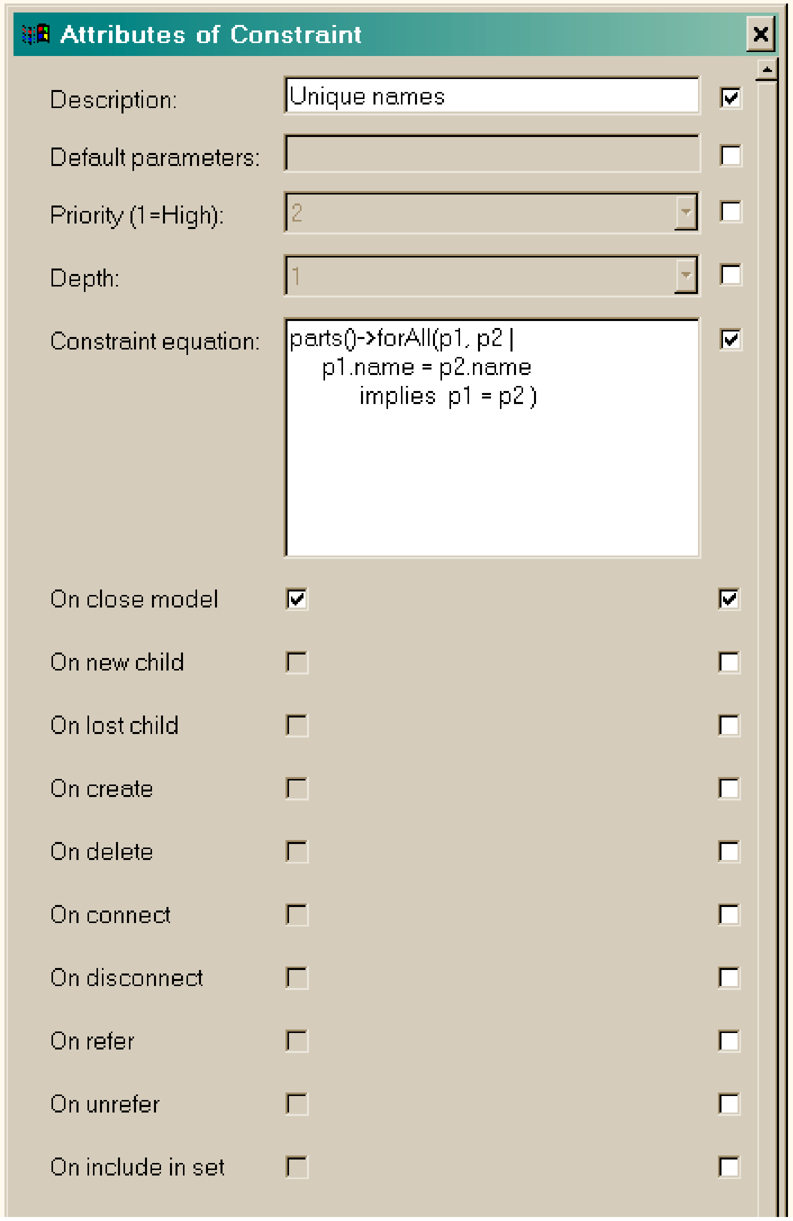
Signal Flow MetaModels


GME2000 - SF2000 Meta - [SF2000]

Name: SF2000
Paradigm
Aspect: ClassDiagram
Base: N/A

Ready
EDIT 100% MetaModeling2000 02:33 PM

Signal Flow Constraints



Description:	Unique names	☑
Default parameters:		☐
Priority (1=High):	2	☐
Depth:	1	☐

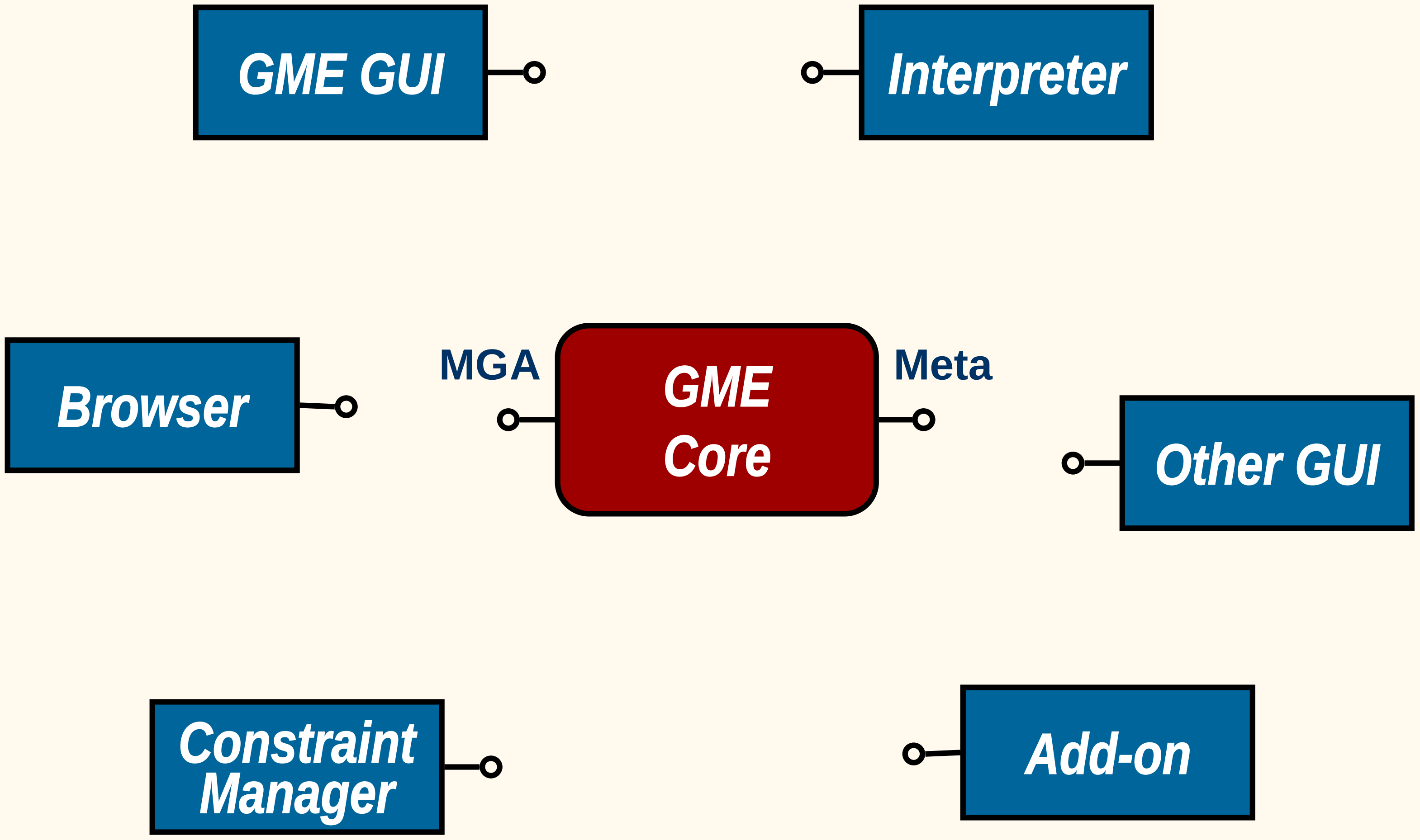
Constraint equation: ☑

```
parts()->forAll(p1, p2 |
    p1.name = p2.name
        implies  p1 = p2 )
```

On close model	☑	☑
On new child	☐	☐
On lost child	☐	☐
On create	☐	☐
On delete	☐	☐
On connect	☐	☐
On disconnect	☐	☐
On refer	☐	☐
On unrefer	☐	☐
On include in set	☐	☐

- Event-based and on-demand
- Context
- Based on the Object Constraint Language (OCL)
- Priority

GME 2000 Components



GME 2000 Features

- *COM-based modular architecture*
- *Database storage*
- *Distributed, multi-user access*
- *Type inheritance*
- *Libraries*
- *Event-based constraint manager*
- *Multi-level undo/redo*
- *GME-, paradigm-, project-specific help*

Design goals

- ***Simplify metamodels (increase readability)***
- ***Reuse existing metamodels***
- ***Compose paradigms from subparadigms***
- ***Create metamodel libraries***
- ***Do NOT modify reused metamodels***

New features

- ***Multi-sheet capability:***
  - ***Proxy: Reference to a UML class***
  - ***represents the exact same object***
  - ***only attribute: abstract***
- ***New operators:***
  - ***Equivalence***
  - ***Implementation inheritance***
  - ***Interface inheritance***

Equivalence (union)

- *Two objects are the “same”, i.e. a new object is created that is the union of the two*
- *Represent the points where two subparadigms join together*
- *Can be emulated by a new UML class derived from both and making the originals abstract*

Implementation Inheritance

- ***Finer control over inheritance***
- ***Analogous to private inheritance in C++***
- ***Inherits what's "inside" a class:***
  - ***Attributes***
  - ***All composition relations where given class is the parent***

Interface Inheritance

- ***Finer control over inheritance***
- ***Analogous to interface inheritance in Java***
- ***Inherits what's "outside" a class:***
  - ***All associations***
  - ***All composition relations where given class is the child***

Inheritance cont’d.

- ***Can be emulated using regular UML, but only by modifying original metamodel***
- ***The union of implementation and interface inheritance is pure UML inheritance (operators are applied sequentially in any order)***