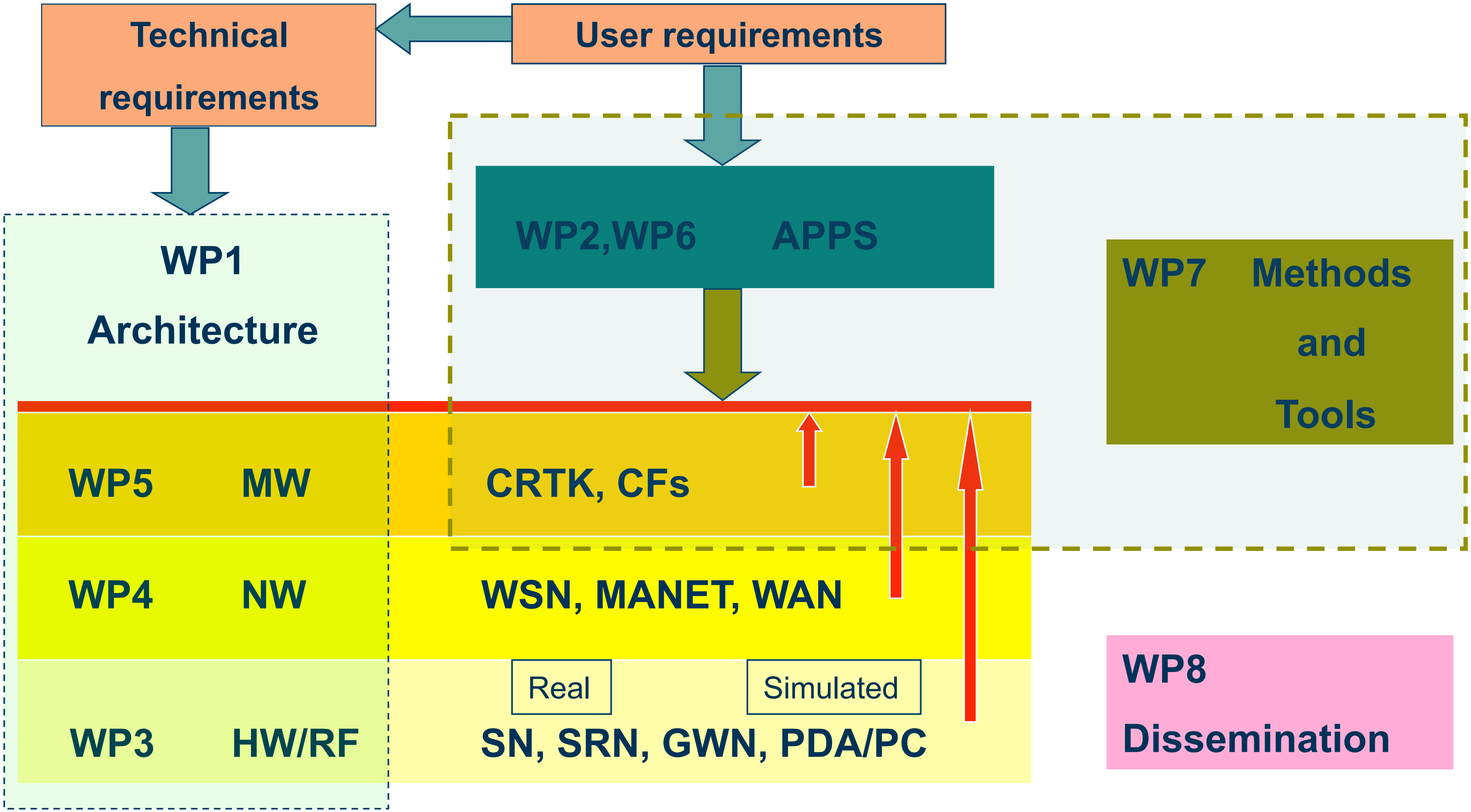
Model Driven Software Development

- Architecture design plays a decisive role in the process
- Product line development style is essential
- Meta-modeling stands in the centre
- Application development concentrates on model creation
- Architecture development concentrates on translator creation
- Run-time platform features are heavily relied on
- Tool support is the enabler of the process

Architecture Design

- Architecture defines the scope of the endeavor in the project
- Well designed architecture provides wide scale applicability of the result
- Architecture design enables easy interfacing among various work-packages in integrated projects. (RUNES is a multi-work-package project !!)

Architecture Design



Meta-modeling

- Meta-modeling defines the domain knowledge formally providing an ontology with abstract syntax and static semantics
- Meta-modeling creates Domain Specific Languages which can refer to each other → It matches multi-work-package research and development processes well
- Meta-modeling provides easy reasoning both for domain experts and domain users
- In RUNES: Scenario-to-Application Development, Semi-Automatic Test Case Generation

RUNES Platform

- RUNES Platform is an intermediate meta-model based on the RUNES middleware's Component Run-Time Kernel abstraction
- It is a UML-profile like classification based wrapping scheme.
- Its run-time implementations provide a reflective causal meta-interface to the connected components deployed in a heterogeneous hardware and software environment on different scales of computer powers.

Run-time platform

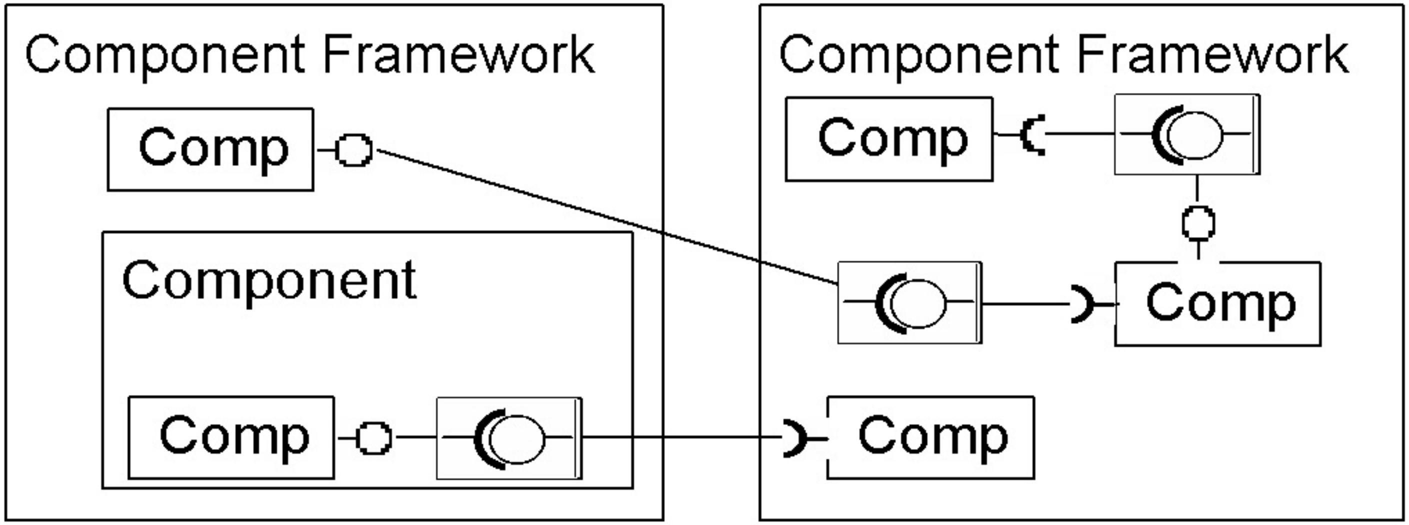
- One meta-model can be transformed onto different platforms providing various run-time features
- Feature selection is important as it is the reason behind profiling
- In RUNES:
  - Contiki CRTK in Telos motes (resource scarceness)
  - Java CRTK in laptops (easy portability)
  - C CRTK in gateways (efficiency)
  - Erlang CRTK in application servers (robustness, redundancy)

Erlang CRTK

- Robust
  - Fault tolerant, Highly available
- Reconfigurable
  - Adaptability to environmental changes
- Erlang
  - Ericsson's preferred language
  - Language elements support robust, reconfigurable behavior
  - Support for distributed deployment
- Component
  - Separation of functionality
  - Structured, reusable code
- System
  - Application neutral framework

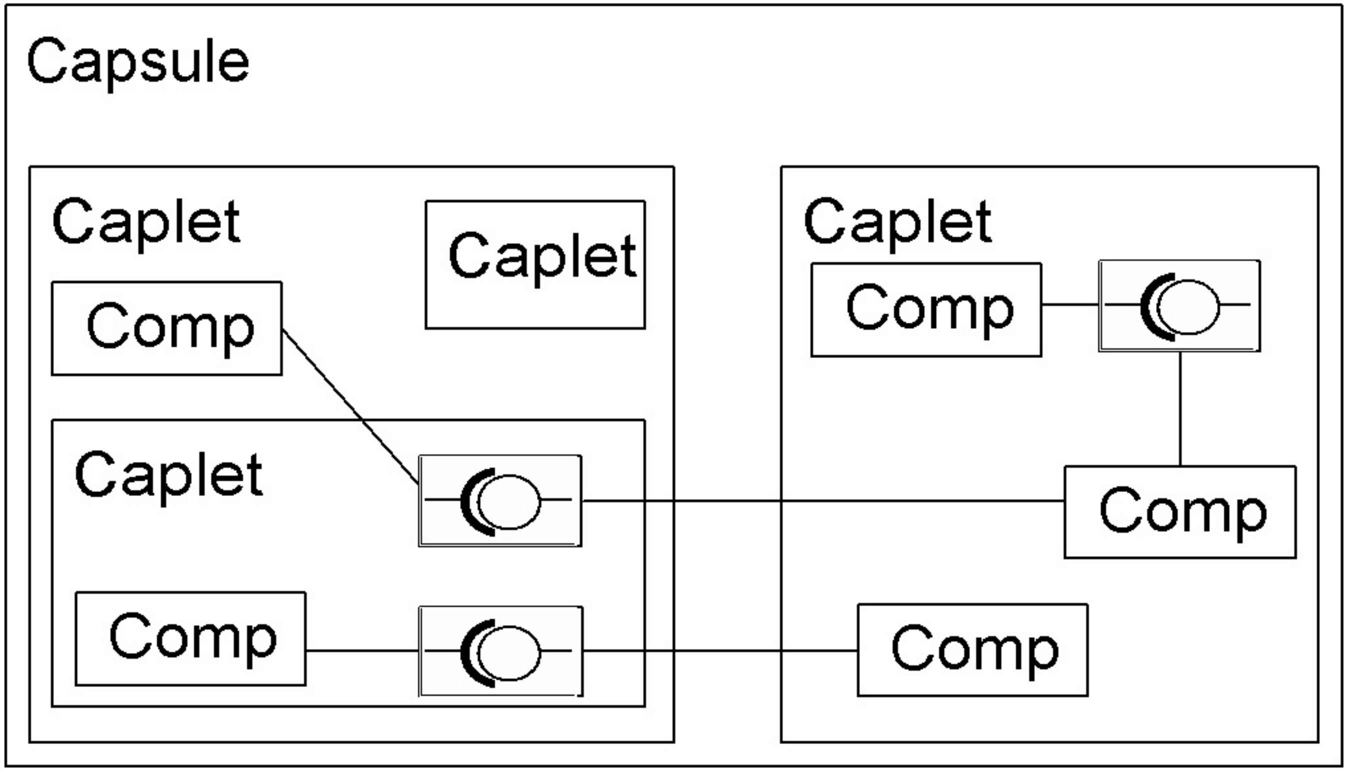
Functional Model

- Component, Composite Component – Functionality Owner
- Interface, Receptacle – Interaction Point Owner
- Binding – Communication Owner
- Component Framework – Constraint Owner

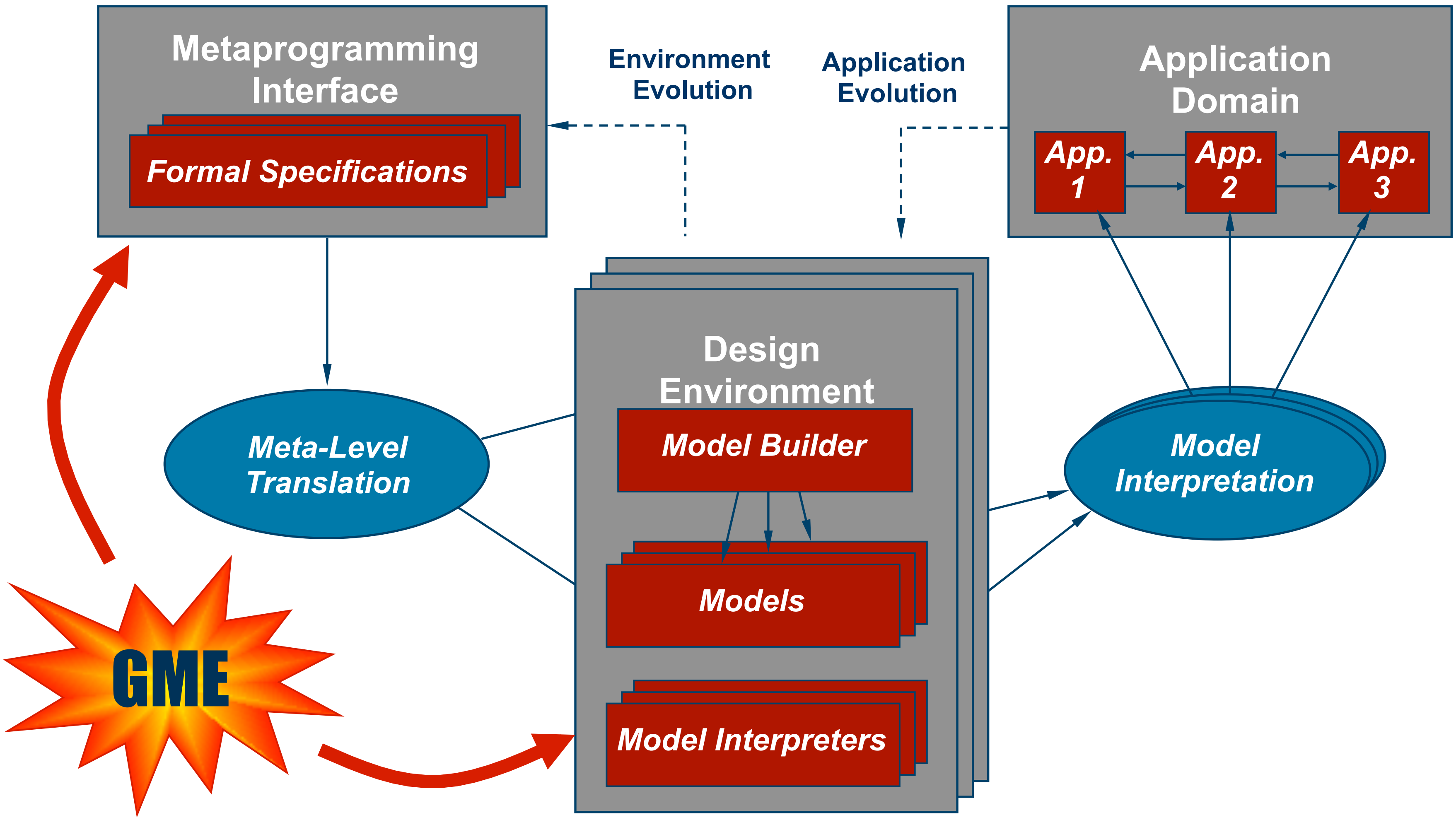


Deployment Model

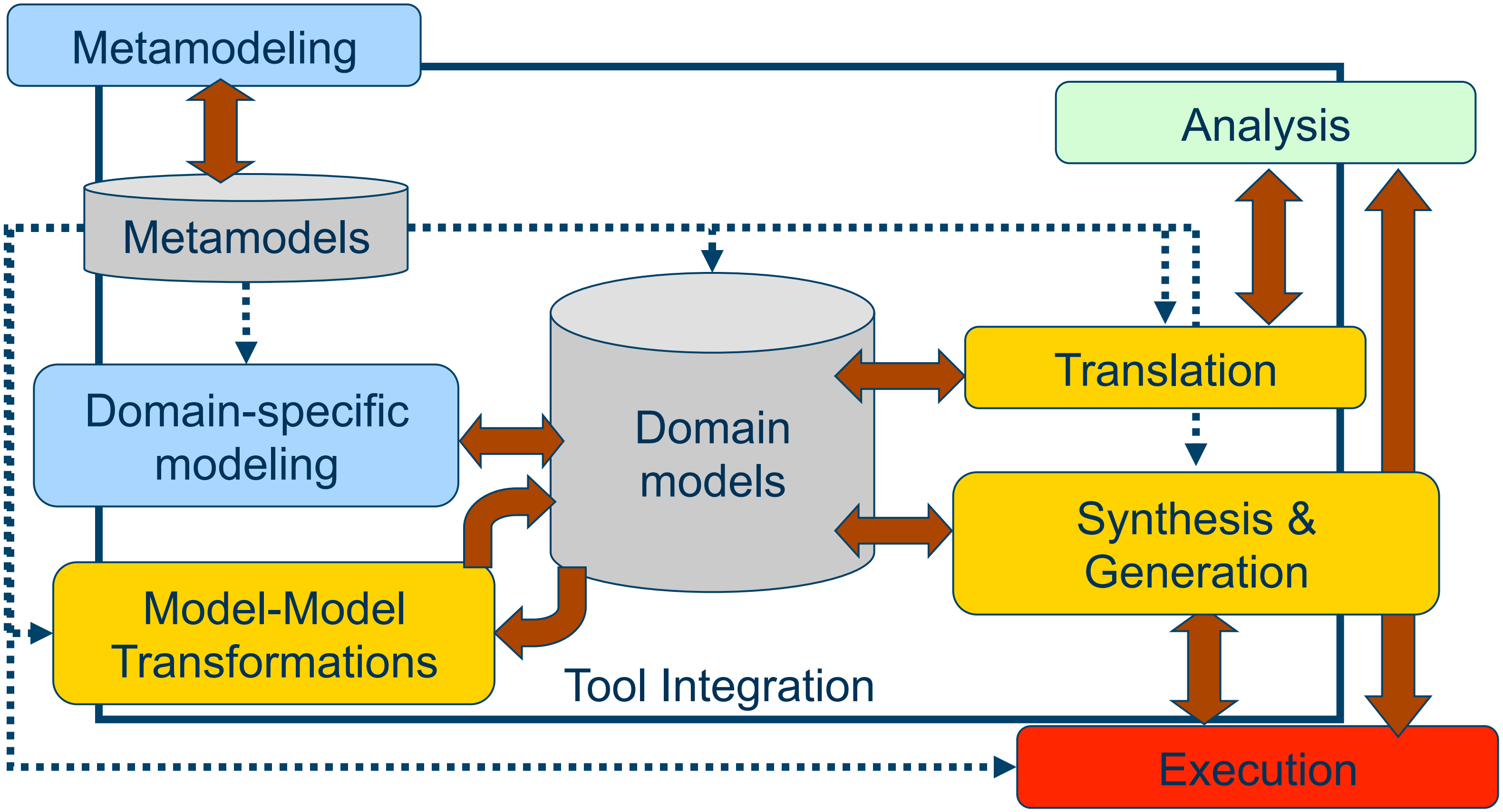
- Capsule – Supervision Owner
- Caplet – Component Owner
- Component – Functionality Owner



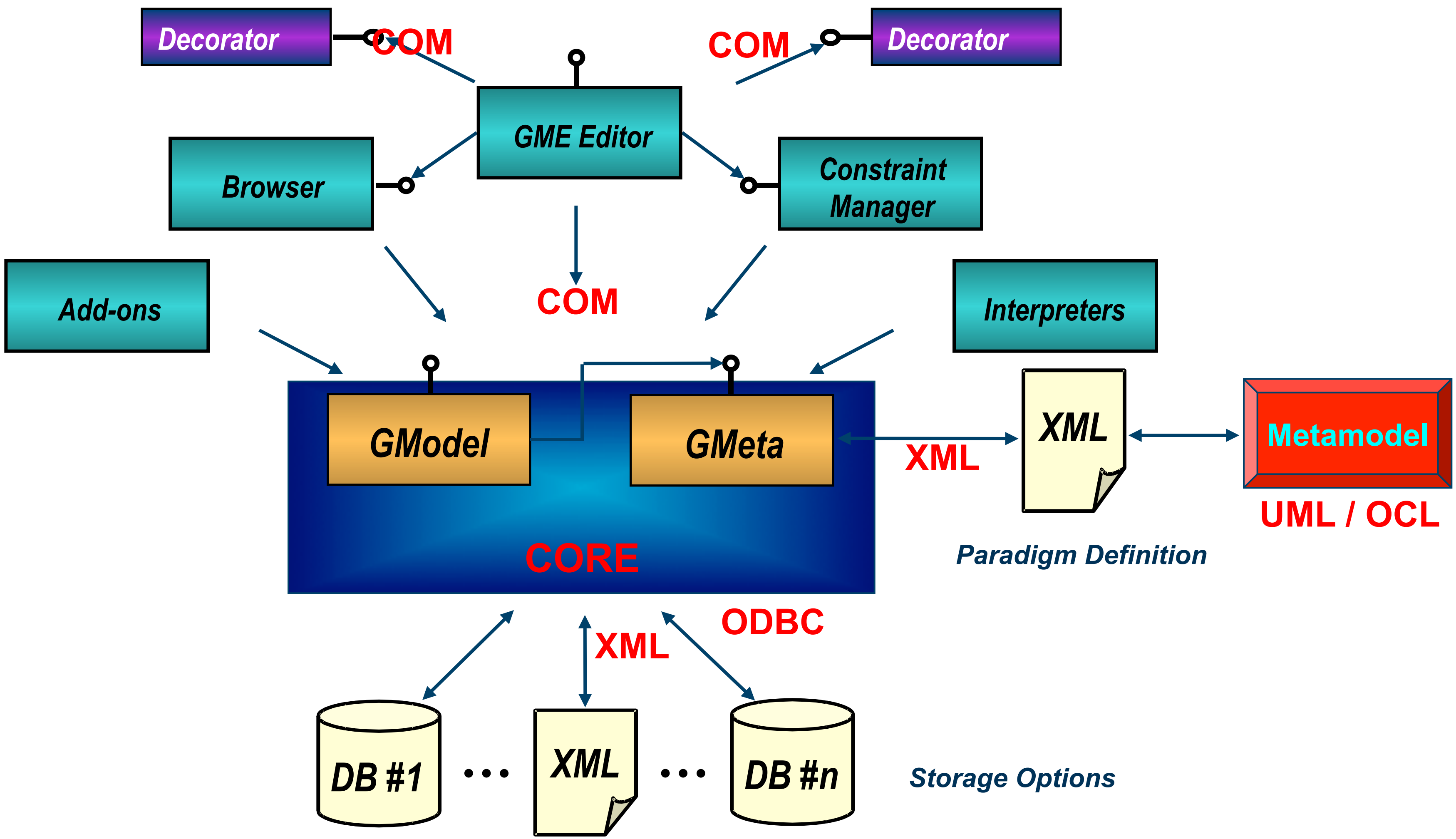
Model Integrated Computing



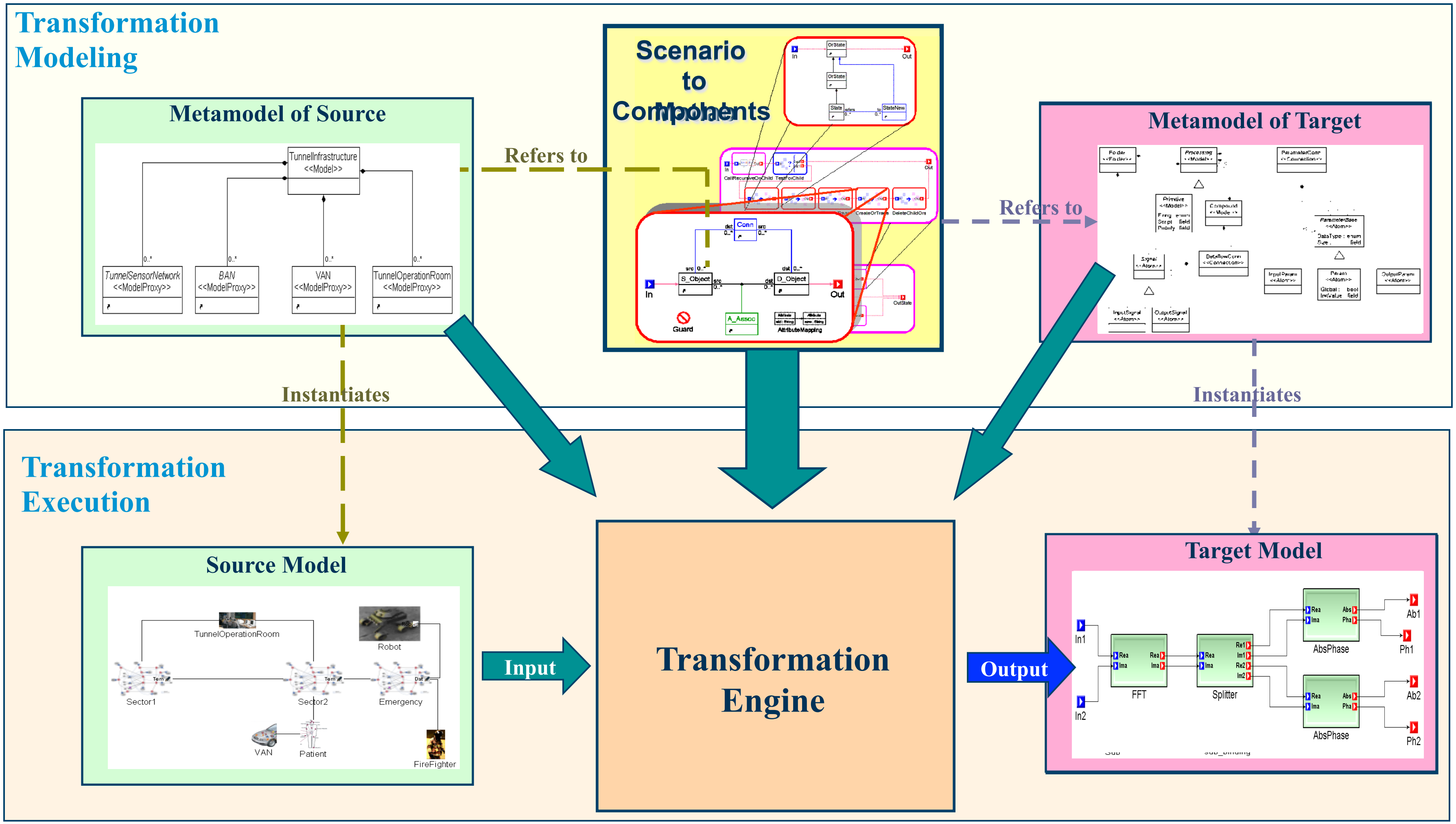
Model Integrated Computer Tool Chains



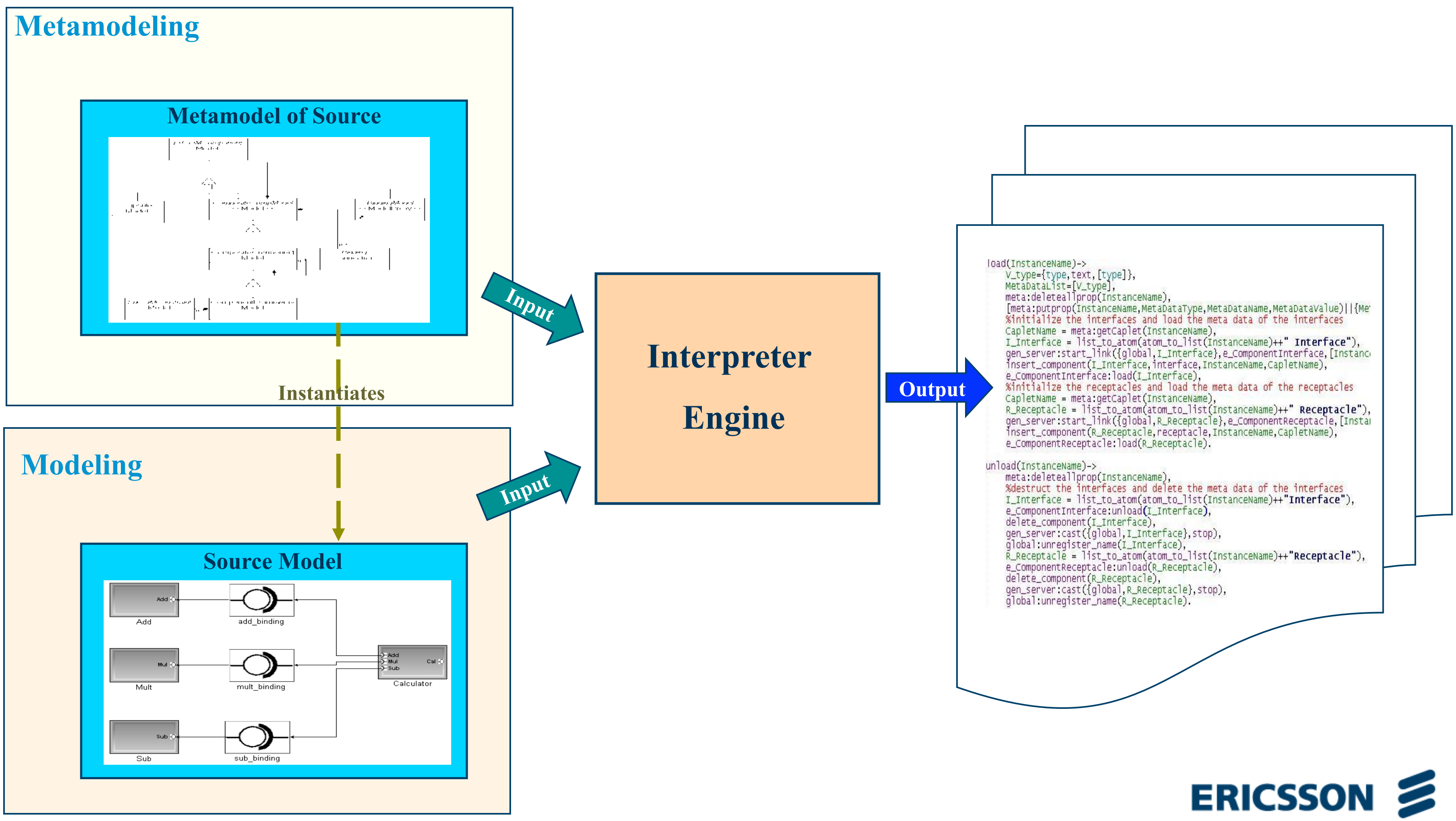
GME Architecture



Model Transformation



Model Transformation



ERICSSON
TAKING YOU FORWARD

Continuous Modelling

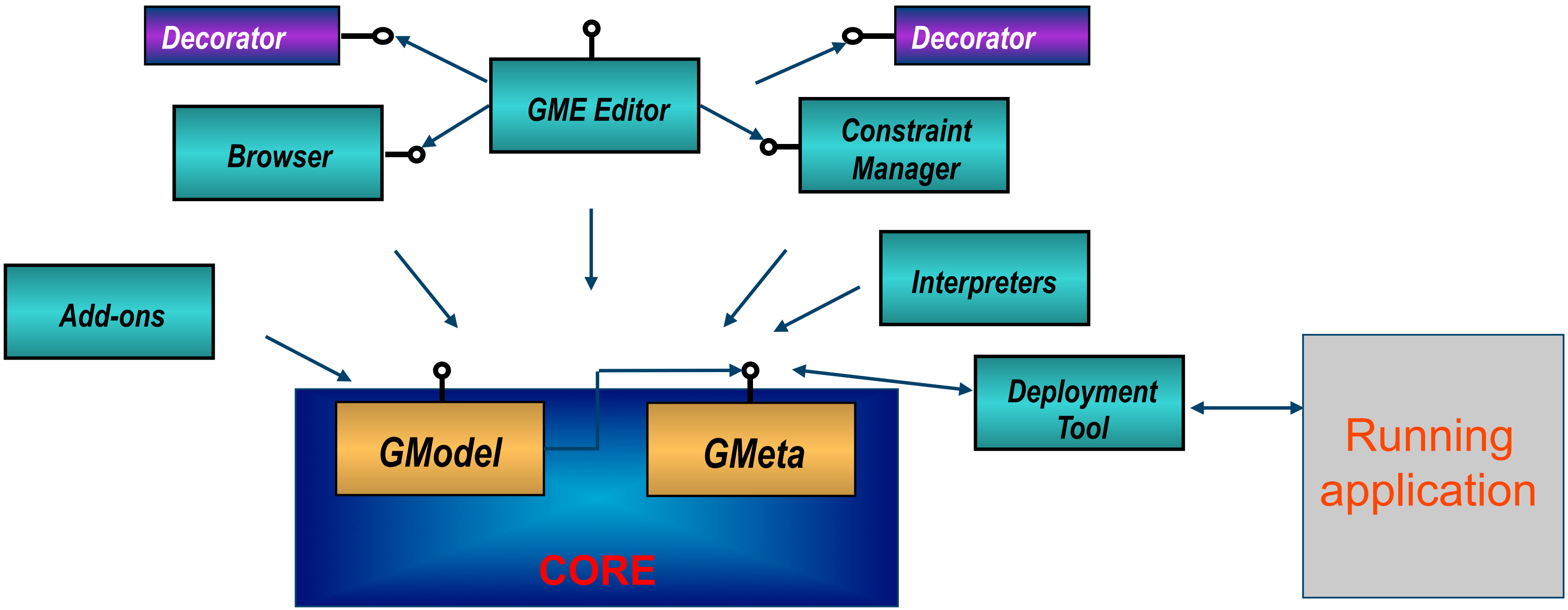
- Why can the modeler not be used as a Operation and Maintenance tool for the running application?
  - Source of the application is a model in GME
  - Code is reflective ➔it knows its meta-model

Reflect the changes in the running application into the model

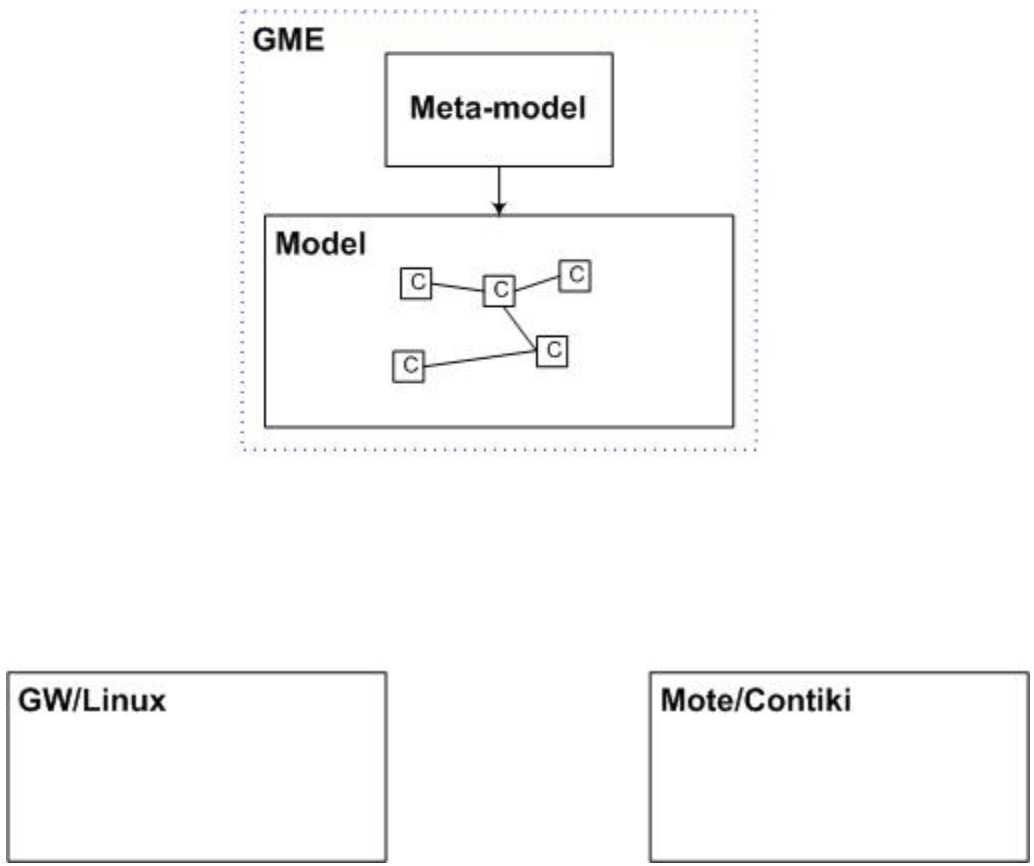
Deployment Tool

- Deploys components in a distributed system
- Stores the current configuration of the system
- Receives configuration change messages and modifies the model accordingly
- If the current state of the system is saved it can be redeployed accordingly later on
- Implementation platform (Erlang, C, Java) independent storage of the system state

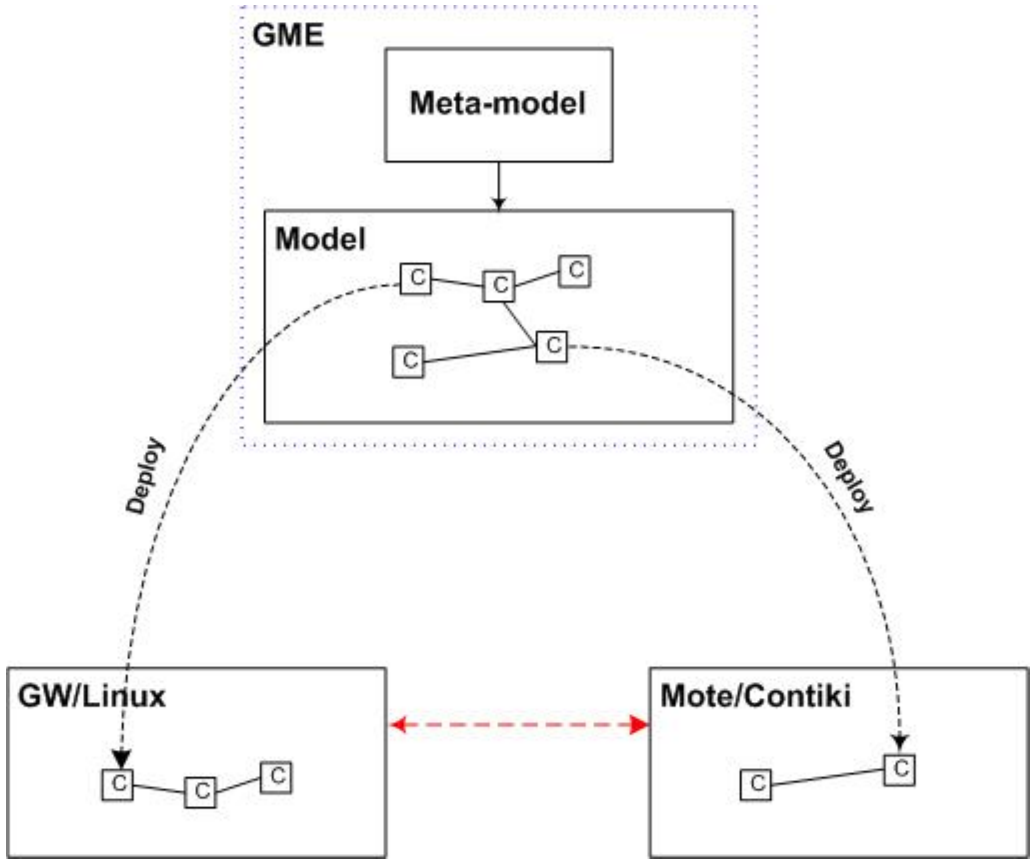
Deployment Tool in the GME Architecture



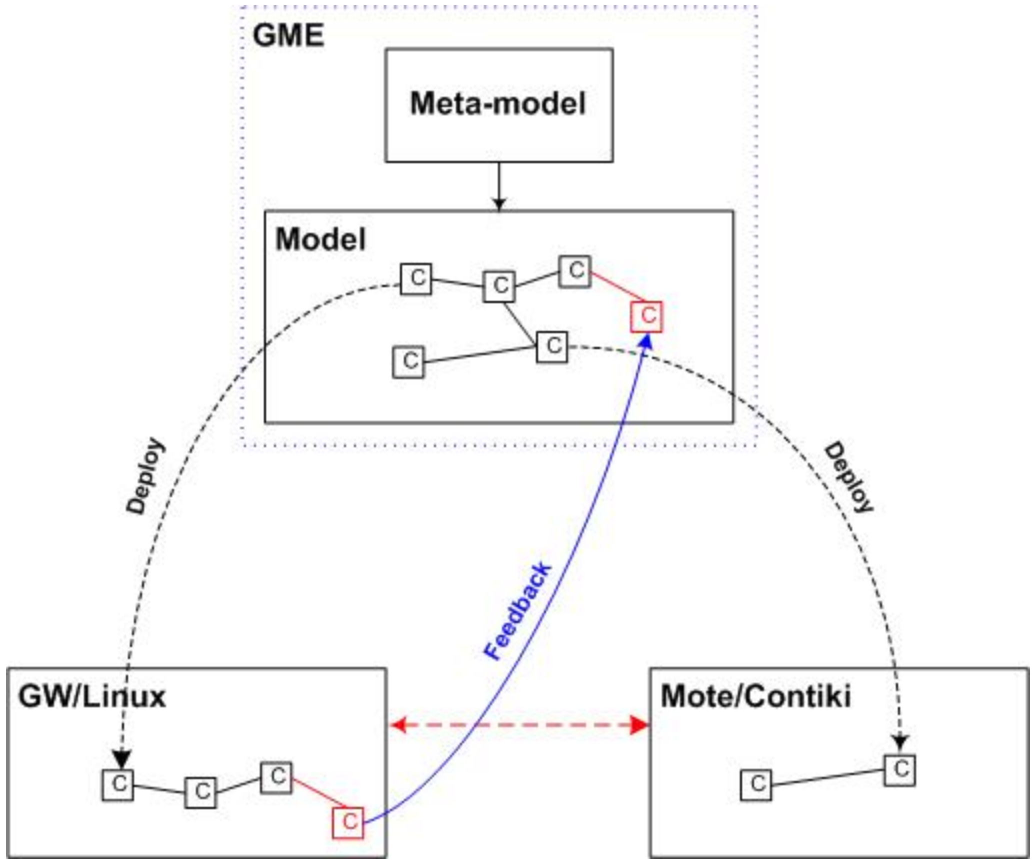
Behaviour of the Deployment Tool



Behaviour of the Deployment Tool

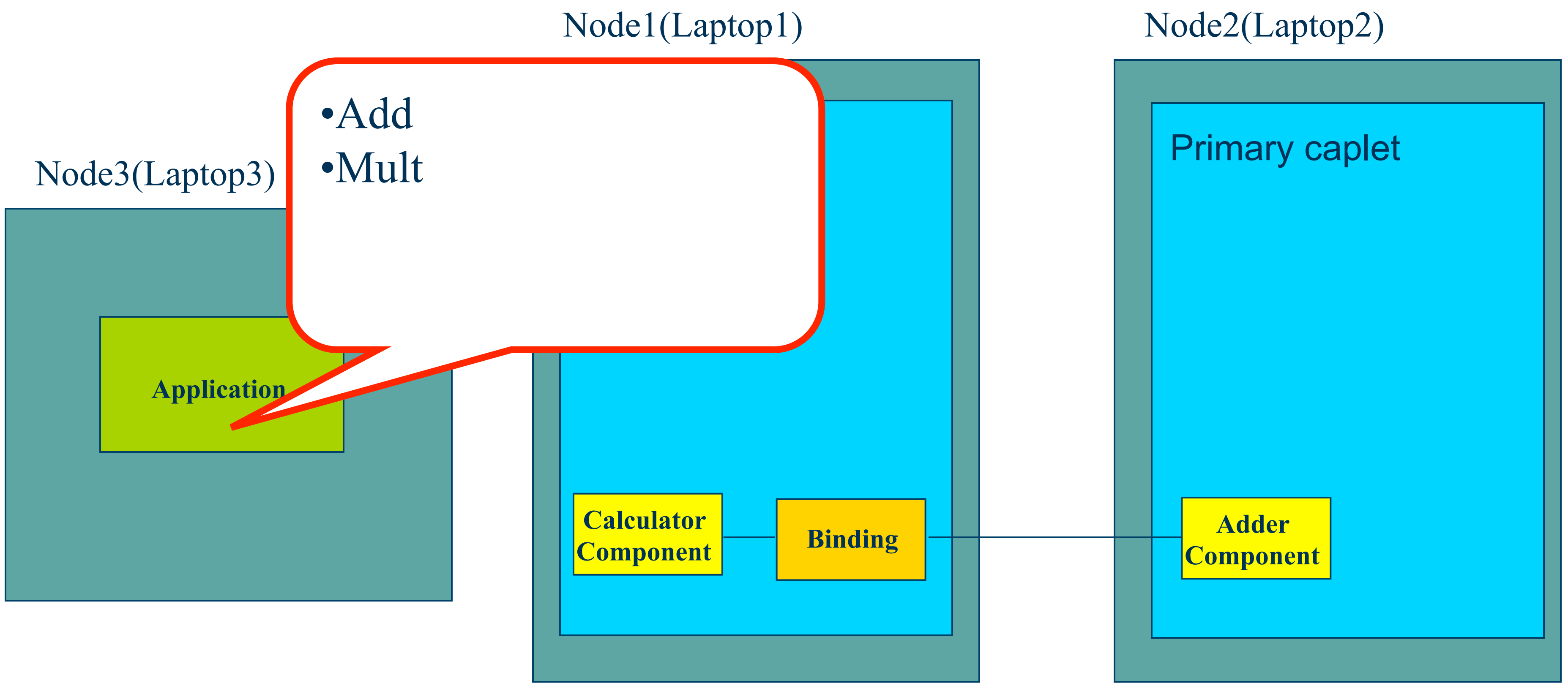


Behaviour of the Deployment Tool

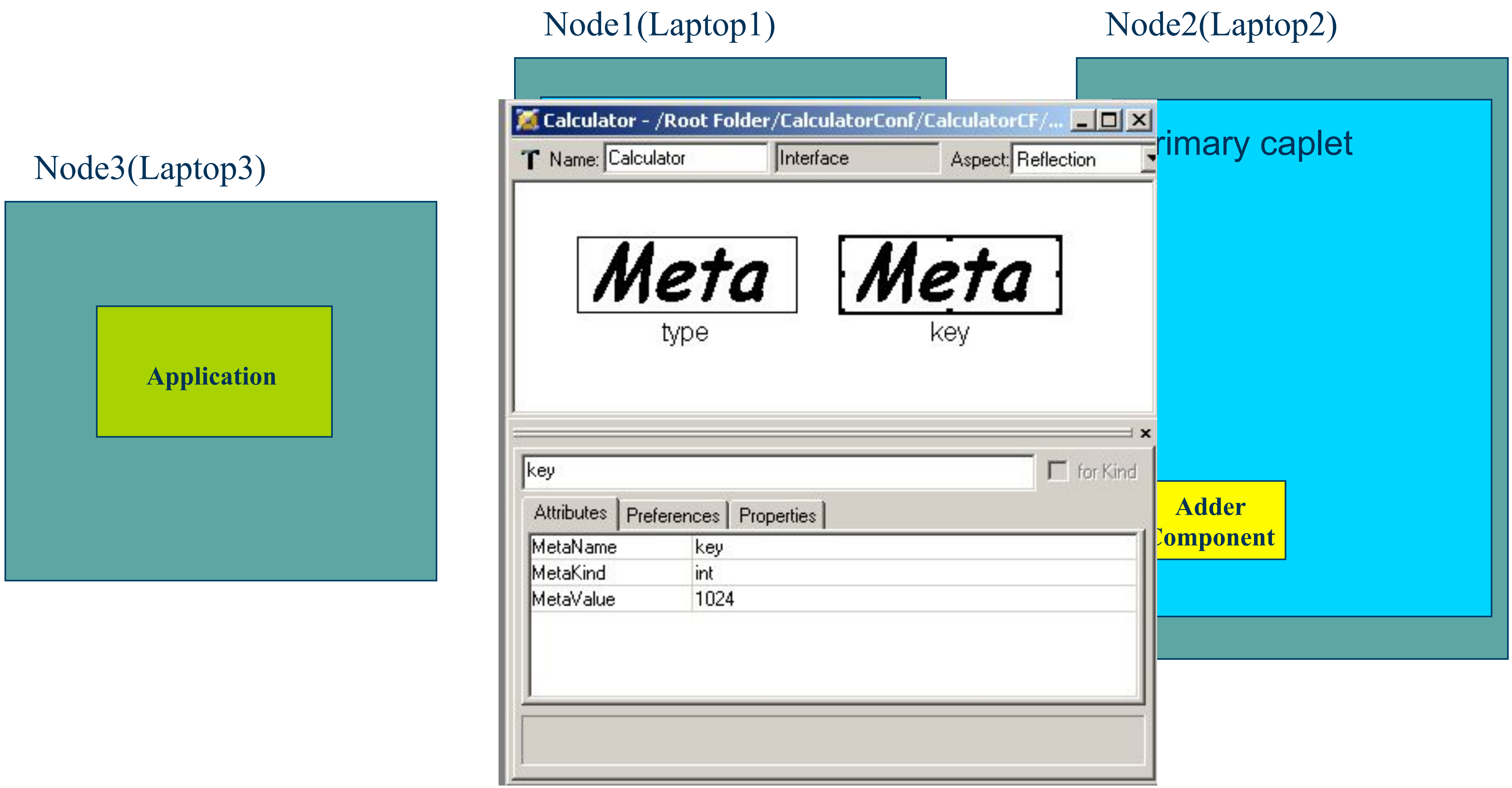


Demo

Initialization state

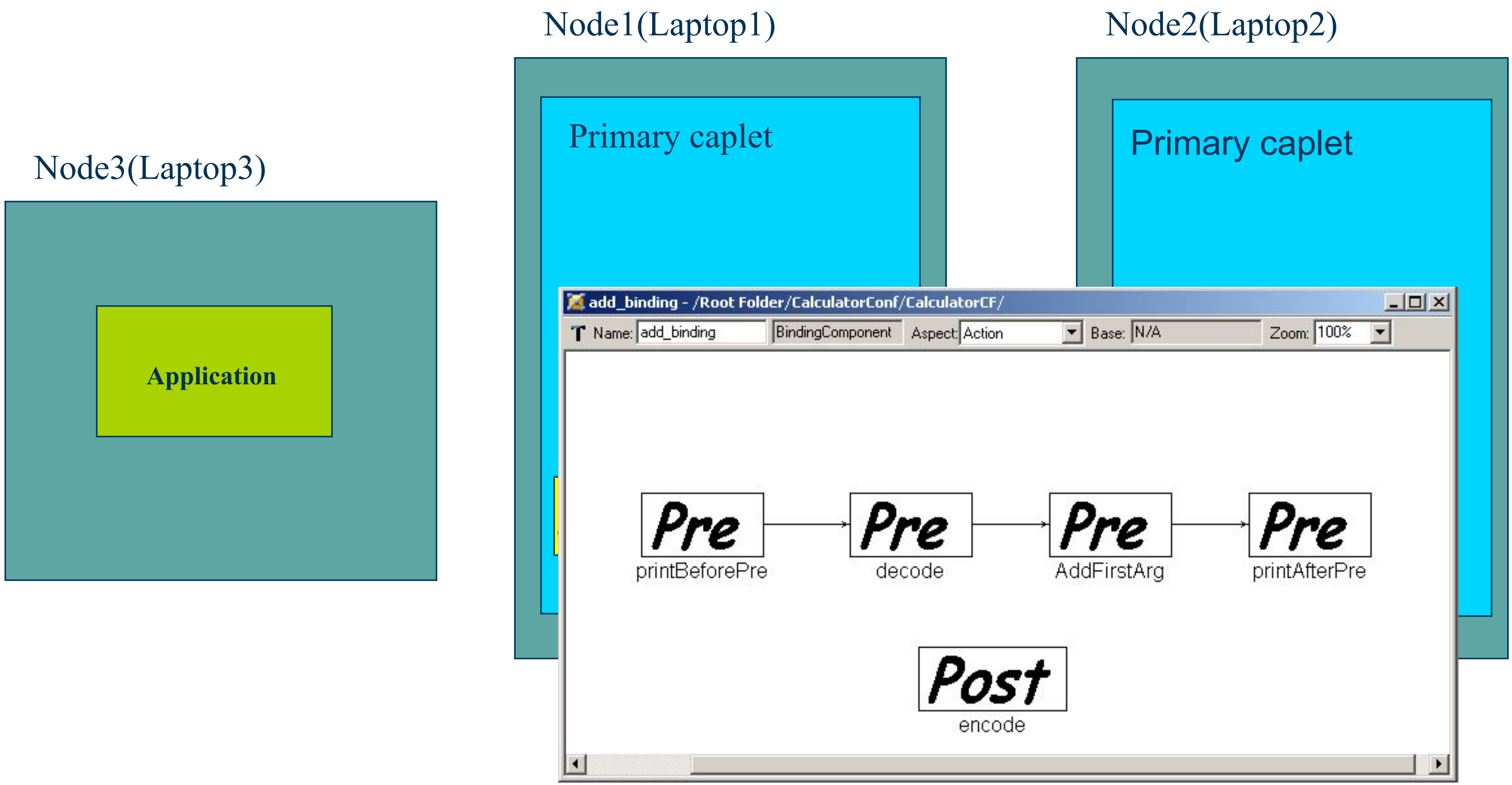


Meta Data

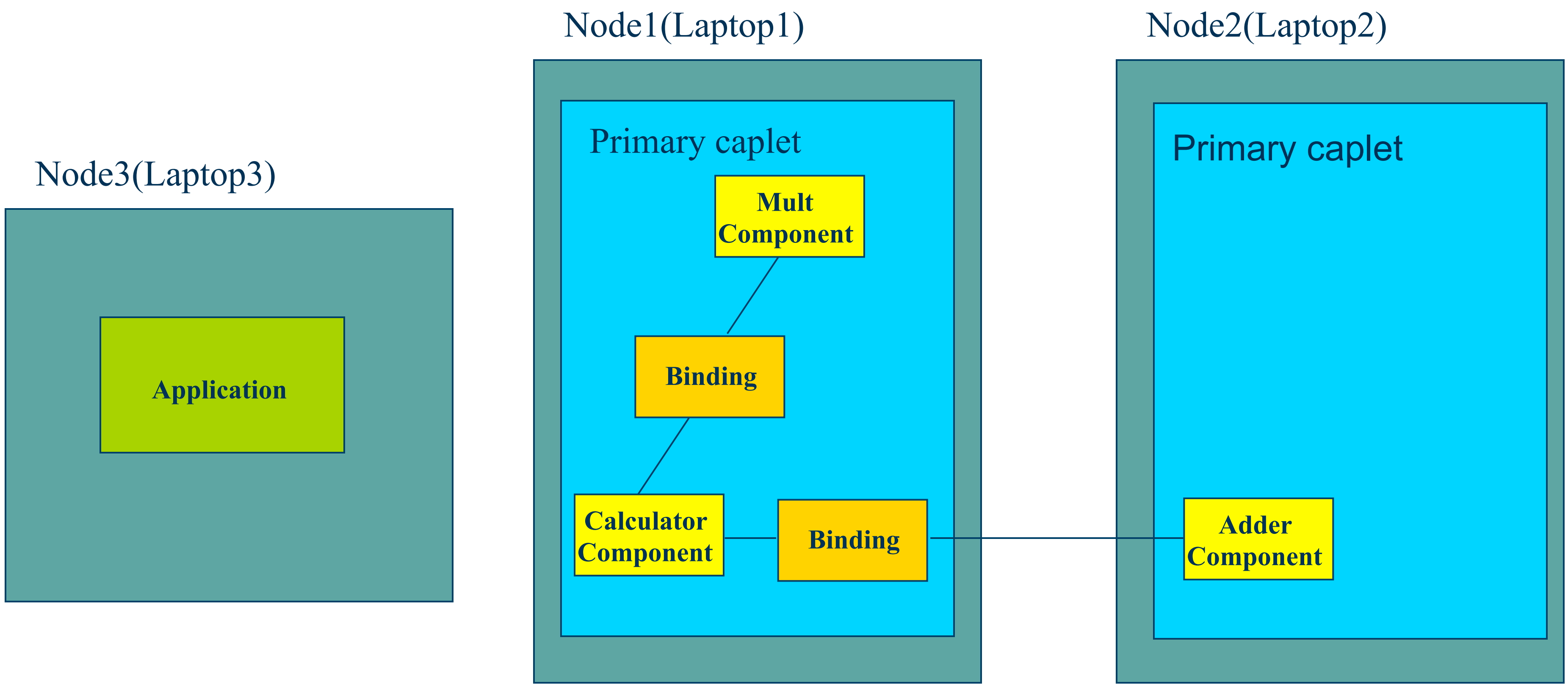


Node1(Laptop1)
Node2(Laptop2)
Node3(Laptop3)
Application
Calculator - /Root Folder/CalculatorConf/CalculatorCF/...
Name: Calculator
Interface
Aspect: Reflection
Meta
type
Meta
key
key
for Kind
Attributes
Preferences
Properties
MetaName key
MetaKind int
MetaValue 1024
rimary caplet
Adder
Component

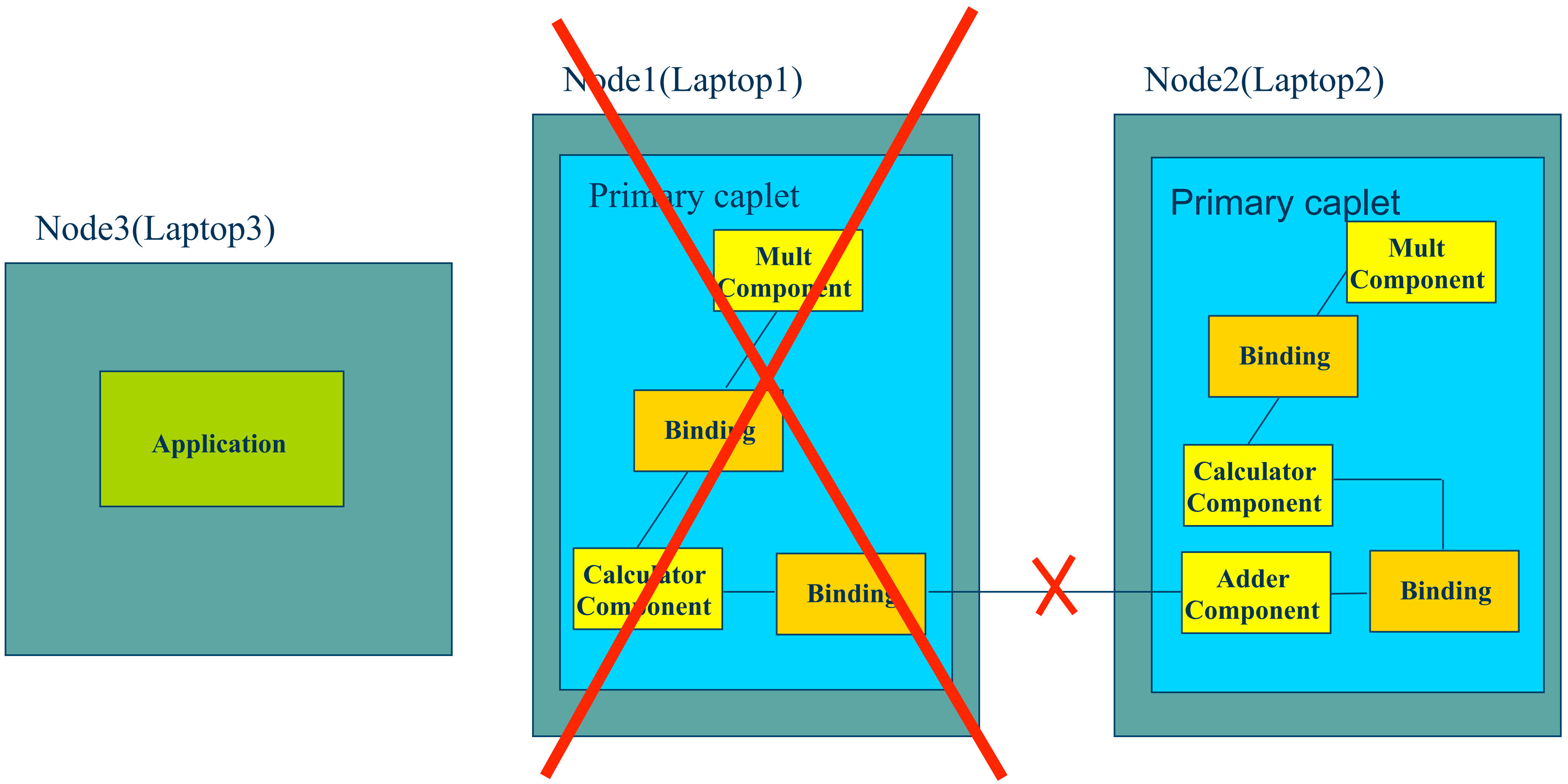
Dynamic Interception



Creating new components



Reconfiguration



Questions ?

ERICSSON

TAKING YOU FORWARD