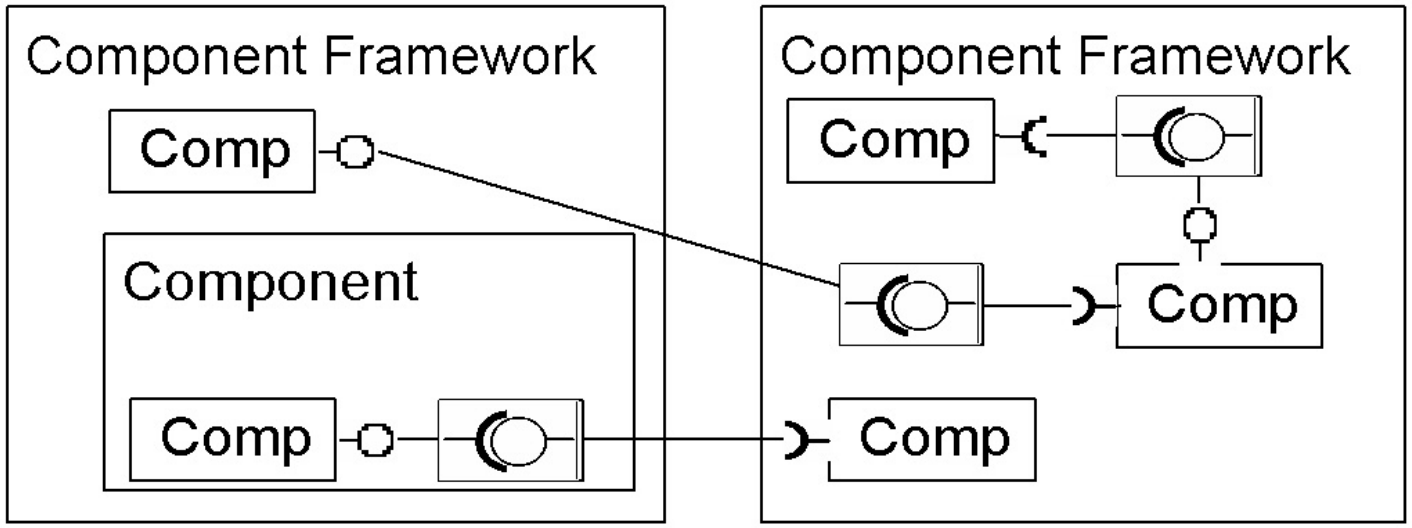
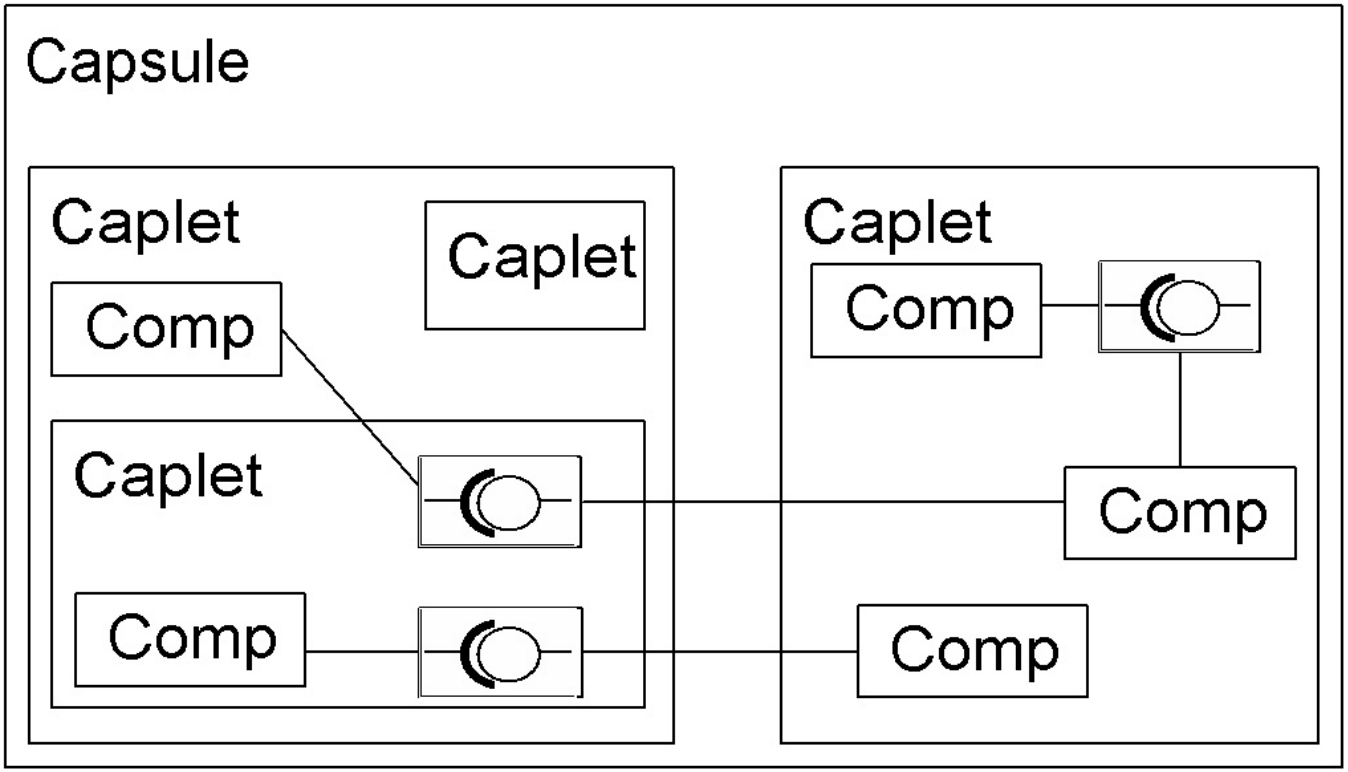
Component Model

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



Robust Component Model

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



Basic concepts

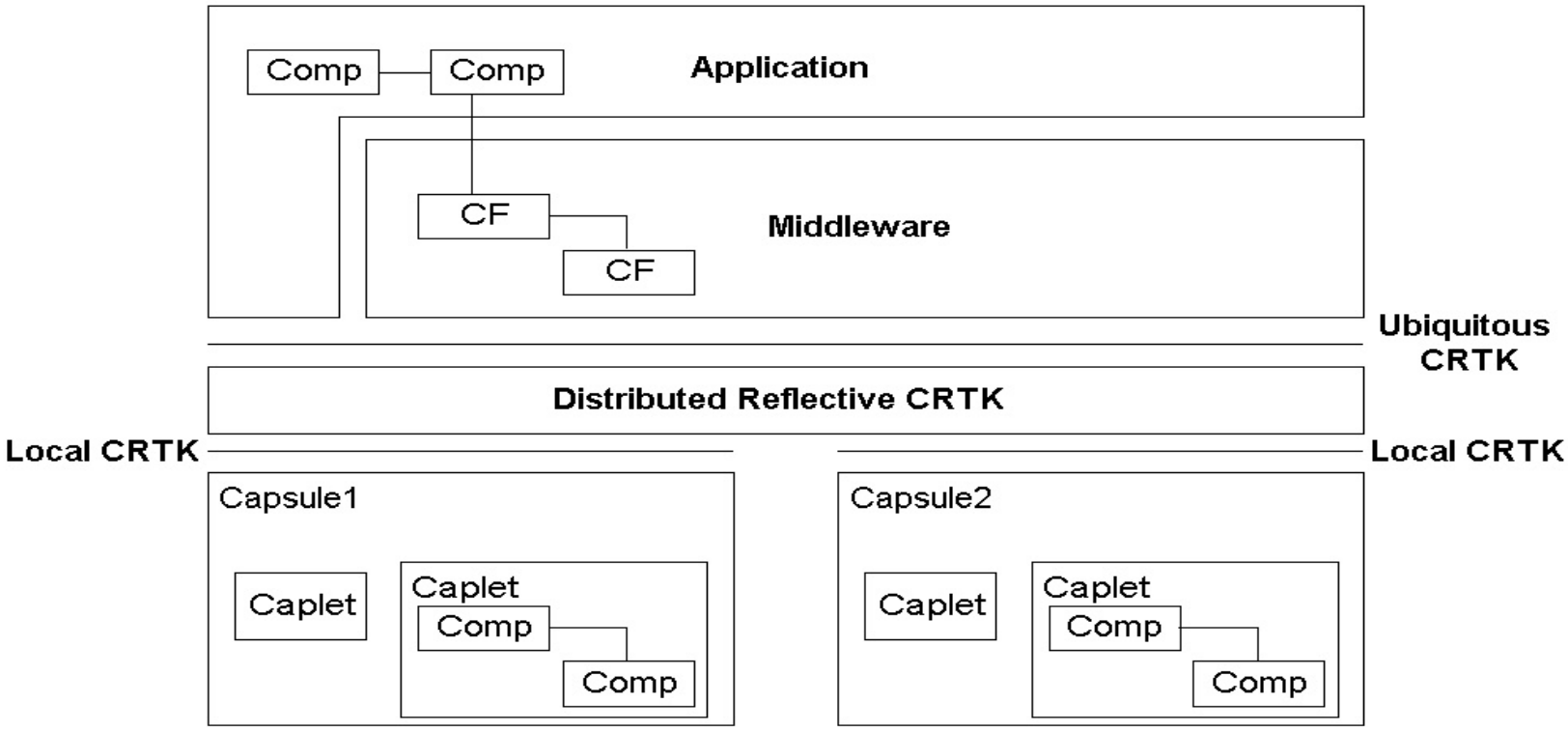
Concept	Erlang	ErlCOM
Concurrent entity	Process	Component
Communication	Message Passing*	Binding**
State	Local state information	Reflective Repository

*No reasoning about messages

**Interception consists of pre/post actions → synchronous communication

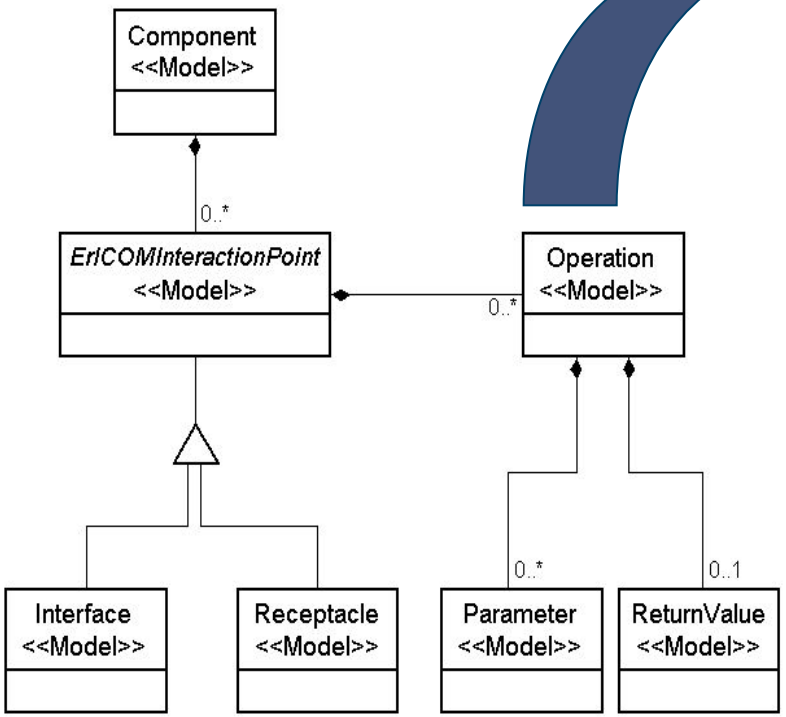
RUNES Example

- Gateways represented by Capsules (running Erlang)
- Middleware and Application Component Frameworks using the services of the Ubiquitous CRTK



Implementation details(1)

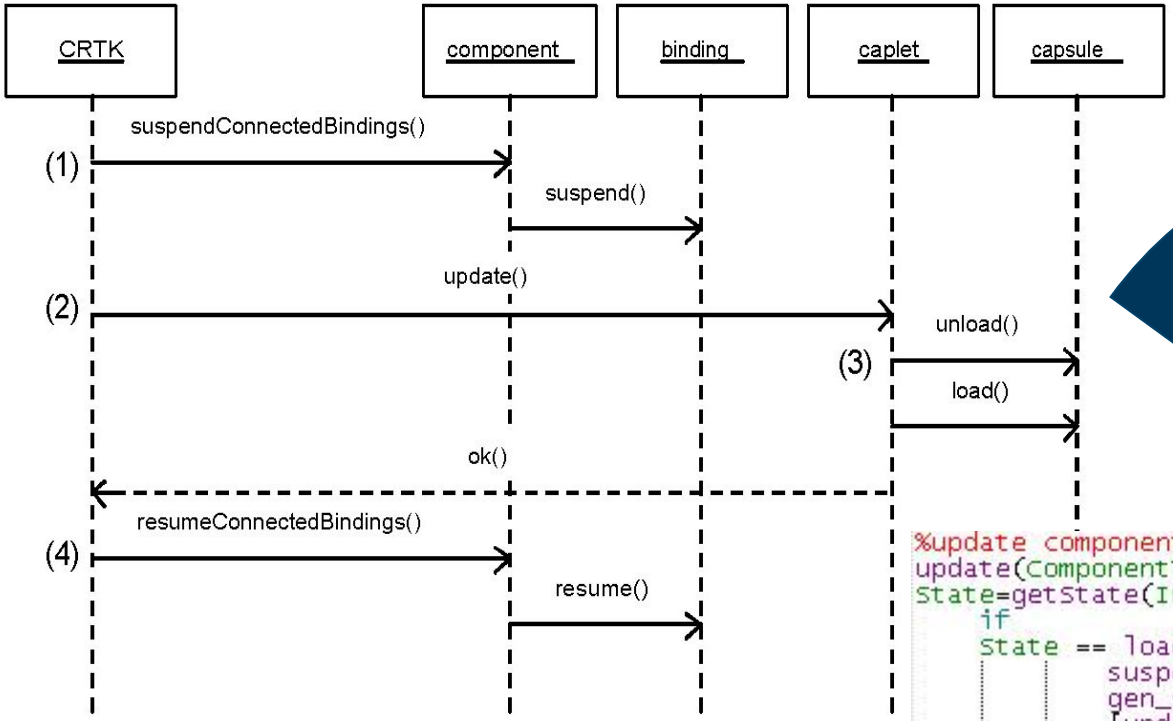
- Entity Relationship Diagram (ERD) + generated Erlang Code




```
load(InstanceName)->
    V_type={type,text,[type]},
    MetaDataList=[V_type],
    meta:deleteallprop(InstanceName),
    [meta:putprop(InstanceName,MetaDataType,MetaDataName,MetaDataValue)||{Me
    %initialize the interfaces and load the meta data of the interfaces
     pletName = meta:getCaplet(InstanceName),
     nterface = list_to_atom(atom_to_list(InstanceName)++" Interface"),
     server:start_link({global,I_Interface},e_ComponentInterface,[Instanc
     component(I_Interface,interface,InstanceName,CapletName),
     onentInterface:load(I_Interface),
     tialize the receptacles and load the meta data of the receptacles
    CapletName = meta:getCaplet(InstanceName),
    R_Receptacle = list_to_atom(atom_to_list(InstanceName)++" Receptacle"),
    gen_server:start_link({global,R_Receptacle},e_ComponentReceptacle,[Instal
    insert_component(R_Receptacle,receptacle,InstanceName,CapletName),
    e_ComponentReceptacle:load(R_Receptacle).

unload(InstanceName)->
    meta:deleteallprop(InstanceName),
    %destruct the interfaces and delete the meta data of the interfaces
    I_Interface = list_to_atom(atom_to_list(InstanceName)++"Interface"),
    e_ComponentInterface:unload(I_Interface),
    delete_component(I_Interface),
    gen_server:cast({global,I_Interface},stop),
    global:unregister_name(I_Interface),
    R_Receptacle = list_to_atom(atom_to_list(InstanceName)++"Receptacle"),
    e_ComponentReceptacle:unload(R_Receptacle),
    delete_component(R_Receptacle),
    gen_server:cast({global,R_Receptacle},stop),
    global:unregister_name(R_Receptacle).
```

Implementation details(2)

- Message Sequence Chart (MSC) + Erlang Code (CRTK)



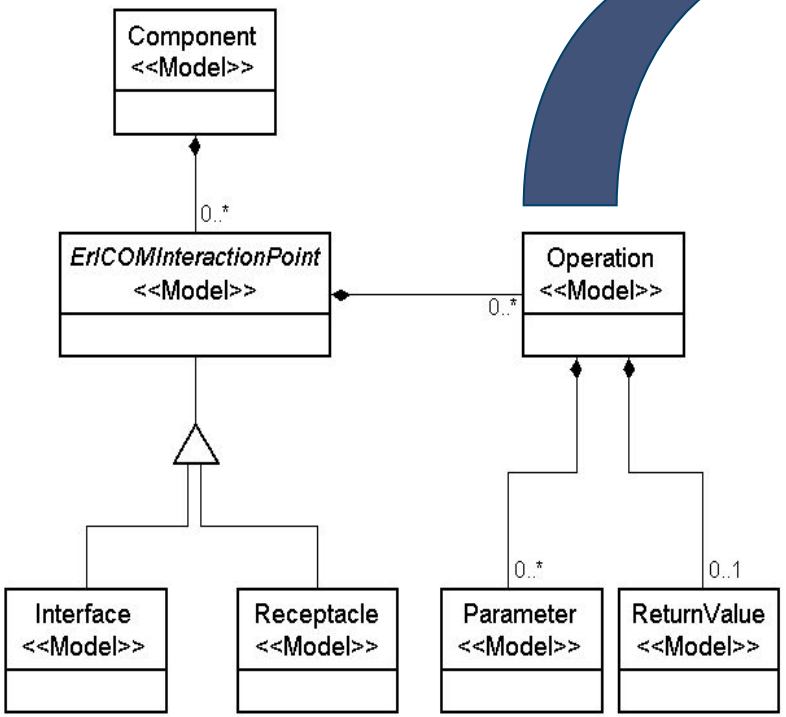

```
%update component in CRTK
update(ComponentType,InstanceName)->
State=getState(InstanceName),
    if
    State == loaded->
        suspendConnectedBindings(InstanceName),
        gen_server:call({global,CapletName},
        {update,ComponentType,InstanceName}),
        resumeConnectedBindings(InstanceName);
    State == true->
        io:format("~w should be loaded before updat
end.

%update component in caplet:
update(ComponentType,InstanceName)->
    LoaderName = getLoader(InstanceName),
    unload_component(InstanceName),
    load_component(LoaderName,ComponentType,InstanceName).
```

How to manage complexity?

- Custom libraries (API design)
  - Code maintenance problem (version handling)
- Custom behavior (language design and virtual machine)
  - Too intrusive , too costly
- Generative Programming (meta-modeling, modeling and transformation design)
  - New design approach → Domain Specific Language engineering and supporting IDE needed

Implementation details(1)

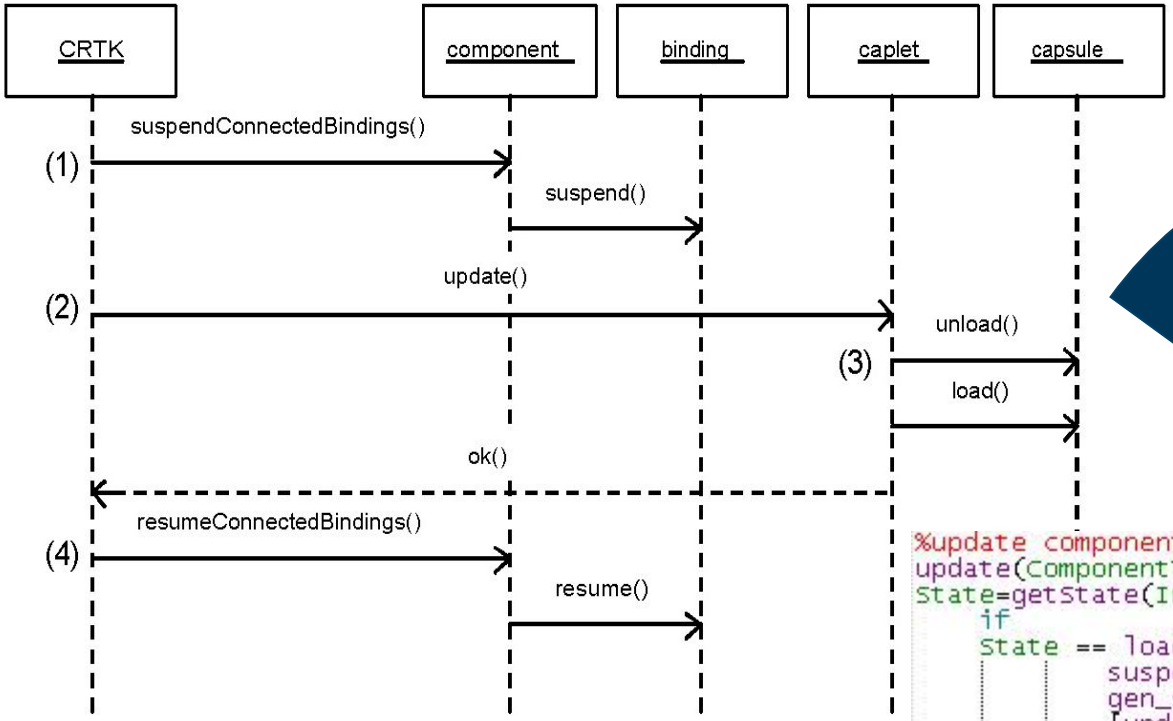
- Entity Relationship Diagram (ERD) + generated Erlang Code




```
load(InstanceName)->
    V_type={type,text,[type]},
    MetaDataList=[V_type],
    meta:deleteallprop(InstanceName),
    [meta:putprop(InstanceName,MetaDataType,MetaDataName,MetaDataValue)||{Me
    %initialize the interfaces and load the meta data of the interfaces
     pletName = meta:getCaplet(InstanceName),
     nterface = list_to_atom(atom_to_list(InstanceName)++" Interface"),
     server:start_link({global,I_Interface},e_ComponentInterface,[Instanc
     component(I_Interface,interface,InstanceName,CapletName),
     onentInterface:load(I_Interface),
     tialize the receptacles and load the meta data of the receptacles
    CapletName = meta:getCaplet(InstanceName),
    R_Receptacle = list_to_atom(atom_to_list(InstanceName)++" Receptacle"),
    gen_server:start_link({global,R_Receptacle},e_ComponentReceptacle,[Instal
    insert_component(R_Receptacle,receptacle,InstanceName,CapletName),
    e_ComponentReceptacle:load(R_Receptacle).

unload(InstanceName)->
    meta:deleteallprop(InstanceName),
    %destruct the interfaces and delete the meta data of the interfaces
    I_Interface = list_to_atom(atom_to_list(InstanceName)++"Interface"),
    e_ComponentInterface:unload(I_Interface),
    delete_component(I_Interface),
    gen_server:cast({global,I_Interface},stop),
    global:unregister_name(I_Interface),
    R_Receptacle = list_to_atom(atom_to_list(InstanceName)++"Receptacle"),
    e_ComponentReceptacle:unload(R_Receptacle),
    delete_component(R_Receptacle),
    gen_server:cast({global,R_Receptacle},stop),
    global:unregister_name(R_Receptacle).
```

Implementation details(2)

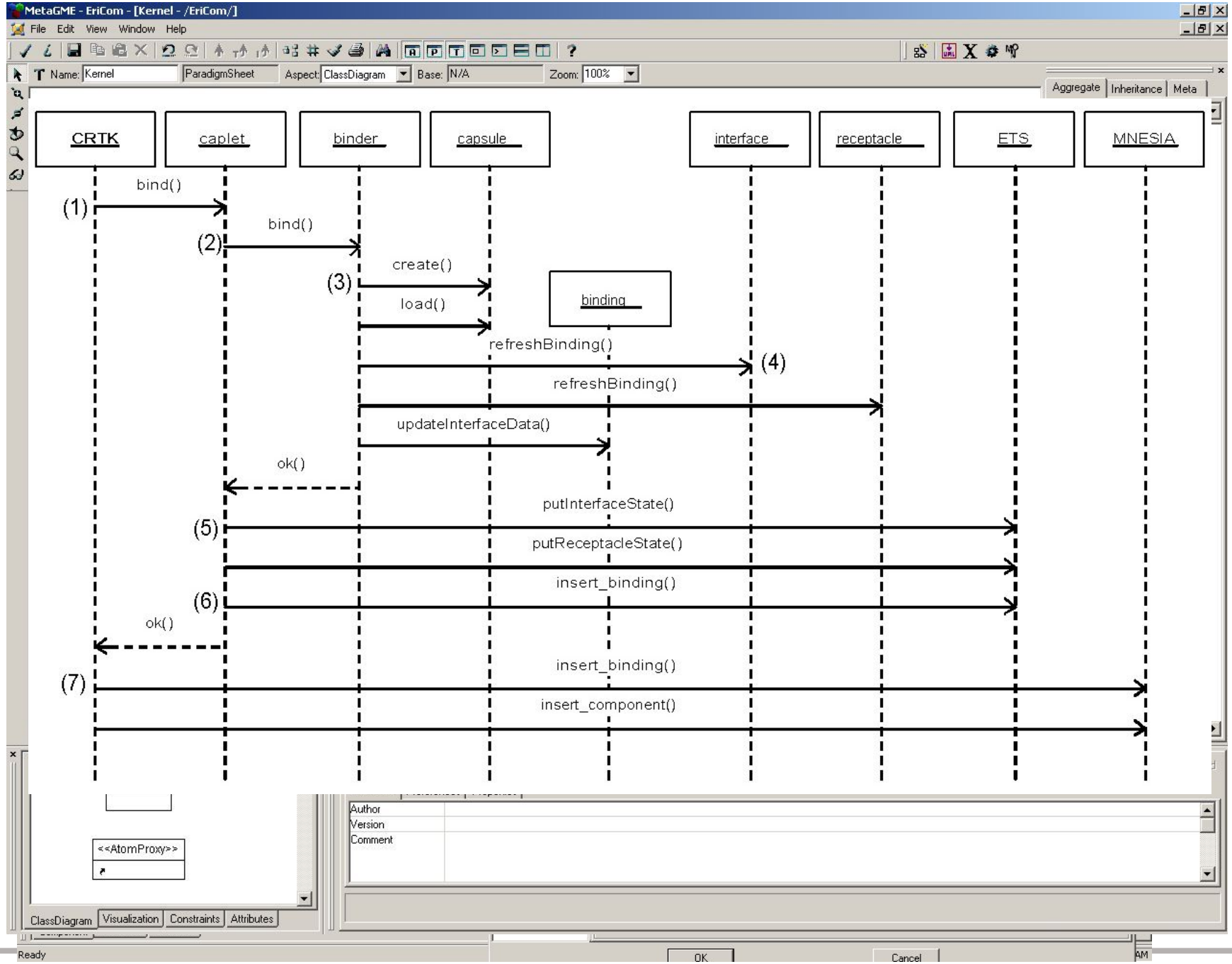
- Message Sequence Chart (MSC) + Erlang Code (CRTK)




```
%update component in CRTK
update(ComponentType,InstanceName)->
State=getState(InstanceName),
    if
    State == loaded->
            suspendConnectedBindings(InstanceName),
            gen_server:call({global,CapletName},
            {update,ComponentType,InstanceName}),
            resumeConnectedBindings(InstanceName);
        State == true->
            io:format("~w should be loaded before updat
end.

%update component in caplet:
update(ComponentType,InstanceName)->
    LoaderName = getLoader(InstanceName),
    unload_component(InstanceName),
    load_component(LoaderName,ComponentType,InstanceName).
```

What we have learnt?

- Entity Relationship Diagram helps identify concepts
- MSC helps describe dynamics
- Erlang provides versatile platform for component based systems
- Current implementation provides a feasible proof of concept for RUNES CRTK.
- BUT, the implementation code is COMPLEX


MetaGME - EriCom - [Kernel - /EriCom/]
File Edit View Window Help
Name: Kernel
ParadigmSheet
Aspect: ClassDiagram
Base: N/A
Zoom: 100%
Aggregate
Inheritance
Meta
CRTK
caplet
binder
capsule
interface
receptacle
ETS
MNESIA
bind()
(1)
bind()
(2)
create()
(3)
load()
binding
refreshBinding()
(4)
refreshBinding()
updateInterfaceData()
ok()
putInterfaceState()
(5)
putReceptacleState()
insert_binding()
(6)
ok()
insert_binding()
(7)
insert_component()
<<AtomProxy>>
Author
Version
Comment
ClassDiagram
Visualization
Constraints
Attributes
Ready
OK
Cancel