# The COSA Framework

## A Cognitive System Architecture with its implementation based on a CORBA-wrapped SOAR process

*Henrik J. Putzer*

*Institut für Systemdynamik and Flugmechanik*
*Universität der Bundeswehr München - Germany*

# Who are we ?

# Who are we ?

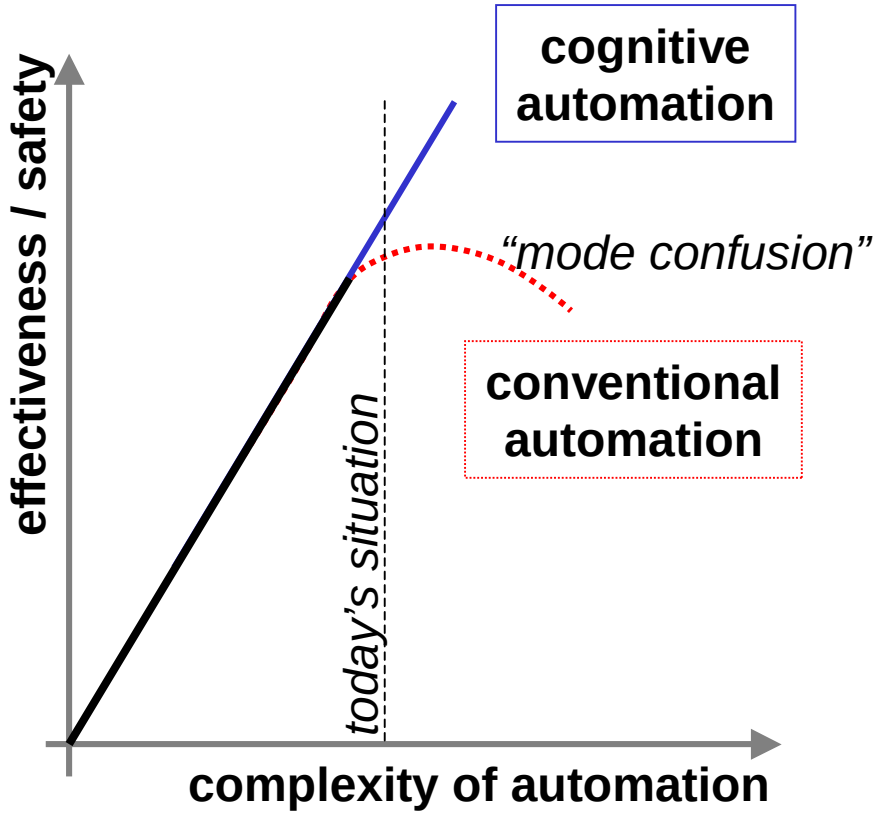
- **Institut für Systemdynamik und Flugmechanik Universität der Bundeswehr München, Germany**
- **Research objectives**
  - *flight guidance and control*
  - *“Human Engineering”, not Psychology*
  - *top down (architecture), not bottom up (sensors)*
  - *cognitive systems (assistants, tutors, UAV, etc.)*
  - *architecture with target system in mind*
- **first contact with SOAR one year ago**
  - *while searching for knowledge processors via the web*
  - *but not much experience so far (focus on architecture)*

# What are we doing ?

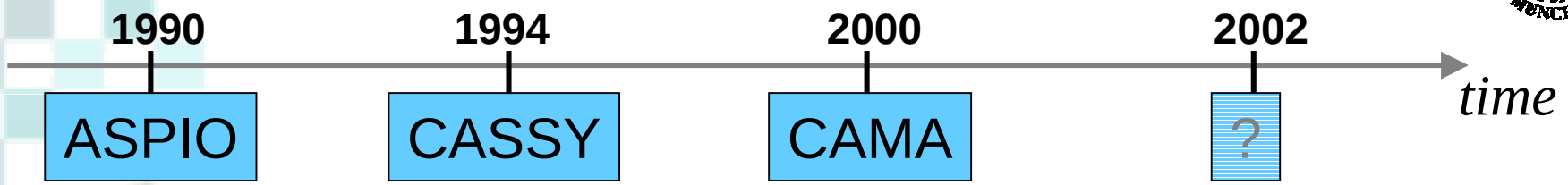
# Motivation

- **Increasing:**
  - *system complexity*
  - *automated functions*
  - *complexity of situation*
  - *complexity of mission*
  - *complex planning and decisions*

- **but:**
  ***constant crew resources***

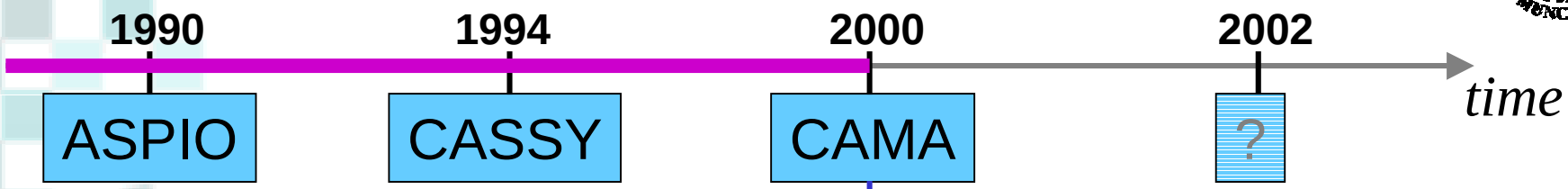


# System Evolution




- **research yielded operational systems**
- **systems improved over time in ...**
  - *software development*
  - *architecture*
  - *functionality*

# System Evolution - CAMA




## *Crew Assistant Military Aircraft*

- **functional extension of CASSY**
  (for military transport missions)
  - *modular architecture*
  - *central situation representation*
  - *based on CASSY, coded in C and C++*
- **successfully flight tested in 2000**
- **great acceptance by pilots**

*... but:*
*grown over years and now hard to maintain or extend.*

# Analysis

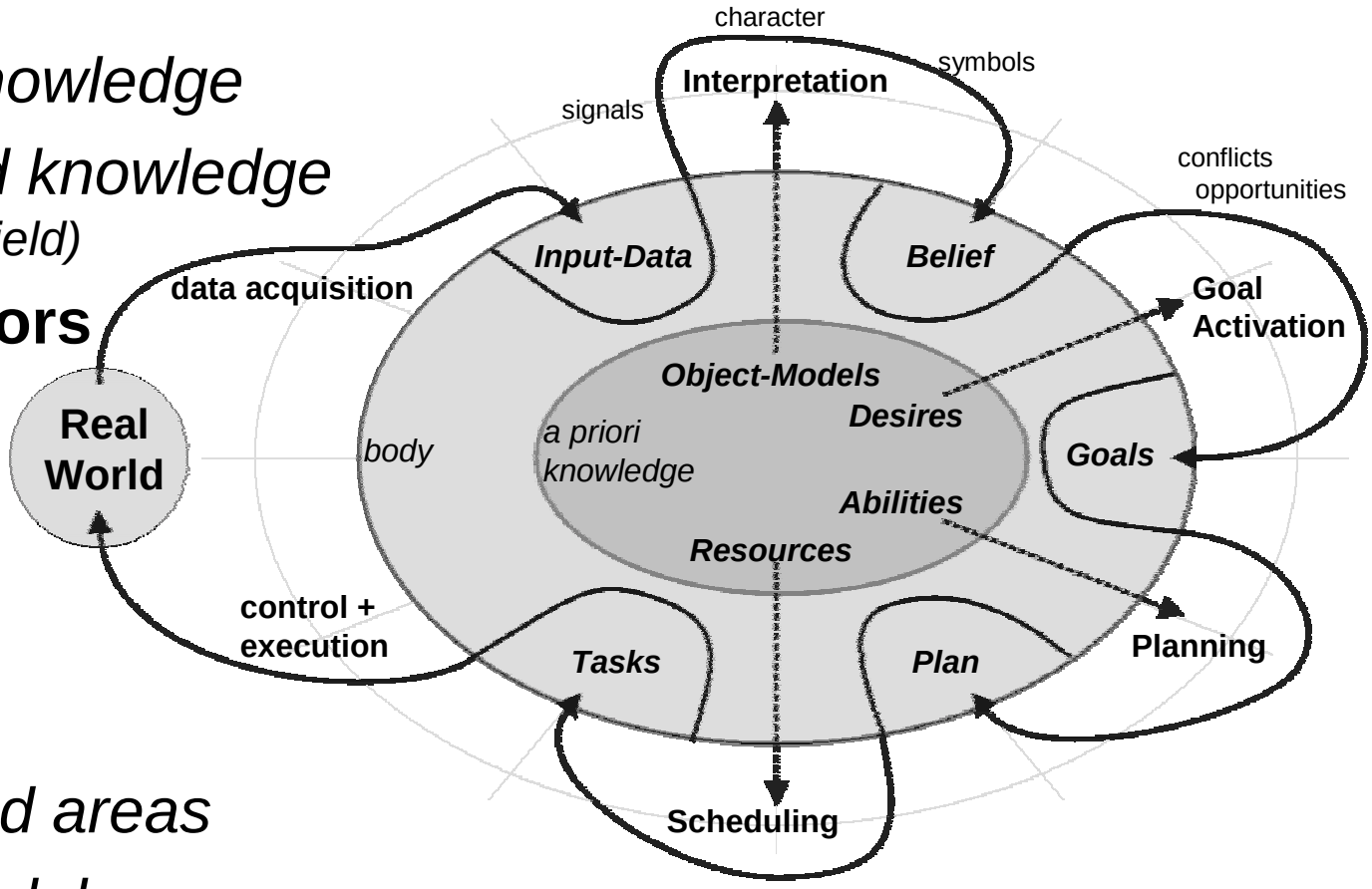
- **functional view**
  - *cognitive system*
  - *cooperative system*
  - *symbolic knowledge processing*
  - *simulating human behavior*
    *(system's behavior is understandable)*

***the Cognitive Process***

- **architectural view**
  - *distributed system and knowledge*
  - *separate architecture from application*
  - *maintainability, extendibility*
  - *uniform representation*
  - *knowledge processor*

# Substructure for all Cognitive Processes

- **body** (data)**:**
  - *a-priori knowledge*
  - *generated knowledge (=cognitive yield)*
- **transformators** (functions)**:**
  - *reading in whole body*
  - *writing in designated areas*
  - *use of models*



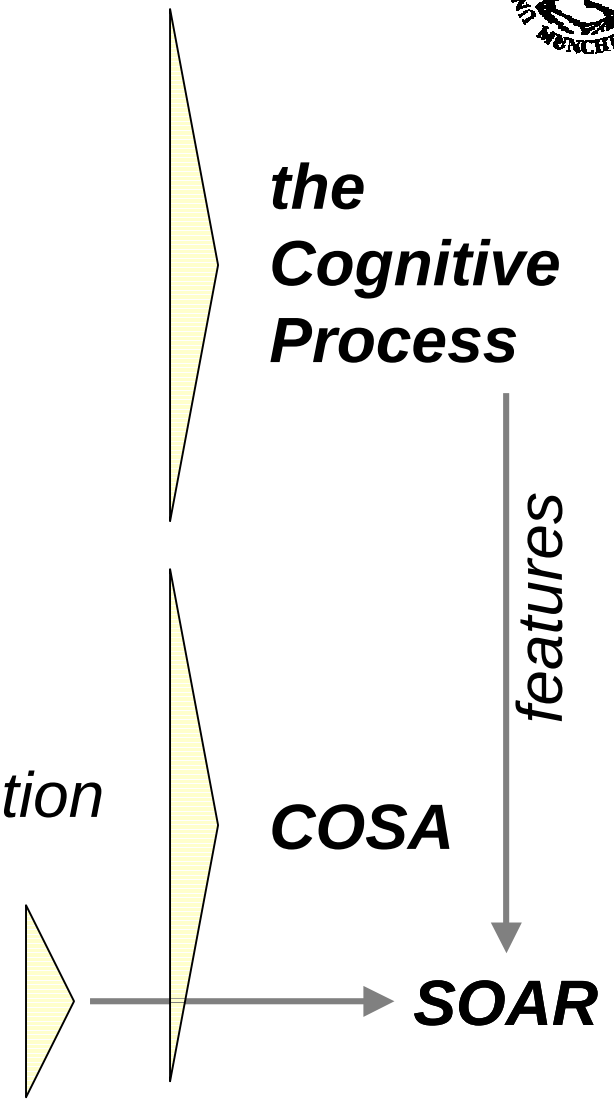

- **cognitive process consists of 4 transformators (+ I/O)**

# Decomposition and OO approach

- **the Cognitive Process is the fix architecture**
  - *target systems are established solely by 'communicating' a priori knowledge into the body*
  - *this knowledge as the uniform structure of models*
- **object models from an image of the real world**
  - *templates have functions describing the behavior of each instance (including creation and deletion)*
  - *instances have data members describing the state*
- **aggregation**
  - *the combination of all micro behaviors of all objects within the body of the cognitive process form the macro behavior of the whole system.*

# Analysis

- **functional view**
  - *cognitive system*
  - *cooperative system*
  - *symbolic knowledge processing*
  - *simulating human behavior*
- **architectural view**
  - *distributed system and knowledge*
  - *separate architecture from application*
  - *maintainability, extendibility*
  - *uniform representation*
  - *knowledge processor*



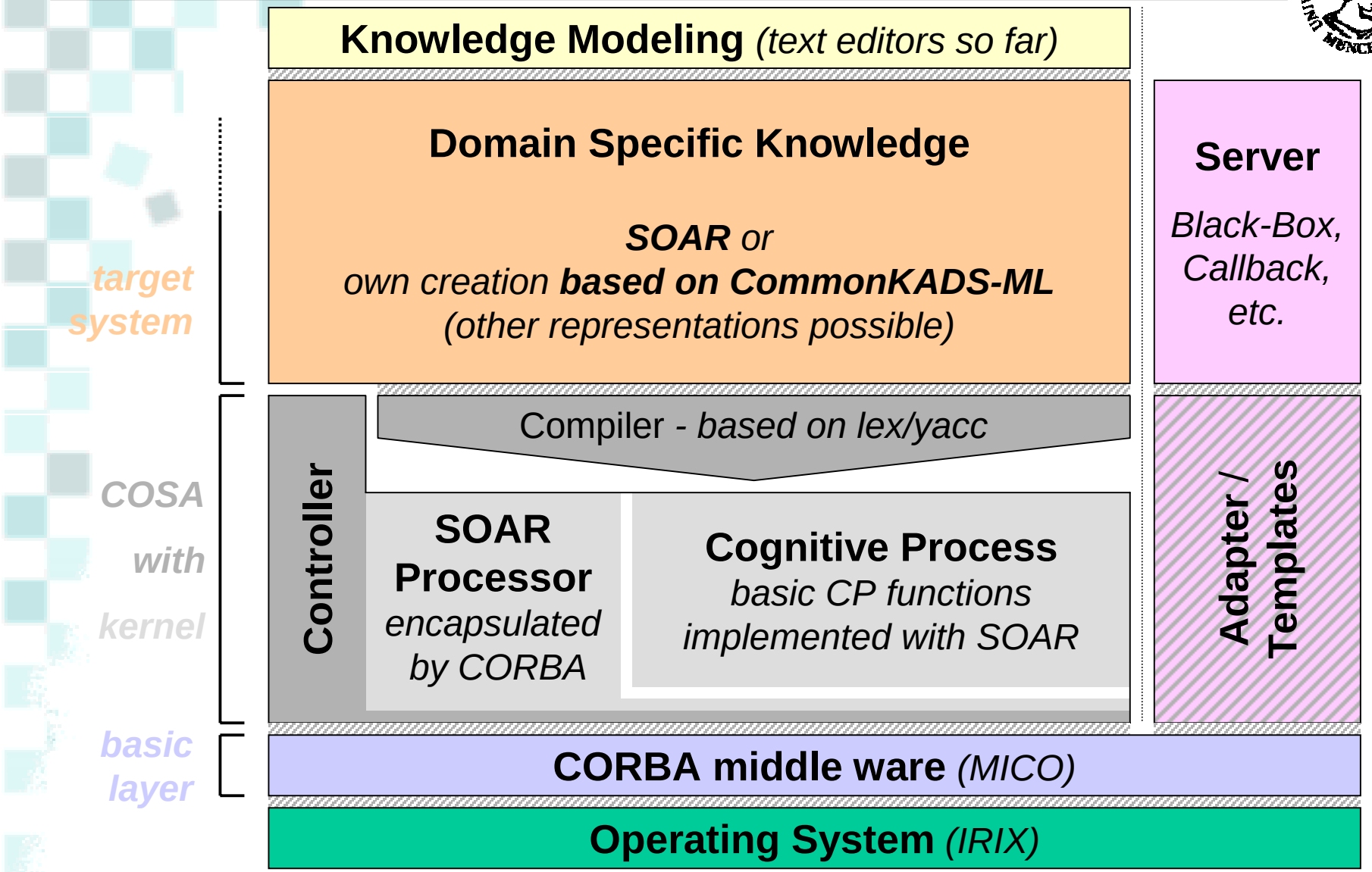
# What is COSA?

# What is COSA ?

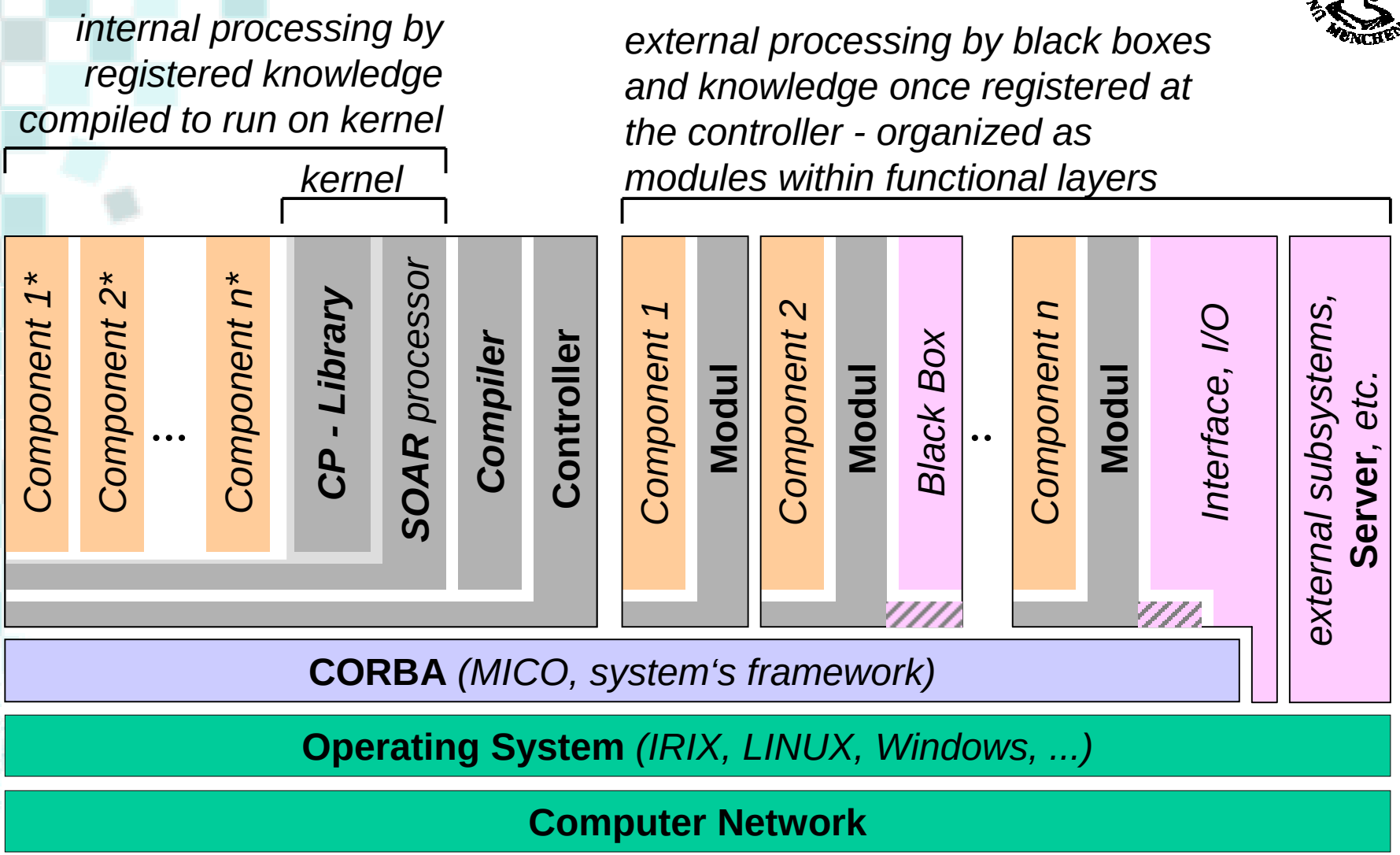
COSA

- **Kernel**
  - *Processor: SOAR*
    - » uniform data (WM)
    - » uniform algorithm / behavior (rules)
  - *Library: Cognitive Process*
    - » realizes the Cognitive Process
    - » object oriented abstraction in SOAR
    - » knows about components
- **CORBA encapsulation**
  - » distributed system / component handling (make use of kernel)
  - » knowledge abstraction (wrapping for distribution via CORBA)
  - » interfacing with other (external) systems (e.g. in the cockpit)
- **Language Front End**
  - *Compiler*
    - » input is knowledge, which is compiled to run on the kernel
    - » other knowledge descriptions (besides SOAR) are possible

# COSA: block model architecture



# COSA: Layer Model of Architecture



# What can COSA be used for?

- **goals**
  - *high level decisions / decision support*
  - *implement the Cognitive Process*
  - *complex symbolic processing*
  - *distributed system*
  - *separation of architecture and target system*
  - *flexible knowledge front end and reuse of knowledge*

- **not addressed** *(but can be done by extern. components)*
  - *high frequent control loops*
  - *number crunching*

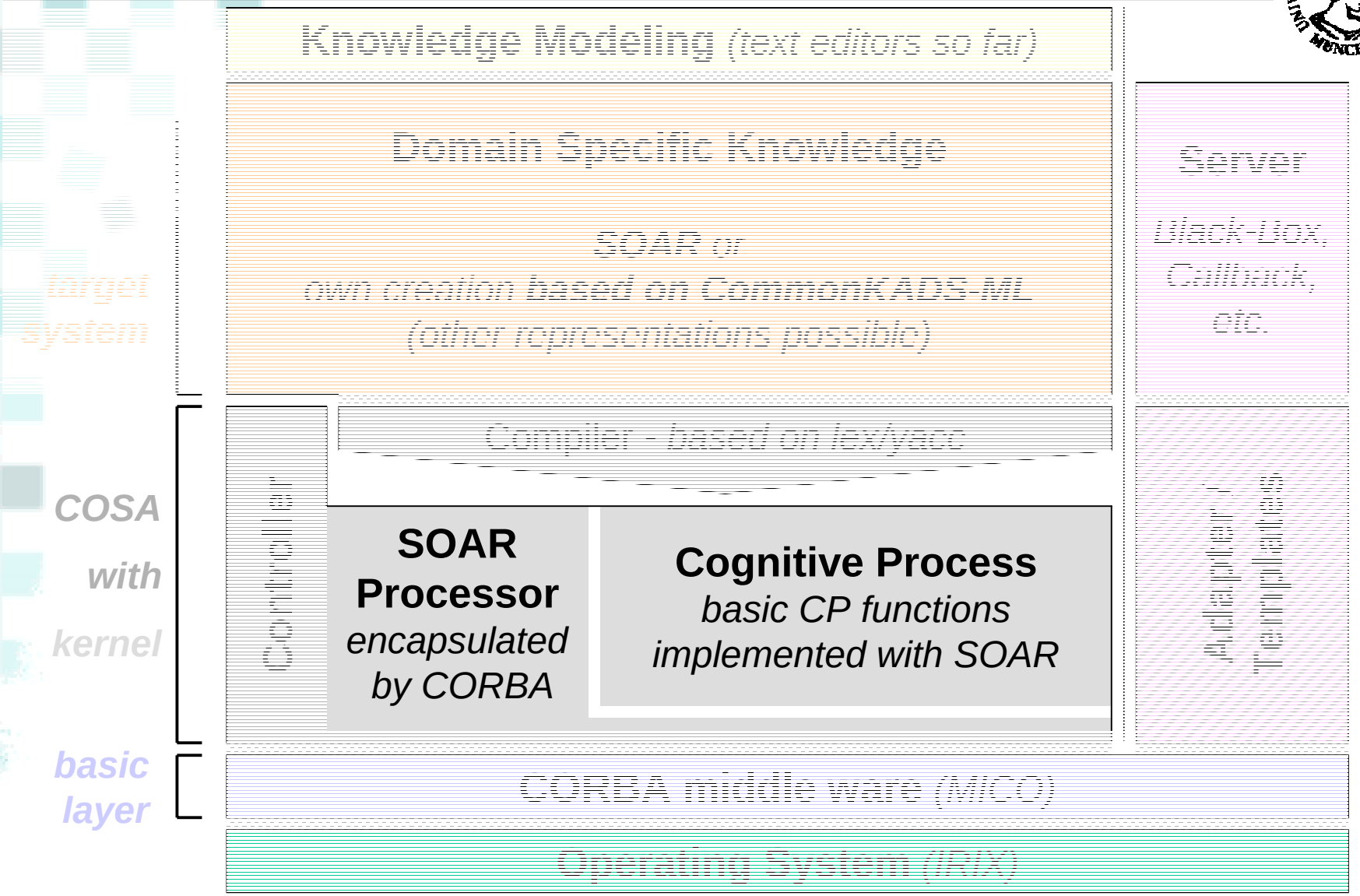
# How do we use SOAR ?

# COSA - Usage of SOAR

COSA

- **Kernel**
  - *Processor: SOAR*
    - » uniform data (WM)
    - » uniform algorithm / behavior (rules)
  - *Library: Cognitive Process*
    - » realizes the Cognitive Process
    - » object oriented abstraction in SOAR
    - » knows about components
- CORBA encapsulation
  - » distributed system / component handling (make use of kernel)
  - » knowledge abstraction (wrapping for distribution via CORBA)
  - » interfacing with other (external) systems (e.g. in the cockpit)
- Language Front End
  - Compiler
    - » input is knowledge, which is compiled to run on the kernel
    - » other knowledge descriptions (besides SOAR) are possible

# Locating SOAR with in COSA

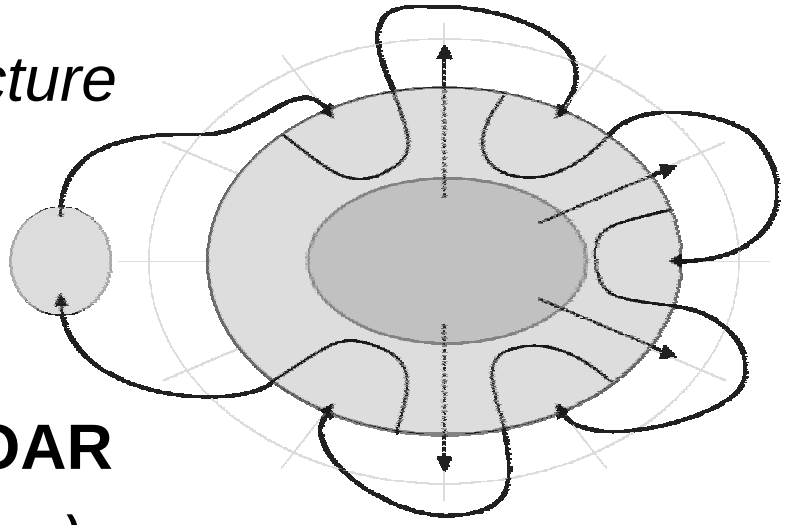


# Usage of SOAR within COSA

- **Kernel is formed by SOAR**
  - *SOAR is the processor*
  - *SOAR library implementing the Cognitive Process (CP-Library)*
- **Why SOAR ?**
  - *Uniform representation of knowledge: WM*
  - *Uniform representation of behavioral parts: productions*
  - *features and research in many areas we need*
    - learning
    - cooperation with other agents
    - using several levels of knowledge
    - **…** *(much more)*

# Features of the CP-Library

- **Cognitive Process is the top level SOAR state**
- **organization of WM**
  - *special area used by architecture*
    - components, signals, ...
  - *a-priori-knowledge*
  - *cognitive yield*
- **object oriented view within SOAR**
  - *classes (= models or templates)*
  - *instances*
  - *process of creation and deletion*
  - *behavior or instances*
  - *inheritance (data members (=attributes) only)*



# CP-Library - The ‘global’ tree

- **‘`global`’ is an augmentation of each state**
  - *automatically linked to every state at creation time*
- **the ‘`global`’ tree within the WM includes**
  - ‘**`trigger`**’ - *for internal synchronization (signals)*
    *(unused so far, but tests are running)*
  - ‘**`components`**’ - *to organize registered components*
    - component dependencies
    - monitor state
      *(activation, errors, ...)*
    - operator selection within SOAR
  - ‘**body**’
    - a-priori knowledge and cognitive yield

```
(S1 ^global G1)
  (G1 ^trigger T1 )
  (G1 ^components C1)
  (G1 ^body B1)
```

# CP-Library - Components

- **representation of**
  - *internal components*
    *(system or transformators)*
  - *external components*
    *(parts of target system)*
- **augmentations**
  - '**name**', '**type**'
  - '**used**' *components*
  - '**connect**' *- true if all used components are found*
- **architecture generates**
  - '**depend**' *- transitive hull of* '**used**'
  - '**active**' *- true if connected and all depend are active*
  - '**rang**' *- kind of comp. hierarchy for operator selection*

```
(S1 ^global.components C1)
 (C1 ^comp C2 C3 C4 ... )

  (C4 ^name |name|)
  (C4 ^type [sys,cpt,model])
  (C4 ^uses <comp>* )
  (C4 ^connect t)

  (C4 ^depend <comp*>)
  (C4 ^active t)
  (C4 ^rang [int])
```

# CP-Library - Body

- **'body' area ...**

  - *is part of the* '**global**' *structure*

  - *represents the data within the cognitive process*
    - a priori data = '**model**'
    - cognitive yield = '**instance**'

```
(S1 ^global.body B1)

 (B1 ^belief B4)
   (B4 ^model M1)
   (B4 ^instance I1)

 (B1 ^goal G4)
   (G4 ^model M1)
   (G4 ^instance I1)

 (B1 ^plan P4)
   (P4 ^model M1)
   (P4 ^instance I1)

 (B1 ^schedule S4)
   (S4 ^model M1)
   (S4 ^instance I1)
```

# CP-Library - Models

- **models are part of components**
- **models consist of**
  - *a general description*
    *This is the 'class' or the 'template' with all possible attributes, optional default values and information about inheritance.*
  - *productions for creation*
  - *productions for the behavior of instances*

```
(S1 ^global.body.belief B9)
 (B9 ^model M1 M2 M3 ... )
 (B9 ^instance I1 I2 I3 ... )

  (M1 ^name aircraft)
  (M1 ^attrib A1 A2 ... )
    (A1 ^name callsign)
    (A2 ^name alt ^default 0)
    ...
  (I1 ^name own-vehicle)
  (I1 ^model M1)
  (I1 ^callsign |D-ADAM| )
  (I1 ^alt 0)
  ...
```

- **architecture provides operators for instantiation and attribute consistency in case of inheritance**

# CP-Library - Overview

- **COSA**
  - *is a system architecture which uses SOAR*

- **SOAR is the kernel of COSA**
  - *SOAR processor*
    - all research of SOAR community (re-) usable
  - *SOAR library implementing the Cognitive Process*
    - organization and object oriented view by models

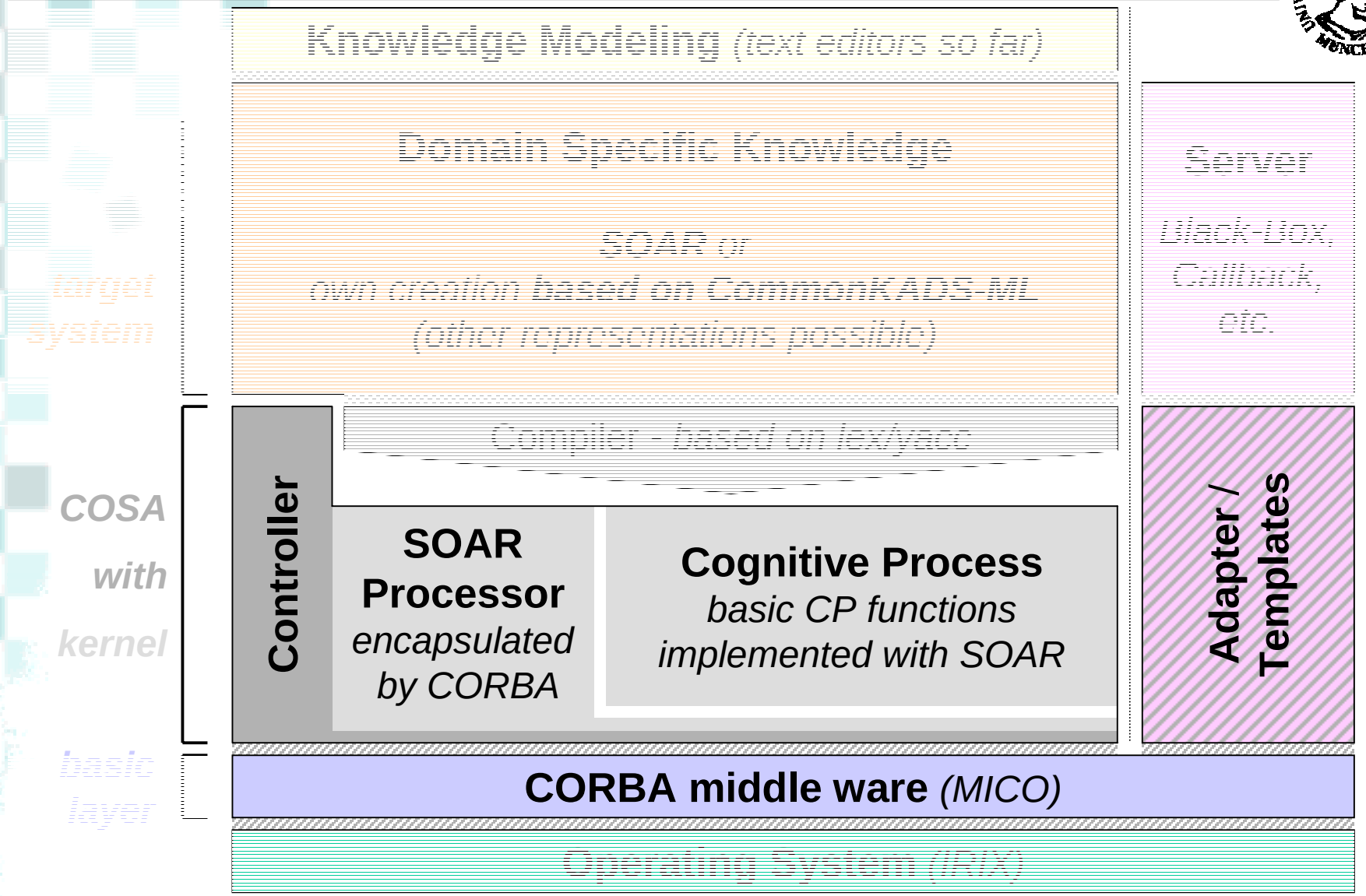

# What about the CORBA encapsulation ?

# COSA - Wrapping SOAR with CORBA

COSA

- Kernel
  - Processor SOAR
    - » uniform data (WM)
    - » uniform algorithm / behavior (rules)
  - Library Cognitive Process
    - » realizes the Cognitive Process
    - » object oriented abstraction in SOAR
    - » knows about components
- **CORBA encapsulation**
  - » distributed system / component handling (make use of kernel)
  - » knowledge abstraction (wrapping for distribution via CORBA)
  - » interfacing with external systems (e.g. FMS in the cockpit)
- Language Front End
  - Compiler
    - » input is knowledge, which is compiled to run on the kernel
    - » other knowledge descriptions (besides SOAR) are possible

# Wrapping with CORBA - COSA architecture



# Wrapping with CORBA - Why wrapping SOAR?

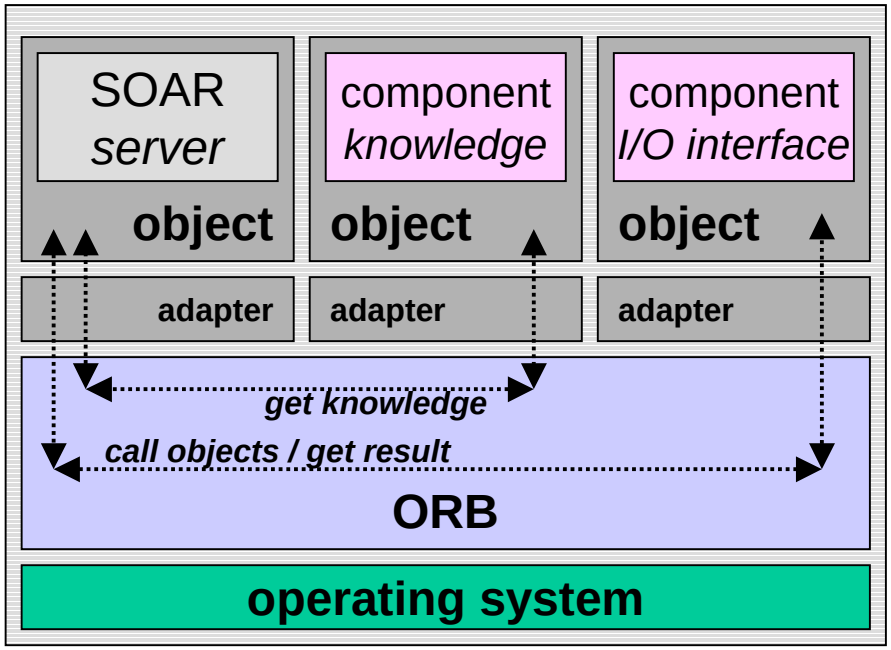

- **SOAR has ...**
  - *central situation representation (working memory)*
  - *efficient implementation of access (rules)*
  - *uniform representation of data (WMEs)*
  - *uniform representation of algorithms (productions)*

- **SOAR lacks ...**
  - *ability to be used in distributed environments*
  - *interface to handle components*

  - ***CORBA is good at these deficiencies***

# Wrapping with CORBA - What is CORBA?

- **Common Object Request Broker Architecture**
  - *industrial standard for distributed systems*
  - *middle ware to connect software components*
  - *client-server system*
  - *OO replacement for RPC*




- **features**
  - *independent of programming language*
  - *independent of operating system*
  - *independent of hardware (even network)*
  - *easy to use*

# Wrapping with CORBA

- **SOAR elements which need to be wrapped**

  - *knowledge*
    - build a suitable abstraction of the working memory
    - have interface to let CORBA objects communicate

  - *I/O functions, RHS functions*
    - call functions via the network
    - transfer and receive small portions of knowledge

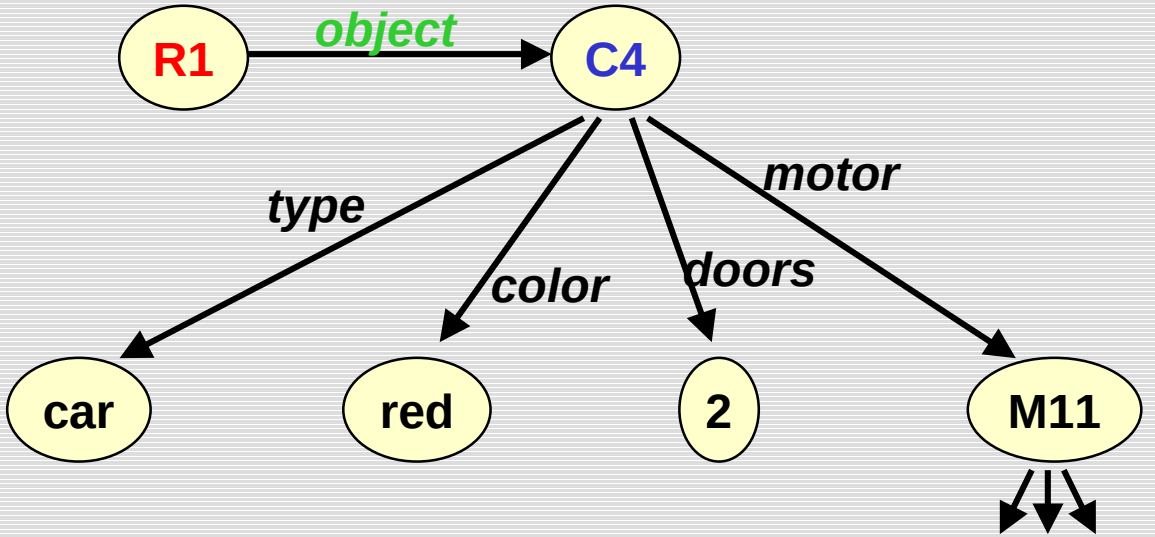
  - *callbacks*
    - no uniform interfacing to SOAR

# Wrapping Knowledge - Concept

- **knowledge abstraction**
  - *similar to conceptual graph (others are possible, e.g. frames)*

```
( R1 ^object C4 )

( C4 ^type  car )
( C4 ^color red )
( C4 ^doors 2 )
( C4 ^motor M11 )
  ( M11 ... )
```




- **interfaces possible**
  - ❶ *copy and reintegrate areas of the WM: easy to use*
  - ❷ *have 'pointer' access: highest flexibility*

# Wrapping Knowledge - Solved Problems

- ☑ **mapping SOAR types to CORBA**
  - *straight forward mapping of values (integer, string, ...)*
  - *special mapping for nodes which are not a value*
- ☑ **depth of copying WMEs**
  - *given by structure of models within COSA*
- ☑ **reintegration with links to other symbols in the WM**
  - *used special mapping and internal SOAR functions*
- ☒ **'pointer' access**
  - *not yet available, but experimented with it*
  - *consequences might not be intended !*
    *(location of knowledge)*

# Wrapping Functions - Concept


## Example: RHS function

❶ **COSA kernel**

- *convert parameters into knowledge graphs*
- *send knowledge to appropriate component*

❷ **component**

- *receive parameters*
- *calculate return result*

❸ **COSA kernel**

- *receive and unpack result*
- *reintegrate into WM*

*Note:*
*RHS-function '**sqt**' must be registered within the controller*

```
sp {test*production
 (state <s> ^operator <o>)
 (<o> ^name calc-sqt
 (<o> ^value <v>)
-->
 (<o> ^result (call sqt <v>))
}
```

# Wrapping Functions - integration into SOAR


- **registering dispatcher as SOAR callbacks**
  - » function: **link** between SOAR callbacks and object oriented world
  - » input: gets **target object's name** as parameters along with call
  - » action: **dispatches the call** to that object
- **using dispatcher for ...**

*↳special RHS functions*

  - » need to use RHS function '**`call`**'
  - » first parameter defines the target CORBA object
  - » following parameters define the parameters to the call

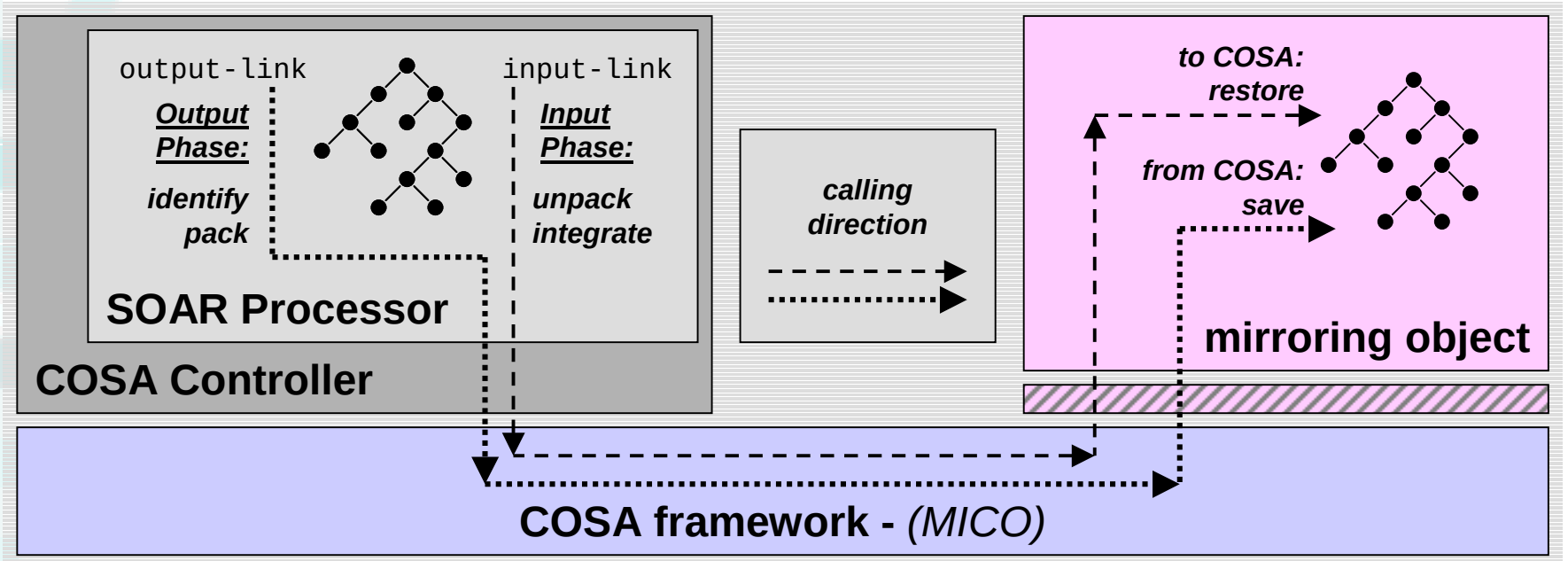
*↳special I/O-callbacks executed during I/O phase*

  - » target CORBA object is derived from structure of **`io-link`**
  - » parts of the **`output-link`** are transmitted while output phase
  - » incoming knowledge (input phase) is stored at the **`input-link`**

*↳callbacks (not implemented so far - not needed so far)*

# Wrapping with CORBA - Benchmark

- **setup of bench:**
  - *COSA server and 'mirroring' object; test I/O phase*




- **duration of one SOAR cycle** *(on a dual 250 MHz Octane, IRIX6.5)*
  - *38ms / 60ms (for 113 nodes and 108 edges, local / via network)*
  - *72ms / 100ms (for 226 nodes and 216 edges, local / via network)*

# Wrapping with CORBA - Implementation


- **using MICO**
  - *CORBA is a standard, MICO is a free implementation (see http://www.mico.org)*
- **using the Standard Template Library STL**
  - *standard C++ library*
  - *easy to use classes to handle knowledge graphs*
- **using the new C API of SOAR**
  - *extended in some areas*
- **documented with “doxygen”**
  - *free tool (see http://www.stack.nl/~dimitri/doxygen)*
  - *generates documentation from special C++ comments*
  - *“doxygen” is used for the C API as well*


# What is the “language front end” ?

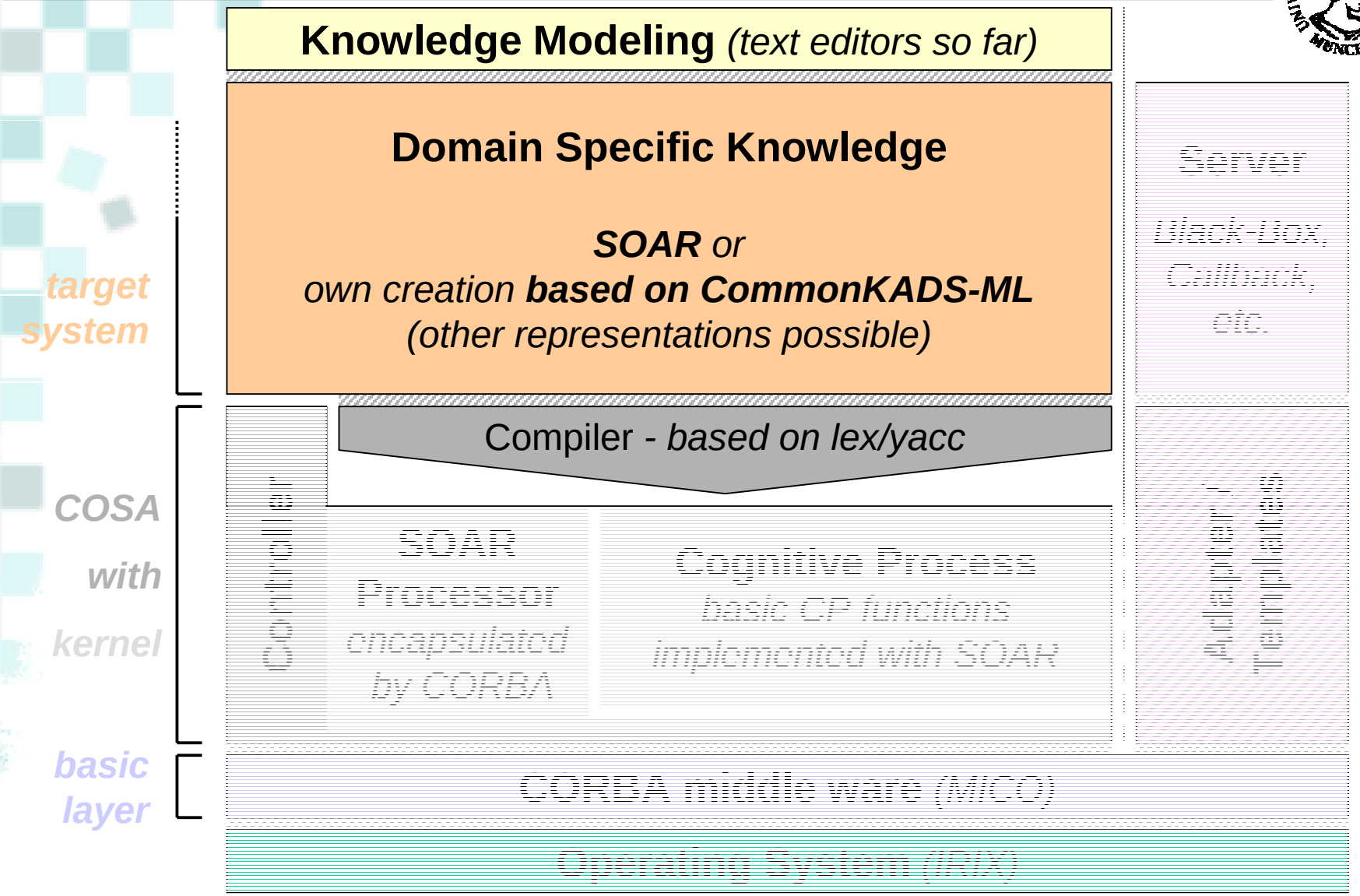
# COSA - language front end

COSA

- Kernel
  - ↳ Processor SOAR
    - » uniform data (WM)
    - » uniform algorithm / behavior (rules)
  - ↳ Library Cognitive Process
    - » realizes the Cognitive Process
    - » object oriented abstraction in SOAR
    - » knows about components
- CORBA encapsulation
  - » distributed system / component handling (make use of kernel)
  - » knowledge abstraction (wrapping for distribution via CORBA)
  - » interfacing with other (external) systems (e.g. in the cockpit)

- **Language Front End**
  - ↳ *Compiler*
    - » input is knowledge, which is compiled to run on the kernel
    - » other knowledge descriptions (besides SOAR) are possible

# Language front end - components of COSA



# Language front end - features

- **language front end**
  - *compiled to run on the COSA kernel*
  - *will save the user from the need of learning SOAR*
- **main problem**
  - *(not only) mapping to SOAR*
  - *mapping on to the kernel of COSA: SOAR and the Cognitive Process library*
  - *languages are basing on own model, not CP*
- **first promising tries are using CommonKADS-ML**
- **others are planned**
  - *more object oriented languages (similar to C++ ?)*

# Summary and Conclusion

# What have we done ?

- **COSA - cognitive system architecture**
  - *new approach towards cognitive systems*
- **wrapped SOAR with CORBA**
  - *knowledge processor of COSA*
  - *distributed system*
  - *knowledge mapping similar to conceptual graph*
- **cognitive process**
  - *implementation on top of SOAR*
  - *introduced an object oriented view (models) to SOAR*
- **languages / knowledge front end**
  - *first abstractions towards other representations*

# Actual State

- **state of implementation**
  - *SOAR wrapper in use*
    - » speed improvements planned if necessary
  - *Prototype using COSA is running (COSY$^{flight}$)*
    - » simple implementation in some areas of the cognitive process
    - » improvements and further development

- **future**
  - *perfecting COSA and the SOAR kernel in it*
  - *improve existing and add new knowledge front ends*
  - *next milestone: build a more complex system (UAV)*

## Benefit for others


- **use experience**
  - *limited; indirect by using COSA*
- **extend COSA to test other theories or languages**
  - *implementation of any block from the architecture can be changed*
- **use COSA as architecture - communicate knowledge**
  - *need to wait until it is ready to be used (2002)*
- **use wrapping of SOAR only**
  - *some minor work to do*
  - *no pure SOAR encapsulation*

# Contact Information


- **postal**
  - *Henrik Putzer*
    *Universität der Bundeswehr München - LRT13*
    *Werner-Heisenberg-Weg 39*
    *85577 Neubiberg*
    *Germany*
- **wire**
  - *Phone: +49-(0)89-6004-3579*
  - *Fax: +49-(0)89-6004-2082*
- **web**
  - *http://www.unibw-muenchen.de/campus/LRT/LRT13*
- **e-mail**
  - *Henrik.Putzer@UniBw-Muenchen.DE*

# References

- **CORBA (standard of the Object Management Group)**
  - *http://www.omg.org*
  - *http://www.corba.org*
- **MICO (free CORMA implementation)**
  - *http://www.mico.org*
- **DOXYGEN (free documentation tool)**
  - *http://www.stack.nl/~dimitri/doxygen*