Training TANGO

Users Session

04.02.2003

TANGO : introduction

A «computing tool» dedicated to the implementation of distributed systems, heterogeneous and oriented control/commande (switch)

TANGO : introduction

- Distributed Systems ?

  The system components are geographically distributed on machines connected through a computing network

- Heterogeneous Systems ?

  A coherent whole made from heterogeneous hardwares and softwares

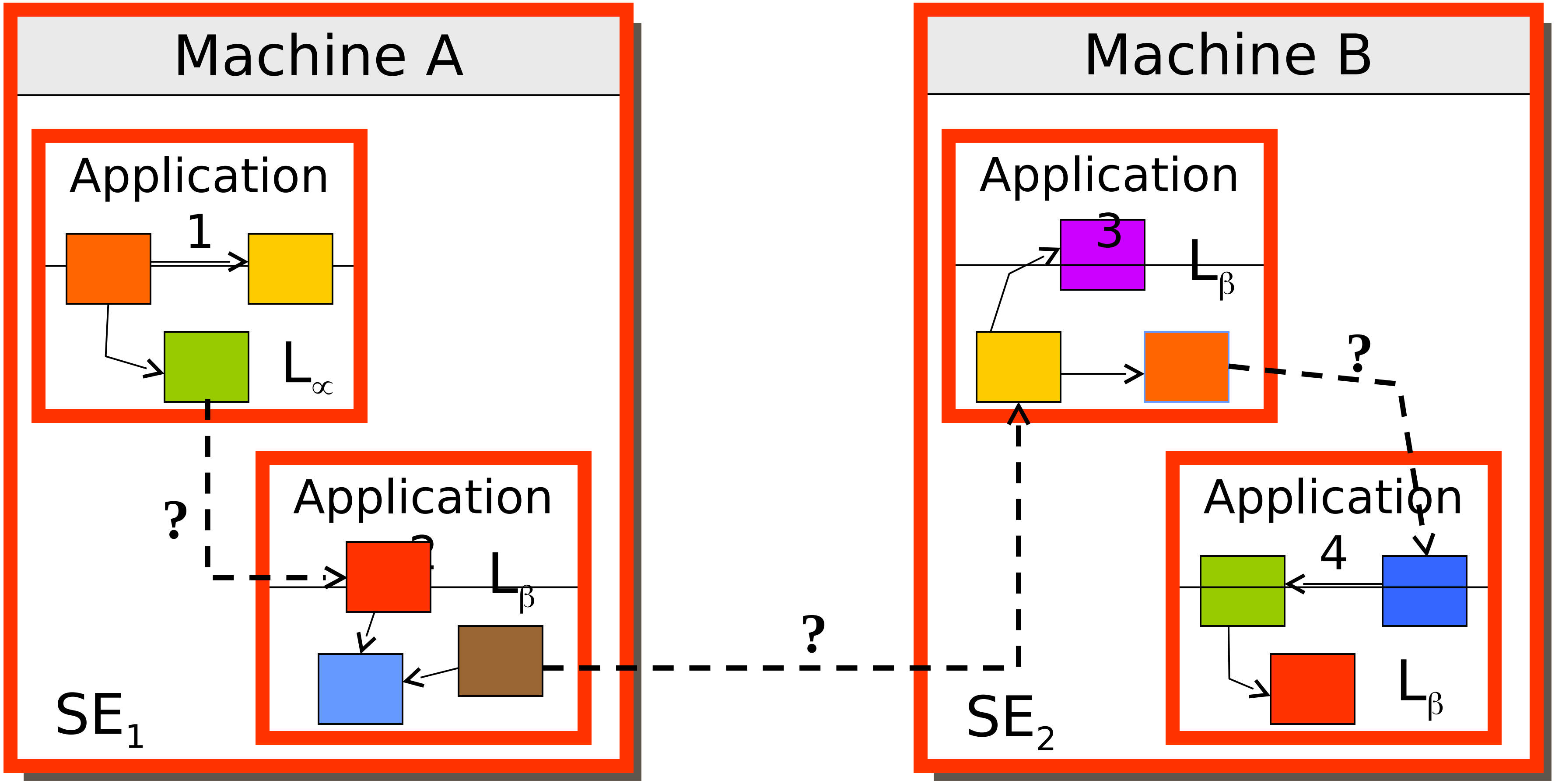
- oriented Systems ctrl/command ?

  Services adapted to a control system (storage, logging, alarms, …)

TANGO : introduction

How does TANGO solve the contraints of activity distribution and of interoperability (interaction) of the heterogeneous components ?

TANGO : introduction



Machine A

Application 1

$L_\propto$

Application 2

L_β

?

SE_1

?

Machine B

Application 3

L_β

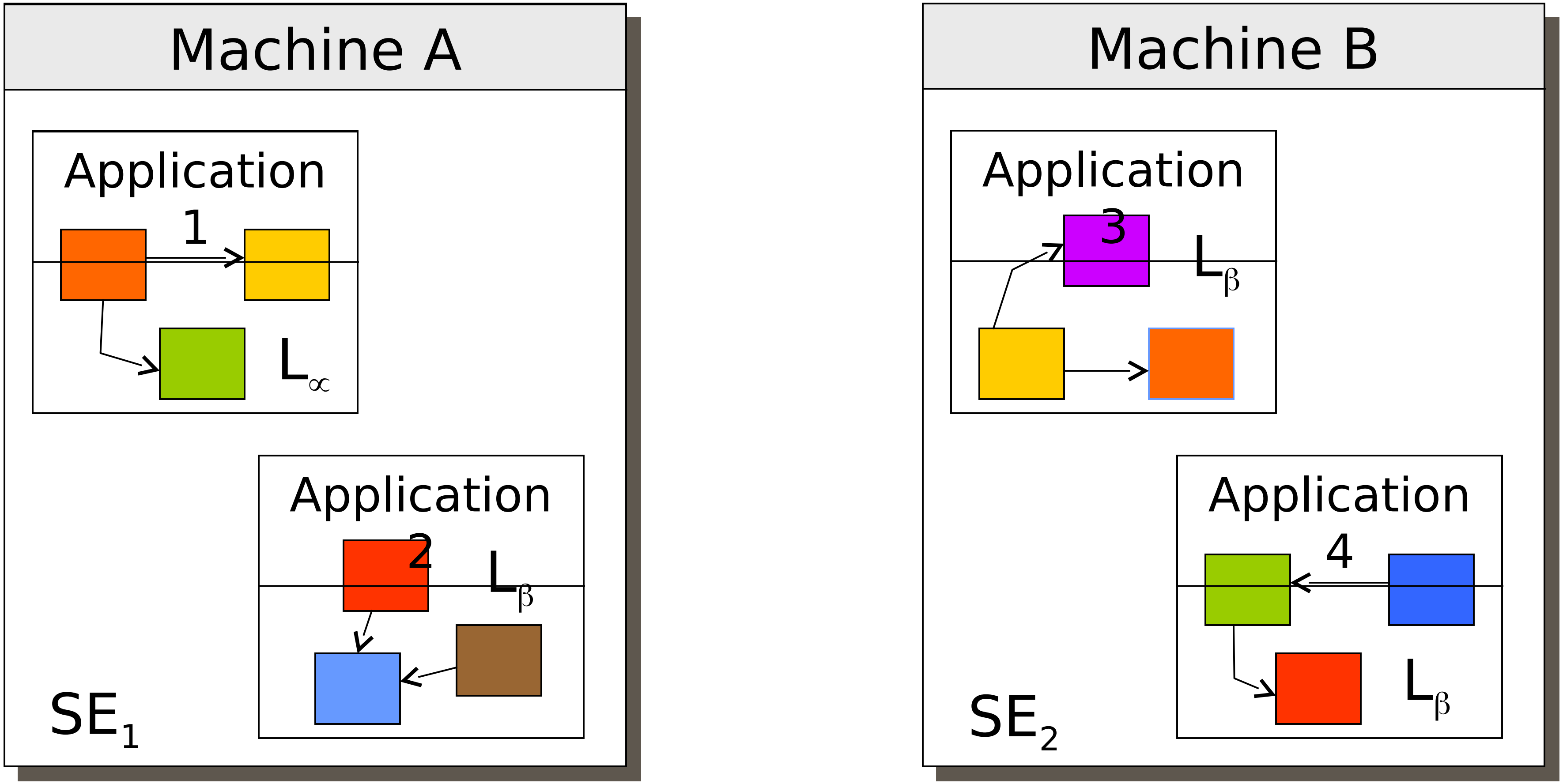
?

Application 4

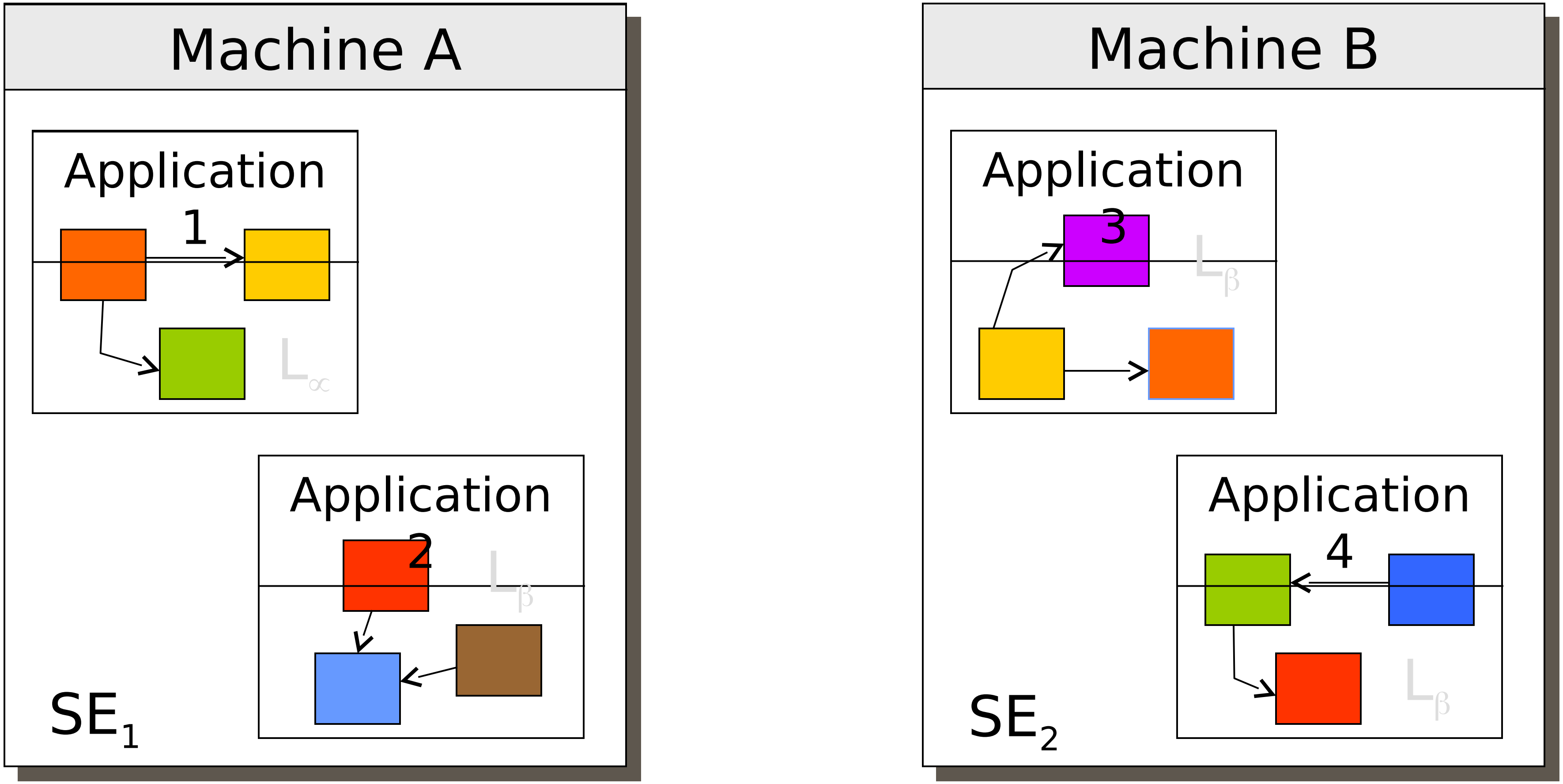
L_β

SE_2

TANGO : introduction



TANGO : introduction



Machine A

Application 1

$L_\propto$

Application 2

L_β

SE_1

Machine B

Application 3

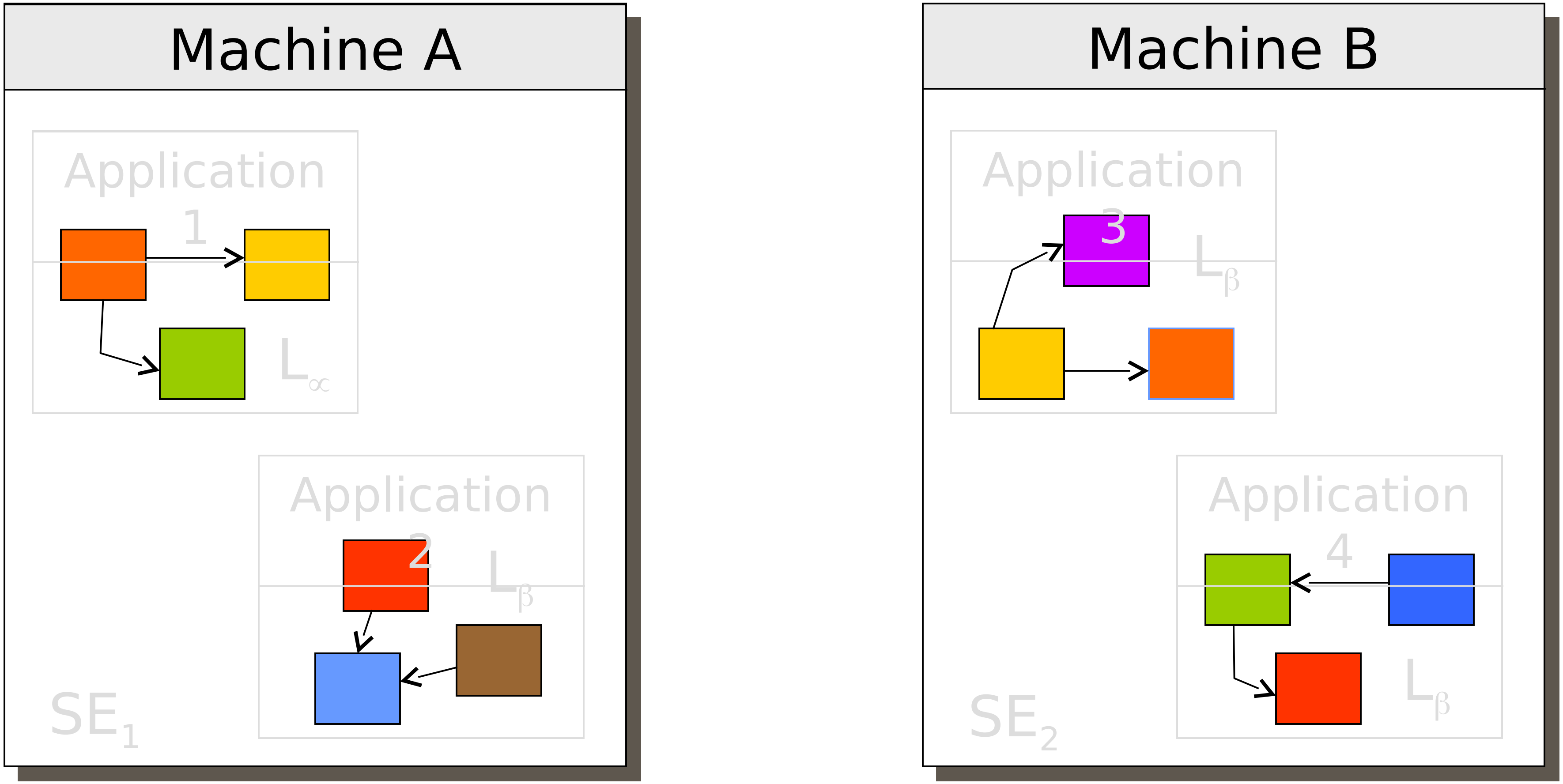
L_β

Application 4

L_β

SE_2

TANGO : introduction



Machine A

Application 1

$L_\propto$

Application 2

L_β

SE_1

Machine B

Application 3

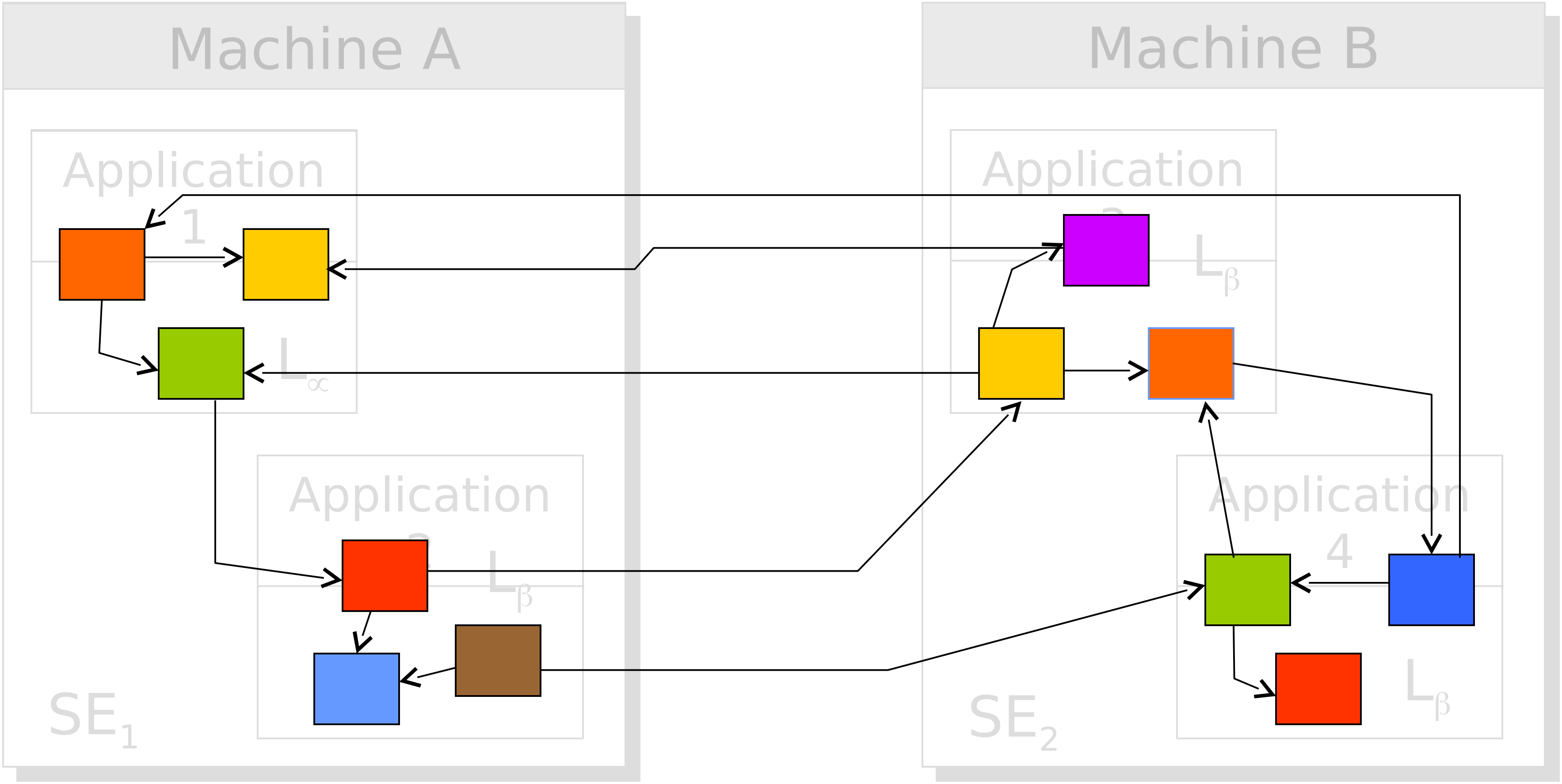
L_β

Application 4

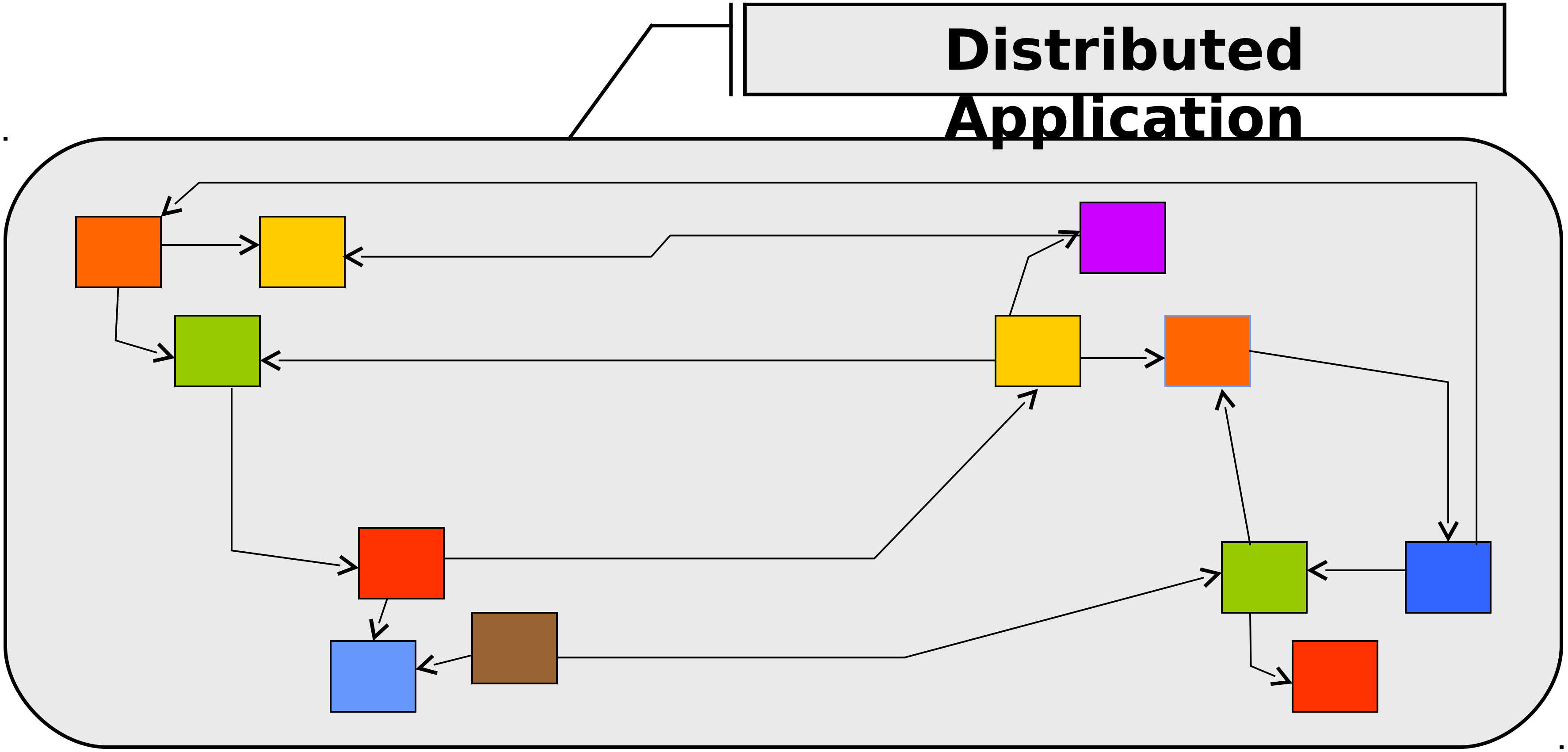
L_β

SE_2

TANGO : introduction

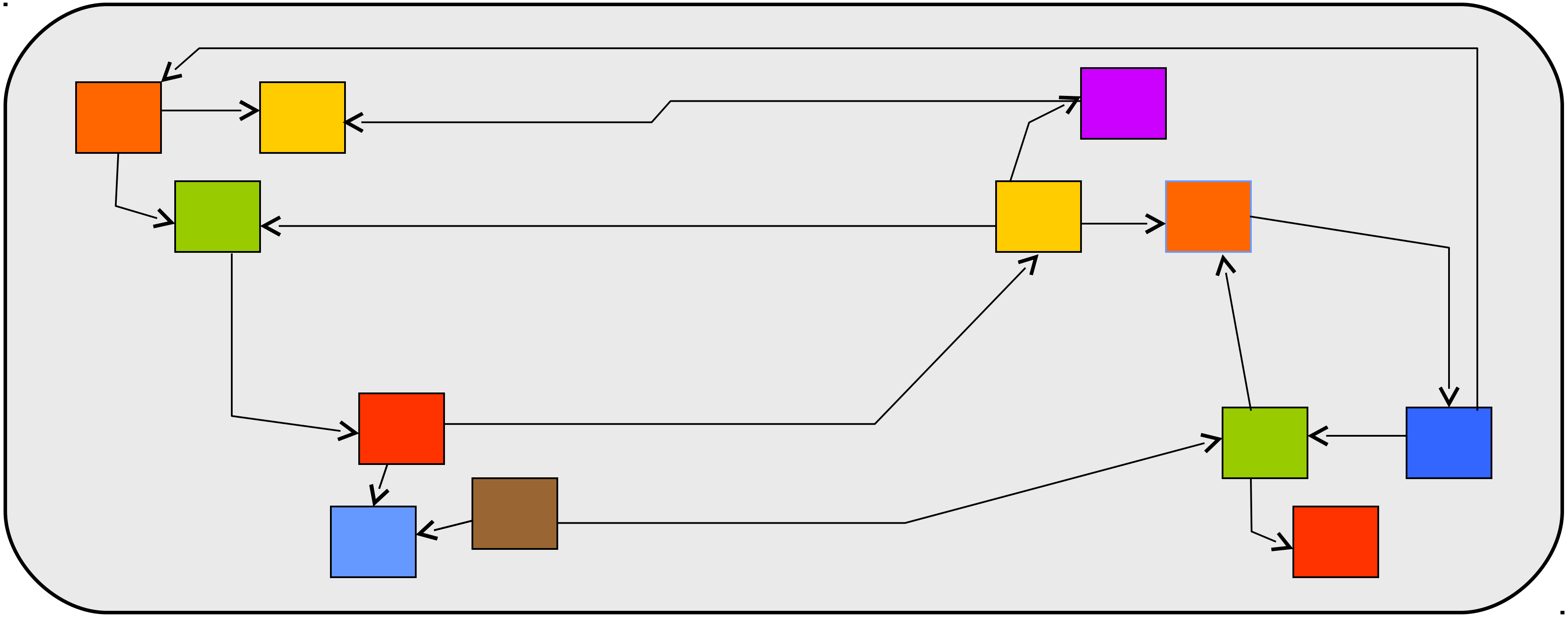


TANGO : introduction

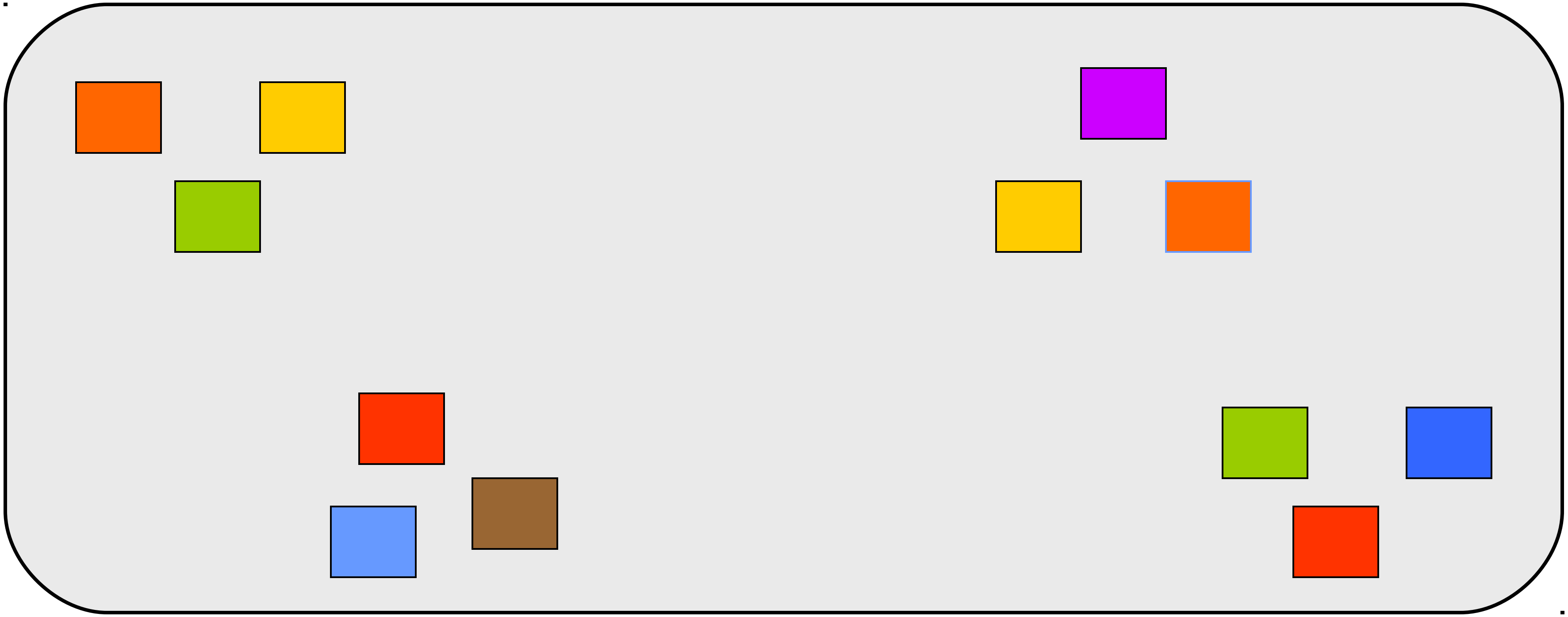


Distributed Application

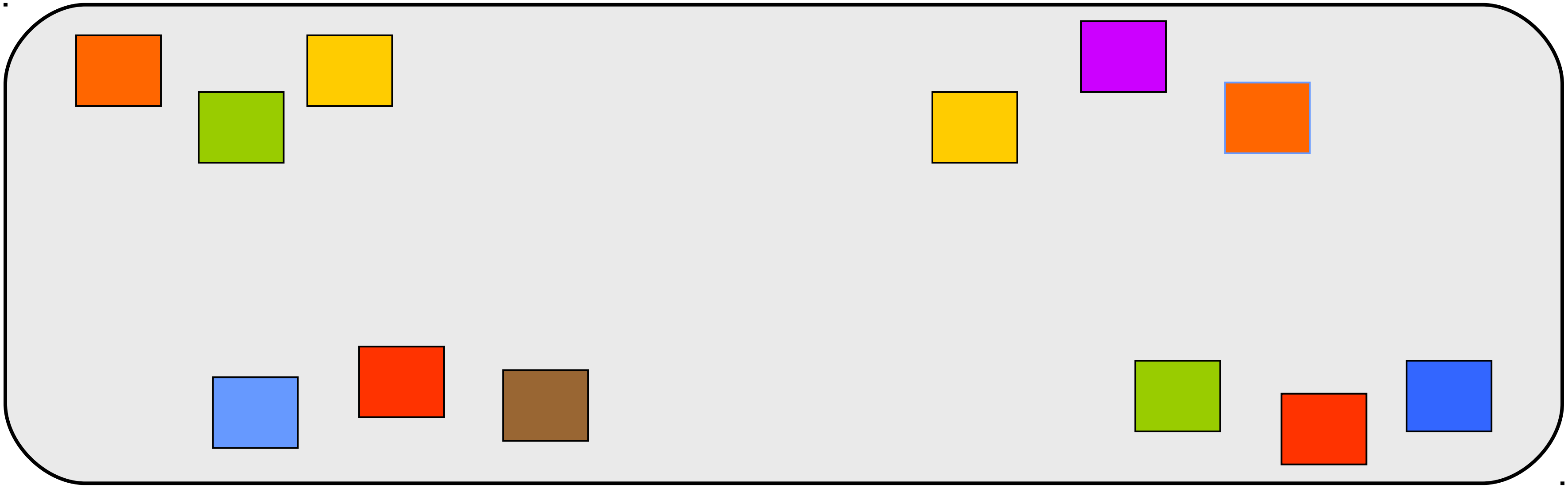
TANGO : introduction



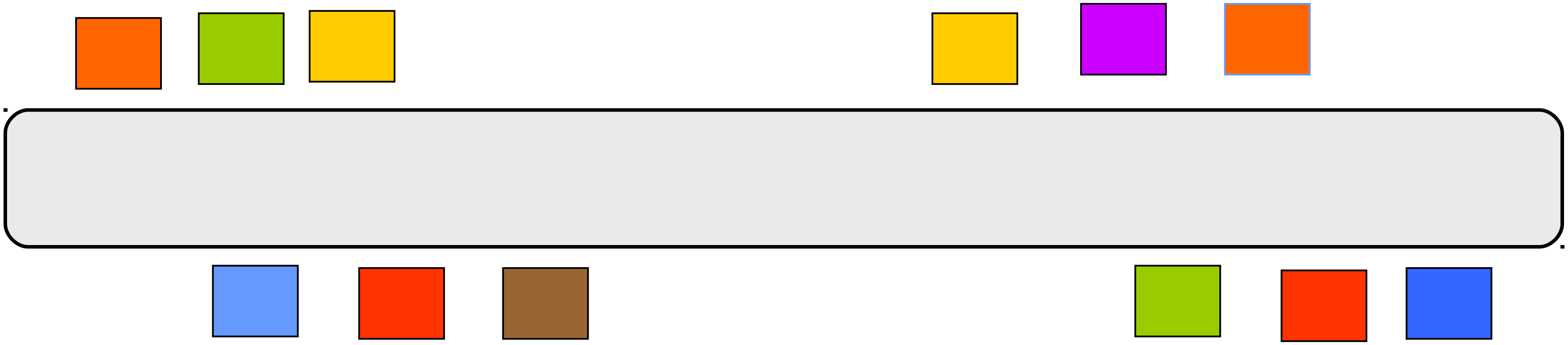
TANGO : introduction



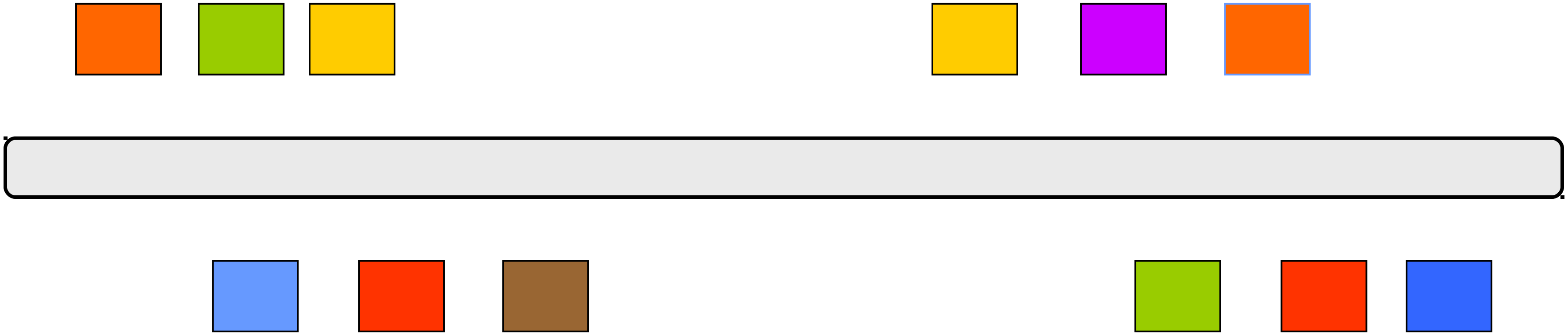
TANGO : introduction



TANGO : introduction



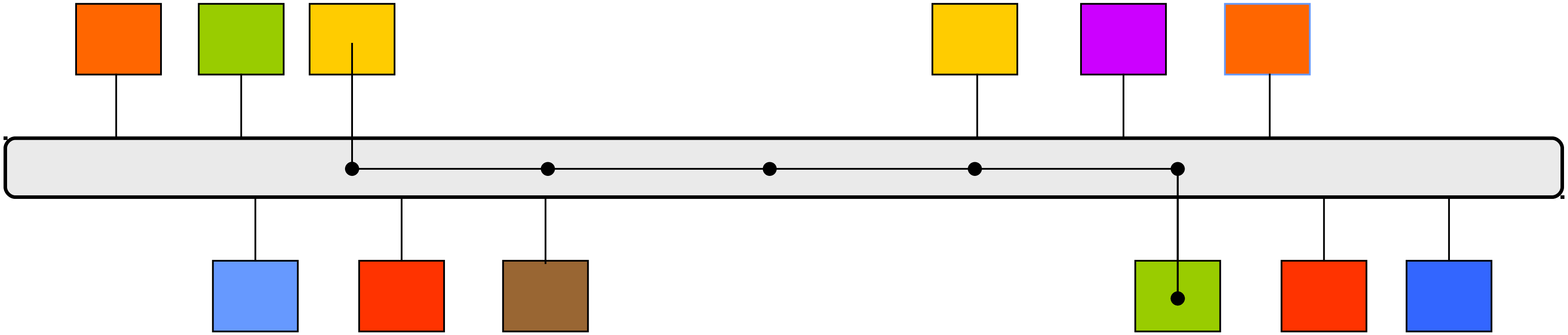
TANGO : introduction



TANGO : introduction

CORBA

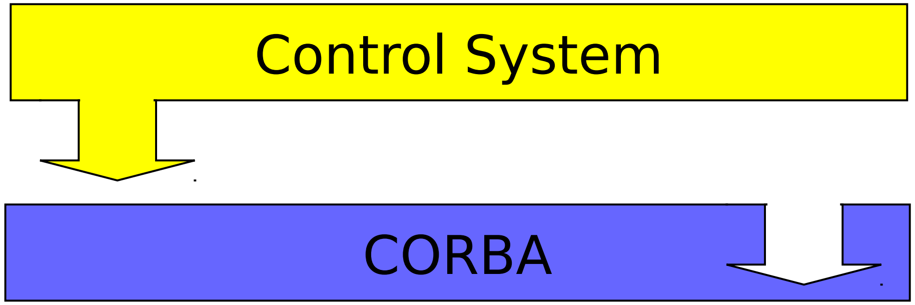
(Common Object Request Broker Architecture)



A standard tool in charge of the communications between software components making up distributed and heterogeneous applications

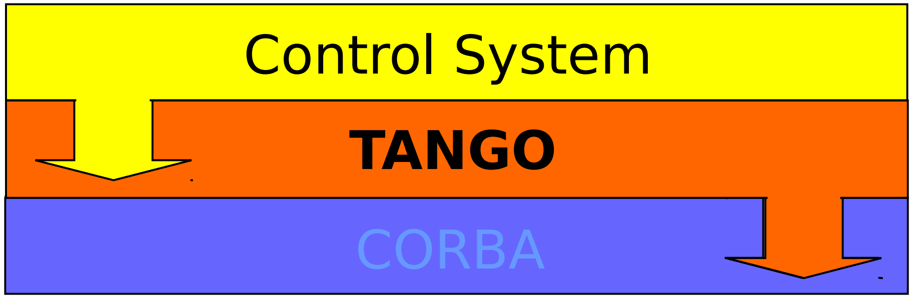
TANGO : introduction

- CORBA = 1 generic tool
  - support to the development of distributed applications
  - A powerful but cumbersome tool
  - No functionalities «ctrl/cmd oriented»



TANGO : introduction

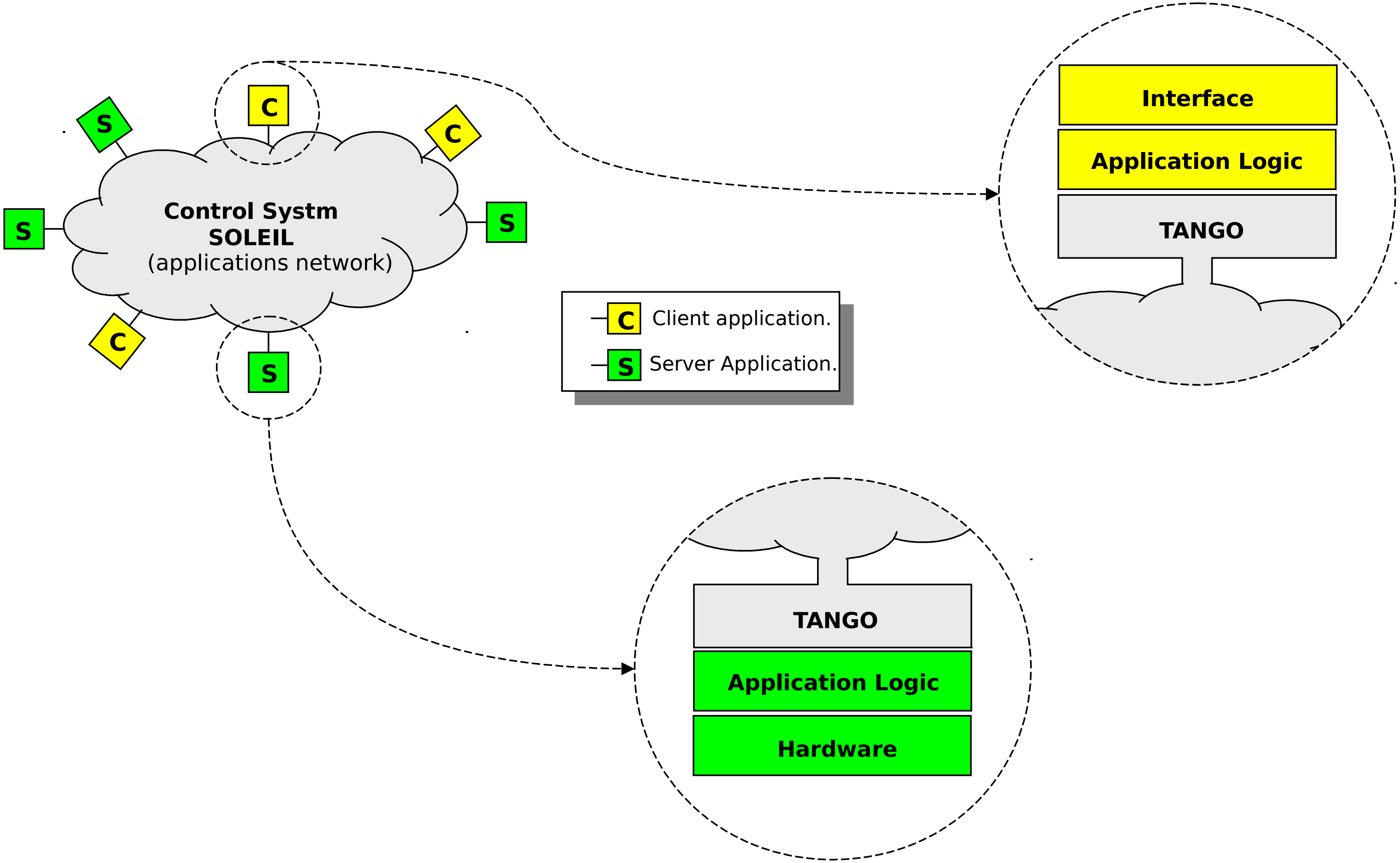
- a framework CORBA ctrl/cmd oriented
  - A toolbox to implement the system
  - A specialization of CORBA adapted to the needs



TANGO : introduction

- Unifier kernel (core) of the system
  - Overall consistency

TANGO : introduction



TANGO : introduction

- Unifier kernel (core) of the system
  - Overall consistency
- factorize the services
  - factorize additions and corrections

TANGO : introduction

- Unifier kernel (core) of the system
  - Overall consistency
- factorize the services
  - factorize additions and corrections
- standardize the applications
  - harmonization of the applications structure

TANGO : introduction

- Unifier kernel (core) of the system
  - Overall consistency
- factorize the services
  - factorize additions and corrections
- standardize the applications
  - harmonization of the applications structure
- masks technical details
  - 1 interface of simplified programmation (APIs)
  - focus on the application logic

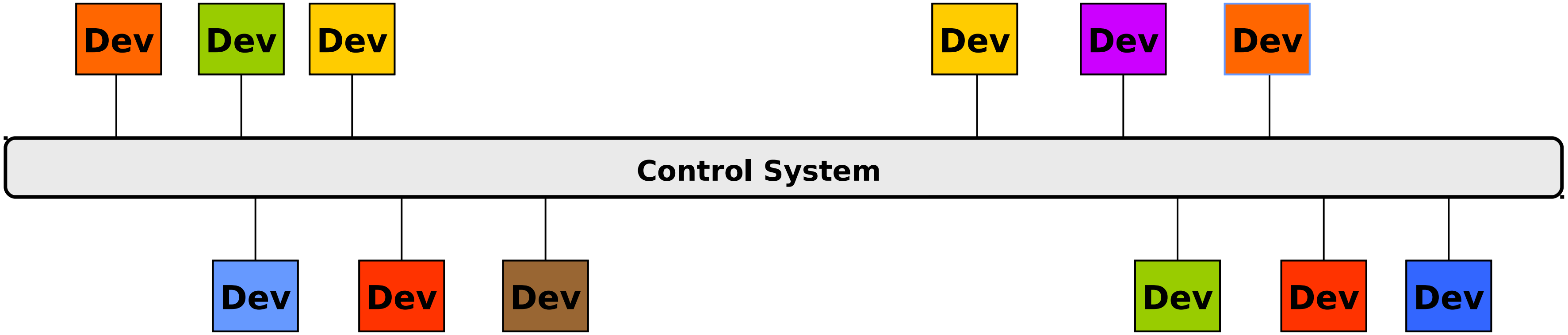
TANGO : introduction

- Philosophy …
  - simplicity
    - mask the CORBA mechanisms
    - propose simplified APIs
  - genericity
    - Enable the writing of generic clients
    - On the communications point of view: 1 single object type

TANGO : introduction

TANGO : device

- an abstract concept : the «device»
  - central component of the structure (architecture)

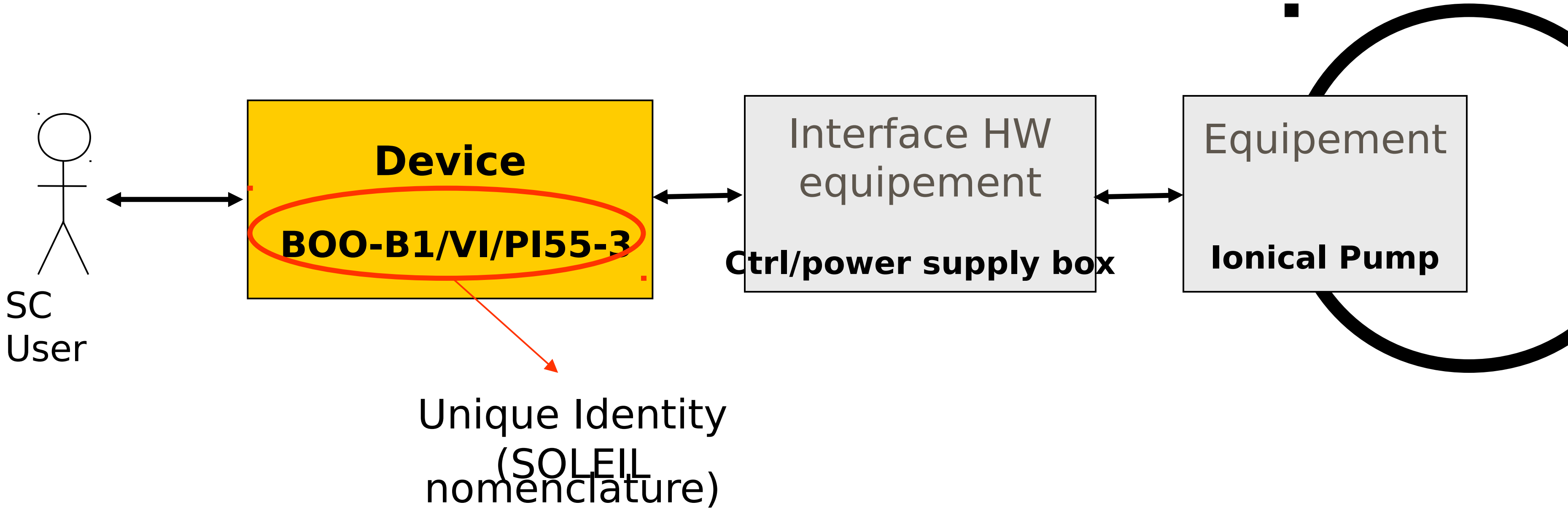


TANGO : device : definition

- device = 1 <entity> to be controlled
  - Hardware of software
  - device «physical» / device «logical»
- device = 1 polymorphous object
  - 1 equipment (ex: 1 power supply)
  - 1 collection of equipments (ex: 1 motor + 1 encoder)
  - 1 devices agregate (ex: a beamline)
  - 1 application (ex: 1 agent of the storage service)

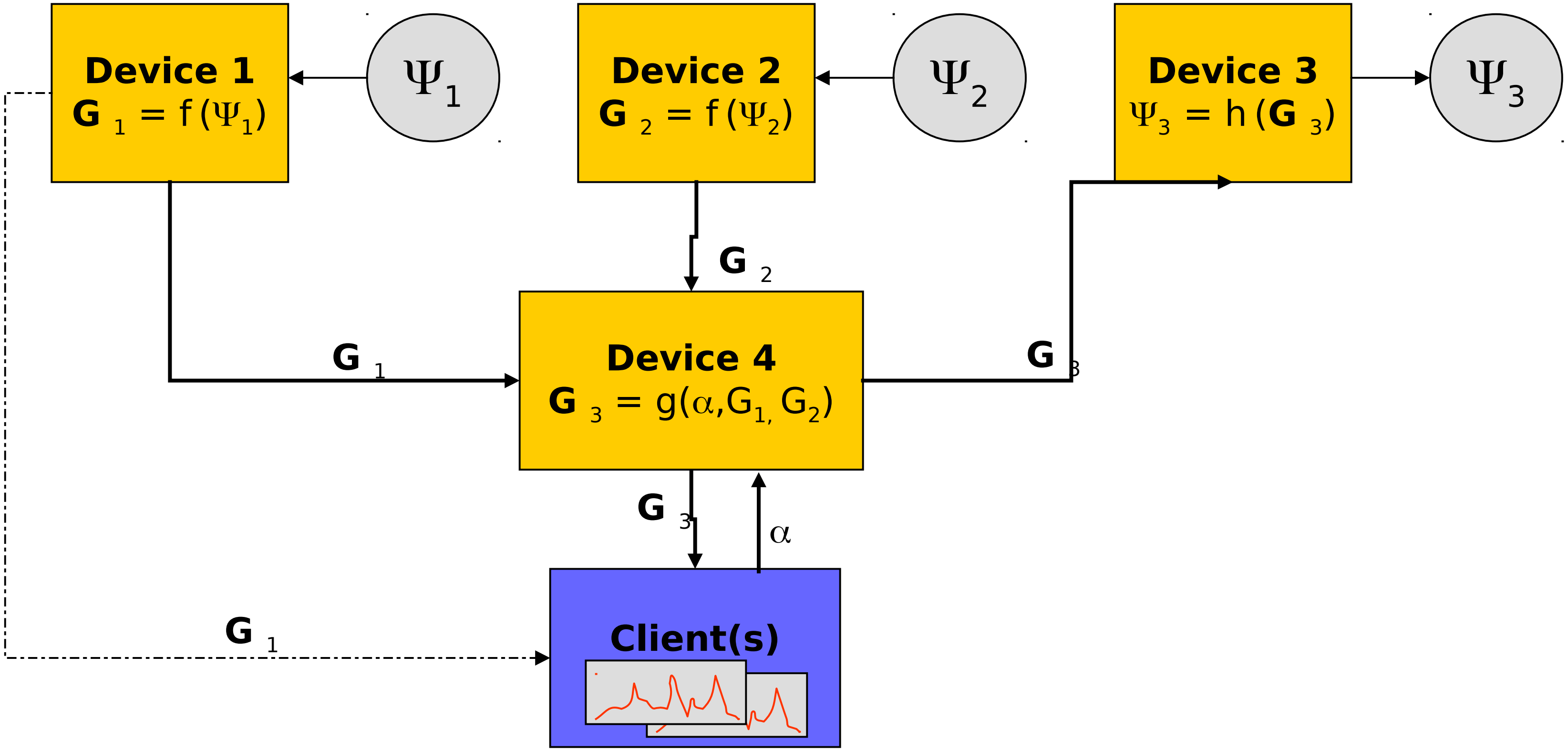
TANGO : device : définition

- device = 1 equipment
  - The most simple and widespread case



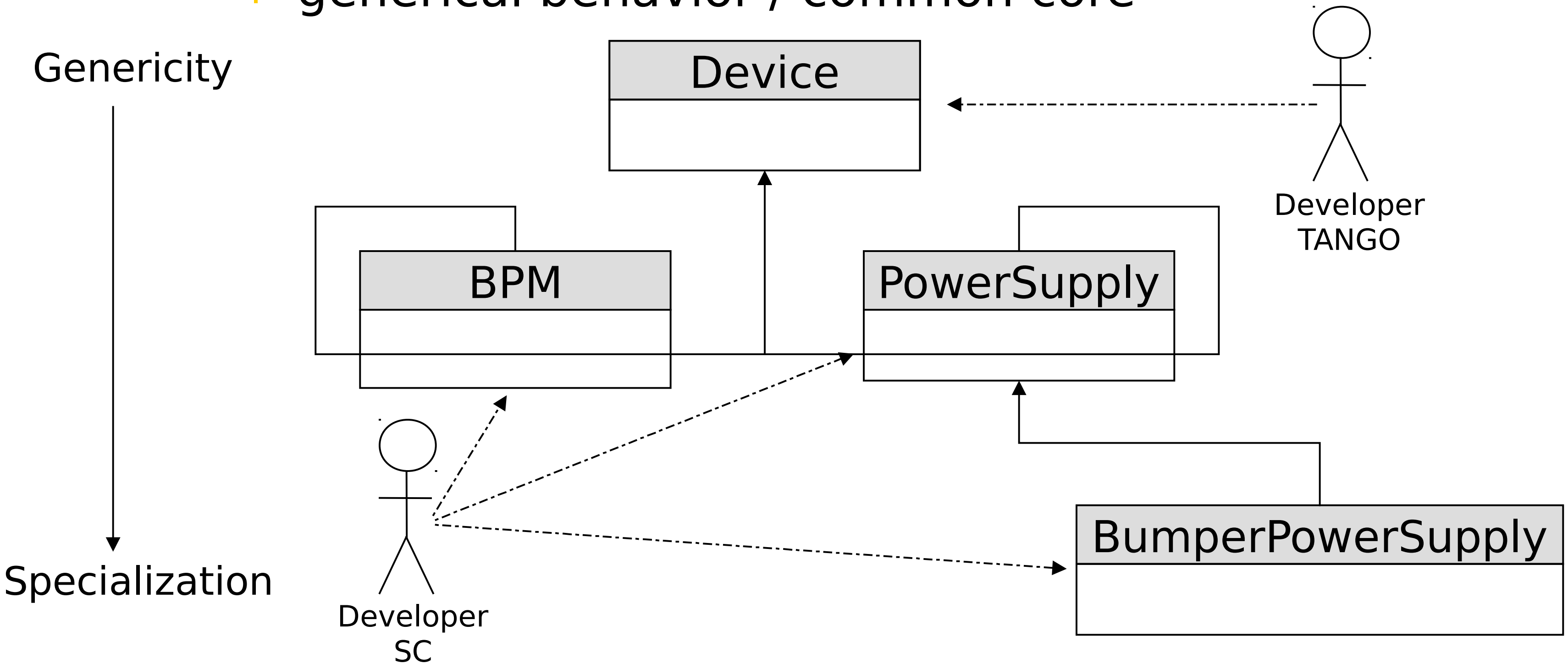
TANGO : device : definition

- device = 1 application
  - Logic device

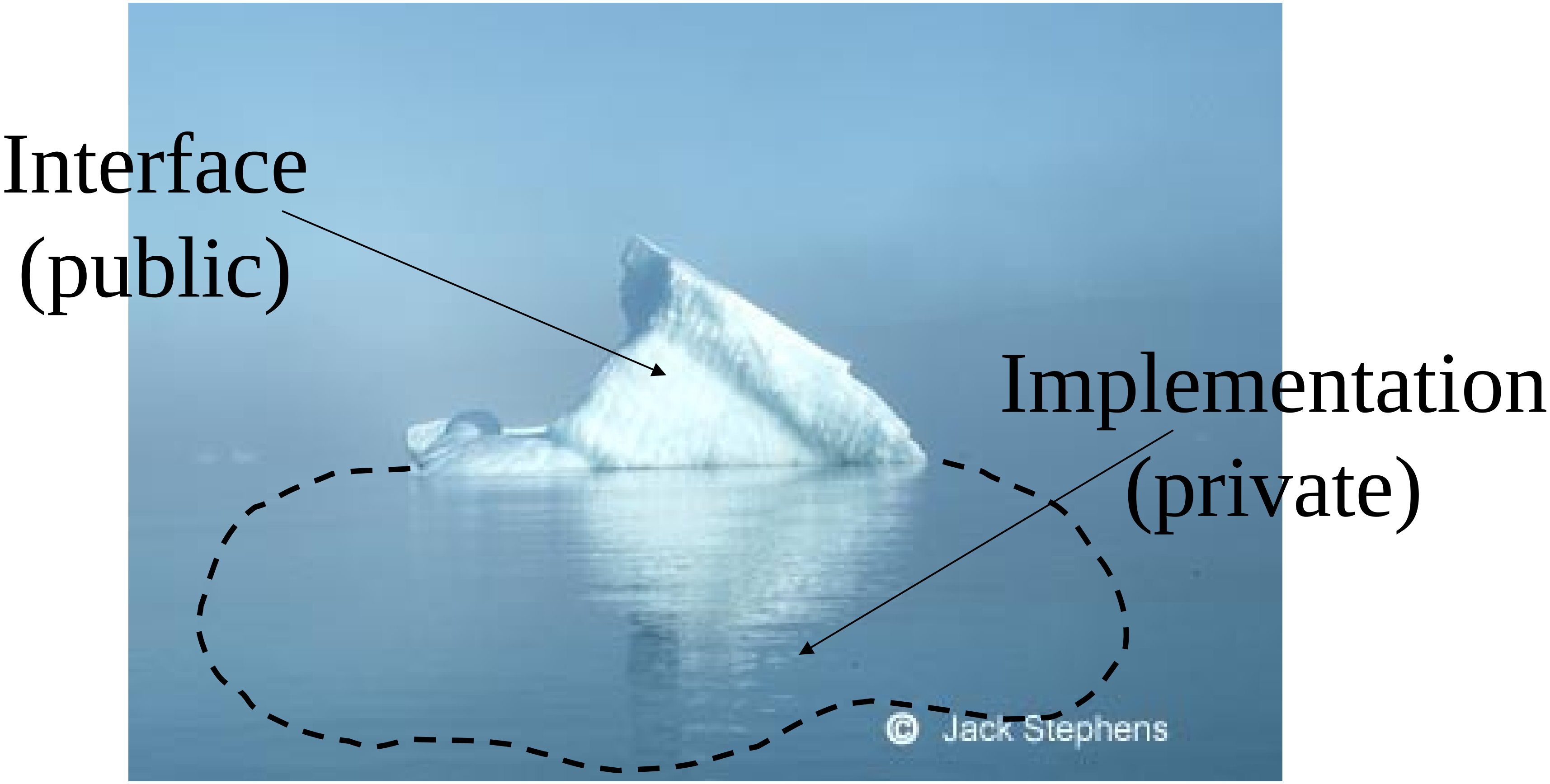


TANGO : device : class

- Belongs to a class
  - member of a devices’ family
  - derived from a basic (common) class
    - generical behavior / common core



TANGO : device : interface



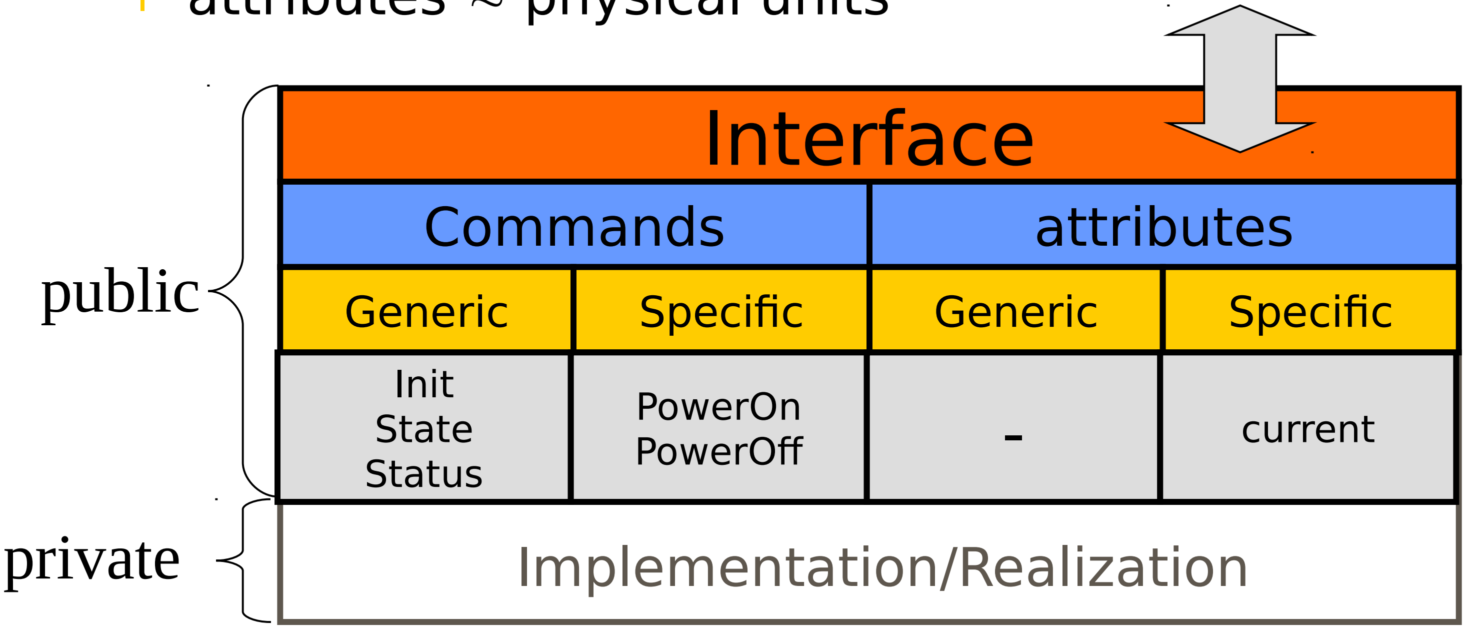
Interface (public)

Implementation (private)

© Jack Stephens

TANGO : device : interface

- Owns a communication interface
  - interface device <=> class
  - interface = commands + attributes
    - commandes ≈ actions
    - attributes ≈ physical units



public

Interface

Commands

attributes

Generic

Specific

Generic

Specific

Init
State
Status

PowerOn
PowerOff

-

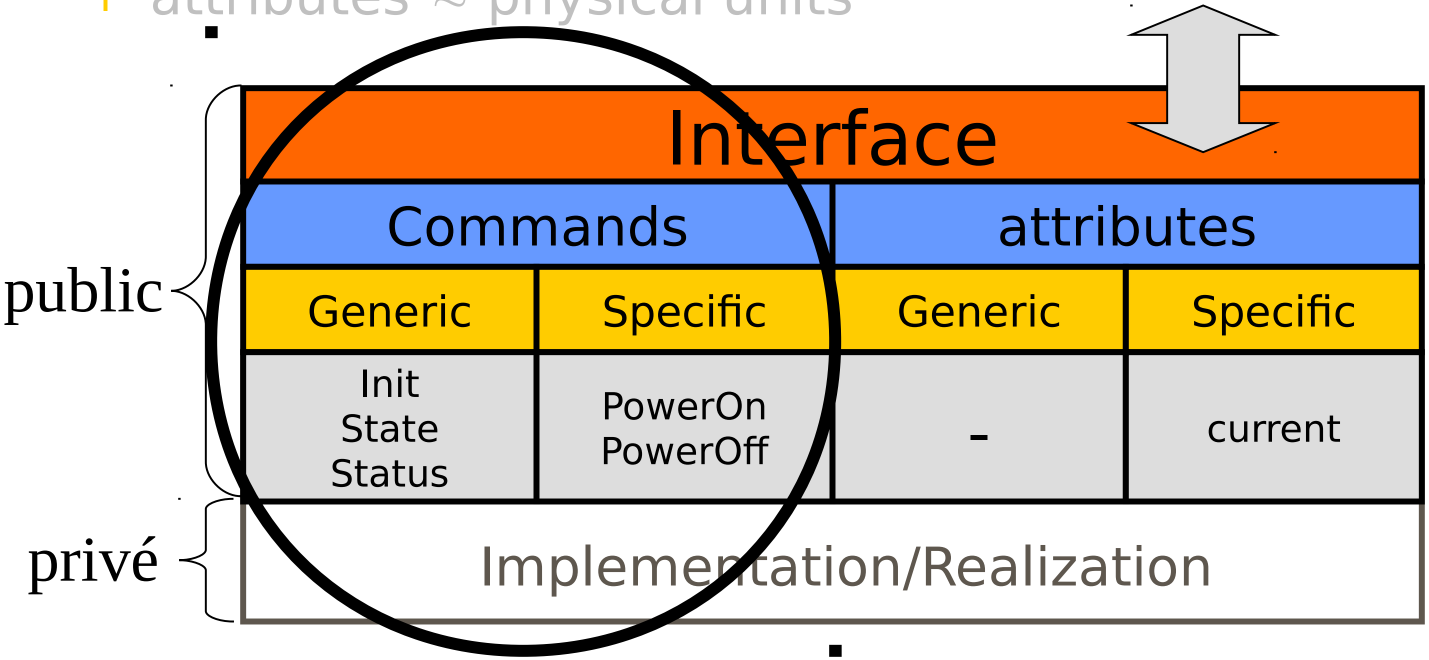
current

private

Implementation/Realization

TANGO : device : interface

- Owns a communication interface
  - interface device <=> class
  - interface = commands + attributes
    - commands ≈ actions
    - attributes ≈ physical units



Interface			
Commands		attributes	
Generic	Specific	Generic	Specific
Init State Status	PowerOn PowerOff	-	current
Implementation/Realization			

TANGO : device : interface : command

- 1 action
- 0 ou 1 entry argument (argin)
- 0 ou 1 exit argument (argout)
- argin & argout = 1 of the 20 TANGO types
- Execution : indirect mecanism
  - -> generic approach of TANGO
  - -> 1 CORBA method : command_inout
    - belongs to the generic interface of the devices
    - dedicated to the fulfillment of non generic commands
      - Device’s specificity (PowerSupply, StepperMotor, …)
    - only one signature : 400 combinations argin/argout !
    - generic containers (CORBA::any)

TANGO : device : interface : command

- About the argin & argout type...

TANGO	Desc	Matlab
DEV_VOID	no argin and/or no argout	-
DEV_STATE	Device status	1-by-n char array
DEV_STRING	Characters chain	1-by-n char array
DEV_BOOLEAN	boolean	1-by-1 uint16 array
DEV_SHORT	Integer 16 bits signed	1-by-1 int16 array
DEV_USHORT	Integer 16 bits non signed	1-by-1 uint16 array
DEV_LONG	Integer 32 bits signed	1-by-1 int32 array
DEV_ULONG	Integer 32 bits non signed	1-by-1 uint32 array

TANGO : device : interface : commande

- About the argin & argout type ...

TANGO	Desc	Matlab
DEV_FLOAT	real 32 bits	1-by-1 single array
DEV_DOUBLE	real 64 bits	1-by-1 double array
DEVVAR_CHARARRAY	Octets chart (i.e. characters)	1-by-n char array
DEVVAR_ SHORTARRAY	Integers chart 16 bits signed	1-by-n int16 array
DEVVAR_ USHORTARRAY	Integers chart 16 bits not signed	1-by-n uint16 array
DEVVAR_ LONGARRAY	Integers chart 32 bits signed	1-by-n int32 array
DEVVAR_ ULONGARRAY	Integers chart 32 bits not signed	1-by-n uint32 array
DEVVAR_ FLOATARRAY	reals chart 32 bits	1-by-n single array

TANGO : device : interface : commande

- A propos du type d’argin & argout…

TANGO	Desc	Matlab
DEVVAR_ DOUBLEARRAY	Reals chart 64 bits	1-by-n double array
DEVVAR_ STRINGARRAY	Non bounded characters chains chart	1-by-n cell array of {1-by-n char array}

DEVVAR_LONGSTRINGARRAY	structure containing an integers chart 32 bits signed and a characters chains chart	1-by-n struct array {
		field **lvalue**:
		1-by-n int32 array
		field **svalue**:
		1-by-n cell array of
		{1-by-n char array}
		}

DEVVAR_DOUBLESTRINGARRAY	structure containing a reals chart 64 bits and a characters chains chart	1-by-n struct array {
		field **dvalue**:
		1-by-n double array
		field **svalue**:
		1-by-n cell array of
		{1-by-n char array}
		}

TANGO : device : interface : command

- Syntaxe
  - Prog. env. OO (C++, Java, Python)

    argout = dev.command_inout (cmd_name, argin)

  - User env. (Matlab, Igor Pro, …)

    argout = tango_command_inout (dev_name, cmd_name, argin)

  - Examples Matlab

```
>> help tango_command_inout
>> dev = ‘tango/tangotest/1’
>> tango_command_inout(dev,‘DevDouble’,pi)
>> tango_command_inout(dev,‘DevVarDoubleArray’,[1,2,3])
>> s.dvalue = [pi, 2*pi, 3*pi]
>> s.svalue = {‘dev’, ‘var’, ‘double’, ‘array’, ‘test’}
>> tango_command_inout(dev,‘DevVarDoubleStringArray’,s)
```

TANGO : device : interface : command

- Name and signature of the commands ?
  - Device’s Documentation
    - `http://controle/DeviceServers/Galil/doc_html`
  - Prog. env.. OO (C++, Java, Python)
    - cmd_list_info = dev.command_list_query ()
    - cmd_info = dev.command_query (cmd_name)
  - User env. (Matlab, Igor Pro, …)
    - cmd_list_info = tango_command_list_query (dev_name)
    - cmd_info = command_query (dev_name, cmd_name)
  - Examples Matlab
    - `>> tango_command_list_query(dev)`
    - `>> tango_command_query(dev,‘DevDouble’)`
    - `>> tango_print_cmd_list(dev)`

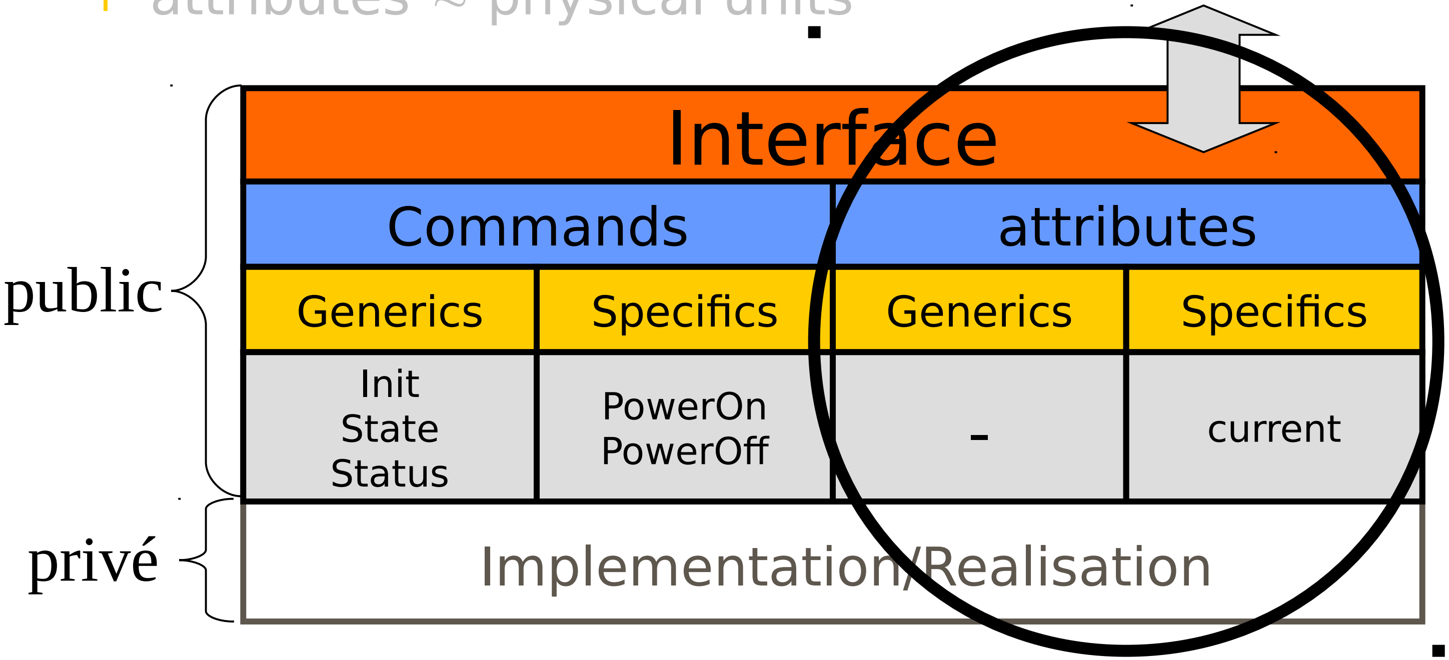
TANGO : device : interface : command

- Errors processing
  - Prog. env.. OO (C++, Java, Python) :
    - exceptions : mechanism try/catch (DevFailed & derived)
  - User env. (Matlab, Igor Pro, …) :
    - error code : updating after each execution of a «command»
  - Examples Matlab:
    - `>> result = tango_command_inout(dev,‘dummy’,pi);`
    - `>> tango_error`
    - if tango_error == -1 then …
      - result is invalid, indéfini
      - result can be not of the expected type !
      - Do not use it !
    - `>> help tango_error`
      - `an example to follow !`

TANGO : device : interface : commands

TANGO : device : interface

- Has a communication interface
  - interface device <=> class
  - interface = commands + attributes
    - commands ≈ actions
    - attributes ≈ physical units



	Interface			
public	Commands		attributes	
	Generics	Specifics	Generics	Specifics
	Init State Status	PowerOn PowerOff	-	current
privé	Implementation/Realisation			

TANGO : device : interface : attribute

- Definition
  - Physical unit produced or administrated by the device
  - ex: a motor’s position, alimentation power supply., …
- Format
  - From 0 to 2 dimensions
    - SACLAR
    - SPECTRUM (i.e. vector)
    - IMAGE (i.e. matrix)
- Type
    - DEV_SHORT, DEV_LONG, DEV_DOUBLE
      - scalar, spectrum or image
    - DEV_STRING
      - scalar only

TANGO : device : interface : attribute

- Accessibility
  - READ
    - accessible in read only
  - WRITE
    - accessible in write only
  - READ_WRITE
    - accessible in read AND in write only
    - Consigne (instructions) vs effective value
  - READ_WITH_WRITE
    - 1 attribute READ linked to 1 attribute WRITE
    - exotic (prefer READ_WRITE)

TANGO : device : interface : attribute

- Features : autodescriptive & parametrizable
  - 1 attribute -> 18 properties
    - generic properties (attribute)
    - 8 non-modifiable properties (developer)
      - name : attribute’s name
      - data_type : data type (DEV_SHORT, DEV_LONG, ...)
      - data_format : data format (SCALAR, SPECTRUM or IMAGE)
      - writable : access mode (READ, WRITE, ...)
      - max_dim_x, max_dim_y : dimensions max
        - dim_x <= max_dim_x
        - dim_y <= max_dim_y
      - disp_level : expert or operator
      - wrt_attr_name : name of the attibute WRITE associated

TANGO : device : interface : attribute

- 10 modifiable properties (user)
  - description : attr. Description (text)
  - label : label associated to the attr. (text)
  - unit : unit in which is expressed the value associated to the attribute (text)
  - standard_unit : conversion factor to the units MKSA (text)
  - display_unit : unit * standard_unit (text)
  - format : display format for the «numerical» attributes (texte)
    - Key-words : fixed, scientific, uppercase, showpoint, showpos, setprecision(), setw()
    - ex : scientific;uppercase;setprecision(3)
  - min_value : min. value of an attribute WRITE or READ_WRITE (text)
  - max_value : max. value of an attribute WRITE or READ_WRITE (text)
  - min_alarm : alarm threshold <low> of an attribute READ or READ_WRITE (text)
  - max_alarm : alarm threshold <high> of an attribute READ or READ_WRITE (text)

TANGO : device : interface : attribute

- List of the attributes ?
  - Documentation of the device
    - `http://controle/DeviceServers/Galil/doc_html`
  - Prog. env. OO (C++, Java, Python)
    - attr_list = dev.get_attribute_list ()
  - User env. (Matlab, Igor Pro, …)
    - attr_list = tango_get_attribute_list (dev_name)
  - Example Matlab
    - `>> attr_list = tango_get_attribute_list(dev)`

TANGO : device : interface : attribute

- Standard configuration of an attribute ?
  - Prog. env. OO (C++, Java, Python)
    - attr_config_list = dev.get_attribute_config(attr_name_list)
    - attr_config_list = dev.attribute_list_query()
    - attr_config = dev.attribute_query(attr_name)
  - User env. (Matlab, Igor Pro, …)
    - attr_config_list = tango_attribute_list_query(dev_name)
    - attr_config_list = tango_get_attributes_config(dev_name, attr_name_list)
    - attr_config = tango_attribute_query(dev_name, attr_name)
    - attr_config = tango_get_attribute_config(dev_name, attr_name)
  - Example Matlab

```
>> help tango_attribute_list_query
>> acl = tango_attribute_list_query(dev)
>> acl(2)
```

TANGO : device : interface : attribute

- Modify an attribute’s configuration ?
  - Take care of the consequences !
    - acts upon all the clients
    - sensitive parameters : min/max_value, min/max_alarm
  - Prog. env.. OO (C++, Java, Python)
    - dev.set_attr_config (attr_config_list)
  - User Env. (Matlab, Igor Pro, …)
    - tango_set_attribute_config (dev_name, attr_config)
    - tango_set_attributes_config (dev_name, attr_config_list)
  - Example Matlab

```
>> help tango_set_attributes_config
>> scc = tango_get_attribute_config(dev, ‘short_scalar’)
>> scc.min_value = num2str(str2num(scc.min_value) / 2)
>> scc.max_value = num2str(str2num(scc.max_value) / 2)
>> scc.description = ‘This is a dummy attribute’
>> tango_set_attribute_config(dev, scc)
```

TANGO : device : interface : attribute

- Obtain the standard value of an attribute ?
  - attributes READ, READ_WRITE ou READ_WITH_WRITE
  - Result of the lecture => structure {read value + infos }
    - name : name of the attribute
    - quality : quality of the returned value
      - ATTR_VALID : ok, the returned value is valid
      - ATTR_ALARM : an alarm threshold has been crossed (*cf. min_alarm and max_alarm*)
      - ATTR_INVALID : error, undefined value
    - dim_x : dim.x of the value (*dim_x <= attr_config.max_dim_x*)
    - dim_y : dim.y of the value (*dim_y <= attr_config. max_dim_y*)
    - timestamp : value stamp
    - value : value of the attribute at this very instant <timestamp>
      - SCALAR
        - READ : [0 : val]
        - READ_WRITE et READ_WITH_WRITE : [0:measure, 1:consigne]
      - SPECTRUM
        - [0 : measure, …, dim_x – 1 : measure]
      - IMAGE
        - [0 : measure, …, dim_x – 1 : measure] x [0 : measure, …, dim_y – 1 : measure]

TANGO : device : interface : attribute

- Obtain the standard of an attribute ?
  - Prog. env. OO (C++, Java, Python)
    - dev.read_attributes (attr_name_list)
  - User Env. (Matlab, Igor Pro, …)
    - tango_read_attribute (dev_name, attr_name)
    - tango_read_attributes (dev_name, attr_name_list)
  - Example Matlab
    - `>> help tango_read_attribute`
    - `>> scv = tango_read_attribute (dev, ‘short_image’)`
    - `>> datestr(scv.time)`
    - `>> for i=1:10 s=tango_read_attribute(dev, ‘short_spectrum’); plot(s.value); drawnow; end;`

TANGO : device : interface : attribute

- Modify the value of an attribute ?
  - attributes WRITE, READ_WRITE et READ_WITH_WRITE
  - attr_config.min_value <= set value <= attr_config.max_value
    - exception API_WAttrOutisideLimit
  - Prog. env. OO (C++, Java, Python)
    - dev.write_attributes (attr_val_list)
  - User Env. (Matlab, Igor Pro, …)
    - tango_write_attribute (dev_name, attr_name, value)
    - tango_write_attributes (dev_name, attr_name_attr_value_struct_list)
  - example Matlab

```
>> help tango_write_attribute
>> tango_write_attribute (dev, ‘short_scalar’, 123456789)
>> tango_print_error_stack
>> tango_write_attribute (dev, ‘short_scalar’, int16(123456789))
>> tango_print_error_stack
>> tango_write_attribute (dev, ‘short_scalar’, int16(1024))
>> tango_read_attribute (dev, ‘short_scalar’)
```

TANGO : device : Status

- 1 device -> 1 status
  - behavior = f (internal status)
    - request -> internal status -> execution or exception
    - Internal status run by the device
  - 14 predefined status
    - ON, OFF, CLOSE, OPEN, INSERT, EXTRACT, MOVING, STANDBY, FAULT, INIT, RUNNING, ALARM, DISABLE, UNKNOWN
    - known and run by the clients (particularly generic)
- Obtain the current status of a device ?
  - Prog. env. OO (C++, Java, Python)
    - dev.state ()
  - User env. (Matlab, Igor Pro, …)
    - dev_state = tango_state (dev_name)
  - example Matlab
    - `>> help tango_state`
    - `>> tango_state(dev)`
    - `>> tango_status(dev)`

TANGO : properties

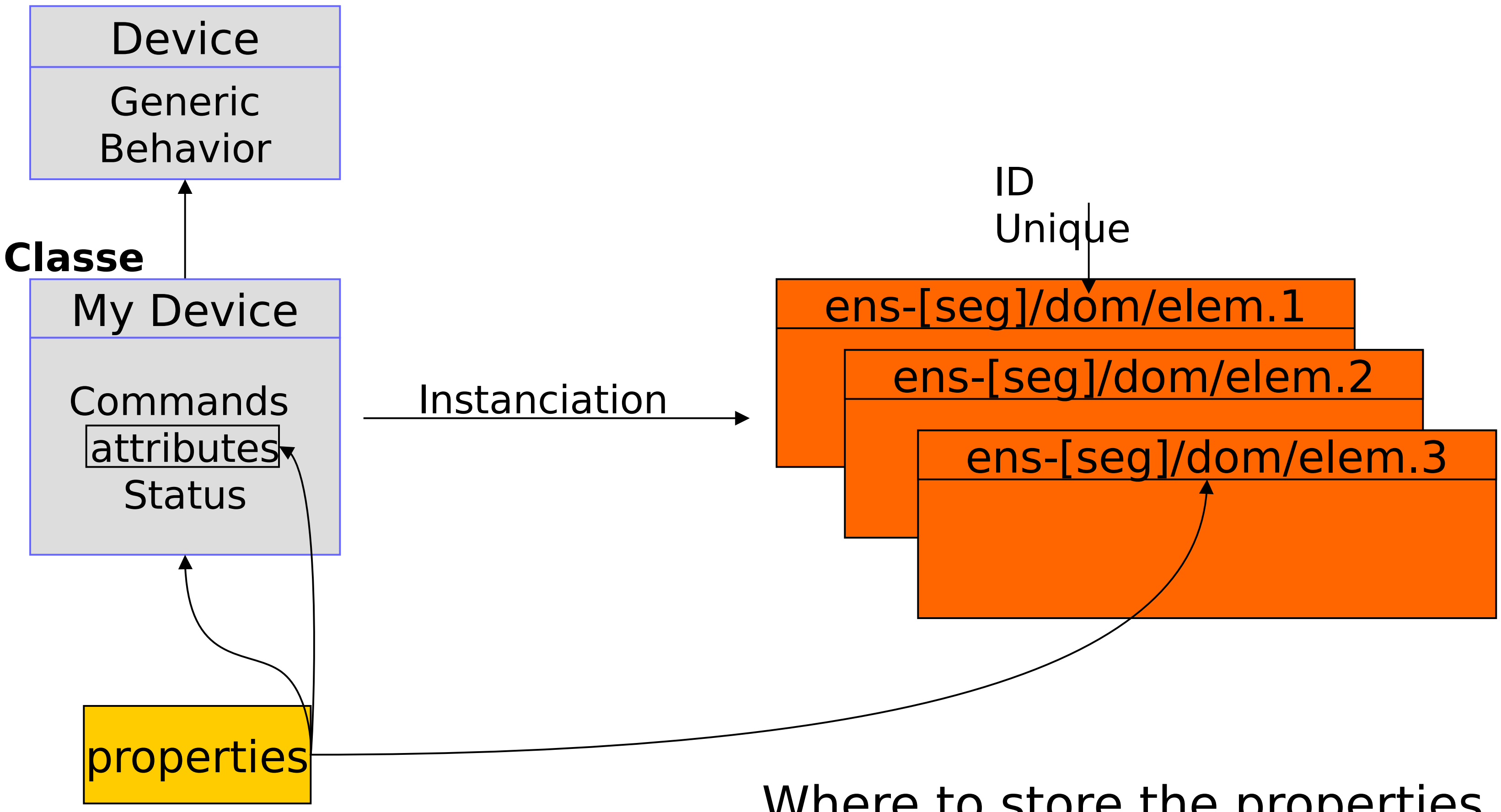
- Definition
  - Configuration Data
  - concept spread to all TANGO entities
    - attribute, device, classe, system
  - Attribute’s property
    - 18 properties TANGO predefined + …
    - … properties defined by the developer
    - ex: initial value of an attribute
  - Device’s property
    - specific to the device
    - defined by the developer
    - ex: adress GPIB of a peripheric
  - Class property
    - shared with all the devices of the class
    - defined by the developer
    - ex: URL of the documentation
  - System’s property
    - shared with all the devices of SC
    - ex: an info related to a centralized service (port n^r of the storage service)

TANGO : properties

- Ex : Manipulate the value of a device’s property ?
  - Beware the consequences !
    - initialization of the devices
  - Prog. env.. OO (C++, Java, Python)
    - indirect way (cf. TANGO doc)
  - User env. (Matlab, Igor Pro, …)
    - prop_val = tango_get_property (dev_name, prop_name)
    - prop_val_list = tango_get_properties (dev_name, prop_name_list)
    - prop_val = tango_put_property (dev_name, prop_name, prop_val)
    - prop_val_list = tango_put_properties (dev_name, prop_name_list)
    - tango_del_property (dev_name, prop_name)
    - tango_del_properties (dev_name, prop_name_list)
  - example Matlab
    - `>> help tango_get_property(dev, ‘mthreaded_impl’)`
    - `>> tango_get_property(dev, ‘mthreaded_impl’)`

TANGO : device

TANGO : device : summary

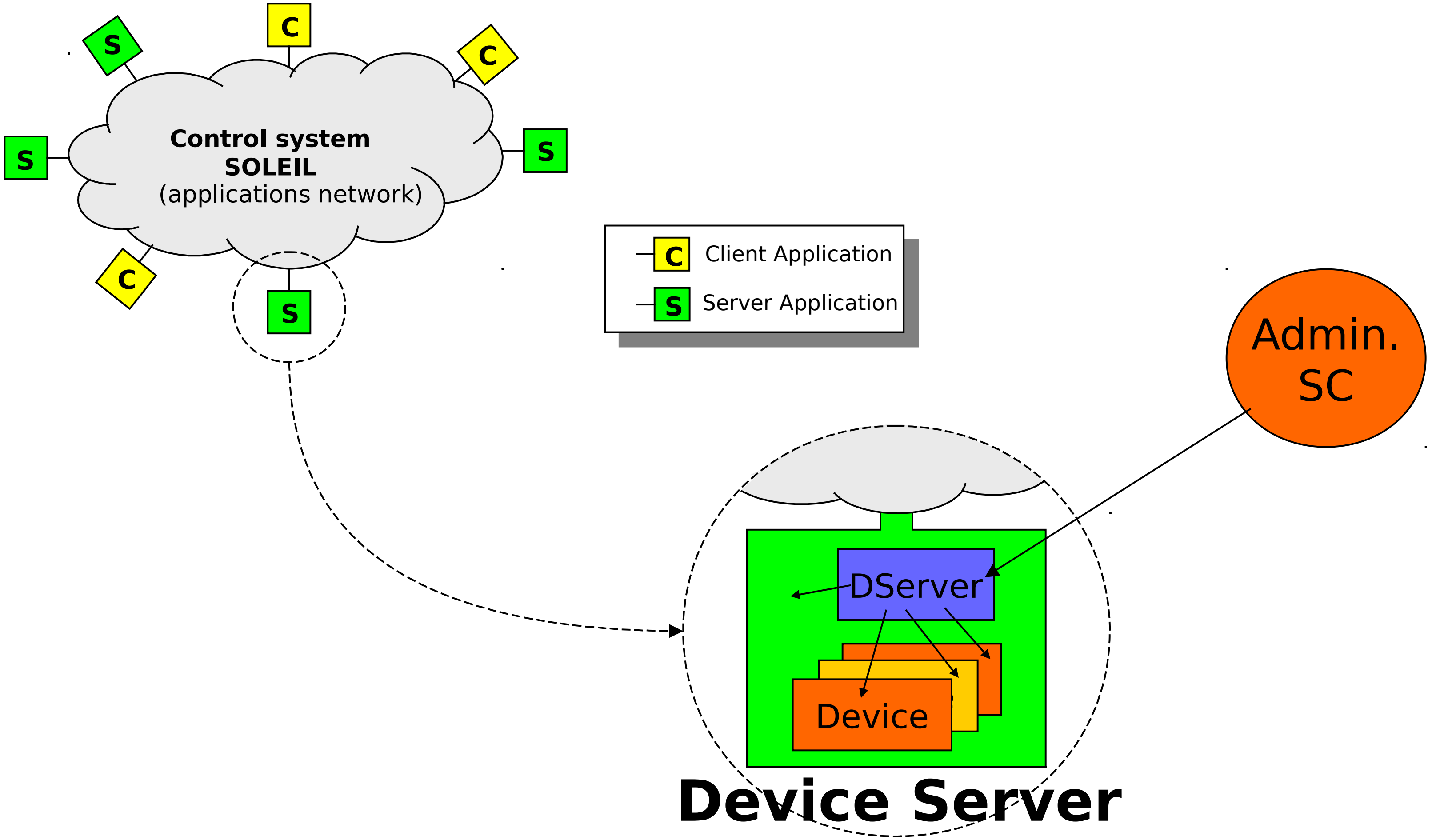


Where to store the properties and all the configuration data ?

TANGO : database (static)

- Database of the configuration
  - critical element of the system
  - The only information source for the devices and the clients
- Content : 7 tables
  - server : infos related to the servers (admin. du SC)
  - device : infos related to the devices (IOR=@particularly the network)
  - property : global properties associated to SC
  - property_class : properties associated to a class of devices
  - property_device : properties associated to a particular device
  - property_attribute_class : properties associated to an attribute (for any device)
  - property_attribute_device : properties associated to an attribute of a particular device
- Implementation
  - 1 dedicated device = interface TANGO of a SGBD
  - TANGO_HOST = host_name:host_port (ex: localhost:20000)

TANGO : device server



TANGO

TANGO system

≈

{Device Servers {Devices}}

+

Static DB

TANGO : APIs and platforms

- APIs/Programming Languages
  - C++ (performances)
  - Java (portability)
  - Python (scripts)

    Serveurs
  - Others (Matlab, Igor Pro, LabView)

    Clients
- platforms
  - Linux
  - Windows NT/2000/XP
  - Sun-Solaris

Questions…

TANGO : Java Tools

- LogViewer
  - Management of messages generated by the devices
- DeviceTree
  - Generic Client : tests, monitoring, …
- Jive
  - Administration of the TANGO database
- Pogo
  - Code generator (dev. devices)