TANGO CONTROLS CONCEPTS

A brief introduction to the Tango Controls Concepts

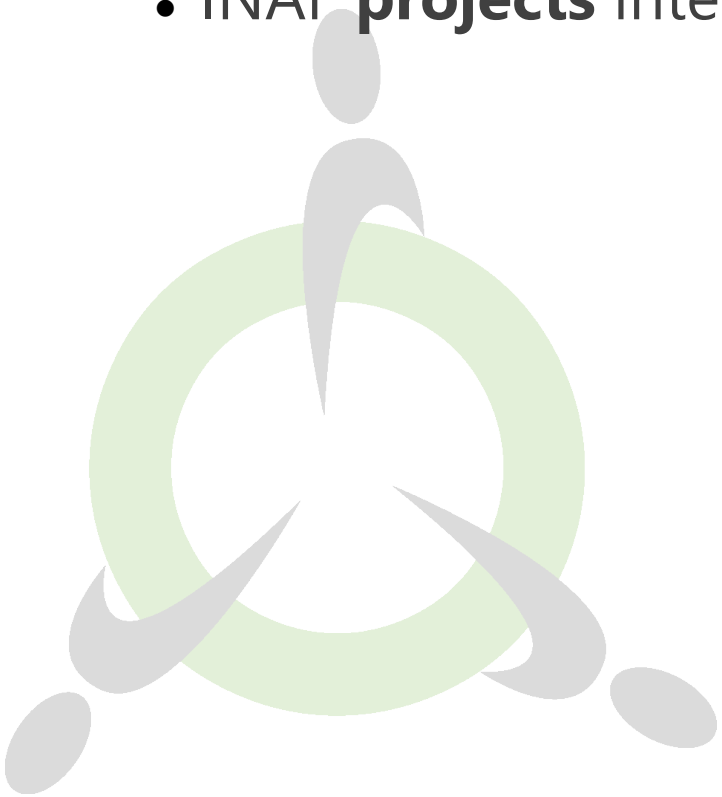
by

Matteo Canzari

INAF - Osservatorio Astronomico d'Abruzzo

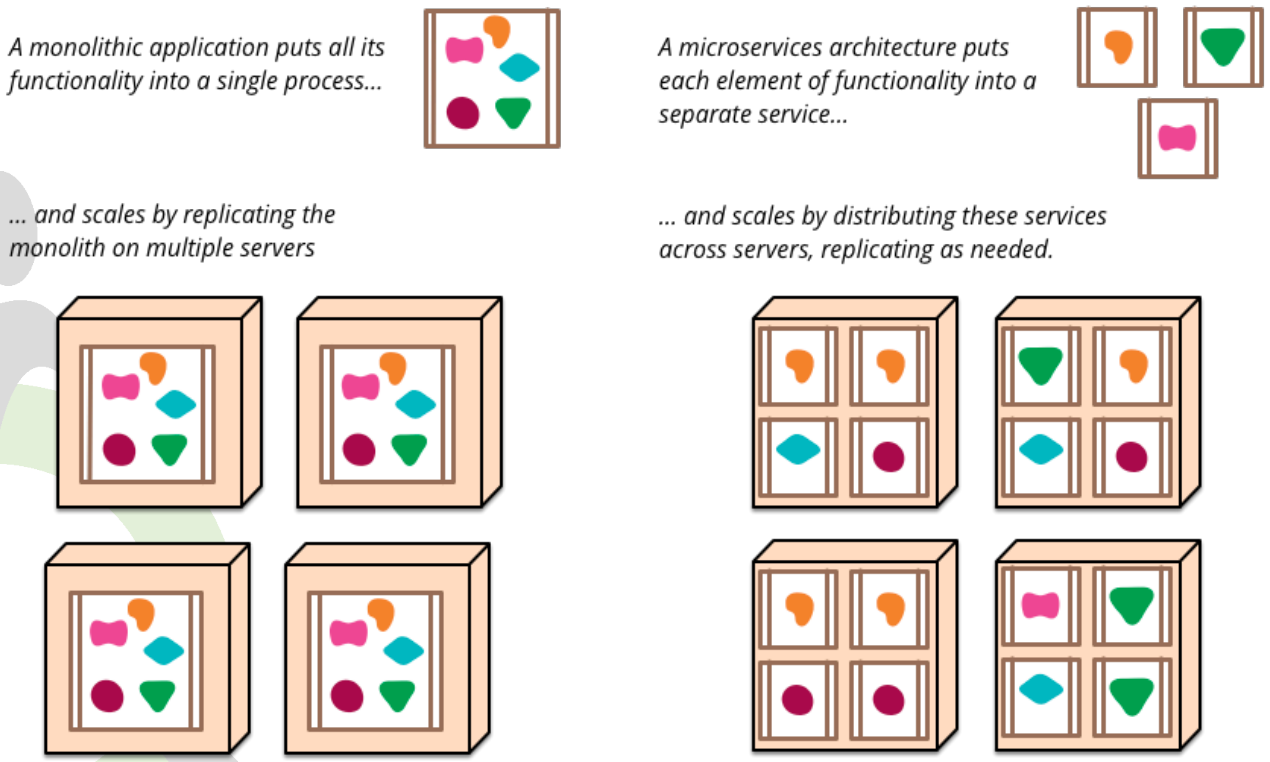
Summary

- Tango Controls Framework **concepts**
- Tango **Community** and Tango **Collaboration**
- The role of **INAF** in Tango
- INAF **people** involved
- INAF **projects** interested in TANGO

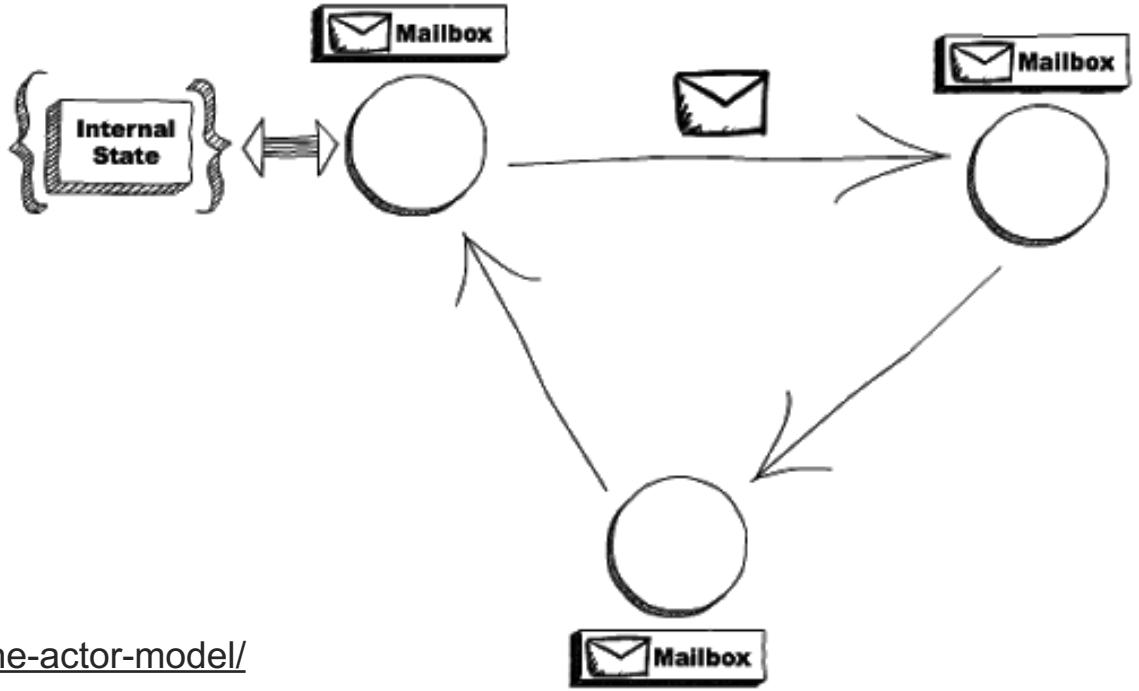


Microservice Architecture

Microservice is an approach to developing a single application as a suite of **small services**, each running in its **own process** and communicating with lightweight mechanisms, often an HTTP resource API



Actor model concept



http://www.brianstorti.com/the-actor-model/

The actor model in computer science is a mathematical model of concurrent computation that treats "actors" as the universal primitives of concurrent computation. In response to a message that it receives, an actor can: **make local decisions**, **create more actors**, **send more messages**, and determine how to respond to the next message received. Actors may modify private state, but can only affect each other through messages (avoiding the need for any locks).

Proposed in 1973 by **Carl Hewitt and others**

10 things you should know about CORBA

1.You don't need to know CORBA to work with TANGO
2.CORBA is the acronym for **C**ommon **O**bject **R**equest **B**roker **A**rchitecture and it is a standard defined by the Object Management Group (OMG)
3.CORBA enables communication between software written in different languages and running on different computers
4.CORBA applications are composed of many objects; objects are running software that provides functionalities and that can represent something in the real world
5.Every object has a type which is defined with a language called IDL (Interface Definition Language)
6.An object has an interface and an implementation: this is the essence of **CORBA** because it allows *interoperability*.
7.CORBA allows an application to request an operation to be performed by a distributed object and for the results of the operation to be returned back to the application making the request.
8.CORBA is based on a *Remote Procedure Call* model
9.The TANGO Device is a **CORBA** Object
10.The TANGO Device Server is a **CORBA** Application

https://tango-controls.readthedocs.io/en/latest/development/corba.html

0mq

0mq (or ZeroMQ, ØMQ, ZMQ)

- high-performance **asynchronous messaging library**
- used in **distributed** or **concurrent** applications
- provides a **message queue**

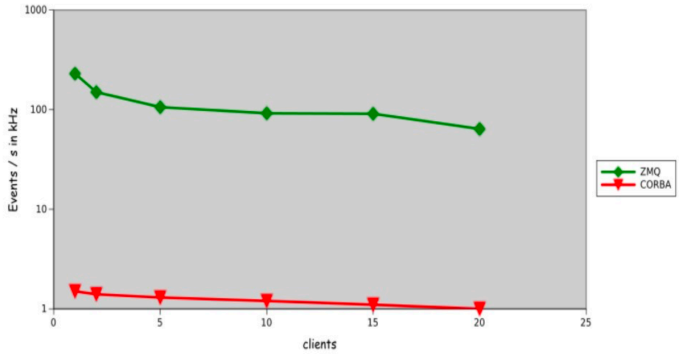
TANGO controls

TANGO and synchronous & asynchronous communication

Tango uses:

- CORBA® for **synchronous** communications (messages between actors)
- ØMQ for **asynchronous** communication (events)

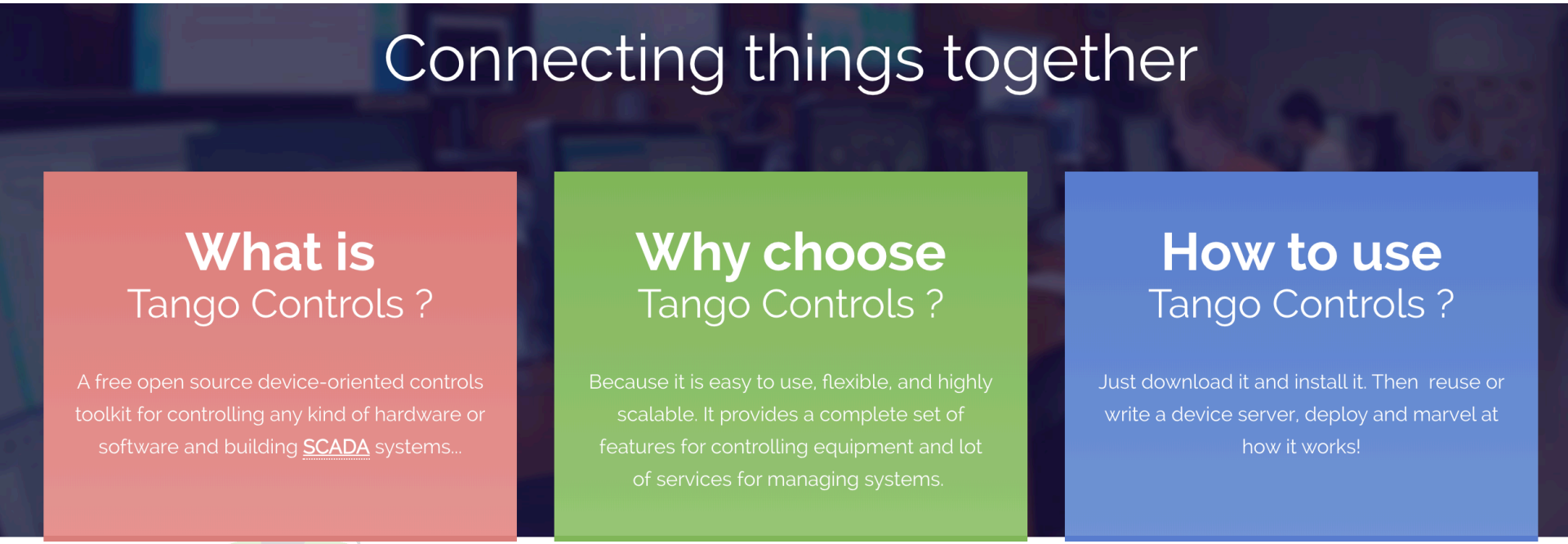
TANGO – CAN ZMQ REPLACE CORBA?
Proceedings of ICALEPCS2013, San Francisco, CA, USA
https://accelconf.web.cern.ch/ICALEPCS2013/papers/tucocb07.pdf



TANGO Events with ZMQ
Xeon 3 GHz
(unix socket)

server to client	data	throughput
Java to java	8 bytes	14.2 k Ev/s
C++ to C++	8 bytes	230 k Ev/s
C++ to C++	1 kbyte	122 k Ev/s
C++ to C++	1 Mbyte	1.2 k Ev/s

What is TANGO

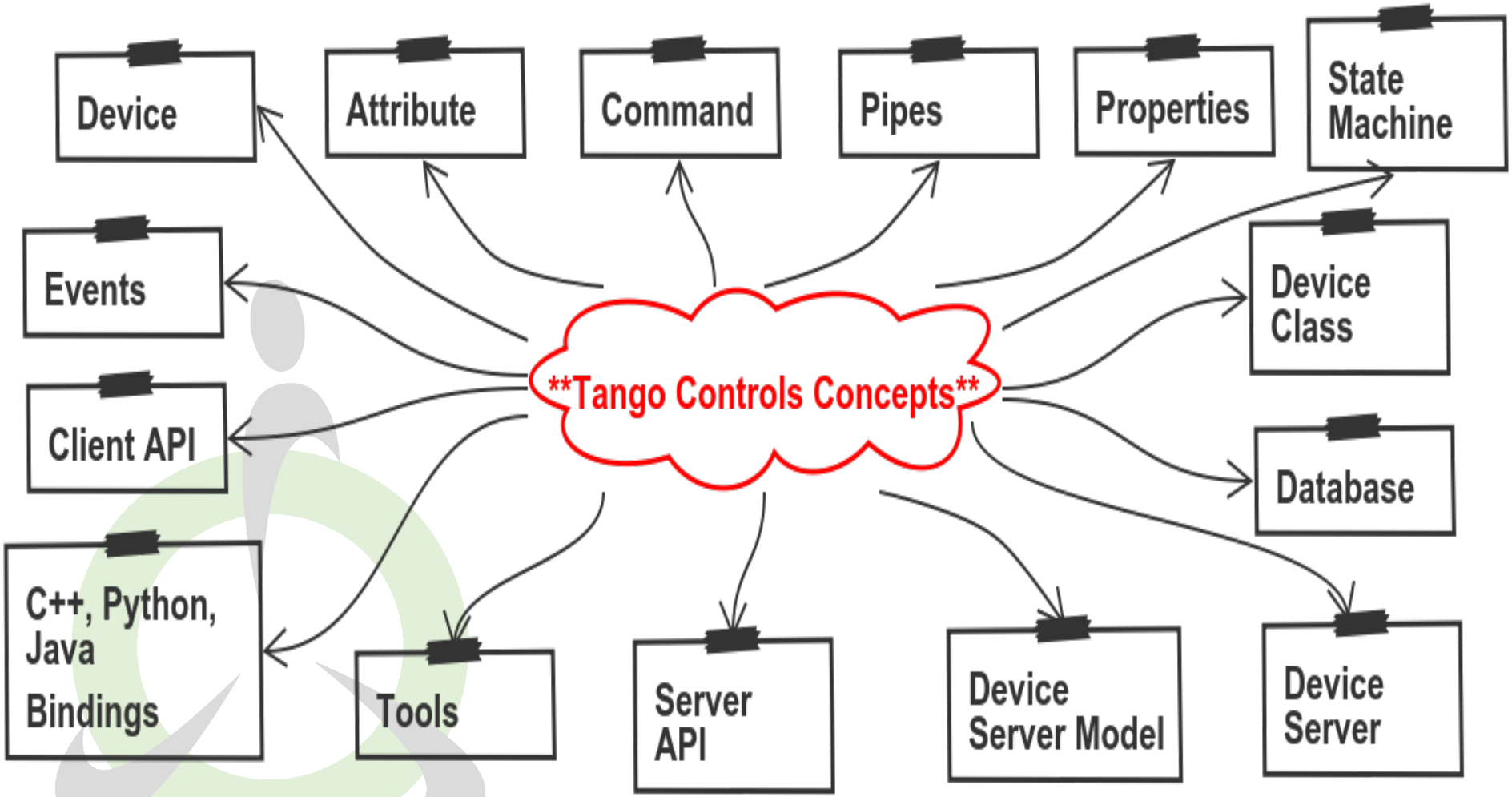


- **Tango** is an **open-source device-oriented** controls toolkit for controlling any kind of hardware and software and building SCADA and DCS
- Born in **1999**
- Mostly used in **Synchrotron**, but also in **industry** and now in **radio astronomy**

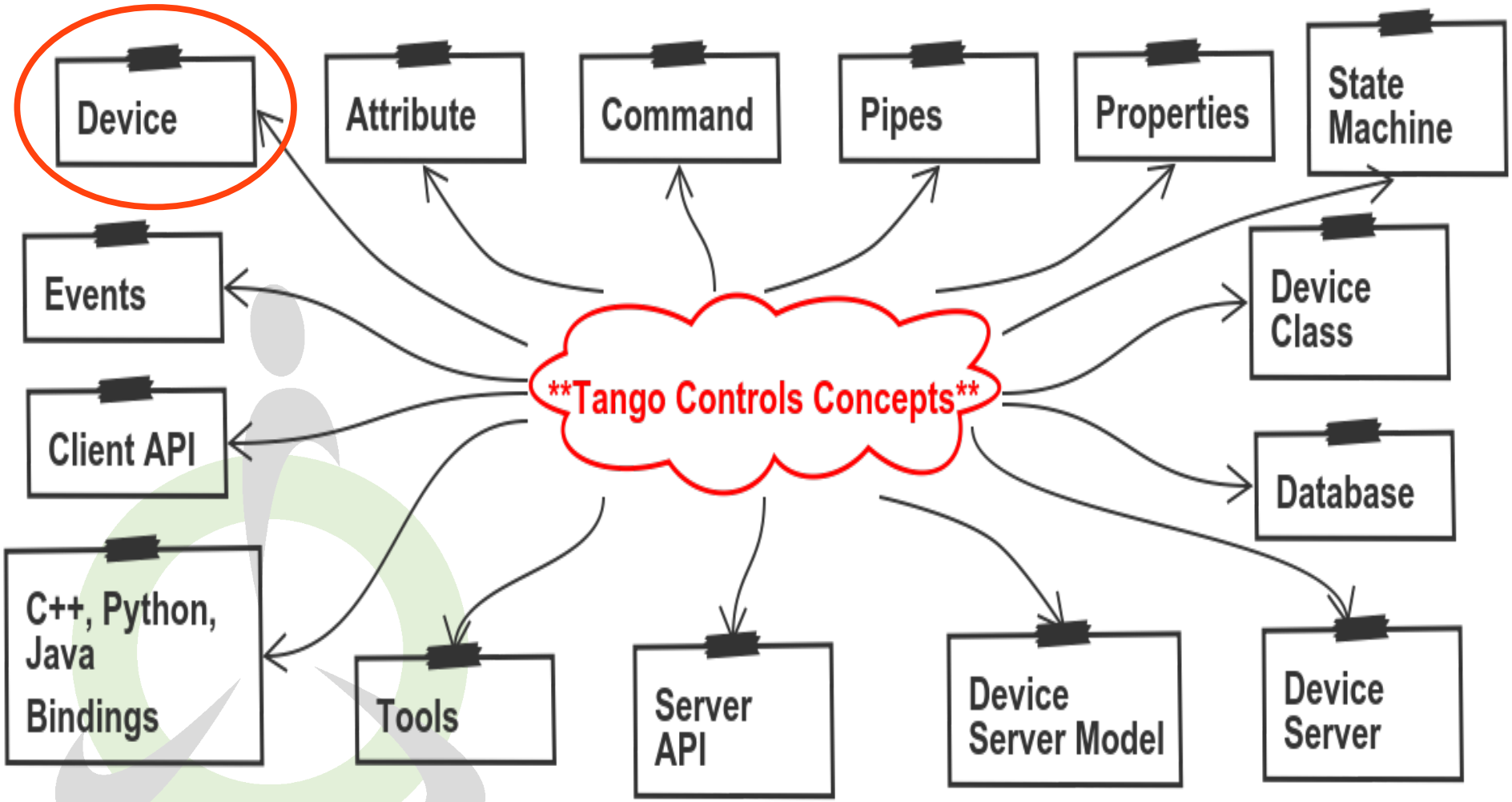
Tango = **actors + microservices**

- Tango is based on the concept of **Distributed Devices**
- This is an implementation of the **Actor Model**
- Device servers implement **Microservices**
- **Tango** = **Actors** + **Microservices**
- Actors + Microservices are in **fashion** today
- TANGO is based on **MODERN** concepts !

Tango basic concepts



Device concept #1



Device concept #1

- Tango **Devices** are the **objects** which implement the microservices of a Tango System. Devices can be **any piece of hardware or software**.

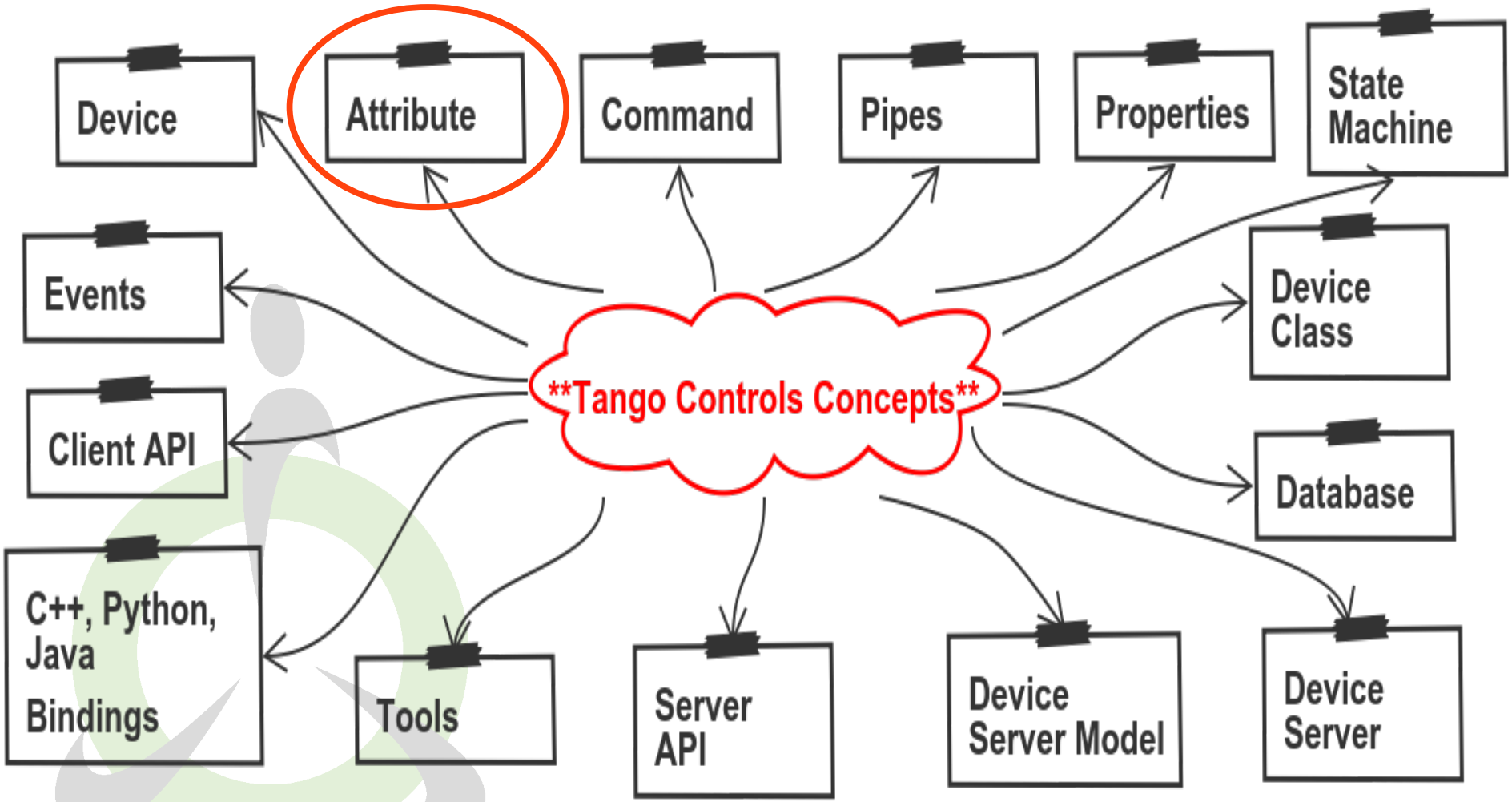
- Examples : *Modbus controller, motor, powersupply, camera, data analysis service, …*

- **Devices** belong to a **Device Class** and are in a **Device Server**. They are stateful i.e. have State. Accessed via a common API. Have a unique 3 field name (D/F/M)

- Device Classes can be implemented in **Python**, **C++** or **Java**

- Devices can be built on top of other Devices

Attribute concept #2



Attribute concept #2

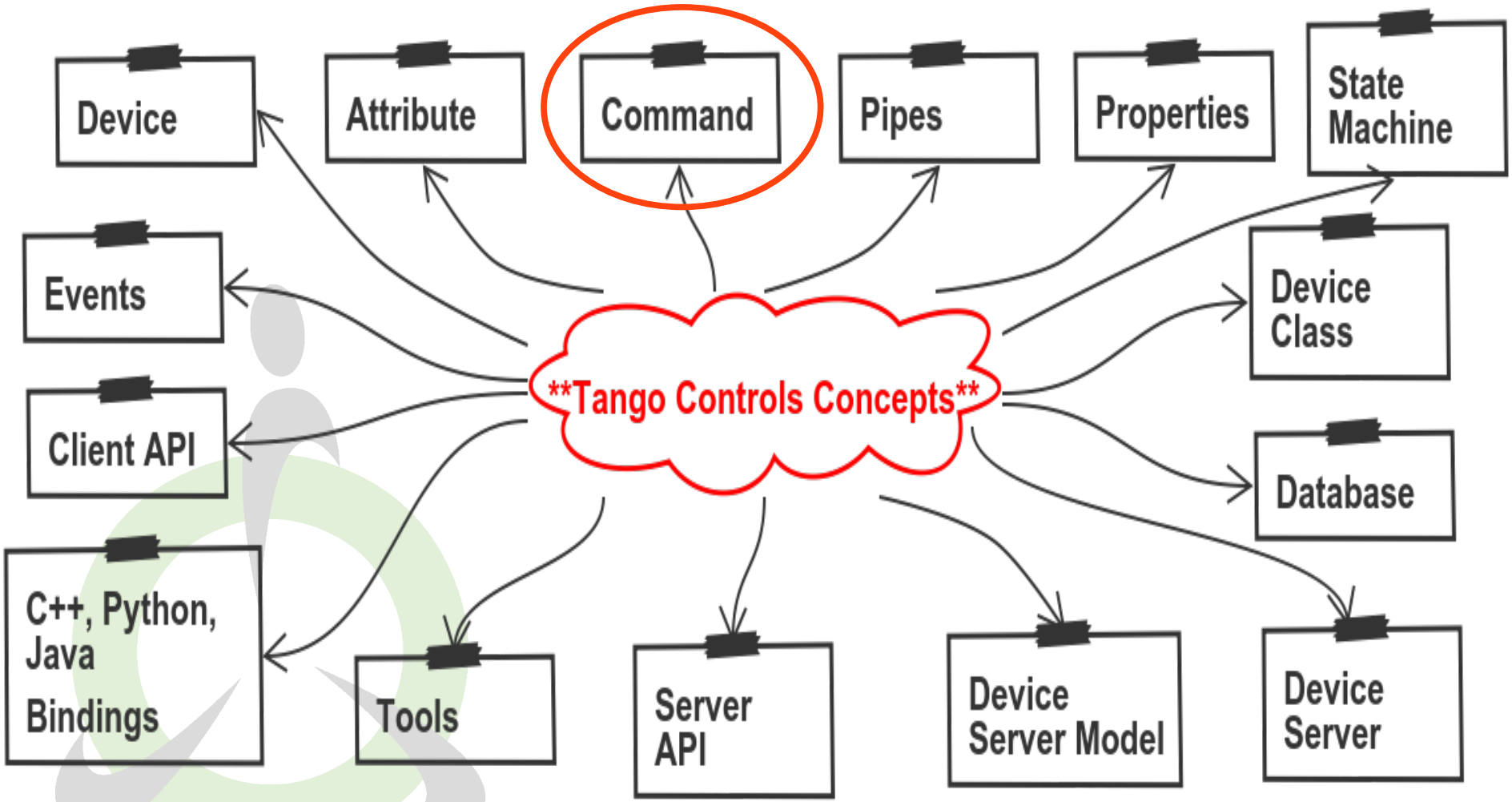
- Tango **Attributes** represents the **data fields** a Device wants clients to **Read** or **Write** or receive **Events**.

- Examples : *modbus register, interlock bit, read/set value, spectrum, image, …*

- Attributes can be **scalar**, **spectrum** (1D) or **images** (2D) and are **self describing** (units, min, max, alarms, display,…)

- **All** Device **data** should be provided **as attributes** (well almost all!). Attributes can be read one by one or many. Device developers have hooks for optimising attributes. Attributes read/write check the State Machine.

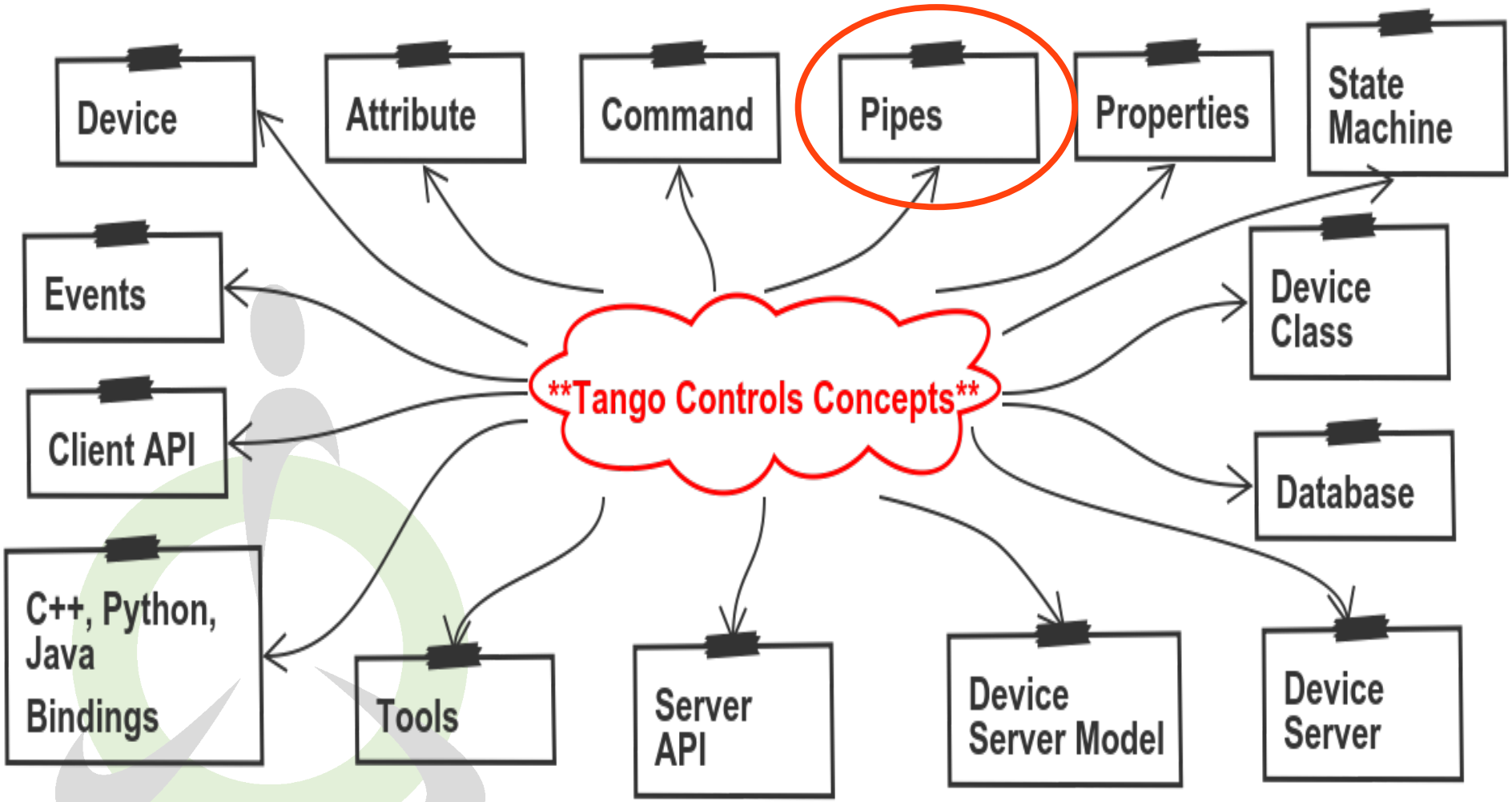
Command concept #3



Command concept #3

- Tango **Commands** are the **actions** of a Device the clients needs to execute. Commands can change the State of a Device (Attributes don't)
- Examples : *On, Off, Calibrate, Move, …*
- Commands take **one input** and **one output** parameter. Parameters can be of any of the 20+ Tango data types.
- Commands always **check** the **State Machine** before and after execution (Attributes only before).

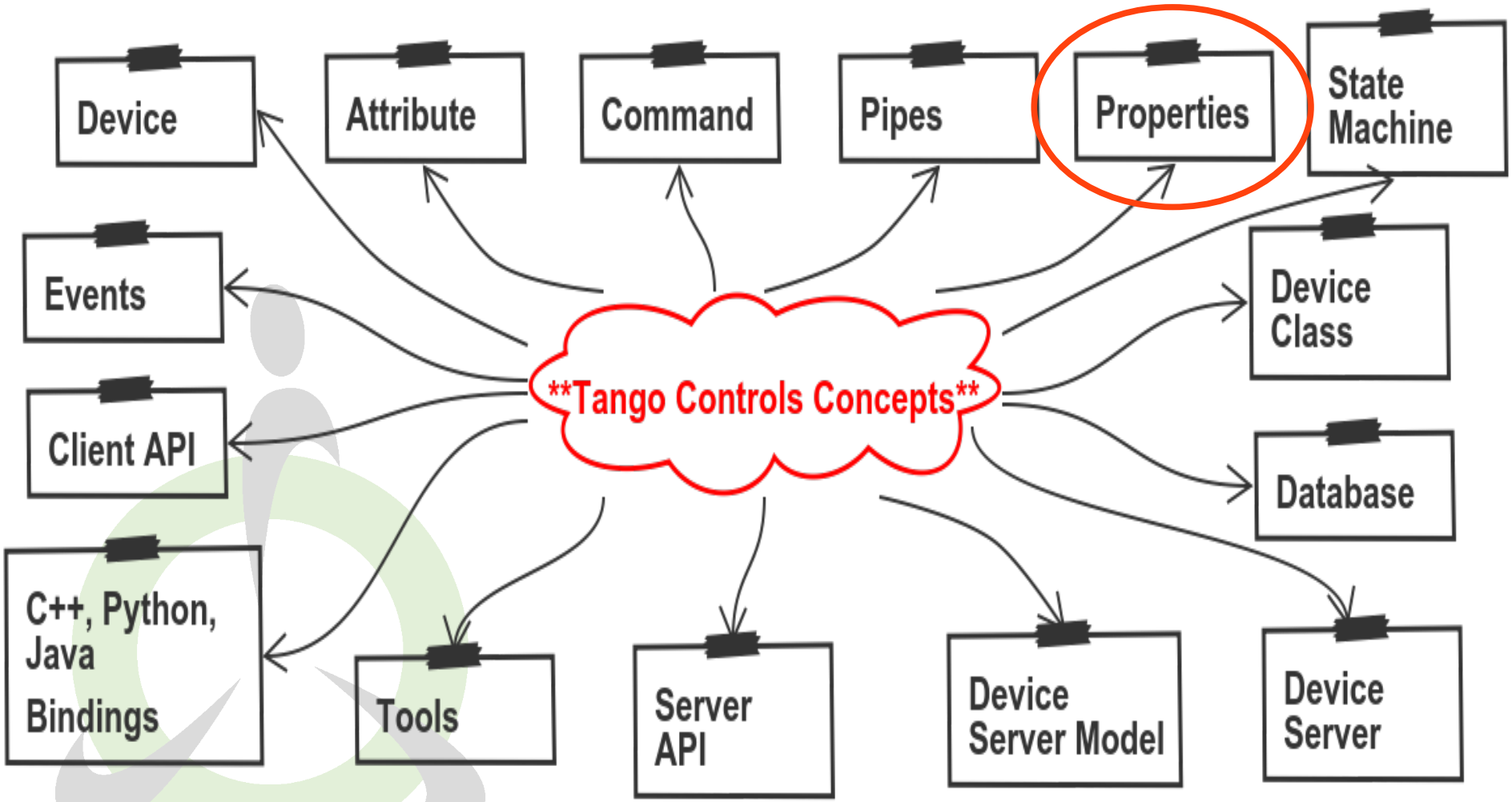
Pipe concept #4



Pipe concept #4

- Tango **Pipes** are **data streams** or channels for exchanging a stream of any Tango data type. Data types can be sent individually or grouped together in a Blob.
- Examples : *scanning data stream of mixed data types*
- Also used to circumvent the fixed data type set of Tango by sending mixed data types or a JSON blob.
- DO NOT only use Pipes (except in special cases)! Pipes were added to Tango in V9.x
- DO use Attributes!

Properties concept #5



Properties concept #5

- Tango **Properties** are **data stored** in the database and used to **configure** Devices at **startup**. Properties can be any Tango Data types. Properties enable Device Classes to be generic. Properties are edited with Jive usually.

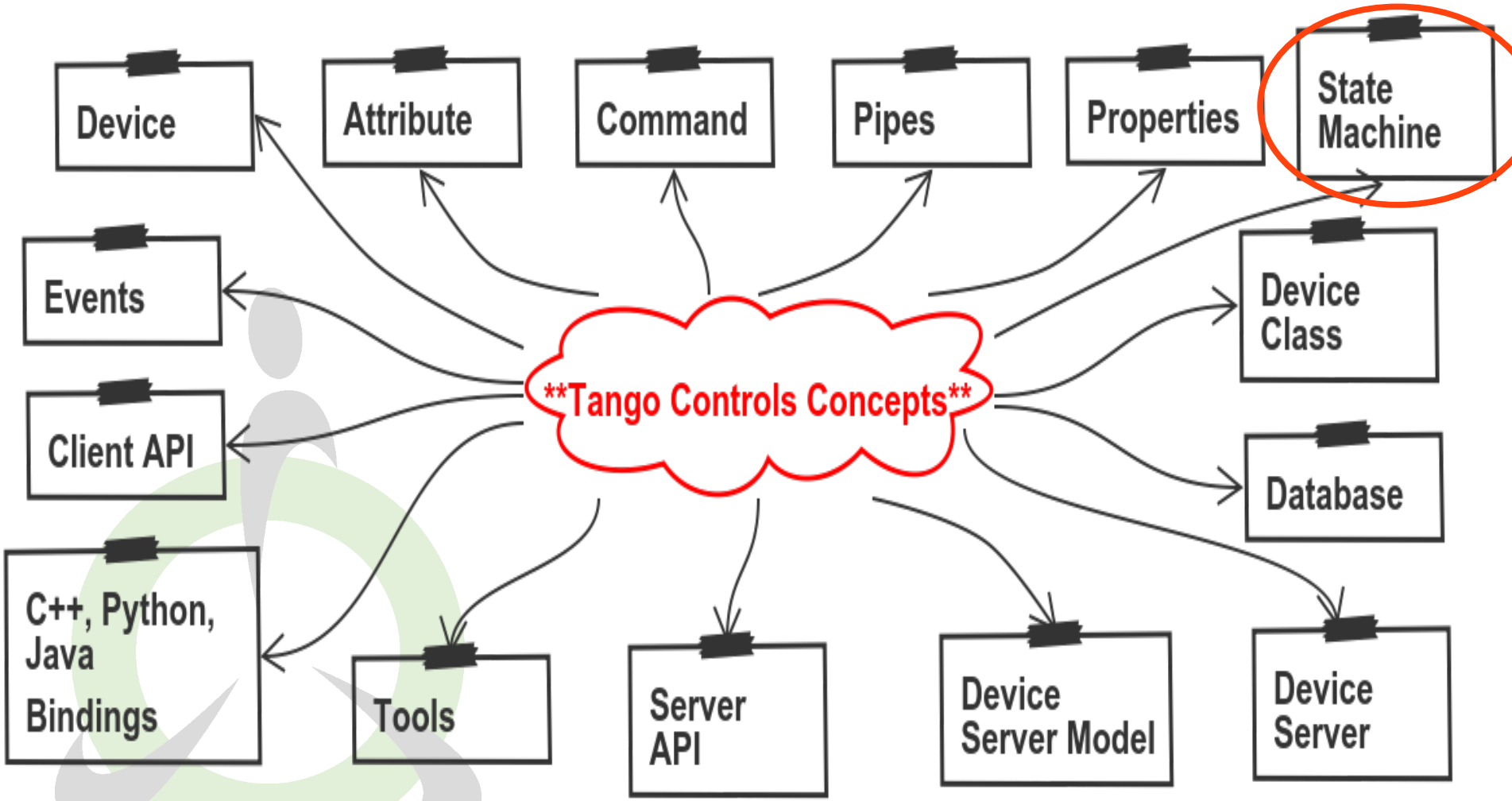
- Examples : *channel address, initial or current settings, sub-device names, ...*

- Changes to Properties can be persisted in the Database.

- DO NOT exit if Properties are wrong!

- DO use sensible default Properties!

State machine concept #6



State machine concept #6

- All Tango Devices have **State**. Tango States are limited to 14 discrete values. Each Tango Device Class **State Machine** implements the state transitions.

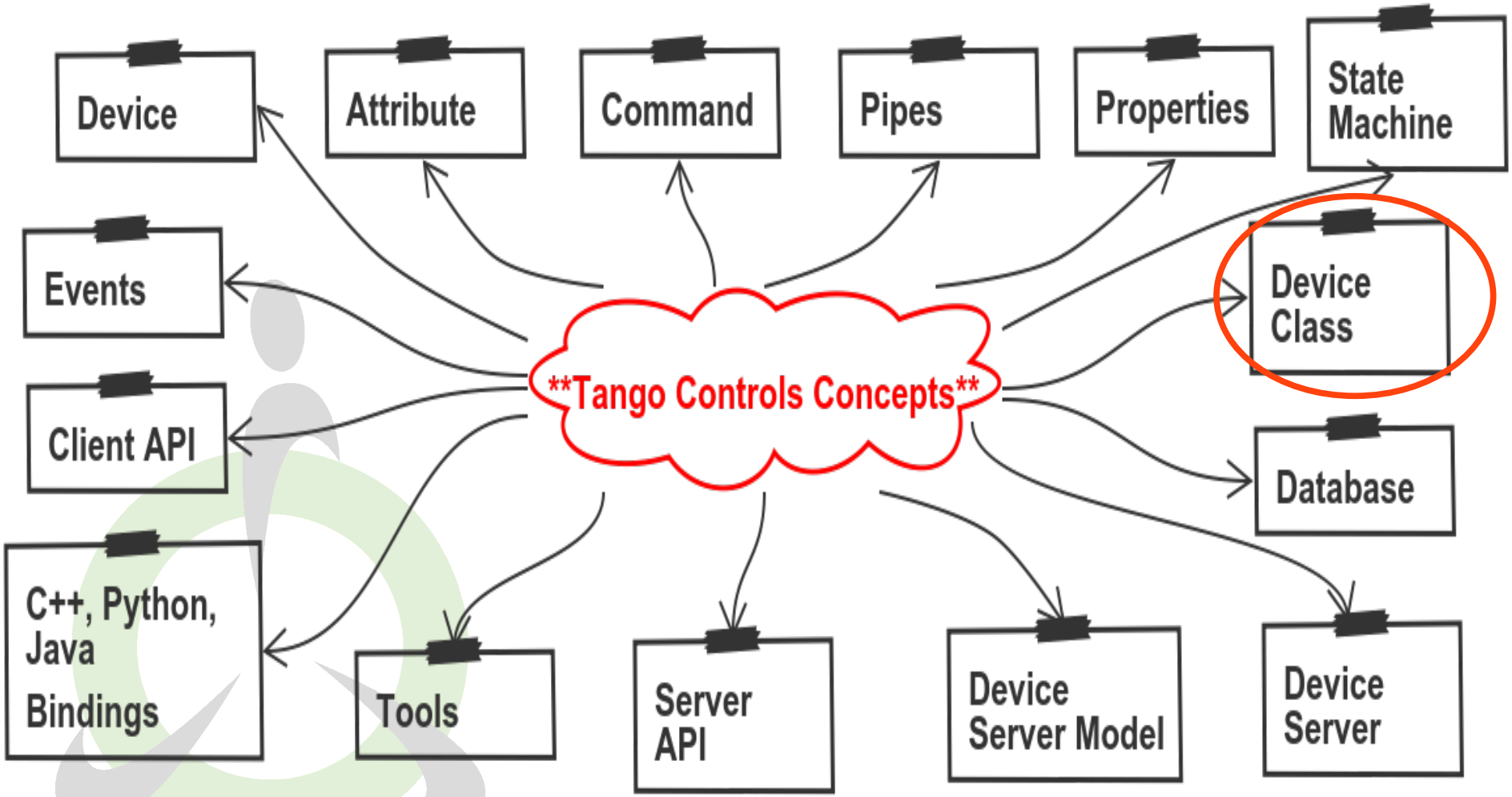
- **ON, OFF, CLOSE, OPEN, INSERT, EXTRACT, MOVING, STANDBY, FAULT, INIT, RUNNING, ALARM, DISABLE**, and **UNKNOWN**

- State is a very powerful mechanism for **protecting** Devices and for communicating changes to clients or servers.

- DO NOT ignore State !

- DO set a default State!

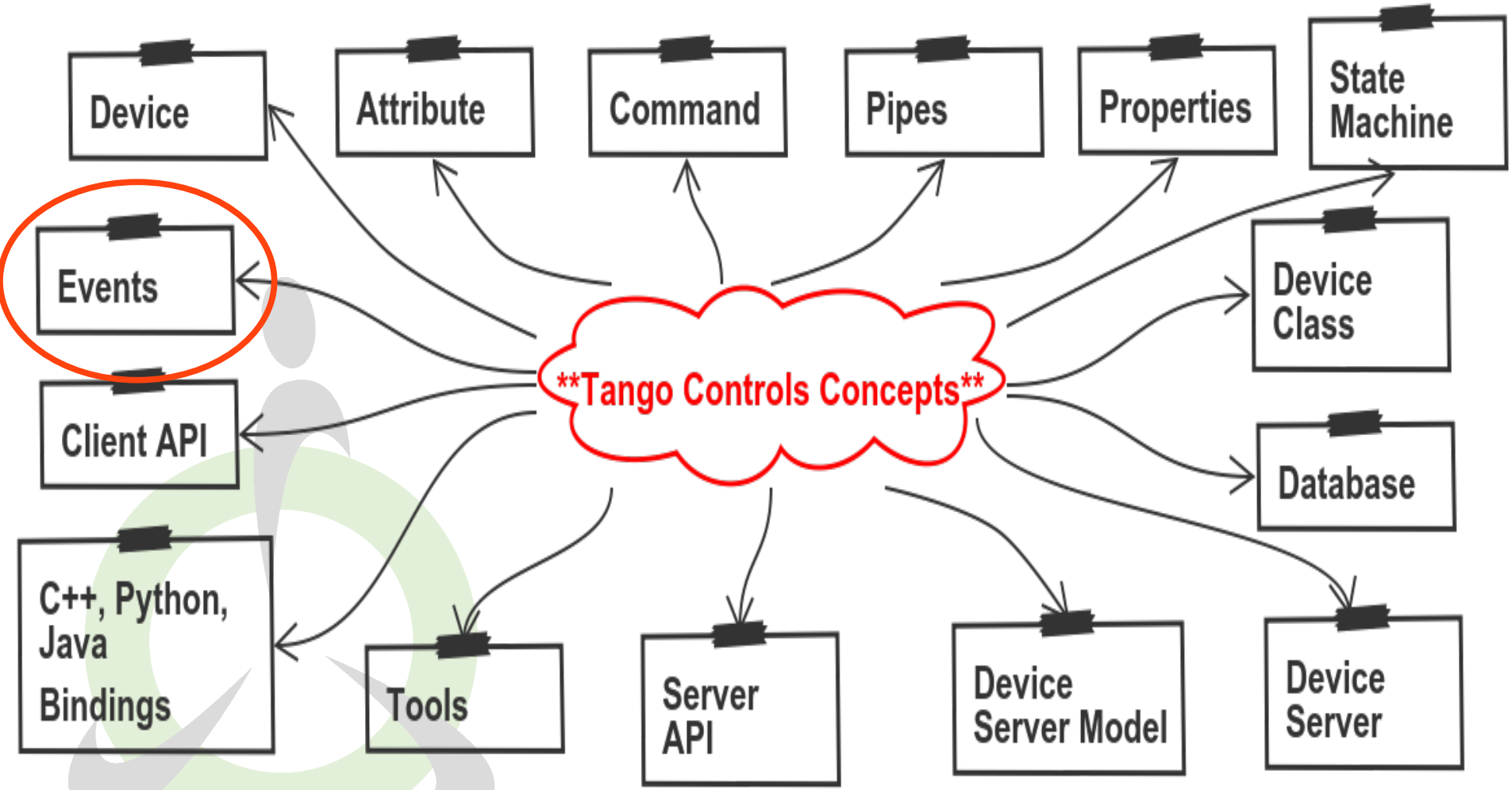
Device class concept #7



Device class concept #7

- All Tango Devices are implement by a **Device Class**. The Device Class implements a **generic** Device **behaviour**. Properties are used to configure the specific Device
- Examples : *PowerSupply, SerialLine, Polly*
- Device Server developers are in fact developing Device Classes
- C++ developers have an extra class to develop – the so-called DeviceClassClass e.g. MyPowerSupplyClass. This is uses one of the Gang of Four patterns. Python and Java have only the DeviceClass.

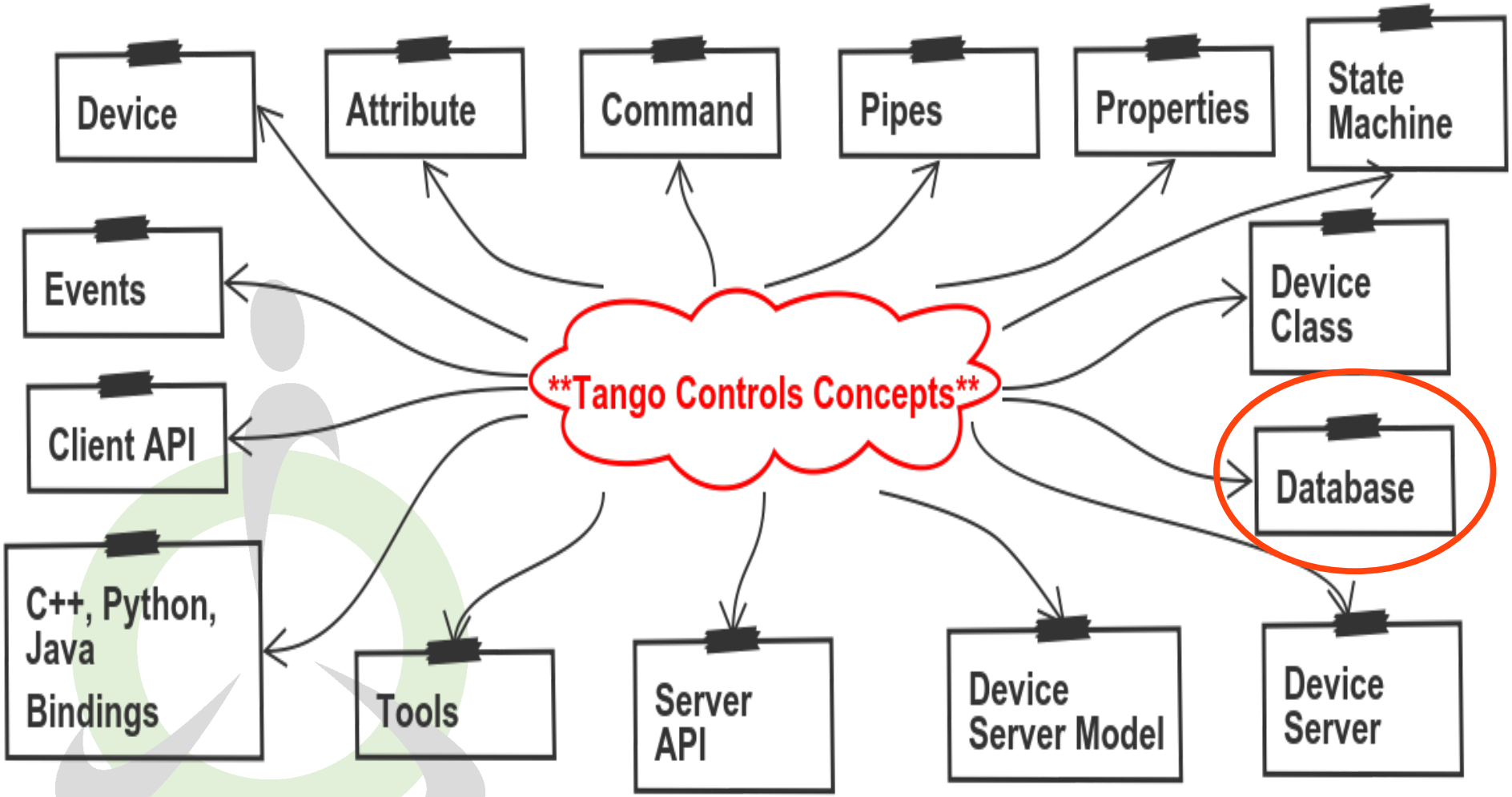
Events concept #8



Events concept #8

- Tango **Events** are a **Pub-Sub** communication between clients and servers. Events are only supported for **Attributes** and **Pipes**. Multiple Event types are supported – **Change**, **Periodic**, **Archive**, **User**, …
- Examples : *Send Event if Attribute changes by x%*
- Events use **ZeroMQ** + are the most efficient way to communicate. Events rely on Polling to be triggered.
- Events are configured in the database or code
- Tango implements a Polling algorithm to trigger events.
- DO NOT only read Attributes, use Events

Database concept #9



Database concept #9

- Tango **Database** implements the **Configuration** and **Naming** Service for Tango. It can also persists **settings** values.

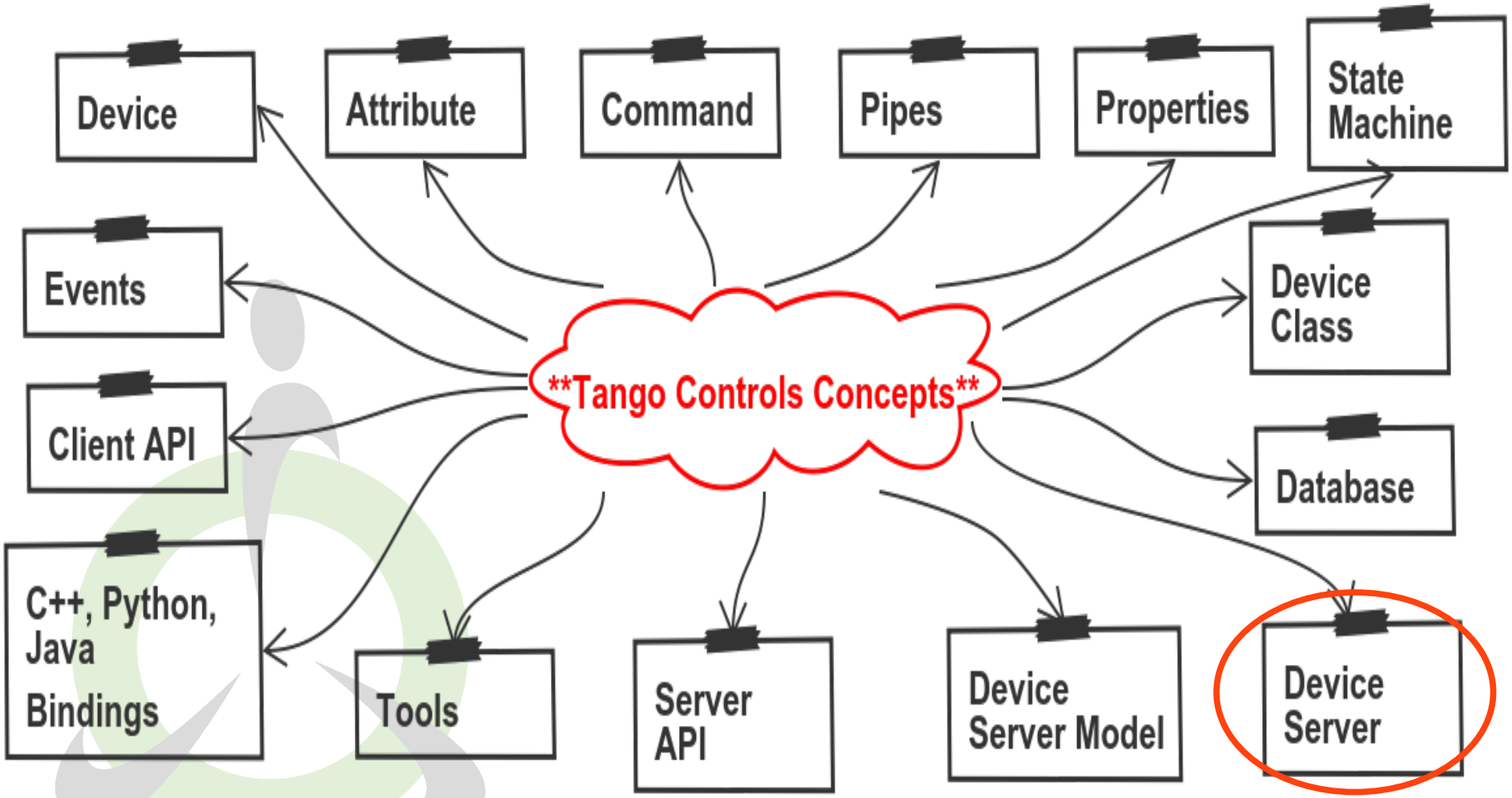
- Examples : *configuration properties, export/import*

- Tango Database is implemented as a Device Server. Clients use the Tango Client API and Data Types to access the Database. Only **MySQL** (or **MariaDB**) is supported. A (non-official) version exists for **SQLite** + yaml. It also possible to use a **single file** (not recommended).

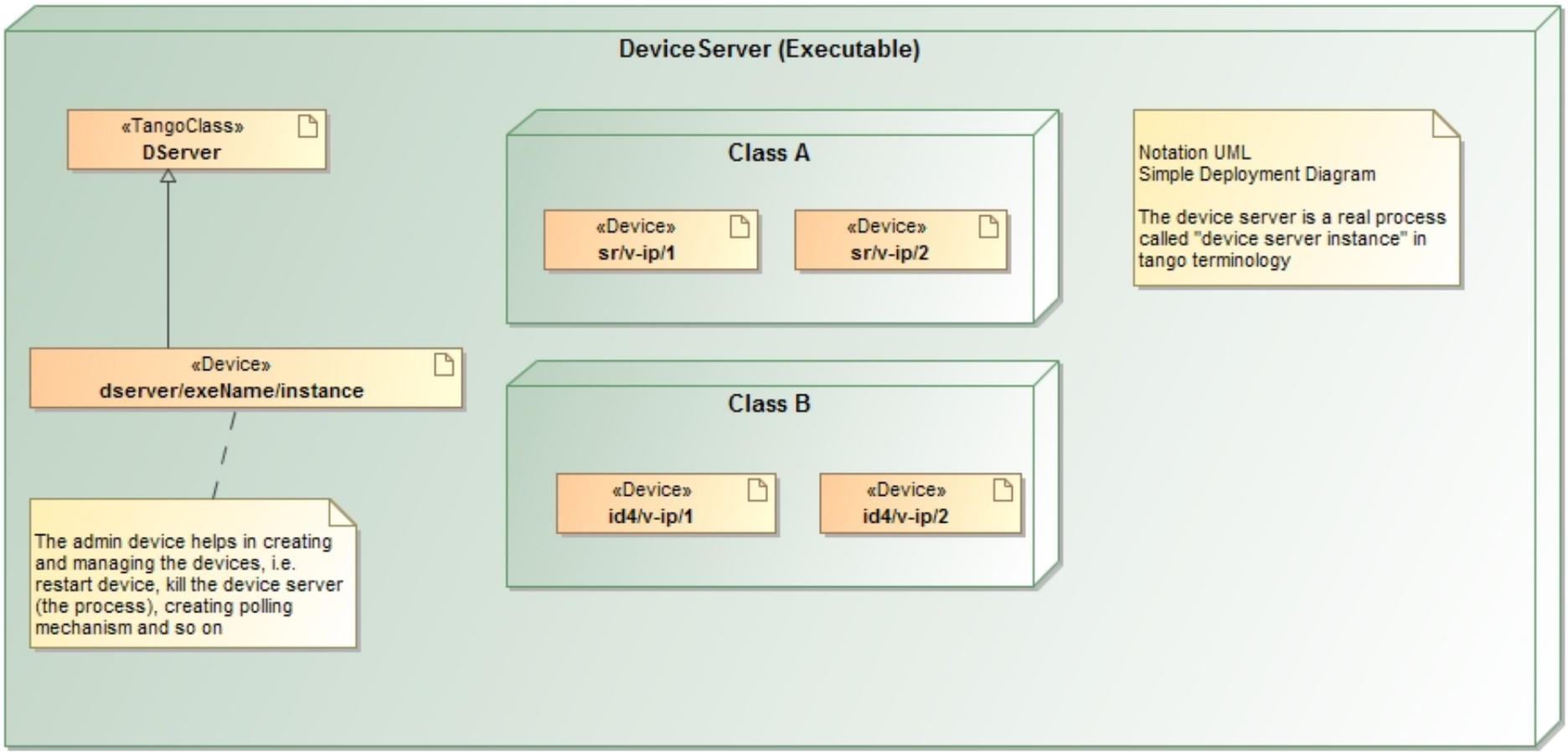
- Database is **only fixed** address `TANGO_HOST=host:port` or `/etc/tango.rc` environment variable.

- **Multiple** Databases supported.

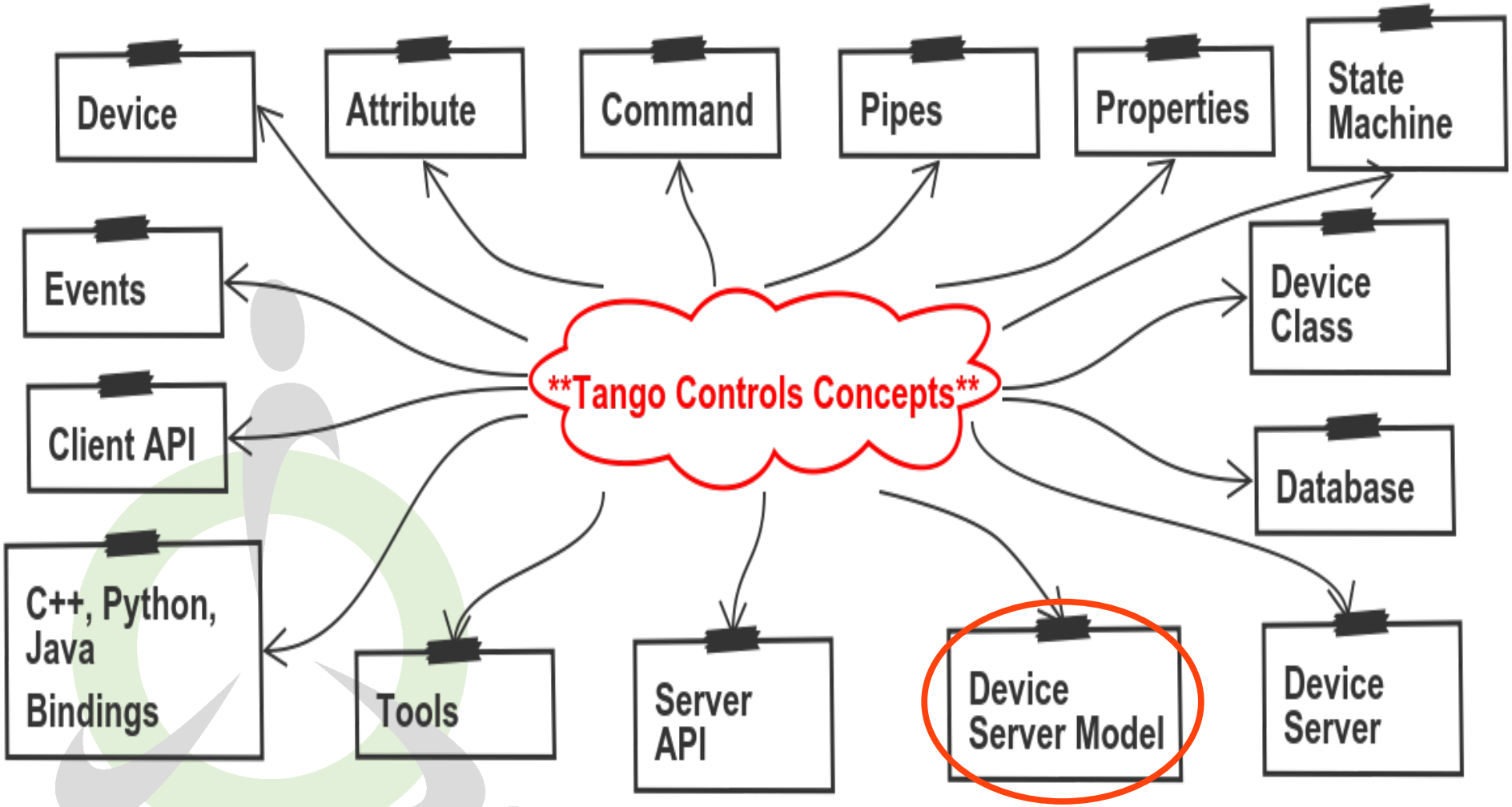
Device server concept #10



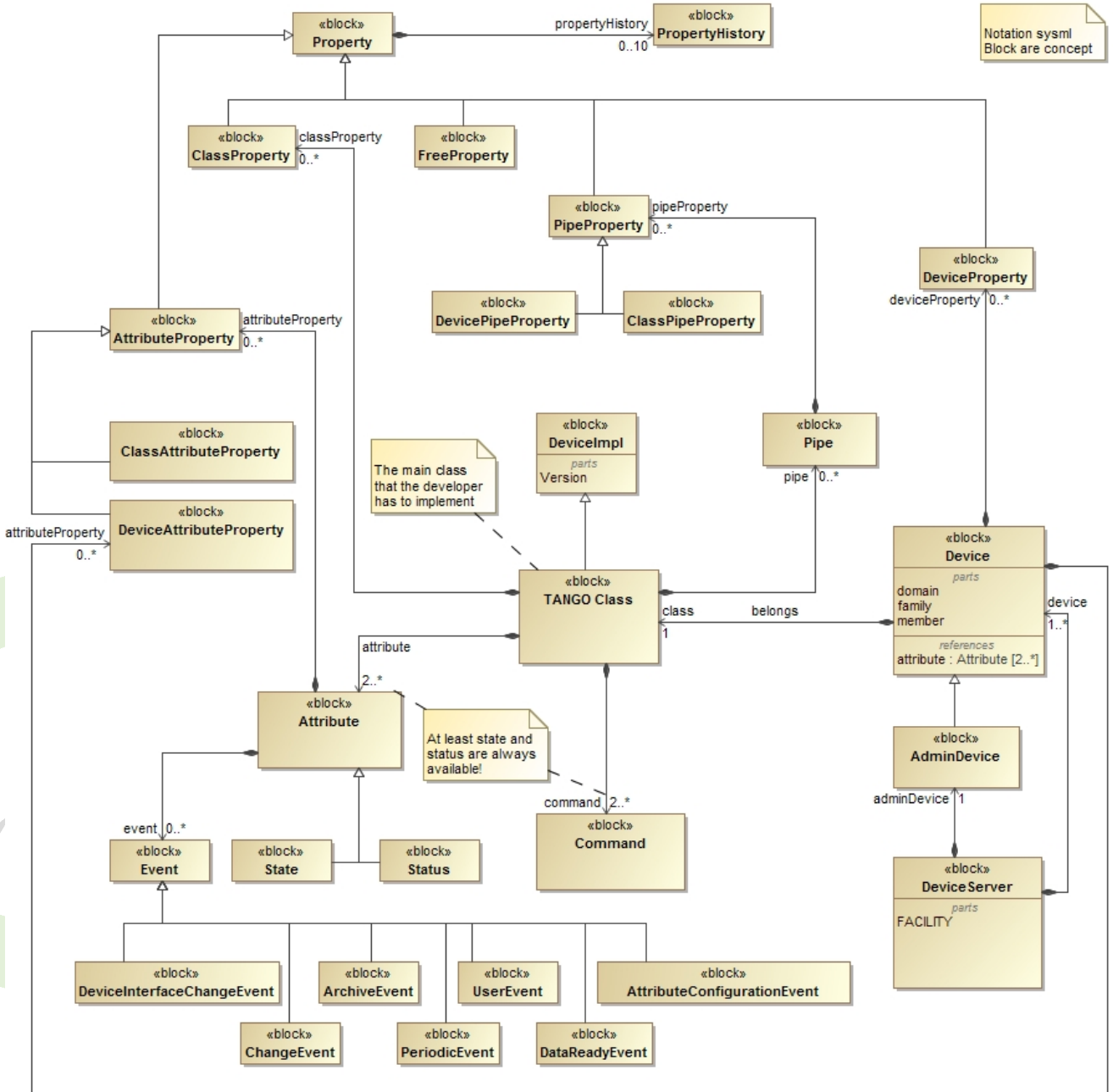
Tango runtime **Device Model**



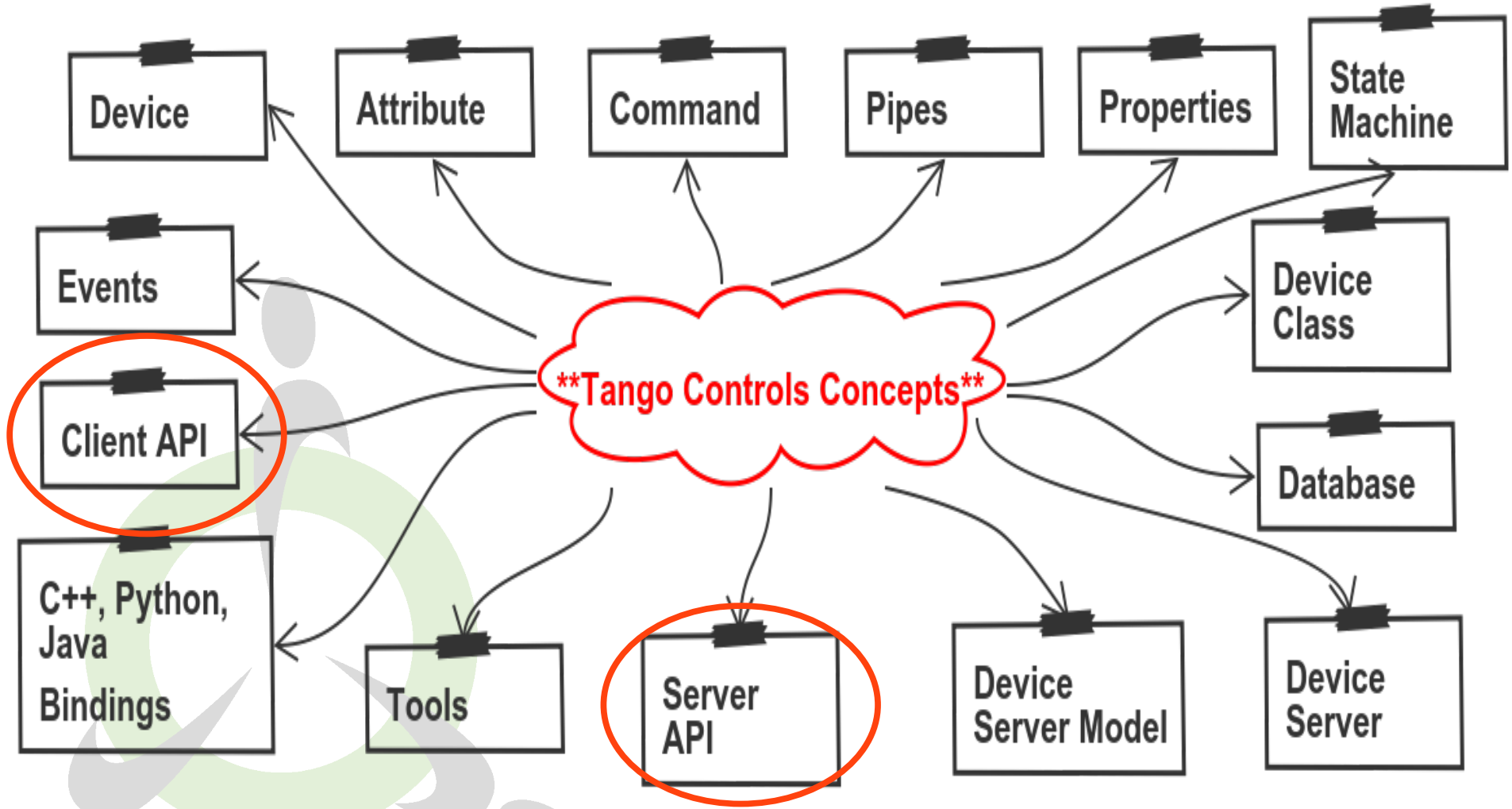
Device server model concept #11



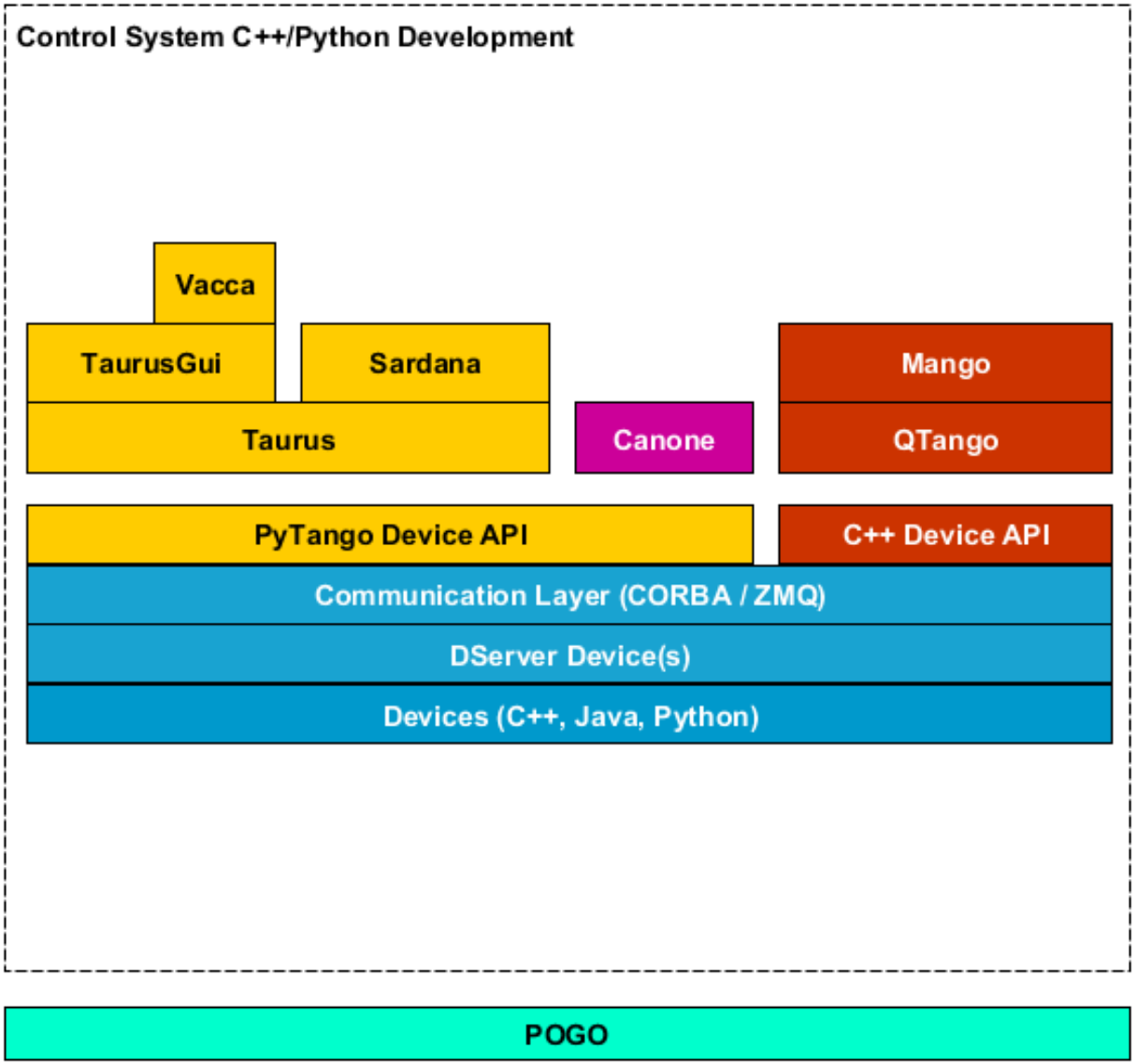
Tango full Device Model



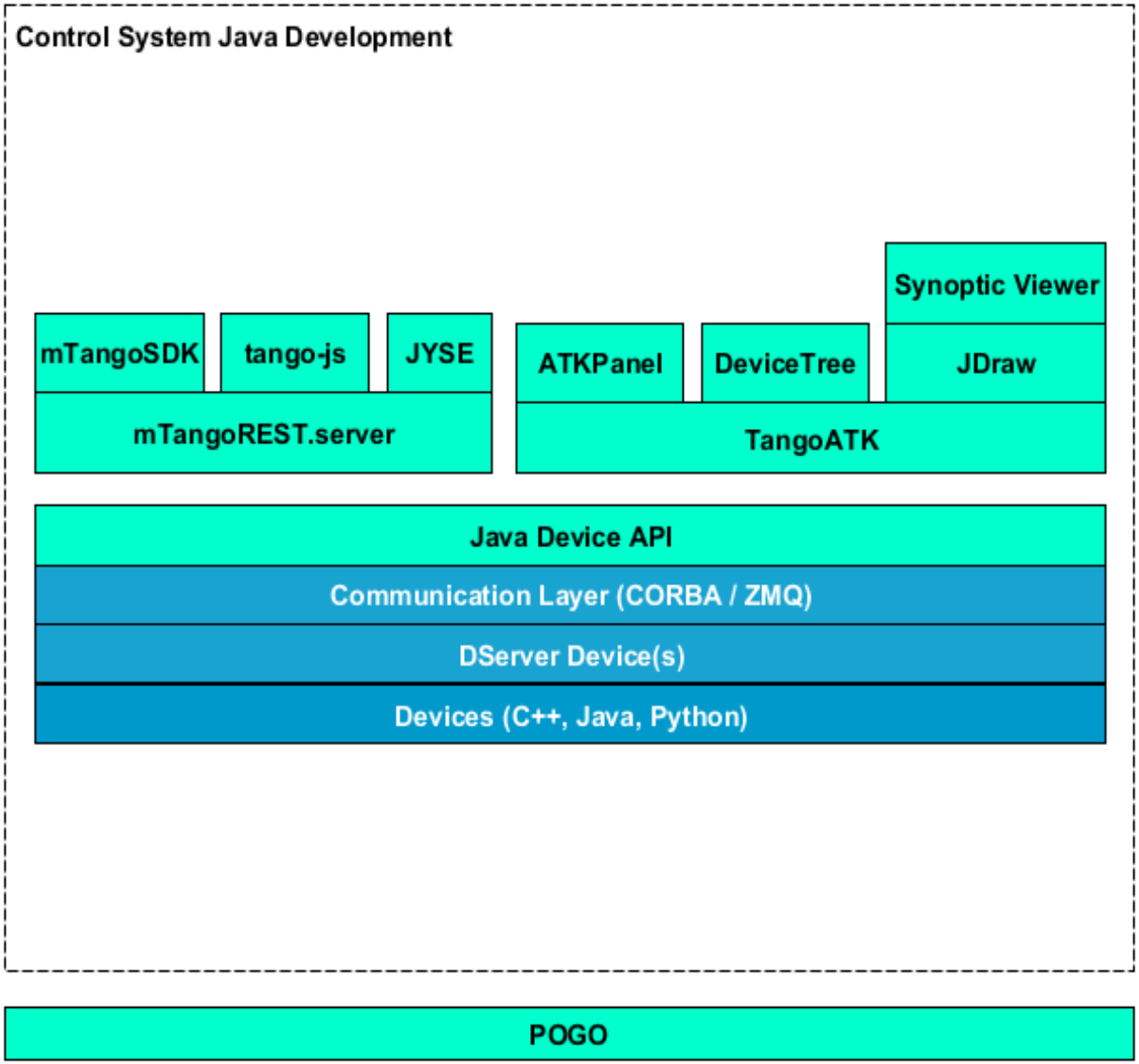
Server + Client api concept #12



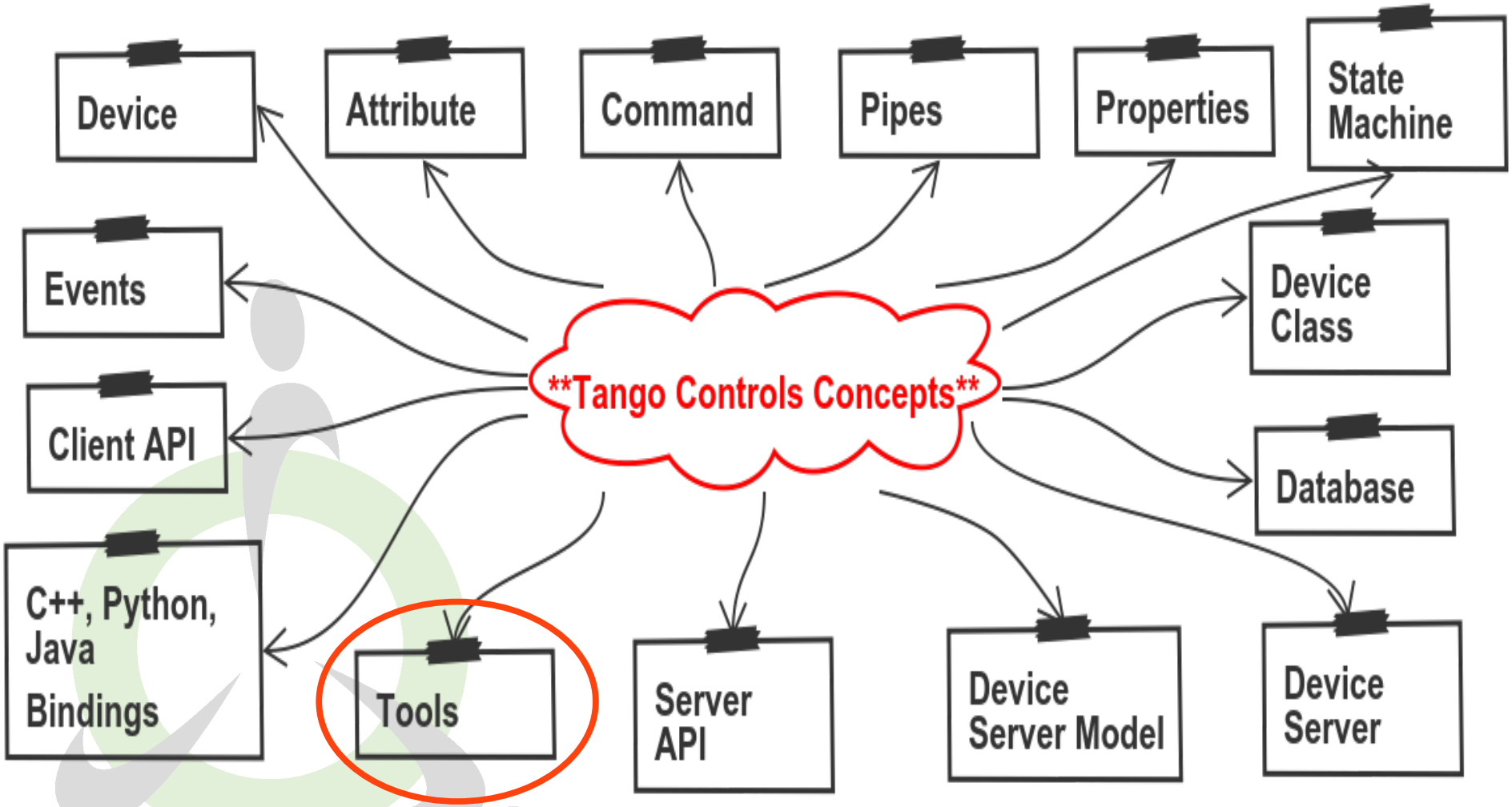
Tango **Developers map**



Tango **Developers map**



Tools concept #13

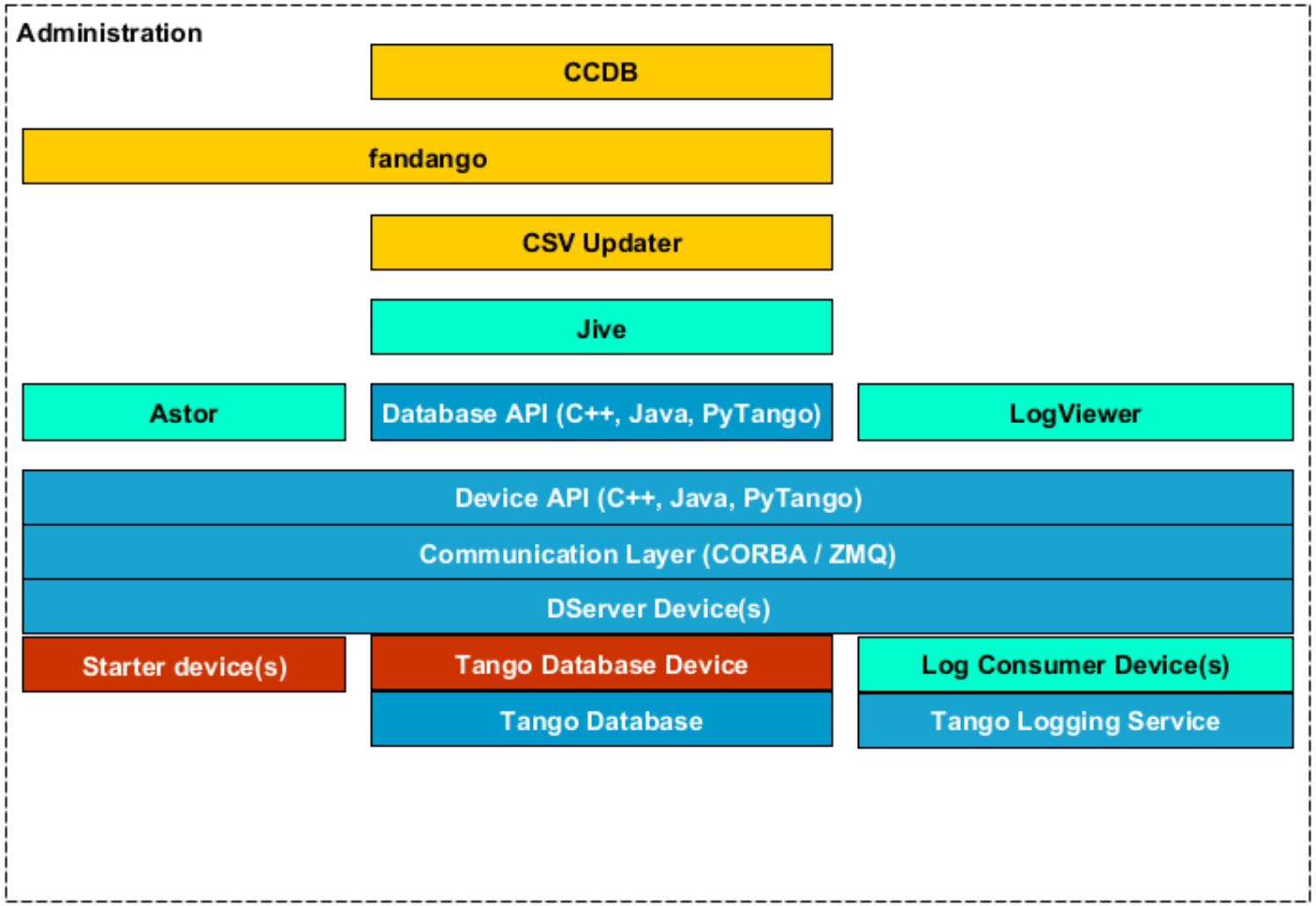


Tools concept #13

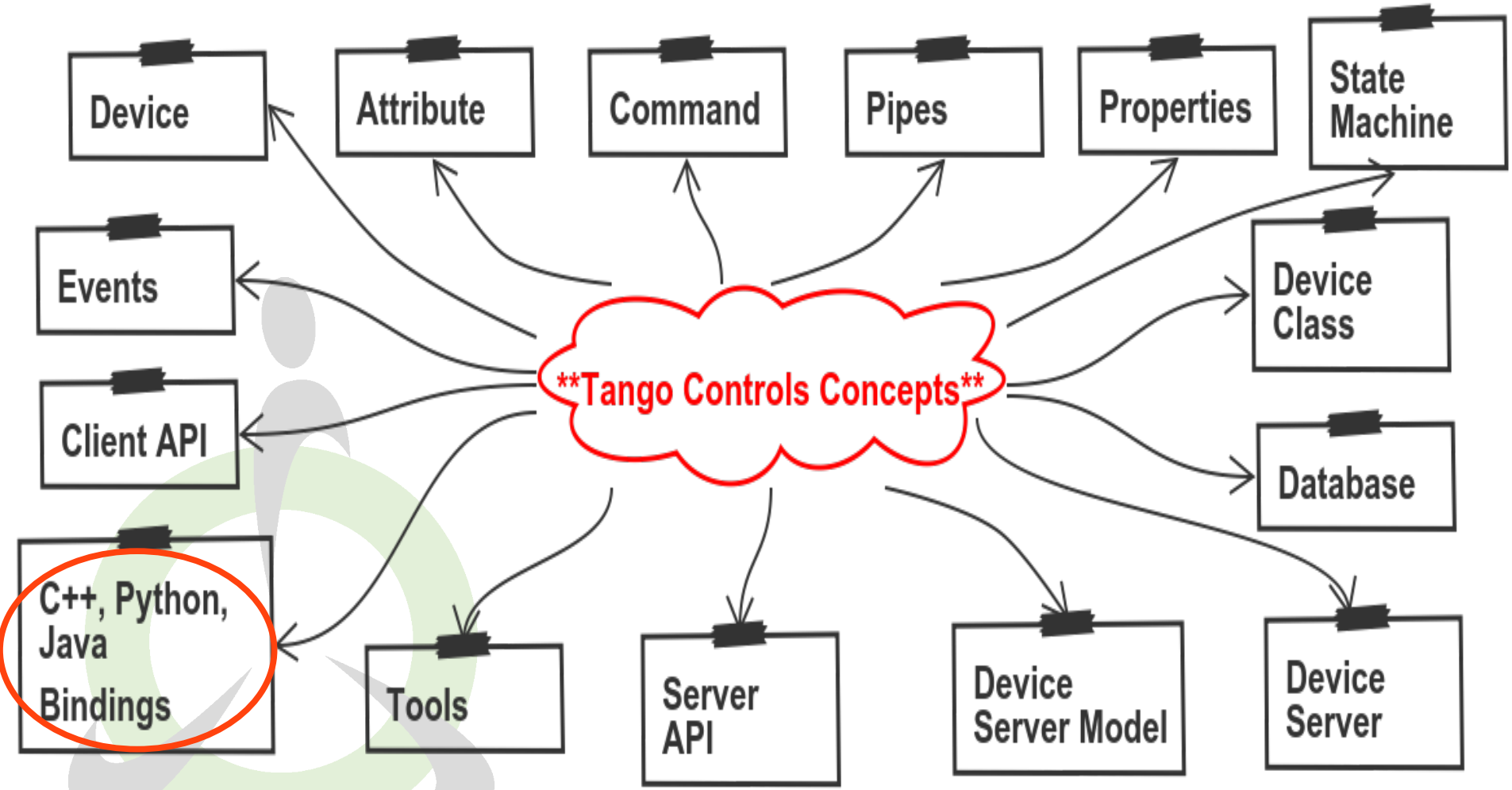
Many **tools** exist for Tango to configure + test Devices

- **Configuration tool**: Jive (alternatives are dsconfig, Waltz)
- **Start/stop control system**: Astor (alternatives system 5, Supervisord)
- **Test + monitor Device tool**: ATKPanel (alternatives are PyTango, Taurus, Waltz, WebJive, Jyse, …)
- **Archiver**: HDB++ (alternatives are HDB)
- **View logs**: Logviewer (alternatives Elasticsearch)
- **Alarm Handling**: Panic

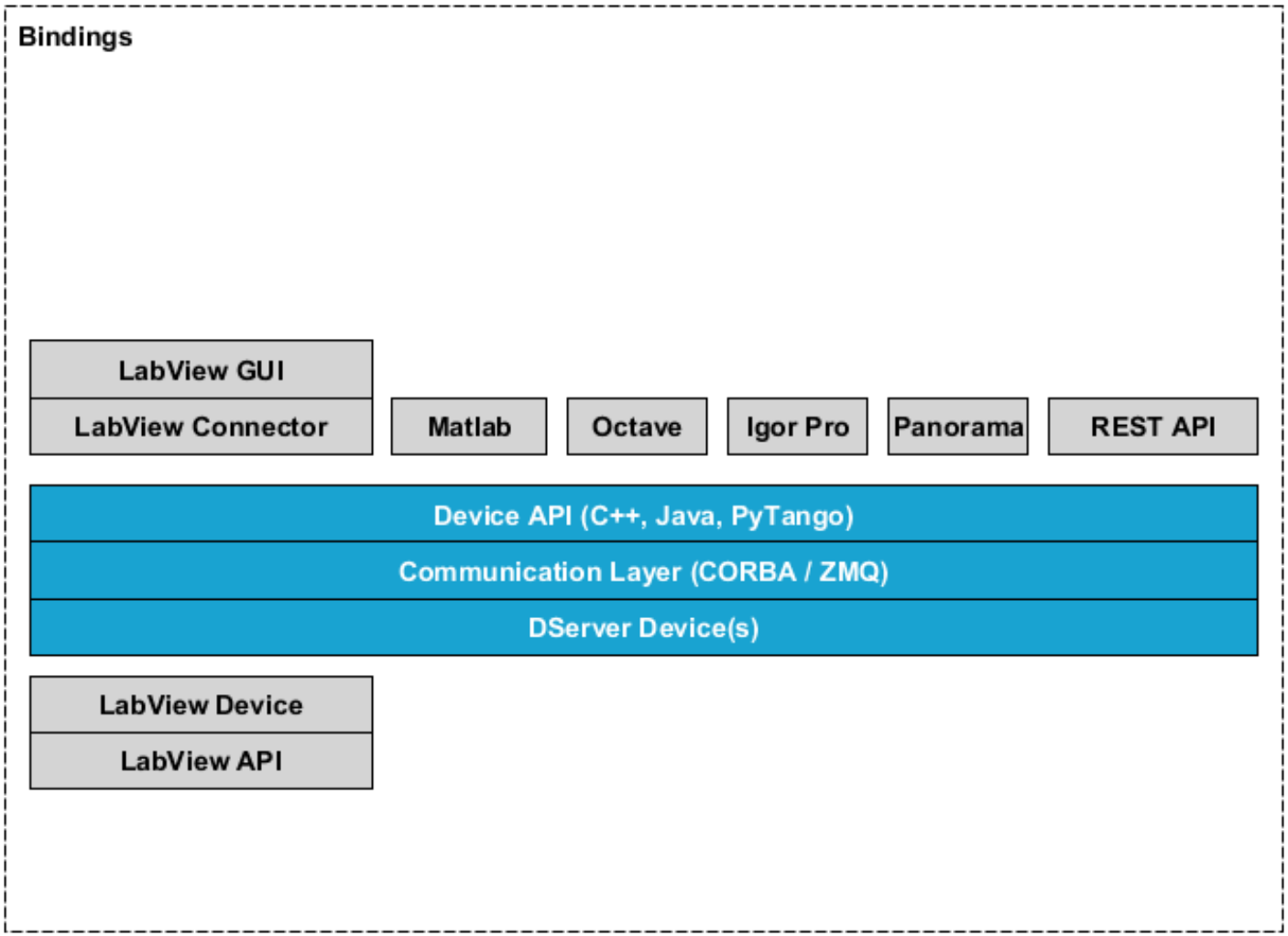
Tango **Tools** map



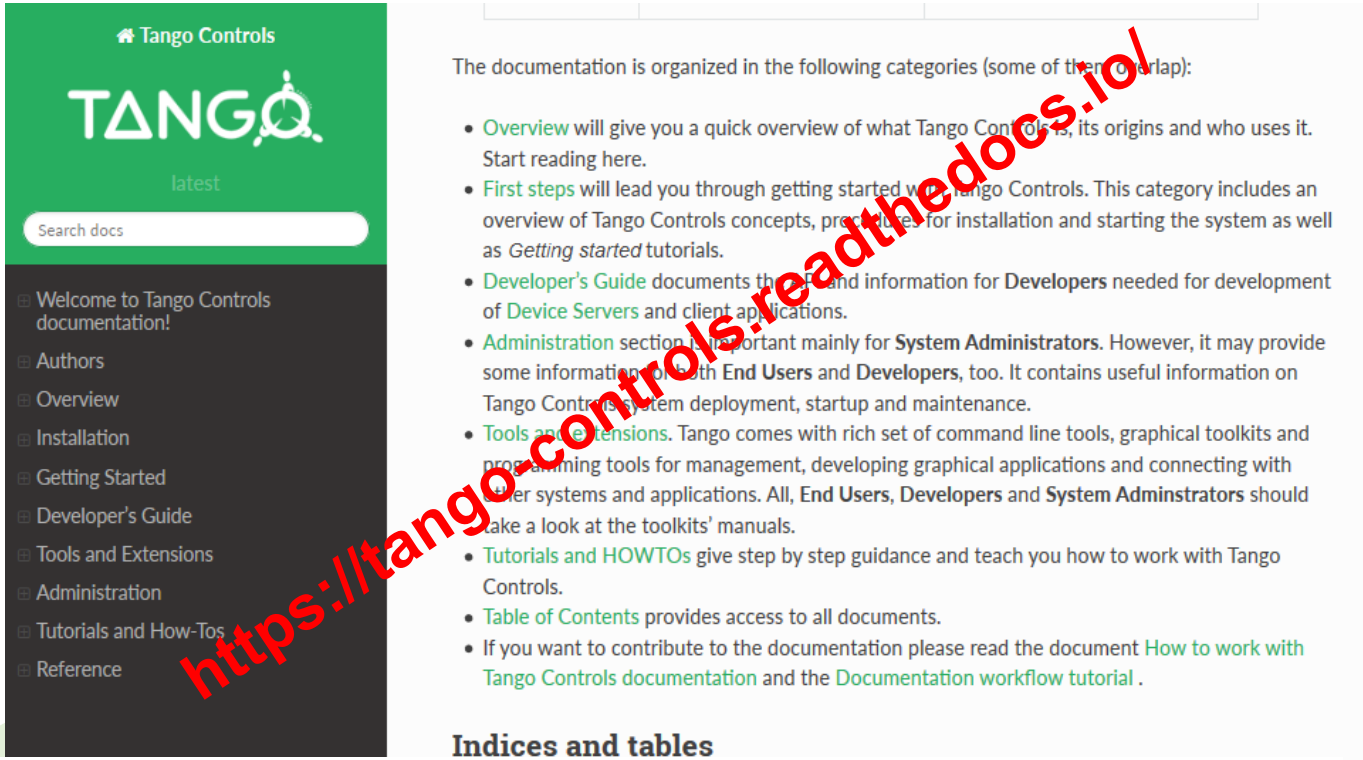
Bindings concept #14

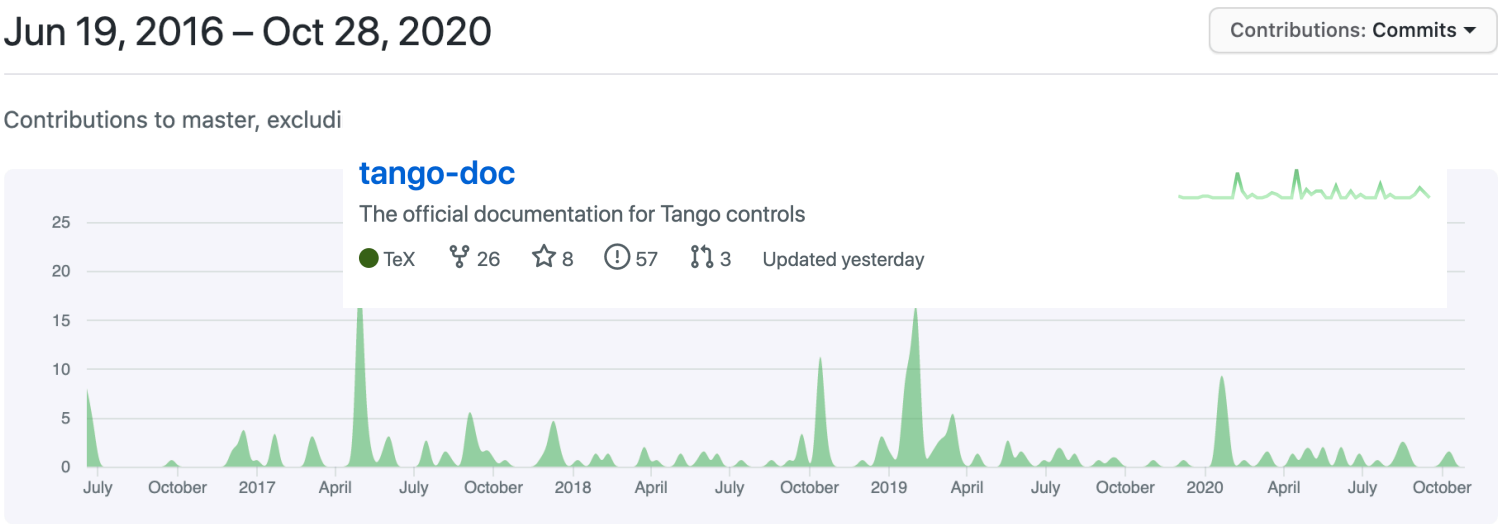


Tango **Bindings** map



Read the Documentation !





GitHub – Tango Controls

Tango Controls Core Projects

The official place for Tango Controls core projects

http://www.tango-controls.org info@tango-controls.org

Repositories 61 | Packages | People 38 | Teams 4 | Projects 1 | Settings

Pinned repositories

Customize pinned reposit

TangoTickets
The official place to create an issue/ticket related to Tango when it impacts several repositories or you don't know where to create your issue
★ 3

cppTango
TANGO kernel c++ implementation
C++ ★ 30 25

pytango
Python binding to Tango C++
Python ★ 36 37

rest-api
Tango REST API specification
★ 8 3

JTango
TANGO kernel Java implementation
Java ★ 7 9

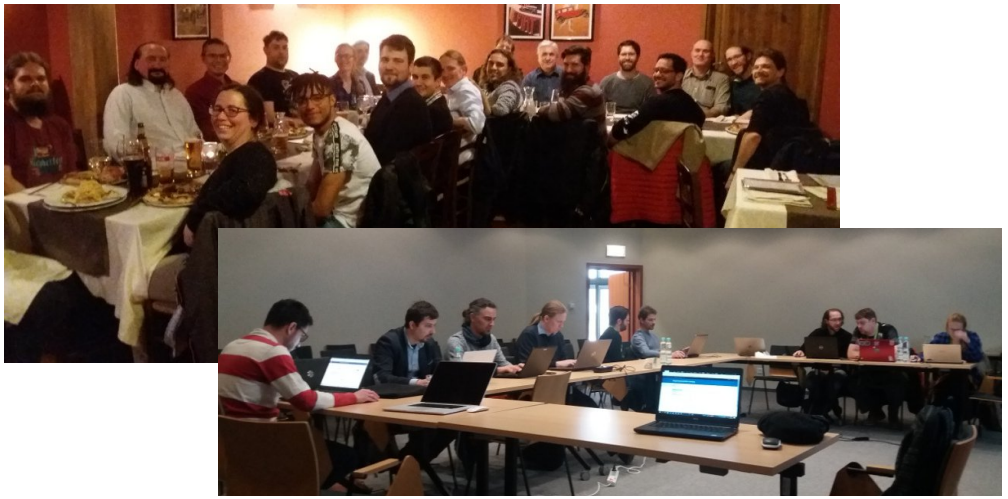
tango-doc
The official documentation for Tango controls
TeX ★ 5 13

TANGO

Tango Community

3 main events

Write the docs camp
next event ???



Kernel meeting
03 Nov 2020 -> ONLINE!

https://www.tango-controls.org/community/news/2020/10/21/1st-tango-kernel-webinar/

St. Petersburg, Russia

Autumn 2020 Status webinar
17 – 18 November 2020

https://www.tango-controls.org/community/events/autumn-2020-status-webinar/

34th Tango Collaboration meeting, 2020
Saint-Petersburg, Russia from 9 Jun 2020 to 11 Jun 2020

INAF @TANGO

Tango Collaboration is an international effort to **finance** and **sustain** the development of Tango in the **long-term**. The Tango Controls executive body is the **Steering Committee**. It makes **strategic decisions** about core developments in the Tango collaboration.

- **INAF** is part of the **TANGO Collaboration** as contributing member
- INAF main contribution is about **SKA**
- SKA **decided** to adopt TANGO as control framework for the whole telescope thanks to INAF

ALBA (core)
ELETTRA (core)
ESRF (core)
SOLEIL (core)
DESY (contributing)
INAF (contributing)
MAX-IV (contributing)
SKAO (contributing)
SKA-ZA (contributing)
SOLARIS (contributing)
ELI-Beamlines (Contributing)

https://www.tango-controls.org/about-us/#steering%20committee

TANGO Event organized by INAF

- 5-6 May 2015 - **Trieste Tango@SKA training course**
- 25-27 March 2015 – Trieste - **LMC Standardization Workshop**
- 06 Jun - 08 Jun 2017 – Firenze - **31st TANGO Collaboration Meeting**
- 2018 – **TANGO Training** @Torino
- 2018 – **TANGO Training** @Cagliari

Main people involved

Name	Affiliation	Skills
Matteo Canzari	OAAb	INAF member representatives, framework, tango related web technlogies
Matteo Di Carlo	OAAb	SKA TANGO CoP chair, framework, container, CI/CD
2 open positions NEW	**OAAb**	Will be involved in SKA and LOFAR2
Elisabetta Giani	INAF Arcetri	framework, tango c++
Carlo Baffa	INAF Arcetri	framework, tango c++
Marina Vela Nunez	OATs	framework
Simone Riggi	OACt	framework
Cristina Knapic	OATs	framework
Riccardo Smareglia	OATs	framework
Elisa Londero	OATs	framework
Andrea Bignamini	OATs	framework
Robert Butora	OATs	framework
David Tosolini	OATs	framework

Industrial Partner	Description
ELETTRA	MoU with Elettra Synchrotron

INAF Projects interested

International collaborations

Project	Description
SKA	Developing the SKA control software
LOFAR2	Monitor and Control Subsystem for LOFAR 2 Station

Internal projects

Project	Description
SRT	Web user interface (proof-of-concept), ACS-TANGO bridge
NADIR	Data archiving and distribution
PRISMA	Monitoring and control full-sky camera

Join the community

THANKS!

Contact me:
matteo.canzari@inaf.it

TANGO controls