



从C++到Python

规划团队的知识结构，提高整体做事能力

同济大学电子与信息工程学院实验中心
裴喜龙
2021年4月



C++构件模型

C++ Component Model, COMO

1. COMO历经近20年的开发，与JVM、CLR并列的技术：<https://gitee.com/tjopenlab>
2. 按规范写C++程序，然后无缝在python中使用；
3. C++的程序执行效率与GDB级的程序调试，可以对接JTAG硬件调试；
4. C++与硬件的直接打交道的能力。
5. Apache授权协议的开源项目，用于学习与工程项目，无法律风险。

<https://gitee.com/tjopenlab/como>

https://gitee.com/tjopenlab/como_pybind

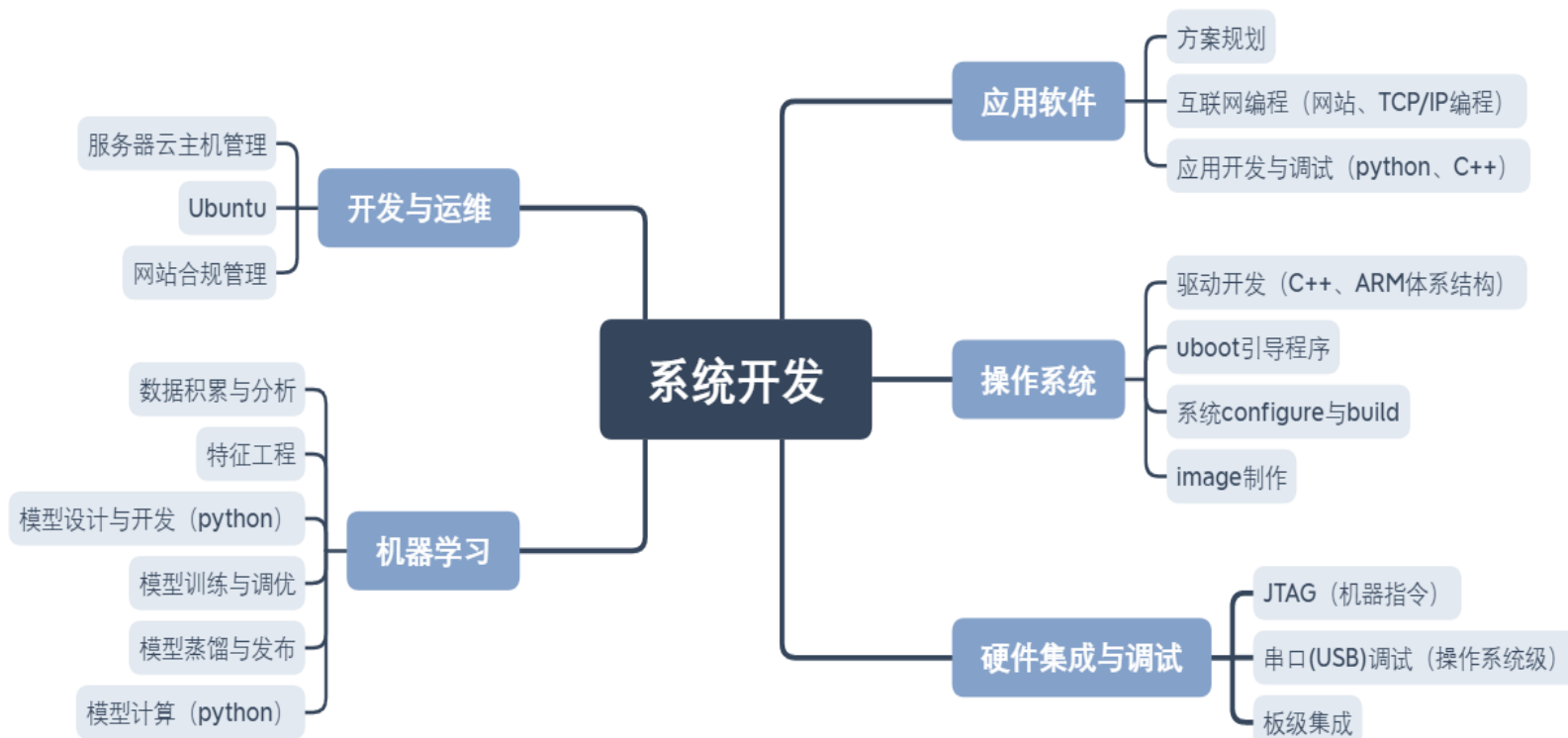


系统开发团队知识结构

- 自动控制专业、微电子专业等靠近硬件的专业。
- 计算机科学与技术、软件工程等靠近计算机应用的专业。

C++侧：计算机硬件、系统级开发

python侧：计算机软件、互联网





C++侧

```
test > runtime > reflection > component > MethodTest.cdl
17 namespace como {
18 namespace test {
19 namespace reflection {
20
21 [
22     uuid(6611bd57-14c2-4b5f-b971-9d47b122fc21),
23     version(0.1.0)
24 ]
25 interface IMethodTest
26 {
27     TestMethod1(
28         [in] Integer arg,
29         [out] Integer& result);
30
31     TestMethod2(
32         [in] Float arg,
33         [out] Float& result);
34
35     TestMethod3(
36         [in] Integer arg1,
37         [in] Long arg2,
38         [in] Boolean arg3,
39         [in] Char arg4,
40         [in] Short arg5,
41         [in] Integer arg6);
42 }
```

```
EXPLORER
...
CMethodTester.cpp X
test > runtime > reflection > component > CMethodTester.cpp
17 #include "CMethodTester.h"
18
19 #include <cstdio>
20
21 namespace como {
22 namespace test {
23 namespace reflection {
24
25 COMO_INTERFACE_IMPL_1(CMethodTester, Object, IMethodTest);
26 COMO_OBJECT_IMPL(CMethodTester);
27
28 CMethodTester::CMethodTester()
29 {
30     printf("==== call CMethodTester() ==== \n");
31 }
32
33 CMethodTester::~CMethodTester()
34 {
35     printf("==== call ~CMethodTester() ==== \n");
36 }
37
38 ECode CMethodTester::TestMethod1(
39     /* [in] */ Integer arg,
40     /* [out] */ Integer& result)
```

<https://gitee.com/tjopenlab/como/blob/master/test/runtime/reflection/component/MethodTest.cdl>



Python侧

```
import como_pybind
import os, sys
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), "../")))

mc = como_pybind.como('ReflectionTestUnit.so')
name = mc.getName();
print ('name is: ' + name)
print ('constants: ', mc.getAllConstants());

cmt = como_pybind.CMethodTester()
print ('load COMO class CMethodTester')

print (cmt.TestMethod1(12392021))
print (cmt.getAllConstants())
```

https://gitee.com/tjopenlab/como_pybind/blob/master/tests/como_pybind_basictest.py



Tensorflow中的算子

1. op和kernel是TF框架中最重要的两个概念，如果一定要做一个类比的话，可以认为op相当于函数声明，kernel相当于函数实现。
2. 为什么要把操作和它的实现分离呢？是为了实现TF代码的可移植性。

<http://vernlium.github.io/2019/07/01/Tensorflow%E6%BA%90%E7%A0%81%E8%A7%A3%E6%9E%90%E2%80%94%E2%80%94%E7%AE%97%E5%AD%A0%E6%B3%A8%E5%86%8C/>



华为昇腾AI处理器支持的算子

(你想开发昇腾AI处理器的算子吗? 没啥文档)

https://support.huaweicloud.com/opl-atlas200dkappc32/atlasos_09_0001.html



利用COMO封装机器学习中的算子

1. 己所不欲，勿施于人。
2. 这么难啃的骨头咱们还是自己啃吧，毕竟让人家把我们的硬件用起来是正道。



未完，
待续