Codeit控制系统

Codeit 简介

Control

机器人学

- 完整的控制框架，控制被控对象的动力学特性，包含控制指令、数据通信、运动学模型、运动规划、运动控制算法。

Develop

C++及软件设计

- 完全基于C++17特性开发；跨系统运行（可在windows下调试开发）；模块化开发。

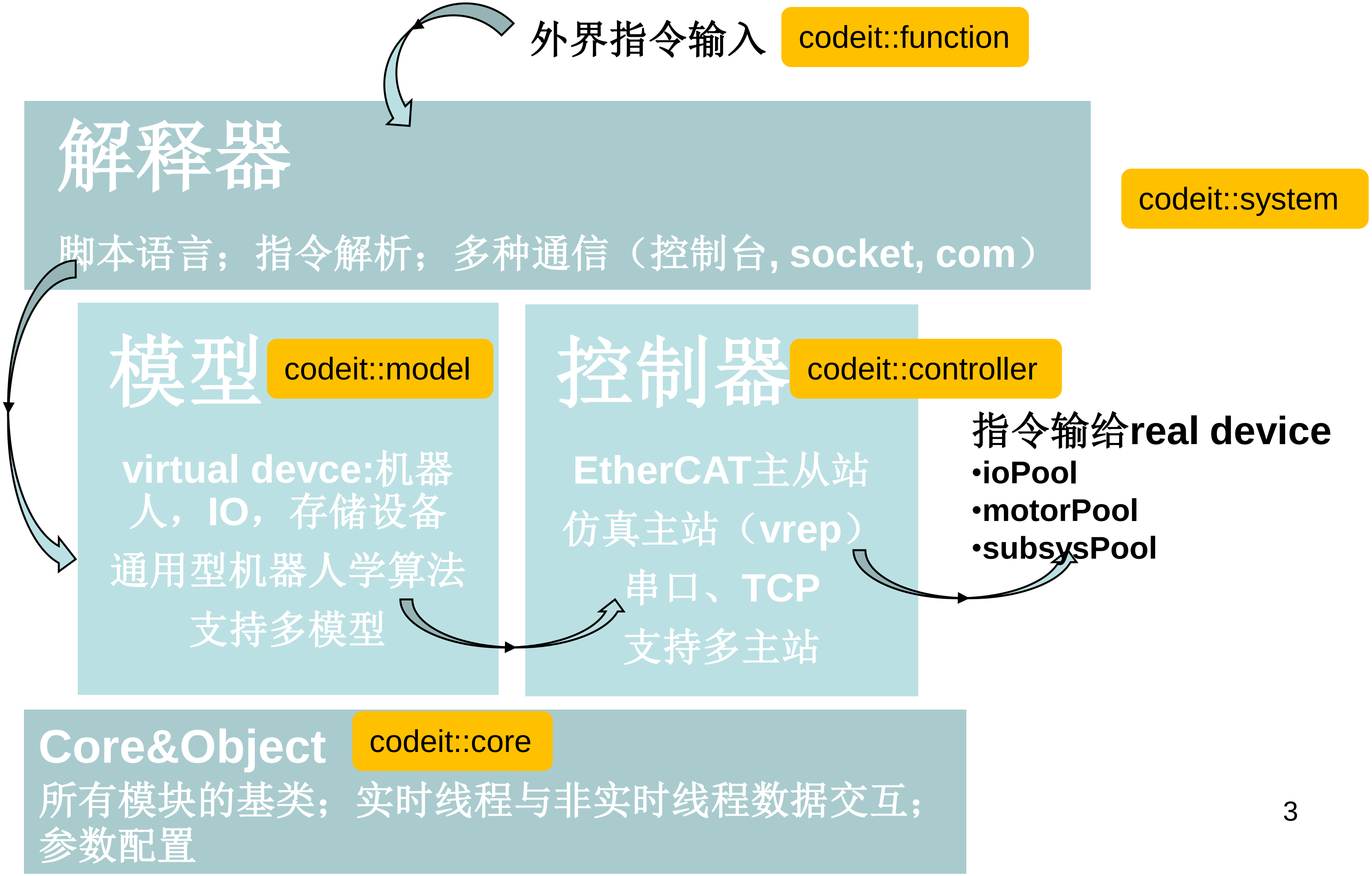
- 控制系统开发平台，功能易于添加、拓展与维护；敏捷协同开发。

It

控制系统组成

- 被控对象的通用性，小到led灯控制，大到整个工厂的控制，可在相同软件框架下实现。

Codeit——架构



外界指令输入 codeit::function

解释器

脚本语言；指令解析；多种通信（控制台, socket, com）

codeit::system

模型 codeit::model

virtual devce:机器人，IO，存储设备

通用型机器人学算法

支持多模型

控制器 codeit::controller

EtherCAT主从站

仿真主站（vrep）

串口、TCP

支持多主站

指令输给real device

•ioPool

•motorPool

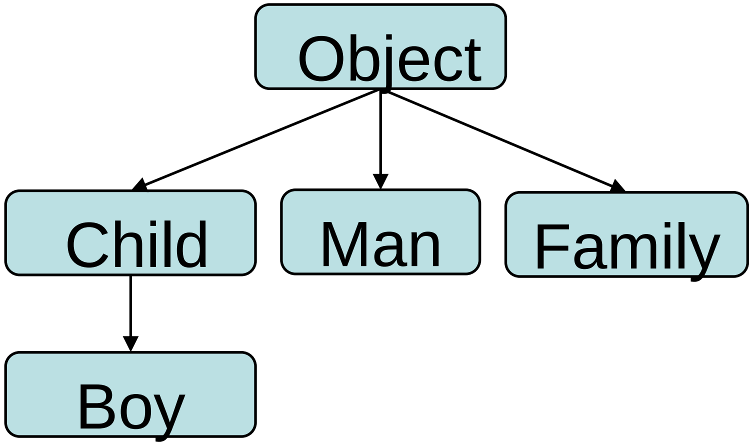
•subsysPool

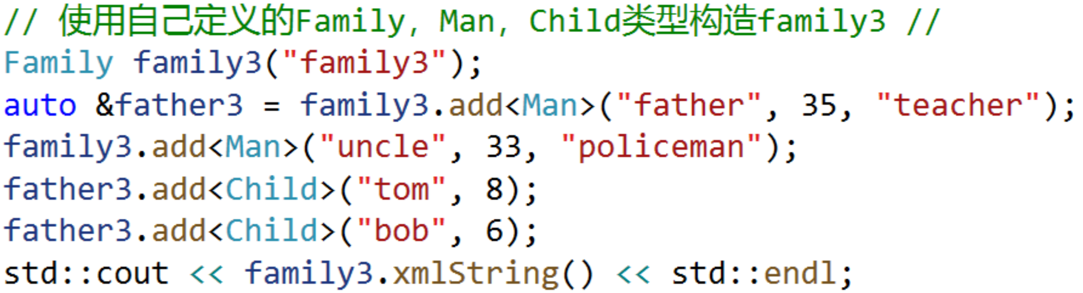
Core&Object codeit::core

所有模块的基类；实时线程与非实时线程数据交互；参数配置

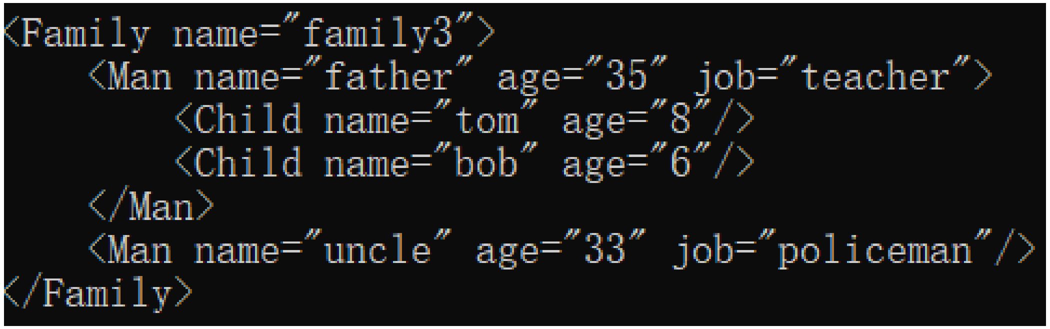
codeit::core——Object

- 面向对象编程：

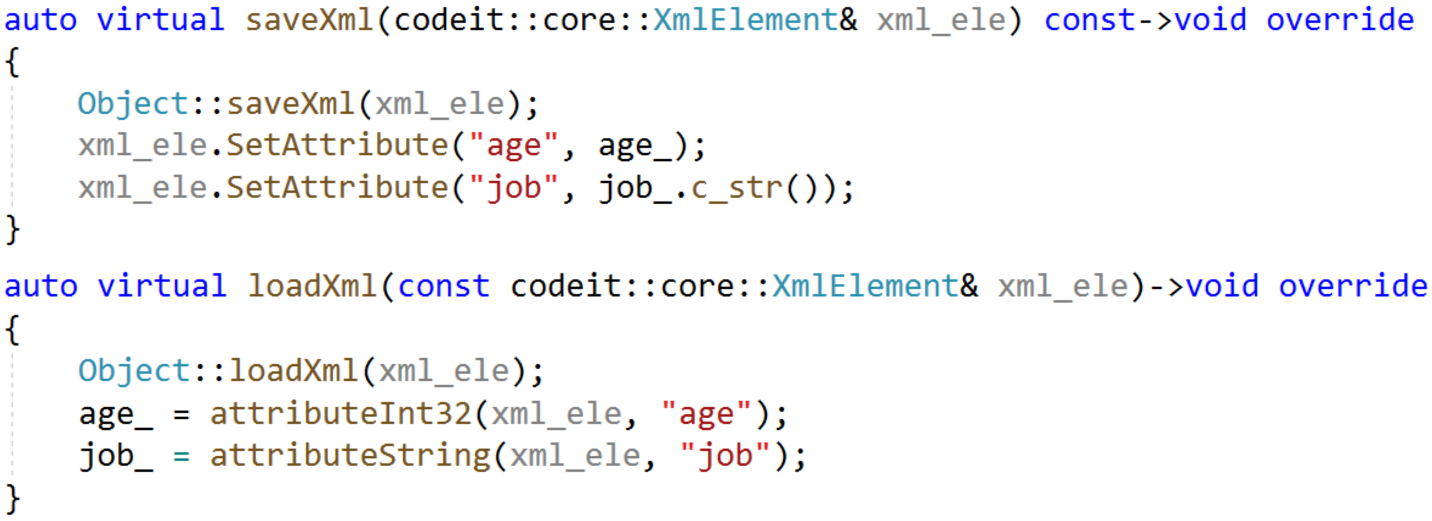




```cpp
// 使用自己定义的Family, Man, Child类型构造family3 //
Family family3("family3");
auto &father3 = family3.add<Man>("father", 35, "teacher");
family3.add<Man>("uncle", 33, "policeman");
father3.add<Child>("tom", 8);
father3.add<Child>("bob", 6);
std::cout << family3.xmlString() << std::endl;
```

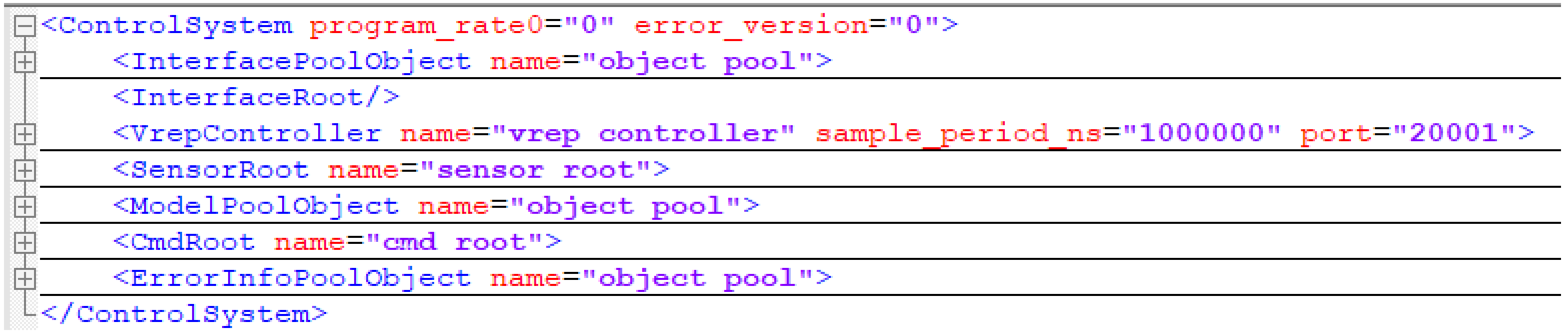


```
<Family name="family3">
    <Man name="father" age="35" job="teacher">
        <Child name="tom" age="8"/>
        <Child name="bob" age="6"/>
    </Man>
    <Man name="uncle" age="33" job="policeman"/>
</Family>
```

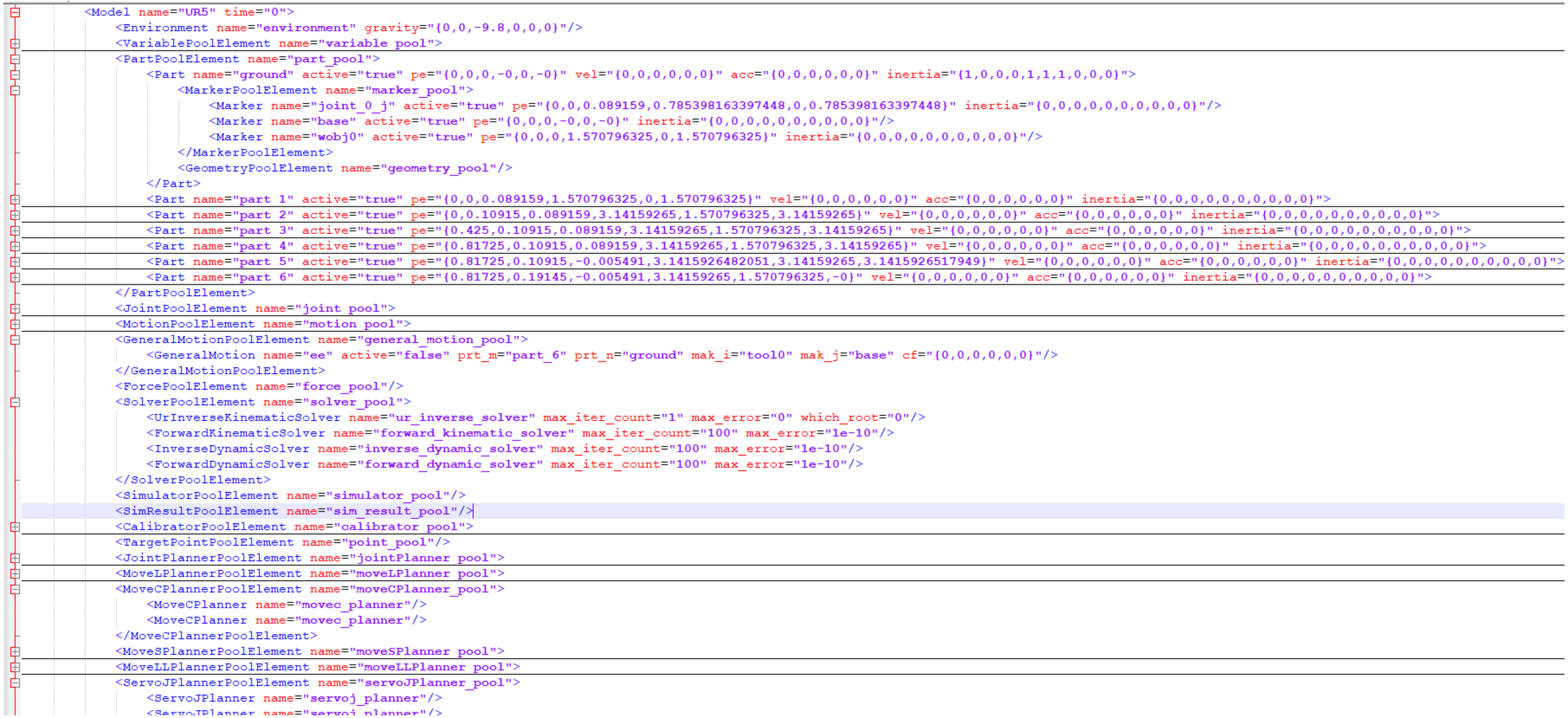


```cpp
auto virtual saveXml(codeit::core::XmlElement& xml_ele) const->void override
{
    Object::saveXml(xml_ele);
    xml_ele.SetAttribute("age", age_);
    xml_ele.SetAttribute("job", job_.c_str());
}
auto virtual loadXml(const codeit::core::XmlElement& xml_ele)->void override
{
    Object::loadXml(xml_ele);
    age_ = attributeInt32(xml_ele, "age");
    job_ = attributeString(xml_ele, "job");
}
```

codeit::core——参数配置



```
<ControlSystem program_rate0="0" error_version="0">
    <InterfacePoolObject name="object pool">
    <InterfaceRoot/>
    <VrepController name="vrep controller" sample_period_ns="1000000" port="20001">
    <SensorRoot name="sensor root">
    <ModelPoolObject name="object pool">
    <CmdRoot name="cmd root">
    <ErrorInfoPoolObject name="object pool">
</ControlSystem>
```



```
<Model name="UR5" time="0">
    <Environment name="environment" gravity="{0,0,-9.8,0,0,0}"/>
    <VariablePoolElement name="variable pool">
    <PartPoolElement name="part_pool">
        <Part name="ground" active="true" pe="{0,0,0,-0,0,-0}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{1,0,0,0,1,1,1,0,0,0}">
            <MarkerPoolElement name="marker_pool">
                <Marker name="joint_0_j" active="true" pe="{0,0,0.089159,0.785398163397448,0,0.785398163397448}" inertia="{0,0,0,0,0,0,0,0,0,0}"/>
                <Marker name="base" active="true" pe="{0,0,0,-0,0,-0}" inertia="{0,0,0,0,0,0,0,0,0,0}"/>
                <Marker name="wobj0" active="true" pe="{0,0,0,1.570796325,0,1.570796325}" inertia="{0,0,0,0,0,0,0,0,0,0}"/>
            </MarkerPoolElement>
            <GeometryPoolElement name="geometry_pool"/>
        </Part>
        <Part name="part 1" active="true" pe="{0,0,0.089159,1.570796325,0,1.570796325}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{0,0,0,0,0,0,0,0,0,0}">
        <Part name="part 2" active="true" pe="{0,0.10915,0.089159,3.14159265,1.570796325,3.14159265}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{0,0,0,0,0,0,0,0,0,0}">
        <Part name="part 3" active="true" pe="{0.425,0.10915,0.089159,3.14159265,1.570796325,3.14159265}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{0,0,0,0,0,0,0,0,0,0}">
        <Part name="part 4" active="true" pe="{0.81725,0.10915,0.089159,3.14159265,1.570796325,3.14159265}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{0,0,0,0,0,0,0,0,0,0}">
        <Part name="part 5" active="true" pe="{0.81725,0.10915,-0.005491,3.1415926482051,3.14159265,3.1415926517949}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{0,0,0,0,0,0,0,0,0,0}">
        <Part name="part 6" active="true" pe="{0.81725,0.19145,-0.005491,3.14159265,1.570796325,-0}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{0,0,0,0,0,0,0,0,0,0}">
    </PartPoolElement>
    <JointPoolElement name="joint pool">
    <MotionPoolElement name="motion pool">
    <GeneralMotionPoolElement name="general_motion_pool">
        <GeneralMotion name="ee" active="false" prt_m="part_6" prt_n="ground" mak_i="tool0" mak_j="base" cf="{0,0,0,0,0,0}"/>
    </GeneralMotionPoolElement>
    <ForcePoolElement name="force_pool"/>
    <SolverPoolElement name="solver_pool">
        <UrInverseKinematicSolver name="ur_inverse_solver" max_iter_count="1" max_error="0" which_root="0"/>
        <ForwardKinematicSolver name="forward_kinematic_solver" max_iter_count="100" max_error="1e-10"/>
        <InverseDynamicSolver name="inverse_dynamic_solver" max_iter_count="100" max_error="1e-10"/>
        <ForwardDynamicSolver name="forward_dynamic_solver" max_iter_count="100" max_error="1e-10"/>
    </SolverPoolElement>
    <SimulatorPoolElement name="simulator_pool"/>
    <SimResultPoolElement name="sim_result_pool"/>
    <CalibratorPoolElement name="calibrator pool">
    <TargetPointPoolElement name="point_pool"/>
    <JointPlannerPoolElement name="jointPlanner pool">
    <MoveLPlannerPoolElement name="moveLPlanner pool">
    <MoveCPlannerPoolElement name="moveCPlanner_pool">
        <MoveCPlanner name="movec_planner"/>
        <MoveCPlanner name="movec_planner"/>
    </MoveCPlannerPoolElement>
    <MoveSPlannerPoolElement name="moveSPlanner pool">
    <MoveLLPlannerPoolElement name="moveLLPlanner pool">
    <ServoJPlannerPoolElement name="servoJPlanner_pool">
        <ServoJPlanner name="servoj_planner"/>
        <ServoJPlanner name="servoj_planner"/>
```

codeit::controller

- 单实时主站：EtherCAT, CAN, Vrep(仿真)
- 多非实时主站：Com, Socket, ModBus…

控制器

EtherCAT主从站

仿真主站（vrep）

串口、TCP

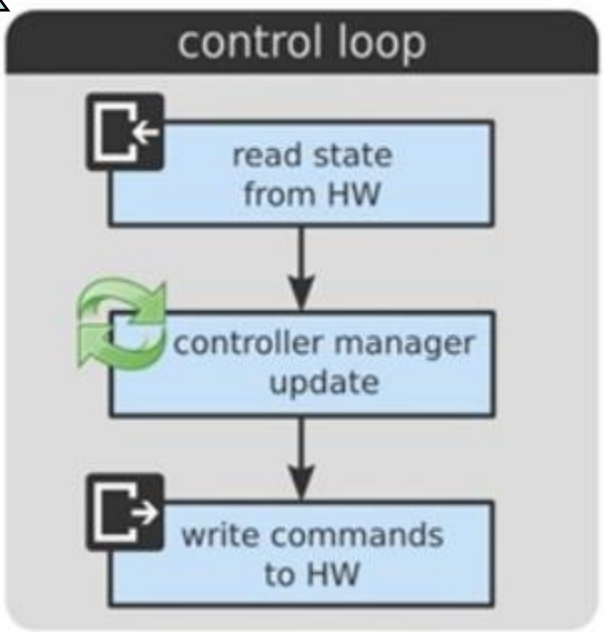
支持多主站

指令输给real device

•ioPool

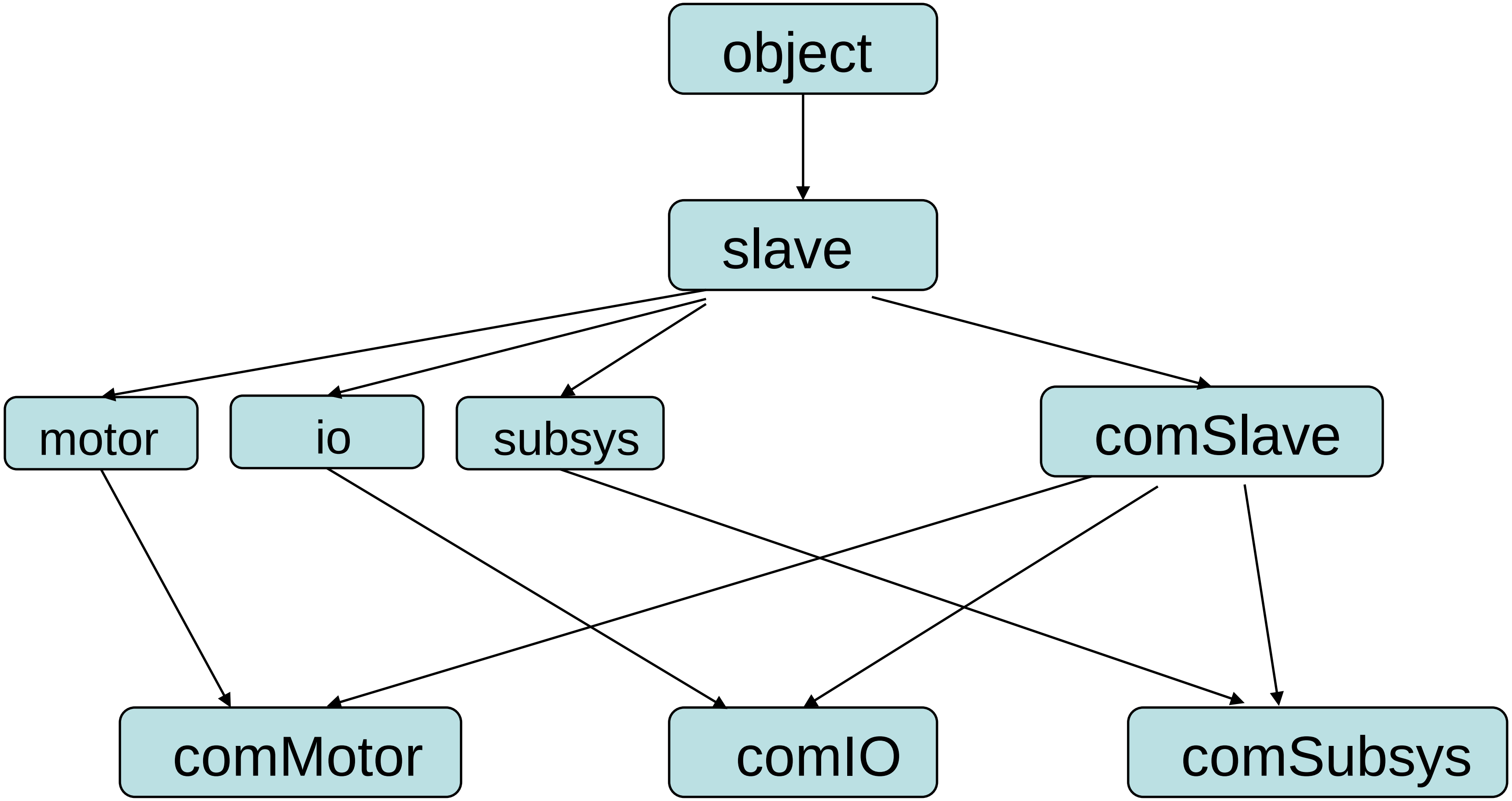
•motorPool

•subsysPool



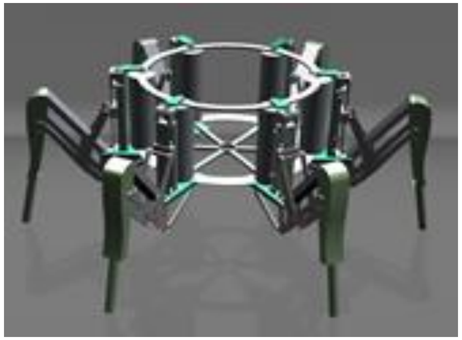
codeit::controller

- com下slave示例：



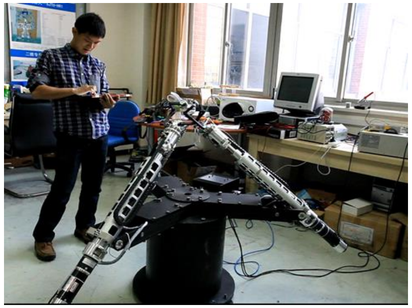
codeit::model——通用型运动学

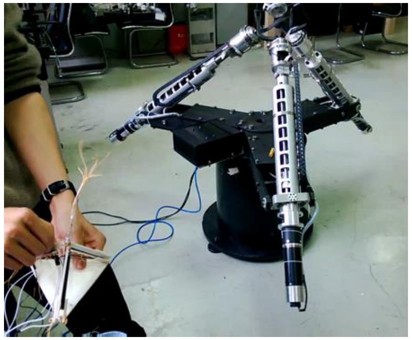
足式机器人



医疗机器人

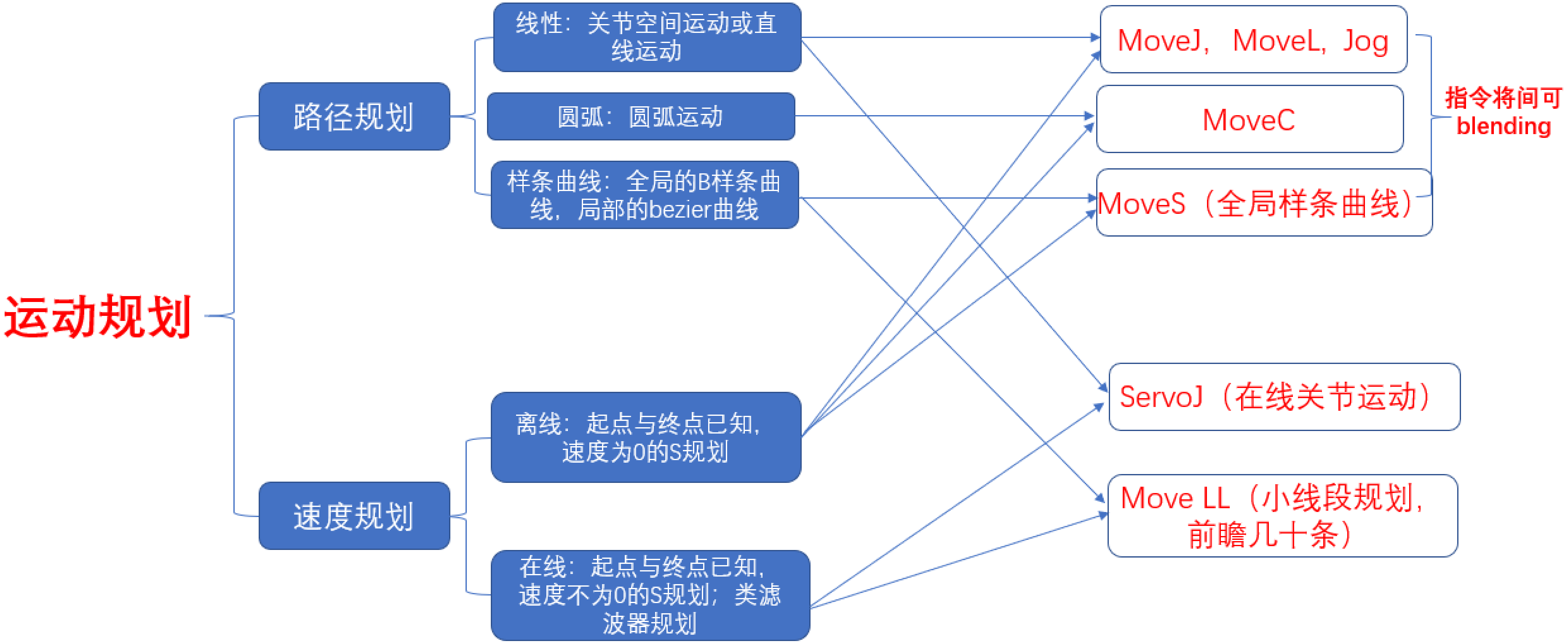
遥操作机器人





工业机器人

codeit::model——通用型运动规划



运动规划

路径规划

线性：关节空间运动或直线运动

圆弧：圆弧运动

样条曲线：全局的B样条曲线，局部的bezier曲线

速度规划

离线：起点与终点已知，速度为0的S规划

在线：起点与终点已知，速度不为0的S规划；类滤波器规划

MoveJ，MoveL，Jog

MoveC

MoveS（全局样条曲线）

指令将间可blending

ServoJ（在线关节运动）

Move LL（小线段规划，前瞻几十条）

codeit::model——通用型动力学

- 基于电流环的动力学

任意串并联机器人动力学模型

- 复杂动力学和摩擦力模型可在伺服周期（**1ms**）计算完毕

动力学模型参数辨识

- 全臂动力学参数辨识
- 负载惯性参数变数

电流环力控制

- 基于电流环的拖动示教；摩擦力精细化处理
- 阻抗算法调节每个方向的刚度与柔性，适应各种工件

codeit::model——通用型动力学

- 基于电流环的动力学

codeit::model——通用型动力学

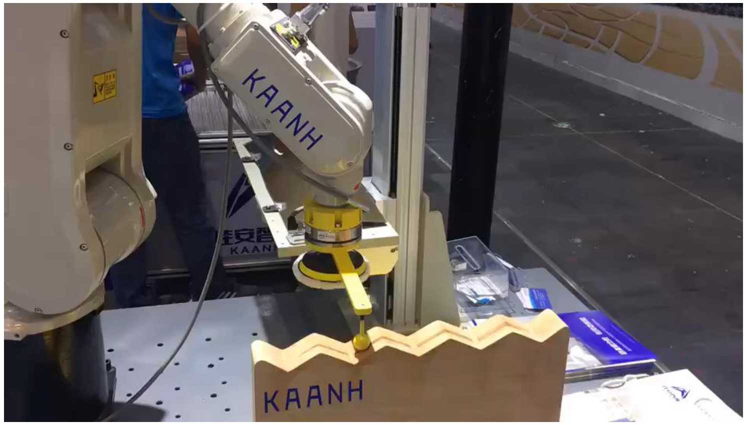
- 基于六轴力传感器的动力学：建立留六轴力传感器与关节状态间的动力学方程。

电流环与力矩传感器混合拖动：末端轻盈拖动，全臂范围辅助拖动，安全过奇异点，负载重力自补偿，启动停止灵敏，轨迹全真复现。

codeit::model——外部传感器引导规划

- 力传感器引导运动规划：

将运动规划与力控技术融合



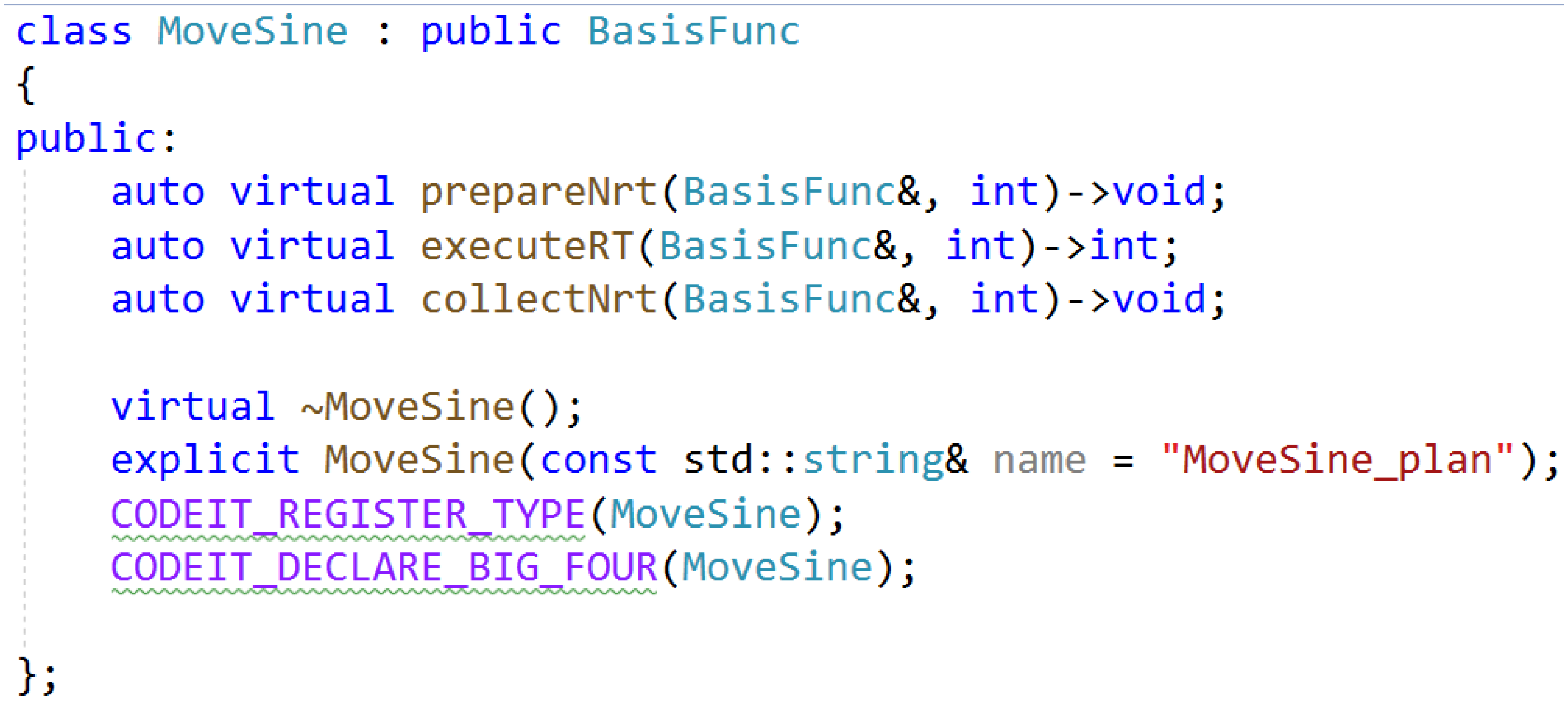
锯齿面贴合：只给出起点与终点平面位置，高度方向自适应

复杂曲面法向恒压力贴合：只给出曲面起点与终点平面位置，曲面法向自适应寻找，并控制法向压力

复杂轮廓贴合：只给出轮廓的数个边角点，完成对整个轮廓的恒力贴合

codeit::function——指令模板

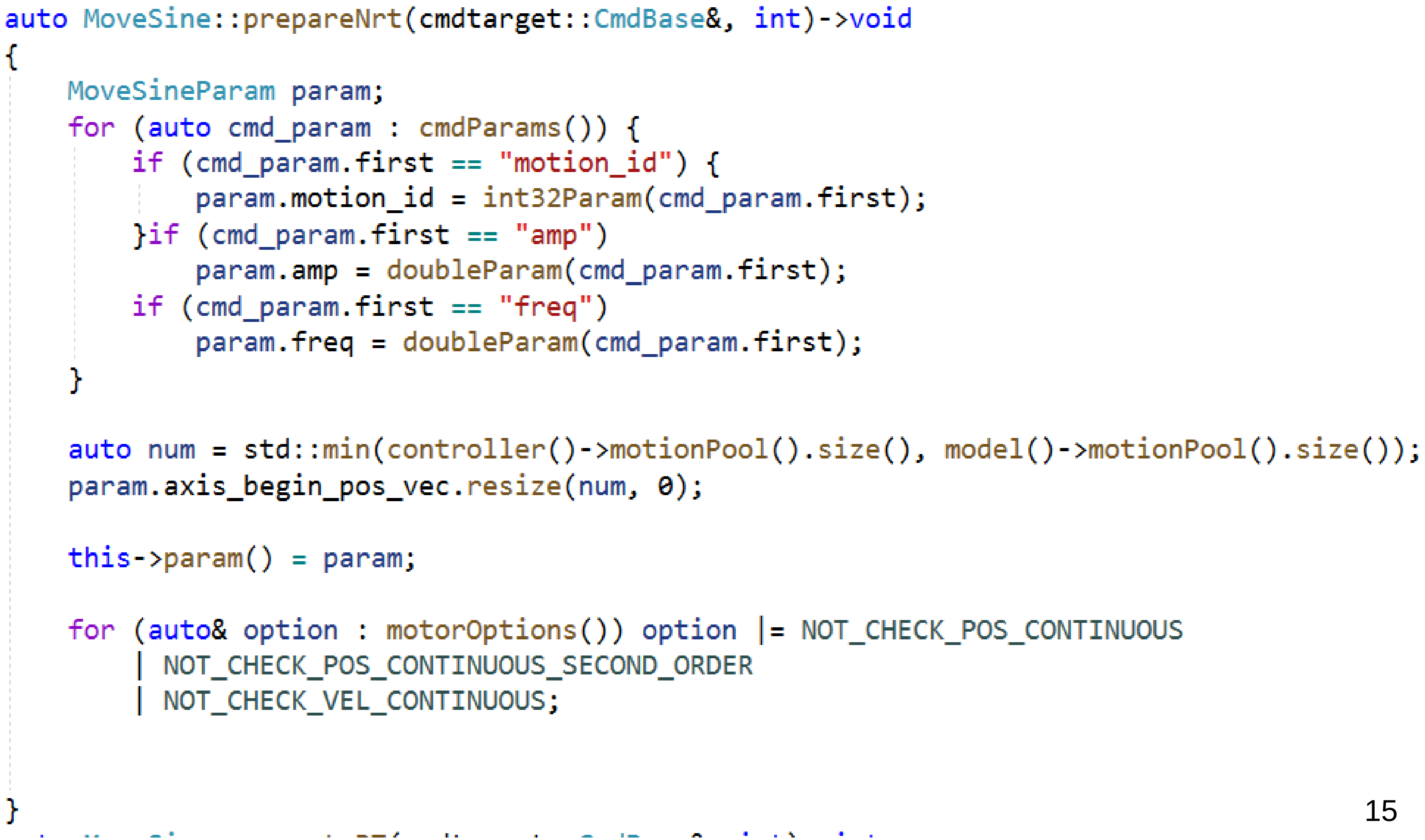
- xfunc.hpp



```cpp
class MoveSine : public BasisFunc
{
public:
    auto virtual prepareNrt(BasisFunc&, int)->void;
    auto virtual executeRT(BasisFunc&, int)->int;
    auto virtual collectNrt(BasisFunc&, int)->void;

    virtual ~MoveSine();
    explicit MoveSine(const std::string& name = "MoveSine_plan");
    CODEIT_REGISTER_TYPE(MoveSine);
    CODEIT_DECLARE_BIG_FOUR(MoveSine);

};
```

codeit::function——指令模板

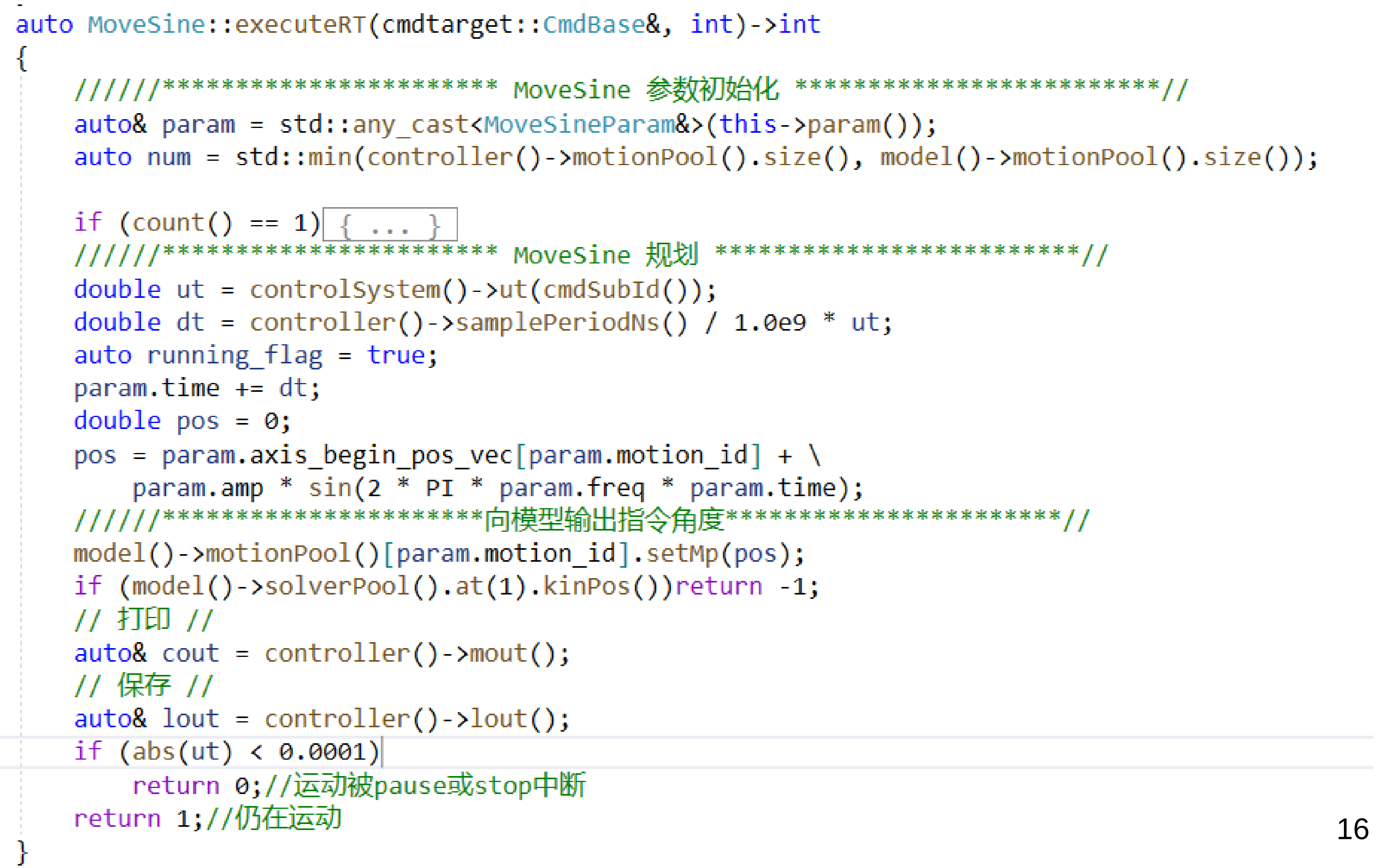
- xfunc.cpp



```
auto MoveSine::prepareNrt(cmdtarget::CmdBase&, int)->void
{
    MoveSineParam param;
    for (auto cmd_param : cmdParams()) {
        if (cmd_param.first == "motion_id") {
            param.motion_id = int32Param(cmd_param.first);
        }if (cmd_param.first == "amp")
            param.amp = doubleParam(cmd_param.first);
        if (cmd_param.first == "freq")
            param.freq = doubleParam(cmd_param.first);
    }

    auto num = std::min(controller()->motionPool().size(), model()->motionPool().size());
    param.axis_begin_pos_vec.resize(num, 0);

    this->param() = param;

    for (auto& option : motorOptions()) option |= NOT_CHECK_POS_CONTINUOUS
        | NOT_CHECK_POS_CONTINUOUS_SECOND_ORDER
        | NOT_CHECK_VEL_CONTINUOUS;


}
```

codeit::function——指令模板

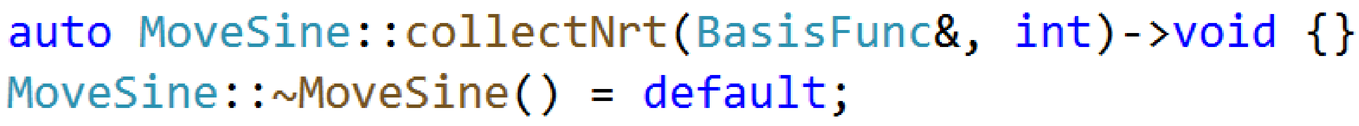
- xfunc.cpp



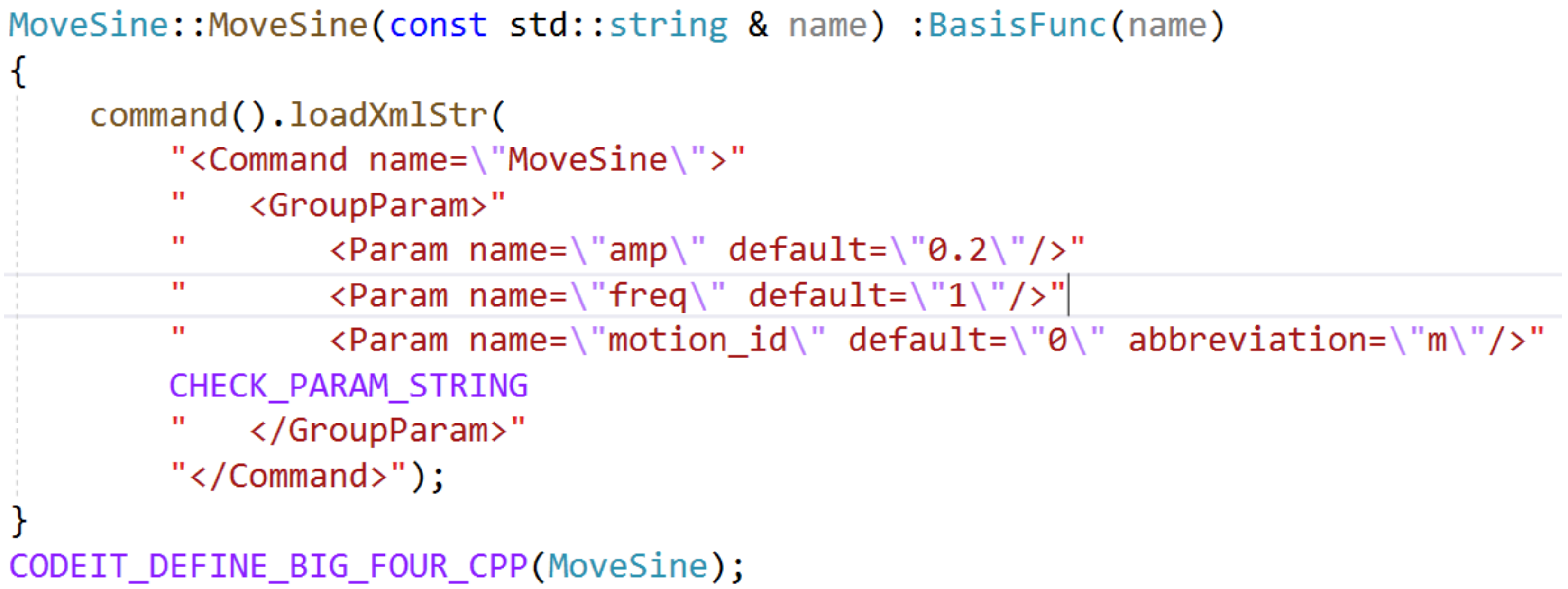
```
auto MoveSine::executeRT(cmdtarget::CmdBase&, int)->int
{
    //////************************ MoveSine 参数初始化 **************************//
    auto& param = std::any_cast<MoveSineParam&>(this->param());
    auto num = std::min(controller()->motionPool().size(), model()->motionPool().size());

    if (count() == 1) { ... }
    //////************************ MoveSine 规划 **************************//
    double ut = controlSystem()->ut(cmdSubId());
    double dt = controller()->samplePeriodNs() / 1.0e9 * ut;
    auto running_flag = true;
    param.time += dt;
    double pos = 0;
    pos = param.axis_begin_pos_vec[param.motion_id] + \
        param.amp * sin(2 * PI * param.freq * param.time);
    //////**********************向模型输出指令角度***********************//
    model()->motionPool()[param.motion_id].setMp(pos);
    if (model()->solverPool().at(1).kinPos())return -1;
    // 打印 //
    auto& cout = controller()->mout();
    // 保存 //
    auto& lout = controller()->lout();
    if (abs(ut) < 0.0001)
        return 0;//运动被pause或stop中断
    return 1;//仍在运动
}
```

codeit::function——指令模板

- xfunc.cpp



```
auto MoveSine::collectNrt(BasisFunc&, int)->void {}
MoveSine::~MoveSine() = default;
```



```
MoveSine::MoveSine(const std::string & name) :BasisFunc(name)
{
    command().loadXmlStr(
        "<Command name=\"MoveSine\">"
        "    <GroupParam>"
        "        <Param name=\"amp\" default=\"0.2\"/>"
        "        <Param name=\"freq\" default=\"1\"/>"
        "        <Param name=\"motion_id\" default=\"0\" abbreviation=\"m\"/>"
        CHECK_PARAM_STRING
        "    </GroupParam>"
        "</Command>");
}
CODEIT_DEFINE_BIG_FOUR_CPP(MoveSine);
```

codeit::function——异常处理

- 异常等级

FATAL:致命性错误；系统需重启

WARNING:警告；调整指令即可

ERROR:一般性错误；通过Clear清楚错误

- prepareNrt()

```
errorinfoPool->add<codeit::system::ErrorInfo>
("an exception", -10, "WARNNING", "一个异常出现", "an exception exists");
THROW_FILE_LINE("an exception");
```

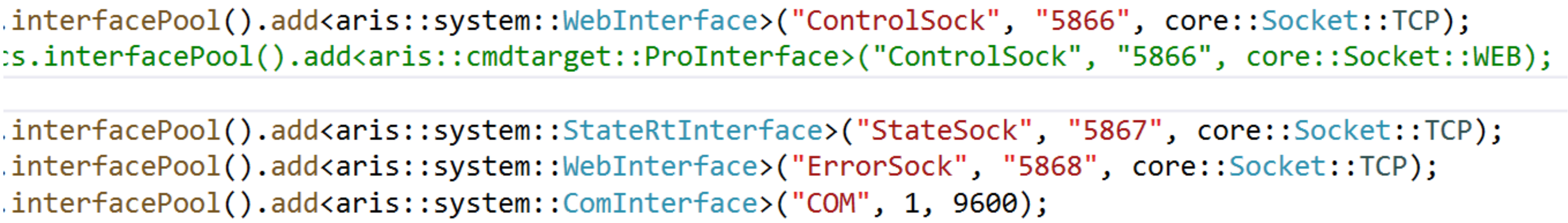
- executeRT()

```
errorinfoPool->add<codeit::system::ErrorInfo>
("plan over time", -2001, "ERROR", "规划超时", "plan over time");
errMap.insert(pair<std::int32_t, string>(-2001, "plan over time"));
return -2001;
```

- collectNrt():不能抛出异常

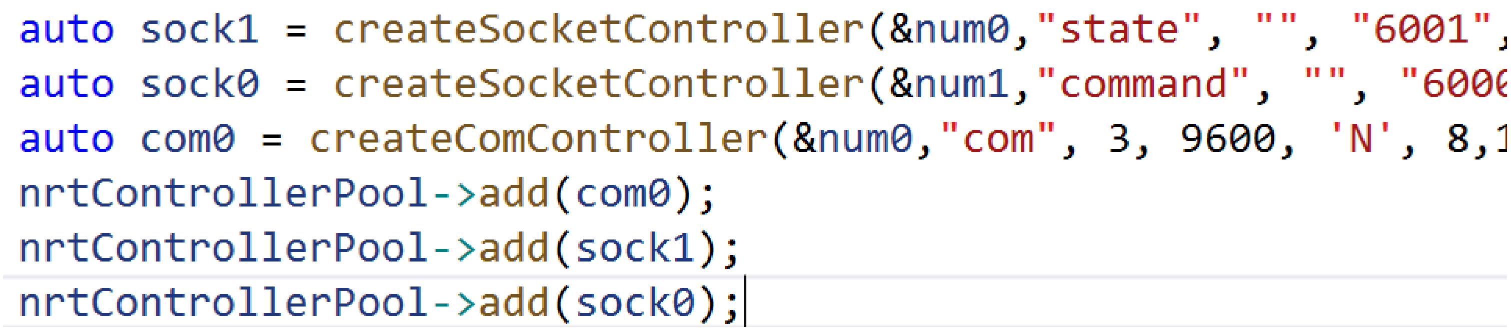
Codeit——其他封装功能

- InterfacePool



```
.interfacePool().add<aris::system::WebInterface>("ControlSock", "5866", core::Socket::TCP);
s.interfacePool().add<aris::cmdtarget::ProInterface>("ControlSock", "5866", core::Socket::WEB);

.interfacePool().add<aris::system::StateRtInterface>("StateSock", "5867", core::Socket::TCP);
.interfacePool().add<aris::system::WebInterface>("ErrorSock", "5868", core::Socket::TCP);
.interfacePool().add<aris::system::ComInterface>("COM", 1, 9600);
```

- ControllerPool



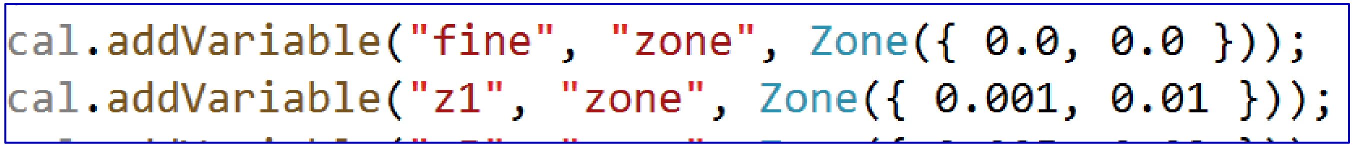
```
auto sock1 = createSocketController(&num0,"state", "", "6001",
auto sock0 = createSocketController(&num1,"command", "", "600
auto com0 = createComController(&num0,"com", 3, 9600, 'N', 8,1
nrtControllerPool->add(com0);
nrtControllerPool->add(sock1);
nrtControllerPool->add(sock0);
```

- Log

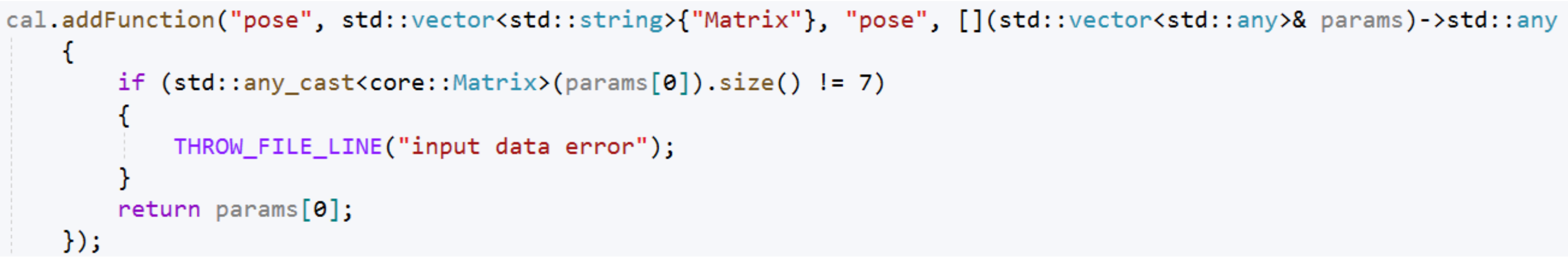
基于streambuf实现线程安全的日志功能，几百行代码；**实时线程数据交互**

Codeit——其他封装功能

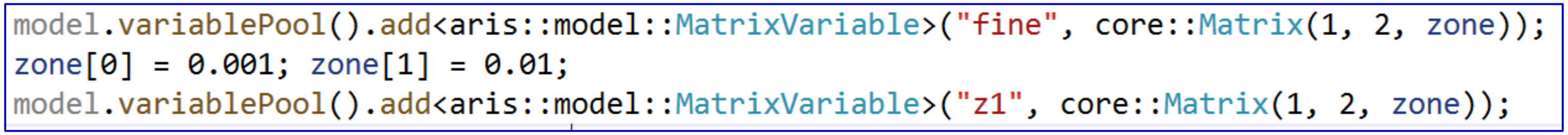
- 用户数据



```
cal.addVariable("fine", "zone", Zone({ 0.0, 0.0 }));
cal.addVariable("z1", "zone", Zone({ 0.001, 0.01 }));
```



```
cal.addFunction("pose", std::vector<std::string>{"Matrix"}, "pose", [](std::vector<std::any>& params)->std::any
    {
        if (std::any_cast<core::Matrix>(params[0]).size() != 7)
        {
            THROW_FILE_LINE("input data error");
        }
        return params[0];
    });
```



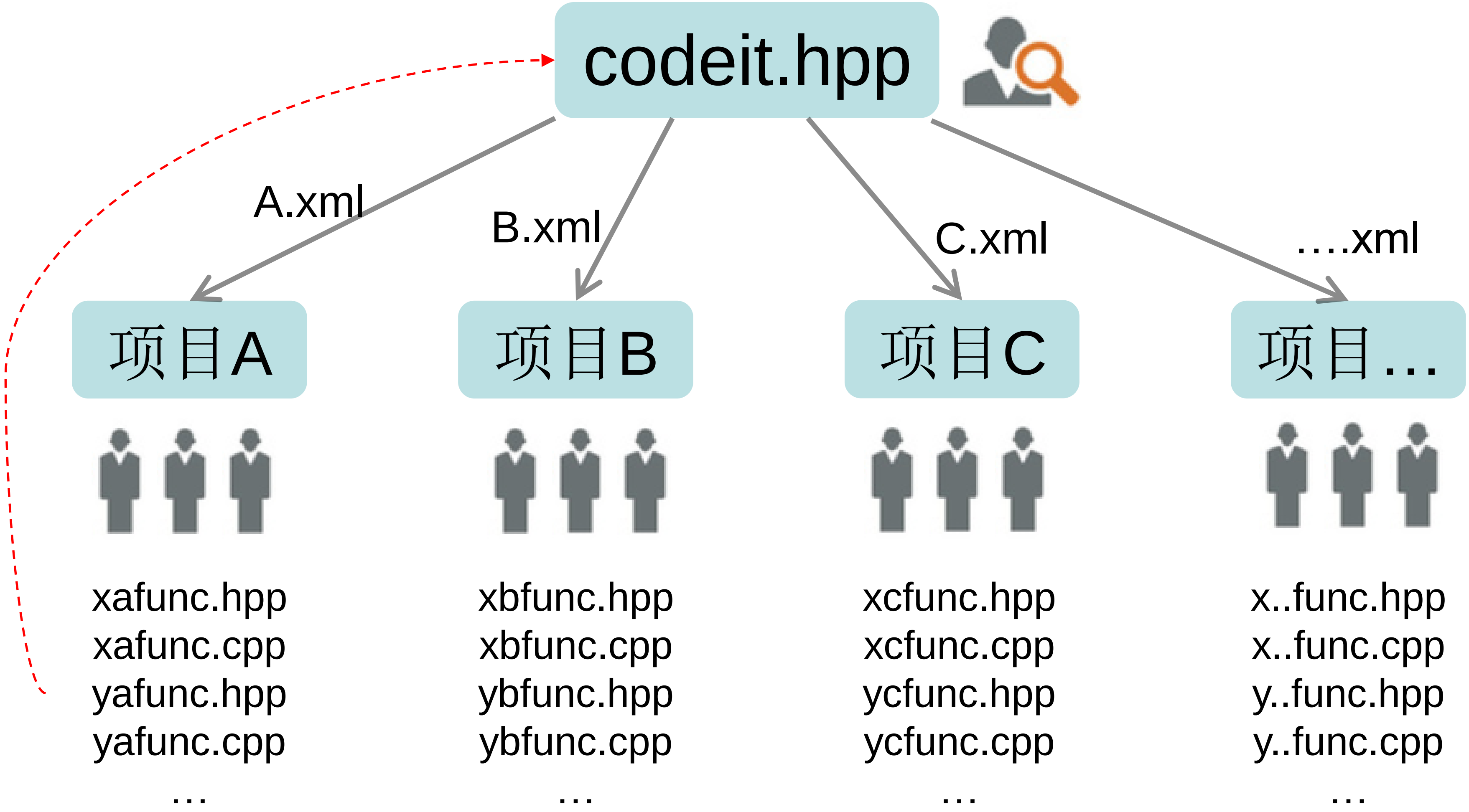
```
model.variablePool().add<aris::model::MatrixVariable>("fine", core::Matrix(1, 2, zone));
zone[0] = 0.001; zone[1] = 0.01;
model.variablePool().add<aris::model::MatrixVariable>("z1", core::Matrix(1, 2, zone));
```

- 精简高效的矩阵库（千行）

- 旋量库：欧拉角、旋转矩阵、四元数、轴角、旋量间转换，一阶导、二阶导间转换

Codeit——版本管控

```
find_package(codeit REQUIRED PATHS C:/codeit/codeit-1.0.0)
```



codeit.hpp

A.xml

B.xml

C.xml

….xml

项目A

项目B

项目C

项目…

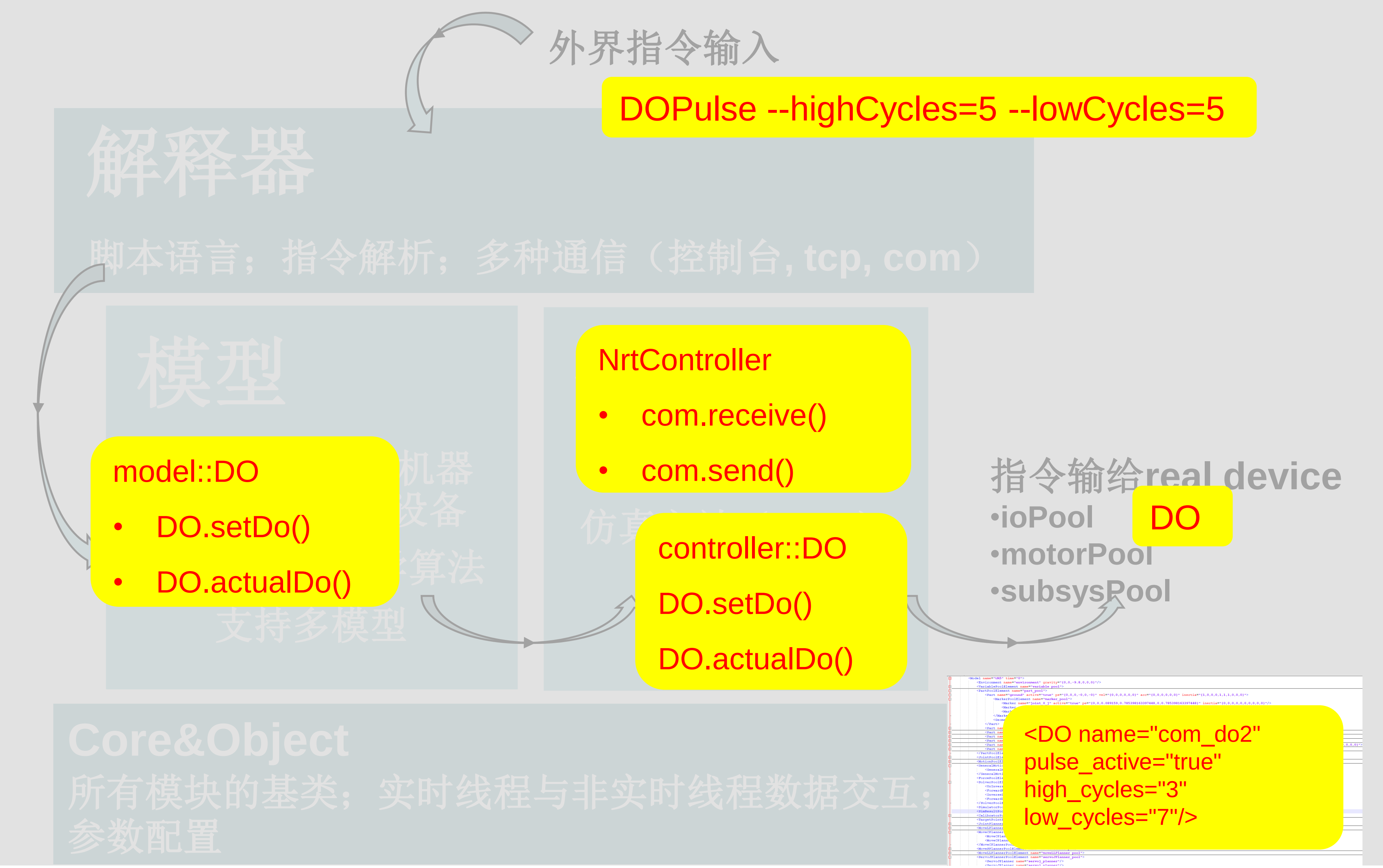
xafunc.hpp
xafunc.cpp
yafunc.hpp
yafunc.cpp
…

xbfunc.hpp
xbfunc.cpp
ybfunc.hpp
ybfunc.cpp
…

xcfunc.hpp
xcfunc.cpp
ycfunc.hpp
ycfunc.cpp
…

x..func.hpp
x..func.cpp
y..func.hpp
y..func.cpp
…

Codeit 开发示例——串口下流水灯控制



外界指令输入

DOPulse --highCycles=5 --lowCycles=5

解释器
脚本语言；指令解析；多种通信（控制台, tcp, com）
模型
机器
设备
算法
支持多模型
仿真
Core&Object
所有模块的基类；实时线程与非实时线程数据交互；参数配置
指令输给real device
•ioPool
•motorPool
•subsysPool

NrtController
- com.receive()
- com.send()

model::DO
- DO.setDo()
- DO.actualDo()

controller::DO

DO.setDo()

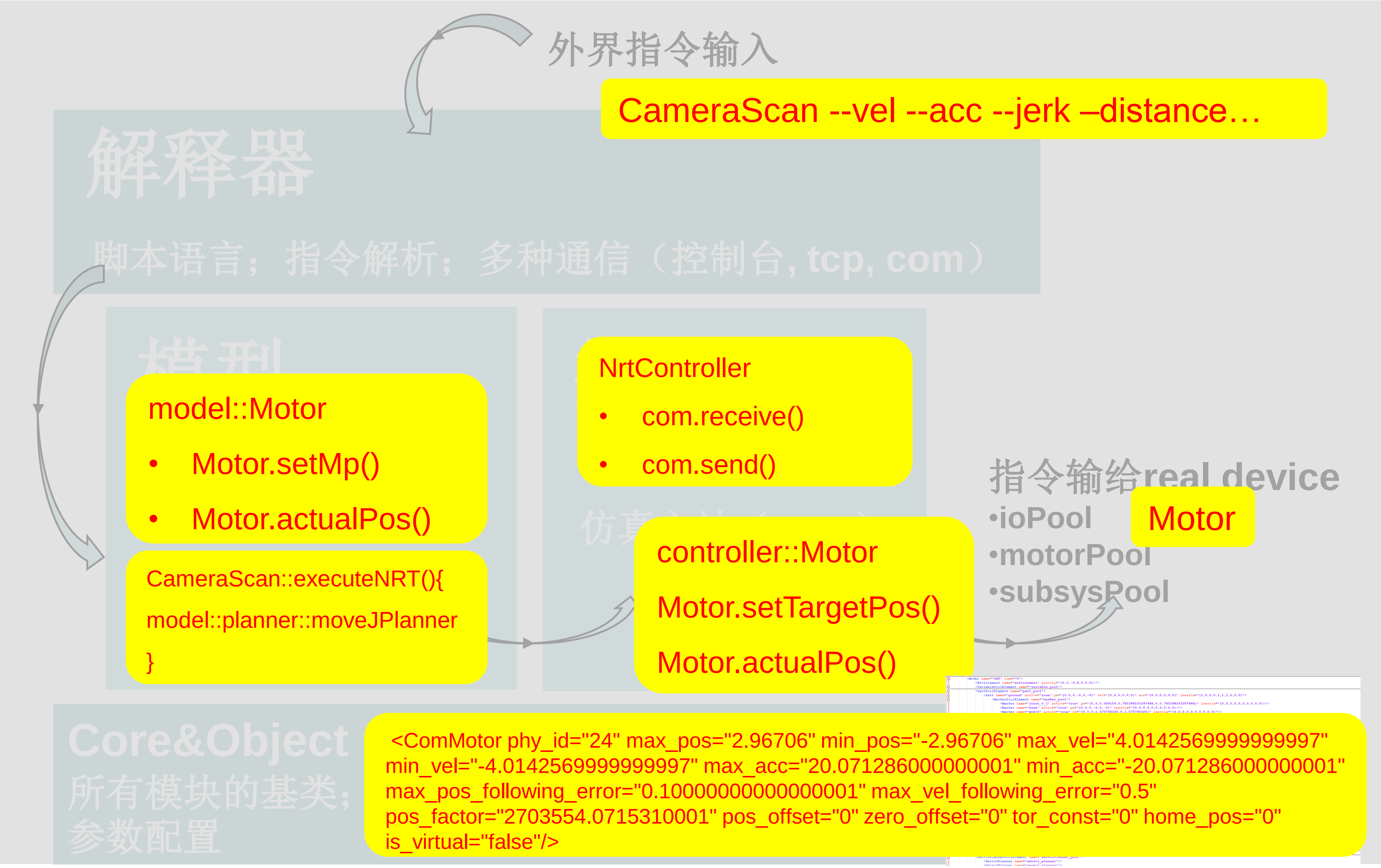
DO.actualDo()

DO

```
<Model name="UR5" time="0">
    <Environment name="environment" gravity="{0,0,-9.8,0,0,0}"/>
    <VariablePoolElement name="variable_pool">
    <PartPoolElement name="part_pool">
        <Part name="ground" active="true" pe="{0,0,0,-0,0,-0}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{1,0,0,0,1,1,1,0,0,0}">
            <MarkerPoolElement name="marker_pool">
                <Marker name="joint_0_j" active="true" pe="{0,0,0.089159,0.785398163397448,0,0.785398163397448}" inertia="{0,0,0,0,0,0,0,0,0,0}"/>
```

<DO name="com_do2"
pulse_active="true"
high_cycles="3"
low_cycles="7"/>

Codeit 开发示例——串口下电机控制



外界指令输入

CameraScan --vel --acc --jerk –distance…

model::Motor

- Motor.setMp()
- Motor.actualPos()

```
CameraScan::executeNRT(){
model::planner::moveJPlanner
}
```

NrtController

- com.receive()
- com.send()

controller::Motor

Motor.setTargetPos()

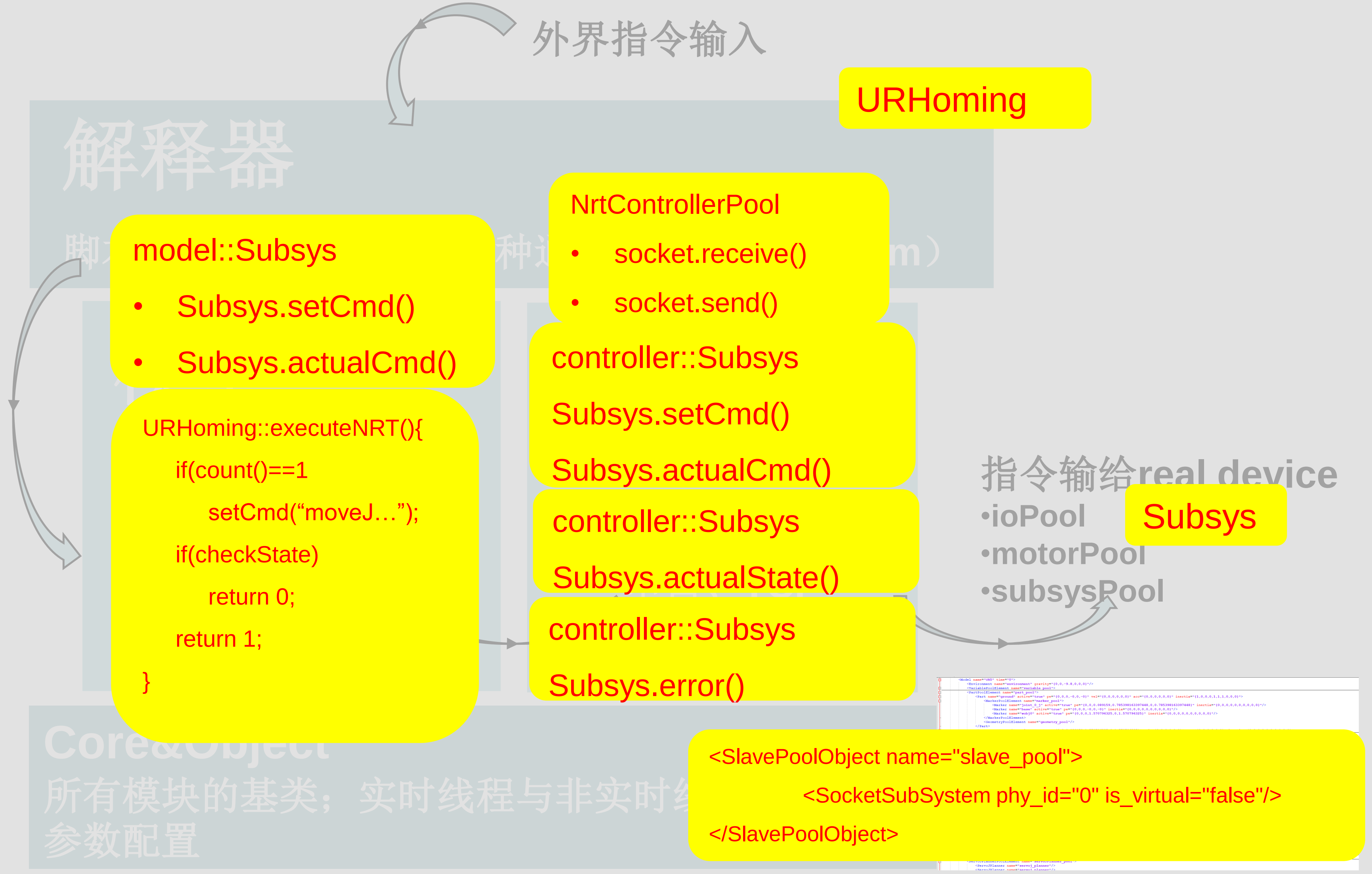
Motor.actualPos()

Motor

```
<Model name="UR5" time="0">
    <Environment name="environment" gravity="{0,0,-9.8,0,0,0}"/>
    <VariablePoolElement name="variable_pool">
    <PartPoolElement name="part_pool">
        <Part name="ground" active="true" pe="{0,0,0,-0,0,-0}" vel="{0,0,0,0,0,0}" acc="{0,0,0,0,0,0}" inertia="{1,0,0,0,1,1,1,0,0,0}">
            <MarkerPoolElement name="marker_pool">
                <Marker name="joint_0_j" active="true" pe="{0,0,0.089159,0.785398163397448,0,0.785398163397448}" inertia="{0,0,0,0,0,0,0,0,0,0}"/>
                <Marker name="base" active="true" pe="{0,0,0,-0,0,-0}" inertia="{0,0,0,0,0,0,0,0,0,0}"/>
                <Marker name="wobj0" active="true" pe="{0,0,0,1.570796325,0,1.570796325}" inertia="{0,0,0,0,0,0,0,0,0,0}"/>
```

```
<ComMotor phy_id="24" max_pos="2.96706" min_pos="-2.96706" max_vel="4.0142569999999997" min_vel="-4.0142569999999997" max_acc="20.071286000000001" min_acc="-20.071286000000001" max_pos_following_error="0.10000000000000001" max_vel_following_error="0.5" pos_factor="2703554.0715310001" pos_offset="0" zero_offset="0" tor_const="0" home_pos="0" is_virtual="false"/>
```

Codeit 开发示例——控制UR的控制器



URHoming

NrtControllerPool

- socket.receive()
- socket.send()

model::Subsys

- Subsys.setCmd()
- Subsys.actualCmd()

```
URHoming::executeNRT(){
    if(count()==1
        setCmd("moveJ…");
    if(checkState)
        return 0;
    return 1;
}
```

controller::Subsys

Subsys.setCmd()

Subsys.actualCmd()

controller::Subsys

Subsys.actualState()

controller::Subsys

Subsys.error()

Subsys

```
<SlavePoolObject name="slave_pool">
        <SocketSubSystem phy_id="0" is_virtual="false"/>
</SlavePoolObject>
```

谢　谢

Q&A