



DroNE 中子实验数据处理框架

用户指南

中国散裂中子源中子科学部谱仪软件系统

Data Analysis & Software Group,

Neutron Science Division of CSNS

状态 发布

创建日期 2017-10-03

编号

文件名 系统设计、开发状态和用户测试

作者 田浩来

田浩来-2018年6月25日版

版本历史

修改者	日期	联系方式	备注
田浩来	2017-12-5	tianhl@ihep.ac.cn	起草
田浩来	2018-2-10	tianhl@ihep.ac.cn	定稿
田浩来	2018-5-10	h.tian@qmul.ac.uk	补充

1 摘要

该文档描述了 DroNE(Data pROcessing software suite for Neutron Experiments, 中子实验数据处理软件框架)项目正式发布版本的设计概要、关键组件和用户使用和测试指南。DroNE 软件框架的开发, 满足了中国散裂中子源实验数据处理对基础软件提出的需求, 实现了《设计报告》中的全部功能要求。本文档最后部分为用户使用和测试指南, 便于用户测试框架功能。

此开发报告作为中国散裂中子源软件系统交付文档的一部分, 是软件系统开发活动初步成果总结, 用于用户了解和测试该系统的功能, 并作为系统维护指南应用在软件运维阶段。该文档应当随软件系统的升级而不断更新, 以全面现实的反映软件开发和运维活动的现状。

关键词: 架构设计, 关键组件, 用户指南, 软件测试, 数据处理软件框架

目录

1 摘要	4
2 前言	6
2.1 文档目标	6
2.2 项目范围	6
2.3 术语定义	7
3 框架设计概况.....	9
3.1 框架设计	9
3.2 技术选型	14
4 框架重要组件.....	15
4.1 任务(Task).....	16
4.2 算法(Algorithm).....	19
4.3 数据服务(DataSvc)	21
4.4 控制服务(CtrlSvc).....	23
4.5 事件通知(Incident)	24
4.6 句柄操作(Handle).....	26
5 用户使用和测试.....	28
5.1 谱仪数据处理作业	28
5.2 谱仪探测器模拟	32
5.2 分布式环境 RPC 交互	34

2 前言

2.1 文档目标

该文档描述了 DroNE(Data pROcessing software suite for Neutron Experiments, 中子实验数据处理软件框架)项目正式发布版本的设计概要、关键组件和用户使用和测试指南。DroNE 软件框架的开发, 满足了中国散裂中子源实验数据处理对基础软件提出的需求, 实现了《设计报告》中的全部功能要求。本文档最后部分为用户使用和测试指南, 便于用户测试框架功能。

此开发报告作为中国散裂中子源软件系统交付文档的一部分, 是软件系统开发活动初步成果总结, 用于用户了解和测试该系统的功能, 并作为系统维护指南应用在软件运维阶段。该文档应当随软件系统的升级而不断更新, 以全面现实的反映软件开发和运维活动的现状。

2.2 项目范围

DroNE 软件框架的开发目标, 是构建中子散射实验通用数据框架, 该框架将会应用在中国散裂中子源首期中子谱仪(通用粉末衍射谱仪、多功能反射谱仪和小角中子散射谱仪)中, 包含样品和探测器模拟、原始数据处理、探测器重建和其他数据处理任务, 运行环境包括在线交互式数据处理环境、离线批处理系统和独立运行的

数据处理软件。该框架未来也将会应用在中国散裂中子源其他待建谱仪的相关数据处理软件中。

框架开发的意义在于, 通过复用通用组件, 减轻具体谱仪数据处理软件的开发难度和工作量。DroNE框架一方面具有高内聚的特性, 通过规范组件接口, 提供框架对组件的反向控制功能, 这一点区别于软件库, 软件库往往被上层用户应用调用; 另一方面, DroNE框架还具有开放性的特点, 可以方便的集成其他软件框架, 各内部组件也可以独立升级。

本文档仅描述DroNE软件框架的总体架构, 介绍关键组件的实现。本文档作为用户指南和测试指南, 不涉及具体数据处理任务和具体算法实现。

2.3 术语定义

CSNS	中国散裂中子源(China Spallation Neutron Source)
DroNE	中子实验数据处理软件框架(Data pROcessing software suite for Neutron Experiments)
SNiPER	非对撞物理实验软件框架 (Software for Non-collider Physics Experiments)
API	应用程序接口(Application Programming Interface)
JUNO	江门中微子实验 (Jiangmen Underground Neutrino Observatory)
CSNS	中国散裂中子源(China Spallation Neutron Source)

LHAASO 高海拔宇宙线观测站

(Large High Altitude Air Shower Observatory)

RPC 远程过程调用(Remote Procedure Call)

田浩来-2018年6月25日版

3 框架设计概况

这一章将会介绍 CSNS 数据处理基础框架 DroNE 正式发布版本的设计和技术。

3.1 框架设计

CSNS 谱仪软件系统自行研发的数据处理框架（DroNE），应该实现一个以数据为中心，数据和算法分离，采用面向接口编程的数据处理框架。这种架构的优势在于，对于特定数据，数据的结构往往是稳定的，数据和算法的分离，可以隔离算法的发展变化；其次，整个数据处理流程都以数据作为中介，算法之间完全解耦合，算法的升级不影响其他算法的运行。

DroNE底层由C++实现，上层由Python实现，并提供C++和Python API供外部软件组件调用。框架底层主要实现数据处理算法和内存/数据管理，C++丰富的编程范式，可以在保证程序运行效率的同时，提供很好的扩展性和灵活性。框架上层主要实现灵活动态的数据处理工作流，以及与外部组件的交互，Python动态脚本语言的特性，以及丰富的第三方软件包，使得开发者和用户不但可以设计针对具体谱仪和特定任务的操作原语，也可以方便快捷的开发图形界面、RESTful Web APIs等交互服务。

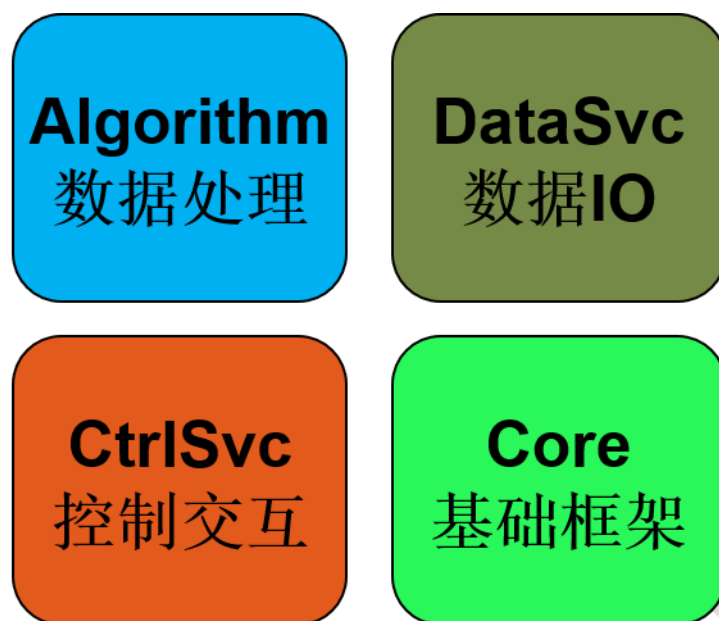


图1 DroNE总体架构

DroNE 数据处理框架由四个模块构成，如图 1 所示，分别是内核模块(Core)、控制服务模块(CtrlSvc)、数据服务模块(DataSvc)和算法模块(Algorithm)。算法是实际处理数据的组件，主要由算法工程师和谱仪科学家开发；框架通过算法接口实现对算法的控制；框架通过数据服务(DataSvc)实现对数据的访问和更新，数据服务由应用程序开发工程师完成，其中瞬态数据结构由谱仪科学家和框架工程师共同确定；框架内部的控制由控制服务(CtrlSvc)实现，该模块主要由应用程序工程师开发；组件之间的交互在内核模块(Core)中定义，内核模块由框架工程师负责。此外 DroNE 应该提供 Python 接口，具体应用的运行和配置应是纯粹的 Python 程序(PyDroNE)，框架的功能也应该可以被其他 Python 程序无缝调用，PyDroNE 模块由谱仪科学家和应用程序工程师共同开发。

3.1.1 内核模块

谱仪数据处理框架的运行，在空间上表现为包括算法和服务的多组件的协同工作；在时间上表现为一系列算法的有序调用，和数据的一系列 IO 操作，这些工作都是在内核中预先定义。组件的发现和调用可以通过接口实现。接口是框架定义的对象与框架交互的协议，只有遵循接口规范的功能模块，才能被框架识别和调用。接口只是协议，其实现表现为由接口派生的具体类。因此，实现可以独立于接口变化，保证了框架接口的稳定和实现的灵活。从计算机的角度来看，最基本的接口应该包括一个 key-value 字典，其中 key 是接口的名字，而 value 是接口所对应的类型。所以对于 C++ 这种强类型语言，通过接口的名字，就能判断接口所对应的类型。接口管理的职责是按照接口名创建、销毁和管理具体的接口实例。接口控制器完成对组件的注册、命名和寻址，组件之间通过接口控制器发现对方。这种方法减少了组件间交互所需的点对点连接的数量，减少了组件集成的复杂度。

3.1.2 数据服务模块

谱仪数据处理框架按照数据的处理特点，把数据分为三类：静态数据，瞬态数据和直方图数据。其中谱仪配置、探测器描述和实验参数设置等数据属于静态数据，在整个数据处理过程中不发生改变；从数据获取系统传输来的探测器电子学信号，以及从控制系统传输过来的样品环境参数等数据，属于瞬态数据，瞬态数据在整个

数据处理过程中不断发生变化；而静态数据和瞬态数据会被处理得到一些直方图数据，直方图数据在程序运行过程中会被算法更新。

数据服务(DataSvc)模块包括一个高效灵活的数据缓存服务和数据输入输出服务。数据缓存服务管理着所有静态数据、瞬态数据和直方图数据的创建、更新、接入和删除；日志服务，实现 log 信息的分级输出，方便用户代码的调试。数据的输入输出服务把数据获取系统和控制系统传输过来的数据，或者从数据文件读入的数据流，加载进入数据缓存服务，并把数据缓存服务中的数据保存到文件中。另外，根据用户使用过程中出现的新需求，不断对其扩展和完善。

3.1.3 控制服务模块

控制服务(CtrlSvc)模块实现了 Incident/Handle 的通讯模式，Incident 发出控制指令，订阅了该 Incident 的 Handle 响应该指令，并可以返回响应结果。Incident 和 Handle 都可以通过 Python 或者 C++ 实现，不同语言的实现不影响其互操作性。同时控制服务不但需要支持程序内部控制，也同时要支持外部控制指令的响应，因此必须提供网络中间件支持。

3.1.4 算法模块

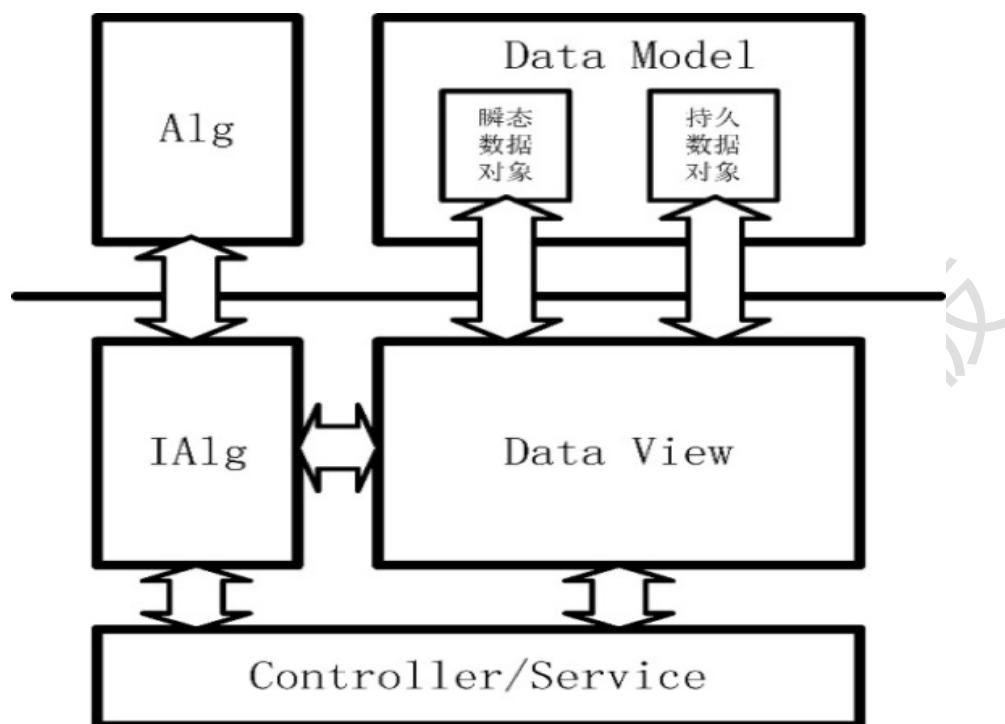


图 2 算法和数据服务的交互

算法是应用处理数据的核心模块，如图2所示，整个数据处理流程都以数据作为中介，算法之间完全解耦合，算法的升级不影响其他算法的运行。图一描述了这种框架的基本思想，算法（Alg）是实际处理数据的组件；框架通过算法接口(IAlg)实现对算法的控制；框架通过数据视图（Data View）实现对数据的访问和更新；数据模型（Data Model）描述的是数据的表现形式，包括以内存数据形式表现的瞬态数据对象和以文件形式表现的持久数据对象；而框架本身的功能由控制器（Controller）和服务（Services）提供。

3.2 参与角色

我们考察了多种基础框架技术，包括欧洲核子中心LHCb实验开发的Gaudi，以及RAL和ORNL联合开发的中子散射数据处理框架Mantid。最终选定SNiPERKernel作为DroNE底层技术

SNiPER是中国科学院高能物理研究所自行研发的数据处理框架，同时服务于JUNO实验和LHAASO实验，有很好的二次开发性和方便的技术支持。SNiPERKernel是为SNiPER提供了核心软件功能。

除此之外，DroNE计划采用CMake作为自身的编译环境，仅依赖C++标准库STL，准标准库Boost和Python2.7。

4 框架重要组件

本章将会介绍 DroNE 框架的重要组件。

模块是框架的基本物理单元，一组具有相似功能的组件被放置在同一文件夹内，成为一个模块。每个模块都会生成独立的动态链接库，在运行时统一加载。

DroNE 的基本逻辑单元为组件，组件是一个能完成特定功能并实现了预定义接口的类。组件通过接口与框架中的其他组件进行交互，组件之间交互的动作，由框架预先规范。

基于 DroNE 框架的应用程序的运行，在空间上表现为各功能组件的相互调用，在时间上表现为组件的依次有序执行。

4.1 任务(Task)

Task 是数据处理作业的执行组件，本章介绍 Task 组件的功能、接口和交互。

4.1.1 功能

Task组件的主要功能，就是管理特定计算任务相关资源的加载、配置和调用。Task所管理的计算任务，可能是长期任务，如DroNE本身就是Task的具体实现，负责管理DroNE进程整个运行周期；也可能是短期任务，如一个远程调用命令的执行。Task基类以C++实现，通过Boost.Python提供Python的API，派生类可以用Python或者C++实现。

Task内部最重要的数据类型为一保存了DLE(可加载动态组件)信息的映射表，可以通过名字加载相应组件。此外Task还管理着与计算任务相关的服务和算法列表。每个数据处理进程必须要有一个顶级Task，负责全局相关组件的管理，Task还可以创建和管理子Task。

4.1.2 重要 Task 简介

1. DroNE，这是DroNE执行离线数据处理任务的顶级Task。该Task从数据源读入原始数据，解析后依脉冲(Pulse)送入算法序列进行处理，输出重建后的中子击中事件。

2 . DroNEOnline, 这是DroNE执行在线数据处理任务的顶级Task。DroNEOnline除了完成原始数据解析和重建, 还可以通过RPC与其他程序交互, 动态改变运行状态。

3 . NeonRPCTask, 这是DroNEOnline通过RPC与其他程序交互的子Task, 该Task通过RPC技术读取远程调用指令, 经过有效性检查后, 调用相应DroNE内部命令(Incident), 内部命令执行完成后, 向RPC返回执行结果。不同于前两个C++实现的顶级Task, 该Task为Python实现的轻量级Task。

4.1.3 重要接口

- 1 . initialize/execute/finalize, 此接口为DroNE组件初始化、执行和结束函数, 有默认实现。
- 2 . addSvc/addAlg/addTask, 此接口为Task创建下属服务、算法和子任务函数。
- 3 . clearSvc/clearAlg/clearTask, 此接口为Task删除下属服务、算法和子任务函数。
- 4 . find, 此接口为Task查找下属服务、算法和子任务函数。
- 5 . handle, 此接口为Task相应其他组件调用的函数。

4.1.4 依赖关系

- 1 . incident, 外界调用指令, 与handle对应使用。

2. DLElement, 可加载动态组件, 计算任务执行的相关组件的基类。
3. Algorithm, 计算任务执行算法。
4. Service, 计算任务所需的服务, 包括必须的数据服务和控制服务。

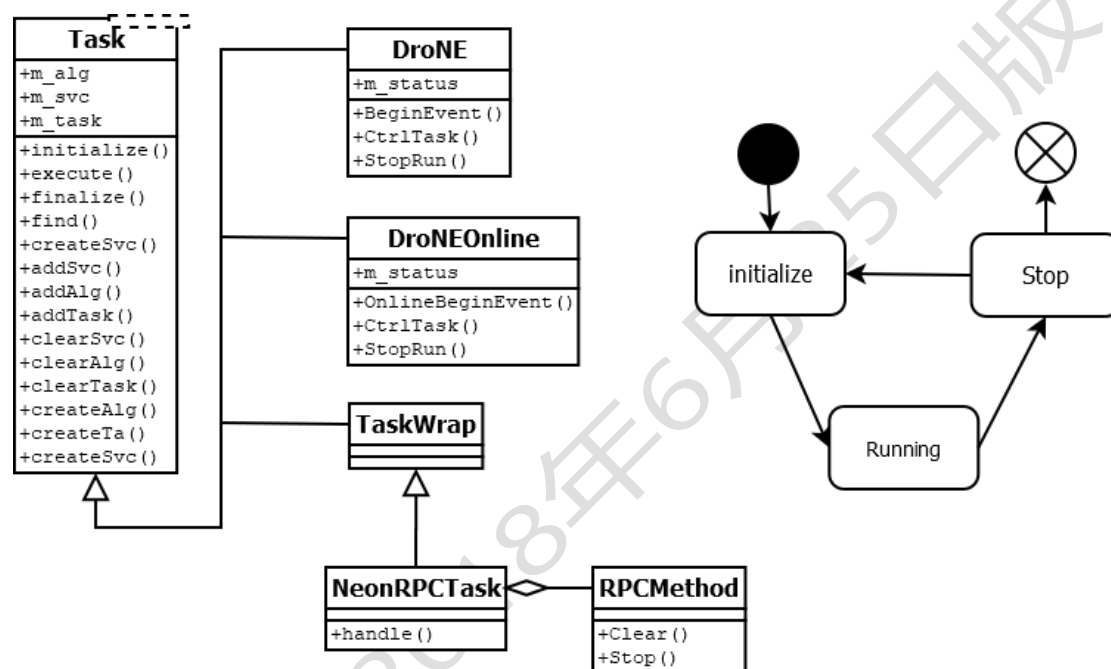


图 3 Task 的类图和 Task 的状态转移图

4.2 算法(Algorithm)

Algorithm 是特定计算的具体执行模块，所有计算任务最终都是由一系列算法构成的，这些计算任务可能是模拟、重建、映射、求和等等。所有 Algorithm 必须由算法基类(AlgBase)派生。

4.2.1 功能

Algorithm组件的主要功能，就是把一个规定数据结构的输入数据集，通过某些特定操作，转换为另一个规定数据结构的输出数据集。这些特定操作依赖对输入数据集的数学变换公式，以及该变换公式的内部参数。这些内部参数，可以由算法自身通过计算得出，比如模拟算法中的随机数；也可以通过外部配置得到。

算法之间不应该有直接的交互，所有算法之间的数据传递，应该通过数据服务(DataSvc)作为中介完成，实现算法之间的解耦和。算法的调用，应该由管理它的Task实现。

4.2.2 重要 Algorithm 简介

- 1 . DumpAlg，这是算法类的示范例子，展示了算法如何从数据服务中获取数据，利用数学变换处理数据，并最终向数据服务中写入数据。
- 2 . *RecAlg，这是相关探测器的重建算法。
- 3 . *SimAlg，这是相关探测器的模拟算法。
- 4 . *MapAlg，这是相关谱仪的映射算法。

注: *代表谱仪探测器名

4.2.3 重要接口

1. initialize/ finalize, 此接口为Algorithm组件初始化和结束函数, 有默认实现。
2. execute, 此接口为Algorithm组件执行函数, 每个算法类都有各种不同的实现。

4.2.4 依赖关系

1. Property, 算法参数的申明和设置。
2. DataSvc, 算法的输入数据源和输出数据目的地。
3. Task, 管理算法的组件。

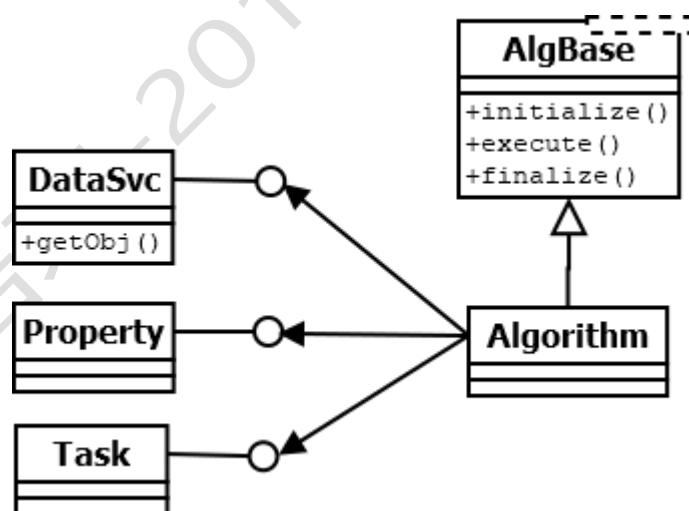


图 4 Algorithm 的类图

4.3 数据服务(DataSvc)

DataSvc 是瞬态数据的“逻辑”存放地址，DroNE 的数据服务管理着一个树形数据结构，每个数据对象(DataObject)都存放在这个树形结构的一个分支上。

4.3.1 功能

DataSvc组件的主要功能，就是管理算法输入和输出的数据对象(DataObject)。DataSvc内部是一个树形结构，DataSvc本身是该树形结构的入口(Root)，树形结构中保存着两类数据结构：分支(Branch)和数据对象(DataObject)。其中分支是一种特殊的数据对象，其特点是分支下面可以生成新的分支，但是其他数据对象下面不能再接入新的数据对象(包括Branch和其他DataObject)。每个数据对象都有自己的路径名和类型，其他组件通过路径名获取数据对象，通过类型解析数据对象。

4.3.2 重要 DataSvc 简介

- 1 . GPPDDataSvc, CSNS通用粉末衍射谱仪的数据服务。
- 2 . MRDataSvc, CSNS多功能反射谱仪的数据服务。
- 3 . SANSDataSvc, CSNS小角中子散射谱仪的数据服务。

4.3.3 重要接口

- 1 . branch, 下级分支。

- 2 . find, 获取数据对象或者下级分支。
- 3 . regObj, 把数据对象挂载在某个分支上

4.3.4 依赖关系

- 1 . DataObject, 树形结构中挂载的数据对象。

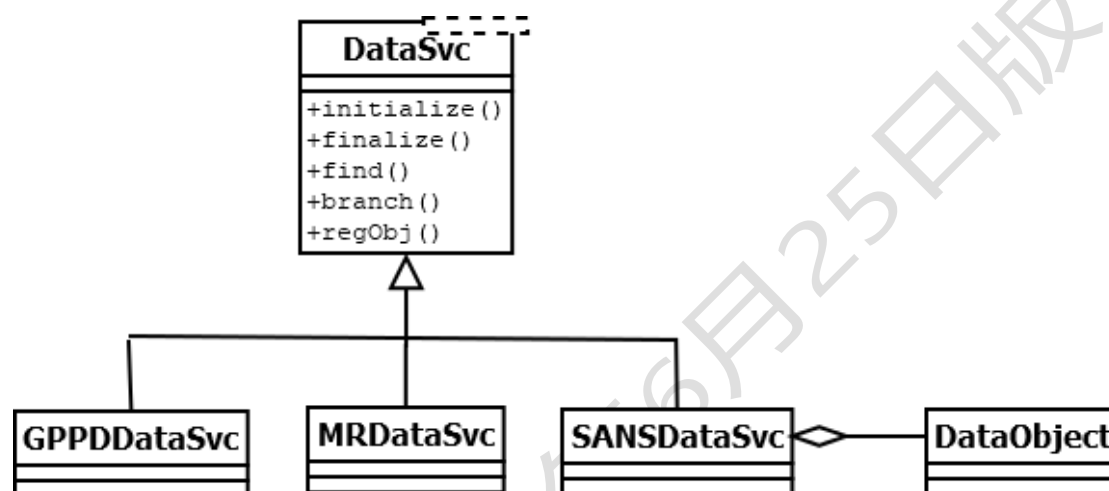


图 5 DataSvc 的类图

4.4 控制服务(CtrlSvc)

CtrlSvc 是管理着整个应用的事件通知及其对应的句柄操作。

4.4.1 功能

CtrlSvc组件的主要功能，就是为应用所有的事件通知，绑定和管理合适的句柄操作。加载不同的控制服务，可以让应用对相同的事件通知，执行不同的句柄操作。

4.4.2 重要 CtrlSvc 简介

- 1 . GPPDCtrlSvc, CSNS通用粉末衍射谱仪的控制服务。
- 2 . MRCtrlSvc, CSNS多功能反射谱仪的控制服务。
- 3 . SANSCtrlSvc, CSNS小角中子散射谱仪的控制服务。

4.4.3 重要接口

- 1 . initialize, 创建事件通知和句柄操作的映射关系。
- 2 . finalize, 清除事件通知和句柄操作的映射关系。

4.4.4 依赖关系

- 1 . incident, 事件通知。
- 2 . handle, 句柄操作。

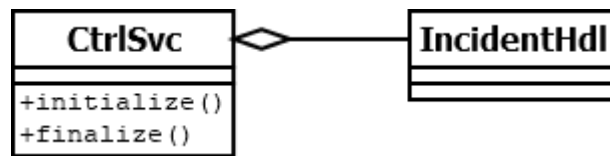


图 6 CtrlSvc 的类图

田浩来-2018年6月25日版

4.5 事件通知(Incident)

Incident 是应用程序某组件调用其他组件规定操作时，发出特定事件通知。

4.5.1 功能

Incident组件的主要功能，就是应用程序某组件需要其他组件执行规定操作时，发出的特定事件通知。在这里需要强调的是，Incident本身并不需要知道哪些组件会响应其发出的事件通知，甚至Incident发出事件通知后，可以无需响应。这样就实现了事件通知和具体操作的解耦和。Incident同时支持以Python和C++编写，其中Python编写的Incident除了具有C++编写的Incident的所有功能，还提供了定时事件通知和周期性事件通知的功能，定时事件可以规定在某个时间点触发该事件通知，周期性事件通知可以规定事件通知周期性触发。

4.5.2 重要 Incident 简介

- 1 . BeginEvt, DroNE在每个脉冲事件(Pulse)执行前，发出的事件通知。
- 2 . EndEvt, DroNE在每个脉冲事件(Pulse)执行后，发出的事件通知。
- 3 . StopRun, DroNE准备结束整个应用程序运行时，发出的事件通知。

4.5.3 重要接口

1. name, 事件通知的名字, 句柄操作根据事件的名字判断是否响应。
2. fire, 事件通知的触发操作, 触发后DroNE会自动调用绑定的句柄操作。

4.5.4 依赖关系

1. Handle, 事件通知对应的句柄操作。
2. CtrlSvc, 管理事件通知和句柄操作的映射关系。

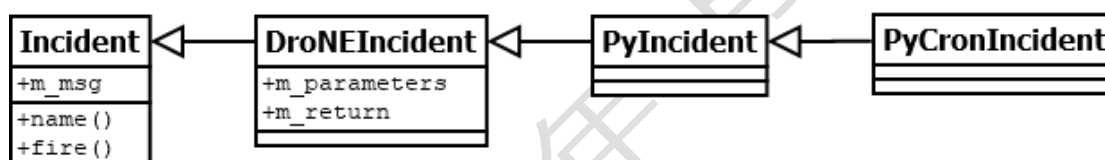


图 7 Incident 的类图

4.6 句柄操作(Handle)

Handle(句柄操作类)响应 Incident 的事件通知, 通过 Incident 传入的参数执行规定操作, 最后向 Incident 返回操作结果。DroNE 的 Handle 同时响应 C++和 Python 编写的 Incident 事件通知。

4.6.1 功能

Handle组件的主要功能, 就是响应Incident事件通知, 解析 Incident传入的操作参数, 执行规定操作, 最后向Incident返回操作结果。其中解析操作参数和返回操作结果为可选动作。

4.6.2 重要 Handle 简介

- 1 . BeginEvtHdl, 每次脉冲事件(Pulse)数据处理算法序列执行前, 加载解析原始数据的句柄操作。
- 2 . EndEvtHdl, 每次脉冲事件(Pulse)数据处理算法序列执行后, 执行后续的清理工作的句柄操作。
- 3 . ClearDataSvcHdl, 清理数据服务指定数据对象的句柄操作。

4.6.3 重要接口

- 1 . regist, 绑定指定Incident事件通知。
- 2 . listen, 监听绑定的Incident事件通知。
- 3 . handle, 接收到绑定Incident事件通知后, 操作执行部分。

4.6.4 依赖关系

1. Incident, 触发句柄操作的事件通知。
2. CtrlSvc, 管理incident事件通知和Handle句柄操作的服务。
3. Task, 规定了事件通知和句柄操作的作用域范围。

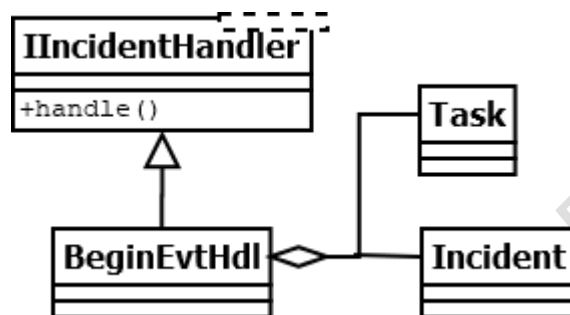


图 8 Handler 的类图

5 用户使用和测试

本框架的最终目的，是为 CSNS 提供一套可复用的软件框架，应用在首期三台中子谱仪，以及后期待建中子谱仪的模拟、离线/在线数据处理环境中。本章介绍首期三台谱仪(GPPD, MR 和 SANS)在线/离线/模拟程序的简单使用。

5.1 谱仪数据处理作业

5.1.1 GPPD

```
1 import DroNECore
2 import CtrlSvc
3 import DataSvc
4
5 task = DroNECore.DroNE("task")
6 task.asTop()
7 task.setLogLevel(3)
```

代码块 1

1-3行 加载DroNE模块

5行 初始化应用程序任务

6-7行 把本任务设置为顶层任务，并设置日志输出级别

```
1 task.property("svcs").append("CtrlSvc")
2 task.property("svcs").append("GPPDDataSvc/DataSvc")
3 task.property("svcs").append("RawDataInputSvc/DataInputSvc")
4 task.property("svcs").append("FileInputSvc/DataProvideSvc")
5
6 import Algorithms
7 task.property("algs").append("GPPDSNDRecAlg")
8 task.property("algs").append("GPPDSNDMapAlg")
10 task.property("algs").append("RunningInfAlg")
```

代码块 2

- | | |
|-------|-----------|
| 1-4行 | 初始化所需服务组件 |
| 6行 | 加载算法模块 |
| 7-10行 | 初始化所需算法组件 |

```
1 task.run()
```

代码块 3

- | | |
|----|--------|
| 1行 | 运行顶级任务 |
|----|--------|

5.1.2 MR

多功能反射谱仪的运行和粉末衍射谱仪类似，只是在需要加载不同的服务模块和算法模块，即把 GPPD 中的代码块 2 替换成代码块 4。

```
1 task.property("svcs").append("CtrlSvc")
2 task.property("svcs").append("MRDataSvc/DataSvc")
3 task.property("svcs").append("He3TRawDataInputSvc/DataInputSvc")
4 task.property("svcs").append("FileInputSvc/DataProvideSvc")
5
6 import Algorithms
7 task.property("algs").append("MRHe3TRecAlg")
8 task.property("algs").append("MRHe3TMapAlg")
10 task.property("algs").append("RunningInfAlg")
```

代码块 4

1-4行	初始化所需服务组件
6行	加载算法模块
7-10行	初始化所需算法组件

5.1.3 SANS

小角中子散射谱仪的运行和粉末衍射谱仪类似，只是在需要加载不同的服务模块和算法模块，即把 GPPD 中的代码块 2 替换成代码块 5。

```
1 task.property("svcs").append("CtrlSvc")
2 task.property("svcs").append("SANSDataSvc/DataSvc")
3 task.property("svcs").append("He3TRawDataInputSvc/DataInputSvc")
4 task.property("svcs").append("FileInputSvc/DataProvideSvc")
5
6 import Algorithms
7 task.property("algs").append("SANSHe3TRecAlg")
8 task.property("algs").append("SANSHe3TMapAlg")
10 task.property("algs").append("RunningInfAlg")
```

代码块 5

- | | |
|-------|-----------|
| 1-4行 | 初始化所需服务组件 |
| 6行 | 加载算法模块 |
| 7-10行 | 初始化所需算法组件 |

5.2 谱仪探测器模拟

5.2.1 GPPD

粉末衍射谱仪探测器模拟与数据处理程序类似，只需加载不同的服务和算法，用代码块 6 替代代码块 2。

```
1 task.property("svcs").append("CtrlSvc")
2 task.property("svcs").append("GPPDDataSvc/DataSvc")
3 task.property("svcs").append("SimNeutronGunInputSvc/DataInputSvc")
4
5 import Algorithms
6 task.property("algs").append("GPPDSNDFastSimAlg")
```

代码块 6

1-3行 初始化所需服务组件

5行 加载算法模块

6行 初始化所需算法组件

5.2.2 REFL

反射谱仪探测器模拟与通用衍射谱仪模拟程序类似，只需加载不同的服务和算法，用代码块 7 替代代码块 6。

```
1 task.property("svcs").append("CtrlSvc")
2 task.property("svcs").append("REFLDataSvc/DataSvc")
3 task.property("svcs").append("SimNeutronGunInputSvc/DataInputSvc")
4
5 import Algorithms
6 task.property("algs").append("REFLMWPCFastSimAlg")
```

代码块 7

1-3行	初始化所需服务组件
5行	加载算法模块
6行	初始化所需算法组件

5.2.3 SANS

小角中子散射谱仪探测器模拟与通用衍射谱仪模拟程序类似，只需加载不同的服务和算法，用代码块 8 替代代码块 6。

```
1 task.property("svcs").append("CtrlSvc")
2 task.property("svcs").append("SANSDataSvc/DataSvc")
3 task.property("svcs").append("SimNeutronGunInputSvc/DataInputSvc")
4
5 import Algorithms
6 task.property("algs").append("SANSHe3TFastSimAlg")
```

代码块 8

1-3行	初始化所需服务组件
5行	加载算法模块
6行	初始化所需算法组件

5.1 分布式环境 RPC 交互

在线运行在分布式环境中的 DroNE 拥有自己的顶级 Task，与代码块 1 不同，需采用代码块 9 显示的任务。

```
1 import DroNECore
2 import CtrlSvc
3 import DataSvc
4
5 task = DroNECore.DroNEOnline("task")
6 task.asTop()
7 task.setLogLevel(3)
```

代码块 9

1-3行 加载DroNE模块

5行 初始化应用程序任务

6-7行 把本任务设置为顶层任务，并设置日志输出级别

此外只需添加 RPC 代码即可，如代码块 10 所表示的那样。

```
1 m_neonRedis = NEON.Neon.NeonRedis(host='127.0.0.1', port=9999, db = 0, isWritable = True)
2 servpob = NEON.Neon.NeonService.POBox(m_neonRedis, '/GPPD/process/detector', '192.168.0.1:drone01')
3 servrpc = NEON.Neon.NeonService.NeonRPC(sendPOBox = servpob, recvPOBox = servpob, isServer=True)
4 nr = NRT.NeonRPCTask("NeonRPC", servrpc, remotedata = m_taskheartbeat)
5
6 m_taskMatrix = NEON.Data.LocalASCIIData("test.txt")
7 m_configure = {"pidstart":3310001,"pidsize":5328}
8 nu = NUT.NeonUpdateDataTask("UpdateRemoteData", remotedata = m_taskMatrix, configure=m_configure)
9
10 ct = DroNECore.CtrlTask("ctrl")
11 hi = HBI.HeartBeatCronIncident("HeartBeat", cron = 2, repeatable = True, remotedata = m_taskheartbeat)
12 ni = NRI.NeonRPCCronIncident("NeonRPC", cron = 2, repeatable = True)
13 pi = PMI.PushMatrixCronIncident("PushMatrix", cron = 15, \
14     repeatable = True, remotedata = m_taskMatrix,\
15     configure = m_configure)
16 ct.add(hi)
17 ct.add(ni)
18 ct.add(pi)
```

代码块 10

- | | |
|--------|------------------------|
| 1-4行 | 初始化NEON网络中间件NeonRPC组件 |
| 6-9行 | 初始化NEON网络中间件NeonData组件 |
| 10-15行 | 初始化相关事件通知 |
| 16-18行 | 事件通知向控制模块注册 |