



# DroNE 中子实验数据处理框架 设计报告

---

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 2013-12-03

编号

文件名 项目需求调研、系统初步设计和开发计划

作者 田浩来

田浩来-2018年6月25日

## 版本历史

修改者	日期	联系方式	备注
田浩来	2014-6-10	tianhl@ihep.ac.cn	起草
田浩来	2017-3-14	tianhl@ihep.ac.cn	定稿

# 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)的项目需求分析, 系统初步架构设计和开发计划。DroNE 软件框架架构设计基于中国散裂中子源实验数据处理对基础软件提出的需求, 包括架构逻辑设计和物理设计。本文档最后部分为建议的开发阶段划分, 以及相应的时间计划。

此设计报告作为中国散裂中子源软件系统交付文档的一部分, 在软件系统设计阶段, 用于架构设计评审, 在软件系统实现阶段, 用于软件开发指导, 在软件运维阶段, 用于系统维护指南。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:** 需求分析, 架构设计, 开发计划, 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 项目需求分析..... 9

    3.1 计算任务 ..... 9

    3.2 参与角色 ..... 10

4 框架初步设计 ..... 13

    4.1 设计准则 ..... 13

    4.2 初步设计 ..... 16

    4.3 组件交互 ..... 21

    4.4 技术选型 ..... 23

5 开发计划 ..... 25

    5.1 开发阶段 ..... 25

    5.2 开发时间安排 ..... 28

## 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)的项目需求分析, 系统初步架构设计和开发计划。DroNE 软件框架架构设计基于中国散裂中子源实验数据处理对基础软件提出的需求, 包括架构逻辑设计和物理设计。

此设计报告作为中国散裂中子源软件系统交付文档的一部分, 在软件系统设计阶段, 用于架构设计评审, 在软件系统实现阶段, 用于软件开发指导, 在软件运维阶段, 用于系统维护指南。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三台中子谱仪(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。未来该框架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软件框架的设计思路和总体架构，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)



## 3 项目需求分析

这一章将会介绍 CSNS 数据处理系统所承担的计算任务，该系统涉及的参与角色，以及系统开发计划。

### 3.1 计算任务

CSNS 中子谱仪对数据处理系统提出了一系列详尽的要求，这些要求使得我们的设计要能满足现在通用数据处理软件所面对的所有任务，涵盖在线数据实时处理、离线数据批处理和单机交互式处理等各种计算环境。总所周知，谱仪软件的任务，是把谱仪电子学获取的电压/电流信号，通过一系列数据处理作业(包括数据收集、原始数据处理、探测器数据重建和映射、直方图数据生成和处理)，转换成为物理数据。这些物理数据要求尽可能的消除仪器特征，仅反映真实的物理过程。开发这些数据处理组件，需要各系统之间的紧密配合，比如数据收集任务不可避免的需要数据获取和电子学专家的参与，探测器重建算法离不开探测器专家的意见，最终数据质量也受到用户和谱仪科学家的影响。

此外，作为谱仪专用软件，CSNS 数据处理系统还要面对一些特殊的问题。第一，数据处理子系统只是谱仪系统的一部分，所以数据处理子系统必须很好的和谱仪其他组成子系统集成交互；第二，

作为大科学装置的 CSNS 支持多达 20 台谱仪的运行，所以数据处理系统必须有很好的通用性，在不显著增加研发和运维压力的情况下，能够支持不同谱仪的运行；第三，CSNS 预计运行周期在 30 年以上，数据处理系统必须要有很好的灵活性，来面对长达几十年的技术升级。

## 3.2 参与角色

数据处理子系统在整个谱仪系统中处于一个中心的位置，其他子系统，如谱仪控制、数据获取、电子学、探测器等都深度参与于数据处理的具体任务，同时数据处理子系统也是用户接触谱仪的直接接口，数据处理结果的质量直接反映谱仪的运行质量。而软件框架作为数据处理子系统的基础，数据处理子系统的合作成员，或多或少的都要和基础框架有所接触。比如实验前执行模拟作业，实验中执行数据质量监视任务，实验后执行数据预处理任务。这里我们简单的使用框架的用户和开发者做一下分类。

1. 用户。这类群体首要关心的是获取结果以及结果的质量，他们有可能会调节数据处理系统的一些参数，以便获得更好的数据，但是他们的改动不应该影响谱仪的其他用户。而通常情况下，他们仅仅依靠谱仪科学家来获得数据，对数据处理的细节完全不感兴趣。
2. 谱仪科学家。这类群体关心的是如何调试谱仪，修改或开发数据处理算法，以便获得更好的数据。他们虽然不是软件开发者，但是他们可能相当了解数据处理算法的细节。他们完完全全的了解整台谱仪的情况，通常情况下，其他人都是通过谱仪科学家来操作谱仪

的。他们对数据处理软件做出的改动，会影响这台谱仪的所有用户，但不应该影响其他谱仪。

3. 算法工程师。这类群体对特定问题有深入的研究，应该说谱仪科学家的一些工作和算法工程师有所重合。此外还有探测器科学家，在开发探测器重建算法和刻度算法时，他们有绝对的权威。虽然有时候，他们仅仅关心整个宏大软件系统中几行代码，或者几个配置参数，但是他们对数据处理结果的质量，有着非常大的影响。然后他们对程序的改动，应该仅限于他们的专业范围内，某台谱仪或者某个算法，不应该影响程序其他组件的操作。

4. 应用软件工程师。这类群体关注的是软件的通用功能开发，在 DroNE 的开发过程中，他们往往涉及数据 IO 和控制流程的实现，他们的工作是全局性的。

5. 配置管理员。他们的工作往往不涉及代码的改动，但是他们的工作决定了应用软件运行的正确性。他们的责任是为特定的计算任务配置合适版本的算法，为特定的算法配置合适的运行参数。随着谱仪的升级，同样功能的算法可能有不同的实现，而由于运行条件不同，类似刻度参数也需要选择不同的版本，这些配置可能需要从 XML 或者数据库中读取，因此他们可能会保存一张 Excel 表，里面有制作菜肴的各种调料配方。

6. 框架开发工程师。这些人负责框架的设计、实现和支持。

## 4 框架初步设计

本章将会介绍 CSNS 数据处理系统的设计准则，物理设计，以及逻辑设计，最后介绍该系统依赖的底层技术选型方案。

数据处理框架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。

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。

### 4.1 设计准则

系统设计的基本原则，就是“隔离变化”，经过调研，我们总结了如下设计准则，作为我们设计的出发点。

#### 4.1.1 数据与算法的隔离

在科学软件的应用中，数据结构往往是稳定的(比如列表、矩阵等)，但是针对相同数据的处理算法通常是多变的，我们常常用不同的算法处理相同的数据，去比较算法的优劣，也时常改变算法的参数去寻找更好的处理结果。基本数据结构的定义由框架开发工程师来确定，算法由谱仪科学家和算法工程师开发。

### 4.1.2 瞬态数据与持久数据的隔离

如上所述，虽然内存中的数据结构(瞬态数据)表达是稳定的，但是数据源却往往不同。比如相同的探测器重建算法，离线环境中原始数据可能来自于文件，而在线环境中可能来自于 DAQ 的网络数据流。同时相同的瞬态数据，我们最终希望保存的方式可能也不尽相同，比如同样的矩阵数据，我们可能会保存成为 ASCII 文件、JSON 文件、XML 文件、或者科学计算通用的 HDF 文件。持久数据组件由应用软件工程师开发，数据源的配置有谱仪科学家或用户在程序运行前确定。

### 4.1.3 以瞬态数据为中心的架构

参考上面的讨论，我们可以看出，在科学软件中，算法是容易变动的，持久数据也是容易变动的，但是所幸的是内存中的瞬态数据结构相对是稳定的，因此我们应当把瞬态数据及其管理作为整个框架的中心。瞬态数据结构的定义，应该由谱仪科学家和框架开发工程师细致研究后确定，一经确定谨慎变动。

### 4.1.4 服务和算法的隔离

谈完了数据，现在我们把注意力转向算法。如前所述，算法的开发者往往是算法工程师和谱仪科学家，作为某领域的专家，他们的注意力往往集中在关键性的几行代码和几个参数上，他们时常查找文献寻找摆弄数据的技巧，为让数据质量仅仅千分之几的提高绞尽脑汁，然而他们对整个软件的其他部分的精妙设计却视而不见。

为了让他们沉浸在自己的世界中，我们应该把无关数据处理但对软件运行至关重要的代码另存他处，以免打搅他们的专注。这部分代码，我们一般称为“服务”，主要涉及数据IO和程序控制，由框架工程师和应用程序工程师开发。

#### 4.1.5 C++ 和 Python 的使用

谈及C++，它的缺点和优点同等显而易见。作为一种面向对象的强数据类型的静态编译型语言，C++在运行效率上值得称道，然而却带来极大的开发复杂度。Python站在它的反面，作为一种简洁优雅而缓慢动态语言，在科研领域获得极大的成功。在我们的设计中，我们希望这两种语言能扬长避短，相互补充。在涉及数据IO和算法的时候，发挥C++高效的特性，在涉及交互方面，发挥Python灵活的特性。此外，作为一种脚本语言，我们希望变动频繁的代码以Python实现。Python代码主要由谱仪科学家和应用程序工程师开发，C++代码由算法工程师、应用程序工程师和框架工程师开发。

#### 4.1.6 接口与实现的隔离

所有的框架组件都将实现框架预定义的接口，每个接口包括一系列特定方法，使得框架能反向控制组件的运行，也让组件之间的交互透明。接口的定义由框架工程师确定，实现由应用程序工程师开发。

### 4.1.7 通用组件的复用

我们所做的一切努力，就是希望实现可复用组件，避免重复开发。相同的组件，不但可以在离线/在线环境中复用，也可以在不同谱仪间复用。所以我们强烈要求开发者在开发过程中，遵循框架的接口定义。

### 4.1.8 中间件集成技术

DroNE框架不应该限定中间集成技术的使用，不过出于运行效率和开发效率方面的考虑，建议数据传输集成技术基于C++实现，外部控制指令基于Python实现，另外为了不影响DroNE数据处理的效率，所有集成技术，尽量采用异步通讯模式完成。

## 4.2 初步设计

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 workflow，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。

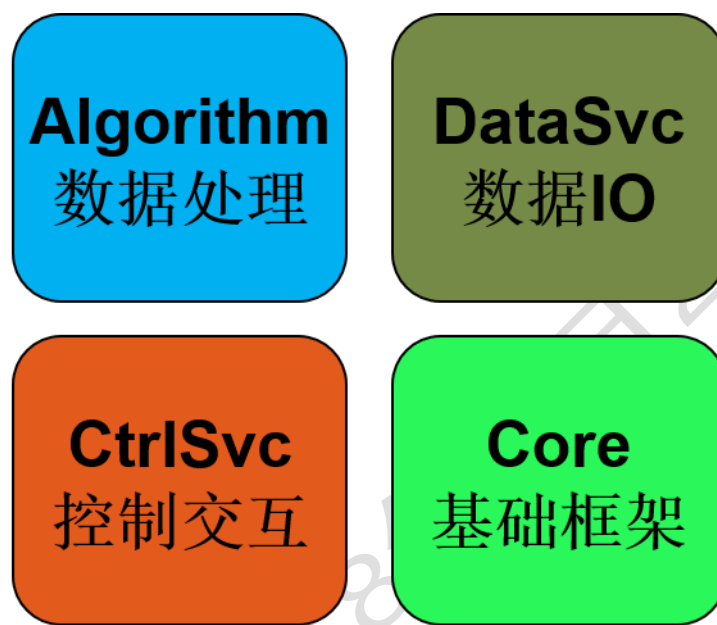


图1 DroNE总体架构

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。

### 4.2.1 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。

### 4.2.2 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。

### 4.2.3 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。

#### 4.2.4 算法模块

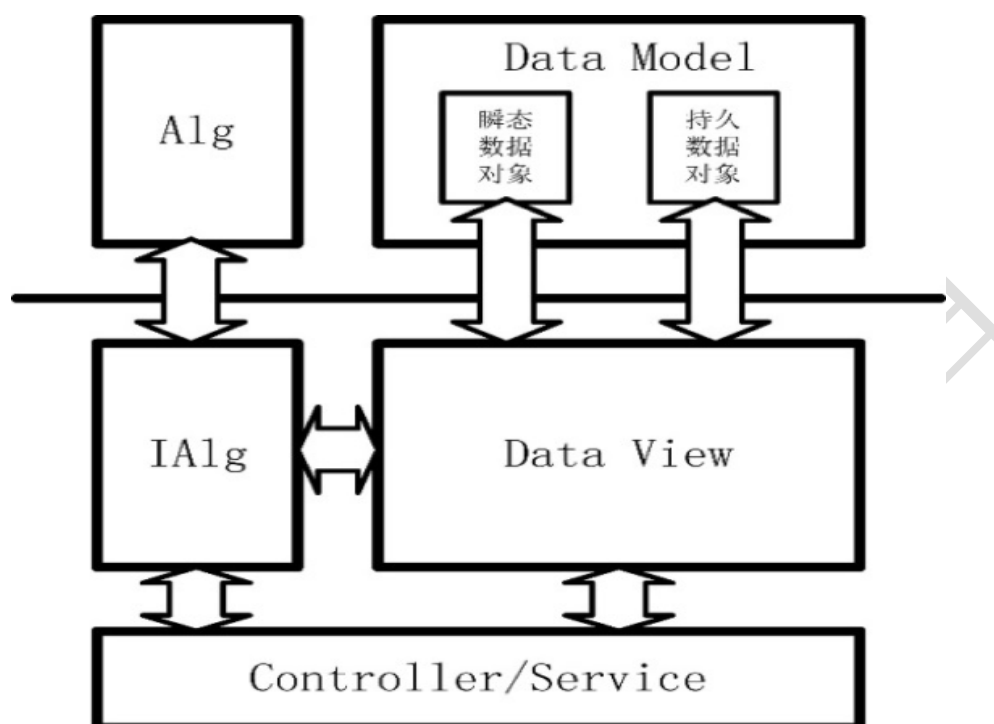


图 2 算法和数据服务的交互

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。

## 4.3 组件交互

### 4.3.1 离线数据处理中的事例循环

- 1 所有模块在调用前必须先加载(import), 必须加载的模块包括DroNECore, DataSvc和CtrlSvc。
- 2 初始化DroNECore, 定义计算任务task。
- 3 在task中初始化控制服务CtrlSvc。
- 4 在task中初始化所需数据服务, 包括DataSvc, DataInputSvc和DataProvideSvc, 其中DataSvc管理瞬态数据对象, DataInputSvc负责从持久对象加载瞬态数据, DataProvideSvc负责持久对象读取。
- 5 在task中加载算法对象。
- 6 运行task, 通过控制服务发出BeginEvt信号, 从持久对象中加载第一个脉冲事例(Pulse)
- 7 数据加载完成后, 依次运行算法对象序列
- 8 算法从DataSvc中读取瞬态数据对象, 经过处理后, 写回DataSvc
- 9 结束算法序列调用, 通过控制服务发出EndEvt信号, 清空瞬态数据对象。
- 10 重复执行操作6-9, 直至结束。

### 4.3.2 在线数据处理中的控制交互

在线数据处理流程和离线数据处理类似，唯一的区别在于执行完算法序列后，task需要执行一个控制指令序列。

- 1 初始化控制服务后，向控制服务注册定时触发Incident
- 2 执行完算法序列后，向控制服务查询是否有需要触发的Incident，如果没有完成本次事例循环。
- 3 如果控制服务中存在需要触发的Incident，触发该Incident，框架会自动执行对应的Handle。
- 4 执行完控制服务中保存的需触发的Incident后，完成本次事例循环。

### 4.3.3 瞬态数据的读取

- 1 收到BeginEvt后，DataSvc调用DataInputSvc，DataInputSvc从自身的缓存中解析出下一个脉冲事例，写入DataSvc。
- 2 如果DataInputSvc缓存全部解析完成，DataInputSvc调用DataProvideSvc从持久对象中读入下一段数据，并保存在自己的缓存中。
- 3 当DataProvideSvc确认所有持久对象都已经读完，发出StopRun指令，结束整个程序运行。

## 4.4 技术选型

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。

Gaudi是一种在高能物理领域广泛应用的数据处理框架，已经有接近20年的开发历史，不过Gaudi复杂的持久对象加载机制带来了潜在的开发难度；此外Gaud和CERN其他软件库和数据存储技术(如ROOT)深度集成，带来了额外的复杂性；最重要的是，虽然Gaudi提供Python接口，但是作为纯C++开发的数据处理框架，Python并不能精细的控制Gaudi的所有运行细节。

Mantid在欧美中子散射实验中有广泛的应用，Mantid不但是一种成熟的软件框架，也提供一整套功能包，是可以独立运行的程序。不过Mantid实现的瞬态数据对象模型已经固定，不支持用户进行二次开发，所以我们决定把Mantid作为CSNS中子散射数据规约软件的基础框架，但是不作为数据处理的基础框架。

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。因此我们选择SNiPER作为DroNE的基础软件框架。

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。

DroNE 现仅准备支持 Linux 平台，编译系统为 GCC

田浩来-2018年6月25日

## 5 开发计划

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。本章把整个开发过程划分为五个阶段，详细论述了各阶段的主要工作和参与角色，最后提出了初步的开发计划。

### 5.1 开发阶段

#### 5.1.1 需求调研

完成该计划的第一步，是全面的深入理解 CSNS 对软件系统的需求，框架的设计就是被这种需求所引导的。在框架的设计中，灵活性和复杂度，是一块跷跷板的两头，提高灵活性必然带来复杂度，降低复杂度必然损失灵活性，框架设计的根本任务，就是平衡灵活性和复杂度，使得整个开发任务能在限定时间和限定资源的情况下完成。而为了达到这种平衡，首要的事情就是对系统功能做到取舍。因此我们需要去了解用户、谱仪科学家和谱仪其他子系统对软件框架的需求，并且确认，哪些需求是必须的，哪些是可选的。需求的确认，一方面来自于系统设计者自身的经验，一方面来自于与合作者的正式或非正式的讨论，还有更重要的是，来自于其他实验软件系统的成熟经验。



### 5.1.2 初步设计和实验版本发布

在需求确认完成后，我们就可以着手设计和实现了。需要明确的是，这样一个大型软件的设计，闭门造车是不行的，我们需要随时和潜在的用户和合作者讨论。我们应该尽快提供一个实验版本供用户尝试，满足用户最紧急的需求，并在此基础上进行功能迭代和升级。同时实验版本也是框架候选技术的验证平台，通过真实的开发活动，熟悉和了解候选技术的特征，为后一阶段最终确定技术平台打好基础。

### 5.1.3 详细设计和预览版本发布

在完成最初几个实验版本迭代后，应该尽快明确关键性模块，并确定这些模块之间的接口，这些关键模块将会成为框架的基石。此时的版本可以定义为预览版本。此前的工作主要由框架开发工程师完成。

预览版的发布预示着整个软件框架的稳定，这时就可以把主要精力放在拓展性功能模块的开发上了。一般情况，拓展性模块都是关键性模块的派生类，因此可以直接复用基类的接口，并实现特定的操作。拓展性模块的开发要注意降低模块之间的耦合，避免因为扩展性模块的升级导致整个系统的失效。与此同时，数据处理算法可以同步开发。算法的调试和计算参数的确定是一个长期的过程，往往伴随特定谱仪的整个生命周期。在这一阶段，应用开发工程师和算法工程师开始参与开发工作。

### 5.1.4 测试版本发布和集成测试

当拓展模块和算法模块可以初步满足软件系统需求后，框架进入测试版。测试版的主要任务是依次进行功能测试、压力测试和集成测试，并根据测试结果对框架进行微调。此阶段，框架的开发进入尾声，重点任务转移至在框架的基础上进行应用程序的开发。

### 5.1.5 正式版本发布与运行维护

当框架和应用程序通过测试后，整个软件系统进入正式版阶段。进入正式版后，开发任务集中在运行、维护和升级工作。开发任务的重要承担者，由前期的框架开发工程师、应用开发工程师和算法工程师，转移为谱仪科学家和配置管理员。

## 5.2 开发时间安排

2014.06.1—2015.03.1	需求调研
2015.03.1—2016.03.1	初步设计与实验版本发布
2016.03.1—2017.03.1	详细设计与预览版本发布
2017.03.1—2017.10.1	测试版本发布与集成测试
2017.10.1—	正式版本发布与运行维护