一体化二进制数据包软件
概要设计说明书

目录

图目录

表目录

1. 引言

1.1. 标识

文档标识：SJSM_XNSIM

标题：一体化二进制数据包软件概要设计说明书

版本号：V 1.0

发布号：V 1.0

1.2. 软件概述

一体化二进制数据包软件是飞行模拟机系统的一个子系统，用于运行飞机仿真模型数据包并与其它子系统交互数据。

在当前飞行模拟机制造领域，随着技术的进步和行业需求的增长，对于飞行模拟训练设备的要求越来越高。传统的二进制数据包（动态链接库）在不同实时仿真系统中部署后，面临着仿真结果一致性难以保证的问题，这直接影响了数据包在实时仿真系统上的集成技术，使得技术问题变得复杂，且难以形成统一品质的飞行训练设备。此外，对于中国商飞等飞机制造商而言，过度依赖外部仿真平台供应商可能导致技术迭代和升级受限，影响数据包仿真模型的自主可控性。

为了解决上述问题，并推动航空仿真技术的发展，我们提出了新一代基于一体化超融合系统架构的仿真模型封装技术。通过自主研发的一体化超融合系统架构——“玄鸟”架构，我们旨在构建一个协同整合的一体化二进制数据包软件。该软件的开发对于整个航空产业链的参与者具有重大意义：

1．对于局方：通过使用唯一经过飞机制造商授权的、高度集成和统一的二进制数据包软件，可以提高鉴定和监管效率，确保模拟飞行训练的有效性，从而提升民航飞行安全。

2．对于飞机制造商：可以更好地保护知识产权，保证模拟飞行训练的品质，形成标准化数据包产品，为飞行培训全产业链提供更好的服务，提高经济效益，并促进数据包开发技术的可持续发展。

3．对于模拟机制造商：可以减少对飞机本身模拟的关注，转而专注于其他系统的研制，有效提升模拟机全机的品质，缩短研制周期，降低成本。

4．对于模拟机运营商：采用一体化数据包软件可以实现不同厂家飞行模拟训练设备的飞机功能与性能仿真一致性，提高引进及运营的效率。

综上所述，自主研发的“玄鸟”架构和一体化二进制数据包软件的开发，不仅能够解决现有的技术难题，还能够为国内飞行模拟机制造领域带来创新和进步，具有重要的战略和实际应用价值。

1.3. 文档概述

本软件概要设计说明书的编写旨在阐述一体化二进制数据包软件的概要设计方案，包括软件架构、功能模块、处理流程、用户界面设计以及与其他系统的接口等关键信息。编制本说明书的主要目的如下：

1．明确设计目标与范围：定义功能需求和性能目标，确保设计满足项目预期和用户需求。

2．指导开发与实现：为软件开发团队提供详细的设计指南，确保开发工作按照既定的设计规范和标准进行。

3．风险评估与管理：识别和记录在设计和实现过程中可能遇到的风险和问题，为风险管理提供依据。

4．质量保证：通过详细的设计文档，为软件测试和质量保证活动提供基础，确保软件质量符合标准。

5．维护与升级：为未来的软件维护和升级工作提供必要的技术文档支持，降低维护成本，提高效率。

6．符合标准与规范：确保软件设计遵循国家和行业的相关标准和规范。

7．项目文档归档：作为项目重要文档之一，本说明书将被归档存储，为项目的历史记录和知识积累提供资料。

通过本说明书，项目团队能够清晰地理解软件的设计理念和实现细节，为后续的详细设计、软件开发、软件测试和软件部署工作奠定坚实的基础。

1.4. 基线

本设计说明书依据的设计基线为 GB/T 8567-2006《计算机软件文档编制规范》。

1.5. 术语

本文中用到的术语及外文缩写的含义如表 1 所示。

表 1 术语

术语/缩写	全称	含义
API	Application Programming Interface	应用程序编程接口
ARINC	Aeronautical Radio Inc.	航空无线电通信公司
CAN	Controller Area Network	控制器局域网总线
DCPS	Data-Centric Publisher-Subscriber	以数据为中心的发布者-订阅者模型
DDS	Data Distribution Setvice	数据分发服务
ICD	Interface Control Document	接口控制文档
IDE	Integrated Development Environment	集成开发环境
IDL	Interface Definition Language	接口定义语言
IP	Internet Protocol	网际互连协议
IRQ	Interupt ReQuest	中断请求
MSVC	Microsoft Visual C++	微软 C++集成开发环境
PIMPL	Pointer to Implementation	指向实现的指针
QoS	Quality of Service	服务质量
QTG	Qualification Test Guide	鉴定测试指南
RTPS	Real-Time Publish-Subscribe protocol	实时发布-订阅协议
SHM	Shared Memory	共享内存
TCP	TransmissionControl Protocol	传输控制协议
TDM	Training Devices Manufacture	飞行模拟机制造商
UDP	User Datagram Protocol	用户数据报协议

2. 引用文件

[1] GB/T 8567-2006,《计算机软件文档编制规范》, 2006-06-01；

3. 设计决策

本章对一体化二进制数据包软件的设计决策进行阐述，将从接口、行为、数据库/数据文件、安全性、灵活性等方面对一体化二进制数据包软件进行设计决策。

3.1. 接口设计决策

一体化二进制数据包软件需要实现一系列的接口，包括：二进制数据包模型间的通信接口、与外部系统的通信接口以及与用户交互的接口。

3.1.1. 模型间通信接口

二进制数据包模型间的通信接口由 ICD（接口控制文档）定义，具有数据量庞大以及需要实时交互的特点。因此，一体化二进制数据包软件拟使用开源的数据分发服务 Fast-DDS 库，可以实现基于共享内存的发布订阅模式进行数据交互，满足大量数据的实时交互需求。Fast-DDS 是 DDS（数据分发服务）规范的一种 C++实现，提供了 API（应用程序编程接口）和通信协议，这些协议部署了 DCPS（以数据为中心的发布者-订阅者模型），可以配置为提供实时功能，保证在指定的时间限制内做出响应，因此能够在实时系统之间建立高效可靠的信息分发。

一体化二进制数据包软件使用 IDL（接口定义语言）将模型 ICD 定义为交互主题，模型可以通过发布/订阅主题的形式进行数据交互。

3.1.2. 外部系统通信接口

一体化二进制数据包软件与外部系统（主要是 TDM 计算机）的通信接口分为以下三种类型：

1．虚拟航空总线通信接口：需要按照飞机各系统间通信总线的协议建立接口标准，并通过 UDP/TCP 实现数据通信。需要实现的虚拟航空总线协议包括用于照明系统的 CAN（ARINC 825）总线、用于航电系统的 ARINC 429/ARINC 664 总线和用于气象雷达的 ARINC 708 总线等。

2．离散量通信接口：一体化二进制数据包软件与 TDM 计算机间有部分离散量需要交互，需要实现通过 UDP/TCP 的离散量数据通信。

3．控制指令接口：一体化二进制数据包软件需要通过 UDP/TCP 接收、响应与反馈 TDM 计算机发出的一系列控制指令。这部分控制指令包括仿真控制指令与模型控制指令等。

3.1.3. 用户接口

一体化二进制数据包软件与用户交互的接口包括界面调试工具等前端软件以及一系列配置文件。界面调试工具是用户与一体化二进制数据包软件交互的重要接口，允许开发者和

用户在运行时监控和调试软件发送和接收的数据。其它前端软件有仿真调度终端和仿真配置终端等，这些工具拥有直观的用户交互界面。通过这些前端软件，用户可以方便地配置和管理各类配置文件与控制软件的运行。这些配置文件的设计决策见 3.3 小节。

3.2. 行为设计决策

一体化二进制数据包软件需要集成并实时调度二进制数据包模型，完成模型间数据交互，同时能够接受外部系统指令控制，因此要对软件的模型集成、模型调度、指令解析等行为进行设计决策。

3.2.1. 模型集成

一体化二进制数据包软件需要实现标准化的模型封装框架，提供二进制数据模型的调用接口。负责模型封装集成的开发人员利用标准化的封装框架提供的接口方法，可以便捷地进行二进制数据包仿真模型的封装，以集成到“玄鸟”架构中。

3.2.2. 模型调度

由于普通的操作系统在 SpinLock，IRQ 上下文方面无法抢占，导致高优先级任务唤醒后并不一定立即得以执行，并且也不支持优先级反转，因此需要操作系统具有硬实时性。Linux 系统需要安装 RT_Preempt 补丁，Windows 系统需要安装 RTX 扩展，使得操作系统能够处理优先级反转，能够实时执行高优先级的任务。同时，通过配置操作系统的 CPU 隔离配置项，可以使特定的几个 CPU 核心只能由一体化二进制数据包软件使用，以免其它进程竞争 CPU 资源。一体化二进制数据包软件在具有硬实时性的操作系统上使用 pthread 库和纳秒睡眠技术来创建线程实现二进制数据包模型的实时调度，使每个线程可单独设置其优先级、内核调度策略、CPU 亲和性等参数。

3.2.3. 指令解析

一体化二进制数据包软件可接受各种不同的控制指令，这些指令通过 UDP/TCP 进行交互。控制指令在控制指令配置文件中定义，软件通过指令解析服务根据定义将指令发往特定的模块执行，只有有效的控制指令才能被软件执行。如果控制指令无效，指令解析服务会返回错误信息。指令支持使用进程名/线程名/服务名/模型名通配符发送，任何满足通配符条件的进程/线程/服务/模型都将执行给定的指令。

3.3. 数据库/数据文件设计决策

一体化二进制数据包软件能够通过运行环境配置文件、模型配置文件灵活配置软件的运行环境和二进制数据包模型参数。运行环境配置文件用于定义软件运行所需的环境参数，包括操作系统版本、工作路径等。这些配置文件确保软件在不同环境中能够正确运行，并提供必要的环境变量设置。通过读取这些配置文件，软件可以动态调整其行为以适应不同的运行环境。模型配置文件用于描述二进制数据包模型的接口和参数。通过模型配置文件，开发者可以灵活地修改二进制数据包模型的接口和调度参数。

一体化二进制数据包软件还需要利用数据库来支持快照的拍摄和调用。考虑到模型间采用 Fast-DDS 进行数据交互，因此可以 Fast-DDS 自带的数据持久化工具 Record and Replay 将快照数据存储为 MCAP 格式的数据库文件。输出的 MCAP 文件可以被任何兼容的工具读取，它包含了读取和重放快照数据所需的所有信息。

3.4. 安全性设计决策

一体化二进制数据包软件采用的 Fast-DDS 通信可以配置为安全通信来保护通信和数据的安全，防止数据在传输过程中被截获或篡改。它提供了三种验证方式保证数据的安全性：远程参与者的身份验证、实体的访问控制和数据加密。

一体化二进制数据包软件通过完善的日志记录和出错告警信息来保证软件的在运行过程中出现错误时能够及时发现和尽快纠正，保证仿真的稳定和安全运行。

3.5. 灵活性设计决策

一体化二进制数据包软件通过实现操作系统抽象层，简化了对底层操作系统的依赖，使得应用程序可以在不同的操作系统上运行而无需进行大量的修改。软件使用开源的跨平台自动化构建系统 CMake 进行项目工程的管理，可以跨平台进行项目工程构建。

4. 结构设计

4.1. 总体设计

4.1.1. 概述

4.1.1.1. 功能描述

4.1.1.1.1. 功能需求

一体化二进制数据包软件应满足以下功能需求：

1．模型封装与集成能力：能够集成离散的二进制数据包仿真模型（动态链接库），并正确模拟飞机本体和各飞机系统的工作；

2．模型实时调度能力：能够以不同的频率实时调度各模型周期性运行；

3．模型间数据交互能力：能够满足模型间数据的交互；

4．外部系统数据交互能力：能够通过 UDP、TCP 或虚拟航空总线与外部系统进行数据交互；

5．仿真监控能力：能够监控仿真运行状态和模型运行数据；

6．仿真可配置性：能够使用配置文件配置仿真运行所需的各类参数，以满足不同的运行需求；

7．接受外部系统指令控制能力：能够接收外部系统的控制指令并响应控制；

8．快照拍摄和调用能力：能够拍摄和调用快照，进行快照数据持久化地存储与读取。

9．可视化交互界面：具有统一的界面调试工具及丰富的前端软件，包括仿真运行终端、仿真配置终端、仿真监控终端等。

4.1.1.1.2. 性能需求

一体化二进制数据包软件还应满足以下性能需求：

1．调度实时性：模型周期性调度的频率误差不超过 1 %；

2．响应实时性：外部输入数据/指令的响应时间不大于 50 ms；

3．监控实时性：监控软件的数据更新频率不大于 1 Hz；

4．软件可靠性：加载全部二进制数据包模型后稳定运行时间不少于 100 小时；

5．软件兼容性：应具备跨操作系统能力；

4.1.1.2. 运行环境

4.1.1.2.1. 硬件环境

1．处理器（CPU）：最低要求 Intel Core i7 或等效的其它处理器，主频 2.4 GHz 以上，核心数不少于 8 个；

2．内存（RAM）：最低要求 4 GB，推荐 8 GB；

3．存储空间（硬盘）：需要至少 200 GB 的可用硬盘空间；

4．网卡：支持 10 Gbps 传输速率；

4.1.1.2.2. 软件环境

1．操作系统：需要具备硬实时性的操作系统。Linux 操作系统最低版本为 Debian 11 、Ubuntu 20.04、CentOS 8 或其它内核版本 5.10.0 以上的 Linux 操作系统，并安装 RT_Preempt 补丁，将内核版本升级为对应版本的 RT 内核；Windows 操作系统最低版本为 Windows 10，并安装 Windows RTX2011 或 Windows RTX64 扩展；

2．编译器：Linux 系统下为 GCC/G++ v10.2.1，Windows 系统下为 MSVC v142（Visual Studio 2019）；

3．IDE：Qt Creator v14.0.1 和 Visual Studio Code v1.93.0

4．项目管理：CMake v3.20

5．依赖项：Fast DDS v3.1.0、Fast CDR v2.2.5、Foonathan Memory Vendor v1.3.1、Asio v1.30.0、TinyXml2 v6.0.0、OpenSSL v3.1.1、Qt v6.7.2

4.1.2. 设计思想

4.1.2.1. 软件构思

一体化二进制数据包软件采用自研的“玄鸟”架构封装集成二进制数据包模型，以实现 4.1.1.1 小节所述的所有功能需求。因此对“玄鸟”架构的设计构思如下：

1．“玄鸟”架构包含一个仿真内核，以实现软件的核心功能，包括：二进制数据包模型的多线程多频率实时调度、基于 Fast DDS 的主题发布/订阅、软件运行环境配置、加载模型配置、加载服务配置、事件处理、服务系统框架、模型封装框架等。

2．“玄鸟”架构包含一个集成了所有二进制数据包模型的模型系统，其中的模型根据仿真内核提供的模型封装框架进行封装集成。模型系统使仿真内核不依赖于具体的模型，这对模型加载提供了灵活性；

3．“玄鸟”架构包含一个用于外部数据交互及其它扩展功能的服务系统，其中的服务根据仿真内核提供的服务系统框架进行开发，可以实现离散量通信服务、虚拟航空总线通信服务、控制指令解析服务、快照服务等。服务系统使仿真内核不依赖于具体的服务，这对仿真内核功能的可扩展性提供了支持；

4．“玄鸟”架构包含一个用于加载与控制仿真内核运行的引擎；

5．“玄鸟”架构实现了操作系统抽象层，简化了对底层操作系统的依赖，使得应用程序可以在不同的操作系统上运行而无需进行大量的修改。

6．“玄鸟”架构包含一系列界面调试工具及丰富的前端软件。其中，仿真运行工具通过直观界面控制平台运行；仿真配置工具提供清晰的参数设置界面方便用户配置运行参数、服务参数、模型参数、接口参数等；仿真监控终端可以在仿真运行过程中监控与采集运行数据，以可视化图表展示运行状态及自定义监控数据，供用户进行分析和调试。

7．“玄鸟”架构提供了模型封装及服务开发所需的 IDE 及依赖环境。

4.1.2.2. 关键技术与算法

“玄鸟”架构采用的关键技术与算法有模型调度算法、PIMPL 设计模式、宏编程和基于发布/订阅的共享内存数据交互。

4.1.2.2.1. 模型调度算法

“玄鸟”架构能够保证模型能够实时调度的一个必要条件是 Linux 实时内核。Linux 实时内核是抢占式内核，通过设置任务的优先级，高优先级任务能够抢占内核来保证及时执行。此外，Linux 系统还需要预先设置 CPU 隔离，使特定的几个 CPU 核只能由 “玄鸟”架构使用，以免其它进程与本架构竞争内核。

“玄鸟”架构通过多线程模式调度模型，线程是由 pthread 库实现的，使每个线程可单独设置其运行优先级、线程的内核调度策略、CPU 亲和性、栈空间大小等参数。同时使用了纳秒睡眠技术保证线程在 Linux 实时内核中能够按照设置的频率正确运行。

“玄鸟”架构中每个线程都可以调度多个不同频率的模型。线程包含一个调度任务表，线程的每个运行周期按顺序依次执行调度表，调度表的设计保证了每个被调度的模型按照预定的频率运行。

4.1.2.2.2. PIMPL 设计模式

PIMPL（Pointer to Implementation，指向实现的指针）设计模式是一种常用的，用来对“类的接口与实现”进行解耦的方法，是一种隐藏实现方法的设计模式，如图 1。它将类的私有

成员封装在一个私有结构体中，仅在原类中暴露用户必须的接口和该私有结构体指针。使用该设计模式的软件，能够较好地隐藏软件内具体实现。

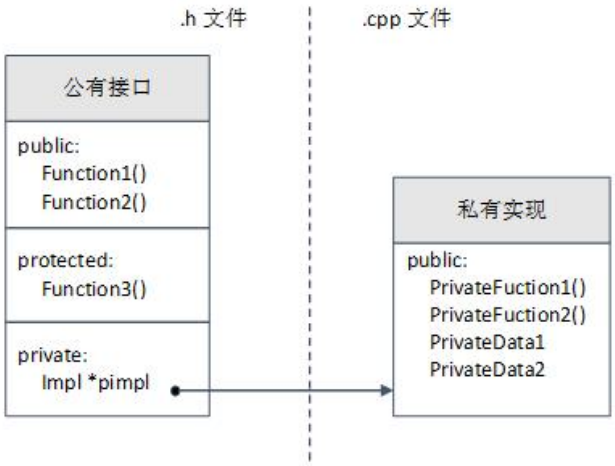

图 1　PIMPL 设计模式

在类中使用 PIMPL，具有如下优点：

1．信息隐藏：私有成员完全可以隐藏在共有接口之外，尤其对于闭源 API 的设计尤其的适合。同时，很多代码会应用平台依赖相关的宏控制，这些琐碎的东西也完全可以隐藏在实现类当中，给用户一个简洁明了的使用接口。

2．加速编译：这通常是用 PIMPL 设计模式的最重要的收益，称之为编译防火墙(Compilation Firewall)，主要是阻断了类的接口和类的实现两者的编译依赖性。这样，类用户不需要额外引用不必要的头文件，同时实现类的成员可以随意变更，而公有类的使用者不需要重新编译。

3．更好的二进制兼容性：通常对一个类的修改，会影响到类的大小、对象的表示和布局等信息，那么任何该类的用户都需要重新编译才行。而且即使更新的是外部不可访问的 private 部分，虽然从访问性来说此时只有类成员和友元能否访问类的私有部分，但是私有部分的修改也会影响到类使用者的行为，这也迫使类的使用者需要重新编译。而对于使用 PIMPL 设计模式，如果实现变更被限制在实现类中，那公有类只持有一个实现类的指针，所以实现做出重大变更的情况下，PIMPL 设计模式也能够保证良好的二进制兼容性。

4．惰性分配：实现类可以做到按需分配或者实际使用时候再分配，从而节省资源提高响应。

4.1.2.2.3. 宏编程

宏编程技术利用 C++ 宏来简化编程所需的代码输入量，提高编程效率。

“玄鸟”架构开发和模型集成过程中有大量的重复性高的代码，例如模型周期性函数注册、模型数据与共享内存映射等，它们的共同特点是代码段绝大部分都相同，仅有少数参数或类型不同。

使用带参数的宏编程或 C++ 模板编程都能解决这个问题，但使用模板函数在生成动态库时不会也不可能推导所有的模板类型以生成符号表，因此只能采用带参数的宏编程来生产“模板”代码。

带参数的宏编程所定义的代码只有在使用该宏的地方才展开成代码并被编译。

4.1.2.2.4. 基于发布/订阅的共享内存数据交互

“玄鸟”架构使用 Fast DDS 进行基于发布/订阅的共享内存数据交互。Fast DDS 是一种基于 DDS 标准的发布/订阅消息中间件，它支持包括共享内存在内的多种通信机制。Fast DDS 通过共享内存实现零拷贝通信，将数据直接写入共享内存区域，避免了多次数据复制，提高了传输效率。在同一主机上的其他订阅进程可以直接访问此内存区域，而无需进行数据拷贝。使用 Fast DDS 进行基于发布/订阅的共享内存数据交互，可以显著提高跨进程通信的效率，尤其是在需要处理大量数据和高频率消息传递的场景中。

Fast DDS 基于发布/订阅模型，其中发布者（Publisher）负责发送数据，订阅者（Subscriber）负责接收数据。数据通过主题（Topic）组织和传输，发布者和订阅者之间是解耦的。在二进制数据包模型集成过程中，根据二进制数据包模型 ICD 中的信息，使用 IDL 定义相应的数据结构，建立模型接口主题。“玄鸟”架构通过标准化的模型封装框架提供主题的发布/订阅接口，使得集成后的二进制数据包模型通过发布/订阅所需的主题进行数据交互。

Fast DDS 设计用于支持实时系统，特别是在低延迟、高吞吐量和确定性要求高的应用中，这保证了二进制数据包模型间能够进行实时数据交互。

4.1.2.3. 关键数据结构

“玄鸟”架构中的关键数据结构为线程调度任务表数据结构。线程调度任务表是一个三维表，数据结构如图 2 所示。

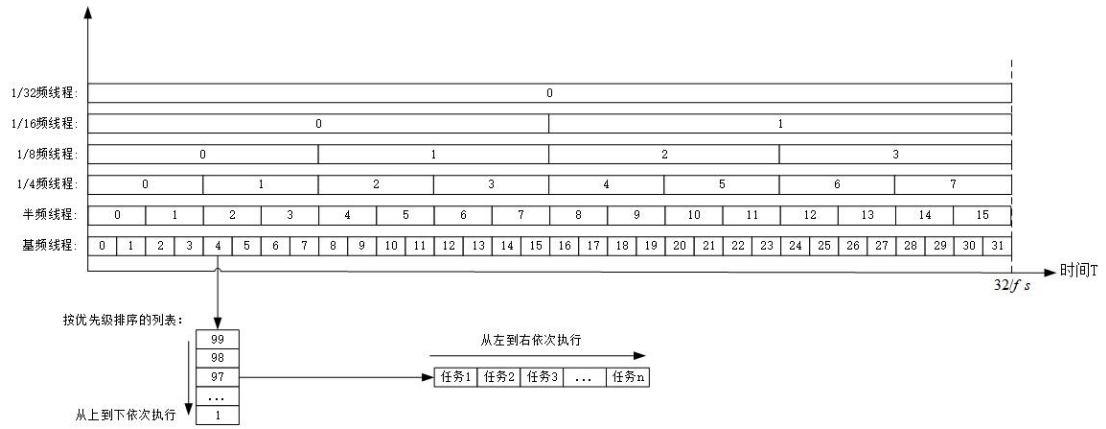

图 2 线程调度任务表

图中为 $32/f$ 秒内的调度任务表，f为运行频率，具体为：

1．表的第一维长度由线程运行频率决定，即坐标轴之内的列表。线程运行频率共有 6 种类型：基频、半频、1/4频、1/8频、1/16频、1/32频，不同频率的线程具有不同的表长度，用户可以在配置文件中指定某些模型运行所在线程的频率类型。

举例说明：以基频运行的线程的调度表中有 32 个位置（节点），每$1/f$秒的时间片内执行一个节点（即运行频率为f），该线程可以调度执行需要以基频运行或不高于基频运行的模型；以半频运行的线程的调度表中有 16 个位置（节点），每$2/f$秒的时间片内执行一个节点（即运行频率为$f/2$），该线程可以调度执行需要以半频运行或不高于半频运行的模型；以此类推，以$1/32$频运行的线程的调度表中只有一个位置（节点），$32/f$秒的时间片内执行一个节点（即运行频率为$f/32$），该线程只能调度以$1/32$频运行的模型。

每个节点中为该时间片内所有需要执行的调度任务（即模型的周期性执行任务）。当使用较高频率的线程调度较低频率的模型时，将根据节点号将模型的调度任务填入对应的位置。举例说明：若有一个基频线程（运行周期为$1/f$），需要调度一个以半频运行的模型（运行周期为$2/f$）时，模型在提交周期性执行任务时需要指定一个节点编号（此时为 0 或 1），表示该周期性执行任务是在每$2/f$的时间段的前$1/f$（对应于节点号 0）还是后$1/f$（对应于节点号 1）执行。这可以保证某些模型间如果具有执行顺序要求时，保证模型间能按照严格的顺序分时运行。

2．由于每个节点包含了该线程在该时间片需要执行的所有任务，因此需要展开为表的第二维度。它是一个以优先级降序排序的列表，在同一时间片内，高优先级任务先执行，低优先级任务后执行。线程在当前时间片内从优先级高的任务列表开始依次执行。

3．由于同一优先级可能会有多个调度任务需要执行，这展开为调度表的第三维度。表

中任务的执行顺序则由任务提交时的先后顺序决定。

举例说明度表的填写与执行过程。假设当前有一个基频线程，需要在该线程中依次提交3个以基频运行的任务（a）、（b）、（c），它们的优先级分别为99、98、98，节点号均为0。还需要提交1个以半频运行的任务（d），优先级为99，节点号为1。所有任务按照上述顺序依次提交，具体的填写过程如下：

1．首先创建一个基频线程，线程的调度表有32个节点，将调度表看作一个第一维长度为32的且其余两维长度不定的数组A[32][x][y]；

2．以基频运行、优先级为99、节点号为0的任务（a）将填写32份在A[n][99][0]处，其中 $n = 0,1,2,\ldots,31$；

3．以基频运行、优先级为98、节点号为0的任务（b）将填写32份在A[n][98][0]处，其中 $n = 0,1,2,\ldots,31$；

4．以基频运行、优先级为98、节点号为0的任务（c）将填写32份在A[n][98][1]处，其中 $n = 0,1,2,\ldots,31$；

5．以半频运行、优先级为99、节点号为1的任务（d）将填写16份在A[m][99][1]处，其中 $m = 1,3,5,\ldots,31$。

调度表填写完成后，每$32/f$秒的调度周期中任务的调度顺序具体为：

1．第0秒，任务执行顺序（a）→（b）→（c）

2．第$1/f$秒，任务执行顺序（a）→（d）→（b）→（c）

3．...

4．第$30/f$秒，任务执行顺序（a）→（b）→（c）

5．第$31/f$秒，任务执行顺序（a）→（d）→（b）→（c）

4.1.3. 基本处理流程

4.1.3.1. 软件流程图

“玄鸟”架构的基本处理流程如图 3 所示。

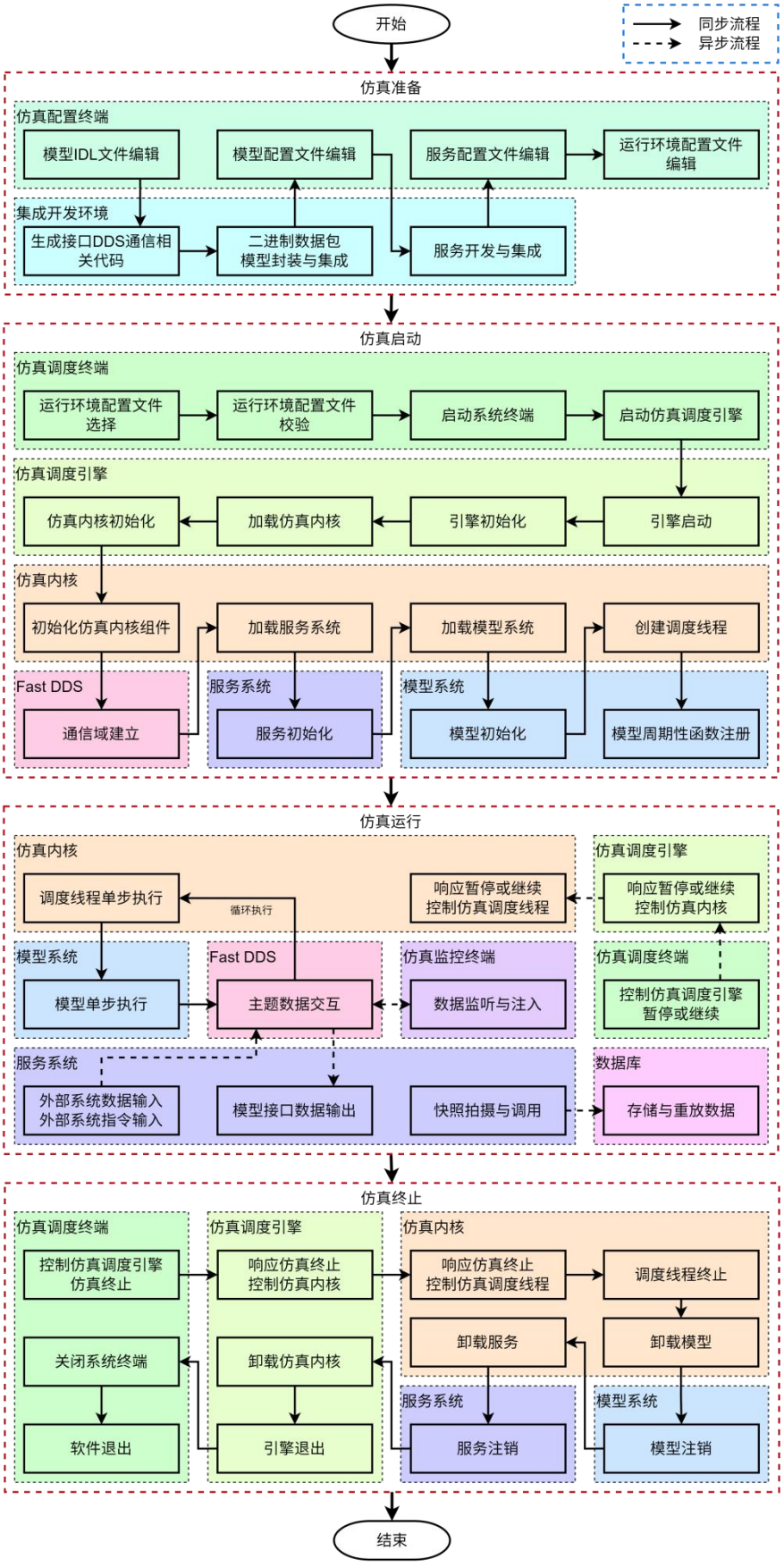

图 3 “玄鸟”架构基本流程图

“玄鸟”架构的基本处理流程包含仿真准备、仿真启动、仿真运行以及仿真终止四个主要流程阶段。每个主要流程阶段具体执行流程的设计见第 5 章运行设计。本节简要描述其基本处理：

1．仿真准备阶段：

1）使用仿真配置终端根据二进制数据包模型的 ICD 文件编写模型接口 IDL 文件；

2）使用集成开发环境中提供的 Fast DDS Gen 工具生成 DDS 通信相关代码；

3）使用集成开发环境中提供的 IDE 工具进行二进制数据包模型的封装与集成，模型封装框架由仿真内核提供；

4）使用仿真配置终端编辑模型配置文件，以配置模型调度运行所需参数；

5）使用集成开发环境中提供的 IDE 工具进行服务开发，以实现特定的扩展功能（如与外部系统通信的功能等），服务开发框架由仿真内核提供；

6）使用仿真配置终端编辑服务配置文件，以配置服务运行所需参数；

7）使用仿真配置终端编辑运行环境配置文件，以配置仿真运行所需参数。

2．仿真启动阶段：

1） 使用仿真调度终端选择运行所需的运行环境配置文件；

2） 仿真调度终端校验运行环境配置文件的有效性；

3） 仿真调度终端启动系统终端，并建立与系统终端的链接以便接收仿真调度引擎的输出；

4） 仿真调度终端通过系统终端启动仿真调度引擎；

5） 仿真调度引擎启动；

6） 仿真调度引擎进行初始化设置（例如设置引擎进程的 CPU 亲和性等）；

7） 仿真调度引擎加载仿真内核动态库；

8） 仿真调度引擎调用仿真内核的初始化；

9） 仿真内核完成其内部组件的初始化工作；

10） 仿真内核调用 Fast DDS 的 API 建立 DDS 通信域；

11） 仿真内核加载服务系统中的所有服务；

12） 服务系统中的服务进行初始化；

13） 仿真内核加载模型系统中的所有模型；

14） 模型系统中的模型进行初始化；

15） 仿真内核创建调度线程；

16） 模型系统中的模型向调度线程注册周期性函数。

3．仿真运行阶段分为同步流程（按固定周期执行的流程）和异步流程（事件触发式流程）：

1） 同步流程：

a. 仿真内核控制所有调度线程单步执行；

b. 调度线程调度模型系统中的模型进行单步执行；

c. 模型通过 Fast DDS 进行主题数据交互；

d. 以上流程循环执行。

2） 异步流程：

a. 用户通过仿真调度终端能够控制仿真的暂停与继续，此时仿真调度终端向仿真调度引擎发出控制指令；

b. 仿真调度终端接收并响应控制指令，同时向仿真内核发出控制指令；

c. 仿真内核接收并响应控制指令，控制调度线程的运行；

d. 仿真监控终端可以通过 Fast DDS 进行数据的监听与注入；

e. 服务系统会响应外部系统输入的指令或数据，通过 Fast DDS 发布至指令或数据的目的地；

f. 服务系统会按照指令采集模型接口数据并向外部系统输出；

g. 服务系统会按照指令进行快照的拍摄与调用，调用 Fast DDS 提供的 Record and Replay 工具的 API；

h. Fast DDS Record and Replay 工具会根据指令进行快照数据的存储与重放。

4．仿真终止阶段：

1） 用户通过仿真调度终端下达仿真终止指令，仿真调度终端向仿真调度引擎发出终止指令；

2） 仿真调度引擎接收并响应终止指令，同时向仿真内核发出终止指令；

3） 仿真内核接收并响应终止指令，终止调度线程；

4） 仿真内核卸载模型系统；

5） 模型系统执行所有模型的注销操作；

6） 仿真内核卸载服务系统；

7） 服务系统执行所有服务的注销操作；

8） 仿真调度引擎卸载仿真内核；

9） 仿真引擎退出；

10） 仿真调度终端关闭系统终端；

11） 仿真调度终端退出，仿真结束。

4.1.3.2. 数据流程图

“玄鸟”架构的数据流程图如图 4 所示。

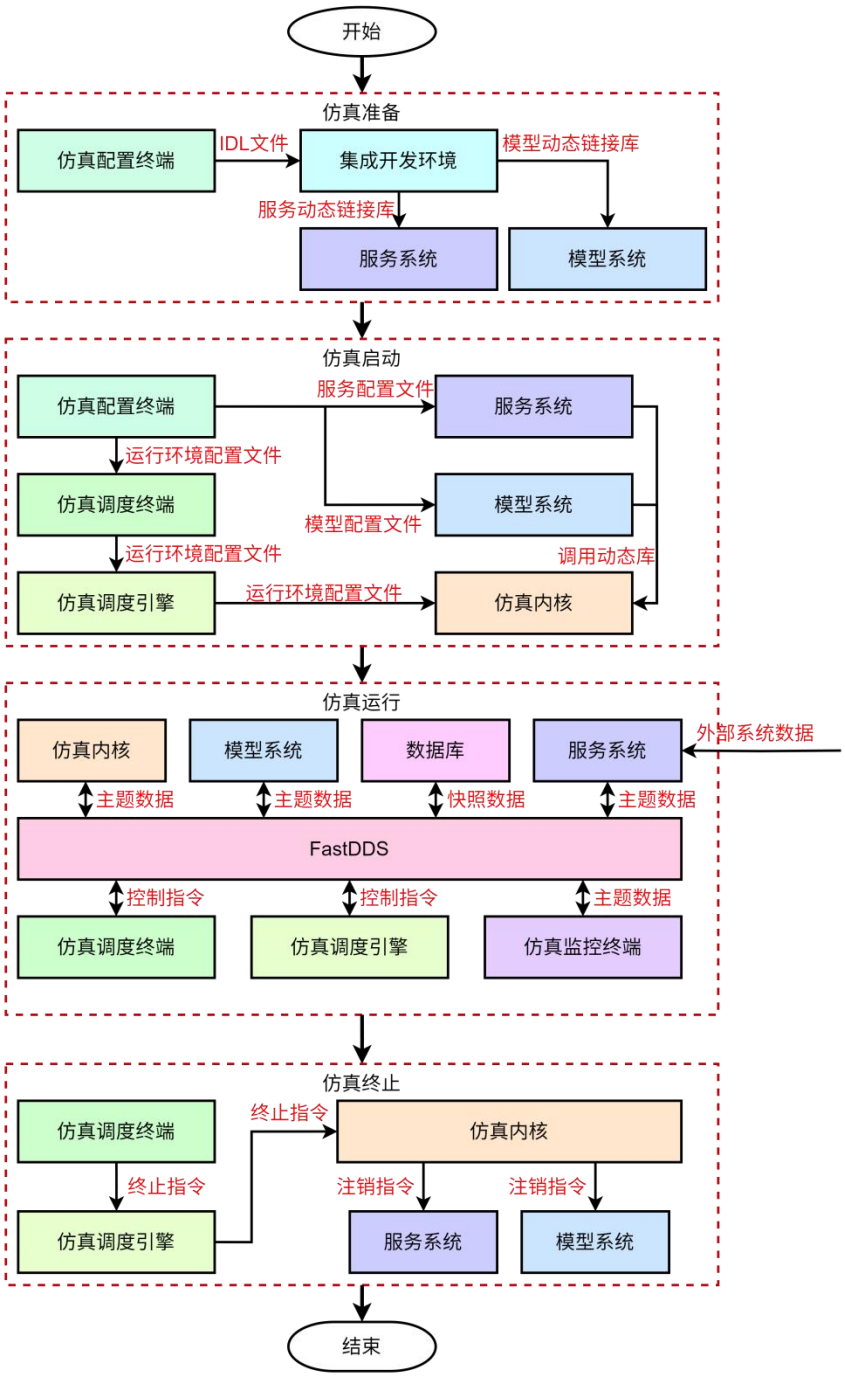

图 4 “玄鸟”架构数据流程图

“玄鸟”架构中的交互数据流程为：

1．仿真准备阶段：

1） 仿真配置终端生成的 IDL 文件将在集成开发环境中进行二进制数据包模型的封装集成时使用；

2） 封装完成的二进制数据包模型以模型动态链接库的形式集成至模型系统中；

3） 开发的服务以服务动态链接库的形式集成至服务系统中。

2．仿真启动阶段：

1） 仿真配置终端生成的运行环境配置文件将由仿真调度终端进行读取；

2） 仿真调度终端将用户选择的运行环境配置文件传递给仿真调度引擎；

3） 仿真调度引擎再将运行环境配置文件传递给仿真内核进行解析；

4） 仿真内核根据运行环境配置文件中的内容调用模型动态链接库和服务动态链接库；

5） 模型初始化时读取仿真配置终端生成的模型配置文件进行参数配置；

6） 服务初始化时读取仿真配置终端生成的服务配置文件进行参数配置。

3．仿真运行阶段：

1） 仿真内核、模型系统、服务系统、仿真监控终端均通过 Fast DDS 的主题发布订阅模式进行主题数据的交互；

2） 仿真调度终端和仿真调度引擎通过 Fast DDS 发送仿真运行控制指令；

3） 数据库通过 Fast DDS 进行快照数据的存储与重放。

4．仿真终止阶段：

1）仿真调度终端向仿真调度引擎发送仿真终止指令；

2）仿真调度引擎接收并响应终止指令，向仿真内核发送仿真终止指令；

3）仿真内核接收并响应仿真终止指令，向模型系统和服务系统发送注销指令。

4.1.4. 软件体系结构

4.1.4.1. 软件配置项

“玄鸟”架构的组成结构如图 5 所示：

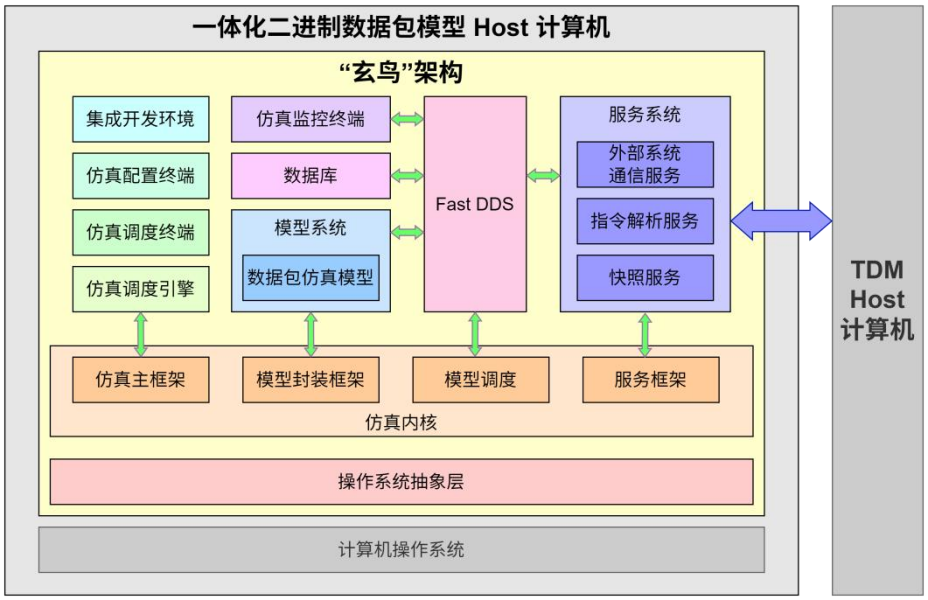

图 5 “玄鸟”架构组成结构图

“玄鸟”架构由以下几个软件配置项组成：

1．**仿真配置终端**：标识符 XNEditor。仿真配置终端是一个可视化前端界面，可以对“玄鸟”架构运行参数、模型参数、接口参数等进行配置；

2．**仿真调度终端**：标识符 XNRunner。仿真调度终端是一个可视化前端界面，可以通过界面配置“玄鸟”架构运行的参数，也可以进行“玄鸟”架构的启动、暂停、继续、停止等控制运行的操作；

3．**仿真调度引擎**：标识符 XNEngine。仿真调度引擎是仿真的主进程，主要负责仿真内核的加载、初始化与运行，同时通过共享内存接受外部控制指令以控制“玄鸟”架构运行；

4．操作系统抽象层：标识符 XNOSLayer。操作系统抽象层使得软件不依赖于底层操作系统，软件不需要进行大量的修改即可在不同操作系统上运行；

5．**仿真内核**：标识符 XNCore。仿真内核主要负责“玄鸟”架构运行参数配置、服务加载管理、模型加载管理、模型周期性调度、运行线程管理、时间步进、共享内存空间管理、运行日志记录等工作；

6．**模型系统**：标识符 XNModels。模型系统是利用仿真内核提供的模型封装框架定制化封装与开发的由仿真内核动态加载的需要周期性调度的模型或其它功能模块的集合，仿真内核通过配置文件中的配置项加载需要调度运行的模型。模型系统不仅可以加载数据包模型，还可以加载其它需要周期性调度的模型，例如数据处理模型能够处理数据包模型产生的数据以供仿真内核或其它模块使用；

7．**服务系统**：标识符 XNServices。服务系统是利用仿真内核提供的服务框架定制化开

发的由仿真内核动态加载的可扩展功能的服务模块集合，仿真内核通过配置文件中的配置项加载需要的服务。服务系统中加载的服务可以执行与外部系统间的离散量与虚拟航空总线通信、控制指令解析与执行、快照拍摄和调用等功能；

8．**Fast DDS**：标识符 FastDDS。Fast DDS 提供了软件内部基于共享内存的发布/订阅模式的交互通道。

9．**数据库**：标识符 XNSnapshot。数据库用于存储软件运行过程中的快照信息。快照服务会根据指令从数据库中读取/写入快照数据；

10．**仿真监控终端**：标识符 XNMonitor。仿真监控终端是一个可视化前端界面，可以在软件运行过程中监控运行数据。仿真监控终端同时具备数据采集记录的功能，能够根据需求将软件运行过程中的数据记录成文件，以供分析和查看；

11．**集成开发环境**：标识符 XNIDE。二进制数据包模型的封装人员使用指定的集成开发环境将模型集成至“玄鸟”架构中。另外，用户还可以根据需要定制化开发其他数据处理模型或服务集成至“玄鸟”架构中。

4.1.4.2. 软件层次结构

“玄鸟”架构的软件层次分为操作系统抽象层、数据层、业务层、表现层四个层级，其结构图如图 6 所示：

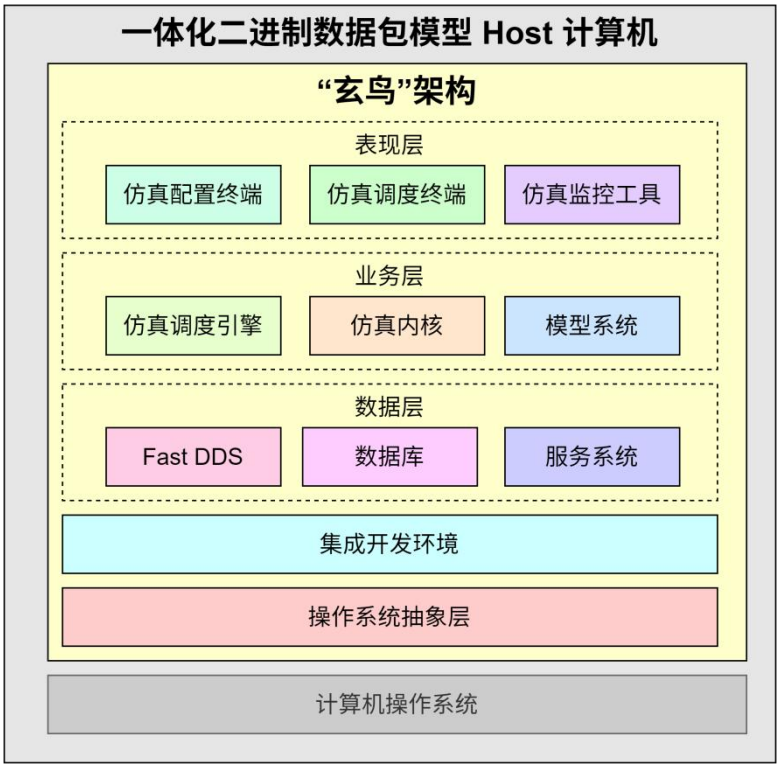

图 6 “玄鸟”架构软件层次结构图

操作系统抽象层解除软件对操作系统的依赖，使软件能够灵活地部署在不同的操作系统上。

集成开发环境用于二进制数据包模型的封装与服务开发。

数据层负责数据的传输与存储，包括：使用 Fast DDS 的基于共享内存的发布/订阅机制数据交互；使用 Fast DDS 的 Record and Replay 对数据库进行快照数据的存取；使用服务系统进行基于 TCP/UDP 的外部系统指令响应和数据交互。

业务层负责软件的基本业务逻辑处理，包括：仿真调度引擎负责加载仿真内核进行仿真；模型系统负责数据包模型的加载；仿真内核负责模型系统中数据包仿真模型的实时调用与其它事件处理。

表现层提供一系列工具给用户，包括：仿真配置终端用于配置软件运行参数；仿真调度终端用于启动仿真调度引擎以及控制引擎运行；仿真监控终端用于软件运行过程中的数据监控与调试。

4.1.4.3. 软件配置项功能描述

本节简要阐述“玄鸟”架构各软件配置项具有的功能。

4.1.4.3.1. 仿真配置终端

仿真配置终端是一个可视化前端界面，用于软件运行参数的配置，包括以下功能：

1. 支持编辑 XML 格式的配置文件类型，同时支持编辑接口定义所用的 IDL 文件；

2. 能够对配置文件进行新建、打开、关闭、保存、另存为等操作；

3. 能够以清晰的表格或列表形式显示各配置参数，并能够对各参数进行增、删、改、查操作；

4. 能够以树型结构展示接口定义 IDL 文件，并能够对各接口进行增、删、改、查操作；

5. 能够提供运行环境配置文件、模型配置文件、服务配置文件和接口 IDL 文件的基础模板。

4.1.4.3.2. 仿真调度终端

仿真调度终端是一个可视化前端见面，用于仿真调度引擎的启动与运行控制，包括以下功能：

1. 能够以下拉列表的形式展示与选择所有运行环境配置文件；

2. 能够根据所选运行环境配置文件调用仿真调度引擎对模型与服务进行预加载，检验配置文件是否正确，以及模型与服务是否可以正常加载；

3. 能够显示仿真调度引擎运行过程中输出的运行消息，并且可以支持用户选择需要显示哪些消息；

4. 能够启动仿真调度引擎，向仿真调度引擎提供所选的运行环境配置文件；

5. 能够控制仿真调度引擎的暂停、继续和停止。

4.1.4.3.3. 仿真调度引擎

1. 能够加载仿真内核动态库；

2. 能够调用仿真内核的运行环境配置文件解析功能完成运行环境配置文件的解析；

3. 能够以预加载模式调用仿真内核，检验配置文件是否正确，以及模型与服务是否可以正常加载；

4. 能够控制仿真内核开始进行仿真；

5. 能够接受仿真调度终端的暂停、继续和停止运行控制指令，并控制仿真内核做出相应的响应；

4.1.4.3.4. 操作系统抽象层

1. 能够根据不同的硬件设备、操作系统、编译器将常用的数据类型抽象出来；

2. 能够将操作系统相关的操作（如线程管理、内存管理、定时器、I/O 等）抽象出来；

3. 能够提供一套与操作系统无关的 API 接口，使得软件可以不依赖于特定的操作系统实现细节。

4.1.4.3.5. 仿真内核

1. 能够以动态库的形式被仿真调度引擎加载；

2. 能够完成运行环境配置文件的解析；

3. 能够以预加载模式进行运行环境配置文件、模型配置文件、服务配置文件的解析，检验配置文件的正确性；

4. 能够以预加载模式进行模型动态库、服务动态库的调用测试，检验模型与服务是否可以正常加载；

5. 能够根据运行环境配置文件中列出的模型与服务，加载对应的模型动态库与服务动态库；

6. 能够解析已加载模型的模型配置文件，设置模型参数；

7. 能够解析已加载服务的服务配置文件，设置服务参数；

8. 能够以多线程多频率调度模型实时运行；

9. 能够通过 Fast DDS 建立模型间数据交互、服务间数据交互、模型与服务间数据交互的发布/订阅接口；

10. 能够在运行过程中产生仿真运行事件来激励某些功能运行，例如：定期向外部系统发送数据等；

11. 能够通过回调机制响应运行过程中仿真内核及其他模块产生的事件；

12. 提供标准化的模型封装框架，提供初始化接口、周期性调度接口及数据发布/订阅接口等；

13. 提供标准化的服务开发框架，提供初始化接口、数据发布订阅接口、事件回调接口及事件提交接口等；

4.1.4.3.6. 模型系统

1. 能够动态加载并调用二进制数据包仿真模型；

2. 能够根据标准化的模型封装框架提供的初始化接口完成二进制数据包模型的初始化；

3. 能够根据标准化的模型封装框架提供的周期性调度接口完成二进制数据包模型的周期性调度；

4. 能够根据标准化的模型封装框架提供的数据发布/订阅接口进行数据交互；

5. 能够进行其它相关数据处理。

4.1.4.3.7. 服务系统

1. 能够根据标准化的服务开发框架完成服务的初始化；

2. 能够根据标准化的服务开发框架提供的数据发布/订阅接口进行数据交互；

3. 能够根据标准化的服务开发框架提供的事件回调接口对事件进行响应；

4. 能够根据需要产生事件，并根据标准化的服务开发框架提供的事件提交接口向仿真内核提交事件；

5. 负责离散量通信的服务能够通过 TCP/UDP 与外部系统进行数据交互，并通过数据发布/订阅接口与仿真内核进行交互；

6. 负责虚拟航空总线通信的服务能够以标准的航空总线协议对数据进行打包/解包，并通过 TCP/UDP 与外部系统进行数据交互。支持 CAN（ARINC 825）总线、ARINC 429/ARINC 664 总线和 ARINC 708 总线。该服务通过数据发布/订阅接口与仿真内核进行交互；

7. 负责控制指令解析的服务能够通过 TCP/UDP 接收外部系统的控制指令，并发送指令

响应信息。该服务能够解析包含通配符的指令，并根据解析完成后的指令对仿真内核或对应的模型/服务进行控制；

8. 负责快照的服务能够根据快照的拍照/调用指令，操作数据库写入/读取快照信息；快照信息通过数据发布/订阅接口提供给仿真内核及其它模型；

4.1.4.3.8. Fast DDS

1. 提供高性能的 DDS 实时发布/订阅框架，支持快速、可靠的数据交换;

2. 提供可靠的数据传输和身份验证机制，以确保数据的机密性和完整性。支持加密和访问控制，保护敏感数据不受未授权方访问。

4.1.4.3.9. 数据库

“玄鸟”架构使用的 Fast DDS Record and Replay 将快照数据存储在数据库文件中，该软件具有以下功能：

1. 将发布到 DDS 环境中的数据保存到 MCAP 格式的数据库中，包括发布时间戳、序列化数据及其格式；

2. 输出的 MCAP 文件是一种用于存储多模态数据的开源容器文件格式，可以被任何兼容的工具读取，因为它包含了读取和重放数据所需的所有信息；

3. 提供了启动、停止和暂停数据记录的远程控制功能。用户可以通过 API 从其他设备发送命令来控制记录工具。

4.1.4.3.10. 仿真监控终端

1. 能够使用 Fast DDS 的订阅机制读取仿真运行状态、线程运行状态、模型运行状态等信息；

2. 能够使用 Fast DDS 的订阅机制读取模型交互数据进行监控和分析；

3. 能够配置并使用列表与图表展示需要监控的数据；

4. 能够设置需要采集的数据、采集时间、采集频率等来进行数据采集并保存为数据文件；

5. 能够读取并加载数据并绘制图表，包括采集的数据或参考数据；

6. 能够使用 Fast DDS 的数据发布机制向模型注入数据以调试模型。

4.1.4.3.11. 集成开发环境

1. 提供二进制数据包模型封装与服务开发的 IDE 工具；

2. 提供二进制数据包模型封装与服务开发所需的依赖库；

3. 提供一系列便于模型封装与服务开发的工具。

4.2. 软件配置项设计

4.2.1. 软件配置项标识

本节用项目唯一标识符列出了“玄鸟”架构的每个软件配置项，见表 2，表中列出了软件配置项的编号、名称、标识符及版本。

表 2 “玄鸟”架构软件配置项标识

编号	名称	标识符	版本
1.	仿真配置终端	XNEditor	V1.0
2.	仿真调度终端	XNRunner	V1.0
3.	仿真调度引擎	XNEngine	V1.0
4.	操作系统抽象层	XNOSLayer	V1.0
5.	仿真内核	XNCore	V1.0
6.	模型系统	XNModels	V1.0
7.	服务系统	XNServices	V1.0
8.	eProsima Fast DDS	FastDDS	V3.1.0
9.	数据库	XNMACP	/
10.	仿真监控终端	XNMonitor	V1.0
11.	集成开发环境	XMIDE	/

4.2.2. XNEditor

4.2.2.1. 概述

1．标识符：XNEditor

2．名称：仿真配置终端

3．开发状态：新开发软件配置项

4.2.2.2. 结构设计

仿真配置终端的结构如图 7 所示，包含 XML 文件编辑、IDL 文件编辑、配置文件管理、配置文件模板和配置文件列表显示与修改五个功能模块。

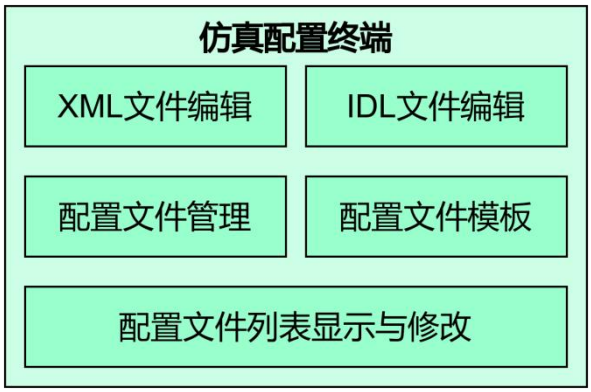

图 7 仿真配置终端结构设计

4.2.2.3. 工作流程

仿真配置终端的工作流程如图 8 所示。

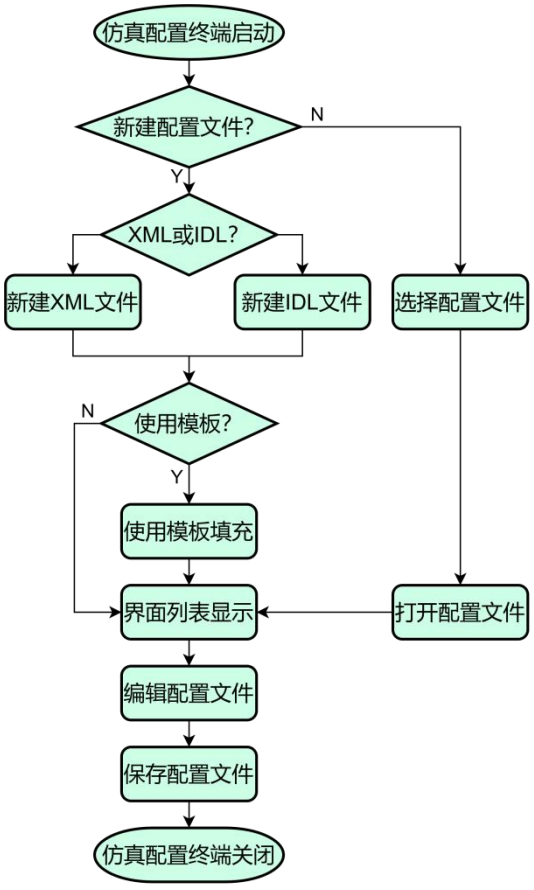

图 8 仿真配置终端工作流程

4.2.2.4. 模块设计

4.2.2.4.1. XML 文件编辑

XML（eXtensible Markup Language，可扩展标记语言）是一种用于存储、传输和交换

数据的标记语言。“玄鸟”架构使用 XML 语言编写的 XML 文件来存储与运行环境、模型配置、服务配置等相关的参数。

仿真配置终端使用 TinyXML2 库进行 XML 文件的读取与修改。TinyXML2 库是一个轻量级的 C++ XML 解析库，它提供了一组简单的类和函数来处理 XML 数据。并且 TinyXML2 可以在多个操作系统上运行，包括 Windows、Linux 和 Mac OS X 等，它使用标准的 C++代码编写，不依赖于特定的操作系统功能。同时，Fast DDS 也恰好依赖 TinyXML2 库，因此使用该库可以避免引入其他 XML 解析库。

4.2.2.4.2. IDL 文件编辑

IDL（Interface Definition Language，接口定义语言）是一种用于描述软件之间通信接口的语言。在 Fast DDS 中，IDL 用于定义数据类型和消息格式，以便在分布式系统中进行通信。在 IDL 中，可以使用 struct 关键字来定义结构体，结构体可以包含多个字段，每个字段可以是不同的数据类型。“玄鸟”架构使用 IDL 语言重新描述模型的 ICD，产生描述模型接口的 IDL 文件。

仿真配置终端以文本文件的读写机制来处理 IDL 文件，允许用户轻松地读取和修改这些关键文件。此外，仿真配置终端还具有 IDL 文件正确性检验功能，确保文件内容符合语法和逻辑规范。

4.2.2.4.3. 配置文件管理

仿真配置终端为用户提供了一套全面的配置文件管理功能，包括创建新配置文件、打开现有文件、保存更改、另存为新文件以及关闭文件等。此外，仿真配置终端采用直观的树状结构来展示与当前运行环境配置文件相链接的模型配置文件和服务配置文件，使得用户能够清晰地理解和管理整个仿真环境的配置层次。这样的设计不仅增强了用户体验，也提高了配置管理的效率和准确性。

4.2.2.4.4. 配置文件模板

仿真配置终端提供以下配置文件的模板：

1. 运行环境配置文件模板：包含 XML 元素名称为操作系统版本、CPU 亲和性掩码、调度频率、工作目录、模型库目录、模型列表、服务列表等 XML 元素的 XML 配置文件模板；

2. 模型配置文件模板：包含 XML 元素名称为模型名称、周期性函数列表、指令列表等 XML 元素的 XML 配置文件模板，周期性函数列表元素包含一个默认名称为函数的子元素，

包含函数名、运行节点和优先级三个默认的 XML 属性；

3. 服务配置文件模板：包含 XML 元素名称为服务名称、自定义事件列表、指令列表等 XML 元素的 XML 配置文件模板，自定义事件列表元素包含一个默认名称为事件的子元素，包含事件名、事件标识和优先级三个默认的 XML 属性；

4. IDL 文件模板：包含一个名为 XNSim 的命名空间，该命名空间下定义有一个名为 Example 的空结构体。

4.2.2.4.5. 配置文件列表显示及修改

仿真配置终端通过友好的用户界面提供配置文件内容的显示与修改。针对 XML 配置文件，仿真配置终端支持 XML 的元素名称、元素文本、属性名称、属性文本的增、删、改、查。

针对 IDL 文件，仿真配置终端支持接口定义的命名空间名称、结构体名称、接口名称、数据类型、数组形式、数组大小等参数的增、删、改、查。

4.2.3. XNRunner

4.2.3.1. 概述

1．标识符：XNRunner

2．软件配置项名称：仿真调度终端

3．开发状态：新开发软件配置项

4.2.3.2. 结构设计

仿真调度终端的结构如图 9 所示，包含运行环境配置、配置文件校验、模型/服务加载清单、仿真调度引擎控制和仿真调度引擎输出显示五个功能模块。

图 9 仿真调度终端结构设计

4.2.3.3. 工作流程

仿真调度终端的工作流程如图 10 所示。

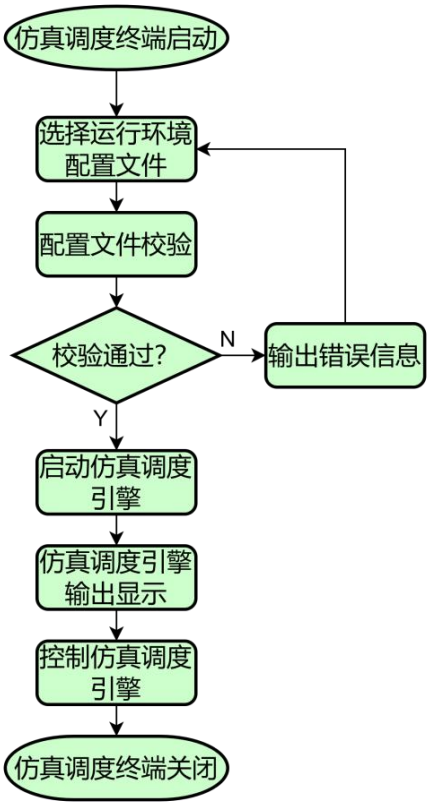

图 10 仿真调度终端工作流程

4.2.3.4. 模块设计

4.2.3.4.1. 运行环境配置

仿真调度终端将在默认的目录下（仿真调度终端工作目录下的 Scenario 文件夹）寻找所有文件后缀名为.xml 或.sce 的文件，以下拉菜单的形式显示所有运行环境配置文件。用户选择一个运行环境配置文件后，仿真调度终端将启动配置文件校验。

4.2.3.4.2. 配置文件校验

仿真调度终端以预加载模式启动仿真调度引擎，验证运行环境配置文件的正确性。运行环境配置文件验证完成后，按照其中配置的模型与服务，验证模型配置文件和服务配置文件的正确性，并尝试动态加载模型与服务的动态链接库。在此过程中将暂时禁用界面其它操作。校验过程结束后，将校验结果输出在仿真调度引擎输出显示，并以不同颜色在模型/服务加载清单中表示它们加载校验的结果。

4.2.3.4.3. 模型/服务加载清单

仿真调度终端能够读取运行环境的配置文件，并从中提取出需要加载的模型和服务的列表。这样的设计使得用户可以一目了然地看到即将被加载的模型和服务，从而更加便捷地进行仿真环境的管理和监控。

4.2.3.4.4. 仿真调度引擎控制

仿真调度终端通过后台启动一个系统终端并使用命令行语句来启动仿真调度引擎，并将所选择的运行环境配置文件通过命令行参数传递给仿真调度引擎。

在仿真调度引擎运行过程中，仿真调度终端可以通过 Fast DDS 向仿真调度引擎发布运行控制指令主题，控制其暂停运行、继续运行或终止运行。

4.2.3.4.5. 仿真调度引擎输出显示

仿真调度引擎启动后，仿真调度终端便能够即时捕获并显示该引擎在系统终端中生成的输出信息，这确保了用户能够随时监控仿真进度和状态。

仿真调度终端还提供了一个用户友好的功能，即通过输出信息配置选项，用户可以轻松实现输出消息的过滤，以便仅展示他们关心的信息。这一特性使得信息管理更加高效，确保用户能够专注于最关键的数据。

4.2.4. XNEngine

4.2.4.1. 概述

1．标识符：XNEngine

2．软件配置项名称：仿真调度引擎

3．开发状态：新开发软件配置项

4.2.4.2. 结构设计

仿真调度引擎的结构如图 11 所示，包含命令参数解析、仿真内核预加载、仿真内核加载及调用和控制指令监听及仿真运行控制四个功能模块。

图 11 仿真调度引擎结构图

4.2.4.3. 工作流程

仿真调度引擎的工作流程如图 12 所示。

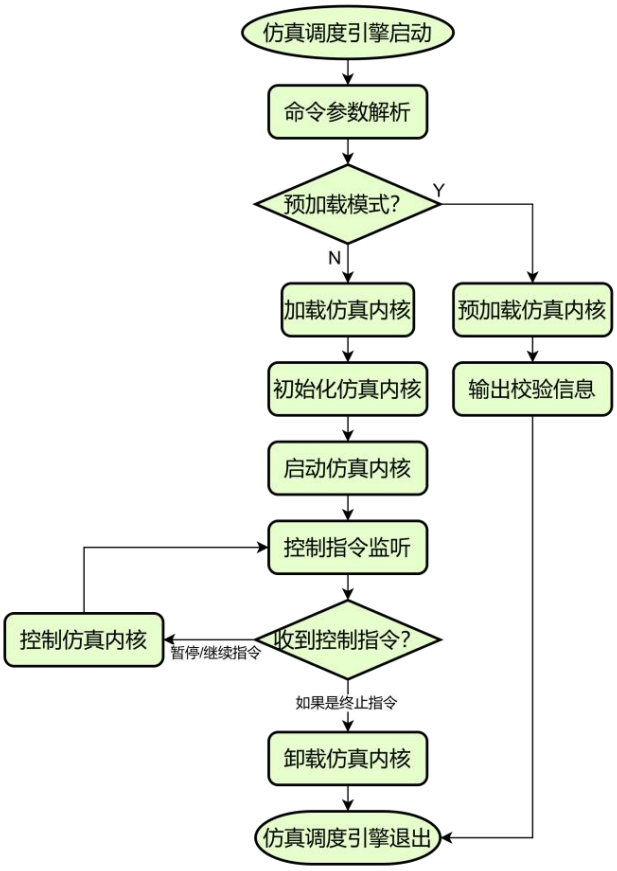

图 12 仿真调度引擎工作流程

4.2.4.4. 模块设计

4.2.4.4.1. 命令参数解析

仿真调度引擎在启动时进行命令行参数解析，以确保正确配置和启动仿真。能够解析的命令行参数包括：运行环境配置文件路径、仿真内核加载模式及其它运行参数。

运行环境配置文件路径是命令行参数中必须提供的第一个参数，且必须指向一个有效的

配置文件路径。这个文件包含了仿真运行所需的关键设置和参数。

仿真内核加载模式通过特定的标识符“-m”来识别是否启用预加载模式。如果该标识符后跟随的参数指示启用预加载，仿真内核将按照预加载模式运行，用于检验配置文件与模型的可用性。

除了上述两个参数外，还可以通过其他特定的标识符来指定额外的运行参数，以满足不同的仿真需求。

4.2.4.4.2. 仿真内核预加载

当命令行参数被指定为预加载模式时，仿真调度引擎对仿真内核进行加载来执行运行环境配置文件校验。在这个过程中，仿真内核将会校验运行环境配置文件及其相关的模型配置文件和服务配置文件，并尝试动态加载所需的模型和服务的动态库。

预加载时如果出现校验失败，会向系统终端及日志文件中打印错误信息；如果校验成功，则向系统终端报告校验通过。校验完成后直接退出引擎。

4.2.4.4.3. 仿真内核加载及调用

当命令行参数被指定为正常加载模式时，仿真调度引擎将正常加载仿真内核来解析运行环境配置文件。在这个过程中，仿真内核完成所有相关配置文件解析、加载相关模型及服务动态库、建立模型调度线程、建立事件监听等仿真准备工作。

一旦上述准备工作全部完成，仿真内核将报告仿真调度引擎，表明仿真环境已准备就绪。在确认仿真内核已做好所有准备工作后，仿真调度引擎将接管控制权，指导仿真内核开始仿真运行。

4.2.4.4.4. 控制指令监听及仿真运行控制

在仿真运行过程中，仿真调度引擎作为仿真的主线程会通过 Fast DDS 订阅运行控制指令主题，根据接收到的运行控制指令通知仿真内核的进行相应的响应：

1. 暂停指令：通知仿真内核运行暂停，仿真内核将暂停运行所有线程；

2. 继续指令：通知仿真内核运行继续，仿真内核将继续运行所有线程；

3. 终止指令：通知仿真内核运行终止，仿真内核将结束运行所有线程，并注销所有已加载的模型与服务。仿真内核完成终止操作后，仿真调度引擎会卸载仿真内核并退出。

4.2.5. XNOSLayer

4.2.5.1. 概述

1．标识符：XNOSLayer

2．软件配置项名称：操作系统抽象层

3．开发状态：新开发软件配置项

4.2.5.2. 结构设计

操作系统抽象层的结构如图 13 所示，包含系统信息及环境变量获取、数据类型抽象、线程管理抽象、文件系统访问抽象、时间管理抽象、内存管理抽象、同步原语抽象、网络接口抽象、动态链接库抽象九个功能模块。

图 13 操作系统抽象层结构图

4.2.5.3. 模块设计

4.2.5.3.1. 系统信息和环境变量获取

操作系统抽象层将不同操作系统提供的系统信息和环境变量相关的 API 接口抽象成统一的 API 接口。在编译代码时，操作系统抽象层会根据当前操作系统的信息和环境变量，将这些抽象的 API 接口转换为特定于该操作系统的 API 接口。操作系统抽象层使用编译器特定的预定义宏获取一些基本的系统信息，例如：

1. “_M_IX86”、“_M_X64”等宏在 MSVC 编译器中指示 CPU 架构；
2. “_WIN32”、“_WIN64”、“linux”等宏指示操作系统平台；
3. “__cplusplus”宏指示编译器支持的 C++标准版本。

操作系统抽象层使用 CMake 构建系统，它可以在编译前运行脚本来检测系统信息和环境变量，并将这些信息通过预处理器指令传递给源代码。

4.2.5.3.2. 数据类型抽象

操作系统抽象层进行数据类型抽象的目的是为了确保应用程序能够在不同的操作系统

上以相同的方式处理数据类型。操作系统抽象层使用条件编译并定义类型别名的方式来统一不同平台上的数据类型，例如：可以利用获取到的操作系统平台类型宏进行条件编译，将不同操作系统上 32 位长度的无符号整数类型变量都定义其别名为“XNINT32”。

操作系统抽象层需要定义类型别名的数据类型包括：

1. 整数类型变量；
2. 布尔类型变量；
3. 字符类型变量；
4. 浮点数类型变量；
5. 字符串类型变量；
6. 指针和地址类型变量。

4.2.5.3.3. 文件系统访问抽象

操作系统抽象层使用条件编译将不同操作系统的文件系统访问 API 抽象成统一的 API 接口，包括以下两部分 API 接口的抽象：

1. 目录操作 API：包括目录的创建、删除、遍历等操作；
2. 文件操作 API：包括文件的打开、关闭、读取、写入、删除等操作。

4.2.5.3.4. 时间管理抽象

操作系统抽象层使用条件编译将不同操作系统的时间管理 API 抽象成统一的 API 接口，包括以下两部分 API 接口的抽象：

1. 时间获取：包括获取当前系统时间、时间运算等操作；
2. 睡眠：实现线程睡眠相关的操作；

4.2.5.3.5. 内存管理抽象

操作系统抽象层使用条件编译将不同操作系统的内存管理 API 抽象成统一的 API 接口，包括以下两部分 API 接口的抽象：

1. 动态内存分配：包括动态内存的分配与释放操作；
2. 内存锁定：包括内存的锁定与解锁操作；

4.2.5.3.6. 网络接口抽象

操作系统抽象层使用条件编译将不同操作系统的内存管理 API 抽象成统一的 API 接口，包括对 UDP/TCP 的套接字创建、建立连接、数据传输、监听等操作的 API 接口抽象。

4.2.5.3.7. 线程管理抽象

操作系统抽象层使用条件编译将不同操作系统的线程管理 API 抽象成统一的 API 接口，主要包括以下四部分 API 接口的抽象：

1. 线程的创建及销毁；
2. 线程的 CPU 亲和性设置；
3. 线程的调度参数设置；
4. 线程睡眠。

4.2.5.3.8. 同步原语抽象

操作系统抽象层使用条件编译将不同操作系统的同步原语及相关 API 抽象成统一别名的同步原语及 API 接口，主要包括以下五种同步原语的抽象：

1. 互斥锁；
2. 读写锁；
3. 信号量；
4. 条件变量；
5. 原子变量。

4.2.5.3.9. 动态链接库抽象

操作系统抽象层使用条件编译将不同操作系统下动态链接库动态加载的 API 抽象成统一的 API 接口，主要包括以下两种 API 的抽象：

1. 动态链接库的动态加载和卸载；
2. 动态链接库内函数的调用。

4.2.6. XNCore

4.2.6.1. 概述

1．标识符：XNCore

2．软件配置项名称：仿真内核

3．开发状态：新开发软件配置项

4.2.6.2. 结构设计

仿真内核的结构如图 14 所示。包含仿真主框架、场景描述管理器、线程管理器、DDS

管理器、时间管理器、事件管理器、模型管理器、服务管理器、日志记录、调度线程、模型封装框架、服务框架共十二个模块。

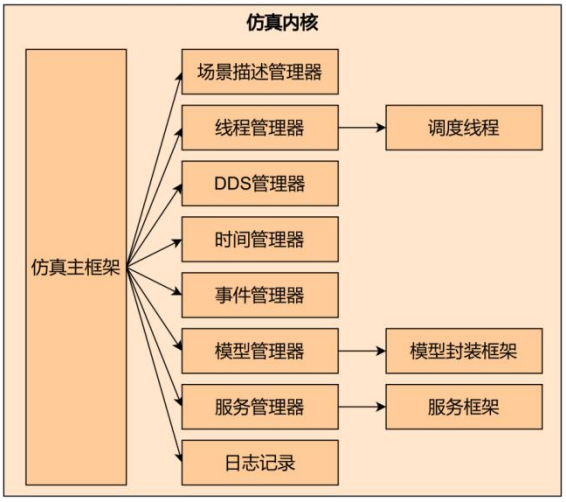

图 14 仿真内核结构图

4.2.6.3. 工作流程

仿真内核的工作流程如图 15 所示。

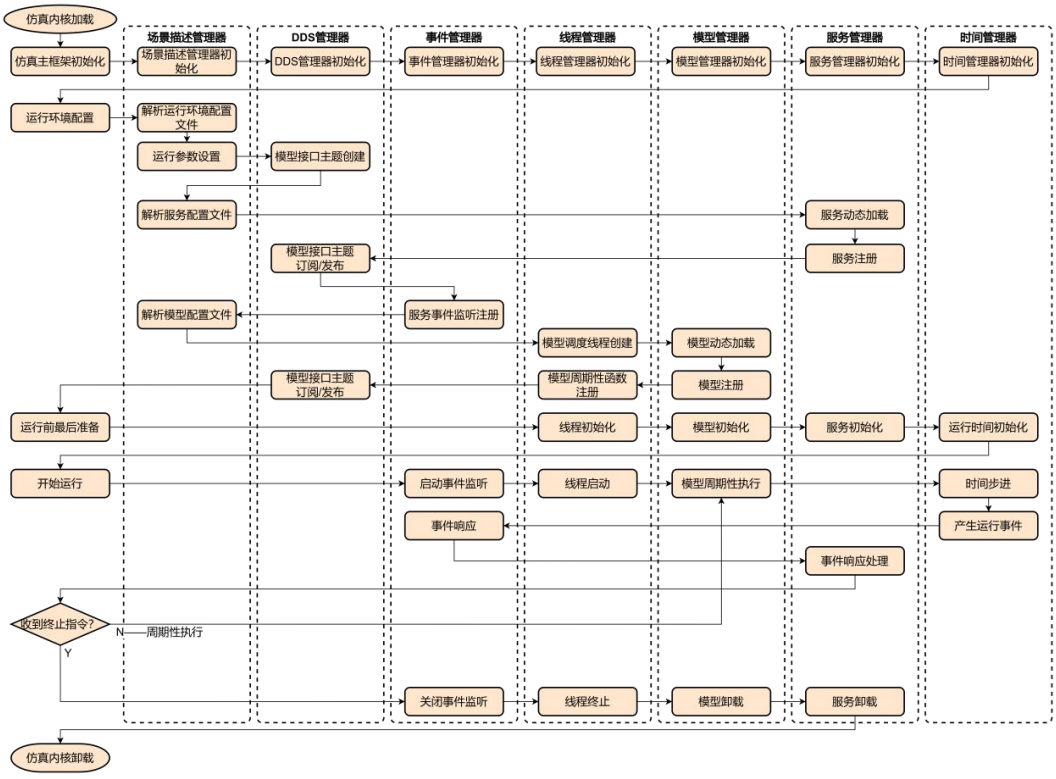

图 15 仿真内核工作流程

4.2.6.4. 模块设计

4.2.6.4.1. 仿真主框架

仿真主框架作为仿真内核的主体，扮演着仿真内核的管理和协调角色。它不仅定义了仿真内核的基本属性和接口，还负责整个仿真环境的管理和协调工作。

仿真主框架定义了仿真运行所需的一些属性及接口，包括：

1. 操作系统相关参数（系统类型、系统版本、系统内核类型等）及其设置与获取接口；

2. 仿真内核相关参数（版本号、作者、修改时间等）及其设置与获取接口；

3. 工作路径相关参数（仿真根目录、模型库目录、服务库目录、接口文件目录等）的及其设置与获取接口；

4. 运行相关参数（CPU 亲和性、基础运行频率等）及其设置与获取接口；

5. 获取其它模块（时间管理器、事件管理器、线程管理器等）智能指针的接口。

仿真主框架负责管理与调用仿真内核的其它模块，直接管理的模块有时间管理器、模型管理器、服务管理器、场景描述管理器、DDS 管理器、线程管理器、事件管理器。仿真主框架为这些模块提供了一个协作平台，使得它们可以通过主框架获取其它模块的智能指针以互相访问和协作。这种设计使得仿真内核能够灵活地响应各种仿真需求，同时保持各模块间的解耦和独立性。

4.2.6.4.2. 场景描述管理器

场景描述管理器负责解析运行环境配置文件并根据这些配置来设置仿真环境。它定义了一些场景相关的属性及接口，包括：

1. 场景描述管理器初始化接口；

2. 场景名称及其设置与获取接口；

3. 仿真开始时间及其设置与获取接口；

4. 飞机初始值（位置、速度、姿态等）及其设置与获取接口。

场景描述管理器的主要工作流程如下：

1. 执行场景描述管理器的初始化，初始化所有参数；

2. 读取运行环境配置文件中的环境配置信息，执行仿真运行参数设置，例如：设置仿真主框架中各参数、设置时间管理器的仿真开始时间等；

3. 读取 IDL 文件列表中所有 IDL 文件，交给 DDS 管理器创建交互数据主题；

4. 读取服务列表中服务动态库的名称，交给服务管理器动态加载各个服务的动态库，

并调用各个服务的预处理函数，完成服务注册；

5. 服务注册成功后会获得其全局唯一 ID，场景描述管理器根据此 ID 从服务管理器获取服务的只能指针，再读取服务配置文件，根据配置文件设置其参数；

6. 调用服务的初始化接口，服务会针对其需要交互的数据主题，访问 DDS 管理器注册成为参与者（发布者/订阅者）；

7. 读取模型列表中模型分组的频率信息，交给线程管理器创建对应频率的调度线程；

8. 读取模型列表中模型动态库的名称，交给模型管理器动态加载各个模型的动态库，并调用各个模型的预处理函数，完成模型注册；

9. 模型注册成功后会获得其全局唯一 ID，场景描述管理器根据此 ID 从模型管理器获取模型的智能指针，再读取其模型配置文件，根据配置文件设置其参数；

10. 调用模型的初始化接口，模型将向线程管理器注册其周期性执行函数。线程管理器会依据其模型配置文件中的参数将其安排到相应线程的调度表的相应位置，确保该周期性函数能够被线程适时调度。同时，模型会针对其需要交互的数据主题，访问 DDS 管理器注册成为参与者（发布者/订阅者）。

4.2.6.4.3. 线程管理器

线程管理器负责管理所有调度线程，定义了调度线程相关的一些属性和接口，包括：

1. 线程管理器初始化接口；

2. 线程管理器运行前最后准备接口；

3. 所有调度线程的控制接口（包括开始、暂停、继续、终止）；

4. 获取所有调度线程运行状态接口；

5. 获取指定调度线程运行状态接口；

6. 调度线程运行相关参数（包括运行频率、运行间隔、运行开始时间、CPU 亲和性等）及其设置与获取接口；

7. 调度线程管理接口（包括创建调度线程、删除调度线程等）；

8. 向调度线程添加周期性函数接口；

线程管理器的主要工作流程如下：

1. 执行线程管理器初始化，清空线程池，初始化所有参数；

2. 根据场景描述管理器解析的模型分组的参数创建线程，并设置其运行频率、CPU 亲和性、运行开始时间等参数，然后将其加入线程池；

3. 根据场景描述管理器解析的模型分组信息，将模型提交注册的周期性函数提交给对应运行频率的调度线程，由调度线程进一步进行调度安排；

4. 当时间管理器执行运行前最后准备时，会调用线程管理器的运行前最后准备接口。线程管理器会执行所有线程分离等操作；

5. 通过调度线程控制接口控制所有调度线程的开始、暂停、继续、终止。

4.2.6.4.4. 调度线程

调度线程负责调度所有模型提交的周期性执行函数，定义了线程运行相关的一些属性和接口，包括：

1. 调度线程初始化接口；

2. 调度线程控制接口（包括开始、暂停、继续、终止）；

3. 调度线程的加入与脱离接口；

4. 获取调度线程运行状态接口；

5. 调度线程名称及其设置与获取接口；

6. 调度线程运行相关参数（运行频率、运行间隔、运行优先级、运行开始时间、CPU亲和性等）及其设置与获取接口；

7. 向调度线程注册模型周期性函数的接口。

调度线程的主要工作流程如下：

1. 线程调度管理器创建调度线程；

2. 设置调度线程的运行相关参数；

3. 将模型注册的周期性函数根据其运行频率和运行节点填入调度表对应的位置；

4. 执行调度线程初始化；

5. 调度线程按照调度表依次执行模型的周期性函数，调度表的数据结构见 4.1.2.3 小节。调度线程在每个调度周期的任务结束后，计算线程下一周期开始的时间，使用纳秒睡眠技术保证线程能够精确地控制自身睡眠时间，实时地重新唤醒以继续调度任务；

6. 可以通过调度线程控制接口原子性地控制线程的开始、暂停、继续与终止。

4.2.6.4.5. DDS 管理器

DDS 管理器负责对整个软件的 DDS 通信进行管理，定义了 DDS 通信相关的一些属性和接口，包括：

1. DDS 管理器初始化接口；

2. DDS 管理器运行前最后准备接口；

3. Fast DDS 相关参数配置（传输层通信方式、身份验证、访问控制、数据加密等）接口；

4. DDS 主题创建接口；

5. DDS 参与者（订阅者、发布者）注册接口；

6. IDL 文件解析接口；

7. DDS 主题注册接口；

8. DDS 主题数据发布/订阅接口；

9. DDS 通信域 ID 及其设置与获取接口；

DDS 管理器的主要工作流程如下：

1. DDS 管理器初始化；

2. 设置 Fast DDS 通信域 ID，建立通信域；

3. 解析所有 IDL 文件，将模型接口注册为主题；

4. 接收所有 DDS 参与者的注册，并调用 Fast DDS 库注册参与者；

5. 完成其它仿真运行前最后准备工作；

6. 通过 DDS 数据发布/订阅接口进行数据主题的发布/订阅。

4.2.6.4.6. 时间管理器

时间管理器负责管理仿真过程中的时间信息，定义了时间相关的一些属性和接口，包括：

1. 时间管理器初始化接口；

2. 时间管理器运行前最后准备接口；

3. 时间管理器线程运行状态及其获取接口；

4. 时间管理器线程运行控制接口（包括开始、暂停、继续、终止）；

5. 当前仿真时间及其设置与获取接口；

6. 仿真开始时间及其设置与获取接口。

时间管理器的主要工作流程如下：

1. 时间管理器初始化，包括初始化当前仿真时间、仿真开始时间等参数；

2. 创建时间管理器线程；

3. 设置时间管理器线程运行频率为仿真基础运行频率，设置其优先级为最高，设置其 CPU 亲和性；

4. 接受场景描述管理器设置的仿真开始时间，并把当前仿真时间也设置为同样的值；

5. 将仿真开始时间发送给线程管理器，设置其管理的所有线程的开始时间；

6. 完成其它仿真运行前最后准备工作，同时调用线程管理器、模型管理器、服务管理器进行仿真运行前最后准备工作；

7. 仿真开始运行后，时间管理器线程将按照基础运行频率运行，每周期对仿真时间进行递增，同时提交周期性系统事件，如仿真单步执行开始事件、仿真单步执行结束事件等；

8. 通过时间管理器线程控制接口可以控制其开始、暂停、继续、终止，同时调用线程管理器对应的运行控制接口进行调度线程的控制。

4.2.6.4.7. 事件管理器

事件管理器负责管理仿真运行过程中产生的事件及事件处理，定义了一些事件相关的属性和接口，包括：

1. 事件管理器初始化接口；

2. 事件管理器运行前最后准备接口；

3. 事件回调注册接口；

4. 事件回调注销接口；

5. 事件提交接口；

事件管理器的主要工作流程如下：

1. 事件管理器初始化，清空事件回调函数绑定列表；

2. 接收其它组件、模型及服务提交的事件回调注册，为事件标识分配唯一事件 ID，并建立事件和回调函数的绑定列表；

3. 完成其它仿真运行前最后准备工作，开启事件监听；

4. 当有任何组件、模型或服务提交事件时，查询事件回调函数绑定列表，调用绑定的回调函数，进行事件响应。

4.2.6.4.8. 模型管理器

模型管理器负责管理所有利用模型封装框架来封装集成的二进制数据包模型，定义了模型管理相关的属性及接口，包括：

1. 模型管理器初始化接口；

2. 模型管理器运行前最后准备接口；

3. 模型动态库动态加载接口；

4. 模型注册接口；

5. 获取模型 ID 接口；

6. 通过 ID 获取模型智能指针接口。

模型管理器的主要工作流程如下：

1. 模型管理器初始化，清空模型列表；

2. 接收场景描述管理器传来的模型动态库路径，动态加载模型动态库；

3. 调用模型动态库的模型预处理函数，模型会通过模型管理器的模型注册接口注册模型；

4. 模型注册时为模型分配唯一 ID，并将模型的唯一 ID 和智能指针保存在模型列表中；

5. 模型注册完成后向场景描述管理器报告已注册模型的 ID；

6. 当时间管理器执行运行前最后准备时，会调用模型管理器的运行前最后准备接口。模型管理器再调用所有已加载模型的运行前最后准备接口，执行仿真运行前的准备工作。

4.2.6.4.9. 服务管理器

服务管理器负责管理所有利用服务框架来开发的具有特定功能的服务，定义了服务管理相关的属性及接口，包括：

1. 服务管理器初始化接口；

2. 服务管理器运行前最后准备接口；

3. 服务动态库动态加载接口；

4. 服务注册接口；

5. 获取服务 ID 接口；

6. 通过 ID 获取服务智能指针接口。

服务管理器的主要工作流程如下：

1. 服务管理器初始化，清空服务列表；

2. 接收场景描述管理器传来的服务动态库路径，动态加载服务动态库；

3. 调用服务动态库的服务预处理函数，服务会通过服务管理器的服务注册接口注册服务；

4. 服务注册时为服务分配唯一 ID，并将服务的唯一 ID 和智能指针保存在服务列表中；

5. 服务注册完成后向场景描述管理器报告已注册服务的 ID；

6. 当时间管理器执行运行前最后准备时，会调用服务管理器的运行前最后准备接口。

服务管理器再调用所有已加载服务的运行前最后准备接口，执行仿真运行前的准备工作。

4.2.6.4.10. 日志记录

日志记录模块负责记录仿真运行过程中的所有日志信息，定义了日志记录相关的属性及接口，包括：

1. 日志记录等级及其设置与获取接口。仿真运行过程的日志信息最多分为 32 个等级，通过设置日志记录等级，可以筛选将哪些日志信息写入日志文件或输出至终端中；

2. 是否向终端输出日志接口；

3. 写日志接口。日志记录模块利用宏编程提供各类日志写入的宏函数，可以在代码编写时减少代码量，提高编写效率。

在仿真内核的加载和运行过程中，所有生成的日志信息都会输出到两个地方：

1. 系统终端：日志信息会实时输出至系统终端上，为操作人员提供即时的反馈和状态更新，便于监控仿真调度引擎的运行状况。

2. 日志记录文件：这些日志信息也会被记录到一个专门的日志文件中，以便于后续的审查、分析和故障排查。

4.2.6.4.11. 模型封装框架

模型封装框架提供统一的二进制数据包模型封装接口，使二进制数据包模型能够与仿真内核解耦。模型封装框架提供了以下统一的接口：

1. 模型预处理接口：模型动态库动态加载时的被模型管理器调用的入口函数，模型通过此函数建立自身的智能指针，并向模型管理器进行注册；

2. 模型参数设置与获取接口：设置与获取模型的唯一 ID、名称、描述、作者、创建时间、修改时间、版本号、二进制数据包模型动态库路径、模型配置文件路径、仿真主框架智能指针等参数的接口；

3. 模型初始化接口：模型向线程管理器注册周期性函数，动态加载二进制数据包模型动态库；

4. 模型运行前最后准备接口：模型向 DDS 管理器注册成为其交互数据主题的参与者（发布者或订阅者），获取二进制数据包模型的入口函数指针；

5. 模型单步执行接口：即模型的周期性执行函数。模型在单步周期内首先根据订阅的输入数据主题更新模型输入数据，接着执行二进制数据包模型的入口函数，最后将二进制数据包模型的输出数据更新至其发布的输出数据主题中。

4.2.6.4.12. 服务框架

服务框架提供统一的服务开发接口，使服务能够与仿真内核解耦。服务框架提供了以下统一的接口：

1. 服务预处理接口：服务动态库动态加载时的被服务管理器调用的入口函数，服务通过此函数建立自身的智能指针，并向服务管理器进行注册；

2. 服务参数设置与获取接口：设置与获取服务的唯一 ID、名称、描述、作者、创建时间、修改时间、版本号、服务配置文件路径、仿真主框架智能指针等参数的接口；

3. 服务初始化接口：服务向事件管理器注册事件及其回调函数；

4. 服务运行前最后准备接口：服务向 DDS 管理器注册成为其交互数据主题的参与者（发布者或订阅者）；

4.2.7. XNModels

4.2.7.1. 概述

1．标识符：XNModels

2．软件配置项名称：模型系统

3．开发状态：新开发软件配置项

4.2.7.2. 结构设计

模型系统是仿真内核中所有动态加载的模型的集合，这些模型专门用于模拟飞机的各个系统。它们是二进制数据包模型（动态链接库）的形式，并通过一个统一的仿真模型封装框架进行封装和集成。模型系统支持动态加载的所有模型如图 16 所示。

图 16 模型系统中模型组成

其中，单个模型由模型预处理、模型初始化、模型运行前最后准备和模型单步执行四个模块组成，组成结构如图 17 所示。

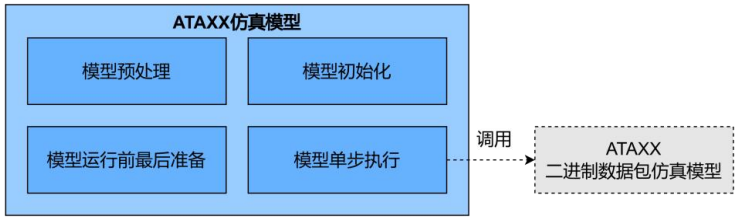

图 17 模型组成结构

4.2.7.3. 工作流程

模型系统单个模型的工作流程如图 18 所示。

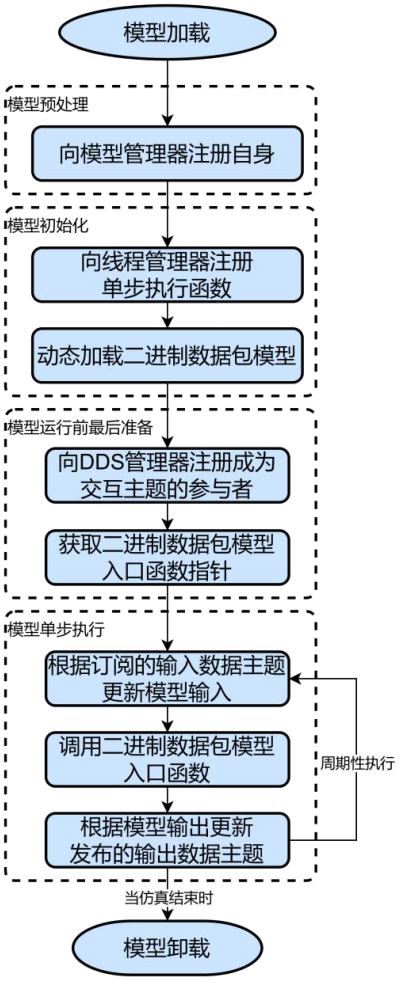

图 18 模型工作流程

4.2.7.4. 模块设计

4.2.7.4.1. 模型预处理

模型预处理是模型动态库在动态加载过程中的关键入口点。在这个模块中，模型首先会

创建一个指向自身的智能指针。随后，它将调用模型管理器提供的模型注册接口，将自身注册到模型管理器中。通过这一过程，模型能够获得其在当前仿真环境中的唯一 ID，确保在整个仿真过程中的准确识别和操作。

4.2.7.4.2. 模型初始化

模型初始化是模型加载完成后必须执行的一系列初始化步骤。在这个模块中，模型首先会解析其模型配置文件，以提取和识别需要注册的周期性函数的相关参数。接着，模型会向线程管理器提交注册这些周期性函数，通常这包括默认的模型单步执行函数。注册完成后，模型将进行下一步操作，即动态加载二进制数据包模型的动态库文件。

4.2.7.4.3. 模型运行前最后准备

在仿真正式启动前，模型需要完成最后的准备工作，以确保其能够顺利参与仿真过程。在这一阶段，模型首先与 DDS 管理器交互，注册成为所需输入数据主题的订阅者以及输出数据主题的发布者，从而确保模型能够接收必要的输入并发送输出数据。

随后，模型将从动态加载的二进制数据包模型的动态库中检索主入口函数的指针。这个指针在模型的单步执行过程中至关重要，因为它指向了模型执行的核心逻辑。

最后，模型将对二进制数据包模型的输入和输出接口数据进行初始化，为仿真的开始做好全面准备。

4.2.7.4.4. 模型单步执行

在仿真启动后，当调度线程调度执行该模型的任务时，将进行模型单步执行。模型的单步执行分为三步进行：

1. 当模型订阅的输入数据主题更新时，模型将会异步接收并保存此次更新的模型输入数据。当模型单步执行开始时，读取保存的模型输入数据；

2. 调用二进制数据包模型的主入口函数，将模型输入数据传输给该函数，进行模型核心逻辑计算，获得模型输出数据；

3. 将模型输出数据更新到模型发布的输出数据主题中。

4.2.8. XNServices

4.2.8.1. 概述

1．标识符：XNServices

2．软件配置项名称：服务系统

3．开发状态：新开发软件配置项

4.2.8.2. 结构设计

服务系统是仿真内核中所有动态加载的服务的集合，这些服务专门用于与外部系统通信、快照管理或完成其它功能。它们通过一个统一的服务开发框架进行开发和集成。服务系统动态加载的部分服务如图 19 所示。

图 19 服务系统中服务组成

其中，单个服务由服务预处理、服务初始化、服务运行前最后准备和事件响应四个模块组成，组成结构如图 20 所示。

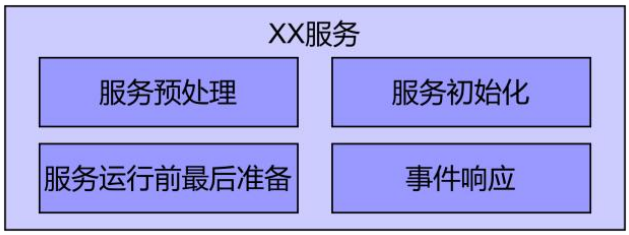

图 20 服务组成结构

4.2.8.3. 工作流程

服务系统中单个服务的工作流程如图 21 所示。

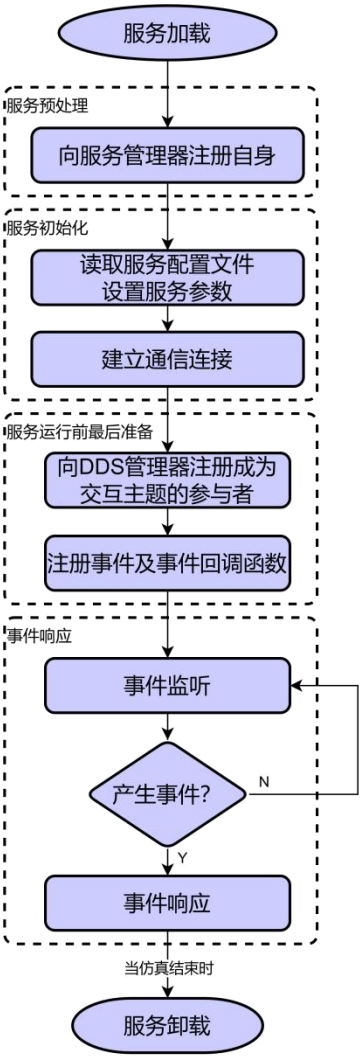

图 21 服务工作流程

4.2.8.4. 模块设计

4.2.8.4.1. 服务预处理

服务预处理是服务动态库在动态加载过程中的关键入口点。在这个模块中，服务首先会创建一个指向自身的智能指针。随后，它将调用服务管理器提供的服务注册接口，将自身注册到服务管理器中。通过这一过程，服务能够获得其在当前仿真环境中的唯一 ID，确保在整个仿真过程中的准确识别和操作。

4.2.8.4.2. 服务初始化

服务初始化是服务加载完成后必须执行的一系列初始化步骤。在这个模块中，服务会解析其服务配置文件，以提取和识别服务具体功能相关的参数并进行设置，例如：

1. 与外部系统通信的服务，需要读取其服务配置文件来设置其传输协议（UDP、TCP、

SHM 等）、通信参数（IP、端口等）、虚拟航空总线协议类型（ARINC 825、ARINC 429、ARINC 664、ARINC 708 等）等参数；

2. 快照服务，需要读取其服务配置文件来设置其对 Fast DDS Record and Replay 工具的调动参数等；

3. 控制指令解析服务，需要读取其服务配置文件来设置其可以识别的指令标识，以及其用于接收指令和发送应答的传输协议（UDP/TCP/SHM）与通信参数（IP、端口）。

在完成参数设置后，服务将根据设置的参数建立与外部系统的通信连接。

4.2.8.4.3. 服务运行前最后准备

在仿真正式启动前，服务需要完成最后的准备工作，以确保其能够顺利参与仿真过程。在这一阶段，服务首先与 DDS 管理器交互，注册成为需要向外部系统输出的数据主题的订阅者以及需要从外部系统输入数据主题的发布者，确保服务能够正确建立外部系统与仿真内核的数据交互连接。

接着，服务需要将所需响应的事件与事件响应对应的回调函数向事件管理器提交注册，以便在事件发生时，能够调用事件响应回调函数进行处理。

4.2.8.4.4. 事件响应

服务需要对内部和外部事件进行监听与响应。服务可能需要对以下外部事件进行响应：

1. 与外部系统通过 UDP/TCP 交互的数据到达或通过共享内存交互的数据更新时，需要对数据进行解包。如果是虚拟航空总线协议数据，还需要对数据进一步解包；

2. 外部系统通过 UDP/TCP 发送的指令到达时，需要对指令进行解包与解析。

服务可能还需要对以下内部事件进行响应：

1. 仿真运行事件：当仿真开始、暂停、继续、终止以及每仿真周期开始时会产生运行事件，服务需要对这些事件进行对应的响应，如周期性向外部系统发送数据等；

2. 快照的拍摄/调用事件：快照服务需要响应事件以写入/读取数据库。

4.2.9. FastDDS

4.2.9.1. 概述

1．标识符：FastDDS

2．软件配置项名称：Fast DDS

3．开发状态：已有的开源库

4．版本：3.1.0

4.2.9.2. 结构

Fast DDS（以前称为 Fast RTPS）是 DDS 规范的高效、高性能实现，DDS 规范是一种用于分布式应用软件的以数据为中心的通信中间件（DCPS）。Fast DDS 的架构如图 22 所示，其中可以看到具有以下几层结构：

1. 应用程序层：利用 Fast DDS API 在分布式系统中实现通信的用户应用程序。

2. Fast DDS 层：DDS 通信中间件的高性能实现。它允许部署一个或多个 DDS 域，其中同一域中的 DomainParticipants 通过在域主题下发布/订阅来交换消息。

3. RTPS 层：实施实时发布-订阅（RTPS）协议，以实现与 DDS 应用程序的互操作性。此层充当传输层的抽象层。

4. 传输层：Fast DDS 可用于各种传输协议，例如不可靠传输协议（UDP）、可靠传输协议（TCP）或共享内存传输协议（SHM）。

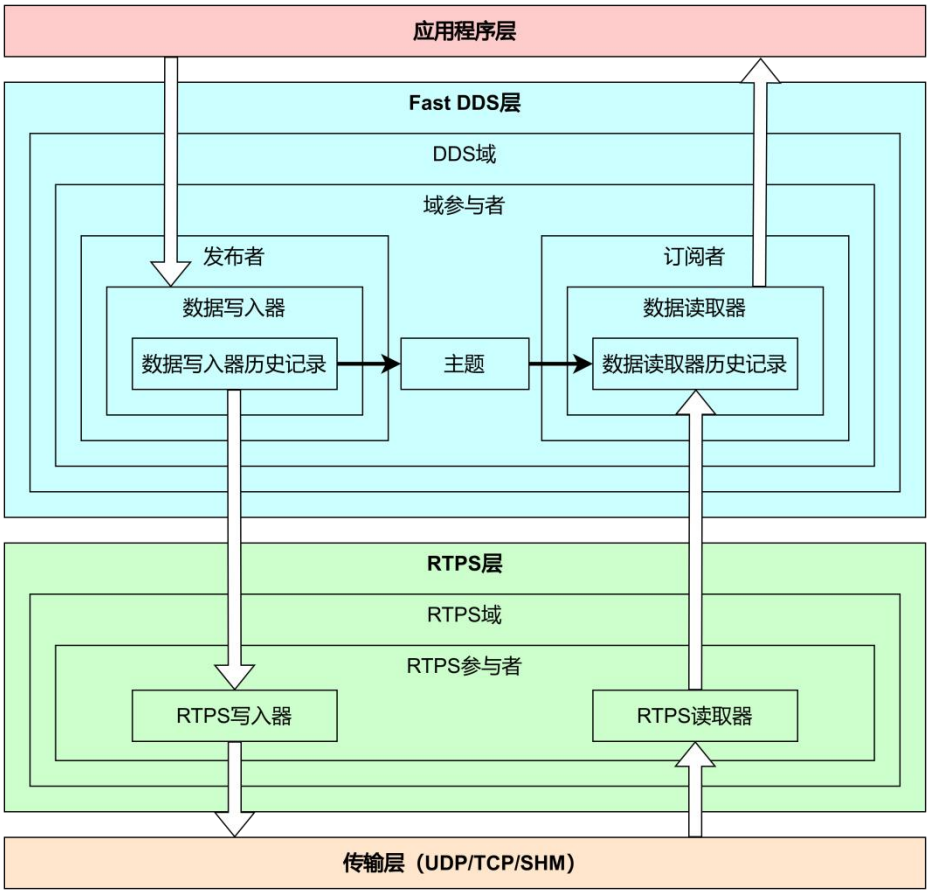

图 22 Fast DDS 架构

4.2.9.3. 功能

4.2.9.3.1. 应用程序层

Fast DDS 的应用程序层即所有使用 Fast DDS API 进行数据交互的所有软件配置项，包括：

1. 仿真调度终端：使用 Fast DDS 发布运行控制指令与仿真调度引擎进行交互；

2. 仿真调度引擎：使用 Fast DDS 订阅运行控制指令与仿真调度终端进行交互；

3. 仿真内核：管理所有通过 Fast DDS 进行发布/订阅的参与者及交互数据；

4. 模型系统：使用 Fast DDS 发布/订阅进行模型间的数据交互；

5. 服务系统：将外部系统的数据通过 Fast DDS 发布/订阅与模型或仿真内核进行交互；

6. 仿真监控终端：使用 Fast DDS 收集模型运行数据并监控仿真运行状态，给用户提供实时的反馈信息；

7. 数据库：拍摄快照时通过 Fast DDS 收集所有数据进行存储，调用快照时读取数据通过 Fast DDS 发布。

4.2.9.3.2. Fast DDS 层

Fast DDS 的 DDS 层定义了几个通信的关键要素。用户将在他们的应用程序中创建这些元素，从而合并 DDS 应用程序元素并创建一个以数据为中心的通信系统。Fast DDS 遵循 DDS 规范，将通信中涉及的这些元素定义为实体。DDS 实体支持服务质量配置（QoS）与监听。DDS 中的实体及其描述和功能：

1. Domain——域：标识 DDS 域的正整数。每个域参与者都分配有一个 DDS 域，以便同一域中的参与者可以通信，以及隔离 DDS 域之间的通信。此值必须由应用程序开发人员在创建参与者时提供；

2. DomainParticipant——域参与者：包含其他 DDS 实体（如发布者、订阅者与主题）的对象；

3. Publisher——发布者：发布者使用数据写入器在指定的主题下发布数据，数据写入器将数据进行传输。一个发布者可能包含多个数据写入器实体；

4. DataWriter——数据写入器：它是负责发布数据的实体。用户在创建此实体时必须提供一个主题，该主题将成为发布数据的主题。发布是通过将数据对象作为更改写入数据写入器历史记录来完成的；

5. DataWriterHistory——数据写入器历史记录：这是对数据对象的更改列表。当数据写

入器继续在指定主题下发布数据时，会在此记录中创建更改。然后，这些更改将发送到订阅该指定主题的数据读取器；

6. Subscriber——订阅者：订阅者使用数据读取器订阅主题，数据读取器从传输中读取数据。一个订阅者可能包含多个数据读取器实体；

7. DataReader——数据读取器：它是订阅主题以接收发布的实体。用户在创建此实体时必须提供订阅主题。数据读取器在其数据读取器历史记录中接收消息作为更改。

8. DataReaderHistory——数据读取器历史记录：它包含数据读取器在订阅某个主题后接收的数据对象中的更改。

9. Topic——主题：将发布者的数据写入器与订阅者的数据读取器绑定的实体。

4.2.9.3.3. RTPS 层

Fast DDS 中的 RTPS 协议允许从传输层抽象 DDS 应用程序实体，有四个主要的实体：

1. RTPSDomain——RTPS 域：它是 DDS 域在 RTPS 协议中的扩展；

2. RTPSParticipant——RTPS 参与者：包含其他 RTPS 实体的实体。它允许配置和创建它所包含的实体；

3. RTPSWriter——RTPS 写入器：消息的来源。它读取数据写入器历史记录中的更改，并将这些更改传输到之前匹配的所有 RTPS 读取器；

4. RTPSReader——RTPS 读取器：消息的接收实体。它将 RTPS 写入器报告的更改写入数据读取器历史记录。

4.2.9.3.4. 传输层

Fast DDS 支持通过各种传输协议进行数据交互的应用程序，例如 UDPv4、UDPv6、TCPv4、TCPv6 和共享内存传输 （SHM）。默认情况下，域参与者使用 UDPv4 和 SHM 传输协议。

4.2.10.XNMACP

4.2.10.1. 概述

1．标识符：XNMACP

2．软件配置项名称：数据库

3．开发状态：已有的开源容器文件

4．版本：无

4.2.10.2. 结构

MCAP（Modular Container and Archive Protocol）是一种开源的容器文件格式，主要用于存储和记录多模态数据。

MCAP 文件的结构如下：

1. Magic：文件必须以特定的魔法字节开始和结束，这些字节用于标识文件格式和版本；

2. Header：文件的头部记录，包含文件的基本信息；

3. Data Section：数据部分，包含多个记录，每个记录包含一个或多个通道的数据；

4. Summary Section：可选的摘要部分，包含文件的索引和摘要数据，用于快速查找和读取；

5. Summary Offset Section：可选的摘要偏移部分，用于快速定位摘要部分；

6. Footer：文件的尾部记录，包含文件的结束信息；

4.2.10.3. 功能

MCAP 数据文件具有以下功能：

1. 多模态支持：MCAP 文件可以存储多种类型的传感器数据，如摄像头、激光雷达、IMU 等，提供统一的数据接口；

2. 高效且可扩展：二进制数据格式——MCAP 使用二进制数据块，确保高效的存储和传输。动态通道设计——支持在运行时增加或减少通道，增加了其适应性；

3. 格式兼容性：MCAP 支持多种序列化格式，如 Protobuf、DDS (CDR)、ROS、JSON 等，可以记录和回放二进制消息；

4. 高性能写入——MCAP 采用追加结构，数据可以流式传输到磁盘或网络，无需回溯，减少了磁盘 I/O 操作，降低了数据损坏的风险；

5. 自包含——MCAP 文件包含消息模式，即使代码库 evolves，文件仍然可读；

6. 高效读取和定位——MCAP 文件包含可选的索引，支持快速、高效的读取，即使在低带宽的网络连接上也能高效工作；

7. 可选压缩——MCAP 支持 LZ4 或 Zstandard 压缩，即使在压缩的情况下，也能支持高效的索引读取；

8. 广泛的语言支持——MCAP 提供了 C++、Go、Python、Rust、Swift 和 TypeScript 的原生读写库；

9. 灵活配置——MCAP 支持配置可选功能，如分块、索引、CRC 校验和和压缩，以满

足不同应用的需求。

4.2.11. XNMonitor

4.2.11.1. 概述

1．标识符：XNMonitor

2．软件配置项名称：仿真监控终端

3．开发状态：新开发软件配置项

4.2.11.2. 结构设计

仿真监控终端的结构如图 23 所示，包含软件运行状态监控、模型运行状态监控、模型数据监控、模型数据注入、模型数据采集和数据图表绘制六个功能模块。

图 23 仿真监控终端结构图

4.2.11.3. 工作流程

仿真监控终端的工作流程如图 24 所示。

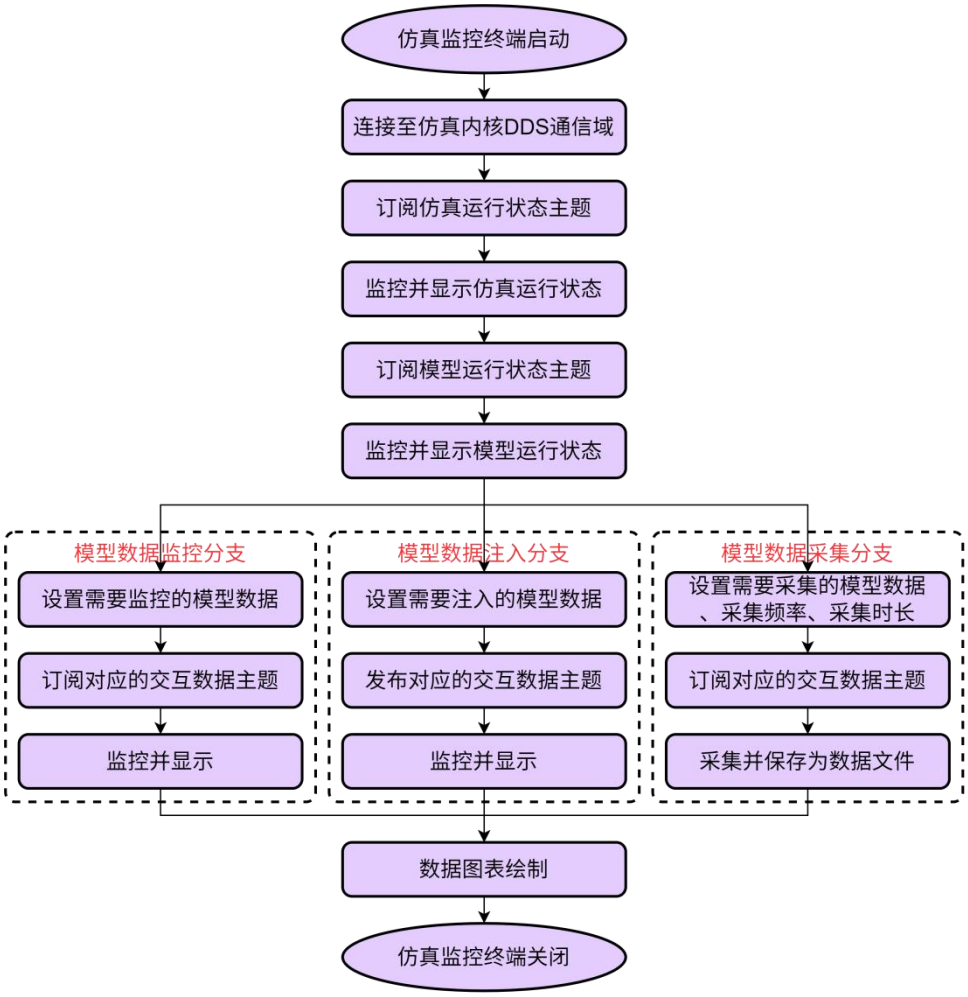

图 24 仿真监控终端工作流程

4.2.11.4. 模块设计

4.2.11.4.1. 仿真运行状态监控

在仿真运行状态监控模块中，仿真监控终端通过订阅仿真内核运行过程中发布的仿真运行状态主题，实时更新界面中显示的仿真运行状态信息。仿真运行状态监控模块可以监控的内容包括：

1. 仿真引擎进程名称、进程 ID；

2. 仿真内核各模块状态（正常、异常）；

3. 仿真运行相关参数：运行状态（未开始、运行中、暂停、运行终止）、设定运行频率、实时运行频率、最小运行频率、最大运行频率、平均运行频率、设定运行周期、实时运行周期、最小运行周期、最大运行周期、平均运行周期、已运行周期数、CPU 核心编号等；

4. 各仿真调度线程相关参数：线程名称、线程 ID、运行状态（未开始、运行中、暂停、

运行终止）、优先级、设定运行频率、实时运行频率、最小运行频率、最大运行频率、平均运行频率、设定运行周期、实时运行周期、最小运行周期、最大运行周期、平均运行周期、已运行周期数、CPU 核心编号等；

5. 选中某个线程可以在调度线程实时监控图中监控其运行周期或运行频率，通过在图表上唤出右键菜单可以选择图表绘制的数据内容。

4.2.11.4.2. 模型运行状态监控

在模型运行状态监控模块中，仿真监控终端通过订阅仿真调度线程及模型运行过程中发布的运行状态主题，实时更新界面中显示的调度线程及模型运行状态信息。模型运行状态监控模块可以监控的内容包括：

1. 各仿真调度线程相关参数：线程名称、线程 ID、运行状态（未开始、运行中、暂停、运行终止）、优先级、设定运行频率、实时运行频率、最小运行频率、最大运行频率、平均运行频率、设定运行周期、实时运行周期、最小运行周期、最大运行周期、平均运行周期、已运行周期数、CPU 核心编号等；

2. 各模型相关参数：模型名称、模型 ID、运行状态（未开始、运行中、暂停、运行终止）、优先级、设定运行频率、实时运行频率、最小运行频率、最大运行频率、平均运行频率、设定运行周期、实时运行周期、最小运行周期、最大运行周期、平均运行周期、已运行周期数等；

3. 选中某个模型可以在模型实时监控图中监控其运行周期或运行频率，通过在图表上唤出右键菜单可以选择图表绘制的数据内容。

4.2.11.4.3. 模型数据监控

在模型数据监控模块中，仿真监控终端支持用户自定义需要监控的模型交互数据，并支持以图表显示监控的数据。模型数据监控模块具有以下功能：

1. 打开模型接口描述 IDL 文件，可以在界面左侧展示模型的所有接口。通过点击接口右侧的“监控”按钮，可以直接监控该接口数据。当不需要监控该模型交互数据时，可以点击模型名称右侧的“关闭”按钮，关闭模型接口描述 IDL 文件；

2. 接口信息列表显示正在监控的接口数据，用户可以在表格的接口名称一列中输入需要监控的接口名称，软件将尝试订阅该接口数据进行监控。

3. 接口信息列表显示监控接口数据的数据名称、数据类型、数据值、单位、监控状态等，同时还支持用户控制监控的开始、暂停、继续、绘图及监控项的删除操作；

4. 点击接口信息列表中某个接口的“绘图”按钮，数据实时监控图中将绘制数据的实时数据。

4.2.11.4.4. 模型数据注入

在模型数据注入模块中，仿真监控终端会根据用户选择的监控数据及注入值，发布对应的数据主题。

4.2.11.4.5. 模型数据采集

在模型数据采集模块中，仿真监控终端支持用户自定义需要采集的模型交互数据及采集参数，采集完成后将采集到的数据保存在数据文件中。模型数据采集模块还支持打开采集的数据文件进行数据预览。模型数据采集模块具有以下功能：

1. 可以自由添加、删除需要采集的数据；
2. 可以设置采集频率、采集时长、保存路径等采集参数；
3. 可以控制数据采集的开始和临时终止；
4. 可以打开采集的数据文件进行预览，并以图表形式展示采集的数据。

4.2.11.4.6. 数据图表绘制

数据图表绘制模块负责绘制仿真运行监控、模型运行监控、模型数据监控和模型数据采集四个模块中的所有图表。它具有以下功能：

1. 能够同时绘制多条曲线，并以不同颜色进行显示；
2. 支持图像的放大、缩小、调整坐标范围、标记等操作；
3. 将绘制的图像保存为多种图片格式；
4. 用于实时显示时，存储一定长度的历史数据。

4.2.12.XNIDE

4.2.12.1. 概述

1．标识符：XNIDE

2．软件配置项名称：集成开发环境

3．开发状态：包含新开发的工具软件与已有的开源软件集合

4．版本号：集成开发环境包含的已有的开源软件及版本号见 4.1.1.2.2 节介绍的软件环境。

4.2.12.2. 结构

集成开发环境包含以下新开发工具：

1. IDL 文件生成工具：将模型接口 ICD 文件转换为 IDL 文件的工具；

2. 代码构建工具：模型封装及服务开发的模板源代码生成工具。

集成开发环境包含以下已有的开源软件：

1. Fast DDS Gen：Java 应用程序，用于生成与 Fast DDS 库兼容的 TypeSupport 代码；

2. VS Code：由微软开发的免费、开源的代码编辑器，支持多种编程语言和框架，是模型封装及服务开发使用的代码编辑器；

3. CMake：跨平台的自动化构建系统，用于管理和构建软件项目；

4. GCC/G++：GCC（GNU Compiler Collection）和 G++是两个紧密相关的编译器工具，广泛用于编译 C 和 C++程序，是模型封装及服务开发使用的代码编译器；

5. Fast DDS、Fast CDR、Qt 等编译依赖库。

4.2.12.3. 功能

使用集成开发环境进行二进制数据包模型封装与集成时具有以下功能：

1. 开发人员能够使用 IDL 文件生成工具读取模型接口 ICD 文件，自动根据 ICD 文件生成 IDL 文件；

2. 开发人员能够使用 Fast DDS Gen 工具读取 IDL 文件，自动根据 IDL 文件生成 DDS 交互主题定义源代码；

3. 开发人员能够使用代码构建工具建立模型封装工程，选择模型数据交互对应的 DDS 交互主题定义源代码，自动生成模型封装的模板源代码与模型配置文件模板；

4. 开发人员能够使用 VS Code 代码编辑器打开模型封装工程，编写模型封装的核心代码（如输入/输出接口对接、模型入口函数调用等）；

5. 开发人员能够使用 CMake 为模型封装工程生成标准的构建文件；

6. 开发人员能够使用 GCC/G++代码编译器进行源代码编译，生成模型动态链接库；

7. 依赖库将在源代码编译过程中提供支持；

8. 将编译得到的模型动态链接库、编辑后的模型配置文件以及二进制数据包模型一起放置在统一的模型库目录中，即可供仿真内核在仿真运行时动态加载。

使用集成开发环境进行服务开发时具有以下功能：

1. 开发人员能够使用 Fast DDS Gen 工具读取服务交互数据的 IDL 文件，自动根据 IDL

文件生成 DDS 交互主题定义源代码；

2. 开发人员能够使用代码构建工具建立服务开发工程，选择服务数据交互对应的 DDS 交互主题定义源代码，将根据仿真内核提供的服务开发框架自动生成服务的模板源代码与服务配置文件模板；

3. 开发人员能够使用 VS Code 代码编辑器打开服务开发工程，编写服务的核心代码（如仿真事件响应、外部系统数据交互、指令解析等）；

4. 开发人员能够使用 VS Code 代码编辑器调用 GCC/G++代码编译器进行源代码编译，生成服务动态链接库；

5. 依赖库将在源代码编译过程中提供支持；

6. 将编译得到的服务动态链接库、编辑后的服务配置文件一起放置在统一的服务库目录中，即可供仿真内核在仿真运行时动态加载。

4.3. 执行概念

本节将从用例图、运行时序图、状态转换图三个方面来描述“玄鸟”架构各软件配置项之间的动态关系。通过这些图形化工具，将清晰地展示各软件配置项在仿真运行期间的交互方式、执行顺序以及状态变化，从而为理解和分析“玄鸟”架构的动态行为提供直观且详细的视角。

4.3.1. 用例图

“玄鸟”架构的用例图如图 25 所示，描述了“玄鸟”架构各软件配置项与外部参与者之间的交互关系。通过用例图可以直观地看到各个参与者如何与“玄鸟”架构中的不同功能模块进行交互，以及这些模块如何协同工作以满足外部参与者的需求。图中清晰地展示了每个用例的具体功能和参与者之间的关联，为理解“玄鸟”架构的业务流程和功能需求提供了重要的参考依据。

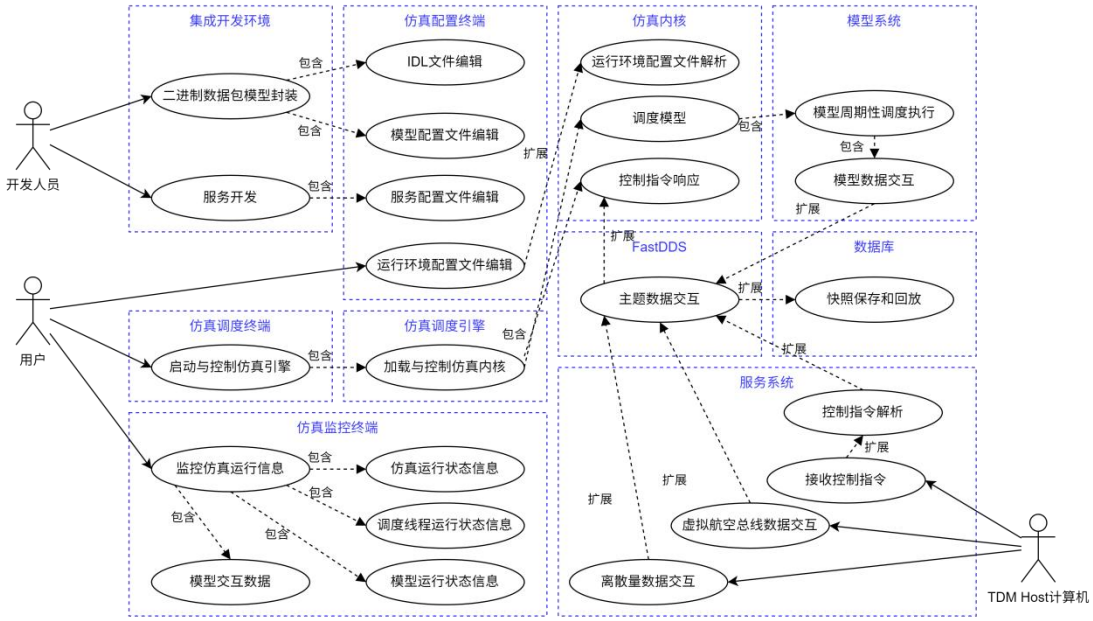

图 25 “玄鸟”架构用例图

4.3.2. 运行时序图

“玄鸟”架构的运行时序图如图 26 所示，描述了“玄鸟”架构各相关软件配置项运行时的时序逻辑。该图通过精确的时间序列，展示了不同软件配置项之间的交互顺序和时间关系，揭示了“玄鸟”架构在运行过程中的动态行为和协同工作模式。通过这一时序图，开发人员和用户能够清晰地理解各软件配置项的执行流程，从而能够更好地优化“玄鸟”架构的性能和确保“玄鸟”架构的稳定运行。

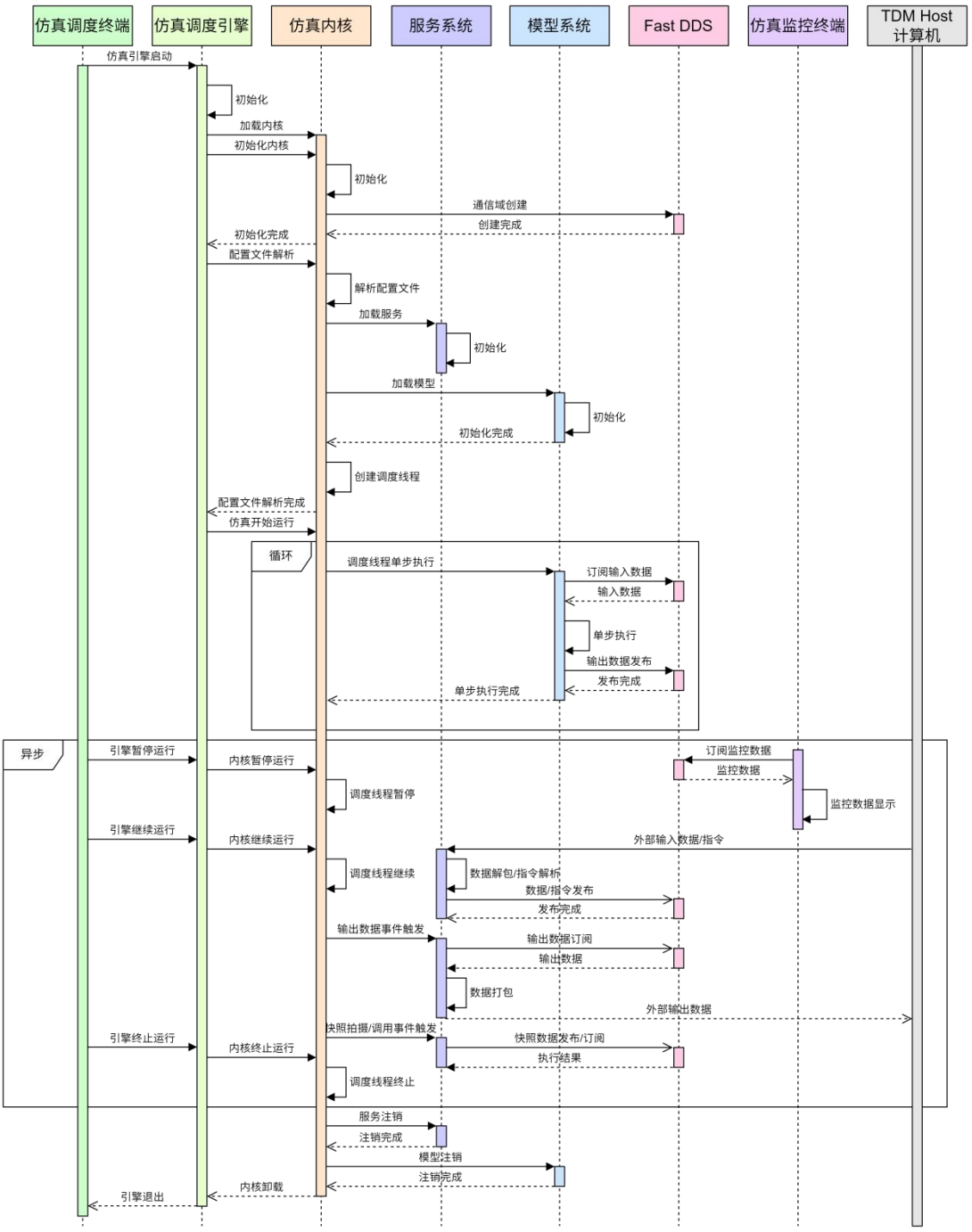

图 26 “玄鸟”架构运行时序图

4.3.3. 状态转换图

“玄鸟”架构在仿真运行阶段的状态转换图如图 27 所示，描述了“玄鸟”架构在仿真运行阶段相关软件配置项的状态转换。该图通过展示各软件配置项在不同事件触发下的状态变化，帮助开发人员和用户理解“玄鸟”架构在仿真过程中的动态行为。

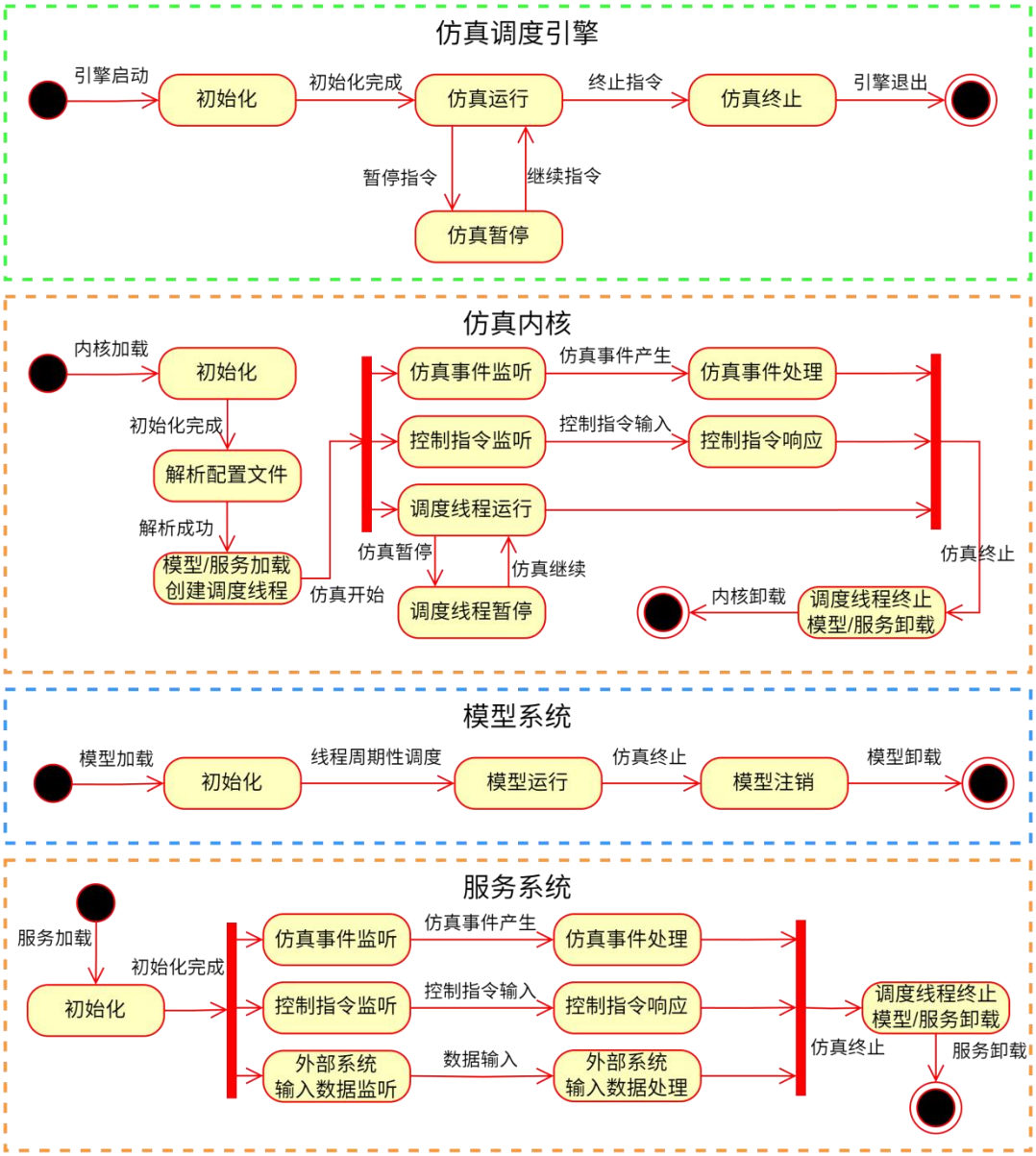

图 27 “玄鸟”架构在仿真运行阶段的状态转换图

4.4. 接口设计

4.4.1. 接口标识和图表

本节用项目唯一标识符列出了“玄鸟”架构的每个内外部接口，并列出接口的名称、编号、版本及相关实体，其中内部接口见表 3，外部接口见表 4。

表 3 “玄鸟”架构内部接口标识表

序号	接口名称	标识符	版本	相关实体
1.	运行环境配置文件接口	XNScenarioFileInterface	V1.0	仿真配置终端 仿真调度终端

				仿真调度引擎 仿真内核
2.	模型配置文件接口	XNModelConfigFileInterface	V1.0	仿真配置终端 仿真内核 模型系统
3.	服务配置文件接口	XNServiceConfigFileInterface	V1.0	仿真配置终端 仿真内核 服务系统
4.	接口描述文件接口	XNIDLFileInterface	V1.0	仿真配置终端 仿真内核 模型系统 服务系统 仿真监控终端
5.	仿真运行控制接口	XNRuntimeControlInterface	V1.0	仿真调度终端 仿真调度引擎 仿真内核 FastDDS
6.	仿真运行状态接口	XNRuntimeStatusInterface	V1.0	仿真调度终端 仿真调度引擎 仿真内核 FastDDS 仿真监控终端
7.	调度线程运行状态接口	XNThreadStatusInterface	V1.0	仿真调度引擎 仿真内核 FastDDS 仿真监控终端
8.	模型运行状态接口	XNModelStatusInterface	V1.0	仿真调度引擎 仿真内核 FastDDS 仿真监控终端
9.	模型 DDS 数据交互接口	XNDDSDataInterface	V1.0	仿真内核 模型系统 服务系统 FastDDS 仿真监控终端
10.	快照存储与读取接口	XNSnapshotInterface	V1.0	服务系统 FastDDS 数据库

表 4 “玄鸟”架构外部接口标识表

序号	接口名称	标识符	版本	相关实体
1.	仿真控制指令接口	XNCommandInterface	V1.0	仿真内核 服务系统

				TDM Host 计算机
2.	ARINC 429 虚拟航空总线通信接口	XNARINC429Interface	V1.0	服务系统 TDM Host 计算机
3.	ARINC 664 虚拟航空总线通信接口	XNARINC664Interface	V1.0	服务系统 TDM Host 计算机
4.	ARINC 708 虚拟航空总线通信接口	XNARINC708Interface	V1.0	服务系统 TDM Host 计算机
5.	ARINC 825 虚拟航空总线通信接口	XNARINC825Interface	V1.0	服务系统 TDM Host 计算机
6.	离散量通信接口	XNDiscreteDataInterface	V1.0	服务系统 TDM Host 计算机
7.	仿真配置终端交互界面接口	XNEditorInterface	V1.0	仿真配置终端用户
8.	仿真调度终端交互界面接口	XNRunnerInterface	V1.0	仿真调度终端用户
9.	仿真监控终端交互界面接口	XNMonitorInterface	V1.0	仿真监控终端用户

4.4.2. XNScenarioFileInterface

4.4.2.1. 概述

1. 接口名称：运行环境配置文件接口
2. 接口类型：数据存储与检索接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNScenarioConfigFile
        b. 非技术（自然语言）名称：运行环境配置文件；
        c. 缩写：XNSceCfg
    2) 数据元素类型：XML 格式的配置文件；
4. 通信方法：XML 文件读写；
5. 通信协议：XML 语法规范。

4.4.2.2. 数据元素定义

运行环境配置文件（XNScenarioConfigFile）是以 XML 编写的用以存储一体化二进制数据包软件运行环境及运行配置相关参数的配置文件，它包含以下内容：

1. 运行环境参数（XNEnvironment）：

1） 操作系统名称及版本（XNOSVersion）：仿真运行所在计算机的操作系统名称及版本，例如：Windows 11 或 Debian 11；

2） Linux 内核版本/WindowRTX 版本（XNCoreVersion）：仿真运行所在计算机的实体操作系统内核/补丁版本，例如：RTX64（Windows）或 5.10.0-32-rt-amd64（Linux）；

3） CPU 亲和性（XNCPUAff）：限定仿真运行在 CPU 的哪些核心上；

4） 基础运行频率（XNBaseFreq）：限定仿真调度周期的最高运行频率；

5） 工作目录（XNWorkPath）：绝对路径，仿真运行的工作根目录路径；

6） 模型库目录（XNModelsPath）：相对于工作目录的相对路径，是存放模型系统所有模型动态链接库的路径；

7） 服务库目录（XNServicePath）：相对于工作目录的相对路径，存放服务系统所有服务动态链接库的路径；

8） 终端输出控制（XNTerOutput）：控制仿真运行过程中是否向控制台终端打印各个等级的日志消息；

9） 日志输出控制（XNLogOutput）：控制仿真运行过程中是否向日志文件中记录各个等级的日志消息；

2. 模型列表（XNModelsList）列出本次仿真需要动态加载的所有模型信息，每个模型的信息包括：

1） 模型名称（XNModelName）；

2） 版本号（XNModelVersion）；

3） 模型动态库名称（XNModelLibName）：封装后模型动态链接库的名称，不包括前缀“lib”和后缀名“.so”，封装后模型动态链接库必须位于模型库目录下；

4） 二进制数据包模型动态链接库路径（XNModelLibPath）：相对于模型库目录的相对路径，是存放该模型对应的二进制数据包模型动态链接库所在的路径。

3. 服务列表（XNServicesList）列出本次仿真需要动态加载的所有服务信息，每个服务的信息包括：

1） 服务名称（XNServiceName）；

2） 版本号（XNServiceVersion）；

3） 服务动态库名称（XNServiceLibName）：服务动态链接库的名称，不包括前缀“lib”和后缀名“.so”，服务动态链接库必须位于服务库目录下。

4.4.3. XNModelConfigFileInterface

4.4.3.1. 概述

1. 接口名称：模型配置文件接口
2. 接口类型：数据存储与检索接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNModelConfigFile
        b. 非技术（自然语言）名称：模型配置文件；
        c. 缩写：XNMdlCfg
    2) 数据元素类型：XML 格式的配置文件；
4. 通信方法：XML 文件读写；
5. 通信协议：XML 语法规范。

4.4.3.2. 数据元素定义

模型配置文件（XNModelConfigFile）是以 XML 编写的用以存储封装后的二进制数据包模型的参数的配置文件，需要与封装后模型动态链接库在同一目录下，它包含以下内容：

1. 模型基本信息（XNModelInfo）：

1） 模型名称（XNModelName）：需要与运行环境配置文件中配置的对应模型名称保持一致；

2） 版本号（XNModelVersion）：需要与运行环境配置文件中配置的对应版本的模型版本号保持一致；

3） 作者（XNModelAuthor）；

4） 创建时间（XNModelBldTime）：该模型第一个版本封装的时间；

5） 修改时间（XNModelMdfTime）：该版本模型的封装时间。

2. 函数列表（XNModelFunList）列出该模型需要被周期性调度的所有函数的信息，每个函数的信息包括：

1） 函数名称（XNModelFunName）：与模型封装代码中的函数名称保持一致；

2） 函数调度节点（XNModelFunNode）：以“x-y”的形式填写，其中：

a. “x”为函数被调度运行的频率分组号：“0”表示以基础频率进行调度，

“1”表示以基础频率的一半进行调度，依次类推，最大为“5”；

b. “y”为函数被调度运行的节点号，举例说明：当“x”为 1 时，函数以基础频率的一半进行调度，因此过两个调度周期才调度执行一次该函数。“y”决定该函数在两个调度周期的前一个周期（“y”为 0）还是后一个周期（“y”为 1）进行调度。因此“y”的值不大于“$x^2 - 1$”。

3） 函数运行优先级（XNModelFunPro）：决定函数调度运行的优先级顺序，最高为“99”，最低为“0”。

3. 指令列表（XNModelCmdList）列出了该模型可以响应的所有指令信息，每个指令的信息包括：

1） 指令标识（XNModelCmdName）：该条指令的唯一标识符，必须为形如“模型名_指令名”的格式；

2） 指令含义（XNModelCmdDscr）：解释该条指令的含义；

3） 响应函数（XNModelCmdCall）：响应该条指令的函数名称。

4.4.4. XNServiceConfigFileInterface

4.4.4.1. 概述

1. 接口名称：服务配置文件接口
2. 接口类型：数据存储与检索接口
3. 接口数据元素：
   1) 数据元素名称：
      a. 项目唯一标识符：XNServiceConfigFile
      b. 非技术（自然语言）名称：服务配置文件；
      c. 缩写：XNSrvCfg
   2) 数据元素类型：XML 格式的配置文件；
4. 通信方法：XML 文件读写；
5. 通信协议：XML 语法规范。

4.4.4.2. 数据元素定义

服务配置文件（XNServiceConfigFile）是以 XML 编写的用以存储服务参数的配置文件，需要与服务动态链接库在同一目录下，它包含以下内容：

1. 服务基本信息（XNServiceInfo）：

1） 服务名称（XNServiceName）：需要与运行环境配置文件中配置的对应服务名称保持一致；

2） 版本号（XNServiceVersion）：需要与运行环境配置文件中配置的对应版本的服务版本号保持一致；

3） 作者（XNServiceAuthor）；

4） 创建时间（XNServiceBldTime）：该服务第一个版本完成开发的时间；

5） 修改时间（XNServiceMdfTime）：该版本服务完成开发的时间。

2. 指令列表（XNServiceCmdList）列出了该模型可以响应的所有指令信息，每个指令的信息包括：

1） 指令标识（XNServiceCmdName）；

2） 指令含义（XNServiceCmdDscr）；

3） 响应函数（XNServiceCmdCall）：响应该条指令的函数名称。

3. 自定义参数（XNServiceParam）中列出了服务自身功能相关的一些参数，包括但不限于以下参数：

1） UDP/IP 通信相关参数：IP 地址、端口号等参数；

2） TCP/IP 通信相关参数：IP 地址、端口号等参数；

3） 虚拟航空总线通信相关参数：航空总线协议类型等参数；

4） 指令解析相关参数：仿真控制指令列表等参数；

5） 数据库相关参数：数据库文件路径等参数。

4.4.5. XNIDLFileInterface

4.4.5.1. 概述

1. 接口名称：接口描述文件接口
2. 接口类型：数据存储与检索接口
3. 接口数据元素：
   1) 数据元素名称：
      a. 项目唯一标识符：XNIDLFile
      b. 非技术（自然语言）名称：接口描述文件；
      c. 缩写：XNIDL

2) 数据元素类型：IDL 格式的接口描述文件；

4. 通信方法：IDL 文件读写；

5. 通信协议：IDL 语法规范。

4.4.5.2. 数据元素定义

接口描述文件（XNIDLFile）是以 IDL 编写的用于描述二进制数据包模型间接口的文件。IDL（Interface Definition Language，接口定义语言）是一个用于描述软件组件接口的抽象语言。它不依赖于任何特定的编程语言，允许开发者定义数据类型、接口、方法和其他软件组件的规格。这些定义随后可以被转换成多种编程语言的代码，从而实现跨语言的通信和集成。

接口描述文件包含以下内容：

1. 命名空间（NameSpace）：使用命名空间将相关的接口组织在一起，同时避免不同接口间的命名冲突；

2. 结构体（Struct）：结构体用于将一组相关的数据成员封装在一起，形成一个逻辑上的整体，例如将二进制数据包模型的输入接口与输出接口放置在两个不同的结构体中；

3. 成员变量（MemVar）：成员变量用于存储结构体中的数据，这些数据可以是基本类型、数组、字符串、其他结构体等，例如二进制数据包模型的一个输入/输出接口。

4.4.6. XNRuntimeControlInterface

4.4.6.1. 概述

1. 接口名称：仿真运行控制接口

2. 接口类型：实时数据传送接口

3. 接口数据元素：

   1) 数据元素名称：

      a. 项目唯一标识符：XNRuntimeControl

      b. 非技术（自然语言）名称：仿真运行控制指令；

      c. 缩写：XNRtCmd

   2) 数据元素类型：结构体；

4. 通信方法：Fast DDS 通信；

5. 通信协议：DDS 通信协议。

4.4.6.2. 数据元素定义

仿真运行控制指令（XNRuntimeControl）用于控制仿真调度引擎的运行，可以由仿真调度终端或其它控制终端通过 Fast DDS 发送。仿真运行控制指令结构体组成内容如表 5 所示。

表 5 仿真运行控制指令结构体

序号	元素名称	元素标识	数据类型	元素含义
1.	仿真控制指令	XNSimCmd	INT32	控制仿真运行：0-无指令，1-暂停，2-继续，3-结束
2.	调度线程控制指令	XNThrCmd	INT32	从低位到高位，每两个二进制位控制一个线程的运行：00-无指令，01-暂停，10-继续，11-结束

4.4.7. XNRuntimeStatusInterface

4.4.7.1. 概述

1. 接口名称：仿真运行状态接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
   1) 数据元素名称：
      a. 项目唯一标识符：XNRuntimeStatus
      b. 非技术（自然语言）名称：仿真运行状态；
      c. 缩写：XNRtSta
   2) 数据元素类型：结构体；
4. 通信方法：Fast DDS 通信；
5. 通信协议：DDS 通信协议。

4.4.7.2. 数据元素定义

仿真运行状态（XNRuntimeStatus）用于向仿真调度引擎和仿真监控终端报告当前仿真运行的状态及参数。仿真运行状态结构体组成内容如表 6 所示。

表 6 仿真运行状态结构体

序号	元素名称	元素标识	数据类型	元素含义
1.	仿真调度引擎进程名称	XNEngineName	string	/
2.	仿真调度引擎	XNEngineID	INT32	/

	进程 ID			
3.	仿真调度引擎运行状态	XNEngineSt	INT32	0-未开始，1-运行中，2-暂停，3-终止，其它-未知状态
4.	仿真调度引擎 CPU 亲和性	XNEngineAff	INT32	CPU 亲和性掩码，每一位都表示一个 CPU 核心，置 1 表示“绑定”。最低位表示第一个逻辑 CPU，最高位表示最后一个逻辑 CPU。
5.	仿真内核加载状态	XNCoreSt	struct	结构体定义见表 7
6.	仿真运行信息	XNRuntimeSt	struct	结构体定义见表 8

表 7 仿真内核加载状态结构体

序号	元素名称	元素标识	数据类型	元素含义
1.	主框架加载状态	XNFWSatus	INT32	0-未加载，1-正常，其它-异常
2.	时间管理器加载状态	XNTMStatus	INT32	0-未加载，1-正常，其它-异常
3.	事件管理器加载状态	XNEMStatus	INT32	0-未加载，1-正常，其它-异常
4.	场景描述管理器加载状态	XNSDStatus	INT32	0-未加载，1-正常，其它-异常
5.	线程管理器加载状态	XNThMStatus	INT32	0-未加载，1-正常，其它-异常
6.	模型管理器加载状态	XNMMStatus	INT32	0-未加载，1-正常，其它-异常
7.	服务管理器加载状态	XNSMStatus	INT32	0-未加载，1-正常，其它-异常
8.	DDS 管理器加载状态	XNDMStatus	INT32	0-未加载，1-正常，其它-异常

表 8 仿真运行信息结构体

序号	元素名称	元素标识	数据类型	元素含义
1.	线程数	XNThCnt	INT32	调度线程总数
2.	运行周期数	XNRunCnt	INT32	仿真调度引擎运行周期数目
3.	实时频率	XNCurFreq	DOUBLE	仿真调度引擎当前实际运行频率，单位 Hz
4.	设定频率	XNSetFreq	DOUBLE	仿真调度引擎设定的运行频率，单位 Hz

4.4.8. XNThreadStatusInterface

4.4.8.1. 概述

1. 接口名称：调度线程运行状态接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNThreadStatus
        b. 非技术（自然语言）名称：调度线程运行状态；
        c. 缩写：XNThSta
    2) 数据元素类型：结构体；
4. 通信方法：Fast DDS 通信；
5. 通信协议：DDS 通信协议。

4.4.8.2. 数据元素定义

调度线程运行状态（XNThreadStatus）用于向仿真监控终端报告当前调度线程运行的状态及参数。调度线程运行状态结构体组成内容如表 9 所示。

表 9 调度线程运行状态结构体

序号	元素名称	元素标识	数据类型	元素含义
1.	调度线程名称	XNThreadName	string	/
2.	调度线程 ID	XNThreadID	INT32	/
3.	调度线程运行状态	XNThreadSt	INT32	0-未开始，1-运行中，2-暂停，3-终止，其它-未知状态
4.	调度线程 CPU 亲和性	XNThreadAff	INT32	CPU 亲和性掩码，每一位都表示一个 CPU 核心，置 1 表示“绑定”。最低位表示第一个逻辑 CPU，最高位表示最后一个逻辑 CPU。
5.	调度线程优先级	XNThreadPro	INT32	最高 99，最低 0
6.	调度线程运行周期数	XNThRunCnt	INT32	调度线程运行周期数目
7.	调度线程实时频率	XNThCurFreq	DOUBLE	仿真调度线程当前实际运行频率，单位 Hz
8.	调度线程设定频率	XNThSetFreq	DOUBLE	仿真调度线程设定的运行频率，单位 Hz

4.4.9. XNModelStatusInterface

4.4.9.1. 概述

1. 接口名称：模型运行状态接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNModelStatus
        b. 非技术（自然语言）名称：模型运行状态；
        c. 缩写：XNMdlSta
    2) 数据元素类型：结构体；
4. 通信方法：Fast DDS 通信；
5. 通信协议：DDS 通信协议。

4.4.9.2. 数据元素定义

模型运行状态（XNModelStatus）用于向仿真监控终端报告当前模型运行的状态及参数。模型运行状态结构体组成内容如表 10 所示。

表 10 模型运行状态结构体

序号	元素名称	元素标识	数据类型	元素含义
1.	模型名称	XNModelName	string	/
2.	模型 ID	XNModelID	INT32	/
3.	模型运行状态	XNModelSt	INT32	0-未开始，1-运行中，2-暂停，3-终止，其它-未知状态
4.	模型所属线程 ID	XNModelThID	INT32	调度该模型的线程 ID
5.	模型运行节点	XNModelNode	INT32	模型提交的周期性函数所在的运行节点
6.	模型运行优先级	XNModelPro	INT32	最高 99，最低 0
7.	模型运行周期数	XNMdlRunCnt	INT32	模型运行周期数目
8.	模型运行实时频率	XNMdlCurFreq	DOUBLE	模型当前实际运行频率，单位 Hz
9.	模型运行设定频率	XNMdlSetFreq	DOUBLE	模型设定的运行频率，单位 Hz

4.4.10. XNDDSDataInterface

4.4.10.1. 概述

1. 接口名称：模型 DDS 数据交互接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNDDSData
        b. 非技术（自然语言）名称：模型 DDS 数据交互；
    2) 数据元素类型：结构体；
4. 通信方法：Fast DDS 通信；
5. 通信协议：DDS 通信协议。

4.4.10.2. 数据元素定义

模型 DDS 数据交互（XNDDSData）主要用于模型间数据交互，同时仿真监控终端可以监控交互数据。模型 DDS 数据交互结构体的内容由 Fast DDS 提供的 Fast DDS Gen 工具根据接口描述文件（XNIDLFile）生成，因此其数据元素内容与 4.4.5.2 节中描述的一致。

4.4.11. XNSnapshotInterface

4.4.11.1. 概述

1. 接口名称：快照存储与读取接口
2. 接口类型：数据存储与检索接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNSnapshot
        b. 非技术（自然语言）名称：快照数据；
        c. 缩写：XNSnp
    2) 数据元素类型：MCAP 文件；
4. 通信方法：Fast DDS 通信；
5. 通信协议：DDS 通信协议。

4.4.11.2. 数据元素定义

快照数据（XNSnapshot）使用 Fast DDS 提供的 Record and Replay 工具记录和回放发布到 DDS 环境中的数据，它将消息保存为 MCAP 格式的数据库。MCAP 文件包含以下内容：

1. 发布时间戳：记录每个消息发布的时间戳，这使得数据可以按照原始发布时间进行精确回放。

2. 序列化数据：保存实际发布的消息内容，这些数据经过序列化处理，以便在不同的操作系统和平台上进行传输和存储。

3. 数据序列化类型和格式：定义了数据的序列化方式，包括使用的编码格式（如 CDR、JSON 等）。这确保了数据在不同的软件之间可以正确解析和反序列化。

4. DDS 主题（Topics）：记录了每个消息所属的 DDS 主题，这有助于在回放时正确地将消息分发到相应的订阅者。

5. 数据类型：记录了消息的数据类型，这有助于在回放时正确地解析和处理消息。

6. 实体信息：记录了发布者和订阅者的实体信息，包括域 ID、参与者 ID 等，这有助于在复杂的网络环境中正确地识别和匹配消息。

4.4.12.XNCommandInterface

4.4.12.1. 概述

1. 接口名称：仿真控制指令接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNCommand
        b. 非技术（自然语言）名称：仿真控制指令；
        c. 缩写：XNCmd
    2) 数据元素类型：结构体；
4. 通信方法：UDP；
5. 通信协议：UDP/IP。

4.4.12.2. 数据元素定义

仿真控制指令（XNCommand）由 TDM 计算机发送，通过控制指令解析服务接收并进

行指令解析，之后发送至对应的模块完成指令响应。仿真控制指令结构体的内容包括：

表 11 模型运行状态结构体

序号	元素名称	元素标识	数据类型	元素含义
1.	指令标识符	XNCmdID	char[20]	1） 用以识别指令类型的标识符，只有在模型配置文件（XNModelConfigFile）和服务配置文件（XNServiceConfigFile）中配置过的指令才能被识别； 2） 指令标识符支持通配符（*符号）来进行模糊识别； 3） 指令标识符长度限制为 20 个字节。
2.	指令数据	XNCmdData	char[]	控制指令所需传输的序列化的参数数据

4.4.13.XNARINC429Interface

4.4.13.1. 概述

1. 接口名称：ARINC 429 虚拟航空总线通信接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
    3) 数据元素名称：
        a. 项目唯一标识符：XNARINC429Data
        b. 非技术（自然语言）名称：ARINC 429 总线数据；
        c. 缩写：XN429Data
    4) 数据元素类型：数据字；
4. 通信方法：UDP；
5. 通信协议：ARINC 429 通信协议与 UDP/IP。

4.4.13.2. 数据元素定义

ARINC 429 总线数据（XNARINC429Data）是 TDM 计算机与二进制数据包模型间的交互数据，它们通过 ARINC 429 虚拟航空总线通信服务进行交互。ARINC 429 的数据传输单位是字，每个字由 32 位组成，格式如下：

表 12 ARINC 429 数据格式

序号	位号	标识	功能	含义
1.	1～8	LABEL	标号位	标记出这个传送字内信息的类型
2.	9～10	SDI	源/目的识别位	指示信息的来源或信息的终端
3.	11～29	DATA	数据位	/
4.	30～31	SSM	符号状态位	指出数据的特性，如方向、符号等（南，北，正，负）
5.	32	Parity	奇偶校验位	用于检查发送的数据是否有效

4.4.14.XNARINC664Interface

4.4.14.1. 概述

1. 接口名称：ARINC 664 虚拟航空总线通信接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
   1) 数据元素名称：
      a. 项目唯一标识符：XNARINC664Data
      b. 非技术（自然语言）名称：ARINC 664 总线数据；
      c. 缩写：XN429Data
   2) 数据元素类型：AFDX 数据帧；
4. 通信方法：UDP；
5. 通信协议：ARINC 664 通信协议与 UDP/IP。

4.4.14.2. 数据元素定义

ARINC 664 总线数据（XNARINC664Data）是 TDM 计算机与二进制数据包模型间的交互数据，它们通过 ARINC 664 虚拟航空总线通信服务进行交互。ARINC 664 的数据包帧格式与 IEEE802.3 以太网的帧格式基本相同，格式如表 13 所示。

表 13 ARINC 664 数据格式

序号	字节数	标识	功能	含义
1.	7	Preamble	帧头	每个字节值为 0xAA，用于同步发送器和接收器的时钟，确保接收器能够准确识别数据帧的开始位置
2.	1	SFD	帧起始分隔符	用于标识以太网帧的开始，为 0xAB
3.	6	DA	目的 MAC 地址	数据帧接收者的硬件地址
4.	6	SA	源 MAC 地址	数据帧发送者的硬件地址
5.	2	Ipv4	IPV4 类型	标识数据帧的协议类型，为 0x0800

6.	20	IP	IP 结构	IP 的首部结构
7.	8	UDP	UDP 结构	标识数据包的传输层信息，确保数据包的正确传输
8.	17~1471	AFDX Payload	AFDX 负载	封装应用层数据，如传感器数据、控制命令等
9.	1	SN	序列号	用于维护虚拟链路（VL）中的数据顺序，确保数据的完整性和顺序。序列号在设备启动或重置时初始化为 0，达到 255 后回到 1
10.	4	FCS	帧校验序列	用于错误检测的校验和，确保数据帧在传输过程中没有发生错误
11.	12	IG	帧间间隔	用于帧之间的间隔，确保接收器有足够的时间处理前一个帧。

在 AFDX 数据帧的有效数据载荷（AFDX Payload）部分，由多个功能数据集（Functional Data Set, FDS）组成，每个 FDS 包含一组功能状态集（Functional Status Set, FSS）和数据集合（Data Set, DS），每个 FSS 包含 4 个功能状态位（Functional Status Bytes, FSB），每个 FSB 对应一组 DS，每个 FSB 用来指示这组 DS 的数据有效性，每组 DS 包含若干数据，其数据结构如图 28 所示。

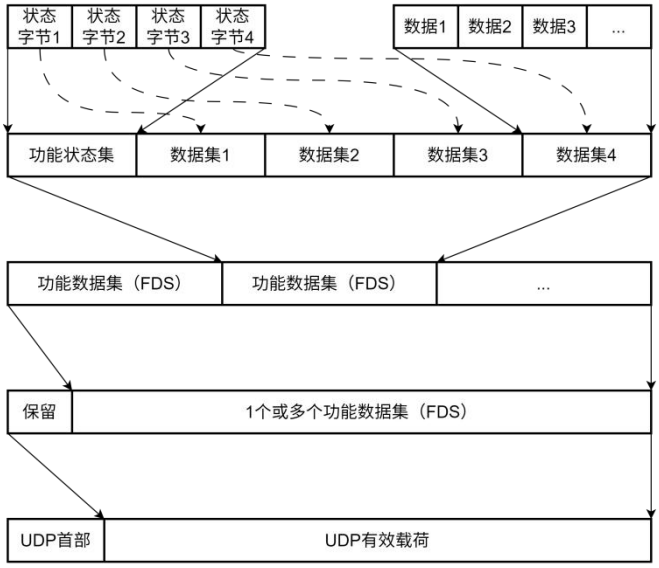

图 28 AFDX 数据帧功能数据集结构

FSB 的定义如表 14 所示。

表 14 AFDX 数据帧 FSB 定义

状态	含义	值	优先级
ND	无数据	0x00	1
NO	一般操作	0x03	4

FT	功能测试	0x0c	3
NCD	原始数据	0x30	2

4.4.15.XNARINC708Interface

4.4.15.1. 概述

1. 接口名称：ARINC 708 虚拟航空总线通信接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNARINC708Data
        b. 非技术（自然语言）名称：ARINC 708 总线数据；
        c. 缩写：XN708Data
    2) 数据元素类型：ARINC 708 数据帧；
4. 通信方法：UDP；
5. 通信协议：ARINC 708 通信协议与 UDP/IP。

4.4.15.2. 数据元素定义

ARINC 708 总线数据（XNARINC708Data）是 TDM 计算机与二进制数据包模型间的交互数据，它们通过 ARINC 708 虚拟航空总线通信服务进行交互。ARINC 708 数据字由 64 位的头部和 512 个 3 位的数据值组成，总长度为 1600 位。用于机载脉冲多普勒天气雷达系统的头部信息具体格式如表 15 所示。

表 15 ARINC 708 数据字头部

序号	位号	功能	描述
1.	1~8	标签	始终为 0b10110100
2.	9~10	控制接受	是否接受控制
3.	11	从属	0=主（正常），1=从属
4.	12~13	保留	
5.	14~18	模式通告	每一位表示一种模式的状态，0=正常，1=该模式启动
6.	19~25	故障	每一位表示一种故障，0=正常，1=故障
7.	26	稳定化	0=关闭，1=开启
8.	27~29	运行模式	000=测试模式，001=手动模式，010=自动模式等
9.	30~36	倾斜角度	0b000000~0b111111 表示 -60°～+60°，以 4° 为步进
10.	37~42	增益	0b000000~0b111111 表示最低增益到最高增益

11.	43~48	范围	0b000000~0b111111 表示从零开始的海里范围，以 10 海里为步进
12.	49	保留	
13.	50~51	数据接受	是否接受数据
14.	52~63	扫描角度	0x000~0xfff 表示 0°～360°，以 1°为步进
15.	64	保留	

数据部分包含 512 个 3 位的数据值，每个数据值表示一个像素的颜色编码。这些数据值用于在雷达显示屏上绘制天气模式。

4.4.16.XNARINC825Interface

4.4.16.1. 概述

1. 接口名称：ARINC 825 虚拟航空总线通信接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNARINC825Data
        b. 非技术（自然语言）名称：ARINC 825 总线数据；
        c. 缩写：XN825Data
    2) 数据元素类型：ARINC 825 数据帧；
4. 通信方法：UDP；
5. 通信协议：ARINC 825 通信协议与 UDP/IP。

4.4.16.2. 数据元素定义

ARINC 825 总线数据（XNARINC825Data）是 TDM 计算机与二进制数据包模型间的交互数据，它们通过 ARINC 825 虚拟航空总线通信服务进行交互。ARINC 825 数据帧基于 CAN 2.0B 扩展帧格式，具体结构如表 16 所示。

表 16 ARINC 825 数据帧

序号	长度	功能	标识	描述
1.	29 位	标识符	Identifier	ARINC 825 标识符，见表 17
2.	6 位	控制字段	Control Field	4 位数据长度码（DLC），表示数据字段的长度，范围从 0 到 15 字节。2 位保留位。
3.	0~15 字节	数据字段	Data Field	实际传输的数据，长度由 DLC 字

				段指定
4.	15 位	校验字段	CRC Field	CRC 校验码，用于检测数据传输中的错误。
5.	1 位	应答字段	ACK Field	应答位 (ACK)： 0：接收器正确接收到数据帧。 1：接收器未接收到数据帧或接收到错误。
6.	7 位	帧结束	EOF	7 个连续的显性位，表示帧的结束。

表 17 ARINC 825 标识符

序号	位号	功能	标识	描述
1.	29~27	逻辑通信信道	LCC	000：异常事件通道（EEC） 001：正常工作通道（NOC） 010：节点服务通道（NSC） 011：用户自定义通道（UDC） 100：测试与维护通道（TMC） 101：CAN 基本帧兼容通道（FMC）
2.	26~19	功能代码标识符	FID	用于识别消息的源
3.	18~17	保留	RSD	
4.	16	本地	LCL	0：消息目的地不限于本地总线段。 1：消息目的地仅限于本地总线段。
5.	15	私有	PVT	0：消息为公共用途。 1：消息为特殊用途。
6.	14~8	数据对象代码	DOC	用于识别消息的有效载荷
7.	7~0	冗余通道标识符	RCI	用于识别四个冗余源之一

4.4.17.XNDiscreteDataInterface

4.4.17.1. 概述

1. 接口名称：离散量通信接口
2. 接口类型：实时数据传送接口
3. 接口数据元素：
   1) 数据元素名称：
      a. 项目唯一标识符：XNDiscreteData
      b. 非技术（自然语言）名称：离散量数据；
      c. 缩写：XNDisData
   2) 数据元素类型：UDP 数据报；

4. 通信方法：UDP；

5. 通信协议：UDP/IP。

4.4.17.2. 数据元素定义

离散量数据（XNDiscreteData）是 TDM 计算机与二进制数据包模型间的一些不通过虚拟航空总线交互的数据，它们通过离散量通信服务进行交互。离散量数据采用的 UDP 数据报格式如所示。

表 18 离散量数据 UDP 数据报

序号	字节数	功能	标识	描述
1.	2	源 IP 地址	SIP	接收方的 IP 地址
2.	2	目的 IP 地址	DIP	接收方的 IP 地址
3.	1	保留	RSD	
4.	1	协议号		值为 17，表示 UDP 协议
5.	2	UDP 长度		UDP 数据报的总长度，包括 UDP 头部和数据部分，用于计算校验和
6.	2	源端口号	SP	发送方的端口号
7.	2	目的端口号	DP	接收方的端口号
8.	2	UDP 总长度		UDP 数据报的总长度，包括 UDP 头部和数据部分。
9.	2	校验和	CS	校验和字段用于检测 UDP 数据报在传输过程中是否发生错误。
10.	可变	数据部分	Data	数据部分包含应用程序要传输的实际数据

4.4.18.XNEditorInterface

4.4.18.1. 概述

1. 接口名称：仿真配置终端交互界面接口

2. 接口类型：用户交互界面接口

3. 接口数据元素：

   1) 数据元素名称：

      a. 项目唯一标识符：XNEditorWindow

      b. 非技术（自然语言）名称：仿真配置终端交互界面；

      c. 缩写：XNEd

   2) 数据元素类型：交互界面；

4. 通信方法：图形化界面交互；

5. 通信协议：无。

4.4.18.2. 数据元素定义

仿真配置终端交互界面（XNEditorWindow）是用户对运行环境配置文件（XNScenarioConfigFile）、模型配置文件（XNModelConfigFile）、服务配置文件（XNServiceConfigFile）等 XML 文件的查看与修改提供可视化交互的界面，其简单示意图如图 29 所示。

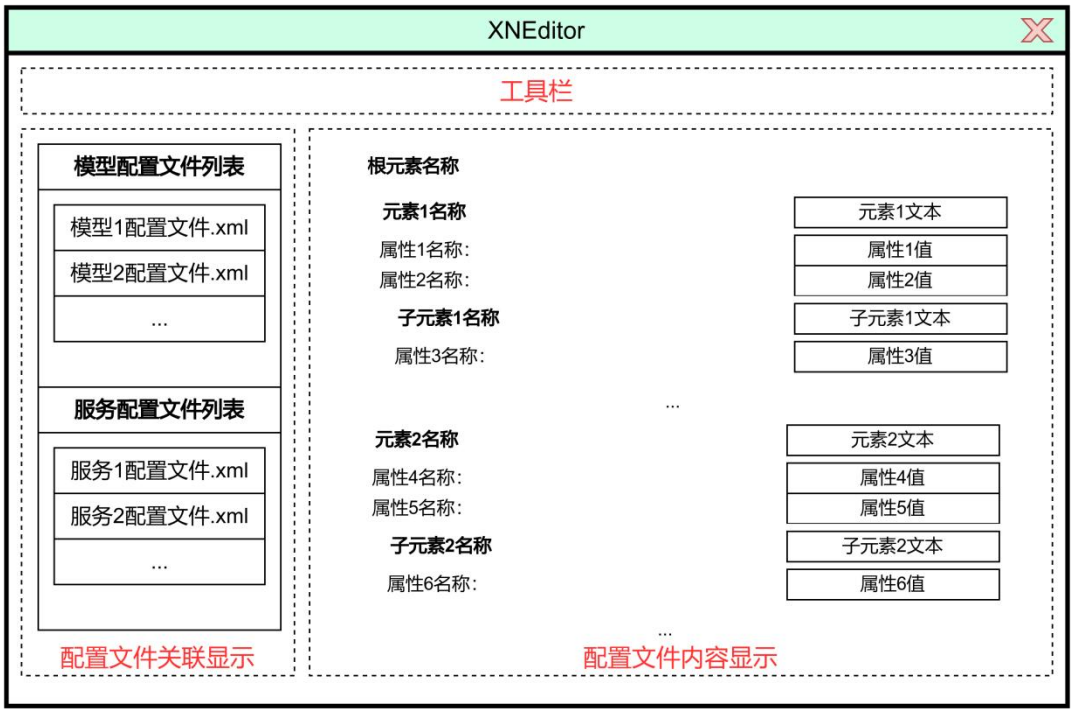

图 29 仿真配置终端 XML 文件配置交互界面示意图

仿真配置终端交互界面也支持查看与修改接口描述文件（XNIDLFile），其简单示意图如图 30 所示。

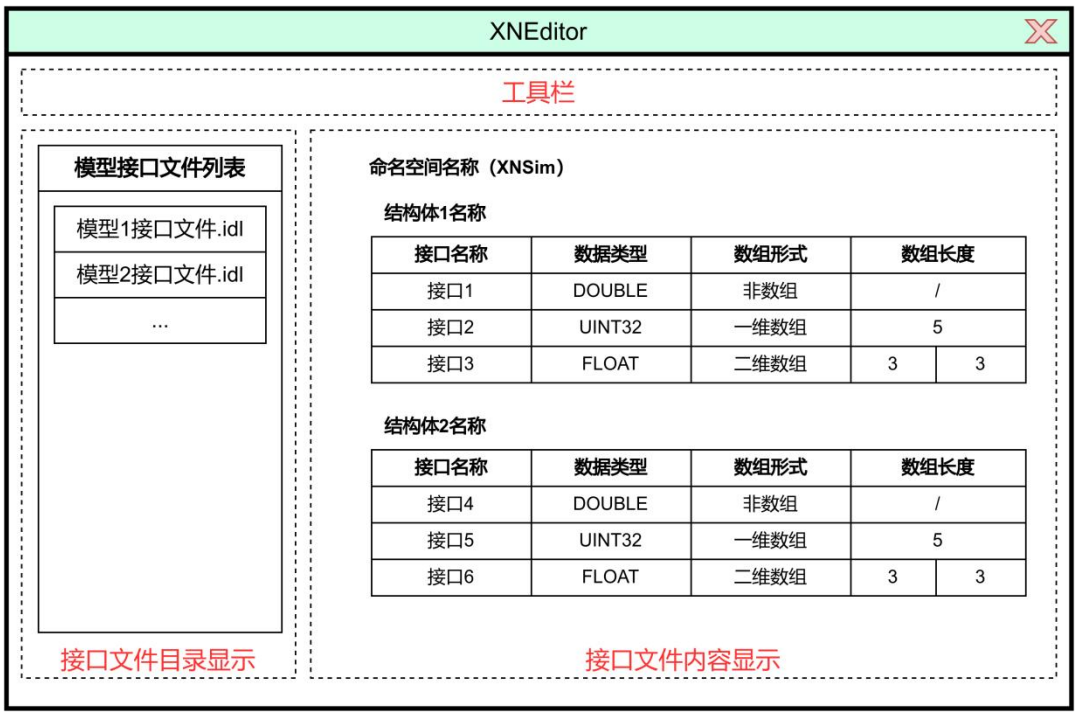

图 30 仿真配置终端 IDL 文件配置交互界面示意图

4.4.19.XNRunnerInterface

4.4.19.1. 概述

1. 接口名称：仿真调度终端交互界面接口
2. 接口类型：用户交互界面接口
3. 接口数据元素：
    1) 数据元素名称：
        a. 项目唯一标识符：XNRunnerWindow
        b. 非技术（自然语言）名称：仿真调度终端交互界面；
        c. 缩写：XNRn
    2) 数据元素类型：交互界面；
4. 通信方法：图形化界面交互；
5. 通信协议：无。

4.4.19.2. 数据元素定义

仿真调度终端交互界面（XNRunnerWindow）为用户提供仿真运行控制的可视化交互界面，其简单示意图如图 31 所示。

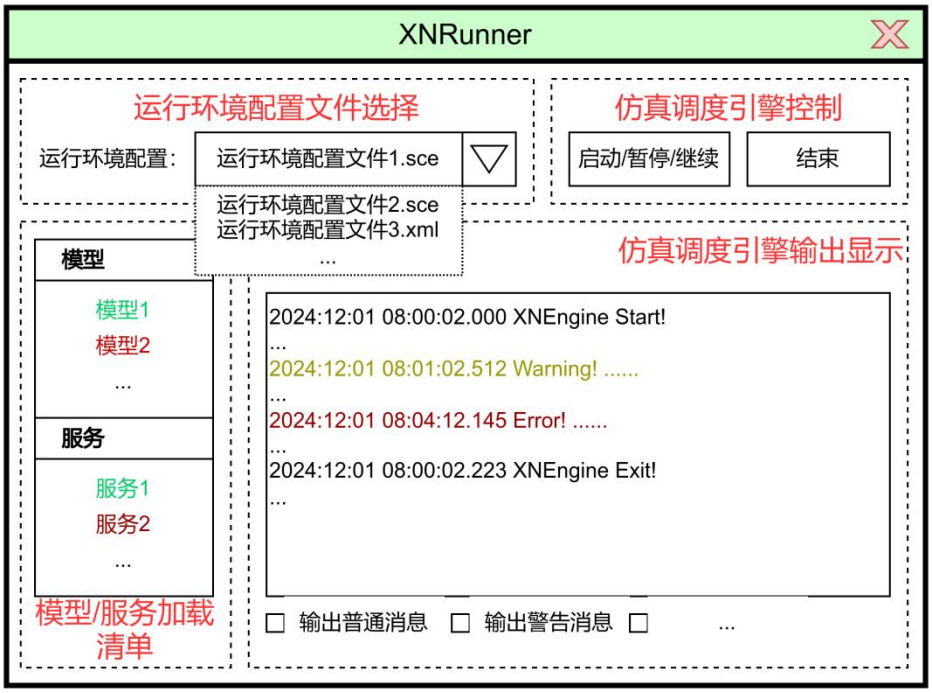

图 31 仿真调度终端交互界面示意图

4.4.20.XNMonitorInterface

4.4.20.1. 概述

1. 接口名称：仿真监控终端交互界面接口
2. 接口类型：用户交互界面接口
3. 接口数据元素：
   1) 数据元素名称：
      a. 项目唯一标识符：XNMonitorWindow
      b. 非技术（自然语言）名称：仿真监控终端交互界面；
      c. 缩写：XNMn
   2) 数据元素类型：交互界面；
4. 通信方法：图形化界面交互；
5. 通信协议：无。

4.4.20.2. 数据元素定义

仿真监控终端交互界面（XNMonitorWindow）为用户提供仿真运行监控、模型运行状监控、模型数据监控与模型数据采集的可视化交互界面。其中，各交互界面的简单示意图如

图 32、图 33、图 34、图 35 所示。

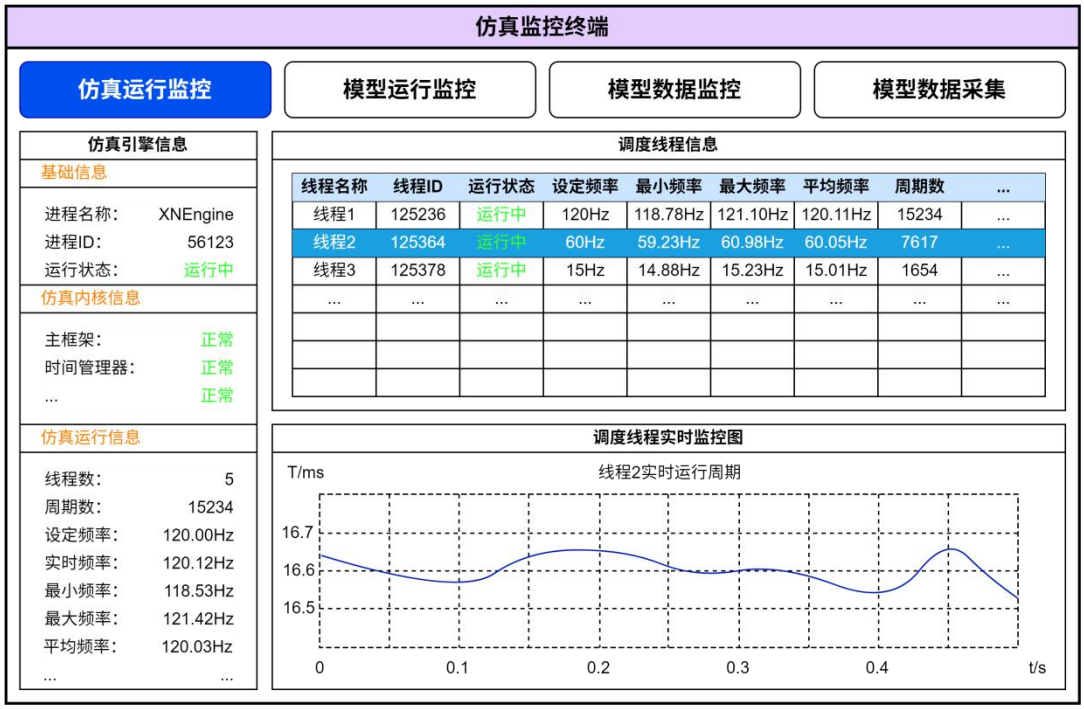

图 32 仿真监控终端仿真运行监控交互界面示意图

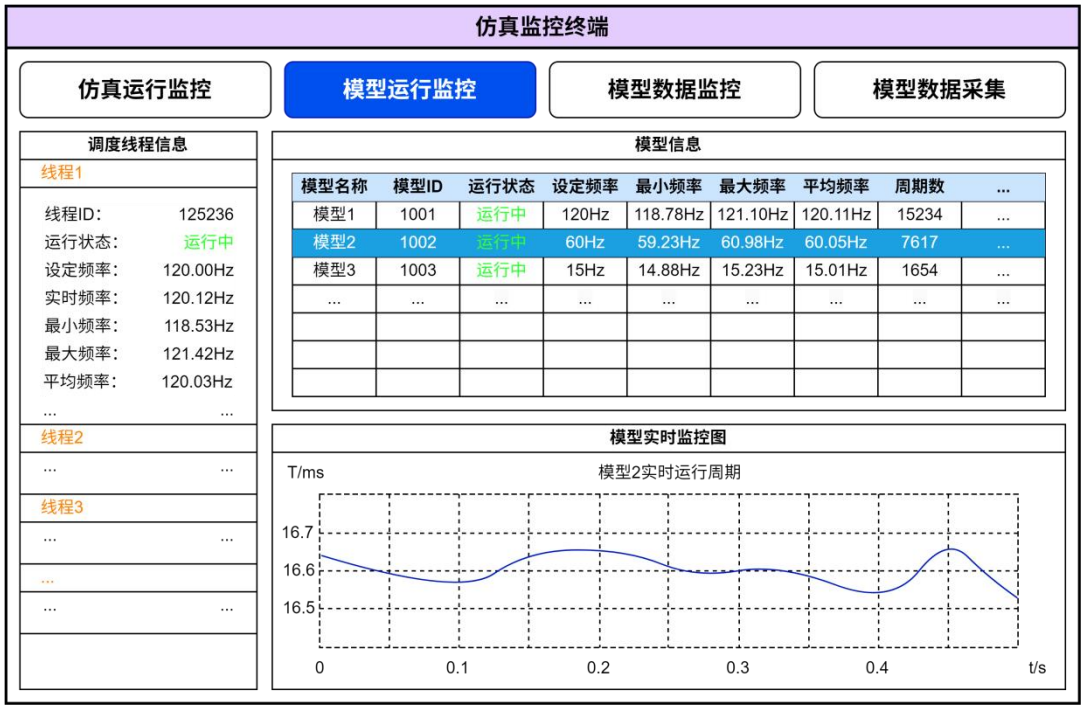

图 33 仿真监控终端模型运行监控交互界面示意图

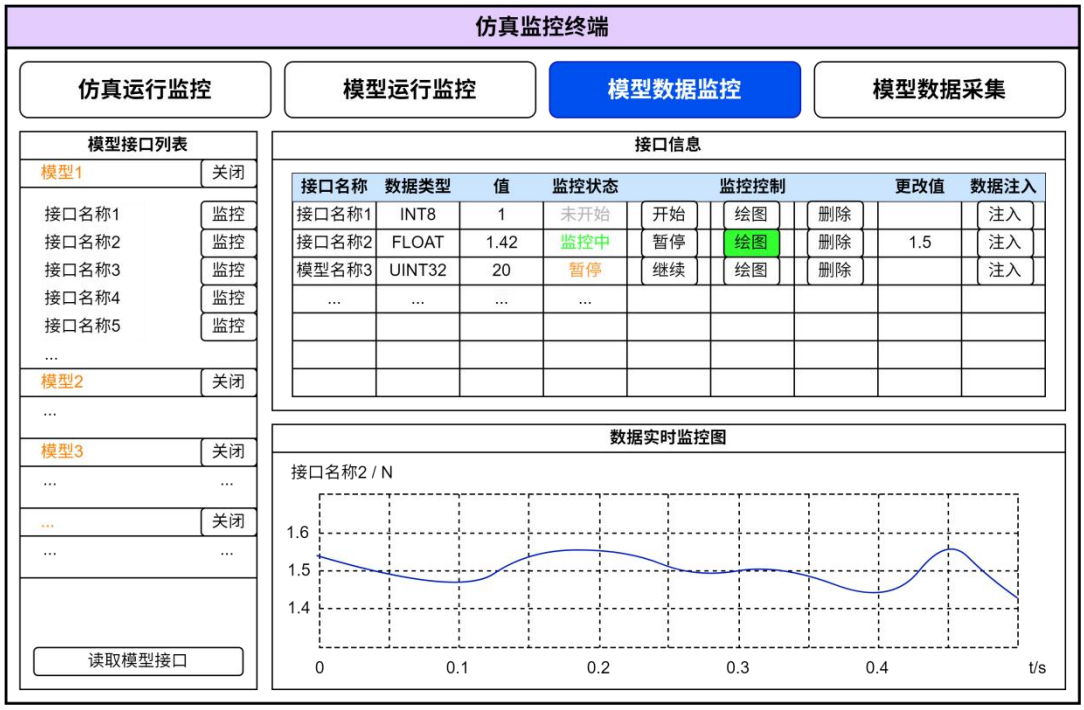

图 34 仿真监控终端模型数据监控交互界面示意图

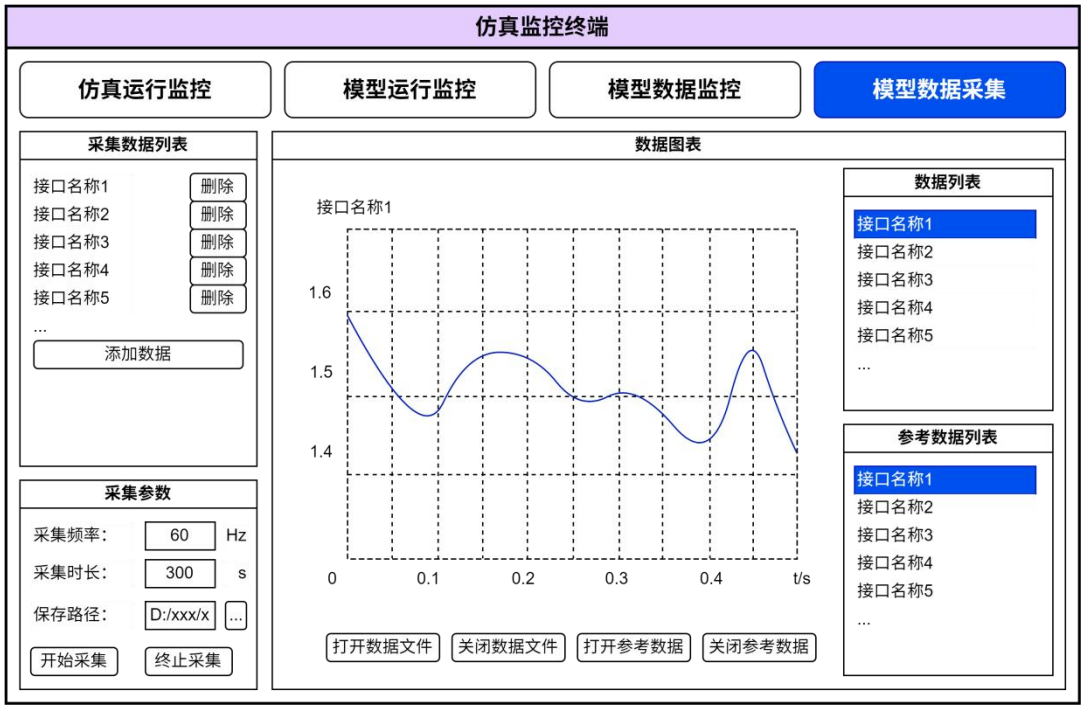

图 35 仿真监控终端模型数据采集交互界面示意图

5. 运行设计

本节对“玄鸟”架构的运行流程进行详细地阐述，涵盖从仿真准备、仿真启动到仿真运行，再到仿真终止的全过程，为开发人员、维护人员以及用户提供清晰、准确地操作指南和参考依据，确保“玄鸟”架构能够高效、稳定地运行，满足设定地功能需求与性能指标。

5.1. 仿真准备

“玄鸟”架构在仿真准备阶段的详细处理流程如图 36 所示。

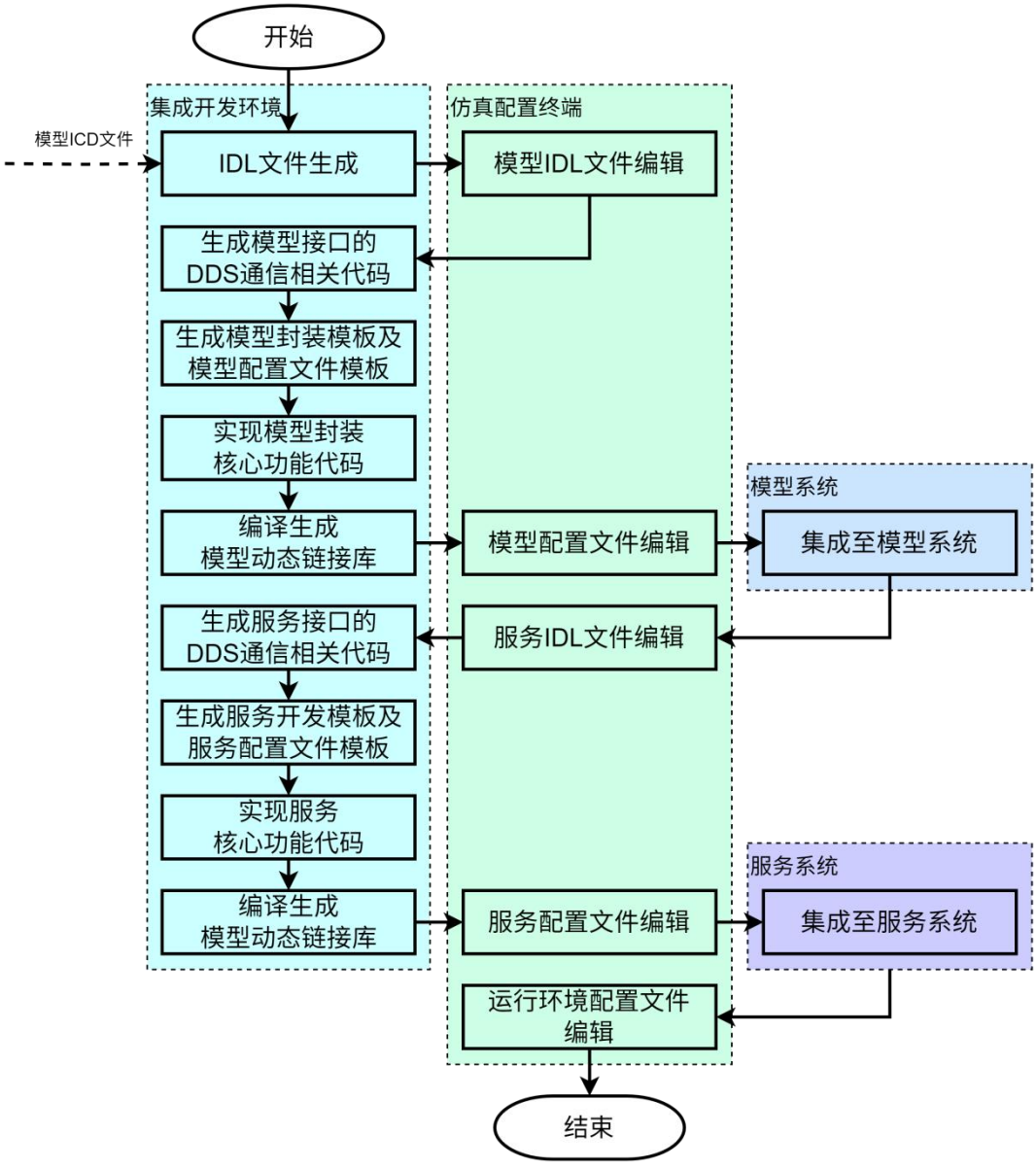

图 36 “玄鸟”架构仿真准备阶段处理流程

详细处理流程如下：

1. 首先使用 IDL 文件生成工具根据模型 ICD 文件生成模型的 IDL 文件，IDL 文件用于定义模型间 DDS 通信的交互主题；

2. 使用仿真配置终端或其它文本编辑器编辑模型 IDL 文件。在编辑过程中，确保模型 IDL 文件中全面涵盖了模型的所有接口，并且仔细检查是否存在语法错误；

3. 使用 Fast DDS Gen 工具，根据模型的 IDL 文件生成模型接口的 DDS 通信相关代码。这些代码不仅定义了交互主题的数据类型，还包括交互数据的序列化等内容；

4. 使用代码构建工具创建模型封装工程，并详细设置模型封装参数，包括但不限于模型名称、版本号、二进制数据包模型动态链接库的相对路径及其入口函数等。在此基础上，将模型对应的 DDS 通信相关代码文件添加进工程之中。代码构建工具将根据仿真内核提供的模型封装框架，自动生成模型封装代码模板与模型配置文件模板；

5. 使用 VS Code 编辑器打开模型封装工程，编写代码以完善模型封装的其它功能，包括：二进制数据包模型接口的数据对接，输入/输出数据必要的转换、计算等处理操作；

6. 编写 CMakeLists.txt 配置文件，指定模型封装工程的编译选项、源文件等信息。接着使用 CMake 进行模型封装工程的自动化构建，通过 GCC/G++对源代码进行编译，生成模型动态链接库；

7. 使用仿真配置终端或其它文本编辑器编辑模型配置文件，确保模型配置文件包含了模型的参数信息、周期性函数信息、可接受的指令信息等在 4.4.3 小节中定义的数据元素，并且仔细检查是否存在语法错误；

8. 将生成的模型动态链接库与模型配置文件一起放置在模型库目录中，从而实现模型在模型系统中的顺利集成；

9. 重复 1~8 的内容封装与集成所需的所有二进制数据包模型；

10. 使用仿真配置终端或其它文本编辑器编辑服务接口的 IDL 文件。在编辑过程中，确保服务 IDL 文件中全面涵盖了服务所需的所有接口，并且仔细检查是否存在语法错误；

11. 使用 Fast DDS Gen 工具生成服务接口的 DDS 通信相关代码；

12. 使用代码构建工具创建服务开发工程，并详细设置服务参数，包括但不限于服务名称、版本号等。在此基础上，将服务对应的 DDS 通信相关代码文件添加进工程之中。代码构建工具将根据仿真内核提供的服务开发框架，自动生成服务代码模板与服务配置文件模板；

13. 使用 VS Code 编辑器打开服务开发工程，编写代码以实现不同功能的服务，具体包

括：

1） UDP/TCP 通信相关功能：包括建立连接、监听端口、数据接收/发送、封包/解包等；

2） 虚拟航空总线通信相关功能：包括 ARINC429、ARINC664、ARINC708、ARINC825 等航空总线协议数据的封包/解包等；

3） 指令解析相关功能：包括指令识别、指令分发等；

4） 仿真事件相关功能：仿真事件监听、仿真事件处理等；

5） DDS 数据交互相关功能：数据发布/订阅、数据的转换、数据计算等；

6） 快照相关功能：通过 Fast DDS Record and Replay 工具的 API 调用，实现对数据的记录与回放等；

14. 编写 CMakeLists.txt 配置文件，指定服务开发工程的编译选项、源文件等信息。接着使用 CMake 进行服务开发工程的自动化构建，通过 GCC/G++对源代码进行编译，生成服务动态链接库；

15. 使用仿真配置终端或其它文本编辑器编辑服务配置文件，确保服务配置文件包含了服务的参数信息、可接受的指令信息等在 4.4.4 小节中定义的数据元素，并且仔细检查是否存在语法错误；

16. 将生成的服务动态链接库与服务配置文件一起放置在服务库目录中，从而实现服务在服务系统中的顺利集成；

17. 重复 10~16 的内容开发与集成所需的所有服务；

18. 使用仿真配置终端或其它文本编辑器编辑运行环境配置文件，确保运行环境配置文件中包含了仿真运行所需的参数、需要加载的服务列表和模型列表等在 4.4.2 小节中定义的数据元素，并且没有语法错误。

19. 运行环境配置文件中不同的参数、服务列表和模型列表可以使仿真具有不同的构型，用户在仿真运行时通过选择不同的运行环境配置文件进行构型切换。

5.2. 仿真启动

“玄鸟”架构在仿真启动阶段的详细处理流程如图 37 所示。

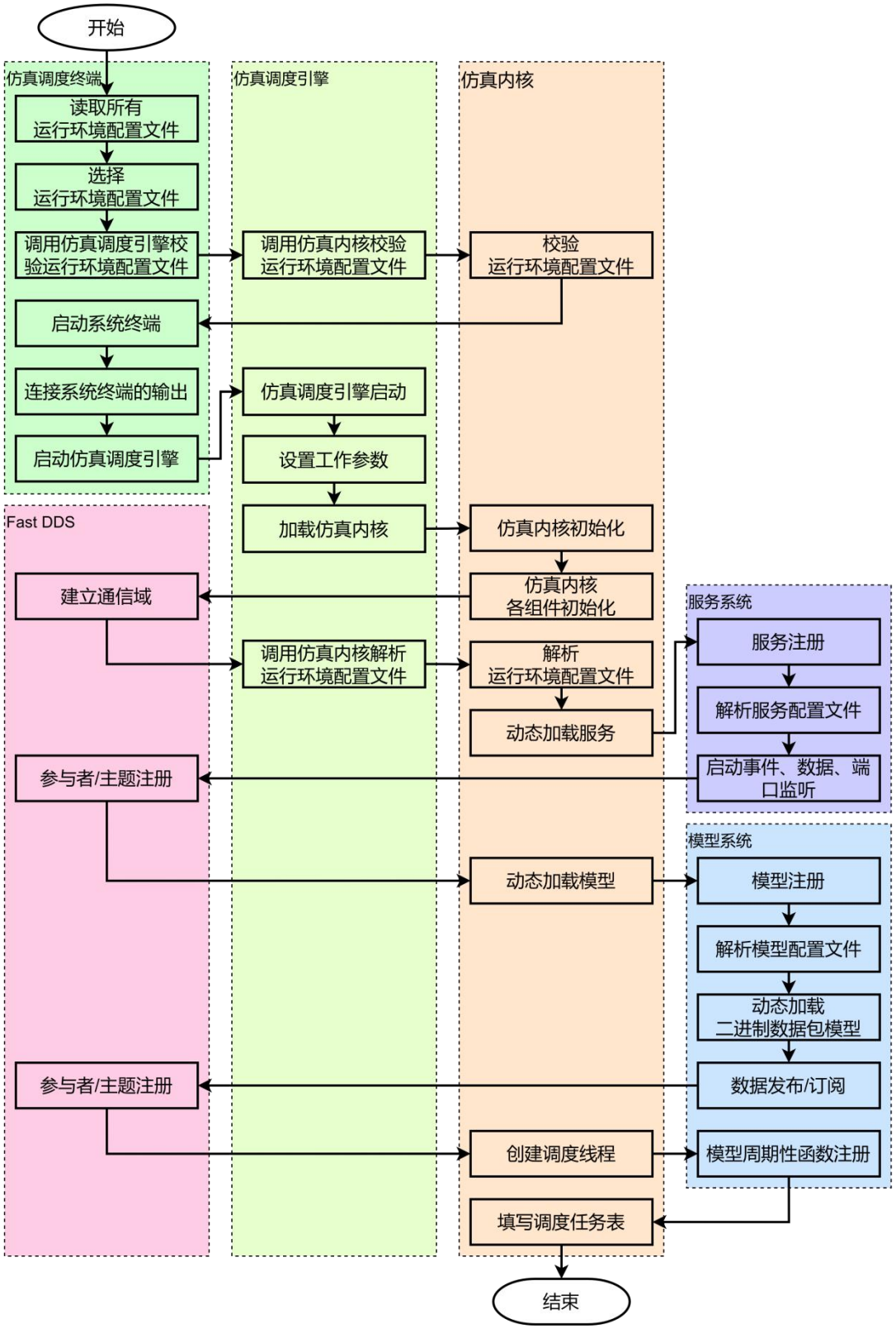

图 37 “玄鸟”架构仿真启动阶段处理流程

详细处理流程如下：

1. 启动仿真调度终端，自动扫描指定目录下的所有运行环境配置文件，并在用户交互

界面中展示，供用户选择以切换不同的仿真构型；

2. 用户完成运行环境配置文件的选择后，仿真调度终端自动启动运行环境配置文件的校验流程，调用仿真调度引擎执行校验任务；

3. 仿真调度引擎进一步调用仿真内核，执行运行环境配置文件的校验工作；

4. 仿真内核对运行环境配置文件进行详细解析与校验，包括以下关键内容：

1） 检查仿真运行参数设置是否合理；

2） 校验服务列表中各服务的服务配置文件是否有效；

3） 确认服务列表中各服务动态链接库是否能够加载；

4） 校验模型列表中各模型的模型配置文件是否有效；

5） 确认模型列表中各模型动态链接库是否能够加载。

5. 运行环境配置文件校验通过，用户可点击仿真调度终端用户交互界面中的仿真开始按钮启动仿真，此时仿真调度终端将执行启动系统终端的操作；

6. 仿真调度终端与系统终端建立连接，以便接收系统终端中所有的输出信息；

7. 仿真调度终端向系统终端中输出命令行指令，启动仿真调度引擎；

8. 仿真调度引擎启动后，首先进行工作参数设置等初始化操作，包括但不限于：

1）设置工作目录；

2）创建日志记录文件；

3）设置 CPU 亲和性：用于分配仿真运行的计算资源；

4）内存锁定：防止因内存不足或数据迁移导致的性能下降或错误；

9. 仿真调度引擎进行仿真内核的加载；

10. 仿真内核进行初始化操作，创建一系列关键组件，这些组件构成了仿真的核心架构，即仿真主框架、场景描述管理器、线程管理器、DDS 管理器、事件管理器、模型管理器和服务管理器；

11. 仿真内核对仿真主框架执行初始化操作，包括但不限于以下操作：

1） 仿真主框架作为整个仿真内核的控制中心，负责调用其它各组件的初始化；

2） 其它各组件初始化其内部的数据结构和控制逻辑；

3） 线程管理器创建初始线程池；

4） 事件管理器注册所有仿真事件，建立事件监听机制；

5） DDS 管理器会初始化 DDS 通信环境；

12. DDS 管理器将在 Fast DDS 中执行 DDS 通信域的创建操作；

13. 仿真调度引擎调用仿真内核进行运行环境配置文件解析；

14. 仿真内核进行运行环境配置文件解析，逐项读取配置文件中的仿真运行参数，并根据这些参数对仿真环境进行相应的设置；

15. 仿真内核根据运行环境配置文件中的服务列表，动态地加载所有配置的服务动态链接库，并调用它们的预处理函数；

16. 服务动态链接库成功加载并完成预处理后，将立即执行服务的初始化操作。初始化完成后，服务会向服务管理器注册自身，告知服务管理器其已经完成初始化；

17. 服务读取并解析自身的服务配置文件。服务配置文件中包含了服务运行所需的各种参数和设置，通过解析这些信息，服务可以对自己的运行状态和行为进行设置，确保能够按照预设的方式提供服务功能；

18. 服务根据自身所承担的功能，进行仿真事件监听、通信端口监听或 DDS 主题数据的发布/订阅操作；

19. 服务向 Fast DDS 中注册其所需交互的主题，并将自身注册为该主题的参与者；

20. 仿真内核完成服务列表中所有服务的加载后，会继续根据运行环境配置文件中的模型列表动态加载配置的所有模型动态链接库，并调用它们的预处理函数；

21. 模型动态链接库成功加载并完成预处理后，将立即执行模型的初始化操作。初始化完成后，模型会向模型管理器注册自身，告知模型管理器其已经完成初始化；

22. 模型在完成注册后，会读取并解析自身的模型配置文件。模型配置文件中包含了模型运行所需的各种参数和设置，通过解析这些信息，模型可以对自己的运行状态和行为进行设置，以确保能够按照预设的方式参与仿真，输出准确的仿真结果；

23. 模型动态加载与其关联的二进制数据包模型动态链接库；

24. 模型对其所需交互的数据主题进行发布/订阅；

25. 模型向 Fast DDS 中注册其所需交互的主题，并将自身注册为该主题的参与者；

26. 仿真内核完成模型列表中所有模型的加载后，将控制线程管理器在线程池中创建所有调度线程；

27. 模型将周期性执行函数向线程管理器进行注册;

28. 线程管理器根据模型提交注册时提供的参数将这些周期性执行函数填写的到对应进程的调度任务表的对应位置中。

5.3. 仿真运行

“玄鸟”架构在仿真运行阶段的详细处理流程如图 38 所示。

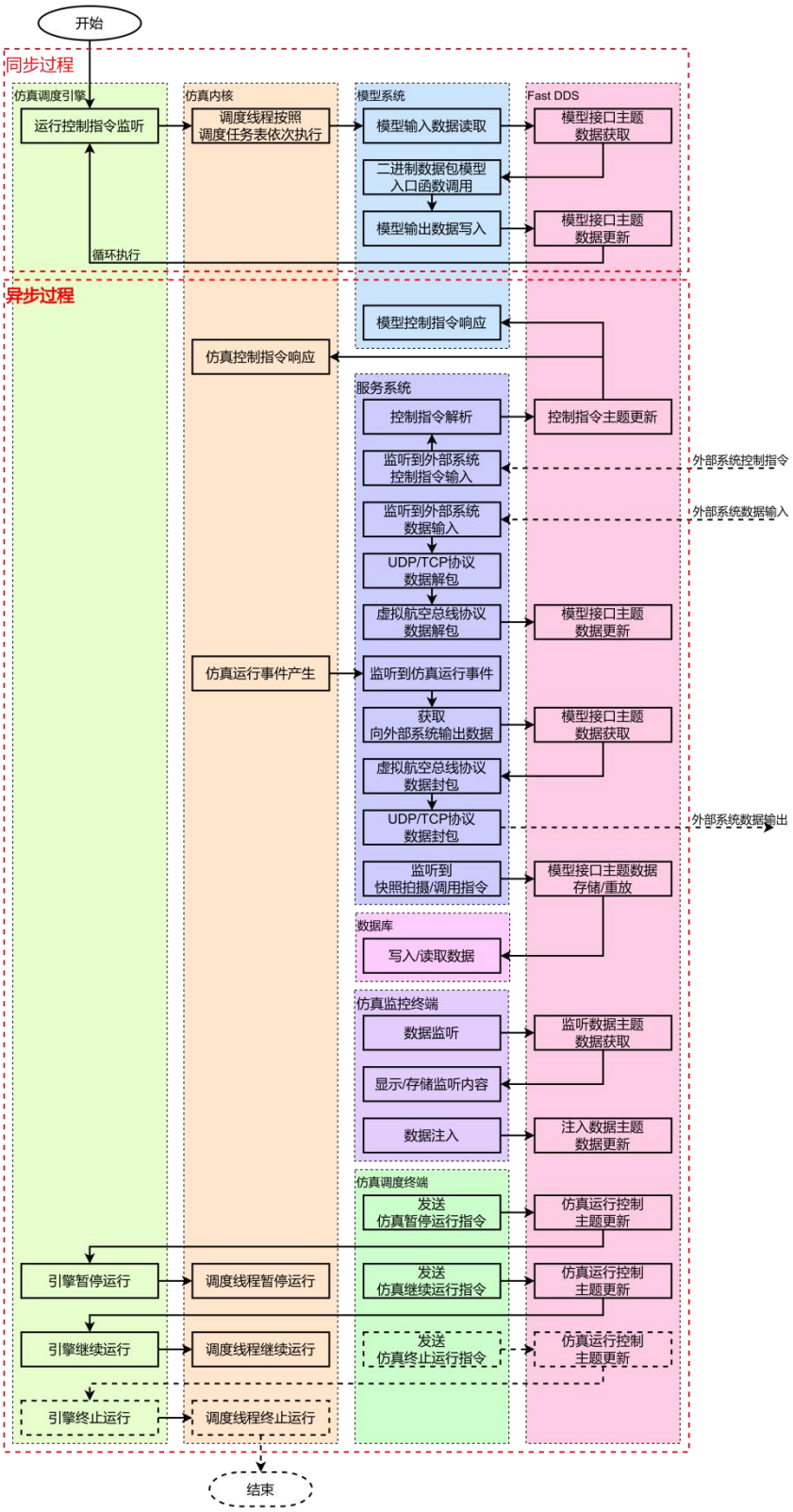

图 38 “玄鸟”架构仿真运行阶段处理流程

仿真运行阶段的同步处理流程如下：

1. 仿真调度引擎开始仿真，进行仿真运行控制指令的监听，当监听到仿真运行控制指令后启动异步流程进行仿真运行控制；

2. 仿真内核各调度线程按照调度任务表依次调度模型的周期性执行函数；

3. 模型的周期性执行函数进行模型输入数据的读取；

4. 读取模型输入数据需要从 Fast DDS 中获取模型输入接口的主题数据；

5. 模型的周期性执行函数调用二进制数据包模型动态链接库的入口函数，进行仿真计算，得到模型输出数据；

6. 模型的周期性执行函数将输出数据发布到 Fast DDS 中；

7. Fast DDS 将模型数据接口的主题数据进行更新；

8. 循环执行 1～7 的步骤，继续执行后续的仿真调度任务。

仿真运行阶段的异步处理流程如下：

1. 外部系统控制指令响应异步流程：

1） 当外部系统有控制指令输入时，服务系统中的控制指令解析服务监听到该控制指令的输入；

2） 控制指令解析服务对控制指令进行解析与识别，确定控制指令的目标对象及指令参数；

3） 控制指令解析服务在 Fast DDS 中更新对应的控制指令主题；

4） 如果是模型控制指令，模型接收到该主题的指令后完成对应的指令响应操作，如模型参数设置、模型冻结等；

5） 如果是仿真控制指令，仿真内核接收到该主题的指令后完成对应的响应，如调度线程参数更改、调度线程冻结等。

2. 外部系统数据输出响应异步流程：

1）当外部系统有控制指令输入时，服务系统中的通信服务（虚拟航空总线通信服务及离散量通信服务）监听到该数据的输入；

2）通信服务根据采用的通信协议（UDP 或 TCP）对接收到的数据进行解包；

3）如果是虚拟航空总线通信服务，还需要根据采用的航空总线协议（ARINC429、ARINC664、ARINC708 或 ARINC825）对解包后的数据进一步进行解包；

4）将解包完成的数据发布到 Fast DDS 中完成模型接口主题数据的更新。

3. 仿真运行事件响应异步流程：

1） 当有仿真运行事件(一般是需要向外部系统输出数据的仿真运行事件)产生时，服务系统中的通信服务（虚拟航空总线通信服务及离散量通信服务）监听到该事件的产生；

2） 通信服务获取需要向外部系统输出的数据；

3） 从 Fast DDS 中获取到需要向外部系统输出的数据；

4） 如果是虚拟航空总线通信服务，需要根据采用的航空总线协议（ARINC429、ARINC664、ARINC708 或 ARINC825）对输出数据进行封包；

5） 通信服务根据采用的通信协议（UDP 或 TCP）对需要输出的数据进一步进行解包；

6） 通信服务使用 UDP 或 TCP 向外部系统发送输出数据。

4. 快照拍摄/调用异步流程：

1） 当快照的拍摄/调用指令产生（仿真运行事件触发或外部系统控制指令触发）时，快照服务将监听到该指令；

2） 快照服务调用 Fast DDS Record and Replay 工具的 API，进行快照数据的采集/重放；

3） Fast DDS Record and Replay 工具将所有模型接口主题数据进行采集/重放；

4） 采集/重放的主题数据存放于 MCAP 数据库文件中。

5. 仿真数据监控异步流程：

1） 仿真监控终端订阅需要监控的数据主题，包括：仿真运行状态数据主题、调度线程运行状态数据主题、模型运行状态数据主题、模型 DDS 交互数据主题等；

2） Fast DDS 将订阅的监控数据主题发送至仿真监控终端；

3） 仿真监控终端对接收到的数据展示在仿真监控终端的各类用户界面上，包括：仿真运行状态交互界面、模型运行状态交互界面、模型数据监控交互界面与模型数据采集交互界面；

6. 仿真数据注入异步流程：

1）仿真监控终端的模型数据监控交互界面可以进行数据注入操作。用户选择需要注入的接口数据名称和数据值，仿真监控终端将发布注入数据所在的主题；

2）Fast DDS 更新该主题数据以完成数据注入；

7. 仿真运行控制异步流程：

1）用户可以通过仿真调度终端控制仿真运行。当用户发送仿真暂停指令时，仿真

调度终端发布仿真运行控制指令主题；

2）Fast DDS 更新仿真运行控制指令主题；

3）仿真引擎监听到仿真运行控制指令主题更改，暂停仿真引擎运行，同时控制仿真内核暂停运行；

4）仿真内核暂停所有调度线程的运行；

5）当用户发送仿真继续指令时，执行同样的流程控制仿真引擎和调度线程继续运行；

6）当用户发送仿真终止指令时，执行同样的流程控制仿真引擎和调度线程终止运行，此部分仿真终止过程在下一节中详细描述。

5.4. 仿真终止

“玄鸟”架构在仿真终止阶段的详细处理流程如图 39 所示。

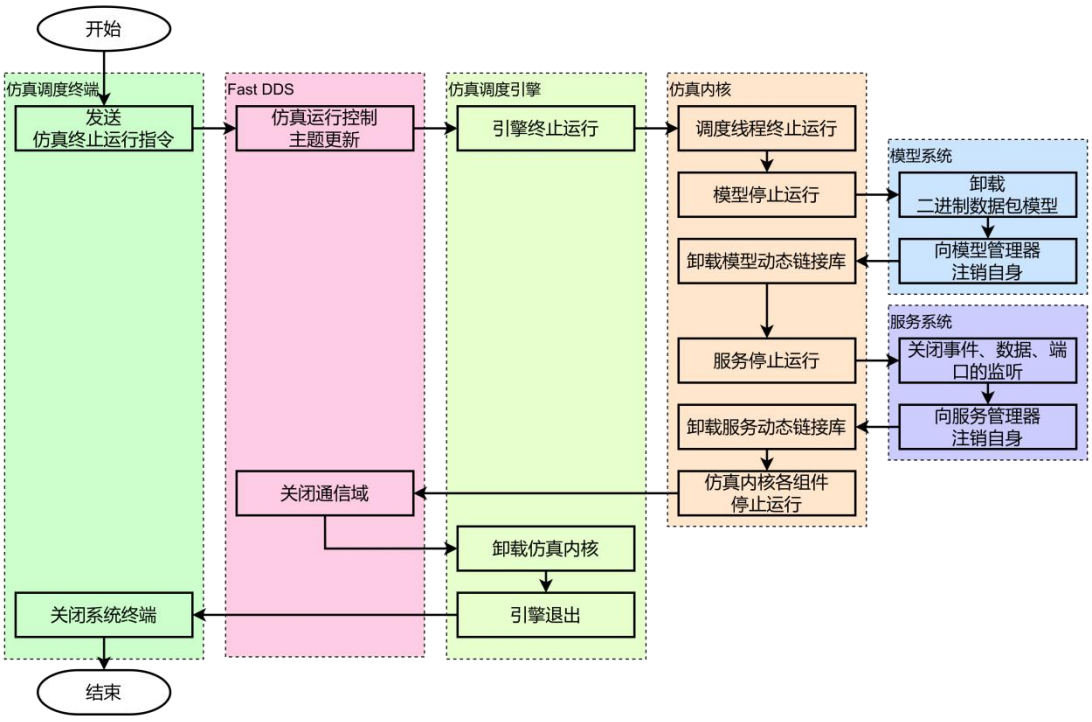

图 39 “玄鸟”架构仿真终止阶段处理流程

详细处理流程如下：

1. 当用户发送仿真终止指令时，仿真调度终端发布仿真运行控制指令主题；

2. Fast DDS 更新仿真运行控制指令主题；

3. 仿真引擎监听到仿真运行控制指令主题更改，终止仿真引擎运行，同时控制仿真内核终止运行；

4. 仿真内核终止所有调度线程的运行；

5. 调度线程终止所有模型周期性函数的调度，清理调度任务表；

6. 模型卸载二进制数据包模型动态链接库；

7. 模型向模型管理器注销自身；

8. 仿真内核卸载模型动态链接库；

9. 仿真内核完成所有模型的卸载后，停止所有服务的运行；

10. 服务关闭所有的事件监听、数据监听和通信监听，并断开与外部系统的连接；

11. 服务向服务管理器注销自身；

12. 仿真内核完成所有服务的卸载后，停止仿真内核各组件的运行；

13. DDS 管理器会关闭 Fast DDS 通信域；

14. 仿真引擎卸载仿真内核；

15. 仿真引擎退出；

16. 仿真调度终端断开与系统终端的连接，关闭系统终端；

6. 出错处理设计

6.1. 出错信息

“玄鸟”架构的出错信息及故障处理表如表 19 所示。

表 19 “玄鸟”出错信息及故障处理表

序号	错误码	错误等级	出错信息	故障处理
1.	0x1000	致命	引擎启动参数中没有运行环境配置文件路径	引擎退出
2.	0x1001	致命	引擎启动参数中运行环境配置文件路径无效	引擎退出
3.	0x1002	致命	引擎启动参数中运行环境配置文件后缀名错误	引擎退出
4.	0x1020	致命	引擎设置 CPU 亲和性失败	引擎退出
5.	0x1021	致命	引擎设置内存锁定失败	引擎退出
6.	0x1022	致命	框架初始化失败	引擎退出
7.	0x1040	致命	解析运行环境配置文件失败	引擎退出
8.	0x2000	致命	初始化场景描述管理器失败	引擎退出
9.	0x2010	致命	初始化时间管理器失败	引擎退出
10.	0x2020	致命	初始化线程管理器失败	引擎退出

11.	0x2030	致命	初始化 DDS 管理器失败	引擎退出
12.	0x2040	致命	初始化事件管理器失败	引擎退出
13.	0x2050	致命	初始化模型管理器失败	引擎退出
14.	0x2060	致命	初始化服务管理器失败	引擎退出
15.	0x2100	致命	打开运行环境配置文件失败	引擎退出
16.	0x2121	致命	无法找到模型库目录	引擎退出
17.	0x2130	致命	无法创建仿真运行控制主题	引擎退出
18.	0x2131	致命	无法注册成为仿真运行控制主题的参与者	引擎退出
19.	0x2132	严重	某个 DDS 交互主题无法创建	警告信息：无法使用该主题进行数据交互
20.	0x2133	严重	某个实体无法注册成为某个 DDS 交互主题的参与者	警告信息：该实体无法使用该主题进行数据交互
21.	0x2150	严重	某模型的动态链接库中没有预处理函数	警告信息：该模型将不会被加载
22.	0x2151	严重	某模型注册失败	警告信息：该模型将不会被加载
23.	0x2152	严重	无法找到某模型的动态链接库	警告信息：该模型将不会被加载
24.	0x2160	严重	无法找到某模型的二进制数据包模型动态链接库	警告信息：该模型将不会调用二进制数据包模型动态链接库
25.	0x2161	严重	无法打开某模型的配置文件	警告信息：该模型将不会被调度任何周期性函数
26.	0x2162	严重	模型某个周期性函数名称配置错误	警告信息：该周期性函数将不会被调度
27.	0x2163	严重	模型某个周期性函数运行频率配置错误	警告信息：该周期性函数将不会被调度
28.	0x2164	严重	模型某个周期性函数运行节点配置错误	警告信息：该周期性函数将不会被调度
29.	0x2210	严重	某调度线程初始化属性失败	警告信息：该调度线程将不会执行
30.	0x2211	严重	某调度线程设置栈空间失败	警告信息：该调度线程将不会执行
31.	0x2212	严重	某调度线程设置调度策略失败	警告信息：该调度线程将不会执行
32.	0x2213	严重	某调度线程设置优先级失败	警告信息：该调度线程将不会执行
33.	0x2214	严重	某调度线程设置调度器继承失败	警告信息：该调度线

				程将不会执行
34.	0x2215	严重	某调度线程创建失败	警告信息：该调度线程将不会执行
35.	0x2216	严重	某调度线程设置 CPU 亲和性失败	警告信息：该调度线程将不会执行

6.2. 补救措施

“玄鸟”架构故障出现后可采取的补救措施见表 20。

表 20 “玄鸟”架构故障补救措施

序号	错误码	错误等级	补救措施
1.	0x1000	致命	引擎启动命令行指令中添加运行环境配置文件路径
2.	0x1001	致命	检查引擎启动命令行指令中运行环境配置文件路径是否存在
3.	0x1002	致命	检查引擎启动命令行指令中运行环境配置文件路径是否是.xml 或.sce 文件
4.	0x1020	致命	检查引擎是否以管理员权限启动 检查系统环境是否是实时操作系统 查看终端或日志中记录的错误信息
5.	0x1021	致命	检查引擎是否以管理员权限启动 查看终端或日志中记录的错误信息
6.	0x1022	致命	查看框架初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2000~0x20ff
7.	0x1040	致命	查看分析运行环境配置文件过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2100~0x21ff
8.	0x2000	致命	查看场景描述管理器初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2001~0x200f
9.	0x2010	致命	查看时间管理器初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2011~0x201f
10.	0x2020	致命	查看线程管理器初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2021~0x202f
11.	0x2030	致命	查看 DDS 管理器初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2031~0x203f
12.	0x2040	致命	查看事件管理器初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2041~0x204f
13.	0x2050	致命	查看模型管理器初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2051~0x205f
14.	0x2060	致命	查看服务管理器初始化过程中在终端或日志中输出的具体错误信息，对应错误码为 0x2061~0x206f
15.	0x2100	致命	运行环境配置文件有格式错误，查看在终端或日志中输出的具体错误位置
16.	0x2121	致命	没有找到模型库所在目录，检查运行环境配置文件中模型库

			目录配置是否正确
17.	0x2130	致命	查看 Fast DDS 在终端或日志中输出的具体错误原因
18.	0x2131	致命	查看 Fast DDS 在终端或日志中输出的具体错误原因
19.	0x2132	严重	查看 Fast DDS 在终端或日志中输出的具体错误原因
20.	0x2133	严重	查看 Fast DDS 在终端或日志中输出的具体错误原因
21.	0x2150	严重	模型封装时缺少了预处理函数的实现，修改并重新封装模型
22.	0x2151	严重	模型加载数量超过 1000 个，尝试减少模型加载
23.	0x2152	严重	检查运行环境配置文件中模型动态库名称是否配置错误
24.	0x2160	严重	检查运行环境配置文件中模型的二进制数据包模型动态库路径是否配置错误
25.	0x2161	严重	检查模型配置文件是否和该模型动态链接库在同一路径下
26.	0x2162	严重	检查模型配置文件中周期性函数名称是否和代码中一致
27.	0x2163	严重	检查模型配置文件中周期性函数运行节点第一个数字是否是 0～5 之间的值
28.	0x2164	严重	检查模型配置文件中周期性函数运行节点第二个数字是否小于等于 2 的（运行节点第一个数字）次幂
29.	0x2210	严重	查看 pthread 库在终端或日志中输出的具体错误原因
30.	0x2211	严重	查看 pthread 库在终端或日志中输出的具体错误原因
31.	0x2212	严重	查看 pthread 库在终端或日志中输出的具体错误原因
32.	0x2213	严重	查看 pthread 库在终端或日志中输出的具体错误原因
33.	0x2214	严重	查看 pthread 库在终端或日志中输出的具体错误原因
34.	0x2215	严重	查看 pthread 库在终端或日志中输出的具体错误原因
35.	0x2216	严重	查看 pthread 库在终端或日志中输出的具体错误原因

7. 维护设计

为了确保“玄鸟”架构的维护方便，设计中采取了多种策略和安排：

1. 模块化设计：“玄鸟”架构分解多个独立的软件配置项，各软件配置项又分解为多个模块，每个模块负责特定的功能。每个模块的职责清晰、接口明确、模块间耦合低，可以独立开发、测试和维护，提高了“玄鸟”架构的可维护性；

2. 代码规范和文档：

1）制定并遵循统一的代码格式、命名规则和编程风格。

2）在代码中添加必要的注释，编写详细的开发文档，包括设计文档、API 文档和用户手册等；

3）定期进行代码审查，确保代码质量符合规范。

3. 日志记录：

1）支持多种日志级别（如 DEBUG、INFO、WARNING、ERROR），根据需要选择合适的日志级别；

2）记录关键的操作步骤、错误信息等；

4. 配置管理：

1） 将配置参数放在外部文件中，如运行环境配置文件、模型配置文件、服务配置文件等，便于修改和管理；

2） 使用仿真配置终端进行自动化配置。

5. 测试：

1） 编写单元测试，确保每个模块的独立功能正确；

2） 编写集成测试，确保模块之间的协作正确以及各软件配置项之间的协作正确；

6. 监控：仿真监控终端能够实时监控“玄鸟”架构的运行情况，及时发现并处理问题；

7. 版本控制：使用版本控制系统管理代码和文档的版本，便于回溯和协作；

8. 代码重构：定期对代码进行重构，优化代码结构，提高代码的可维护性和性能；

9. 用户反馈：及时收集用户的意见和建议，不断对“玄鸟”架构进行优化。

通过以上这些设计安排，可以显著提高“玄鸟”架构的可维护性，减少维护成本，确保仿真的长期稳定运行。

8. 尚待解决的问题

无。

9. 需求的可追踪性

9.1. 正向追踪性

“玄鸟”架构的需求与各软件配置项的正向追踪关系如表 21 所示。

表 21 需求与软件配置项的正向追踪关系

序号	需求	软件配置项	说明
1.	模型封装与集成能力	仿真内核 模型系统 集成开发环境	模型封装人员根据仿真内核提供的标准化模型封装框架利用集成开发环境将离散的二进制数据包模型封装到模型系统中

2.	模型实时调度能力	仿真调度引擎 仿真内核 模型系统	仿真调度引擎用于加载仿真内核，使得仿真内核通过标准化模型封装框架中的接口多线程多频率地调度模型系统中的模型运行
3.	模型间数据交互能力	仿真内核 模型系统 Fast DDS	模型系统中的模型通过仿真内核提供的标准化模型封装框架的数据发布/订阅接口利用 Fast DDS 进行数据交互
4.	外部系统数据交互能力	服务系统	与外部系统通信的各类服务实现了与外部系统进行交互的功能，包括离散量的 UDP/TCP 通信服务、虚拟航空总线通信服务、控制指令解析服务等
5.	仿真监控能力	仿真监控终端 Fast DDS	仿真监控终端通过 Fast DDS 的发布/订阅机制进行模型交互数据读取和调试数据注入
6.	仿真可配置性	仿真配置终端 仿真内核	仿真配置终端能够管理与编辑配置文件，仿真内核根据配置文件中的参数配置仿真的运行
7.	接受外部系统指令控制能力	服务系统	控制指令解析服务能够进行外部系统控制指令的接收、解析、响应与回复
8.	快照拍摄和调用能力	服务系统 Fast DDS 数据库	快照服务能够对 Fast DDS 中交互的数据进行快照拍摄和调用。快照信息存储在数据库中，由快照服务读取和写入
9.	可视化交互界面	仿真监控终端 仿真调度终端 仿真配置终端	仿真监控终端是用于监控模型交互数据的界面调试工具，仿真运行终端用于启动与控制仿真，仿真配置终端用于软件配置文件的管理和修改
10.	调度实时性	仿真内核	仿真内核使用使用纳秒睡眠机制保证模型周期性调度的频率误差不超过 1 %
11.	响应实时性	服务系统	服务系统通过独立线程实时响应外部输入的数据/指令，实现响应时间不大于 50 ms
12.	监控实时性	仿真监控终端 Fast DDS	仿真监控终端通过 Fast DDS 实时获取模型交互数据并进行显示，实现数据更新频率不大于 1 Hz
13.	软件可靠性	仿真内核	仿真内核提供充分的出错处理设计及从错误中恢复的能力，保证加载全部二进制数据包模型后稳定运行时间不少于 100 小时
14.	软件兼容性	操作系统抽象层	对不同操作系统进行抽象，使软件具备跨操作系统能力

9.2. 反向追踪性

“玄鸟”架构的需求与各软件配置项的反向追踪关系如表 22 所示。

表 22 需求与软件配置项的反向追踪关系

序号	软件配置项	功能需求	说明
1.	仿真配置终端	仿真可配置性 可视化交互界面	仿真配置终端具有可视化的交互界面来编辑配置文件，从而配置仿真运行所需的各类参数
2.	仿真调度终端	可视化交互界面	仿真调度终端具有可视化的交互界面来控制仿真运行
3.	仿真调度引擎	模型实时调度能力	仿真调度引擎加载仿真内核进行实时仿真
4.	操作系统抽象层	软件兼容性	操作系统抽象层使软件具备跨操作系统能力
5.	仿真内核	模型封装与集成能力 模型实时调度能力 模型间数据交互能力 仿真可配置性 调度实时性 软件可靠性	仿真内核提供了模型封装框架用以封装与集成二进制数据包模型；仿真内核使用实时调度线程调度模型运行；仿真内核提供了 Fast DDS 通信的数据发布/订阅接口；仿真内核通过解析配置文件来设置仿真运行的各类参数；仿真内核使用使用纳秒睡眠机制保证模型周期性调度的频率误差不超过 1 %；仿真内核提供充分的出错处理设计及从错误中恢复的能力
6.	模型系统	模型封装与集成能力 模型实时调度能力 模型间数据交互能力	使用模型封装框架封装的模型集成于模型系统中供仿真内核调用；模型系统中的模型注册需要被调度的周期性执行函数供实时调度线程调度；模型系统中的模型使用 Fast DDS 通信的数据发布/订阅接口进行模型间数据交互
7.	服务系统	外部系统数据交互能力 接受外部系统指令控制能力 快照拍摄和调用能力 响应实时性	服务系统中的服务实现了与外部系统进行数据/指令交互的功能，包括离散量的 UDP/TCP 通信服务、虚拟航空总线通信服务、控制指令解析服务等，并且通过独立线程实时响应外部输入的数据/指令，实现响应时间不大于 50 ms；快照服务能够对 Fast DDS 中交互的数据进行快照拍摄和调用

8.	Fast DDS	模型间数据交互能力 仿真监控能力 快照拍摄和调用能力 监控实时性	Fast DDS 提供了基于共享内存的发布/订阅数据交互功能，实现了模型间数据的实时交互；Fast DDS 为仿真监控终端和快照拍摄/调用提供通信通道及实时性的支持；
9.	数据库	快照拍摄和调用能力	快照信息存储在数据库中，由快照服务读取和写入
10.	仿真监控终端	仿真监控能力 可视化交互界面 监控实时性	仿真监控终端具有可视化的交互界面以监视仿真运行及模型间数据交互，数据更新频率不大于 1 Hz
11.	集成开发环境	模型封装与集成能力	模型封装人员根据仿真内核提供的标准化模型封装框架利用集成开发环境将离散的二进制数据包模型封装到模型系统中

10. 总结

依据本软件概要设计说明书，开发团队可以使用自研的“玄鸟”架构封装与集成二进制数据包模型，最终形成一体化的二进制数据包软件。本软件概要设计说明书为一体化的二进制数据包软件的开发与维护提供全面指导。通过阐述软件架构、功能模块、处理流程、用户界面设计以及与其他系统的接口等关键信息，该文档确保开发团队对软件的整体结构和功能有清晰的理解，从而能够高效协作，实现软件的开发与稳定运行。同时，它也为后续的运行维护和功能扩展提供了明确的参考，确保软件能够适应不断变化的需求。