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：GYSJSM_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集成技术。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
XNSim	XuanNiao Simulation	“玄鸟”架构

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实现一系列关键接口，涵盖：二进制数据包模型间的通信接口、与外部系统的交互接口以及用户界面交互接口。

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，具有数据量庞大且需实时交互的特点。因此，一体化二进制数据包软件计划采用开源的数据分发服务 Fast-DDS 库，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），并可配置为提供实时功能，确保在指定的时间限制内做出响应，从而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，并实现了通用化、模板化的数据交互接口。模型通过该系统以发布/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 计算机间存在部分离散量需交互，需通过 UDP/TCP 实现离散量数据通信。

3．控制指令接口：一体化二进制数据包软件需通过 UDP/TCP 接收、响应并反馈接收到的外部系统发出的一系列控制指令。这些控制指令涵盖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主要由一个仿真综合管理平台前端软件构成。仿真综合管理平台作为用户与一体化二进制数据包软件进行交互的核心接口，扮演着至关重要的角色。该平台是一个基于 Web 技术的综合性管理平台，集成了众多功能模块，包括仿真运行、仿真监控、数据监控、模型开发、服务开发、配置管理以及用户管理等多个方面。通过这些功能模块，平台能够全面覆盖用户在使用一体化二进制数据包软件时的各项需求。

具体而言，仿真综合管理平台提供了直观且友好的用户交互界面，使得开发者和用户能够在软件运行过程中实时监控和调试所发送和接收的数据包。这不仅大大提升了数据处理的透明度和效率，还方便了用户对仿真过程的全面掌控。此外，平台还支持对仿真构型配置进行灵活管理，用户可以根据实际需求调整各项参数，确保仿真结果的准确性和可靠性。

在监控系统状态方面，仿真综合管理平台能够实时反馈系统的运行状况，及时发现并处理潜在问题，保障系统的稳定运行。同时，平台还提供了强大的模型和服务开发功能，用户可以在平台上进行模型的构建和服务的定制，满足不同场景下的应用需求。

3.2. 行为设计决策

一体化二进制数据包软件在开发过程中，必须实现集成功能，以便能够将多个二进制数据包模型无缝整合到软件系统中。此外，该软件还需具备实时调度的能力，确保各个二进制数据包模型能够根据实际需求高效地进行数据交互和处理。为了实现这一目标，软件系统还需设计一套完善的指令解析机制，以便能够准确接收并解析来自外部系统的指令控制信号。因此，在软件开发过程中，必须对模型集成的方式、模型调度的策略以及指令解析的具体实现等关键行为进行详细的设计和决策，以确保软件系统的稳定运行和高效性能。

3.2.1. 模型集成

一体化二进制数据包软件具备标准化的模型集成框架和自动化的模型集成模板生成功能，为开发人员在二进制数据包模型集成过程中提供了高效且统一的方法。负责模型集成与集成的开发人员，通过利用标准化集成框架所提供的丰富接口方法，能够以更便捷、高效的

方式完成二进制数据包仿真模型的集成工作。这使得模型可以顺利集成到“玄鸟”架构中，从而提升整体系统的性能和兼容性。标准化框架的应用不仅简化了开发流程，还确保了模型集成的一致性和稳定性，为“玄鸟”架构的扩展和优化奠定了坚实基础。

3.2.2. 模型调度

由于普通操作系统在 SpinLock 和 IRQ 上下文方面无法实现抢占，导致高优先级任务唤醒后不一定能立即执行，且不支持优先级反转，因此需要操作系统具备硬实时性。为此，Linux 系统需安装 RT_Preempt 补丁，Windows 系统则需安装 RTX 扩展，以确保操作系统能够处理优先级反转，并能实时执行高优先级任务。此外，通过配置操作系统的 CPU 隔离选项，可使特定 CPU 核心仅供一体化二进制数据包软件使用，避免其他进程竞争 CPU 资源。

一体化二进制数据包处理软件在具备硬实时性特性的操作系统环境中，通过充分利用 pthread 库的高效线程管理功能以及纳秒级精度的睡眠技术，实现了专门用于模型调度的仿真实时线程。这些线程能够精确地实现二进制数据包模型的实时调度机制，确保数据处理的高效性和实时性。在此过程中，每个线程都被赋予了独立设置其优先级及 CPU 亲和性的权限，可以根据实际需求灵活调整其内核调度策略。这种精细化的线程管理方式，不仅提升了系统的整体性能，还极大地增强了数据处理的稳定性和可靠性，确保了在高负载情况下依然能够保持优异的实时响应能力。

3.2.3. 指令解析

一体化二进制数据包软件具备强大的指令接收和处理能力，能够接受并处理来自外部系统的各种不同类型的控制指令。这些控制指令的传输和交互主要通过 UDP（用户数据报协议）和 TCP（传输控制协议）这两种网络协议来实现。

一体化二进制数据包软件实现了高效且精准的事件管理功能。具体而言，控制指令的执行采用了事件触发-响应的机制，即当特定事件被触发时，相应的控制指令会被激活并执行。

为了实现这一机制，控制指令在各需要响应的模块中进行了明确的定义，并且将这些模块的响应函数注册为对应的事件。指令解析服务在接收到外部系统指令后，会通过事件管理器根据事件的唯一标识符，将指令精准地发往特定的模块进行执行。这一过程中，只有经过验证且有效的控制指令才会被软件所执行，确保了系统的安全性和稳定性。

如果控制指令无效或存在错误，指令解析服务会立即识别并返回相应的错误信息，避免了无效指令对系统造成潜在影响。此外，指令解析服务对指令的解析非常灵活，支持外部系

统使用进程名、线程名、服务名或模型名等通配符进行发送。这意味着，任何满足通配符条件的进程、线程、服务或模型都将接收到并执行给定的指令，极大地提升了指令的覆盖范围和执行效率。

3.3. 数据库/数据文件设计决策

一体化二进制数据包软件使用 SQL 关系型数据库管理构型配置、接口配置、模型配置、服务配置等其它数据。

构型配置表定义了软件运行所需的环境参数，涵盖构型 ID、构型名称、操作系统名称、操作系统版本、CPU 亲和性、日志记录等级、调度线程参数、加载模型列表、加载服务列表等关键要素。这些参数确保软件在不同环境下能够正确运行并精准记录日志。软件通过读取构型配置表，完成仿真的初始化设置，配置相应的运行环境，从而实现不同构型间的灵活切换。

接口配置表定义了二进制数据包模型的交互接口参数，涵盖接口名称、接口数据类型、接口数据大小等关键信息。这种设计保证了软件能够依托接口配置表，自动构建模型间进行 Fast-DDS 交互的数据交互接口。

模型配置表定义了模型开发与运行的各项细节，涵盖名称、版本、所属构型、运行频率、运行节点、二进制数据包模型信息、输入输出接口信息及控制指令等内容。该设计确保软件能够基于模型配置表，自动生成模型集成模板代码，并在仿真运行过程中提供模型调度所需的必备参数。

服务配置表定义了服务开发与运行的各项细节，涵盖名称、版本及控制指令等内容。该设计确保软件能够基于服务配置表，自动生成服务开发模板代码，并在仿真运行过程中提供服务加载所需的必备参数。

一体化二进制数据包软件需借助数据库支持快照的拍摄与调用。鉴于模型间通过 Fast-DDS 进行数据交互，因此可利用 Fast-DDS 自带的 Record and Replay 数据持久化工具，将快照数据存储为 MCAP 格式的数据库文件。生成的 MCAP 文件可被任何兼容工具读取，其中包含读取和重放快照数据所需的所有信息。

此外，数据库中配备了一系列用于系统管理的数据表，涵盖机型数据表、ATA 章节数据表、模型名称数据表、系统日志数据表、版本管理数据表、用户信息数据表、出错处理数据表等。

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。远程参与者的身份验证通过证书机制实现，确保每个通信参与者的身份真实可信。实体的访问控制则通过设置不同的权限级别，限制对敏感数据和功能的访问，从而防止未授权的操作。数据加密则采用先进的加密算法，对传输的数据进行加密处理，即使数据在传输过程中被截获，也无法被轻易解密，有效保障了数据的机密性。这些安全机制的组合使用，为一体化二进制数据包软件提供了全面的安全保护。

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。日志记录涵盖了软件运行的关键事件、操作记录、异常信息等，为问题排查和性能分析提供了详实的数据支持。出错告警信息则通过明确的提示和错误代码，快速定位问题所在，便于维护人员迅速采取措施进行修复。这种全面的日志和告警机制，不仅提升了软件的可靠性和稳定性，还大大缩短了问题响应和解决的时间，确保了仿真任务的高效执行。

一体化二进制数据包软件采用了基于非对称加密技术的 OpenSSL 数字签名验证机制，该机制涉及一对密钥：私钥与公钥。私钥负责生成签名，而公钥则用于对签名进行验证。这种技术的运用旨在防范供应链攻击、代码注入攻击以及 DLL 劫持攻击。这些攻击手段通常通过替换或伪造动态链接库文件，向程序中注入恶意代码或劫持程序的执行流程。利用 OpenSSL 数字签名技术，可以显著提升动态链接库的安全性。通过恰当的实施与管理，能够有效地阻止动态链接库遭受恶意替换。

3.5. 灵活性设计决策

一体化二进制数据包软件通过设计和实现一个高效的操作系统抽象层，成功地将对底层操作系统的依赖性大幅简化。这一技术手段使得应用程序在跨平台运行时，能够极大地减少因操作系统差异而必须进行的繁琐修改工作。具体而言，该软件通过抽象层提供的统一接口和功能封装，屏蔽了不同操作系统在底层实现上的复杂性，从而确保了应用程序的兼容性和可移植性。这样一来，开发者在开发跨平台应用时，无需再针对每个操作系统进行逐一适配和优化，极大地提升了开发效率和降低了维护成本。最终，应用程序得以在多样化的操作系

统环境中顺畅运行，用户体验也得到了显著提升。

一体化二进制数据包软件采用了开源的跨平台自动化构建系统——CMake，来全面负责项目工程的管理工作。CMake 以其强大的功能和灵活性，能够高效地处理项目中的各种构建任务，确保项目工程在不同操作系统和硬件架构上都能顺利进行构建。通过使用 CMake，一体化二进制数据包软件不仅实现了跨平台的工程构建，还极大地提升了项目管理的便捷性和可维护性，使得开发者在面对多样化的开发环境时，能够轻松应对，确保项目的稳定性和一致性。无论是 Windows、Linux 还是 macOS 等操作系统，CMake 都能提供一致且可靠的构建体验，从而极大地简化了跨平台开发的复杂度，提升了开发效率。

一体化二进制数据包软件采用了插件化开发方法，这种开发方式使得软件在构建和扩展功能时更加灵活和高效。通过将不同的功能模块设计成独立的插件，一体化二进制数据包软件能够根据用户的具体需求，灵活地动态加载或卸载相应的插件模块，从而实现功能的定制化和模块化管理。这种插件化开发方法不仅提高了软件的可维护性和可扩展性，还极大地提升了软件的开发效率和用户体验。

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：能够灵活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．综合管理交互界面：能够进行仿真运行配置、仿真运行控制、仿真状态监控、仿真数据监控等功能的综合管理交互界面。

10．可扩展性：系统应具备可扩展的功能，以适应未来对系统功能的添加需求。

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：仿真监控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：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

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（XNCore），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、数据交互接口管理、构型配置、加载模型配置、加载服务配置、事件管理，并提供数据交互接口框架、服务框架、模型集成框架等；

2．“玄鸟”架构包含一个数据交互接口（XNDDSInterface），基于仿真内核提供的数据交互接口框架建立所有交互的 FastDDS 通信接口、UDP/TCP 通信接口及数据监控接口。数据交互接口使仿真内核不依赖于具体的接口数据定义，这对接口数据的更改提供了灵活

性；

3．“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（XNModels），其中的模型基于仿真内核提供的模型集成框架进行集成。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；

4．“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（XNServices），其中的服务基于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；

5．“玄鸟”架构包含一个用于加载与控制仿真内核运行的引擎（XNEngine）。仿真调度引擎是仿真系统的主进程，负责加载仿真内核并控制仿真的初始化、启动、暂停、继续与终止；

6．“玄鸟”架构包含一个仿真综合管理平台及配套的后端服务，提供直观的用户交互界面（XNSimPortal）。该平台具备构型配置、接口配置、模型配置、服务配置、系统运行状态监控、交互数据监控、交互数据注入以及交互数据采集等核心功能。同时，仿真综合管理平台还需提供一系列用户友好功能，包括模型集成与服务开发模板代码的生成、编译、上传及下载；

7．“玄鸟”架构设计了一个操作系统抽象层（XNOSAL），这降低了应用程序对底层操作系统的直接依赖性。通过这一抽象层的介入，应用程序得以在不同的操作系统平台上运行，而无需针对每个操作系统进行大量修改工作。这种设计不仅提升了应用程序的跨平台兼容性，还极大地简化了开发和维护流程，使得开发者能够更加高效地应对多样化的操作系统环境。

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仿真构型管理、插件式开发、PIMPL 设计模式、模型调度算法和基于发布/订阅的共享内存数据交互。

4.1.2.2.1. 仿真构型管理

仿真构型管理是指对仿真系统运行所必需的各类构型数据进行全面管理的一项技术，这些构型数据包括但不限于运行参数、模型参数、服务参数以及接口参数等。具体而言，仿真构型管理涉及对这些数据进行严格的版本控制，确保每个版本的准确性和可追溯性；同时，还需要对数据进行有效性标识，以便在仿真过程中能够快速识别和验证数据的合法性；此外，

动态加载技术也是仿真构型管理的重要组成部分，它允许系统在运行时根据需要灵活加载不同的构型数据，从而提高仿真的灵活性和适应性。

构型的本质可以理解为系统运行的“蓝图”，它详细规定了仿真系统的结构、行为以及数据流的组织和传递方式。换句话说，构型就像是仿真系统的“设计图”，决定了系统在仿真过程中如何运作，包括各个模块之间的交互、数据的流向以及系统的整体性能。

在“玄鸟”架构中，通过以下方式管理构型数据：

1. 构型的基本配置参数通过数据库统一管理，包括：

   1） 仿真运行环境参数：操作系统名称、版本等;

   2） 仿真运行参数：CPU 亲和性、DDS 通信域等；

   3） 日志记录参数；

   4） 调度线程参数：运行频率、优先级、CPU 亲和型等；

   5） 加载模型参数：模型名称、模型运行频率、模型运行节点等；

   6） 加载服务参数：服务名称、服务其它配置参数等；

   7） 数据交互接口定义：接口名称、接口所属结构体名称、接口数据大小等。

2. 基于数据库中构型的基本配置参数，在仿真运行工作目录下构建构型目录，构型目录中存储以下内容：

   1） IDL 目录：存储构型的数据交互接口 IDL 文件、数据交互接口自动化构建的源码；

   2） ModelProjects 目录：存储构型的模型集成源码；

   3） Models 目录：存储构型集成完成的模型动态连接库；

   4） Packages 目录：存储二进制数据包动态库及头文件；

   5） PluginCode 目录：存储用于数据监控的数据监控器自动化构建的源码；

   6） Plugins 目录：存储用于数据监控的数据监控器动态连接库。

3. 在仿真运行过程中，根据以下过程实现构型的动态加载：

   1） 从数据库中读取基本配置参数进行仿真初始化；

   2） 根据加载服务参数在仿真根目录中动态加载对应的服务并配置服务参数；

   3） 在构型目录中动态加载模型、数据交互接口、数据监控器等动态插件。

通过这种构型管理技术，用户可以根据实际需要配置构型，大大提高了仿真系统的灵活性和可扩展性。此外，“玄鸟”架构还提供了完善的构型管理用户交互页面，使得用户可以通过这些交互页面对构型数据进行查询、修改和删除等操作。

4.1.2.2.2. 插件式开发

插件式开发技术是一种模块化软件架构模式，通过动态加载独立功能模块（插件）扩展核心应用功能，实现“即插即用”的灵活扩展能力。核心系统仅包含最小化基础功能，不涉及具体业务逻辑。插件模块为独立的功能单元，通过预定义接口与核心系统交互。插件式开发技术的设计原则为：

1. 开放封闭原则：核心系统对扩展开放（支持新插件），对修改封闭（无需改动核心代码）；

2. 依赖倒置原则：核心系统与插件均依赖抽象接口，而非具体实现。

插件式开发技术具有以下优势：

1. 功能能够模块化与解耦，不同团队可以并行开发插件，也能够实现插件的热插拔；

2. 其它开发者也可根据抽象接口自主开发插件；

“玄鸟”架构的数据交互接口、模型系统、服务系统以及 DDS 监控后端服务均采用了插件式开发技术。不同的构型对应不同的接口数据定义，因此，数据交互接口与 DDS 监控后端服务均需解耦与接口数据强耦合的功能模块。“玄鸟”架构在实现过程中预留了这些强耦合模块的插件抽象接口，并提供了插件自动化构建功能（即数据交互接口生成后端服务、DDS 监控后端服务中的数据监控器插件生成模块）。在实际使用过程中，系统可根据数据库中存在的数据，自动化生成插件代码并编译发布，仿真运行时能够动态加载这些插件并完成对应功能。

同理，不同构型亦对应不同的模型与服务，因此，模型系统与服务系统同样采用插件式设计，并提供了自动化构建功能（即模型集成后端服务与服务开发后端服务）。用户可通过自动化构建功能获取插件模板，再根据自身需求完善其功能。在仿真运行中，系统可动态加载模型与服务，以实现所需功能。

4.1.2.2.3. 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。

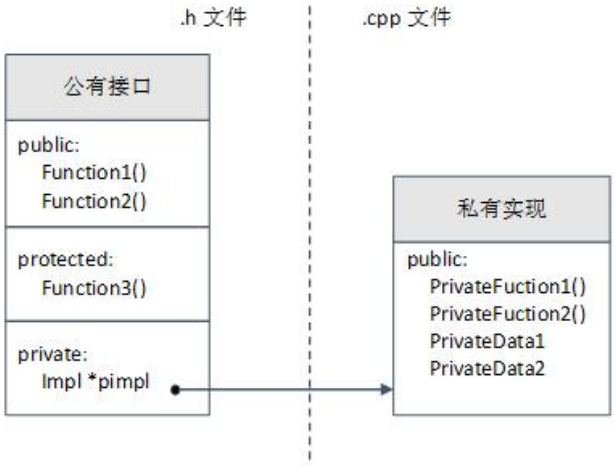

图 1　PIMPL 设计模式

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。

4.1.2.2.4. 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。

4.1.2.2.5. 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集成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。

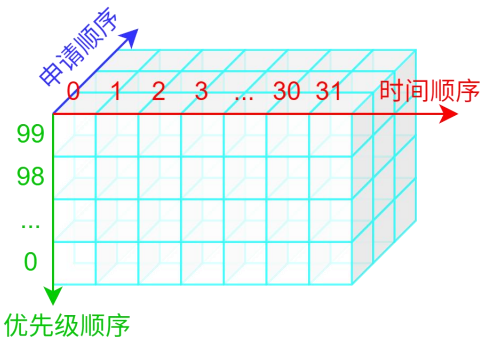

图 2 线程调度任务表

图中每个蓝色方块为一个任务，每个方块在时间顺序上的宽度表示其在当前线程的一个时间片（即 $1/f$ 秒）内执行，f为线程运行频率，具体为：

1．表的第一维长度固定为 32。由于模型在线程中的运行频率共有 6 种类型：f、$f/2$、$f/4$、$f/8$、$f/16$、$f/32$，因此 32 长度表正好能满足 6 种模型的运行频率。用户可以在模型配置中指定某个模型在其所属线程的频率类型。

举例说明：线程的调度表在时间顺序方向有 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），线程可以调度执行需要以f运行或不高于f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：若有一个线程（运行周期为$1/f$），需要调度一个以$f/2$运行的模型时，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，高优先级任务列表先执行，低优先级任务列表后执行。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。假设当前有一个以f频率运行的线程，需要在该线程中依次提交 3 个以f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$f/2$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 $m = 1,3,5,\ldots,31$。

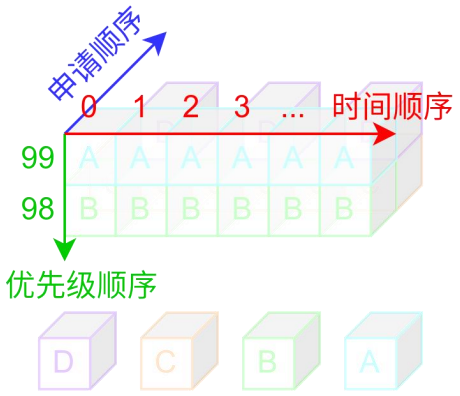

图 3 线程调度任务表填写示例

调度表填写完成后的调度表如图 3 所示，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 4 所示。

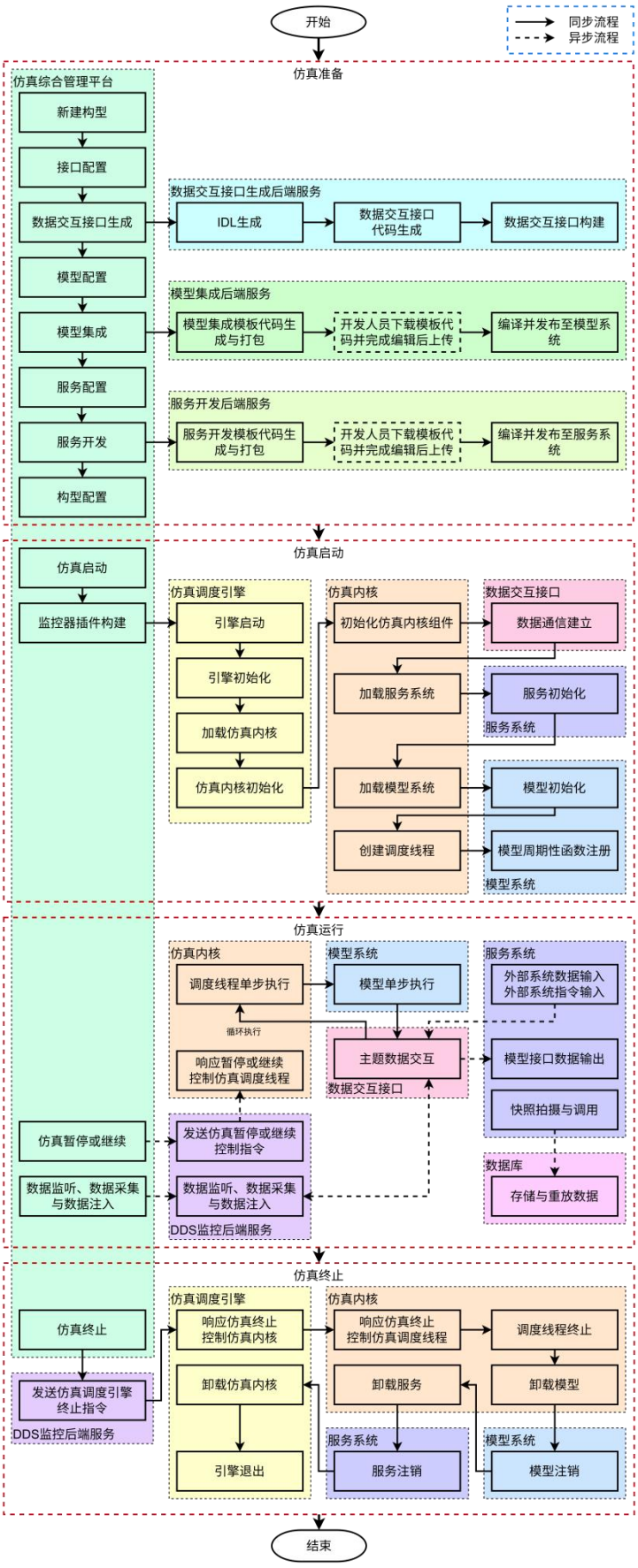

图 4 “玄鸟”架构基本流程图

“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）使用仿真综合管理平台创建构型，配置构型基础参数；

2）使用仿真综合管理平台配置接口，接口定义支持从 ICD 文件导入；

3）通过仿真综合管理平台创建数据交互接口。在此过程中，仿真综合管理平台调用后端服务根据接口定义来构建数据交互接口。该后端服务从数据库中读取接口数据，生成 IDL 文件，并利用 FastDDS 生成 DDS 通信接口代码。随后，进一步生成用于模型交互、TCP/UDP 通信及数据监控所需的接口代码。最终，执行代码的编译、构建与发布，形成当前构型的数据交互接口。。

4）通过仿真综合管理平台创建并配置模型，需要配置模型集成模板代码生成以及模型调度运行所需的各类参数；

5）通过仿真综合管理平台集成、构建并发布模型。在此过程中，仿真综合管理平台调用后端服务根据模型配置生成模型集成模板代码，并提供模型集成的开发人员下载。开发人员对模型集成代码进行修改完善后上传代码。接着，后端服务对模型代码进行构建并发布至模型系统中；

6）通过仿真综合管理平台创建并配置服务，需要配置服务开发模板代码生成以及服务加载运行所需的各类参数；

7）通过仿真综合管理平台开发、构建并发布服务。在此过程中，仿真综合管理平台调用后端服务根据服务配置生成服务开发模板代码，并提供给服务开发人员下载。开发人员对服务代码进行功能开发后上传代码。接着，后端服务对服务代码进行构建并发布至服务系统中；

8）通过仿真综合管理平台对构型进行进一步的配置，设置构型需要加载的服务以及需要调度的模型。

2．仿真启动阶段：

1） 通过仿真综合管理平台启动仿真；

2） 仿真综合管理平台会自动开始构建数据监控器。此时，仿真综合管理平台会调用后端服务生成 DDS 监控后端服务所需的数据监控器插件代码，并进行构建与发布；

3） 仿真综合管理平台启动仿真调度引擎，记录引擎进程 ID，并与仿真调度引擎

建立 SSE 连接以获取引擎的控制台输出信息；

4） 仿真调度引擎启动；

5） 仿真调度引擎进行初始化设置（例如设置日志记录等级、引擎进程的 CPU 亲和性等）；

6） 仿真调度引擎加载仿真内核动态库；

7） 仿真调度引擎调用仿真内核的初始化；

8） 仿真内核完成其内部组件的初始化工作；

9） 仿真内核与数据交互接口建立 DDS 数据通信；

10） 仿真内核加载服务系统中的所有服务；

11） 服务系统中的服务进行初始化；

12） 仿真内核加载模型系统中的所有模型；

13） 模型系统中的模型进行初始化；

14） 仿真内核创建调度线程；

15） 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. 模型通过数据交互接口进行主题数据交互；

d. 以上流程循环执行。

2） 异步流程：

a. 用户通过仿真综合管理平台能够控制仿真的暂停与继续，此时仿真综合管理平台通过 DDS 监控后端服务向仿真调度引擎发出控制指令；

b. 仿真调度引擎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. 仿真综合管理平台通过调用 DDS 监控后端服务使用数据交互接口进行交互数据的监听、注入与采集；

e. 服务系统会响应外部系统输入的指令或数据，通过数据交互接口发布至指令或数据到目的地；

f. 服务系统会接收到需要向外部输出的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） 用户通过仿真综合管理平台下达仿真终止指令，DDS 监控后端服务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） 仿真调度引擎退出；

10） 仿真综合管理平台关闭与仿真调度引擎的 SSE 连接，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 5 所示。

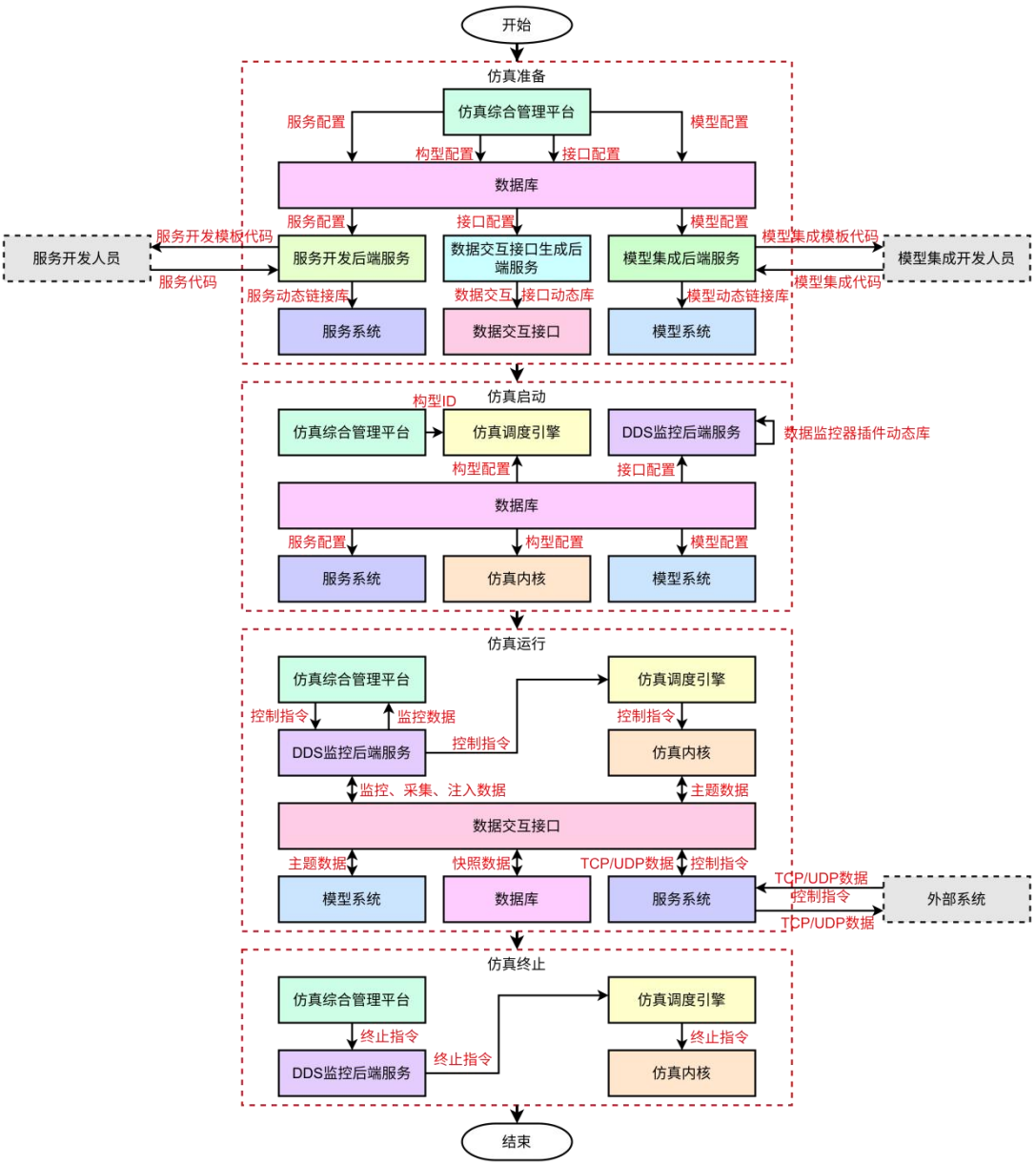

图 5 “玄鸟”架构数据流程图

“玄鸟”架构中的交互数据流程为：

1．仿真准备阶段：

1） 仿真综合管理平台将构型配置、接口配置、模型配置、服务配置存储于数据库中；

2） 数据交互接口生成后端服务读取数据库中的接口配置构建数据交互接口动态库；

3） 模型集成后端服务读取数据库中的模型配置并生成模型集成模板代码；

4） 模型集成开发人员下载模型集成模板代码并进行进一步的完善开发，然后将完成的模型集成代码上传；

5） 模型集成后端服务对上传模型集成代码进行构建，生成集成后的模型动态库并发布至模型系统中；

6） 服务开发后端服务读取数据库中的服务配置并生成服务开发模板代码；

7） 服务开发人员下载服务开发模板代码并进行进一步的完善开发，然后将完成的服务代码上传；

8） 服务开发后端服务对上传的服务代码进行构建，生成服务动态库并发布至服务系统中。

2．仿真启动阶段：

1） DDS 监控后端服务读取数据库中的接口数据生成监控交互数据所需的数据监控器插件的动态库；

2） 仿真综合管理平台将当前运行仿真的构型 ID 传递给仿真调度引擎；

3） 仿真调度引擎读取数据库中的构型配置并设置自身运行参数；

4） 仿真内核读取数据库中的构型配置并设置自身运行参数，同时根据构型配置加载模型的动态链接库和服务的动态链接库；

5） 模型初始化时读取数据库中的模型配置进行参数配置；

6） 服务初始化时读取数据库中的服务配置进行参数配置。

3．仿真运行阶段：

1） 仿真内核、模型系统、服务系统、DDS 监控后端服务及存储快照的数据库均通过数据交互接口进行 DDS 主题数据、TCP/UDP 数据、控制指令的交互；

2） 仿真综合管理平台从 DDS 监控后端服务获取监控数据，并通过 DDS 监控后端服务向仿真调度引擎发送仿真运行控制指令；

3） 仿真调度引擎接收到控制指令后将控制指令发送至仿真内核并进行仿真运行控制；

4） 外部系统输入的 TCP/UDP 数据或控制指令由服务系统进行接收处理，并通过数据交互接口发送至对应的模块；

5） 服务系统从数据交互接口获取 TCP/UDP 数据，发送至外部系统。

4．仿真终止阶段：

1）仿真综合管理平台通过 DDS 监控后端服务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，仿真终止。

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 6 所示：

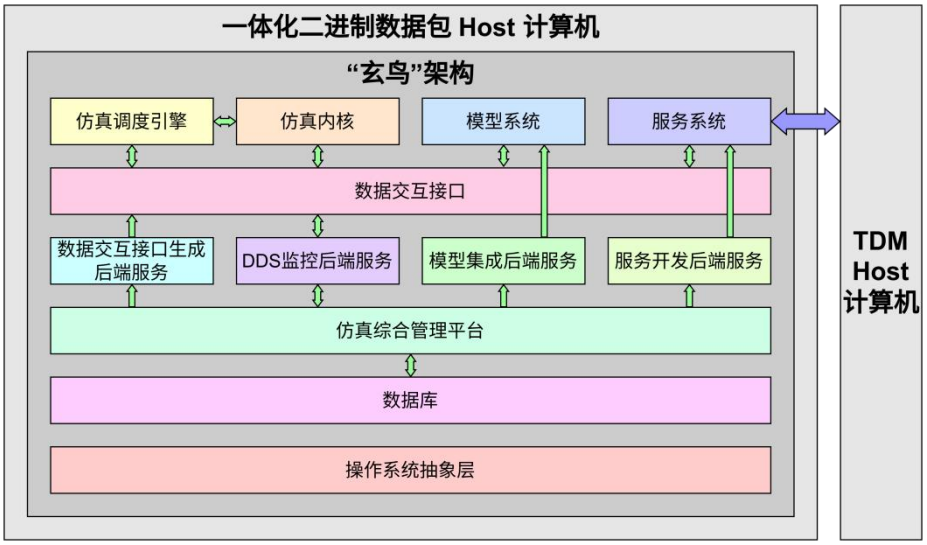

图 6 “玄鸟”架构组成结构图

“玄鸟”架构由以下几个软件配置项组成：

1. **操作系统抽象层**：标识符 XNOSAL。操作系统抽象层将与操作系统耦合的数据结构及 API 接口进行抽象，为“玄鸟”架构提供统一的 API 接口，供其它模块调用，简化了软件的开发和维护工作。它屏蔽了底层操作系统的复杂性和差异性，使得其它模块能够更加专注于业务逻辑的实现。同时，操作系统抽象层还增强了软件的移植性和可扩展性，为软件的长期发展和升级提供了有力支持。；

2. **数据库**：标识符 XNDatabase。数据库用于存储构型配置、接口配置、模型配置、服务配置等“玄鸟”架构仿真所需的关键数据。用户可通过仿真综合管理平台对这些配置进行编辑，而其他模块则会从数据库中读取配置以进行初始化操作。此外，数据库还负责存储仿真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或写入快照数据。

3. **仿真综合管理平台**：标识符 XNSimPortal。仿真综合管理平台提供了一个可视化前端界面，能够对“玄鸟”架构进行灵活配置，并执行仿真运行控制、运行状态监控，以及仿真数据的实时监控、采集与注入等多种操作；

4. **仿真调度引擎**：标识符 XNEngine。仿真调度引擎是“玄鸟”架构仿真的主进程，主要负责仿真内核的加载、初始化与运行，同时通过数据交互接口接受控制指令以控制仿真运行；

5. **仿真内核**：标识符 XNCore。仿真内核主要负责“玄鸟”架构仿真运行的构型配置、服务加载与管理、模型加载与管理、模型周期性调度、调度线程管理、数据交互接口管理、运行日志记录等工作；

6. **数据交互接口生成后端服务**：标识符 XNInterfaceGenServer。数据交互接口生成后端服务用于自动化生成数据交互接口。该后端服务由仿真综合管理平台驱动，依据数据库中存储的接口配置数据，动态构建能够灵活支撑模型间数据交互、UDP/TCP 数据传输以及前端交互数据监控的数据交互接口；

7. **数据交互接口**：标识符 XNDDSInterface。数据交互接口基于仿真内核所提供的接口框架，并结合构型实际数据接口的定义，由数据交互接口生成后端服务自动构建而成。该接口由仿真内核动态加载，旨在为其他模块提供统一的 FastDDS 主题数据交互接口、TCP/UDP 数据交互接口以及数据监控交互接口；

8. **模型集成后端服务**：标识符 XNModelGenServer。模型集成后端服务用于自动化集成模型。该后端服务由仿真综合管理平台驱动，基于数据库中存储的模型配置数据，动态生成二进制数据包模型的集成模板代码，并支持用户下载压缩后的模板代码。同时，该服务提供自动化模型编译功能，可自动将用户上传的模型集成代码解压后编译并发布至模型系统中

9. **模型系统**：标识符 XNModels。模型系统是模型集成后端服务利用仿真内核提供的模型集成框架，进行定制化集成与开发的，由仿真内核动态加载的，需要周期性调度的模型或其他功能模块的集合。仿真内核通过构型配置中的配置项，加载需要调度运行的模型。模型系统不仅能够加载数据包模型，还可以加载其他需要周期性调度的任务模型，例如数据处理模型，它能处理数据包模型产生的数据，供仿真内核或其他模块使用；

10. **服务开发后端服务**：标识符 XNServiceGenServer。服务开发后端服务用于自动化开发服务。该后端服务由仿真综合管理平台驱动，基于数据库中存储的服务配置数据，动态生成服务的开发模板代码，并支持用户下载压缩后的模板代码。同时，该服务提供自动化服务编译功能，可自动将用户上传的服务代码解压后编译并发布至服务系统中；

11. **服务系统**：标识符 XNServices。服务系统是基于仿真内核提供的服务框架进行定制化开发的，由仿真内核动态加载的可扩展功能的服务模块集合。仿真内核通过构型配置中的配置项来加载所需的服务。服务系统中加载的服务能够执行与外部系统间的离散量通信、虚拟航空总线通信、控制指令解析与执行、快照拍摄及调用及其它定制化功能等多项功能；

12. **DDS 监控后端服务**：标识符 XNMonitorServer。DDS 监控后端服务用于监控系统运行与监控交互数据。该后端服务由仿真综合管理平台驱动，负责向仿真综合管理平台提供监

控数据，并具备仿真运行控制、仿真数据采集以及向仿真系统中注入数据的功能；

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 7 所示：

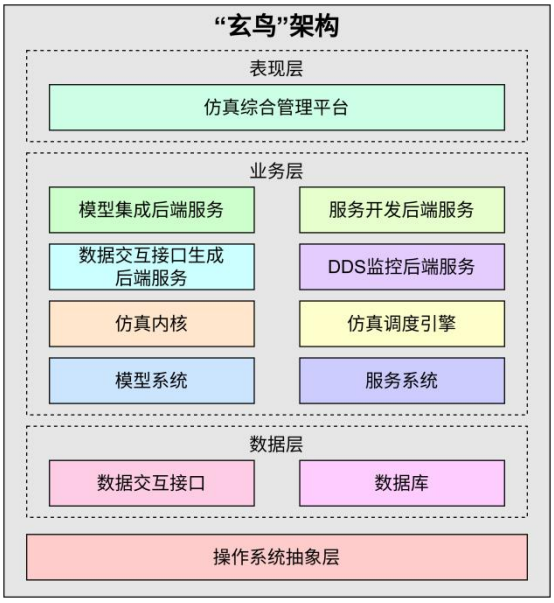

图 7 “玄鸟”架构软件层次结构图

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。

数据层负责数据的传输与存储，具体包括以下几项功能：

1) 数据交互接口：基于 Fast DDS 的共享内存发布/订阅机制数据交互、TCP/UDP 数据交互、以及监控数据的提取与注入。

2) 数据库：存储了“玄鸟”架构配置及运行仿真所需的所有数据，并利用 Fast DDS 的 Record and Replay 功能进行快照数据的存取。

业务层负责软件的基本业务逻辑处理，具体包括以下几项功能：

1) 仿真调度引擎负责加载仿真内核以进行仿真操作；

2) 模型系统负责加载数据包模型；

3) 仿真内核负责实时调用模型系统中的数据包仿真模型，并处理其他相关事件；

4) 服务系统负责处理外部系统指令的响应，并通过数据交互接口进行数据处理；

5) DDS 监控后端服务从数据交互接口提取和注入监控数据；

6) 数据交互接口服务用于自动化生成数据交互接口；

7) 模型集成后端服务用于自动化模型集成流程；

8) 服务开发后端服务用于自动化服务开发流程。

表现层主要是仿真综合管理平台，其功能在于向用户提供丰富的可视化操作界面，以支持模型集成、服务开发、仿真配置、仿真运行以及仿真监控等操作。

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. 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。

4.1.4.3.2. 数据库

“玄鸟”架构使用数据库保存以下信息：

1. 构型配置：构型 ID、构型名称、操作系统名称、操作系统版本、实时补丁版本、CPU 亲和性、DDS 通信域 ID、控制台输出等级、日志记录等级、调度线程名称、调度线程频率、调度线程优先级、加载模型列表、加载服务列表等；

2. 接口配置：接口名称、所属构型、接口所属结构体名称、接口类型、接口数据大小、接口含义等；

3. 模型配置：模型名称、版本号、所属构型、作者、描述、创建时间、修改时间、运行频率分组、运行节点、运行优先级、二进制数据包模型相关参数、指令列表等；

4. 服务配置：服务名称、版本号、作者、描述、创建时间、修改时间、指令列表、其它参数等；

5. 快照数据：Fast DDS Record and Replay 保存的 MCAP 格式的数据库中的快照数据；

6. 用户信息：用户 ID、用户名、密码、权限、其它信息；

7. 错误信息：错误码、错误消息、错误位置、解决方法等；

8. 其它数据：机型表、ATA 章节编号表、模型类型表、更新记录表等。

4.1.4.3.3. 仿真综合管理平台

仿真综合管理平台是一个 web 端的可视化前端界面，包括以下功能：

1. 权限管理相关功能：用户登录/登出、用户注册、用户管理、权限控制等；

2. 系统信息功能：系统状态、版本信息、更新记录、使用帮助、问题解答等；

3. 构型配置功能：构型选择、构型参数配置、构型加载模型配置、构型加载服务配置等；

4. 接口配置功能：接口参数配置、接口参数查询、ICD 导入、数据交互接口生成等；

5. 模型集成功能：模型参数配置、二进制数据包模型上传、模型集成模板代码下载、模型集成代码上传、模型代码构建与发布等；

6. 服务开发功能：服务参数配置、服务开发模板代码下载、服务开发代码上传、服务代码构建与发布等；

7. 仿真运行功能：仿真测试运行、仿真运行、仿真运行控制、仿真运行日志查看等；

8. 资源监控功能：监控仿真运行时的计算机资源占用等；

9. 仿真运行监控功能：监控仿真运行情况，包括：仿真运行状态、各调度线程 ID、各调度线程运行频率等；

10. 模型运行监控功能：监控模型运行情况，包括：模型运行状态、模型 ID、模型运行频率等；

11. 数据监控功能：监控模型间交互数据、模型间交互数据注入、通过 CSV 文件注入模型间交互数据、图表展示监控数据等；

12. 数据采集功能：上传采集脚本、模型间交互数据采集、加载采集结果并通过图表展示采集数据等；

13. 网络监控功能：抓取特定 IP、端口的 UDP/TCP 数据包并显示等；

14. QTG 功能：支持仿真模型的 QTG 测试。

4.1.4.3.4. 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构型解析功能完成数据库中构型配置的解析；

3. 能够以测试模式调用仿真内核，检验构型配置是否正确，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. 能够接收并响应 DDS 监控后端服务发出的暂停、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；

4.1.4.3.5. 仿真内核

1. 能够以动态库的形式被仿真调度引擎正确加载；

2. 能够读取数据库完成构型配置的解析；

3. 能够以测试模式进行构型配置、模型配置、服务配置的解析，检验数据库中配置的正确性；

4. 能够以测试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. 能够根据构型配置中列出的加载模型与加载服务，加载对应的模型动态库与服务动态库；

6. 能够解析数据库中已加载模型的模型配置，设置模型参数；

7. 能够解析数据库中已加载服务的服务配置，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. 能够通过数据交互接口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. 提供标准化的模型集成框架，提供初始化接口、周期性调度接口及数据交互接口等；

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

4.1.4.3.6. 数据交互接口生成后端服务

1. 提供能够被仿真综合管理平台调用的接口；

2. 能够根据构型 ID 从数据库中读取对应构型的接口配置；

3. 能够根据接口配置生成 FastDDS 通信所需的 IDL 文件；

4. 能够使用 IDL 文件生成 FastDDS 通信所需的主题接口定义代码；

5. 能够根据仿真内核提供的数据交互抽象接口生成适合实际接口配置的数据交互接口代码；

6. 能够生成构建配置文件；

7. 能够自动构建并发布指定的目录供仿真内核加载。

4.1.4.3.7. 数据交互接口

数据交互接口由数据交互接口生成后端服务自动化生成，它实现了实际接口配置的数据交互接口，具有一下功能：

1. 提供 FastDDS 主题数据的发布/订阅接口；
2. 提供 TCP/UDP 数据包的打包与解包；
3. 提供仿真监控所需的字符串形式数据交互接口。

4.1.4.3.8. 模型集成后端服务

1. 提供能够被仿真综合管理平台调用的接口；
2. 能够生成模型集成模板代码，包括：模型代码文件、模型构建配置文件等；
3. 能够压缩/解压缩模型集成代码；
4. 能够自动构建并发布模型集成代码。

4.1.4.3.9. 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集成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集成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集成框架提供的数据交互接口进行数据交互；
5. 能够进行其它相关数据处理。

4.1.4.3.10. 服务开发后端服务

1. 提供能够被仿真综合管理平台调用的接口；
2. 能够生成服务开发模板代码，包括：服务代码文件、服务构建配置文件等；
3. 能够压缩/解压缩服务开发代码；
4. 能够自动构建并发布服务开发代码。

4.1.4.3.11. 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；快照信息通过数据交互接口提供给仿真内核及其它模型；

4.1.4.3.12. DDS 监控后端服务

1. 提供能够被仿真综合管理平台调用的接口；

2. 能够通过 FastDDS 进行数据发布/订阅；

3. 能够根据数据库中的接口配置，自动化生成数据监控器插件的代码并构建；

4. 能够动态加载自动化构建的数据监控器；

5. 能够订阅仿真运行状态主题数据，并由仿真综合管理平台获取；

6. 能够订阅模型运行状态主题数据，并由仿真综合管理平台获取；

7. 能够通过数据监控器插件从数据交互接口获取字符串形式的监控数据，并由仿真综合管理平台获取；

8. 能够接收仿真综合管理平台的指令，通过数据监控器插件单步或连续地向数据交互接口注入数据；

9. 能够接收仿真综合管理平台的指令，从 CSV 文件读取数据通过数据监控器插件持续向数据交互接口注入数据；

10. 能够按照仿真综合管理平台提供的数据采集脚本，按照脚本中定义的采集数据、采集时间、采集频率等通过数据监控器插件从数据交互接口进行数据采集并保存为 CSV 格式的数据文件。

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.	操作系统抽象层	XNOSAL	V1.0
2.	数据库	XNDatabase	V1.0
3.	仿真综合管理平台	XNSimPortal	V1.0
4.	仿真调度引擎	XNEngine	V1.0
5.	仿真内核	XNCore	V1.0
6.	数据交互接口生成后端服务	XNInterfaceGenServer	V1.0
7.	数据交互接口	XNDDSInterface	V1.0
8.	模型集成后端服务	XNModelGenServer	V1.0
9.	模型系统	XNModels	V1.0
10.	服务开发后端服务	XNServiceGenServer	V1.0
11.	服务系统	XNServices	V1.0
12.	DDS 监控后端服务	XNMonitorServer	V1.0

4.2.2. XNOSLayer

4.2.2.1. 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

4.2.2.2. 结构设计

操作系统抽象层的结构如图 8 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 8 操作系统抽象层结构图

4.2.2.3. 模块设计

4.2.2.3.1. 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。

4.2.2.3.2. 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。

4.2.2.3.3. 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。

4.2.2.3.4. 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；

4.2.2.3.5. 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；

4.2.2.3.6. 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。

4.2.2.3.7. 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。

4.2.2.3.8. 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。

4.2.2.3.9. 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。

4.2.3. XNDatabase

4.2.3.1. 概述

1．标识符：XNDatabase

2．软件配置项名称：数据库

3．开发状态：已有的开源数据库容器文件（SQLite 3）

4．版本：SQLite 3

4.2.3.2. 结构设计

数据库包含的数据表如表 3 所示。

表 3 数据库数据表定义

序号	数据表标识符	数据表名称	备注
1.	users	用户信息表	存储用户名、密码、权限等信息
2.	Plane	机型表	存储飞机机型信息，用于区分构型
3.	Configuration	构型配置表	存储仿真所需构型信息
4.	LoadModelGroups	加载模型组配置表	存储构型所需加载的模型组信息
5.	LoadModels	加载模型配置表	存储构型所需加载的模型组中模型信息
6.	LoadServices	加载服务配置表	存储构型所需加载的服务信息
7.	ATAChapters	ATA 章节表	存储 ATA 章节信息，用于区分模型
8.	XNModels	模型表	存储各章节的模型信息
9.	XNModelsVersion	模型版本表	存储不同版本的各模型配置信息
10.	XNServices	服务表	存储服务信息
11.	XNServiceVersion	服务版本表	存储不同版本的服务配置信息
12.	version	版本记录表	存储系统版本更新记录

13.	SystemLog	系统日志表	存储仿真综合管理平台的日志
14.	ModuleCode	模块编码表	存储“玄鸟”架构各模块的编号
15.	SubModuleCode	子模块编码表	存储“玄鸟”架构各模块的子模块编号
16.	ErrorType	错误类型表	存储错误消息的类型识别编码
17.	ErrorLevel	错误等级表	存储错误消息的等级识别编码
18.	ErrorCode	错误码表	存储“玄鸟”架构所有错误码及其错误消息等信息

4.2.4. XNSimPortal

4.2.4.1. 概述

1．标识符：XNSimPortal

2．名称：仿真综合管理平台

3．开发状态：新开发软件配置项

4.2.4.2. 结构设计

仿真综合管理平台的结构如图 9 所示，包含用户权限管理、系统信息、构型配置、接口配置、模型集成、服务开发、仿真运行、资源监控、仿真运行监控、模型运行监控、数据监控、数据采集、网络监控与 QTG 共十四个功能模块。

图 9 仿真综合管理平台结构设计

4.2.4.3. 工作流程

仿真综合管理平台的主要工作流程如图 10 所示。

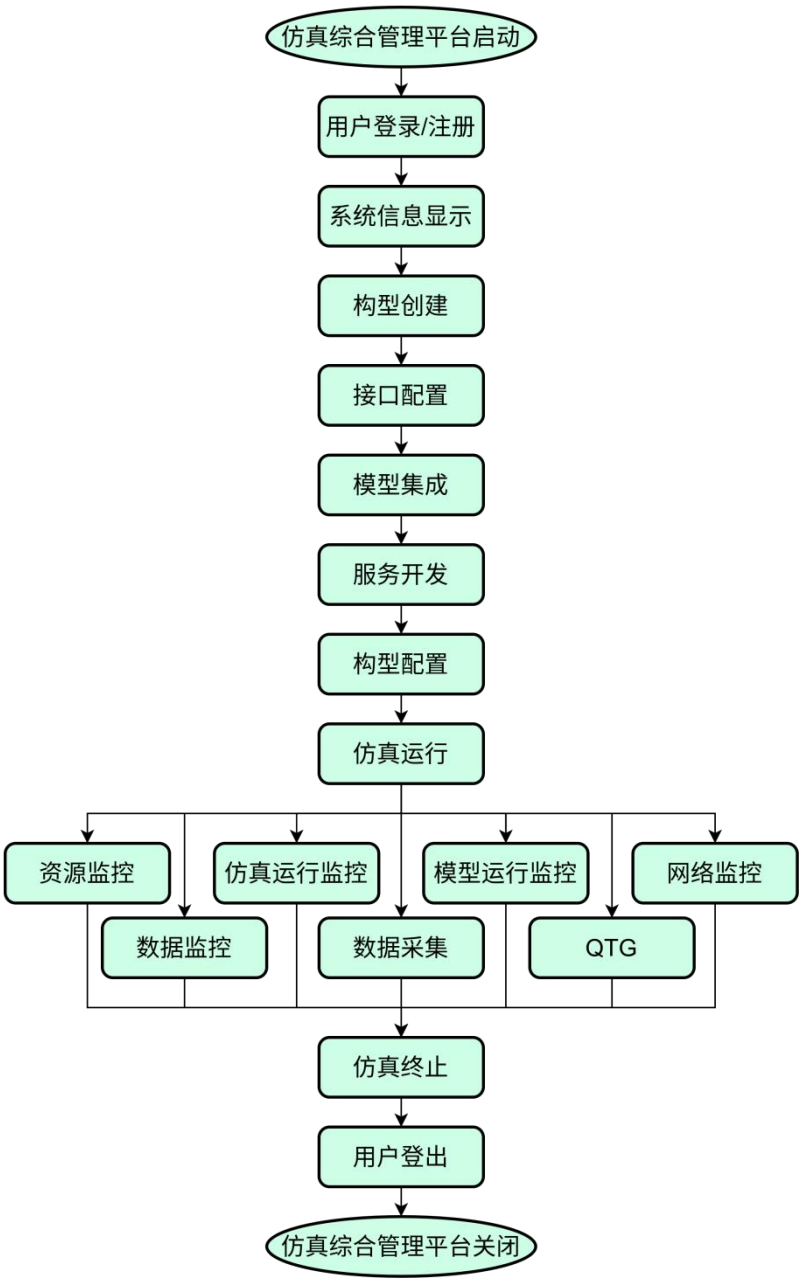

图 10 仿真综合管理平台工作流程

4.2.4.4. 模块设计

4.2.4.4.1. 用户权限管理

仿真综合管理平台的用户权限管理提供了用户登录、注册、登出、个人信息修改、密码修改、用户管理等多种用户权限管理相关功能。

用户信息部分加密存储在数据库中，且使用专用的 C++后端读取写入加密数据，避免加密算法暴露。其中：

1. 用户密码采用加盐 SHA-256 哈希算法加密存储，算法具有抗碰撞性（极难找到两个

不同输入产生相同的哈希值）、单向性（哈希值无法逆向推导原始数据）及雪崩效应（输入数据的微小变化会导致输出哈希值 50%以上比特位变化）的优点，非常适合密码存储与比对，并且将加密盐值选择为用户名与固定字符串的组合，进一步提高加密的安全性；

2. 用户权限等级使用五级用户权限，分别是管理员、组长、开发人员、用户、访客五级，其权限矩阵如所示。权限等级采用 AES-256-CBC 对称加密算法加密存储，其 256 位的密钥空间巨大，暴力破解几乎不可能，但拥有密钥时可以快速地反向解密出原权限等级数据以供查询；

表 4 用户权限矩阵

功能	管理员	组长	开发人员	用户	访客
登录/登出/注册	√	√	√	√	√
查看系统概览/帮助/Q&A	√	√	√	√	√
查看更新记录	√	√	√	√	
提交更新记录	√	√	√		
查看/修改待办事项	√	√	√		
查看构型/接口/模型/服务配置	√	√	√	√	
修改构型/接口/模型/服务配置	√	√	√		
删除构型/接口/模型/服务配置	√	√			
仿真运行/暂停/继续/终止	√	√	√	√	
查看仿真运行日志	√	√	√	√	
查看资源监控	√	√	√	√	√
查看仿真运行监控	√	√	√	√	√
查看模型运行监控	√	√	√	√	√
查看数据监控	√	√	√	√	
数据注入	√	√	√		
数据采集	√	√	√	√	
网络监控	√	√	√	√	
QTG	√	√	√		
用户权限设置	√				
修改个人信息	√	√	√	√	√

3. 其它用户信息使用明文存储。

4.2.4.4.2. 系统信息

仿真综合管理平台的系统信息功能模块提供了系统概览、更新记录、待办事项、帮助、Q&A 等用户交互功能页面。其中：

1. 系统概览页面展示了“玄鸟”架构的概述，包括其核心功能、设计理念以及技术亮点等方面的简要介绍。同时，页面还标注了当前发布的版本号，以便用户快速了解系统版本信息。此外，系统概览还列出了近期版本的更新记录，详细描述了每次版本更新所包含的新

功能、优化改进以及已知问题的修复情况，帮助用户及时掌握系统更新动态。页面还实时显示了当前的日期和时间，确保用户在使用过程中能够准确了解当前时间节点，方便进行时间管理和任务规划；

2. 更新记录页面呈现了“玄鸟”架构自创建以来的所有历史更新记录，包括每次更新的具体内容、更新时间、负责人员等信息。此外，该页面还提供了便捷的接口，允许开发人员随时提交新的更新记录。所有提交的更新记录将以明文形式直接保存在数据库中，以便于后续的查询、审核和追溯；

3. 待办事项页面将呈现与当前登录用户紧密相关的所有待办事项列表，确保用户能够掌握自己的任务情况。此外还具有一系列操作功能，包括但不限于对各项待办事项进行修改、删除、标记完成或未完成状态以及分配执行人等操作；

4. 帮助页面详细列出了适用于不同使用场景和用户需求的仿真综合管理平台的用户指南，其中涵盖了从基础操作到高级功能的各项说明，以助力用户更好地使用该平台。

5. Q&A 页面提供了一个便捷的交流平台，用户可以根据自己的需求和疑惑提出各种问题，开发人员针对用户提出的问题进行详细解答。通过这种双向沟通的方式，不仅能够帮助用户解决实际问题，还能促进用户与开发人员之间的紧密联系，进一步提升用户体验和产品服务质量。

4.2.4.4.3. 构型配置

仿真综合管理平台的系统信息功能模块提供了构型新建、构型基本参数编辑、加载模型组参数编辑、加载服务列表编辑、构型删除等功能。

1. 构型新建功能能够新建一条空的构型配置数据，也可以选择一个已有构型作为模板新建。当选择已有构型作为模板时，已有构型的加载模型组参数、加载模型参数、加载服务参数与接口配置都会复制到新构型；

2. 构型基本参数编辑功能可以编辑构型的操作系统名称、操作系统版本、操作系统实时补丁版本、CPU 亲和性、DDS 通信域 ID、是否在控制台输出（调试、普通、警告、错误）消息、是否在日志文件中输出（调试、普通、警告、错误）消息等参数；

3. 加载模型组参数编辑功能可以增加模型组，删除模型组，修改模型组的运行频率、优先级、CPU 亲和型，在加载模型列表中增加、删除模型等；

4. 加载服务列表编辑功能可以在加载服务列表中增加、删除服务；

5. 构型删除能够删除构型配置数据，同时也会删除构型关联的加载模型组参数、加载

模型参数、加载服务参数与接口配置。

4.2.4.4.4. 接口配置

仿真综合管理平台的接口配置模块提供了接口配置的增、删、改、查操作，还能从 ICD 文件直接导入接口数据，并且能够根据接口配置直接构建数据交互接口动态库。具体为：

1. 使用表格展示所有接口配置数据，包括：所属机型、ATA 章节、接口结构体名、接口名、数据类型、数据大小、备注；

2. 能够手动添加接口配置数据；

3. 能够删除任意一条接口配置数据，也能够批量删除至多 100 条的接口配置数据；

4. 能够修改任意一条接口配置数据；

5. 能够根据接口名查找搜索任意一条接口数据，也能够根据 ATA 章节或接口结构体名筛选接口数据；

6. 能够从 ICD 文件导入接口数据，但对 ICD 文件有限定格式：

   1） ICD 文件是.xlsx 后缀的 Excel 文件；

   2） ICD 文件包含两个工作表，分别命名为 Inputs 和 Outputs；

   3） Inputs 工作表必须有表头（即第一行），且表头必须包含以下几列：Input Variable Name、Variable Type、Description、Variable Dimensions；

   4） Outputs 工作表必须有表头（即第一行），且表头必须包含以下几列：Output Variable Name、Variable Type、Description、Variable Dimensions。

7. 能够从 IDL 文件导入接口数据，但对 IDL 文件有限定格式：

   1） IDL 文件必须是.idl 后缀的 IDL 语言文件；

   2） IDL 文件的必须具有三层模块，第一层模块名必须为 XNSim，第二层模块必须为有效的机型名称，第三层模块必须为有效的 ATA 章节号，其格式示例如图 11 所示。

```
module XNSim {
    // 机型名称
    module C909 {
        // ATA章节名称
        module ATAxx {
            // 输入结构体名称
            struct InputStrName {
                @optional 变量类型 变量名称;
                ...
            }
            // 输出结构体名称
            struct OutputStrName {
                @optional 变量类型 变量名称;
                ...
            }
            ...
        }
    }
}
```

图 11 IDL 文件格式示例

8. 能够根据当前接口配置数据，调用数据交互接口生成后端服务，自动化构建数据交互接口动态库，具体构建过程见。

4.2.4.4.5. 模型集成

仿真综合管理平台的模型集成模块提供了模型参数配置、模型分类显示、二进制数据包模型上传、模型集成模板代码下载、模型集成代码上传、模型自动化集成构建等功能，具体为：

1. 能够按照 ATA 章节号将所有模型分类展示；
2. 能够按照模型名称将所有不同版本的模型分类展示；
3. 能够新建模型参数配置；
4. 能够编辑模型参数配置，包括：
    1） 模型描述；
    2） 模型版本号；
    3） 作者；
    4） 修改时间；
    5） 运行频率组：基频，半频，四分之一频，八分之一频，十六分之一频，三十二分之一频；
    6） 运行节点：根据不同的运行频率组，可以选择的节点号为：0、0～1、0～3、0～7、0～15、0～31；

7） 优先级：0～99，99 为最高优先级；

8） 指令列表：每条需要输入指令名称、指令描述、调用函数三个参数。其中，指令是由仿真事件触发的，指令名称对应于事件的名称，调用函数对应于事件触发的函数。

5. 能够上传二进制数据包模型，但有以下限制条件：

1） 上传时必须选择二进制数据包的文件夹；

2） 文件夹中只有两个文件，一个为二进制数据包模型动态库文件（包含.so 字段），一个为头文件（后缀为.h）文件。

6. 二进制数据包模型上传后自动放置在当前所选构型的 Packages 目录下；

7. 根据上传的二进制数据包模型自动填写以下模型参数配置：

1） 数据包路径：Packages 目录下上传的二进制数据包模型所在目录；

2） 数据包名称：Packages 目录下上传的二进制数据包模型动态库文件名称；

3） 数据包头文件名称：Packages 目录下上传的二进制数据包模型头文件名称；

4） 数据包入口点：自动分析二进制数据包模型动态库符号表，找到含有 EntryPoint 字段的入口函数名；

5） 数据包接口参数名称：根据入口函数名分析函数输入参数的类型；

6） 分析二进制数据包模型头文件中的结构体定义，将输入、输出、心跳结构体的变量名称作为备选，供用户手动配置数据库中输入、输出、心跳结构体与头文件中输入、输出、心跳结构体的对应关系，用于模型集成模板代码的生成。

8. 能够调用模型集成后端服务根据已配置的模型参数生成模型集成模板代码，模板代码生成在当前所选构型的 ModelProjects 目录下；

9. 开发人员能够下载模型集成模板代码的压缩文件，格式为.zip 压缩文件；

10. 开发人员完成集成代码后，再次将集成代码压缩为.zip 文件，能够从仿真综合管理平台上传至所选构型的 ModelProjects 目录下。仿真综合管理平台将调用模型集成后端服务自动解压并覆盖之前的模板代码；

11. 能够调用模型集成后端服务自动构建模型集成代码，生成模型集成动态库并发布至所选构型的 Models 目录下。

4.2.4.4.6. 服务开发

仿真综合管理平台的服务开发模块提供了服务参数配置、服务分类显示、服务开发模板代码下载、服务开发代码上传、服务自动化构建等功能，具体为：

1. 能够按照服务名称将所有不同版本的服务分类展示；

2. 能够新建服务参数配置；

3. 能够编辑服务参数配置，包括：

1） 服务描述；

2） 服务版本号；

3） 作者；

4） 修改时间；

5） 指令列表：每条需要输入指令名称、指令描述、调用函数三个参数。其中，指令是由仿真事件触发的，指令名称对应于事件的名称，调用函数对应于事件触发的函数；

6） 其它参数：服务运行所需的其它参数，以 JSON 字符串的形式配置，服务开发时需要由开发人员完成 JSON 字符串的解析。

4. 能够调用服务开发后端服务根据已配置的服务参数生成服务开发模板代码，模板代码生成在仿真根目录的 ServiceProjects 目录下；

5. 开发人员能够下载服务开发模板代码的压缩文件，格式为.zip 压缩文件；

6. 开发人员完成开发后，再次将代码压缩为.zip 文件，能够从仿真综合管理平台上传至仿真根目录的 ServiceProjects 目录下。仿真综合管理平台将调用服务开发后端服务自动解压并覆盖之前的模板代码；

7. 能够调用服务开发后端服务自动构建服务开发代码，生成服务动态库并发布至仿真根目录的 Services 目录下。

4.2.4.4.7. 仿真运行

仿真综合管理平台的仿真运行模块提供了模型与服务展示、仿真测试、仿真运行控制、仿真进程实时监控与连接管理、仿真输出显示与日志管理功能，具体为：

1. 模型与服务展示：

1） 展示当前构型下的所有模型组参数，包括：模型组名称、运行频率（Hz）、优先级设置、CPU 亲和性配置；

2） 展示模型组内模型列表及其版本信息；

3） 展示构型中包含的所有服务及其版本信息。

2. 仿真测试：以测试模式启动仿真引擎，用于验证仿真配置的正确性；

2. 仿真运行控制：

1） 仿真启动：自动化构建 DDS 监控后端服务的数据监控器插件，启动 DDS 监控后端服务，初始化 DDS 监控后端服务的仿真引擎控制，正常启动仿真引擎进程，建立 SSE 实时输出连接；

2） 仿真暂停/继续：动态控制仿真运行状态；

3） 仿真停止：安全停止仿真进程，清理相关资源。

3. 仿真进程实时监控与连接管理：

1） SSE 事件源连接：建立 Server-Sent Events 连接，实时接收仿真输出；

2） 自动重连机制：支持连接断开后的自动重连，最多重试 3 次；

3） 状态检测：自动检测已有仿真进程，支持链接丢失后的状态恢复；

4） 进程状态监控：实时监控仿真进程的运行状态。

4. 输出显示与日志管理

1） 实时输出显示：实时显示仿真的标准输出和错误输出；

2） ANSI 转义序列处理：支持终端颜色、字体样式等格式化显示；

3） 自动滚动：输出内容自动滚动到最新位置；

4） 日志文件轮询：通过轮询机制读取日志文件内容；

5） 能够读取并显示历次仿真引擎输出的日志文件内容并按照日志消息等级筛选日志内容。

4.2.4.4.8. 资源监控

仿真综合管理平台的资源监控模块是一个实时监控系统硬件资源使用情况的模块，提供直观的图形化界面来展示系统性能和资源状态。具体为：

1. 系统概览信息：

1） 操作系统信息：显示当前运行的操作系统名称和内核版本；

2） CPU 核心数：显示系统 CPU 的核心数量；

3） IP 地址：显示当前系统的网络 IP 地址；

4） 已隔离 CPU 核心号：显示被隔离的 CPU 核心编号（用于实时系统）。

2. 实时监控图表：提供 4 个可配置的监控图表，支持的监控指标有：内存使用率（%）、磁盘使用率（%）、网络带宽（Mbps）、每个 CPU 核心的使用率（CPU0 使用率、CPU1 使用率等）。监控图表具有以下特性：

1） 实时数据更新：

2） 保存最近 60 个数据点的历史记录；

3） 支持平滑的曲线显示；

4） 使用不同颜色区分不同指标：

a） 网络带宽：绿色（下载）、红色（上传）；

b） CPU 使用率：紫色系；

c） 内存使用率：蓝色；

d） 磁盘使用率：橙色

4.2.4.4.9. 仿真运行监控

仿真综合管理平台的仿真运行监控模块是一个专门用于监控 XNSim 仿真引擎运行状态的模块，提供实时的引擎状态监控、线程信息跟踪和性能数据可视化功能。具体为：

1. 监控状态管理：

1） 自动监控启动：模块激活时自动启动系统监控；

2） 状态指示器：实时显示监控状态（未监控/监控中/监控错误）；

3） 智能状态检查：每秒检查一次监控状态，确保监控持续运行；

4） 依赖检查：启动监控前检查 DDS 监控后端服务是否已初始化。

2. 引擎信息监控：

1） 引擎名称：显示当前运行的仿真引擎名称（默认 XNSim）；

2） 引擎 ID：显示引擎的进程 ID；

3） 引擎状态：实时显示引擎运行状态：未运行（灰色）、运行中（绿色）、暂停中（橙色）、错误（红色）；

4） 引擎亲和性：显示 CPU 亲和性设置；

5） 线程数：显示引擎管理的线程总数。

3. 核心组件状态监控，监控 XNSim 引擎的 8 个核心组件状态，状态分别为未加载（灰色）、初始化完成（黄色）、正常（绿色）、异常（红色）。这 8 个核心组件是：

1） 主框架状态；

2） 时间管理器状态；

3） 事件管理器状态；

4） 构型管理器状态；

5） 线程管理器状态；

6） 模型管理器状态；

7） 服务管理器状态；

8） DDS 管理器状态。

4. 线程详细信息监控，以表格形式显示：

1） 线程名称：线程的标识名称；

2） 线程 ID：系统分配的线程 ID；

3） 状态：线程当前运行状态；

4） 优先级：线程执行优先级；

5） 运行次数：线程已执行的次数；

6） 设定频率(Hz)；

7） 平均频率(Hz)；

8） 最大频率(Hz)；

9） 最小频率(Hz)；

10） 设定周期(ms)；

11） 平均周期(ms)；

12） 最大周期(ms)；

13） 最小周期(ms)。

5. 实时图表监控能够显示线程的实时性能数据，支持频率和周期两种显示模式，每秒更新图表数据，能同时显示多个线程的性能曲线，且每个线程保留最近 30 个数据点。

4.2.4.4.10. 模型运行监控

仿真综合管理平台的模型运行监控模块是一个专门用于监控 XNSim 仿真系统中各个模型运行状态的模块，提供实时的模型状态监控、性能数据跟踪和可视化分析功能。具体为：

1. 监控状态管理

1） 自动监控启动：组件激活时自动启动模型监控服务；

2） 状态指示器：实时显示监控状态（未监控/监控中/监控错误）；

3） 智能状态检查：每秒检查一次监控状态，确保监控持续运行；

4） 依赖检查：启动监控前检查 DDS 监控后端服务是否已初始化；

5） 错误恢复机制：连续错误超过 3 次时自动停止监控并提示用户。

2. 模型信息监控：

1） 模型名称：显示仿真模型的标识名称；

2） 模型 ID：显示模型的唯一标识符；

3） 模型状态：实时显示模型运行状态：未运行（灰色）、运行中（绿色）、暂停中（橙色）、错误（红色）；

4） 线程 ID：显示模型所属线程的标识；

5） 节点：显示模型运行的节点信息；

6） 优先级：显示模型的执行优先级；

7） 运行次数：显示模型已执行的次数；

8） 设定频率(Hz)：模型配置的目标执行频率；

9） 平均频率(Hz)：模型实际执行的平均频率；

10） 最大频率(Hz)：模型执行过程中的最高频率；

11） 最小频率(Hz)：模型执行过程中的最低频率；

12） 设定周期(ms)：模型配置的目标执行周期；

13） 平均周期(ms)：模型实际执行的平均周期；

14） 最大周期(ms)：模型执行过程中的最长周期；

15） 最小周期(ms)：模型执行过程中的最短周期。

3. 实时图表监控模型性能：

1） 双模式显示：支持频率和周期两种显示模式切换；

2） 实时数据更新：每秒更新图表数据；

3） 多模型跟踪：同时显示多个模型的性能曲线；

4） 历史数据：保留最近 30 个数据点；

5） 颜色区分：不同模型使用不同颜色标识。

4.2.4.4.11. 数据监控

仿真综合管理平台的数据监控模块是一个功能强大的仿真数据监控和注入工具，提供实时的数据监控、数据注入、CSV 批量注入和可视化图表功能，是 XNSim 仿真系统数据交互的核心组件。具体为：

1. 监控状态管理：

1） 自动监控启动：组件激活时自动启动数据监控服务；

2） 状态指示器：实时显示监控状态（未监控/监控中/监控错误）；

3） 智能状态检查：每秒检查一次监控状态，确保监控持续运行；

4） 依赖检查：启动监控前检查 DDS 监控后端服务是否已初始化；

5） 错误恢复机制：连续错误时自动停止监控并提示用户。

2. 接口展示：以树形结构展示所有接口名称，接口按接口结构体名称自动分组显示，支持按接口名称或结构体名称搜索。用户可以在树型结构中选择需要监控的接口。

3. 实时数据监控使用表格展示监控数据，并具有以下特性：

1） 实时数据更新：每秒更新监控数据；

2） 数据验证：验证注入值格式和类型；

3） 状态显示：显示每个接口的监控和注入状态；

4. 单次数据注入：

1） 值输入：支持单个数值或数组数据输入；

2） 格式验证：实时验证输入格式的正确性；

3） 类型检查：根据接口类型验证数据格式；

4） 即时反馈：注入后立即显示结果。

5. 连续数据注入：

1） 频率控制：可设置注入频率（1-1000Hz）；

2） 状态管理：显示注入状态（注入中/停止）；

3） 操作限制：注入时禁用其他操作；

4） 自动停止：支持手动停止连续注入。

6. CSV 批量数据注入：

1） CSV 文件上传：

a. 文件验证：验证 CSV 文件格式和大小（最大 10MB）；

b. 头部验证：验证 CSV 文件头部与接口表的匹配；

c. 数组索引：支持数组接口的索引标注（如：接口名(1)或接口名(1_2)）；

d. 错误处理：详细的错误提示和文件清理。

2） 批量注入：

a. 自动停止：注入前自动停止所有持续注入；

b. 状态管理：显示注入状态和进度；

c. 操作限制：注入时禁用其他操作；

d. 文件管理：支持文件上传和删除。

7. 可视化图表功能使用浮动图表窗口显示监控数据，具有以下特性：

1） 多窗口支持：支持同时打开多个图表窗口；

2） 实时更新：图表数据实时更新；

3） 窗口管理：支持窗口拖拽、调整大小、关闭；

4） 数据解析：自动解析各种数据格式。

4.2.4.4.12. 数据采集

仿真综合管理平台的数据采集模块是一个专门用于仿真数据采集和可视化的模块，支持脚本驱动的数据采集、实时状态监控、采集结果可视化和文件下载功能，是 XNSim 仿真系统数据采集的核心工具。具体为：

1. 监控状态管理：

1） 自动监控启动：组件激活时自动启动监控状态检查；

2） 状态指示器：实时显示监控状态（未监控/监控中/监控错误）；

3） 智能状态检查：每秒检查一次监控状态，确保系统正常运行；

4） 依赖检查：检查 DDS 监控后端服务是否已初始化；

5） 错误恢复机制：网络错误时自动重试。

2. 脚本文件管理：

1） 脚本文件上传与格式验证：只支持.dcs 格式的脚本文件；

2） 脚本内容解析：

a. 采集列表解析：解析脚本中 collect_list 中的接口列表，验证脚本中的接口是否在系统中存在，以及验证数组索引是否在有效范围内；

b. 数组索引支持：支持一维和二维数组接口（如：接口名(1)或接口名(1_2)）；

c. 参数解析：解析 for duration at frequency 格式的采集参数，duration 为采集时长，frequency 为采集频率；

d. 输出文件解析：解析 put/extend/all result 指定的输出文件。

3） 脚本卸载功能：卸载时自动删除服务器上的脚本文件。

3. 数据采集控制：

1） 采集参数配置：采集频率、采集时长、输出文件均从脚本文件中读取并自动填写；

2） 采集状态管理：能够跟踪采集状态（未加载脚本/已加载脚本/数据采集中/采集

完成），并在采集完成后自动停止并更新状态。

4. 接口列表管理：需要采集数据的接口按树形结构列出，并按结构体分组显示；

5. 采集结果可视化采用浮动图表窗口绘制数据，具有以下特性：

1） 多窗口支持：支持同时打开多个图表窗口；

2） 实时更新：采集完成后立即显示数据；

3） 窗口管理：支持窗口拖拽、调整大小、关闭。

6. 采集结果数据文件提供下载功能。

4.2.4.4.13. 网络监控

仿真综合管理平台的网络监控模块是一个专门用于 UDP/TCP 数据包抓取和解析的模块，提供实时的网络数据监控、数据包详情查看和协议解析功能，是 XNSim 仿真系统网络通信监控的重要工具。具体为：

1. 网络配置管理：

1） 监听 IP 地址配置：可设置目标 IP 地址（默认 127.0.0.1）；

2） 监听端口配置：可设置监听端口（范围 1024-65535，默认 54321）；

3） 参数验证：实时验证 IP 和端口参数的有效性；

4） 状态同步：与后端监控状态保持同步；

2. 实时数据监控：

1） 定时获取：每秒从后端获取新的数据包；

2） 数据累积：持续累积抓取的数据包；

3） 容量限制：最多保留 1000 条数据包记录；

4） 自动清理：超出限制时自动清理旧数据。

3. UDP/TCP 数据包列表显示：

1） 时间戳显示：显示数据包的精确时间戳（包含毫秒）；

2） 来源信息：显示数据包的来源地址；

3） 数据大小：显示数据包的大小（字节数）；

4. UDP/TCP 数据包详情解析：

1） 格式化显示：按 16 字节每行格式化显示；

2） 偏移地址：显示每行的十六进制偏移地址；

3） 字节对齐：整齐的字节对齐显示；

5. UDP/TCP 数据包头部解析功能：

1） 协议标识：识别 XNSim 协议标识（0xA6）；

2） 数据类型：解析机型数据类型；

3） ATA 章节：解析 ATA 章节号；

4） 模型编号：解析模型编号；

5） 结构体类型：识别输入/输出/心跳结构体；

6） 传输方向：识别输入/输出方向；

7） 数据包大小：解析数据包大小信息。

4.2.4.4.14. QTG

仿真综合管理平台的 QTG 模块是用于支持二进制数据包模型进行 QTG 测试的模块，其功能暂未设计。

4.2.5. XNEngine

4.2.5.1. 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

4.2.5.2. 结构设计

仿真调度引擎的结构如图 12 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 12 仿真调度引擎结构图

4.2.5.3. 工作流程

仿真调度引擎的工作流程如图 13 所示。

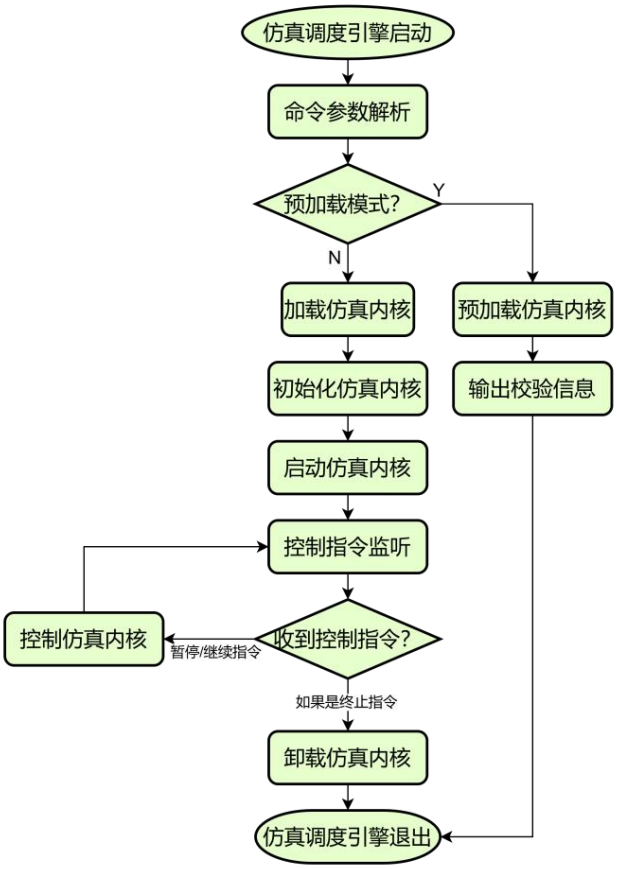

图 13 仿真调度引擎工作流程

4.2.5.4. 模块设计

4.2.5.4.1. 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：构型 ID、仿真内核加载模式、版本号查询、帮助。

1. 构型 ID 参数用于指定仿真运行所使用的构型 ID，引擎将根据此 ID 从数据库中查找构型配置，参数输入格式为 “-id <ID>”；

2. 仿真内核加载模式通过特定的参数“-test”来识别是否启用预加载模式。如果命令行参数中有此参数，仿真内核将按照预加载模式运行，用于检验构型配置是否有效；

3. 版本号查询使用“-v”参数，引擎将输出当前版本号信息并退出；

4. 帮助使用“-h”参数，引擎将输出所有支持的命令行参数及其说明并退出。

4.2.5.4.2. 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构型配置校验。在这个过程中，仿真内核将会校验构型配置及其相关的模型配置和服务配置，并尝

试动态加载所需的模型和服务的动态链接库。

预加载时如果出现校验失败，会向控制台及日志文件中打印错误信息；如果校验成功，则向控制台报告校验通过。无论校验是否成功，校验完成后直接退出引擎。

4.2.5.4.3. 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构型配置。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链接库、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。

4.2.5.4.4. 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。

4.2.6. 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、构型管理器、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、模型集成框架、服务框架、数据交互抽象接口共十三个模块。

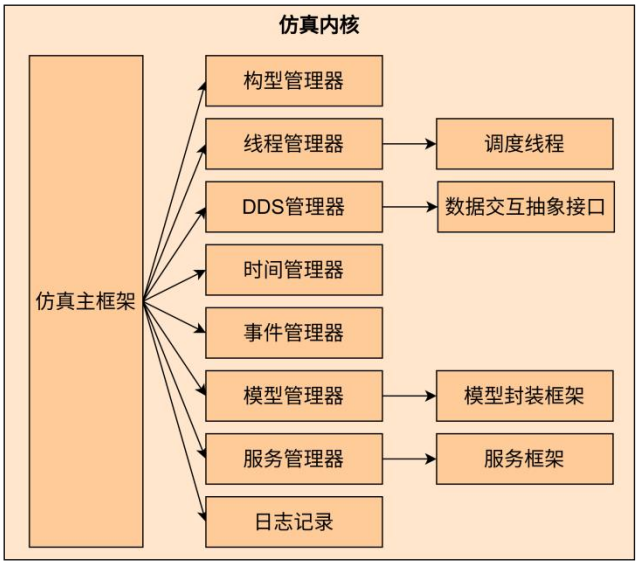

图 14 仿真内核结构图

4.2.6.3. 工作流程

仿真内核的工作流程如图 15 所示。

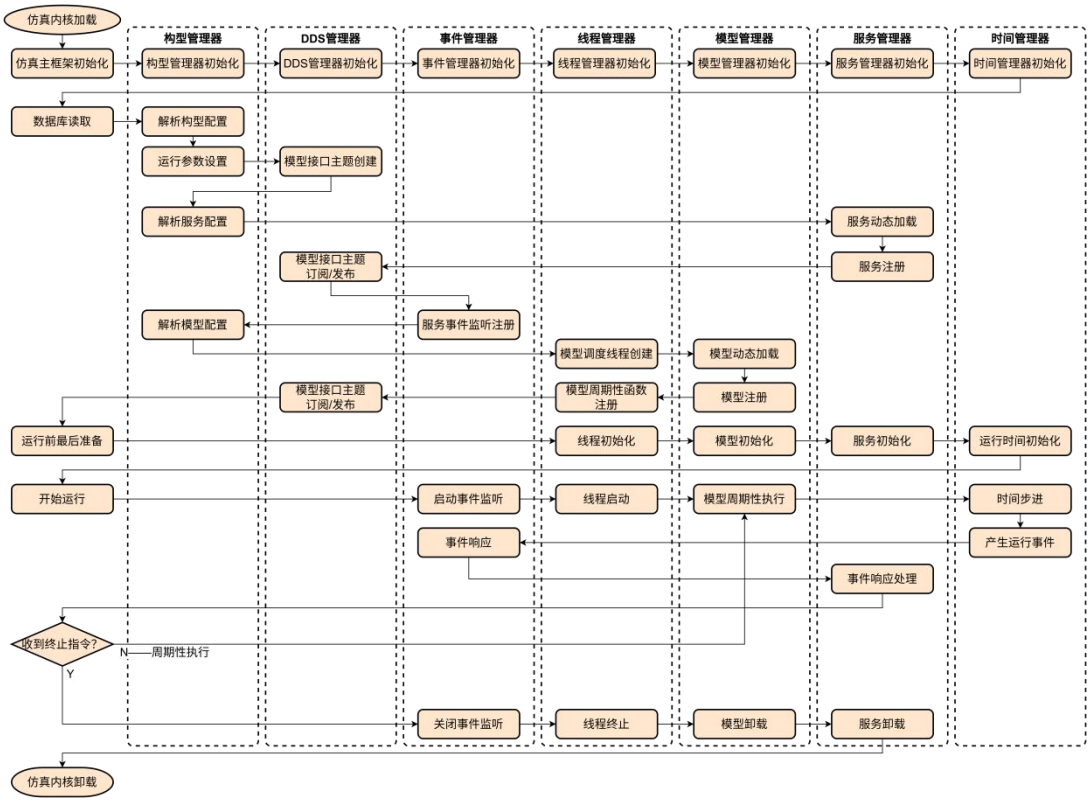

图 15 仿真内核工作流程

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、构型管理器、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. 构型管理器

构型管理器负责解析构型配置并根据这些配置来设置仿真环境。它定义了一些构型相关的属性及接口，包括：

1. 构型管理器初始化接口；

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。

构型管理器的主要工作流程如下：

1. 执行构型管理器的初始化，初始化所有参数；

2. 读取构型配置中的环境配置信息，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. 读取构型加载服务列表中的服务名称和版本，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；

4. 服务注册成功后会获得其全局唯一 ID，构型管理器根据此 ID 从服务管理器获取服

务的智能指针，再读取服务配置设置其参数；

5. 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）；

6. 读取构型加载模型组列表中模型组的频率、优先级、亲和性等信息，交给线程管理器创建对应的调度线程；

7. 读取构型加载模型列表中的模型名称和版本，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；

8. 模型注册成功后会获得其全局唯一 ID，构型管理器根据此 ID 从模型管理器获取模型的智能指针，再读取其模型配置设置其参数；

9. 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. 根据构型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. 根据构型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. 接收所有 DDS 参与者的注册，并调用 Fast DDS 库注册参与者；

4. 完成其它仿真运行前最后准备工作；

5. 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. 接受构型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集成框架来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. 接收构型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. 模型注册完成后向构型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. 接收构型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. 服务注册完成后向构型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控制台中；

2. 是否向控制台输出日志接口；

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. 模型集成框架

模型集成框架提供统一的二进制数据包模型集成接口，使二进制数据包模型能够与仿真内核解耦。模型集成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、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、服务其它配置参数、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）；

4.2.6.4.13. 数据交互抽象接口

数据交互抽象接口主要用于实现数据的序列化、反序列化和网络传输功能。它提供了统一的数据处理接口，支持多种数据类型的自动转换和传输。核心功能为：

1. 数据序列化与反序列化：
   1） 字节数组转换: 将各种数据类型转换为字节数组用于网络传输；
   2） 字符串转换: 将数据转换为字符串格式用于监控前端交互；
   3） 类型安全: 使用模板元编程确保类型安全，支持算术类型和 std::array 类型。
2. 网络通信支持：
   1） UDP 包处理: 生成和解析 UDP 数据包，支持最大 40KB 的数据包；
   2） 头部管理: 包含 8 字节的固定头部，用于数据包识别；
   3） 数据分片: 当数据超过最大包大小时自动分片处理；
3. 数据类型支持：
   1） 基本算术类型: int, float, double, char 等；
   2） 数组类型: std::array，支持嵌套数组（最多两层嵌套）；
   3） 可选类型: 使用 FastDDS 的可选类型包装，支持空值处理。

4.2.7. XNInterfaceGenServer

4.2.7.1. 概述

1．标识符：XNInterfaceGenServer

2．软件配置项名称：数据交互接口生成后端服务

3．开发状态：新开发软件配置项

4.2.7.2. 结构设计

数据交互接口生成后端服务的结构如图 16 所示。包含仿接口配置读取、IDL 文件生成、FastDDS 接口生成、数据交互接口生成、构建文件生成、自动化构建共六个模块。

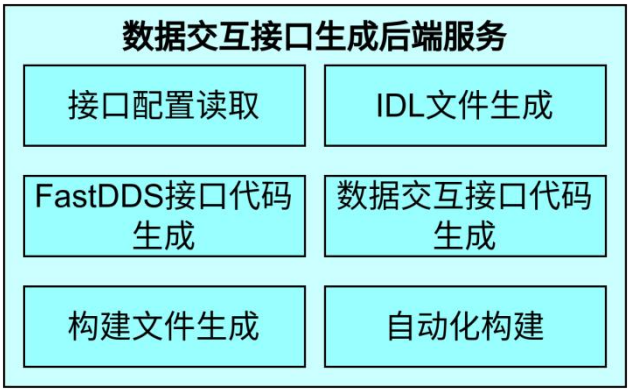

图 16 数据交互接口生成后端服务结构图

4.2.7.3. 工作流程

数据交互接口生成后端服务的工作流程如所示。

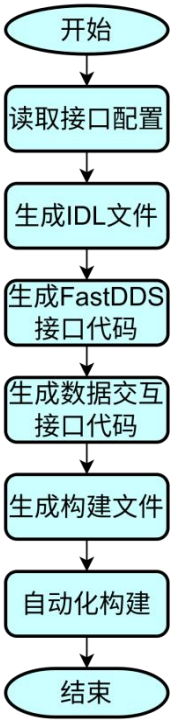

图 17 数据交互接口生成后端服务工作流程

4.2.7.4. 模块设计

4.2.7.4.1. 接口配置读取

接口配置读取模块根据输入的构型配置 ID，从数据库中读取构型对应的构型名称（ConfigurationName）和接口配置数据，并使用特定的数据结构保存接口信息。模块采用五层数据结构保存接口信息：

AllInterfaceData （所有接口信息）
├── systemName （系统名称，默认为 XNSim）
├── planeName （机型名称）
└── ataInterfaceData （每个 ATA 章节的接口信息列表）
　　└── ATAInterfaceData（单个 ATA 章节的接口信息）
　　　　├── ataName （ATA 章节名称）
　　　　└── structInterfaceData （每个接口结构体的信息列表）
　　　　　　└── StructInterfaceData（单个接口结构体的信息）
　　　　　　　　├── modelStructName （接口结构体名称）
　　　　　　　　└── interfaceData （接口结构体中接口信息列表）
　　　　　　　　　　└── InterfaceData（单个接口信息）
　　　　　　　　　　　　├── interfaceName（接口名称）
　　　　　　　　　　　　├── interfaceType （接口类型）
　　　　　　　　　　　　├── interfaceIsArray （是否为数组）
　　　　　　　　　　　　├── interfaceArraySize_1 （一维数组大小）
　　　　　　　　　　　　├── interfaceArraySize_2 （二维数组大小）
　　　　　　　　　　　　└── interfaceNotes （接口注释）

4.2.7.4.2. IDL 文件生成

IDL 文件生成模块根据保存的接口信息，按照符合 IDL 语法的方式将 IDL 文件生成在构型目录的 IDL 目录下。生成的 IDL 文件命名为“ConfigurationName.idl”，其内容格式为：

```
module XNSim{
    module planeName{
        module ataName1{
            struct modelStructName1{
```

```
                @optional interfaceType interfaceName; //interfaceNotes
                ...
            } //end of modelStructName1
            struct modelStructName2{
                ...
            } //end of modelStructName2
            ...
        }//end of ataName1
        module ataName2{
            ...
        }//end of ataName2
        ...
    }//end of planeName
}//end of XNSim
```

4.2.7.4.3. FastDDS 接口生成

FastDDS 接口生成模块将在构型目录的 IDL 目录下，运行 FastDDS-Gen 的 bash 指令“fastddsgen ConfigurationName.idl -replace”，使用 IDL 文件生成 FastDDS 接口定义文件，这一过程将产生以下文件：

1. ConfigurationName. hpp：定义 FastDDS 所需的数据接口类型；

2. ConfigurationNameCdrAux. hpp 与 ConfigurationNameCdrAux. ipp：定义 FastDDS 所需的数据序列化方法；

3. ConfigurationNamePubSubTypes. hpp 与 ConfigurationNamePubSubTypes. cxx：定义 FastDDS 主题交互所需的发布/订阅类型；

4. ConfigurationNameTypeObjectSupport. hpp 与 ConfigurationNameTypeObjectSupport. cxx：定义 FastDDS 主题交互所需的类型对象支持。

4.2.7.4.4. 数据交互接口代码生成

数据交互接口代码生成模块将在构型目录的 IDL 目录下，根据仿真内核提供的数据交互抽象接口生成对应的数据交互接口代码。具体生成流程为：

1. 模块会为每一个 ATA 章节建立一个单独的目录；

2. 在每一个 ATA 章节目录下针对该章节的每一个接口结构体实现它的抽象仿真接口；

3. 在每一个 ATA 章节目录下添加 CMakeLists.txt 构建文件；

4. 在 IDL 目录下添加 ConfigurationName_Interface.h，引用之前创建的所有.hpp 文件。

这一过程将在 IDL 目录下建立以下目录结构：

IDL

├──ataName1

　　├── modelStructName1.hpp

　　├── modelStructName1.cxx

　　├── ...

　　└── CMakeLists.txt

├──...

└── ConfigurationName_Interface.h

4.2.7.4.5. 构建文件生成

构建文件生成模块将在 IDL 目录下生成一个 CMakeLists.txt 文件，该文件将包含整个数据交互接口自动化构建的所有配置信息。主要包含以下配置：

1. 基础配置：CMake 版本需求、项目名称、C++版本等；

2. 构建类型配置：Release 模式；

3. 依赖库配置：XNCore、FastDDS、FastCDR、OpenSSL;

4. 子目录配置：添加所有 ATA 章节目录；

5. 编译配置：编译文件、编译链接库等；

6. 安装配置：生成的动态链接库将被安装到仿真根目录的 lib 目录下。

4.2.7.4.6. 自动化构建

自动化构建模块将在 IDL 目录下新建 build 目录，在 build 目录下执行“cmake .. && make && make install”的 bash 指令。这将调用 CMake 工具对数据交互接口进行构建并安装。

在安装完成后，还会执行“sudo ldconfig”的 bash 指令来刷新系统的动态链接库配置。

4.2.8. XNDDSInterface

4.2.8.1. 概述

1．标识符：XNDDSInterface

2．软件配置项名称：数据交互接口

3．开发状态：无需开发，由 XNInterfaceGenServer 自动化构建的软件配置项

4.2.8.2. 结构设计

数据交互接口的结构如图 18 所示，包含 FastDDS 交互接口、模型数据交互接口、TCP/UDP 数据交互接口和数据监控交互接口共四个模块。

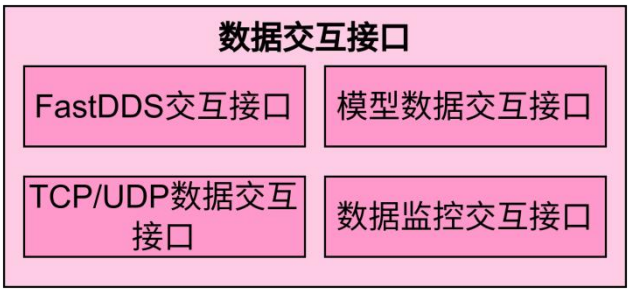

图 18 数据交互接口结构图

4.2.8.3. 工作流程

数据交互接口只用于向外部提供通信的接口，因此不具备自身的主要工作流程。其并行的简要数据交互流程如所示。

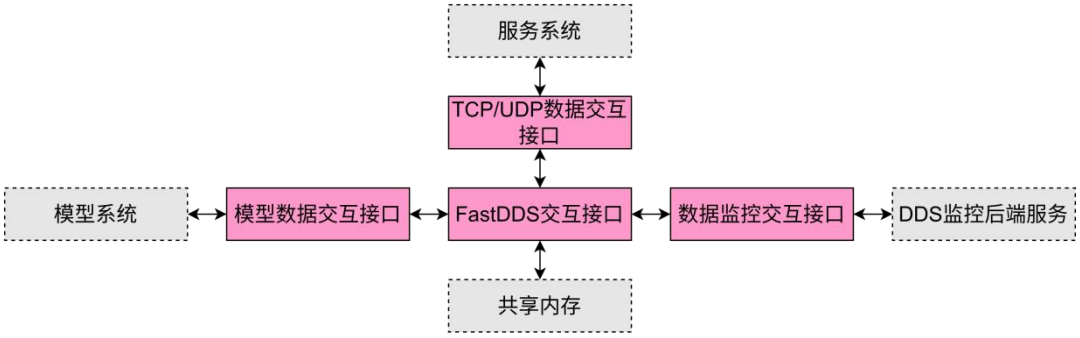

图 19 数据交互接口的数据交互流程

4.2.8.4. 模块设计

4.2.8.4.1. FastDDS 交互接口

FastDDS 交互接口模块是整个数据交互接口的核心模块，所有交互数据最终均通过该模块在共享内存中实现读写操作。该模块利用 Fast DDS 提供的 API，基于 Fast DDS 的发布/订阅模式，高效地在共享内存中进行实时数据交互。

在 FastDDS 交互接口中，模型接口数据结构体的每个成员均被设定为可选模式。这样，在进行发布/订阅交互时，无需完整填写结构体的所有数据即可实现数据交换，有效避免了

不必要的数据覆盖和获取冗余数据的问题。

4.2.8.4.2. 模型数据交互接口

模型数据交互接口模块将二进制数据包模型中定义的结构体形式的接口数据与FastDDS中定义的主题数据进行双向映射，能够实现模型数据与主题数据的双向转换。具体为：

1. 使用模板元编程技术，在编译期即确定每个数据接口的数据类型及转换方式；

2. 提供统一的模型数据与主题数据的双向转换接口，模型只需简单使用该接口即可完成所需的数据转换。

4.2.8.4.3. TCP/UDP 数据交互接口

TCP/UDP 数据交互接口将用于网络通信的字节数组与 FastDDS 中定义的主题数据进行双向映射，能够实现 FastDDS 主题数据的序列化与反序列化。具体为：

1. 使用模板元编程技术，在编译期即确定每个数据接口的数据类型及序列化方式；

2. 提供统一的序列化与反序列化接口，服务只需使用该接口即可将 FastDDS 主题数据序列化为字节数组或将字节数组反序列化为主题数据。

4.2.8.4.4. 数据监控交互接口

数据监控交互接口将用于数据监控的字符串数据与 FastDDS 中定义的主题数据进行双向映射，能够实现 FastDDS 主题数据的另一种序列化与反序列化。具体为：

1. 使用模板元编程技术，在编译期即确定每个数据接口的数据类型及序列化方式；

2. 提供统一的序列化与反序列化接口，服务只需使用该接口即可将 FastDDS 主题数据序列化为字符串用于显示或将字符串形式的数据反序列化为主题数据。

4.2.9. XNModelGenServer

4.2.9.1. 概述

1．标识符：XNModelGenServer

2．软件配置项名称：模型集成后端服务

3．开发状态：新开发软件配置项

4.2.9.2. 结构设计

模型集成后端服务的结构如图 20 所示，包含模型集成模板代码生成、模型代码压缩/

解压缩、模型自动化构建共三个模块。

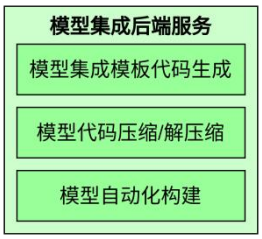

图 20 模型集成后端服务结构图

4.2.9.3. 工作流程

模型集成后端服务的工作流程如所示。

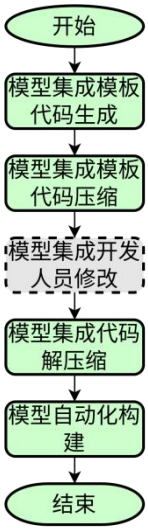

图 21 模型集成后端服务工作流程

4.2.9.4. 模块设计

4.2.9.4.1. 模型集成模板代码生成

模型集成模板代码生成模块读取数据库中模型的配置，基于仿真内核提供的模型集成框架，实现大部分的模型集成代码。模型代码将生成在当前构型的 ModelProjects 目录下，具体生成以下内容：

1. 根据仿真内核提供的模型集成框架搭建基础模型集成代码；

2. 根据数据库中模型配置，完成二进制数据包模型的动态加载与入口函数调用相关代码；

3. 根据数据库中模型配置，引用数据交互接口头文件，调用模型数据交互接口，完成模型接口结构体数据与 FastDDS 主题数据的映射相关的代码；

4. 生成 CMakeLists.txt 文件，用于自动化构建并发布模型。

4.2.9.4.2. 模型代码压缩/解压缩

模型代码压缩/解压缩模块提供了模型集成模板代码压缩为.zip 文件以及解压缩.zip 文件并覆盖模型集成代码的功能。这一功能主要用于方便模型集成开发人员下载模板代码与上传完成集成后的代码。

4.2.9.4.3. 模型自动化构建

模型自动化构建模块将在模型集成代码目录下新建 build 目录，在 build 目录下执行“cmake .. && make && make install”的 bash 指令。这将调用 CMake 工具对模型集成代码进行构建并安装。模型动态链接库将被安装在当前构型目录的 Models 目录下。

4.2.10.XNModels

4.2.10.1. 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

4.2.10.2. 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集成框架进行集成。模型系统支持动态加载的所有模型如图 22 所示。

图 22 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 23 所示。

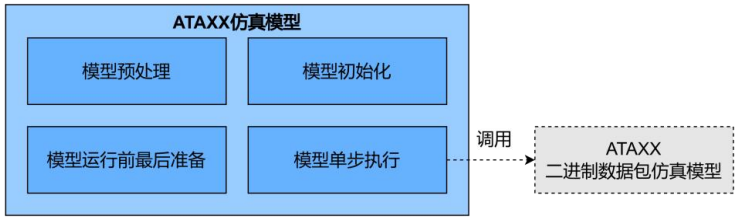

图 23 模型组成结构

4.2.10.3. 工作流程

模型系统单个模型的工作流程如图 24 所示。

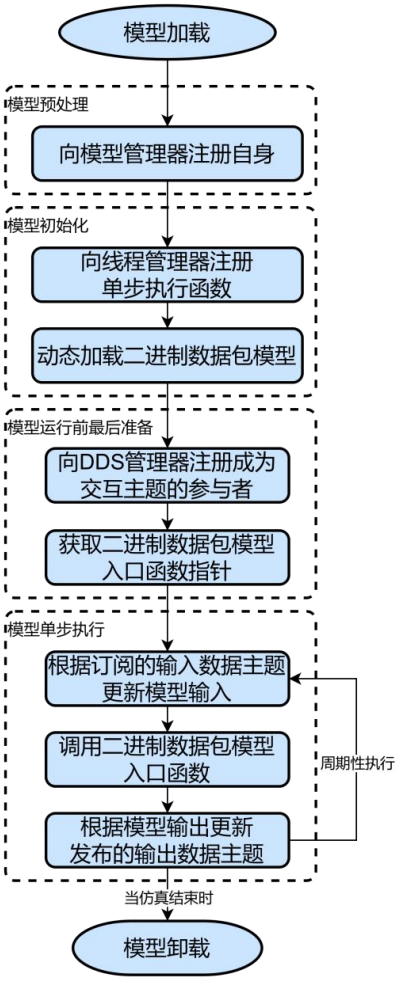

图 24 模型工作流程

4.2.10.4. 模块设计

4.2.10.4.1. 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。

4.2.10.4.2. 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，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。

4.2.10.4.3. 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。

4.2.10.4.4. 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。

4.2.11. XNServiceGenServer

4.2.11.1. 概述

1．标识符：XNServiceGenServer

2．软件配置项名称：服务开发后端服务

3．开发状态：新开发软件配置项

4.2.11.2. 结构设计

服务开发后端服务的结构如图 20 所示，包含服务开发模板代码生成、服务代码压缩/解压缩、服务自动化构建共三个模块。

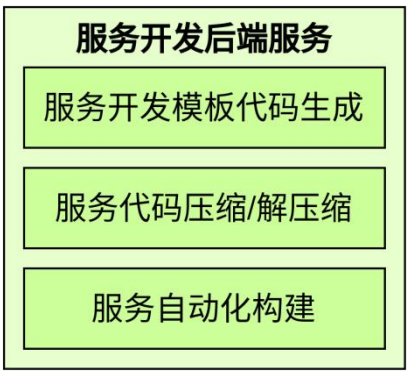

图 25 服务开发后端服务结构图

4.2.11.3. 工作流程

服务开发后端服务的工作流程如所示。

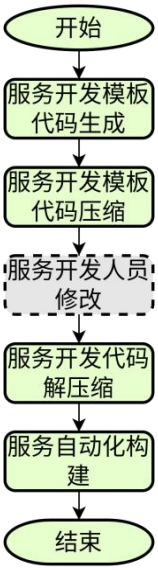

图 26 服务开发后端服务工作流程

4.2.11.4. 模块设计

4.2.11.4.1. 服务开发模板代码生成

服务开发模板代码生成模块读取数据库中副刷的配置，基于仿真内核提供的服务开发框架，实现大部分的服务开发代码。服务代码将生成在当前构型的 ServiceProjects 目录下，具

体生成以下内容：

1. 根据仿真内核提供的服务开发框架搭建基础服务开发代码；

2. 生成 CMakeLists.txt 文件，用于自动化构建并发布服务。

4.2.11.4.2. 服务代码压缩/解压缩

服务代码压缩/解压缩模块提供了服务开发模板代码压缩为.zip 文件以及解压缩.zip 文件并覆盖服务开发代码的功能。这一功能主要用于方便服务开发人员下载模板代码与上传完成开发后的代码。

4.2.11.4.3. 服务自动化构建

服务自动化构建模块将在服务开发代码目录下新建 build 目录，在 build 目录下执行“cmake .. && make && make install”的 bash 指令。这将调用 CMake 工具对服务代码进行构建并安装。服务动态链接库将被安装在仿真根目录的 Services 目录下。

4.2.12.XNServices

4.2.12.1. 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

4.2.12.2. 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 27 所示。

图 27 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 28 所示。

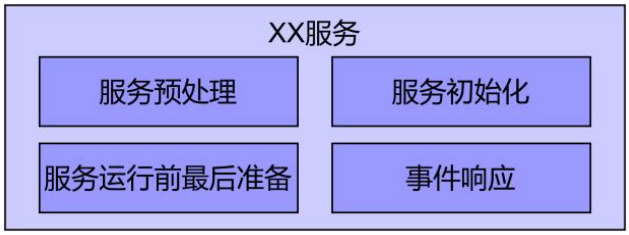

图 28 服务组成结构

4.2.12.3. 工作流程

服务系统中单个服务的工作流程如图 29 所示。

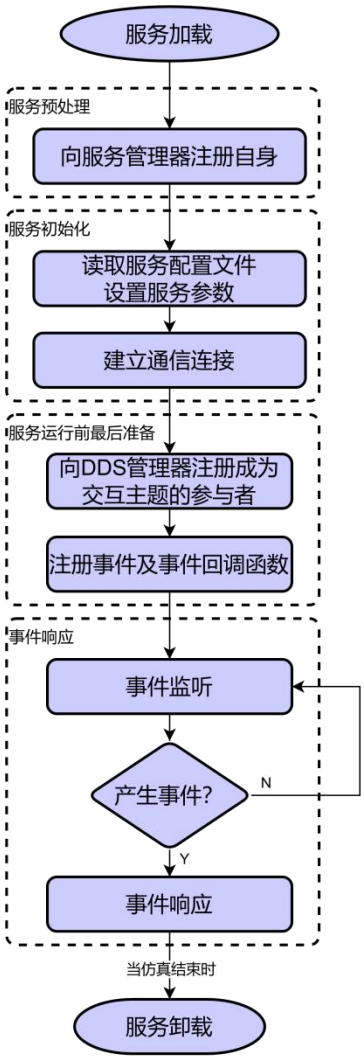

图 29 服务工作流程

4.2.12.4. 模块设计

4.2.12.4.1. 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。

4.2.12.4.2. 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，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。

4.2.12.4.3. 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。

4.2.12.4.4. 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。

4.2.13.XNMonitorServer

4.2.13.1. 概述

1．标识符：XNMonitorServer

2．软件配置项名称：DDS 监控后端服务

3．开发状态：新开发软件配置项

4.2.13.2. 结构设计

DDS 监控后端服务的结构如图 30 所示，包含仿真运行状态监控、模型运行状态监控、仿真运行控制、数据监控器插件管理、数据监控器抽象接口、数据监控器生成、模型数据监控、模型数据采集、模型数据注入共九个功能模块。

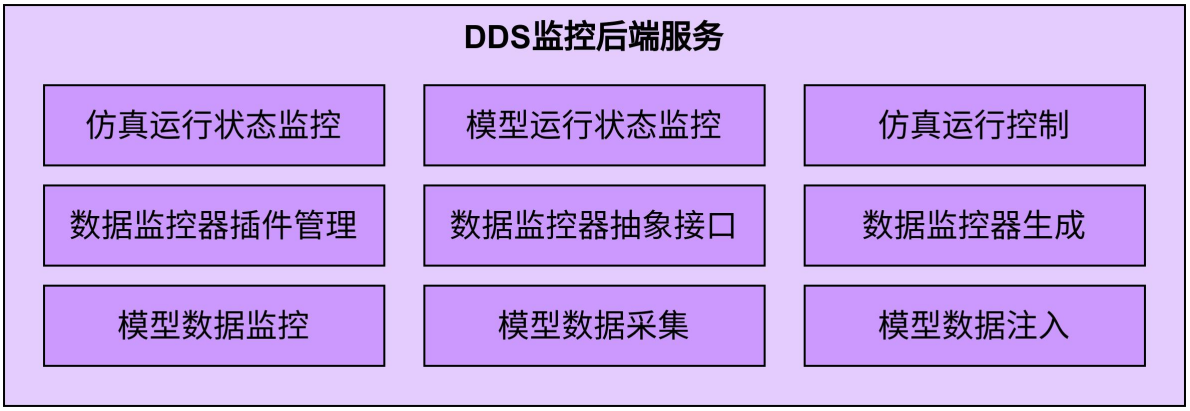

图 30 DDS 监控后端服务结构图

4.2.13.3. 工作流程

DDS 监控后端服务由综合仿真管理平台驱动，它具有多个工作流程，如图 31 所示。

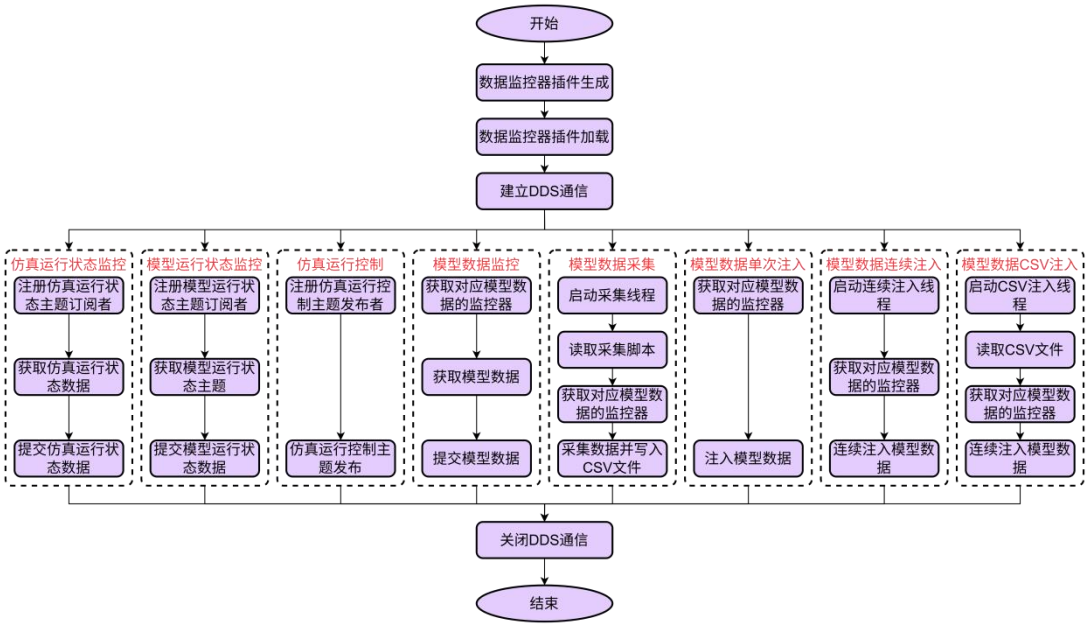

图 31 DDS 监控后端服务工作流程

4.2.13.4. 模块设计

4.2.13.4.1. 仿真运行状态监控

DDS 监控后端服务的仿真运行状态监控模块中，通过订阅仿真内核运行过程中发布的仿真运行状态主题，实时向仿真综合管理平台提交仿真运行状态数据。仿真运行状态监控模块可以监控的内容包括：

1. 仿真调度引擎进程名称、进程 ID、CPU 亲和性；

2. 仿真内核各模块状态（正常、异常）；

3. 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 亲和性等。

4.2.13.4.2. 模型运行状态监控

DDS 监控后端服务的模型运行状态监控模块中，通过订阅模型运行过程中发布的运行状态主题，实时向仿真综合管理平台提交模型运行状态数据。模型运行状态监控模块可以监控的内容包括各模型相关参数：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。

4.2.13.4.3. 仿真运行控制

DDS 监控后端服务的仿真运行控制模块中，通过发布仿真运行控制主题，实时向仿真引擎发送控制指令。仿真运行控制模块可以进行的控制有：

1. 控制引擎的暂停、继续与停止；

2. 控制某个线程的暂停、继续与停止；

4.2.13.4.4. 数据监控器插件管理

DDS 监控后端服务的数据监控器插件管理系统负责在每次仿真监控过程中加载所需的数据监控器插件。鉴于 DDS 监控后端服务所监控的模型数据构型可能会发生变化，因此对于不同的构型，必须有与特定的数据交互接口定义相对应的数据监控器。为了防止在数据交互接口发生变更时重新编译整个 DDS 监控后端服务，将数据监控器实现为插件形式，可以将 DDS 监控后端服务的核心功能与数据监控器的构建过程分离。

数据监控器插件具有的主要功能为：

1. 插件生命周期管理：

   1） 加载插件：加载自动化构建的插件；

   2） 卸载插件：安全地卸载当前加载的插件；

   3） 状态检查：检查插件是否已加载。

2. 监控器实例管理：

   1） 获取监控器：根据接口名称获取对应的数据监控器实例；

   2） 缓存机制：缓存已创建的监控器实例，避免重复创建；

   3） 支持接口查询：获取插件支持的所有接口列表。

3. 插件信息管理：

   1） 设置插件信息：设置从数据库接口配置构建的数据监控器插件信息；

   2） 插件路径构建：自动构建数据监控器插件库的完整路径。

4.2.13.4.5. 数据监控器抽象接口

DDS 监控后端服务的数据监控器抽象接口定义了数据监控器插件与 DDS 监控后端服务之间的标准接口，采用 C 风格接口设计以确保 ABI（应用程序二进制接口）兼容性。主要功能为：

1. 插件信息：定义了插件的基本信息，包括插件名称、描述信息和接口版本。接口版

本必须与 DDS 监控后端服务兼容，这是确保插件能够正常工作的关键。

2. C 风格函数指针类型定义。为了确保 ABI 兼容性，所有插件接口都使用 C 风格函数指针，包括：

1） 获取插件信息函数：返回插件的基本信息结构体指针，插件管理器通过此函数获取插件元数据。

2） 创建监控器实例函数：根据接口名称创建对应的监控器实例，返回智能指针类型的监控器对象。

3） 销毁监控器实例函数：根据接口名称销毁对应的监控器实例，用于资源清理。

4） 获取支持的接口列表函数：返回插件支持的所有接口名称数组，通过参数返回接口数量，主程序通过此函数了解插件的功能范围。

5） 释放函数：释放由插件分配的内存，防止内存泄漏。

4.2.13.4.6. 数据监控器生成

DDS 监控后端服务的数据监控器生成模块负责从数据库读取配置信息自动生成插件代码。它实现了从配置到可执行插件的完整自动化流程，是数据监控器插件的生成工具。主要功能有：

1. 数据库配置加载：

1） 从数据库中读取配置信息：

2） 根据构型 ID 获取构型名称作为插件名称；

3） 查询接口配置表获取接口信息；

2. 代码生成：生成插件的 C++ 源代码文件，包含：插件信息结构体定义、支持的接口列表、所有必需的接口函数实现、基于模板的监控器实例创建逻辑；

3. 构建系统生成：生成 CMake 构建文件，配置项目依赖、库链接关系、安装路径设置、编译选项；

4. 自动编译：创建构建目录、执行 CMake 配置、执行 Make 编译、执行 Make Install 安装。

4.2.13.4.7. 模型数据监控

DDS 监控后端服务的模型数据监控模块负责接收仿真综合管理平台传递的待监控交互接口名称，并从数据交互接口中获取序列化为字符串格式的交互接口数据，随后将这些数据返回给仿真综合管理平台。该模型数据监控模块具备以下功能：

1. 采用工厂模式创建并管理所需监控数据的主题监控器，确保每个主题仅对应一个监控器；

2. 监控器负责通过数据交互接口获取相应主题序列化为字符串的数据。

4.2.13.4.8. 模型数据采集

DDS 监控后端服务的模型数据采集模块，负责从仿真系统中采集指定数据并保存到 CSV 文件中。它支持多线程数据采集，能够根据脚本配置的频率和时长进行精确的数据记录。主要功能：

1. 解析.dcs 数据采集脚本文件，提取关键信息：
    1） 从脚本文件中读取需要采集的接口字段列表；
    2） 解析接口字段的数组维度信息（支持一维和二维数组）；
    3） 解析采集时长和频率设置；
    4） 确定输出文件名和路径。
2. 多线程数据采集
    1） 启动独立的数据采集线程；
    2） 按照指定的频率进行数据采集；
    3） 使用高精度时钟确保采集时间间隔的准确性；
    4） 在指定时长内持续采集数据；
    5） 自动停止采集并清理资源。
3. 数据格式处理：
    1） 支持标量数据、一维数组和二维数组的提取；
    2） 根据配置的字段索引提取数组中的特定元素；
    3） 将数据格式化为 CSV 格式并写入文件；
    4） 处理缺失数据，用零值填充。
4. 文件管理：
    1） 自动创建输出 CSV 文件；
    2） 写入表头信息，包含时间戳和所有数据字段；
    3） 处理文件重名情况，自动添加序号后缀；
    4） 在采集结束时自动关闭文件。

4.2.13.4.9. 模型数据注入

DDS 监控后端服务的模型数据注入模块具有三种工作模式：单次注入、连续注入和 CSV 注入。

1. 单次注入：直接使用数据监控器一次性向数据交互接口注入字符串形式的数据，数据交互接口进行反序列化后发布至 DDS 中。

2. 连续注入：使用数据注入线程持续向数据交互接口注入数据。它提供了可控制频率的数据注入能力，支持动态更新注入数据和频率。具体功能为：

1） 数据注入控制：

a. 持续向指定的数据交互接口注入数据；

b. 在独立线程中运行，不阻塞主程序；

c. 按照指定的频率定期注入数据；

d. 支持动态更新要注入的数据内容；

e. 支持运行时调整注入频率；

f. 提供完整的线程控制功能;

2） 精确频率控制：

a. 使用高精度时间控制实现准确的注入频率；

b. 根据频率计算睡眠时间间隔；

c. 使用条件变量避免 CPU 空转；

d. 支持毫秒级的时间精度。

3. CSV 注入：使用基于 CSV 文件的数据注入线程从 CSV 文件中读取数据并按照时间戳精确地向数据监控器注入数据。具体功能为：

1） CSV 文件解析：

a. 读取 CSV 文件头，解析数据字段信息；

b. 支持复杂的字段格式，包括数组索引；

c. 自动识别字段的数据类型和维度；

d. 建立字段与监控器接口的映射关系。

2） 时间戳控制的数据注入：

a. 从 CSV 文件中读取时间戳；

b. 根据时间戳计算下一次执行时间；

c. 解析对应的数据行内容；

d. 处理文件结束情况，自动停止注入。

3） 复杂数据结构支持：

a. 标量数据：直接字段值；

b. 一维数组：字段名(索引)格式；

c. 二维数组：字段名(行_列)格式；

d. 自动处理数组索引的转换和映射。

3） 精确频率控制：

a. 使用高精度时钟和条件变量；

b. 根据 CSV 中的时间戳进行精确的时间同步；

c. 支持毫秒级的时间精度控制；

d. 避免时间漂移和累积误差。

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 32 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。

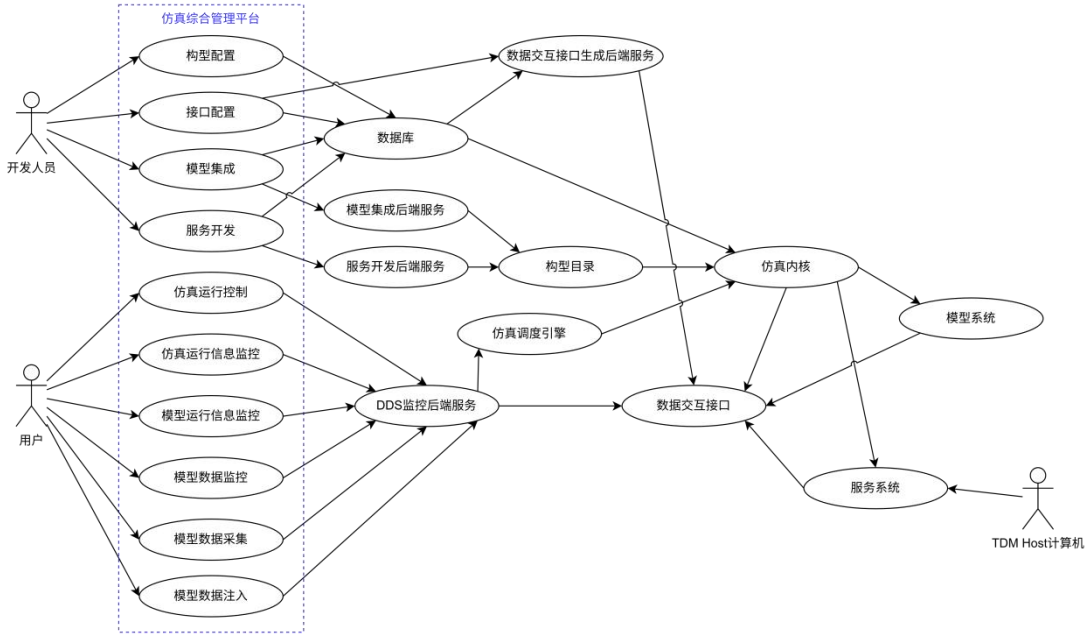

图 32 “玄鸟”架构用例图

4.3.2. 运行时序图

“玄鸟”架构的运行时序图如图 33 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。

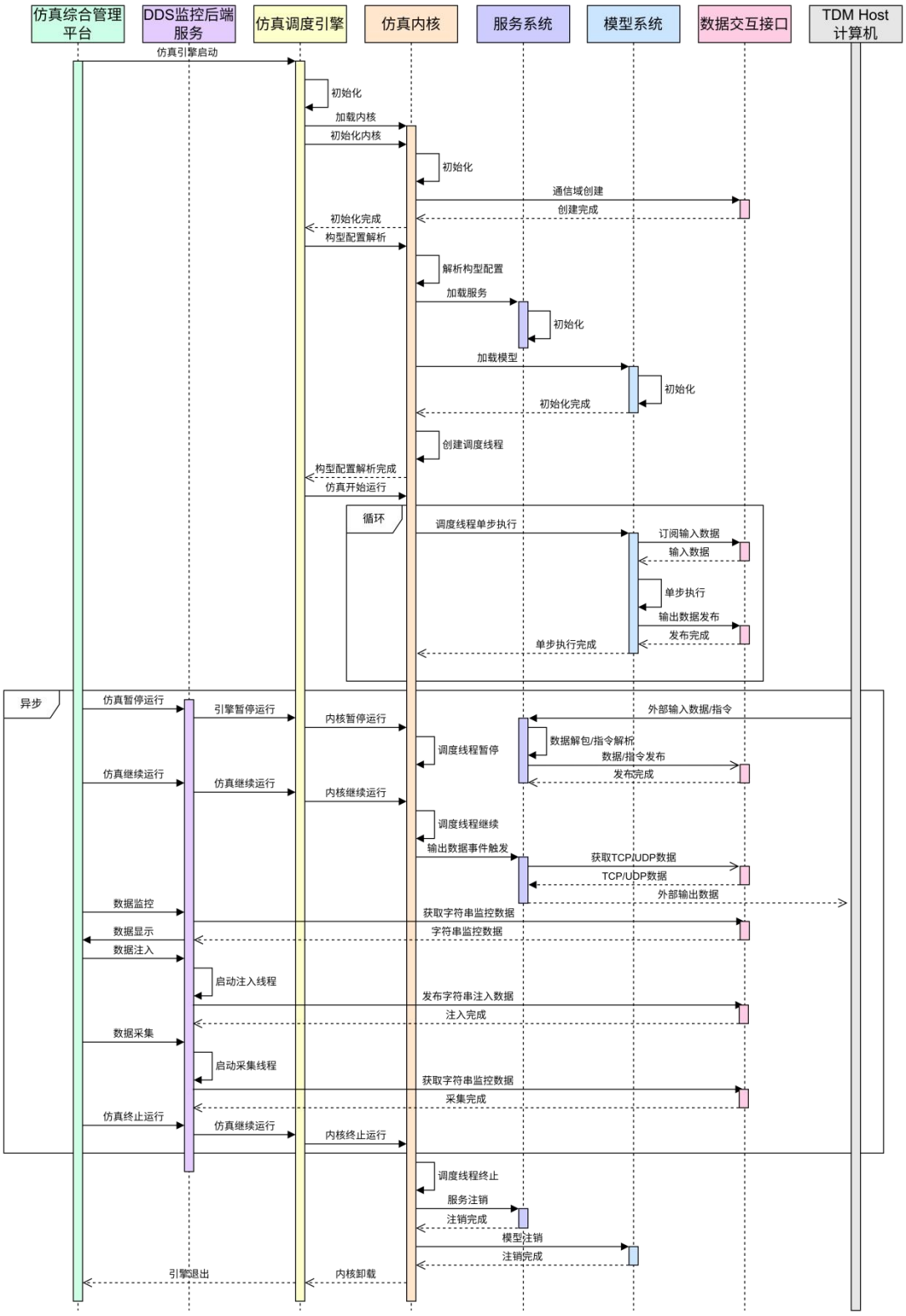

图 33 “玄鸟”架构运行时序图

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 34 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。

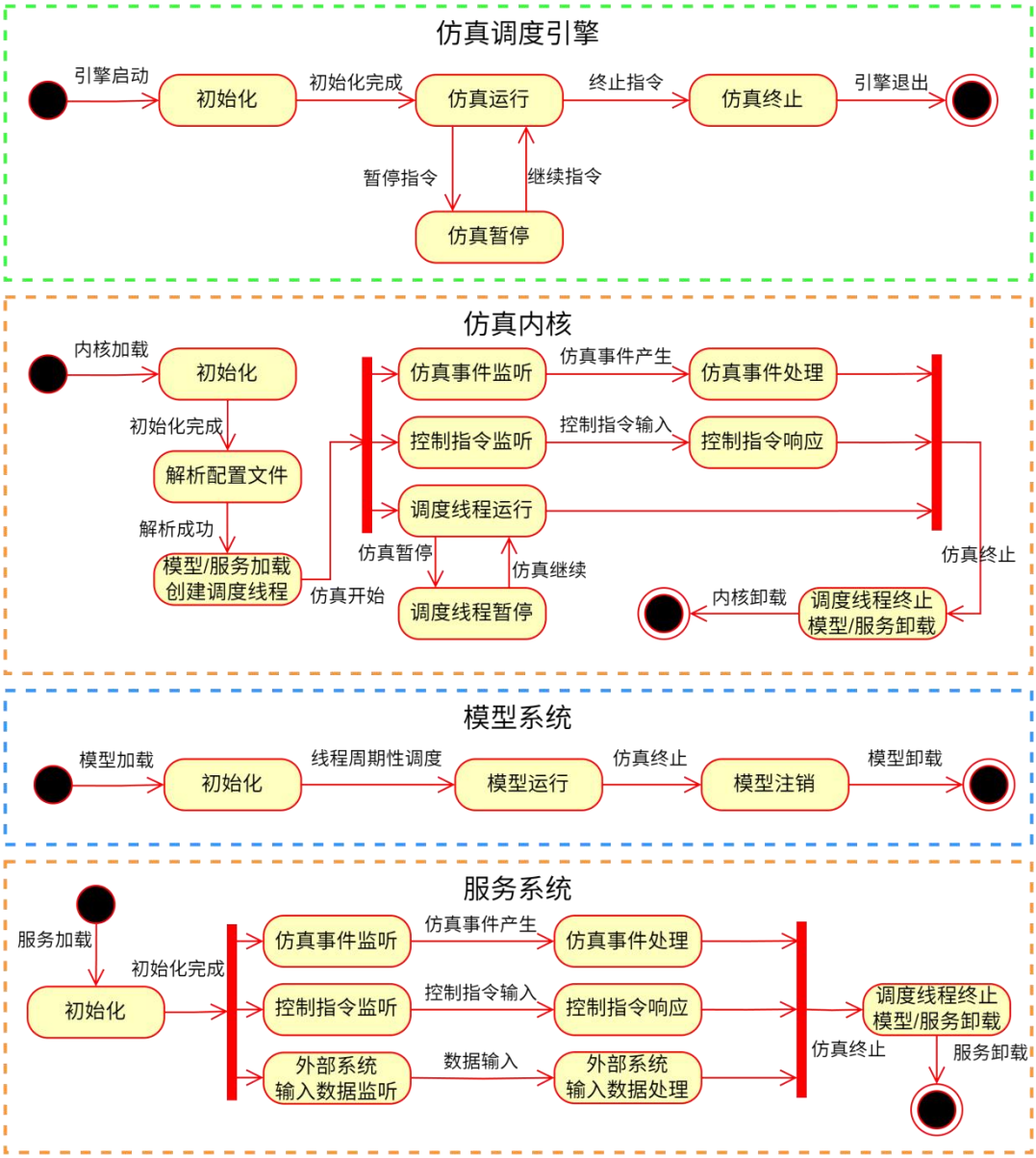

图 34 “玄鸟”架构在仿真运行阶段的状态转换图

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 5，外部接口见表 6，人机交互接口见表 7。

表 5 “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.	构型配置	XNConfigurationInterface	V1.0	仿真综合管理平台 仿真调度引擎 仿真内核 数据库
2.	接口配置	XNInterfaceConfigInterface	V1.0	仿真综合管理平台 数据交互接口生成后端服务 数据交互接口 数据库
3.	模型配置	XNModelConfigInterface	V1.0	仿真综合管理平台 仿真内核 模型系统 数据库
4.	服务配置	XNServiceConfigInterface	V1.0	仿真综合管理平台 仿真内核 服务系统 数据库
5.	仿真运行控制	XNRuntimeControlInterface	V1.0	仿真综合管理平台 DDS 监控后端服务 仿真调度引擎
6.	仿真运行状态	XNRuntimeStatusInterface	V1.0	仿真综合管理平台 DDS 监控后端服务 仿真调度引擎 仿真内核
7.	线程运行状态	XNThreadStatusInterface	V1.0	仿真调度引擎 DDS 监控后端服务 仿真内核
8.	模型运行状态	XNModelStatusInterface	V1.0	仿真综合管理平台 DDS 监控后端服务 仿真内核
9.	模型数据交互	XNDDSDataInterface	V1.0	仿真综合管理平台 DDS 监控后端服务 模型系统 服务系统 数据交互接口

表 6 “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	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	XNARINC429Interface	V1.0	服务系统 TDM Host 计算机
3.	ARINC 664 虚拟航空总线通信	XNARINC664Interface	V1.0	服务系统 TDM Host 计算机
4.	ARINC 708 虚拟航空总线通信	XNARINC708Interface	V1.0	服务系统 TDM Host 计算机
5.	ARINC 825 虚拟航空总线通信	XNARINC825Interface	V1.0	服务系统 TDM Host 计算机
6.	离散量通信	XNDiscreteDataInterface	V1.0	服务系统 TDM Host 计算机

表 7 “玄鸟”架构人机交互接口标识表

序号	接口名称	标识符	版本	相关实体
1.	用户登陆界面交互接口	XNLogin	V1.0	仿真综合管理平台用户
2.	用户注册界面交互接口	XNRegister	V1.0	仿真综合管理平台用户
3.	系统概览界面交互接口	XNOverview	V1.0	仿真综合管理平台用户
4.	更新记录界面交互接口	XNUpdateInfo	V1.0	仿真综合管理平台用户
5.	帮助界面交互接口	XNHelp	V1.0	仿真综合管理平台用户
6.	问答界面交互接口	XNQAndA	V1.0	仿真综合管理平台用户
7.	构型配置界面交互接口	XNConfigEdit	V1.0	仿真综合管理平台用户
8.	接口配置界面交互接口	XNInterfaceEdit	V1.0	仿真综合管理平台用户
9.	模型集成界面交互接口	XNModelEdit	V1.0	仿真综合管理平台用户
10.	服务开发界面交互接口	XNServiceEdit	V1.0	仿真综合管理平台用户
11.	仿真运行界面交互接口	XNRunSim	V1.0	仿真综合管理平台用户
12.	运行日志界面交互接口	XNSimLog	V1.0	仿真综合管理平台用户

13.	资源监控界面交互接口	XNResourceMonitor	V1.0	仿真综合管理平台用户
14.	仿真监控界面交互接口	XNSystemMonitor	V1.0	仿真综合管理平台用户
15.	模型监控界面交互接口	XNModelMonitor	V1.0	仿真综合管理平台用户
16.	数据监控界面交互接口	XNDataMonitor	V1.0	仿真综合管理平台用户
17.	数据采集界面交互接口	XNDataCollect	V1.0	仿真综合管理平台用户
18.	网络监控界面交互接口	XNNetMonitor	V1.0	仿真综合管理平台用户
19.	QTG 界面交互接口	XNQTG	V1.0	仿真综合管理平台用户
20.	个人中心界面交互接口	XNProfileCenter	V1.0	仿真综合管理平台用户
21.	用户管理界面交互接口	XNUserManager	V1.0	仿真综合管理平台用户

4.4.2. XNConfigurationInterface

4.4.2.1. 概述

1. 接口名称：构型配置
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：XNConfiguration
        b. 非技术（自然语言）名称：构型配置；
        c. 缩写：XNCfg
    2) 数据元素类型：SQL 数据表；
4. 通信方法：SQLite 数据库读写；
5. 通信协议：SQL 语句。

4.4.2.2. 数据元素定义

构型配置（XNConfiguration）是在数据库中存储的一体化二进制数据包软件运行环境及运行配置相关的参数，它包含以下内容：

1. 构型标识：

1） 构型 ID（ConfID）：每个构型都具有唯一的 ID；

2） 机型（PlaneName）：构型所属的机型；

3） 构型名称（ConfName）。

2. 运行环境参数：

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

2） 操作系统版本（OSVersion）：仿真运行所在计算机的操作系统版本；

3） 实时补丁版本（RTXVersion）：仿真运行所在计算机的操作系统内核/补丁版本，例如：RTX64（Windows）或 5.10.0-32-rt-amd64（Linux）；

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

5） DDS 通信域 ID（DomainID）：用于 DDS 通信的域 ID；

6） 控制台输出控制（ConsoleDebug/ConsoleInfo/ConsoleWarning/ConsoleError）：控制仿真运行过程中是否向控制台打印调试/消息/警告/错误日志消息；

7） 日志输出控制（LogDebug/LogInfo/LogWarning/LogError）：控制仿真运行过程中是否向日志文件中记录调试/消息/警告/错误日志消息；

3. 加载模型组列表（LoadModelGroups）列出本次仿真需要创建的调度线程信息，每个模型组信息包括：

1） 模型组 ID（GroupID）：每个模型组具有全局唯一的 ID；

2） 构型 ID（ConfID）：模型组所属构型的 ID；

3） 模型组名称（GroupName）；

4） 运行频率（Freq）；

5） CPU 亲和型（CPUAff）；

6） 优先级（Priority）。

4. 加载模型列表（LoadModels）列出各个模型组中需要动态加载的所有模型信息，每个模型的信息包括：

1） 模型组 ID（GroupID）：模型所属的模型组 ID；

2） 模型 C++类名称（ClassName）；

3） 版本号（ModelVersion）；

4） 模型名称（ModelName）。

5. 加载服务列表（LoadServices）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） 构型 ID（ConfID）：服务所属构型的 ID；

2） 服务 C++类名称（ClassName）；

3） 版本号（ServiceVersion）；

4） 服务名称（ServiceName）。

4.4.3. XNInterfaceConfigInterface

4.4.3.1. 概述

1. 接口名称：接口配置
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：XNInterfaceConfig
      b. 非技术（自然语言）名称：接口配置；
      c. 缩写：XNInterCfg
   2) 数据元素类型：SQL 数据表；
4. 通信方法：SQLite 数据库读写；
5. 通信协议：SQL 语句。

4.4.3.2. 数据元素定义

接口配置（XNInterfaceConfig）是在数据库中存储的二进制数据包模型间所需交互的数据接口信息，包含以下内容：

1. 接口名称（InterfaceName）；
2. 系统名称（SystemName）：默认为“XNSim”；
3. 机型名称（PlaneName）；
4. ATA 章节名称（ATAName）；
5. 接口所属结构体名称（ModelStructName）；
6. 接口数据类型（InterfaceType）；
7. 接口是否是数组类型（InterfaceIsArray）；

8. 接口数组的两个维度大小（InterfaceArraySize_1、InterfaceArraySize_2）；

9. 接口注释（InterfaceNotes）。

4.4.4. XNModelConfigInterface

4.4.4.1. 概述

1. 接口名称：模型配置

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：XNModelConfig

      b. 非技术（自然语言）名称：模型配置；

      c. 缩写：XNMdlCfg

   2) 数据元素类型：SQL 数据表；

4. 通信方法：SQLite 数据库读写；

5. 通信协议：SQL 语句。

4.4.4.2. 数据元素定义

模型配置（XNModelConfig）是在数据库中存储的集成二进制数据包模型与模型运行相关的参数，它包含以下内容：

1. 模型基本信息：

   1） 机型名称（PlaneName）：模型所属机型名称；

   2） 构型 ID（ConfID）：模型所属构型 ID；

   3） 模型 C++类名（ClassName）；

   4） 模型名称（Name）；

   5） 版本号（Version）；

   6） 作者（Author）；

   7） 描述（Description）；

   8） 创建时间（CreateTime）；

   9） 修改时间（ChangeTime）；

   10） 运行频率组（RunFreqGroup）：“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”；

11） 运行节点（RunNode）：函数被调度运行的节点号，举例说明：当运行频率组为 1 时，函数以线程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。运行节点决定该函数在两个调度周期的前一个周期（为 0）还是后一个周期（为 1）进行调度。因此运行节点的值需小于 2 的运行频率组次方。

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

2. 二进制数据包模型集成相关：

1） 二进制数据包路径（DataPackagePath）；

2） 二进制数据包动态链接库名称（DataPackageName）；

3） 二进制数据包头文件名称（DataPackageHeaderName）；

4） 二进制数据包入口点函数名称（DataPackageEntryPoint）；

5） 二进制数据包入口点参数类型（DataPackageInterfaceName）；

6） 输入接口对应关系（InputStruct）：存储输入接口的结构体定义与主题定义的对应关系，用 JSON 字符串存储；

7） 输出接口对应关系（OutputStruct）：存储输出接口的结构体定义与主题定义的对应关系，用 JSON 字符串存储；

8） 心跳接口对应关系（HeartStruct）：存储心跳接口的结构体定义与主题定义的对应关系，用 JSON 字符串存储；

3. 指令列表（CmdList）以 JSON 数组的形式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（CmdName）：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（CmdDscr）：解释该条指令的含义；

3） 响应函数（CmdCall）：响应该条指令的函数名称。

4.4.5. XNServiceConfigInterface

4.4.5.1. 概述

1. 接口名称：服务配置

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：XNServiceConfig

b. 非技术（自然语言）名称：服务配置；

c. 缩写：XNSrvCfg

3) 数据元素类型：SQL 数据表；

4. 通信方法：SQLite 数据库读写；

5. 通信协议：SQL 语句。

4.4.5.2. 数据元素定义

服务配置（XNServiceConfig）是在数据库中存储的服务运行所需的参数，它包含以下内容：

1. 服务基本信息：

1） 服务 C++类名（ClassName）；

2） 服务名称（Name）；

3） 版本号（Version）；

4） 作者（Author）；

5） 创建时间（CreateTime）；

6） 修改时间（ChangeTime）；

7） 描述（Description）。

2. 指令列表（CmdList）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（CmdName）；

2） 指令含义（CmdDscr）；

3） 响应函数（CmdCall）：响应该条指令的函数名称。

3. 自定义参数（OtherParam）中列出了服务自身功能相关的一些参数，以 JSON 字符串形式存储，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。

4.4.6. 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 DDS 监控后端服务通过 Fast DDS 发送。仿真运行控制指令结构体组成内容如表 8 所示。

表 8 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综合管理平台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 9 所示。

表 9 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.	仿真调度引擎进程 ID	XNEngineID	INT32	/
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 10
6.	仿真运行信息	XNRuntimeSt	struct	结构体定义见表 11

表 10 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.	构型管理器加	XNSDStatus	INT32	0-未加载，1-正常，其它-异常

	载状态			
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 11 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）用于向仿真综合管理平台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 12 所示。

表 12 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）用于向仿真综合管理平台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 13 所示。

表 13 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，同时使 DDS 监控后端服务可以监控交互数据。模型 DDS 数据交互结构体的内容由二进制数据包模型的 ICD 文件

中定义。

4.4.11. XNCommandInterface

4.4.11.1. 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。

4.4.11.2. 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 14 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（XNModelConfigFile）和服务配置（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

4.4.12.XNARINC429Interface

4.4.12.1. 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：
    1) 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
    2) 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。

4.4.12.2. 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 15 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

4.4.13.XNARINC664Interface

4.4.13.1. 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。

4.4.13.2. 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 16 所示。

表 16 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 35 所示。

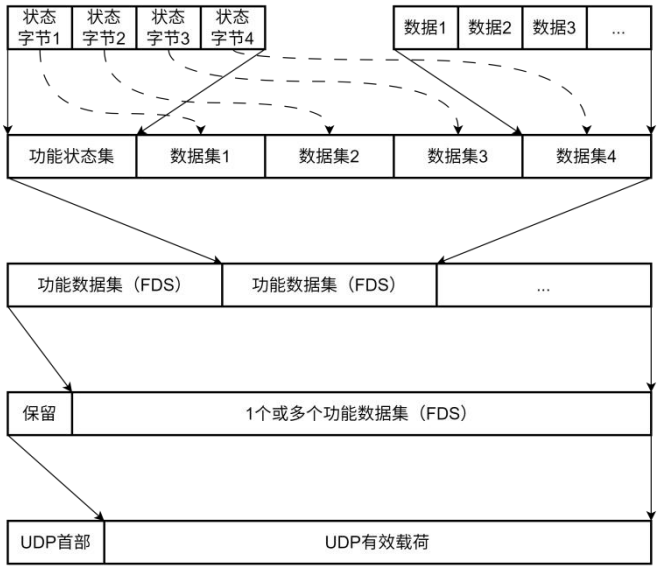

图 35 AFDX 数据帧功能数据集结构

FSB 的定义如表 17 所示。

表 17 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

4.4.14.XNARINC708Interface

4.4.14.1. 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。

4.4.14.2. 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 18 所示。

表 18 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。

4.4.15.XNARINC825Interface

4.4.15.1. 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。

4.4.15.2. 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 19 所示。

表 19 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 20
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 20 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

4.4.16.XNDiscreteDataInterface

4.4.16.1. 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。

4.4.16.2. 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 21 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

4.4.17.XNLogin

4.4.17.1. 概述

1. 接口名称：用户登陆界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.17.2. 界面布局

用户登陆界面的布局如图 36 所示。

图 36 用户登陆界面布局

4.4.17.3. 交互元素

用户登陆界面的关键控件列表如表 22 所示。

表 22 用户登陆界面交互元素

控件类型	ID	状态	行为规则
注册页面切换按钮	btn_register	激活/未激活	点击后切换至用户注册界面
用户名输入框	edit_username	无	点击后可以输入用户名
密码输入框	edit_password	隐藏/显示	点击后可以输入密码
密码显示切换按钮	btn_pwdvis	激活/未激活	点击后可以切换密码输入框中密码的隐藏/显示状态
登陆按钮	btn_login	无	点击后触发用户名和密码校验： 1. 校验通过显示欢迎信息，跳转至概览页面； 2. 校验失败提示用户名或密码错误。

4.4.18.XNRegister

4.4.18.1. 概述

1. 接口名称：用户注册界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.18.2. 界面布局

用户注册界面的布局如图 37 所示。

图 37 用户注册界面布局

4.4.18.3. 交互元素

用户注册界面的关键控件列表如表 23 所示。

表 23 用户注册界面交互元素

控件类型	ID	状态	行为规则
登陆页面切换按钮	btn_login	激活/未激活	点击后切换至用户登陆界面
用户名输入框	edit_username	无	点击后可以输入用户名
密码输入框	edit_password	隐藏/显示	点击后可以输入密码
密码显示切换按钮	btn_pwdvis	激活/未激活	点击后可以切换密码输入框中密码的隐藏/显示状态
确认密码输入框	edit_pwdrequire	隐藏/显示	点击后可以输入确认密码
确认密码显示切换按钮	btn_pwdreqvis	激活/未激活	点击后可以切换确认密码输入框中密码的隐藏/显示状态
姓名输入框	edit_name	无	点击后可以输入姓名
电话输入框	edit_number	无	点击后可以输入电话
邮箱输入框	edit_email	无	点击后可以输入邮箱
部门输入框	edit_department	无	点击后可以输入部门
职位输入框	edit_position	无	点击后可以输入职位
注册按钮	btn_login	无	点击后触发用户名和密码校验： 1. 校验通过显示注册成功； 2. 校验失败提示用户名或密码无效。

4.4.19.XNOverview

4.4.19.1. 概述

1. 接口名称：系统概览界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.19.2. 界面布局

系统概览界面的布局如图 38 所示。

图 38 系统概览界面布局

4.4.19.3. 交互元素

系统概览界面的关键控件列表如表 24 所示。

表 24 系统概览界面交互元素

控件类型	ID	状态	行为规则
简介卡片	card_introduce	无	显示软件简介
关于卡片	card_about	无	显示软件关于信息
版本信息卡片	card_version	无	显示软件当前版本
时间卡片	card_time	无	显示当前系统时间
日期卡片	card_date	无	显示当前系统日期
更新记录卡片	card_uphistory	无	显示最近更新的历史记录
Q&A 按钮	btn_QAndA	无	点击后跳转问答界面
帮助按钮	btn_help	无	点击后跳转帮助界面
更新记录按钮	btn_update	无	点击后跳转更新记录页面

4.4.20.XNUpdateInfo

4.4.20.1. 概述

1. 接口名称：更新记录界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.20.2. 界面布局

更新记录界面的布局如图 39 所示。

图 39 更新记录界面布局

4.4.20.3. 交互元素

更新记录界面的关键控件列表如表 25 所示。

表 25 更新记录界面交互元素

控件类型	ID	状态	行为规则
更新记录列表	list_update	无	列出软件所有的更新记录
提交更新按钮	btn_commit	无	点击后可以提交更新记录
详细记录查看按钮	btn_detail	激活/未激活	点击后可以切换显示单条更新记录的详细信息
分页切换按钮	btn_page	无	点击后可以切换更新记录的分页页面

4.4.21.XNHelp

4.4.21.1. 概述

1. 接口名称：帮助界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.21.2. 界面布局

帮助界面的布局如图 40 所示。

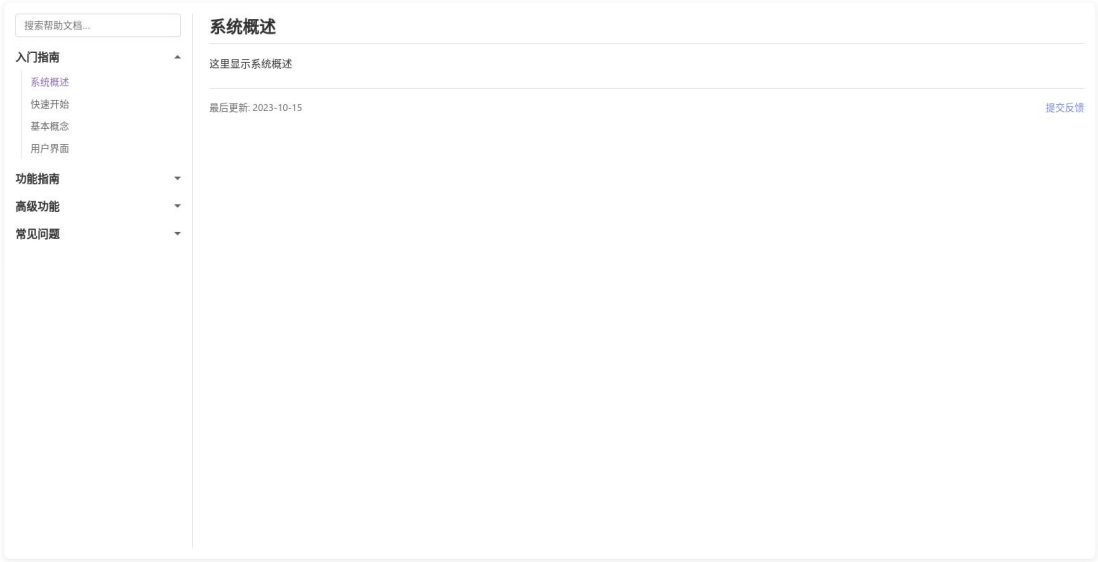

图 40 帮助界面布局

4.4.21.3. 交互元素

帮助界面的关键控件列表如表 26 所示。

表 26 帮助界面交互元素

控件类型	ID	状态	行为规则
帮助搜索框	edit_search	无	输入后可以搜索帮助文档内容
帮助索引列表	list_helpindex	无	点击对应的列表条目可以打开对应的帮助文档
帮助文档显示页面	page_helpdoc	无	显示对应条目的帮助文档

4.4.22.XNQAndA

4.4.22.1. 概述

1. 接口名称：问答界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.22.2. 界面布局

问答界面的布局如图 41 所示。

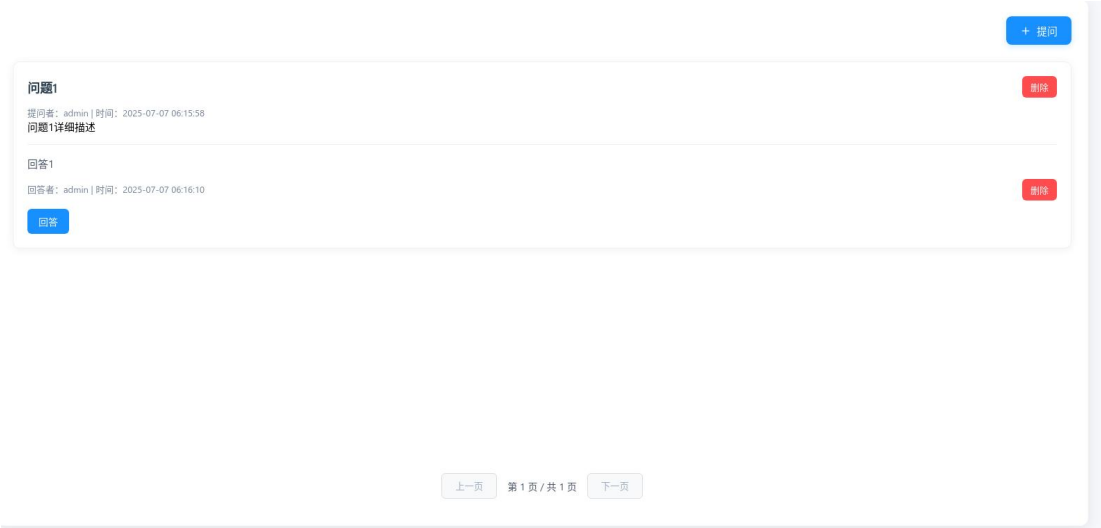

图 41 问答界面布局

4.4.22.3. 交互元素

问答界面的关键控件列表如表 27 所示。

表 27 问答界面交互元素

控件类型	ID	状态	行为规则
提问按钮	btn_question	无	点击后可以进行提问
问题列表	list_question	无	列出所有问题及回答
回答按钮	btn_answer	无	点击后可以进行回答
删除问题按钮	btn_deleteq	无	点击后可以删除问题及其回答
删除回答按钮	btn_deletea	无	点击后删除回答
分页切换按钮	btn_page	无	点击后可以切换问答的分页页面

4.4.23.XNConfigEdit

4.4.23.1. 概述

1. 接口名称：构型配置界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.23.2. 界面布局

构型配置界面的布局如图 42 所示。

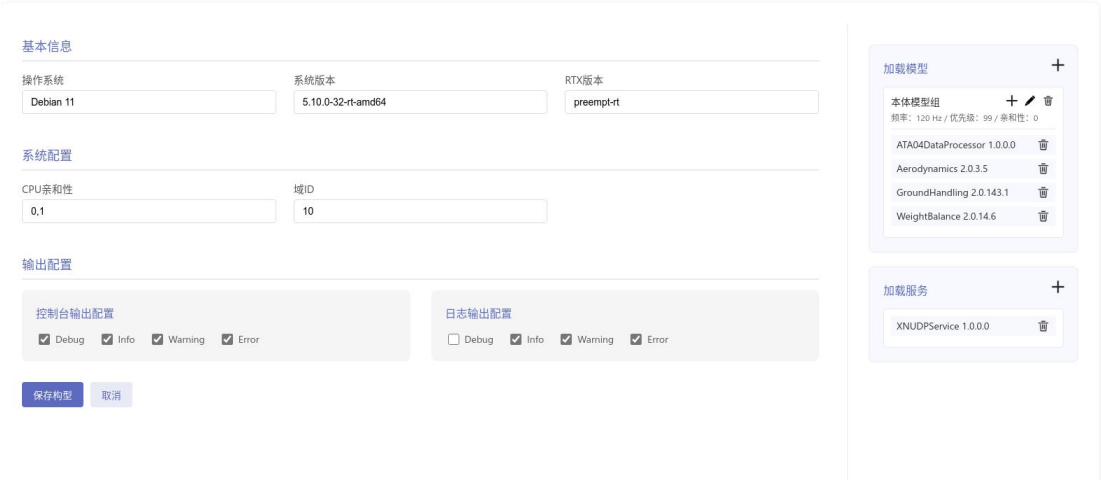

图 42 构型配置界面布局

4.4.23.3. 交互元素

构型配置界面的关键控件列表如表 28 所示。

表 28 构型配置界面交互元素

控件类型	ID	状态	行为规则
操作系统输入框	edit_osname	无	点击后可以编辑操作系统名称
系统版本输入框	edit_osvision	无	点击后可以编辑操作系统版本
RTX 版本输入框	edit_rtxvision	无	点击后可以编辑 RTX 补丁版本
CPU 亲和性输入框	edit_cpuaff	无	点击后可以编辑 CPU 亲和性
域 ID 输入框	edit_domainid	无	点击后可以编辑通信域 ID
控制台输出配置	edit_console	无	点击后可以修改控制台输出设置
日志输出配置	edit_log	无	点击后可以修改日志输出设置
保存构型按钮	btn_save	无	点击后保存所有设置
取消按钮	btn_cancel	无	点击后撤销所有修改
添加模型组按钮	btn_addgroup	无	点击后添加一个模型组
编辑模型组按钮	btn_editgroup	无	点击后可以编辑模型组参数
删除模型组按钮	btn_deletegroup	无	点击后删除该模型组
添加模型按钮	btn_addmodel	无	点击后可以为模型组添加一个模型
删除模型按钮	btn_deletemodel	无	点击后删除该模型
添加服务按钮	btn_addserver	无	点击后添加一个服务
删除服务按钮	btn_deleteserver	无	点击后删除该服务

4.4.24.XNInterfaceEdit

4.4.24.1. 概述

1. 接口名称：接口配置界面交互接口

2. 接口类型：人机交互接口

3. 适用终端：Web 浏览器。

4.4.24.2. 界面布局

接口配置界面的布局如图 43 所示。

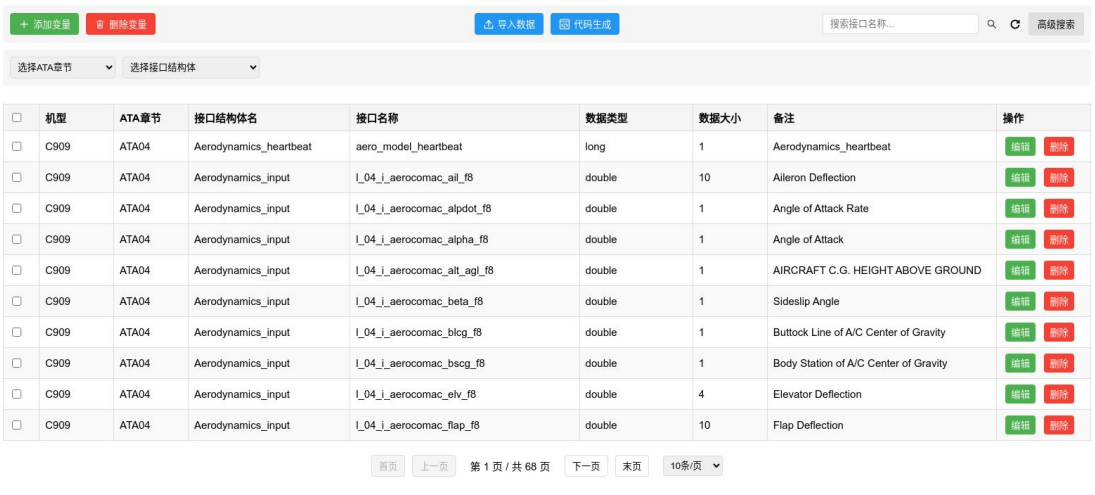

图 43 接口配置界面布局

4.4.24.3. 交互元素

接口配置界面的关键控件列表如表 29 所示。

表 29 接口配置界面交互元素

控件类型	ID	状态	行为规则
添加变量按钮	btn_addinter	无	点击后可以向接口列表添加一条数据
删除变量按钮	btn_deleteinter	无	点击后可以删除接口列表中选择的数据
导入数据按钮	btn_importdata	无	点击后可以从 ICD 导入接口数据
代码生成按钮	btn_codegen	无	点击后将调用数据交互接口后端服务进行生成
搜索输入框	edit_search	无	点击后可以编辑搜索内容
搜索按钮	btn_search	无	点击后将根据搜索内容筛选接口列表数据
重置搜索按钮	btn_reset	无	点击后将清除搜索输入框，并取消接口列表的筛选
高级搜索按钮	btn_adsearch	激活/未激活	激活时将显示 ATA 章节选择按钮和接口结构体选择按钮
ATA 章节选择按钮	btn_ataselect	无	点击后可以选择 ATA 章节并筛选接口列表数据
接口结构体选择按	btn_structselect	无	点击后可以选择接口结构体并筛选

钮			接口列表数据
接口表	table_inter	无	以表格形式显示接口的所属机型、ATA 章节、结构体名、名称、数据类型、数据大小、备注
选择接口按钮	btn_select	激活/未激活	激活时选中接口
编辑接口按钮	btn_edit	无	点击后可以编辑该条接口数据
删除接口按钮	btn_delete	无	点击后可以删除该条接口数据
切换分页按钮	btn_page	无	点击后可以切换接口数据的分页页面

4.4.25.XNModelEdit

4.4.25.1. 概述

1. 接口名称：模型集成界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.25.2. 界面布局

模型集成界面的布局如图 44 所示。

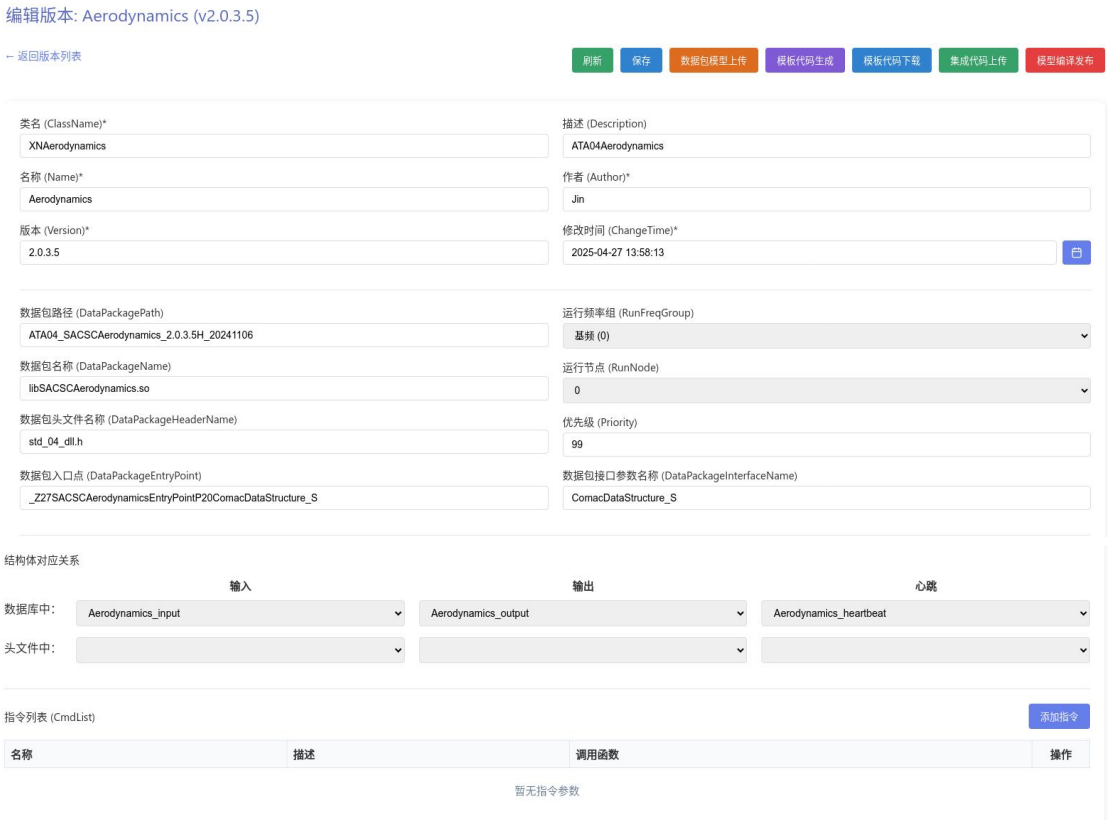

图 44 模型集成界面布局

4.4.25.3. 交互元素

模型集成界面的关键控件列表如表 30 所示。

表 30 模型集成界面交互元素

控件类型	ID	状态	行为规则
刷新按钮	btn_refresh	无	点击将重新从数据库获取数据
保存按钮	btn_save	无	点击将修改保存到数据库
数据包模型上传按钮	btn_uploadpkg	无	点击后将上传二进制数据包模型并自动解析，然后根据解析结果自动填充数据包路径输入框、数据包名称输入框、数据包头文件名输入框、数据包入口点输入框、数据包接口参数名称输入框
模板代码生成按钮	btn_codegen	无	点击后将调用模型集成后端服务生成集成模板代码
模板代码下载按钮	btn_download	无	点击后将下载集成模板代码的压缩包
集成代码上传按钮	btn_upload	无	点击后将上传集成代码的压缩包
模型编译发布	btn_compile	无	点击后将调用模型集成后端服务编译集成代码并发布
类名输入框	edit_class	无	自动填充，不可编辑
名称输入框	edit_name	无	自动填充，不可编辑
版本输入框	edit_version	无	点击后可以编辑版本号
描述输入框	edit_descrip	无	点击后可以编辑模型描述
作者输入框	edit_author	无	点击后可以编辑作者
修改时间输入框	edit_chtime	无	点击后可以编辑修改时间
修改时间选择按钮	btn_time	无	点击后可以选择一个修改时间
数据包路径输入框	edit_pkgpath	无	可以手动编辑，也可以上传二进制数据包模型以自动填充
数据包名称输入框	edit_pkgname	无	可以手动编辑，也可以上传二进制数据包模型以自动填充
数据包头文件名输入框	edit_headname	无	可以手动编辑，也可以上传二进制数据包模型以自动填充
数据包入口点输入框	edit_entry	无	可以手动编辑，也可以上传二进制数据包模型以自动填充
数据包接口参数名称输入框	edit_param	无	可以手动编辑，也可以上传二进制数据包模型以自动填充
运行频率组选择按钮	btn_freq	无	点击后选择一个运行频率组
运行节点选择按钮	btn_node	无	点击后选择一个运行节点
优先级输入框	edit_priority	无	点击后可以编辑运行优先级
输入结构体选择按	btn_input	无	点击后可以选择输入结构体的主题

钮			名称与头文件中定义的对应关系
输出结构体选择按钮	btn_output	无	点击后可以选择输出结构体的主题名称与头文件中定义的对应关系
心跳结构体选择按钮	btn_heart	无	点击后可以选择心跳结构体的主题名称与头文件中定义的对应关系
指令列表	list_cmd	无	显示模型支持的指令列表
添加指令按钮	btn_addcmd	无	点击后可以添加一条指令
删除指令按钮	btn_deletecmd	无	点击后可以删除一条指令
编辑指令按钮	btn_editcmd	无	点击后可以修改一条指令

4.4.26.XNServiceEdit

4.4.26.1. 概述

1. 接口名称：服务开发界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.26.2. 界面布局

服务开发界面的布局如图 45 所示。

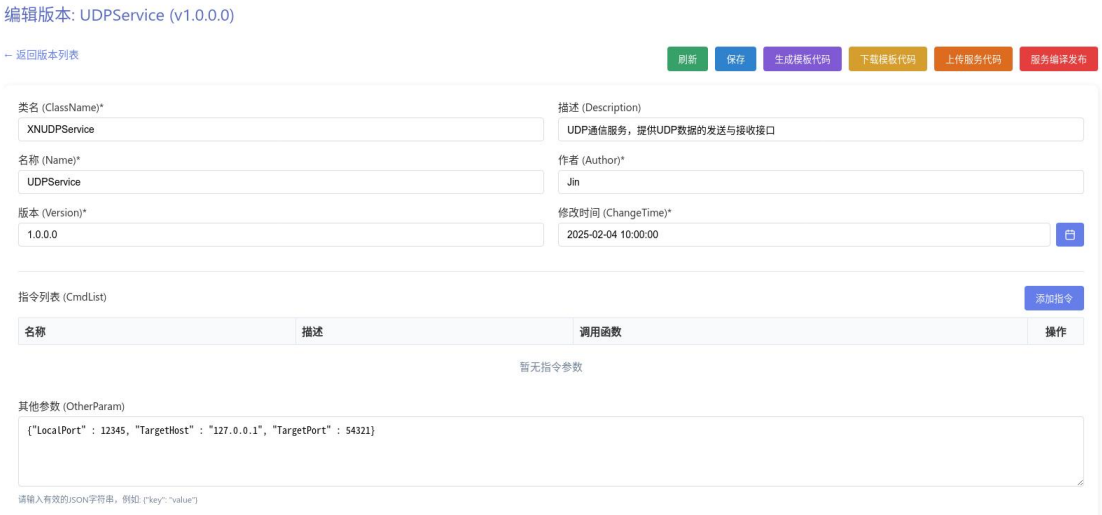

图 45 服务开发界面布局

4.4.26.3. 交互元素

服务开发界面的关键控件列表如表 31 所示。

表 31 服务开发界面交互元素

控件类型	ID	状态	行为规则
刷新按钮	btn_refresh	无	点击将重新从数据库获取数据
保存按钮	btn_save	无	点击将修改保存到数据库
模板代码生成按钮	btn_codegen	无	点击后将调用服务开发后端服务生成服务模板代码
模板代码下载按钮	btn_download	无	点击后将下载服务模板代码的压缩包
服务代码上传按钮	btn_upload	无	点击后将上传服务代码的压缩包
服务编译发布	btn_compile	无	点击后将调用服务开发后端服务编译服务代码并发布
类名输入框	edit_class	无	自动填充，不可编辑
名称输入框	edit_name	无	自动填充，不可编辑
版本输入框	edit_version	无	点击后可以编辑版本号
描述输入框	edit_descrip	无	点击后可以编辑服务描述
作者输入框	edit_author	无	点击后可以编辑作者
修改时间输入框	edit_chtime	无	点击后可以编辑修改时间
修改时间选择按钮	btn_time	无	点击后可以选择一个修改时间
指令列表	list_cmd	无	显示模型支持的指令列表
添加指令按钮	btn_addcmd	无	点击后可以添加一条指令
删除指令按钮	btn_deletecmd	无	点击后可以删除一条指令
编辑指令按钮	btn_editcmd	无	点击后可以修改一条指令
其它参数输入框	btn_otherparam	无	点击后可以编辑其它参数，必须为 JSON 字符串格式

4.4.27.XNRunSim

4.4.27.1. 概述

1. 接口名称：仿真运行界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.27.2. 界面布局

仿真运行界面的布局如图 46 所示。

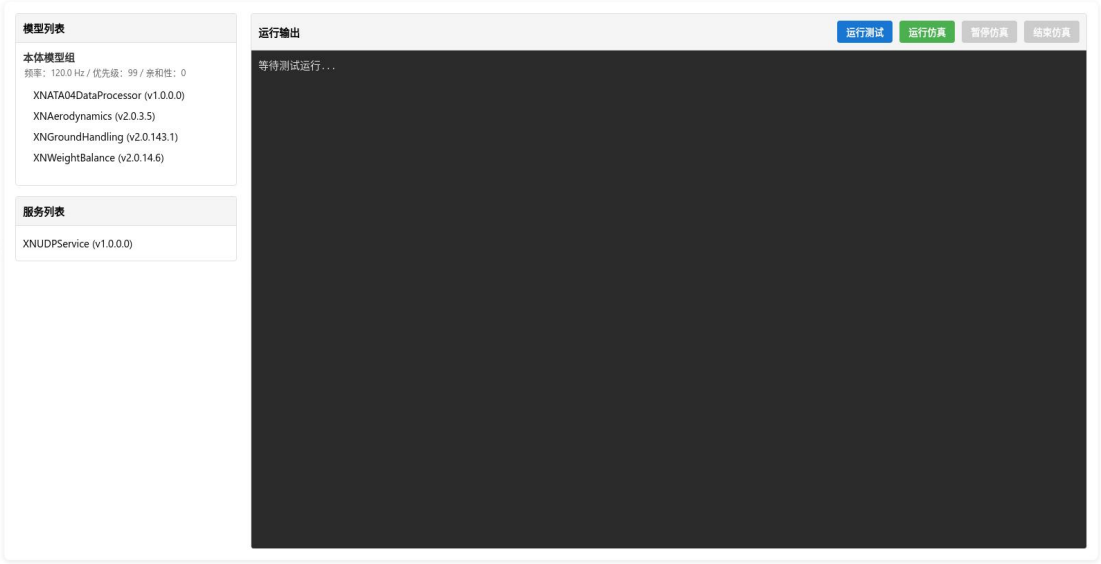

图 46 仿真运行界面布局

4.4.27.3. 交互元素

仿真运行界面的关键控件列表如表 32 所示。

表 32 仿真运行界面交互元素

控件类型	ID	状态	行为规则
模型列表	list_model	无	显示本次仿真加载的模型列表
服务列表	list_server	无	显示本次仿真加载的服务列表
运行测试按钮	btn_runtest	无	点击后以测试模式启动仿真引擎
运行仿真按钮	btn_runsim	无	点击后启动仿真引擎
暂停/继续仿真按钮	btn_pcsim	激活/未激活	激活时暂停仿真引擎，未激活时取消仿真引擎的暂停
结束仿真按钮	btn_stopsim	无	点击后终止仿真引擎
运行输出页面	page_out	无	显示仿真引擎在控制台的输出

4.4.28.XNSimLog

4.4.28.1. 概述

1. 接口名称：运行日志界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.28.2. 界面布局

运行日志界面的布局如图 47 所示。

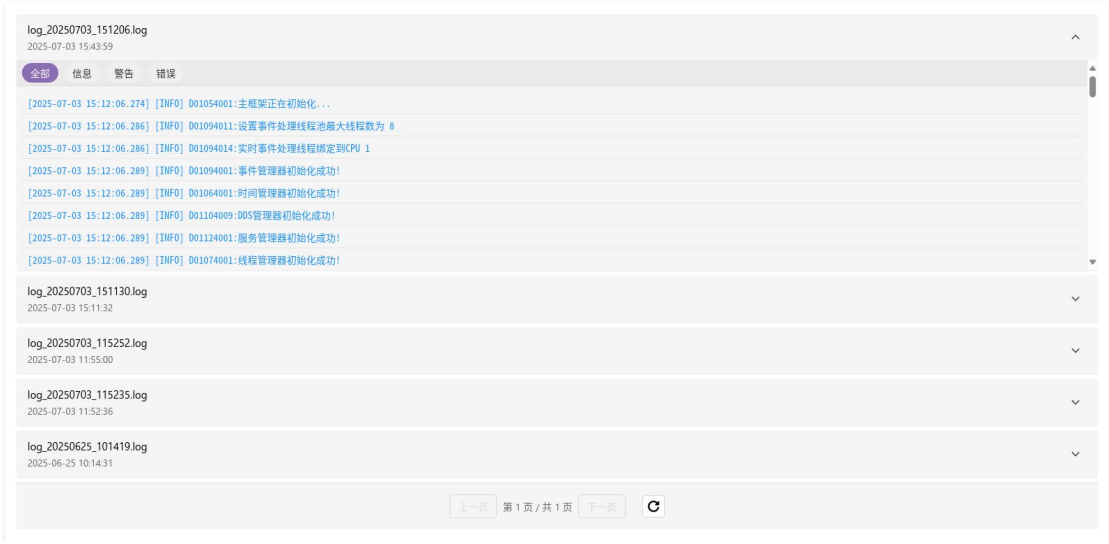

图 47 运行日志界面布局

4.4.28.3. 交互元素

运行日志界面的关键控件列表如表 33 所示。

表 33 运行日志界面交互元素

控件类型	ID	状态	行为规则
日志列表	list_log	无	列出所有仿真运行日志
日志详细内容按钮	btn_detail	激活/未激活	激活后显示日志详细内容
显示全部按钮	btn_all	激活/未激活	激活后显示日志所有内容
只显示信息按钮	btn_info	激活/未激活	激活后只显示日志中的信息类记录
只显示警告按钮	btn_warning	激活/未激活	激活后只显示日志中的警告类记录
只显示错误按钮	btn_error	激活/未激活	激活后只显示日志中的错误类记录
切换分页按钮	btn_page	无	点击可以切换日志的分页显示

4.4.29.XNResourceMonitor

4.4.29.1. 概述

1. 接口名称：资源监控界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.29.2. 界面布局

资源监控界面的布局如图 48 所示。

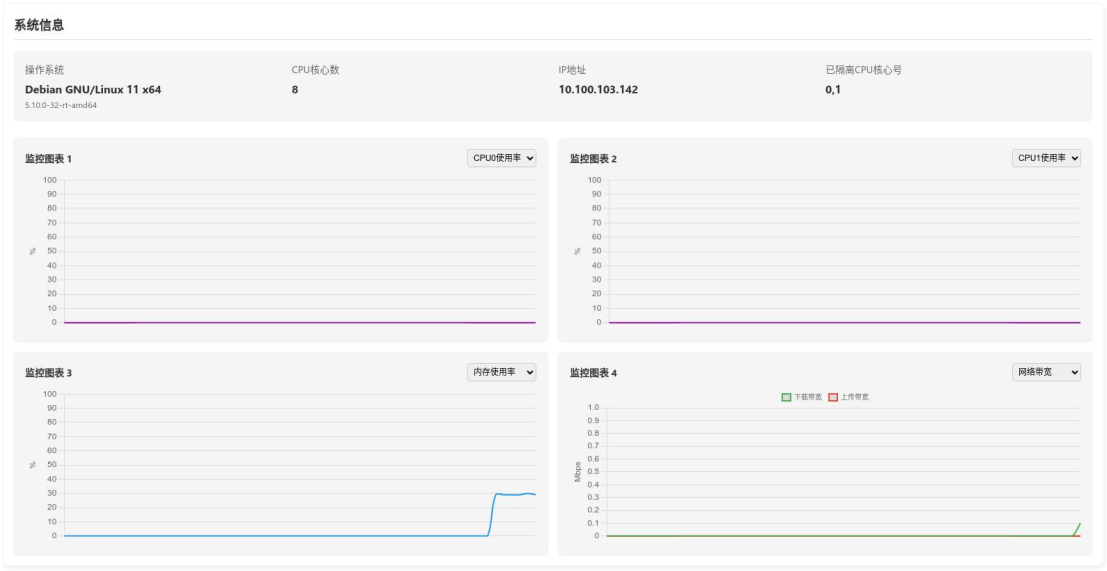

图 48 资源监控界面布局

4.4.29.3. 交互元素

资源监控界面的关键控件列表如表 34 所示。

表 34 资源监控界面交互元素

控件类型	ID	状态	行为规则
系统信息显示	label_systeminfo	无	显示系统名称、CPU 核心数、IP、隔离 CPU 核心编号等信息
监控图表	chart_monitor	无	以四个图表显示需要监控的资源信息
图表切换按钮	btn_change	无	点击可以切换图表显示的内容

4.4.30.XNSystemMonitor

4.4.30.1. 概述

1. 接口名称：仿真监控界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.30.2. 界面布局

仿真监控界面的布局如图 49 所示。

图 49 仿真监控界面布局

4.4.30.3. 交互元素

仿真监控界面的关键控件列表如表 35 所示。

表 35 仿真监控界面交互元素

控件类型	ID	状态	行为规则
监控状态指示	label_status	未监控/监控中/错误	未监控-DDS 监控后端服务未启用 监控中-DDS 监控后端服务已启用 错误-DDS 监控后端服务监控出错
引擎名称显示	label_enginename	无	显示引擎进程名
引擎 ID 显示	label_engineid	无	显示引擎进程 ID
引擎状态显示	label_enginestatus	未运行/运行中/暂停/错误	未运行-引擎未在运行 运行中-引擎正在运行中 暂停-引擎暂停运行 错误-无法获取引擎运行状态
引擎亲和性显示	label_engineaff	无	显示引擎的 CPU 亲和性
仿真线程数显示	label_threadcount	无	显示调度线程个数
主框架状态显示	label_framwork	未加载/正常/异常	未加载-该内核模块未加载 正常-该内核模块正常加载 异常-该内核模块加载异常
时间管理器状态显示	label_timemng	未加载/正常/异常	未加载-该内核模块未加载 正常-该内核模块正常加载 异常-该内核模块加载异常
事件管理器状态显示	label_eventmng	未加载/正常/异常	未加载-该内核模块未加载 正常-该内核模块正常加载 异常-该内核模块加载异常

构型管理器状态显示	label_confmng	未加载/正常/异常	未加载-该内核模块未加载 正常-该内核模块正常加载 异常-该内核模块加载异常
模型管理器状态显示	label_modelmng	未加载/正常/异常	未加载-该内核模块未加载 正常-该内核模块正常加载 异常-该内核模块加载异常
服务管理器状态显示	label_servermng	未加载/正常/异常	未加载-该内核模块未加载 正常-该内核模块正常加载 异常-该内核模块加载异常
DDS 管理器状态显示	label_ddsmng	未加载/正常/异常	未加载-该内核模块未加载 正常-该内核模块正常加载 异常-该内核模块加载异常
线程信息表	table_thread	无	以表格形式显示线程的名称、ID、状态、优先级、运行次数、设定频率、平均频率、最大频率、最小频率、设定周期、平均周期、最大周期、最小周期
线程监控图表	chart_thread	频率/周期	绘制线程的实时频率/实时周期
图表显示切换按钮	btn_chart	激活/未激活	激活时线程监控图表绘制频率，未激活时绘制周期

4.4.31.XNModelMonitor

4.4.31.1. 概述

1. 接口名称：模型监控界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.31.2. 界面布局

模型监控界面的布局如图 50 所示。

图 50 模型监控界面布局

4.4.31.3. 交互元素

模型监控界面的关键控件列表如表 36 所示。

表 36 模型监控界面交互元素

控件类型	ID	状态	行为规则
监控状态指示	label_status	未监控/监控中/错误	未监控-DDS 监控后端服务未启用 监控中-DDS 监控后端服务已启用 错误-DDS 监控后端服务监控出错
模型信息表	table_model	无	以表格形式显示模型的名称、ID、状态、所属线程 ID、优先级、运行次数、设定频率、平均频率、最大频率、最小频率、设定周期、平均周期、最大周期、最小周期
模型监控图表	chart_model	频率/周期	绘制模型的实时频率/实时周期
图表显示切换按钮	btn_chart	激活/未激活	激活时模型监控图表绘制频率，未激活时绘制周期

4.4.32.XNDataMonitor

4.4.32.1. 概述

1. 接口名称：数据监控界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.32.2. 界面布局

数据监控界面的布局如图 51 所示。

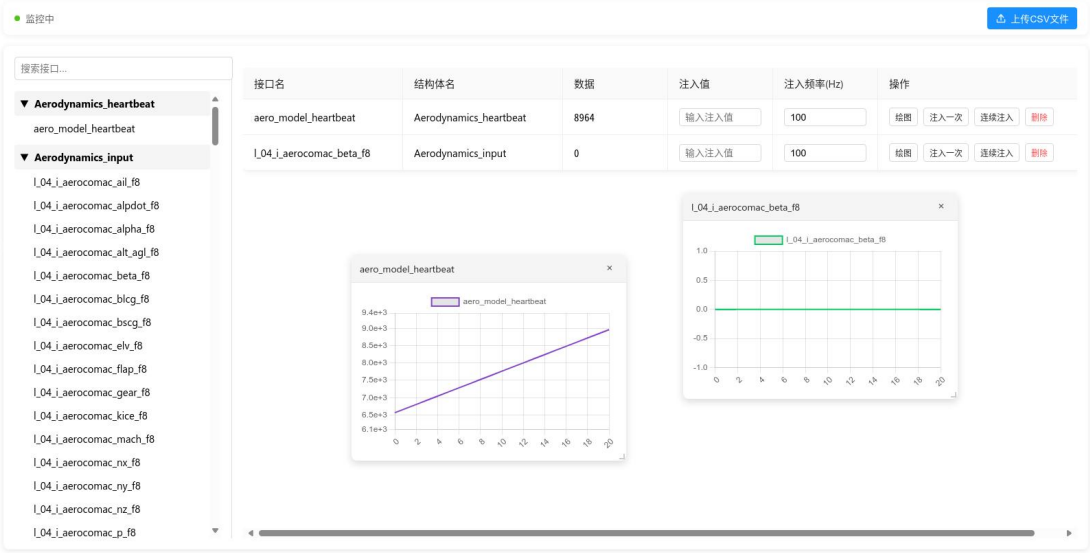

图 51 数据监控界面布局

4.4.32.3. 交互元素

数据监控界面的关键控件列表如表 37 所示。

表 37 数据监控界面交互元素

控件类型	ID	状态	行为规则
监控状态指示	label_status	未监控/监控中/错误	未监控-DDS 监控后端服务未启用 监控中-DDS 监控后端服务已启用 错误-DDS 监控后端服务监控出错
接口列表	list_inter	无	列出所有可以监控数据的接口，双击接口可以将它添加到监控表格中
接口列表搜索框	edit_search	无	可以根据输入的内容搜索接口列表
接口监控表	table_monitor	无	以表格的形式显示监控的接口的名称、结构体名、数据
注入值输入框	edit_injectdata	无	可以输入需要注入的数据
注入频率输入框	edit_injectfreq	无	可以输入注入数据的频率
绘图按钮	btn_chart	激活/未激活	激活时弹出对应接口的实时数据绘图，未激活时关闭绘图
注入一次按钮	btn_injectonce	无	点击后向模型注入一次数据
连续注入按钮	btn_injectctn	激活/未激活	激活时按指定的注入频率向模型注入数据，未激活时停止注入
删除按钮	btn_delete	无	点击后停止对该接口的监控
上传 CSV 按钮	btn_upload	无	点击后可以上传 CSV 文件进行数据注入
CSV 注入按钮	btn_csvinject	激活/未激活	激活时读取 CSV 文件并按照设定的时间戳向模型注入数据，未激活时停止注入

4.4.33.XNDataCollect

4.4.33.1. 概述

1. 接口名称：数据采集界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.33.2. 界面布局

数据采集界面的布局如图 52 所示。

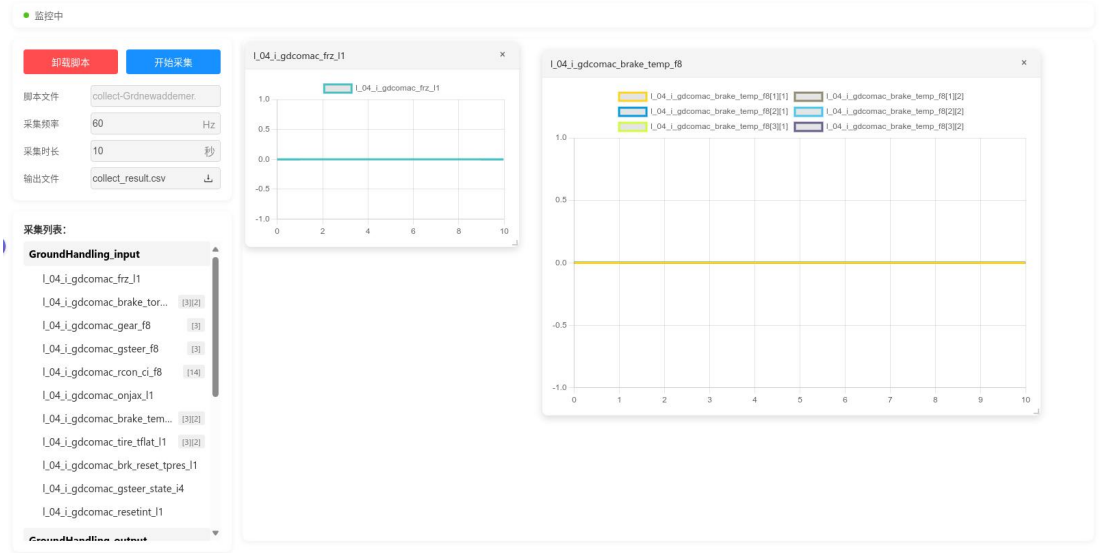

图 52 数据采集界面布局

4.4.33.3. 交互元素

数据采集界面的关键控件列表如表 38 所示。

表 38 数据采集界面交互元素

控件类型	ID	状态	行为规则
监控状态指示	label_status	未监控/监控中/错误	未监控-DDS 监控后端服务未启用 监控中-DDS 监控后端服务已启用 错误-DDS 监控后端服务监控出错
脚本加载/卸载按钮	btn_dcs	激活/未激活	激活时上传 DCS 采集脚本，解析并填充脚本文件输入框、采集频率输入框、采集时长输入框、输出文件输入框和采集列表；未激活时卸载脚本，并清空以上控件的内容
开始/停止采集按钮	btn_collect	激活/未激活	激活时开始进行数据采集；未激活时停止数据采集

脚本文件输入框	edit_dcsfile	无	无法编辑，上传脚本自动填充
采集频率输入框	edit_cllfreq	无	无法编辑，上传脚本自动填充
采集时长输入框	edit_clltime	无	无法编辑，上传脚本自动填充
输出文件输入框	edit_outfile	无	无法编辑，上传脚本自动填充
输出文件下载按钮	btn_download	无	点击后下载采集完成的 CSV 文件
采集列表	list_collect	无	列出 DCS 脚本中需要采集的接口列表，采集完成后双击接口名可以打开采集数据的绘图
采集数据绘图	chart_collect	无	绘制对应接口采集到的数据

4.4.34.XNNetMonitor

4.4.34.1. 概述

1. 接口名称：网络监控界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.34.2. 界面布局

网络监控界面的布局如图 53 所示。

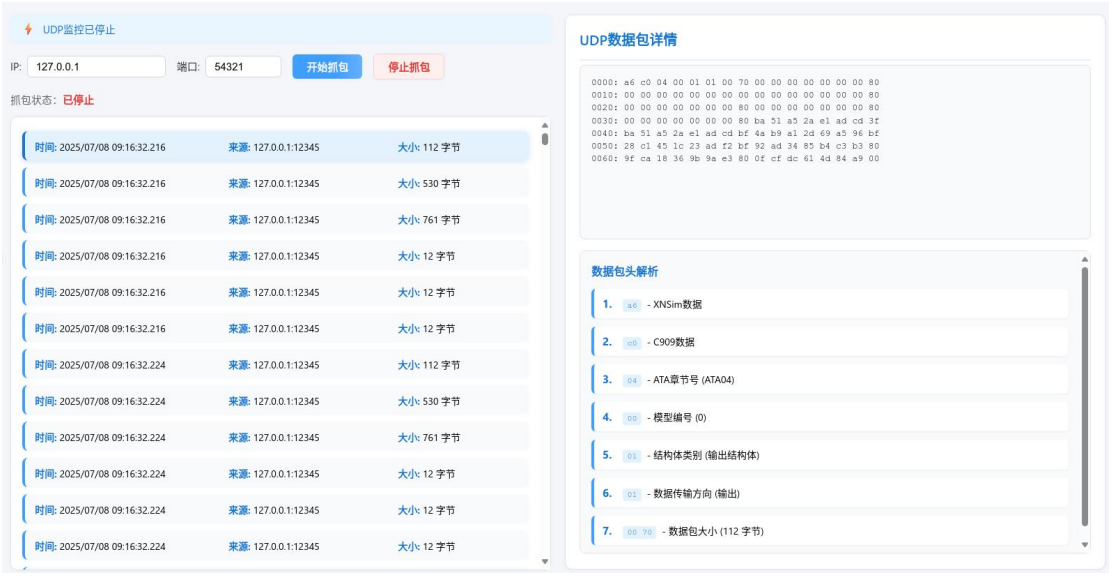

图 53 网络监控界面布局

4.4.34.3. 交互元素

网络监控界面的关键控件列表如表 39 所示。

表 39 网络监控界面交互元素

控件类型	ID	状态	行为规则
IP 输入框	edit_ip	无	可以输入需要监控的 IP 地址
端口输入框	edit_port	无	可以输入需要监控的端口号
开始抓包按钮	btn_start	无	点击后开始监控对应 IP 和端口号的数据包
停止抓包按钮	btn_stop	无	点击后停止监控对应 IP 和端口号的数据包
抓包状态显示	label_status	已开始/已停止	对应显示当前的监控状态
数据包列表	list_pkg	无	列出监控到的数据包的时间戳、来源 IP 和端口、包大小
数据包内容显示	label_pkg	无	显示数据包的十六进制内容
数据包头解析结果	label_pkgheader	无	解析数据包的头部并显示解析结果：包括 XNSim 数据头、机型头、ATA 章节头、模型编号头、结构体类型头、数据传输方向头、数据大小头

4.4.35.XNQTG

4.4.35.1. 概述

1. 接口名称：QTG 界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.35.2. 界面布局

QTG 界面的布局暂未设计。

4.4.35.3. 交互元素

QTG 界面的交互元素暂未设计。

4.4.36.XNProfileCenter

4.4.36.1. 概述

1. 接口名称：个人中心界面交互接口
2. 接口类型：人机交互接口
3. 适用终端：Web 浏览器。

4.4.36.2. 界面布局

个人中心界面的布局如图 54 所示。

图 54 个人中心界面布局

4.4.36.3. 交互元素

个人中心界面的关键控件列表如表 40 所示。

表 40 个人中心界面交互元素

控件类型	ID	状态	行为规则
用户头像显示	icon_user	无	显示用户的头像图标
编辑头像按钮	btn_editicon	无	编辑用户头像图标
用户基本信息显示	label_info	无	显示用户的用户名、姓名、权限级别、部门、职位、ID
用户联系方式显示	label_contact	无	显示用户的邮箱和电话
编辑信息按钮	btn_editinfo	无	点击可以修改用户的姓名、部门、职位、邮箱和电话
修改密码按钮	btn_editpwd	无	点击可以修改用户密码

4.4.37.XNUserManager

4.4.37.1. 概述

1. 接口名称：用户管理界面交互接口

2. 接口类型：人机交互接口

3. 适用终端：Web 浏览器。

4.4.37.2. 界面布局

用户管理界面的布局如图 55 所示。

图 55 用户管理界面布局

4.4.37.3. 交互元素

用户管理界面的关键控件列表如表 41 所示。

表 41 用户管理界面交互元素

控件类型	ID	状态	行为规则
用户信息表	table_users	无	列表显示用户的 ID、用户名、姓名、权限等级
修改信息按钮	btn_editinfo	无	点击后修改对应用户的信息
调整权限按钮	btn_editaccess	无	点击后修改对应用户的权限等级
重置密码按钮	btn_resetpwd	无	点击后重置对应用户的密码
删除按钮	btn_delete	无	点击后删除对应用户

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 56 所示。

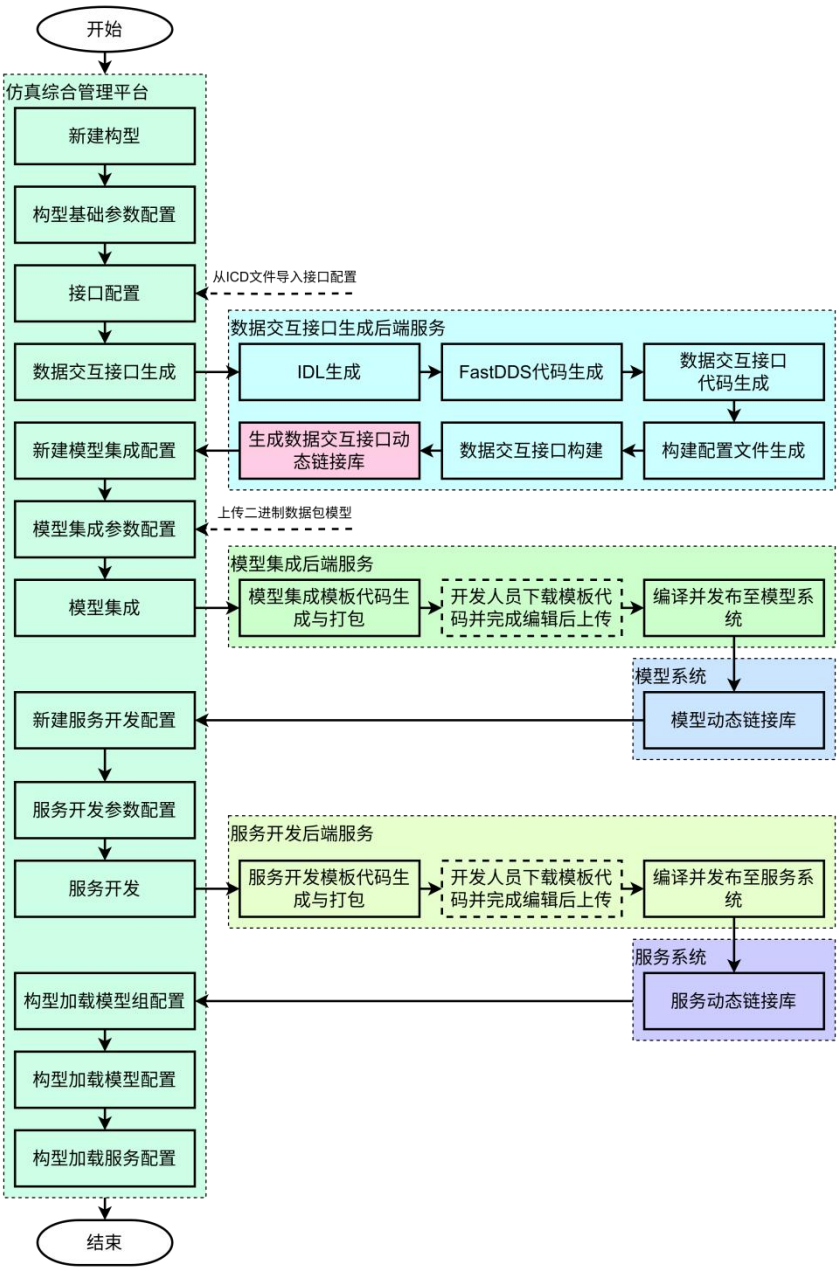

图 56 “玄鸟”架构仿真准备阶段处理流程

详细处理流程如下：

1. 启动仿真综合管理平台，使用有构型配置权限的用户登录后，在构型配置模块新建一个仿真构型；

2. 新建构型配置后，配置构型基础参数，如操作系统名称、版本、实时补丁版本、DDS通信域与日志记录等级等；

3. 在接口配置模块配置该构型的数据交互接口定义，支持从 ICD 文件直接导入接口配置，也支持手动配置各个接口。接口配置将实时存入数据库中；

4. 使用接口配置模块提供的数据交互接口生成功能进行数据交互接口的自动化构建，在这一过程中仿真综合管理平台将调用数据交互接口生成后端服务进行以下步骤：

1） 读取数据库中已经保存好的接口配置，生成 IDL 文件；

2） 调用 FastDDS-Gen 工具，解析 IDL 文件并生成 FastDDS 交互所需的代码；

3） 依据仿真内核提供的数据交互抽象接口，针对 FastDDS 各数据主题实现数据交互接口代码；

4） 生成数据交互接口的构建配置文件；

5） 执行自动化地构建流程；

6） 生成数据交互接口动态链接库。

5. 使用模型集成模块新建需要参与仿真的模型的集成配置；

6. 对模型集成配置参数进行编辑，需要填写以下内容：

1） 基础参数：版本号、作者、描述、运行频率分组、运行节点等；

2） 二进制数据包模型参数：可以上传二进制数据包模型并由模型集成模块自动分析并填写这部分参数，也可以手动填写。

7. 使用模型集成模块提供的模板代码生成功能可以生成模型集成模板代码；

8. 开发人员下载模板代码并进行所需的修改后再上传完成的集成代码；

9. 模型集成模块将自动化构建模型集成代码，生成模型动态链接库并发布至模型系统中；

10. 使用服务开发模块新建需要参与仿真的服务配置；

11. 对服务配置参数进行编译；

12. 使用服务开发模块提供的模板代码生成功能可以生成服务开发模板代码；

13. 开发人员下载模板代码并进行所需的开发后再上传完成的服务代码；

14. 服务开发模块将自动化构建服务开发代码，生成服务动态链接库并发布至服务系统

中；

15. 在构型配置模块配置仿真要加载的模型组（即仿真调度线程）及其参数，包括：频率、优先级、CPU 亲和性等；

16. 在构型配置模块配置各模型组中需要加载的模型；

17. 在构型配置模块配置仿真需要加载的服务；

18. 仿真准备完成。

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 57 所示。

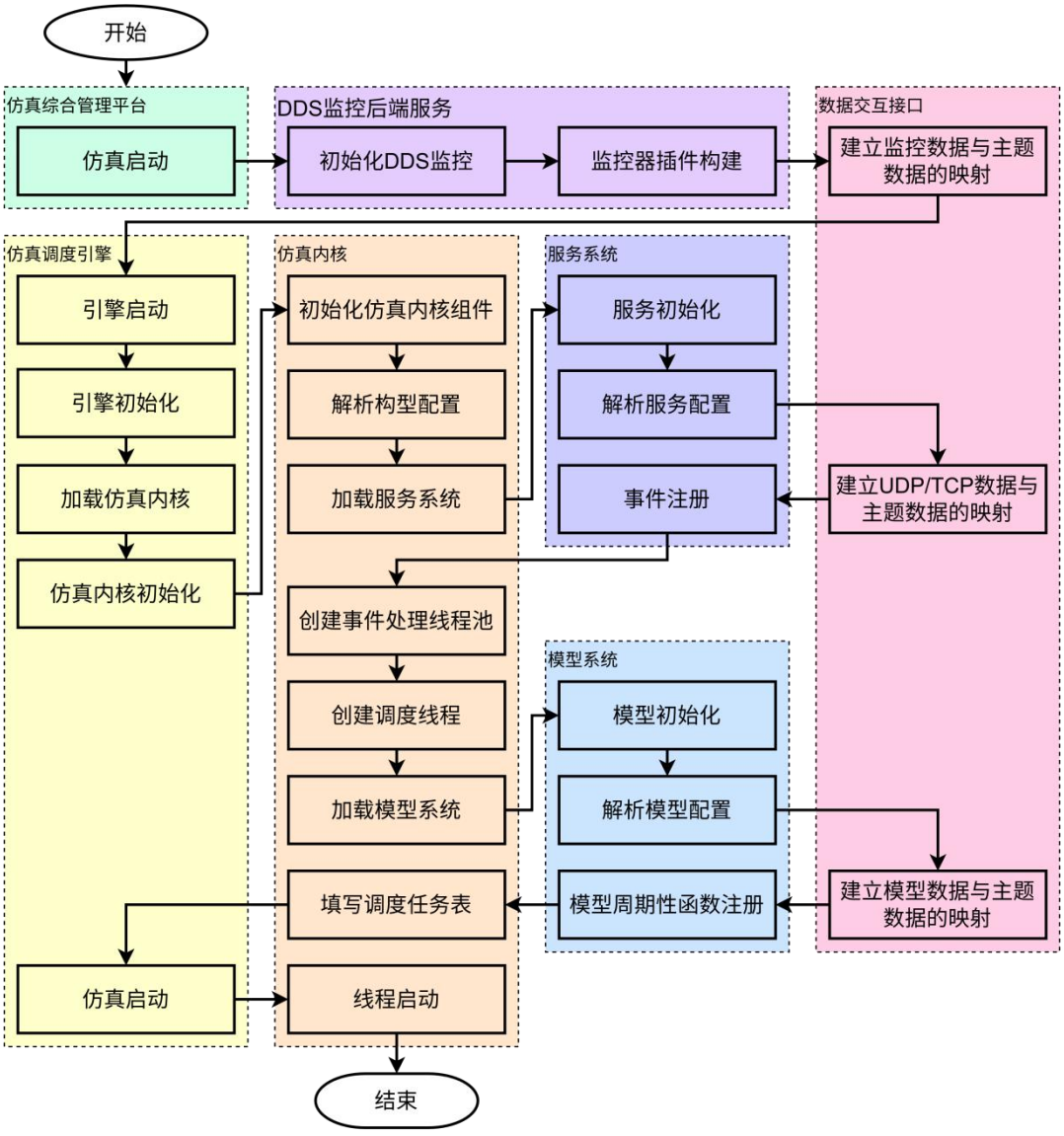

图 57 “玄鸟”架构仿真启动阶段处理流程

详细处理流程如下：

1. 在综合仿真管理平台的仿真运行控制模块启动仿真；

2. 综合管理平台会首先调用 DDS 监控后端服务，初始化 DDS 监控；

3. DDS 监控后端服务会动态构建数据监控器插件，用于监控模型数据；

4. 数据监控器插件将调用数据交互接口，建立监控数据与主题数据的映射接口；

5. 启动仿真调度引擎，进行工作参数设置等初始化操作，包括但不限于：

1）创建日志记录文件；

2）设置 CPU 亲和性：用于分配仿真运行的计算资源；

3）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；

6. 仿真调度引擎进行仿真内核的加载；

7. 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；

8. 仿真内核进行构型配置解析，逐项读取构型配置中的仿真运行参数，并根据这些参数对仿真环境进行相应的设置；

9. 仿真内核根据构型配置中的加载服务列表，动态地加载所有配置的服务动态链接库；

10. 服务动态链接库成功加载并完成初始化；

11. 服务初始化完成后，服务读取并解析自身的服务配置。服务配置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；

12. 服务根据自身需要，通过数据交互接口建立 UDP/TCP 数据与主题数据的映射接口；

13. 服务根据自身需要，向仿真内核的事件管理器注册事件；

14. 仿真内核收到注册的事件后，建立事件处理线程池；

15. 仿真内核根据构型配置中的加载模型组列表创建调度线程；

16. 仿真内核根据模型组中的加载模型列表，动态加载所有模型动态链接库；

17. 模型动态链接库成功加载并完成初始化；

18. 模型初始化完成后，读取并解析自身的模型配置。模型配置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；

19. 模型根据自身交互所需的数据接口，通过数据交互接口建立模型数据与主题数据的映射接口；

20. 模型将周期性执行函数向线程管理器进行注册;

21. 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；

22. 调度表填写完成后，仿真已经可以开始运行，仿真引擎将发出启动指令；

23. 仿真内核的线程管理器启动所有仿真调度线程；

24. 仿真开始运行。

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 58 所示。

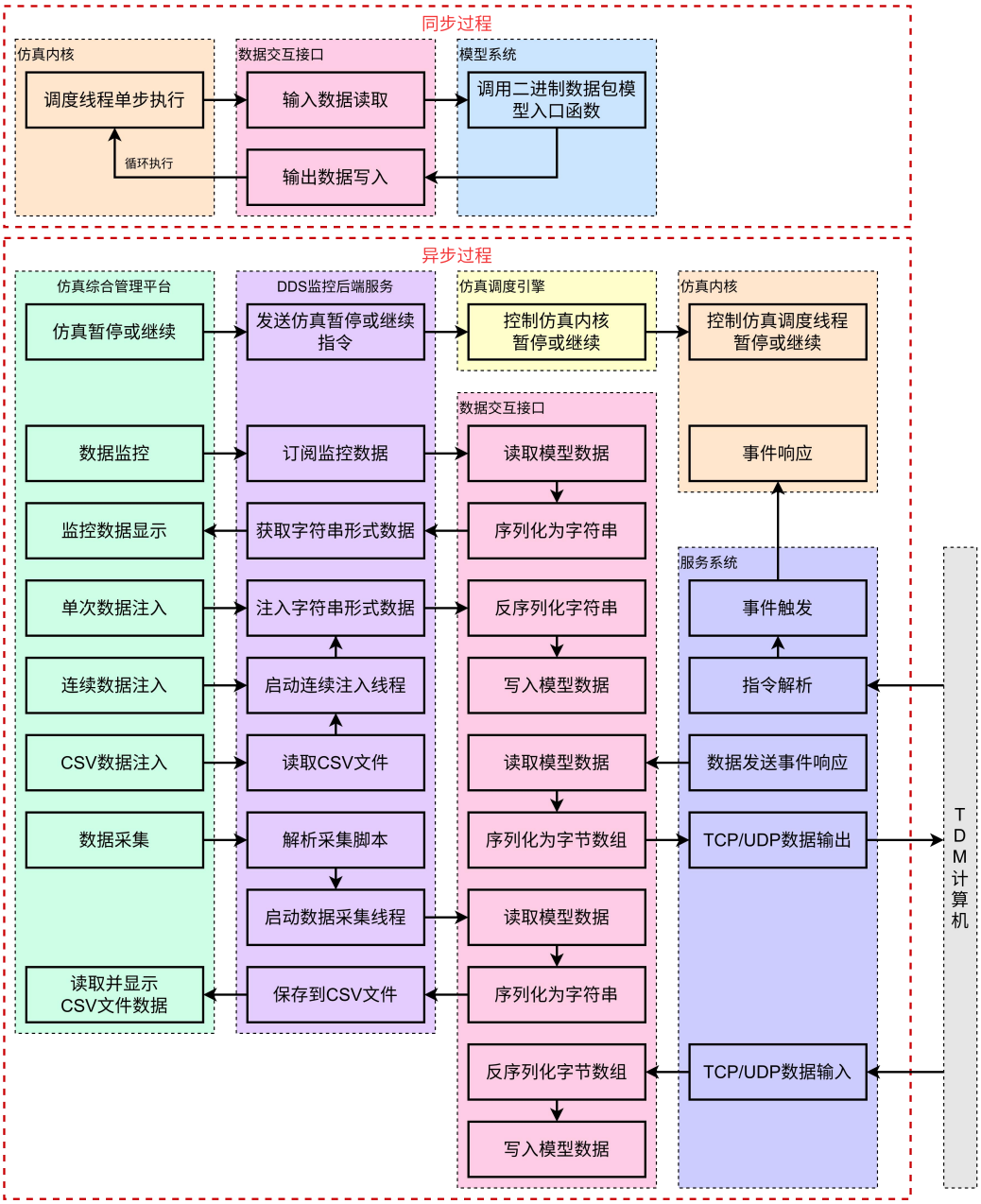

图 58 “玄鸟”架构仿真运行阶段处理流程

仿真运行阶段的同步处理流程如下：

1. 仿真内核各调度线程按照调度任务表依次单步执行，调度模型的周期性执行函数；

2. 模型的周期性执行函数从数据交互接口进行模型输入数据的读取；

3. 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；

4. 模型的周期性执行函数将输出数据写入到数据交互接口中；

5. 循环执行 1～4 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. 仿真运行控制异步流程：

1）用户可以通过仿真综合管理平台控制仿真运行。当用户发送仿真暂停指令时，仿真综合管理平台通过 DDS 监控后端服务发送仿真运行控制指令主题；

2）仿真调度引擎接收到仿真运行控制指令主题，暂停仿真调度引擎运行，同时控制仿真内核暂停运行；

3）仿真内核暂停所有调度线程的运行；

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

2. 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；

2） 控制指令解析服务触发指令对应的事件；

3） 仿真内核的事件管理器收到事件触发信号后会执行对应的事件响应。

3. 仿真数据输出异步流程：

1） 当有仿真数据输出事件产生时，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）将响应该事件；

2） 通信服务从数据交互接口获取需要向外部系统输出的数据；

3） 数据交互接口将数据序列化为字节数组形式发送给通信服务；

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）对输出数据进行打包；

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打包；

6） 通信服务使用 UDP 或 TCP 向外部系统发送输出数据。

4. 仿真数据输入异步流程：

1）当外部系统有数据输入时，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送到数据交互接口中；

5）数据交互接口将对该字节数组进行反序列化；

6）数据交互接口将反序列化完成的数据写入共享内存。

5. 仿真数据监控异步流程：

1） 仿真综合管理平台向 DDS 监控后端服务发送需要监控的数据；

2） DDS 监控后端服务将向数据交互接口订阅对应的监控数据主题；

3） 数据交互接口从共享内存中读取需要监控的数据；

4） 数据交互接口将需要监控的数据序列化为字符串形式；

5） DDS 监控后端服务读取序列化的监控数据；

6） 仿真综合管理平台从 DDS 监控后端服务读取监控数据并显示。

6. 仿真数据单步注入异步流程：

1）仿真综合管理平台向 DDS 监控后端服务发送需要注入的数据；

2）DDS 监控后端服务向数据交互接口注入一次字符串形式的数据；

3）数据交互接口反序列化该字符串数据；

4）数据交互接口向共享内存写入该数据；

7. 仿真数据连续注入异步流程：

1）仿真综合管理平台向 DDS 监控后端服务发送需要连续注入的数据；

2）DDS 监控后端服务启动连续注入线程；

3）连续注入线程向数据交互接口连续注入字符串形式的数据；

4）数据交互接口反序列化该字符串数据；

5）数据交互接口向共享内存写入该数据；

8. 仿真数据 CSV 注入异步流程：

1）仿真综合管理平台向 DDS 监控后端服务发送需要进行 CSV 注入的.csv 文件；

2）DDS 监控后端服务读取.csv 文件获取需要注入的数据；

3）DDS 监控后端服务启动连续注入线程；

4）连续注入线程向数据交互接口连续注入字符串形式的数据；

5）数据交互接口反序列化该字符串数据；

6）数据交互接口向共享内存写入该数据；

9. 仿真数据采集异步流程：

1）仿真综合管理平台向 DDS 监控后端服务发送需要采集数据的脚本；

2）DDS 监控后端服务解析脚本获取需要采集的数据；

3）DDS 监控后端服务启动数据采集线程；

4）数据交互接口获取共享内存中的模型数据；

5）数据交互接口将模型数据序列化为字符串数据；

6）数据采集线程获取该字符串数据并写入 CSV 文件中；

7）采集完成后，仿真综合管理平台读取并显示采集结果。

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 59 所示。

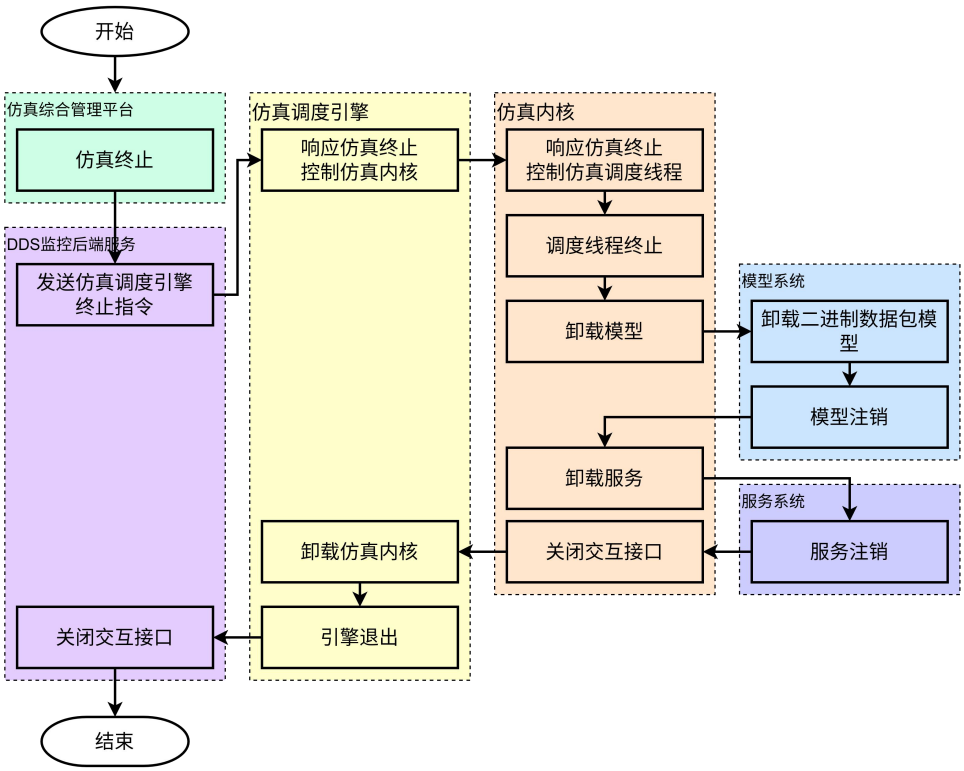

图 59 “玄鸟”架构仿真终止阶段处理流程

详细处理流程如下：

1. 当用户通过仿真综合管理平台发送仿真终止指令时，DDS 监控后端服务发布仿真运行控制指令主题；

2. 仿真调度引擎接收到仿真运行控制指令主题，终止仿真调度引擎运行，同时控制仿真内核终止运行；

3. 仿真内核终止所有调度线程的运行；

4. 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；

5. 模型卸载二进制数据包模型动态链接库；

6. 模型向仿真内核的模型管理器注销自身，仿真内核卸载模型动态链接库；

7. 仿真内核完成所有模型的卸载后，停止所有服务的运行；

8. 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；

9. 服务向服务管理器注销自身；

10. 仿真内核完成所有服务的卸载后，关闭所有数据交互接口，停止仿真内核各组件的运行；

11. 仿真调度引擎卸载仿真内核；

12. 仿真调度引擎退出；

13. DDS 监控后端服务关闭数据交互接口；

14. 仿真终止。

6. 出错处理设计

“玄鸟”架构的出错信息及故障处理表见附录 A。

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） 将配置参数放在数据库中，如构型配置、模型配置、服务配置等，便于修改和管理；

2） 使用仿真综合管理平台进行可视化地配置。

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：仿真综合管理平台能够实时监控“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

“玄鸟”架构目前还有以下问题尚待解决：

1. 二进制数据包模型的 QTG 功能尚未设计实现；

2. 尚未设计虚拟航空总线通信服务与数据交互接口之间如何进行对接。

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 42 所示。

表 42 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.	模型集成与集成能力	仿真内核 模型系统 模型集成后端服务	模型集成人员根据仿真内核提供的标准化模型集成框架将离散的二进制数据包模型集成到模型系统中
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集成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	模型系统 数据交互接口	模型系统中的模型通过数据交互接口进行数据交互
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	仿真综合管理平台 DDS 监控后端服务 数据交互接口	仿真综合管理平台通过 DDS 监控后端服务从数据交互接口中读取或写入数据以实现数据监控和数据注入
6.	仿真可配置性	仿真综合管理平台 数据库 仿真内核	仿真综合管理平台能够管理与编辑各项配置并存储在数据库中，仿真内核根据数据库中的内容配置仿真的运行

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 数据交互接口 数据库	快照服务能够通过数据交互接口对共享内存中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	仿真综合管理平台	仿真综合管理平台提供了仿真配置、仿真运行、仿真监控等多种可视化交互功能
10.	可扩展性	仿真综合管理平台 服务开发后端服务 服务系统	仿真综合管理平台通过调用服务开发后端服务，提供了服务的自动化代码生成与构建，使服务系统能够扩展“玄鸟”架构具备的能力
11.	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 %
12.	响应实时性	仿真内核 服务系统	服务系统及仿真内核提供的事件管理器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
13.	监控实时性	仿真综合管理平台 DDS 监控后端服务 数据交互接口	仿真综合管理平台通过 DDS 监控后端服务实时从数据交互接口获取模型数据并进行显示，实现数据更新频率不大于 1 Hz
14.	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
15.	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 43 所示。

表 43 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.	仿真综合管理平台	仿真可配置性 仿真监控能力 可视化交互界面 监控实时性	仿真综合管理平台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；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
2.	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
3.	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

4.	仿真内核	模型集成与集成能力 模型实时调度能力 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集成框架用以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；仿真内核通过解析数据库中的配置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
5.	模型系统	模型实时调度能力	模型注册需要被调度的周期性执行函数以供实时调度线程调度
6.	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 可扩展性	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
7.	数据交互接口	模型间数据交互能力 仿真监控能力 外部系统数据交互能力	数据交互接口提供了基于共享内存的 FastDDS 发布/订阅数据交互功能；实现了模型间数据交互接口、TCP/UDP 数据交互接口、数据监控接口；
8.	数据库	仿真可配置性 快照拍摄和调用能力	仿真配置信息和快照信息存储在数据库中
9.	DDS 监控后端服务	仿真监控能力	DDS 监控后端服务为仿真综合管理平台提供仿真数据监控和注入的接口
10.	模型集成后端服务	模型集成与集成能力	使用模型集成框架集成二进制数据包模型于模型系统中供仿真内核调用
11.	服务开发后端服务	可扩展性	使用服务开发框架开发服务并集成于服务系统中以扩展仿真系统能力

10. 总结

依据本软件概要设计说明书，开发团队可以充分利用自研的“玄鸟”架构，将二进制数据包模型进行高效集成，最终构建出一套完整且一体化的二进制数据包软件。本软件概要设计说明书不仅为一体化二进制数据包软件的开发与维护提供了详尽且全面的指导，还涵盖了软件开发的各个环节。通过详细阐述软件的整体架构设计、各个功能模块的具体划分、数据处理流程的优化方案，以及与其他系统之间的接口对接方式等关键信息。该文档确保开发团队能够对软件的整体结构和各项功能有一个清晰且深入的理解。这样一来，开发团队成员之间能够高效协作，顺利推进软件的开发工作，并确保软件在运行过程中保持稳定性和可靠性。同时，该文档也为后续的运行维护和功能扩展提供了明确的参考依据，确保软件能够灵活应对不断变化的市场需求和用户反馈，持续优化和提升软件的性能与用户体验。

附录 A “玄鸟”架构出错信息及故障处理表

序号	错误码	出错信息	故障原因及处理
1	A01021001	不支持的类型转换	日志记录语句的参数不支持转换为 std::string
2	A01021002	单条日志参数数量超过限制	日志记录语句的参数最多为 9 个
3	D01054001	主框架正在初始化...	重要消息
4	B01052002	主框架初始化失败	事件管理器初始化失败导致
5	B01052003	主框架初始化失败	时间管理器初始化失败导致
6	B01052004	主框架初始化失败	DDS 管理器初始化失败导致
7	B01052005	主框架初始化失败	服务管理器初始化失败导致
8	B01052006	主框架初始化失败	线程管理器初始化失败导致
9	B01052007	主框架初始化失败	模型管理器初始化失败导致
10	B01052008	主框架初始化失败	构型管理器初始化失败导致
11	D01054009	主框架初始化成功	重要消息
12	D01054010	开始解析构型文件...	重要消息
13	D01054011	解析构型文件成功	重要消息
14	B01052012	主框架解析构型失败	构型管理器初解析构型参数失败导致
15	B01052013	主框架准备执行失败	事件管理器准备执行失败导致
16	B01052014	主框架准备执行失败	时间管理器准备执行失败导致
17	B01052015	主框架准备执行失败	DDS 管理器准备执行失败导致
18	B01052016	主框架准备执行失败	服务管理器准备执行失败导致
19	B01052017	主框架准备执行失败	线程管理器准备执行失败导致
20	B01052018	主框架准备执行失败	模型管理器准备执行失败导致
21	B01052019	主框架准备执行失败	构型管理器准备执行失败导致
22	D01054020	主框架准备就绪	重要消息
23	D01064001	时间管理器初始化成功	重要消息
24	D01064002	时间管理器准备就绪	重要消息
25	D01064003	仿真开始运行	重要消息
26	D01064004	仿真终止运行	重要消息
27	D01064005	仿真暂停运行	重要消息
28	D01064006	仿真继续运行	重要消息
29	A01063007	仿真已经运行,请勿重复运行	请勿重复发送仿真开始指令

30	A01063008	仿真已经终止运行,请勿重复终止	请勿重复发送仿真终止指令
31	A01063009	仿真不是运行状态,无法暂停	请勿在仿真未运行时发送仿真暂停指令
32	A01063010	仿真不是暂停状态,无法继续	请勿在仿真非暂停时发送仿真继续指令
33	D01074001	线程管理器初始化成功	重要消息
34	D01074002	线程管理器正在初始化所有线程...	重要消息
35	D01074003	线程管理器准备就绪	重要消息
36	D01074004	添加线程成功,运行频率: f Hz, 运行间隔: T ns.	重要消息，同时输出线程的运行频率 f 和运行间隔 T
37	D01074005	模型 [ID] 注册周期性函数成功! 运行节点: G-N 优先级: P	重要消息，输出模型 ID 及其注册的周期性函数的运行节点 G-N 与优先级 P
38	A01073006	模型 [ID] 的运行频率分组不在 0~5 之间,注册失败!	某 ID 的模型的运行频率分组必须是 [0,5] 之间的数
39	A01073007	模型 [ID] 的运行节点 N 超出当前频率分组最大节点数 Nmax,注册失败!	某 ID 的模型的运行节点号 N 必须小于等于 Nmax=(2 的频率分组值的次方）
40	B01072008	线程 [ID] 不存在,模型 [ID] 注册周期性函数失败!	内部错误，某 ID 的模型向不存在的线程 ID 注册周期性函数
41	B01072009	线程 [ID] 初始化失败!	某 ID 的线程初始化失败导致
42	D01084001	构型管理器初始化成功	重要消息
43	D01084002	构型管理器准备就绪	重要消息
44	C01082003	打开构型配置文件 Path 出错, 错误信息: Msg	打开位于 Path 的构型配置文件时出错，查看具体的错误消息 Msg
45	C01082004	构型配置文件 Path 的 XML 解析失败!	无法解析位于 Path 的 XML 文件，可能是 XML 文件格式错误
46	A01083005	DDS 通信域 ID 设置错误, 使用默认域 ID: 10	构型配置文件中 DomainID 属性必须为 0～255 之间的值
47	D01084006	正在加载服务 Path	重要消息
48	A01082007	模型组 GroupName 优先级设置错误(0~99)，优先级值：P	构型配置文件中的模型组优先级必须在 0~99 之间
49	A01082008	模型组 GroupName 的 CPU 亲和性设置错误, CPU 亲和性值: Ag,进程 CPU 亲和性值: Ap	模型组的 CPU 亲和性设置必须是进程 CPU 亲和性的子集
50	D01084009	正在加载模型 Path	重要消息
51	A01082010	使用数据库构型配置时必须设置 XNCore 环境变量!	需要设置 XNCore 环境变量，且 XNCore 环境变量的 database 目录下有 XNSim.db 数据库文件

52	C01082011	打开数据库 Path 失败, 错误信息: Msg	打开位于 Path 的数据库时出错，输出无法打开数据库的错误消息 Msg
53	C01082012	数据库查询时准备 SQL 语句失败, 错误信息: Msg	查看无法查询数据库的具体错误消息 Msg
54	C01082013	数据库查询时绑定参数失败, 错误信息: Msg	查看无法查询数据库的具体错误消息 Msg
55	A01082014	未在数据库中找到构型 ID 为 confID 的记录	数据库 Configuration 表中找不到构型 ID 为 confID 的数据
56	A01083015	DDS 通信域 ID 设置错误, 使用默认域 ID: 10	数据库中构型配置的 DomainID 字段必须为 0～255 之间的值
57	C01082016	数据库查询时准备 LoadServices 表的 SQL 查询语句失败, 错误信息: Msg	查看无法查询数据库的具体错误消息 Msg
58	C01082017	数据库查询时绑定 LoadServices 表的参数失败, 错误信息: Msg	查看无法查询数据库的具体错误消息 Msg
59	D01084018	正在加载服务 Path	重要消息
60	C01082019	数据库查询时准备 LoadModelGroups 表的 SQL 查询语句失败, 错误信息: Msg	查看无法查询数据库的具体错误消息 Msg
61	C01082020	数据库查询时绑定 LoadModelGroups 表的参数失败, 错误信息: Path	查看无法查询数据库的具体错误消息 Msg
62	A01082021	模型组 GroupName 优先级设置错误(0~99)，优先级值：P	数据库中 LoadModelGroup 表中的模型组优先级必须在 0~99 之间
63	A01082022	模型组 GroupName 的 CPU 亲和性设置错误, CPU 亲和性值: Ag,进程 CPU 亲和性值: Ap	数据库中 LoadModelGroup 表中的 CPU 亲和性设置必须是构型 CPU 亲和性的子集
64	C01082023	数据库查询时准备 LoadModels 表的 SQL 查询语句失败, 错误信息: Msg	查看无法查询数据库的具体错误消息 Msg
65	C01082024	数据库查询时绑定 LoadModels 表的参数失败, 错误信息: Msg	查看无法查询数据库的具体错误消息 Msg
66	D01084025	正在加载模型 Path	重要消息
67	D01094001	事件管理器初始化成功	重要消息
68	D01094002	事件管理器准备就绪	重要消息
69	A01093003	注册的事件名称或回调函数为空，注册失败	注册事件时需要非空的事件名称和回调函数
70	D01094004	注册 同步/异步 事件处理器, 事件名称: Name, 处理器 ID:	重要消息

		CPUID (对象 ID: ObjectID，本地 ID: EventID)	
71	B01093005	移除事件 Event 时，事件不存在	内部错误，移除事件时找不到事件名称
72	D01094006	移除事件 Event 的处理器 ID: EventID	重要消息
73	B01093007	移除事件 Event 的处理器 ID: EventID 时，处理器不存在	内部错误，移除事件时找不到事件处理器
74	D01094008	移除事件 Event 的处理器 ID: EventID	重要消息
75	B01093009	移除事件 Event 时，处理器不存在	内部错误，移除事件时找不到事件处理器
76	B01092010	事件 Event 的处理器 ID: EventID 执行时发生异常: Msg	查看事件执行异常的具体错误消息 Msg
77	D01094011	设置事件处理线程池最大线程数为 count	重要消息
78	C01102001	创建主题 topicName 失败	FastDDS 异常
79	C01102002	创建主题 topicName 的发布者 publisherID 失败	FastDDS 异常
80	C01102003	创建主题 topicName 的发布者 publisherID 的数据写入器失败	FastDDS 异常
81	D01104004	创建主题 topicName 的发布者 publisherID 的数据写入器成功	重要消息
82	C01102005	创建主题 topicName 失败	FastDDS 异常
83	C01102006	创建主题 topicName 的订阅者 subscriberID 失败	FastDDS 异常
84	C01102007	创建主题 topicName 的订阅者 subscriberID 的数据读取器失败	FastDDS 异常
85	D01104008	创建主题 topicName 的订阅者 subscriberID 的数据读取器成功	重要消息
86	D01104009	DDS 管理器初始化成功	重要消息
87	D01104010	DDS 管理器准备就绪	重要消息
88	C01102011	DDS 管理器创建域参与者失败	FastDDS 异常
89	D01104012	DDS 管理器创建域参与者成功	重要消息
90	D01114001	模型管理器初始化成功	重要消息
91	D01114002	模型管理器准备就绪	重要消息
92	B01113003	分配模型 ID 失败，模型 ID 已用完，模型无法注册	注册的模型数量超过了 10000 个

93	B01113004	动态链接库 Path 中未找到模型 className	位于 Path 的动态链接库内没有找到名为 className 的模型
94	B01113005	动态链接库 Path 中未找到 initialclassName 函数	位于 Path 的动态链接库内没有找到名为 initialclassName 的函数
95	C01113006	动态链接库 Path 加载失败！错误: Msg	位于 Path 的动态链接库加载失败，查看具体加载错误消息 Msg
96	B01113007	模型 ID 对应的模型不存在	根据 ID 获取模型时找不到对应 ID 的模型
97	B01113008	模型 ID 对应的模型未注册	根据 ID 获取模型时模型未注册
98	B01113009	模型 ID 不合法	根据 ID 获取模型时 ID 不合法
99	B01113010	模型 ID 注册周期性函数失败，模型不存在	模型注册周期性函数失败，模型 ID 对应的模型不存在
100	D01124001	服务管理器初始化成功	重要消息
101	D01124002	服务管理器准备就绪	重要消息
102	B01123003	分配服务 ID 失败，服务 ID 已用完，服务无法注册	注册的服务数量超过了 10000 个
103	B01123004	动态链接库 Path 中未找到服务 className	位于 Path 的动态链接库内没有找到名为 className 的服务
104	B01123005	动态链接库 Path 中未找到 initialclassName 函数	位于 Path 的动态链接库内没有找到名为 initialclassName 的函数
105	C01123006	动态链接库 Path 加载失败！错误: Msg	位于 Path 的动态链接库加载失败，查看具体加载错误消息 Msg
106	B01123007	服务 ID 对应的服务不存在	根据 ID 获取服务时找不到对应 ID 的服务
107	B01123008	服务 ID 对应的服务未注册	根据 ID 获取服务时服务未注册
108	B01123009	服务 ID 不合法	根据 ID 获取服务时 ID 不合法
109	C01132001	线程 Name 初始化属性失败	pthread 异常或引擎没有足够的权限
110	C01132002	线程 Name 设置栈空间失败	pthread 异常或引擎没有足够的权限
111	C01132003	线程 Name 设置调度策略失败	pthread 异常或引擎没有足够的权限
112	C01132004	线程 Name 设置优先级失败	pthread 异常或引擎没有足够的权限
113	C01132005	线程 Name 设置调度器继承失败	pthread 异常或引擎没有足够的权限
114	C01132006	线程 Name 创建失败	pthread 异常或引擎没有足够的权限
115	B01133007	线程 Name 获取主框架失败	内部错误，会导致线程无法发送其运行数据
116	B01133008	线程 Name 获取 DDS 管理器失败	内部错误，会导致线程无法发送其运行数据
117	B01133009	线程 Name 获取 DDS 数据写入	内部错误，会导致线程无法发送其运行

		器失败	数据
118	D01134010	线程 Name 准备就绪	重要消息
119	D01134011	线程 Name 开始执行	重要消息
120	D01134012	线程 Name 暂停执行	重要消息
121	D01134013	线程 Name 继续执行	重要消息
122	D01134014	线程 Name 强制停止执行	重要消息
123	D01134015	线程 Name 正常停止执行	重要消息
124	C01133017	线程 Name 设置 CPU 亲和性失败	pthread 异常或引擎没有足够的权限
125	C01143001	模型配置文件 Path 打开失败	无法打开位于 Path 的模型配置文件
126	C01143002	模型配置文件 Path 解析失败	无法解析位于 Path 的模型配置文件，可能是 XML 格式错误
127	A01143003	模型配置文件 Path 中模型名称不一致, 模型名称: Name1, 配置文件中名称: Name2	模型配置文件中的模型名称与模型类名不一致
128	A01143004	模型配置文件 Path 中运行节点属性值不是 x-x 格式	模型配置文件中的运行节点属性值必须为 频率组-节点号 的形式，且频率组为[0,5]之间的值，节点号小于等于 2 的频率组次方
129	D01144005	模型 Name 加载数据包动态库 Path 成功	重要消息
130	A01143006	模型 Name 加载数据包动态库 Path 失败,将不调用数据包模型	模型无法找到或无法加载数据包模型动态库
131	A01142007	未设置 XNCore 环境变量, 无法解析模型配置	需要设置 XNCore 环境变量，且 XNCore 环境变量的 database 目录下有 XNSim.db 数据库文件
132	C01142008	打开数据库失败: Msg	打开数据库时出错，输出无法打开数据库的错误消息 Msg
133	C01142009	准备 SQL 语句失败: Msg	查看无法查询数据库的具体错误消息 Msg
134	C01142010	绑定参数失败: Msg	查看无法查询数据库的具体错误消息 Msg
135	A01142011	未找到机型为 PlaneName, 模型名称为 ClassName, 版本号 Version 的记录	未在数据库中找到对应模型版本的数据
136	D01144012	模型 Name 加载数据包动态库 Path 成功	重要消息
137	A01143013	模型 Name 加载数据包动态库 Path 失败,将不调用数据包模型	模型无法找到或无法加载数据包模型动态库

138	B01143014	获取主框架失败	内部错误，会导致模型无法发送其运行数据
139	B01143015	获取 DDS 管理器失败	内部错误，会导致模型无法发送其运行数据
140	B01143016	注册 DDS 参与者失败	内部错误，会导致模型无法发送其运行数据
141	B01143017	获取主框架失败	内部错误，会导致模型无法注册事件
142	B01143018	获取事件管理器失败	内部错误，会导致模型无法注册事件
143	B01143019	获取主框架失败	内部错误，会导致模型无法触发事件
144	B01143020	获取事件管理器失败	内部错误，会导致模型无法触发事件
145	C01153001	服务配置文件 Path 打开失败	无法打开位于 Path 的服务配置文件
146	C01153002	服务配置文件 Path 解析失败	无法解析位于 Path 的服务配置文件，可能是 XML 格式错误
147	A01153003	服务配置文件 Path 中服务名称不一致, 服务名称: Name1, 配置文件中名称: Name2	服务配置文件中的服务名称与服务类名不一致
148	A01152004	未设置 XNCore 环境变量, 无法解析模型配置	需要设置 XNCore 环境变量，且 XNCore 环境变量的 database 目录下有 XNSim.db 数据库文件
149	C01152005	打开数据库失败: Msg	打开数据库时出错，输出无法打开数据库的错误消息 Msg
150	C01152006	准备 SQL 语句失败: Msg	查看无法查询数据库的具体错误消息 Msg
151	C01152007	绑定参数失败: Msg	查看无法查询数据库的具体错误消息 Msg
152	A01152008	未找到服务名称为 ClassName, 版本号 Version 的记录	未在数据库中找到对应模型版本的数据
153	C01152009	解析其他参数失败：JSON	无法解析服务的其它参数，可能不是有效的 JSON 格式字符串
154	B01153010	获取主框架失败	内部错误，会导致服务无法注册事件
155	B01153011	获取事件管理器失败	内部错误，会导致服务无法注册事件
156	B01153012	获取主框架失败	内部错误，会导致服务无法触发事件
157	B01153013	获取事件管理器失败	内部错误，会导致服务无法触发事件
158	A01161001	T 必须是算术类型或 std::array 类型	获取 UDP 数据包时，某接口的数据类型不符
159	A01161002	T 必须是算术类型或 std::array 类型	通过 UDP 数据包注入数据时，某接口数据类型不符
160	A01161003	T 必须是算术类型或 std::array	获取数组类型的 UDP 数据包时，某接口

		类型	的数据类型不符
161	A01161004	T 必须是算术类型或 std::array 类型	通过 UDP 数据包注入数组类型数据时，某接口数据类型不符
162	A01161005	T 必须是算术类型或 std::array 类型	获取数据的字符串格式数据时某接口数据类型不符
163	A01163006	数组大小不一致，设置数据失败	通过字符串格式数据设置某接口数据时，数组长度不一致
164	A01161007	T 必须是算术类型或 std::array 类型	通过字符串格式数据设置某接口数据时，某接口数据类型不符
165	A01161008	T 必须是算术类型或 std::array 类型	获取数组类型的数据的字符串格式数据时，某接口数据类型不符
166	A01161009	T 必须是算术类型或 std::array 类型	通过字符串格式数据设置数组类型的接口数据时，某接口数据类型不符
167	A01163010	无法解析第 x 个元素：Msg	通过字符串格式数据设置数组类型的接口数据时，无法解析第 x 个元素，并输出错误信息
168	A01161011	模板参数 T2 必须是算术类型	对模型接口赋值时，某模型接口不是算术类型
169	A01161012	模板参数 T1 必须可以转换为 T2 类型	对模型接口赋值时，某模型接口的 DDS 类型与其在结构体中定义的类型不能转换
170	A01161013	模板参数 T1 是 std::array 类型时，它的数组嵌套不能超过两层	对模型接口赋值时，某模型接口如果是数组类型，不能超过 2 维
171	A01161014	模板参数 T1 是 std::array 类型时，它的 value_type 必须是算术类型或 std::array 类型	对模型接口赋值时，某模型接口如果是数组类型，其子类型必须为数组或算术类型
172	A01161015	模板参数 T1 必须是算术类型或 std::array 类型	对模型接口赋值时，某模型接口的 DDS 类型不符
173	A01161016	模板参数 T2 必须是算术类型	从模型接口取值时，某模型接口不是算术类型
174	A01161017	模板参数 T2 必须可以转换为 T1 类型	从模型接口取值时，某模型接口的 DDS 类型与其在结构体中定义的类型不能转换
175	A01161018	模板参数 T1 是 std::array 类型时，它的数组嵌套不能超过两层	从模型接口取值时，某模型接口如果是数组类型，不能超过 2 维
176	A01161019	模板参数 T1 是 std::array 类型时，它的 value_type 必须是算术类型或 std::array 类型	从模型接口取值时，某模型接口如果是数组类型，其子类型必须为数组或算术类型
177	A01161020	模板参数 T1 必须是算术类型或 std::array 类型	从模型接口取值时，某模型接口的 DDS 类型不符

178	A02012001	输入参数太少！使用 -h 查看帮助	XNEngine 的启动参数不能少于一个
179	A02012002	在-f 参数后，未指定构型配置文件路径，引擎将退出	使用-f 参数启动引擎时，必须在-f 后提供构型配置文件路径
180	A02012003	在-id 参数后，未指定构型 ID，引擎将退出	使用-id 参数启动引擎时，必须在-id 后提供数据库中已有的构型 ID
181	A02012004	无法识别的参数 p，引擎将退出	启动引擎时使用了无法识别的输入参数
182	A02012005	请不要同时使用 -f 和 -id 参数，引擎将退出	-f 和-id 参数只能使用其中之一
183	A02012006	构型配置文件不是 .xml 或 .sce 文件，引擎将退出	所选的构型配置文件不是.xml 或 .sce 文件
184	A02012007	构型配置文件没有后缀名，引擎将退出	所选的构型配置文件没有后缀名
185	C02012008	打开构型配置文件失败，引擎将退出	无法打开所选的构型配置文件，构型配置文件可能不存在
186	C02012009	解析构型配置文件失败，引擎将退出	无法解析所选的构型配置文件，可能是构型配置文件 XML 格式错误
187	A02012010	构型配置文件中未找到 Scenario 根元素，引擎将退出	构型配置文件的根元素必须为 Scenario
188	A02013011	无效的 CPU 亲和性值	构型配置文件的 CPU 亲和性属性设置错误
189	C02012012	设置引擎 CPU 亲和性失败	无法设置引擎的 CPU 亲和性，可能是权限不足
190	D02014013	成功设置引擎 CPU 亲和性	重要消息
191	C02012014	锁定引擎内存失败	无法锁定引擎的内存，可能是权限不足
192	D02014015	成功锁定引擎内存	重要消息
193	A02013016	构型配置文件中未找到 ConsoleOutput 元素，控制台将输出所有日志	构型配置文件中未配置 ConsoleOutput 元素
194	A02013017	配置文件中未找到 Log 元素，日志文件将记录所有日志	构型配置文件中未配置 ConsoleOutput 元素
195	B02012018	引擎初始化失败，引擎将退出	主框架初始化失败引起
196	D02014019	引擎初始化成功	重要消息
197	D02014020	引擎测试通过	重要消息
198	B02012021	引擎准备执行失败，引擎将退出	主框架准备执行失败引起
199	B02012022	无法发送引擎运行状态，引擎将退出	引擎无法获取到 DDS 管理器指针，无法发送引擎运行状态

200	A02012023	未设置 XNCore 环境变量，引擎将退出	需要设置 XNCore 环境变量，且 XNCore 环境变量的 database 目录下有 XNSim.db 数据库文件
201	C02012024	打开数据库失败: Msg	打开数据库时出错，输出无法打开数据库的错误消息 Msg
202	C02012025	准备 SQL 语句失败: Msg	查看无法查询数据库的具体错误消息 Msg
203	C02012026	绑定参数失败: Msg	查看无法查询数据库的具体错误消息 Msg
204	A02012027	未在数据库中找到构型 ID 为 confID 的记录	数据库 Configuration 表中找不到构型 ID 为 confID 的数据
205	A02013028	无效的 CPU 亲和性值	数据库中构型的 CPU 亲和性字段设置错误
206	C02012029	设置引擎 CPU 亲和性失败	无法设置引擎的 CPU 亲和性，可能是权限不足
207	D02014030	成功设置引擎 CPU 亲和性	重要消息
208	C02012031	锁定引擎内存失败	无法锁定引擎的内存，可能是权限不足
209	D02014032	成功锁定引擎内存	重要消息