ICS
CCS

团　　体　　标　　准

T/CIE XXX-2025

CAE 通用后处理引擎设计规范

CAE General Design Specifications for Post-Processing Engine

(草案)

2025-XX-XX 发布　　2025-XX-XX 实施

中国电子学会　发 布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由中国电子学会提出并归口。

本文件起草单位：杭州电子科技大学、大连理工大学、国家超级计算天津中心、粤港澳大湾区（广东）国创中心、北京神舟航天软件技术股份有限公司。

本文件主要起草人：徐岗、许佳敏、胡小飞、段莉莉、曾仲光、王磊。

CAE 通用后处理引擎设计规范

1 范围

本文件规定了CAE通用后处理引擎设计的基本原则、基本模块和技术架构。

本文件适用于CAE软件研发企业和CAE有限元仿真企业在设计、开发和使用CAE后处理引擎架构时参考。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1 CAE（Computer-Aided Engineering）

CAE（计算机辅助工程）是指利用计算机软件和数值方法对工程问题进行分析、仿真和优化的技术，帮助工程师在设计阶段预测产品性能、验证可靠性并改进方案。

3.2 API（Application Programming Interface）

API（应用程序编程接口）是一组预定义的规则、协议和工具，允许不同软件系统或组件之间安全、标准化地交换数据与功能。

3.3 CAE 通用软件架构（CAE Common Software Architecture）

面向计算机辅助工程（CAE）软件开发与集成的标准化软件架构框架，通过分层设计、接口规范与跨平台适配，实现仿真应用的快速开发、系统集成与持续演进。其核心特征包括：

- 架构分层性：划分为表示层、业务逻辑层、数据访问层与系统适配层，支持独立演化与模块解耦，提升软件维护性与可测试性。
- 接口标准化：定义统一的服务接口规范（如REST、gRPC、Python API）和数据交换协议（如JSON、HDF5、VTK），实现模块间的高效互操作。
- 平台无关性：支持主流操作系统（Windows、Linux）与部署环境（本地、云端、HPC集群），通过容器化与中间件屏蔽系统差异。
- 开发生态化：通过标准SDK与开发框架支持第三方插件与行业定制模块接入，构建可持续演进的CAE软件生态体系。

3.4 过滤器（Filter）

面向CAE数据处理流程的标准化处理组件，过滤器（Filter）是一类以模型、场数据或结构数据为输入，执行特定操作后生成新的模型或衍生数据的功能模块。通过管道式组合与参数化配置，支持复杂仿真预处理、后处理与中间转换操作的自动化与复用。其核心特征包括：

- 输入输出明确性：每个过滤器以特定类型的CAE对象（如网格、场、几何体）作为输入，输出处理后的对象或衍生结果，可串联成处理链。
- 处理能力多样性：支持常见操作如网格简化、网格优化、拓扑清理、几何映射、场重建、边界提取等，也可用于计算标量场、矢量场或导出中间特征。
- 组合执行性：过滤器之间可通过图结构或流水线方式组合执行，实现复杂数据流处理链的构建，如“几何体→网格生成→边界标识→边界场构建”。
- 参数可配置性：每个过滤器具备独立参数集，可通过GUI、脚本或API动态设置执行策略、精度控制、目标规则等。

3.5 模块（Module）

CAE通用后处理引擎通常根据不同的功能进行模块化划分，例如核心模块、过滤器模块和渲染模块等，以优化性能并提高系统的灵活性。这种结构化设计不仅有助于提升计算效率，还能增强系统的可扩展性，使其能够适应各种工程应用需求。

3.6 问题等级

对根据验收标准测试不通过的问题，按照其外部条件、出现概率、功能类型、影响程度、恢复条件等维度，及一定规则进行量化综合评估，根据评估结果对问题进行等级界定，共分为提示问题、一般问题、严重问题和致命问题四个级别。

问题级别定义方法

致命：必现，程序崩溃、闪退，主要功能流程失败，主要需求功能缺失。

严重：必现，功能出错，部分流程失败，卡在某一界面，通过重启等操作功能恢复正常

一般：不常用功能出错 不影响主要流程，不常用功能缺失，样式与设计不符，信息显示出错

提示：用户体验改进（样式、字体、弹框等），用户不易察觉的错误

3.7 缺陷指数 DI（Defect Index）

根据业界软件公司的通用方法，按照一定规则根据问题级别及问题数量计算项目 DI 值：

提示问题分值：0.1；

一般问题分值：1；

严重问题分值：3；

致命问题分值：10；

4 CAE 后处理软件架构总体架构设计

4.1 总体设计思路

设计思路如下：

1. 总体架构划分：

软件系统划分为四个模块：核心模块、过滤器模块、渲染模块和GUI模块。其中，核心模块、过滤器模块和渲染模块共同构成内核系统。内核系统内部各模块通过接口机制协同工作，实现模块间的解耦与高效协作。GUI模块则为内核系统提供用户界面支持，承担与用户交互的功能。核心模块负责实现系统的基础功能与核心服务；过滤器模块通过内核系统提供的标准接口与核心模块交互，用于实现特定的算法处理或功能扩展；渲染模块同样通过接口访问核心模块和过滤器模块所提供或处理的数据，并将其可视化呈现。

2. 数据流标准化处理：

对数据流进行标准化处理，确保数据结构的一致性以及转换接口的通用性，从而提升系统的可维护性与扩展性，同时简化过滤器模块的开发与集成工作。

3. 功能点规范化提炼：

梳理系统功能，设计统一的API调用接口，使过滤器及渲染模块能够通过标准化方式访问内核系统的各项功能，从而降低系统耦合性，提升模块的复用率和扩展能力。

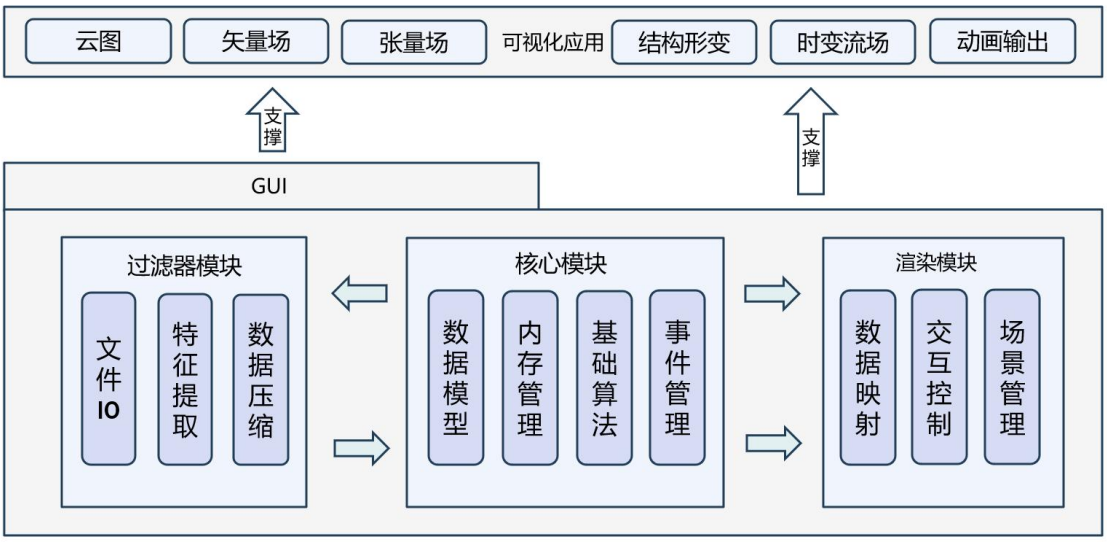
4. 业务流规范化提炼：

对业务流程进行标准化梳理，明确职责划分并定义模块，支持任务流程的自定义配置，从而提升系统的灵活性与可配置性，满足多样化业务场景的需求。

4.2 架构体系设计

架构体系设计遵循内核系统+GUI模块架构。

图1 整体架构体系设计图



软件系统采用分层架构设计，整体由GUI模块和内核系统两大部分构成。GUI模块作为用户操作的入口，连接上层的多种可视化应用需求，包括云图、矢量场、张量场、结构形变、时变流场和动画输出等，提供统一的界面支撑和交互通道。

内核系统作为系统的核心处理单元，是整个 CAE 后处理平台的关键支撑层，承担数据管理、计算支持、任务调度、可视化渲染与交互控制等核心功能，并向上层模块提供标准化的程序调用接口。内核系统由核心模块、过滤器模块和渲染模块构成，三者通过标准化的 API 接口协同工作，形成从数据处理、计算逻辑到图形渲染的完整技术链，全面支撑航天航空、汽车零部件、工业机械等领域的复杂可视化应用需求。

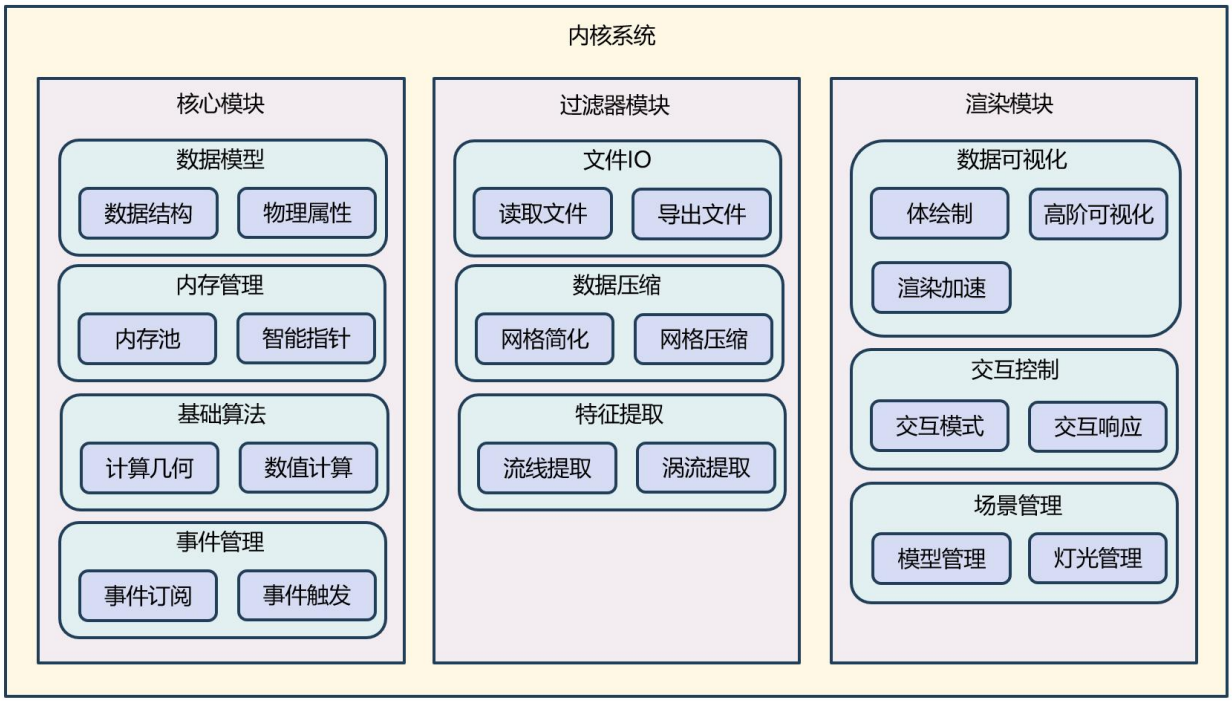
1. 核心模块则负责数据结构组织、内存分配与释放、通用算法执行及系统事件管理，构建了系统运行的基础逻辑平台，是支撑高性能处理的关键部分。

2. 过滤器模块主要负责数据处理及后处理算法的定义与执行，涵盖文件读写、数据类型转换、特征提取、网格处理、可视化算法应用，以及数据压缩与降维等操作，保障后续可视化任务的质量与效率。

3. 渲染模块则将处理后的数据进行可视化呈现，实现与用户的交互控制，并通过场景管理维护复杂图形结构和可视化状态，从而输出高质量的图形结果，满足不同应用场景下的可视化需求。

各模块之间既相互独立又密切协作，确保平台具备高性能、高扩展性与高可维护性的系统架构。其中，内核系统所包含的能力如下图所示：

图2 内核系统架构图



4.3 核心模块

核心模块是整个系统的基础，负责数据的组织、存储与计算支撑，主要包括数据结构、内存管理、基础算法和事件管理四个子模块。

4.3.1 数据结构子模块

数据结构模块负责对CAE仿真数据的统一表示与管理，为后续的处理提供坚实的数据基础。为满足复杂数据存储需求，其设计被划分为两大关键组成部分：一是网格结构，负责组织和管理几何数据，需要兼容多样化的网格类型，包括表面网格、体网格、结构化网格与非结构化网格等不同网格；二是属性，

用于描述附着在网格上的各类场信息，包括标量场（如温度、压力）、矢量场（如速度、应力方向）以及张量场（如应力张量、应变张量）等。

数据结构的设计需着重考虑以下关键特性，确保其可扩展、高效及实用：

a) 可扩展性：需要充分考虑网格结构和物理属性的可扩展性，能够兼容多种类型的网格数据，能够为未来出现的多种类型的网格数据与物理场数据提供灵活的集成方法。

b) 节约内存：需要优化数据存储，降低内存占用，以支撑大规模仿真数据处理能力与可视化。

c) 高效访问：设计高效的访问接口，确保对网格结构和属性数据的读写操作具有高效的性能，减少数据访问瓶颈，为后续数据处理与高性能可视化提供高效的底层支撑。

4.3.2 内存管理子模块

内存管理子模块主要负责系统运行过程中的内存资源控制，通过引入内存池机制以及智能指针等现代C++技术手段，有效降低频繁内存分配与释放带来的系统开销，提升系统的内存安全与稳定，避免内存泄漏。需要满足以下关键特性：

a) 健壮性：需具备容错能力，确保在发生错误或遭遇异常情况时，能够有效维持核心功能运行，避免因局部故障导致整个系统崩溃。

b) 高效性：提供高效的内存管理及访问机制，减少不必要的内存操作开销和延迟，以提升系统的整体处理速度与响应能力。

c) 稳定性：保障系统能够长时间持续稳定运行，防止出现内存泄漏性能衰减等问题。

4.3.3 基础算法子模块

基础算法子模块提供丰富的通用计算能力，涵盖常见的计算几何算法（如点面关系判定、包围盒计算等）以及数值计算算法（如线性代数运算、插值算法等数学库函数）。这些算法作为数据处理的基本工具，为数据压缩、特征提取与可视化渲染等功能模块提供底层算力支持。需要满足以下关键特性：

a) 高效性：内部算法需采用优化的数据结构和实现策略，能够支撑高频调用情景。

b) 通用性：确保算法的可复用性和模块化，能够为不同的功能提供同一的算法接口。

c) 可靠性：需具有良好的数值稳定性和鲁棒性。

4.3.4 事件管理子模块

事件管理子模块提供系统级的事件调度机制，支持模块之间的异步通信与状态同步。通过建立标准化的事件发布、订阅、触发机制，确保用户交互、数据更新、渲染刷新等操作在系统内能够快速联动并稳定响应，提高了系统整体的交互性与可维护性。

4.4 过滤器模块

过滤器模块是对数据处理算法的抽象中间层，通过标准化流程实现数据输入输出处理、压缩优化及特征提取等关键功能，提升整个系统的可扩展性。其核心子模块设计如下：

a) 文件 IO 子模块：支持多种常见 CAE 数据格式的读取与写出，如 VTK、VTU、CGNS 等，实现数据的无缝导入与导出，确保不同数据的兼容性与读写效率。

b) 数据处理子模块：定义丰富的过滤器类，用于实现多样化的后处理功能。例如，数据压缩过滤器支持对表面网格与体网格的简化策略及压缩编码技术，有效降低内存占用和存储开销；交互式特征提取过滤器则支持从仿真场中提取关键结构，如构建流线可视化中的路径线、自动检测与抽取旋涡区域等。

引入过滤器机制作为系统的扩展方式，支持动态加载和卸载过滤器，使 CAE 软件能够灵活应对不同需求。过滤器间采用标准接口进行通信，确保高度的独立性。过滤器机制赋予系统强大的扩展能力和灵活性，将功能模块作为独立的过滤器进行开发和部署，实现系统的模块化和可插拔性。

4.5 渲染模块

渲染模块是实现数据可视化与用户交互体验的核心组件，旨在支撑高性能、高质量的图形渲染与多模式交互操作，服务于后处理与分析平台的可视化需求。该模块整体架构划分为以下三个子模块：

a) 数据可视化子模块：负责多类型仿真或分析数据的渲染，支持多种渲染技术，包括基于体数据的体绘制（Volume Rendering），透明与不透明表面的多通道渲染，以及高阶参数曲面和曲线的可视化渲染，满足不同类型数据的表达需求。

b) 交互控制子模块：提供灵活且高效的用户操作能力与场景管理机制，支持模型树管理以组织和调度可视化对象，涵盖模型的加载、卸载与动态切换，同时支持视图控制与光照参数配置，确保复杂场景下渲染结果的表现力。此外，交互模块支持自由旋转、缩放、对象拾取、动态高亮与参数调节等功能，为用户提供直观、高效的分析流程。

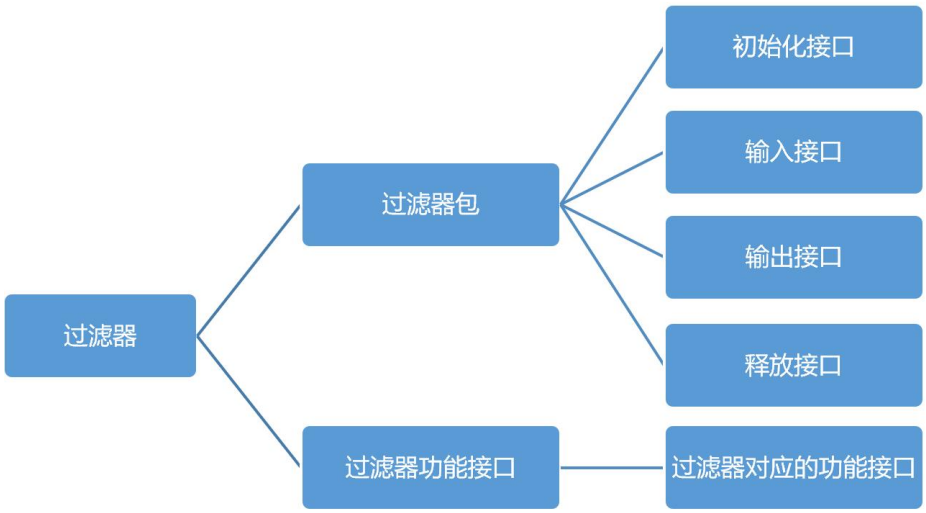
c) 高性能计算子模块：面向大规模数据处理，采用CPU-GPU混合渲染架构，并结合并行计算框架以充分挖掘异构硬件资源的计算潜力，同时支持大规模并行渲染能力，显著提升系统在高负载可视化任务中的响应速度与稳定性。

5 过滤器开发设计

本章节介绍了接入 CAE 通用后处理引擎的过滤器开发标准及开发过程中需注意的要点，旨在为过滤器设计与开发人员提供规范化的指导和参考。

5.1 过滤器的组成设计

图3 过滤器的组成设计图



过滤器应提供以下两类接口，均须对外导出，确保可被CAE通用后处理引擎调用：

- 过滤器标准接口：指CAE通用后处理引擎规定的通用交互接口，用于实现系统与插件之间的统一通信机制。过滤器需按规范实现相应功能。标准接口通常包括：

a) 初始化接口 (Initialization Interface)

b) 输入接口 (Input Interface)

c) 输出接口 (Output Interface)

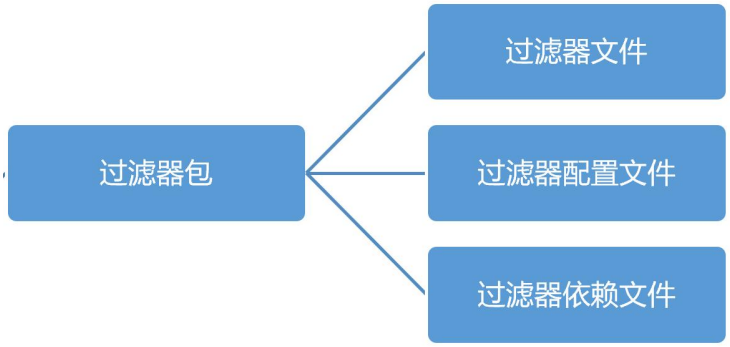
d) 释放接口 (Relea Interface)

- 过滤器功能接口：指过滤器为实现特定业务或算法功能所定义的核心接口。此类接口通常为过滤器自主设计，且数量不定，具体依据过滤器实现的功能模块而定。CAE通用后处理引擎通过调用这些接口完成具体功能的访问与执行。

注意：这两类接口都必须导出来，可被 CAE 通用后处理引擎调用的接口。

5.2 过滤器包的组成设计

图4 过滤器包的组成设计图

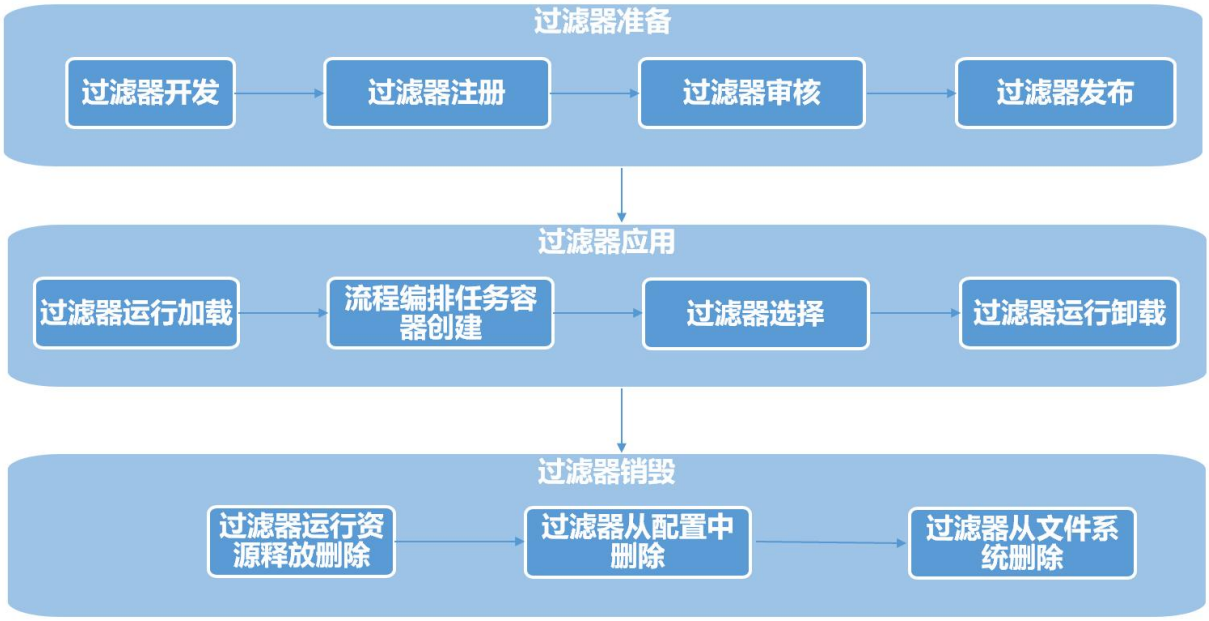


过滤器开发完成后，需要把过滤器打包成一个整体文件，方便CAE通用后处理引擎注册使用。一般包括三类文件：

- 过滤器文件：是按CAE通用后处理引擎标准开发生成的可执行文件以及其依赖文件。
- 过滤器配置文件：是按CAE通用后处理引擎注册要求用于描述过滤器信息的配置文件，信息内容通常包括过滤器名、过滤器类型、过滤器版权、过滤器校验、过滤器作者等信息。
- 过滤器依赖文件：是指过滤器运行过程中可能会用到的运行依赖库存或者第三方的插件信息。

5.3 过滤器生命周期设计

图5 过滤器生命周期设计图



过滤器生命周期可分为三个阶段：

1. 过滤器准备

包括过滤器开发、过滤器注册、过滤器审核、过滤器发布阶段。

a) 过滤器开发：是指开发人员按照过滤器的规范标准创建完成一定功能的模块。

b) 过滤器注册：是指过滤器开发完成后需要在CAE通用后处理引擎下完成注册，注册过程中需要填写一定的过滤器信息，包括过滤器名、过滤器描述、过滤器版权、过滤器应用范围、过滤器权限归属等。

c) 过滤器审核：是指过滤器上传到CAE通用后处理引擎待系统或者管理员审核。

d) 过滤器发布：是指在系统或者管理员审核通过后，会正式发布到CAE通用后处理引擎平台，使用CAE通用后处理引擎的相关人员可使用该过滤器（如果为单机版时只能供当前机使用）。

2. 过滤器应用

包括过滤器运行加载、流程编排任务容器创建、选择过滤器、过滤器运行卸载。

a) 过滤器运行加载：是指过滤器做为CAE通用后处理引擎的一部分，在系统运行后加载到内存（也可以在真正使用到的时候进行加载）。

b) 流程编排任务容器创建：是指在过滤器正式应用前需要有任务容器做为载体，因为每个任务容器相当于需要实现特定API功能的容器，而任务容器需要实现特定功能必须依赖过滤器提供的API功能。

c) 选择过滤器：是指任务容器的功能可能会被多个过滤器的实现API实现，需要选择特定的过滤器来实现该容器的API功能。

d) 过滤器运行卸载：是指在CAE程序退出后调用过滤器提供的释放接口释放过滤器的资源，从而在程序退出前完成过滤器的卸载。

3. 过滤器销毁

包括过滤器运行资源释放、过滤器从配置中删除、过滤器从文件系统中删除。

a) 过滤器运行资源释放：是指CAE通用后处理引擎调用过滤器的释放资源接口完成过滤器运行状态资源的释放。

b) 过滤器从配置中删除：是指将过滤器的配置信息从CAE通用后处理引擎配置文件或者数据库中删除，从而达到程序不会再次进行加载。

c) 过滤器从文件系统中删除：是指销毁过滤器后还需要把过滤器的文件从文件系统中删除。

5.4 过滤器代码开发设计

1. 明确实现功能所属的模块，比如过滤器需要实现前处理功能，或者要实现求解器功能等。

2. 明确实现具体的 API，每个 API 的输入输出参数必须与 CAE 通用后处理引擎所要求的一致。

3. 明确开发过滤器的工具，编程语言，如选择 VS & C++2022 & QT 方式。

4. 开发，调试，测试过滤器实现的 API 是否达到预期。

5. 导出 API 供 CAE 平台可调用，CAE 平台调用过滤器时可通过动态库或者通过 Python 脚本的方式加载。

6. 生成过滤器配置文件，按照 CAE 通用后处理引擎注册过滤器的要求，配置好过滤器的信息。

7. 打包过滤器，过滤器及过滤器配置文件按照 CAE 通用后处理引擎的要求放置好，最终生成 CAE 通用后处理引擎所要求的过滤器包格式。

6 界面与交互设计

6.1 设计原则

1、简化

a) 功能的简化：功能的简化不等同于功能目标的减少，有时候为了更快实现功能目标，反而会增加功能项，比如默认参数、智能选项等。功能的简化可以将复杂功能进行简化实现，更高效地达成功能

目标；

b) 流程的简化：在一般情况下，交互的步长越短，交互的效率越高。在一个步骤里不应粗暴塞入过多的信息，同时在确保功能的情况下适当简化流程步数，明确流程逻辑，才能即提高用户满意度，也可以提高操作效率；

c) 页面布局的简化：根据席勒定律（一个人面临的选择越多，所需要作出决定的时间就越长），在页面布局设计中，应充分利用相近原则、对比原则，使得同一层级下的选择达到一个尽可能少的合理数量，以避免造成决策瘫痪与认知负担。

2、可用性与易学性

可用性与易学性往往是相关的，对于工具类软件与平台来说，可用性的优先级高于易学性。用户对此类软件使用频率是很高的，所以对效率的追求优先于对易学性的追求。因此我们不仅仅需要关注短期的学习曲线，更应该关注长期的学习曲线。

3、一致性

a) 平台与工具间的一致性：在设计语言与交互规则上，同平台下的多产品与工具之间应尽可能地保持一致性，避免增加用户的认知与学习成本；

b) 设计语言的一致性：应拥有统一的设计规范，以保证设计语言的一致性，避免风格样式与交互规则的错乱和冲突；

c) 与用户预期的一致性：不应该在任何情况下挑战用户对世界的基本认知，比如：我们通常认为将一个文件拖动到一个文件夹是一个‘移动’的动作，那么就不要对这个动作有‘移动’以外的定义。如果用户期待某个操作以某种方式进行，尽可能地满足这种期待；

d) 不一致性：当元素行为不同时，其设计语言应该具有明显的不一致，既设计学中的对比原则。

4、用户需求预测

a) 尽力去预知用户的需求。不要期待用户离开当前的页面去寻找他所需要的信息，所需的信息和工具必须放在合适的位置并且要明显可见；

b) 在交互设计时，需站在频繁使用软件的用户视角去审视元素操作，也需在设计前进行充分的调研和分析，在设计后进行足量的测试。密切关注每一个环节的反馈，从而进行调优。

5、及时有效的反馈

对用户的操作有所回应，让用户明确自己每一步操作是否有效，缺少反馈会使用户产生焦虑。

a) 回应用户操作：用户发起的所有操作都应该伴随一个反馈，比如：用户按下按钮的时候，必须伴随一个回馈，表示按钮操作已经被我们接收到了；

b) 进度反馈：对于一些比较长的流程以及较长的等待时间，可以用进度条的形式，告诉用户现在的进程状态以提高用户的可控感，从而降低等待的焦虑；

c) 系统内外变化：产品的运行可能依赖于服务器、网络、设备等等因素，如因外部条件导致产品无法正常运行，也需要及时的反馈与提示。产品内的变化，也需要及时反馈，比如：审核成功提醒，状态的变更等。

6.2 界面元素设计

6.2.1 文字

1. 字体选择

a) 界面上呈现的字体设计方案应保持一致，建议采用无衬线字体（如 Harmony OS Sans 或系统默认字体），以提高可读性。避免使用花哨或难以辨认的字体，确保文字在不同分辨率和设备上清晰可见。

2. 字号与对比度

a) 文字与背景应具有良好的颜色对比，普通文本对比度至少为 4.5:1，主要标题和关键信息对比度至少为 3:1；

b) 主要功能及关键信息的字号不小于 18dp 或 pt，次要信息的字号不小于 14dp 或 pt。

3. 行间距与段落

a) 段落内文字的行距应至少为 1.3 倍，段落间距应至少比行距大 1.3 倍；

b) 重要信息应通过加粗、高亮或分段的方式突出显示，避免大段文字堆砌。

6.2.2 颜色

1. 颜色用途与无障碍设计

a) 颜色应与其他编码技术组合使用，不应作为传递信息的唯一方式。例如，警告信息应同时使用红色和文字提示；

b) 遵循 W3C WCAG 2.2 标准，确保文本与背景的对比度至少为 4.5:1（AA 级）或 7:1（AAA 级）；

c) 在工业 CAE 后处理可视化中，颜色编码应符合行业标准，如红色表示警告、黄色表示注意、绿色表示正常。

2. 颜色数量与对比度

a) 除黑白色外，界面上的颜色数量不宜超过 6 种，以避免视觉疲劳；

b) 前景色和背景色应具有显著对比，建议背景色使用淡灰色或白色，避免高饱和度颜色。

3. 数据可视化

a) 在数据可视化（如标量云图、热力图等）中，建议使用渐变色谱图，并提供图例说明颜色与数据的对应关系；

b) 设计不同色谱图时，尽量避免使用对色盲用户不友好的颜色组合（如绿色和红色）。

6.2.3 符号/图标

1. 一致性与可理解性

a) 图标的设计风格、大小、形状和颜色应保持一致，建议采用统一的图标库；

b) 图标应符合用户的认知习惯，避免使用过于抽象或难以理解的符号。不带文字说明的图标，应能够让 80%的人通过图像辨别理解。

2. 尺寸与布局

a) 图标尺寸应统一，建议主要功能图标尺寸不小于 24×24dp 或 pt，次要功能图标不小于 18×18dp 或 pt；

b) 为非常见图标提供文字说明或悬停提示，确保用户能快速理解其含义。

3. 工业应用中的符号规范

在工业 CAE 后处理可视化中，符号和图标应符合 ISO 9241-110:2020 中关于符号使用的建议，确保符号在不同场景下的可理解性和一致性。

6.2.4 按钮

1. 类型与用途

a) 根据使用目的选择适宜的按钮类型（如主要按钮、次要按钮、危险按钮等）；

b) 主要按钮应突出显示，建议使用高对比度颜色（如蓝色或绿色），次要按钮使用低对比度颜色（如浅蓝色）或者中性色（如灰色）。

2. 交互状态

a) 按钮的交互状态（如悬停、点击、禁用）应清晰可辨，建议通过颜色变化、阴影或边框变化体现；

b) 禁用状态的按钮应保持可见，但通过灰度化或透明度降低表明不可操作。

3. 尺寸与布局

a) 按钮尺寸应适中，建议主要按钮的最小点击区域为 36×36dp 或 pt，次要按钮为 28×28dp 或 pt；

b) 按钮间距应足够大，避免误触，建议按钮间水平间距不小于 8dp 或 pt，垂直间距不小于 8dp 或 pt。

6.2.5 表格

1. 信息组织与展示

a) 表格应用于将信息组织成视觉上可区分的子集，建议采用层次化展示（如主表、子表）。

2. 标题与分隔

a) 表格的行和列标题应始终可见，建议使用固定表头或滚动表头；

b) 表格中应有明显的栏间距，不同栏之间以适当的方式分隔（如浅色分隔线或背景色）。

3. 交互功能

a) 支持表格的排序、筛选和搜索功能，建议在表头提供排序箭头和筛选下拉菜单；

b) 对于大量数据，建议提供分页功能，并在页脚显示当前页码和总页数。

6.2.6 弹窗

1. 类型与用途

a) 根据使用场景选择适宜的弹窗类型（如提示弹窗、确认弹窗、输入弹窗等）；

b) 弹窗应简洁明了，避免信息过载，建议在弹窗中突出显示关键信息。

2. 交互与布局

a) 弹窗的关闭按钮应明显且易于点击，建议放置在弹窗的右上角；

b) 弹窗的功能按钮应明显且易于点击，建议在下方放置确认和取消，且确认放置在右侧；

c) 弹窗的尺寸应适中，避免遮挡过多内容，建议最大宽度不超过屏幕宽度的 60%。

6.3 界面布局设计

6.3.1 概述

界面布局设计应遵循一致性、简洁性和易用性原则，确保用户能够快速找到所需信息并完成操作。布局设计应支持用户在不同页面的相同位置找到相似的信息，同时适应不同屏幕尺寸和设备类型。

6.3.2 界面信息的位置

1. 信息分类与组织

a) 界面信息应按照逻辑进行分类和组织，例如将工具栏、数据展示区、操作区和状态栏分开。界面信息也应按照业务流程和常用操作习惯进行组织；

b) 同时常用功能和关键信息应放置在用户容易访问的位置，例如屏幕顶部或左侧。

2. 用户期望的位置

a) 信息的位置应符合用户期望和任务要求，例如工具栏通常放置在屏幕顶部，数据可视化区域放置在中间；

b) 避免信息位置的频繁变化，以减少用户的学习成本。

6.3.3 界面分隔与间距

1. 信息分组

a) 将界面信息按照功能或语义进行分组，有助于用户快速找到和理解信息；

b) 可通过间隔、边框或背景色等方式区分不同的信息组。避免为了增加信息而减小字符大小，从而影响可读性。

2. 间距

同层级的功能模块间保持相同的间距，最小间距不低于 8dp 或 pt。

6.3.4 信息显示密度

1. 显示信息密度

a) 显示信息密度不应过于拥挤或过于疏松，建议实际字符数占总字符数的比例为 40%左右；

b) 对于图形用户界面，图形元素（如线条、按钮和图标）应合理布局，避免增加不必要的拥挤感。

2. 重点信息突出

重要信息应通过颜色、大小或位置等方式突出显示，避免信息淹没在大量次要信息中。

6.3.5 窗口长度

1. 窗口大小控制

窗口大小应适宜，以支持窗口的主要目的和使用。在设计窗口大小时要考虑到窗口与窗口之间的间距和互相影响。例如，避免组件窗口过大导致渲染窗口过小。

2. 快速导航

如窗口长度超过整个屏幕或者最佳可视化长度，应提供分页或者滚动页的功能，同时可快速导航为第一页或者顶部，例如滚动条或“返回顶部”按钮。

6.3.6 页面滚动方式

1. 滚动优化

宜使页面垂直滚动最小化，可通过将重要信息放在页面顶部，并为页面下方的信息提供快捷入口的方式实现。或采用折叠的设计，将信息分组，进行统一显示或者隐藏。

2. 避免水平滚动

尽量避免页面水平滚动，如需水平滚动，应提供明确的提示和操作引导。

6.3.7 工具栏与操作区

1. 工具栏设计

a）工具栏应放置在用户容易访问的位置，通常为屏幕顶部或左侧；

b）工具栏中的按钮应按照功能相关性进行分组，并提供悬停提示或图标说明。

2. 操作区布局

a）操作区应支持用户快速完成任务，例如将常用操作按钮放置在显眼位置；

b）操作区的布局应支持多任务处理，例如提供标签页或多窗口支持。

6.3.8 数据可视化区域

1. 数据展示

a）数据可视化区域应支持多维度展示，例如云图可视化、图标可视化、流场可视化等；

b）数据可视化区域应提供交互功能，例如缩放、平移和筛选。

2. 信息层次

a）数据可视化区域的信息应依据逻辑，按层次展示，避免信息过载；

b）提供图例和说明，确保用户能够快速理解数据含义。

6.3.9 状态栏

a）状态栏应放置在屏幕底部或顶部，用于显示当前操作状态、进度和系统信息；

b）状态栏应保持简洁，避免信息过载。

6.4 人机交互设计

6.4.1 输入模块

1. 输入方式

主要输入方式为两种，为键盘输入和鼠标选择。例如，在仿真参数设置中，用户可以通过键盘快速输入数值或者通过交互界面进行参数的选择（比如切面），预计未来可通过上传脚本进行快速设定。

2. 输入效率

优化输入流程，减少不必要的输入步骤。例如，提供最近文件路径，方便用户输入最近模型；提供默认值或推荐值，用户可以通过简单的确认操作完成设置。

3. 输入校验

在用户输入过程中，提供即时的输入校验和反馈，避免错误输入导致的可视化失败。例如，当用户输入的参数超出合理范围时，系统应立即提示并建议修正（如法向量定义为全 0）。

6.4.2 输出模块

1. 视觉反馈

用户操作后，界面应立即提供视觉反馈，例如按钮变色、图标动画或进度条更新，以确认操作已被系统接收。例如，当用户选择“导入模型”后，进度条会进行更新，显示导入进度。

2. 错误反馈

当用户操作出错时，系统应提供明确的错误提示，并建议解决方案。例如，当用户输入的参数或者选择的选项不符合要求时，系统可以弹出提示框，指出错误原因并提供修正建议。

6.4.3 动效

1. 动效原则

a）动效设计应遵循舒适性、高效性和流畅性原则，确保用户操作的自然性和流畅性；

b）动效时长应控制在 200ms 至 500ms 之间，避免过长或过短的动画影响用户体验。如果模型过大导致算法时间过长，可以采用迭代结束更新的设计，避免中间过程更新导致高延迟效果。

2. 动效类型

a）引导动效：用于引导用户注意力，例如在新功能上线时，通过动画高亮显示关键按钮或区域。

b）过渡动效：在页面切换或功能切换时，使用平滑的过渡动画，减少视觉断裂感。

c）反馈动效：在用户完成操作后，通过动画效果确认操作结果，例如文件载入时的进度条动画。

d）动效应用：在工业 CAE 后处理中，动效可用于展示仿真结果的变化过程，例如通过动画展示流场的变化，用户更直观地理解模型中数据流的变化状况。

6.4.4 控制逻辑

1. 交互规则

系统应提供清晰的交互规则，确保用户能够理解如何与系统进行交互。例如，在多步骤且有前后步骤明确要求的流程中，在未执行上一操作时，无法进行下一步操作的选择。

2. 错误处理

系统应具备完善的错误处理机制，能够在用户操作出错时及时捕获错误并提供解决方案。例如，当可视化因内存不足而失败时，系统可以提示用户释放部分模型。

6.5 UI 代码设计标准

1. 组件类的接口

a）接口名称遵守统一的命名风格；

b）对于每个widget，必须确保窗口的打开、关闭、隐藏和显示四种状态下的数据更新，初始化代码力求完整，显示和隐藏需要关注组件的数据更新，关闭需要遵守数据清空原则。

2. 功能性要求

a) 支持模型背景颜色的整体变更；

b) 支持人机交互设计章节中的视觉效果，包括颜色、字体、形状、动效等；

c) 界面控件需要根据产品的需求设定对应事件。

7 CAE 通用后处理引擎的验证测试标准

本章节规定了 CAE 通用软件架构验证测试的流程规范、管理规范和文档规范；适用于 CAE 软件研发企业和 CAE 有限元仿真企业在验证测试时参考。

7.1 测试流程规范

对测试活动的流程进行明确，过程进行规范，使产品测试活动更加科学有序的进行，提高产品质量。迭代过程包含以下测试活动：

7.1.1 版本测试方案设计

版本测试方案设计用于验证软件的功能、性能和稳定性，以确保软件能够按照预期的方式运行，用于指导后续软件测试活动的高效执行。要求如下：

1. 版本测试方案设计，需要包括如下核心内容：
   - 确定测试目标：在设计版本测试方案之前，需要明确测试的目标和范围。这包括确定要测试的功能、性能指标和预期的稳定性要求。
   - 制定测试计划：根据测试目标，制定一个详细的测试计划。测试计划应包括测试的时间表、资源需求、测试环境的设置和测试用例的设计。
   - 测试用例设计指导：说明测试用例的设计方法和准则，包括功能点的覆盖、边界条件的考虑、异常情况的测试等。
   - 测试范围：功能、性能、稳定性、安全等，同时需要给出与下游联调的计划和主要场景。
   - 自动化测试方案：描述整个版本的自动化方案，包括需要引入的工具，测试平台。
   - 测试环境：需要设置一个合适的测试环境进行版本测试，包括安装和配置测试软件、模拟真实用户的行为、设置测试数据等。
2. 版本测试方案需要进行评审，评审人员包括测试经理、测试工程师和相关开发人员；
3. 评审后的版本测试方案需要归档；
4. 版本测试方案设计的输出包括：版本测试方案、版本测试方案评审纪要、测试计划。

7.1.2 测试用例设计

测试人员在开发进行代码编写时，进行用例的编写。测试用例设计需要明确测试的范围、设计测试数据和确定预期结果。在提测后，根据实际测试过程中对需求、功能和用例的进一步验证过程中，可以对用例进行完善。要求如下：

1. 理解需求：需要仔细阅读和理解软件的基线需求，包括功能需求、性能需求，安全需求等非功能需求，确定要测试的功能和特性。
2. 确定测试目标：根据需求和测试的目标，确定测试的范围和重点，明确测试用例的设计方向和覆盖范围。
3. 划分测试场景：根据软件的功能和特性，将测试用例划分为不同的场景。每个场景应该覆盖一个特定的功能或使用情况。
4. 设计测试数据：为每个测试场景设计合适的测试数据。测试数据应该具有不同的输入值和边界条件，以验证软件的正确性和鲁棒性。
5. 确定预期结果：对于每个测试场景，确定预期的输出结果，辅助判断软件是否按照预期工作，并检测出潜在的问题。
6. 设计测试用例：根据测试目标和场景，设计正向测试用例来验证软件的功能是否按照预期工作。正向测试用例应包括正确的输入和预期的输出；设计逆向测试用例用于验证软件的健壮性和错误处理能力。
7. 确定优先级：为每个测试用例确定优先级和覆盖率。根据测试目标和资源限制，确定哪些测试用例应该优先执行，以确保测试的有效性和效率。
8. 测试用例评审：用例编写完成后，需要组织评审，评审人员包括 TSE、SE、开发、开发 PL；
9. 测试用例设计的输出包括：测试用例，测试用例评审会议纪要。

7.1.3 版本迭代测试

在每个迭代周期内，测试团队与开发团队紧密合作，进行测试工作，确保版本迭代测试的全面性和准确性，及时发现和解决软件产品中的缺陷和问题。版本迭代测试过程中要求要严格按照测试用例进行测试。测试的主要指标及质量标准如下：

- 需求覆盖率：代码功能对需求覆盖；质量标准：100%
- DI值：根据缺陷严重情况计算得到分数；质量标准：<15分（不允许出现致命缺陷）
- 缺陷闭环比例：除了遗留问题，其他问题回归关闭比例；质量标准：100%

- 版本提测成功率：开发提交测试无明显致命问题导致无法开始测试的比例；质量标准：100%

编写迭代测试报告对测试过程总结和回顾，给出版本质量评价，缺陷收敛情况。

对测试结果进行记录、分析和总结，包括测试覆盖率、通过的测试用例数量、失败的测试用例数量等统计数据，对遗留问题和影响进行风险分析，版本迭代测试报告需要经过评审，

输出：版本测试报告，测试报告评审会议纪要。

7.2 测试管理规范

责任分工：

角色	职责	要求
测试经理	制定测试计划，监控进度	计划偏差≤3 天
开发人员	修复优先级为“高”的缺陷	24 小时内响应
测试工程师	执行用例并提交缺陷	缺陷描述完整

7.3 测试文档规范

7.3.1 版本测试方案要求：

引言：目的、背景、范围、定义

测试内容：测试功能清单

测试规则：通用规则，测试方法（黑盒、灰盒、白盒测试）、测试要点、测试工具

测试环境：硬件环境、软件环境、特定测试环境要求

项目任务：测试规划，测试设计，测试执行准备，测试执行，测试总结

文档输出：版本测试方案/计划

7.3.2 测试用例要求：

统一测试用例编写的规范，为测试设计人员提供测试用例编写的指导，提高编写的测试用例的可读性，可执行性、合理性。为测试执行人员更好执行测试，提高测试效率，最终提高整个产品的质量。

7.3.3 用例编写规范

用例按照以下要素填写：

测试用例八要素：（用例编号，测试项目（所属模块），测试标题，重要级别（优先级），前置条

件，测试输入，操作步骤，预期结果）

1、用例编号：（规则：由字符和数字组成的字符串，具有唯一性，易识别性）

2、测试项目（所属模块）：（对应测试用例编号中的测试子项名 系统测试

3、测试标题：（体现测试出发点关注点以及测试用例期盼的测试结果）

4、重要级别、优先级别：（重要级别分为 P0，P1，P2，P3）

5、前置条件：测试用例在执行时需要满足一些前提条件，环境的设置

6、测试输入：（测试执行中需要加工的外部信息，避免用描述性语言，要具体，根据测试用例具体情况，有手工输入，文件，数据库记录）

7、操作步骤： 执行当前用例需要经过的操作步骤，需要明确的给出每一个步骤的描述

8、预期结果： 需要判断测试对象是否正常工作

7.3.4 测试报告规范

测试报告包含以下内容：

1、编写目的

本测试报告为 XXX 版本测试报告，目的在于总结测试阶段的测试情况以及分享测试结果，描述系统是否符合用户需求，是否已达到需求文档中预期的功能目标，并对测试质量进行分析。

测试报告参考文档提供给测试人员、开发人员、产品人员、运营人员、和需要阅读本报告的高层管理人员阅读。

2、测试范围

本测试报告是针对<XXX 系统需求规格说明书>中规定内容进行测试，包括以下需求功能：

需求大纲（当前版本包含的需求点）

3、测试概要

3.1、进度回归（整个 xx 项目的测试时间从 xx 年 x 月 x 日开始，到 xx 年 x 月 x 日上线止，期间各阶段工作情况如下）测试时间，测试人员（测试起止时间）

3.2、测试执行

此次测试严格按照项目计划和测试计划执行，按时完成了测试计划规定的测试对象的测试。针对测试计划规定的测试策略，在测试执行中都有体现，在测试执行过程中，依据测试计划和测试用例，对系统进行了完整的测试

4、测试环境，测试设备（用到哪些测试收集，客户端环境，浏览器）

5、Bug 数据分析（从多个维度分析：bug 等级分布，遗留 bug 分析，bug 类型分布。模块 bug 分布，bug

激活次数分

6、测试用例报告（报告用例执行情况）

7、已知风险、未知风险

8、测试结论（是否达到发布标准，是否可发布）

9、测试总结（从测试角度，对版本存在的问题，提出建议）

附 录 A
（资料性）
XXXXXXXXXXXXXXXX

A.1 XXXXXXXXXXXX。

表 A.1 XXXXXXXXXX 表