

**硕士学位论文**

**基于CCSDS SEDS的业务的描述方法的研究与设计**

**作者姓名： 张梁**

**指导教师： 吕良庆 正高级工程师**

**中国科学院国家空间科学中心**

**学位类别： 电子信息硕士**

**学科专业： 计算机技术**

**培养单位： 中国科学院国家空间科学中心**

**2025年 6 月**

**Research and design of business description method based on CCSDS SEDS**

**A dissertation/thesis submitted to**

**University of Chinese Academy of Sciences**

**in partial fulfillment of the requirement**

**for the degree of**

**Master of Electricity and Information Engineering**

**in Computer Technology**

**By**

**ZHANG Liang**

**Supervisor: LYU Liangqing**

**National Space Science Center ，Chinese Academy of Sciences**

**June 2024**

**中国科学院大学**

**研究生学位论文原创性声明**

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。承诺除文中已经注明引用的内容外，本论文不包含任何其他个人或集体享有著作权的研究成果，未在以往任何学位申请中全部或部分提交。对本论文所涉及的研究工作做出贡献的其他个人或集体，均已在文中以明确方式标明或致谢。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日 期：

**中国科学院大学**

**学位论文使用授权声明**

本人完全了解并同意遵守中国科学院大学有关收集、保存和使用学位论文的规定，即中国科学院大学有权按照学术研究公开原则和保护知识产权的原则， 保留并向国家指定或中国科学院指定机构送交学位论文的电子版和印刷版文件，且电子版与印刷版内容应完全相同，允许该论文被检索、查阅和借阅，公布本学位论文的全部或部分内容，可以采用扫描、影印、缩印等复制手段以及其他法律许可的方式保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名： 导师签名：

日 期： 日 期：

书脊（此页仅用于制作书脊，不用单独打印放入论文）

黑体小四号，论文题目、作者、中国科学院大学

英文和阿拉伯数字用Times New Roman

|  |
| --- |
| 3cm左右  **论文题目**  **李四**  **中国科学院大学**  3cm左右 |

**摘 要**

关键词与摘要间空一行

**关键词：**，，，，

如有直接粘贴引用pdf文档文本内容到word的话会出现很多换行符，这种情况是因为pdf和word软件是属于两种体系，pdf的文本内容每行直接拷贝出来的话会是一行会加换行符号的内容。对于这种情况可以看这篇百度经验[pdf复制到word有空格间隙和换行问题解决办法-百度经验 (baidu.com)](https://jingyan.baidu.com/article/3a2f7c2e646e5867aed61107.html)

这里的宏窗体是[单击键入：关键词1]，如果操作失误没有使得宏窗体消失便输入，直接在宏窗体前面输入内容的话，会默认继承前面的文本"关键词："的格式，因为其加粗了，所以其后输入的内容全部默认加粗。正确点选宏窗体输入便不会出现这种问题。

**Abstract**

关键词与摘要间空一行

**Key Words:**

，，，，

注（注的内容在最后的论文里需要删除）：英文摘要与中文摘要的内容应一致，在语法、用词上应准确无误。关键词间用逗号分隔。如无多个关键字请删除多余的关键字。

如有直接粘贴引用pdf文档文本内容到word的话会出现很多换行符，这种情况是因为pdf和word软件是属于两种体系，pdf的文本内容每行直接拷贝出来的话会是一行加换行符号的内容。对于这种情况可以看这篇百度经验[pdf复制到word有空格间隙和换行问题解决办法-百度经验 (baidu.com)](https://jingyan.baidu.com/article/3a2f7c2e646e5867aed61107.html)

**目 录**

目录

[第1章 绪论 9](#_Toc192160560)

[1.1 研究背景及意义 9](#_Toc192160561)

[1.2 国内外研究现状 12](#_Toc192160562)

[1.2.1 即插即用国内外研究现状 12](#_Toc192160563)

[1.2.2 电子数据单国内外研究现状 14](#_Toc192160564)

[1.3 研究目标与内容 16](#_Toc192160565)

[1.3.1 研究目标 16](#_Toc192160566)

[1.3.2 研究内容 16](#_Toc192160567)

[1.3.3 创新点 17](#_Toc192160568)

[1.3.4 论文组织结构 17](#_Toc192160569)

[1.4 小结 18](#_Toc192160570)

[第2章 S-SEDS建模设计 19](#_Toc192160571)

[2.1 S-SEDS建模设计 19](#_Toc192160572)

[2.1.1 引言 19](#_Toc192160573)

[2.1.2 核心设计理念 19](#_Toc192160574)

[2.1.3 设计架构 19](#_Toc192160575)

[2.2 S-SEDS概念模型 21](#_Toc192160576)

[2.2.1 概念模型介绍 21](#_Toc192160577)

[2.2.2 概念模型组成 21](#_Toc192160578)

[2.3 业务流程与数据流的模型化 23](#_Toc192160579)

[2.3.1 SEDS层次结构介绍 23](#_Toc192160580)

[2.3.2 S-SEDS层次结构介绍 24](#_Toc192160581)

[2.3.3 S-SEDS建模流程 27](#_Toc192160582)

[2.4 业务描述信息的层次结构 29](#_Toc192160583)

[2.4.1 S-SEDS中信息的层次结构 29](#_Toc192160584)

[2.4.2 标签类型设计 32](#_Toc192160585)

[2.5 小结 35](#_Toc192160586)

[第3章 S-SEDS与OWL转换及其应用 36](#_Toc192160587)

[3.1 SEDS 到 OWL 转换需求分析 36](#_Toc192160588)

[3.1.1 SEDS到OWL转换需求分析 36](#_Toc192160589)

[3.1.2 转换面临的挑战 36](#_Toc192160590)

[3.2 SANA提供的XSD与OWL转换方法分析 36](#_Toc192160591)

[3.2.1 XSD文件介绍 37](#_Toc192160592)

[3.2.2 XML文件介绍 37](#_Toc192160593)

[3.2.3 OWL文绍 37](#_Toc192160594)

[3.2.4 soisOWLTools实现 38](#_Toc192160595)

[3.3 S-SEDS到OWL转换设计 38](#_Toc192160596)

[3.3.1 转换目标 38](#_Toc192160597)

[3.3.2 转换策略 39](#_Toc192160598)

[3.4 转换过程实现 41](#_Toc192160599)

[3.4.1 XSD文件解析 41](#_Toc192160600)

[3.4.2 OWL模板生成 42](#_Toc192160601)

[3.4.3 数据绑定与序列化 43](#_Toc192160602)

[3.5 小结 43](#_Toc192160603)

[第4章 S-SEDS工具链的设计与实现 45](#_Toc192160604)

[4.1 应用场景分析 45](#_Toc192160605)

[4.2 S-SEDS模板设计 45](#_Toc192160606)

[4.3 S-SEDS工具链设计 46](#_Toc192160607)

[4.3.1 工具链架构设计 46](#_Toc192160608)

[4.3.2 工具链具体功能设计 47](#_Toc192160609)

[4.3.3 S-SEDS格式转换工具 49](#_Toc192160610)

[4.4 小结 49](#_Toc192160611)

[第5章 S-SEDS在星载业务中的应用与验证 50](#_Toc192160612)

[5.1 星载业务具体指令设计 50](#_Toc192160613)

[5.2 星载业务软件实现 51](#_Toc192160614)

[5.2.1 系统架构设计 51](#_Toc192160615)

[5.2.2 核心模块实现 51](#_Toc192160616)

[5.2.3 核心技术实现 52](#_Toc192160617)

[5.3 实验结果与分析 53](#_Toc192160618)

[5.3.1 事件表业务实例验证 53](#_Toc192160619)

[5.3.2 在线监视业务实例验证 54](#_Toc192160620)

[5.3.3 结果分析 54](#_Toc192160621)

[5.4 小结 54](#_Toc192160622)

[第6章 总结与展望 55](#_Toc192160623)

[6.1 本文工作总结 55](#_Toc192160624)

[6.2 未来工作展望 55](#_Toc192160625)

[参考文献 57](#_Toc192160626)

[附录 59](#_Toc192160627)

[附录一 61](#_Toc192160628)

[附录二 63](#_Toc192160629)

[致 谢 65](#_Toc192160630)

[作者简历及攻读学位期间发表的学术论文与其他相关学术成果 67](#_Toc192160631)

**图目录**

[图3‑1 点选插入题注 5](#_Toc105403921)

[图3‑2 新建标签 6](#_Toc105403922)

[图3‑3 新建图标签 6](#_Toc105403923)

[图3‑4 设置图标签格式 6](#_Toc105403924)

[图3‑5 7](#_Toc105403925)

[图3‑6 图修改位置 7](#_Toc105403926)

[图3‑7 点选插入新图序 8](#_Toc105403927)

[图3‑8 添加新图序 8](#_Toc105403928)

[图3‑9 删除多余空格 9](#_Toc105403929)

[图3‑10 表修改位置 10](#_Toc105403930)

[图3‑11 点选插入新表序 10](#_Toc105403931)

[图3‑12 添加新表序 11](#_Toc105403932)

[图3‑13 错误序号的二级标题 13](#_Toc105403933)

[图3‑14 重新调整编号 14](#_Toc105403934)

**和上面的附录、附录一和附录二一样，图序和图题会空一个汉字符位（2空格），标准的只需要一个空格，所以需要手动删除一个空格。**

**表目录**

[表3‑1 9](#_Toc105402813)

[表3‑2 11](#_Toc105402814)

[表3‑3 续表表格示例 11](#_Toc105402815)

[表3‑4 表格示例 12](#_Toc105402816)

[表3‑5 表格示例1 12](#_Toc105402817)

[表3‑6 表格示例2 12](#_Toc105402818)

**和上面的附录、附录一和附录二一样，表序和表题会空一个汉字符位（2空格），标准的只需要一个空格，所以需要手动删除一个空格。**

# 绪论

## 研究背景及意义

随着航天任务类型多样化、需求复杂化，空间系统结构日趋复杂，模块化设计是解决这一问题的有效手段。模块化设计可使各子系统在遵循统一接口规范的前提下独立、并行开发与测试，提高研发效率[21]。采用多模块设计可以满足航天器在轨长期运行和动态升级的需要[19-20]。但是航天器一般是定制化设计的，部件集成需要在约定好通信协议的情况下，通过特定接口来连接，形成比较紧密的耦合关系。这种做法对于系统开发而言，接口关系复杂，任务量繁重，数据通用性差，系统可靠性难以保证，不利于实现快速装配和灵活更换。为此，在航天器设计中引入即插即用（Plug-and-Play， PnP）技术是一个值得研究的方向。即插即用技术利用标准化接口和通信协议，采用模型化加数据化的开发方式，实现部件的快速识别、配置和互联，有助于缩短系统集成和测试时间 [22]，降低因定制化设计所带来的系统维护复杂性和研制成本，支持系统的灵活扩展和升级 [23]。同时航天器因航天任务需求的不同，所需的设备和业务也不尽相同。而且在发展航天器智能能力的背景下，对发展星载即插即用的能力也提出了需求[3]。

为此空间数据系统咨询委员会（Consultative Committee for Space Data Systems，CCSDS）的航天器接口业务（Space Onboard Interface Services，SOIS）领域提出了SOIS架构，规定了星载信息系统的层次划分和标准功能业务，并引入了电子数据单（Electronic Data Sheet，EDS）技术，规定了SOIS架构的EDS标准（SEDS）。SEDS旨在描述航天器设备及其接口关系，采用XML语言来描述设备的协议、数据、通信接口及功能，可用于星载软件的开发、测试用例的生成和文档的自动化过程[1]。

目前SEDS主要是面向设备的，但它也可以用于描述业务。本文在SEDS基础上提出了一种星载业务的接口信息及其关系的标准化描述方法。同时，为了支持业务数据的描述和处理，设计开发了相应的电子数据单工具链，用于对SEDS的设计、修改、生成、转换等操作，为电子数据单的开发提供方便。为了便于文件数据的共享，提出一种模板文件到本体的映射规则方法。

## 国内外研究现状

### 即插即用技术

#### 概述

即插即用是指当一个设备接入系统时，系统在运行过程中能动态地进行检测和配置的能力[2-3]。

即插即用技术在地面应用主要局限于设备级，而在航天器中的应用则有所扩展，不仅涉及设备级的识别，还涵盖功能与服务的识别。这一变化的主要原因在于，地面系统在完成设备识别与连接后，其应用由人为决策，而航天器上层应用的智能化水平较低，设备接入系统后，其应用对象通常是更高层次的系统。因此，为了有效发挥即插即用技术的作用，还需进一步解决业务层面的即插即用问题。

即插即用技术在航天器中的主要应用场景包括：地面系统集成测试阶段的网络快速配置与设备即插即用，在轨飞行任务期间各组件的分离与对接。此外，该技术还能支持应用层软件的可重用性与可配置性。

#### 即插即用研究现状

美国空军研究实验室（Air Force Research Laboratory，简称 AFRL）自2004年起开展空间即插即用技术研究，并于同年年底获美国航空宇航学会（AIAA）批准，成立工作组专门研究“空间飞行器即插即用电子设备”（SPA）相关标准。并于2005年12月在快速反应空间实验平台上首次验证了模块化、即插即用卫星体系架构的可行性。

2007年，AFRL启动首颗完全采用SPA架构的卫星PnPSat-1的研制，旨在通过标准化接口减少重复操作，缩短研制周期。2009年，AFRL进一步开展PnPSat-2的研制，以深化SPA技术研究。2011年，AIAA对SPA标准进行修订，将其全称调整为空间即插即用电子体系，并于2013年进一步完善标准，提高SPA技术性能。

根据AIAA官网，2013版SPA标准包含八个子标准：Networking、Logical Interface、Physical Interface、28V Power Service、System Timing、Ontology、Test Bypass和Spacewire Subnet Adaptation，以及两本指导手册——Standards Development Guidebook和System Capability。未来，SPA标准将继续发展，光纤与无线网络技术将纳入其中，形成SPA-O（Optical）和SPA-W（Wireless）子网规范。

SPA的分层结构见表1-1。

表 1‑1 SPA分层结构

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 任务层 | 遥感系统 | | | | | 通信系统 | | 导航系统 | |
| 功能层 | 任务管理 | | | | | | | | |
| 载荷管理 | | | | | | 飞行器管理 | | |
| 应用层 | 任务请求产生 | | 调度 | 命令 | | 监控 | 姿态控制 | 轨道控制 | 其他 |
| 中级别服务层 | 飞行动力学 | | 载荷控制 | 空间链路节点控制 | | 飞行器系统控制 | 任务与调度 | 软件和配置管理 | 其他 |
| 低级别服务层 | 卫星数据管理（SDM） | | | | | |  | | |
| 服务器 | 操作系统（可选） | | | | | | | | |
| 协议层 | xTEDS | | | | | | | | |
| XML | | | | CCSDS | | | TCP | |
| SPA-U（USB） | | | SPA-S（即插即用SPaceware） | | | SPA-E（Ethernet） | SPA-X（其他） | |
| 设备层 | ASIM | | | 智能传感器 | | 智能执行器 | 内存 | 连接器 | 处理器 |
| 传感器 | 执行器 | |

从SPA的分层结构来看，其核心为卫星数据管理（Satellite Data Manager, SDM）单元。在协议层，SPA采用可扩展标记语言（XML），并遵循CCSDS协议和TCP协议，以定义航天器总线及有效载荷的快速开发接口驱动标准。这些接口标准基于计算机工业广泛应用的规范，包括面向低速数据传输、基于USB1.1标准的SPA-U接口，面向高速数据传输、基于SpaceWire标准的SPA-S接口，以及Ethernet等子网协议，并具备良好的扩展性。

在设备层，传感器和执行器通过应用传感器接口模块（Applique Sensor Interface Module, ASIM）接入系统，而具备智能功能的传感器与执行器可直接连入。在上层，SDM支持平台化的服务层与应用层功能，同时涵盖功能层和任务层的构建。SPA的层次结构全面覆盖了应用层及以上的层级划分与内容。从星地服务的对等架构（CCSDS MOIMS领域）角度来看，SPA可视为其中的一种架构实例。

南京航空航天大学王明东团队基于FPGA主控芯片的片上系统，实现了嵌入式传感器接口模块（ASIM），使传感器采集的数据经FPGA处理后，通过SPA-U接口传输至星载计算机。杜娟团队对ASIM的工作原理进行了分析，并完成了其硬件与软件的设计。

清华大学尤政院士团队研究了即插即用技术在微小卫星综合电子系统中的应用，并提出了六项关键技术。基于微小卫星常用的UART总线，团队对其进行了即插即用功能改造，并设计并实现了多UART接口的即插即用系统。该系统的即插即用服务由即插即用核心系统负责，主要承担端口检测、设备初始化和资源分配等管理任务。

复旦大学游红俊团队针对航天器在轨服务需求，研究了即插即用技术的应用，并提出了一种可扩展的航天器电子系统架构模型。该模型采用模块化设计，通过下位机装配信息表实现电源配置与管理，并利用EDS和下位机任务桩对可更换模块进行识别与动态维护管理。

国防科技大学研究团队围绕SPA软件体系开展了一系列研究：张元昭[38]通过解构星载数据管理器（SDM）核心架构，系统剖析其容错机制、运行模式及通信模型，创新性地实现了符合空间数据系统咨询委员会（CCSDS）星上接口服务（SOIS）协议的命令数据获取、时间访问、消息传输和文件管理等四项基础服务模块，并以动态链接库形式完成Windows系统部署，同时验证了其在Linux环境的跨平台移植能力，李轩[39]提出基于硬件抽象层的一体化卫星软件架构，构建了脱离硬件依赖的仿真开发环境，为星载软件的全周期开发提供方法论支撑，李彦琛[40]则针对新一代SPA标准，研制了以SpaceWire总线为载体的应用层软件系统，实现了航天器内部设备的高效互联。

哈尔滨工业大学的鲁文帅[41]聚焦SpaceWire总线协议，提出SpW-PnP分层协议栈模型，为航天器内部高速数据网络的即插即用提供理论框架，闫瑞东[42]借鉴早期SPA中间件设计理念，创新性构建基于软总线的卫星姿态控制系统仿真平台，通过软件复用技术实现了控制算法的模块化集成，邹吉炜[43]系统解构SPA三大核心技术——即插即用接口、部件自描述和软件架构，搭建功能验证平台并完成核心技术的工程化演示，为航天器快速集成提供实践范例。

以上对于航天器的即插即用主要是在设备层面，除了对物理设备的信息识别，还需要考虑功能业务层面的识别，因此星载应用还需要解决星载系统对业务、功能层面的信息识别与配置问题，电子数据单技术是实现即插即用的业务描述的主要内容之一。

### 电子数据单技术

#### 概述

电子数据单（Electronic Data Sheet， EDS）是一种用于描述系统中部件实体（如设备、业务等）的信息载体，主要解决信息传递过程中的部件信息识别和匹配问题，支持即插即用技术[1]。EDS起到数据隔离的作用，使外部系统无需关心部件内部的具体实现和内部逻辑，就可对该部件进行识别、配置和更改。

EDS的概念最早可追溯至美国国家标准技术研究所（NIST）和电气与电子工程师协会（IEEE）提出的IEEE 1451智能传感器协议族[2]。目前，该技术已广泛应用于互联网通信及传感器领域，并逐步推广至航天领域。例如，空间数据系统咨询委员会（CCSDS）提出了基于可扩展标记语言的遥测遥控交换标准（XTCE）、航天器接口业务领域（SOIS）的EDS（SEDS）标准，以及开放档案信息系统（OAIS）中用于信息对象标识的XML格式化数据单元（XFDU）标准[3-6]。

#### IEEE1451标准

美国国家标准技术研究所（NIST）、电气与电子工程师协会（IEEE）及仪器与测量协会传感技术委员会自1993年起联合制定智能传感器通用通信接口标准，并于1997年陆续发布IEEE 1451智能传感器协议族。该协议族包括适用于不同应用场景的变换器电子数据单（TEDS），并已广泛应用于地面系统和网络系统[4-5]。

IEEE1451制定了一系列子标准，用于将智能传感器连接到网络系统，分为软件接口和硬件接口两部分。软件接口包括1451.0和1451.1，采用面向对象模型，定义了一套软件接口规范、TEDS格式和通信协议，使各种标准或非标准的变换器能够顺利接入不同网络。硬件部分则定义了针对性的模型、协议和TEDS的具体应用。内容简介请参见表1-2。

表 1‑2 IEEE1451标准协议族的内容

|  |  |
| --- | --- |
| 1451标准号 | 名称 |
| 1451.0-2007 | 常规功能、通信协议和TEDS格式 |
| 1451.1-1999 | 网络能力应用处理器接口模型 |
| 1451.2-1997 | 变换器与微处理器之间的通信协议和TEDS格式 |
| 1451.3-2003 | 分布式多路系统的数字通信和TEDS格式 |
| 1451.4-2004 | 混合模式的通信协议和TEDS格式 |
| 1451.5-2007 | 无线通信协议和TEDS格式 |
| 1451.6 | 为本质安全性应用构建的CANopen变换器网络接口 |
| 1451.7-2010 | 射频识别系统通信协议和TEDS格式 |

TEDS是整个协议族的核心内容，用来描述各种转换器的接口信息。IEEE1451标准协议族规定的多种类型TEDS有不同的适用性，通过接口标准化和模块化进行TEDS的交换，自动识别和自动适配接入的传感器和执行器，用于实现即插即用。

#### SEDS标准

CCSDS在2013年提出了航天器接口业务（Spacecraft Onboard Interface Service，SOIS）领域的SOIS架构[44]，该架构目前仍在持续发展和演进中。CCSDS SOIS为解决EDS建立和使用问题，采用XML制定了SOIS EDS（SEDS）的XML规格[45]和术语字典（Dictionary of Term，DoT）规格标准草案[46]，并于2019年提出了SEDS标准建议书[47]，用以描述星载设备及其接口关系。

NASA和ESA在国际上率先应用SEDS。NASA的戈达德飞行中心在其cFE核心飞行软件架构中实现了SEDS，并通过JAVA开发工具支持SEDS的生成和解析。NASA的cFE核心软件系统已应用于多个型号，基于软件总线进行消息通信，开发了配套的组件配置工具，源代码公开，适用于航天器和无人机等系统。NASA计划将该软件架构参考SOIS体系架构定义的接口，进一步完善以适应SOIS体系架构。在cFE核心飞行软件架构中，SEDS技术用于自动生成软件配置信息，便于软件的按需配置和组装。SEDS不仅能定义设备的接口，还能定义所有软件构件的接口。工具通过扫描代码头文件生成SEDS，修改SEDS后可再生成新代码，同时生成测试用遥控指令并解析遥测数据。NASA开发了一款工具，将软构件SEDS和任务配置文件作为输入，生成C语言头文件，包含消息定义和工程单元转换。该工具集成在cFS创建系统中，被多个NASA中心使用。

欧洲的SCISYS公司长期研究EDS，开发了支持SEDS生成和解析的工具。该工具采用JAVA开发，可根据SEDS自动生成星载软件代码。SEDS生成及使用过程包括：初期根据设备参数生成SEDS文件，通过工具生成参数和验证文档，生成的SEDS文件可生成星载部分构件如设备驱动，还可生成仿真输入。在项目开发过程中，新增设备或参数改变时可直接更新SEDS，系统模型或数据变化也可生成SEDS，修改后的SEDS文件通过工具自动生成并更新文档。在综合测试时，EDS直接作为输入，省去文档，因为SEDS中包含所有文档数据。

国内对于电子数据单的研究包括复旦大学提出的系统架构模型，北京空间飞行器总体设计部提出的SEDS设计实例，还有中国科学院国家空间科学中心设计的电子数据单工具链。

复旦大学提出了一种可扩展的航天器电子系统架构模型（SCMSES），利用电子数据表（EDS）和下位机任务桩来实现航天器下位机的动态维护和管理。通过这种架构，可以在不修改航天器软件系统的前提下，实现功能模块和下位机的在轨更换与扩展。EDS用于标识和描述下位机的详细信息，而下位机任务桩则用于实时监控和管理下位机的状态[33]。

北京空间飞行器总体设计部基于 SOIS 架构对 SEDS 的概念、应用场景做了说明，对星载综合电子系统研制过程做出了顶层设计，使用SEDS定义和描述设备及服务接口，提出SEDS到XTCE的转换[4]。

中国科学院国家空间科学中心出 SEDS 解析器的设计思路，给出了解析器的应用场景，并以 1553B 数据总线的通信协议为例做了验证，并以数据注入业务为例给出了相应的建模实例化工具，还给出了描述星载业务的电子数据单S-SEDS的设计思路[7]。

### 包应用标准

#### 概述

包应用标准（Packet Utilization Standard，PUS）是由欧洲空间局（European Space Agency，ESA）操作和地面支持系统标准委员会（Committee for Operations and EGSE Standards，COES）负责开发并于1994年推出的。PUS标准的主要目的是规范航天器与地面系统之间的数据通信，它提出了地面操作与航天器交互的标准化业务模型，定义了每项业务的请求和报告内容。

#### 包应用标准简介

欧洲空间标准化合作组织（European Cooperation for Space Standardization， ECSS）于 2003 年发布了欧洲空间局的包应用标准（Packet Utilization Standard， PUS）A 版本，该版本针对航天器应用方面的需求制定了标准业务模型的集合，对每个业务都进行了操作内容、业务部署以及相关的请求、报告规定。经过十余年的实践积累与改进，对 A 版本规定的星载业务规格及其请求、报告格式进行了梳理，于 2016 年提出了 C 版本[24-25]。PUS 定义的标准业务模型适用于不同任务和不同层次的服务能力，在实际使用时又可以根据任务需求、适用性进行个性化的部署和剪裁。

PUS 中的请求概念对应了业务的遥控数据，报告概念对应了业务的遥测数据。它规定了各项业务的设计要求，不同的型号任务可以根据需要对内部的数据结构和处理流程进行灵活的本地化设计。

PUS定义了20个标准业务模型（standardized service type model），提供一致的模型规格定义，并允许任务定义特定的业务模型（mission-specific service type model）。每个业务模型包括请求、报告和业务规格三部分内容，可在具体任务中进行剪裁和扩充。PUS标准业务模型见表1-3。

表 1‑3 PUS标准业务模型

|  |  |  |
| --- | --- | --- |
| 业务类型编号 | 业务类型名称 | 功能描述 |
| ST[01] | 请求验证 | 验证请求的接收完整性、目标应用进程和服务的可用性，生成执行验证报告。 |
| ST[02] | 设备访问 | 向设备发送命令，从设备获取数据。 |
| ST[03] | 常规遥测 | 生成和管理常规遥测数据，用于监控航天器状态。 |
| ST[04] | 参量统计报告 | 计算并报告参数的统计信息（如最大值、最小值、平均值等）。 |
| ST[05] | 事件报告 | 报告航天器上发生的事件（如故障、异常等）。 |
| ST[06] | 存储器管理 | 管理内存内容，包括加载、转储和检查内存。 |
| ST[07] | / | 保留 |
| ST[08] | 功能管理 | 执行特定功能（如模式切换、子系统控制等）。 |
| ST[09] | 时间管理 | 生成和报告航天器时间，支持与地面时间同步。 |
| ST[10] | / | 保留 |
| ST[11] | 基时调度 | 根据预设时间表释放请求，支持任务自动化执行。 |
| ST[12] | 星上监控 | 监控航天器参数，当参数超出预设范围时触发事件。 |
| ST[13] | 大数据包传输 | 支持大容量数据传输（如文件传输）。 |
| ST[14] | 实时转发控制 | 控制数据的实时转发，用于实时通信。 |
| ST[15] | 星上存储与检索 | 管理航天器上的数据存储和检索。 |
| ST[16] | / | 保留 |
| ST[17] | 测试 | 支持航天器系统的测试功能。 |
| ST[18] | 星上控制程序 | 执行航天器控制程序（OBCP）。 |
| ST[19] | 事件操作 | 根据事件触发预设动作。 |
| ST[20] | 参量管理 | 管理航天器参数的定义和配置。 |
| ST[21] | 请求序列 | 管理请求的序列化，支持复杂任务调度 |
| ST[22] | 基于（轨道）位置调度 | 根据航天器位置信息释放请求。 |
| ST[23] | 文件管理 | 管理航天器上的文件系统，支持文件的创建、读取、写入和删除。 |

每个业务模型都有各自的数据结构组织，借用管理信息库 (Management Information Base，MIB)的概念，业务模型内部数据信息通过 MIB 进行管理，同时用 EDS 描述包括外部请求和报告在内的业务接口信息，建立业务模型的请求数据库和报告数据库，开发用于配置的自动化设计工具[3]。这种MIB+EDS 的模式可以用于简单到复杂系统的业务设计。

### 网络本体语言

#### 概述

网络本体语言（Web Ontology Language，OWL）是一种基于语义网的标准化本体描述语言，旨在增强机器对网络内容的理解和处理能力。[]。XML被认为是Web上进行数据交换的标准语言，研究人员基于XML语法开发了多种本体描述语言，包括SHOE（Simple HTML Ontology Extensions，简单HTML本体扩展）、XOL（XML-Based Ontology Exchange Language，基于XML的本体交换语言）、OML（Ontology Markup Language，本体标记语言）、DAML-ONT（DARPA Agent Markup Language Ontology，DARPA智能代理标记语言本体）、OIL（(Ontology Inference Layer，本体推理层）、DAML（DARPA Agent Markup Language，DARPA智能代理标记语言）+OIL等，最终形成了OWL[]。OWL的演化过程如图1-1。

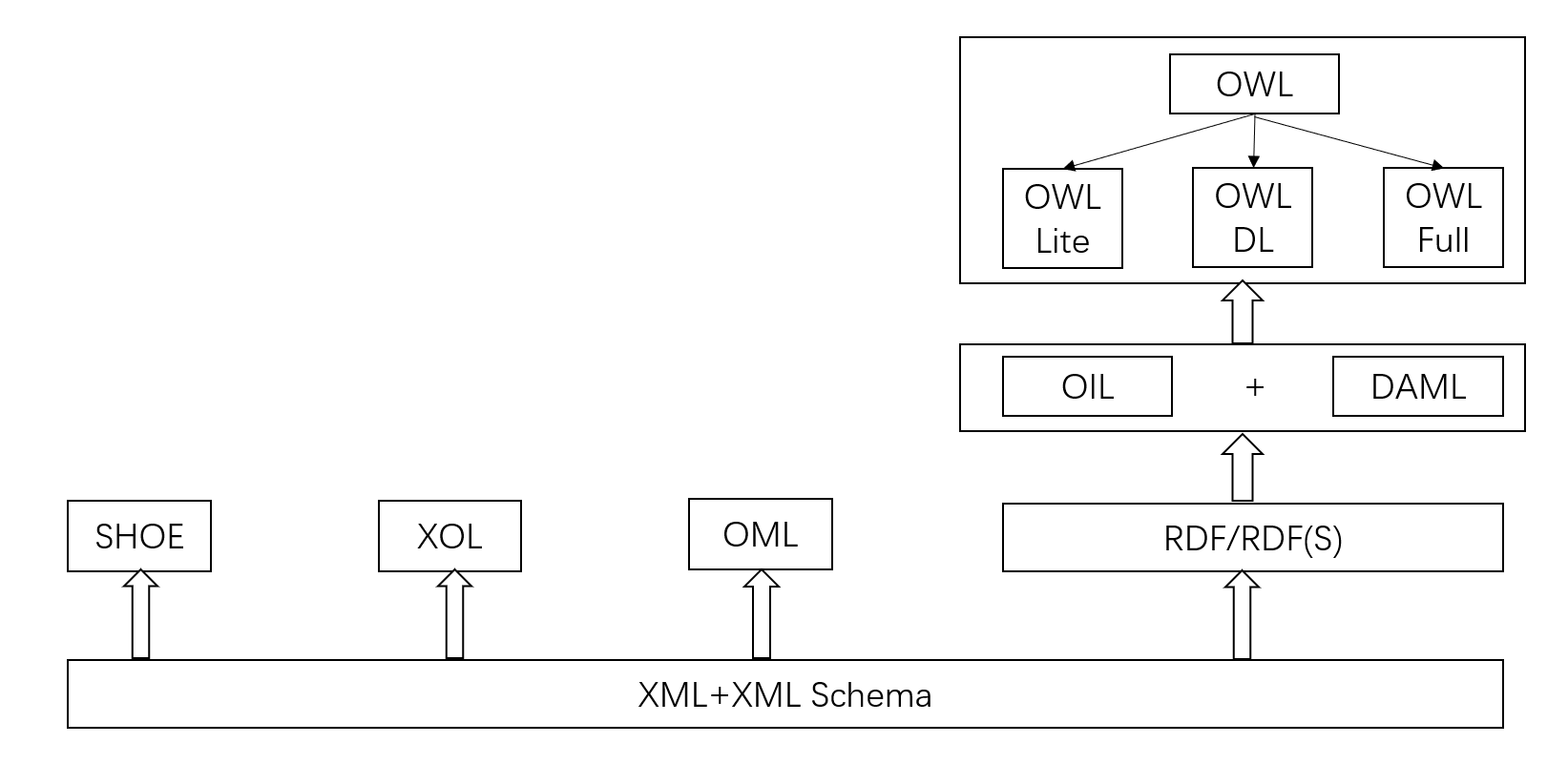


图 1‑1 基于XML的本体语言演化过程

RDF 是 W3C 提出的语义网核心标准，旨在通过结构化方式描述网络资源的语义关系。其核心模型基于三元组（主体-谓词-客体），数据以有向图形式组织，节点表示资源或字面量，边表示属性关系，支持嵌套和链式表达。[]RDF用于描述对象（“资源”）和它们之间的的关系，为这些数据模型提供了一个简单的语义，这些数据模型可以用XML语法表示。

RDF Schema（RDFS）是对 RDF 的语义扩展，用于定义领域词汇和逻辑规则。RDFS 是一个用于描述RDF资源的属性和类的词汇表，其语义用于描述此类属性和类的泛化层次结构，RDF和RDFS共同构成语义网的核心框架[]。

DAML+OIL是早期语义网标准化的核心本体语言之一，由美国国防高级研究计划局（DARPA）主导开发。其设计目标是通过形式化的知识表示框架，解决传统Web数据在语义层面的异构性与互操作性问题。

OWL是借鉴DAML+OIL的设计思想和经验设计的，它包含OWL Lite、OWL DL和OWL Full三种子语言， 分别适用于不同的用户需求。OWL提供了丰富的词汇和形式语义，支持复杂的逻辑推理和语义关系，可以用于构建领域本体，支持知识共享和复用[]。

#### XML和XML Schema

为了克服超文本标记语言（HTML）在大规模万维网（Web）应用中的扩展性不足，万维网联盟（W3C）于1998年2月基于标准通用标记语言（SGML）和HTML提出了XML1.0版本，作为推荐的标记标准语言。 XML在语言功能上属于标记语言，通过元素标签标记任意类型数据，并明确表达数据含义，其特点是采用树状结构设计元素标签，并规定了描述数据时的结构化原则。

XML可分为面向数据（data-oriented）的XML和面向文档（document-oriented）的XML，提供了结构化文档的表层语法，但没有施加语义约束。 XML语言有两种定义文档结构的方法：文档类型定义（DTD）和XML Schema。DTD最初作为XML标准的一部分提出，主要用于定义标记符号的语法规范，包括标记符号的类型、属性和顺序关系等[59]。在实际应用中，由于DTD在自动化信息交换和文档结构规范描述方面存在问题，W3C于2001年5月提出了XML Schema作为定义XML文档结构的标准模式[60]。XML结构定义（XSD）用于描述XML文档中元素节点的结构，保存为XSD格式的文件，可作为XML文档的模板，

XML Schema是一种用于描述 XML 文档结构与内容的语言，使用数据类型扩展了XML，通过定义元素、属性、数据类型及约束规则，实现文档的严格模式验证。它是由W3C定义用于描述 XML 文档的结构、数据类型及语义约束，其核心目标是为 XML 数据提供强类型化验证与结构化建模能力。

XML Schema 与 XML 共同用于协作完成具有一定语义表达能力的结构化XML文档。XML规范描述底层语法的规则，不涉及语义内涵。XML Schema规范描述上层语义的规则，明确元素、属性及其数据类型的语义角色，并规范其逻辑关系，它描述的是一种语义结构。

#### OWL

OWL是一种基于DAML+OIL的语言，相较于其前身本体语言，增加了更多的词汇来描述属性和类，可以清晰表达词汇的含义及其相互关系。W3C于2002年7月发布了OWL的第一个版本，2003年8月成为W3C候选推荐标准，2004年2月10日正式成为推荐标准。OWL根据不同应用需求，提供了三种表达能力递增的子语言：

1）OWL Lite：适用于需要简单分类层次和基础约束的用户。例如，它支持基数约束，但仅允许基数为0或1。

2）OWL DL：适用于需要最强表达能力且要求保持计算完备性和可判定性的用户。计算完备性是指所有结论都能被计算出来，而可判定性是指所有计算在有限时间内完成。OWL DL包含OWL语言的所有成分，但使用时必须遵循一定约束。

3）OWL Full：适用于不需要计算保证但需要最强表达能力和完全自由的RDF语法的用户。例如，在OWL Full中，一个类可以同时被视为个体集合和个体本身，并允许扩展预定义词汇表的含义。

在表达和推理能力上，每个子语言都是对前者的扩展。这三种子语言的关系如下：

1）每个合法的OWL Lite本体都是一个合法的OWL DL本体。

2）每个合法的OWL DL本体都是一个合法的OWL Full本体。

3）每个有效的OWL Lite结论都是一个有效的OWL DL结论。

4）每个有效的OWL DL结论都是一个有效的OWL Full结论。

### 小结

综合上述对国内外研究现状的介绍，设备即插即用过程是通过自动化识别、协议适配与动态配置实现设备无缝接入系统的过程。当新设备接入时，系统在链路层通过轮询发现设备并分配地址，设备向系统传递EDS配置信息。系统解析收到的EDS，将信息存入MIB，并根据应用需求与设备协商可配置参数（如性能指标），通过多轮交互完成动态配置。运行中系统可依据外部指令或环境变化重新调整配置。设备移除时，系统可通过多次轮询确认后释放MIB资源以避免冗余，以便后续接入时重新使用。整个过程依托EDS的自描述性、标准协议框架和分层协商机制，实现端到端交互的灵活接入，兼容未知设备的接入，方便系统资源的动态管理。

EDS在即插即用过程中作为设备和系统间的“数字身份证”，通过标准化元数据描述实现设备自描述能力与系统配置的解耦。主要体现在三个方面：首先，EDS可通过结构化语言对设备接口协议、功能参数、通信规范等信息进行机器可读的封装，使用时通过设备与接入系统之间的自动交互，使得不再需要工程师预先掌握和配置设备的详细参数或协议细节。其次，系统基于EDS构建MIB，通过EDS模板，将设备参数与配置逻辑分层，通过解析EDS、存储参数、匹配模板、生成文件的流程实现设备的自动化集成。最后，EDS的使用使得航天器在不改变星载软件的情况下能够进行灵活的功能配置和修改，提升其适应性。

通过EDS对业务进行标准化描述（例如PUS的20项标准业务），可以结构化定义业务功能、输入输出参数、绑定协议，实现业务逻辑与底层硬件的解耦。EDS描述的业务信息通过协议规则屏蔽通信接口的差异，使得工具相应的EDS工具链能够对业务进行建模、生成和转换，支持业务的即插即用。

航天任务中有不同业务，而同一业务也会产生不同的EDS文件，从而产生大量的文件，这些数据分散在不同过程的文档中，不方便管理和复用。将业务EDS结构转换为OWL语义本体，采用OWL类-属性体系对EDS元素进行形式化定义，通过建立领域知识库，表达业务之间的依赖关系，支持对于数据文件的管理和知识信息的共享。

因此本文要解决业务的即插即用问题就需要先解决业务的EDS描述问题。

1）星载业务描述的标准化需求。

目前，遥控、遥测、星载设备等部件的信息描述已得到一定程度的标准化（如XTCE标准、XML格式数据化单元（XFDU）标准、SEDS标准等数据标准化模型）支持，各航天机构也开展了相关软件研制方法的研究。但随着航天器自主能力的提升，星载业务（如设备管理、参数监视、事件报告等）的描述需求日益增加。为此，基于SEDS标准进行扩展，增加业务特征的信息，以支持对星载业务的描述。

2）SEDS文件自动生成和管理需求。

SEDS技术在使用中会产生大量模板和数据文件，显然，手工设计和管理方式难以支持对数据描述的编辑、生成、转换等需要，也不利于数据信息的复用。为此，需要开发SEDS辅助工具实现自动化生成与管理。

3）建立领域知识库需求。

引入SEDS到OWL的映射规则，提升SEDS标准的语义表达能力和互操作性，便于领域信息的管理和知识共享。

## 研究目标与内容

### 研究目标

在SEDS基础上，提出一种星载业务的接口信息及其关系的标准化描述方法，同时基于此描述方法提出一种XSD向OWL信息转换的映射规则，并设计了模板生成工具、文件生成工具和格式转换工具，供用户实现对星载业务的设计使用。具体目标有：

1. 研究SEDS标准如何描述设备，以及如何进一步描述业务。
2. 研究SEDS向OWL的转换，主要是映射规则和内容解析。

（3） 设计SEDS工具链，支持SEDS的编辑、生成和转换。

### 研究内容

本研究首先分析了SEDS标准，深入探讨了它如何描述航天器的设备接口和数据一致性。随后提出S-SEDS的扩展设计来支持更加复杂的航天器业务描述。

通过设计基于OWL的本体模型，提出了一种从SEDS到OWL的映射规则，该规则能够将航天器任务数据通过语义化描述进行标准化处理。通过构建对应的本体，为航天器不同任务之间的数据互操作性和知识共享提供了有效的支持。

设计并实现三个核心工具：模板设计工具、文件生成工具和格式转换工具。模板设计工具支持根据业务需求生成SEDS标准的模板，文件生成工具通过数据填充生成符合SEDS标准的XML文件，而格式转换工具则将SEDS数据模型转换为其他格式文件，包括OWL本体格式，支持领域知识构建和知识共享。

以星载事件表业务和在线监视业务为例，对工具链开展了实例验证，证明了其可行性和有效性。本研究提出的星载业务描述方法及其工具链实现为业务数据的标准化描述和处理提供了支持，有助于提升星载信息系统的即插即用能力，便于不同系统的知识共享。

本文的主要贡献：

1）提出了基于SEDS的描述星载业务的S-SEDS模型：

S-SEDS模型继承了SEDS标准在航天器任务功能和数据结构描述方面的特点，将描述范围从设备扩展至功能业务和应用，支持子业务模块化与通用数据类型复用，并提出了业务建模的具体步骤。

2）提出了基于S-SEDS向OWL转换的映射规则：

提出S-SEDS向OWL转换的映射规则，将星载业务数据进行标准语义化处理，给出具体解析转换过程。

3）设计实现S-SEDS工具链：

提出S-SEDS工具链方案，设计实现了模板设计、文件生成和格式转换工具。

### 论文组织结构

本文主要分为六章。第一章介绍了研究背景、国内外研究现状以及研究目标和内容，第二章研究了SEDS的建模设计，给出基于CCSDS SEDS的业务描述方法，第三章探讨了S-SEDS与OWL之间的映射规则，提出了相应的转换需求和方案，第四章则设计并实现了支持S-SEDS编辑、转换和解析的工具链，第五章通过系统功能验证和实际应用案例，展示了工具链的效果与应用价值，第六章总结了研究成果，并展望了未来的研究方向。

本文研究思路见图1-1。

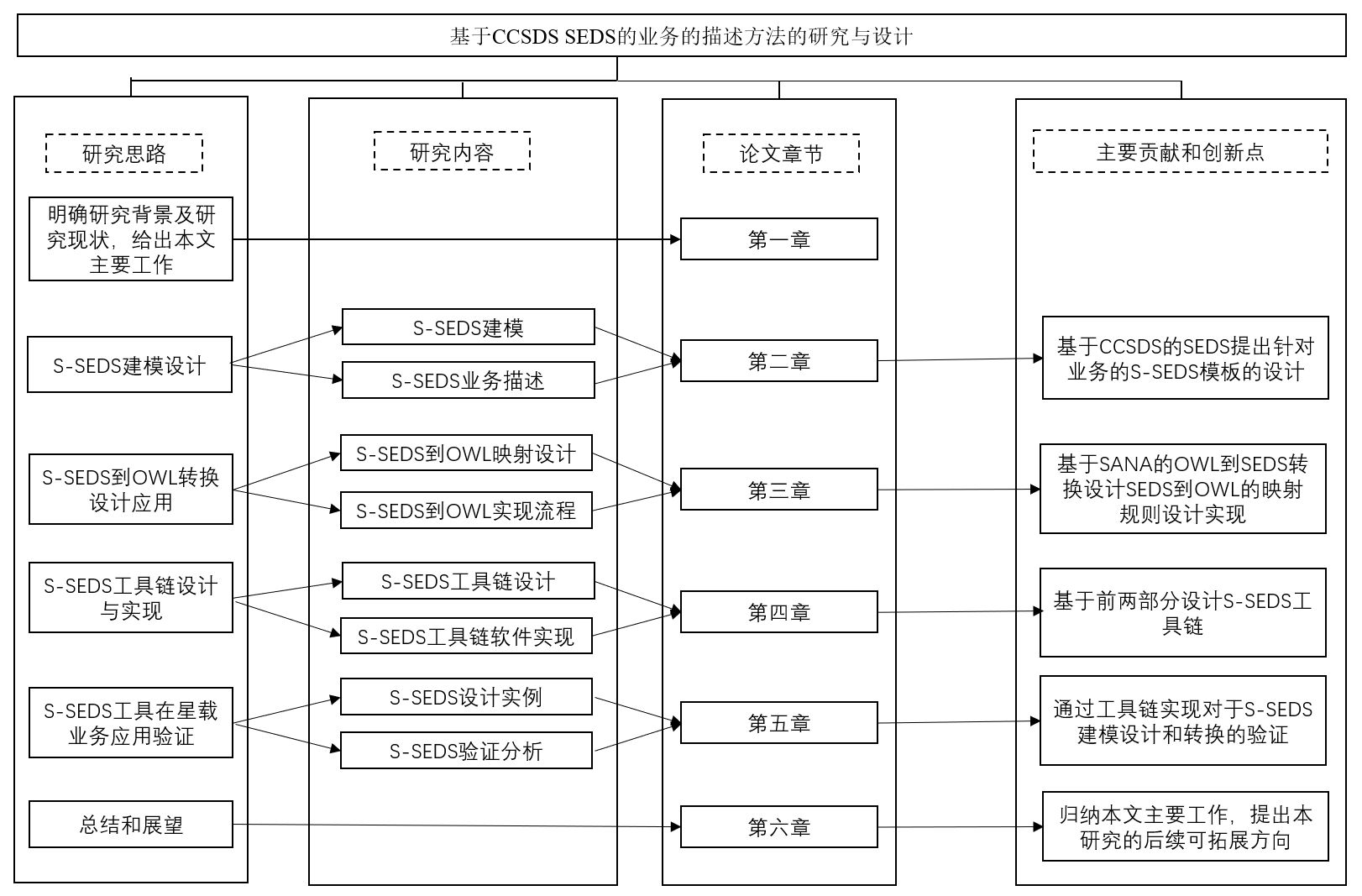


图 1‑2 研究路线图

## 本章小结

本章主要介绍了课题的背景和研究意义，从PnP技术、EDS技术、PUS和OWL四个方面梳理了国内外研究现状。

在PnP技术方面，以AFRL提出的SPA架构为例，剖析了SPA架构的分层设计及其在SDM中的作用，对比了国内外研究团队在接口模块、总线协议及仿真平台开发方面的实践进展。

在EDS技术方面，介绍了CCSDS提出的SEDS标准及其在NASA、ESA的应用实践，还介绍了国内对于SEDS工具链的开发。

在PUS方面，介绍了ESA的PUS演进过程，对20类标准业务通过EDS与MIB实现业务逻辑与硬件解耦。

在OWL方面，探讨OWL的演化路径及其在知识共享的作用。

最后梳理了课题的研究目标和研究内容，介绍了本文的编写逻辑和组成结构。

# SEDS建模设计

## 信息视角建模分析

空间数据系统可以从不同视角进行描述。在空间数据系统参考架构（RASDS）[61]中，定义了五种基本视角：企业视角、功能视角、连接视角、通信视角和信息视角。除企业视角外，功能视角描述系统应具备的功能，连接视角描述系统的组成和连接关系，通信视角描述系统各部分之间的通信协议关系，信息视角描述通信交换数据的语义、语法、规则和约束等。这些视角对同一对象的描述应保持一致性和相关的逻辑关系。

信息视角主要关注信息对象和数据对象及其定义和配置的描述，包括元素、结构（语法）、语义、相互关系、使用约束、传输规则、存储规则以及访问策略等。信息对象在功能对象之间传递，由信息基础设施管理（如存储、定位、访问和发布）。一个信息对象对应一组连接对象，其静态模板对应连接对象的可能状态，每个状态变化对应连接对象之间的行为、状态和环境互动关系，或连接对象的内部动作。

信息视角的核心元素是数据对象及其逻辑连接和关系。数据对象可以由元数据和其他数据对象构成。元数据是描述数据的数据，用于定义语法（结构）、语义、关系、约束、规则和策略等。数据对象内容可以包含自描述信息，能够被人和机器访问。系统中使用的数据对象需有详尽、严格的描述，通常表现为数据模型或数据模板。

信息对象没有输入输出接口，而是通过模板来描述其结构、使用和传输规则，以及访问和保持策略。信息对象由一组抽象的数据元素构成，使用特定的语言和框架进行定义。管理接口主要关注规则、模板、元素类型、约束和持久性等，核心能力包括结构、语义、关系、类型转换和约束检查等。

在描述信息对象时，通过类型来描述数据、元数据、信息、包、模板、模型和元模型等元素。数据是对信息对象的语义描述，可以实例化为具体数据对象并参与系统活动。元模型是领域内用于构建模型的规则模型。元素可以是任何可实例化的对象，其属性包括名称、类型、长度、结构、语法、语义、保持、出处、实现、规则和策略等。出处标明信息对象文件的位置、真实性和处理过程。这些内容可以组织成模板，作为信息对象实例化的操作载体。

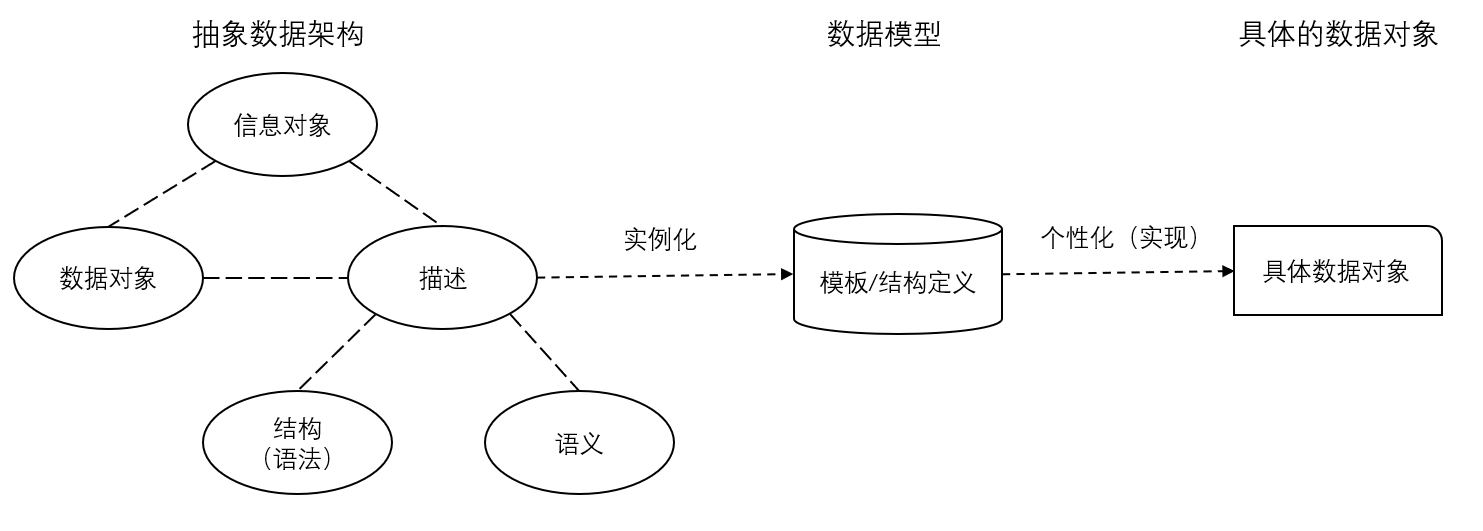
在建模时，系统中的信息对象通常通过多个视图进行描述，从非常抽象到非常具体。信息视角主要阐明数据的抽象规格，并提供从抽象到具体的语言描述。转换过程中的不同视图如图2-1所示。

图 2‑1 信息对象到具体数据对象的建模过程[]

抽象数据架构主要关注数据元素的定义、特定数据类型以及信息对象之间的关系，这些关系可以通过“相关”、“部分”和“被使用”等形式来表达。抽象数据结构一般在模型建立时出现。

从抽象数据架构到数据模型的过程是实例化的过程，即将抽象的信息对象和数据对象的特征信息具体化为数据模型的过程，通常在系统设计阶段进行。

数据模型主要包括数据模板结构、数据元素的名称和排列顺序等，可以是结构定义、数据库模板定义、外部文件或其他存储元数据的形式，通常需要单独描述。数据模型通常在系统设计阶段通过实例化过程生成。

从数据模型到具体数据对象的过程是个性化的过程，即根据不同信息流的使用场景和目的，在模板和结构的基础上填入具体数值。这个过程通常在系统进入详细设计或使用阶段时出现。

具体数据对象是一组数据流或字节流，用于信息的保存和交换。个性化是一个具体系统对特定信息生成、处理和解析识别的过程，从而允许异构系统之间交换不同形式的数据。

## 基于SEDS的建模设计

### SEDS介绍

SEDS的层次结构如图2-2，其标签结构包含DataSheet、Device、Package、Metadata、Datatype、Interface和Component等核心元素，这些元素共同构建了一个框架，用于系统化地描述设备的数据接口、参数、命令及其相关的行为和状态机。

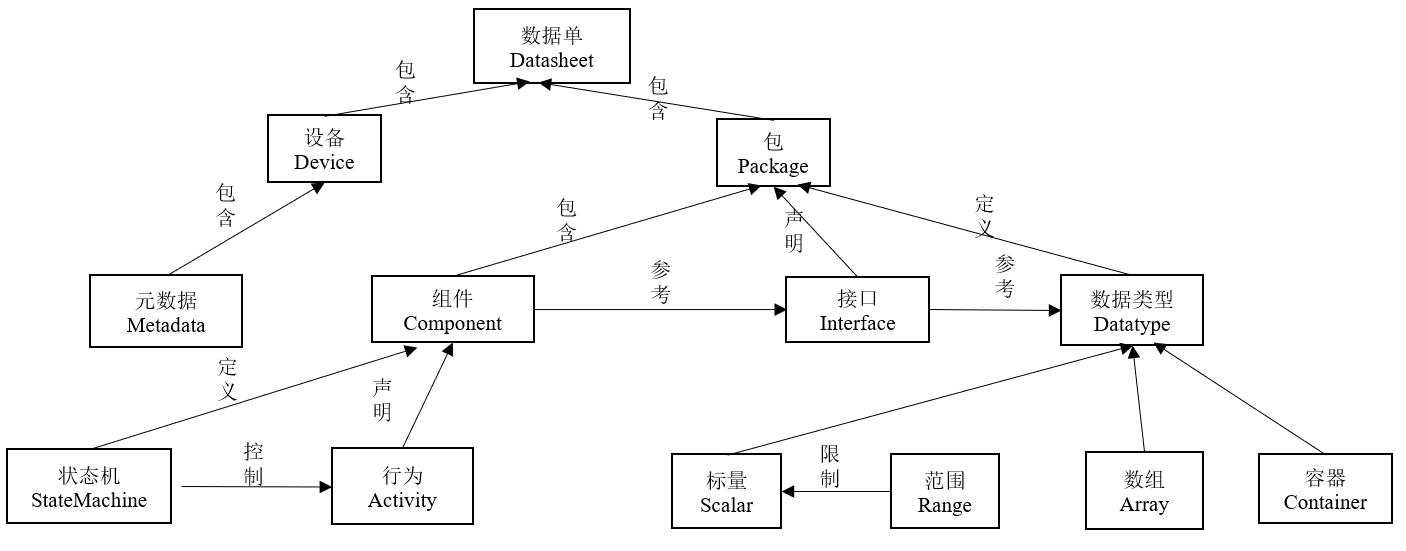


图 2‑2 SEDS标准的元素节点结构

设备（Device）元素主要存储 SEDS 实例化后的配置信息，如制造商、设备和接口属性等，同时可通过 数据类型（Datatype） 元素定义并赋值。

元数据（Metadata）元素用于描述数据本身的信息，包括其结构、内容、质量及应用条件，便于理解和管理。

包（Package）元素负责封装接口、组件及数据类型信息。

组件（Component）元素负责接口的具体实现，允许外部业务通过接口名称调用其实现函数。其作用包括 协议层参数映射、参数在活动（Activity）元素中的输入输出顺序，实现接口逻辑。

状态机（State Machine）元素用于响应事件并调度活动的执行。它由多个状态（State）、进入状态（EntryState）、退出状态（ExitState）和转换（Transition）组成，每个状态可以具有进入（OnEntry）和退出（OnExit）时的操作​

行为（Activity）元素指外部可执行代码，通过活动元数据进行描述和引用，活动执行前后设备的状态转换关系由状态机定义。

接口（Interface）元素作为一种特殊的数据类型，定义设备参数及可执行指令。

数据类型（DataType）元素规定了接口参数的语法和语义，包括 整型（IntegerDataType）、浮点型（FloatDataType）、字符串型（StringDataType）、二进制型（BinaryDataType）、布尔型（BooleanDataType）、枚举型（EnumeratedDataType）、数组（ArrayDataType）和容器（ContainerDataType）。SEDS 还允许定义自定义数据类型，将多个基本类型组合为新类型，类似 C 语言中的结构体，提高数据表达的灵活性。定义接口参数时，可直接引用这些数据类型，并进行更精细的设计。

标量（Scalar）元素是单值数据类型，与数组或容器等结构化类型相对。例如，整数（IntegerDataType）和浮点数（FloatDataType）都是标量类型​。

范围（Range）元素用于约束数据类型的值，以进行验证和识别。例如，整数或浮点数数据类型可以具有最小-最大范围（MinMaxRange），用于定义合法的数值区间。

数组（Array）元素用于表示相同类型数据的连续重复，每个元素可以通过索引访问。数组元素的数据类型由 DataTypeRef 属性定义，数组维度由 DimensionList 规定。

容器（Container）元素是具有命名条目的聚合数据类型，每个条目可以是任意类型的数据。它可以用于描述协议数据单元（PDU），并支持不同的编码方式。

SEDS各元素的属性标签需要根据元素所要表达的内容进行定义，主要的属性标签如表2-1所示。

表 2‑1 SEDS各元素的属性标签

|  |  |  |
| --- | --- | --- |
| 元素标签 | 作用 | 主要属性标签 |
| DataSheet | SEDS 文件的根元素，描述设备整体结构 | Device，Package |
| Device | 存储 SEDS 实例化后的设备配置信息 | Metadata，Package |
| Metadata | 设备描述信息，如设备型号、版本等 | Model，Version，… |
| Package | 封装接口、组件及数据类型信息 | Interface，Component，DataType |
| Interface | 定义设备参数、指令及已有接口的引用 | Parameter，BaseInterface，… |
| Component | 具体实现接口功能，支持参数映射及活动调用 | StateMachine，Activity |
| DataType | 定义参数的数据类型，包括基本类型、数组和容器 | Container，Array，Scalar，IntegerDataType，FloatDataType，… |
| StateMachine | 负责响应事件并调度活动的执行 | State |
| Activity | 外部可执行代码，可用于业务逻辑实现 | Input，Output |
| Scalar | 代表单值数据类型，如整数、浮点数等 | Type |
| Range | 约束数据类型值的合法范围 | MinRange，MaxRange，… |
| Array | 具有相同类型元素的集合 | DimensionList，DataTypeRef，… |
| Container | 复合数据结构，包含多个字段 | Field，… |

表中的属性“，…”表示属性标签可根据需要扩充。

### MIB与EDS的系统部署关系

基于CCSDS RASDS进行建模过程实例化时，最重要的是配置系统内部的MIB。MIB是一个根据业务、设备和系统内部应用建立的信息实体，可以是EDS的全集或是添加了系统集成综合信息后的超集，并通过EDS形式对外表达。

航天器上不同部分的MIB和EDS使用需求和方式各异，涉及MIB和EDS的部署关系及格式内容问题。

一个系统的组成部件可以抽象为三类实体：应用（Application）、业务（Service）和设备（Device）。它们在系统架构中按上、中、下三个层次进行部署，如图2-2所示。

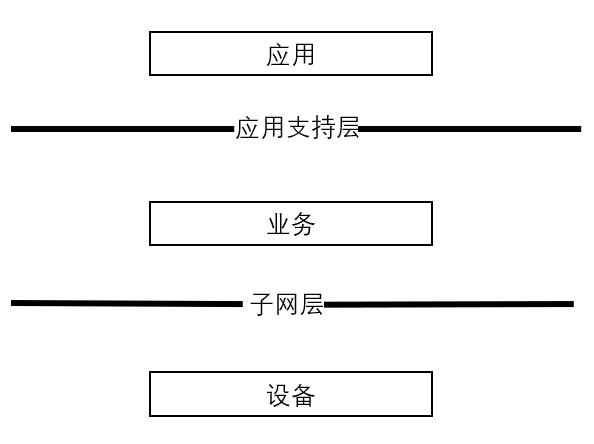


图 2‑3 系统层次部件划分关系

三者分别有各自的EDS，分别是EDS.A、EDS.S和EDS.D，三种EDS内容存在包含关系，通过EDS.A可以访问到所需的EDS.S和EDS.D内容。

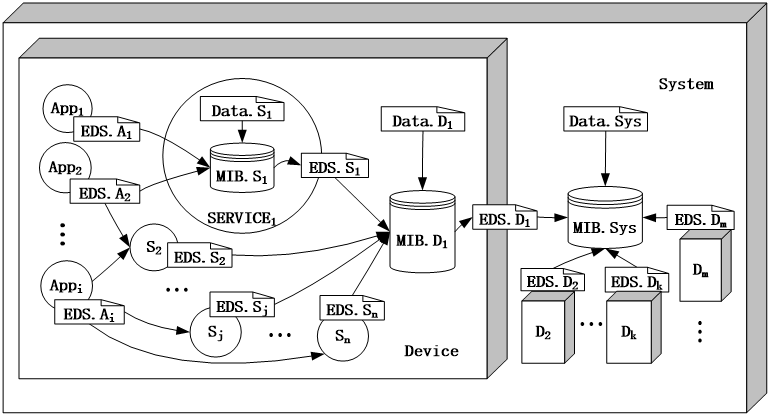
从系统部署视角来看，MIB可以部署在三个层次上，即业务、设备和系统（分别称为MIB.S、MIB.D、MIB.Sys）。这三层MIB与三层EDS之间的关系如图2-4所示。

图 2‑4 EDS与MIB部署关系

图中各部件只表达数据关系，内部的逻辑描述被省略。应用和业务之间的对应关系可以固定或按需变化。应用的MIB属于应用内部的设计，其对外关系已经通过EDS.A描述了，因此可以不做应用MIB的约束。系统级EDS.Sys是否需要要根据系统之上是否还有更大的系统需要构建，或者与其他系统进行交换。

### 基于MIB与EDS的建模设计思路

## 基于S-SEDS建模设计

PUS所提供的各项星上业务具有一个共同特点，即每项业务内部都有一个数据结构，支撑业务对外接收请求（指令）和发送报告（遥测）。该数据结构即为业务的MIB。这种设计理念符合先确定数据结构，再进行编程的程序设计原则。

一个系统可以拥有一个全局MIB.sys，并配有一个专门的MIB管理业务进行统一管理。MIB中保存的是EDS项，MIB管理业务可以提供对EDS项的增删、检索、修改等功能。根据系统复杂程度，全局MIB.sys也可以包含各个MIB.D、MIB.S，或者对它们进行索引，以便从系统全局对各组成部分进行直接管理和使用。反之，MIB.D、MIB.S也可以以同样的方式进行内部管理。当部件没有处理能力或比较简单时，其MIB可以等同于EDS项，由MIB.sys统一管理。

由于每个EDS项的内容差异较大，为充分反映其对象属性，组织格式可以采用变长包格式。EDS项在MIB中的保存方式是平面式的，无层次关系，以便于检索。各层EDS内容的包含关系可通过在EDS项中设计关联域来表达，而不是采用洋葱头式的层层包装方式，方便系统通过动态配置建立和撤销各种关联关系。

在使用方法上，SOIS EDS由关键词、术语和描述内容组成。关键词是XML描述EDS的语法，参照已有的XML语言和工具。术语是对EDS数据内容的语义描述，如功能标识、接口标识等。EDS内容需根据不同业务需求进行设计。为满足PUS的在线监视业务而设计的在线监视表，即为该业务的MIB。而EDS的内容应至少包括在线监视表及其外部访问方式。

### S-SEDS介绍

SEDS标准已被用于描述航天器设备接口，但仍需解决业务层面的即插即用问题，因此参照现有SEDS标准，对其元素结构和功能进行扩展，提出描述业务的SEDS（Service SEDS，S-SEDS）概念，以描述业务信息。按照图2-2的元素节点结构，设计的S-SEDS的元素结构如图2-7。

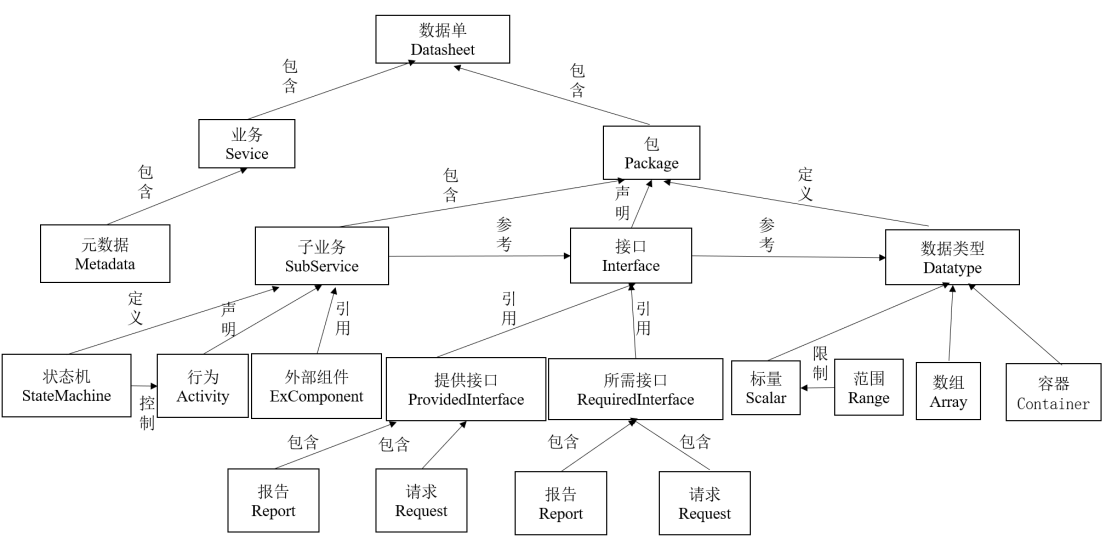


图 2‑5 S-SEDS元素节点结构

S-SEDS相较于SEDS做出改进如下：

1）引入业务元素（Service），替代设备元素（Device），以利用相同的元数据结构来描述业务层面的信息。

2）扩展包元素：在接口元素（Interface）上增加提供接口（ProvidedInterface）和所需接口（RequiredInterface）元素，以便描述业务单元的输入输出关系。

3）引入子业务（SubService）元素，描绘业务内部的组成结构。

4）增加请求（Request）和报告（Report）元素，描述业务的请求关系。

5）在子业务元素中嵌入外部组件（ExComponent）元素，实现对组件的逻辑调用关系，增强组件的复用性设计。

对应的改进效果如表2-3。

表 2‑2 S-SEDS改进效果

|  |  |  |  |
| --- | --- | --- | --- |
| 改进点 | SEDS | S-SEDS | 改进效果 |
| 面向对象 | Device | Service | 从设备管理扩展到业务管理 |
| 接口管理 | Interface | ProvidedInterface + RequiredInterface | 明确业务输入输出关系 |
| 层次关系 | 无 | SubService | 业务结构清晰，支持层次化设计 |
| 交互模块 | 无 | Request + Report | 业务交互更加直观和可管理 |
| 组件复用 | Component | ExComponent | 提高业务复用性 |

以上改进的优势：

1）扩展了SEDS标准的适用范围，从描述设备到描述功能业务，使得业务的即插即用成为可能。

2）提高了业务数据的可复用性，使用子业务和外部组件描述业务内部的组成结构和调用关系，使子业务和组件在不同任务中复用。

3）S-SEDS继承了SEDS已有的要素，使得对设备和业务以至于应用的描述有可能统一。

S-SEDS与业务即插即用的关系如表2-4。

表 2‑3 S-SEDS与业务即插即用关系

|  |  |  |
| --- | --- | --- |
| 序号 | 子能力 | S-SEDS结构元素 |
| 1 | 业务发现和访问能力 | 数据单DataSheet 包Package |
| 2 | 业务虚拟能力 | 业务 Service、子业务 SubService |
| 3 | |  | | --- | | 业务接口能力 |  |  | | --- | |  | | 接口 Interface、提供接口 ProvidedInterface、所需接口 RequiredInterface、请求 Request、报告 Report |
| 4 | 业务控制能力 | 组件 Component、外部组件 ExComponent、状态机 StateMachine、活动 Activity |
| 5 | 数据表示和存储能力 | 数据类型 Datatype、容器 Container、数组 Array、范围 Range、标量 Scalar |

由此可见S-SEDS的结构可以支持业务即插即用各自能力的需求，同时在SEDS标准基础上进行扩展，可用于实现兼容。S-SEDS的层次结构的元素标签如表2-5。

表 2‑4 S-SEDS层次化元素标签

|  |  |  |
| --- | --- | --- |
| 元素标签 | 作用 | 主要属性 |
| DataSheet | S-SEDS 文件的根元素，描述业务整体结构 | Service、Package、DocumentID、CreationDate、Version |
| Service | 存储 S-SEDS 实例化后的业务配置信息 | |  | | --- | |  |  |  | | --- | | Model、Version、ServiceName、Description、Timestamp | |
| Metadata | 业务描述信息，如业务名称、版本等 | Model，Version |
| SubService | 描述业务内部的组成结构 | SubServiceID、ParentServicID、Dependency |
| Package | 封装业务接口、组件及数据类型信息 | Interface、Component、DataType、PackageID、Dependencies |
| ProvidedInterface | 描述业务提供的接口，支持输入输出关系 | Parameter、BaseInterface、InterfaceID、Protocol、DataFormat |
| RequiredInterface | 描述业务所需的接口，支持输入输出关系 | Parameter、BaseInterface、InterfaceID、Protocol、DataFormat |
| Component | 具体实现业务接口功能，支持参数映射及活动调用 | StateMachine、Activity、ExComponent、ComponentID、ImplementationLanguage、API |
| ExComponent | 外部组件，增强组件复用性 | StateMachine、Activity、ExternalInterface、Vendor、Version |
| DataType | 定义业务参数的数据类型 | Container，Array，Scalar，IntegerDataType，FloatDataType， |
| StateMachine | 负责响应事件并调度活动的执行 | State、Transition、StateID、Condition、Action |
| Activity | 外部可执行代码，可用于业务逻辑实现 | Input、Output、ActivityID、ExecutionEnvironment、Timeout |
| Scalar | 代表单值数据类型，如整数、浮点数等 | Type、Unit、Scale |
| Range | 约束数据类型值的合法范围 | MinRange，MaxRange |
| Array | 具有相同类型元素的集合 | DimensionList、DataTypeRef、IndexingMethod、StorageOrder |
| Container | 复合数据结构，包含多个字段 | Field、FieldOrder、EncodingMethod |
| Request | 描述业务的请求关系 | Parameter、RequestID、OperationType |
| Report | 描述业务的报告关系 | Parameter、ReportID、Frequency |

S-SEDS的元素结构是在SEDS基础上扩展的，相较于SEDS元素结构的新增的元素主要有：

DataSheet 是 S‑SEDS 文件的根元素，主要用于描述整个业务的整体结构。它不仅包含了 Service 和 Package 两个核心部分，用以定义业务配置和接口封装，还携带了 DocumentID（文件唯一标识）、CreationDate（文件创建时间）以及 Version（版本号）属性。

Service元素取代了SEDS中的Device元素，专门用于描述业务层面信息。它包含业务的元数据，定义业务之间的输入输出关系。每个Service可以包含多个SubService。其中Model 表示业务所遵循的模型或架构，Version 标识该业务配置的版本信息，ServiceName 表示业务的名称，Description 则对业务的功能、特性及用途进行详细说明，Timestamp 记录了业务元数据的创建或最新更新时间。

SubService元素用于描述业务的内部层次，支持业务的模块化和层次化，将大型业务拆分为多个子业务，更好组织和管理复杂业务逻辑。它本质上是Service的组成部分，依赖于Service元素进行关联。

ProvidedInterface元素表示业务的提供接口，描述业务的输出功能。这使得外部业务可以通过接口与业务单元进行交互，实现数据的输出。ProvidedInterface包含Interface的参数定义和基础接口信息，有助于明确接口功能用途。

RequiredInterface元素表示业务的所需接口，描述业务的输入功能。它定义了业务单元依赖的外部接口，表明所属的数据信息。RequiredInterface可以保障业务与外部组件进行交互。其主要属性包括：SubServiceID唯一标识每个子业务，ParentServiceID指明该子业务所属的上层业务，Dependency描述子业务之间的依赖关系。

ExComponent元素是一种外部组件，通过嵌入外部组件可以调用外部系统或服务，从而使S-SEDS系统能集成更广泛的功能。其主要属性包括：StateMachine，用于定义组件内部的状态机和行为流程，Activity描述组件执行的具体操作，ExternalInterface指明与外部系统交互的接口信息，Vendor记录外部组件的供应商信息，Version用于标识组件的版本号。

Request元素描述业务的请求关系，代表一个业务单元对外部单元的请求。它通过请求特定的参数，使得业务系统内部和与外部其他系统更好的交互。其核心属性包括 ：Parameter指定请求涉及的业务参数，RequestID唯一标识该请求，以便跟踪和管理，OperationType定义请求的操作类型，如查询、修改或执行特定任务。

Report元素描述业务的报告关系，代表业务的报告或反馈信息。它能帮助业务记录操作结果、状态信息或进行反馈，从而是用户或其他业务单元能够获得更详细的信息，它也是业务交互的重要组成部分。其核心属性包括 ：Parameter指定报告涉及的业务参数，ReportID唯一标识该报告，用于追踪和管理，Frequency定义报告的生成频率，如定时上报、事件触发或按需生成。

相较于SEDS同名称元素也有不同的内涵，主要包括：

Metadata 元素在SEDS中主要用于描述数据的结构、内容、质量及应用条件，以便于数据的管理、存储和解析。在 S-SEDS 中，Metadata 的概念得到扩展，不仅用于数据描述，还增强了对业务信息的适配。S-SEDS 在 Metadata 中引入了Model（模型）、Version（版本）、ServiceName（业务名称）、Description（描述信息）和 Timestamp（时间戳）属性。

Package元素 主要用于封装业务相关的接口（Interface）、组件（Component）和数据类型（DataType），以支持模块化设计并提高系统的组织性和复用性。相比于 SEDS，S-SEDS 更加强调业务逻辑的层次化管理。Package 通过 PackageID 唯一标识自身，封装 Interface 以定义业务交互方式，包含 Component 作为功能单元，管理 DataType 以确保数据结构的一致性，同时通过 Dependencies 维护与其他 Package 之间的关联。

Interface 元素在SEDS中定义了设备的参数及可执行指令，使外部系统能够通过接口访问设备的功能。在 S-SEDS 中，Interface 的概念进一步细化，分为 ProvidedInterface（提供的接口） 和 RequiredInterface（所需的接口）。ProvidedInterface 用于定义业务单元对外提供的功能，使其他业务单元或外部系统可以调用其操作和数据参数，RequiredInterface 用于定义业务单元所依赖的外部接口，明确其需要从其他业务单元或外部系统获取的输入或服务。

这些新增的元素和变化的元素为S-SEDS提供了更加丰富的业务表述能力，是层次结构更加清晰，使得业务之间的交互、复用变得更加方便。

### 标准业务模型

对于业务的即插即用除了需要有业务的电子数据单的概念模型，还需要明确业务中EDS的输入输出关系和数据管理过程，在实现方式上采用SEDS的MIB+EDS的设计思路。即插即用的标准业务模型如图2-8。

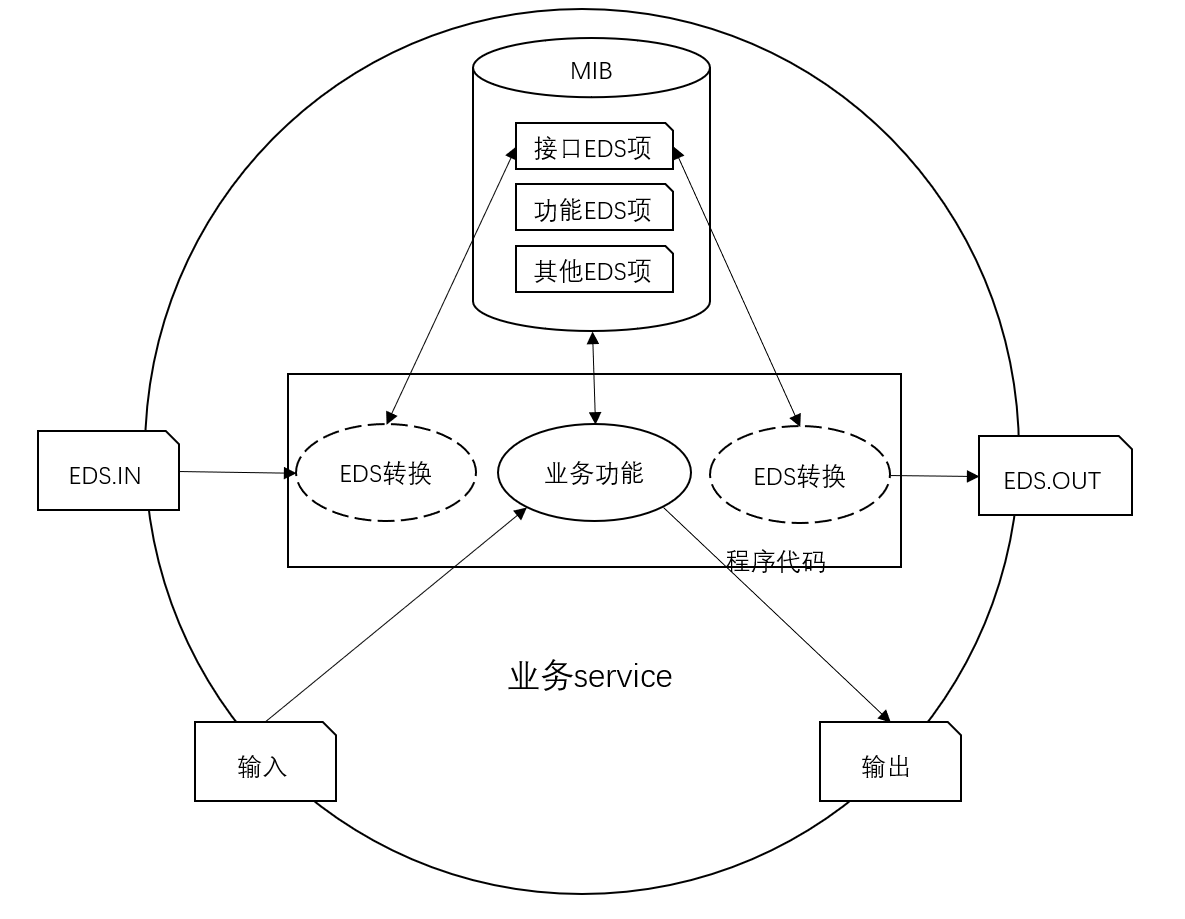


图 2‑6 即插即用业务模型[]

业务的内部数据结构可以统一在MIB的概念下，EDS内容可以源自业务模型所需的数据结构，包括业务的输入、输出以及内部MIB管理的信息。其核心思路是业务模型的MIB应有明确的格式，且内容可配置。EDS在格式设计上应具备可修改性和可扩展性，以允许同一业务模型的不同业务实例进行设计，从而实现业务的即插即用，适应不同系统的配置需求。业务程序部分除了常规的业务功能及其输入、输出外，为了解决业务可配置和自描述的问题，可基于MIB设计相应的EDS转换功能。如果输入输出的EDS与业务MIB中的EDS项内容格式匹配，则可以省略这种转换，与功能业务的输入输出保持一致。

### S-SEDS建模实现流程

S-SEDS的建模依据 EDS Schema 语法结构，将需求作为已知输入，使用 S-SEDS 模板设计工具定义数据的模板结构。通常“需求”可以根据业务或设备的制定方提供的接口控制文件（Interface Control Document，ICD）中定义的内容，描述接口所需的数据类型、接口、排列顺序、行为等信息。根据 S-SEDS 相关接口标准，提取星载业务的公共数据特征，进行通用化设计，包括该对象的输入、输出接口，以及内部数据的组织关系（MIB）等，形成 S-SEDS 模板文件，即 S-SEDS 模板设计过程。

具体的建模过程如图2-5所示。

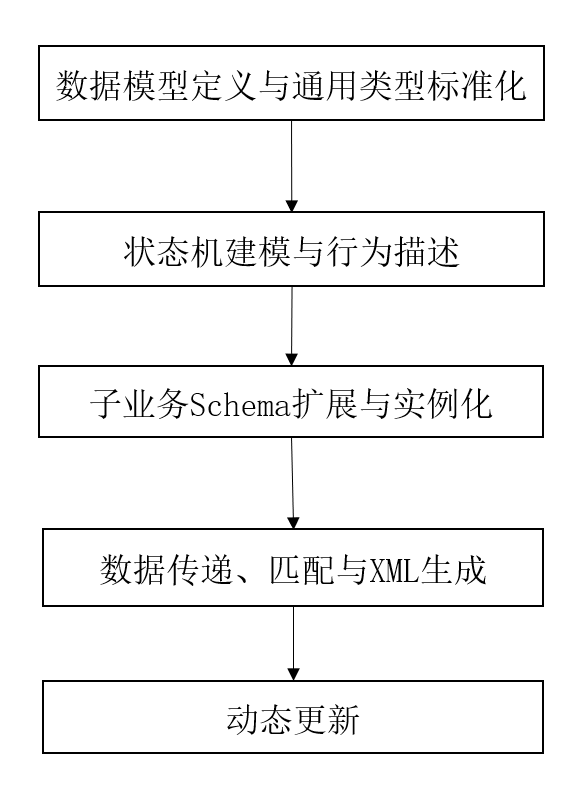


图 2‑5 S-SEDS建模过程设计

可以将建模设计过程分为以下五部分：

1）数据模型定义与通用类型标准化

首先进行数据模型定义，核心组件为DataSheet及其类型模板DataSheetType。通过Common.xsd文件集中定义通用数据类型（如StringDataType字符串类型、BianDataType二进制类型等），确保数据语义的全局一致性。在“数据填写”阶段，设计者使用SEDS模板工具选择这些预定义类型，并配置模块参数（如输入/输出参数、性能约束），最终生成结构化元数据。例如，定义事件表业务时，Common.xsd中的IntegerDataType可直接用于描述事件ID，而TimestampDataType则规范时间戳格式。此步骤为后续业务扩展与交互奠定标准化基础。

2）状态机建模与行为描述

状态机（StateMachine）及其相关类型（StateType、TransitionType）用于描述业务模块的动态行为逻辑。StateType定义模块状态（如“待执行”“执行中”“完成”），TransitionType则规定状态转换条件（如时间触发、指令触发）。在“标签匹配”阶段，系统根据SEDS Schema规则，将用户输入的触发条件（如“温度超过阈值”）自动映射为状态机事件。

3）子业务Schema扩展与实例化

针对具体星载业务（如事件表、在线监视），需通过子业务Schema设计扩展通用数据类型，并构建专属元素与属性。例如，在Common.xsd的IntegerDataType基础上，可派生EventIDType，添加业务特定约束（如取值范围1~1000）。在“数据绑定”阶段，dom4j工具将用户通过Map集合提交的参数（如{"EventID": 1001， "Priority": "High"}）转换为子业务XML元素（如<EventTable>标签及其子节点）。此过程确保业务逻辑既符合全局标准，又满足个性化需求。

4）数据传递、匹配与XML生成

用户填写的数据通过Map集合封装（键值对形式）传递至后端。系统在“数据抽取”阶段提取关键字段（如模块ID、性能指标），随后遍历SEDS Schema进行“标签匹配”，将字段映射至预定义元素（如Map中的Priority映射为<Priority>标签）。在“序列化”阶段，根据XML节点的从属关系排序，生成以DataSheet为根元素的EDS模板文件。例如，事件表业务的XML文件以<DataSheet>为根，包含<EventTable>、<StateMachine>等子结构，完整描述业务逻辑与数据流。

5）动态更新

为适应需求变化（如新增事件类型），模型支持动态扩展Common.xsd中的数据类型，并通过版本控制机制更新子业务Schema。例如，新增CustomEventType后，系统在“序列化”阶段自动检测并适配新类型，确保历史数据的向后兼容性。

## 本章小结

本章围绕航天器即插即用的建模方法展开研究。分别包括信息视角建模分析、SEDS标准建模设计、S-SEDS扩展设计与建模三个方面来展开。

首先基于RASDS信息视角框架，系统梳理了数据对象从抽象架构到具体实例的转换逻辑。通过定义信息对象的元数据描述规则（结构、语义、约束等），提出分层建模方法：抽象数据架构→数据模型→数据对象实例。

针对设备接口描述问题，构建了SEDS分层模型，定义7类核心元素及属性。结合MIB-EDS部署关系，提出工具链驱动的建模流程，支持航天器设备的快速装配与动态配置。

为解决业务即插即用问题，创新性扩展SEDS标准为S-SEDS框架，设计五阶段建模流程，结合通用类型库（Common.xsd）与动态Schema扩展机制，实现星载业务的灵活适配。

本章从理论、模型、实现方法三个层面进行分析，设计了支持业务即插即用的业务描述方法，为航天器业务即插即用提供解决方案。

# S-SEDS与OWL转换设计

## 基于SANA 的OWL到XSD转换方法分析

SANA官方文档通过提供一系列文件，详细定义了SOIS（Space Operations Interface Specification）EDS（Event Data Sheets）和DoT（Definition of Terms）的构成元素。具体来说，SANA官网提供了8个文件和一个压缩包，这些文件之间的相互关系如图3-1。

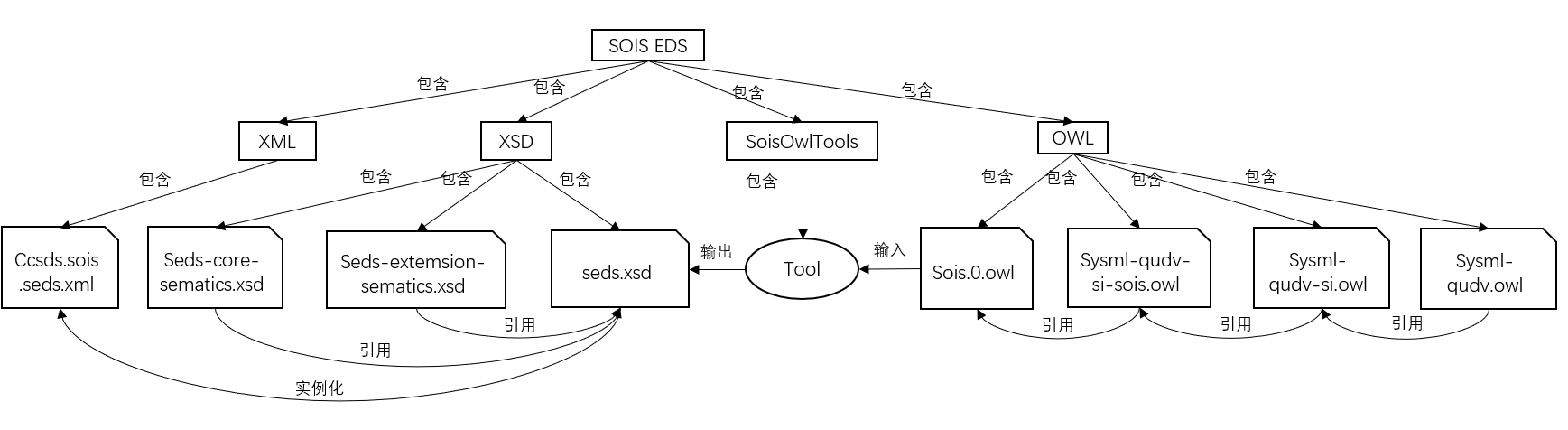


图 3‑1 SoisOwlTools文件转换关系

根据SANA官网的介绍，SOIS EDS的内容分为四类：XML文件、XSD文件、OWL文件和SoisOwlTools工具。接下来将详细介绍这些文件的构成和功能。

### XSD相关文件分析

SANA提供的XSD文件包括seds-core-sematics.xsd、seds-extension-sematics.xsd和seds.xsd三大类。这些文件各自承担了不同的功能，具体如下：

seds-core-sematics.xsd：此文件定义了SEDS的核心语义，包含了多个基础数据类型（如DoTQuantityKind、DoTUnit等），这些类型用于描述物理量的种类、单位及坐标系等标准化元素。

seds-extension-sematics.xsd：该文件定义了SEDS的扩展语义，支持用户在标准基础上进行定制，包括本地错误控制类型（ExtErrorControlType）、编码精度类型（ExtEncodingAndPrecisionType）等。

seds.xsd：作为SEDS的主XML Schema文件，seds.xsd定义了SEDS文档的核心结构，包括数据单（DataSheet）和包文件（PackageFile）的根元素，以及设备（Device）和包（Package）的数据结构等。

这三者之间的关系如下所示：seds-core-sematics.xsd和seds-extension-sematics.xsd共同为seds.xsd提供语义扩展，后者则引用了这两个文件以形成完整的XML Schema架构，此外，seds.xsd还包含了Web标准所定义的XML标签，如图3-2所示。



图 3‑2 seds.xsd根元素代码

### XML相关文件分析

在XML文件方面，ccsds.sois.seds.xml是用于定义SEDS数据字典的实例化文件。它包含了多个通用数据类型的定义，如整数（INT8、INT16、INT32、INT64）、无符号整数（UINT8、UINT16、UINT32、UINT64）、浮点数（FLOAT32、FLOAT64）以及布尔值（BOOLEAN）等。此外，它还包括一些复杂的数据类型，如四元数（Quaternion）和位置向量（Position），具体使用如图3-3所示。

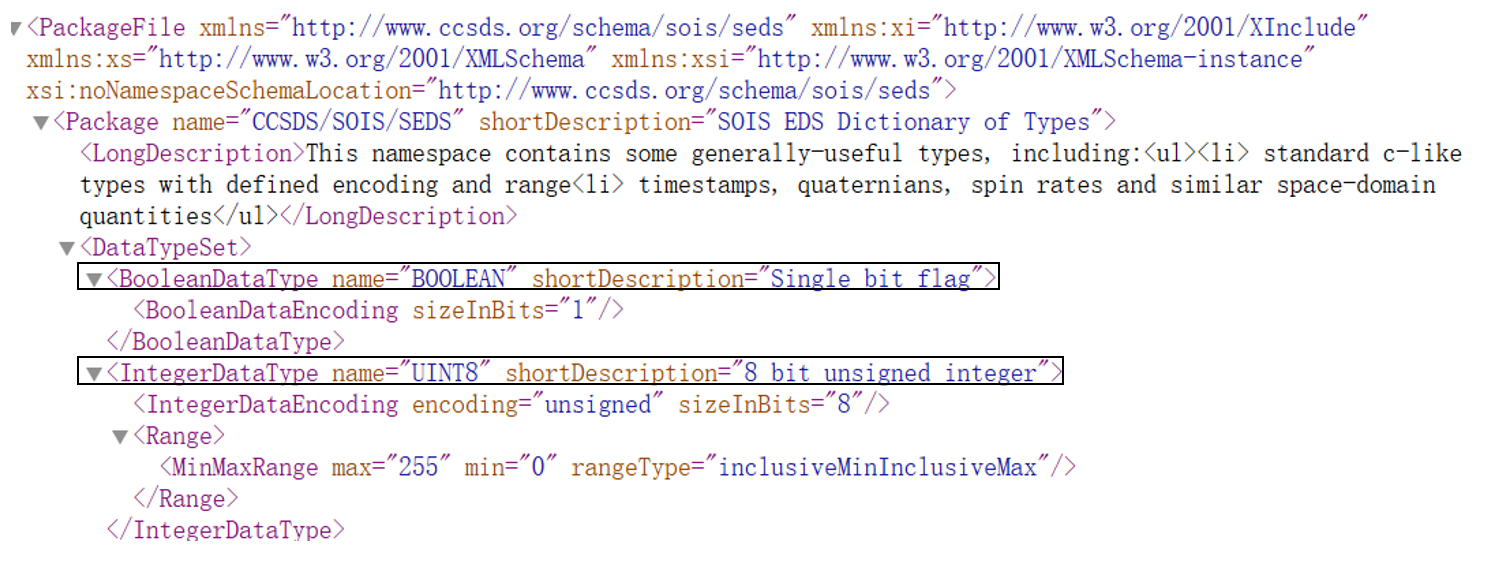


图 3‑3 ccsds.sois.seds.xml示例代码

ccsds.sois.seds.xml与seds.xsd的关系可以通过实例化来理解。seds.xsd定义了数据结构，而ccsds.sois.seds.xml则为这些结构提供了具体的数值和配置，以完成对数据类型的实例化。通过这种方式，XML文件能够实现对seds.xsd中定义的所有数据类型的具体赋值，确保数据文档符合SEDS标准。

### OWL相关文件分析

SANA还提供了四个主要的OWL文件，包括sysml-qudv.owl、sysml-qudv-si.owl、sysml-qudv-si-seds.owl和sois.0.owl。其引用关系如图3-4所示。

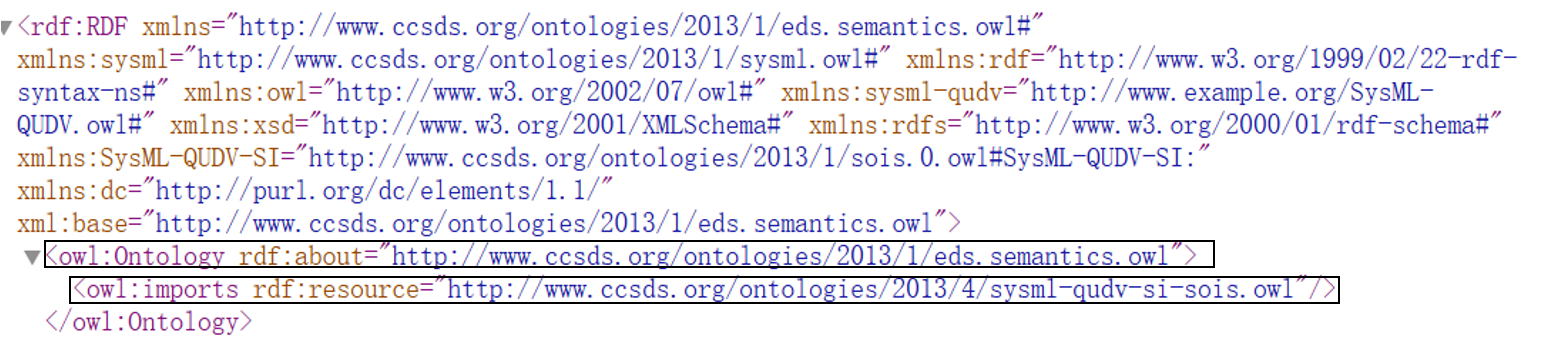


图 3‑4 sois.0.owl根元素代码

这些OWL文件通过引用关系紧密关联。Sois.0.owl引用sysml-qudv-si-sois.owl文件，sysml-qudv-si-sois.owl引用sysml-qudv-si.owl文件，而sysml-qudv-si.owl引用了sysml-qudv.owl文件。

其中每个文件承担了不同的功能：

1）sysml-qudv.owl：定义了OWL中的核心概念，包括量的种类（QuantityKind）和单位（Unit）等。

2）sysml-qudv-si.owl：扩展了核心OWL文件，定义了与国际单位制（SI）相关的量和单位，如设备量种类（DevicedQuantityKind）和设备单元（DevicedUnit）。

3）sysml-qudv-si-seds.owl：在sysml-qudv-si.owl的基础上，添加了与SEDS相关的量纲，如模数转换计数（aToDCount）和帧数（frameCountQK）等。

4）sois.0.owl：此文件主要用于描述CCSDS模板操作（ModelOfOperation）和数据传输速率（DataTransferRateRange）等内容。

### SoisOwlTools工具分析

SoisOwlTools的主要功能是将OWL文件转换为多种格式，包括HTML、XSD和TXT文件。转换过程通过以下几个步骤完成：首先构建模板文件规定映射规则，然后输入一个OWL文件（例如sois.0.owl），然后使用模板生成工具根据OWL文件的内容构建对应的模板。该模板支持三种不同格式：文本（AsText）、HTML（AsHtml）和XSD（AsXsd）。最后根据所选模板，格式转换工具会生成相应的格式文件。OWL文件格式转换过程如图3-5所示。

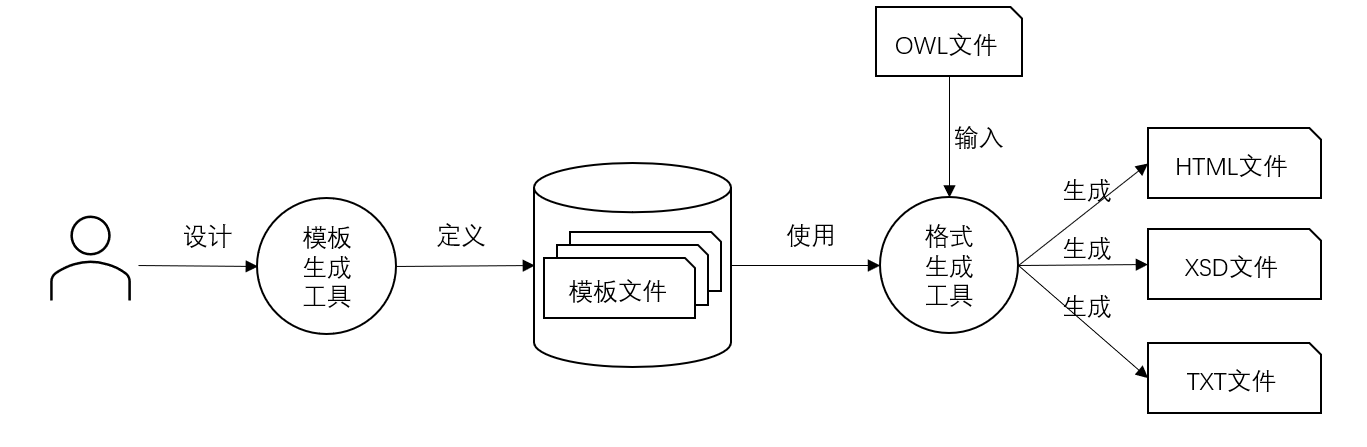


图 3‑5 OWL文件格式转换过程

该工具的具体转换流程以航天领域本体（OWL）为起点，通过解析本体文件中定义的类（如设备、协议、计量单位）、属性及约束关系，依据预置的语义映射规则，将形式化语义逐层转换为标准化的XSD架构。基于基础本体（sysml-qudv.owl）定义核心语义文件（如seds-core-semantics.xsd），定义通用数据类型（如DoTUnit单位类型），然后定义扩展语义文件（如seds-extension-semantics.xsd），根据核心语义文件和扩展语义文件生成航天领域本体文件（sysml-qudv-si-seds.owl），支持航天任务定制参数（如激光通信频段），根据定义的映射规则转换为符合seds.xsd的模板文件。生成的XSD文件进一步通过XML实例文件（ccsds.sois.seds.xml）实现双向验证：一方面验证数据实例的语法合规性（如数值范围、嵌套结构），另一方面通过嵌入RDF语义注解（如rdf:type）关联本体概念，确保数据在逻辑层符合领域知识定义。

在模板构建过程中，系统通过解析OWL/RDF文档、识别命名空间和遍历文档元素等步骤，将文档中的对象属性、数据类型属性和OWL类等信息提取并根据定义的映射规则转换为适用于不同格式的输出。通过调用相应的方法（如AssimilateObjectProperty、AssimilateDatatypeProperty等）实现对各类数据的精确解析和分类。

### OWL到XSD映射规则

基于对SANA的转换工具的流程和具体实现的文件分析，可以总结出以下的映射规则：

1. 基础元素映射。将OWL中的核心语义元素转换为XSD的基本数据类型与结构，如表3-1。
2. 复杂结构映射。处理OWL中的复合类、容器类型及继承关系，生成XSD的嵌套结构，如表3-2。
3. 约束与验证规则映射。将OWL中的逻辑约束转换为XSD的语法验证规则，主要包括数值范围约束、基数约束和唯一约束，如表3-3。

表 3‑1 OWL到XSD的基础元素映射规则

|  |  |  |
| --- | --- | --- |
| OWL元素 | XSD标签 | 示例 |
| owl：Class | xs:complexType | sois:Device → <xs:complexType name="DeviceType"> |
| owl:DatatypeProperty | xs:element+xs:attribute | hasID → <xs:element name="ID" type="xs:string"/> |
| owl:ObjectProperty | 嵌套xs:element | hasComponent → <xs:element name="component" type="ComponentType" maxOccurs="unbounded"/> |
| rdfs:subClassOf | xs:extension | Quaternion继承Vector → <xs:extension base="VectorType"> |

“owl:Class → xs:complexType”代表OWL类（如设备、传感器）映射为XSD复杂类型，定义实体结构（如<xs:complexType name="DeviceType">），保留属性和嵌套元素。“owl:DatatypeProperty → xs:element+xs:attribute”代表简单属性（如ID、数值）转换为XSD元素或属性（如<xs:element name="ID" type="xs:string">），绑定数据类型（如字符串、浮点数）“owl:ObjectProperty → 嵌套xs:element”代表对象属性（如“包含组件”）映射为嵌套元素（如<xs:element name="component" type="ComponentType">），通过类型引用表达实体关联。rdfs:subClassOf → xs:extension继承关系通过扩展父类实现（如<xs:extension base="VectorType">），子类复用父类结构并添加新属性（如四元数新增标量）。

表 3‑2 OWL到XSD的复杂结构映射

|  |  |  |
| --- | --- | --- |
| OWL元素 | XSD标签 | 示例 |
| 容器类型 | xs:complexType + xs:sequence | Quaternian类 → <xs:complexType name="Quaternian"> |
| rdfs:subClassOf | xs:extension | SpaceWireLinkState继承ModelOfOperation → <xs:extension base="ModelOfOperationType"> |

将OWL中的复合类（如Quaternian）映射为XSD的嵌套结构（xs:sequence），通过xs:sequence定义嵌套元素的有序结构。子类通过xs:extension继承父类的复杂类型，并添加新的属性或元素。

表 3‑3 约束与验证规则映射

|  |  |  |
| --- | --- | --- |
| OWL元素 | XSD标签 | 示例 |
| owl:Restriction + xsd:minInclusive/xsd:maxInclusive | xs:minInclusive/xs:maxInclusive | <br><xs:minInclusive value="-3.141592653589793"/><br> <xs:maxInclusive value="3.141592653589793"/><br> </xs:restriction><br> |
| owl:minCardinality/owl:maxCardinality | xs:element的minOccurs/maxOccurs属性 | <br><xs:element name="component" type="ComponentType" minOccurs="1"/> |
| OWL键约束 | xs:key | <br><xs:key name="InterfaceNameKey"><br> <xs:selector xpath="./seds:Interface"/><br> <xs:field xpath="@name"/><br></xs:key> |

上述三个示例展示了OWL的语义约束转换为XSD的结构化验证规则：​数值范围约束通过OWL的owl:Restriction限制角度值在-π到π之间，映射为XSD的xs:minInclusive/xs:maxInclusive，基数约束利用owl:minCardinality强制设备至少包含一个组件，对应XSD的minOccurs="1"，唯一性约束通过OWL的注解声明接口名称唯一性，转换为XSD的xs:key规则，防止字段重复。

## XSD到OWL转换设计

### 转换目标

SANA的工具链仅支持OWL到XSD的单向转换，无法逆向构建具备语义表达能力的本体模型，所以提出了基于SANA方案的S-SEDS的XSD到OWL转换方法。并通过以下改进实现S-SEDS的XSD到OWL转换：

1. 业务语义完整性。需遵循W3C OWL 2规范确保将XSD中的业务流程、服务接口及业务按照规则转换为OWL本体，按照S-SEDS设计的标签层次结构进行转换，确保业务逻辑的形式化表达。
2. 领域可扩展性。基于CCSDS 876.0-B-1标准，采用模块化本体设计（owl:imports），支持第三方通过预定义的sois:ExtensionPoint类进行扩展，定义common.xsd作为公共xsd实现扩展复用。
3. **安**全性控制。依据ECSS-Q-ST-80C标准[7]，通过OWL注解属性标注数据密级，结合dcterms:accessRights定义访问策略，对敏感参数（如DataTransferRateRange）通过owl:Restriction实施控制。

通过上述改进，将支持业务逻辑与数据结构的映射，实现业务语义的表达。能通过第三方插入新业务逻辑，减少重复开发成本。还能进行敏感参数标注，保障数据安全。

### XSD到OWL映射规则

#### SANA映射规则对比

为实现S-SEDS到OWL的语义转换，将S-SEDS通过分层结构分为子业务、接口、状态机等标签，在表2-3的原则上对于这些标签结合OWL2规则将S-SEDS标签逐层映射为可具备推理的本体元素。并在SANA定义的OWL到XSD转换的基础上做出以下修改：

1. 将XSD元素逆向化转变为OWL，给出逆向对应关系。
2. 将XSD的元素按照S-SEDS的标签结构映射为关联的领域OWL，通过sois:CCSDSProtocol本体类绑定标准接口。
3. 将XSD注释中的密级标记转换为sois:SecurityLevel，并通过owl:Restriction动态限制访问。

按照这个改动的核心标签到OWL的映射规则可以分为业务逻辑与模块化封装、接口与交互协议、数据类型与约束、行为和状态控制、安全与扩展机制这五部分。

#### 业务逻辑映射规则

为更好表达业务的结构层次关系需要对业务模块的嵌套关系进行定义，具体的映射规则如表3-4。

表 3‑4 业务逻辑映射规则

|  |  |  |
| --- | --- | --- |
| S-SEDS标签 | OWL元素 | 设计逻辑 |
| Service | owl:Class | 将业务逻辑单元定义为独立的类，通过rdfs:subClassOf关联领域本体，支持业务分类与推理。 |
| Package | owl:ObjectProperty | 包作为业务模块的封装关系，通过containsPackage属性表达层级结构。 |
| SubService | owl:Class + rdfs:subClassOf | 子业务继承父业务的结构，复用父类属性，同时扩展特有逻辑。 |

S-SEDS到OWL的语义映射规则通过分层标签体系实现了航天电子数据业务模型的结构化与知识化表达。​业务（Service）​被映射为独立的OWL类（如TelemetryProcessingService），通过rdfs:subClassOf关联航天领域本体（如sois:CCSDSProtocol），继承标准化数据处理规则（如帧格式解析、数据校验算法），确保遥测/遥控业务符合空间数据系统咨询委员会（CCSDS）的协议规范，包（Package）​以owl:ObjectProperty（如containsDataFlow）显式定义数据业务的模块化流程（如数据采集包嵌套数据清洗、压缩、加密等子流程），支持任务驱动的流程动态重构（如深空任务中启用数据压缩子模块以节省带宽），子业务（SubService）​通过继承机制（rdfs:subClassOf）复用父类核心逻辑（如指令验证规则），同时扩展任务专属策略（如某型号卫星新增激光通信数据的解码算法），实现航天数据业务的标准化与场景灵活性的统一。

#### 接口与交互协议映射规则

接口协议是跨模块、跨系统协作的核心枢纽，其定义的精确性与一致性直接影响数据交互的可靠性。S-SEDS通过语义化映射规则，将XSD接口描述转化为可推理的OWL模型。具体规则如表3-5。

表 3‑5接口与交互协议映射规则

|  |  |  |
| --- | --- | --- |
| S-SEDS标签 | OWL元素 | 设计逻辑 |
| Interface | owl:ObjectProperty | 定义双向交互协议，通过owl:inverseOf声明逆向关系。 |
| ProvidedInterface | |  |  | | --- | --- | | owl:ObjectProperty |  | | 表示服务提供方的接口能力，通过owl:inverseOf关联需求方。 |
| RequiredInterface | owl:ObjectProperty | 表示服务依赖方的接口需求，通过逆向属性实现双向绑定。 |

接口（Interface）​被定义为owl:ObjectProperty，例如卫星遥测数据接口（TelemetryInterface），通过owl:inverseOf声明逆向关系链（如providesTelemetry与requiresTelemetry），明确数据流的双向语义依赖。​提供接口（ProvidedInterface）​映射为服务供给方的能力描述（如卫星平台的数据发送接口），而需求接口（RequiredInterface）​则表征服务依赖方的调用需求（如地面站的数据接收接口），两者通过逆向属性实现协议的双向绑定。此规则通过OWL的逆向关系公理，支持接口协议的动态兼容性验证（如频段匹配、数据格式一致性）与权限约束注入（如加密算法校验）。

#### 数据类型与约束映射规则

S-SEDS通过本体化的数据类型映射规则，将航天电子数据的结构定义与约束条件转化为可推理的语义模型。具体如表3-6。

表 3‑6 数据类型与约束映射规则

|  |  |  |
| --- | --- | --- |
| S-SEDS标签 | OWL元素 | 设计逻辑 |
| Datatype | owl:DatatypeProperty（基本类型）或owl:Class（复合类型） | 基本类型映射为数据属性，复合类型映射为类，支持嵌套结构。 |
| Scalar | owl:DatatypeProperty + xsd类型绑定 | 绑定XSD原生类型，确保数据格式兼容性。 |
| Range | owl:Restriction + xsd:minInclusive/xsd:maxInclusive | 定义参数取值范围，通过数值约束保障数据合法性。 |
| Array | rdf:List | 有序集合通过RDF列表结构表示，保留元素顺序性。 |

数据类型（DataType）分为基本数据类型和复杂数据类型，基本数据类型​（如整型、浮点数）映射为owl:DatatypeProperty并绑定XSD原生类型（如xsd:float），确保跨系统数据格式解析的一致性，复合数据类型​（如遥测数据帧）则映射为嵌套的owl:Class，通过属性链（如hasTimestamp、hasAttitude）定义层次化结构，支持复杂数据实体的形式化建模。​标量（Scalar）​作为原子数据单元（如传感器温度值、电压值），通过owl:DatatypeProperty直接关联XSD类型（如xsd:double），严格约束数据格式（如双精度浮点数精度要求），避免因类型歧义导致的数据解析错误。Range（范围）范围约束（如姿态角范围[-π, π]），采用owl:Restriction结合xsd:minInclusive/xsd:maxInclusive进行显式声明，Array​（数组）通过rdf:List结构（rdf:first、rdf:rest）保留元素顺序性，适配航天业务对时序敏感性的严格要求。

#### 行为与状态控制映射规则

S-SEDS通过语义化行为-状态映射规则，将离散的操作指令与状态转移关系转化为可推理的OWL模型，具体如表3-7。

表 3‑7 行为与状态控制映射规则

|  |  |  |
| --- | --- | --- |
| S-SEDS标签 | OWL元素 | 设计逻辑 |
| StateMachine | owl:Class + owl:oneOf（状态枚举） | 将状态机映射为状态枚举类，通过owl:oneOf定义状态集合。 |
| Activity | owl:Class | 原子操作映射为行为类，通过owl:ObjectProperty关联执行时序。 |
| Request | owl:Class | 操作指令映射为请求类，关联输入参数。 |
| Report | owl:Class | 响应结果映射为事件类，关联操作状态，支持事件溯因。 |

​状态机（StateMachine）​被映射为OWL类，通过owl:oneOf枚举所有合法状态（如卫星电源系统的Normal、Safe、Fault），定义状态转移的离散边界，原子操作（Activity）​映射为独立的行为类（如DataEncryption），通过owl:ObjectProperty（如hasNextStep）显式声明操作时序，约束任务流程的完整性，操作指令（Request）​映射为请求类（如RebootCommand），关联输入参数（如延迟时间delay）并通过owl:Restriction强制类型（xsd:integer）与范围（minInclusive=0），响应结果（Report）​映射为事件类（如ThermalResponse），关联操作终止状态（如Overheated）与触发原因（如HighPowerLoad），支持异常溯因与状态回滚。

#### 安全与扩展映射规则

|  |  |  |
| --- | --- | --- |
| S-SEDS标签 | OWL元素 | 设计逻辑 |
| Metadata | owl:AnnotationProperty | 描述性元数据（如版本号、创建时间）作为语义注解，支持非逻辑性信息标注。 |
| Container | owl:Class + owl:intersectionOf | 组合模式实现参数引用，通过owl:intersectionOf继承父类属性并扩展新参数。 |
| 安全控制 | sois:SecurityLevel + owl:Restriction | 敏感参数（如DataTransferRate）标注密级（如“机密”），通过owl:Restriction限制访问权限 |

元数据（Metadata）​映射为owl:AnnotationProperty（如dcterms:created、sois:version），以语义注解形式记录非逻辑性信息（如版本号、创建时间），支持数据溯源与生命周期管理，容器（Container）​映射为owl:Class，通过owl:intersectionOf组合继承父类属性（如通用量纲common:Unit）并扩展型号专属参数（如激光通信波长），实现参数引用的动态适配，安全控制通过sois:SecurityLevel标注敏感数据密级，结合owl:Restriction限制访问权限（如仅控制团队可读写）。

## XSD到OWL转换过程实现

### XSD到OWL转换流程

XSD到OWL转换分为XSD 文件解析、OWL 模型生成、数据绑定与序列化的实现过程与技术方案，重点介绍如何将 S-SEDS 数据模型从 XSD 文件转换为 OWL 本体，并将实例数据绑定至 OWL 个体，最终生成符合 RDF/XML 格式的 OWL 文件。具体的转换过程如图3-10所示。

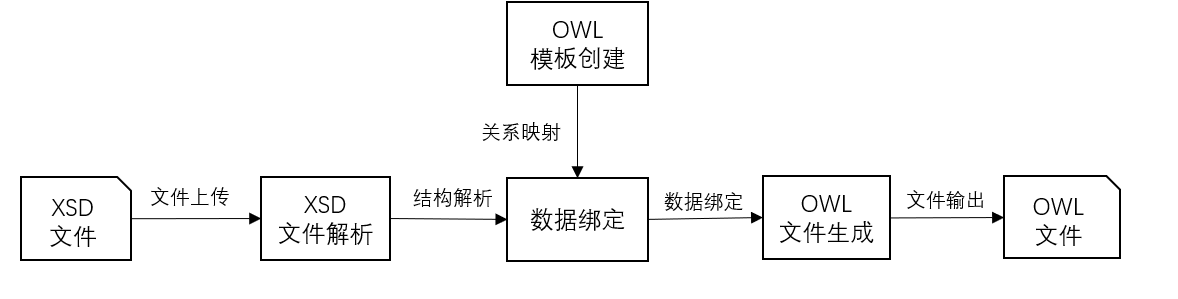


图 3‑6 XSD到OWL转换过程

S-SEDS从XSD到OWL的转换流程通过四阶段逐步生成符合W3C标准的OWL文件。首先上传XSD文件然后通过XSD文件解析模块进行结构解析，然后创建OWL模版，将解析完的结构映射到对应的OWL元素，然后通过RDF转换和格式编码实现数据绑定和序列化，最后通过OWL文件生成模块输出OWL文件。

### XSD文件解析

本阶段的目标是读取 XSD 文件，提取出其中的数据类型、接口定义和组件关系，并为后续的 OWL 模型生成提供必要的结构信息。关键在于构建 XSD 文件的命名空间和标签层次映射表，以确保 OWL 本体中各个元素的准确映射。

具体实现过程：

1）读取 XSD 文件：

使用 XML 解析器（Java 的 JAXB）对 XSD 文件进行解析。XSD 文件定义了 XML 数据的结构、数据类型和约束条件。在解析过程中，需要提取文件中的复杂类型（xs:complexType）、简单类型（xs:simpleType）、元素（xs:element）和属性（xs:attribute）等关键信息，为后续处理奠定基础。

2）提取数据类型和接口定义：

对 XSD 中的各类元素进行详细分析。复杂类型（xs:complexType）将映射为 OWL 中的类（rdf:Class），而简单类型（xs:simpleType）则对应 OWL 的数据属性（rdf:DatatypeProperty）。元素和属性的定义也需要根据其在 XSD 中的属性（如 minOccurs、maxOccurs）和数据类型进行处理和映射。这样能使 OWL 本体结构中既保留数据类型的约束，又能反映数据间的关系。

3）构建命名空间和标签层次映射表：

XSD 文件中的命名空间定义对于本体的生成至关重要。需要将 XSD 文件中定义的命名空间与标签层次结构映射到 OWL 本体中，以避免命名冲突，并确保 OWL 本体的层次关系与 XSD 文件中的标签层次一致。每个命名空间将映射为 OWL 本体中的一个命名空间，使得生成的 OWL 文件符合语义网的标准。

上面实现流程具体实现需要使用Java 中的 JAXB 解析工具读取 XSD 文件，并通过递归算法遍历嵌套结构，提取其中的元素类型、接口定义和关系信息。最后将解析的 XSD 元素转换为内存中结构化的数据模型，确保后续处理过程中信息的完整性与准确性。

### OWL模板创建

基于从 XSD 文件解析得到的结构信息，生成 OWL 本体。此阶段的核心任务是将 S-SEDS 数据类型和组件映射为 OWL 中的类、数据属性和对象属性，并建立类之间的继承关系、属性约束和对象关系。

具体实现过程：

1）创建 OWL 本体：

使用 OWLAPI库创建 OWL 本体。在 OWL 本体中，为每个 XSD 文件中的复杂类型（xs:complexType）创建对应的 OWL 类（rdf:Class），并为每个简单类型（xs:simpleType）定义 OWL 数据属性（rdf:DatatypeProperty）。通过命名空间的定义，确保 OWL 类和属性在全局范围内的唯一性。

2）将 S-SEDS 数据类型映射为 OWL 类和属性：

XSD 文件中的每个复杂类型映射为 OWL 本体中的一个类，而每个简单类型则映射为 OWL 的数据属性。特别地，对于带有 minOccurs 和 maxOccurs 限制的元素，需要在 OWL 中通过约束条件（如 rdf:range 和 rdfs:range）来表示数据类型的范围和多重性限制。同时，对所有映射的类和属性，需确保 OWL 本体的层次结构与 XSD 文件保持一致。

3）建立类继承、属性约束、对象关系：

在 OWL 本体中，类之间的继承关系通过 rdfs:subClassOf 来表示。例如，XSD 文件中的继承关系（如 xs:extension）应转化为 OWL 中的类继承关系。同时，XSD 中的属性约束（如最小值、最大值）将映射为 OWL 本体中的数据属性约束（rdfs:range）。如果 XSD 中存在对象关系（如 xs:sequence、xs:choice），则这些关系在 OWL 中通过对象属性（rdf:ObjectProperty）表示，从而构建起类之间的关联与层次。

上述实现流程具体需要在 OWLAPI 中，通过 OWLClass 和 OWLDataProperty 创建 OWL 类与数据属性。类之间的继承关系通过 RDFS.subClassOf 表达，而属性约束使用 RDFS.range 和 OWLRestrictions 进行约束。

### 数据绑定

将从 XML 数据实例化的 S-SEDS 数据转化为 OWL 个体，并绑定其属性值，最终生成符合 RDF/XML 格式的 OWL 文件。此阶段的核心任务是将数据与本体模型进行绑定，并通过适当的序列化格式输出。

具体实现过程如下：

1. 将 XML 数据实例化为 OWL 个体：

基于 XSD 中定义的结构和属性约束，使用 XML 数据实例创建 OWL 个体（rdf:Individual）。例如，XML 中的 <event> 元素将映射为一个 OWL 个体，其中 eventID、eventType 等属性将映射为 OWL 的数据属性。通过该过程，能够将 XML 文件中的数据转换为与 OWL 本体兼容的形式，便于后续的推理和查询。

2）映射 S-SEDS 属性值为 OWL属性的具体值：  
XML 数据中的每个元素和属性值将根据其类型映射到 OWL 数据属性中。具体地，属性值将依据其数据类型（如字符串、整数等）映射为 OWL 数据属性的具体值。需要特别注意数据类型的映射和数据约束的遵循，例如，整数类型的数据应映射为 OWL 中的 xsd:int 数据类型，字符串类型的数据应映射为 xsd:string。

上述实现流程在 OWLAPI 中需要使用 OWLModelManager 将 OWL 个体与属性值进行绑定，生成无序的OWL文件。

### OWL文件生成

将无序的OWL文件生成序列化的OWL文件，核心是对于OWL文件的序列化。

在完成数据绑定后，系统通过本体序列化技术将结构化数据与本体模型整合，生成符合语义网标准的OWL文件。通过 OWLAPI 或 RDFLib 将 OWL 本体和数据实例序列化为符合 RDF/XML 格式的 OWL 文件。这一过程确保了生成的 OWL 文件符合语义网标准，能够被其他语义网工具进行处理和推理。

上述实现流程在 OWLAPI 中使用 RDFXMLDocumentFormat 将 OWL 本体及其个体序列化为 RDF/XML 格式。

## 本章小结

本章围绕航天数据模型的语义化转换展开研究，旨在解决生成数据文件不便于复用管理，并基于SANA给出的OWL到XSD的映射规则提出改进的XSD到OWL逆向转换方法。

首先分析了SANA工具链的组成与功能，详细解析了其XSD、XML、OWL文件及SoisOwlTools工具的映射规则。

然后提出了基于SANA的XSD到OWL转换设计，通过核心标签映射、接口协议定义、安全控制规则及模块化扩展机制，实现业务逻辑、数据类型与安全约束的语义化表达。

最后设计了转换流程的具体实现，涵盖XSD解析、OWL本体生成、数据绑定与序列化，最终生成符合RDF/XML标准的OWL文件，支持航天数据的推理与共享。

本章通过逆向映射规则与领域适配设计，基于SANA工具链的语义化进行反向设计，为航天型号知识管理提供了标准化、安全的本体建模方案。

# S-SEDS工具链的设计与实现

## 工具链需求与设计目标

### 应用场景分析

结合星载业务即插即用应用场景，S-SEDS数据对象的生成工具链由三部分组成：S-SEDS模板设计工具、S-SEDS文件生成工具以及S-SEDS格式转换工具，S-SEDS应用场景如图4-1所示。

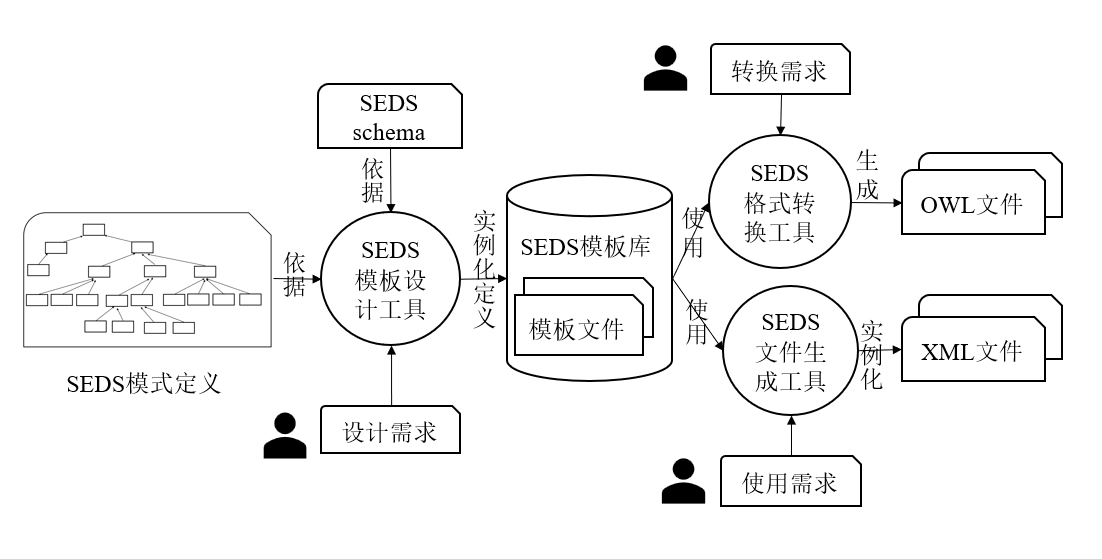


图 4‑1 S-SEDS应用场景

S-SEDS模板设计工具依据S-SEDS规范生成和调整业务数据模板，支持设计者自定义配置信息。该工具根据业务需求中的数据元素，查找并选择S-SEDS Schema中的标签。通过调用XML读写操作函数，将这些数据元素与相应标签进行绑定，并依据S-SEDS标准规则和业务执行逻辑进行标签的序列化处理，生成对应的XSD文件。

S-SEDS业务元数据及其定义的数据结构包含在模板文件中，存储于S-SEDS模板库。模板设计工具通过动态链接库（DLL）实现对模板库中的模板文件的增删改查操作，从而实现模板文件的设计和选用。

S-SEDS文件生成工具使用挑选出的模板文件，实例化所需的业务数据文件，将其存储于业务数据文件库中，并可对库中的文件进行局部调整，添加配置参数以生成新的业务数据文件。

S-SEDS格式转换工具用于将生成的XSD文件转化为适用于各类系统和工具的OWL文件。格式转换工具提高了业务数据的互操作性和可交换性，使得数据能够在异构系统间得到有效利用，支持后续基于OWL文件的知识存储[14-15]。由于遵循了S-SEDS元素的约束，该工具能够根据不同航天器的业务格式进行转换，从而实现相同的业务数据模板和业务数据跨任务使用。

### 工具链核心功能和目标

为满足星载业务即插即用场景中设备快速装配、灵活更换与跨系统互操作的迫切需求，S-SEDS工具链以标准化数据建模与自动化语义转换为核心，旨在实现航天器接口数据从设计到应用的全生命周期管理。工具链通过以下核心功能支撑航天器研制与在轨任务。基于S-SEDS标准的电子数据单应用数据流如图4-2。

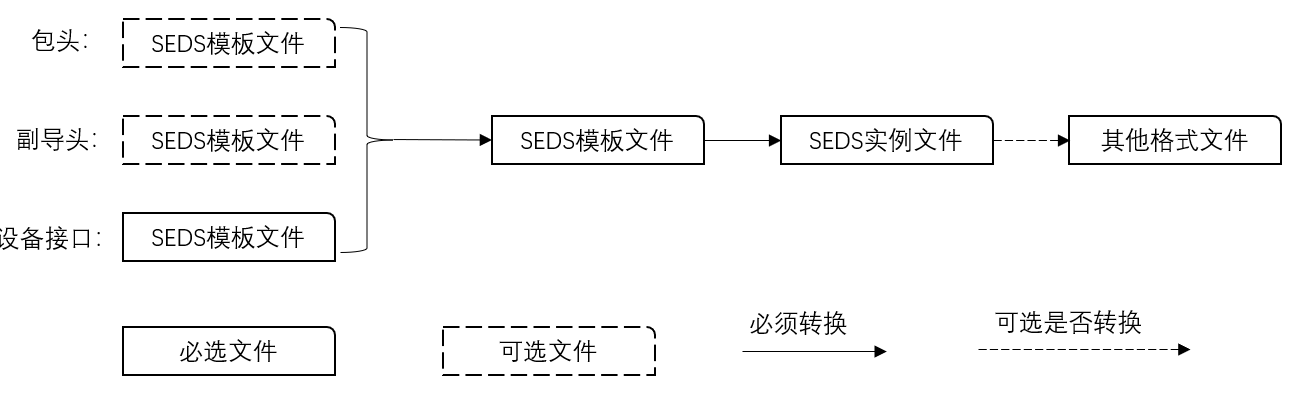


图 4‑2 基于S-SEDS标准的设备接口数据处理的数据流

S-SEDS描述的对象是业务自描述信息“名片”。S-SEDS的数据可以是任意自定义格式，也可以是包格式，各部分可以有各自的模板文件。在应用时再选择模板进行组合，将信息数据填入，生成S-SEDS实例文件。不同类型的设备有可能接受不同形式的数据文件，可根据需要进行数据格式的转换，如转换为二进制文件。

S-SEDS设计、生成、转换的过程设计的工具如表4-1。

表 4‑1 S-SEDS设计、生成、转换工具

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 工具 | 说明 | 使用场景 |
| 1 | S-SEDS模板设计工具 | 根据标准和用户需求定义XSD模板 | 设计阶段 |
| 2 | S-SEDS文件生成工具 | 模板实例化赋值生成S-SEDS的XML文件 | 设计阶段和在轨运行阶段 |
| 3 | S-SEDS格式转换工具 | 根据映射规则将XSD模板转换为其他格式文件 | 设计阶段和在轨运行阶段 |

按照表4-1的工具链可以在航天器研制和应用阶段实现通过S-SEDS工具链对于数据的处理，实现航天器软件的快速开发，便于设备的快速装配和灵活更换。

## 工具链软件架构

### 分层架构设计

基于星载业务即插即用需求，S-SEDS工具链采用分层模块化架构设计，通过解耦用户交互、业务逻辑与数据管理功能，实现高效、灵活的业务数据建模与转换。软件架构如图4-3所示，分为表示层（Presentation Layer）、业务逻辑层（Business Logic Layer）和数据访问层（Data Access Layer），各层协同工作，支持模板设计、文件生成与格式转换全流程。

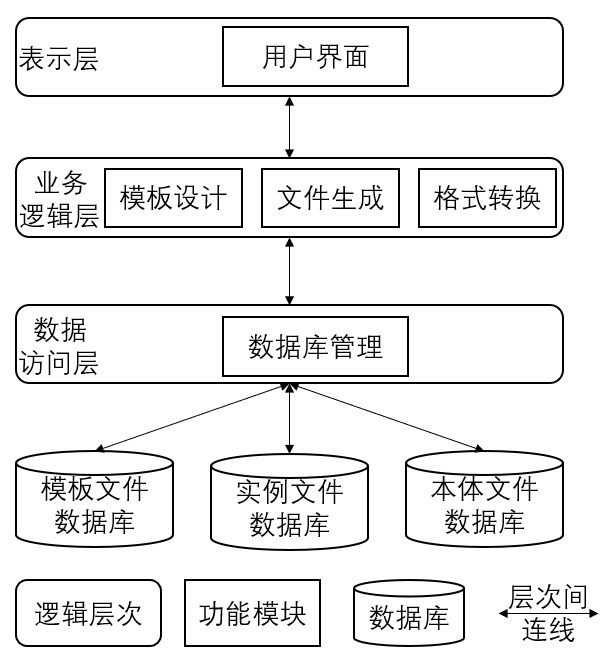


图 4‑3 软件架构层次设计

表示层作为用户交互入口，通过Vue.js构建响应式单页应用（SPA），集成拖拽式模板设计、参数输入表单及OWL映射规则配置面板，提供实时数据校验与可视化预览功能。用户可通过Element UI组件库动态配置S-SEDS标签层级结构，生成符合航天器接口规范的XML模板，并实时渲染结果。前端通过WebSocket实现异步消息通知（如转换进度提示），确保交互高效性与用户体验流畅性。

业务逻辑层为核心功能引擎，包含模板设计、文件生成与语义转换三大模块。模板设计引擎基于JAXB解析S-SEDS Schema，结合XSLT动态生成标准化模板，文件生成器通过DOM4J实现XML数据绑定，利用XPath定位节点并填充实例数据，语义转换器依托OWLAPI将XSD映射为OWL本体，结合Jena框架执行SPARQL查询以验证语义一致性。业务逻辑层采用Spring Boot微服务架构，通过RESTful API与表示层交互，集成全局异常拦截器捕获数据校验错误（如数据类型冲突），确保数据流转的可靠性。

数据访问层采用多模态数据库架构，通过MySQL、MongoDB与MinIO+Jena TDB协同管理模板、实例及本体数据，满足航天器业务数据的多样化存储需求。MySQL关系型数据库存储结构化模板元数据（如标签层级、版本号），利用B+树索引实现高效检索（响应时间<50ms），MongoDB文档数据库支持非结构化XML实例文件的灵活存储，通过Sharding技术横向扩展至PB级数据量，MinIO分布式对象存储结合语义引擎，实现OWL本体文件的高可用存储，支持复杂规则推理。技术选型兼顾ACID事务、横向扩展与语义能力。数据访问层通过乐观锁机制实现多用户协作下的数据一致性及安全访问，角色权限细化至字段级控制。

### 工具链协同工作流程

工具链协同工作流程以用户需求为驱动，通过分层协作与模块化联动，实现从模板设计到语义转换的无缝衔接。其核心流程分为三个阶段，各阶段跨层协作机制如表4-2。

表 4‑2工具链系统工作流程

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | 用户层 | 逻辑层 | 数据层 |
| 模板设计 | 拖拽配置S-SEDS标签，预览模板结构 | 解析配置，生成符合CCSDS标准的XSD模板 | 存储模板至MySQL，记录版本与元数据 |
| 文件生成 | 选择模板并输入参数（如时间戳、指令编码） | 参数映射至XML标签，生成结构化实例文件 | 实例文件存入MongoDB，记录操作日志 |
| 格式转换 | 配置XSD到OWL的映射规则，触发转换任务 | 基于OWLAPI生成RDF/XML本体文件 | OWL文件存储至MinIO，构建语义索引 |

该工具链通过模板设计、文件生成与格式转换三阶段实现星载业务数据的全流程管理：用户通过可视化界面拖拽配置S-SEDS标签并输入业务参数（如时间戳、指令编码），逻辑层动态生成符合CCSDS标准的XSD模板，并将参数映射为XML实例文件，最终通过OWLAPI转换为RDF/XML本体，数据层采用多模态存储（MySQL管理模板版本、MongoDB存储实例文件、MinIO+Jena TDB处理本体语义），结合Redis缓存与分布式架构保障高并发性能。流程以自动化闭环为核心，通过REST API实现跨层数据流转，异常实时反馈至用户界面，确保高效性与可靠性，为复杂星载业务提供标准化、语义化的数据支撑。

### 工具链设计采用的技术

#### DOM解析与数据动态绑定

在该系统中，DOM（文档对象模型）解析技术被广泛应用于处理XML模板文件。通过解析XML模板文件，系统能够识别并读取其中的标签和元素结构。用户输入的数据将被动态地映射到这些XML标签中，确保数据在不同的业务场景中被准确、有效地呈现。使用JavaScript或相关前端技术库DOMParser来实现这一过程，能够提供灵活、实时的交互体验。

在数据绑定过程中，系统通过观察用户输入的字段并实时更新相应的XML结构。这使得用户可以在浏览器端直接与模板进行交互，并且快速查看数据如何映射到最终的XML文件中，从而提高用户操作的直观性和效率。

#### XML序列化与反序列化

序列化与反序列化技术用于处理文件的读取和写入操作。序列化是将系统内的对象转换为XML格式，从而实现数据的持久化存储。反序列化则是将XML文件中的数据加载回系统内部的数据结构，以便进一步操作。

在实现中，JavaScript对象（如JSON）和XML文件之间的转换是通过自定义的序列化/反序列化方法进行的。通过这些方法，系统能够将XML文件快速转换为内部的对象模型，便于后续的操作和处理。而当需要将数据存储回XML文件时，系统会将内部的数据结构重新转换为XML格式，并通过文件操作进行保存。

#### 前后端技术

为了提升用户体验，前端技术在此系统中起到了至关重要的作用。使用了现代的前端框架，如Vue.js，它能够高效地构建单页应用（SPA）并进行组件化开发。Vue.js 提供的数据绑定和组件化特性，使得数据与视图之间的同步变得更加简便和高效。

通过Vue的双向数据绑定，用户的输入能够实时反映到界面上，而系统的反馈也能即时更新到用户界面。例如，当用户选择模板或填写数据时，Vue会自动更新相应的XML结构并显示在页面上。这种高效的交互方式显著提高了用户体验，并且能够降低开发和维护的复杂度

## 工具链功能模块

### 模板设计工具

S-SEDS模板设计工具的核心功能涵盖定义共享数据类型、构建数据单元结构、设计特定子业务的XML Schema，以及搭建状态机和数据类型。该工具通过在common.xsd文件中集中管理共享数据类型，支持数据类型的再利用和统一更新，从而保证整个项目的统一性。此外，工具还允许扩展或细化这些共享类型，以适应特定子业务的需求，增强了模板的灵活性和适应性。具体实现如图4-4所示：

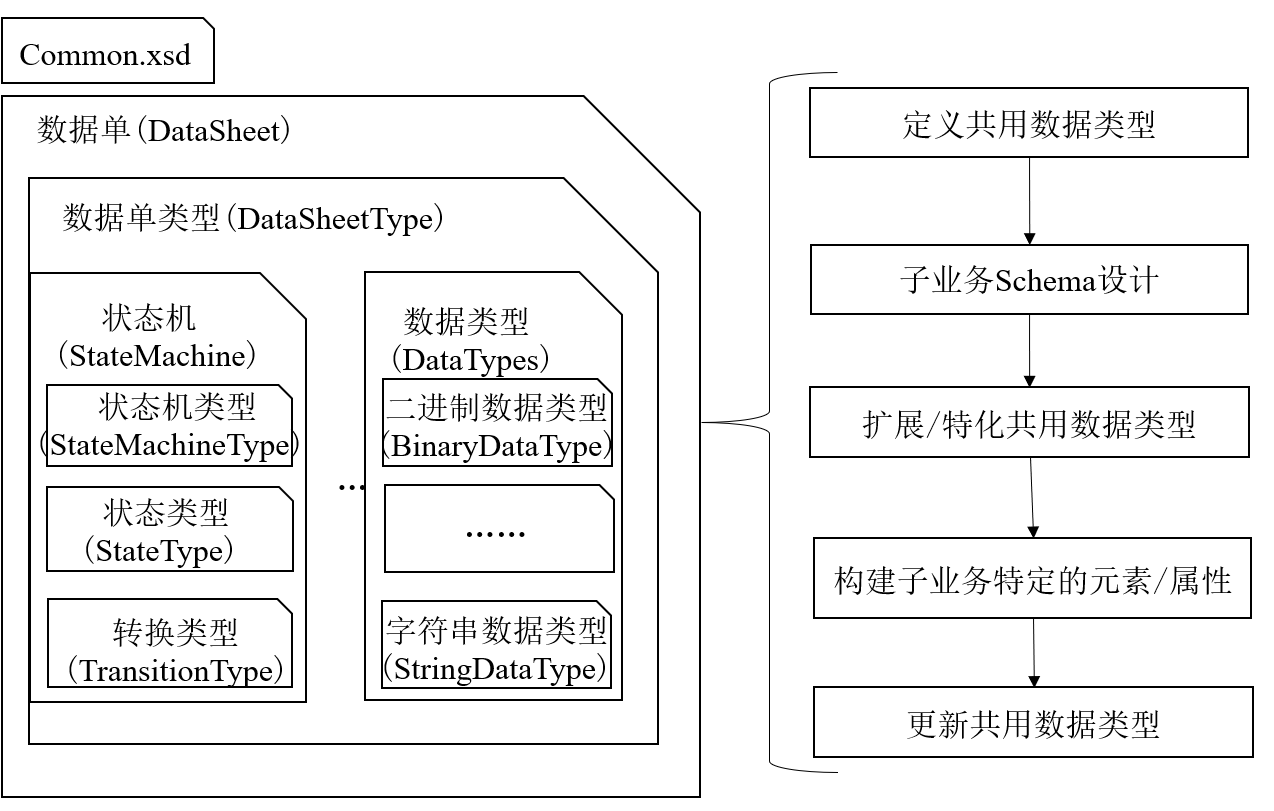


图 4‑4 S-SEDS模板设计流程

在common.xsd中首先定义通用数据类型，随后创建数据单类型（DataSheetType），并为每个子业务设计Schema，其中包含状态机和数据类型定义。接着，根据子业务的具体需求扩展或细化通用数据类型，并构建特定的元素和属性。在整个设计过程中，对通用数据类型的任何更新都会自动应用于引用这些类型的子业务模板中，从而确保数据的一致性和准确性。

为了实现异构系统间数据的一致性交互、后续数据的重用和继承，传递的

EDS 应遵循统一的标准，即S-SEDS标准。数据设计者在特定的系统架构下，依据 S-SEDS Schema 语法结构（S-SEDS Schema），将需求作为已知输入，使用 S-SEDS 模板设计工具定义数据的模板结构。通常“需求”可以根据业务或设备的制定方提供的接口控制文件（Interface Control Document，ICD）中定义的内容，描述接口所需的数据类型、接口、排列顺序、行为等信息。根据 S-SEDS 相关接口标准，提取星载业务的公共数据特征，进行通用化设计，包括该对象的输入、输出接口，以及内部数据的组织关系（MIB）等，形成 S-SEDS 模板文件，即 S-SEDS 实例化过程。在有具体用户需求时，结合应用场景，补充其余设计，形成具体 S-SEDS 数据文件，即 S-SEDS 个性化过程。该文件声明信息对象的所有特性和要求，通过特定信息流的形式，将部件的数据信息提供给系统识别。

具体的建模过程如下所示。

1)数据填写

使用 S-SEDS 模板设计工具，设计者选择 S-SEDS 数据类型，定义参数、包装、接口、组件、限定信息等。

2)数据传递

使用 Map 集合将填写好的数据封装传递到软件后端处理。

3)数据抽取

取出 Map 集合中保存的用户数据。

4)标签匹配

遍历 S-SEDS Schema 中的标签自动选择合适的元素描述用户数据，这个过程遵循 S-SEDS 规则建模。

5)数据绑定

使用 dom4j 工具将已匹配好的元素转换成 XML 标签，一次匹配成功的过程就生成一次 XML 标签，最后生成无序的 XML 标签集。

6)序列化

工具系统后端根据 XML 节点的从属关系进行标签排序，生成 S-SEDS 模板文件。

### 文件生成工具

用户首先从模板库中挑选适合其业务需求的S-SEDS模板，然后在工具界面中输入模板所需的参数值，同时利用自动填充功能，依据历史数据或业务规则预填部分常见字段。在用户完成输入后，系统将对数据进行验证，以确保所输入的参数符合S-SEDS的标准。具体如图4-5所示。

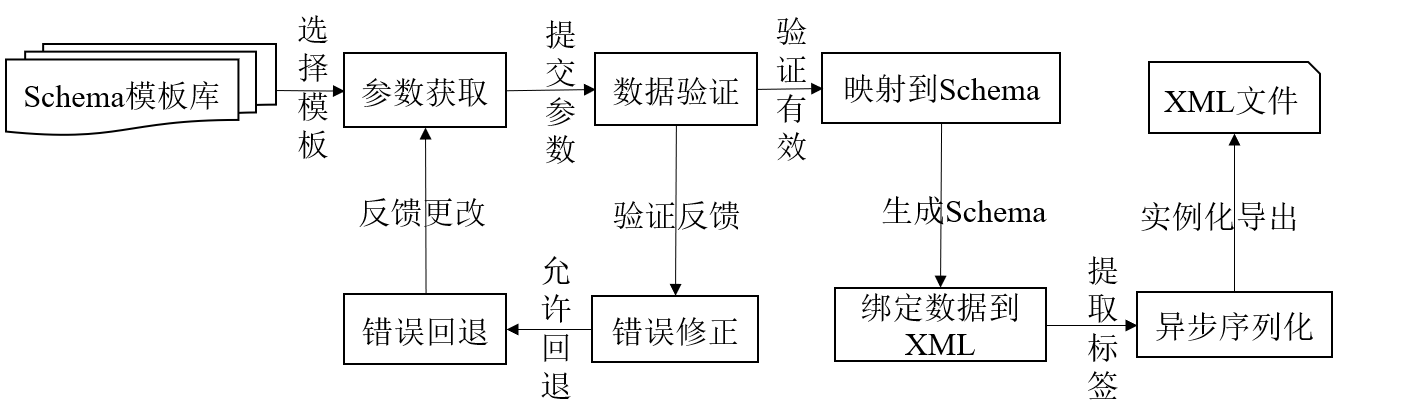


图 4‑5 S-SEDS文件生成流程

在用户输入的数据通过验证后，这些数据会被映射到预定义的模板中。具体实现方式跟模板设计工具类似。该工具采用Map集合来保存用户输入的键-值对，这种数据结构确保了用户指定的参数可以与S-SEDS Schema中的标签精确匹配[16]。借助这一映射机制，系统能够确保数据被准确放置到XML模板的相应位置。

在完成数据映射后，工具会遍历S-SEDS Schema中的标记，并借助DOM API将Map集合内的数据与XML标记关联起来，从而生成一棵树状结构的XML文档[17]。此过程中，系统支持对XML文档进行动态调整，用户可以在文档生成时添加、修改、删除或搜索元素。

最终，该工具对生成的XML文档进行序列化处理。用户能够导出完全符合S-SEDS标准的XML文件，用于后续的业务操作或数据交换。

### 格式转换工具

实现S-SEDS格式转换工具的目标是将S-SEDS的XSD文件转换为OWL本体，从而深化数据语义的理解并实现自动化处理。该流程首先借助XML解析器读取输入的XSD文件，提取其结构与内容，包括复杂类型、简单类型、元素声明和属性声明。随后，工具基于解析的XSD结构开始构建OWL本体。为了确保一致性，首先定义与XSD文件中targetNamespace相匹配的OWL命名空间。接着，复杂类型被转化为OWL类，而XSD中的元素和属性则映射为OWL的对象属性或数据属性，并且处理简单类型和枚举类型，生成相应的OWL数据类型和枚举类[18]。具体如图4-6所示。

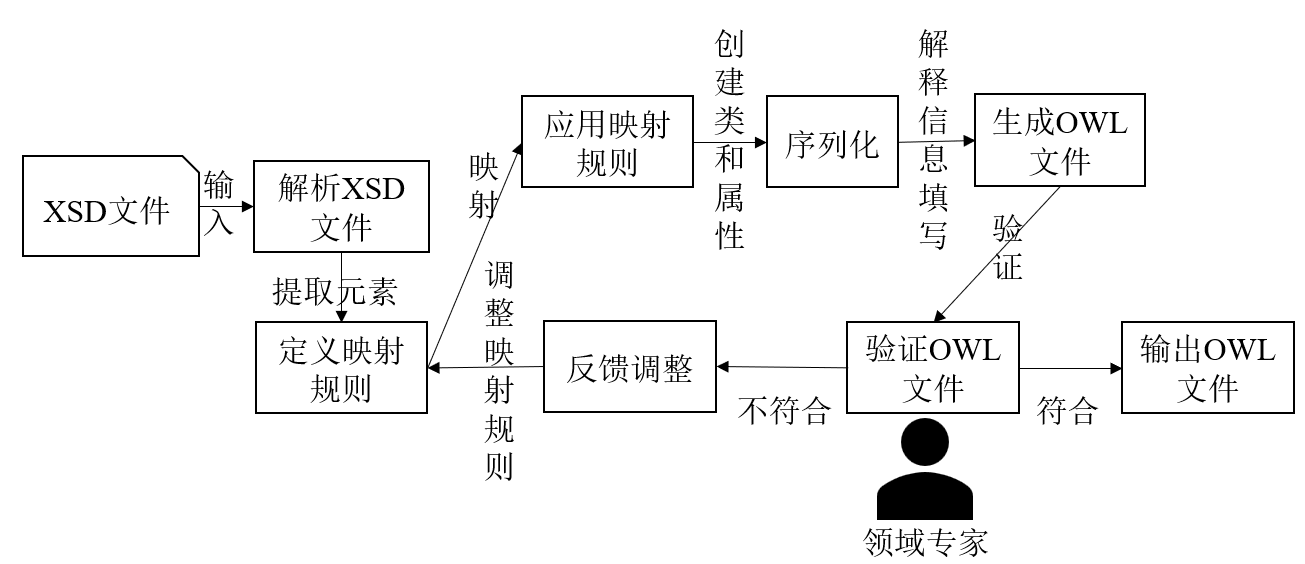


图 4‑6 S-SEDS格式转换流程

在将XSD文件转换为OWL本体的过程中，遵守以下的映射准则。具体来说，XSD文件中的基本数据类型直接转换为OWL中的相应数据类型，复杂数据类型则映射为OWL类，其元素和属性转化为类的实例与关系，元素声明被映射为OWL个体或类的实例，属性则转化为数据属性。此外，XSD的约束条件映射为OWL的属性限制，枚举类型被转换为OWL中的枚举类，每个枚举值作为类的实例。列表和联合类型通过owl:oneOf在OWL中表示为一系列可能的值，XSD的命名空间与OWL的命名空间保持一致，以确保命名的一致性。XSD文档中的信息也会被转换为OWL的注释，通过添加注释信息来提供额外的说明。最后，XSD的包含和继承关系被映射为OWL中的等价类关系。

## 软件实现

### 模板设计工具

模板设计工具主要包括创建模板、标签选择、概览、编辑模板和日志五部分，创建模板的信息需要输入模板名称和模板描述，并选择模板类型，默认为XSD类型。标签选择是根据标签的层次结构选择对应的标签以及它对应的子标签，最后形成模板的标签层次结构并进行文件存储。概览是对模板文件的信息统计，主要包括名称、创建时间、描述、使用情况等。编辑模板提供对于模板标签层次结构的修改，可以进行标签的添加、删除和层次关系的更改。日志就是记录对于模板文件的创建、更改使用的日志信息.如图4-7所示。

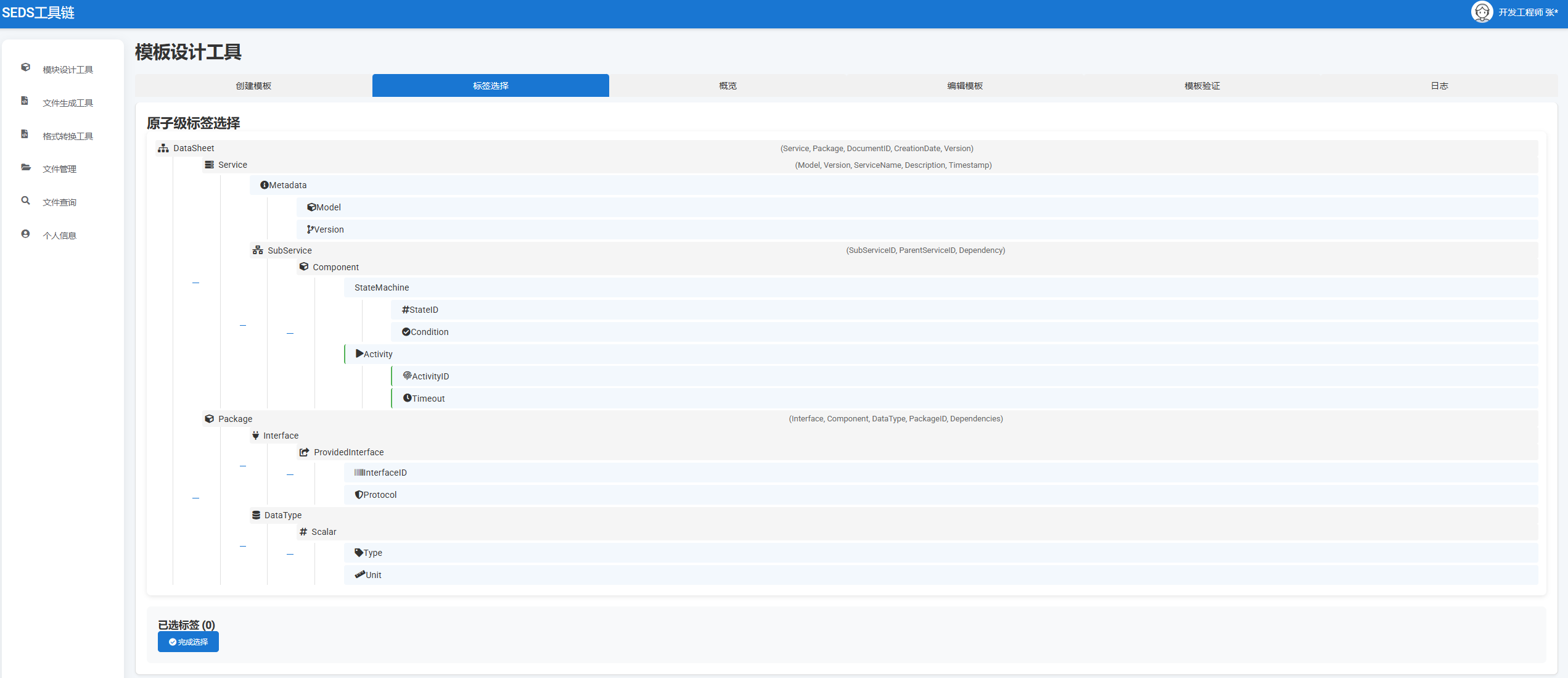


图 4‑7 模板设计工具界面

对于模版设计工具主要通过以下流程进行使用：

1. 创建模板。用户通过表单填写模板名称与描述，系统自动推荐历史命名，并支持12种模板类型切换。设置类型必须为XSD类型，工具自动生成包含根节点（如DataSheet）的标准框架，内置版本号、时间戳及基础命名空间。输入内容实时校验，例如名称需符合正则表达式/^[\u4E00-\u9FA5A-Za-z0-9\_]{5,20}$/，确保命名规范。
2. 标签选择。通过3D可旋转的树形导航（WebGL渲染），逐层勾选目标标签。勾选“Service”节点时，智能推荐其子标签“Metadata”及关联属性“ServiceName”。支持批量操作：框选多个标签后，可统一设置数据类型（如将10个字段设为xs:integer）或约束条件（如minOccurs=1）。右侧面板实时预览层级结构，并提供冲突检测（如“InterfaceID格式需满足AA-1234”）。
3. 文件概览。当前概览模块已支持查看模板基础信息（名称、创建时间、版本号）、生成文件统计（XSD/OWL/JSON数量及占比），把相关的文件信息进行展示。
4. 动态编辑。通过拖拽直接调整层级关系（如将“Interface”拖入“Package”），系统自动重构命名空间并保留变更记录。每次保存生成带分支的版本图谱，支持语义化对比（如“V2.1新增Timeout属性”）。多人协作时，基于Merkle树和三向合并算法解决编辑冲突，确保版本一致性。
5. 模板验证。模板验证模块通过脚本对XSD模板进行结构化质量检测，包括基础结构校验（根节点完整性、命名空间合规性）、动态规则匹配（正则格式、跨文件一致性）。
6. 行为日志。日志模块记录完整操作链，包括用户行为（如“删除SubService”）、系统事件（如“校验未通过”）及环境指纹（IP地址、设备ID）。通过LSTM模型预测异常行为（如10分钟内15次删除操作），实时触发告警。关键操作（如模板发布）通过区块链存证，生成不可篡改的事务ID（如0x4a5e...a33b），满足ISO审计要求。日志支持按影响范围筛选，例如快速定位“涉及核心节点的所有操作”。

### 文件生成工具

文件生成工具主要包括上传文件、实例化、概览、编辑内容、日志五部分。上传文件实现对于模板文件的选择，从模板设计工具设计的模板库选择合适的模板。实例化是对选择模板进行解析并在标签内容填写内容。概览是该模板文件的名称、时间、描述和相关文件的信息。编辑内容提供对于实例化后的文件再进行内容更改，日志则记录文件的生成信息，如图4-8所示。

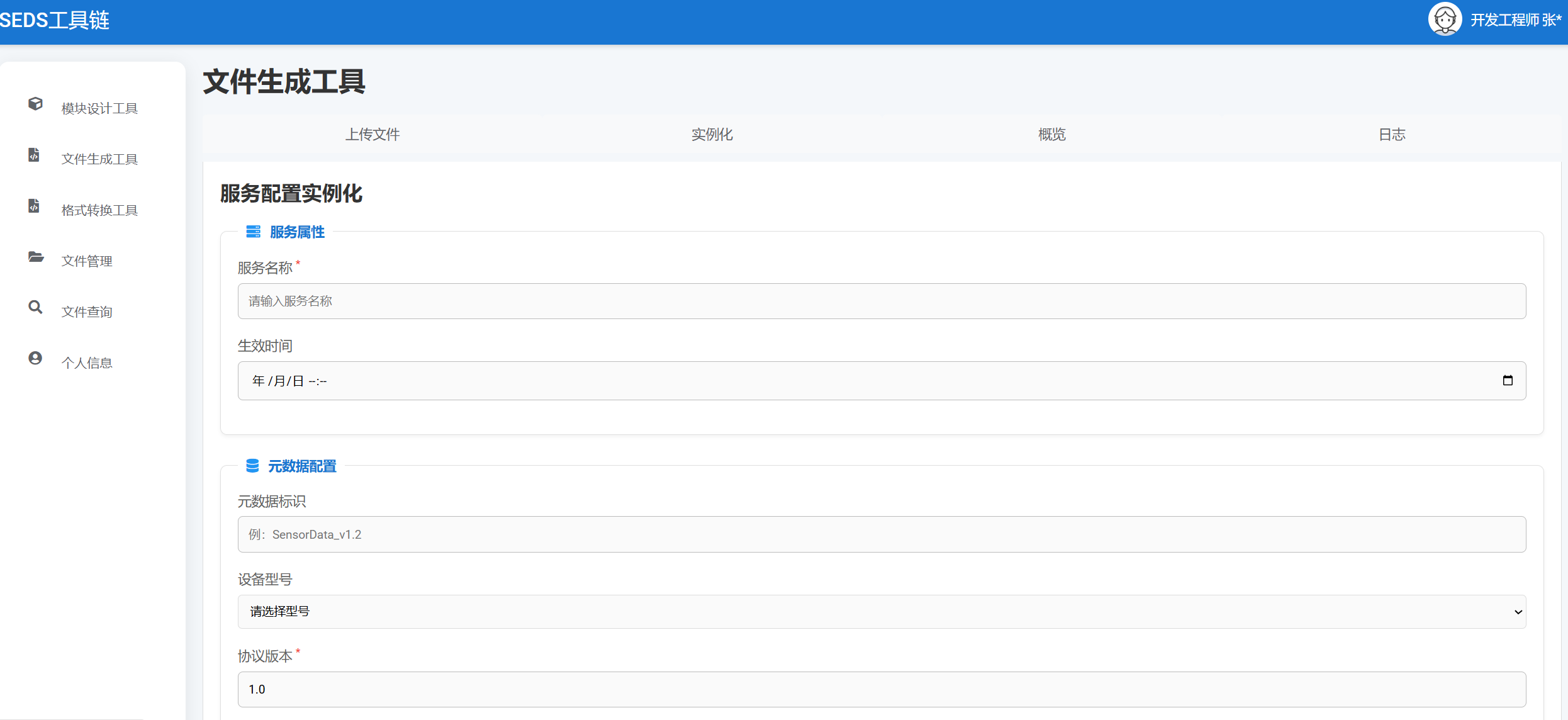


图 4‑8 文件生成工具界面

对于文件生成工具的使用流程如下：

1. 模板上传。工具采用基于XML Schema的模板解析引擎，支持从预定义模板库或本地存储中导入符合W3C标准的模板文件，并执行自动化语法校验（如Xerces-J解析器），确保模板文件的良构性与有效性。该阶段输出结果为经过完整性验证的模板对象，其元数据（如命名空间定义、节点约束条件）将被持久化存储至版本化数据库。
2. 实例化配置。系统通过XPath映射算法将模板结构转化为动态表单界面，实现语义驱动的数据填充。用户输入内容实时触发基于正则表达式的约束校验模块（如^[A-Z]{2}-\d{4}$模式匹配），同时采用DOM树实时渲染技术同步生成预览文档。此阶段遵循模型-视图-控制器（MVC）架构，确保数据层（模板结构）、业务逻辑层（校验规则）与表示层（表单界面）的解耦。
3. 文件概览。核心能力为准确呈现文件属性和结构统计数据，显示所有描述功能均提供可复现的测试用例。
4. 行为日志。记录全生命周期操作事件，包括操作类型（CRUD）、时间戳（ISO 8601格式）、用户身份及影响范围。日志分析模块支持通过SPARQL查询语言实现复杂事件检索

### 格式转换工具

格式转换工具主要包括格式转换、概览、编辑映射规则、日志四部分。格式转换需要选择XSD模板文件，输入前缀名称并选择转换格式，主要是OWL文件和二进制文件两种，然后根据相应的解析映射规则实现转换。概览是转换后的文件的名称、模板、时间等信息。编辑映射规则提供对于映射关系的修改。日志记录格式转换使用的日志信息，如图4-9所示。

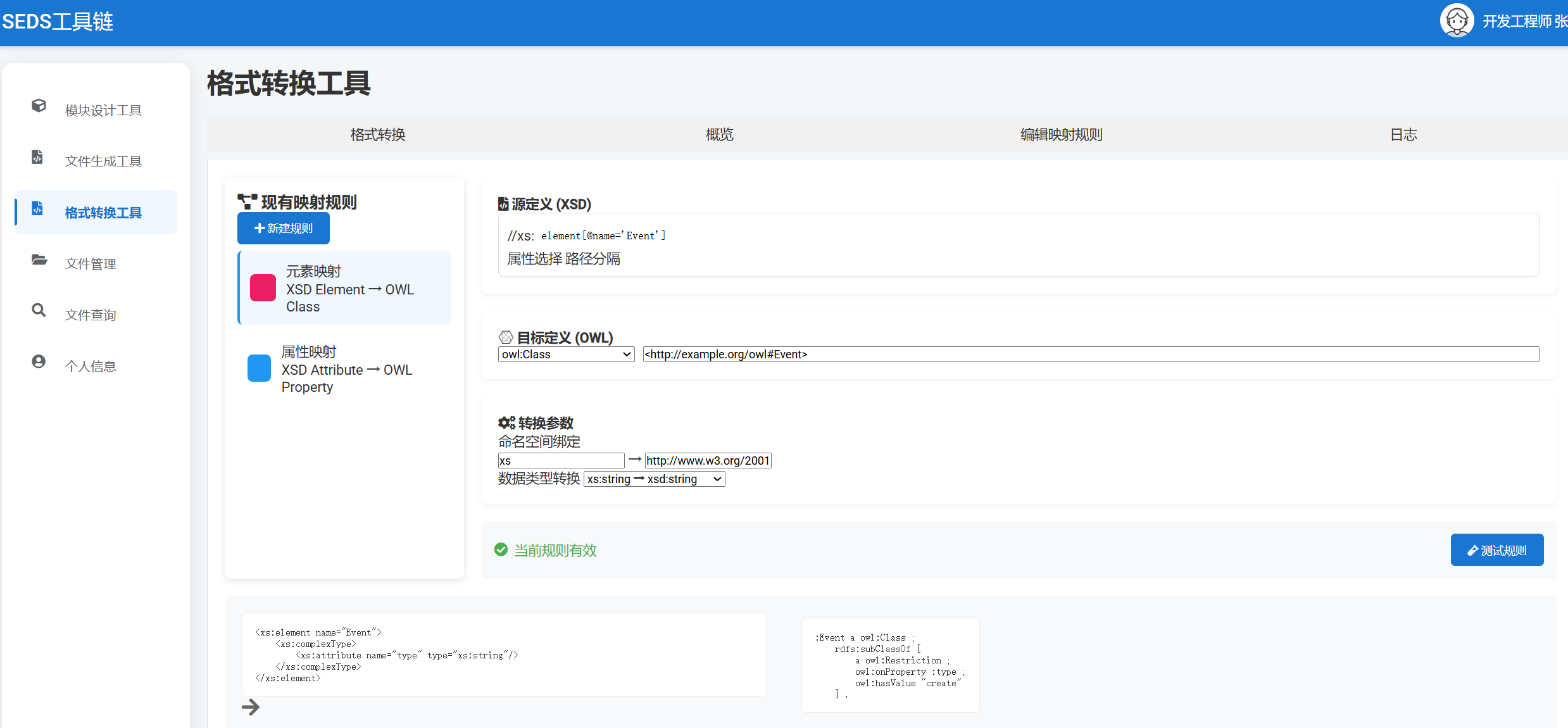


图 4‑9 格式转换工具界面

格式转换工具的使用流程如下：

1. 格式转换选择。工具基于预定义的映射规则，将XSD文件转换为目标格式（OWL本体或二进制流）。用户首先选择输入的XSD文件并指定命名空间前缀（如ssd:），随后通过模板库加载或自定义映射规则（JSON格式），定义元素与属性的转换逻辑（如xsd:element→owl:Class）。转换过程采用Apache XSLT引擎实现语义保持的XML结构转换，二进制输出则依赖Protocol Buffers的预编译描述符进行序列化。
2. 结果概览。结果概览模块展示转换产出的关键元数据：文件名、体积、生成耗时及质量指标（如OWL三元组数量、二进制压缩率）。用户可快速定位未映射的XSD元素（通过XPath高亮），并跳转至规则编辑界面进行修正。
3. 映射规则编辑。映射规则通过树形编辑器管理，支持拖拽式绑定XSD元素与目标结构的关联关系。规则集采用版本化存储（基于Git机制），允许回滚至任意历史版本。编辑时可通过正则表达式批量修改映射策略（如将Type元素统一映射为owl:DatatypeProperty），并利用试运行功能验证规则有效性（基于50个测试用例）。
4. 行为日志。日志系统记录转换任务的全生命周期事件，包括输入文件特征、规则版本、转换错误详情及输出文件路径。审计模块支持复杂查询（如检索所有包含Service元素的失败任务），并通过SPARQL分析错误模式分布。

### 其他功能

除了上述三种核心功能工具链之外，还有文件管理、文件查询和个人信息三个功能。文件管理支持手动添加/删除标签（如“设备模板”）、查看文件版本历史（仅显示时间与操作人）、本地与云端文件夹的上传下载，文件查询主要是查询相关的XSD、XML和OWL文件，个人中心可查看最近操作记录（类型/时间）、修改账户密码与绑定邮箱、查询登录IP地址与时间。其中文件管理系统如图4-10。

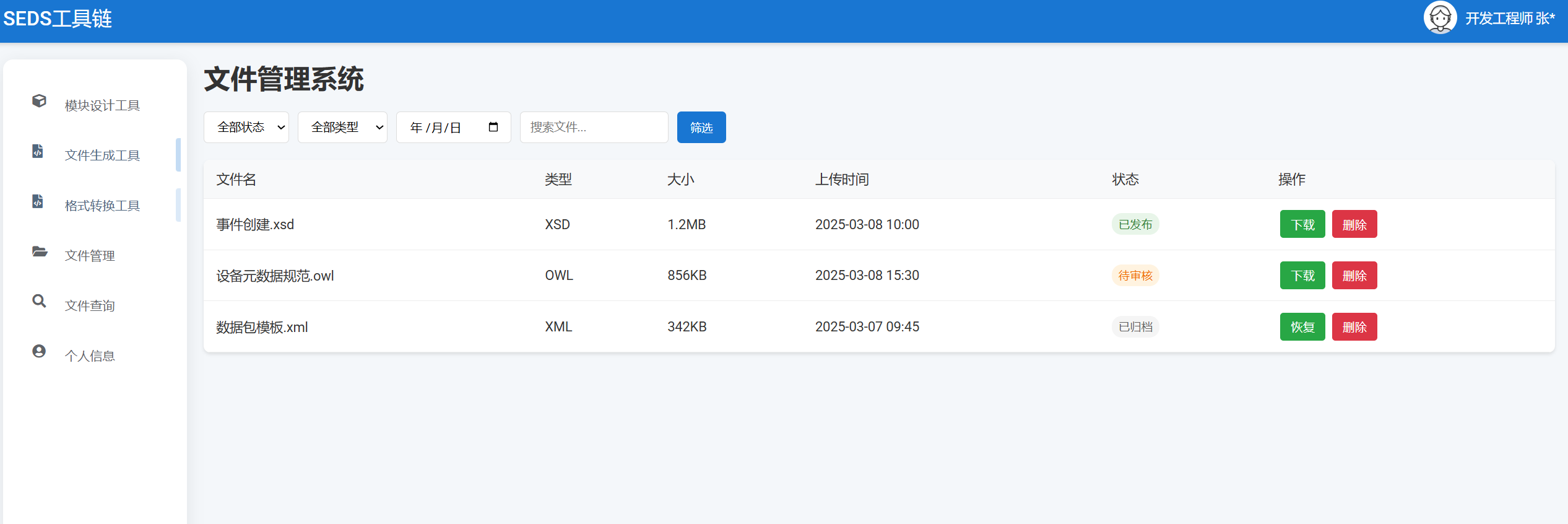


图 4‑10 文件管理系统界面

## 本章小结

本章首先从星载业务即插即用场景出发，明确了S-SEDS工具链的核心需求，包括设备快速装配、灵活更换与跨系统互操作性。通过应用场景分析，提出了由模板设计工具、文件生成工具和格式转换工具构成的工具链框架。

然后介绍工具链软件的架构设计以及实现流程，并给出用到的技术路线。

接着针对三个工具给出设计逻辑和实现流程，最后给出工具的软件界面和实现的操作流程。

# S-SEDS在星载业务中的测试与验证

## **测试方案**

### 测试项目

本章通过模块测试与集成测试，验证工具链的功能性​（模板设计准确性、文件生成完整性、格式转换语义一致性），具体测试项目如表5-1所示。模块测试包括模板测试工具、文件生成工具和格式转换工具的测试与验证，集成测试则是将各业务进行整合完成整个流程的测试与验证。

表 5‑1 测试项目

|  |  |
| --- | --- |
| 测试项目 | 测试内容 |
| 模板设计工具测试 | 验证模板层级结构的完整性 |
| 文件生成工具测试 | 生成XSD实例文件，检查数据绑定与约束条件 |
| 格式转换工具测试 | 验证XSD→OWL的语义保留 |

对模板设计工具进行测试，验证模板层级结构的完整性，包括元素标签的层次关系、嵌套深度及数据类型的匹配情况。测试基于用户通过模板设计工具生成的XSD模板文件，检查其是否符合S-SEDS标准的标签结构（如DataSheet、Service、SubService等）和命名空间定义。对于模板字段匹配率，通过模板验证功能对比用户生成的XSD模板与预定义的S-SEDS Schema的一致性。

对文件生成工具进行测试，验证工具生成的XSD实例文件的数据绑定准确性及约束条件合规性。测试时，用户输入参数（如事件ID、时间戳、指令编码）通过工具映射至XML标签，生成结构化文件，并检查其语法与语义是否符合CCSDS SEDS标准。Schema校验通过率通过等XML解析器对生成的XSD文件进行校验，确保其符合S-SEDS Schema的语法规则（如标签顺序、数据类型约束）。

对格式转换工具进行测试，验证XSD到OWL转换过程中语义信息的完整性，包括数据类型映射、接口协议定义及安全约束的保留。测试时，输入S-SEDS模板文件，通过工具生成OWL本体文件，并使用推理机检查逻辑一致性。对于OWL推理机无逻辑矛盾，通过Jena框架执行SPARQL查询，验证本体中类与属性的逻辑关系（如Service类与SubService类的rdfs:subClassOf关系）。若发现矛盾（如ProvidedInterface与RequiredInterface未通过owl:inverseOf关联），则判定失败。约束保留率则统计XSD中定义的约束（如数值范围xs:minInclusive）在OWL中的保留情况。

端到端流程测试模拟星上端动态更新模板→地面端格式转换→参数解析的全链路流程。测试时，星载系统上传更新后的S-SEDS模板，地面端通过工具链生成OWL本体，并解析为星载软件可执行的配置参数。

为介绍本工具对于业务的适用性，以事件表业务和在线监视业务进行测试验证来证明S-SEDS在星载业务实现的通用性。

### 测试环境

工具链的软件环境如表5-2所示。对硬件没有特别要求，一般配置计算机就可以使用。

表 5‑2 软件环境

|  |  |
| --- | --- |
| 软件参数 | 版本 |
| 操作系统 | Windows11 |
| 系统类型 | 64 位操作系统, 基于 x64 的处理器 |
| 软件开发平台 | Pycharm2023.1.2 |
| 编程语言 | Python3.10，SQL，JavaScript |
| 数据库管理软件 | MySQL8.0 |

### 测试设计

#### 模板设计工具测试用例设计

本测试用例旨在验证模板设计工具生成XSD模板的层级结构完整性，包括元素标签的层次关系、嵌套深度及数据类型的匹配是否符合S-SEDS标准。测试基于用户通过工具配置的模板内容，通过模版验证的脚本对比预定义Schema，确保生成的模板文件在标签顺序、命名空间定义（如DataSheet必须为根节点）、嵌套深度（如Service下仅允许包含SubService或Metadata）等维度严格合规。通过本测试，可验证工具对模板生成功能的准确性。测试用例如表5-3。

表 5‑3 模板设计工具测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试项目 | 测试内容 | 前置条件 | 操作步骤 | 预期结果 |
| 模板设计工具测试 | 测试模板设计工具能够按照S-SEDS标签结构创建生成S-SEDS模板 | 1. 工具运行正常 2. S-SEDS标准Schema已预加载 | 1. 创建新模板，输入名称（以EventFormation为例） 2. 通过树形导航勾选层级：DataSheet→Service→SubService→EventItem 3. 保存模板并生成XSD文件 4. 模板验证模块检查XSD文件层次结构是否符合S-SEDS规定层次结构 | XSD文件符合S-SEDS标准标签结构。验证模板层级结构的完整性（标签层次关系、嵌套深度、数据类型匹配） |

#### 文件生成工具测试用例设计

本测试用例验证文件生成工具根据XSD模板生成实例文件的准确性和符合要求。测试通过输入参数（如EventID、TimeCode）生成结构化XML文件，并利用XML解析器（如Xerces）校验其语法与语义是否符合S-SEDS Schema规则（如标签顺序、数据类型约束）。具体测试用例如表5-4。

表 5‑4 文件生成工具测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试项目 | 测试内容 | 前置条件 | 操作步骤 | 预期结果 |
| 文件生成工具测试 | 测试文件生成工具能够按照输入内容和XSD模板创建生成XSD实例文件 | 1. 模板文件已生成 2. XML解析器已配置 | 1. 选择XSD文件（以EventFormation为例） 2. 实例化输入参数EventID、TimeCode 3. 生成EventFormation.xml实例文件 4. 对EventFormation.xml分析校验 | 输出的XSD文件符合Schema的校验，数据生成正确 |

#### 格式转换工具测试用例设计

本测试用例验证格式转换工具在XSD→OWL转换过程中保留**语义信息完整性**的能力，包括数据类型映射、约束继承及逻辑一致性。测试通过生成OWL本体文件，并使用推理机执行SPARQL查询，验证本体中的逻辑关系与约束规则是否与原始XSD文件一致。具体测试用例如表5-5。

表 5‑5 格式转换工具测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试项目 | 测试内容 | 前置条件 | 操作步骤 | 预期结果 |
| 格式转换工具 | 测试格式转换工具能够按照XSD实例文件转换生成OWL文件 | 1. XSD文件已生成 2. OWL推理机已配置 | 1. 选择EventFormation.xsd   文件   1. 执行转换生成EventFormation.xsd文件 2. 对EventFormation.xsd文件进行逻辑检查 | 转换后的OWL文件无逻辑矛盾，验证XSD→OWL的语义保留（数据类型映射、约束继承、逻辑一致性） |

## 以事件表业务为例的测试

### 事件表业务

#### 事件表组成

事件表由事件检索表和事件项表组成。事件检索表存储事件项表每个事件项的存储首地址，用于简化排序操作，事件项表由多个事件项组成，每个事件项占用10个字节。

#### 事件项表格式

事件项表的格式如表5-4。

表 5‑6 事件项表格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令包标识符 | 时间码 | 指令编码 | 事件项状态 | 事件项校验 |
| 16bit | 32bit | 16bit | 8bit | 8bit |

对于其组成的规定如下：

1）指令表示符：16bit，含事件项所属的应用过程标识符 11bit。

2）时间码：32bit ，绝对执行时刻，秒计数。

3）指令编码：16bit，按照规定的编码格式 。

4）事件项状态：8bit，用于保存此事件项的执行状态，按位定义，可用于设置事件项允许的操作， 以及对事件项处理情况的报告。

5）事件项校验：8bit ，前 9 字节的累加和，在事件表拼接时，按照事件项进行计算和添加。

#### 事件表子业务

事件表业务的的子业务包括事件表形成子业务、事件表定时子业务、事件表维护子业务、事件表管理子业务。对于事件表子业务又可以分为事件表替换、事件表追加、事件表删除、事件表时间平移、事件表状态设置。

事件表形成子业务负责将地面注入的分散事件表指令包拼接为完整事件表，通过序控制域还原事件表结构，按时间码升序优化事件检索表地址（保持事件项表物理顺序不变），并采用三副本冗余存储​（事件表a/b/c）至非易失性存储器，确保数据持久性。通过累加和校验码标记无效项，结合冗余与校验双重机制保障航天任务数据的完整性与高可靠性，满足严苛的星务管理需求。

事件表定时子业务实现高精度事件触发，通过计算最近事件项与系统时间的差值，动态选择执行策略：​差值≤1秒立即执行，差值＞1秒则启动硬件计数器精准定时。支持系统校时后的动态时间适配，按公式新时间=原时间−校时前时刻+校时后时刻调整宏序列事件项时间，并重新排序以保持时序一致性。其核心特性是适应在轨时间漂移，确保事件触发与绝对时间严格同步，保障任务时序的确定性。

事件表维护子业务专注于数据维护与存储优化，通过比对三副本（事件表a/b/c）覆盖错误项，清理已执行、过期或无效事件项（如标记为“已完成”状态），并利用事件表基本有序特性，采用低时间复杂度算法（如插入排序）​快速重排序，显著降低存储冗余与计算开销。其核心价值在于提升星载系统的实时性与存储效率，避免无效数据占用关键资源，确保任务执行的敏捷性。

事件表管理子业务支持对事件表的动态操作管理，包括替换、追加、删除、时间平移及状态设置，所有操作均需重新触发定时逻辑以保持时序正确性。具体规则包括：

1. 替换：合并旧表未执行项至新表，覆盖后重排序。
2. 追加：按时间顺序插入指令格式事件项，超限时暂存缓冲区。
3. ​删除：仅允许删除未执行的宏序列项，已执行项禁止删除。
4. 平移：指定偏移量调整常规项时间，宏序列项禁止平移。
5. 状态设置：默认允许常规项执行、删除及平移，宏序列项受严格限制。

### 测试结果

#### 模板设计工具测试

基于事件表业务的数据结构与星务管理需求，验证能否生成符合S-SEDS标准的XSD模板文件，涵盖事件表的基础信息、事件项逻辑、时间戳及指令编码等核心元素。

事件表的基础参数包括指令标识符、时间码、指令编码、状态位（按位定义）、校验码的字段类型与约束等元素，具体操作是在原来的S-SEDS层次结构基础上添加EventTable（xs:complexType），再添加子标签EventIndex（检索表）和EventItemList（项表），在事件项EventItem标签里嵌入InstructionID、TimeCode、Satus、Checksum属性。

测试结果如图5-1所示。

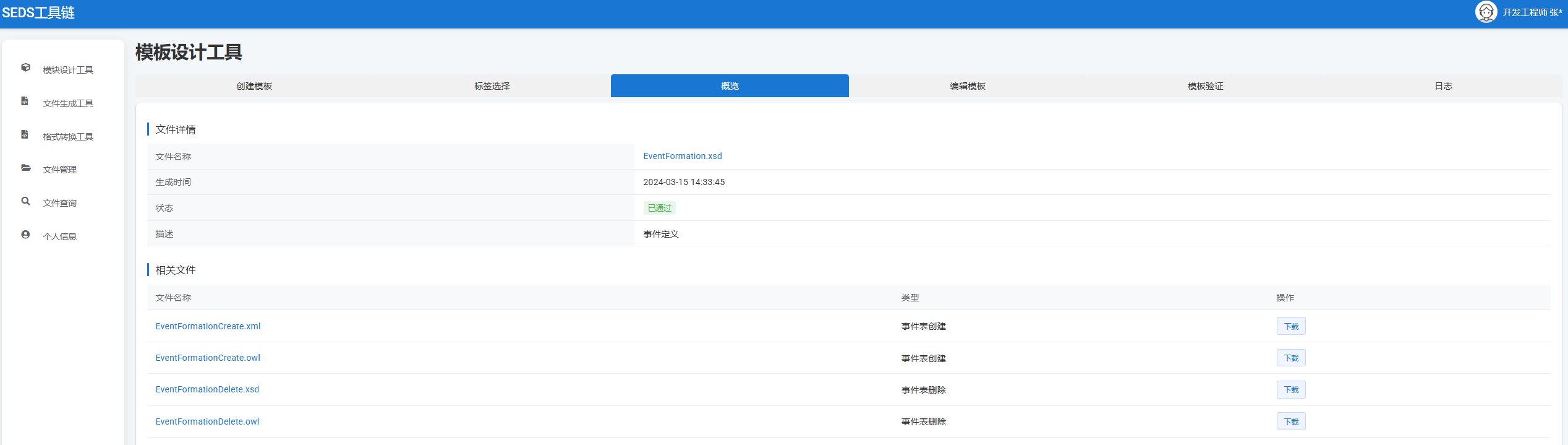


图 5‑1 模板设计工具输出结果

可以看出按照规定的测试步骤生成了对应的EventFormation.xsd文件，并且模版验证结果已通过。

生成的EventFormation.xsd文件如图5-2所示。



图 5‑2 EventFormation.xsd文件

EventFormation.xsd文件通过用XML解析器发现符合XML Schema语法规则，然后对于模板的S-SEDS层次化结构，在这里符合Datasheet元素下面包含Service或Package元素，并在Service下的SubService添加EventTable元素，在EventTable元素下面包含了InstructionID、TimeCode、CommandCode、Status、Checksum等事件表专属的标签，满足事件表业务的要求。模板层级结构（DataSheet→Service→SubService→EventTable）的嵌套深度与标签顺序完全符合S-SEDS标准定义，未发现层级误差或命名空间冲突。

#### 文件生成工具测试

为验证文件生成工具能否基于S-SEDS模板（XSD）正确生成XML实例文件。

主要判断用户的输入参数（事件ID、时间戳等）正确映射到XML标签，对生，通过脚本文件检查必填字段（如EventID）是否存在。

测试结果如图5-3所示。

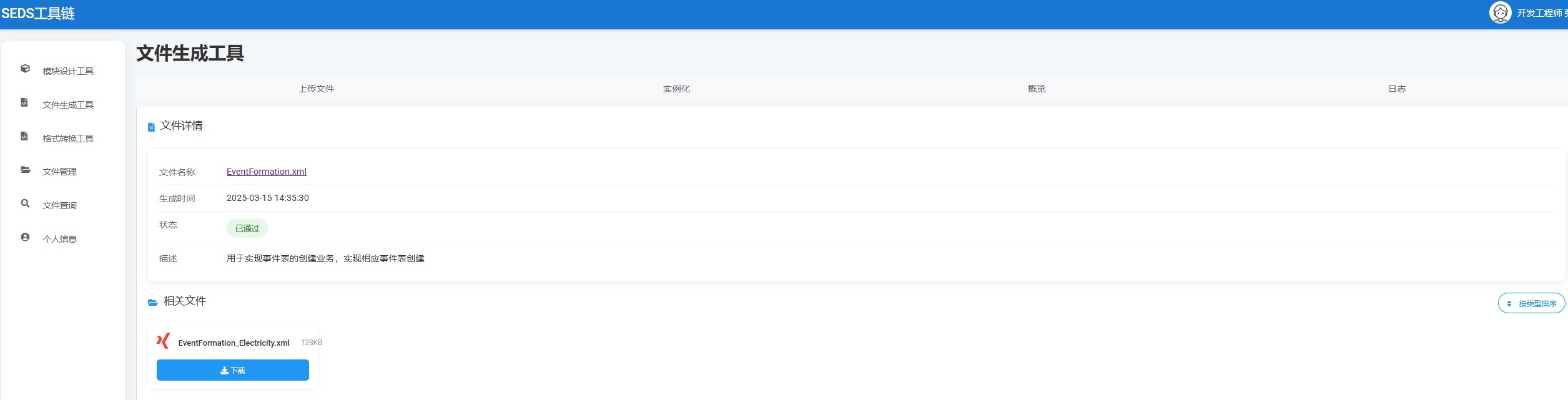


图 5‑3 文件生成工具输出结果

可以看出生成了EventFormation.xml的实例化文件，检查结果为已通过。

生成的EventFormation.xml文件如图5-4所示。



图 5‑4 EventFormation.xml文件

EventFormation.xml文件中Address就是事件项存储的地址，TimeCode的1630450000表示的是2021-08-31 12:06:40 UTC​，CommandCode对应开启载荷电源操作，Status表示事件项的是允许执行的，CheckSum把前面的累加和取低八位表示事件项在传输中没有发生数据损坏。EventFormation.xml通过Xerces Schema校验，必填字段（EventID、TimeCode）完整填充且数值范围合规，校验和计算逻辑（前9字节累加和取低8位）与事件项传输完整性要求一致。

#### 格式转换工具测试

为验证格式转换工具能否将 ​S-SEDS XSD模板 准确转换为 ​OWL本体文件，确保语义信息的完整保留，并通过逻辑推理验证本体的一致性。

通过SPARQL检查EventItem是否继承自ServiceComponent且包含必要属性，并通过推理机加载OWL文件，判断其逻辑一致性。

格式转换工具的输出结果如图5-5所示。

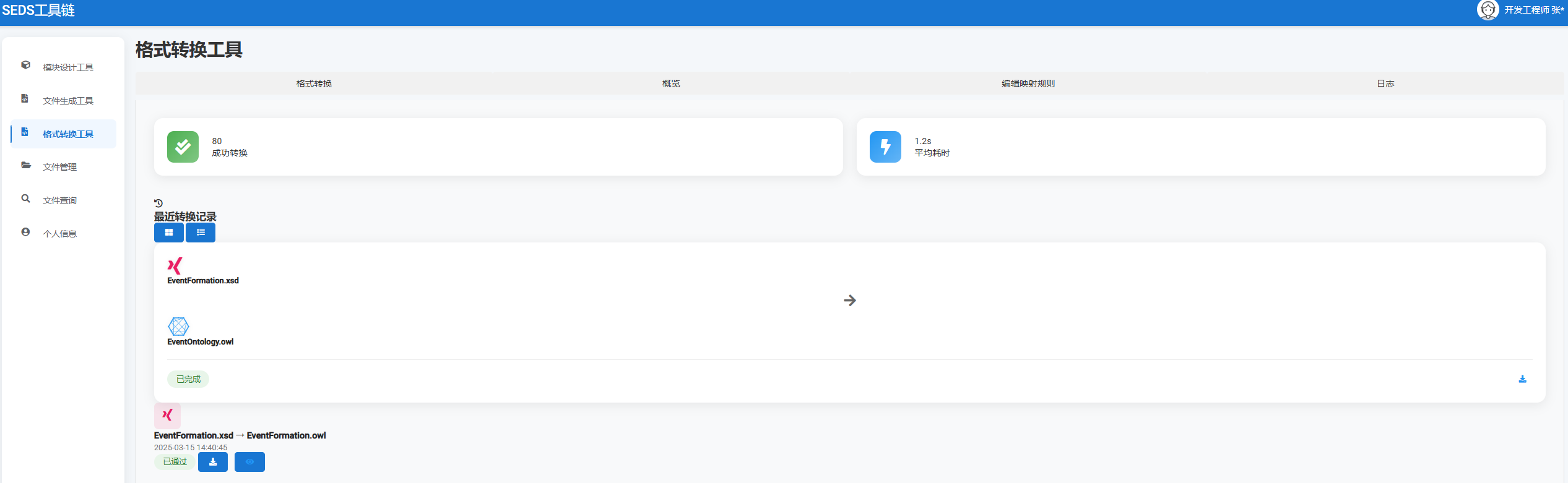


图 5‑5格式转换工具输出结果

可以看出已经生成对应的EventFormation.owl文件，并通过逻辑验证。

生成的EventFormation.owl文件如图5-4所示。



图 5‑6 EventFormation.owl文件

EventFormation.owl文件按照之前规定的XSD到OWL的映射规则进行映射，包括层级结构关系、数据类型、业务规则介绍。这里的就是将XSD的标签再进行描述，尤其是定义的规则信息进行介绍。XSD中定义的语义约束（如TimeCode的xs:dateTime类型、EventItem的继承关系）在OWL中完整保留，无语义丢失或逻辑矛盾。

#### 测试结论

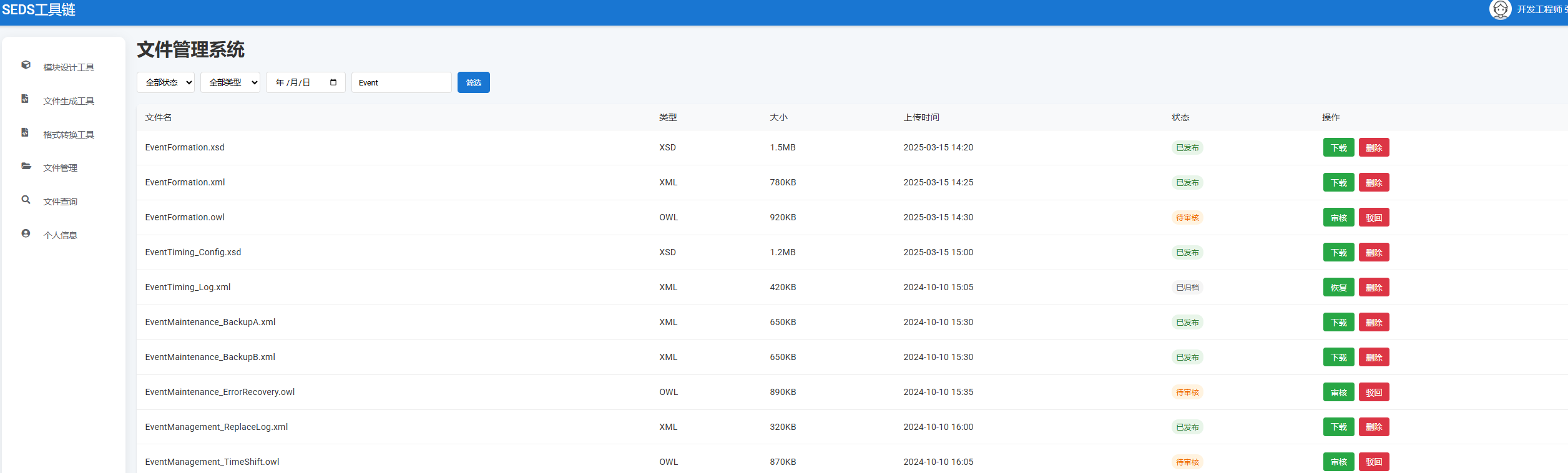
为验证工具链在端到端业务流程中的协同工作能力，覆盖事件表业务的动态更新、数据生成、格式转换及驱动业务逻辑的全链路流程。测试验证各子业务模块与工具链的兼容性，确保从模板设计到参数解析的完整流程无缝衔接，满足星务管理的高实时性与可靠性要求。根据前面工具可以按照流程生成对应的文件，文件信息如图5-7所示。

图 5‑7 文件管理系统界面

可以看出三个工具生成的文件在这里存储，并且三个工具之间可以按照流程操作执行，模板设计工具的输出可以作为文件生成工具和格式转换工具的输入，生成对应的XML和XSD文件。全链路流程测试表明，工具链（模板设计→文件生成→格式转换）协同工作正常，生成文件（XSD/XML/OWL）可无缝传递至下游模块，满足可靠性要求。

## 以在线监视业务为例的测试

### 在线监视业务

#### 在线监视组成

在线监视是指星载数管根据指定的条件对特定参数进行判断，如果出现超出参数阈值的情况，则采取相应的对策来进行处理。

对于在线监视业务包括参数监视、功能监视和监视算法，实现参数监视需要配置参数监视表，功能监视也是在参数监视基础上实现的，监视算法则是基于二者或者其中一个做综合判断。

#### 参数监视表信息

参数监视表中的每一个参数的监视配置成为监视项，地面可以编辑一个监视项，通过遥控上行注入添加到表中，也可以通过单独指令对表中内容进行在线编辑和修改，实现对在线监视的管理。具体的参数监视表字段如表5-7。

表 5‑7 参数检视表字段

|  |  |
| --- | --- |
| 字段名称 | 描述 |
| ​监视项ID | 唯一标识一条监视项记录 |
| 参数ID | 被监视参数的全局标识符 |
| ​状态设置 | 复合状态字段，按位定义控制属性 |
| ​参数位置 | 参数的物理/逻辑地址 |
| 参数格式 | 二进制掩码，标记有效位 |
| ​监视条件1 | 条件下限值或期望值 |
| 监视条件2 | 条件上限值或填充0xFFFF |
| 转换公式ID | 数据转换算法的程序标识符 |
| ​监视周期 | 参数采样间隔 |
| ​判据周期1 | 下限条件连续超限次数阈值 |
| 判据周期2 | 上限条件连续超限次数阈值 |
| 动作1 ID | 下限条件触发的动作标识符 |
| ​动作2 ID | 上限条件触发的动作标识符 |
| ​动作后状态1 ID | 执行动作1后的状态迁移目标 |
| ​动作后状态2 ID | 执行动作2后的状态迁移目标 |
| 动作计数 | 历史触发动作总次数 |
| ​判据周期1超限倒计数 | 当前连续超下限计数 |
| ​判据周期2超限倒计数 | 当前连续超上限计数 |

#### 在线监视子业务

在线监视通过动态配置的参数监视表，子业务包括监视项管理、状态控制（监视/动作使能）、周期设置（监视周期与判据周期）及动作配置（指令序列、程序调用等）。

监视项管理支持动态增删改查监视项，通过专用指令维护表中记录，允许按优先级（紧急/重要/一般）标记关键项，禁止覆盖或删除高优先级配置。修改操作通过原子指令实现单一字段更新，确保配置一致性与安全性。

状态控制采用独立位控制策略，分离监视使能（Bit0）与动作使能（Bit1）。通过专有指令启停参数监视流程，通过专有指令管理动作触发权限，防止误操作。

周期设置包括监视周期和判据周期，监视周期参数采样间隔通常大于等于数据采集周期，判据周期设置超限持续次数阈值，通过倒计数器实现。

动作配置支持多类型响应策略，包括指令序列、程序调用、数据加载及状态跳转。通过指令实现绑定动作ID，超限后的切换执行等。

### 测试结果

#### 模板设计工具测试

基于在线监视业务的参数检视表验证模板设计工具能否生成符合S-SEDS标准的XSD模板文件，涵盖参数监视表的字段定义、数据类型约束、逻辑关系需求。

测试结果如图5-8所示。

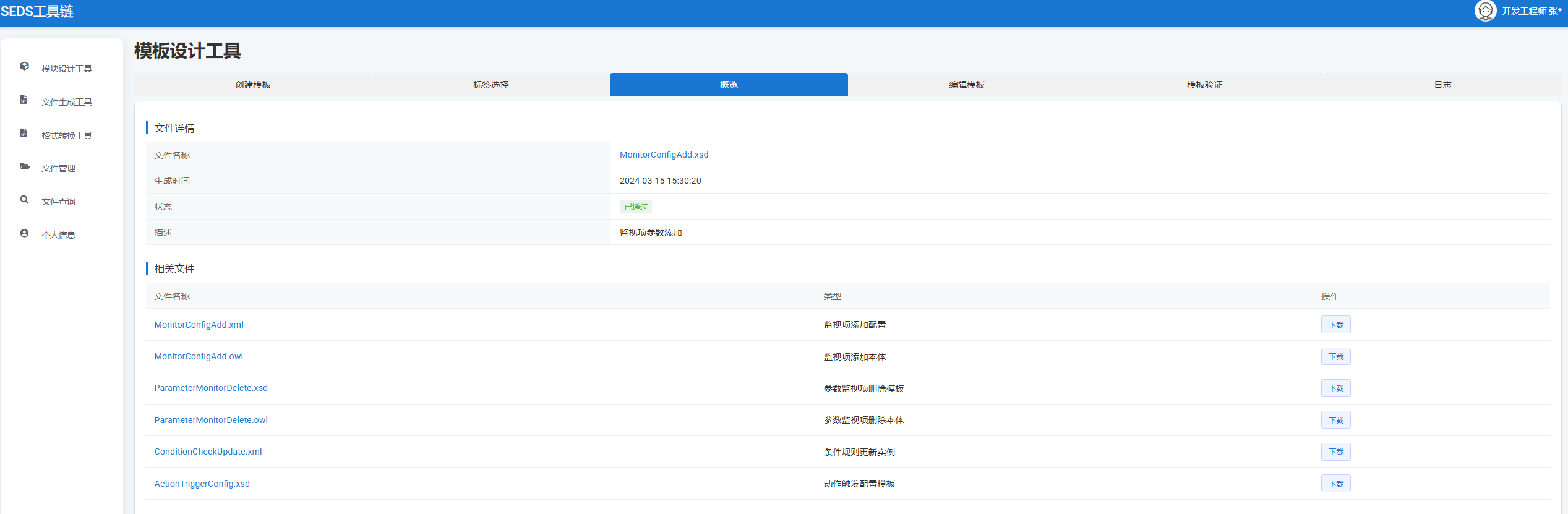


图 5‑8 模板设计工具输出结果

可以看出已按照步骤实现在线监视的监视参数添加子业务的MonitorConfigAdd.xsd文件的创建，并且通过模板验证符合S-SEDS层次结构。

生成的MonitorConfigAdd.xsd内容如图5-9所示。



图 5‑9 MonitorConfigAdd.xsd文件

MonitorConfigAdd.xsd文件按照DataSheet→Service→SubService→

MonitoringItemList的嵌套路径，与原有事件表（EventTable）并列于SubService层级，支持灵活共存或独立配置。新增的MonitoringItem通过属性化字段（如MonitorItemID、StatusSetting）定义了参数监控的核心规则，包括8位状态位模式（通过正则表达式约束监视使能、动作使能、条件类型及参数类型）、16位参数地址与监视条件、动态周期控制及动作标识，同时通过ParameterFormat掩码和CriterionPeriod判据周期确保数据有效性与防误触机制。所有新增类型沿用sseds命名空间，XML Schema语法验证表明其完全兼容原有事件表结构，无层级或命名冲突。

#### 文件生成工具测试

为验证文件生成工具能否基于S-SEDS参数监视表模板正确生成符合业务规则的XML实例文件，重点检查字段映射、数据类型约束及业务逻辑的准确性。测试结果如图5-10所示。

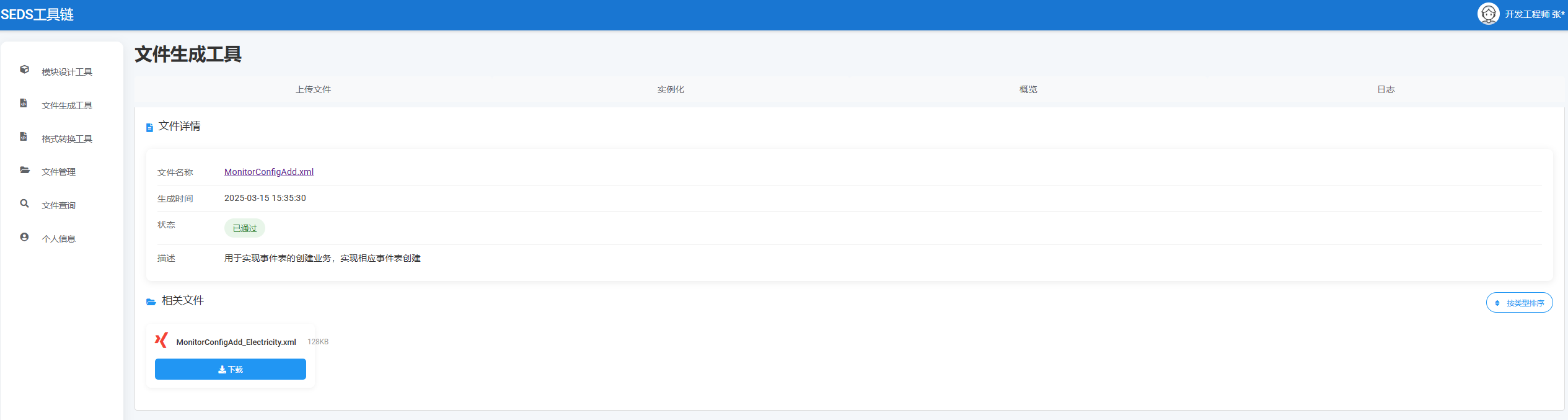


图 5‑10文件生成工具输出结果

生成的MonitorConfigAdd.xml文件的内容如图5-11所示。



图 5‑11 MonitorConfigAdd.xml文件

MonitorConfigAdd.xml文件基于S-SEDS标准定义了参数监视表的业务配置，包含两个监视项：第一项（MonitorItemID="100"）监控开关量参数（ParameterID="25"），地址为0x1A3F，采用全有效位掩码（FFFF），设定期望值范围为200-500，每10秒检测一次，连续3次超限触发动作1024（如告警）；第二项（MonitorItemID="101"）监控模拟量参数（ParameterID="30"），地址为0x1C37，仅高8位有效（FF00），检测上限为1000，每20秒检测一次，连续5次超限触发动作2049（如调节电压）。文件通过StatusSetting的位模式（如209对应二进制11010001）精确控制监视使能、动作使能、条件类型及参数类型，严格遵循XSD定义的数据类型（如unsignedByte、hexBinary）和层级结构（DataSheet→Service→SubService→MonitoringItemList），与原有事件表共享sseds命名空间，与要求一致。

#### 格式转换工具测试

为验证格式转换工具能否将参数监视表模板（MonitorConfigAdd.xsd）准确转换为OWL本体文件，确保语义信息完整保留，并通过逻辑推理验证本体的一致性与业务规则映射。

格式转换工具的输出结果如图5-12所示。

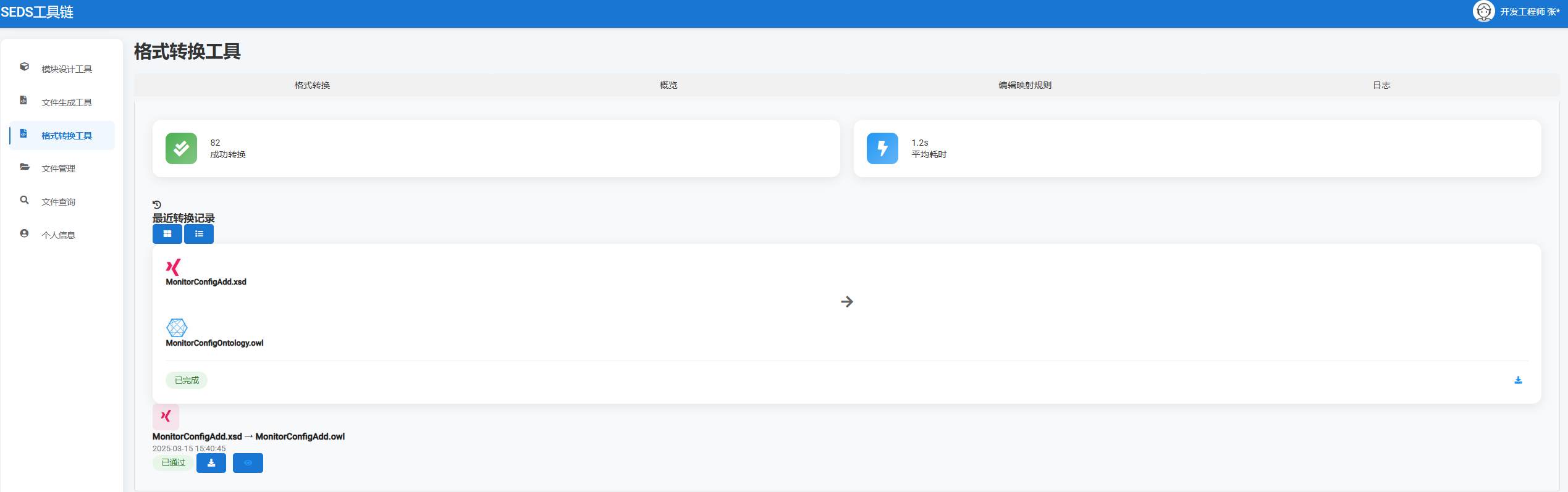


图 5‑12 格式转换工具输出结果

生成的MonitorConfigAdd.owl文件的内容如图5-13所示。



图 5‑13 MonitorConfigAdd.owl文件

基于S-SEDS标准定义了航天参数监视表的语义化结构，通过层级化类（如ParameterMonitoringTable、MonitoringItem）与属性（如StatusSetting、MonitorCondition）规范化卫星在线监控业务。其核心包括：新增参数监视表与事件表并列于SubService层级，支持共存；通过正则表达式约束StatusSetting的8位状态位（定义监视使能、条件类型等）；严格数据类型（如unsignedShort范围阈值、hexBinary掩码格式）；复用事件表的公共属性（ServiceName）并共享命名空间，确保兼容性，满足定义要求。

#### 测试结论

为验证在线监控业务的端到端流程，覆盖模板设计→文件生成→格式转换→参数解析→星载执行全链路，通过三个工具的使用最后顺利生成对应的文件。生成的文件信息如图5-14所示。

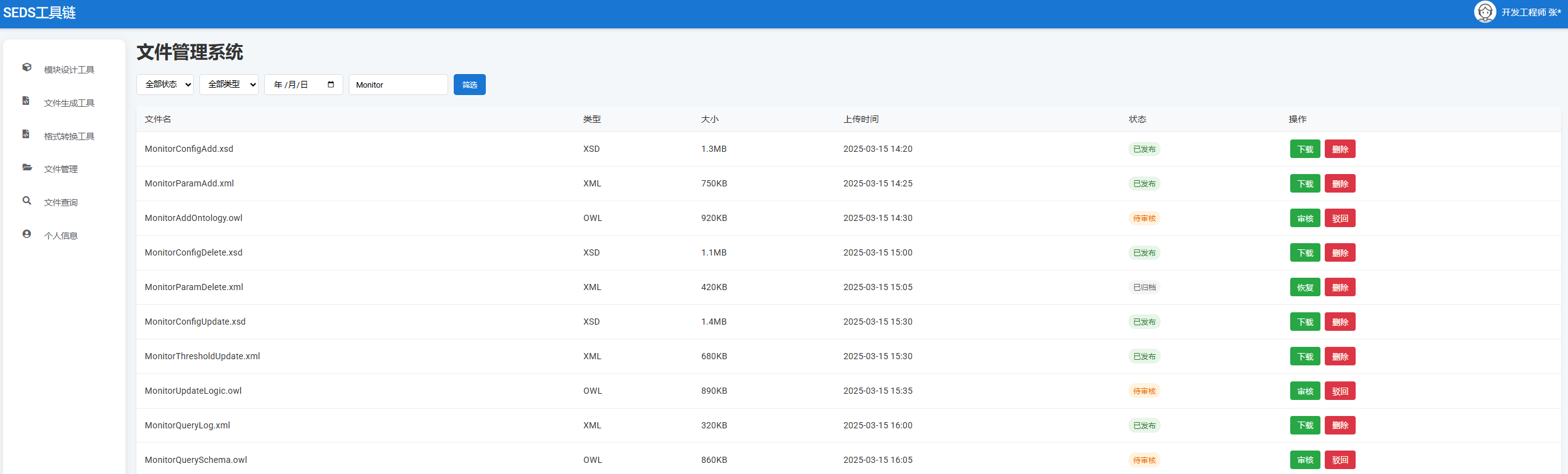


图 5‑14 文件管理系统界面

可以看出，和事件表业务类似，在线监视业务也可以实现三个工具的链路化流程，实现对应文件的生成，满足集成使用的要求。

## 星载业务实现

通过以上测试，工具链能够：

1. 标准化输出：生成的XSD模板、XML实例及OWL本体文件严格遵循S-SEDS标准，层级结构（DataSheet→Service→SubService→具体业务）与数据类型符合规范，无命名空间冲突。
2. 语义一致性：格式转换工具保留XSD的语义约束（如状态位正则模式、判据周期逻辑），推理机验证本体无矛盾，支持语义化查询与冲突检测。

3）星载业务描述的通用性：工具链可适配多种星载业务（以事件表业务和在线监视业务为例），验证了S-SEDS描述星载业务的通用性。

## 本章小结

本章首先通过模块测试和集成测试设计验证工具链的功能性，包括模板设计工具、文件生成工具和格式转换工具。介绍了测试的软件环境，并给出对应的测试用例。

然后分别对事件表业务和在线监视业务进行介绍，对单个工具和集成之后的工具进行测试，结果表明符合测试用例要求。

最后总结验证了本工具链可以可适配多种星载业务，证明了S-SEDS在该软件的通用性。

# 总结与展望

## 总结

本文针对星载业务即插即用描述问题，基于模型驱动设计与本体语义映射技术，开展了面向CCSDS SEDS标准的业务描述方法研究，通过理论建模、工具链开发与场景验证。本研究的主要贡献如下：

1）提出基于S-SEDS的星载业务描述模型：

在SEDS标准描述设备的基础上，通过引入业务元素（Service）、子业务（SubService）及动态接口（ProvidedInterface/RequiredInterface）等扩展设计，构建了支持业务即插即用的S-SEDS模型，解决了业务即插即用所需要的电子数据单描述问题。

2）提出S-SEDS到OWL的语义化转换机制：

提出了一组基于领域知识的XSD到OWL映射规则，涵盖数据类型、接口协议与安全约束的语义保留。通过定义业务逻辑的OWL类继承关系（如Service→SubService）、接口逆向属性（owl:inverseOf）及安全控制（sois:SecurityLevel），实现了业务模型的形式化表达与推理支持。实验表明，转换后的OWL本体完整保留了XSD的结构化约束（如参数范围、枚举类型），并通过Jena框架验证了逻辑一致性。

3）全流程工具链的设计与实现：

开发了由模板设计工具、文件生成工具和格式转换工具组成的工具链，支持从业务需求到标准化数据文件的自动化生成。工具链采用分层架构设计（Vue.js前端+Spring Boot后端+多模态数据库），实现模板动态配置、参数实时绑定及语义转换功能。通过事件表业务与在线监视业务的验证，工具链生成的文件（XSD/XML/OWL）在语法合规性、语义完整性和跨系统互操作性方面均满足设计要求。

## 展望

基于本文研究成果，未来可在以下方向深化星载业务即插即用技术的研究与应用，推动理论与工程实践的深度融合：

1）业务即插即用适配机制研究：

本文提出的S-SEDS模型及工具链为星载业务即插即用提供了理论框架与实现路径，但其实际效能需通过航天器在轨任务验证。未来可结合航天器综合电子系统平台，开展S-SEDS在动态设备接入、业务实时配置及多任务协同中的应用验证，重点突破跨层级协议适配、状态同步容错等关键技术。通过构建全链路仿真环境，量化评估即插即用系统的响应时间、资源占用率及鲁棒性指标，优化模型参数与工具链性能，为工程部署提供理论支撑与实践依据。

2）OWL本体与仿真深度融合：

当前OWL本体主要用于静态知识表达，未来可探索其在动态仿真与自主决策中的应用。基于转换后的本体模型，结合数字孪生技术构建航天器业务仿真引擎，通过本体推理（如SWRL规则）实现异常诊断、任务重规划等动态场景的自动化推演。进一步融合强化学习算法，开发本体知识引导的自主决策模块，使航天器能够基于语义化业务模型自适应调整资源配置与任务时序，提升在复杂空间环境（如深空探测、星座组网）中的智能化水平。此外，可研究本体与多智能体系统的协同机制，支持异构航天器间的语义互操作与协同任务执行，为下一代航天器自主运行体系提供方法论基础。

# 参考文献

1. CCSDS. CCSDS 876.0-B-1 Spacecraft Onboard Interface Services - XML Specification for Electronic Data Sheets [S]. Blue Book， Washington D.C.: CCSDS， 2019.
2. 王石记， 周庆飞， & 安佰岳. IEEE 1451 网络化智能传感器接口技术. 计算机测量与控制， 2012，20(10)， 2600-2602+2616.
3. 吕良庆. (2018). 航天器智能软件体系架构设计与应用研究 [D]. 北京：中国科学院大学.
4. Yang L， He X， Chen B， Du Y， Li L， Mao L. The Application Design of SOIS Electronic Data Sheets in Onboard Integrated Electronic System [C] // Aerospace Conference. New York: IEEE， 2019.
5. 何熊文、徐明伟.航天器接口业务标准化和软件架构现状与发展展望[J].中国航天，2020，9:1-7.
6. CCSDS. CCSDS 660.0-G-1 XML Telemetric and Command Exchange (XTCE) - Element Description [S]. Green Book， Washington D.C.: CCSDS， 2012.
7. 卢广佑，吕良庆，张峻巍. 基于CCSDS RASD框架信息对象建模方法的数据注入工具设计[J]. 航天器工程，2023.
8. Ccsds. 850.0-G-2 Spacecraft Onboard Interface Services[S]. Washington D.C.:CCSDS，2013
9. 李长春. 航天器即插即用技术及开放网络架构的研究与实现[D]. 南京航空航天大学，2019.
10. 朱烜硕. 基于CAN总线的卫星即插即用系统的设计与实现[D]. 郑州大学，2019:7-11.
11. Space Assigned Numbers Authority (SANA). (n.d.). Retrieved November 1， 2024， from <https://sanaregistry.org/r/sois/>
12. CCSDS. (2013). Spacecraft Onboard Interface Services [S]. Washington: CCSDS.
13. 林龙成. 语义网中OWL本体概述及其构建方法研究[J]. 电脑知识与技术， 2020.
14. 何杰，屈国兴. 基于XML Schema分块的快速本体构建方法[J]. 吉林大学学报（理学版）， 2022.
15. 代风，翟翔，施国强，杜臣勇. 面向航天产品研制的知识网络本体建模方法[J]. 浙江大学学报(工学版)， 2018.
16. 沈艺. 开放数据的元数据映射技术途径[J]. 计算机系统应用， 2021， 30(7): 265-270.
17. Jamal， S.， Rahman， C.， & Abdulkarim， M. (2022). XML Schema Validation Using Java API for XML Processing. UKH Journal of Science and Engineering， 6(1)， 33-41.
18. Eberhart， A.， & Hitzler， P. (2020). A Functional API for OWL. Data Semantics Lab， Kansas State University， USA.
19. 崔乃刚， 王平， 郭继峰， 等. 空间在轨服务技术发展综述[J]. 宇航学报， 2007(04): 805-811.
20. 贾平. 国外在轨装配技术发展简析[J]. 国际太空， 2016(12): 61-64.
21. 刘伟伟， 程博文， 汪路元， 等. 一种分布式航天器综合电子系统设计[J]. 航天器工程， 2016.
22. 李新洪， 张永乐， 姜南. 模块化航天器应用需求及应用体系[J]. 装备学院学报， 2014， 25(04):70-74.
23. Fronterhouse D， Lyke J， Achramowicz S. Plug-and-play Satellite (PnPSat)[C]// AIAA Infotech@Aerospace 2007 Conference and Exhibit. Washington D C， USA: AIAA， 2007: 1-13.
24. ECSS-E-ST-70-41A.Space engineering.Telemetry and telecommand packet uti lization[S].Noordwijk， The Netherlands: ECSS， 2003.
25. ECSS. ECSS-E-ST-70-41C. Space engineering， telemetry and telecommand packet utilization[S]. Noordwijk， Netherlands: ECSS，2016.
26. 张鹏. 美国即插即用卫星的发展[J]. 国际太空， 2014(06): 30-33.
27. 顾常青， 李一， 田宏存. 美国即插即用卫星应用对小卫星综合电子发展的启示[J]. 国际太空， 2016(3):17-22.
28. 成飞. 新一代卫星综合总线系统即插即用关键技术研究[D].上海交通大学，2014.
29. 张 军 . 基 于 CANoe软件的 电 动 汽 车 CAN 总 线 测 试 系 统 的 研 究 [J]. 传动技术，2012，26(03):10-13+42.
30. 李新洪， 张育林. 美军“作战响应空间”分析及启示[J]. 装备指挥技术学院学报， 2007， 18(6): 33-36
31. CCSDS.660.1-G-1XML Telemetric and Command Exchange(XTCE)-Element description[S]. Washington D.C.: CCSDS，2012
32. CCSDS.660.0-B-2-SXML Telemetric and Command Exchange-Version1.2[S].

Washington D.C.: CCSDS，2020

1. CCSDS.660.2-G-2 XML Telemetric and Command Exchange(XTCE)[S]. Was hington D.C.: CCSDS，2021
2. 杜娟， 张孝君. 面向在轨模块更换的高级信号接口模块设计研究[J]. 现代电子技术， 2014， 37(24): 37-40.
3. 尤政，田贺祥，李滨，等. 微小卫星综合电子系统中的即插即用技术[J]. 清华大学学报(自然科学版)， 2009， 49(11): 1765-1769.
4. 游红俊， 陈伟男， 周学功， 等. 可扩展的航天器电子系统架构模型[J]. 系统工程与电子技术， 2013， 35(2): 263-269.
5. Broekstra J， Kampman A， van Harmelen F. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema[M/OL]. (2002)[2023-10-10]. <http://sesame.aidministrator.nl/>.
6. McGuinness D L， Fikes R. DAML+OIL: An Ontology Language for the Semantic Web[M/OL]. (2002)[2023-10-10]. <http://www.daml.org/>.
7. Dean M， Schreiber G. OWL Web Ontology Language Reference[M/OL]. (2004)[2023-10-10]. <http://www.w3.org/TR/owl-ref/>.
8. Nejdl W， Wolpers M， Capelle C. The RDF Schema Specification Revisited[M/OL]. (2000)[2023-10-10]. <http://www.kbs.uni-hannover.de/>.
9. Decker S， Melnik S， van Harmelen F， et al. The Role of XML and RDF[M/OL]. IEEE Internet Computing， 2000， 4(5):63-74.
10. McGuinness D L， van Harmelen F. OWL Web Ontology Language Overview[M/OL]. (2004)[2023-10-10]. <http://www.w3.org/TR/owl-features/>.
11. 靖争. XML Schema 到 OWL DL 本体映射的研究[D]. 东北大学信息科学与工程学院， 2008.
12. 温昱. 智能物联网语义建模及解析方法研究[M]. 北京: 北京邮电大学出版社， 2023: 1-47.
13. Duan X， Chaves-Fraga D， Dimou A. XSD2SHACL: Capturing RDF Constraints from XML Schema[C/OL]. In Knowledge Capture Conference 2023 (K-CAP ’23)， December 05–07， 2023

# 附录

如果存在多个附录则删除该页，保留“附录一”，“附录二”等，其中“附录”二字（多个附录则为“附录一”等）与题名间空一个汉字符位。

# 附录一

如没有附录一删除该页

# 附录二

如没有附录二删除该页

# 致 谢

行文至此，掩卷回望，三年的求学历程如星轨般交织着探索的艰辛与收获的喜悦。在此，谨以最诚挚的谢意，献给所有在科研道路上给予我支持与启迪的师长、同窗与亲友。

首先，衷心感谢我的导师吕良庆正高级工程师。从论文选题、方法设计到成果凝练，导师始终以严谨的治学态度和深邃的学术视野为我指明方向。导师对航天事业的热忱与对技术细节的苛求，更让我领悟到科研工作者的责任与使命。

感谢集成室、研究生部老师在学习生活中的帮助。承蒙薛长斌老师、徐海涛老师、赵勋峰老师等集成室老师于实验资源配置方面提供保障。也感谢李俊老师、何京勇老师、云婷老师、许贺楠老师等研究生部老师在生活中给予足够的关怀。

同窗之谊，弥足珍贵。感谢何睿师姐在我课题上的帮助，在论文攻坚期的耐心指导。感谢童博涵、王一鸣、李超一起深夜讨论的思维碰撞。感谢张峻巍、卢广佑、秦昇等同门之间愉快的相处时光。

漫漫时光，知己难觅，挚友难寻。感谢杜晓龙、曹宇两位挚友三载同窗相伴，求职季简历上逐字推敲的批注映着星光，厨房灶台边饭菜的雾气包含着论文灵感。那些共享的焦虑与顿悟、挫败与欢欣，终在录用通知中凝成破局秘钥，更于人生海海里铸就锚点。此间肝胆相照之谊，人生又能得几何。也感谢许永、安泓宇、于天、龙明兵、武尊杰、沈东臻等同学在生活中的帮助。

最后，谨以此文献予我的家人。父母数十年如一日的默默支持，姐姐在远程学习中的鼓励，是我追寻星辰的永恒底气。他们的理解与信任，让我在挫折中重拾勇气，在迷茫时锚定初心。

星河浩渺，探索未止。愿以此刻为起点，继续躬耕，不负韶华，不负师恩。

2025年6月

# 作者简历及攻读学位期间发表的学术论文与其他相关学术成果

**作者简历：**

2018年9月——2022年6月，在郑州大学获得学士学位。

2022年9月——2025年6月，在中国科学院大学国家空间科学中心攻读硕士学位。

**已发表（或正式接受）的学术论文：**（书写格式同参考文献）

**参加的研究项目及获奖情况：**