Computer Science and Application 计算机科学与应用, 2020, 10(2), 334-339
Published Online February 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.102034

Hans汉斯

General Design of Flight Data Ground Process System Based on Reflection Technology of .NET

Shouquan Wang[1,2], Dong Xia[2], Guangsheng Lai[3]

[1]College of Automation Engineering of NUAA, Nanjing Jiangsu
[2]Qingdao Campus of Naval Aeronautics Universityty, Qingdao Shandong
[3]PLA 91954 Troop, Yongzhou Hunan
Email: xcdo@163.com

Received: Feb. 3rd, 2020; accepted: Feb. 18th, 2020; published: Feb. 25th, 2020

Abstract

With development of aeronautic technology, flight data recorded in flight become various in types and huge in amount. Special data process software for given data type is widely utilized nowadays, which brings problems of various kinds, poor generalization and poor extension in ground process software. Flight data ground process software's general design by .Net's reflection technology was studied in this paper, in order to carry out the extension of data types. .NET is an objective developing tool language which is used widely, and its reflection technology support posterior banding operation. As .NET reflection technology was introduced in ground process system, flight data type extensibility would be realized, and general design of system could also be achieved. New data type could be added without any changes of main program as reflection technology is used, which could improve efficiency significantly.

Keywords

Flight Data, Ground Process, Reflection, Data Type Extension

基于.NET反射技术的飞行数据地面处理系统通用性设计

王守权[1,2]，夏　栋[2]，赖光生[3]

[1]南京航空航天大学自动化学院，江苏　南京
[2]海军航空大学青岛校区，山东　青岛
[3]解放军91954部队，湖南　永州

文章引用: 王守权, 夏栋, 赖光生. 基于.NET 反射技术的飞行数据地面处理系统通用性设计[J]. 计算机科学与应用, 2020, 10(2): 334-339. DOI: 10.12677/csa.2020.102034

Email: xcdo@163.com

收稿日期：2020年2月3日；录用日期：2020年2月18日；发布日期：2020年2月25日

摘　要

随着航空技术的进步，越来越多的飞行数据类型需要被记录。而目前针对记录数据的地面分析系统普遍采用各种专用的数据处理软件，这造成了地面处理软件种类繁多、数据单一，通用性差，不具备扩展数据类型的能力。本文将研究利用.NET的反射技术实现飞行数据地面处理软件的通用性设计，并增加对扩展数据类型的支持。.NET是一种广泛受到欢迎的面向对象开发语言工具，它的反射技术支持晚绑定操作，将.NET反射技术引入到地面处理系统，能够实现飞行数据类型扩展，进而实现系统的通用性设计。采用反射技术可以在不改动原程序的情况下添加新的数据类型，极大地提高开发效率。

关键词

飞行数据，地面处理，反射，数据类型扩展

1. 引言

飞行数据记录了飞机各系统的工作状态和飞行员的操作过程，对指导飞行、地面保障和科学研究都具有重要的意义[1] [2]。随着航空技术的发展，需要记录数据的种类和数量越来越多。不同型号飞机记录的数据类型存在非常大的差异[3]，同时飞机在生命周期内也存在增加记录数据类型的需求，这给配套的地面数据处理系统提出了较高的要求。不同数据单独记录并配套专用的地面数据处理软件是当前飞行数据记录普遍存在的现状[4] [5] [6] [7]。随着电子技术的进步和对飞行数据需求的提高，综合采集和地面通用处理已成为未来飞行数据应用的趋势。通用飞行数据地面处理系统用于完成多种航空记录数据的地面管理和使用，与传统的飞行数据地面处理系统相比，存在以下特点：

(1) 数据集中存储，不同客户端完成对数据的操作；

(2) 支持数据源即数据类型的扩展；

(3) 支持数据应用方式扩展；

这里面最重要的是数据类型的扩展，只有有效解决了在不改动原有程序的前提下实现新数据类型的扩展，处理系统才真正实现了通用性，这也是目前地面处理系统没有解决的难题。本文将研究利用.NET反射技术实现飞行记录数据源扩展，进而实现飞行数据地面处理系统的通用性设计。

2. 飞行数据通用地面处理系统

飞行数据通用地面处理系统用来实现不同单位、不同机型飞行记录数据的地面综合管理、调度和基本使用。为了实现机载记录数据的有效管理和使用，通用飞行数据地面处理系统还需要具备管理人员、设备、用户的功能。具体来讲，通用飞行数据地面处理系统功能包括数据卸载与上传、数据解码、数据预处理、数据管理与调度、数据处理、数据回放、系统管理等，详细内容如图 1 所示。

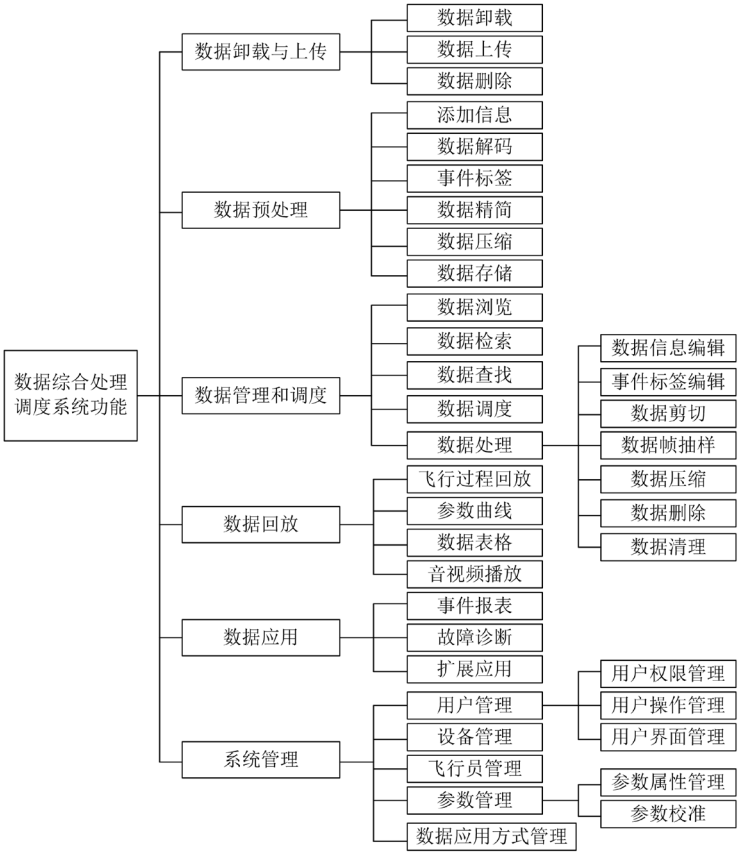

Figure 1. Function schematic diagram of data synthesize process system
图 1. 数据综合处理调度系统功能示意图

1) 数据卸载与上传。系统通过客户端从机载记录设备或者转录设备中卸载数据，也可在客户端将数据写入到记录设备或者转录设备，并且可以删除记录设备或者转录设备中指定的数据。

2) 数据解码。为了用户能够操作和使用数据，系统需要将原始数据解析为有物理意义的数值数据。

3) 数据预处理。在机载记录数据卸载到客户端之后、存储到数据服务器之前，需要对数据进行野值剔除、滤波、和初步的数据处理工作等。

4) 数据管理和调度。对飞行数据进行管理是数据综合处理调度系统的一项重要内容，数据管理内容包括数据的浏览、检索、查找、数据保护和数据处理。系统还必须具有数据调度的功能，即将不同用户需要的数据合理分配到正确的客户端，并将经过客户端修改后的数据更新到数据服务器。

5) 数据处理。数据处理包括对数据信息、事件标签、部门日志和对数据本身的编辑和处理，其中对数据本身的处理又包括数据精简、压缩、删除等。

6) 数据回放。数据回放是将记录数据内容进行再现的过程，包括飞行过程回放、参数曲线、数据表格、音视频回放等。系统将用户选择数据从服务器传递到用户所在客户端以后，由客户端完成数据的回放。

7) 数据应用。系统将用户选择数据从服务器传递到用户所在客户端以后，用户在客户端完成对数据的应用。对数据的应用方式包括事件报表、故障诊断和其它扩展的应用方式。

8) 系统管理。为了保证系统正常运行，系统还需要对用户、设备、飞行员、参数库、数据应用方式等内容进行管理。

3. .NET 反射技术

.NET 的反射技术是指软件在运行过程中动态发现类型信息的能力[8]。软件系统有时需要在运行时处理一些信息，而这些信息在设计时是未知的。反射支持后期绑定，并且允许程序与设计时未知的代码进行交互，因此反射技术极大地提高了.NET 程序的可扩展性。反射技术可以在主程序开发完成后，在不改变原有代码的前提下实现程序扩展，这也是反射技术的价值所在。

要理解反射技术首先要熟悉程序集(Assembly)的概念。在.NET 平台中，软件组件是以程序集的方式存在的，它包含模块，而模块包含类，类又包含成员。程序集是.NET 所采用的应用程序的构造模块，它替代了以前的 EXE 与 DLL 程序结构，构成了部署、版本控制、重复使用、激活范围控制和安全权限的基本单元[9]。程序集具有以下优点：

1) 形成安全边界。程序集是请求和授予权限的单元。

2) 形成类型边界。每一类型的标识均包括该类型所驻留的程序集的名称。

3) 形成引用范围边界。程序集的清单包含用于解析类型和满足资源请求的程序集元数据。它指定在该程序集之外公开的类型和资源。该清单还枚举它所依赖的其他程序集。

4) 形成版本边界。程序集是公共语言运行库中最小的可版本化单元，同一程序集中的所有类型和资源均会被版本化为一个单元。

5) 形成部署单元。当一个应用程序启动时，只有该应用程序最初调用的程序集必须存在。其他程序集(例如本地化资源和包含实用工具类的程序集)可以按需检索，按需加载。

反射技术支持显式加载一个程序集、动态发现类型和类型成员的信息，以及调用这些类型和成员的操作。反射通常具有以下用途[10]：

1) 使用 Assembly 定义和加载程序集，加载在程序集清单中列出的模块，以及从此程序集中查找类型并创建该类型的实例；

2) 使用 Module 获取模块的程序集以及模块中的类等信息；

3) 使用 ConstructorInfo 获取构造函数的名称、参数、访问修饰符和实现详细信息等；

4) 使用 MethodInfo 来获取方法的名称、返回类型、参数、访问修饰符和实现详细信息等；

5) 使用 FieldInfo 来获取字段的名称、访问修饰符和实现详细信息等，并获取或设置字段值；

6) 使用 EventInfo 来获取事件的名称、事件处理程序数据类型、自定义属性、声明类型和反射类型等，并添加或移除事件处理程序；

7) 使用 PropertyInfo 来获取属性的名称、数据类型、声明类型、反射类型和只读或可写状态等，并获取或设置属性值；

8) 使用 ParameterInfo 来获取如下的类似信息，如参数的名称、数据类型、参数是输入参数还是输出参数，以及参数在方法签名中的位置等。

4. 反射技术在地面处理系统通用性设计中的应用

.NET 的反射技术支持晚绑定，允许程序与设计时未知的代码进行交互，因此具有强大的可扩展性，合理利用.NET 的反射技术可以实现地面处理系统的通用性设计。地面处理系统通用性设计的关键在于通用解码器的设计，通用地面处理系统的解码器不仅能够对所有已有的数据源进行解码，同时还需要能够扩展新的数据源中的新数据类型。而不同类型数据源的数据存储格式存在较大差异，用同一个解码器来实现所有数据类型的解码是不现实的。本文采用的方法是采用一个动态解码器集来实现不同类型数据的解码，用户可以通过.NET 反射技术添加新的解码器到动态解码器集来扩展新数据源，解析器根据数据文件头中包含的数据类型信息确定是否属于本身处理的数据。

4.1. 定义 IParameterAnalyze 接口

IParameterAnalyze 接口规范了参数解析器需要实现的功能，每个解析器类都必须实现该接口。IParameterAnalyze 接口的定义如图 2。

Figure 2. Interface structure of parameter analyzor
图 2. 参数解析器接口结构

该接口提供了参数解析相关的四个方法，各方法的详细解释如下：

1) 确定解析器是否适用于该类型的记录数据

Boolean FlightFileVerify(String flightFile,out FlightFileInfo filghtInfo);

2) 完成对源文件进行数据解析

Boolean DataDecode(String SourceFile, List<String> Parameters,out List<parameterValues> parameters-Value, out List<IncidentReportItem> IncidentReports);

3) 提供该解析器包含的所有参数的详细信息

List<parameter_info> GetAllParametersInfo();

4) 参数校准；

Boolean ParameterAdjust(String planeNo, String nickName, List<ParameterAdjustItem> adjustItems);

4.2. 创建解析器集

为了保证后续数据类型的动态扩展，系统应包含私有的、一个长度可变的 IParameterAnalyze 接口列表即解析器集，用来记录已注册的所有数据解析器。

List<IParameterAnalyze> _ParameterAnalyzator = new List<IParameterAnalyze>();

4.3. 开发适用的新解析器，并在数据库或配置文件中注册该解析器

每个新数据源都必须提供相关的解析器类，每个解析器类必须继承 IParameterAnalyze 接口并实现其所有的方法。新数据源解析器开发完毕后，需要在配置文件或数据库中注册新解码器信息，包括解析器类、所在程序集等，然后系统才能使用新解析器。

4.4. 解析器集加载

系统读取保存在数据库或配置文件中的解析器信息，并通过反射技术建立所有解析器实例，并添加到动态解码器集_ParameterAnalyzator 中。下面为从数据库中读取解析器信息，并建立解析器的示例代码。

```
//**************************************
String setFile = curRow["解析程序集"].ToString();     //读取解析集文件
if (File.Exists(setFile))
{
Assembly a = Assembly.LoadFrom(setFile); //载入解析器所在程序集
String analy = curRow["解析器"].ToString();
```

```
foreach (Type curAnaly in a.GetTypes())
{
String name = curAnaly.Name;
if (name == analy)
{
IParameterAnalyze newAnalyzor = (IParameterAnalyze) a.CreateInstance (curAnaly.FullName); //创建解析器实例
_ParameterAnalyzator.Add(newAnalyzor);//添加到动态解析器集
break;
}
}
}
//*************************************
```

新数据源对应的解析器被加载、添加到动态解析器_ParameterAnalyzator 以后，就可以利用该解析器实现对新类型数据源的参数解析。

5. 结论

通过引入.NET 反射技术，可以动态扩展航空客观检查飞行记录数据类型，进而保证飞行数据地面处理系统通用性设计。扩展新数据类型只需要开发新的解码器组件并注册组件信息就可以实现，而不需要改动和重新编译生成已有系统。因此，利用.NET 反射技术既可以实现数据源的灵活扩展，又可以避免原有系统被修改的风险。

参考文献

[1] 李映颖, 谭光宇, 陈友龙. 基于飞行数据的航空发动机健康状况分析[J]. 哈尔滨理工大学学报, 2011, 16(5): 43-46.

[2] 李利荣, 孙立伟, 杨浩, 等. 基于 Spark 的飞机试验数据预处理技术研究[J]. 计算机测量与控制, 2018, 26(12): 266-270.

[3] 张宁, 张安, 张耀中. 飞行数据综合回放系统设计[J]. 火力与指挥控制, 2013, 38(12): 43-46.

[4] 杜宏银. 一种遥测数据处理系统设计[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2011.

[5] 朱菲菲, 高艳辉, 肖前贵, 胡寿松. 观测器/卡尔曼滤波在飞行数据处理中的应用[J]. 电子设计工程, 2016(20): 91-93.

[6] 贾雨, 吴海东, 齐禅颖, 王爽. 飞行试验 IENA 数据实时处理技术研究[J]. 电子设计工程, 2018(9): 12-15.

[7] 高歌梦月, 刘荣林. 飞参数据综合记录分析系统的设计与实现[J]. 指挥控制与仿真, 2018(1): 118-122.

[8] 章志, 都金康, 卓凤军. 基于.NET 反射机制的 GIS 插件技术研究[J]. 测绘科学, 2011, 36(4): 151-153.

[9] 蒋鸿翔, 徐锦法, 高正. 无人直升机地面站系统组件化分布式设计与实现[J]. 南京航空航天大学学报, 2007, 39(4): 475-480.

[10] 李鹏, 周德俭, 刘电霆. 基于.NET 反射机制和组件复用的中小型离散制造企业 MES 配置系统研究[J]. 软件导刊, 2012, 11(8): 91-93.