计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(2):117−124 [doi: 10.15888/j.cnki.csa.006205]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 ARINC 661 协议的 DF 文件验证方法①

崔诗娴

(中电科航空电子, 成都 611731)
通讯作者: 崔诗娴, E-mail: cuisx@cetc.net.cn

摘　要: 随着民用航空电子系统的逐步集成化和模块化, IMA (Integrated Modular Avionics) 系统应运而生, 它对民用航电系统中多个独立的应用 UA (User Application) 进行了综合管理和部署. 在以 IMA 为核心的民用航电系统中, CDS (Cockpit Display System) 提供了对 UA 界面进行显示的功能. 同时 CDS 管理的 IDU (Integrated Display Unit) 显示设备提供了 HMI 界面, 将飞行员的操作反馈给 IMA 上的应用. 并行开发单个 UA 的代码和其符合 ARINC 661 协议的显示加载文件 DF (Definition File), 却分别运行在不同的目标机上 (IMA 和 IDU), 两者具有特殊的相关性和独立性. 为了符合 DO-178C 开发和验证的要求, UA 和 DF 交付物通过适航, 对显示加载文件 DF 的开发和验证的过程和方式进行了研究.

关键词: 民用航空电子系统; IMA; CDS; UA; ARINC 661; DF

引用格式: 崔诗娴.基于 ARINC 661 协议的 DF 文件验证方法.计算机系统应用,2018,27(2):117–124. http://www.c-s-a.org.cn/1003-3254/6205.html

Method of DF Verification Based on ARINC 661 Protocol

CUI Shi-Xian

(CETCA-China Electronics Technology Group Avionics Corporation, Chendu 611731, China)

Abstract: With the development of integrated and modularized civil aircraft electronic system, IMA (Integrated Modular Avionics) system comes out. It manages and deploys many individual UAs (User Applications). CDS (Cockpit Display System) which is in the civil avionics system focusing on IMA, provides the display function for UA. At the same time, CDS manages the IDUs (Integrated Display Unit) which provides the HMI for cockpit giving the feedback to UA. Developing the UA code and ARINC 661 protocol DF (Definition File) which is loaded on IDU are the responsibility of UA developers. But they are not implemented in one target PC. So, they are individual and related. In order to according with DO-178C and CAAC's certification, we do the research of development and verification of DF.

Key words: civil avionics system; IMA; CDS; UA; ARINC 661; DF

在开发驻留在 IMA 系统中的 UA 软件时, 需要对 UA 本身的人机界面进行定义和设计, 并且开发负责与 IDU 通信的 ARINC 661 协议模块以及存储和管理界面控件的模块. 同时负责生成一个符合 A661 协议的 DF 文件, 包括了所有 UA 定义的图形控件, 用于 IDU 加载, 使得该 UA 的界面得以显示. UA 代码中对图形控件的定义和加载在 IDU 上的 DF 文件中的定义必须一致, UA 才能提供完善的功能供飞行员使用. DF 文件由 UA 开发人员创建, 却加载在 CDS 开发人员提供的 IDU 上. 因此, 对于和 DF 相关的需求和验证, 就应该由 UA 开发者和 CDS 开发者共同承担, 这就是当前国内民机项目研制时, 需要解决的, 且在国外民机项目中所没有的问题. 对于有丰富的民用航空系统研发经验的 Rockwell Collins, Honeywell 和 GE 等公司来

① 收稿时间: 2017-05-05; 修改时间: 2017-05-26; 采用时间: 2017-06-08

说, 他们参与的民用飞机项目中, 是将 DF 文件的所有开发和验证, 全部交由 CDS 开发者完成. 根据业界的经验总结, 这样的解决方案会出现以下这些普遍性问题:

(1) UA 开发者和 CDS 开发者的沟通不当, 如 UA 需求的理解或表达不当, 会导致 CDS 开发者开发的 DF 文件不能支持 UA 的功能.

(2) UA 功能的频繁更新, 导致 DF 文件的更新不一致.

(3) CDS 开发者更新控件库, UA 开发者不能同步修改相关功能以及控件属性.

(4) UA 开发者只能等待 DF 文件开发完成后, 才能进行自己的界面交互功能验证, 在验证执行中, 多以接口 (ICD) 方式进行验证, 这种验证方式不直观, 不能准确表达飞行员的真实感受.

(5) CDS 开发者工作量巨大, 需要完成所有 UA 的 DF 文件设计, 并且在等待 UA 开发者完成开发后, 才能进行 DF 文件和 UA 的联调, 开发周期较长.

综上所述, 传统的 DF 文件开发方式, 容易出现多次的异步同步问题, 导致开发周期延长, 追踪性不完整, 工作协调复杂, 耗费大量人力物力. 因此, 我们在国内民用飞机系统中, 首次尝试由 UA 开发者承担 DF 文件的开发任务. UA 开发者可以按照软件的需求, 开发满足需求的 DF 文件, 最后将 DF 文件交付给 CDS 开发者, 统一进行集成测试.

由于 DF 文件并不是可执行代码, 因此, 验证方式与典型的民用航电系统软件方式有所不同. 为了达到 DO-178C 的要求和 CAAC 对软件的审定要求, DF 文件的验证必须采取分析, 评审和测试的综合方式, 达到最终的验证目的. 民用飞机的研制是一个巨大的系统和工作, 包括波音和空客, 均采用全球供应商模式, 类似于 DF 文件这样需要多个供应商共同承担验证的情况日益趋多, 本文介绍的仅仅是 UA 开发者对 DF 文件的验证方式[1,2].

1 Definition File

目前, 主流的民用飞机采用的是 CDS 来进行显示控制. CDS 包括多个 IDU, 每个 IDU 提供一个 ARINC 661 Server. ARINC 661 Server 的主要工作包括:

(1) 定义界面大小, 位置, 形状等公共窗口属性.

(2) 发送握手协议, 与 UA 建立会话, 维护通信.

(3) 加载 DF 文件.

(4) 发送保活报文, 监控 UA 状态, 管理会话和通信.

由于民用飞机航电系统软件的高安全性的特性, C 级以上的软件多数都有显示备份, UA 可以和两个或两个以上的 IDU 进行通信. 通信并不表示可以显示, 只是该 IDU 具备了显示该 UA 的配置. 一般情况下同一个 UA 界面可以同时在 2 个或 2 个以上的 IDU 上进行显示. ARINC 661 Server 负责管理显示事件, IDU 在系统配置时, 获得显示某个 UA 权限后, 将与 UA 建立通信, 在收到显示事件后, 加载该 UA 的 DF 文件. 需要在 IDU 上进行显示的 UA, 直接和相应的 IDU 上的 ARINC 661 server 进行通信. DF 文件就是用于 UA 和 IDU 进行通信的静态非可执行代码数据. 因此, DF 文件必须包含以下内容:

(1) UA 页面的架构信息, 页面之间的树形关系.

(2) 每一个页面的控件信息及其属性.

(3) 页面之间的逻辑关系, 页面之间的跳转, 按钮链接的界面信息.

UA 开发者使用 SCADE DISPLAY 工具, 加载由 CDS 开发者提供的 workplace(定义了控件属性), 通过画图的方式, 完成界面设计. SCADE DISPLAY 工具自动生成 DF 文件, 生成的 DF 文件有三种格式, 分别是 Binary, XML 和 Hexa, 同时生成记录 DF 文件生成过程的日志文件.

首先需要说明的是, 参照 DO-178C 的 section 2.5.1, “A data set that influences the behavior of software without modifying the Executable Object Code and is managed as a separate configuration item is called Parameter Data Item”. 尽管 DF 文件是和 UA 结合开发, 但是并不在 UA 的目标机上加载, 因此 DF 文件并不会影响 UA 的运行, DF 文件不是 UA 的参数数据项. 相反, 部署在 IDU 上的 ARINC 661 Server 才会使用 DF 文件去实施 UA 界面的显示, ARINC 661 Server 通过 ARINC 661 协议和 UA 进行通信, 实时显示 UA 的控制界面, IMA 和 CDS 的部署如图 1 所示.

驻留在 IMA 上的 UA 与相应的 ARINC 661 Server 进行通信, 通常使用 ARINC 664 接口 (空客称之为“AFDX”接口) 传输 ARINC 661 报文. DF 文件预先根据系统配置, 存放在相应的 IDU 上. 当 ARINC 661 Server 在收到来自 UA 的显示事件后, 会加载该 UA 的 DF 文件, 对该 UA 进行显示. 这样的部署, 达到了灵活备份, 减少冗余和快速切换的目的. 比如 UA 可以在

IDU1, IDU3 和 IDU6 上进行显示, 但是同时只能在两个 IDU 上进行显示. 那么, 这三个 IDU 将配置该 UA 的 DF 文件, 当 UA 已经在 IDU1 和 IDU3 上进行显示时, IDU6 将不会加载 UA 的 DF 文件, 不允许 UA 的显示. 当 UA 在 IDU1 上的显示失效后, IDU6 上将收到 UA 的显示事件, 加载 UA 的 DF 文件, UA 将在 IDU3 和 IDU6 上进行显示. UA 维持所有获得显示权限的 A661 Server 的通信会话, 只发出允许显示的界面个数的显示事件[3,4].

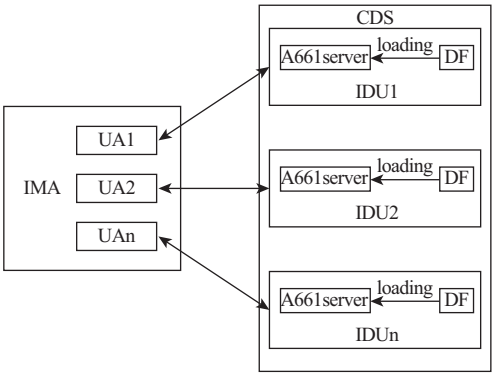

图 1 IMA, CDS 和 DF 文件的部署图

2 DF 文件的开发和验证

2.1 DF 文件的开发流程

DF 文件的开发既然是结合 UA 同时进行, 那么 UA 开发和验证的所有阶段, 都将对 DF 文件进行同时开发和验证. 如图 2 所示,

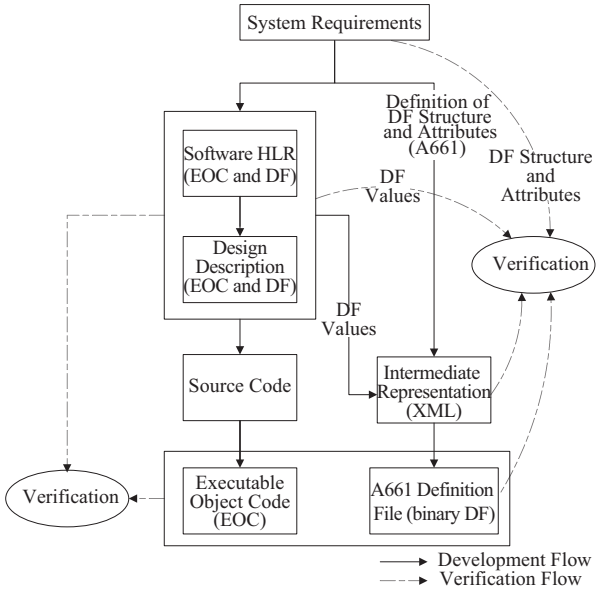

图 2 DF 文件开发验证流程图

在系统需求阶段, 需要将 DF 文件的结构和属性定于在系统需求中. 比如 DF 文件中所包含的界面的 layer ID 的定义, 以及 layout 的定义. 在软件高级需求和设计 (低级需求) 的开发中, 需要将 DF 文件中各个控件的具体参数定义. 比如一个控件的属性是 button 还是 data entry, 字体的 style set 值 (用于定于字体的颜色和大小) 等. 编写代码的同时, 根据需求和设计, 通过 SCADE DISPLAY 工具, 完成 DF 文件的开发. DF 文件的开发同样需要满足民机航电系统开发的基本要求:

(1) 自顶向下的开发流程, 从系统需求到高级需求, 再到低级需求和设计.

(2) 双向追踪性, DF 文件中的代码数据可以逆向追踪到低级需求的控件的定义.

(3) 覆盖率测试满足 DO-178C 的要求, 不出现 dead code, inactive code 等.

(4) 根据相应的软件等级, 满足开发和验证的独立性.

(5) DF 文件的开发需要和 UA 代码同步进行, 同步更新和验证.

2.2 DF 文件的开发流程

在系统需求开发阶段, 该阶段的交付物是系统需求文档, 因此在该环节只需要验证 DF 文件的架构定义和属性定义. 验证人员通过评审的方式, 验证与 DF 文件相关的需求. 评审 DF 文件的需求, 需要结合 UA 软件的系统需求, 必须保证以下三点:

(1) 符合 UA 软件系统需求对软件显示层次的要求.

(2) 对 DF 文件的框架属性定义完整, 所有需求和 UA 的软件高层界面需求一致.

(3) 能够实现 UA 软件的显示需求.

在高级需求和设计 (低级需求) 开发阶段, DF 文件中的参数定义已经明确, 界面需求已经完善. 该阶段的交付物是高级需求, 设计文档 (包括低级需求), 因此, 在该环节需要验证的是 DF 文件中控件的定义, 逻辑关系以及各种属性参数值. 同样, 该阶段也必须结合 UA 相应的需求, 必须与 UA 所定义的界面需求保持一致.

在经过编码阶段后, code 和 DF 文件都已经完善. 验证人员需要结合软件系统需求, 软件高级需求, 软件低级需求和软件设计文档, 对 SCADE DISPLAY 工具产生的具有可读性的 XML 格式的 DF 文件进行评审, XML 格式的 DF 文件内容如图 3 所示.

```
<a661_widget name="FieldTiltle_MsgLog" type="A661_LABEL">
 <model>
  <prop name="WidgetIdent" value="466" />
  <prop name="Anonymous" value="A661_FALSE" />
  <prop name="Visible" value="A661_TRUE" />
  <prop name="PosX" value="150" />
  <prop name="PosY" value="16065" />
  <prop name="SizeX" value="4000" />
  <prop name="SizeY" value="930" />
  <prop name="RotationAngle" value="0" />
  <prop name="StyleSet" value="2" />
  <prop name="MaxStringLength" value="32" />
  <prop name="MotionAllowed" value="A661_TRUE" />
  <prop name="Font" value="1" />
  <prop name="ColorIndex" value="24" />
  <prop name="Alignment" value="A661_LEFT" />
  <prop name="LabelString" value="MESSAGE LOG" />
 </model>
</a661_widget>
```

图 3　XML 格式的 DF 文件内容

依据需求文档, 对 XML 文件中的控件参数进行 review, 查看框架结构 (控件对象继承性) 和控件参数是否和需求一致. 其次, 需要进行覆盖率验证, XML 文件必须覆盖所有需求定义的层次, 结构, 控件和参数, 同时不能有多出需求定义的表达.

在验证 UA 的代码是否符合 UA 的高级需求和低级需求的测试中, 会将 UA 运行在目标机 IMA 上, SCADE DISPLAY 工具产生的 BIN 文件会加载到 IDU 上. 在此过程中, 只要 IDU 能够正确显示 UA 界面, 并且支持完成所有的 UA 验证, 我们认为 BIN 文件通过测试. UA 的功能验证和 BIN 文件验证的环境图, 如图 4 所示.

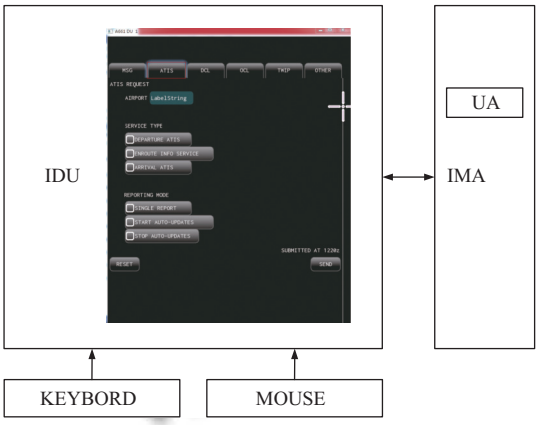

图 4　BIN 文件格式的 DF 文件验证环境示意图

通过键盘和鼠标对 IDU 输入数据, 输入数据通过 ARINC 664 接口, 封装为 ARINC 661 协议, 传输至 IMA 上的 UA. UA 接收并处理这些数据后, UA 做出相应的功能响应, 同时将界面的显示处理反馈至 IDU, IDU 上的 ARINC 661 Server 处理该反馈信息后, 更新 DF 文件的运行数据, 更新界面的显示. IDU 上的界面实时显示数据, 通过获取 DF 文件的运行数据得到, DF 文件的运行数据和 UA 本地存储的界面数据一致, 实现对 UA 用户界面的实时显示.

但是, 通过加载 DF 的 BIN 格式文件, 结合 UA 的功能测试验证 DF 文件的验证方式会出现以下问题:

(1) UA 开发者并不会购买 SCADE DISPLAY 的 DF 文件生成器的认证包 (价格相对昂贵), 但又必须验证 DF 文件的正确性.

(2) 加载的 BIN 文件不具备可读性, 无法通过评审和分析的形式进行验证.

(3) 无法将 DF 文件的非可执行代码追踪到相应的需求.

(4) 即使正确支持了 UA 所有的功能测试, 也不能说明 DF 文件的开发完全符合需求定义.

(5) 不能对 BIN 格式的 DF 文件进行覆盖率测试和静态代码走查

但是, 只要能验证 BIN 文件和已经通过评审的 XML 文件信息一致, 就可以说明 BIN 文件也满足需求定义. 那么便需要对 BIN 格式的 DF 文件和 XML 格式的 DF 文件进行一致性验证. 由于 BIN 文件是二进制格式, 而 XML 文件是可读字符, 二者无法进行直接的验证. 那么, 此时需要应用 SCADE DISPLAY 产生的 HEXA 格式的 DF 文件作为中间件, 验证 BIN 和 XML 文件的一致性, 验证文件关系如图 5 所示.

图 5　三种格式的 DF 文件关系图

HEXA 文件中包含了 XML 文件控件的描述, 同时又将该段参数转换成了十六进制的数据, HEXA 文件内容如图 6 所示.

```
 # A661_Create_Structure (TabbedPanelGroup)
CA01 # A661_CMD_CREATE
0024 # CommandSize
A330 # WidgetType (A661_TABBED_PANEL_GROUP)
006E # WidgetIdent
0000 # ParentIdent
01 # Enable (A661_TRUE)
01 # Visible (A661_TRUE)
00000000 # PosX
000003B1 # PosY
00003E80 # SizeX
00004623 # SizeY
0000 # StyleSet
0000 # ActiveTabbedPanelID
11 # TabPosition (A661_TOP)
01 # AutomaticInsetSizeFlag (A661_TRUE)
0000 # UnusedPad
```

图 6　HEXA 格式的 DF 文件内容

在验证了 XML 文件和 HEXA 文件的描述字段和参数一致后, 将 HEXA 文件的十六进制数据和 BIN 文件的二进制数据进行匹配, 只要数据一致, 那么一致性便得到了验证.

由于 XML 文件是根据需求进行评审, BIN 文件根据需求进行测试, 测试用例分别基于高级需求和低级需求进行开发, XML 文件和 BIN 文件又通过 HEXA 文件, 进行了一致性验证, 那么就可以得到如图 7 所示的追踪关系.

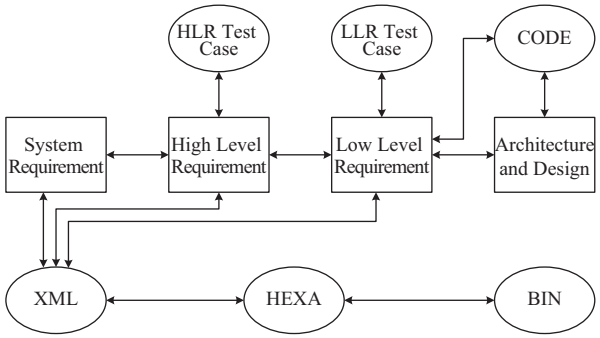

图 7 DF 文件验证的追踪性

执行高级需求测试用例和低级需求测试用例, 验证 BIN 文件是否支持 UA 的功能性测试, DF 文件内容定义和 UA 实现的一致. 根据系统需求, 高级需求和低级需求以及设计文档对 XML 文件进行评审, 对 XML 文件进行了验证. 同时, 也完成了 DF 文件的追踪性验证.

综上所示, UA 开发者负责的 DF 文件的验证具有以下特点:

(1) DF 文件验证具有独立性, 同时结合了 UA 的功能性验证, 保证了双向的数据关系一致.

(2) DF 文件的验证具备完善的追踪性, 适航证据完整.

(3) DF 文件的验证从系统需求阶段开始, 贯穿了整个软件开发流程.

(4) 通过完善的追踪性, 能够达到 UA 界面需求跟新, DF 文件需求更新, 到 DF 文件同步更新的目的.

(5) 最终得到的 DF 文件, 交付至 CDS 开发者后, CDS 开发者只需将其作为 ARINC 661 Server 的参数项进行验证, 与 UA 功能无关.

2.3 DF 文件的测试规程和结果

在使用 SCADE DISPLAY 工具后, 产生了 XML, HEXA, 和 BIN 格式的 DF 文件. 将 UA 运行在 IMA 上, BIN 文件加载在 IDU 上, 通过功能测试, 验证 IDU 显示的界面是否符合需求定义. 例如需求中定义的界面如图 8 所示.

此时通过 IDU 显示出来的界面, 如图 9 所示.

CDS 开发者提供的 workspace 与设计需求文档中的界面, 因为实际使用的画图工具的差别, 会出现细微的差别. 此时, 通过功能验证, 目测界面的颜色字体, 能够符合最终的系统成员规范, 此时的 DF 文件便可以通过测试.

第二步是根据需求评审 XML 文件. 例如 ATIS 界面的一个控件描述, 如图 10 所示.

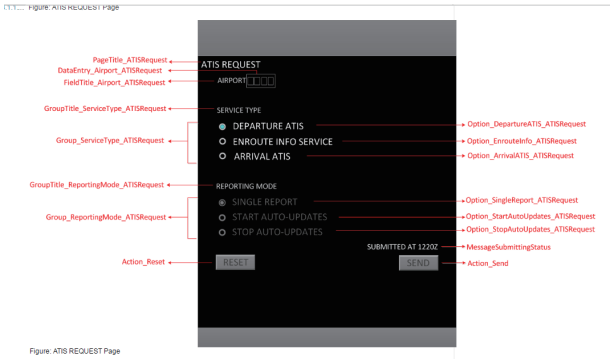

图 8 需求设计界面

图 9 IDU 的真实显示界面

```
<a661_widget name="CheckButton" type="A661_CHECK_BUTTON">
 <model>
  <prop name="WidgetIdent" value="127" />
  <prop name="Enable" value="A661_TRUE" />
  <prop name="Visible" value="A661_TRUE" />
  <prop name="PosX" value="1200" />
  <prop name="PosY" value="12260" />
  <prop name="SizeX" value="4500" />
  <prop name="SizeY" value="930" />
  <prop name="StyleSet" value="50" />
  <prop name="NextFocusedWidget" value="0" />
  <prop name="MaxStringLength" value="32" />
  <prop name="CheckButtonState" value="A661_UNSELECTED" />
  <prop name="Alignment" value="A661_LEFT" />
  <prop name="AutomaticFocusMotion" value="A661_FALSE" />
  <prop name="PicturePosition" value="A661_LEFT" />
  <prop name="LabelString" value="DEPARTURE ATIS" />
 </model>
```

图 10 Check Button 的 XML 表达

在完成 XML 文件的评审后, 进行 HEXA 文件和 XML 文件的一致性验证, 与上图对应的 ATIS 界面的一个控件的 HEXA 表达, 如图 11 所示.

将 XML 中的表达和 HEXA 的二进制对应的映射表, 如表 1 所示.

值得注意的是, 在控件身份表达中, 增加了一个 Parent Identity. 它是 SCADE DISPLAY 给该控件添加的父节点属性, 在 XML 文件中并没有表达出来, 因为 XML 文件是按照树结构进行组织的, 树形结构便可以表达出子节点和父节点的继承关系. 而二进制表达式无法显示树形结构, 便添加父节点属性来体现控件对象的继承性. 在对 DF 文件的覆盖率验证中, 必须说明类似 Parent Identity 这样的参数存在的原因, 才能通过覆盖率的测试和分析.

```
 # A661_Create_Structure (CheckButton)
CA01 # A661_CMD_CREATE
0038 # CommandSize
A050 # WidgetType (A661_CHECK_BUTTON)
007F # WidgetIdent
0082 # ParentIdent
01 # Enable (A661_TRUE)
01 # Visible (A661_TRUE)
000004B0 # PosX
00002FE4 # PosY
00001194 # SizeX
000003A2 # SizeY
0032 # StyleSet
0000 # NextFocusedWidget
0020 # MaxStringLength
00 # CheckButtonState (A661_UNSELECTED)
13 # Alignment (A661_LEFT)
00 # AutomaticFocusMotion (A661_FALSE)
13 # PicturePosition (A661_LEFT)
0000 # UnusedPad
44455041 # LabelString (DEPA)
52545552 # (RTUR)
45204154 # (E AT)
495300 # (IS\x00)
00 # UnusedPad
```

图 11 IDU 的真实显示界面

表 1 Check Button 控件的 XML 和 HEXA 表达的一致性对照

Item Name	XML Expression	Binary Expression
Declare of widget check button	<a661_widget name="CheckButton" type="A661_CHECK_BUTTON">	# A661_Create_Structure (CheckButton) CA01 # A661_CMD_CREATE 0038 # CommandSize A050 # WidgetType (A661_CHECK_BUTTON)
Widget identity and parent identity	<prop name="WidgetIdent" value="127" />	007F # WidgetIdent 0082 # ParentIdent
Enable and visable	<prop name="Enable" value="A661_TRUE" /> <prop name="Visible" value="A661_TRUE" />	01 # Enable (A661_TRUE) 01 # Visible (A661_TRUE)
Size	<prop name="PosX" value="1200" /> <prop name="PosY" value="12260" /> <prop name="SizeX" value="4500" /> <prop name="SizeY" value="930" />	000004B0 # PosX 00002FE4 # PosY 00001194 # SizeX 000003A2 # SizeY
Color and font	<prop name="StyleSet" value="50" />	0032 # StyleSet
Attribution	<prop name="NextFocusedWidget" value="0" /> <prop name="MaxStringLength" value="32" /> <prop name="CheckButtonState" value="A661_UNSELECTED" /> <prop name="Alignment" value="A661_LEFT" /> <prop name="AutomaticFocusMotion" value="A661_FALSE" /> <prop name="PicturePosition" value="A661_LEFT" />	0000 # NextFocusedWidget 0020 # MaxStringLength 00 # CheckButtonState (A661_UNSELECTED) 13 # Alignment (A661_LEFT) 00 # AutomaticFocusMotion (A661_FALSE) 13 # PicturePosition (A661_LEFT) 0000 # UnusedPad
Label name	<prop name="LabelString" value="DEPARTURE ATIS" />	44455041 # LabelString (DEPA) 52545552 # (RTUR) 45204154 # (E AT) 495300 # (IS\x00) 00 # UnusedPad

如表 2 所示, 对于 Check Button 来说, Radio box 就是他的父控件, 所以在定义 Check Button 时, 需要将 Radio Box 的 WidgetIdent 0082 赋值给 ParentIdent 来体现对象的继承性.

表 2 Check Button 与 Radio Box 的表达关系

控件图(Check Button和Radio Box)	父节点Radiobox
	# A661_Create_Structure (RadioBox) CA01 # A661_CMD_CREATE 000C # CommandSize A2B0 # WidgetType (A661_RADIO_BOX) 0082 # WidgetIdent 0073 # ParentIdent 01 # Enable (A661_TRUE) 01 # Visible (A661_TRUE)

验证了 XML 文件和 HEXA 的一致性后, 需要验证 BIN 文件和 HEXA 文件的一致性. 在 BIN 文件中找到相应的二进制内容, 如图 12 所示.

阴影部分, 便是上述控件 Check Button 的二进制表达.

通过上述验证过程, 将验证结果记录到 EXCEL 表格中, 形成测试记录, 如图 13 所示.

XML 文件的评审验证了控件参数是否符合需求, BIN 文件的加载和 UA 的功能验证, 验证了控件是否支持 UA 的操作性功能, BIN 文件和 XML 文件的一致性验证, 完善了 DF 文件的追踪性验证, 同时进行的覆盖率分析, 全面验证了 DF 文件, 最后通过形成测试记录, 提供了 DF 文件的验证证据. 使得 DF 文件的验证符合了 DO-178C 的要求, 同时又能通过 CAAC 的适航审定[5–10].

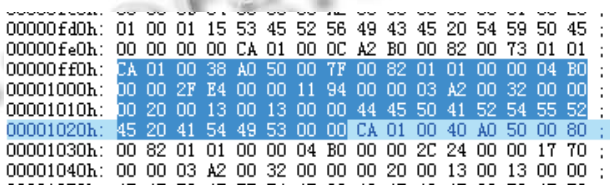

图 12 Check Button 在 BIN 文件中的表达

	A	B	C	D	E
1	Widgt Name	XML	HEXA	BIN	Result
	A661_CHECK_BUTTON	<a661_widget name="CheckButton" type="A661_CHECK_BUTTON"> <model> <prop name="WidgetIdent" value="127" /> <prop name="Enable" value="A661_TRUE" /> <prop name="Visible" value="A661_TRUE" /> <prop name="PosX" value="1200" /> <prop name="PosY" value="12260" /> <prop name="SizeX" value="4500" /> <prop name="SizeY" value="930" /> <prop name="StyleSet" value="50" /> <prop name="NextFocusedWidget" value="0" /> <prop name="MaxStringLength" value="32" /> <prop name="CheckButtonState" value="A661_UNSELECTED" /> <prop name="Alignment" value="A661_LEFT" /> <prop name="AutomaticFocusMotion" value="A661_FALSE" /> <prop name="PicturePosition" value="A661_LEFT" /> <prop name="LabelString" value="DEPARTURE ATIS" /> </model> </a661_widget>	# A661_Create_Structure (CheckButton) CA01 # A661_CMD_CREATE 0038 # CommandSize A050 # WidgetType (A661_CHECK_BUTTON) 007F # WidgetIdent 0082 # ParentIdent 01 # Enable (A661_TRUE) 01 # Visible (A661_TRUE) 000004B0 # PosX 00002FE4 # PosY 00001194 # SizeX 000003A2 # SizeY 0032 # StyleSet 0000 # NextFocusedWidget 0020 # MaxStringLength 00 # CheckButtonState (A661_UNSELECTED) 13 # Alignment (A661_LEFT) 00 # AutomaticFocusMotion (A661_FALSE) 13 # PicturePosition (A661_LEFT)	CA 01 00 38 A0 50 00 7F 00 82 01 01 00 00 04 B0 00 00 2F E4 00 00 11 94 00 00 03 A2 00 32 00 00 00 20 00 13 00 13 00 00 44 45 50 41 52 54 55 52 45 20 41 54 49 53 00 00	PASS

图 13 DF 文件验证记录

3 结论与展望

本文所阐述的基于 ARINC 661 协议的 DF 文件的验证, 解决了 UA 开发者对软件代码以外的附属交付物的验证问题. 在民用航电系统的集成测试中, IMA 上集成了几十个大大小小的软件, 并且由多个开发者分别负责一个或多个软件的开发, 对基于 ARINC 661 协议的 DF 文件的验证, 是普遍存在的问题. 使用本文的验证方式, 减轻了 CDS 开发者对来自于不同 UA 开发者提供的 DF 文件的验证工作. 按照本文的验证方式验证通过的 DF 文件, 排除了 UA 开发者引起的兼容性, 一致性和完整性的错误, CDS 开发者直接使用和加载 DF 文件, 将 DF 文件视为参数数据项进行验证. 这种 DF 文件的验证方法规范了 UA 开发者的 DF 文件的验证流程, 提高了 DF 文件的验证质量, 缩减了 CDS 的开发周期, 同时减少了 UA 和 CDS 开发者在购买 SCADE DISPLAY 的 DF 生成器上的巨大开销. 在民用航电系统快速发展的时代, 不同开发者, 站在不同角度, 使用不同方式, 共同承担一个产物的开发和验证的模式必将成为大的趋势, 这样的验证方式也将逐步趋于成熟和完善[11–13].

参考文献

1 刘天华. 民用飞机数据链通信管理技术. 电讯技术, 2010, 50(5): 84–88.

2 伊恩·莫伊尔, 阿伦·西布里奇, 马尔科姆·朱克斯. 飞机航空

电子系统. 支超有, 秦成, 译. 2 版. 北京: 国防工业出版社, 2015.
3 RTCA. RTCA/DO-178C Software considerations in airborne systems and equipment certification. RTCA, 2011.
4 ARINC. ARINC Specification 661-4, Cockpit display system interfaces to user systems. ARINC, 2010.
5 陈颖, 苑仁亮, 曾利. 航空电子模块化综合系统集成技术. 北京: 国防工业出版社, 2013.
6 赵志勇, 毛忠阳, 张嵩, 等. 数据链系统与技术. 北京: 电子工业出版社, 2014.
7 田莉蓉. 机载电子产品适航工程方法. 北京: 航空工业出版社, 2016.
8 郭艳颖, 吴洪坤, 刘志刚. 航空电子技术基础. 西安: 西北工业大学出版社, 2016.
9 霍曼. 飞速发展的航空电子. 北京: 航空工业出版社, 2007.
10 中航工业成都凯天电子股份有限公司. 机载设备适航工作指南. 北京: 航空工业出版社, 2014.
11 崔诗娴, 陈春晓, 宫伟祥. GUI 自动化测试工具在民用航空数据链系统集成中的应用. 计算机系统应用, 2016, 25(7): 66–71. [doi: 10.15888/j.cnki.csa.005249]
12 冯秋燕. 基于 UML 模型的系统级测试用例生成方法. 计算机应用, 2014, 34(1): 276–280. [doi: 10.11772/j.issn.1001-9081.2014.01.0276]
13 谷多玉, 申浩, 叶曙光, 等. 基于图的航空图像与 GIS 模型匹配算法. 计算机工程, 2003, 39(10): 187–191. [doi: 10.3321/j.issn:1002-8331.2003.10.061]