1 Introduction

This manual explains how to use the Open CASCADE Application Framework (OCAF). It provides basic documentation on using OCAF. For advanced information on OCAF and its applications, see our E-learning &Training offerings.

这个手册解释了如何使用 OpenCASCADE 应用框架（OCAF)。它提供了使用 OCAF 的基本文档。有关 OCAF 及其应用的高级信息，请参阅我们的电子学习和培训课程。

1.1 Purpose of OCAF

OCAF (the Open CASCADE Application Framework) is an easy-to-use platform for rapidly developing sophisticated domain-specific design applications. A typical application developed using OCAF deals with two or three dimensional (2D or 3D) geometric modeling in trade-specific Computer Aided Design (CAD) systems, manufacturing or analysis applications, simulation applications or illustration tools.

OCAF（the Open CASCADE Application Framework）是一个易于使用的平台，用于快速开发复杂的、特定领域中设计类应用程序。典型的应用程序开发是利用 OCAF 处理特定行业中的计算机辅助设计（CAD）系统、制造或分析应用程序、仿真应用程序或图形工具中的 2D 或 3D 几何建模。

Developing a design application requires addressing many technical aspects. In particular, given the functional specification of your application, you must at least:

开发一个设计类应用程序需要处理许多技术方面的问题。特别是，考虑到您的应用程序的功能规范，您至少必须：

• Design the architecture of the application— definition of the software components and the way they cooperate;

•设计应用程序的体系结构——软件组件的定义以及它们的协作方式

• Define the data model able to support the functionality required — a design application operates on data maintained during the whole end-user working session;

•定义能够支持所需功能的数据模型——设计应用程序运行在整个终端用户工作会话期间维护的数据上;

• Structure the software in order to:

– synchronize the display with the data — commands modifying objects must update the views;

—同步显示与数据——命令修改对象必须更新视图

– support generalized undo-redo commands — this feature has to be taken into account very early in the design process;

—支持通用的取消重做命令——这个特性在设计过程的早期就必须考虑进去

• Implement the function for saving the data — if the application has a long life cycle, the compatibility of data between versions of the application has to be addressed;

—实现保存数据的功能——如果应用程序有一个较长的生命周期，则必须解决应用程序版本之间的数据兼容性;

• Build the application user interface.

•构建应用程序用户界面。

•Architectural guidance and ready-to-use solutions provided by OCAF offer you the following benefits:

•OCAF 提供的架构指导和现成的解决方案为您提供以下好处：

• You can concentrate on the functionality specific for your application;

•您可以专注于应用程序的特定功能

• The underlying mechanisms required to support the application are already provided;
•已经提供了支持应用程序所需的底层机制
• The application can be rapidly be prototyped thanks to the coupling the other Open CASCADE Technology modules;
•由于其他 OpenCASCADE 技术模块的耦合，应用程序可以快速地构建出软件原型。
• The final application can be developed by industrializing the prototype — you don't need to restart the development from scratch.
•最终的应用程序可以通过软件原型来实现工业化的开发——您不需要从头开始重新开发
• The Open Source nature of the platform guarantees the long-term usefulness of your development.
•平台开源保证了你开发的长期有效性。
OCAF is much more than just one toolkit among many in the OpenCASCADE Object Libraries. Since it can handle any data and algorithms in these libraries – be it modeling algorithms, topology or geometry – OCAF is their logical supplement.
OCAF 不仅仅是众多 OpenCASCADE 对象库一个工具箱。因为它可以处理这些库中的任何数据和算法—无论是建模算法、拓扑关系处理还是几何建模—OCAF 是对它们逻辑上的补充。
The table below contrasts the design of a modeling application using object libraries alone and using OCAF.
下面的表格对比了单独使用对象库和使用 OCAF 的建模应用程序的设计

Table 1: Services provided by OCAF

Development tasks	Comments	Without OCAF	With OCAF
Creation of geometry	Algorithm Calling the modeling libraries	To be created by the user	To be created by the user
Data organization	Including specific attributes and modeling process	To be created by the user	Simplified
Saving data in a file	Notion of document	To be created by the user	Provided
Document-view management	To be created by the user	Provided	
Application infrastructure	New, Open, Close, Save and Save As File menus	To be created by the user	Provided
Undo-Redo	Robust, multi-level	To be created by the user	Provided
Application-specific dialog boxes	To be created by the user	To be created by the user	

OCAF uses other modules of Open CASCADE Technology — the Shape is implemented with the geometry supported by the Modeling Data module and the viewer is the one provided with the Visualization module.Modeling functions can be implemented using the Modeling Algorithms module.
OCAF 使用了 OpenCASCADE 技术中的其他模块——Shape 被应用于有建模数据模块支持的几何图形建模实现中，而 Viewer 是由可视化模块提供的。建模函数可被用于建模算法模块中。
The relationship between OCAF and the Open CASCADE Technology (OCCT) Object Libraries can be seen in the image below.
OCAF 和 OpenCASCADE 技术（OCCT）对象库之间的关系可以在下图中看到。

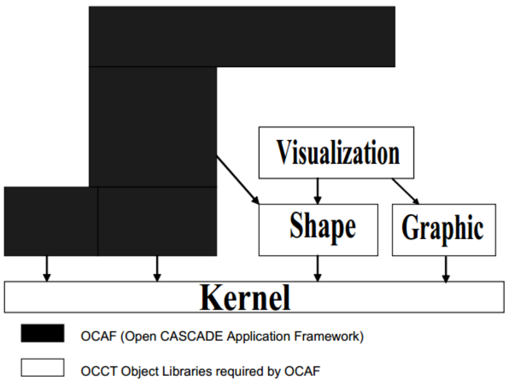
Visualization
Shape
Graphic
Kernel
OCAF (Open CASCADE Application Framework)
OCCT Object Libraries required by OCAF

Figure 1: OCCT Architecture

In the image, the OCAF (Open CASCADE Application Framework) is shown with black rectangles and OCCT Object Libraries required by OCAF are shown with white rectangles.

在图像中，OCAF（OpenCASCADE 应用程序框架）以黑色矩形和 OCAF 所要求的 OCCT 对象库显示为白色矩形。

The subsequent chapters of this document explain the concepts and show how to use the services of OCAF.

本文档的后续章节解释了这些概念并展示了如何使用 OCAF 的服务。

1.2 Architecture Overview

OCAF provides you with an object-oriented Application-Document-Attribute model consisting of C++ class libraries.

OCAF 为您提供了一个面向对象的应用程序文档-属性模型，该模型由 C++类库组成

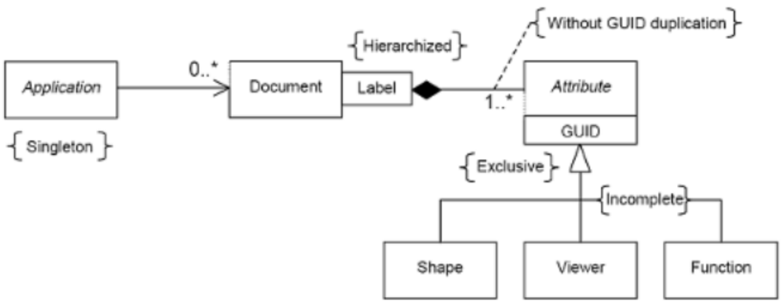

Figure 2: The Application-Document-Attribute model

1.2.1 Application

The Application is an abstract class in charge of handling documents during the working session, namely:

该应用程序是一个抽象类，负责处理工作过程中的文档，即：

• Creating new documents;

•创建新文档

• Saving documents and opening them;

•保存文档并打开文档

• Initializing document views.

•初始化文档视图

1.2.2 Document

The document, implemented by the concrete class Document, is the container for the application data. Documents offer access to the data framework and serve the following purposes:

由具体类文档实现的文档是应用程序数据的容器。文档提供对数据框架的访问，并提供以下用途：

• Manage the notification of changes

•管理更改通知

• Update external links

•更新外部链接

• Manage the saving and restoring of data

•管理数据的保存和恢复

• Store the names of software extensions.

•存储软件扩展名的名称

• Manage command transactions

•管理命令事务

• Manage Undo and Redo options.

•管理撤销和重做选项

Each document is saved in a single flat ASCII file defined by its format and extension (a ready-to-use format isprovided with OCAF).

每一个文档都保存在一个由其格式和扩展定义的单一平面 ASCII 文件中（一种可随时使用的格式由 OCAF 提供）。

Apart from their role as a container of application data, documents can refer to each other; Document A, for example,can refer to a specific label in Document B. This functionality is made possible by means of the reference key.

除了作为应用程序数据容器的角色之外，文档还可以相互引用;举例来说，文档 A 可以引用文档 B 中的特定标签，这个功能是通过引用键实现的。

1.2.3 Attribute

Application data is described by Attributes, which are instances of classes derived from the Attribute abstract class,organized according to the OCAF Data Framework.

应用程序数据是由属性描述的，它是根据 OCAF 数据框架组织的从属性抽象类派生的类的实例

The OCAF Data Framework (p. 10) references aggregations of attributes using persistent identifiers in a single hierarchy. A wide range of attributes come with OCAF, including:

OCAF 数据框架（p.10）在单一层次结构中使用持久标识符来引用属性聚合。OCAF 有很多属性，包括：

• Standard attributes (p. 37) allow operating with simple common data in the data framework (for example:integer, real, string, array kinds of data), realize auxiliary functions (for example: tag sources attribute for the children of the label counter), create dependencies (for example: reference, tree node)....;

•标准属性（p.37）允许在数据框架中使用简单的公共数据（例如：整型、实数、字符串、数组类型的数据），实现辅助功能（例如：标签计数器的子元素的标签源属性），创建依赖关系（例如：引用，树节点）。;

• Shape attributes (p. 28) contain the geometry of the whole model or its elements including reference to the shapes and tracking of shape evolution;

•形状属性（p.28）包含整个模型的几何形状或它的元素，包括对形状引用和对形状演化的追踪;

• Other geometric attributes such as Datums (points, axis and plane) and Constraints (tangent-to,

at-a-given distance, from-a-given-angle, concentric, etc.)
•其他的几何属性，比如数据（点、轴和平面）和约束（切线，给定一个距离，从一个给定的角度，同心，等等)。

• User attributes, that is, attributes typed by the application
•用户属性，也就是由应用程序输入的属性

• Visualization attributes (p. 42) allow placing viewer information to the data framework, visual representation of objects and other auxiliary visual information, which is needed for graphical data representation.
•可视化属性（第 42 页）允许将查看器信息放置到数据框架、对象的可视化表示和其他辅助可视化信息中，这对于图形化数据表示是必需的。

• Function services (p. 44) — the purpose of these attributes is to rebuild objects after they have been modified(parameterization of models). While the document manages the notification of changes, a function manages propagation of these changes. The function mechanism provides links between functions and calls to various algorithms.
•函数服务（第 44 页）—这些属性的目的是在他们被修改后来重构对象（模型的参数化)。当文档管理变更通知时，一个函数管理这些更改的传播。函数机制提供了函数和各种算法的调用之间的链接。

In addition, application-specific data can be added by defining new attribute classes; naturally, this changes the standard file format. The only functions that have to be implemented are:
此外，可以通过定义新的属性类来添加特定于应用程序的数据;当然，这会改变标准的文件格式。唯一需要实现的功能是:

• Copying the attribute
•复制的属性

• Converting it from and persistent data storage
•将它从持久的数据存储中转换

1.3 Reference-key model

In most existing geometric modeling systems, the data are topology driven. They usually use a boundary representation (BRep), where geometric models are defined by a collection of faces, edges and vertices, to which application data are attached. Examples of data include:
在大多数现有的几何建模系统中，数据是由拓扑驱动的。它们通常使用一个边界表示(BRep)，其中的几何模型是由一组面孔、边和顶点组成的，这些数据是附加在应用程序数据上的。数据的例子包括:

• a color;
•色彩

• a material;
•材质

• information that a particular edge is blended.
•特定边缘混合的信息

When the geometric model is parameterized, that is, when you can change the value of parameters used to build the model (the radius of a blend, the thickness of a rib, etc.), the geometry is highly subject to change. In order to maintain the attachment of application data, the geometry must be distinguished from other data.
当几何模型被参数化时，也就是说，当你可以改变用于构建模型的参数的值（the radius of a blend, the thickness of a rib, etc.）时，几何图形会受到很大的改变。为了维护应用程序数据的

附件，必须将几何图形与其他数据区分开来。

In OCAF, the data are reference-key driven. It is a uniform model in which reference-keys are the persistent identification of data. All accessible data, including the geometry, are implemented as attributes attached to reference keys. The geometry becomes the value of the Shape attribute, just as a number is the value of the Integer and Real attributes and a string that of the Name attribute.

在 OCAF 中，数据是由引用键驱动的。它是一种统一的模型，在这种模型中，引用键是数据的持久标识。所有可访问的数据，包括几何图形，都被实现为附加到引用上的属性。几何图形成为形状属性的值，就像一个数字是整数和实数属性的值和 Name 属性的字符串一样。

On a single reference-key, many attributes can be aggregated; the application can ask at runtime which attributes are available. For example, to associate a texture to a face in a geometric model, both the face and the texture are attached to the same reference-key.

在单个引用键上，可以聚合许多属性;应用程序可以在运行时询问哪些属性可用。例如，在几何模型中，将纹理与面联系起来，面和纹理都被连接到同一个引用键上。

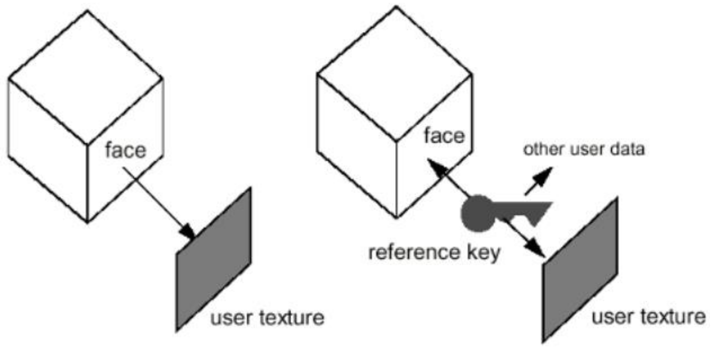

Figure 3: Topology driven versus reference-key driven approaches

Reference-keys can be created in two ways:

可以用两种方式创建引用键:

• At programming time, by the application

•在编程的时候，通过应用程序

• At runtime, by the end-user of the application (providing that you include this capability in the application)

•在运行时，由程序的终端用户（提供在应用程序中包含此功能）

As an application developer, you generate reference-keys in order to give semantics to the data. For example, a function building a prism may create three reference-keys: one for the base of the prism, a second for the lateral faces and a third for the top face. This makes up a semantic built-in the application's prism feature. On the other hand, in a command allowing the end-user to set a texture to a face he/she selects, you must create a reference-key to the selected face if it has not previously been referenced in any feature (as in the case of one of the lateral faces of the prism).

作为应用程序开发人员，为了给数据提供语义，您需要生成引用键。例如，构建棱柱的功能可以创建三个参考键：一个用于棱柱的基础，另一个用于侧面的面，另一个用于顶面。这构成了一个语义内置的应用程序的棱柱特性。另一方面,在命令行中允许终端用户设置一个他选择的纹理面，如果你此前并没有任何关于特性的引用的话，您必须创建一个引用键用来选择面（就像棱柱的侧面一样）。

When you create a reference-key to selected topological elements (faces, edges or vertices), OCAF attaches to the reference-key information defining the selected topology — the Naming attribute.

For example, it may be the faces to which a selected edge is common to. This information, as well as information about the evolution of the topology at each modeling step (the modified, updated and deleted faces), is used by the naming algorithm to maintain the topology attached to the reference-key. As such, on a parametrized model, after modifying the value of a parameter,the reference-keys still address the appropriate faces, even if their geometry has changed. Consequently, you change the size of the cube shown in the figure above, the user texture stay attached to the right face.

当您为选定的拓扑元素（面孔、边或顶点）创建一个引用键时，OCAF 附加应用信息被定义为所选择的拓扑元素—命名属性。例如，边常被选择的那些面。这个信息以及在每个建模步骤中（修改的、更新的和删除的面）拓扑结构的演变信息，都由命名算法来维护拓扑关系并附加到引用上。同样，在一个参数化模型中，在修改参数值之后，即使它们的几何图形发生了变化，引用仍然会处理相对应的面模型。因此，您改变了上图中所示的立方体的大小，用户纹理依然会附着在正确的面上。

Note As Topological naming is based on the reference-key and attributes such as Naming (selection information)and Shape (topology evolution information), OCAF is not coupled to the underlying modeling libraries. The only modeling services required by OCAF are the following:

注意，由于拓扑命名基于引用键和属性，如命名（选择信息）和形状（拓扑演化信息），OCAF 并没有与底层的建模库耦合。OCAF 所需要的唯一的建模服务如下：

• Each algorithm must provide information about the evolution of the topology (the list of faces modified, updated and deleted by the algorithm)

•每个算法都必须提供关于拓扑演化的信息（算法进行面模型修改、更新和删除的列表）。

• Exploration of the geometric model must be available (a 3D model is made of faces bounded by close wires,themselves composed by a sequence of edges connected by their vertices)

•对几何模型的探索必须是可行的（3D 模型是由闭合线框为边界的面构成的，而面本身是由顶点所连接而成的一系列边组成）

Currently, OCAF uses the Open CASCADE Technology modeling libraries.

目前，OCAF 使用 Open CASCADE 技术建模库。

To design an OCAF-based data model, the application developer is encouraged to aggregate ready-to-use attributes instead of defining new attributes by inheriting from an abstract root class. There are two major advantages in using aggregation rather than inheritance:

为了设计一个基于 OCAF 为基础的数据模型，我们鼓励应用程序开发人员聚合现成的属性，而不是通过继承抽象的基类来定义新属性。使用聚合而不是继承有两个主要优点：

• As you don't implement data by defining new classes, the format of saved data provided with OCAF doesn't change; so you don't have to write the Save and Open functions

•由于不通过定义新类来实现数据，所以用 OCAF 提供的保存数据的格式不会改变;所以你不需要写保存和打开的函数

• The application can query the data at runtime if a particular attribute is available

•如果某个特定属性可用，应用程序可以在运行时查询数据

Summary

• OCAF is based on a uniform reference-key model in which:

• OCAF 是基于一个统一的引用模型，在这个模型中：

– Reference-keys provide persistent identification of data;

-引用键提供持久的数据识别;

– Data, including geometry, are implemented as attributes attached to reference-keys;

-包括几何在内的数据被实现为附加到引用键的属性;

－ Topological naming maintains the selected geometry attached to reference-keys in parametrized models;

-在参数化模型中， 拓扑命名维护被选择的几何参数;

• In many applications, the data format provided with OCAF doesn’t need to be extended;

•在许多应用程序中，OCAF 提供的数据格式不需要扩展;

• OCAF is not coupled to the underlying modeling libraries.

• OCAF 与底层的建模库没有耦合。