内核架构与事务

MFG事业群/共性技术研究院/几何内核技术中心

太志伟

2024年7月30日

什么是“内核”?

- 市场上的主流内核架构
- 内核的事务系统

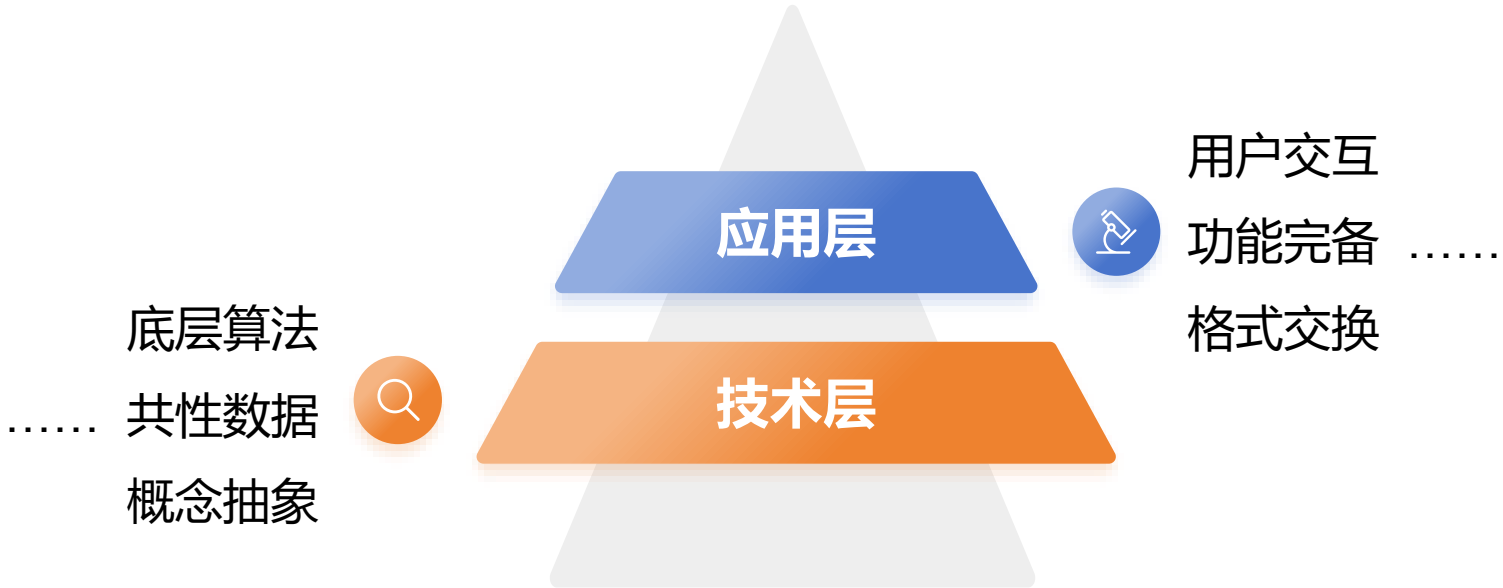
主流内核架构

Part 1

内核：CAx软件的底座

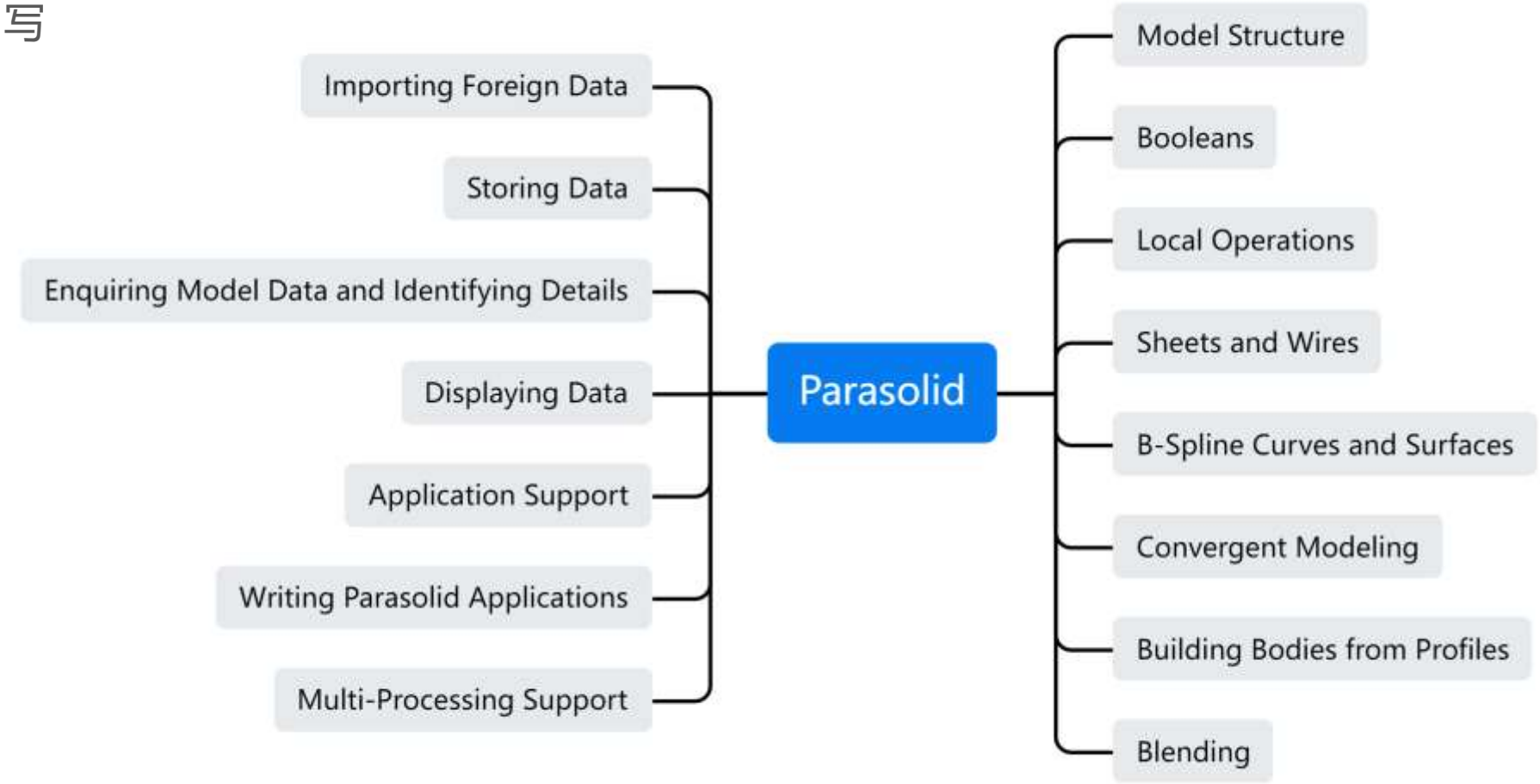
Others....

应用/技术



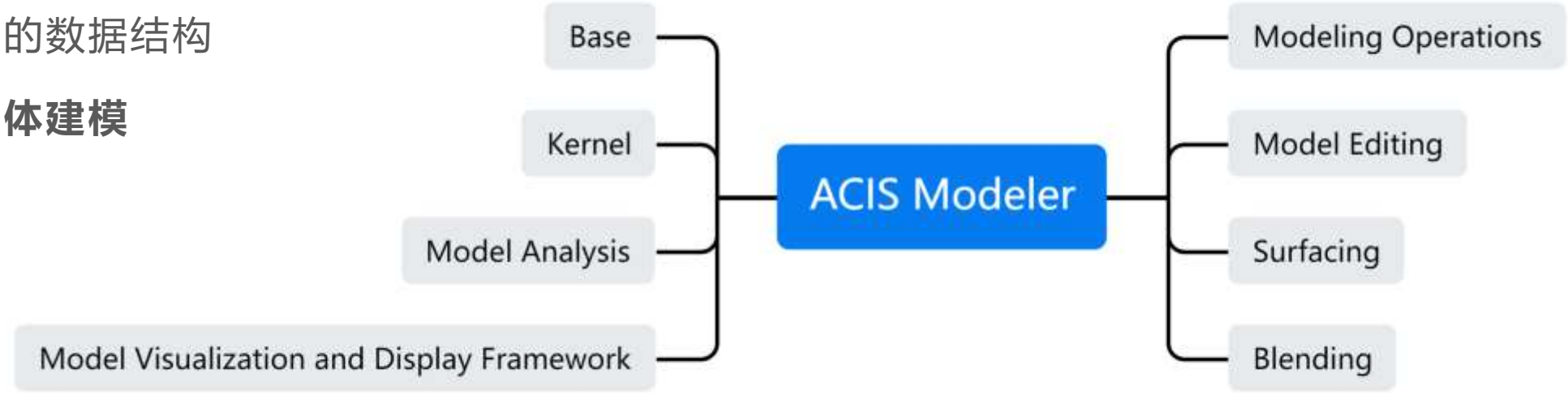
Parasolid

- 80年代至今，C语言编写
- 过程式API
- What is Parasolid
  - **实体建模**
  - **局部操作**
  - **曲面建模**
  - **收敛建模**
  - **应用支持**



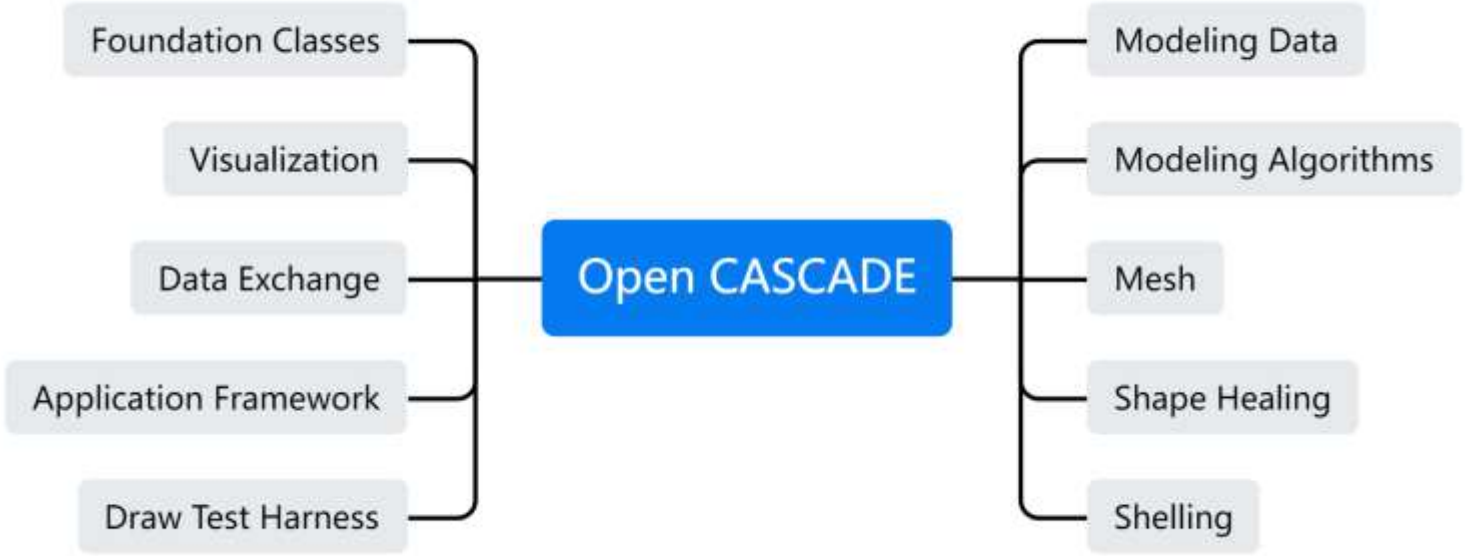
ACIS

- 89年发布初版，C++编写
- 面向对象的三维几何/实体建模引擎
- 基于**一致、统一**的数据结构
- **线框、曲面和实体建模**
- **开放**的体系架构



OpenCascade Technology

- 99年母公司被收购后开源
- 面向对象的C++类库
- 支持**曲线曲面、实体建模、应用框架**
- 建模功能稍弱，主要是CAE用户



Coding with Parasolid

```
// 创建第一个block
PK_BODY_t block1;
PK_BODY_create_solid_block(10, 10, 10, nullptr, &block1);

// 创建第二个block
PK_BODY_t block2;
PK_BODY_create_solid_block(5, 15, 20, nullptr, &block2);

// 进行布尔并运算
PK_BODY_boolean_o_t options;
PK_BODY_boolean_o_m(options);
options.function = PK_boolean_unite_c;
int n_results;
PK_BODY_t* results = nullptr;
PK_ERROR_t err = PK_BODY_boolean(block1, 1, &block2, &options,
    &n_results, &results);
PK_BODY_t united_ody = results[0];
```

- PK_BODY_t为一个int，不暴露对象指针
- C风格，面向过程
- 依靠接口返回值表示执行状态

Coding with ACIS

```
// 创建第一个block
BODY* block1 = nullptr;
outcome out = api_solid_block(
    SPAposition(0.0, 0.0, 0.0), SPAposition(1.0, 1.0, 1.0), block1
);
EXPECT_TRUE(out.ok());

// 创建第二个block
BODY* block2 = nullptr;
out = api_solid_block(
    SPAposition(0.0, 0.0, 0.0), SPAposition(1.0, 1.0, 1.0), block1
);
EXPECT_TRUE(out.ok());

// 进行布尔并运算
BOOL_TYPE bool_type = BOOL_TYPE::UNION;
out = api_boolean(block1, block2, bool_type);
EXPECT_TRUE(out.ok());
```

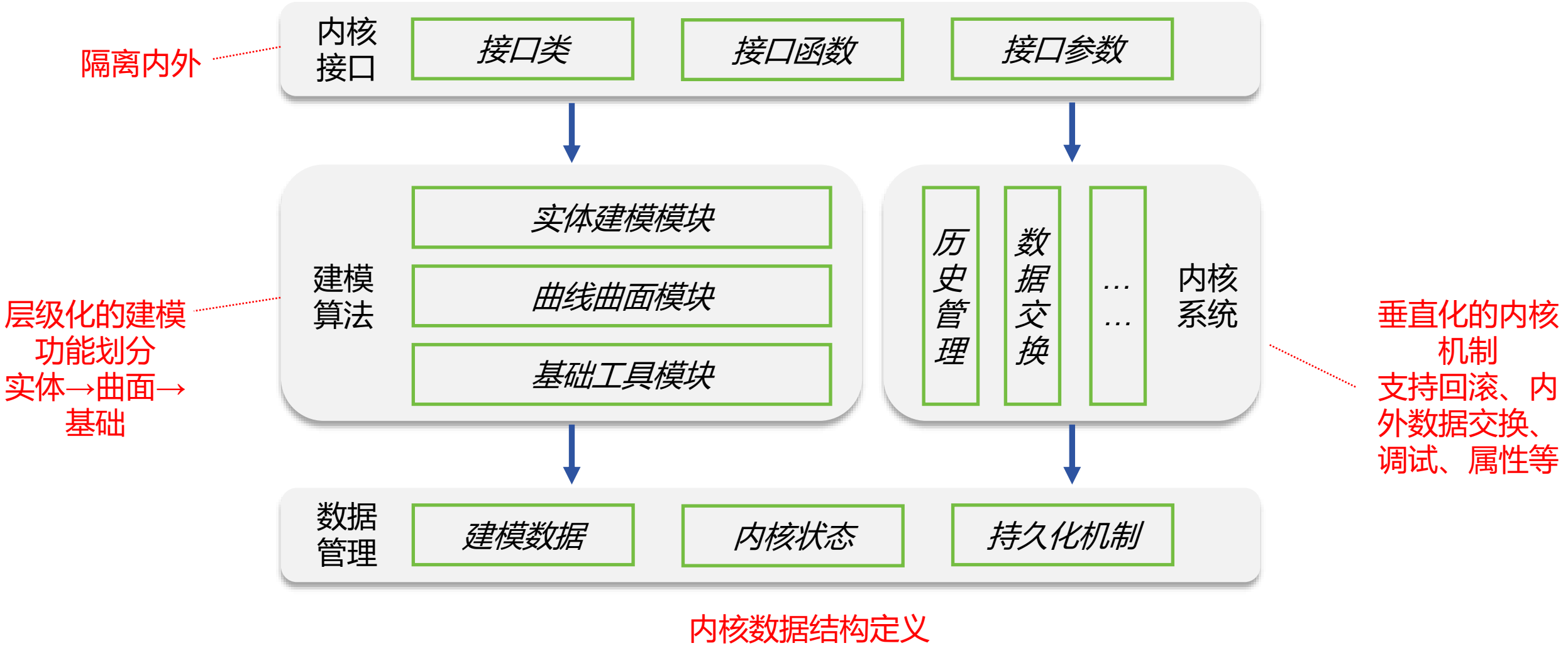
- C++，面向对象（C with Class）
- 接口操作Entity指针，抛出异常
- 提供了用户自定义api的能力

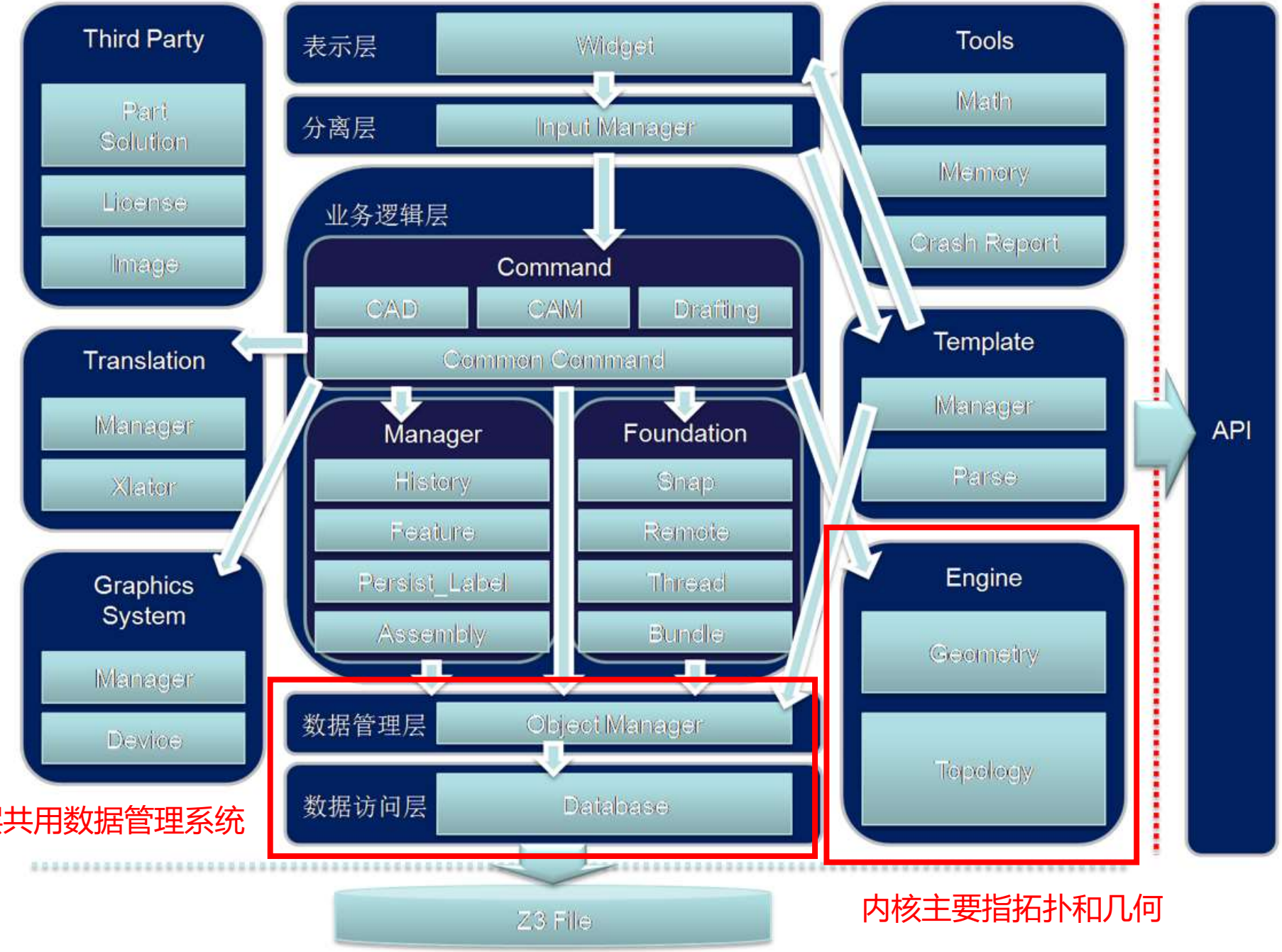
Coding with OCCT

```
// 创建第一个block
gp_Pnt aPnt1(0, 0, 0);
Standard_Real aXSize1 = 100;
Standard_Real aYSize1 = 50;
Standard_Real aZSize1 = 30;
BRepPrimAPI_MakeBox aBoxMaker1(aPnt1, aXSize1, aYSize1, aZSize1);
TopoDS_Shape aBox1 = aBoxMaker1.Shape();

// 创建第二个block
gp_Pnt aPnt2(50, 20, 10);
Standard_Real aXSize2 = 60;
Standard_Real aYSize2 = 70;
Standard_Real aZSize2 = 80;
BRepPrimAPI_MakeBox aBoxMaker2(aPnt2, aXSize2, aYSize2, aZSize2);
TopoDS_Shape aBox2 = aBoxMaker2.Shape();

// 进行布尔并运算
BRepAlgoAPI_Fuse aFuseMaker(aBox1, aBox2);
TopoDS_Shape aResultShape = aFuseMaker.Shape();
```

- C++，面向对象
- 代码风格统一，一切皆对象

主流内核架构

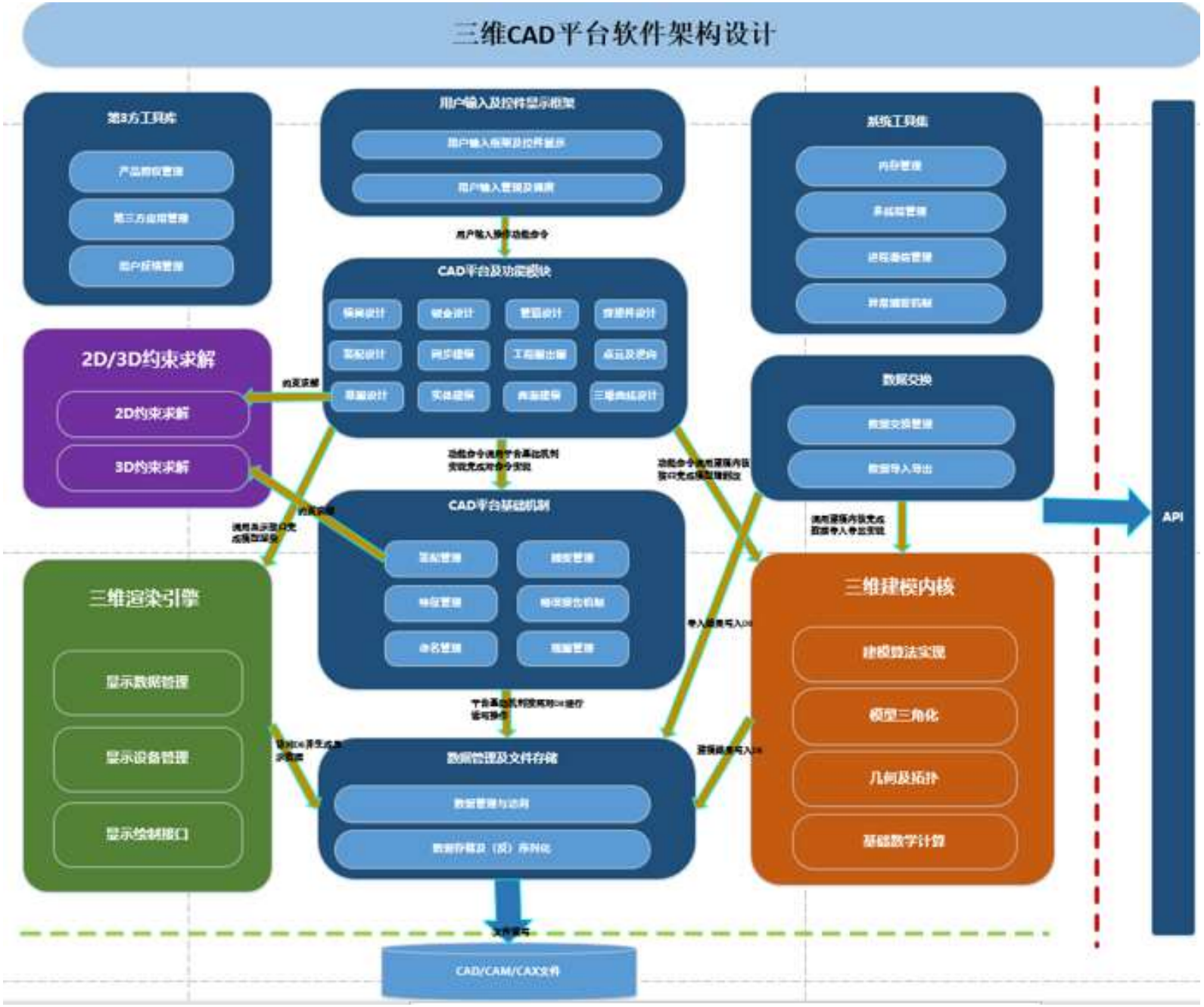

隔离内外
内核接口
接口类
接口函数
接口参数
层级化的建模
功能划分
实体→曲面→基础
建模算法
实体建模模块
曲线曲面模块
基础工具模块
历史管理
数据交换
……
内核系统
垂直化的内核机制
支持回滚、内外数据交换、调试、属性等
数据管理
建模数据
内核状态
持久化机制
内核数据结构定义



底层和上层共用数据管理系统

内核主要指拓扑和几何

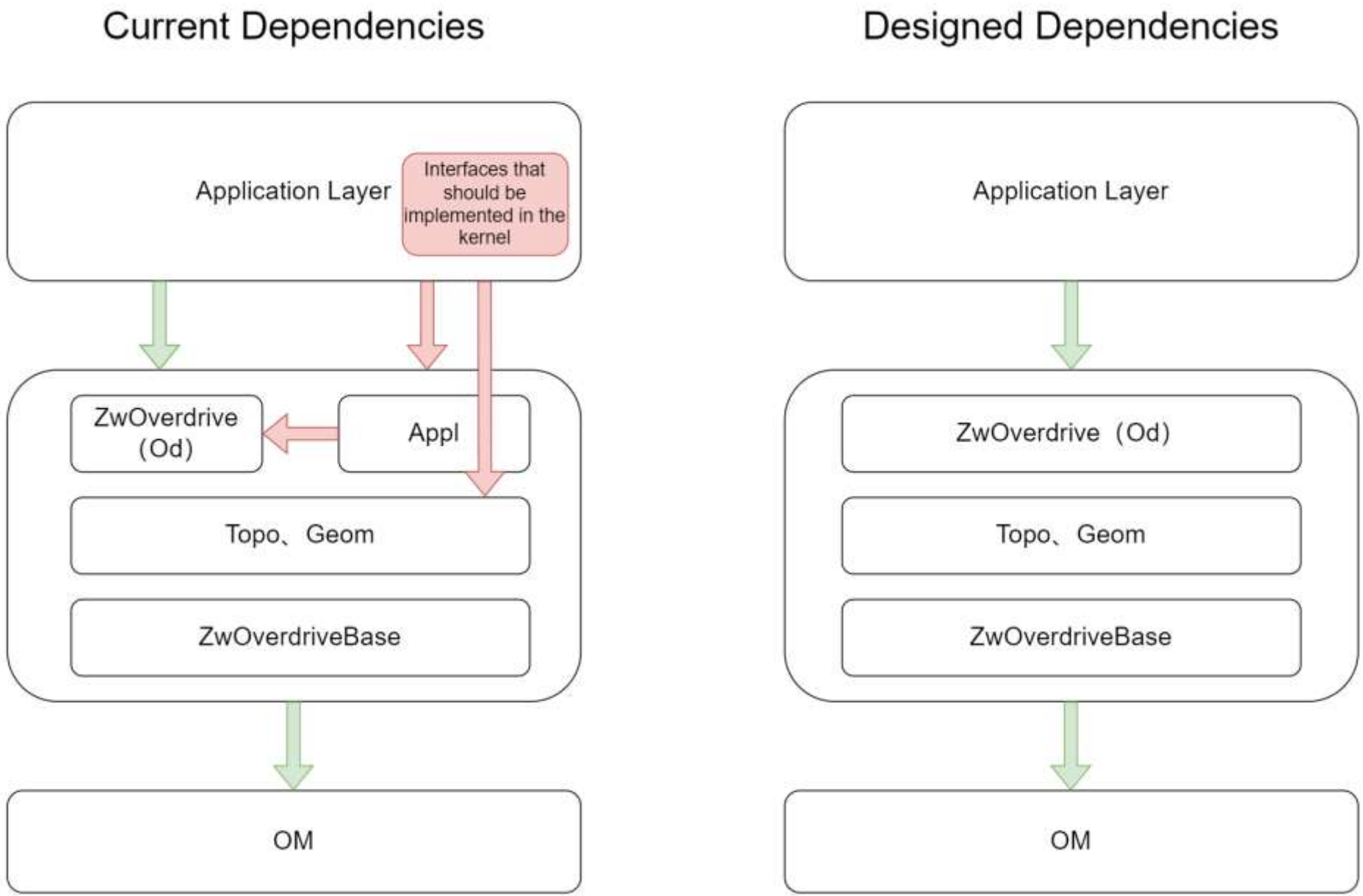
2022年新员工培训《ZW3D软件架构》



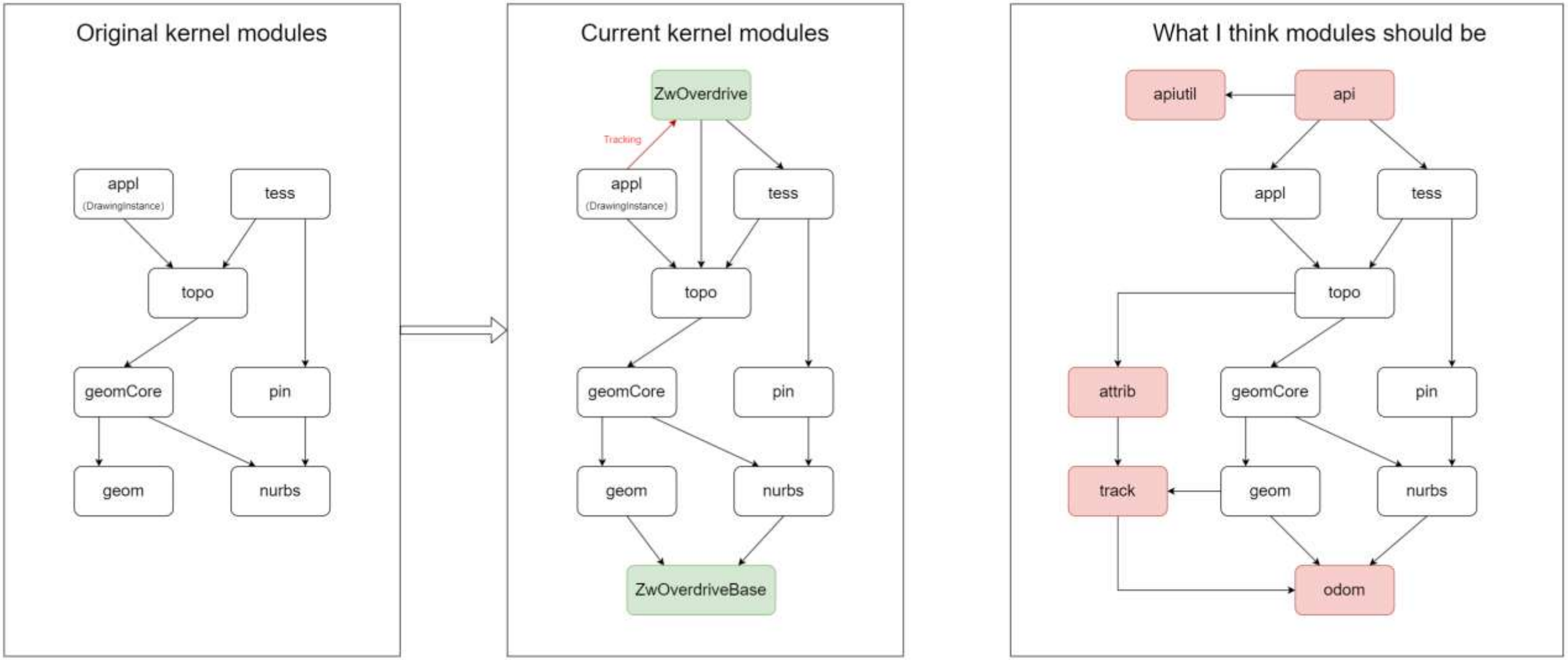
建模算法相关的函数定义为内核

ZW3D的压力

- BUG解决推进困难
- 竞争力弱于主流CAD软件
- 项目对几何建模核心能力提出要求



《What’s New in FM Kernel》https://zwiki.zwcax.com/display/Platformarchitecture/What%27s+New+in+FM+Kernel



《What's New in FM Kernel》https://zwiki.zwcax.com/display/Platformarchitecture/What%27s+New+in+FM+Kernel

何以为“内核”

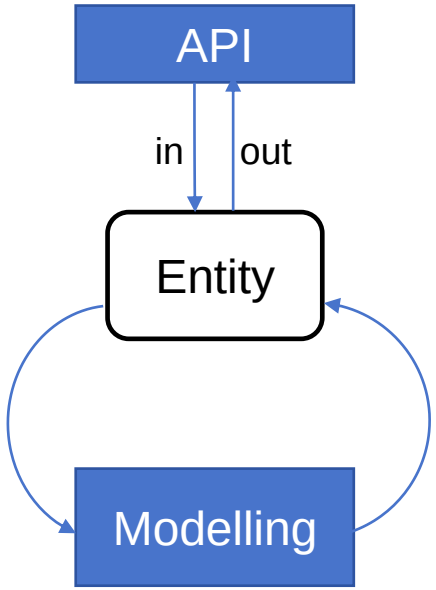
- 操作系统内核：隔离**计算机硬件**的复杂性
- 几何建模内核：隔离**建模算法和建模数据**的复杂性

内核事务/历史机制

Part 2

内核数据管理：历史与回滚

如果不管理建模历史数据——“计算器”



× Undo/Redo
× 错误处理
× 重生成
× ……

事务 / 历史

- 事务（Transaction）
  - 数据库领域，对数据库的一组不可再分的**操作**集合
  - ACID：Atomicity，Consistency，Isolation，Durability
  - **ZW3D、OCCT**

- 历史（History）
  - 在建模数据发生修改前进行备份，产生历史建模**数据**
  - 历史数据之间可以滚动，代表了回滚、前滚
  - **Parasolid、ACIS**

- 相同：内核需要有能力记录并回滚Entity的变化
- 不同：事务侧重操作，历史侧重数据

内核事务/历史机制

1. 如何记录单个Entity的变化?

2. 如何记录所有Entity的某一状态?

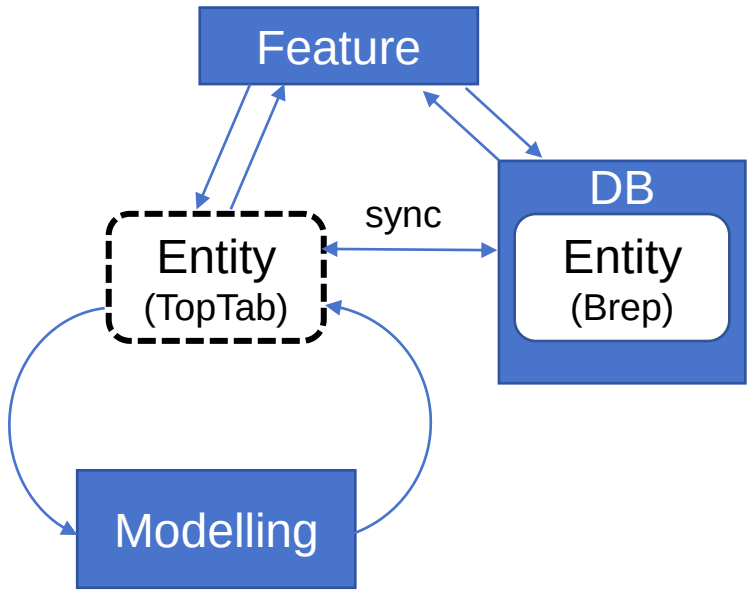
3. 如何更新所有Entity到指定状态?

ZW3D数据结构

- ZW3D的数据引擎主要由**OM、DB**模块定义
- 推荐阅读：
  - 《模块发展》，石一琳，2023.01.13
  - 《OM/DB的常见概念及全局变量介绍》，石一琳，2023.01.13
  - 《事务的记录与提交》，蒙祖锰，2023.2.16
  - 《ZW3D的Undo与Redo》，王鹏强，2023.6.6
  - 《拓扑表结构》，黄恺斌，2023.5.12
  - 《拓扑表入库》，黄恺斌，2023.1.12
  - 《拓扑表出库》，朱光，2023.1.13

ZW3D数据结构

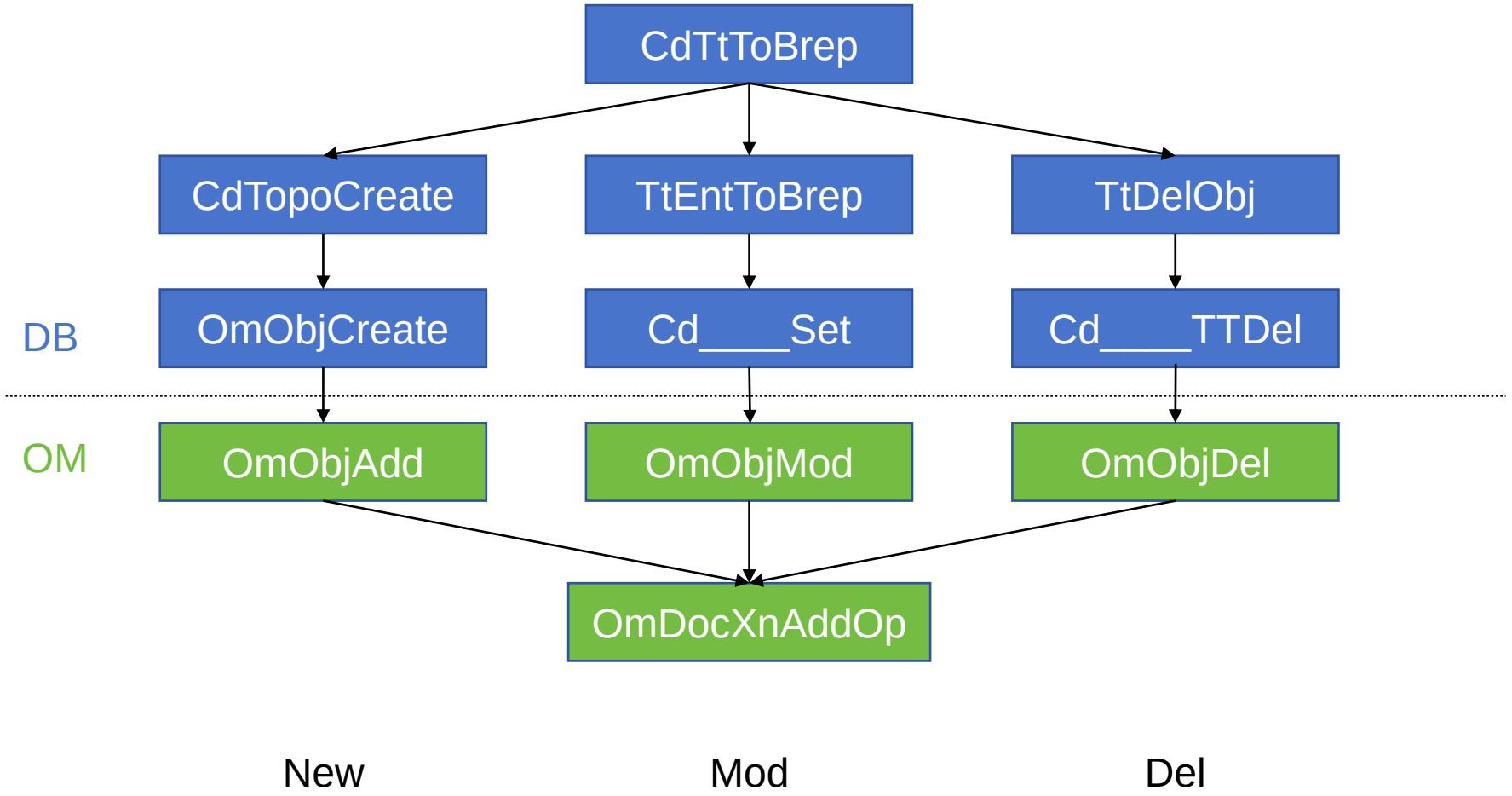
- DB内外**两套数据结构**：TT（TopTable）- 建模；Brep - 持久化
  - TT → Brep：根据变化类型修改DB对象
  - Brep → TT：基于DB对象加载拓扑表数据



ZW3D事务模型

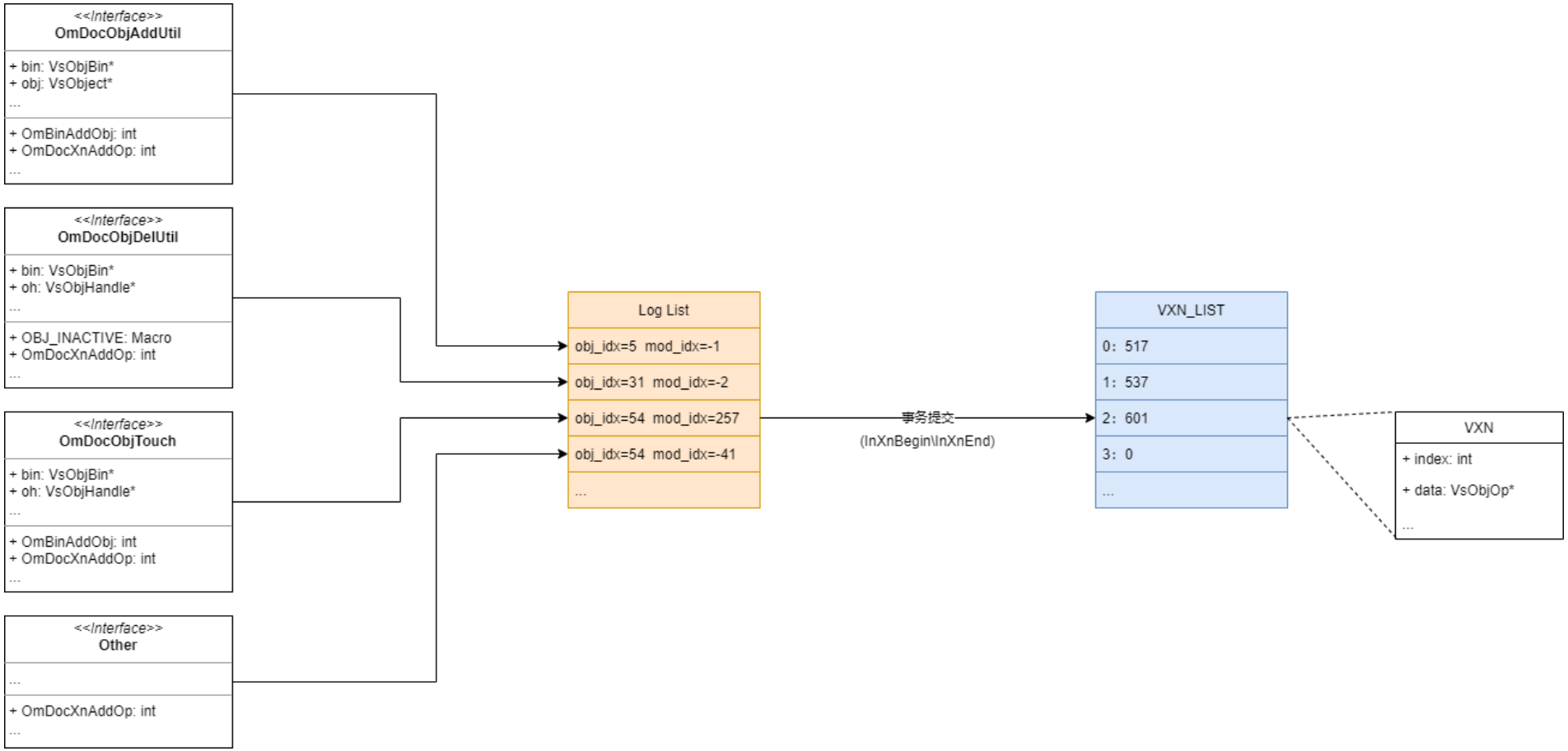
ZW3D事务机制

1. 单个Entity的变化：由CdTtToBrep管理



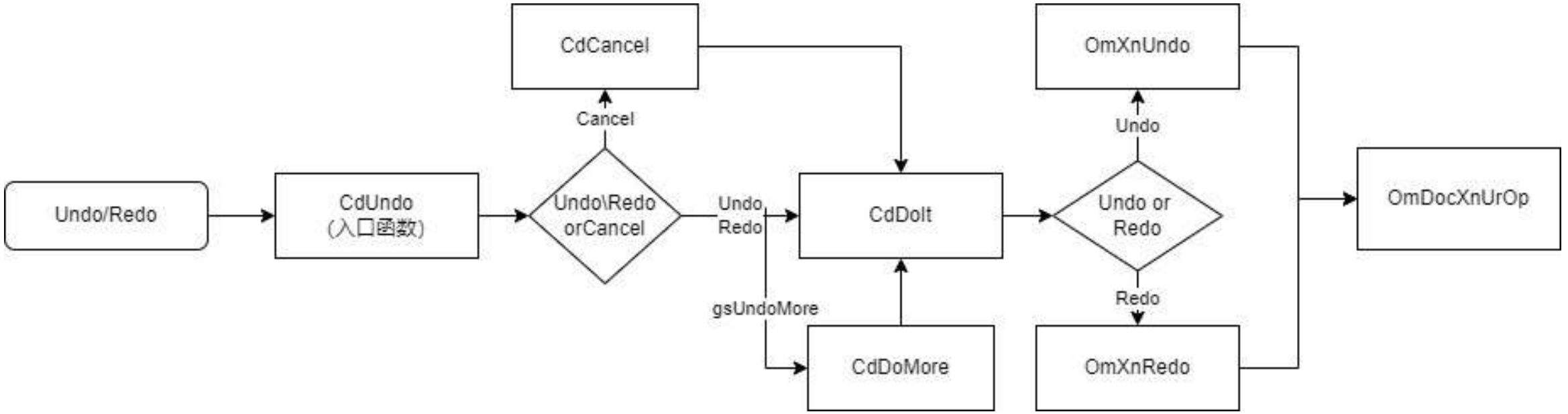
ZW3D事务机制

1. 单个Entity的变化：由CdTtToBrep管理
2. Entity的某一状态：多个事务组合后的状态



ZW3D事务机制

1. 单个Entity的变化：由CdTtToBrep管理
2. Entity的某一状态：多个事务组合后的状态
3. 状态之间切换：Undo/Redo



主流内核数据结构

- 接口直接操作单个Entity的指针/索引
- Entity之间以**侵入式链表**的形式组织

```
class Edge
{
public:
    std::vector<int> halfedge_indices;
    int vertex_indices[2];
};

class Vertex
{
public:
    std::vector<int> edges;
};

class TopTable
{
public:
    std::vector<Shape> shapes;
    std::vector<Shell> shells;
    std::vector<Loop> loops;
    std::vector<Edge> edges;
    std::vector<Vertex> vertices;
};
```

TopTable（简化）

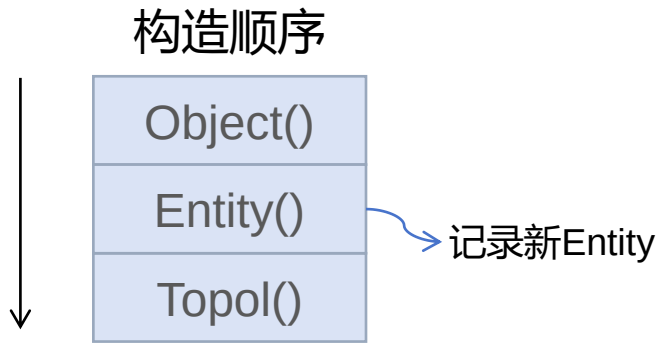
```
class Body : public Topol
{
public:
    Region* region;
    //Shell* shell;
    //Edge* edge;
    //Vertex* vertex;
};

class Region : public Topol
{
public:
    Body* body;
    Region* previous;
    Region* next;
    Shell* shell;
};

class Shell : public Topol
{
public:
    Region* region;
    Shell* next;
    Face* face;
    //Edge* edge;
    //Vertex* vertex;
};
```

Parasolid Topo Entity（简化）

- 可直接访问和遍历
- 拓扑结构的局部修改
- 操作聚焦单个Entity

主流内核历史机制

1. 单个Entity的变化：统一变化时机
   - new：构造函数（ACIS、OCC）；统一接口（Parasolid）
   - mod：Set成员函数（ACIS、OCC）；宏（Parasolid）
   - del：统一接口



New

```
int Topol::ChangeField2(double val) const
{
    if (is_partition_roll_on)
    {
        Backup();
    }
    field_2 = val;
    return 0;
}
```

记录Entity修改前数据

Mod

```
int DeleteEntity(Entity* entity)
{
    RemoveEntityFromXXX(entity);
    entity->Lose();
    return 0;
}
```

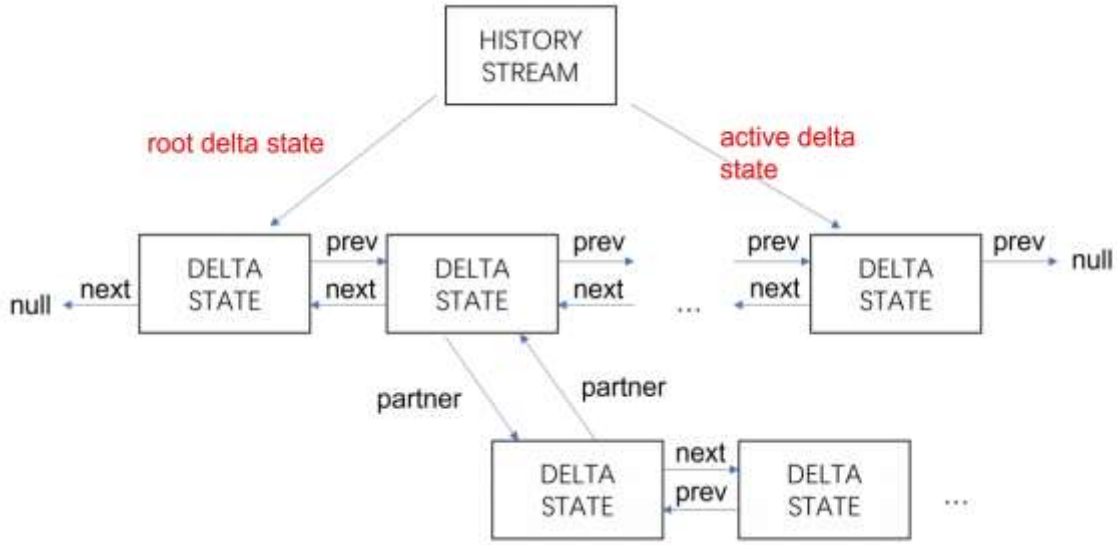
转移Entity，不直接释放内存

Del

主流内核历史机制

1. 单个Entity的变化：统一变化时机
2. Entity的某一状态：由上层应用调用接口
   - ACIS：api_note_state
   - Parasolid：PK_PARTITION_make_pmark
   - OCCT：TDF_Delta::OpenTransaction; TDF_Delta::CommitTransaction

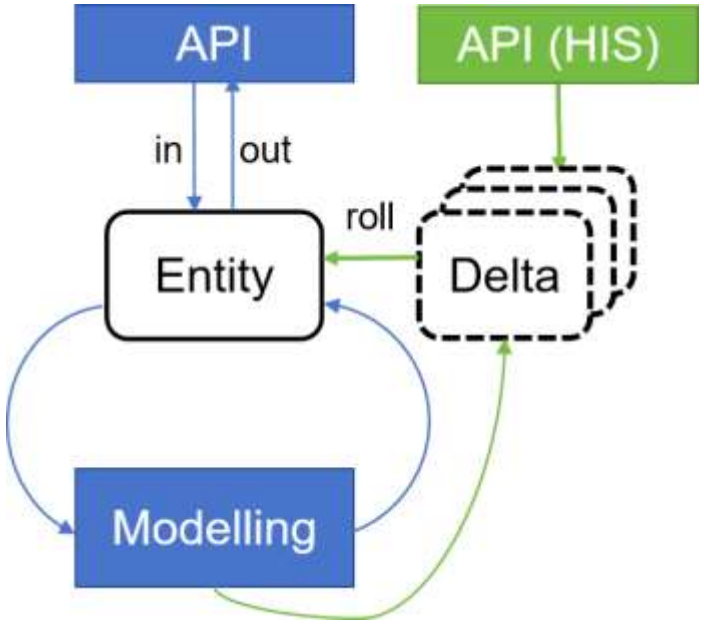
```
void do_something()
{
  // 生成两个球体
  api_make_sphere(...);
  api_make_sphere(...);
  api_note_state (...);

  // 生成一个立方体
  api_make_cuboid(...);
  api_note_state (...);
}
```



ACIS Delta State

主流内核历史机制

1. 单个Entity的变化：统一变化时机
2. Entity的某一状态：由上层应用调用接口
3. 状态之间切换：提供接口
    - ACIS：DELTA_STATE::Roll()
    - Parasolid：PK_PMARK_goto
    - OCCT：TDF_Data::Undo



ZW3D HQR（History Quick Rollback）

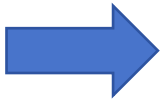
- 用于特征快速回滚（CAD层）
- 实现参考了Undo/Redo机制，同时依赖了DB、OM中的一些机制或流程
- 和主流内核的历史机制理念相似

1. 单个Entity的变化：特征执行中，在DB和OM层收集增删改
2. Entity的某一状态：特征结束后，压缩BRep数据，存入DB
3. 状态之间切换：CdFtrQuickRollback，回滚到指定Feature状态

ZW3D Feature Modeling Framework

- 去年底开启的ZW3D重构规划中的初期阶段
- 吸纳了内核独立化思想，重新设计**数据结构、数据管理、事务机制**
- 数据结构：加强类型系统

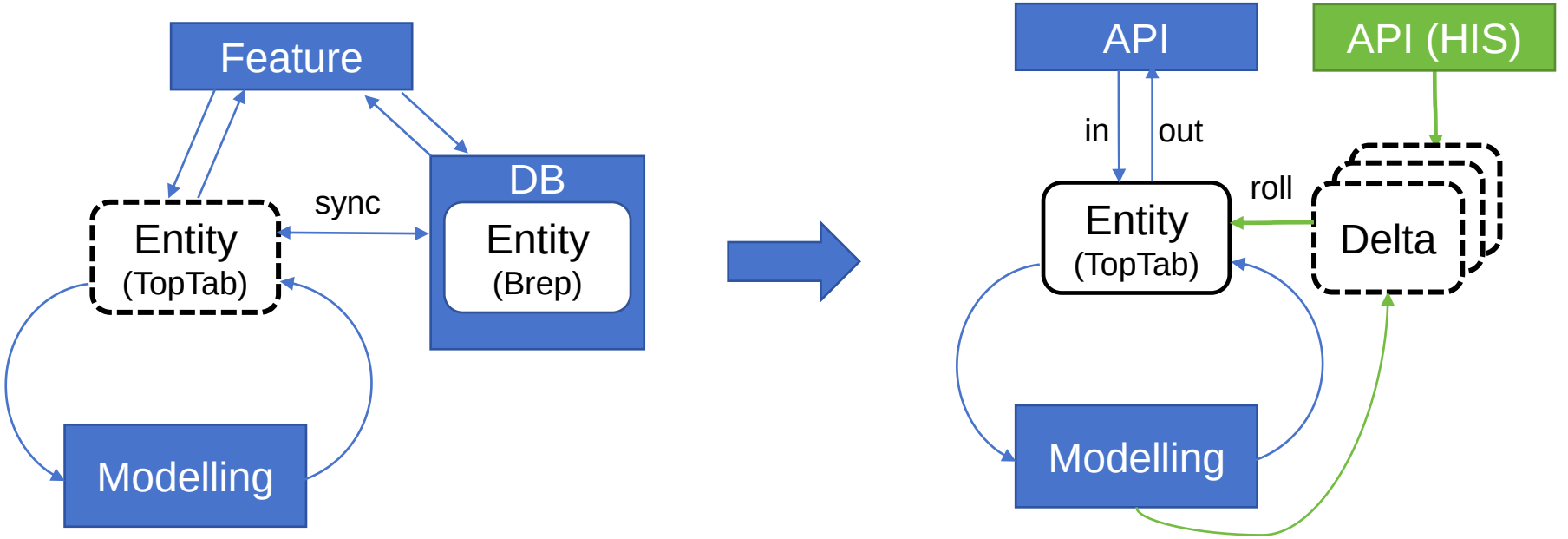
```
typedef struct VsEdge
{
    /* Geometry information. */
    ZS_TOPO_VAR_PRIV int db_idx;      /* location in the database index list */
    ZS_TOPO_VAR_PRIV mod_flag mod;    /* Modification flag. */
    ZS_TOPO_VAR_PRIV VsAppData data; /* application data */
    void *pzHeadAttr;                 /* propagable attribute circular list */
    ZS_TOPO_VAR_PRIV double dtol_accessed_only_through_function_calls;
                                      /* largest distance (D1) from the unrefi
                                         refined edge evaluation plus the gap
                                         evaluation. "tol = 0.0" for an exact e
    ZS_TOPO_VAR_PRIV VsLim1 interval; /* Domain of definition. */
    ZS_TOPO_VAR_PRIV int curv;        /* location in the local index list */
    ZS_TOPO_VAR_PRIV edge_type type;  /* Edge type. */

    /* Topology information. */
    ZS_TOPO_VAR_PRIV VsListObj *list_img; /* List of indices to all pre-edges,
                                             slot number.  (VsXRef2 structures).
                                             the loop and the second is the pre-
    ZS_TOPO_VAR_PRIV void *edg_rep;       /* sampled 3D edge data */
    ZS_TOPO_VAR_PRIV int vrtx_indx[2];    /* Index into vertex table. */
    ZS_TOPO_VAR_PRIV int db_idx2;         /* used during 2D sectioning to keep t
                                             of the original during copying (it
                                             the end of the command) */
```



```
class ZS_TOPO_API VsEdgePrototype : public TopologyObject
{
    DB_OBJECT_DECLARE_DERIVED(VsEdgePrototype, TopologyObject);
    DB_OBJECT_VERSION_DEFINE(1); // 3000
    DBOBJECT_MEMBER_FUNCTION_DECLARE;

public:
    explicit VsEdgePrototype(const ZwRef<Partition> &pPartition, mod_flag eMode = ADDED);
    virtual ~VsEdgePrototype() = default;

public:
    // get method
    edge_type get_type() const;
    const VsLim1 &get_interval() const;
    double get_tolerance() const;
    ZwRef<ZsGeomCurve> get_curve() const;
    om_const_vector<ZwRef<VsPreEdgePrototype>> get_pre_edges() const;
    void *get_edge_rep() const;
    ZwRef<VsVertexPrototype> get_first_vertex() const;
    ZwRef<VsVertexPrototype> get_second_vertex() const;
    char get_exact_flag() const;

    // set method
    void set_type(edge_type t);
    void set_interval(const VsLim1 &lim1);
    void set_tolerance(double dTol);
```

ZW3D Feature Modeling Framework

- 去年底开启的ZW3D重构规划中的初期阶段
- 吸纳了内核独立化思想，重新设计**数据结构、数据管理、事务机制**
- 数据结构：加强类型系统
- 数据管理：保持一套Entity数据结构
- 事务机制：实现内核事务/历史机制



ZW3D Feature Modeling Framework

1. 单个Entity的变化：统一变化时机
    - new：构造函数
    - mod：set方法
    - del：ZwDelete
2. Entity的某一状态：OdPartition::create_pmark
3. 状态之间切换：XnContainer::goto_mark

- 建模算法改动较大，对Entity的修改和访问都通过**BaseOperation**和**Handle**进行

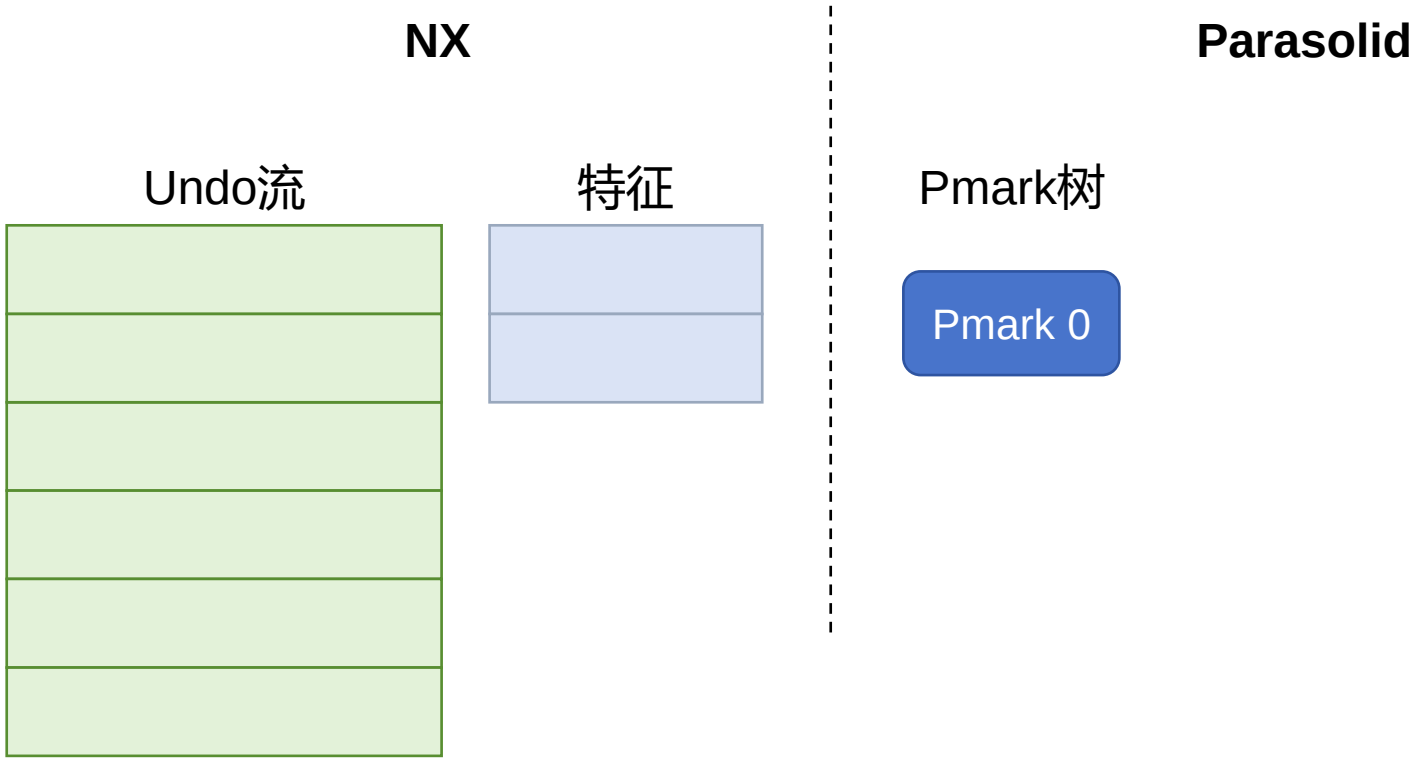
总结

- 主流内核架构
  - **应用层 - 技术层：定好边界**
  - 主流内核架构
  - ZW3D架构变迁

- 内核事务/历史机制
  - **事务/历史概念：三个问题**
  - ZW3D事务机制
  - 主流内核历史机制
  - HRQ、FMF

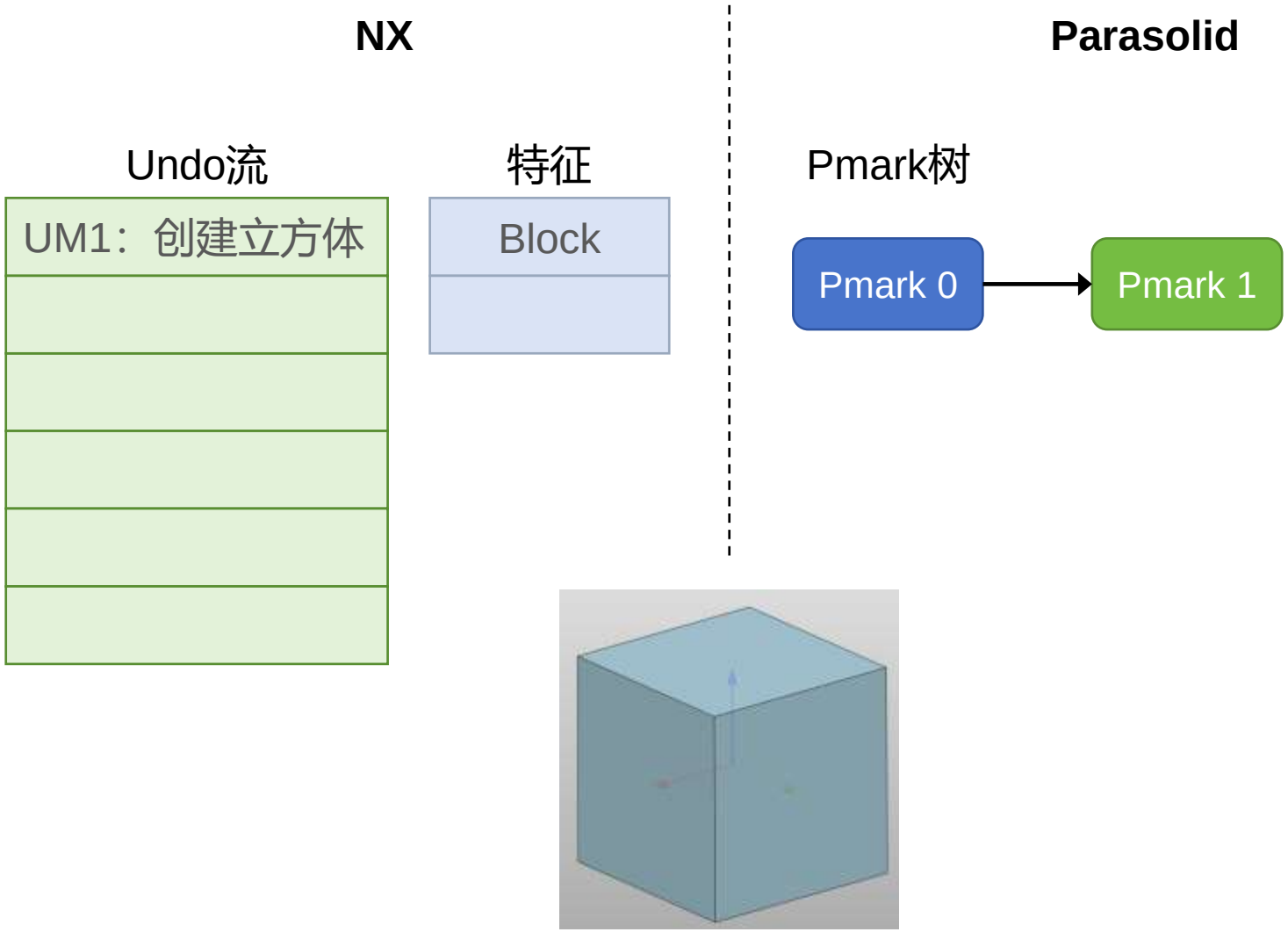
NX

- 初始状态



NX

- 初始状态
- 创建立方体



NX

- 初始状态
- 创建立方体
- 做圆角

NX

Undo流

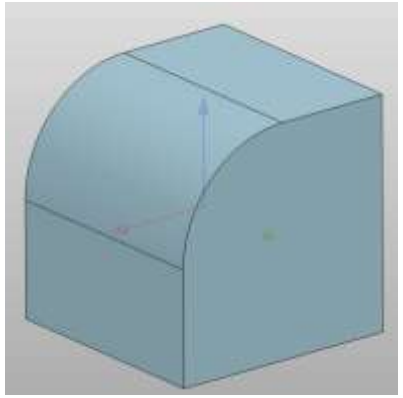
UM1：创建立方体
UM2：圆角

特征

Block
Blend

Parasolid

Pmark树



NX

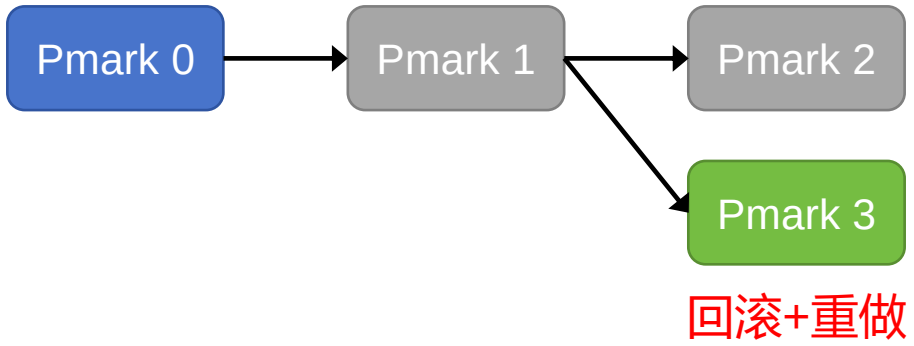
- 初始状态
- 创建立方体
- 做圆角
- 编辑圆角特征

NX

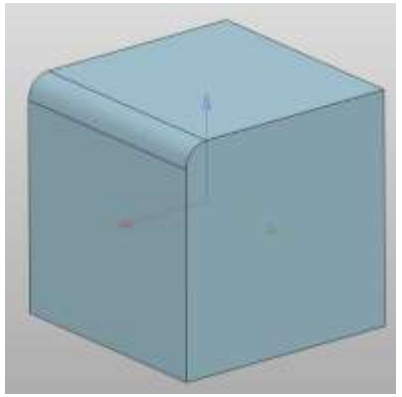
Undo流	特征
UM1：创建立方体	Block
UM2：圆角	Blend
UM3：编辑圆角	

Parasolid

Pmark树



回滚+重做



NX

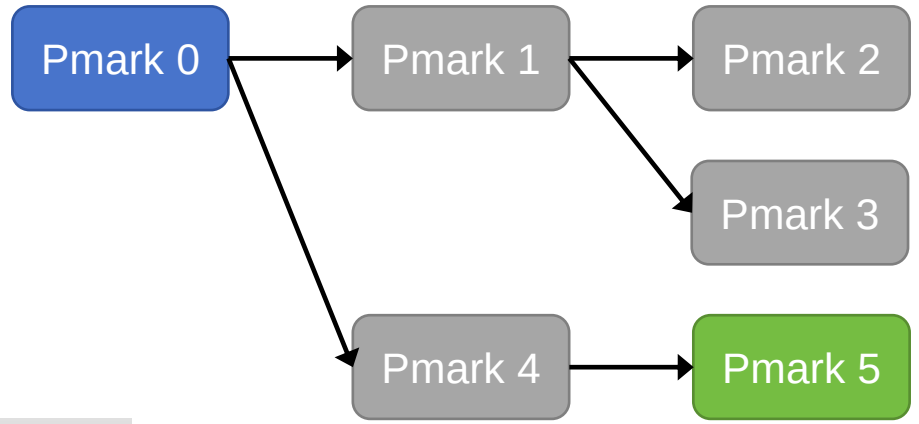
- 初始状态
- 创建立方体
- 做圆角
- 编辑圆角特征
- 编辑立方体特征

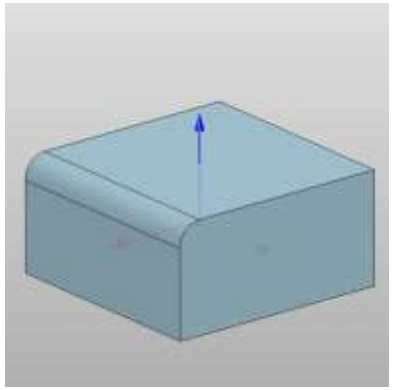
NX

Undo流	特征
UM1：创建立方体	Block
UM2：圆角	Blend
UM3：编辑圆角	
UM4：编辑立方体	

Parasolid

Pmark树





NX

- 初始状态
- 创建立方体
- 做圆角
- 编辑圆角特征
- 编辑立方体特征
- Undo

NX

Undo流

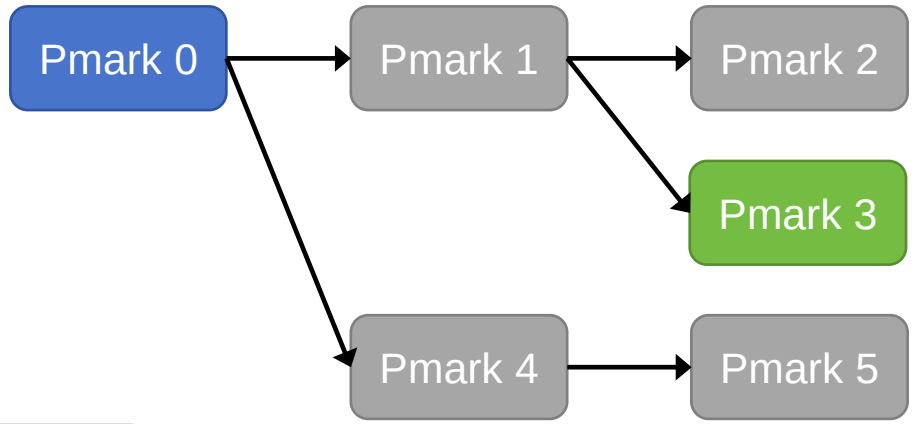
UM1：创建立方体
UM2：圆角
UM3：编辑圆角
~~UM4：编辑立方体~~

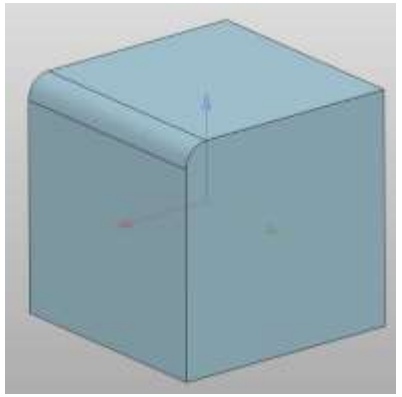
特征

Block
Blend

Parasolid

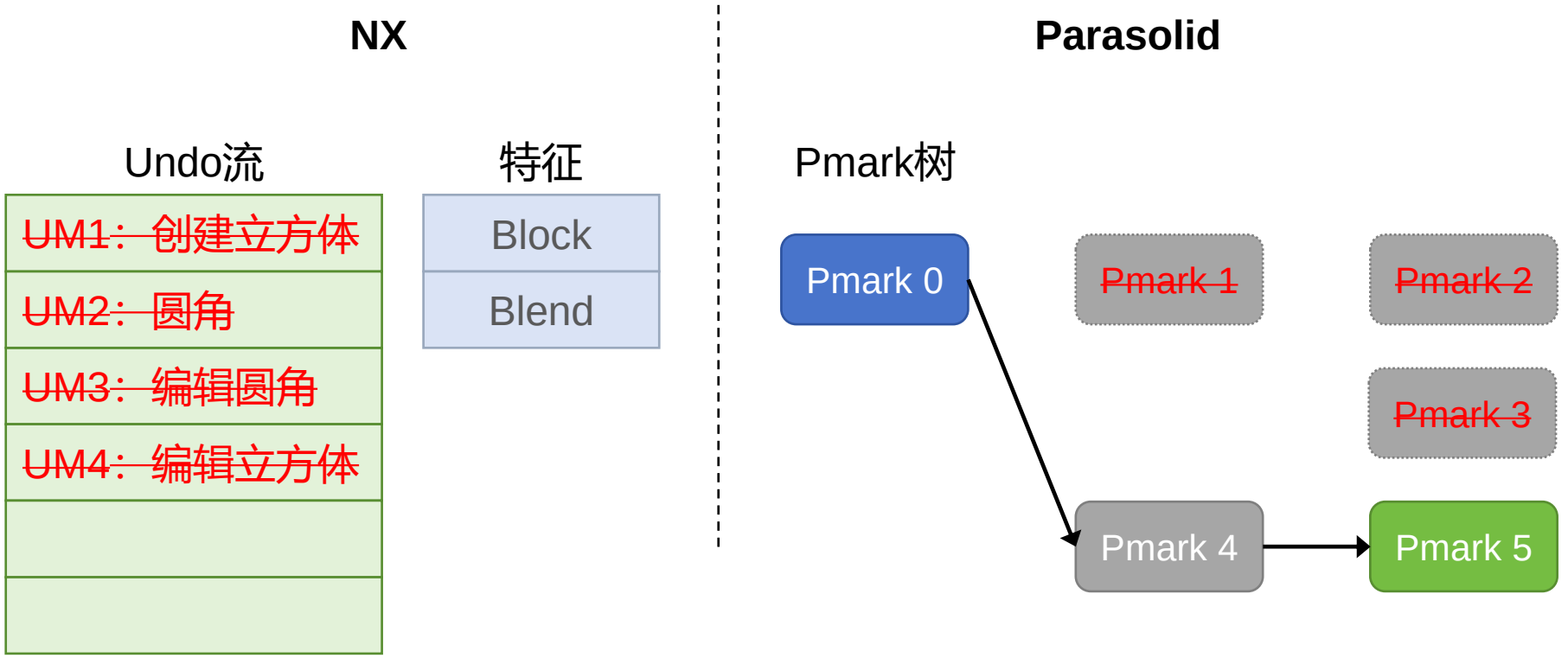
Pmark树





NX

- 初始状态
- 创建立方体
- 做圆角
- 编辑圆角特征
- 编辑立方体特征
- Undo/Redo
- 保存Part文件



NX

- 初始状态
- 创建立方体
- 做圆角
- 编辑圆角特征
- 编辑立方体特征
- Suppress Blend

NX

Undo流

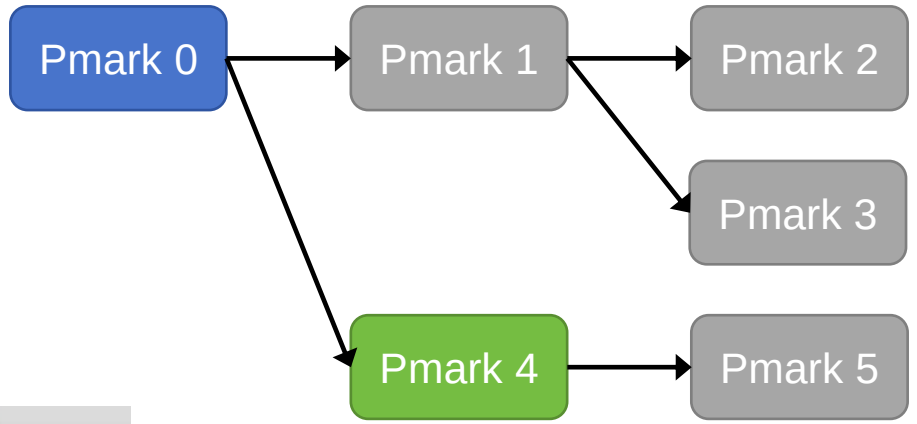
UM1：创建立方体
UM2：圆角
UM3：编辑圆角
UM4：编辑立方体
UM5：Sup. Blend

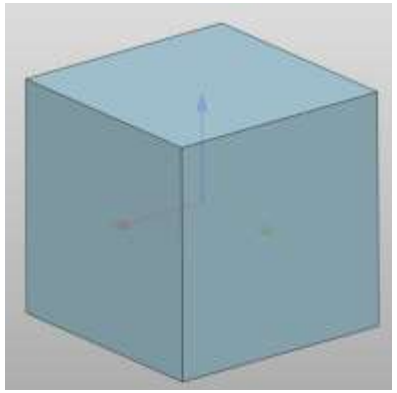
特征

√ Block
Blend

Parasolid

Pmark树



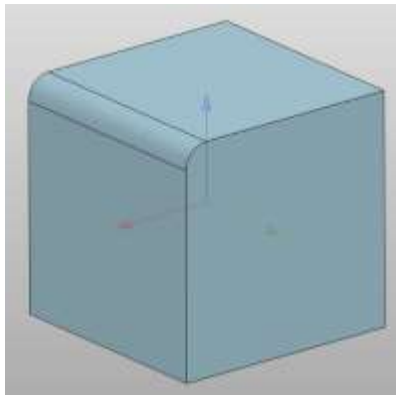


NX

- 初始状态
- 创建立方体
- 做圆角
- 编辑圆角特征
- 编辑立方体特征
- Suppress Blend
- Unsuppress

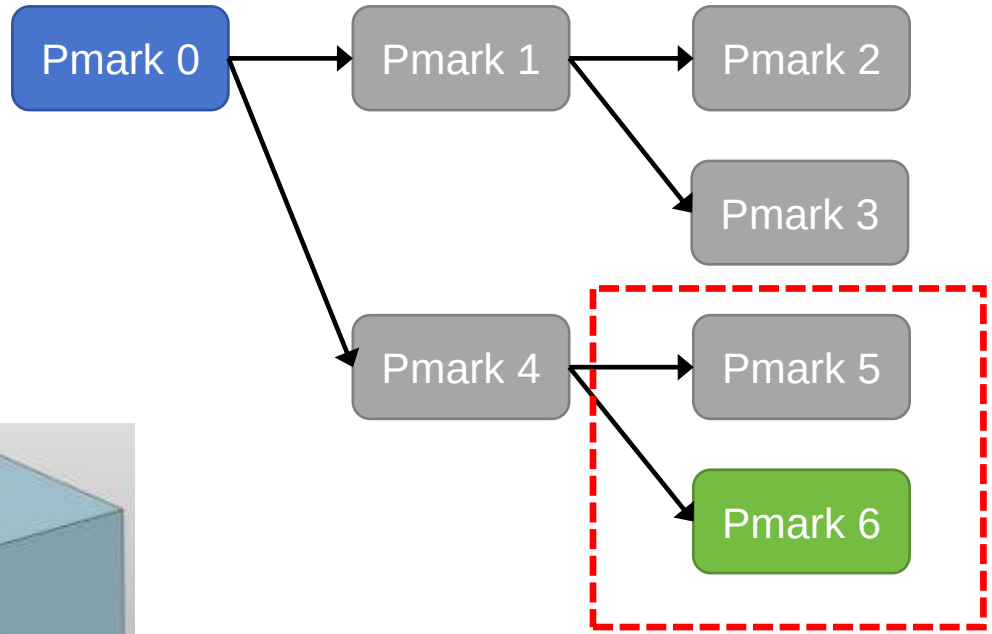
NX

Undo流	特征
UM1：创建立方体	√ Block
UM2：圆角	√ Blend
UM3：编辑圆角	
UM4：编辑立方体	
UM5：Sup. Blend	
UM6：Unsup.	



Parasolid

Pmark树



重做特征而非前滚