VTUFileIO 开发日志（按日期整理，2025.7.11-2025.7.22）

2025.7.22

- *notcharlatan* 完成本文档书写。
- 修复VTK立方体模型导入时的数据错误（如坐标偏移、顶点索引错乱），修正解析逻辑：

```
// 校正立方体顶点顺序，确保拓扑结构正确
void fixCubeVertexOrder(QList<Vertex>& vertices) {
    if (vertices.size() == 8) { // 立方体固定8个顶点
        std::swap(vertices[2], vertices[3]); // 修复特定顶点的位置映射
    }
}
```

  效果：导入VTK立方体模型后，顶点坐标与拓扑关系准确。

- *jyy3371* 合并`main`分支，同步最新代码（含立方体导入修复与文档更新）。
- *SYngFlame* 合并`main`分支，确保功能修复代码与用户手册更新同步。
- *notcharlatan* 两次更新《用户手册.md》，补充VTK立方体导入的操作步骤与注意事项（如"导入前需确认文件顶点数为8"）。
- *notcharlatan* 为用户手册添加功能截图（含"Import VTK Part"按钮位置、导入成功后的模型显示效果）。
- *notcharlatan* 完成《用户手册.md》主体内容，包含导出/导入功能的完整流程、常见错误排查（如"导出失败提示'零件未加载'时，需检查是否调用`odb.ReadParts()`"）。
- *jyy3371* 合并`main`分支，同步文档与代码修复内容。
- *notcharlatan* 修复错误提示无法输出的BUG，通过自定义消息处理器实现弹窗提示：

```
class MessageHandler {
public:

  static void ReportExportErr(int err);
  static void ReportExportInfo(int pointNum, int elemNum);
  static void ReportImportErr(int err);
  static void ReportImportInfo(int pointNum, int elemNum);
};
```

2025.7.21

- *SYngFlame* 提交 VTUFileIO 模块v1.0最终代码，整合所有功能（导出VTK、导入VTK、菜单集成等），优化核心类结构：
- *SYngFlame* 更新开发文档，记录 v1.0 版本功能清单（支持的文件格式：VTK Legacy； 兼容 SAM 版本：v2023+）。
- *jyy3371* 提交 “test doc”，验证用户手册中 Markdown 表格（如“导出参数说明表”）的格式兼容性。
- **notcharlatan* 与 *SYngFlame* 合作完成了GUI按钮移植至File下。

```
    for (QAction* action : menuBar->actions()) {
  if (action->text().contains("File", Qt::CaseInsensitive)) {
    fileMenu = (SAMMenu*)action->menu();
    break;
  }
}
if (fileMenu == nullptr) return;
for (QAction* action : fileMenu->actions()) {
  if (action->text().contains("Import", Qt::CaseInsensitive)) {
    importMenu = (SAMMenu*)action->menu();
    qDebug() << "VTUFileIO: Import Menu detected.";
  }
  if (action->text().contains("Export", Qt::CaseInsensitive)) {
    exportMenu = (SAMMenu*)action->menu();
    qDebug() << "VTUFileIO: Export Menu detected.";
  }
}
```

- *SYngFlame* 合并`main`分支，同步测试文档与代码。
- *jyy3371* 删除冗余的`doc/测试报告.docx`，统一使用Markdown格式管理文档。
- *jyy3371* 上传补充资源（含测试用VTK文件 `cube_test.vtk`），用于功能验证。

```
# vtk DataFile Version 3.0
SAMModel Output
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 8 float
0 0 0 1 0 0 0 1 0
1 1 0 0 0 1 1 0 1
0 1 1 1 1 1
CELLS 1 9
8 0 1 2 3 4 5 6 7

CELL_TYPES 1
11
```

- *SYngFlame* 临时调整装配体导出逻辑，支持导出模型中所有零件（而非仅装配体子零件）：

```
// 临时修改：导出模型中所有零件
QList<Part*> getAllParts(Model* model) {
    return model->parts(); // 不再筛选，直接返回所有零件
}
```

2025.7.20

- *SYngFlame* 完成VTK Legacy文件导入功能的全量代码，支持解析节点、单元数据并生成SAM零件：

```
// 导入VTK文件并生成SAM零件
Part* importVTKLegacy(const QString& filePath) {
    VTKReader reader(filePath);
    auto nodes = reader.readNodes(); // 读取节点坐标
    auto elements = reader.readElements(); // 读取单元拓扑
    return PartCreator::create(nodes, elements); // 生成SAM零件对象
}
```

- *notcharlatan* 读取 step 部分无法正常获取 frame，多天多方尝试后仍然无法解决。**代码搁浅**，并上传至分支 `getStep`

2025.7.19

- *SYngFlame* 搭建VTK文件读取的主框架，定义`VTKReader`类核心接口：

```
  class VTUFormatReader : public FormatReader {
  public:
    VTUFormatReader(const QString& path, VTUDataContainer* VTKData);
    int Read();
  };
```

- *notcharlatan* 完成VTK3.0版本文件读取至内存代码，上传至分支readVTK

```
int VTKLegacyFormatReader::Read30() {
  while (!stream->atEnd()) {
    QString line = stream->readLine().trimmed();

    if (line.startsWith("POINTS")) {
          ...
          }
      ...
      }
  }
```

2025.7.18

- *jyy3371* 微调代码（无功能变更，优化变量命名与注释）。
- *SYngFlame* 移除 `VTUFileToolsetGui` 的单例继承，改用局部对象管理生命周期，避免多线程内存冲突：

```
class VTUFileIOToolsetGui : public SAMToolsetGui
  {
      Q_OBJECT
  public:

      // Constructor and destructor:
      VTUFileIOToolsetGui();
      ~VTUFileIOToolsetGui();
  }
```

- *jyy3371* 修正头文件重复引用（如 `#include <odb.h>` 重复），通过前置声明减少编译时间：

```
// 用前置声明替代重复include
class odb; // 前置声明
// 而非重复#include <odb.h>
```

- *SYngFlame* 完成"单个零件导出为VTK文件"功能，支持节点、单元数据写入：

```
bool exportPartToVTK(Part* part, const QString& path) {
    QFile file(path);
    if (!file.open(QIODevice::WriteOnly)) return false;
    QTextStream out(&file);
    // 写入节点
    out << "POINTS " << part->nodeCount() << " float\n";
    for (auto node : part->nodes()) {
        out << node.x() << " " << node.y() << " " << node.z() << "\n";
    }
    // 写入单元（简化示例）
    out << "CELLS " << part->elementCount() << "...\n";
    return true;
}
```

2025.7.17

- *SYngFlame* 完成VTK文件写入核心代码，规范数据格式（如 ASCII 编码、换行符统一）。

```
class VTUFormatWriter : public FormatWriter {
public:
   VTUFormatWriter(const QString& path, VTUDataContainer* VTKData);
```

```
    int Write();
};

class VTUFormatReader : public FormatReader {
public:
    VTUFormatReader(const QString& path, VTUDataContainer* VTKData);
    int Read();
};

#endif //VTUFormatIO
```

- *notcharlatan* 完善SAM后处理结果数据PART部分读取流程，解决两个关键问题：
    1. **无法通过odb获取part**：调用`odb.Read()`和`odb.ReadParts()`确保零件加载：

```
odb.Read();         // 完整读取odb数据
odb.ReadParts();    // 显式加载零件信息
auto parts = odb.parts(); // 成功获取零件列表
```

    2. **element无法直接获取**：通过odb获取ddb，再提取单元数据：

```
ddbMesh ddbMesh = odb.GetDDBMesh();
auto elemData = ddbMesh.elementData(); // 获取单元数据
```

2025.7.16

- *SYngFlame* 创建 VTK/VTU 格式输出模板，统一文件头与数据块结构：

```
// VTK文件头模板
VTUFormatWriter::VTUFormatWriter(const QString& path, VTUDataContainer*
VTKData)
{
  this->data = VTKData;

  QString targetPath = path;
  if (QFileInfo(path).suffix() != "vtu") targetPath = path + ".vtu";

  QFile* file = new QFile(targetPath);
  if (!file->open(QIODevice::WriteOnly | QIODevice::Truncate |
QIODevice::Text)) {
    delete file;
    file = nullptr;
  }
  else QTextStream* stream = new QTextStream(file);
  stream->setRealNumberPrecision(8);
  stream->setRealNumberNotation(QTextStream::ScientificNotation);
  }
```

```
  int VTUFormatWriter::Write() {
      return 0;
  }

  VTUFormatReader::VTUFormatReader(const QString& path, VTUDataContainer*
VTKData) {

  }
  int VTUFormatReader::Read() {
    return 0;
  }
```

- *SYngFlame* 实现文件选择器与路径校验，确保导出路径合法：

```
QString getVTKSavePath() {
    return QFileDialog::getSaveFileName(
        nullptr, "Save VTK", "", "VTK Files (*.vtk)"
    );
}
```

- *notcharlatan* 代码合并后删除分支 `writeVTK`。

2025.7.15

- *jyy3371* 合并 `main` 分支，同步功能代码与文档框架。
- *notcharlatan* 提交未完成的测试报告，梳理测试用例（如"导出后VTK文件的节点数与SAM零件一致"）。
- *notcharlatan* 删除旧版 `插件测试报告.md`，整理文档结构。
- *notcharlatan* 新增未完成的 MarkDown 文档，计划补充"功能兼容性说明"。
- *jyy3371* 创建项目文档目录，初始化开发规范与接口说明。

2025.7.13

- *notcharlatan* 创建新分支 `writeVTK` 来保存不利用VTK库编写vtk文件部分代码
- *SYngFlame* 在 `printAll()` 方法中添加单元搜索功能，支持打印单元 ID 与类型：

```
            const bmeMesh* objectMesh = flpart-
>ConstGetMesh(bdoDefaultInstId);
        if (!objectMesh->NumNodes()) return 0;
        const bmeNodeData& nodeData = objectMesh->NodeData();
        utiCoordCont3D nodeContainer = nodeData.CoordContainer();
        cowListInt nodeList;
        nodeData.GetUserNodeLabels(nodeList);
        for (int n = 0; n < nodeList.Length(); ++n) {
            int userLabel = nodeList.ConstGet(n);
            float x, y, z;
```

```
            nodeContainer.GetCoord(nodeData.GetMeshNodeIndex(userLabel), x,
y, z);

            QString pointMsg = QString("Node Label %1 (%2 ,%3, %4)").
                    arg(userLabel).
                    arg(x, 8, 'f', 2, ' ').
                    arg(y, 8, 'f', 2, ' ').
                    arg(z, 8, 'f', 2, ' ');
            qDebug(qPrintable(pointMsg));
        }
```

2025.7.12

- *SYngFlame* 实现SAM工作区数据打印功能，验证节点、零件读取是否正常：

```
omuPrimitive* SAMVTUFileIOFragment::printAll(omuArguments& args)
  {
  basBasis* bas = basBasis::Instance();
  basMdb mdb = bas->Fetch();
  basModelMap modelsMap = mdb.GetModels();
  cowListString modelList = modelsMap.Keys();

  for (int i = 0; i < modelList.Length(); ++i) {
  ...
      }
  }
```

2025.7.11

- *SYngFlame* 初始化项目，创建分支并提交基础代码框架。
- *SYngFlame* 上传工作区配置文件，包含SAM接口依赖与编译选项。
- *jyy3371* 更新 `.gitignore`，忽略编译生成的冗余库文件（如 `Debug/Example1.lib`）。
- *notcharlatan* 删除`lib/Debug/`下的冗余库文件（如`SAM.Pre.Example1Toolset.exp`），精简项目。
- *SYngFlame* 合并 `main` 分支，同步初始化代码。

待办事项

1. **未解决：读取 step 部分无法正常获取 frame**
    - 问题：`odbSequenceFrame` 的 `get(int)` 和 `operator[]` 调用触发内存访问冲突（内存异常 `0xC0000005`）；`constGet(int)` 可获取 frame，但 `fieldoutput` 值无法自动刷新，且无手动刷新接口。
    - 计划：搁置留待后续开发。