|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 密级: | 内部 | | 阶段: |  | |
| 故障逻辑推理与定位软件  技术方案 |
| 文件编号：  现行版本：V1000  总 页 数：63页 |
| 北京旋极信息技术股份有限公司 |
|  |

|  |
| --- |
|  |
| 故障逻辑推理与定位软件  技术方案 |
|  |
| |  |  | | --- | --- | |  |  | | 编 制： |  | | 审 核： |  | | 会 签： |  | |  |  | | 标 准 化： |  | | 批 准： |  | |  |  | |

更改历史

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目 | 签名 | 日期 | 产品型号及名称 | （图册编号） | |
| 设计 |  |  |  |  |
| 校对 |  |  |  |  |
| 审核 |  |  | 第张 | 共张 |
| 标准化 |  |  | 空司通信修配厂制 | |
| 批准 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 更改日期 | 更改方法/内容/原因 | 更改人 | 批准 |
| V1000 | 20240104 | 初稿 | 林枫 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1 主题内容与适用范围 1](#_Toc4105)

[2 规范性引用文件 1](#_Toc10713)

[3 术语、定义和符号 1](#_Toc2698)

[4 项目总体描述 1](#_Toc11397)

[5 系统概述 2](#_Toc20253)

[5.1 设计原则 2](#_Toc4014)

[5.1.1 标准化 2](#_Toc20255)

[5.1.2 跨平台 2](#_Toc24267)

[5.1.3 高可配置性 2](#_Toc4475)

[5.2 系统结构 3](#_Toc12611)

[6 测试性建模与分析工具 3](#_Toc21393)

[6.1 模型的建立 3](#_Toc11300)

[6.2 模型编辑功能 4](#_Toc22181)

[6.2.1 系统工作模式和系统配置 4](#_Toc8807)

[6.2.2 编辑视图 4](#_Toc25986)

[6.2.3 导入导出功能 6](#_Toc20600)

[6.2.4 图形信息修改 8](#_Toc15476)

[6.2.5 模型修改权限 10](#_Toc21723)

[6.2.6 图形化配置故障模式 11](#_Toc3522)

[6.3 模型评估与分析计算功能 11](#_Toc22312)

[6.3.1 支持多种测试类型和应用场景的分别分析 12](#_Toc28498)

[6.3.2 诊断分析和优化选项 12](#_Toc6067)

[6.3.3 定性分析 13](#_Toc8737)

[6.3.4 定量分析-自动计算检测率和隔离率 14](#_Toc13719)

[6.3.5 测试性设计报告 14](#_Toc27306)

[6.3.6 多种故障率概率密度分布曲线 16](#_Toc9727)

[6.3.7 诊断策略导出 17](#_Toc10625)

[6.4 与第三方平台集成能力 17](#_Toc28393)

[6.5 总体建模流程 18](#_Toc7581)

[6.5.1 准备阶段 19](#_Toc2219)

[6.5.2 顶层与系统初步建模阶段 19](#_Toc17967)

[6.5.3 分系统初步建模 19](#_Toc9482)

[6.5.4 分系统详细建模 20](#_Toc18423)

[6.5.5 LRU级建模 20](#_Toc20388)

[6.6 建模步骤 21](#_Toc6771)

[6.6.1 收集系统文档 21](#_Toc15090)

[6.6.2 创建系统的结构化模型 21](#_Toc9057)

[6.6.3 添加模块 22](#_Toc26172)

[6.6.4 模块属性定义 23](#_Toc23397)

[6.6.5 决定建模深度 25](#_Toc27688)

[6.6.6 添加故障模式 26](#_Toc11480)

[6.6.7 物理故障与功能故障 26](#_Toc1744)

[6.6.8 参数故障或指标故障 27](#_Toc1539)

[6.6.9 添加连接 27](#_Toc32133)

[6.6.10 添加测试点和测试 28](#_Toc8797)

[6.6.11 添加信号名称 30](#_Toc4520)

[6.6.12 模型分析计算 31](#_Toc27265)

[6.7 分析算法 38](#_Toc27451)

[6.7.1 模型表达 38](#_Toc2286)

[6.7.2 算法求解问题 38](#_Toc19230)

[6.7.3 问题假设 40](#_Toc480)

[6.7.4 数据结构定义 41](#_Toc14358)

[6.7.5 诊断树序列问题定义 42](#_Toc20150)

[6.7.6 最佳诊断序列搜索算法 44](#_Toc17667)

[7 故障逻辑推理与定位工具 48](#_Toc16834)

[7.1 软件需求分析 48](#_Toc16897)

[7.2 接口设计 48](#_Toc23937)

[7.3 软件流程设计 50](#_Toc30526)

[7.4 功能模块设计 52](#_Toc25213)

[7.4.1 通用模块 52](#_Toc16331)

[7.4.2 诊断算法模块 54](#_Toc12658)

[7.4.3 数据存储功能 57](#_Toc19382)

[7.4.4 人机界面功能 61](#_Toc19310)

故障逻辑推理与定位软件技术方案

# 主题内容与适用范围

本文档用于对某系统故障逻辑推理与定位软件的整体设计方案进行阐述，适用于在项目初期规划实施方案，为后续项目开发提供支撑。

本文档主要介绍了某系统故障逻辑推理与定位软件的软件整体设计，包括软件设计原则、整个架构、功能模块及数据库设计等。软件设计人员和开发人员在当前项目实施时需要基于此方案实施。

# 规范性引用文件

OSA-CBM UML Specification 3.3.1Release – June. 29, 2010

CRIS\_V3-2-3\_Documentation.doc

GJB 1391-2006 故障模式、影响及危害性分析指南.pdf

# 术语、定义和符号

PHM：Prognostics and Health Management 故障诊断与健康管理

OSACBM：OpenSystemArchitectureforCondition BasedMaintenance开放式健康管理架构体系

OSA-EAI：企业应用集成的开放系统架构

HA：Health Assessment 健康评估层

BIT：Built-In Test 机内测试

FMECA：failure mode，effects and criticality analysis故障模式、影响和危害性分析

# 项目总体描述

某系统故障逻辑推理与定位软件可为用户提供图形化的建模方法，支持层次化设计方法，支持模型由上向下和由下向上的开发方式构建。允许用户根据系统对象（机械系统、电子系统及液压系统等）的不同建立系统级、子系统级、LRU级的结构功能模型，并根据各单元间的故障或功能依存关系建立它们的连接，从而使模型更接近真实系统的功能组成结构。生成的测试性模型可转换为故障诊断专家知识，为系统健康管理系统所调用。

本项目需要交付测试性建模与分析工具（以下简称“TADS软件”）和故障逻辑推理与定位工具（以下简称“故障诊断软件”）两个软件。项目的主要工作包括通过测试性建模软件建模，并根据模型开展故障诊断软件开发，用于系统健康管理系统调用，同时完成测试性建模的技术支持。

# 系统概述

## 设计原则

对于故障逻辑推理与定位工具，根据项目总体要求和自身的功能需求，软件主要基于以下设计原则设计实现：

* + 1. 标准化

故障诊断软件在数据处理部分的设计主要基于OSA-CBM标准（开放式健康管理架构体系），软件功能模块按照标准中的健康评估层来设计实现，从而实现软件功能模块的标准化，模块接口的标准化。

同时，对于数据的存储，故障诊断软件主要基于OSA-EAI标准（企业应用集成的开放系统架构）定义了故障诊断相关的数据信息。用于存储故障诊断结果，同时支持相关的信息查询功能。

* + 1. 跨平台

故障诊断软件的跨平台设计主要基于QT自身的跨平台特性，可以确保在源代码一致的情况下基于不同的操作系统都能够成功编译生成目标代码；同时，在软件实现过程中，通过获取当前操作系统相关信息，区分处理不同操作系统需要用到的库文件、函数等，来实现在跨平台过程中软件代码的一致性。

故障诊断软件目前设计运行在国产麒麟操作系统，基于跨平台的设计思想可以保证在源代码不变的情况下完成到Windows、Linux等其他操作系统的移植工作。

* + 1. 高可配置性

高可配置性是指可以故障诊断软件可根据不同的需求和环境灵活配置，软件的功能模块加载、模块输入输出信息等都可以在配置文件中进行定义。故障诊断软件在运行初始化阶段加载配置文件，并根据配置文件自动加载功能模块，根据各模块输入输出关系实现数据信息的传递。

基于高可配置性的设计原则，故障诊断软件可以通过系统配置灵活、方便的删减软件需要加载的功能模块，有利于后续软件的扩展和更新。

## 系统结构

本项目需要部署TADS软件和故障诊断软件，其中TADS软件主要用于完成系统测试性模型的构建，得出测试性设计分析结果，输出能够用于故障诊断的模型文件；故障诊断软件主要用于能够导入诊断模型文件，用于系统健康管理系统调用，能够根据输入数据，实现故障诊断与定位的功能，同时支持可视化展示。

软件的整体结构及相互关系如下所示：



1. 软件系统结构图

# 测试性建模与分析工具

## 模型的建立

TADS以系统的结构化信息作为测试性设计、故障诊断分析的建模基础，以对信号的作用及故障效应的传播过程作为分析对象，从而将复杂系统的测试性问题分解成为局部和简单问题的组合，从而加速建模过程。用户以图形化的方式使用单元模块（包括单元模块端口的信息描述）和连接关系的描述来建立系统模型，并支持层次化的模型描述。支持表格编辑视图、图形编辑视图和模型浏览视图等多种编辑和查看界面。

TADS为用户提供了图形化的建模方法，支持层次化设计方法，支持模型由上向下和由下向上的开发方式构建。允许用户根据分析对象（机械系统、电子系统及液压系统等）的不同建立系统级、子系统级、LRU级的结构功能模型，并根据各单元间的故障或功能依存关系建立它们的连接，从而使模型更接近真实系统的功能组成结构。TADS 允许用户对组成系统的各单元进行属性定义，包括可靠度、维修成本、更换成本、任务内工作时间、运行模式、运行状态等。

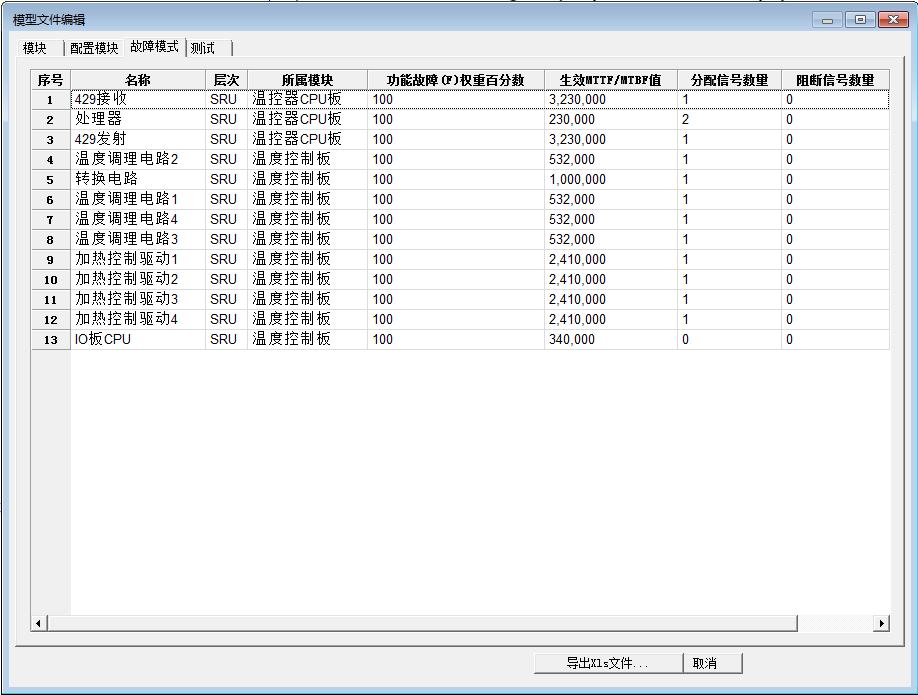
1. 多信号模型：在系统结构和功能分析基础上，以分层有向图表示信号流向和各组成单元（故障或故障模式）的构成及相互连接关系，并通过定义信号及其组成单元、测试与信号之间的关联性，来表征系统组成、功能、故障及测试之间相关性的一种模型表示方式。多信号模型中的信号分为正常和异常两种状态，相应的测试结论为通过和不通过；
2. 多故障模型：能够分析出并定位多个故障的模型；
3. 全局故障：指一旦发生将使系统完全丧失完成功能能力的故障；
4. 功能故障：只影响某些参数或指标的偏移或下降，而不会导致系统整体功能无法完成的故障。例如:功率放大故障、滤波故障等。功能故障导致系统丧失部分功能，系统工作不完全中断。

## 模型编辑功能

* + 1. 系统工作模式和系统配置

TADS支持对系统工作模式和系统配置进行自定义。模型文件可以使用不同的工作模式和系统配置，分析不同工作模式和系统配置情形下的测试性指标。

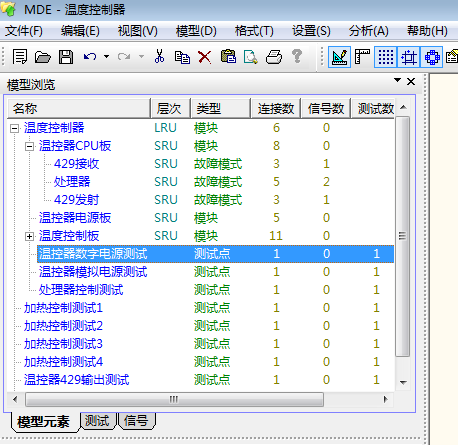
* + 1. 编辑视图



1. TADS表格编辑视图

TADS模型元素的编辑支持多种编辑视图，视图中有模块、配置模块、故障模式和测试等模块元素列表。可以在编辑界面点击模型元素调出属性界面，编辑模型元素属性。

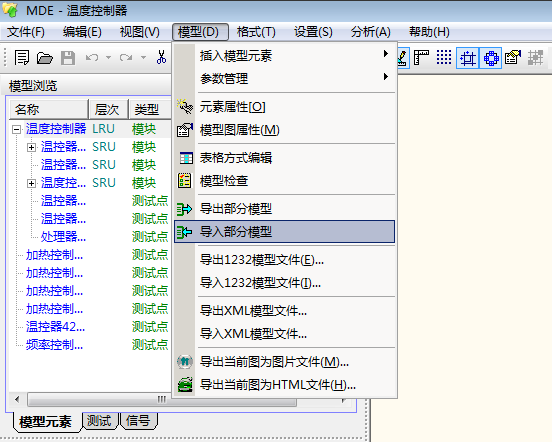
在图形编辑视图，点击图形元素调出属性界面，编辑模型元素属性。



1. TADS模型浏览视图

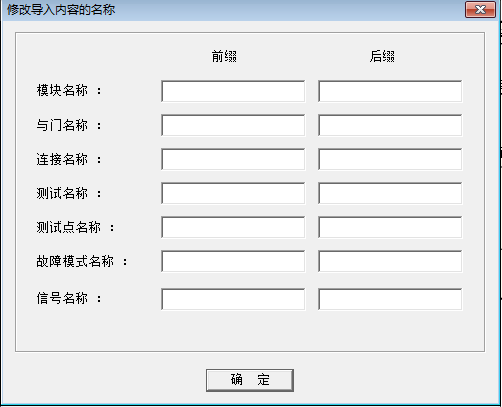
在模型浏览视图，有树形结构的模型元素、测试和信号列表，可以浏览所有的模型元素及其层次、类型、连接和信号数量等信息，并可以点击元素来定位模型元素，进行修改。

* + 1. 导入导出功能



1. 模型导入导出功能视图

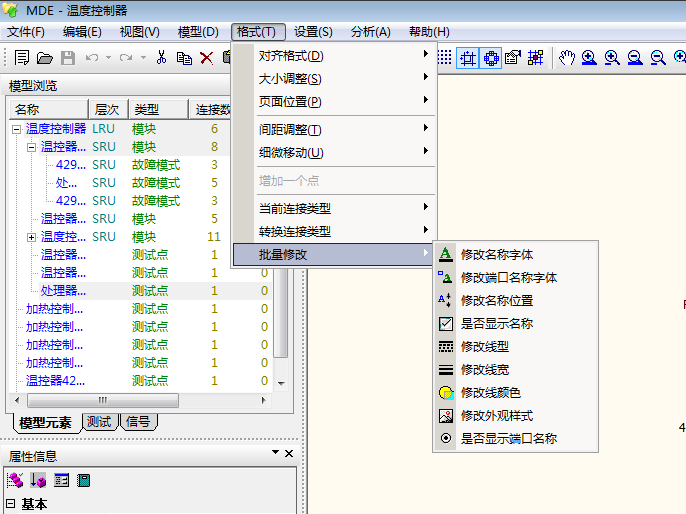
对于已经建立的模块或测试具有重用性。系统支持模型数据、测试性建模分析数据的导入导出功能，可以导出模块，以便以后复用此模块，导入模块，复用以前建立的模块。功能支持模型数据以通用表格等形式导出，模型导入导出格式有IEEE std 1232标准、XML文件标准支持.mif文件格式。



1. 模型导入内容编辑界面

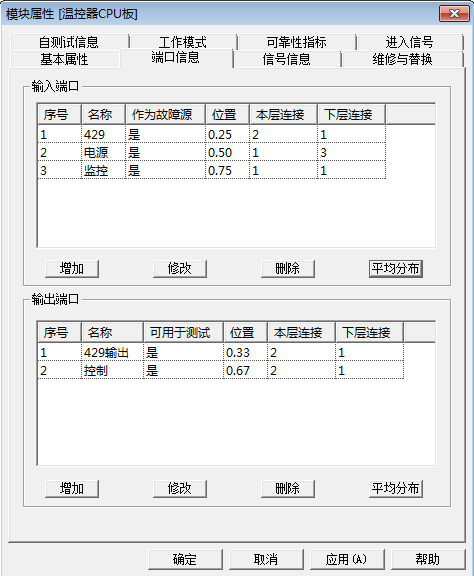
如上图所示，导入模型时，可以增加模型各元素名称的前缀和后缀，批量修改导入模型中所有元素的名称。

* + 1. 图形信息修改



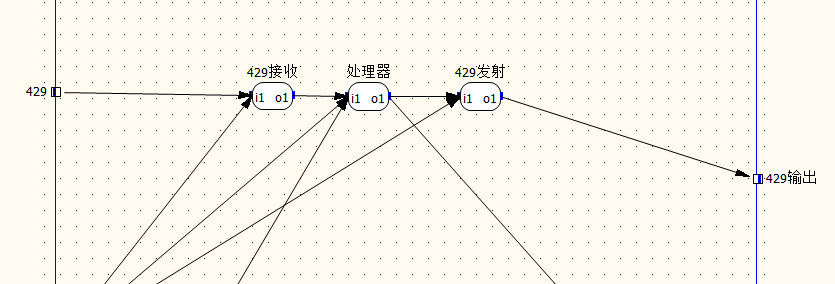
1. 图形信息修改界面

TADS支持对模型元素的批量选择和批量修改功能，可更改字体、端口名称线型、线宽、颜色、样式等。



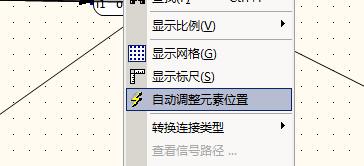
1. 模块属性

在端口信息属性界面，点击平均分布可以让模块端口进行自动排序和排列，方便用户对各模型进行连接。



1. 模型连接编辑界面

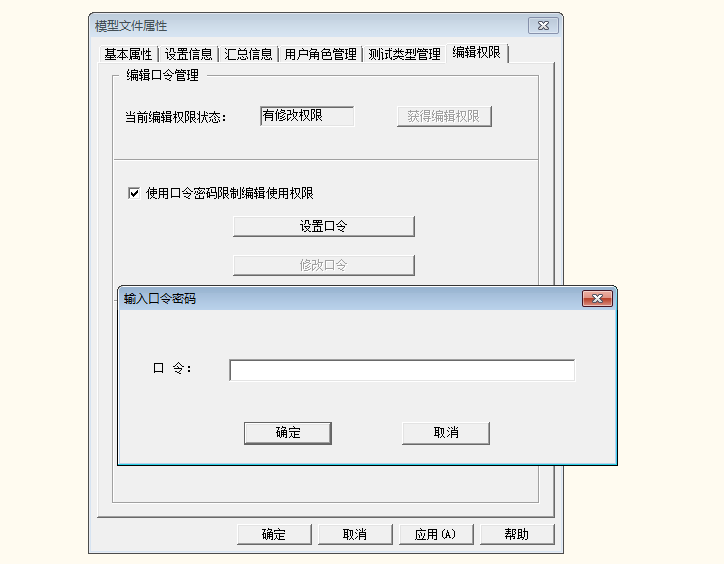
可以支持对图形元素的网格定位、可以使用键盘上的CRTL键加上、下、左、右等键，按元素的不同方向相互对齐图形元素。



1. 自动调整元素位置

在模型图上，点击右键，再点击自动调整元素位置，软件会把元素位置自动调整使之适应模型图大小。

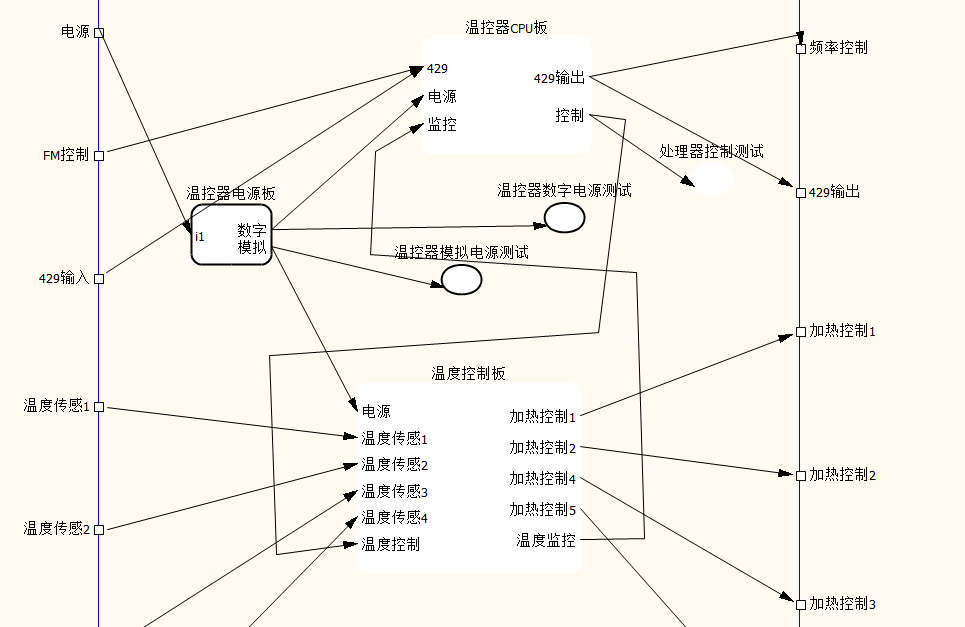
* + 1. 模型修改权限



1. 模型修改权限

模型文件支持编辑权限设置功能，当用户选中限制编辑选项时，设置密码，模型文件可以被密码保护。需要输入对应的密码才能进行编辑，否则只能进行浏览。

* + 1. 图形化配置故障模式

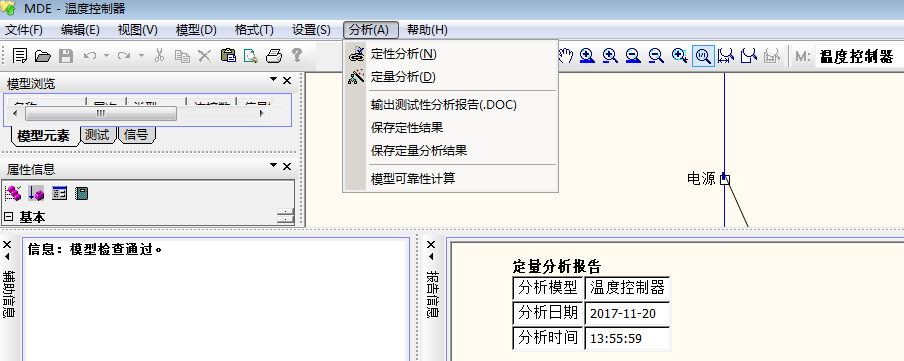


1. 拖拽图形化配置

TADS支持在模型上构建通过拖拽配置的方式构建故障模式，支持模型仿真故障传递功能，在界面上选取任何一个故障模式或模块，可高亮显示所有可检测该故障模式或模块的测试点，在界面上选取任何一个测试点，可高亮显示其覆盖的所有故障。这样可以直观的看到测试和模块的关系，大致判断自己所建立模型的正确与否。

## 模型评估与分析计算功能

TADS能够针对测试性模型进行定性分析和指标定量分析，能进行系统及设备的故障检测率、隔离率的预计和BIT预计等。包括进行冗余和模糊度分析帮助客户了解测试性指标。



1. 评估与分析功能界面
   * 1. 支持多种测试类型和应用场景的分别分析



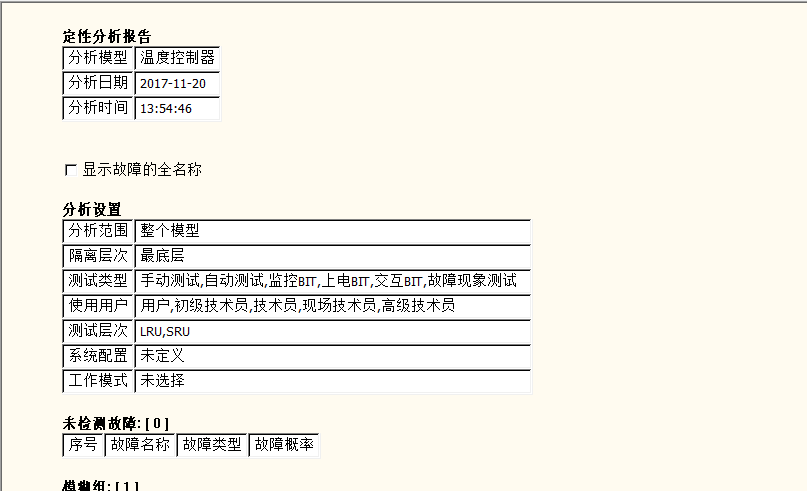
1. 定量分析设置界面

软件支持对测试类型进行自定义，测试类型数量不受限制。在分析设置界面选择不同的测试类型、测试层次和用户角色等选择需要的测试，在工作模式和系统配置界面选择不同的工作模式和系统配置，对不同的使用情况下的可测试性指标进行预计。可以准确的反映真实系统的测试构成情况和应用场景，同时不同测试方法的组合情况也为集成诊断分析的各种场景提供了依据。

* + 1. 诊断分析和优化选项

TADS允许用户设置测试费用以及花费的时间等属性，以进行故障诊断分析和诊断策略的优化，例如按照测试费用最少的优化模式、测试时间最少的优化模式、或者加权成本最少的优化模式。算法上采用与或树搜索配合估值函数的方式保证得到优化的诊断序列。

* + 1. 定性分析



1. 定性分析报告界面

测试性定性分析内容包括：未测试故障、模糊组信息、冗余测试、隐藏故障、冒充故障、反馈回路等信息。

1. 未检测故障：表示故障存在，但是系统不能检测到；
2. 模糊组：两个（两个以上）的故障，系统无法区分出来。就将这些故障视为模糊组；
3. 冗余测试：多组测试能检测的故障模式相同，该组测试视为冗余测试。某一个测试未定义任何测试内容也被认为是冗余测试；
4. 冒充故障：多故障模式下，多个故障的表现形式和多故障之外的某个单故障的表现形式相同。则这几个多故障被认为冒充故障；
5. 隐藏故障：多故障模式下，多个故障的表现形式和多故障之内的某个单故障的表现特征相同，则这几个多故障被认为隐藏故障。
   * 1. 定量分析-自动计算检测率和隔离率



1. 定量分析报告界面

测试性分析定量指标包括：故障检测率、故障隔离率（不同模糊度下）、平均模糊组大小、平均隔离步骤、平均故障检测时间、平均故障检测费用、平均故障隔离时间、平均故障隔离费用等指标。

* + 1. 测试性设计报告

TADS提供了以下分析报告：

1. 未检测故障报告；
2. 冗余重复测试报告；
3. 故障模糊组报告；
4. 故障反馈回路报告；
5. 隐藏故障报告。

为评估被分析系统的测试性设计提供多种定性或定量的描述和设计指导，便于进行设计改进。可自动生成word文档格式的测试性分析报告，报告中应包含报告标题，分析结果和数据统计表格，未检测故障、测试方案和模糊组汇总、诊断流程方案等内容。

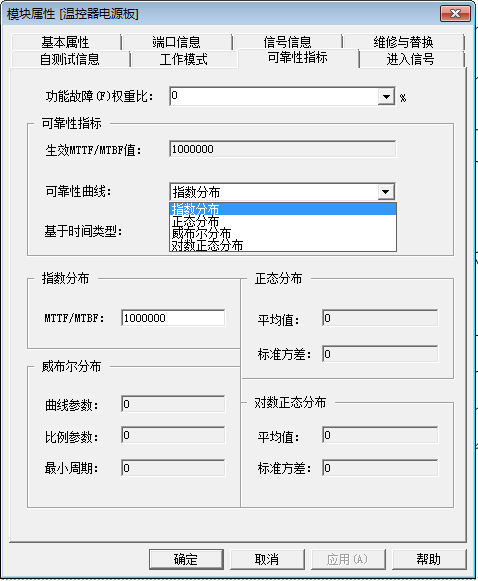


1. 成品测试性分析报告示例

定性分析和定量分析都可导出模型对应的依存矩阵，依存矩阵以文本或表格形式存储，可供给第三方软件使用。



1. 导出的依存矩阵
   * 1. 多种故障率概率密度分布曲线



1. 导出故障概率密度分布编辑图

TADS中可以为不同种类产品的各个故障模式指定不同的故障概率密度分布，使得测试性指标的分析可以适应更广范围的产品和系统，也与可靠性预计工作保持兼容，并且可根据可靠性的结果优化对应的诊断方案。TADS中支持的故障概率密度分布曲线有：

1. 指数分布；
2. 正态分布；
3. 对数正态分布；
4. 威布尔分布。
   * 1. 诊断策略导出



1. 诊断树浏览

可自动生成诊断树，且能够针对不同的测试条件和设置生成不同的诊断树，故障诊断树能以DiagML格式的XML文件进行导出，供给第三方软件使用，并可以BMP格式的图形文件保存。



1. 综合诊断

可支持综合诊断，能支持ATE环境下调用模型进行自动故障推理。在自动测试软件中调用测试性模型作为数据库来判断故障。

## 与第三方平台集成能力

具备与第三方平台集成功能的能力。

测试性建模软件可导入六性一体化平台建立的设计数据，避免手工方式重复定义系统组成和架构信息并避免人工建模误差，并在此基础上建立测试性模型，并进行测试性指标预计和分析，并将分析结果返回到六性一体化平台进行展示。

导入导出文件可以使用IEEE 1232标准交换文件或XML格式文件进行。

IEEE 1232是关于在测试诊断系统之间应用人工智能的数据交换和服务的规范。主要目的是在系统测试和诊断领域内为推理系统提供一套数据交换和软件服务的标准。该交换和服务是建立在测试环境的信息模型上，可用提供的标准框架来识别诊断所必需的信息，并且以机器可处理的定义诊断信息，同时，因为它规范了软件接口，就保证了诊断工具的兼容性和可靠性。它描述的是在测试系统中的数据和知识表示的规范集，是一种中立的（数据和知识）交换格式，同时提供诊断信息的形式化模型，所以诊断推理机之间模型信息互换成为可能。

交换即指数据（或知识）的互换，之所以规范数据互换格式，是为了在不使用信息管理系统的前提下，提供在相容系统间交换知识库的方法。

IEEE 1232是以知识处理为基础的人工智能技术应用到产品的测试诊断，使诊断系统能相互兼容和独立于诊断过程，达到测试诊断知识可移植、重用、共享，构建综合诊断的开放式体系结构。

甲方后续如需平台建设，可经双方商榷商务条款后实施。

## 总体建模流程

测试性建模的步骤可以大致分为以下阶段：



1. 建模流程
   * 1. 准备阶段

建模过程首先从建模准备开始。一个具体的项目中应统一定义建模的层次级别，这样在系统建模时才能够将众多由不同人员在不同设备级别上建立的模型统一和综合起来。

根据系统的功能划分、构成层次以及维修方案的要求，确定系统功能层次，例如：顶层是系统或分系统，第二级是现场可更换单元LRU，第三级是维修车间可更换单元SRU（可以是部件、可供货的电路插件板等）。而且需要把这些层次之间的关系使用建模工具建立起来。

* + 1. 顶层与系统初步建模阶段

接下来是顶层与系统初步建模阶段，顶层与系统级初步建模的主要任务是将顶层和各系统的组成进行初步定义，并且将顶层功能逐级分配到各个系统和分系统上，明确每个系统的输入和输出功能，确定各个分系统之间的接口约束与交联（这个时刻输入的分系统之间的互联不是真实的交联关系，更多是一个示意表达）。

这个阶段上设计的主要目的是，制定接口文件和设计约束和其它设计要求（可靠性分配，测试性分配等）。

在这个阶段上，由于实际维护活动并不需要过多关注系统级（故障隔离到本级是不够的），另外测试性指标（故障检测率和故障隔离率） 通常不在这一级进行计算，所以实际的测试实现只在系统或分系统的边界（输入输出信号或功能）上定义是不足够的。所以在这一级通常不进行完整的模型构造，而主要进行系统级功能结构模型的输入，具体体现在功能分配，分系统之间信号交联定义、分系统功能要求，指标要求等。

在这个阶段如果上层已经确切知道系统组成情况可以按照系统组成情况进行建模，如果还没有系统组成的知识，可以建立全机功能性模型，如果已经具有系统组成情况，可以在初步模型中输入各个分系统，以及相互之间的依存关系。

建立起所有系统的上层层次结构和系统级功能描述信息后下一步可以根据设计的流程与进行情况分别进入“分系统初步建模”环节或者“顶层与系统详细建模环节”。

* + 1. 分系统初步建模

分系统初步建模对应分系统设计流程和阶段中的初步设计阶段，主要描述本分系统中和其它系统的交联情况和功能初步设计。具体本步骤的内容应按照实际设计流程对初步建模与详细建模阶段之间的任务进行合理的分配和协调。

分系统级定义的故障通常是对分系统的故障和状态的一种描述，在初步设计阶段通常根据分系统级的功能来枚举分系统级故障定义，即按照对应系统的功能FMEA分析结果进行定义。

分系统级定义的故障一般与功能或故障现象相对应，而不与单个具体的LRU故障或故障原因直接对应。但对于比较简单的系统，也可以通过描述部分主要LRU来表达分系统的故障状态。在下一步LRU初步建模过程中就可以通过对各个LRU的功能详细描述来进一步确定分系统级的故障集合。

定义分系统故障时不一定要把每个分系统功能失效都列举出来，原因是很多功能是互相相关和互相影响的（由于这些功能共同由相同的软硬件模块实现），可能在特定故障模式下这些相关的功能会同时失效或同时工作，建模人员需要分辨出可能出现的故障模式，将其列出。

分系统级故障列表只是一个示意性列表，通常并不参与维护活动，也不使用其进行测试性分析，因此不一定需要写入模型文件中，但是它是初步设计过程中系统级设计的一个阶段，可以用来帮助指导后续设计，并且比较重要的分系统中的某些状态和故障还会被选择定义为FDE或EICAS信息（分为警告、注意、提示），用来通知飞行员各重要分系统的状态并采取相应的措施。

分系统初步设计主要粗略设计每个分系统和别的分系统的信号交联关系，通过信号交联关系定义本分系统所需要具有的功能。

完成此步骤以后，下一步可以根据设计的流程与进行情况分别进入“分系统详细建模”环节或者回到“顶层与系统详细建模环节”。

* + 1. 分系统详细建模

分系统详细建模对应分系统设计流程和阶段中的详细设计阶段，主要描述本分系统中LRU的划分与功能分配，分系统级故障定义等。具体本步骤的内容应按照设计所的实际设计流程对初步建模与详细建模阶段以及LRU初步设计之间的任务进行合理的分配和协调。本步骤应定义出各个系统的输入输出和接口关系以约束和规范LRU级建模，并且为LRU的测试项目设计提出系统对应的要求。

* + 1. LRU级建模

LRU级建模主要描述LRU内部的结构组成信息、故障依存信息、测试覆盖信息，这些信息可以分为两部分，一部分是为了LRU本级使用的故障信息模型元素，另外一部分是可以向上传递到系统的故障信息模型元素。例如LRU内部的部件组成信息只供本层分析使用，设计的测试方案中一般只有BIT测试可以供系统（外场）使用，各个SRU的故障模式信息会进行综合以后更新原有的分系统模型中各个LRU的故障模型中的故障率信息和故障依存信息。

所以LRU级建模一般使用单独的模型文件进行描述，分系统级综合时并不需要直接合并LRU级模型文件，而是从中选择需要的信息对原有的分系统级模型文件进行更新即可。

LRU初步设计中的功能设计主要是将分系统级设计中形成的各个LRU的设计需求和功能需求加以细化和分解，形成更进一步的功能描述。

推理关系中主要描述某一项测试通过的推论（或前提），这些推论表达的是若某一项测试通过（或若要想让某项测试通过），那么能够确定哪些诊断结论和哪些测试结论。若某一项测试失败，那么能够确定哪些故障可能存在以及哪些测试可能失败（需要进一步执行）。

## 建模步骤

建模的步骤可以大致分为以下阶段。

1) 收集所有的系统文档

2) 创建系统的结构化模型

3) 添加故障模式

4) 在模块之间添加连线表示依存关系或者信号信息流（电、机械、液压）

5) 添加测试点和测试

6) 向故障源和测试添加功能名称

7) 执行各种测试性分析

8) 评估产生的测试策略和测试报告，考察是否需要进行设计修改

9) 根据需要更新模型重新分析

* + 1. 收集系统文档

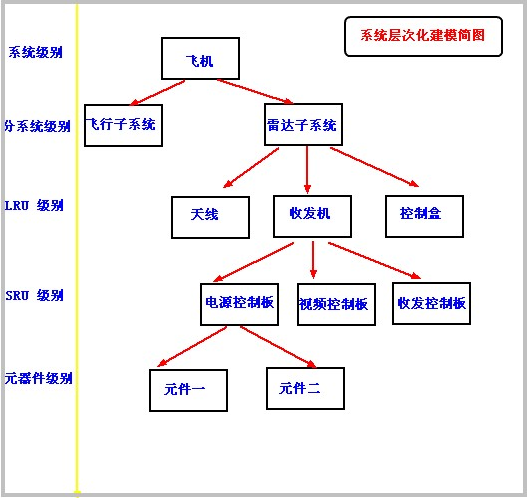
建模之前建模人员首先应尽量收集并仔细阅读待建模系统和子系统相关的以下资料：

* 电路图
* 工作原理
* 维护手册
* 可靠性和FMECA数据

建模人员通过阅读所收集的资料和文档应能够熟悉并熟练掌握待建模系统以及其支持和接口的其它系统的以下知识：

* 系统或设备是如何工作的？
* 系统或设备会以什么方式损坏或失效
* 可以执行哪些测试以检测故障
  + 1. 创建系统的结构化模型

本步骤中按照系统既有的层次和结构关系建立起结构化模型，例如一个系统由LRU1和LRU2组成，其中LRU1又由3个SRU组成，LRU2由2个SRU组成，那么需要使用模块嵌套的方式表示出这种组成关系。下图时是将这种层次关系表达在同一张图上的示意图，不是实际的模型图，模型图是将这种层次关系分成多个图形进行表达的，层次越深，图形越多，但每个图形上都只包含同一结构内相互关联的同一层次模块的关系。



1. 模型层次示例
   * 1. 添加模块

添加模块在不同的情况可以代表不同的内容和实质，如果在较高层次上建模，一开始添加的一个模块可能代表由许多复杂设备共同组成的一个庞大系统，在某个模块内部添加的子模块可能也是一个由多个设备组成的分系统，或者是一个单独的电子设备或机械设备，或者是一块电路板或者一个机械组件，甚至一个电子元件和一个机械零件和标准件。

添加模块的步骤如下：

1.点击“添加模块”按钮

2.移动鼠标到画布区的期望位置。

3.点击鼠标左键，弹出对话框提示输入模块名称：



1. 添加模块

输入模块名称以后，系统显示模块属性面板（见下图），建模人员此时应根据事先收集的资料（可靠性信息，工作原理信息等）将模块对应的设备或者部件的信息输入到属性面板中。

输入必要的模块信息后，可以编辑模块的输入和输出端口，需要注意的是，在这里建模人员不要按照实际电路或者机械连接来建立端口，需要牢记在本模型中流动的是故障，因此模块的端口与实际的连接插头的插钉不是一致的。要根据故障对下游模块的影响情况和被上游模块的故障影响的情况安排端口。

每个输入和输出端口可以指定不同的名称，编辑修改和输入名称的步骤如下：

新建端口

点击“模型”－>“插入模型元素”－>“端口”子菜单。或者单击“模型元素”工具栏的“增加端口”按钮。

移动端口

鼠标移动至端口处,此时鼠标变为十字,按下鼠标左键,并拖动端口即可,在理想位置松开左键.完成移动动作.

删除端口

鼠标移动至端口处,此时鼠标变为十字,按下“Delete”键完成删除。

模块创建以后，可以使用鼠标将光标移动到指定模块之上，点击鼠标左键并保持，就可以拖放移动到画布区域的其它任意位置。一般来说，应当按照模块的逻辑功能流向在画布区域安排模块的位置，输入端口在模块的左侧，输出端口在模块的右侧，所以故障在模型中也是从左流向右侧，从输入端口流入一个模块，再从模块的输出端口流出。

* + 1. 模块属性定义

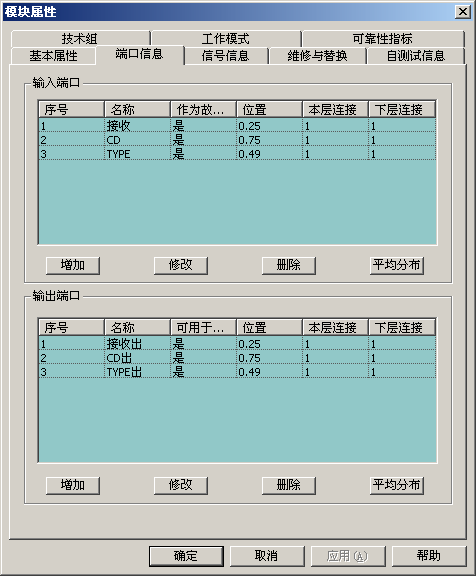
模块属性页中分为以下几个页面:

基本属性页：

****

1. 模块基本属性

输入输出端口属性页



1. 模块输入输出端口属性
   * 1. 决定建模深度

从前面可以看出，在模型中添加一个模块以后，选择该模块后双击即可进入该模块内部添加子模块进行详细的建模，在下一层再重复此过程即可进行更为详细的建模。在较低级别上进行详细建模相当于深入设备的内部观察和描述故障的传递过程，能够得到更为符合实际的故障模型。然而，在过低的层次下建模可能导致建模工作量成指数级别的增长，在复杂和大规模系统中只更深入一层就会使建模工作量以几个数量级进行增长。因此，建模人员需要在建模过程中尽早确定最终的建模深度。

一般来说，建模深度取决于维护场景和维护人员工作内容，如果模型的使用者--该设备/系统的维护人员是更换元件或者机械零件进行修理，那么应该将模型的底层模块代表各个元件，如果维护人员是更换设备中的电路板或者机械组件进行修理，那么应该将模型的底层模块表示各个电路板或机械组件。如果是直接更换设备维护系统工作的正常，那么最详细的模块应该建立在设备级别上。即建模应以维护中的可更换件为底层模块。

如果一个系统有多个维修层次和流程在同一组织和机构中同时存在，那么应该以较为深层次的维护活动为准进行建模，因为建模时进行较为深入的建模，分析诊断时仍然可以在较为高层次进行使用。

一旦选定了最底层的模块，即开始添加故障模式：

如果需要更换元器件进行修理，那么应该在元器件上添加元件的故障模式：

* + 1. 添加故障模式

添加故障是测试性建模的核心和关键，设备部件之间的结构存在着确定性的关系，可以按照空间拓扑准确的定义，然而故障的定义需要建模人员具有相当的维护经验和工作原理知识才能做到准确建模，特别是复杂模块的故障定义。这是由于由简单模块的简单故障模式复合而成的组合具有多样性的表现。

* + 1. 物理故障与功能故障

一个模块即可以描述物理性损坏，也可以描述功能性失效，物理性损坏一般描述一个组件的损坏情况，例如O形圈破裂，管路堵塞，电线断开；而功能性故障一般从该组件应该完成的功能上去进行描述，这个功能性故障是由物理性损坏所引起的后果，例如某条电线的断裂导致某个电信号接收不到。

一般情况下，建模人员在添加故障时，要么同时添加物理性损坏，要么同时添加功能性失效，因为将两类故障同时进行添加很容易造成重复添加，为以后的分析带来困难。即使是添加同一类故障也需要注意不要使这些故障有相互重叠和包含的关系。

例如简单的导线进行故障建模时可能的物理性故障有：断路故障，与其它导线短路的故障。与其它导线短路的故障可能包括与地短路以及和相邻的每一条信号线短路。

* 如果是屏蔽电缆，除了芯线故障以外还有额外的屏蔽层断路故障等。
* 铅蓄电池的物理故障：电池溶液不足，电池极板间短路，电极腐蚀等。
* 铅蓄电池的功能故障：无输出，输出电压低，输出电流小等。
* 稳压电源的物理故障：滤波电容短路，滤波电容开路，输出短路，变压器开路等。
* 稳压电源的功能故障：无输出，输出电压低，输出电流小，输出纹波大等
* 白炽灯的物理故障：灯丝断，灯泡漏气等。
* 白炽灯的功能故障：无输出（不亮）

如果一个部件的某个故障模式并不会导致系统发生可察觉的故障和功能缺失，也就是说该故障在外部和系统级别不可检测，那么这个故障不需要被建模。

有的部件的几个不同的故障模式，在外部和系统级别在目前情况下无论使用任何方法都不可能能够区分开来，那么可能需要将这个部件的这几个故障模式进行合并。

如果一个部件只有一种损坏的情况和模式，那么在该模块上不需要添加故障模式。

* + 1. 参数故障或指标故障

不同级别的模块具有不同的属性和参数，除了那些明确的非此即彼的离散性质的特性以外，还存在一些由数值定义的模拟参数和指标。那么这些指标和参数的偏离是否需要定义为故障呢？这个问题没有明确的答案，需要根据具体的情况和场合进行分析，对于此类建模问题首先要问此模块上的指标超出范围的故障能否被感知或被检测？例如对于照明设备在出厂时有很多的指标进行规范，例如光通量，显色指数，功率因素，波峰系数等，但是在维护阶段基本不会对这类问题进行检测和维护。例如在某些电路上某个电阻或电容的阻值和电容值超差10%甚至以上也不会引起整个电路有任何可感觉的异常。

将具有相同测试签名的故障模式进行合并：例如如果“无输出”和“输出低”导致相同的故障现象和排故行为，那么将这两种故障模式合并为“输出故障”。

举例：

跳开关的功能故障：

* 额定为20安培的跳开关在15安培的时候断开。
* 无负载的时候断开。
* 短路时不断开。

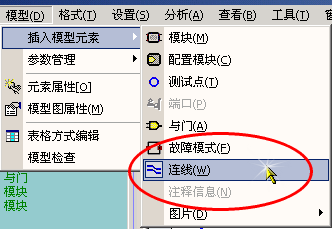
这里面的问题就是在第一种情况下是否添加为故障模式，这个问题的答案取决于如果这种情况出现是否会引起某种可以观察到的故障现象或者后果，或者有没有维护阶段可行的操作检查来检测出这种情况的发生。

另外一个可能影响这个问题答案的就是应用场景，如果这个跳开关的负载是个恒定负载，不会发生变化，那么就可以将第一个故障和第二个故障进行合并。

* + 1. 添加连接

在模块之间添加连线的主要作用是用来表示依存关系流（电子，机械，液压，气压等），即当模块发生故障时会由哪个模块影响另外的哪些模块的正常工作，哪些模块的故障互相没有关联。另外连线具有以下作用和特性:

* 模块上定义的功能会沿着连线进行传播。
* 一条连接连线上可以同时传播多个功能。
* 连线是有方向性的，总是从输出端口向输入端口方向进行连接。
* 连线可以使用直线或者折线方式进行连接。
* 可以通过设置颜色和宽度来改变连线的外观，这样可以区分不同的功能流。



1. 添加连线
   * 1. 添加测试点和测试

测试点：

测试点表示系统中实际进行发生测试和测量的物理位置。

在同一个测试点上可以执行多种测试，那么在模型中一个测试点上也可以定义多个测试。

测试类型：

故障现象类测试：

故障现象类测试通常是系统使用者能够观察和感觉到的测试输出或系统行为。一般都在系统的输出端进行，而不应该是侵入式测试，测试描述用以表示系统是否能够正常进行指定的系统功能，例如：

汽车启动系统：测试描述：当点火钥匙插入并进行点火启动时发动机无反应。

家庭音响系统：选择FM调谐器时无声音。

飞机： 驾驶舱显示“油压低”

操作检查类测试：

操作检查类测试一般都在系统的输出端进行，而不应该是侵入式测试，通常用以检查系统的整体性能和功能。一般在系统的操作手册中进行说明。

机上测试类：

机上测试用以在系统运行时刻监控关键的系统参数，主要供给TADS-RT来实时访问系统的健康状态。

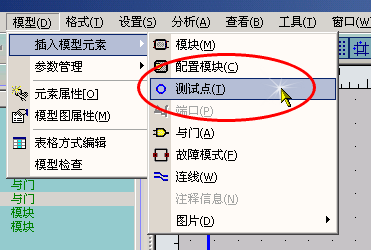
故障隔离类测试：

故障隔离类测试通常需要在系统或设备的内部（系统内部就是设备与设备之间）进行测量和施加激励信号，用来进行故障隔离，一般在系统的排故和维护手册中进行说明。

完成前述测试性要素编辑工作后,需要增加测试点,并编辑测试点对应的测试信息.

新建测试点

arrow选择“模型”－>“插入模型元素”－>“测试点”子菜单。或者单击“模型元素”工具栏的“增加测试点”按钮。



1. 插入测试点

arrow鼠标移至编辑区，点击鼠标左键，放置测试点。

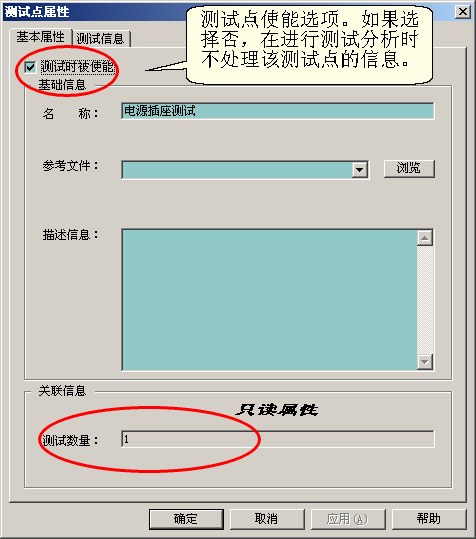
arrow定义测试点与被测模块之间的连接.

修改测试点属性

arrow鼠标左键选中需要修改的测试点，点击右键菜单“元素属性”，在弹出的对话框中修改属性信息。

arrow鼠标左键选中需要修改的测试点，选择“模型”菜单－>“元素属性”菜单，在弹出的对话框中修改属性信息

1、基本属性页：



1. 测试点属性

说明：

测试点使能选择框：

选择使能，测试分析时会处理这个测试点的信息。

选择否，测试分析时不处理这个测试点的信息，这个测试点当不存在。

基础信息：

名称： 测试点的名称。（必选项）

参考文件：点击“浏览”按钮，指定参考文件的路径与名称。（可选项）

描述信息：对此测试点的详细描述。（可选项）

关联信息： 此组的信息是只读信息，表示该测试点中包含的测试个数。

* + 1. 添加信号名称

添加信号到故障源和测试：

-在结构化模型上直接添加功能(故障)信息。

-结构化模型只表示单一的一组依存关系，通过在结构化模型上添加一组功能实际形成了多个不同的依存关系，这在传统结构化模型上需要使用多个结构化模型才能表达出来，现在只需要一个结构化模型加上功能属性。

-功能对应系统的一个独立变量或者独立的功能参数或属性。

例如：一个放大器的功能是提供“增益”功能，如果放大器失效，那么就会影响“增益”功能。

滤波器的功能是调节“频谱”，如果一个滤波器失效，那么我们说会影响到“频谱”功能。

-在模块上定义功能

在模型层次中最低层的模块上分配功能。

功能既可以在分配时新加也可以从原来定义的列表中选择。

一个模块上可以定义一个以上的功能。

-在测试上定义功能

测试上定义的功能是从预先定义的功能列表中进行选择

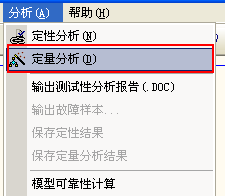
测试上也可以定义一个以上的功能。

-功能加上连线用以将模块（故障模式）和测试相关联起来，以产生依存矩阵。传统的结构模型只使用连线表达模块和测试之间的关系。

一般来说，如果一个故障源和一个测试都分配了相同的功能，并且测试位于该故障源连线关系的下游，那么这个测试能够检测到该故障。

* + 1. 模型分析计算

定量分析时点击“分析”菜单－>“定量分析”菜单。



1. 模型分析

定量分析设置共有六个属性页.分别是:

 基本设置

 输入输出端口设置

 优化

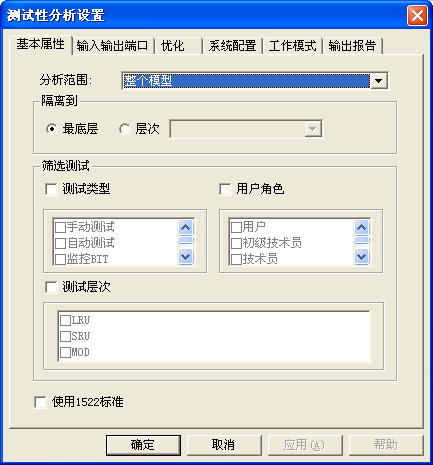
 系统配置

 工作模式

 输出报告

下面将各个属性页内容逐一介绍。

基本属性页



1. 分析设置

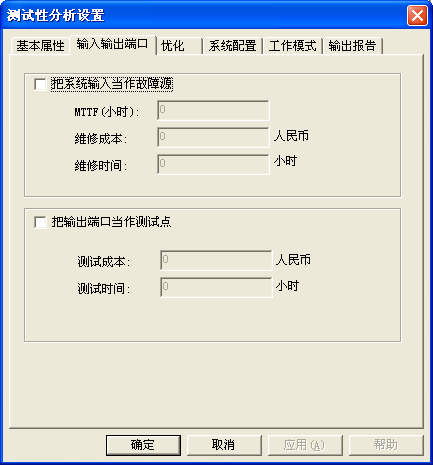
分析范围:表示整个静态分析的分析范围，如果用户选择“整个模型”，系统进行分析时,会分析整个模型文件中所有的模型元素.如果用户选择分析指定的模块,则只分析该模块包含的元素与子模型图的元素。

隔离选项：选择静态分析时，诊断的故障隔离哪级层次。默认隔离到最底层。用户也可以选择“层次”多选按钮，选择隔离到任一层次。

通过“测试类型”、“使用角色”和“层次”多选框，选择测试分析时有效地测试。支持多选。其中：“使用角色”多选列表的内容是根据“模型”菜单－>“参数管理”－>“模型文件属性”菜单的“用户角色”属性页中的内容。用户可以自行修改

是否使用1522标准:静态分析此项设置为灰色,不可选.

输入输出端口属性页



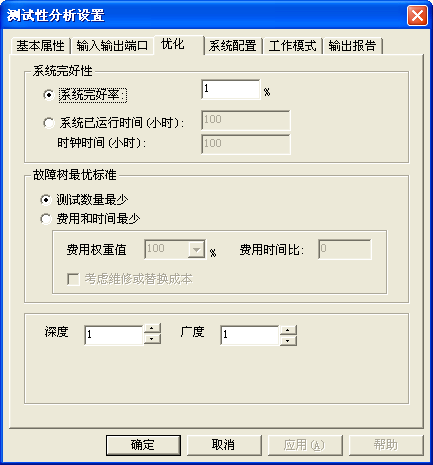
1. 分析设置固有故障源和测试点

本属性页需要和模型元素的“端口属性”属性页配合使用。此处的设置为全局设置。详细说明参考作为《端口属性页》的“[作为故障源](#topic_pin_propertyset_pin)[”和“作为信号源](#topic_pin_propertyset_signal)”选择项。

 是否把系统输入当作故障源

 是否把输出端口当作故障源

优化选项属性页



1. 分析设置优化

系统完好性:本节选项与系统可靠性指标的计算相关.可靠性基于时间改变，对于有一些部件是基于其在系统中的运行时间改变，有一些部件则是基于其寿命（时钟时间）

系统完好率 = 1-系统中所有模块的故障概率之和,是个估计值。

系统运行时间：该模块在本系统已经运行了多长时间。默认为100小时。

时钟时间:模块的出厂后寿命，就是说这种部件即使不运行可靠性也在变坏.

故障树最优标准:选择测试最优的方法.是选择使用测试最少的方式还是选择测试费用和时间最少的方式.

测试数量最少：故障分析过程中，选择测试数量最少的方式进行故障树的分析。每一个测试的费用和时间成本等都不考虑，默认为1；只考虑测试数量最少的测试方法。

费用和时间最少：将每个测试的费用和时间合计后，找出使用的费用和时间最少的故障树的方法。

费用权重值是表示费用考虑在权值计算中的比例.

总权值＝费用×费用权重＋时间×费用时间比×（1-费用权重);

其中:

时间×费用时间比×(1-费用权重);表示的是把时间转换成某种可以和费用相比较的量

费用:来自于模块的维修与替换的成本值.

费用权重比:测试费用权重值.

费用时间比:测试费用和测试时间的比例.

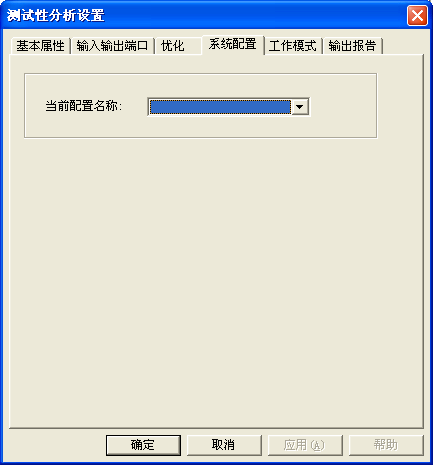
是否考虑维修或者替换成本:默认为否.选择"是"的话,在分析过程中,会把"维修与替换"属性页中定义的各项成本计算在内.

分析选项:分析故障树时候的选项

是否使用多故障模型生成故障树:默认系统均为单故障模型.如果选择此选项,系统按照多故障模型进行分析.

是否把替换当作测试:默认不把测试当作测试.如果选择此选项,系统分析时,会增加测试的内容.

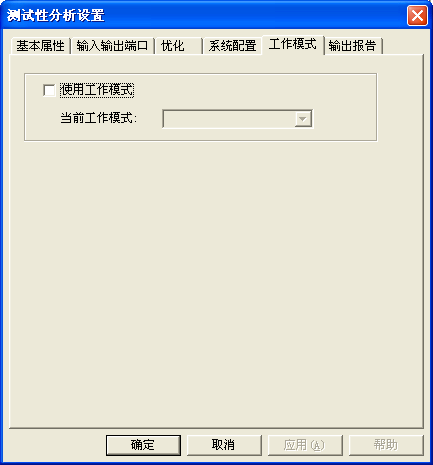
系统配置属性页



1. 分析设置系统配置

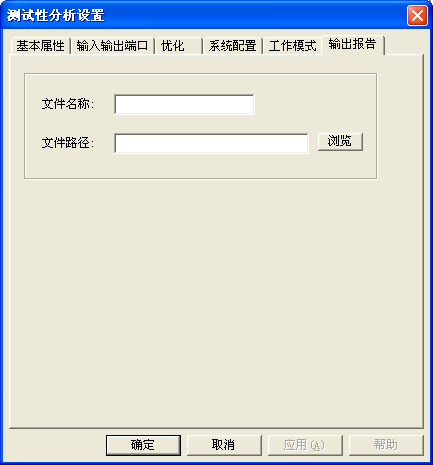
选择当前配置名称：选择当前静态分析时所使用的配置名称。单选。

工作模式属性页



1. 分析设置工作模式

输出报告属性页



1. 分析设置输出

输出报告文件名称：输入测试报告的名称。

输出报告文件保存路径选择指定：输入测试报告的保存路径。

## 分析算法

* + 1. 模型表达

测试性模型主要表达设备的测试诊断策略的能力，所表示的是对于故障系统或者设备的测试项目组合检测与隔离能力，即对应一个设备/系统中，所有可能的故障是如何分布的，对应的系统结构中有哪些测试项目，根据每一步测试输出的不同，应该如何判断和推测系统所处状态的问题。

数学理论上，故障诊断问题可以由以下的五元组问题（S,p,T,c,D）进行描述，其中

S = {s1, s2, ... , sm}是一个系统状态的有限集合， si (1≤i≤m)表示系统不同的故障状态；

p = [p(s1),p(s2),...,p(sm)]T表示系统各种故障状态的先验概率矢量

T = {t1,t2,...,tn}表示系统中可以使用的n个测试的有限集合；

c = [c1,c2,...,cn]T表示测试代价的矢量，这个测试成本可以表示为测试所需要花费的时间，所需要人力或者其它财务因素。

D=[dij]，表示测试与故障源之间的依存关系矩阵，其中，对于每一个测试tj,1≤j≤n，表示一个大小为m的二进制列矢量dj,表示测试tj能够检测到的故障，如果tj能够检测到某个故障状态si,那么测试矢量dj的第i个数值就为1。如果该测试不能够检测到该故障状态，那么测试矢量的对应位置的数值就为0。在被测试系统使用有向图的形式进行表达以后，可以对该有向图实行可达性分析，根据可达性分析的结果可以得到一个二进制可达性矩阵D=[dij]，用以表示故障词典。

* + 1. 算法求解问题

故障诊断方案问题就是根据以上五元组条件，如何求解最优的测试诊断顺序和方案，使得测试诊断所花费的成本和代价为最低。

即寻找测试序列使：

.........................(1)

式中：m为诊断树的分支数或终端节点数量，即确定每个故障或状态的结果。

Ki为第i个分支的过程节点数,即测试节点数，即为了得到这个诊断结果所经历的测试步骤数量。

Cj为得到该诊断结果的测试步骤中的一个测试的成本代价，这个代价可以是时间、人力或经济。

对于诊断策略的最优化问题可以分为两个方面，一个是正确地确定系统状态，另外一个是花费最小的代价。目前阶段，人们往往比较重视第一个方面，而不太重视第二个方面，但是从以往的维护实践活动，这两个问题是相关联的，而且故障往往最后总是能够被隔离出来，但是花费的时间和成本却会很高，难以在预估和控制的范围内完成。有时候维护人员常用的诊断方法和最优方法相差尽管不多，但是由于机载设备的维护活动往往在很大范围内有多个维护单位多次重复，又需要重复很长的时间，这样些微的差别也会带来巨大的资源和成本浪费，甚至影响正常的任务。

根据以上定义，如果不考虑实现需要的时间，寻找最佳诊断策略的方法可以用如下步骤表示：

1.建立UUT的相关性矩阵

建立的UUT相关性矩阵应包括无故障状态，并且进行简化，例如去掉冗余测试点，识别出隔离的模糊组。

2.列出所有可能的测试序列(诊断树)

列出所有的测试序列（诊断树）的方法可以采用逐步分割UUT的相关性矩阵。因为每一个测试都会将系统的可能状态进一步区分为不同的两组状态集合，那么测试过程就是不断使用测试将UUT依存矩阵分割为两个子矩阵的过程，那么下一个测试就会进一步分割这两个子矩阵之一，直到分割后的各子矩阵成为单行为止。然后选用另一个测试分割另外一个UUT子矩阵，重复上述分割过程，直到使用各可用测试点对UUT矩阵分割一遍为止，这样就可列出UUT所有可能的诊断树。

3.计算各个诊断序列的测试费用：

..........................(2)

4.找出最少费用的诊断序列

依次比较上一步中每个测试方法的测试费用，找出测试费用最小的那个测试方法。

根据以上定义，测试策略的数学模型由根节点、分支节点和终端节点（即树叶）组成，每个节点对应的一个测试点（或）测试和相应的测试代价。选用的测试不同，所需的测试代价不同，形成的分支及诊断树也就不同。对应一个UUT可能构成多种诊断树，这与UUT的相关性矩阵具体组成情况有关。各种诊断树都对应有其所需的测试费用，如果能够构成UUT各种可能的诊断树，计算出每个诊断树的测试费用，则其中费用最少的就是要寻求的最少费用诊断策略。

* + 1. 问题假设

前面已经定义，由有向图表示的系统的各种故障状态及其故障概率分别如下：

S = {s1, s2, ... , sm}是一个系统状态的有限集合， si (1≤i≤m)表示系统不同的故障状态；

p = [p(s1),p(s2),...,p(sm)]T表示系统各种故障状态的先验概率矢量

先假定在系统的所有状态中，在同一时刻只会发生单个状态si，0≤i≤m.

为了简化故障诊断的复杂性，多数系统的诊断策略和算法往往基于单故障假设和前提，在这种情况下，前面计算最优诊断策略或者某个解树的实际成本和代价中需要的节点概率会由于这个单故障的条件或假设发生变化。测试诊断问题转化为新的五元组问题（S,p,T,c,D），其中S,p,D的定义和以前稍有不同：

S = {s0, s1, s2, ... , sm}是一个系统状态的有限集合，s0表示系统的无故障状态，si (1≤i≤m)表示系统不同的故障状态；

p = [p0, p1, p2, ... , pm]T表示系统各种故障状态在单故障条件下的后验概率矢量（即条件概率矢量）,其中P0表示系统无故障的后验条件概率；

D=[dij]，表示测试与故障源之间的依存关系矩阵，其中，对于每一个测试tj,1≤j≤n，表示一个大小为m+1的二进制列矢量dj,表示测试tj能够检测到的故障或状态，如果tj能够检测到某个故障或状态si,那么测试矢量dj的第i个数值就为1。

根据贝叶斯定理： P(A|B) =  可以得到单故障假设下的各个故障的条件概率。

Pi==

P0 ==

可以看出，在单故障假设之下，不能直接使用该故障源的无条件概率来进行代价的估算。

* + 1. 数据结构定义

与/或树是用于表示问题及其求解过程的一种形式化方法，通常用于表示比较复杂问题的求解。对于一个复杂问题，直接求解往往比较困难。此时，可通过下述方法进行简化：

**分解**

把一个复杂问题分解为若干个较为简单的子问题，每个子问题又可继续分解为若干个更为简单的子问题，重复此过程，直到不需要再分解或者不能再分解为止。然后对每个子问题分别进行求解，最后把各子问题的解复合起来就得到了原问题的解。问题的这一分解过程可用一个图表示出来。例如，把问题P分解为3个子问题P1,P2,P3，可用图6-3表示。

诊断过程的最初状态包含着UUT的各种故障状态和无故障状态，是一个待诊断的模糊集。这个最初状态上的任意测试把这个UUT模糊集一分为二成为两个子集，一个表示测试通过时系统的可能状态集合，另外一个表示测试失败时系统的可能状态集合。引出这两个分支后，分支节点上的测试再次把对应的待诊断模糊子集一分为二，成为更小的子集。每次分解都会进一步降低其模糊度（待识别的UUT状态数），沿着各分支节点依次分解对应的子集，直到各个子集只包含一种确定的UUT状态或无法再分解为止。所以，对某个UUT故障进行诊断的过程，实际上就是把UUT的可能状态逐步分解成由大到小的子集，逐步诊断各模糊子集的过程。

**等价变换**

对于一个复杂，除了可用“分解”方法进行求解外，还可利用同构或同态的等价变换，把它变换为若干个较容易求解的新问题。若新问题中有一个可求解，则就得到了原问题的解。

问题的等价变换过程，可以使用一个图表示出来，称为“或”树。例如，问题P被等价变换为新问题P1,P2,P3,可用图6-4表示。其中，新问题P1,P2,P3中只要有一个可解，则原问题就可解，称为P1,P2,P3之间存在“或”关系；节点P称为“或”节点；有P,P1,P2,P3所构成的图是一个“或”树。

与/或树名词定义

**与/或树**

与/或树是一个由初始节点S（根节点）和若干终止分支节点（目标节点）s0,s1,…,sm组成的有向图，初始节点S代表需要解决的问题，终止分支节点对应于分解后的可以解决的子问题。此外，与/或树中还有中间节点，中间节点由3种类型组成：或节点，与节点，端节点。

**本原问题**

不能再分解或变换，而且直接可解的子问题称为本原问题。

例如在故障诊断过程中，如果UUT状态被唯一确定下来，那么这个状态就是一个本原问题。

**端节点与终止节点**

在与或树中，没有子节点的节点称为端节点；本原问题所对应的节点称为终止节点。显然，终止节点一定是端节点，但端节点不一定是终止节点。

例如在在故障诊断过程中，由于没有测试能够进一步分辨UUT的状态集合，那么这个节点就是端节点，但是如果问题的要求是唯一确定UUT状态，那么这个节点就不是终止节点。

**可解节点**

在与或树中，满足下列条件之一，称为可解节点：

* 是终止节点。
* 节点本身为或节点，且其子节点中至少有一个是可解的。
* 节点本身为与节点，且其子节点全部都是可解的。

**不可解节点**

不满足上面三条规则中任何一条的称为不可解节点。

例如，不能使用测试进一步分辨UUT的实际状态，这个节点就是不可解节点。

**解树**

由可解节点所构成，并且由这些可解节点可推出出示节点（对应原始问题）为可解节点的子树称为解树。

例如，在故障诊断中，由UUT原始状态及依次的测试步骤和各种测试结论对应的确定的UUT状态就是一个解树。对于一个本原问题，解树可能不止一个，但是解树之间可能存在优劣之分。

* + 1. 诊断树序列问题定义

诊断树测试序列问题按以下方式进行对应：



1. 与或树与诊断序列对应

初始节点对应诊断过程中最开始的UUT状态，这时候还未进行任何测试，UUT状态完全不确定。

中间节点对应诊断过程的中间状态，这时候已进行了一些测试，但是UUT状态还存在一定的模糊组，需要进一步加以分辨才能最终确定。

目标节点对应诊断的结论，这时候已经明确地获得UUT所处的状态，即已经确定出UUT所具有的故障。

与节点对应一个测试节点，这时候测试的输出可能会有“测试通过”和“测试失败”等结论组成，这时候无论是“测试通过”所形成的UUT状态节点还是“测试失败”所形成的UUT状态节点都必须能够是可解的，这个测试节点才能是可解的。解树中如果包含了一个测试节点，那么应该包含它的所有结论分支。

或节点对应一个中间状态节点，即对模糊状态节点，这时候无论使用哪一个剩余的测试来继续进行测试，只要任意的测试节点（与节点）可解，这个或节点就是可解的。解树中如果包含了一个或节点，那么只需要包含它的一个分支节点即可。这个分支节点对应最优解。

* + 1. 最佳诊断序列搜索算法

诊断序列的求解问题在本软件中等效为与或树的搜索问题，就是搜索与或树寻找和标识可解节点和不可解节点的过程，一个节点是否可解是由它的子节点确定的，对于一个“与节点”，只有当其子节点全部为可解节点时，它才为可解节点，只要子节点中有一个为不可解节点，它就是不可解节点；对于一个“或节点”，只要子节点中有一个是可解节点，它就是可解节点，只有当全部子节点都是不可解节点时，它才是不可解节点。像这样由可解子节点来确定父节点、祖父节点等为可解节点的过程称为可解标识过程；由不可解子节点来确定其父节点、祖父节点等为不可解节点的过程称为不可解标识过程。在与或树的搜索过程中将反复使用这两个过程，直到初始节点（即原始问题）被标识为可解或不可解节点为止。

对于排故过程来说，与或树的搜索就是确定诊断策略的故障树，但是通过与或树搜索出来的解树不一定就是最优排故过程，只是表示可以通过解树表示的测试过程确定出所有的故障。

与或树的一般搜索过程

把原始问题作为初始节点，并把它作为当前节点，即包含所有可能的UUT状态的节点。

应用分解或等价变换对当前节点进行扩展，对于排故过程来说分别对应于选择下一步测试和选择测试输出。

为每个子节点设置指向父节点的指针。

选择合适的子节点作为当前节点，反复执行第2步和第3步，在此期间要多次调用可解标识过程和不可解标识过程，直到初始节点被标识为可解节点或不可解节点为止。

与或树搜索的目标是寻找解树，从而获得原始问题的解。如果在搜索的某一个时刻，通过可解标识过程可确定初始节点是可解的，则由此初始节点及其下属的可解节点就构成了解树。如果选择被扩展的节点实际不可扩展，同时也不是终止节点，那么此节点就是不可解节点。此时可应用不可解标识过程来确定初始节点是否为不可解节点，如果可以肯定初始节点是不可解的，则搜索失败，否则继续扩展节点。

可解和不可解标识过程都是自下而上进行的，即由子节点的可解性确定父节点的可解性。由于与或树搜索的目标是寻找解树，因此，如果已确定某个节点为可解节点，则其不可解的子节点就不再有用，可从搜索树中删去；如果已确定某个节点为不可解节点，则其全部子节点都不再有用，可从搜索树中删去，但当前这个不可解节点还不能删去，因为在判断其父节点、祖父节点等的可解性时还需要用到。这两个算法可以提高搜索效率。

与或树的广度优先搜索

与或树的广度优先搜索算法如下：



1. 与或树的广度优先搜索算法

把初始节点放入open表

把open表中的第一个节点取出放入closed表

如果节点可扩展，执行以下工作：扩展节点n,依次将扩展后的子节点放入open表的尾部，并为每个子节点配置指向父节点的指针；依次考察扩展后的子节点是否为终止节点，如果有，将终止节点标识为可解节点，并应用可解标识过程对其父节点、祖父节点等先辈节点中的可解节点进行标识。如果初始节点也被标识为可解节点，就得到了解树，搜索成功，退出搜索过程；如果初始节点的状态还是未知，则从open表中删去具有可解先辈的节点；然后转到第2步重复执行。

如果节点不可扩展，执行以下工作：标识节点n为不可扩展节点,并应用不可解标识过程对其父节点、祖父节点等先辈节点中的不可解节点进行标识。如果初始节点也被标识为不可解节点，则搜索失败，退出搜索过程；如果初始节点的状态还是未知，则从open表中删去具有不可解先辈的节点；然后转到第2步重复执行。

与或树的深度优先搜索



1. 与或树的深度优先搜索算法

把初始节点放入open表

把open表中的第一个节点取出放入closed表

如果节点可扩展，执行以下工作：扩展节点n,依次将扩展后的子节点放入open表的头部，并为每个子节点配置指向父节点的指针；依次考察扩展后的子节点是否为终止节点，如果有，将终止节点标识为可解节点，并应用可解标识过程对其父节点、祖父节点等先辈节点中的可解节点进行标识。如果初始节点也被标识为可解节点，就得到了解树，搜索成功，退出搜索过程；如果初始节点的状态还是未知，则从open表中删去具有可解先辈的节点；然后转到第2步重复执行。

如果节点不可扩展，执行以下工作：标识节点n为不可扩展节点,并应用不可解标识过程对其父节点、祖父节点等先辈节点中的不可解节点进行标识。如果初始节点也被标识为不可解节点，则搜索失败，退出搜索过程；如果初始节点的状态还是未知，则从open表中删去具有不可解先辈的节点；然后转到第2步重复执行。

# 故障逻辑推理与定位工具

## 软件需求分析

故障诊断软件的功能需求如下：

1. 能够加载测试性建模软件导出的测试性模型和依存矩阵；
2. 能够输出测试性模型中包含的系统组成信息；
3. 能够输出测试性模型中包含的所有测试点信息；
4. 能够输出测试型模型中包含的所有故障模式信息；
5. 能够根据输入的所有测试项的测试结果进行诊断推理，输出诊断结果；
6. 诊断结果包含模糊组id和对应的故障模式列表等信息；
7. 如果输出的诊断结果为多个故障模式的可能，能够根据测试性建模软件里录入的信息给出故障模式的可能性排序。

软件的技术指标要求如下：

1. 软件加载诊断模型文件的时间不大于2S；
2. 故障逻辑推理时间不大于2S。

## 接口设计

对于故障诊断软件来说，对外的接口主要包括获取测试数据、输出诊断结果以及支持输出测试性诊断模型中的系统信息给整个健康管理系统使用（支持直接存储到数据库中）。软件的对外接口图如下所示：



1. 对外接口图
   * 1. 加载模型接口

接口类型：API

方向：健康管理系统→故障诊断软件

功能：导入测试性诊断模型

数据内容：模型文件

* + 1. 获取测试数据接口

接口类型：API

方向：健康管理系统→故障诊断软件

功能：根据测试性模型内容存储所有的故障模式信息

数据内容：测试失败/成功列表。

* + 1. 输出诊断结果接口

接口类型：API

方向：故障诊断软件→健康管理系统

功能：根据输入的测试结果列表推理出诊断结果

数据内容：故障时间、模糊组id（可关联查询故障模式或故障设备可能性列表）

* + 1. 输出系统组成信息接口

接口类型：API

方向：故障诊断软件→数据库

功能：根据测试性模型内容存储所有的系统组成信息

数据内容：子系统名称/id列表

* + 1. 输出测试信息接口

接口类型：API

方向：故障诊断软件→数据库

功能：根据测试性模型内容存储所有的系统测试信息

数据内容：测试点名称/id

* + 1. 输出故障模式信息接口

接口类型：API

方向：故障诊断软件→数据库

功能：根据测试性模型内容存储所有的故障模式信息

数据内容：故障模式id/名称/故障率列表

## 软件流程设计

故障诊断软件启动后，首先导入基于测试性的诊断模型，获取诊断推理知识。同时，通过模型文件的内容获取系统组成、测试点列表、故障模式列表等系统相关信息存储在本地数据库中。

当软件通过对外接口获取到满足诊断模型要求的一组测试结果数据后，故障诊断软件将测试结果解析后传递给诊断算法模块；诊断算法模块基于导入的诊断知识进行故障推理定位，将诊断结果返回。

故障诊断软件将诊断结果通过外部通信结果发给健康管理系统的其他模块，同时存储在本地数据库中。

对于故障诊断软件中的人机界面模块，主要通过访问数据库获取当前系统故障信息，通过可视化界面实现故障告警提醒。

软件的运行流程如下图所示：



1. 故障诊断流程图

对于故障诊断软件中的故障推理过程，首先通过基于测试性的诊断模型获取对应的故障诊断知识，当获取到系统测试结果后，诊断算法通过所有测试的结果（通过或失败），来逐步排除不可能的故障模式或设备，从而将故障定位到一定的层次。

由于实际的系统都比较复杂，对应的系统测试性设计很难达到100%的故障隔离率，因此最终推理的结果可能是多个故障模式的可能。故障诊断软件在加载诊断模型后，会根据不同的可能性建立当前系统的模糊组列表，每个模糊组对应一种故障可能，同时将此信息存储在本地数据库中。

故障诊断软件在实际运行时，只输出故障模糊组信息，软件界面或健康管理系统的其他模块可通过数据库查询此模糊组id对应的故障模式信息以及故障模式对应的设备西南西，同时根据每个故障模式包含的故障率大小来给出可能故障模式的排序。

## 功能模块设计

本项目中故障逻辑推理与定位工具基于标准化和可扩展的设计思想，模块输入输出的数据设计封装为标准数据结构，通过标准数据通信接口实现。软件作为健康管理系统的一个功能模块能够被加载到系统中调用。

软件在接收到输入数据后，通过解析数据中的数据值传递给算法输入；算法完成运算后，将算法的输出结果封装成标准数据发送出去；同时，软件支持输出从测试性模型文件中获取的一系列系统设计信息（包括系统组成、故障模式列表、测试点列表等），可以存储到数据库中用于整个健康管理系统的信息获取。

软件包括了CBM通用模块、基于测试性模型的故障诊断算法模块、数据存储模块以及可视化界面。其中，CBM通用模块按照OSA-CBM设计实现，模块包含一个标准XML格式的配置文件，该配置文件描述了该模块有哪些输出端口，从哪个下游模块的哪个端口输入数据，该模块实现了哪些算法等。算法模块中包含了对应的算法模型文件。

故障诊断软件的组成如下图所示：



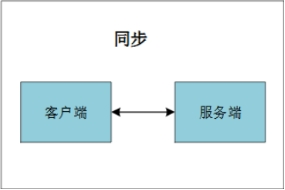
1. 诊断模块组成图
   * 1. 通用模块

通用模块主要负责对外输入输出数据，对内调用算法，用于实现数据处理的整体运行流程。主要功能包括：模块初始化，获取输入数据，调用算法获取算法计算结果，发送输出数据等。

通用模块采用2种OSA-CBM标准接口对外通信，以确保不同开发单位开发的PHM数据处理模块可以相互交换数据和配置信息。

1）同步通信接口

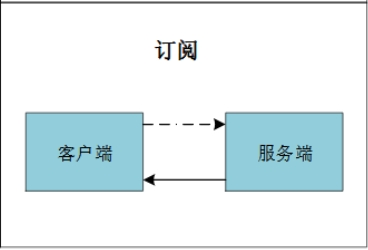
通过同步通信方式来获取状态数据信息，在同步接口的参数中定义需要获取的监控数据信息，返回数据内容。



1. 同步转发

2）订阅通信接口

对于装备的状态数据，通常通过订阅通信方式来获取实时数据信息，在系统初始化时先订阅需要的数据，其他各功能模块会自动发送已订阅的数据到指定功能模块。



1. 订阅转发

其中，模块配置文件由输入端口配置信息、输出端口配置信息、算法配置信息、模块描述信息和扩展配置信息五部分组成。算法配置信息描述了当前模块包含的算法的相关配置信息，包括算法描述信息、输入输出端口描述等；输入端口配置描述了当前模块的所有输入数据的接口信息，包括输入端口的名称，id和数据类型等；模块描述信息用于描述模块的基本信息；输出端口配置描述了当前模块输出端口的接口信息，包括输出数据的名称、id和数据类型等，一个模块的输出端口可作为下一个模块的输入端口。扩展配置描述了模块一些额外信息，用于模块的功能扩展。

通用模块的处理流程如下图所示。软件启动后，首先实现模块初始化功能，加载模块配置文件，获取当前诊断模块的需要输入输出的数据信息，需要调用的算法信息等。

然后，模块启动运行，等待获取输入数据；获取到模块需要的测试结果列表后，模块调用对应的测试性模型诊断算法模块，将测试结果列表传给算法模块，算法通过加载测试性模型来实现故障诊断定位功能，返回诊断结果。

通用模块获取诊断结果后，基于输出信息将数据发送给其他需要的模块（存储模块和健康管理系统）。



1. 通用模块运行流程
   * 1. 诊断算法模块

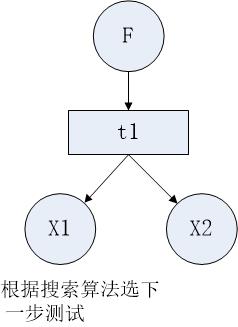
算法模块封装为一个动态链接库，在模块初始化时被加载，一个数据处理模块可以包括多个算法，所有的算法全部按照同一输出格式进行封装，在程序中按统一接口进行调用执行。算法包括接口如下：

1. 算法初始化接口
2. 算法调用接口
3. 关闭算法接口

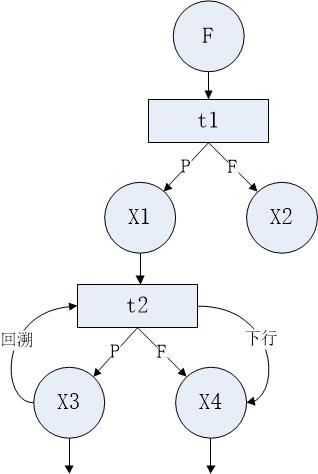
算法模型文件是算法对应的配置信息。主要包括算法在运算过程中用到的参数配置，在本项目中基于依存矩阵的诊断模型文件。诊断模型文件通过测试性建模与分析工具已经进行了优化处理，生成诊断的依存矩阵和模糊组id，软件在读取文件时不需要再去进行模糊组遍历，能够直接获取当前系统的所有模糊组信息和对应的故障模式信息。整个加载模型文件时间不大于1s。

基于测试性模型的故障推理的过程是遍历所有测试点不断的对故障集合进行隔离，直到故障集达到相应的故障隔离要求，或者不能够进一步进行隔离，则输出推理结论，具体思路如下：

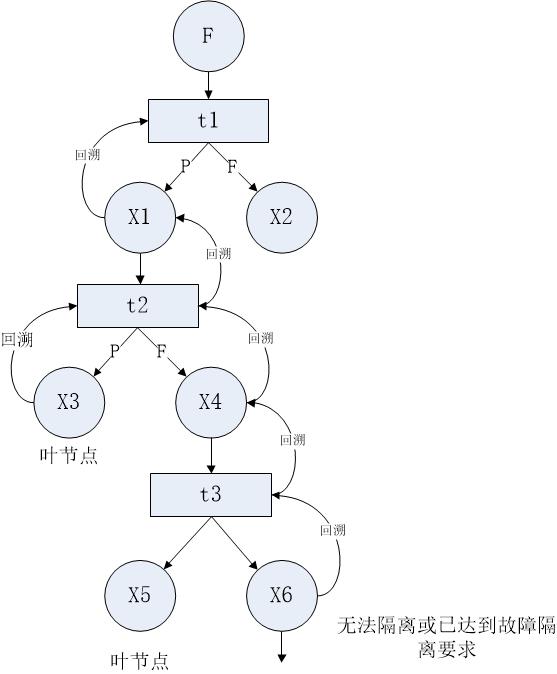
1. 假设根结点所对应的故障集为，在根结点处选取一个初始测试点，构成一个相应的与节点，根据测试通过与否分别产生两个分枝点，通过该测试的故障集和末通过该测试的故障集对应两个分枝点。在产生与节点和分枝点的过程中，先标记与节点上面的分枝点：然后按照左边优先的顺序，对与左侧的分支点，计算该分文点包含的故障集合和测试集合。



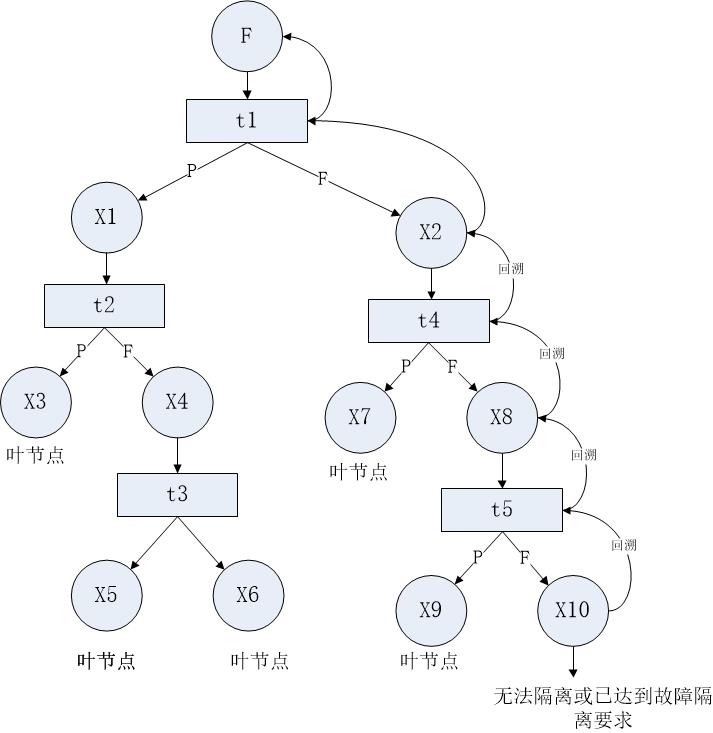
1. 故障诊断推理流程1
2. 对于某分枝点来说，如果其所对应的故障集已经达到故障隔离要求，或者不能够进一步进行隔离，那么就将该分枝点改为叶节点，并进行标记。然后再向上回溯一步，到达上面的与节点，判断该与节点下面是否还有未标记的分枝点，如果有，则继续下行至未标记的分枝点，如下图所示：



1. 故障诊断推理流程2
2. 假如某分枝点所对应的故障集已经达到故障隔离要求，或者不能够进一步进行隔离，那么就与（2）一样，将该分枝点改为叶节点，再回溯到它上面的与节点。如果该与节点下面没有未做标记的枝节点，就继续向上回溯，直到回溯到未标记的枝节点，然后选择新的测试点进行下一步测试，如下图所示：



1. 故障诊断推理流程3
2. 当回溯到根结点，结束算法，此时所有的故障模式已经遍历完成，如下图所示，输出所有被隔离的故障模式结合。



1. 故障诊断推理流程4

在诊断算法推理过程中，算法模块在初始化时会根据依存矩阵的矩阵大小先分配出内存空间用于推理计算，同时在处理过程中基于多线程处理，确保整个诊断推理过程不大于2s。

* + 1. 数据存储功能
       1. 数据存储流程设计

对于故障诊断软件中的数据存储过程，数据存储模块主要通过标准接口获取其他功能模块输出的数据，然后判断对应的数据类型，分别存储到不同的数据表中。

对于在数据存储模块中的数据接收和数据存储两个过程，分别通过不同的子线程来调用保证数据的实时存储效率。

数据存储流程图如下所示：



1. 数据存储流程图

同时，在本项目中数据存储模块还支持基于策略的存储功能，主要通过监控报告来实现数据的触发存储。监控报告主要包含三部分内容：

1）该报告对应的参数配置：报告名称、报告类型、监控频率（每隔多长时间监控一次）、存储数量、存储位置（存储的数据为监控触发前、触发中或者触发后）。

2）监控触发条件：监控参数数据id，参数触发判断（大于/等于/小于某数值等），与下一条监控参数信息的判断关系（与、或）。

3）存储内容：存储参数数据id等信息。

数据监控模块开始运行后，根据监控报告的内容，从数据缓存区读取监控参数的最新数据，然后判断是否满足触发存储的条件。如果条件判断满足，则从缓存区获取对应需要存储的数据存储到数据库中。数据监控模块中可以包括多个监控报告配置信息，每个监控报告的处理保持线程独立性（多线程处理）。

基于监控报告的存储过程如下：



1. 监控报告流程图
   * + 1. 数据表设计

故障诊断软件中的设计表主要用于在软件设计阶段存储系统的相关设计信息，如系统组成、测试信息、FMECA信息等。

下表中用于存储从测试性模型文件中获取的系统组成信息，每一个segment对应一个设备，包括id、名称、类型、危害等级、描述等信息。

1. 系统组成信息表

|  |  |  |
| --- | --- | --- |
| **字段** | **解释** | **大小/数据类型** |
| segment\_site | 主键。表示所属site。 | HEXSTRING(16) NOT NULL |
| segment\_id | id。Site下唯一标识一个segment。 | UINT NOT NULL |
| sg\_db\_site | 外键。唯一标识一个type类型。 | HEXSTRING(16) NOT NULL |
| sg\_db\_id | UINT NOT NULL |
| sg\_type\_code | UINT NOT NULL |
| criticality | 危害等级。 | INT |
| cs\_type\_db\_site | 外键唯一标识危害等级类型。 | HEXSTRING(16) |
| cs\_type\_db\_id | UINT |
| cs\_type\_code | UINT |
| user\_tag\_ident | 描述 | STRING(0:254) NOT NULL |
| name | 名称 | STRING(0:254) |
| long\_description | 长描述，定义长字节。 | STRING(0:4000) |
| gmt\_last\_updated | 上次更新时间 | DATETIME(19:29) |

下表中用于存储从测试性模型文件中获取的测试项信息和测试结果数据，每一个meas\_loc对应一个设备项，包括id、名称、类型、所属设备、描述等信息；对于从整个健康管理系统获取的实时测试结果，存储在测试结果数据表中，包括测试项id、测试值、单位、时间戳等信息。

1. 测试项信息表

|  |  |  |
| --- | --- | --- |
| **字段** | **解释** | **大小/数据类型** |
| meas\_loc\_site | 主键。标识所属site。 | HEXSTRING(16) NOT NULL |
| meas\_loc\_id | 主键。Id，site下唯一标识。 | UINT NOT NULL |
| ml\_db\_site | 外键，标识类型。 | HEXSTRING(16) NOT NULL |
| ml\_db\_id | UINT NOT NULL |
| ml\_type\_code | UINT NOT NULL |
| segment\_site | 外键，关联segment。 | HEXSTRING(16) |
| segment\_id | UINT |
| user\_tag\_ident | 描述 | STRING(0:254) NOT NULL |
| barcode | 编号 | STRING(0:254) |
| name | 名称 | STRING(0:254) |
| gmt\_last\_updated | 上次更新时间 | DATETIME(19:29) |

1. 测试结果数据表

|  |  |  |
| --- | --- | --- |
| **字段** | **解释** | **大小/数据类型** |
| meas\_loc\_site | 主键。id。 | HEXSTRING(16) NOT NULL |
| meas\_loc\_id | UINT NOT NULL |
| gmt\_event | 主键。对应一次Meas\_Event。 | DATETIME(19:29) NOT NULL |
| eu\_db\_site | 主键。关联实数数据的单位信息。 | HEXSTRING(16) NOT NULL |
| eu\_db\_id | UINT NOT NULL |
| eu\_type\_code | UINT NOT NULL |
| data\_value | 存储实数型数据数值 | DOUBLE |
| gmt\_last\_updated | 上次更新时间 | DATETIME(19:29) |

在健康管理数据库中，通过模糊组与FMECA的故障模式关联信息可通过故障诊断结果查询对应的故障信息。

其中，故障模糊组信息主要包括模糊组id、所属设备、类型、关联算法等信息；故障模式主要内容包括故障模式id、名称和故障率等。

1. 故障模糊组信息表

|  |  |  |
| --- | --- | --- |
| **字段** | **解释** | **大小/数据类型** |
| sg\_hyp\_amb\_set\_db\_site | 模糊组id。 | HEXSTRING(16) NOT NULL |
| sg\_hyp\_amb\_set\_db\_id | UINT NOT NULL |
| sg\_hyp\_amb\_set\_id | UINT NOT NULL |
| segment\_site | 外键，关联所属设备id。 | HEXSTRING(16) NOT NULL |
| segment\_id | UINT NOT NULL |
| amb\_set\_db\_site | 外键，关联模糊组类型。 | HEXSTRING(16) NOT NULL |
| amb\_set\_db\_id | UINT NOT NULL |
| amb\_set\_type\_code | UINT NOT NULL |
| by\_agent\_site | 外键，关联算法名称。 | HEXSTRING(16) NOT NULL |
| by\_agent\_id | UINT NOT NULL |
| user\_tag\_ident | 描述 | STRING(0:254) NOT NULL |
| name | 名称 | STRING(0:254) |
| gmt\_last\_updated | 上次更新时间 | DATETIME(19:29) |

1. 故障模式关联信息

|  |  |  |
| --- | --- | --- |
| **字段** | **解释** | **大小/数据类型** |
| sg\_hyp\_db\_site | 主键。故障模式Id。 | HEXSTRING(16) NOT NULL |
| sg\_hyp\_db\_id | UINT NOT NULL |
| sg\_hyp\_event\_id | UINT NOT NULL |
| segment\_site | 外键，关联所属设备id。 | HEXSTRING(16) NOT NULL |
| segment\_id | UINT NOT NULL |
| ev\_db\_site | 外键，类型信息。 | HEXSTRING(16) NOT NULL |
| ev\_db\_id | UINT NOT NULL |
| event\_type\_code | UINT NOT NULL |
| severity\_lev\_db\_site | 外键，表示严酷度。 | HEXSTRING(16) NOT NULL |
| severity\_lev\_db\_id | UINT NOT NULL |
| severity\_lev\_type\_code | UINT NOT NULL |
| gmt\_created | 产生时间。 | DATETIME(19:29) |
| cr\_loc\_hr\_delta | 与本地时间小时偏移量。 | SHORT |
| cr\_loc\_min\_delta | 与本地时间分钟偏移量。 | USHORT |
| by\_org\_asite | 外键，关联算法。 | HEXSTRING(16) |
| by\_agent\_id | UINT |
| user\_tag\_ident | 描述 | STRING(0:254) |
| name | 名称 | STRING(0:254) |
| assoc\_file\_name | 关联数据文件 | STRING(0:254) |
| associated\_blob | 关联数据块 | BASE64BINARY |
| gmt\_last\_updated | 上次更新时间 |  |

* + 1. 人机界面功能

人机交互界面主要通过系统组成和系统模型图来分层展示各级系统的故障状态，实现故障告警提醒功能。

目前故障诊断软件中，各系统通过测试性建模建立诊断模型，在软件初始化时加载对应的诊断模型，实现实时诊断定位功能。软件可视化界面通过当前故障与上次故障杜比，给出故障状态变化。这里故障状态变化包含以下几种情况：

1. 没有变化；
2. 没有故障——有故障
3. 有故障——有新的故障
4. 有故障——有故障消除

故障诊断软件在状态监控界面实现各设备的故障告警功能，当出现告警数据时，软件以图形化方式显示当前故障设备，使用人员可通过界面查看对应的详细告警信息，然后再完成排故工作后，填写维修活动记录。

故障诊断软件的故障告警如下图所示，整个状态监控界面根据系统组成分级显示。当出现告警信息时，界面对应的设备框图变红，用户可通过系统层级快速定位到具体最小可更换单元。点击对应的故障诊断“详情”按钮链接，则界面弹出故障详细信息窗口，如下图所示：



1. 故障告警显示



1. 故障详情显示

其中，故障描述项描述了发生的是什么故障；维修建议项为指导该故障维修的对应IETM章节的链接按钮；发生时间项为故障发生的时间；维修记录项对应填写维修活动记录表链接。

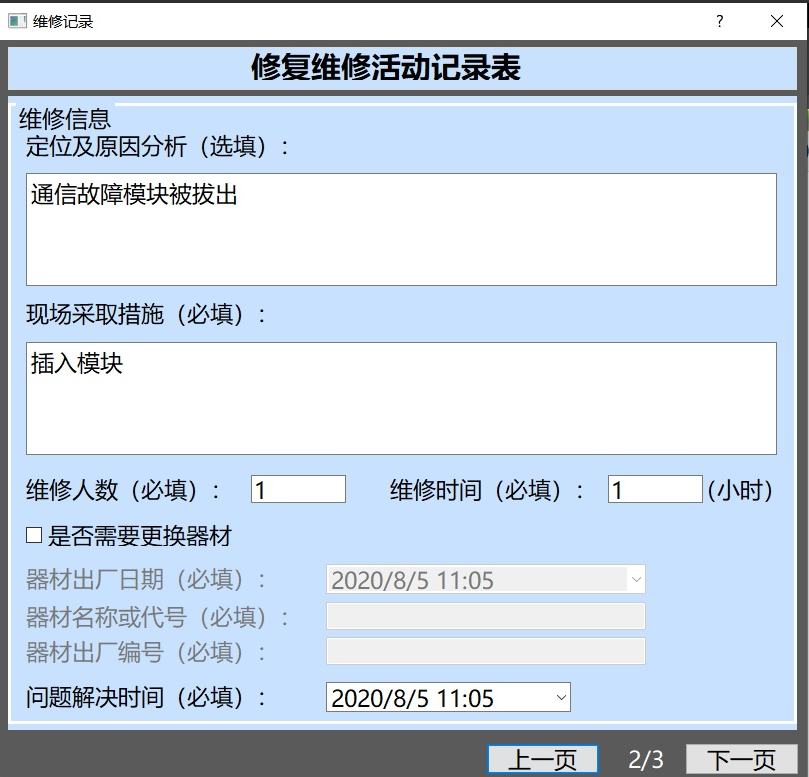
如果当前故障关联多个故障模式，则根据故障模式中的故障率信息按照故障可能性大小分行排序显示。

点击“IETM”链接按钮看到维修手册，维修人员根据IETM显示内容对当前的故障开展维修工作，如下图所示：



1. IETM显示

完成维修工作后，需要点击故障详情中的“维修记录”链接按钮来填写维修活动记录表：



1. 维修记录填写