火龙果 · 整理
uml.org.cn

基于模型的系统工程（ MBSE ）的案例研究，第 1 部分：IBM Rational Harmony 的集中式系统模型

建模自出现以来，一直是系统工程的重要组成部分。在过去十年中，工程师们已经大幅增加基于模型的技术的使用，并发展出一门新的学科，基于模型的系统工程（ Model-Based Systems Engineering, MBSE ）。这门学科与传统的系统工程不同，它强调中央系统模型，该模型同时捕捉系统需求和满足这些需求的设计决策。除了作为系统工程的工作构件的知识库之外，还可以通过模拟系统模型来验证成本、性能研究和设计选择。 IBM Rational Harmony for Systems Engineers 等广泛应用的 MBSE 流程重点关注的是系统功能分析，也就是说，关注如何将功能要求转换为一致的系统操作描述。然后，使用系统操作获得所分配系统架构块之间的端口和接口。这些接口形成了各子系统之间的正式切换的基础。

Mohit Choudhary 是一位基于模型的解决方案的架构师，他在使用先进技术构建软件密集型系统方面具有丰富经验。在捕获、设计、开发和维护软件密集型海军作战系统的需求方面，他拥有超过 12 年的工作经验。

本系列的这一部分旨在通过一个案例研究来探讨标准 MBSE 流程。首先，我们根据 UAV（无人驾驶飞机）地面站控制器的设计来拟定这个案例研究的范围。然后，我们会介绍 Rational Harmony 系统工程流程的基本概念、工作流和工作产品。最后，我们通过定义任务流来实现 UAV 地面站控制器的设计，同时构造每个阶段所需的构件。

在 IBM Bluemix 云平台上开发并部署您的下一个应用。

开始您的试用

案例研究

本案例研究基于对少部分 UAV 地面站控制器的设计分析，这些控制器的功能必须符合表 1 中的要求。

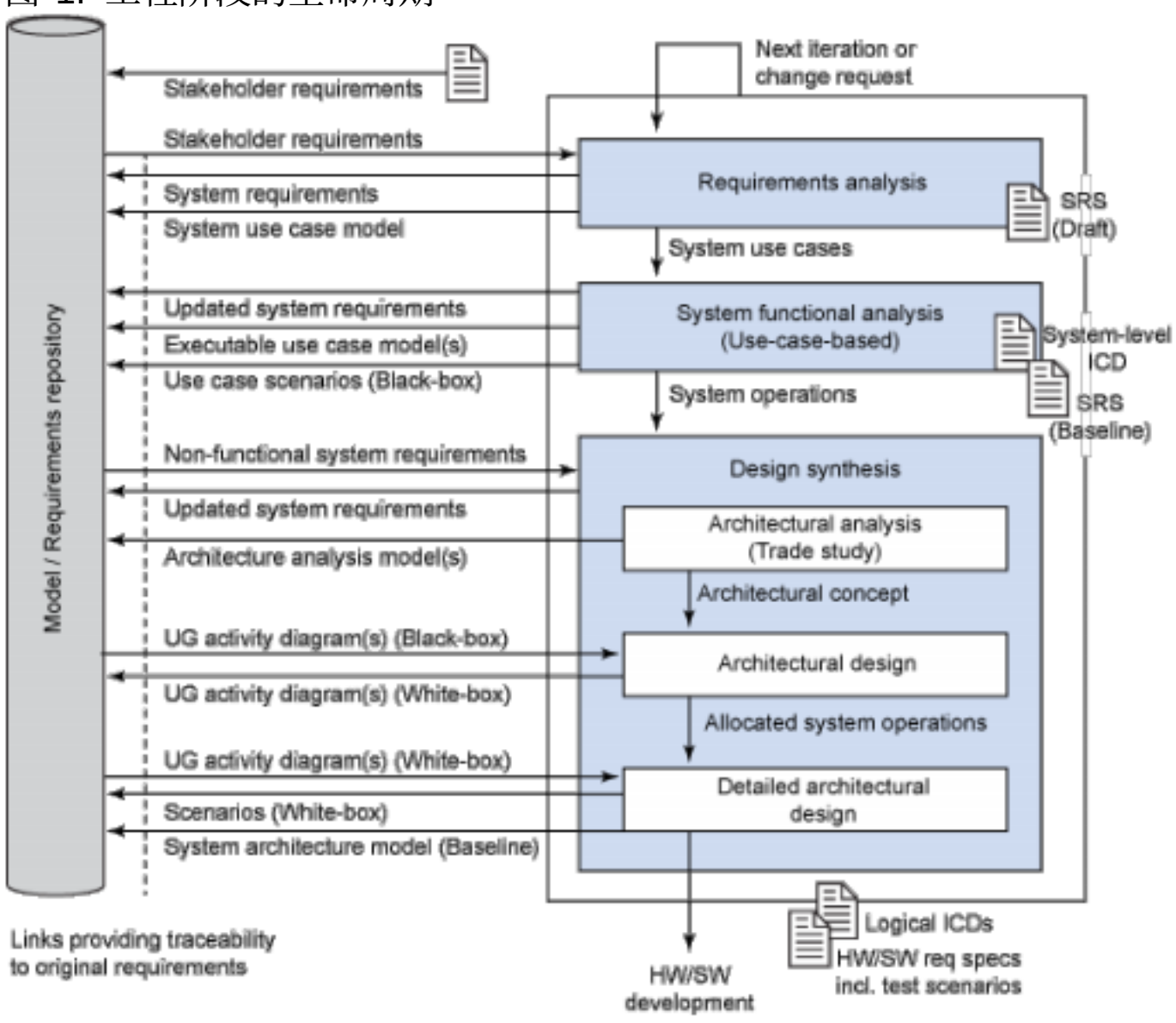
表 1. UAV 地面站控制器需求

需求引用	需求
01	实时飞行中的 UAV 的信息。（身份和传感器负载）
02	允许操作员将搜寻区域分配给选定的飞行中的 UAV。
03	以 1 次更新 /秒的频率接收来自 UAV 的传感器追踪信息。
04	在系统中保持 30 分钟的追踪历史。
05	允许操作员维护包含所采用的系统追踪信息的资料库。
06	最多维护 100 条 System Tracks （系统追踪信息）。
07	允许操作员对系统追踪信息执行生命周期操作（创建 /删除）。
08	每秒更新一次系统追踪信息，如果主传感器追踪信息更新可用，则使用该值进行更新，否则，使用 DR 的值进行更新。
09	使系统追踪信息可在显示屏上显示，并绘制其更新。
10	允许操作员将操作员辅助系统追踪信息与另一台 UAV 的传感器追踪信息相关联。
11	允许操作员将两条独立的系统追踪信息合并成一条。
12	将系统中的重要事件（比如创建和删除系统追踪信息）通知操作员。
13	允许操作员随时中止 UAV 搜索。

Rational Harmony　　系统工程中基于模型的系统工程

Rational Harmony for Systems Engineering　　使您能够识别并推导出所需的系统功能，还能够确定相关的系统模式和状态。此外，您还可以将已确定的系统功能和状态分配给子系统结构，并确定跨子系统的端口和接口。图 1 显示了您在每个工程阶段为了完成系统设计而必须执行的基本输入和输出。

图 1. 工程阶段的生命周期



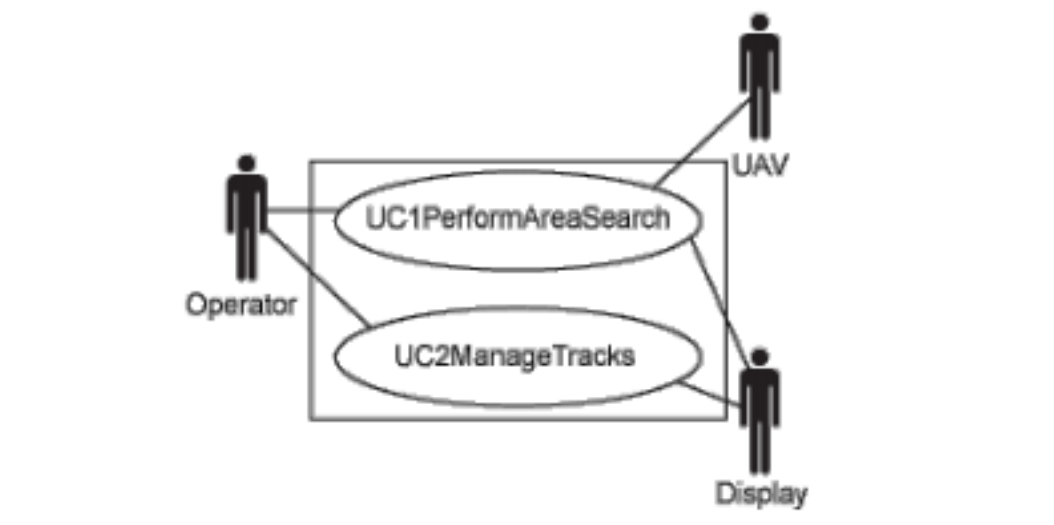
在功能分析阶段，通过一个活动图定义用例的功能流。然后，从活动图推导出用例场景。各场景通过一组序列图表示，创建用例块的端口和接口时需要用到它们。最后，用例基于状态的行为被捕获为一个状态图。

在架构设计阶段，选定的系统块被分解成几部分。最终的系统结构被捕获在　　SysML　块定义图　(BDD)　和 SysML　内部块图　(IBD)　中。每个用例分配都可以通过一个关联的白盒活动图以图形的形式表示。下图是用例的黑盒活动图的副本，但它被划分为泳道　　(swim lane)　。每条泳道代表分解层次的一个块。然后，根据选定的设计理念，将操作　　移动　”到各自的块泳道。这种分配的一个基本要求是要求维护操作之间的初始链接（功能流）。最后，详细架构设计阶段的重点是端口和接口的定义，以及在架构分解的最低层，系统块基于状态的行为的定义。

设计流程　/大纲

UAV 地面站的系统要求被划分成两个用例，如图　　2 所示。为了实现可追溯性，需要将已确定的系统要求与相关的用例相关联。在本文中，假设已经完成需求分析。对于本案例研究，我们将重点关注　　Uc1 PerformAreaSearch　用例。

图 2. UAV　管理系统用例图



功能分析

用例的功能流涵盖的方面包括：将搜索分配给选定的 UAV、接收来自 UAV 传感器的追踪信息、保持系统追踪信息与传感器追踪信息一致、维护所需要的传感器追踪信息更新历史、允许操作员中止搜索。您可以使用该工具在黑盒活动图中详细说明每个功能流，如图 3 所示。

图 3. 黑盒活动图

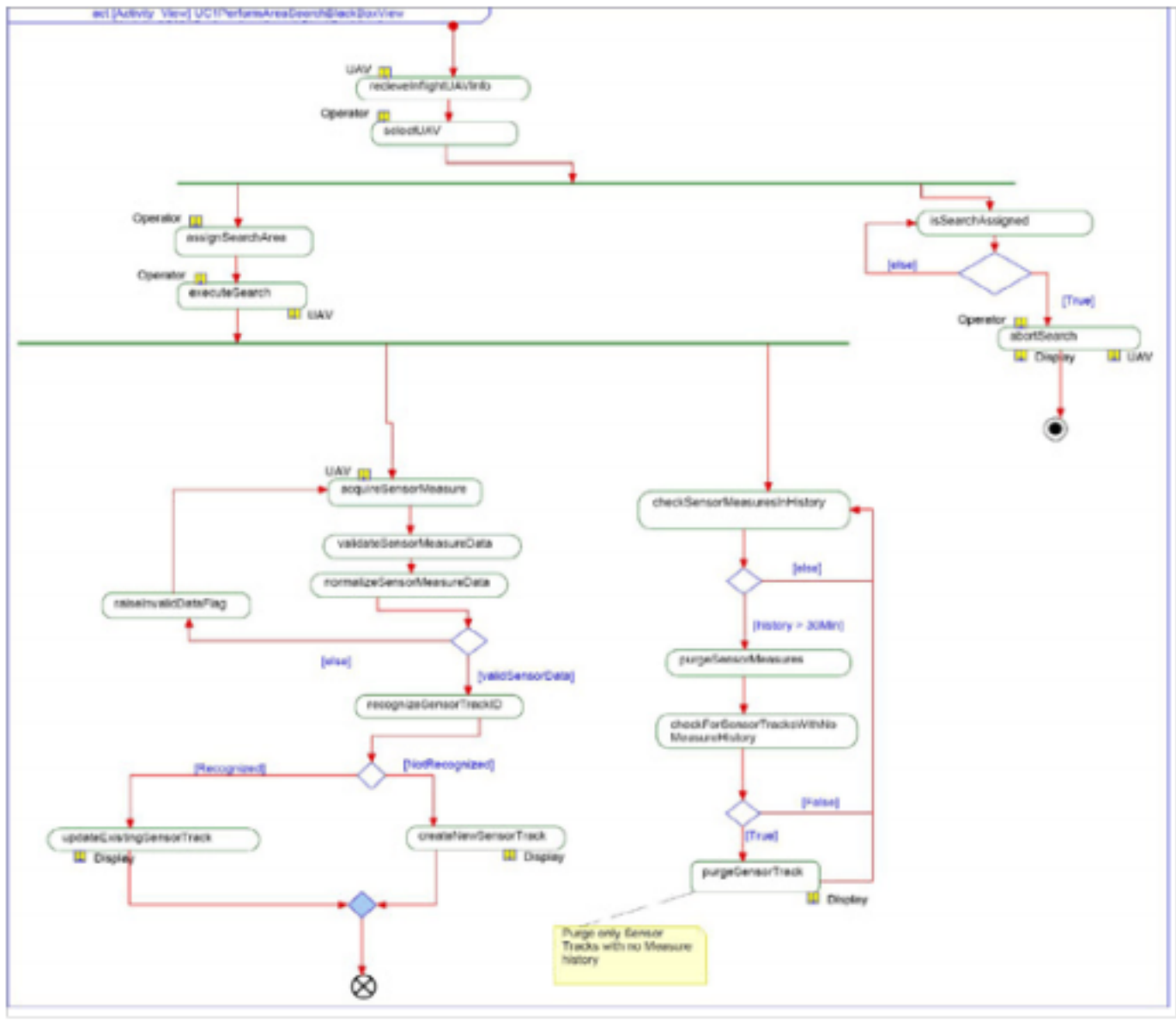


图 3 的大图

用例场景

您可以看到，活动图中的每个流都表示一个不同的用例场景。这些流不仅能帮助我们详细了解功能流中的操作，还能形成在各个开发阶段验证用例行为的基础。在图 4 所示的五个场景中，您可以通过其中三个场景获得我们的用例。

图 4. 黑盒用例场景

火龙果·整理
uml.org.cn

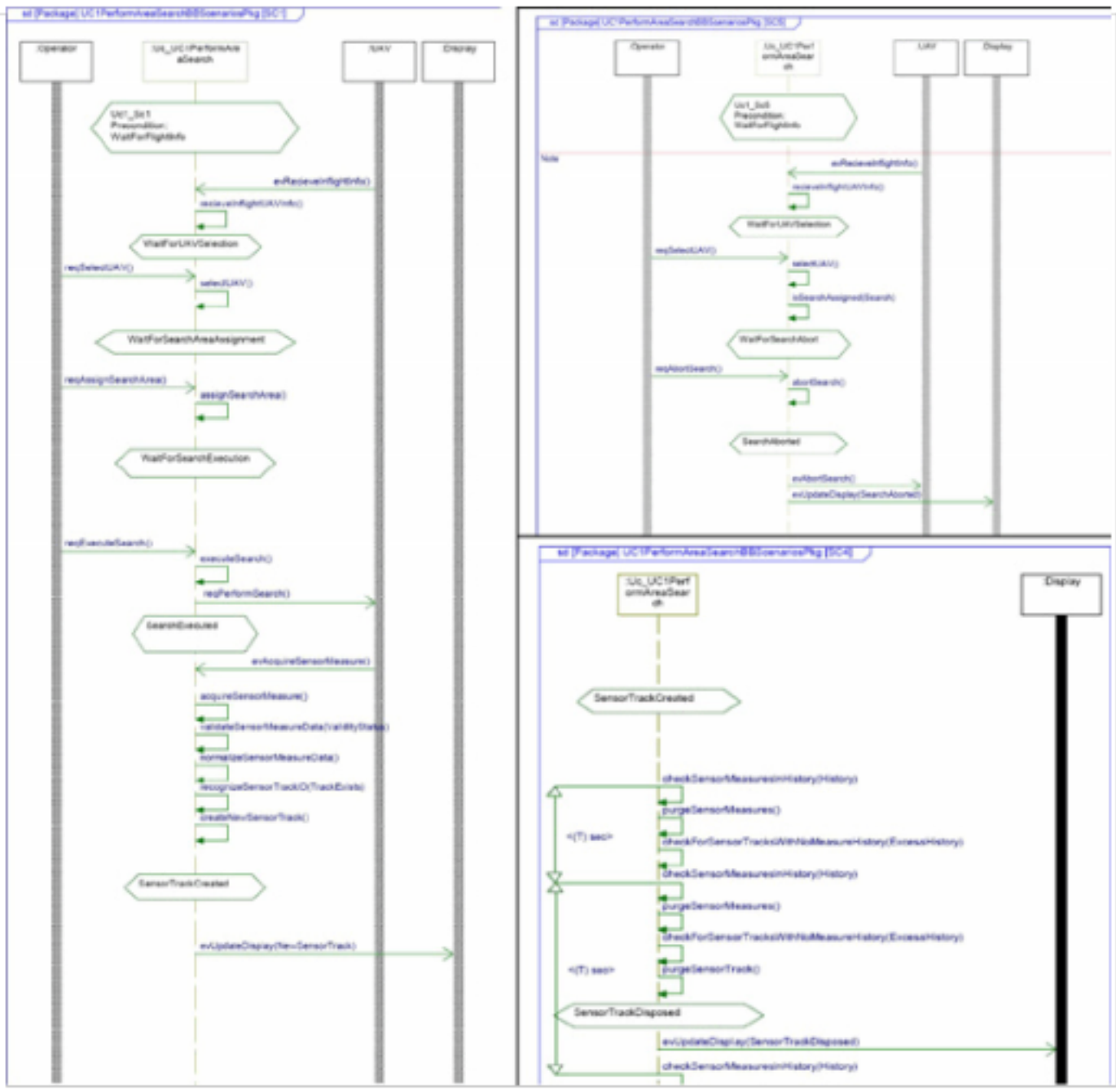


图 4 的大图

用例状态图

在下一步中，您可以使用序列图获得端口和接口。获得端口和接口之后，必须在状态图中捕捉用例的状态行为。最后，为了设定用例的黑盒行为的基线，需要执行状态机，并且将生成的序列图与刚才创建为场景的序列图进行对比。本用例的状态机如图 5 所示。

图 5. 黑盒状态图

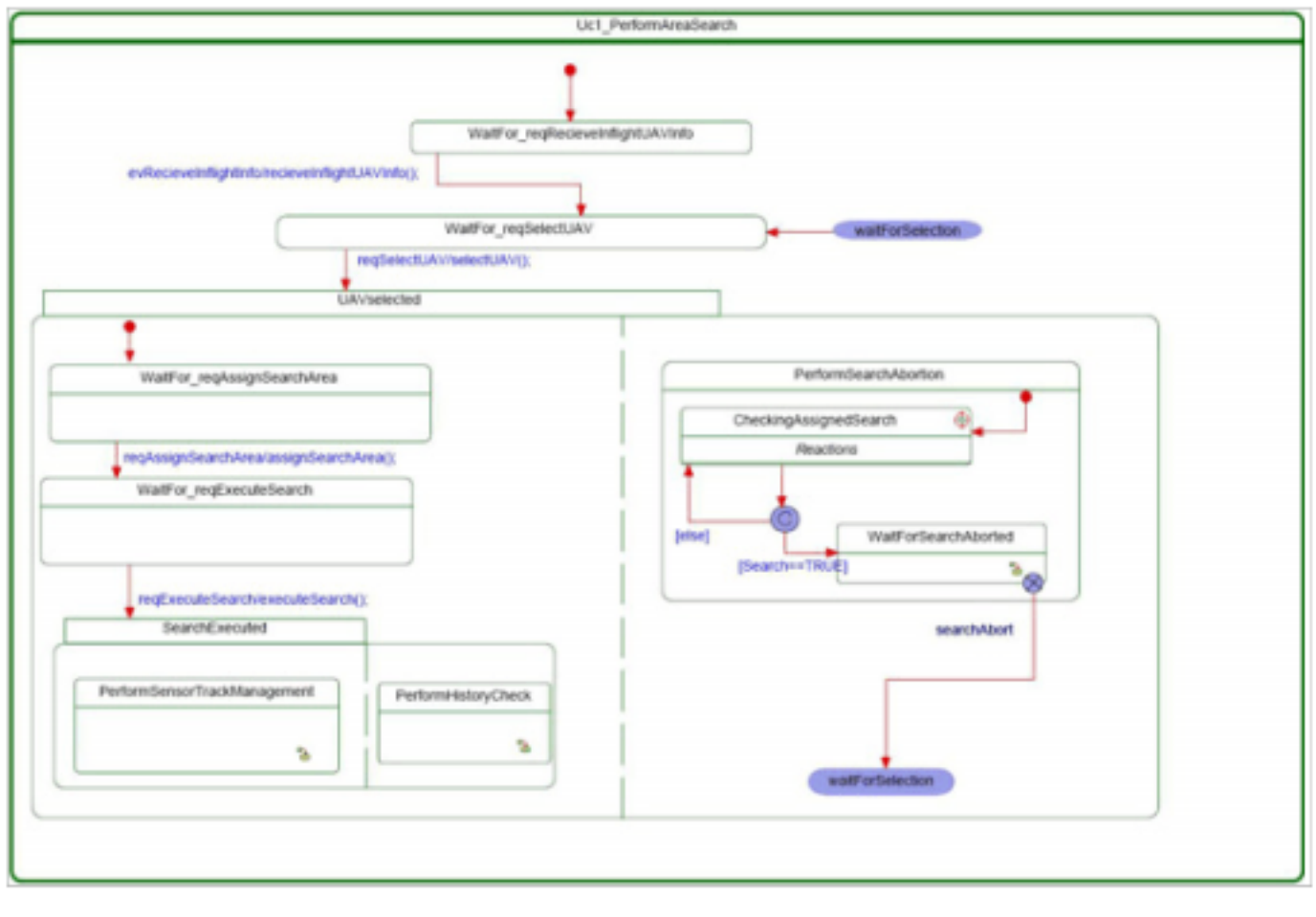


图 5 的大图

状态 “ Search Executed ”有两个 ‘ and’子状态： “ Perform Sensor Track Management 执行传感器追踪信息管理 ”和 “ Perform History Check 。”第一个子状态支持追踪信息的建立或更新，第二个子状态清除大于 30 分钟的传感器追踪信息历史，如果传感器追踪信息没有历史记录，则清除传感器追踪信息本身。

架构设计

在架构设计阶段，您需要重点关注结构分解，以及如何将操作和行为分配给子系统组件。首先，我们描述了将系统结构性分解成子系统的系统 BDD （参见图 6），然后我们将获得 Use Case White-Box Activity

Diagram，并通过它将用例的操作分配给分解后的子系统（参见图　　7）。

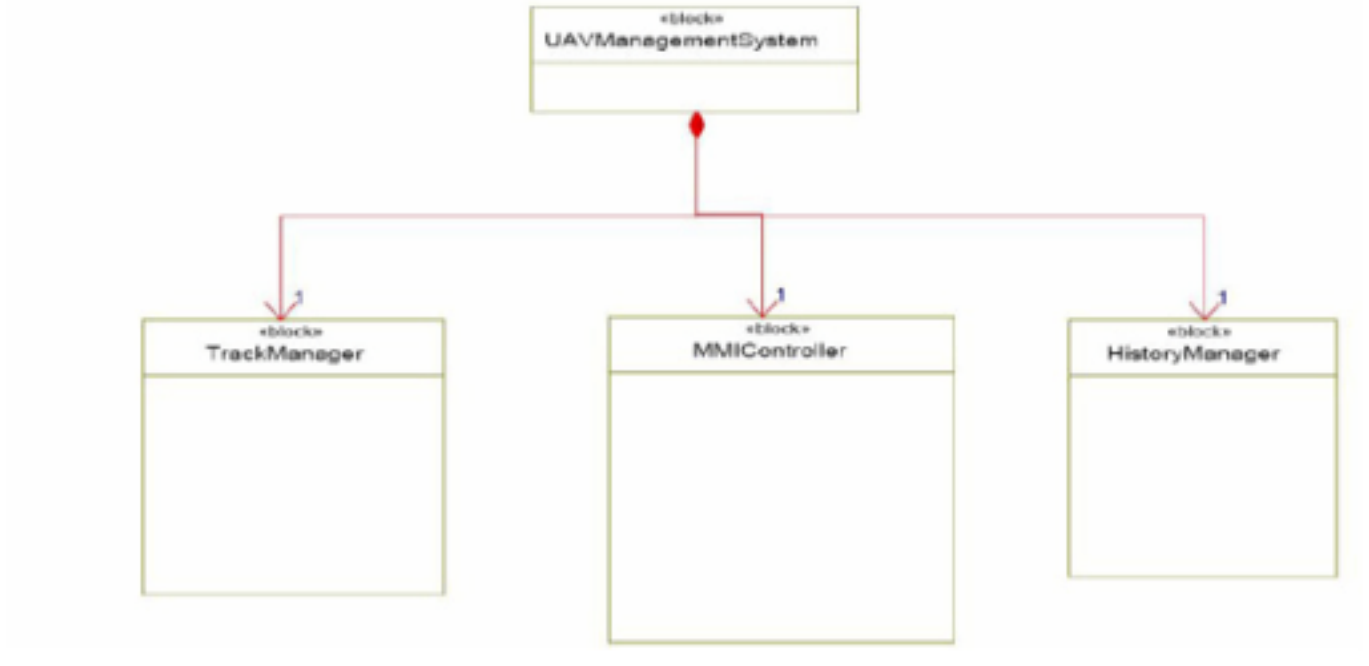
当将系统分解成子块后，它会以关键系统功能的定义为基础。这一阶段的目标是对系统功能进行分组，每个组可以通过一个子系统组件实现。第一步是将相关的系统功能划分为关键系统功能。对于本用例，我们通过用例黑盒活动图的分析，确定了以下三个关键系统功能：

管理传感器追踪信息

控制人机界面

执行历史管理

图 6. UAV 管理系统 BDD



考虑到要使用一些关键系统功能，我们获得了如图　　6 所示的 BDD。因为我们有子系统块，所以接下来的任务是在各个泳道中执行分配操作，以描绘每个独立的子系统块。以下是重要的分配规则：

如果您无法将操作分配到单个块，那么必须将操作分解。在这种情况下，已分解的相关业务必须通过各自的依赖关系链接到父操作。

您可以将一个系统级的操作分配到多个块。在这种情况下，需要将相关的操作复制到相应的块泳道，并将它们集成到功能流中。

图 7. 白盒活动图

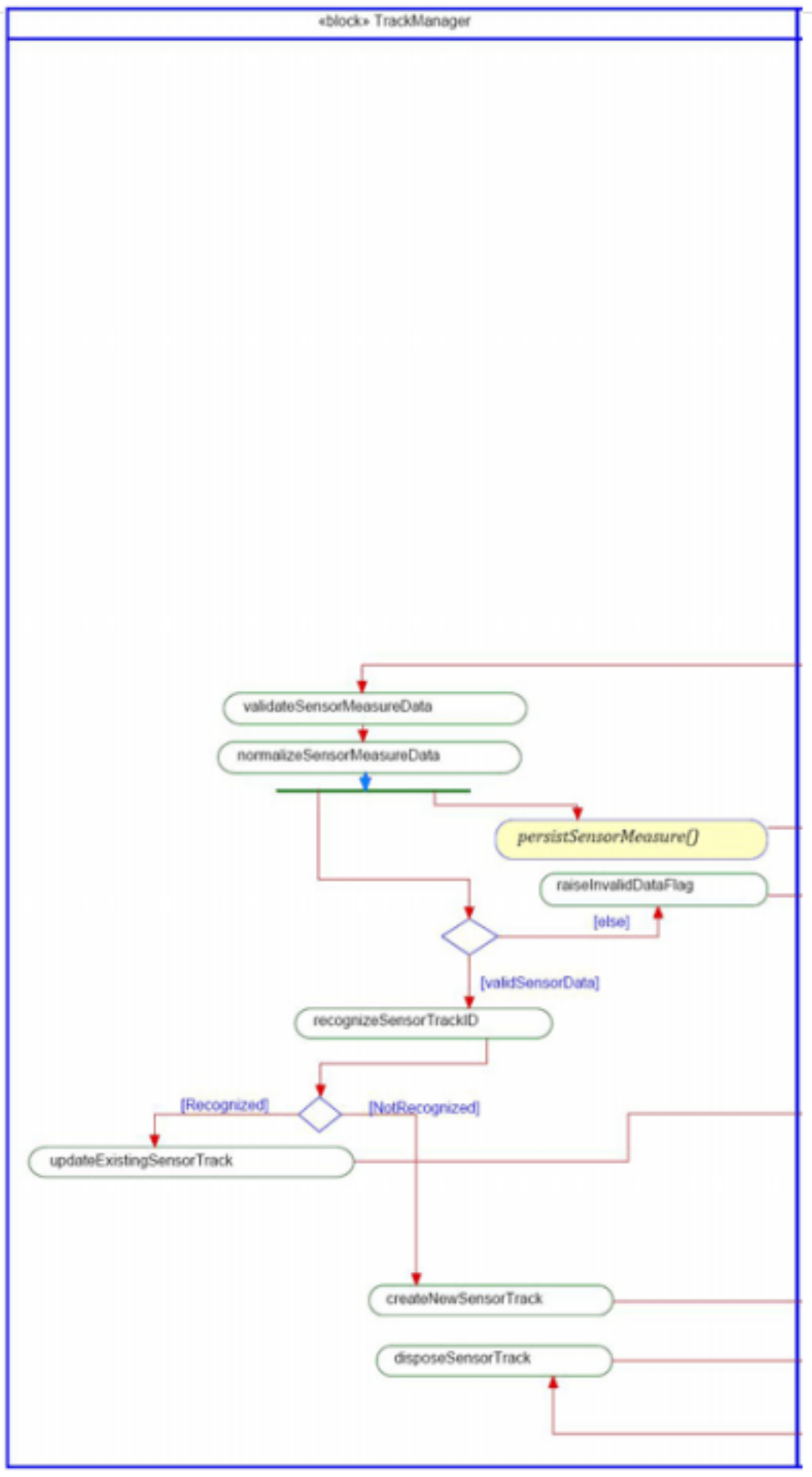

«block» TrackManager
validateSensorMeasureData
normalizeSensorMeasureData
persistSensorMeasure()
raiseInvalidDataFlag
[else]
[validSensorData]
recognizeSensorTrackID
[Recognized]
[NotRecognized]
updateExistingSensorTrack
createNewSensorTrack
disposeSensorTrack

图 7. 白盒活动图

火龙果·整理
uml.org.cn

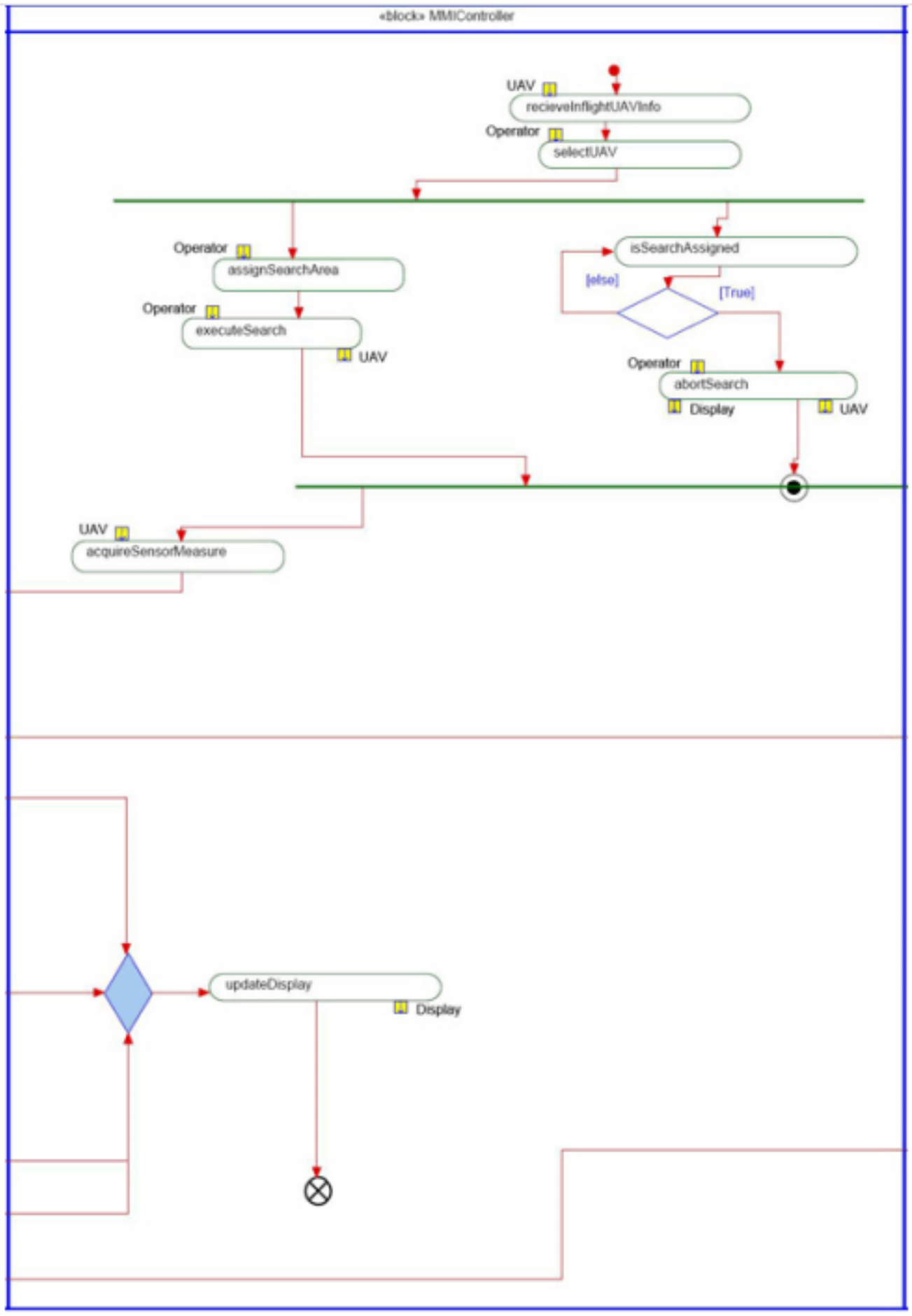
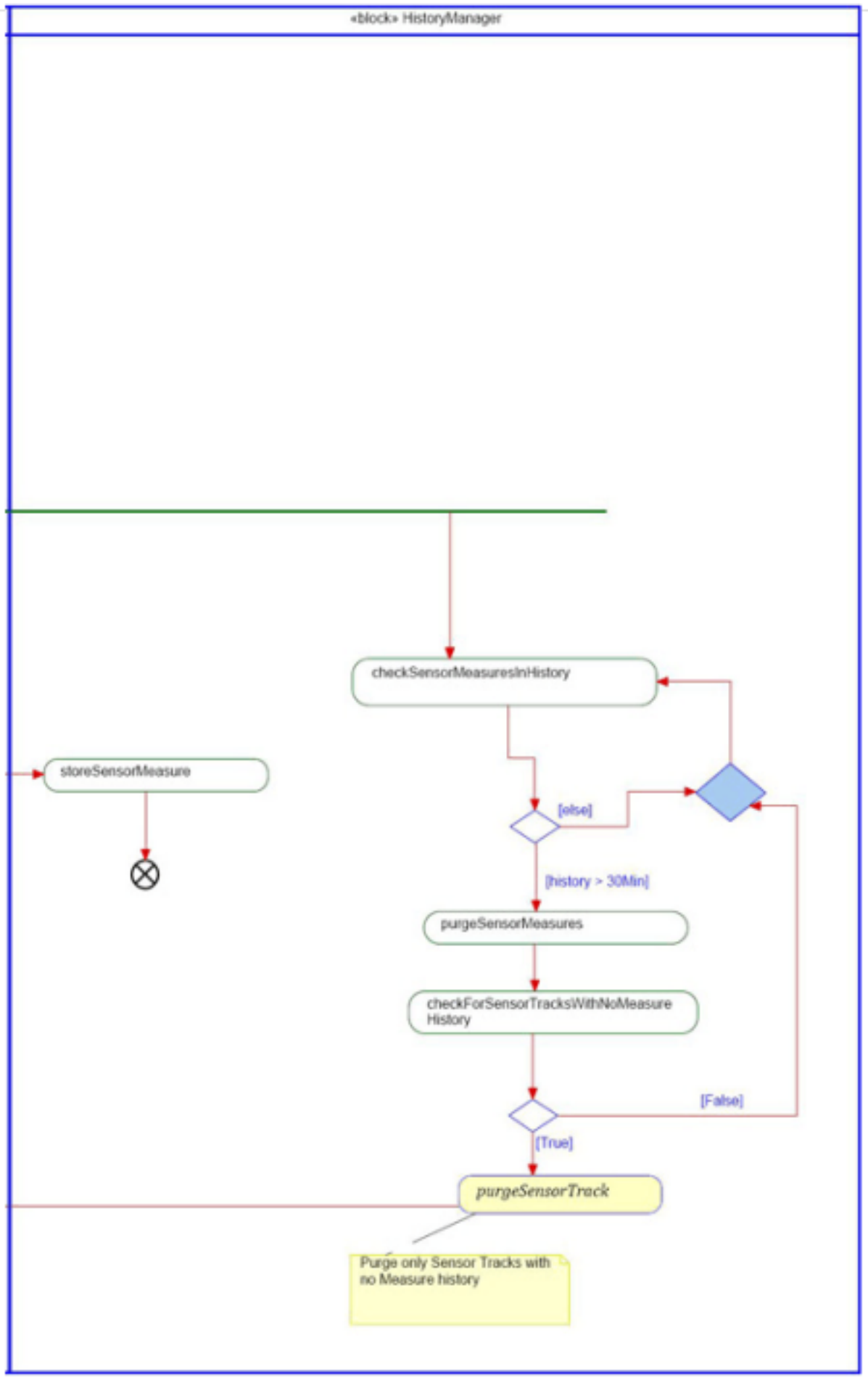


图 7. 白盒活动图



在图 7 中，与操作员交互相关的操作已包含在人机界面 (MMI) 控制器组件中。同样，与创建、更新和处理传感器追踪信息相关的操作被分配到 Track Manager 泳道。而与历史数据管理有关的操作都推送到 History Manager 泳道。在将连续流拆分成两个块的地方，可以利用消息操作来表示从一个块到另一个块的转发请求。这种模式的一个示例是，从 History Manager 组件到 Track Manager 组件的消息操作 purgeSensorTrack() ，该操作请求后一个组件执行 disposeSensorTrack() 。

现在，已将操作分配给泳道，下一步就是执行具体的架构设计。

具体的架构设计

在进行具体的架构设计阶段，需要重点关注端口和接口的定义，以及实现子系统块基于状态的行为。为了做到这一点，必须使用白盒序列图确定所子系统块的端口和接口。

黑盒活动图的重点是确定不同的系统功能（操作）流，而白盒活动图的重点则是不同子系统之间的协作，同时还要考虑到操作的分配。接收到服务请求定义一个块的接口。在定义了端口和接口后，必须将所产生的每个叶块基于状态的行为捕获到某个状态图中。

代理白盒序列图如图 8 所示。序列图显示一个子系统块为了满足场景而向另一个子系统块请求的服务。

图 8. 白盒序列图

火龙果·整理
uml.org.cn

图 8. 白盒序列图

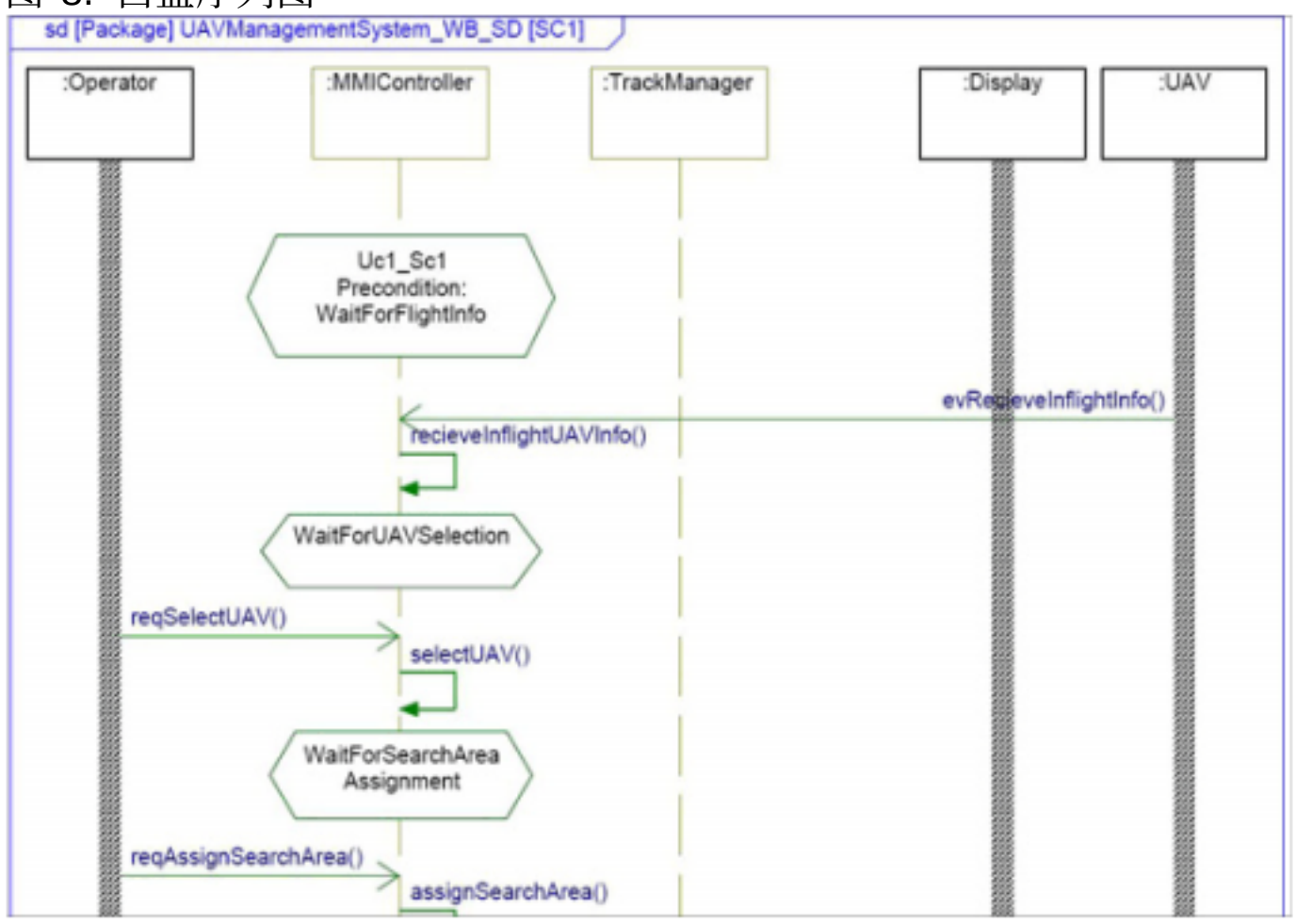


图 8. 白盒序列图

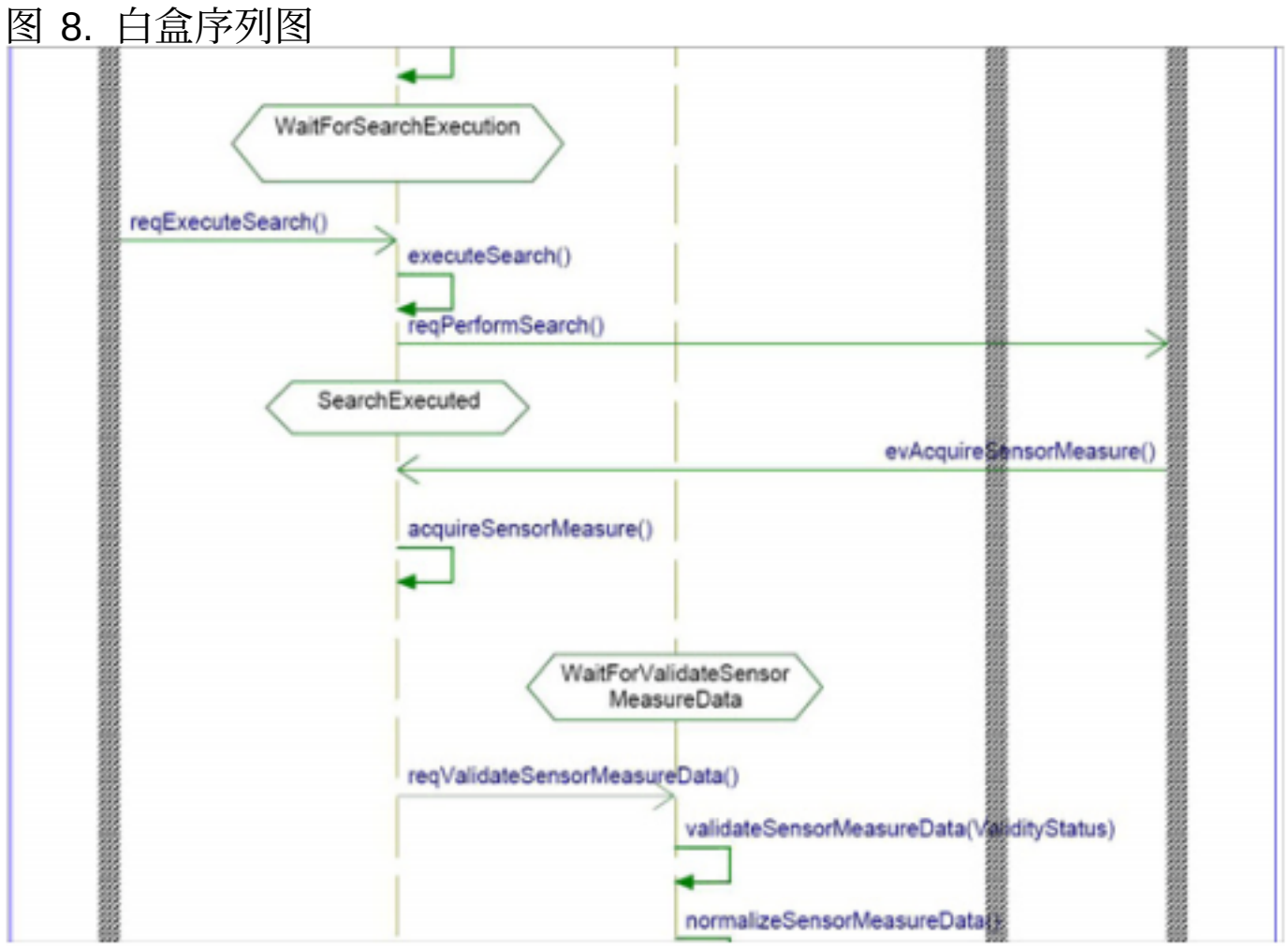


图 8 的大图

我们继续使用白盒序列图来获得子系统之间的端口和接口，并获得代理子系统组件基于状态的行为，如图 9、图 10 和图 11 所示。

火龙果 · 整理
uml.org.cn

图 9. MMI 控制器状态

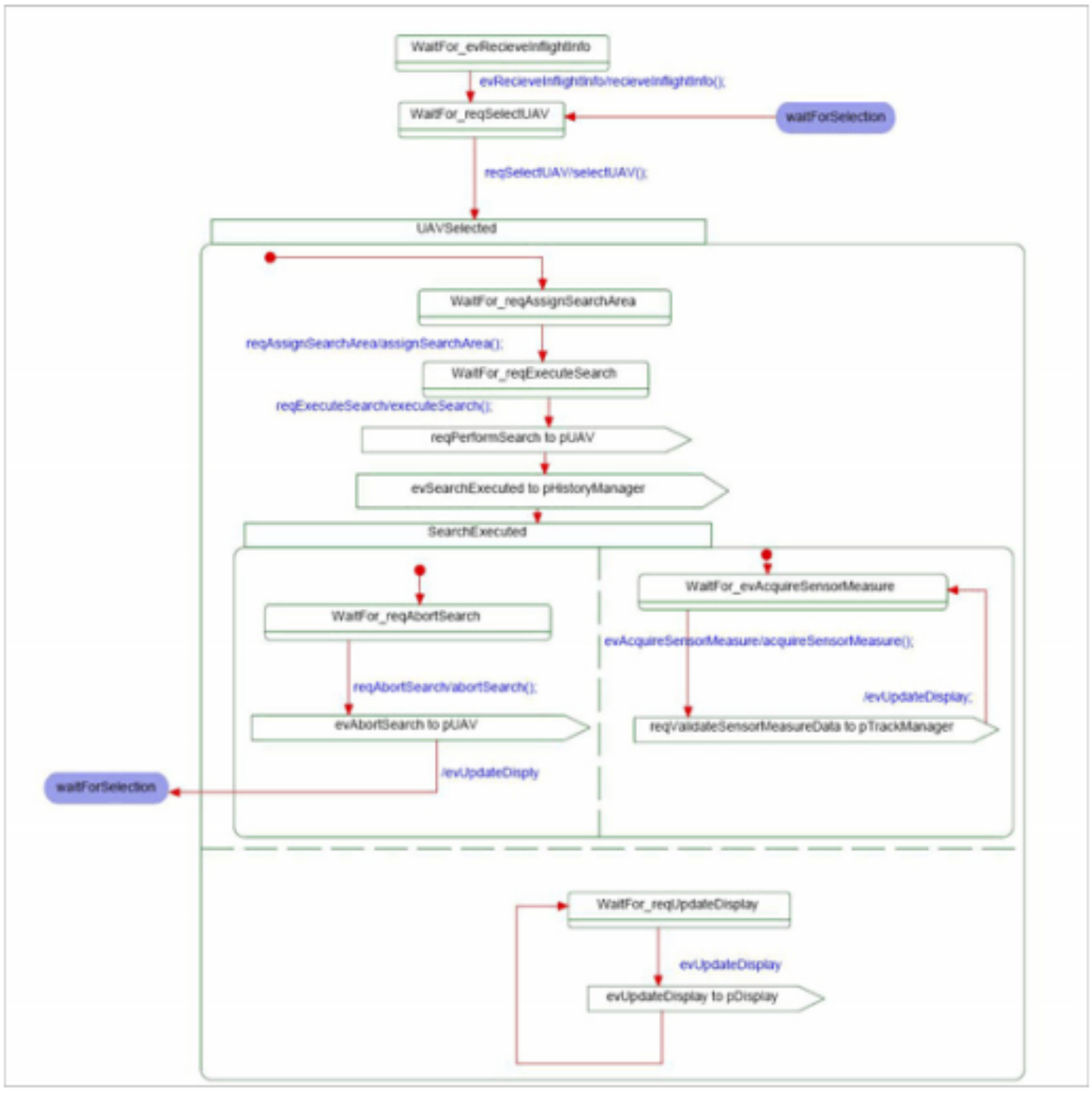


图 9 的大图

图 10. 追踪信息管理器的状态图

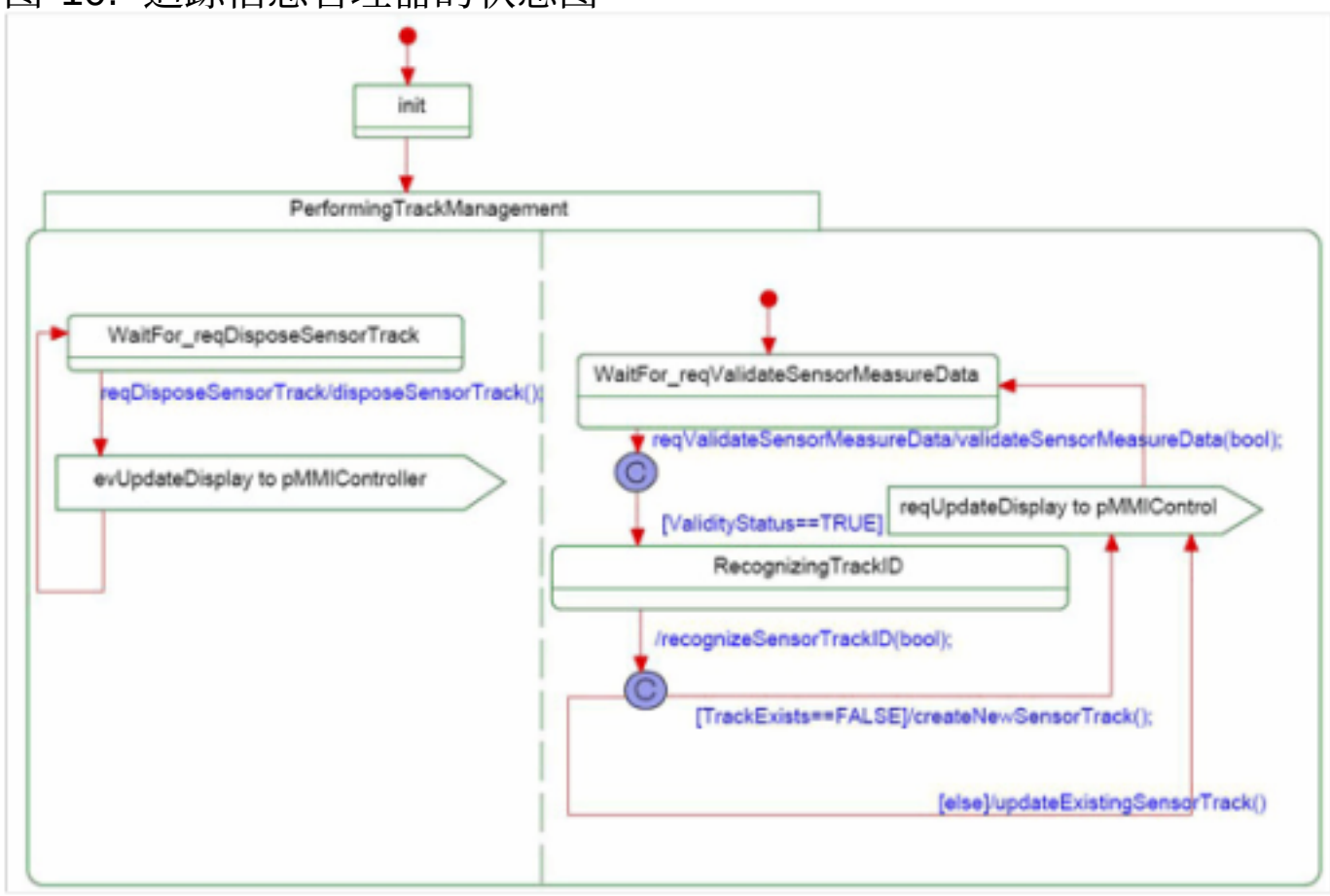


图 10 的大图

图 11. 白盒端口和接口

火龙果·整理
uml.org.cn

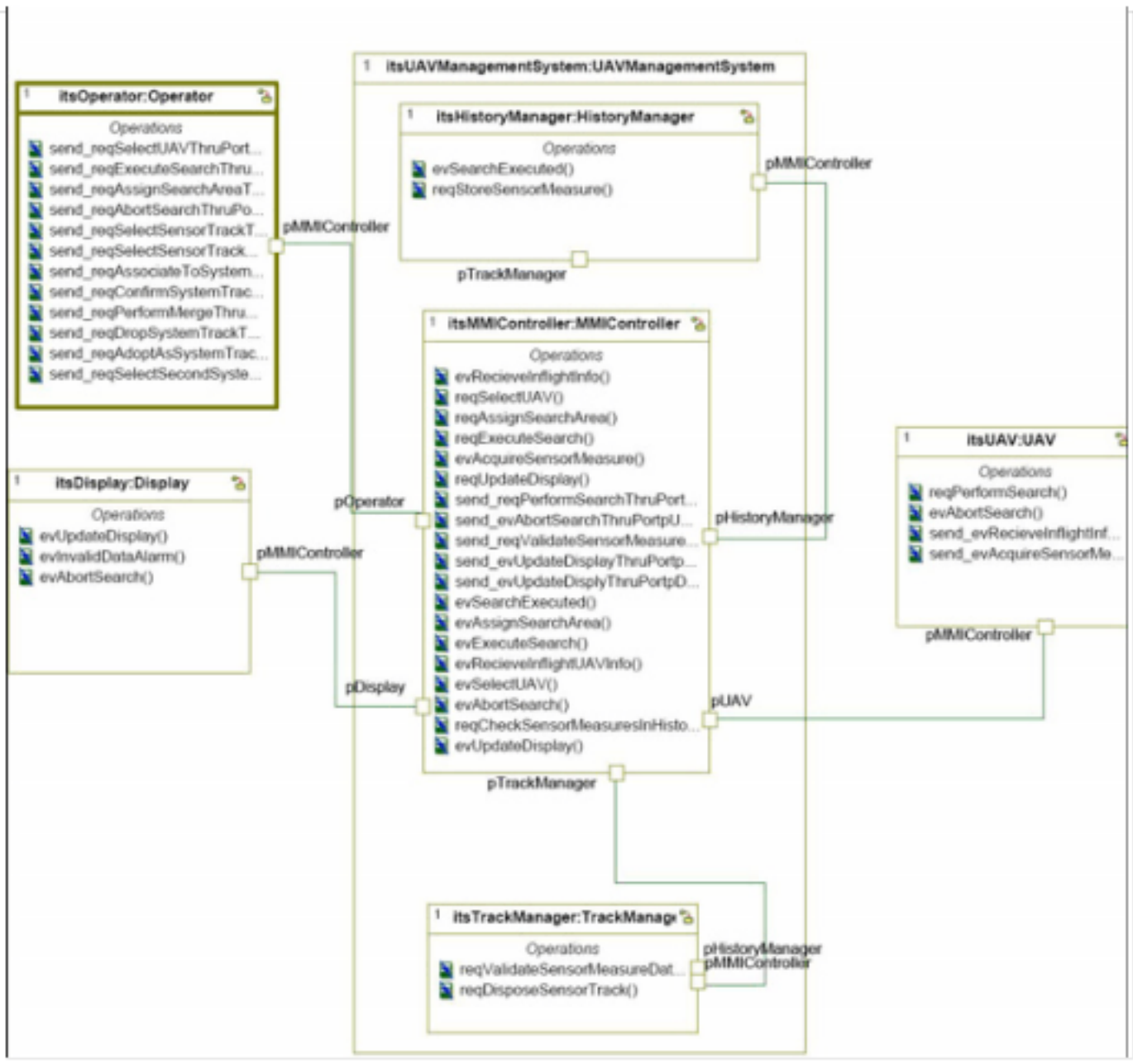


图 11 的大图

结束语

我们描述了如何通过案例研究来应用 Rational Harmony 系统工程流程。从系统工程切换到后续系统开发的关键构件是一个可执行基线模型。该模型是生成规范文档和操作 ICD 的资料库。切换包中包含下列项目：

可执行子系统模型的基线

子系统已分配的操作的定义

子系统端口和 逻辑 接口的定义

子系统行为的定义，捕获在状态图中

测试场景，从系统级用例场景中获得