体系结构 Assignment-2

131250207 丁霄汉
131250181 陈云龙
131250159 曾婧
131250129 梁思宇

一、 NFR

可修改性

Portion of scenario	possible values
source	开发者
stimulus	开发者修改系统用户界面、数据标准、控制逻辑等
artifact	代码
environment	设计时
response	需要修改的部分能被正确的修改，且不影响其他功能
response measure	每个模块的修改可以在 2 人月内完成 修改预算不超过总预算的 10% 不影响无关的系统功能

可移植性

Portion of scenario	possible values
source	开发者
stimulus	将系统从当前平台移植到另一平台
artifact	整个系统，地面部分或探测器部分
environment	开发时刻、部署时刻
response	系统或部分系统可以移植到新平台中并正确运行
response measure	移植工作量不超过 2 人月 修改代码量不超过 10% 移植后系统所有功能都可以正确运行

安全性

Portion of scenario	possible values
source	来自内部/外部的经过了授权/未经过授权的个人或系统
stimulus	试图修改/删除数据，访问系统服务，降低系统服务的可用性
artifact	系统服务、系统中的数据
environment	在线/离线，联网/断网，连接有防火墙或直接连接到了网络上
response	对用户进行验证；加密用户的账户信息；阻止未授权用户访问； 自动侦测攻击，受到攻击后通知管理员并锁死数据访问
response measure	未认证用户无法访问数据和发布控制指令 受到攻击后 1min 内将信息发送给管理员并锁死数据访问 数据被恶意修改/删除后可以在 10min 内进行恢复

可用性

Portion of scenario	possible values
source	系统内部、系统外部
stimulus	系统组件出现故障，出现行为异常或停止响应
Artifact	系统的处理器，通信通道，持久化存储器，进程
environment	运行时
response	系统检测到故障并记录，通知用户，修复故障
response measure	系统在 5min 内检测到故障源 出现故障后，系统可在 1h 内修复 系统应保证每周崩溃不超过 1 次

互操作性

Portion of scenario	possible values
source	系统内部/系统外部
stimulus	探测器与地面进行通信，地面上工作站之间进行通信
artifact	通信服务
environment	运行时刻
response	进行互操作的系统/组件进行可以正确进行通信
response measure	通信正确率 99.99%以上 通信失败或出错时可以在 5 秒钟发送信息给管理员

二、ADD 过程

第一次迭代

针对整个系统，效用树如下。

根据优先级，主要考虑的是可移植性、可修改性、安全性。

分解系统如图所示。探测器控制模块、探测器通信模块部署在探测器端，地面通信服务、地面控制逻辑、图像处理模块、图像存储模块部署在地面，探测器端和地面端都包含数据标准模块和安全性保障服务，并依赖通信服务保持更新。

探测器控制模块：控制探测器根据地面指令完成拍照任务，生成并回传图像。

探测器通信服务：与地面进行通信，接收指令和发送图像。

数据标准模块：管理通信数据格式和图像格式标准。

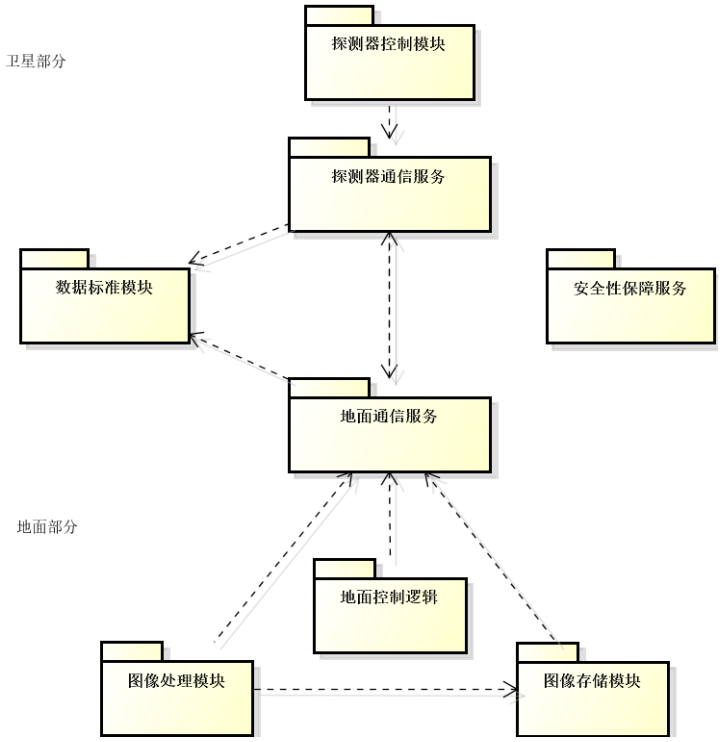
安全性保障服务：保证系统安全性。

地面通信服务：与探测器进行通信，发送指令和接收图像。

地面控制逻辑：通过通信模块发送指令控制探测器上的相机执行任务。

图像存储模块：通过通信模块接收图像，存储在服务器上，允许授权的用户访问。

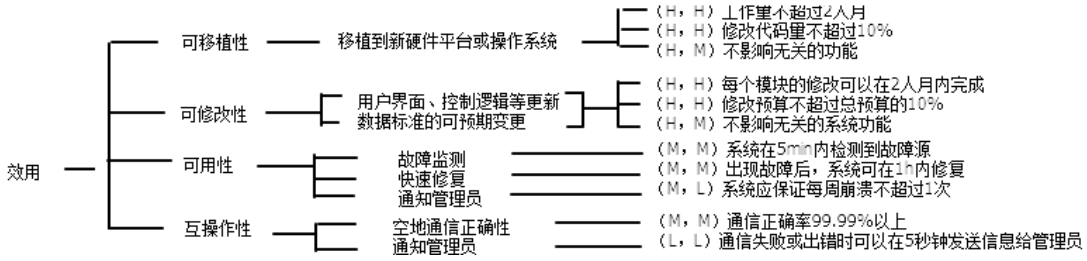
图像处理模块：处理存储的图像，或通过通信模块对原始图像进行即时处理。



第二次迭代

选择元素和 ASR

第二次迭代选择的元素是探测器控制模块。针对这一元素的效用树如下。



对于该系统，硬件和操作系统的升级是可预见的，图像格式和数据标准也是存在潜在变化的。而且探测器发射后对软件系统的修改显然只能远程进行，可修改性显得尤为重要。

因此，第二次迭代选择的 ASR 是可修改性、可移植性。

候选策略表和决策理由

模块分离	采用。分离出数据标准模块，维护通信协议的标准不再由通信标准承担，将可能发生变更的部分独立出来，以应对数据标准的潜在变更。
增强内聚	采用。将底层支持模块分成硬件驱动和操作系统支持模块，前者提供相机的硬件驱动支持，以应对硬件升级需要。后者为系统提供探测器操作系统的接口，实现和探测器其他部分的交互，以应对操作系统变更。
延迟绑定	采用。具体措施同“模块分离”。分离出数据标准模块，专门维护数据标

	准，而非将其写死在系统中，本身也是延迟绑定的措施。
重构	未采用。重构更侧重于维护可变更性的方法，而非体系结构设计决策。
封装	未采用。在体系结构的层面上体现不明显。
使用中间件	未采用。容易造成性能损失。

第二次迭代结果

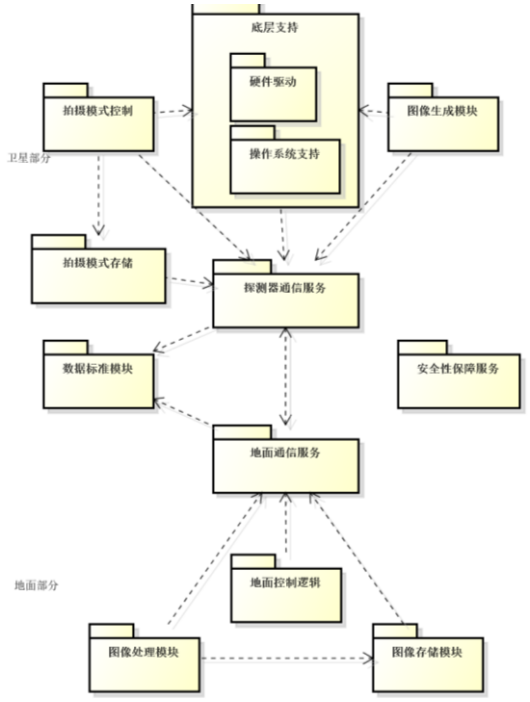
如图所示，新增元素说明如下。

底层支持：提供相机的硬件驱动，并与探测器操作系统交互，依赖通信服务来升级硬件驱动和操作系统。

拍摄模式存储：存储经过定义的拍摄模式，依赖通信服务来增删用户定义的拍摄模式。

图像生成模块：通过相机的传感器读数生成原始图像，依赖底层支持来获取硬件读数。

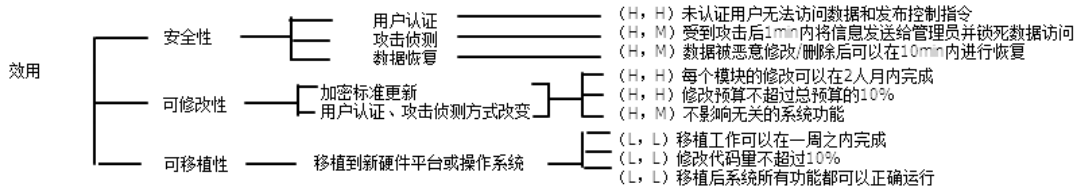
拍摄模式控制：按照存储的拍摄模式和接收的拍摄指令，控制拍摄过程。



第三次迭代

选择元素和 ASR

第三次迭代选择的元素是安全性保障服务。针对这一元素的效用树如下。



针对这一元素，可修改性主要指的是加密标准更新和用户认证、攻击侦测方式的改变。由于此元素跟操作系统和硬件不直接相关，所以可移植性不重要。

一方面，涉及航天工业，硬件昂贵，数据价值高，安全性不容忽视。另一方面，安全措施更新换代快，加上探测器的系统升级只能远程进行，要求可修改性高。

因此，选择的 ASR 是安全性和可修改性。

候选策略表和决策

用户认证	采用。由于系统可能涉及机密信息，对用户的身份认证是必须的。在地面端增加用户认证模块，以加密方式存储用户资料，用户向探测器发布控制指令必须经过身份认证。
数据加密	采用。由于地面和探测器的通信可能被拦截和监听，加密是必须的。
攻击发生时收回数据访问权限	采用。增加攻击侦测模块，当侦测到攻击发生时，图像存储模块将会拒绝一切外界访问，直到确认安全。
通过检验和或哈希值验证数据完整性	未采用。假定在地面和探测器的通信过程中已经采取了完整性验证手段，这种假设在远程通信中是非常合理的。
拒绝可疑访问	未采用。由实际系统特性决定，本系统用户是少数专业用户，而非面向大众的网络服务。
模块分离	采用。提供安全保障的元素模块化，隔离潜在变更。

第三次迭代结果

如图所示，新增元素说明如下。

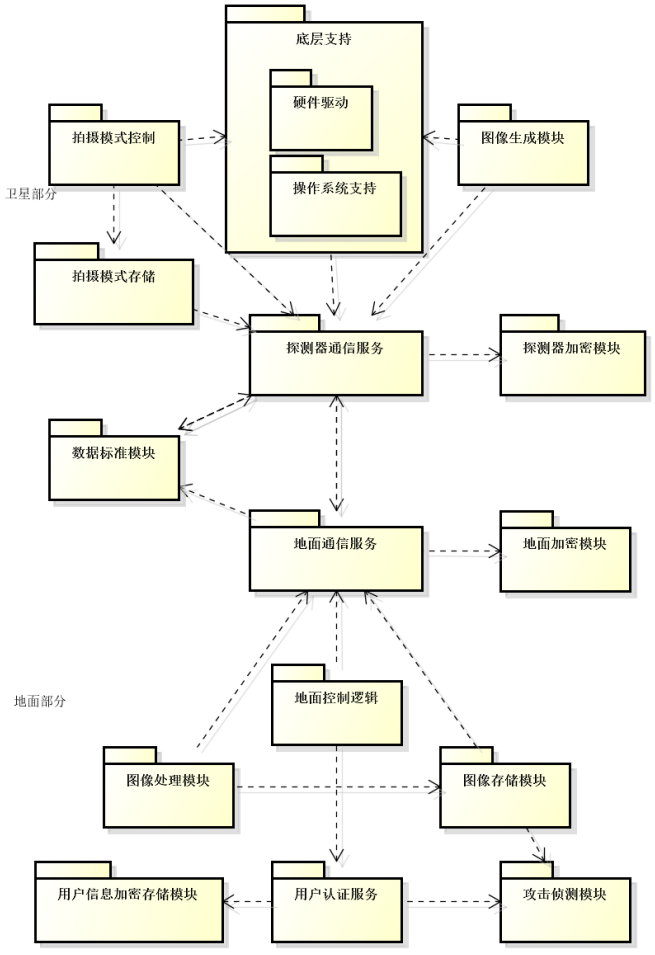
探测器加密模块：加密回传图像数据，解密控制指令。

地面加密模块：加密控制指令，解密图像数据。

用户认证服务：确认用户身份，认证权限。

用户信息加密存储模块：存储加密后的用户信息。

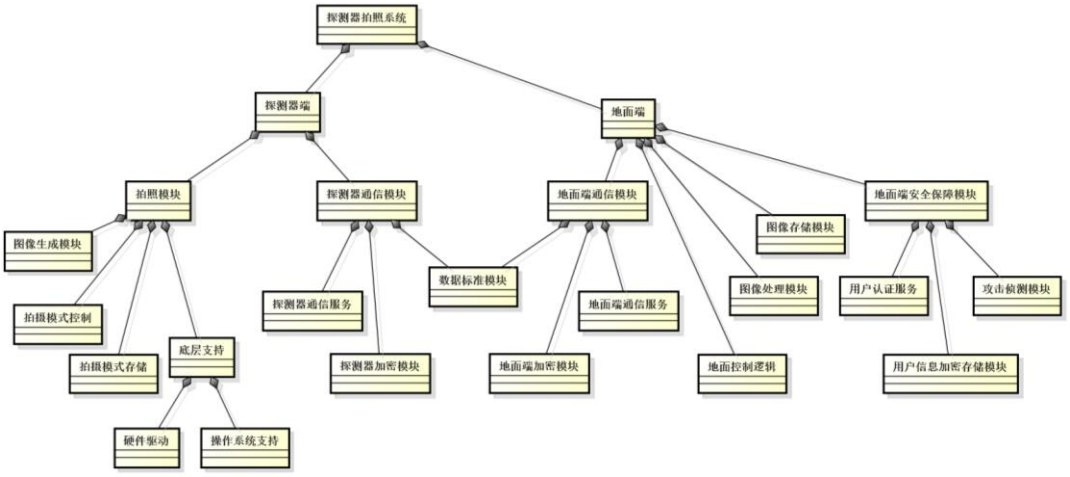
攻击侦测模块：侦测非正常访问，在攻击发生时禁止外界访问图像存储模块。



三、体系结构设计文档

1. 模块视角

1.1 视图的主要表示(分解视图)



1.2 元素目录

在以上分解视图中，根节点探测器拍摄系统为本次架构目标系统，分为探测器端和地面端两部分。

探测器端分为拍照模块和通信模块。拍照模块实现拍照功能，具体有图像生成模块、拍摄模式控制、拍摄模式存储、底层支持。通信模块主要负责和地面端的通信，将图片发送给地面端。通信模块采用统一的数据标准，提高可修改性；还拥有探测器端加密模块，提高安全性。

地面端分为地面端通信模块、地面端安全保障模块、地面控制逻辑、图像存储模块、图像处理模块。控制模块根据用户需要生成控制指令。地面端通信模块负责接收探测器端的图片，采用统一的数据标准，增强可修改性；还拥有地面端加密模块，提高安全性。地面端安全保障模块包括用户认证服务，用户信息加密存储，攻击侦测模块，用来防止系统被恶意攻击或修改。

各元素详细说明如下：

探测器控制模块：控制探测器根据地面指令完成拍照任务，生成并回传图像。

探测器通信服务：与地面进行通信，接收指令和发送图像。

数据标准模块：管理通信数据格式和图像格式标准。

安全性保障服务：保证系统安全性。

地面通信服务：与探测器进行通信，发送指令和接收图像。

地面控制逻辑：通过通信模块发送指令控制探测器上的相机执行任务。

图像存储模块：通过通信模块接收图像，存储在服务器上，允许授权的用户访问。

图像处理模块：处理存储的图像，或通过通信模块对原始图像进行即时处理。

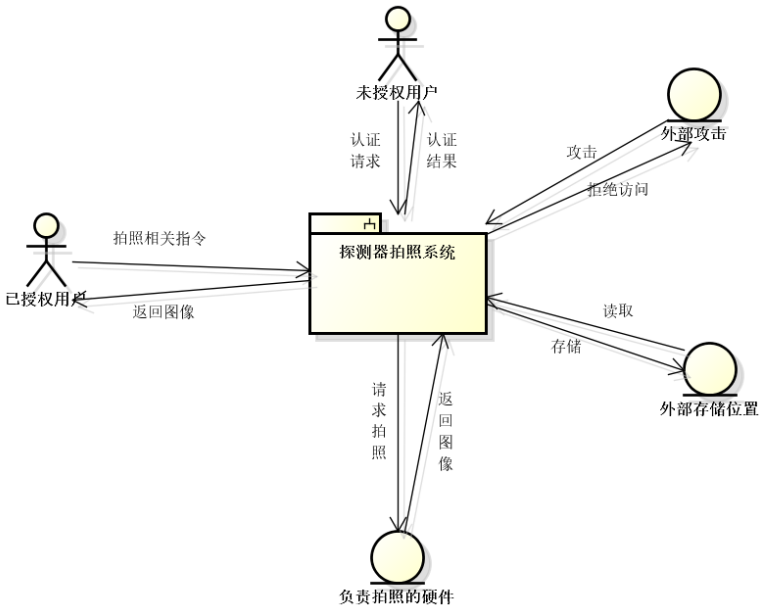
底层支持：提供相机的硬件驱动，并与探测器操作系统交互，依赖通信服务来升级硬件驱动和操作系统。

拍摄模式存储：存储经过定义的拍摄模式，依赖通信服务来增删用户定义的拍摄模式。

拍摄模式控制：按照存储的拍摄模式和接收的拍摄指令，控制拍摄过程。

图像生成模块：通过相机的传感器读数生成原始图像，依赖底层支持来获取硬件读数。
探测器加密模块：加密回传图像数据，解密控制指令。
地面加密模块：加密控制指令，解密图像数据。
用户认证服务：确认用户身份，认证权限。
用户信息加密存储模块：存储加密后的用户信息。
攻击侦测模块：侦测非正常访问，在攻击发生时禁止外界访问图像存储模块。

1.3 上下文图



1.4 变更指南

1.4.1 硬件和操作系统变更

将底层支持模块分成硬件驱动和操作系统支持模块，前者提供相机的硬件驱动支持，以应对硬件升级需要。后者为系统提供探测器操作系统的接口，实现和探测器其他部分的交互，以应对操作系统变更。

1.4.2 数据标准的变更

分离出数据标准模块，维护通信协议的职责由该模块承担，将可能发生变更的部分独立出来，以应对数据标准的潜在变更。

1.4.3 控制逻辑、存储介质、图像处理算法等可能的变更

分别有地面控制逻辑、拍摄模式控制，图像存储模块、拍摄模式存储，图像处理模块等与之对应，可以方便的查找并修改对应部分，降低变更代价。

1.5 合理性说明

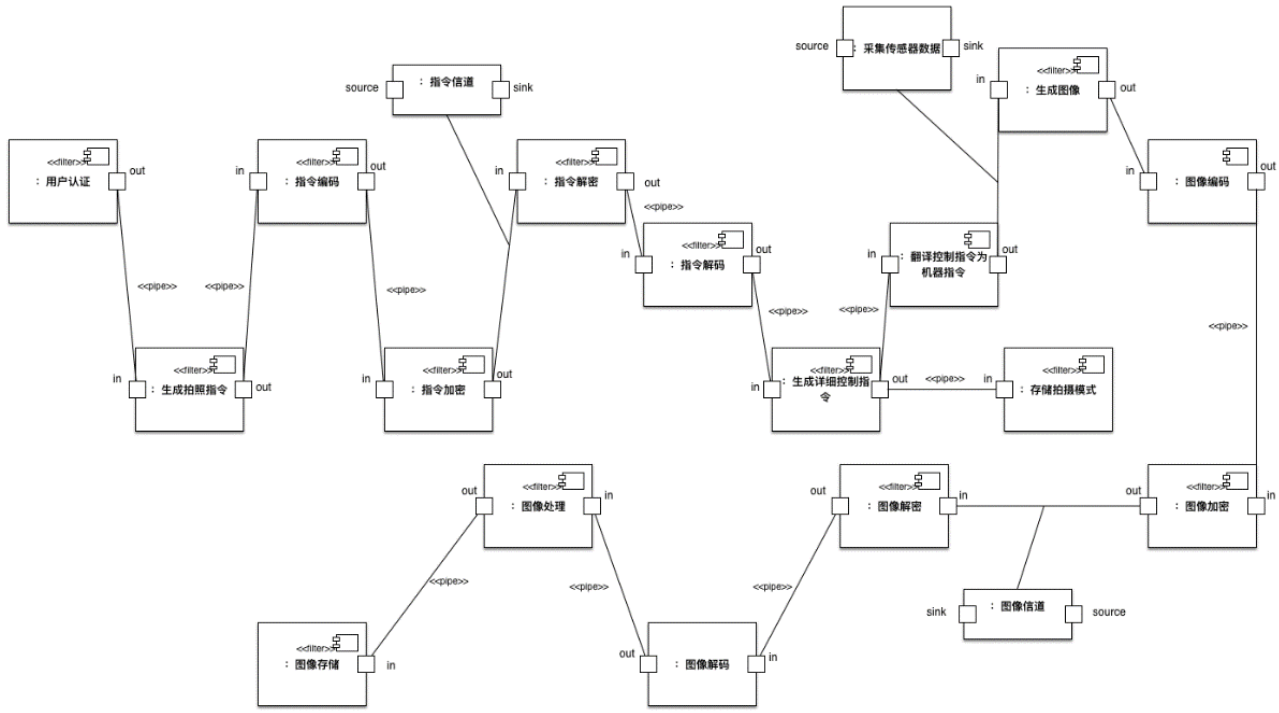
作为探测器拍照系统，由于系统的硬件和操作系统升级可以预期，图像格式和数据标准的变化也时有发生，鉴于系统本身的特殊性，修改代价昂贵，使得可修改性、可移植性显得极其重要。为了应对可修改性、可移植性的要求，系统应该秉承模块化设计的原则，采用模块分离的策略，将可能发生变更的部分独立出来，方便日后的修改。

作为航天工业的重要系统，本身硬件昂贵，数据价值高，安全性不容忽视。因此采用了用户认证机制，并加密存储用户资料，防止未授权用户对系统的恶意修改。同时采取数据加密策略，防止信息被拦截或监听。另外增加攻击侦测模块，保证图像信息的安全。

因此，制定分解视图，可以很好的体现系统的模块化结构和逐步求精的设计思路，体现模块职责的分配，为涉众提供详细指引。

2. C&C 视角

2.1 视图的主要表示（管道过滤器视图）



2.2 元素目录

视图中采用管道和过滤器的形式描述了系统中组件和连接件的连接情况。

过滤器：处理数据。将输入的数据经过处理后输出给下一个或者多个过滤器。

管道：进行单向的数据传输或者通信，保护命令和数据

过滤器和管道之间相互连接，不能有两个管道直接相连，也不能有两个过滤器直接相连。

元素详细说明如下：

用户认证：处理用户登录的用户名和密码，输出验证结果，记录用户状态。

生成拍照指令：根据用户操作生成具体的拍照指令。

指令编码：根据通信协议编码指令数据。

指令加密：根据加密标准加密编码后的指令数据。

指令信道：通过空地通信，加密后的指令数据经过此管道发送给探测器。

指令解密：根据加密标准解密指令数据。

指令解码：根据通信协议解码指令数据。

生成详细控制指令：根据拍照指令，控制逻辑根据探测器和相机状态生成一系列具体的控制指令，如改变镜头朝向、设置曝光时间、选择电磁波波段等。

存储拍摄模式：存储此拍摄模式，以后需要重复拍摄任务时可以简化操作。

翻译控制指令为机器指令：利用底层支持模块，将每一条控制指令翻译为机器指令，驱动硬件系统工作。

采集传感器数据：利用底层支持模块，系统通过此管道采集传感器数据。

生成图像：处理传感器数据，生成原始图像。

图像编码：根据通信协议编码图像数据。

图像加密：根据加密标准加密编码后的图像。

图像信道：通过空地通信，加密后的图像数据通过此管道发送给地面。

图像解密：根据加密标准解密图像数据。

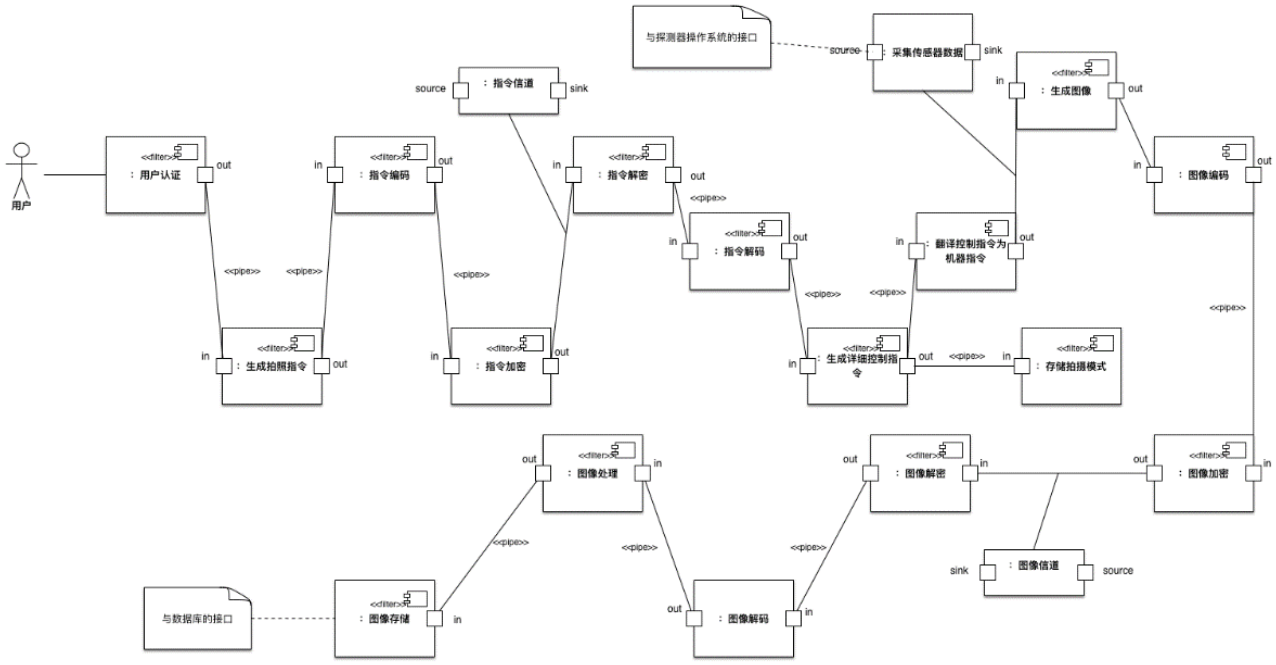
图像解码：根据通信协议还原图像。

图像处理：处理原始图像。

图像存储：存储经过处理的图像。

2.3 上下文图

展示哪些组件和连接器通过哪些接口和协议与外部的组件和连接器相连



2.4 可变性指南

2.4.1 通信协议变更

修改指令编码、指令解码、图像编码、图像解码过滤器即可。

2.4.2 加密标准变更

修改指令加密、指令解密、图像加密、图像解密过滤器即可。

2.4.3 其他变更

找到与之对应的管道或过滤器进行修改即可。

2.5 基本原理

采用管道过滤器视图，是由于可移植性和可修改性的要求，这个系统要被架构成一个松耦合的系统。例如，当在工作站的工作人员向探测器发送拍摄指令的时候，工作人员无需关心探测器是如何解析指令并进行拍照的；当工作站的工作人员对探测器拍摄的照片进行处理的时候，探测器也无需再关注处理照片的过程，也无需关注工作人员对照片的存储。

从工作人员发送的指令，到编码、加密、发送、解密、解码，再到探测器解析命令、拍摄、生成图像、再对图像编码、加密、发送、解密、解码，再到处理照片、存储照片，都是在不断地处理数据、传输数据。在这个过程中，管道负责数据的传递，它把原始数据传递给第一个过滤器，把一个过滤器的输出传递给下一个过滤器，作为下一个过滤器的输入，重复这个过程直到处理结束。

采用管道过滤器视图，使得每一个过滤器只需要实现单一的功能，从而降低了系统的复杂程度，使得过滤器之间的依赖最小，可以以更加灵活的组合来增加、实现新的功能或替换原有的功能。

3. Cross-View

3.1 系统概览

本探测器相机拍照系统分为地面部分和探测器部分。探测器部分负责接收地面指令，利用探测器操作系统和相机硬件执行拍照任务，读取传感器读数，生成图像并发送回地面。地面向探测器发出指令，接收并处理、存储图像。两部分依赖通信服务进行空地通信。

一方面，由于航天系统资源宝贵，本系统重视安全性，在通信过程中使用加密和解密技术，对用户进行身份验证，并在侦测到攻击后锁死图像存储模块以防止数据外泄。

另一方面，由于系统开发周期、成本均有严格要求，硬件系统、操作系统的升级可以预期，数据标准的变化也难以避免，而且探测器端系统升级只能远程进行，所以本系统强调可移植性和可修改性，主要采取模块分离的方式，以隔离潜在变更，降低变更代价。

3.2 视图之间的映射

本文档使用分解视图和管道过滤器视图，主要映射关系如下：

分解视图	**管道过滤器视图**
硬件驱动	翻译控制指令为机器指令
操作系统支持	采集传感器数据
拍摄模式存储	存储拍摄模式
拍摄模式控制	生成详细控制指令
图像生成模块	生成图像
探测器加密模块	指令解密、图像加密
探测器通信服务	指令信道、图像信道
数据标准模块	指令编码、指令解码、图像编码、图像解码
地面通信服务	指令信道、图像信道
地面加密模块	指令加密、图像解密
地面控制逻辑	生成拍照指令
图像处理模块	图像处理
图像存储模块	图像存储
用户认证服务	用户认证

用户信息加密存储模块	用户认证
攻击侦测模块	用户认证

3.3 合理性说明

由于本系统对可移植性、可修改性的高要求，设计时秉承模块化设计原则，采用逐步求精思想，模块与职责的对应、数据的流动过程清晰明确。

因此，对于数据流动中发生的传递和变化，一方面能够容易地找到与之对应的过滤器和管道，一方面能够容易地找到与之对应的系统职责进而找到与之相关的模块，也就能清晰地得出管道过滤器和模块之间的变化，所以说此 cross-view 是自然的、有意义的。

四、个人部分

131250207 丁霄汉

ADD 过程：

在本次团队作业中，我们从系统描述开始提炼 NFR，结合系统实际和生活常识筛选鉴别出其中重要的 ASR，进行了三次迭代过程。第一次迭代，考虑系统功能，构建系统框架，粗略分配各部分职责。第二次迭代，选择探测器控制模块作为目标元素，考虑可移植性和可修改性，按照模块分离的原则，隔离潜在变更，提高可移植性和可修改性。第三次迭代，选择安全性保障服务作为目标元素，考虑安全性和可修改性，参考上课讲过的 Quality Attributes 一章中关于安全性的策略，添加了诸多负责安全性的系统部件。在三次迭代过程中，始终秉承高内聚、低耦合、模块化的原则，保证设计质量。

实践经验：

根据系统要求对 NFR 确定优先级，作决策时优先满足其中高优先级的 NFR。

以常用策略为基础，逐个分析，结合系统实际选择最优决策。

采取逐步求精方法，每次迭代保持注意力集中在某一特定元素上。

个人贡献：

作为团队负责人，召集会议，主持讨论，总结记录讨论成果，分配工作，制定时间表并控制进度。

完成 ADD 三次迭代过程并进行文档化。

汇总团队工作成果，负责最终报告的编制、排版、检查，修正个别疏漏，统一风格，控制质量，督促修改返工。

131250181 陈云龙

在这次作业中，小组进行了激烈的讨论，我也从中学习了如何选择 ASR，巩加深对 ADD 方法的了解。我负责编写全部的非功能需求列表，这些非功能需求是在小组讨论中总结出来的，因此理解了架构质量属性，质量属性高于系统的性能，在设计中有着举足轻重的作用，它可以来评判一个系统的好坏。学会了用质量属性场景来描述质量属性，质量属性场景在质量属性需求规范中的作用与用例在功能需求规范中所扮演的角色相同。

除此之外，我对 ADD 方法也有所了解， ADD 是一种定义软件架构的方法，该方法将分解过程建立在软件必须满足的质量属性之上。它是一个递归分解的过程，其中在每个阶段都选择战术和架构模式来满足一组质量属性场景，然后对功能进行分配，以实例化由该模式所提供的模块类型。ADD 的输入时一组需求。ADD 把功能需求个限制作为输入。在使用 ADD 方法时，首先要分解系统，对分解的系统逐步求精【软件架构实践第二版第七章】。这次作

业我们分为三个迭代，第一个迭代是将系统分为不同的模块，第二、三次迭代我们讨论选择ASR，为 ASR 建立候选策略表，选择合理的策略。

131250159 曾婧

个人使用 ADD 方法的经验：

首先分析了项目中提出的需求，然后探讨项目中涉及的质量属性，并把质量需求表示为一组特定于系统的质量场景。因为 ADD 是一个迭代的过程，在一开始制定好模块之后，在第一次的迭代中先分解整个系统。在这次项目中一共有三个迭代，在第二、第三个迭代中，再细分不同的模块，逐渐细化。

在使用 ADD 方法的时候，还要确定不同功能模块的优先级，优先满足更重要的需求。依据能实现质量属性的策略选择合适的架构模式，并确定子模块，要求要满足驱动因素。还要实例化模块，定义包括服务、交互模式等的子模块接口。接着验证是否满足实例与场景用例。当验证无误，并且得知质量属性将会受到怎样的影响之后，完成迭代一，进入迭代二，选择子模块再进行分解。直到得到该项目的模块分解视图或其他所需视图的最初几个层次，为接下来的归档做好准备工作。

团队工作的个人贡献：

一起分析作业中提出的需求，制定大致模块，将系统分成了天空中的探测器和地上的工作站，由通信模块连接。共同选择 ASR，分析如何将 ADD 方法应用到项目中，如何应用。在文档中，我负责的是部分的视图归档，经过详细分析和与小组其他成员讨论后，我选择了组件－连接器视图，并采用 UML 将视图的主要部分以管道过滤器呈现。

131250129 梁思宇

ADD 方法是一个递归的分解过程，在每个阶段都选择策略和构架模式来满足一组质量属性场景。我们先进行了需求分析，然后运用 ADD 方法定义软件架构。ADD 的结果是构架的模块分解视图和其它视图的最初几个层次，因此我们最终得到的成果是粗粒度的，然而 ADD 方法对实现期望的质量属性来说还是非常重要的。

在使用 ADD 方法的过程中，首先将功能需求和约束作为输入，并把质量需求表示为一组特定于系统的场景。开始 ADD 方法之后，首先选择要分解的系统，接着对分解模块进行求精，然后对需要进一步分解的每个模块重复以上两个步骤。在实验中我们的工作分为了三个迭代，在第一个迭代中，我们将模块分解成子模块，在第二、三迭代中，根据优先级选择了重要的 ASR，为 ASR 建立候选策略表，并从中选择出最合理的策略。选择策略的过程中，既要考虑构架驱动因素本身，又要考虑实施这个策略对其他质量属性的影响，权衡选择。三个迭代完成后，进行验证，看是否通过分解满足了所有质量属性场景。如果质量属性场景不满意当前的分解，就还需要再分解，直到所有的质量属性场景都得到满足。

在这次的小组作业中，我作为成员参加了每一次的小组讨论，与其他成员一起进行需求分析，探讨 ASR 的选择，加深对 ADD 方法的理解，彼此协商交流意见。在最终的报告中，我负责的是视图归档部分，我选择了合适的 UML 图做了有关模块视图的分析并将内容组织为文档。