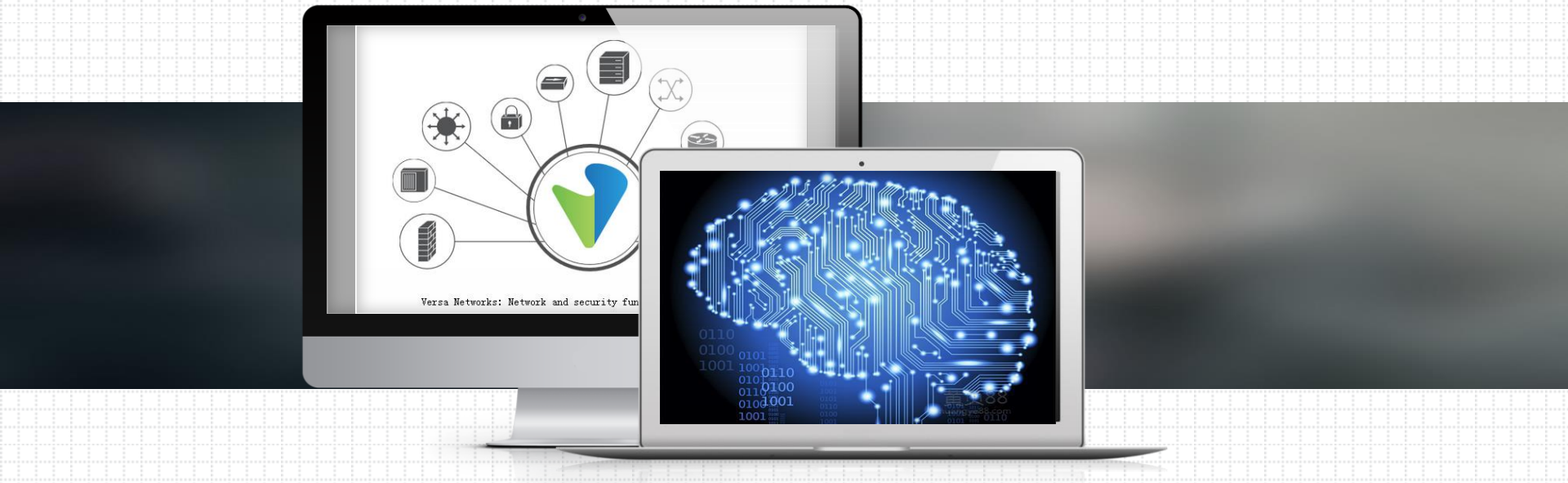
软件架构与设计模式

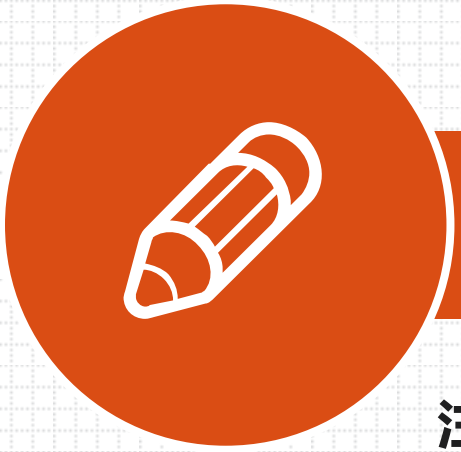
Software Architecture and Design Patterns



吴映波 副教授

Email：wyb@cqu.edu.cn

Office: 虎溪软件学院楼410



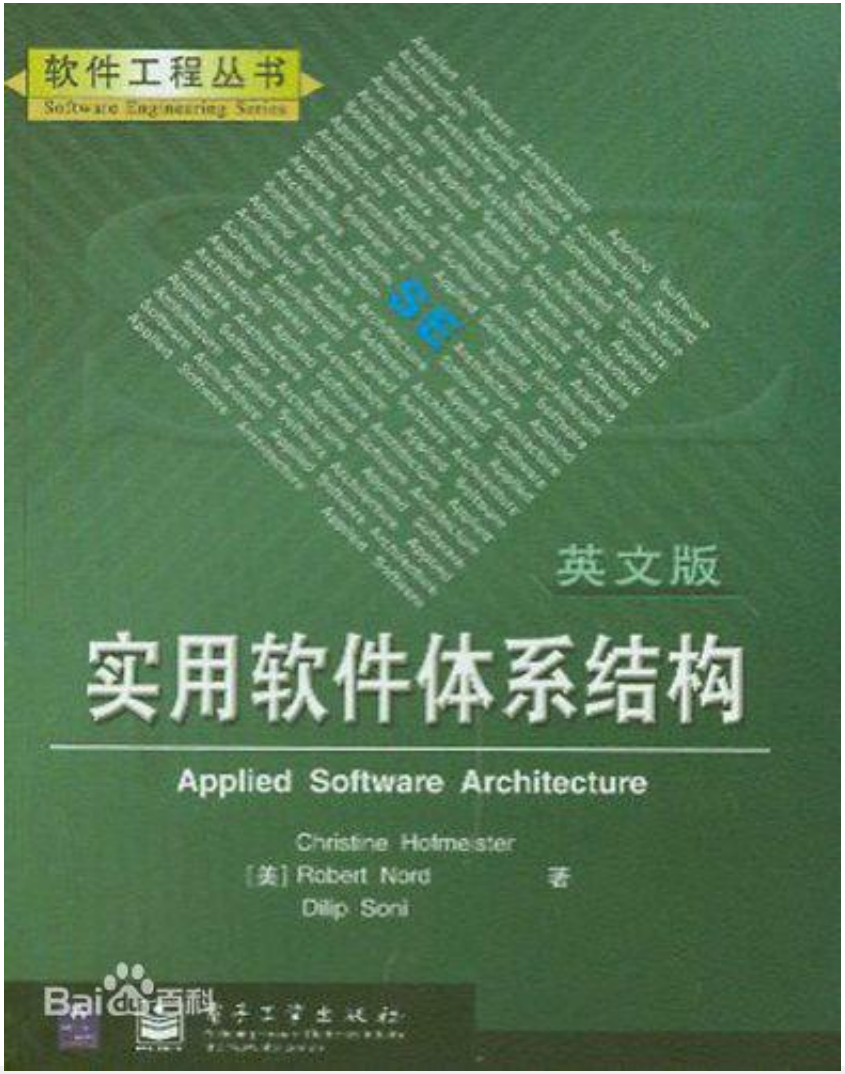
4. 课程案例： IS2000软件体系结构的分析与设计过程

“ PASSION LEADS TO DESIGN, DESIGN LEADS TO PERFORMANCE, PERFORMANCE LEADS TO **SUCCESS**! ”

注：此部分内容不作课程考核要求，仅提供参考学习！

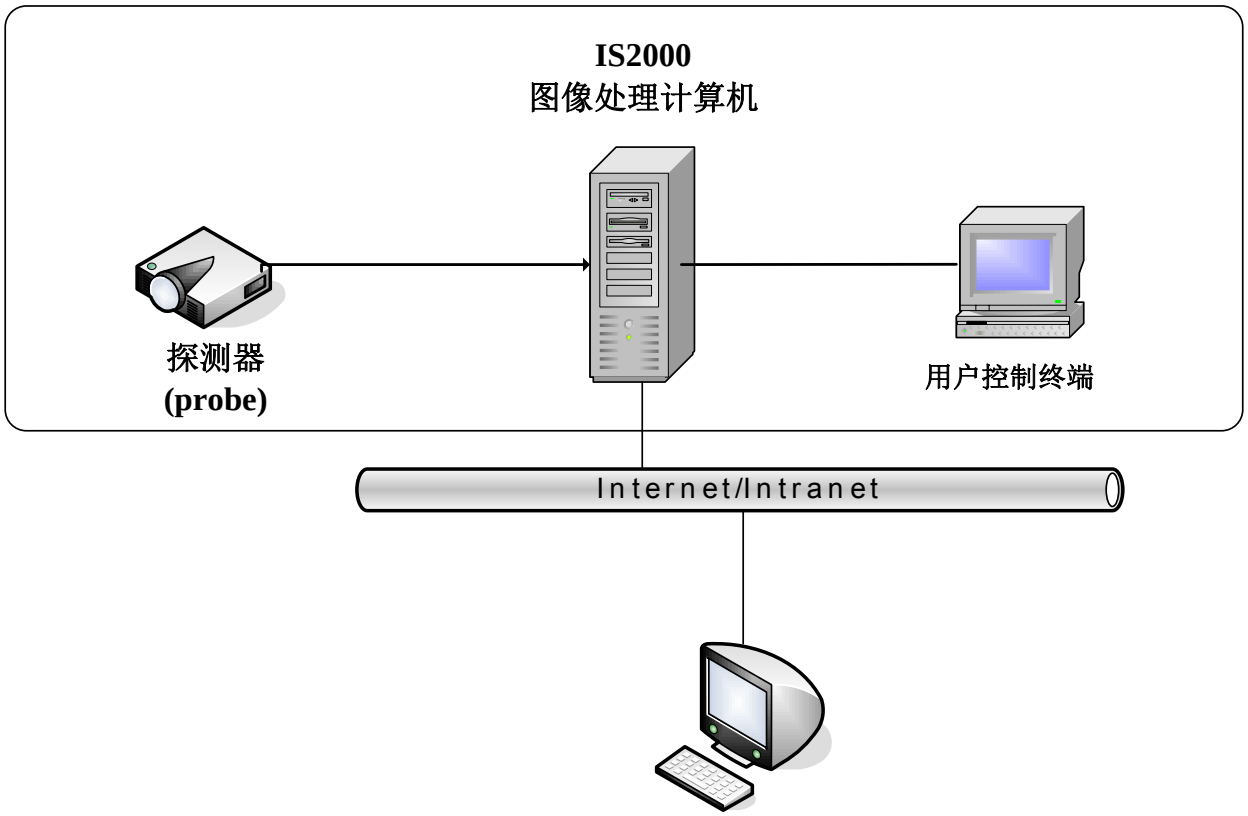
参考书

《实用软件体系结构》，霍夫梅斯特(Christine Hofmeister)，电子工业出版社，2003.



案例– IS2000系统

- IS2000系统简介



图：IS2000的硬件配置

案例– IS2000系统

- **系统需求描述**

IS2000产品基本的功能就是用探测器上的照相机捕获图像，图像源可以是电磁辐射的一种—可见光，也可以是来自热量或者其他的发射源的其他频率阴电磁光谱。为了补充这些图像，传感器要预存一些补充数据， 比如，相机和成橡物体之间的距离。为了生成这个图像，IS2000需要采集原始数据，把它变换成传感器的数据，以及适合显示的图像。

IS2000的用户可能希望获得一张单独的图像、不同时间间隔的一组图像或者是运动中的图像。用户还可能希望得到照相机从固定或者在不同的角度移动获得的照片。所有这些选择都是采集过程的一个实例。这个系统有一系列内置的采集函数，可由用户在采集前或者过程中设定参数，系统还允许用户定义一个新的采集过程。

采集过程开始的时候，用户可以通过观察原始数据映射为原始图像的过程来进行监测。在采集原始图像之后，用户要执行许多操作过程增强图像以用于显示，用户近可以把图像作输到远程的图像系统，以用于远程显示或各处理。

案例– IS2000系统

- **系统需求描述**

IS2000和3个外界事物进行交互：

1、控制采集的用户

2、被探侧的实体

3、将系统和其他显示工作站连接的网络。

对于这些交互，IS2000都有用于协调的硬件：用户控制面板、探测硬件和网络连接器。当IS2000在计算机上运行时，这些硬件都拥有和IS2000软件的接口：

- 用户押制面板的接口用来设定采失参数、监控采集和选择输出的图像。
- 探测饺件的接口用来控制探测器硬件并且接收数据。
- 网络连接器的接口用来通过网络输出图像。

案例– IS2000系统

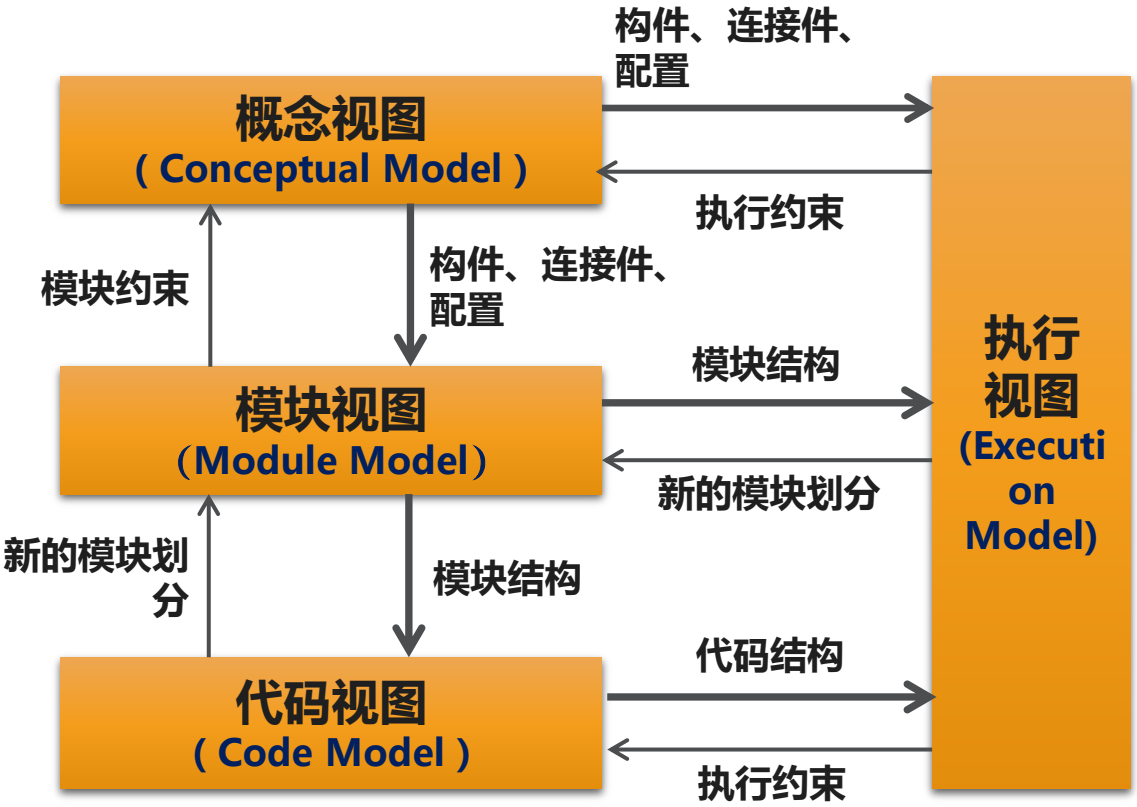
- **系统需求描述（非功能性需求）**

IS2000的设计要只有可扩展性、可维伊性和可移植性。它要有足够的灵活性来适应一些预料之小的变化。产品需求在它的开发和整个生命周期中都会发生变化，照相机、探侧器这些物理器械可能会被新的型号所代替，随着用户对系统的深入了解，用户会希望更有效地控制系统，系统处理能力的增强， 也会使越来越多的工作由用户转向系统。

IS2000的另一个特点也可能变化。随着时间的流逝，内置的采集过程将有进一步发展，数据的处理也会不断进步，还可能加入新类型的过滤器。它需要保证和新的、进化中的文件格式和图像信息传输的标准相一致。

IS2000架构分析设计的四种视图模型

- 霍夫梅斯特(Christine Hofmeister)基于大量的工程实践经验，在《实用软件体系结构》一书中提出软件体系结构建模的四种视图模型



四种视图模型

概念视图（Conceptual Model）

在概念视图的建模中，问题和解决方案主要通过领域术语来考虑的。对于特定的软件及硬件技术，它们应当是相对独立的。概念视图的工程关注点主要包括：

- 系统如何满足需求?
- 商用构件(Commercial Off-The-Shelf Components)怎样组装成整体，怎样在功能层上与系统的其他部分交互?
- 领域特定的硬件与软件如何融入系统?
- 功能是如何被分割并进入产品各版本的?
- 系统如何与之前版本的产品合并?它如何支持未来的版本?
- 如何支持产品线?
- 如何将由需求或领域中所做的变动引起的影响最小化?

四种视图模型

模块视图（Module Model）

在模块视图建模中，概念视图中的构件和连接子被映射为子系统和模块。在这里，架构师强调的是如何用现有的软件平台以及技术实现概念的解决方案。主要的工程关注点有如下几点：

- 产品是如何映射到软件平台的?
- 使用了什么样的系统支持或系统服务?
- 怎样支持测试?
- 如何降低模块间的依赖性?
- 如何将模块与子系统的复用最大化?
- 当商用软件、软件平台或标难发生变动时，采用何种技术在封装产品时可以将它们与产品进行隔离?

四种视图模型

执行视图（Execution Model）

执行视图描述模块如何映射到执行平台，以及这些又如何映射到网络硬件环境（比如内存使用和硬件分配）。

对于执行视图，其主要工程关注点如下：

- 系统如何满足性能、恢复及重新配置方面的需求?
- 如何平衡资源的使用(例如： 负载平衡)?
- 如何达到必需的并发、复制及分布，而个过度增加控制算法的复杂度?
- 如何使运行时平台的改变所引起的影响达到最小?

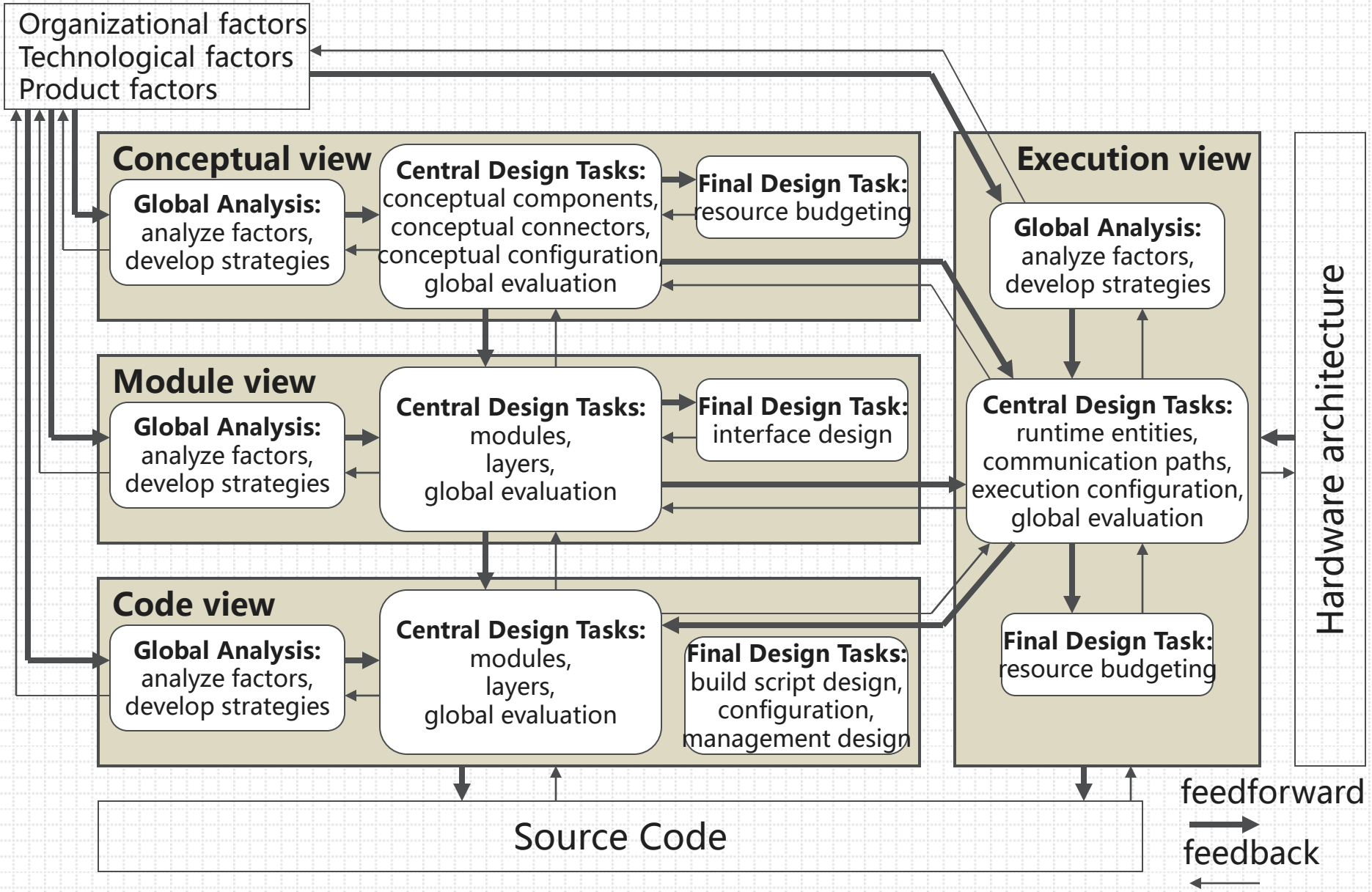
四种视图模型

代码视图（Code Model）

在代码体系结构视图中，架构师决定执行视图中的执行时实体如何映射到部署构件(例如，可执行构件)，决定模块视图中的模块如何映射到源构件，以及部署构件如何从源构件牛成。这种视图巾重要的工程关注点如：

- 如何降低产品升级的时间和费出?
- 如何管理产品版伞及发布?
- 如何降低构造时间?
- 需要什么工具支持开发环境?
- 如何支持集成与测试?

分析设计过程总览（Process Overview）

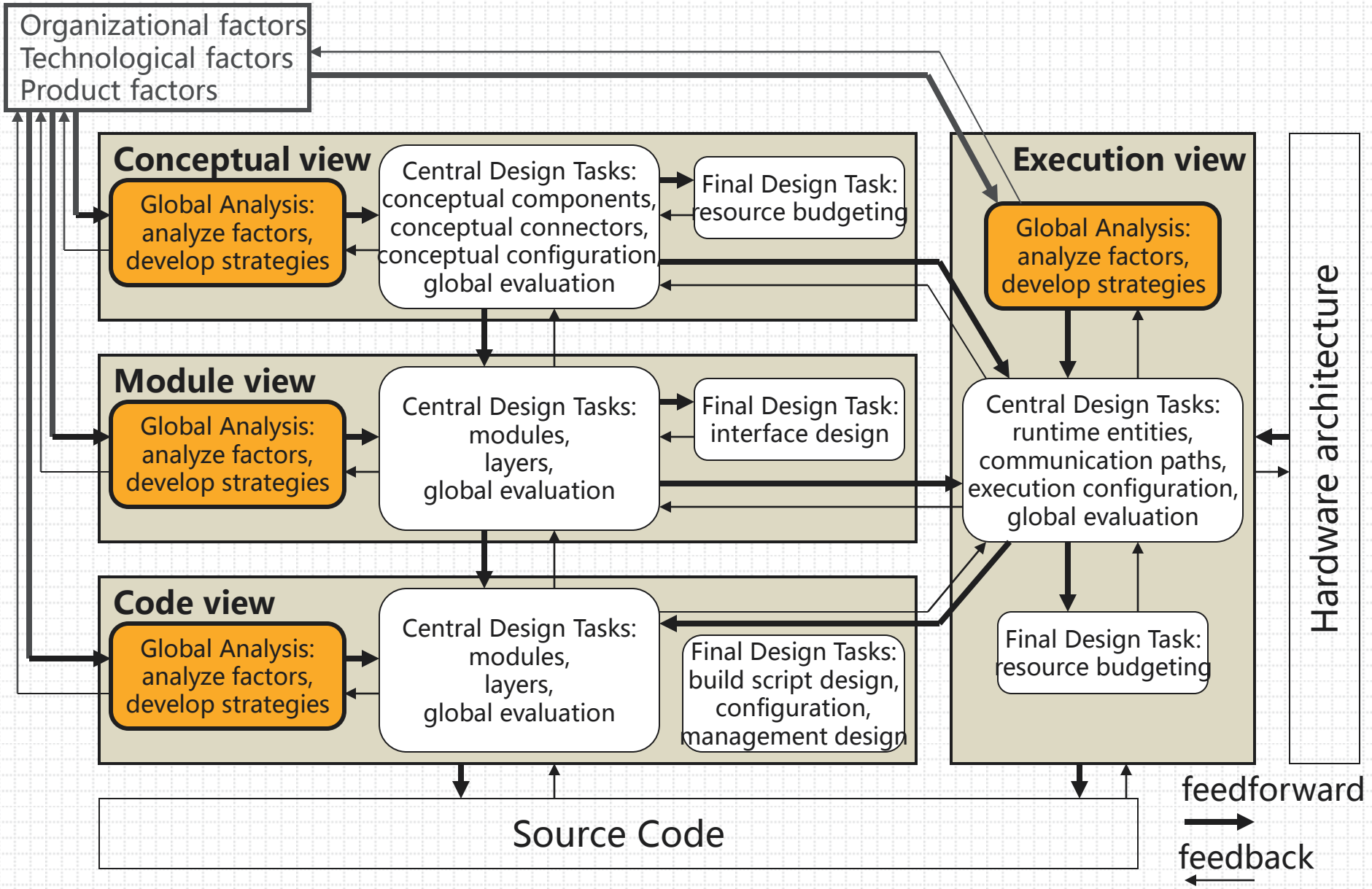


分析设计过程总览（Process Overview）

4种体系结构视图的建立过程都包含三种相同的分析设计任务：

- **全局分析（Global Analysis）**
- **核心设计任务（Central Design Tasks）**
- **最终设计任务（Final Design Tasks）**

全局分析（Global Analysis）



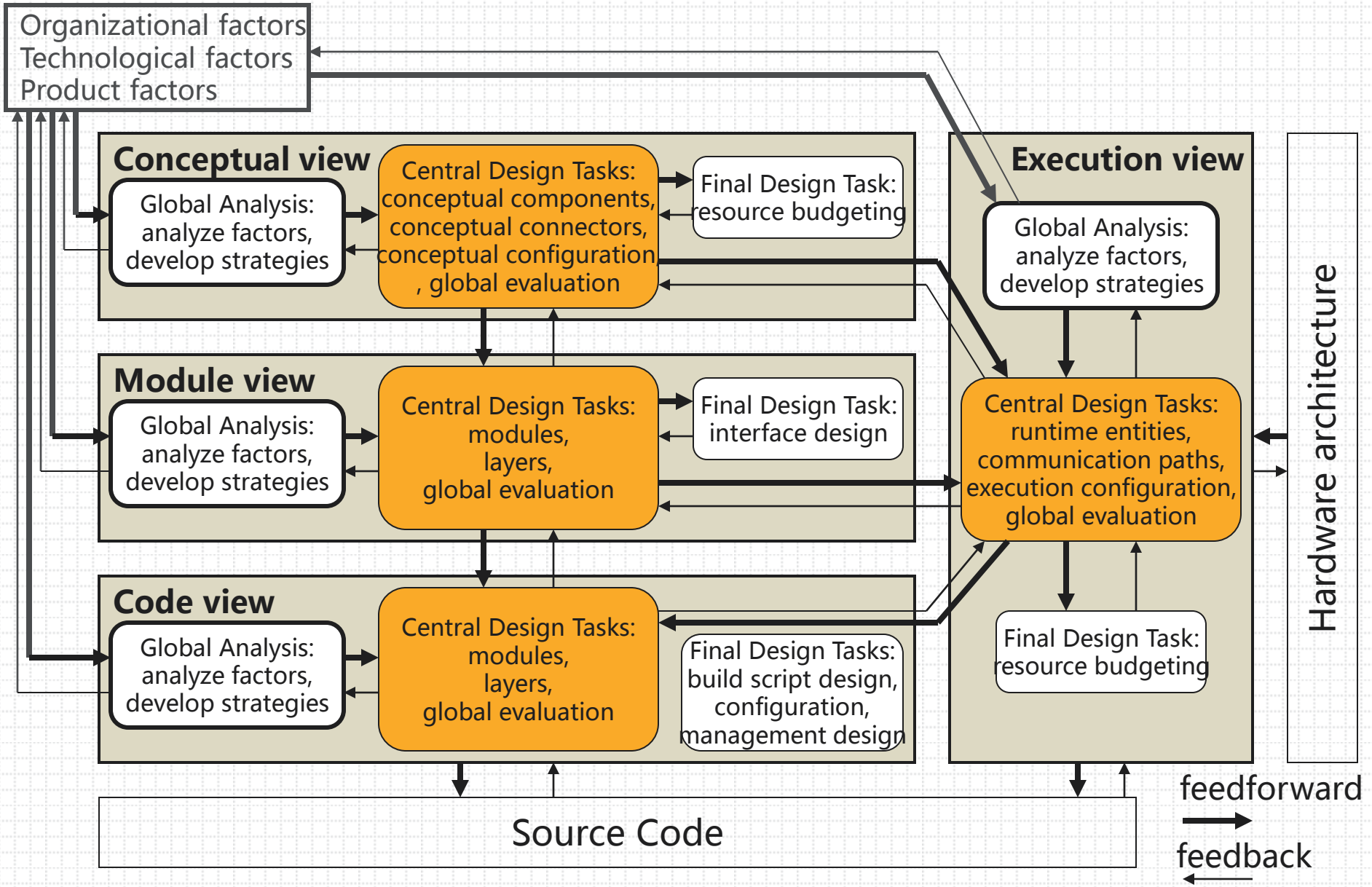
全局分析（Global Analysis）

每个视图模型建立的第1步都要进行全局分析。在全局分析的任务中，首先需要标识可能影响软件体系结构的外部因素和关键需求。然后需要对它们进行分析，以提出设计体系结构的策略。如果不能全部满足这些因素和需求，就需要衡量哪个更重要，修改某些需求，或者改变一些外部因素，以获得体系结构可使用的策略。

The first task in each view is global analysis, which identifies external influences and critical requirements, and designs strategies to deal with them.

If the external influences and requirements cannot all be satisfied, they must be prioritized or renegotiated.

核心设计任务（Central Design Tasks）

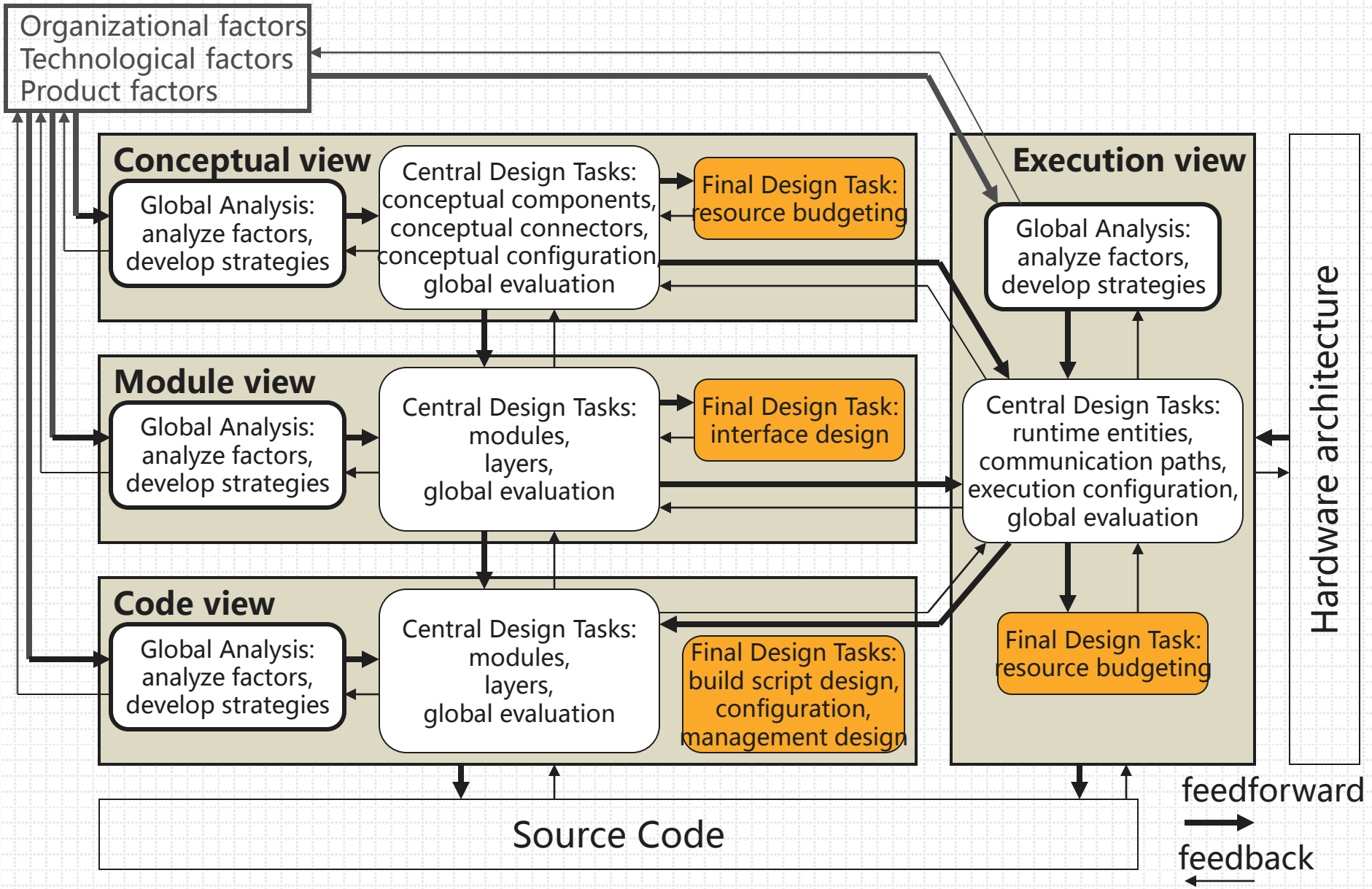


核心设计任务（Central Design Tasks）

每个视图模型建立的第2步是核心设计任务。首先要定义体系结构视图的元素和它们之间的关系，然后(如果适合该视图)定义这些元素如何配置。不同的视图有不同类型的元素，它们可能是概念上的构件、模块、过程、文件或者其他的元素。

These tasks define the elements of the view and the relationships between them. These tasks are highly connected.Global evaluation is an ongoing task. It decides which inputs to use for the other central design tasks, and what the impacts of the other central design tasks of the view are on other prior design tasks.

最终设计任务（Final Design Tasks）



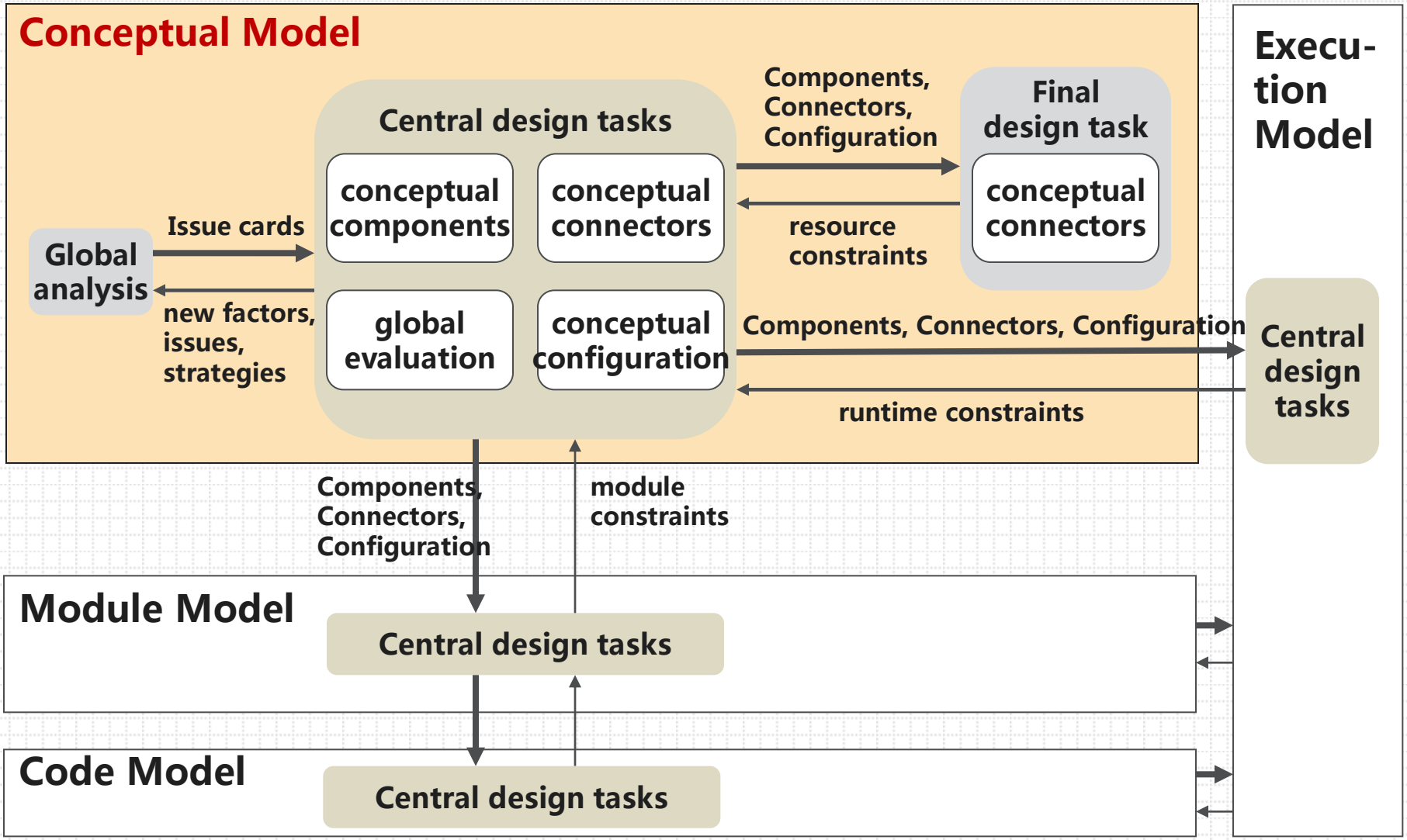
最终设计任务（Final Design Tasks）

每个视图模型建立的第3步是最终设计任务完善细化中心设计任务。如概念视图模型的最终设计任务是进行资源预算，模块视图模型的最后设计任务是接口定义。最终设计任务可能要向核心设计进行一些反馈，但是不会影响到大多数先前的设计。

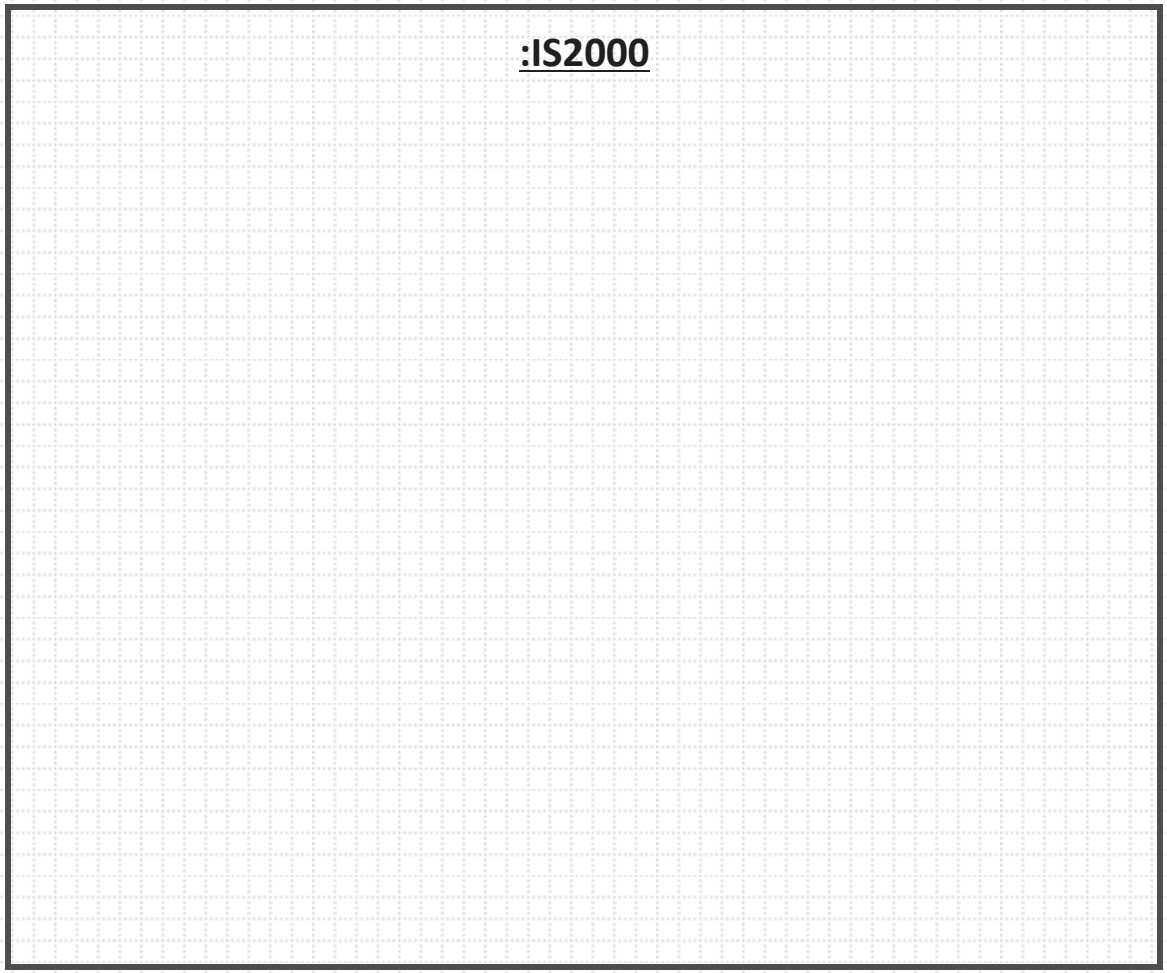
The final design tasks for each view produce the additional outputs of the view. These are less dependent of the rest of the system.

Step1 : Central Design – Conceptual Model

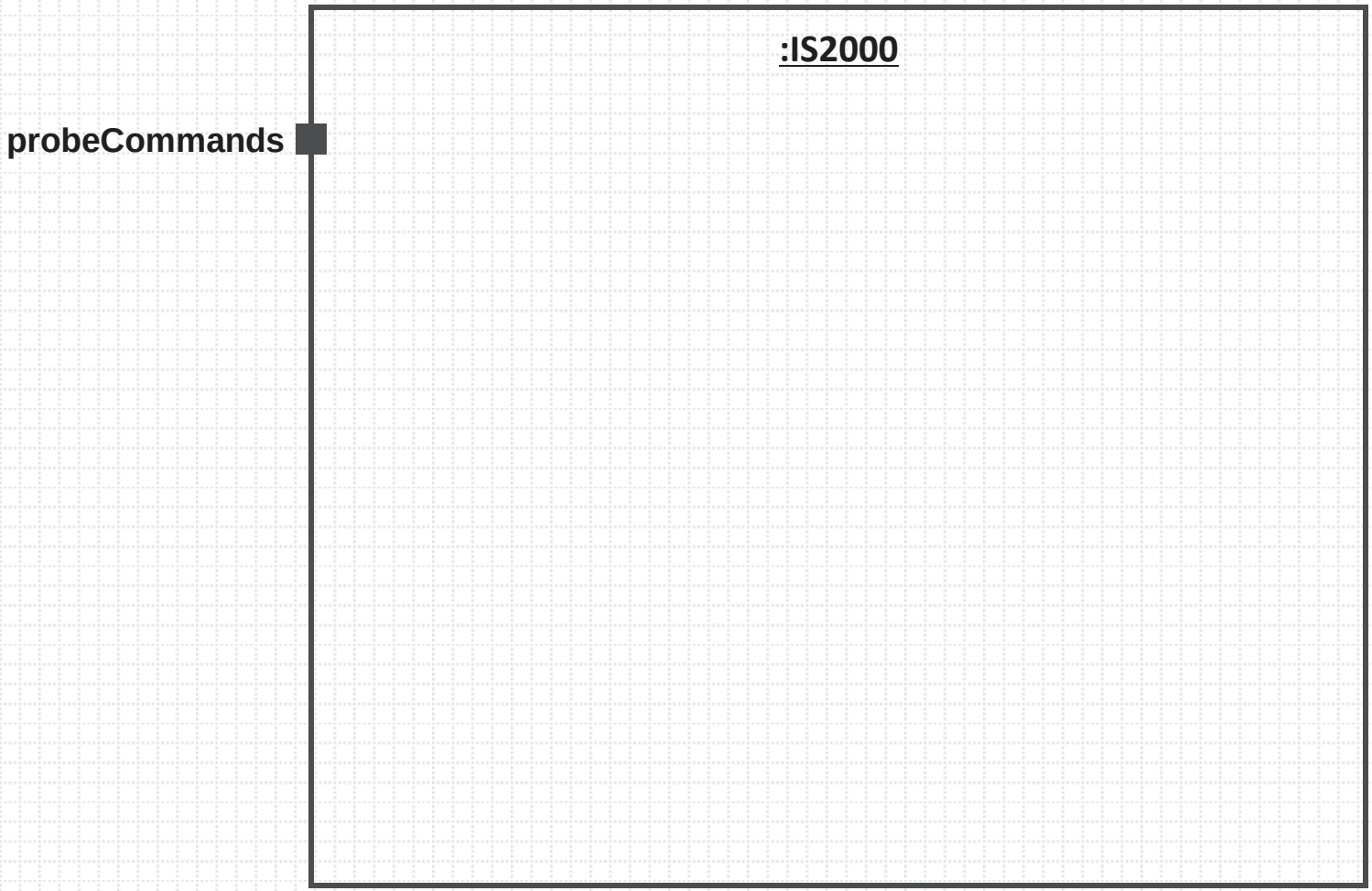
- Conceptual Model



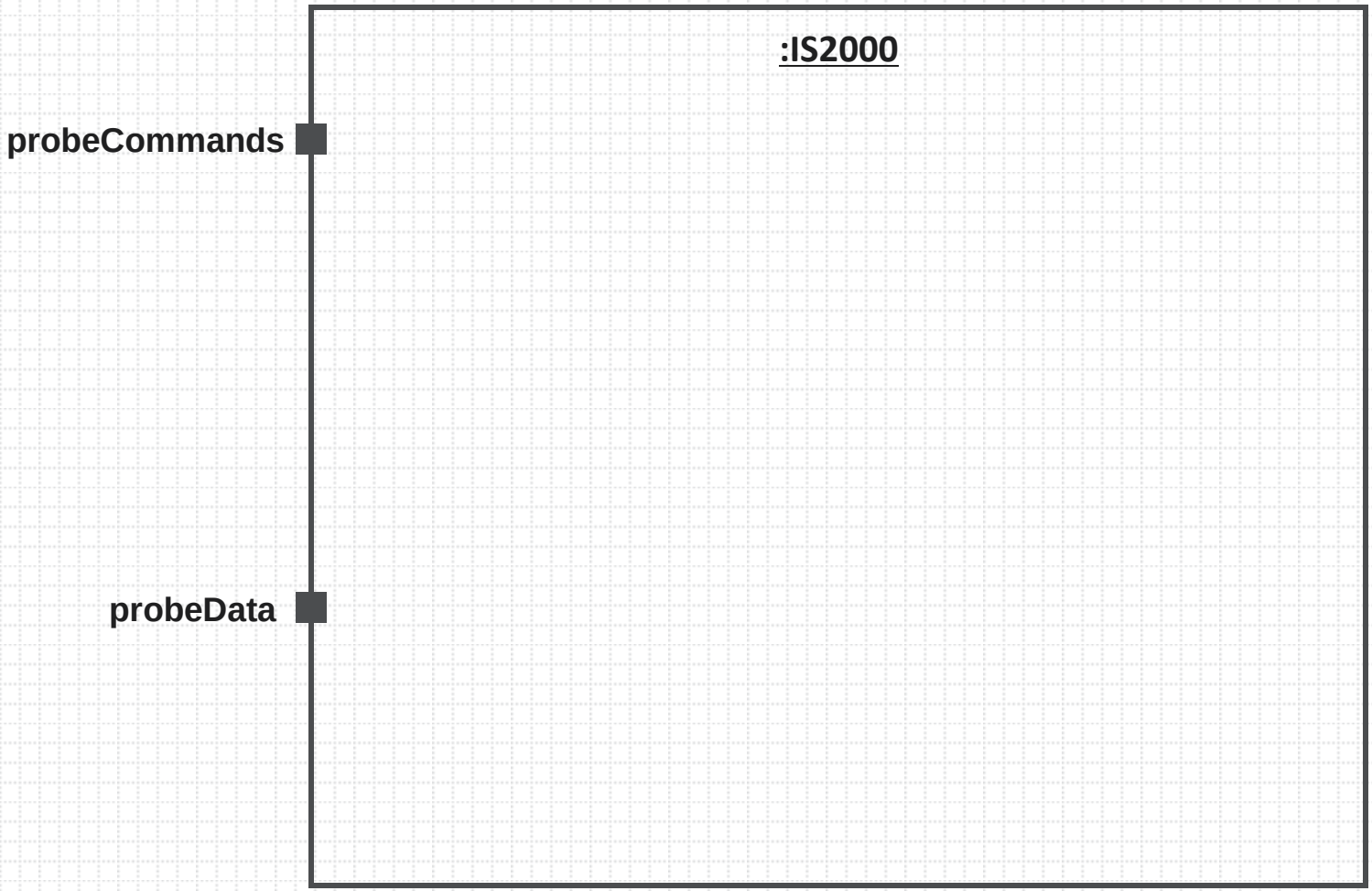
Step 1:Central Design Conceptual Component



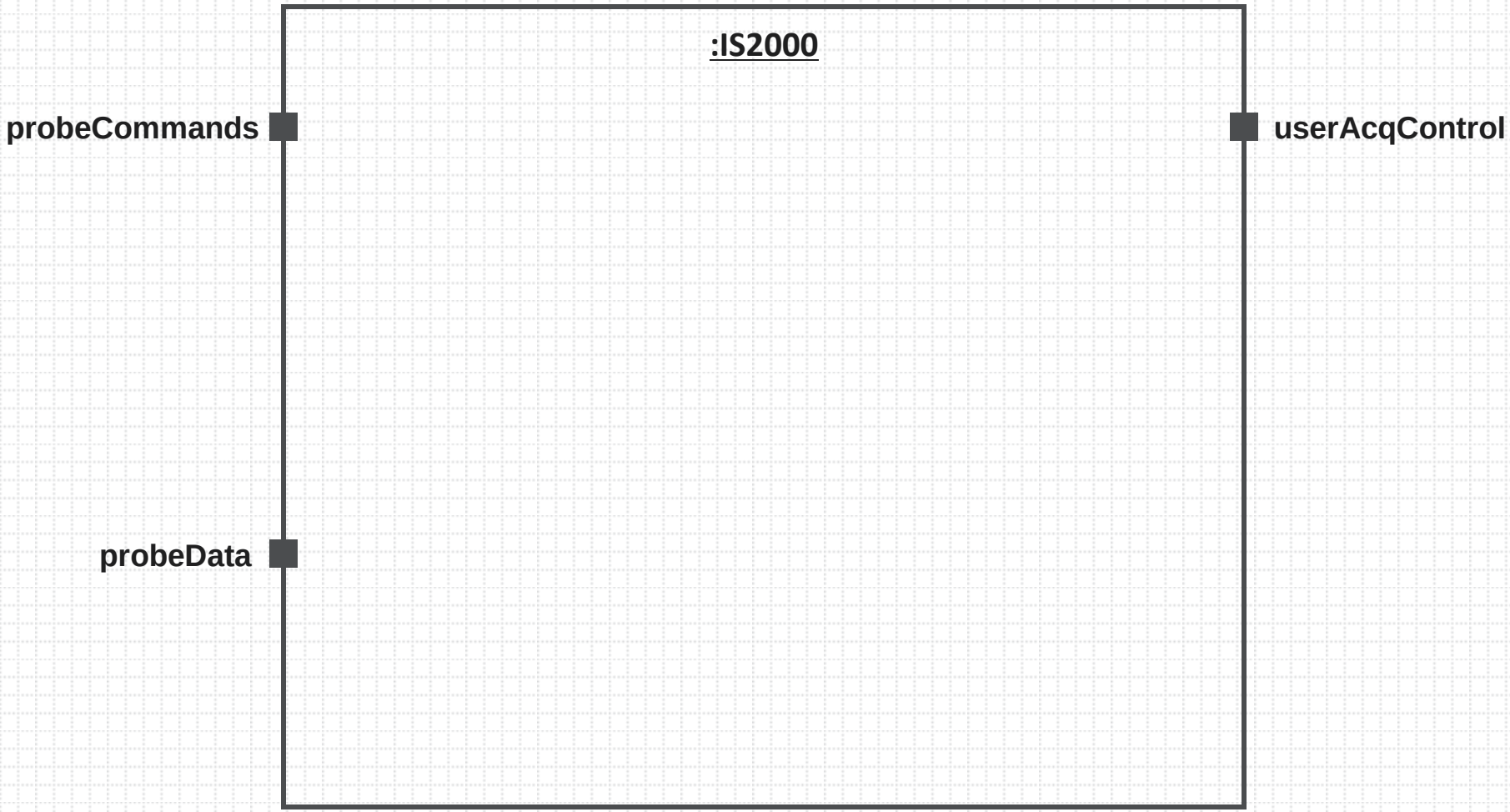
Step 1:Central Design Conceptual Component



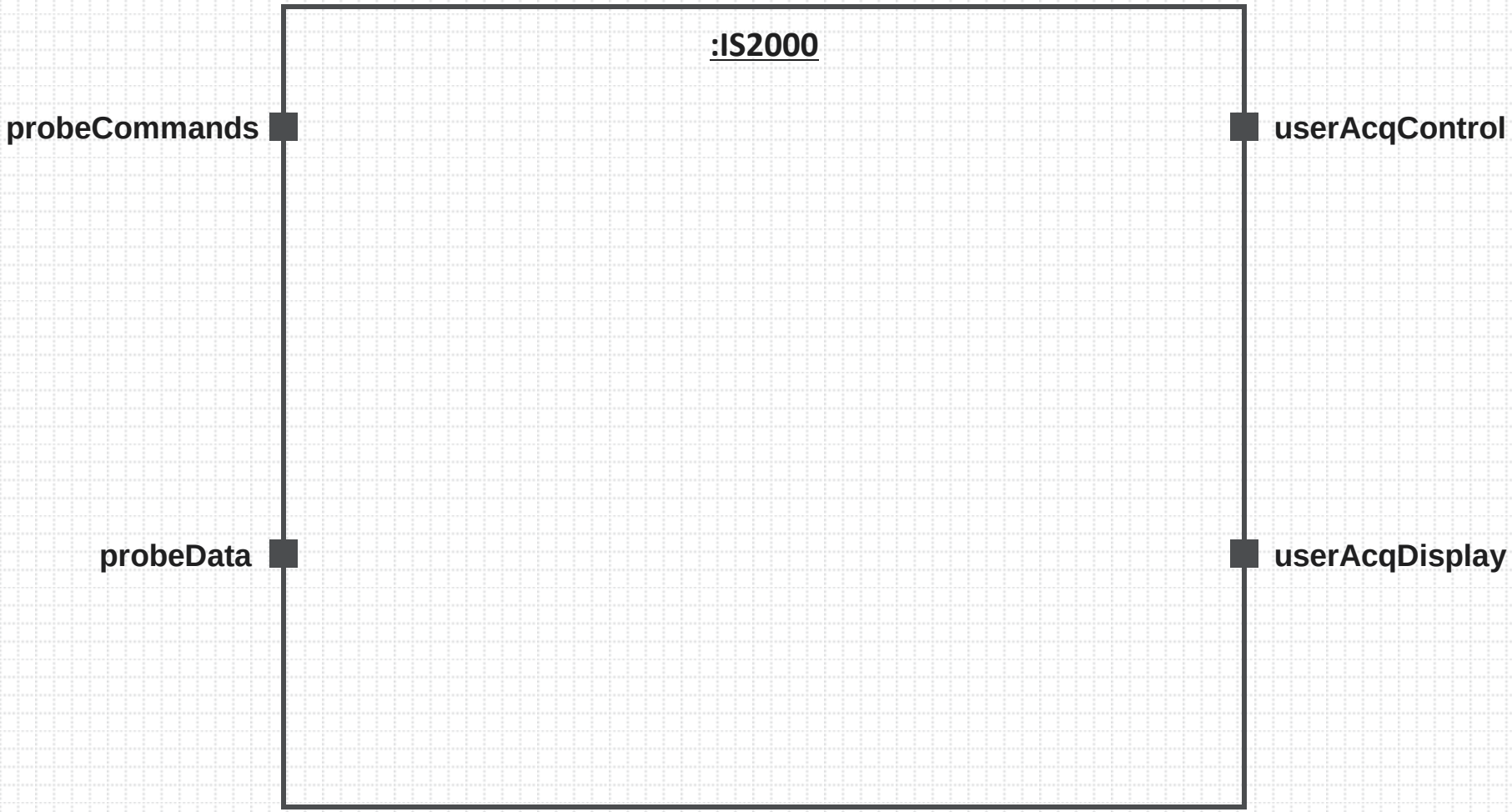
Step 1:Central Design Conceptual Component



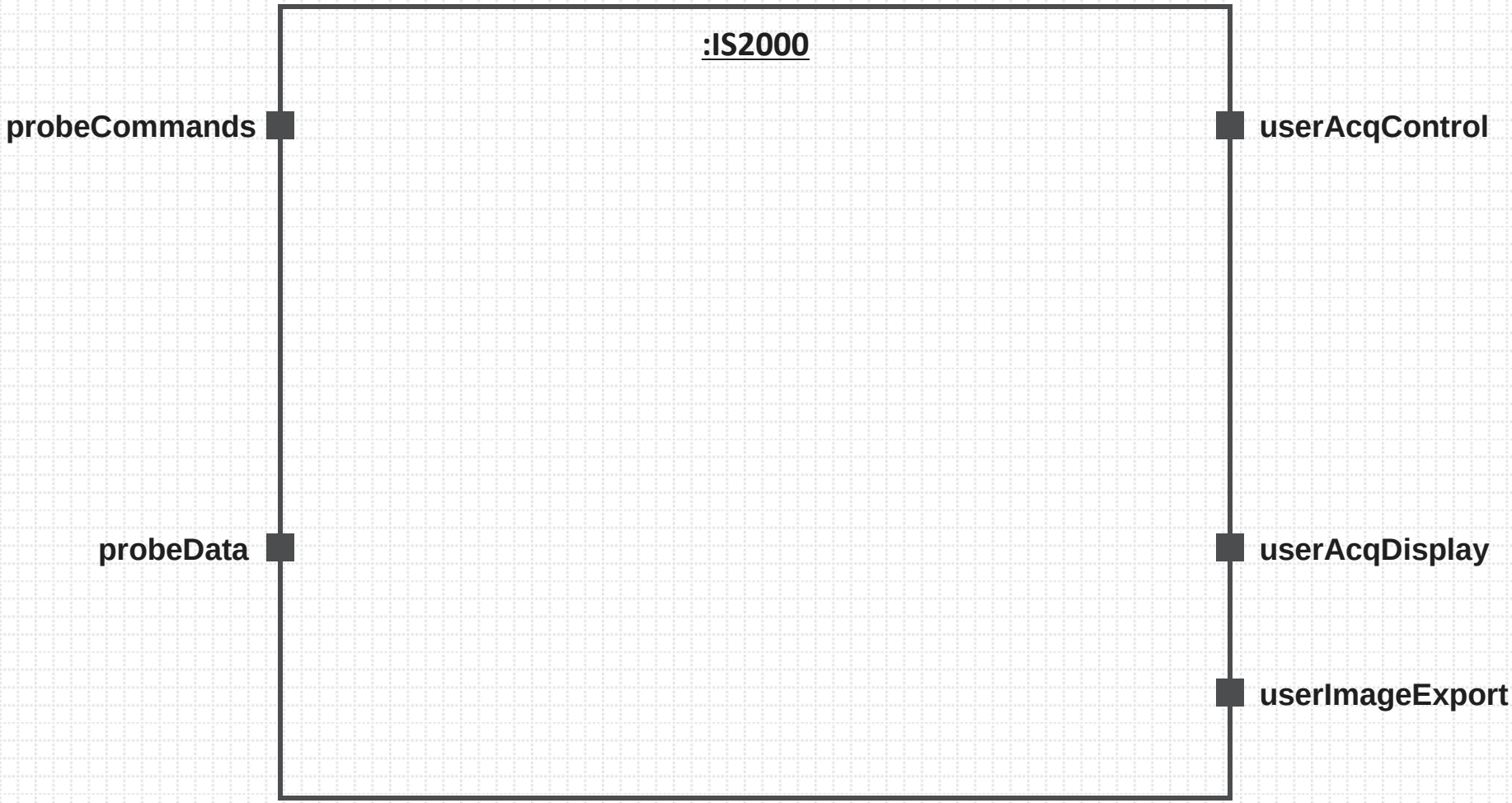
Step 1:Central Design Conceptual Component



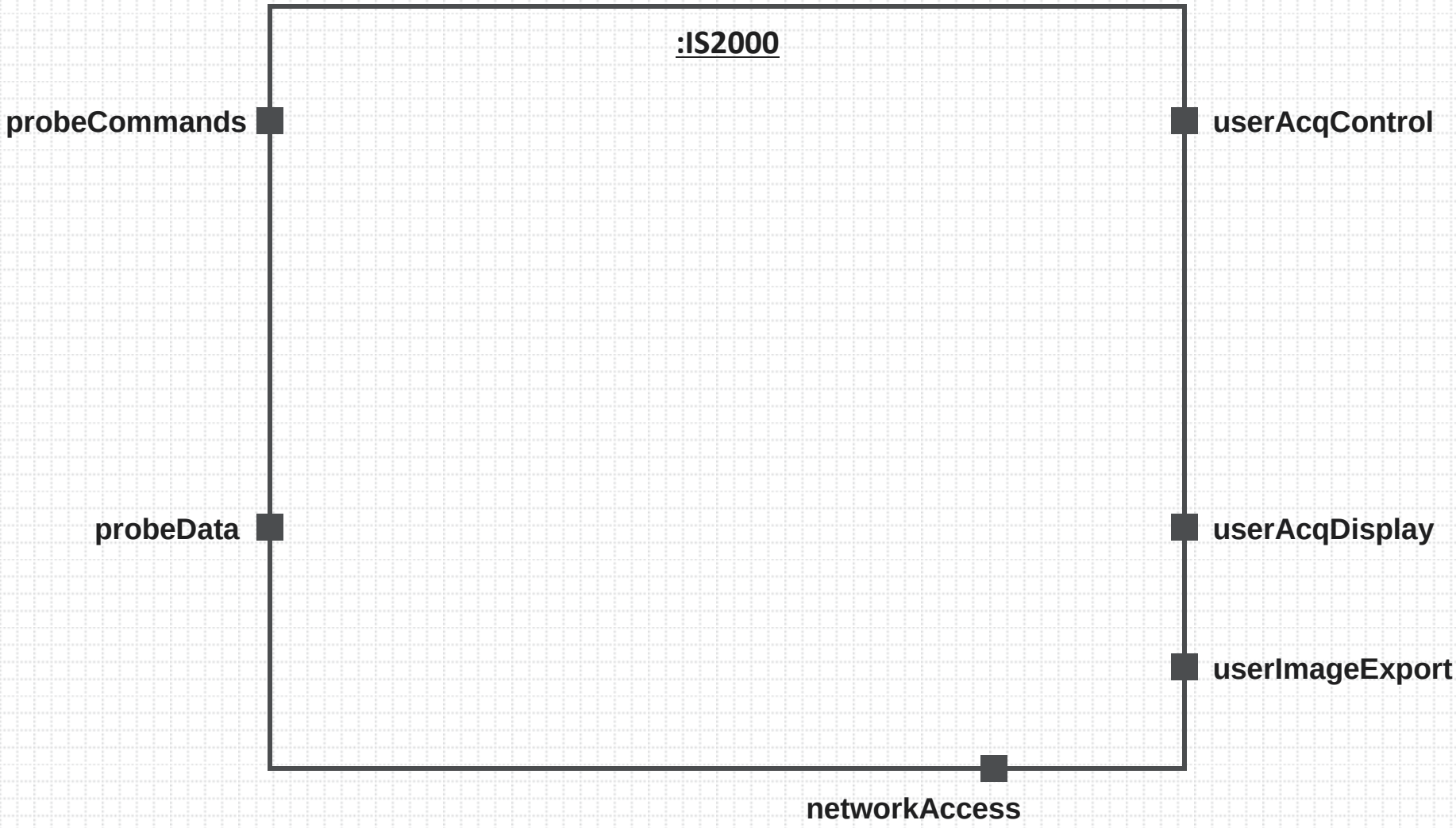
Step 1:Central Design Conceptual Component



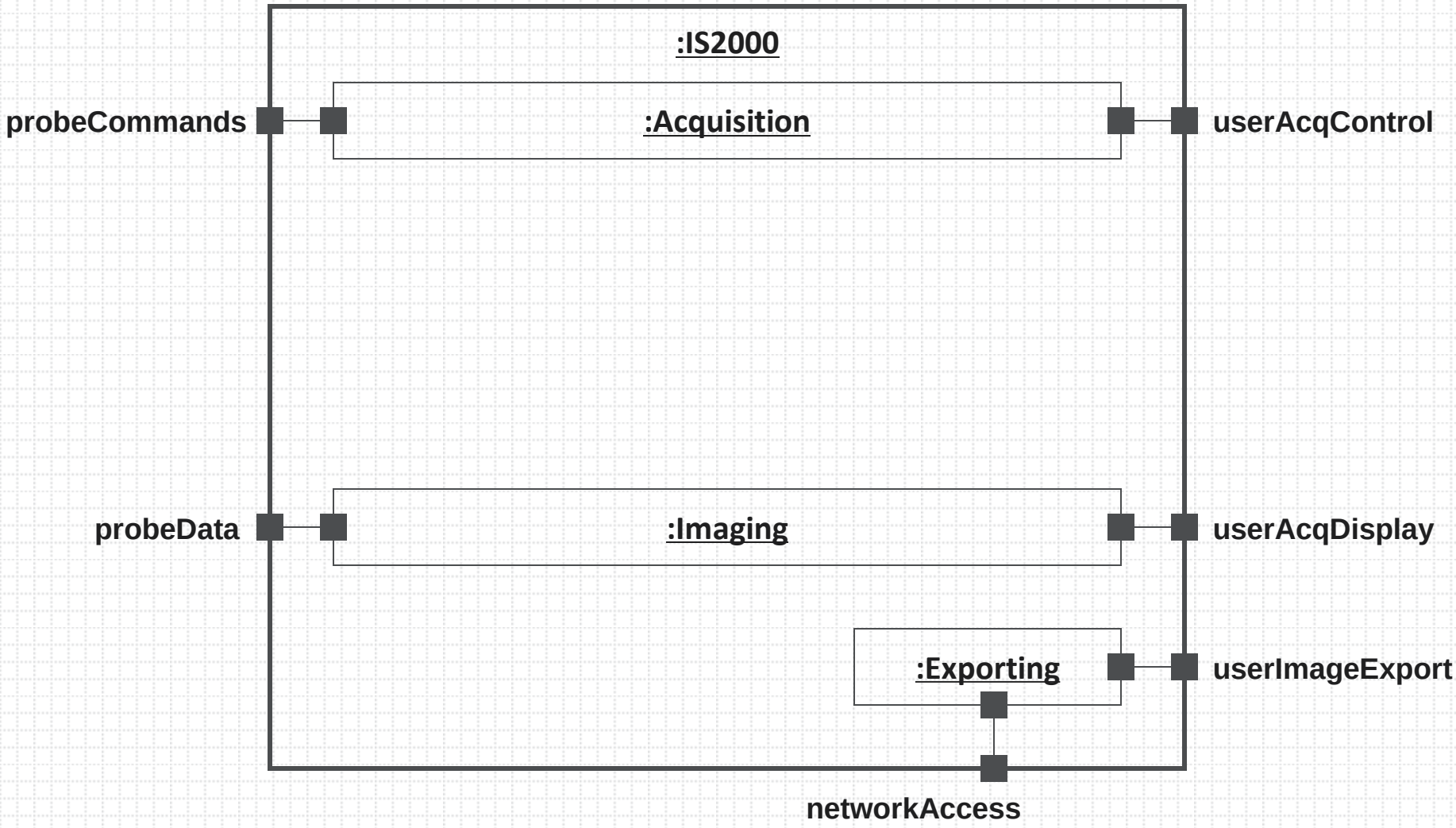
Step 1:Central Design Conceptual Component



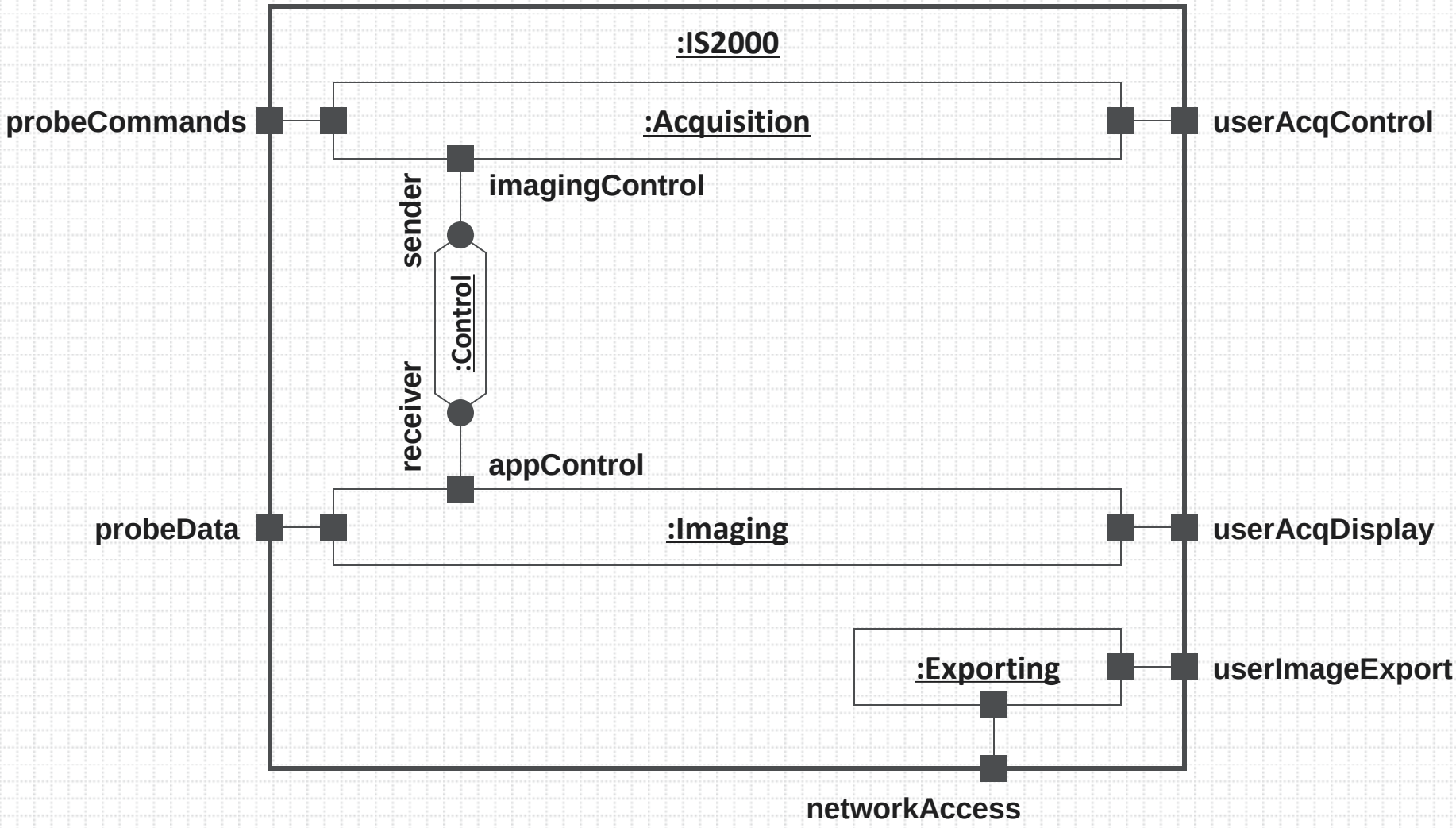
Step 1:Central Design Conceptual Component



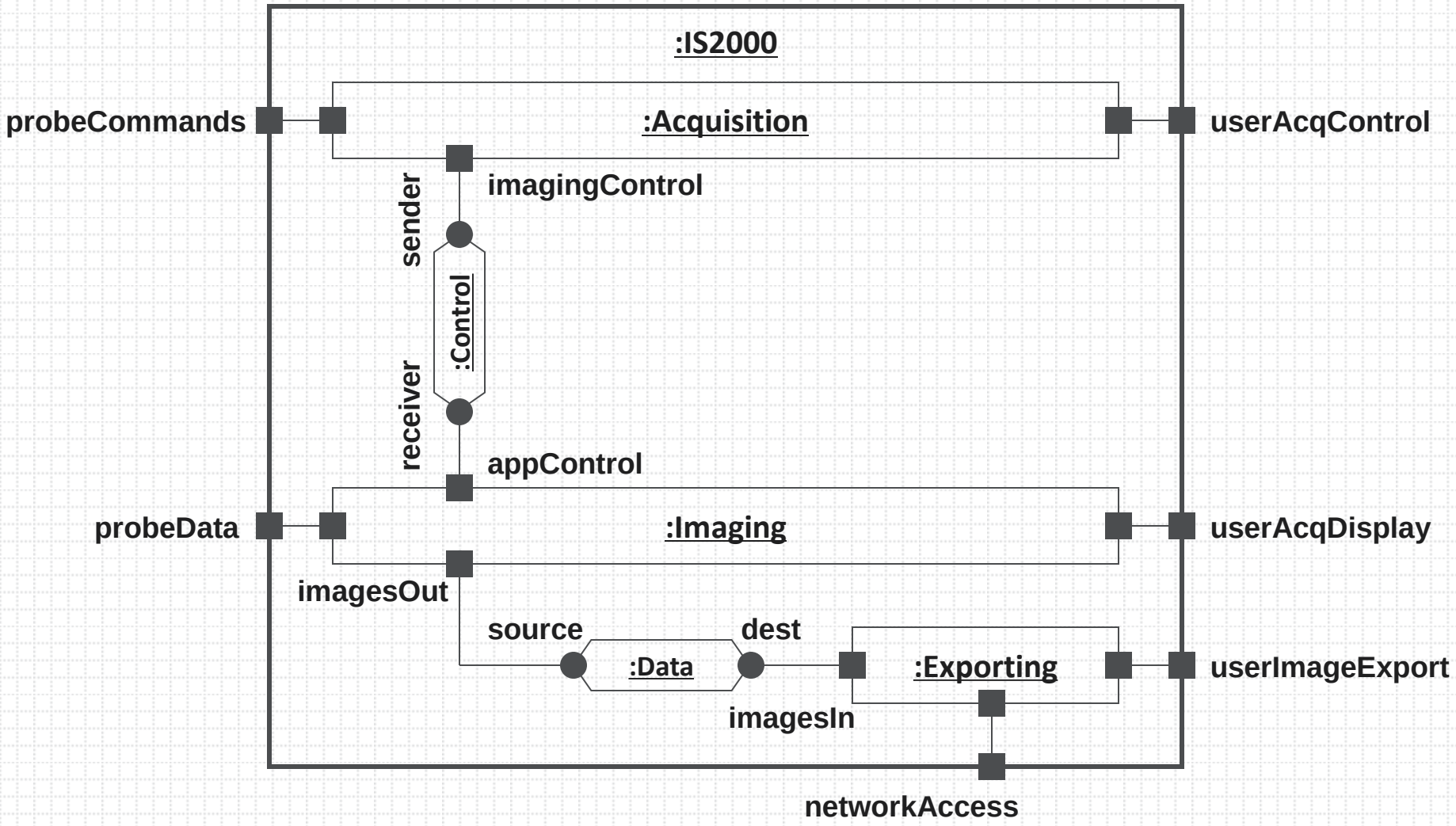
Step 1:Central Design Conceptual Component



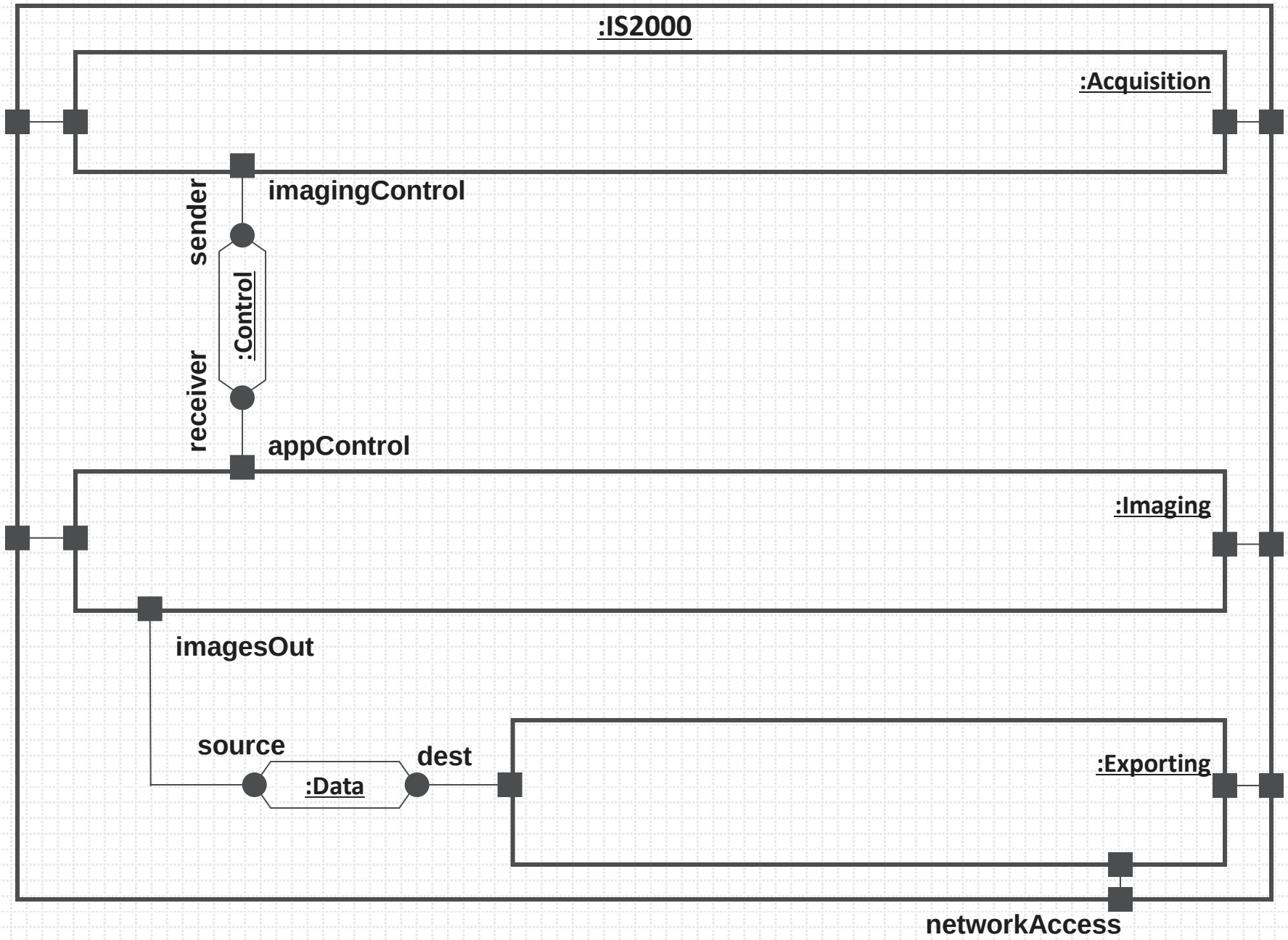
Step 1:Central Design Conceptual Connector



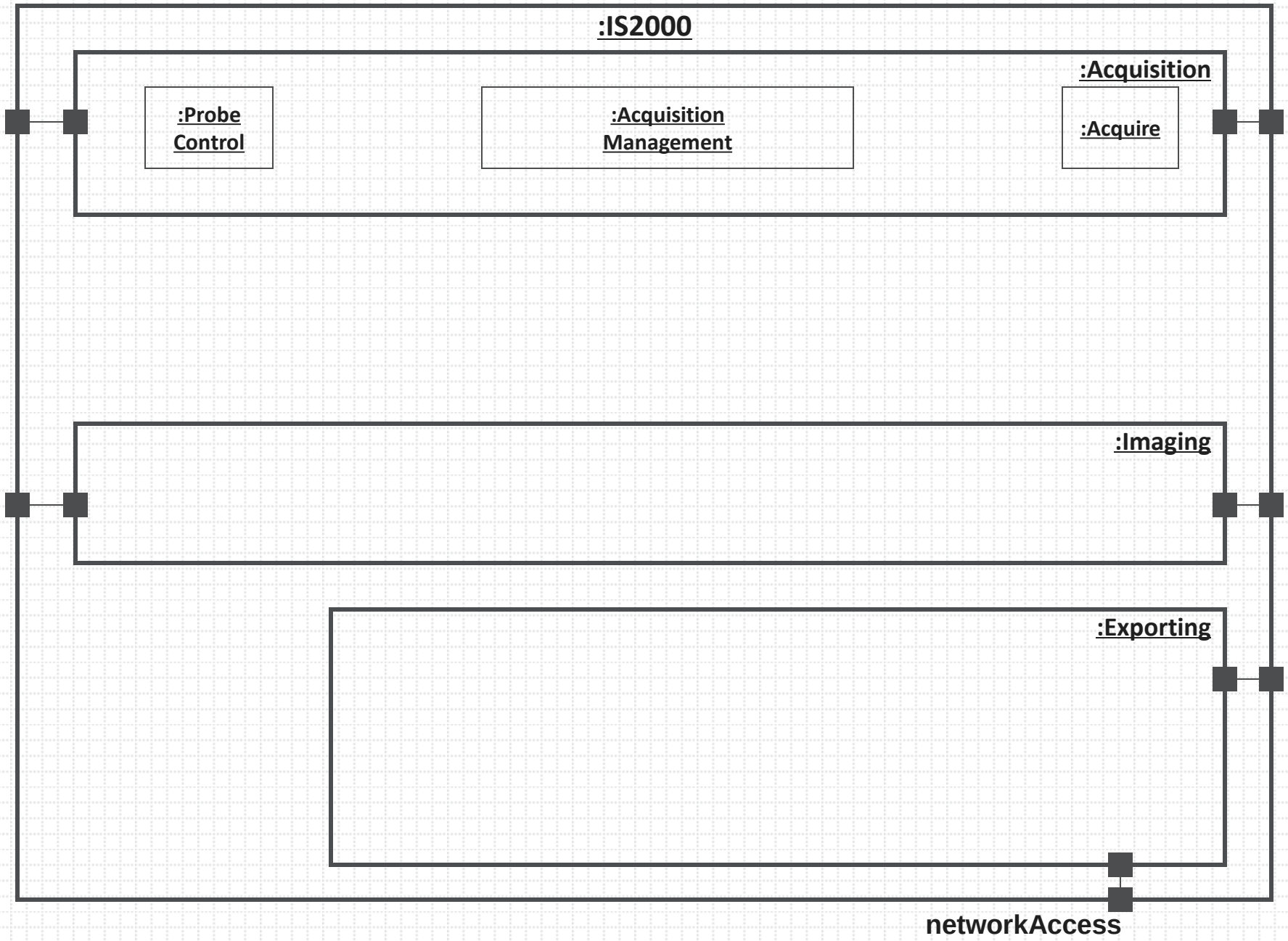
Step 1:Central Design Conceptual Connector



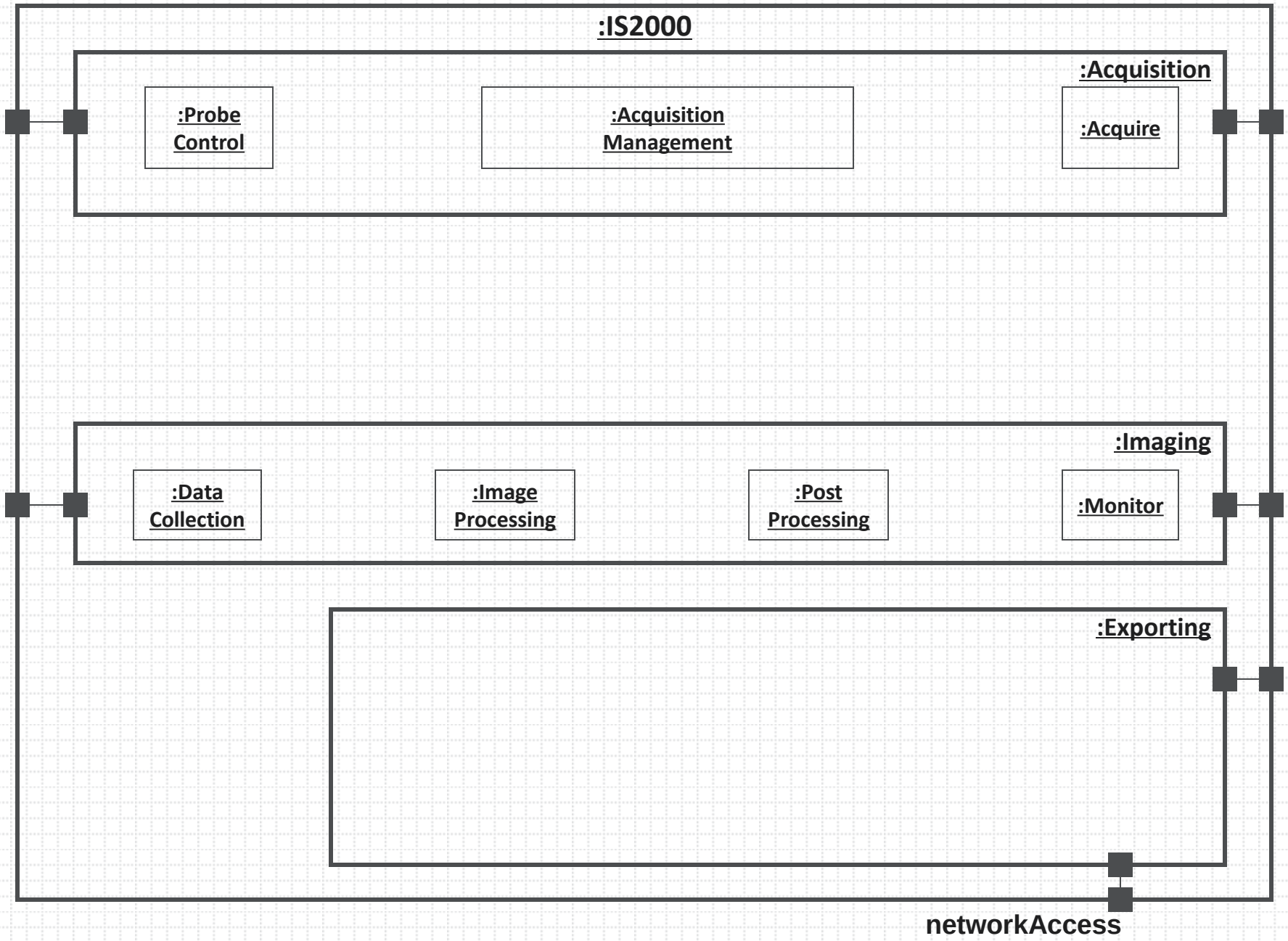
Step 1: Central Design – Conceptual Model(高层的)



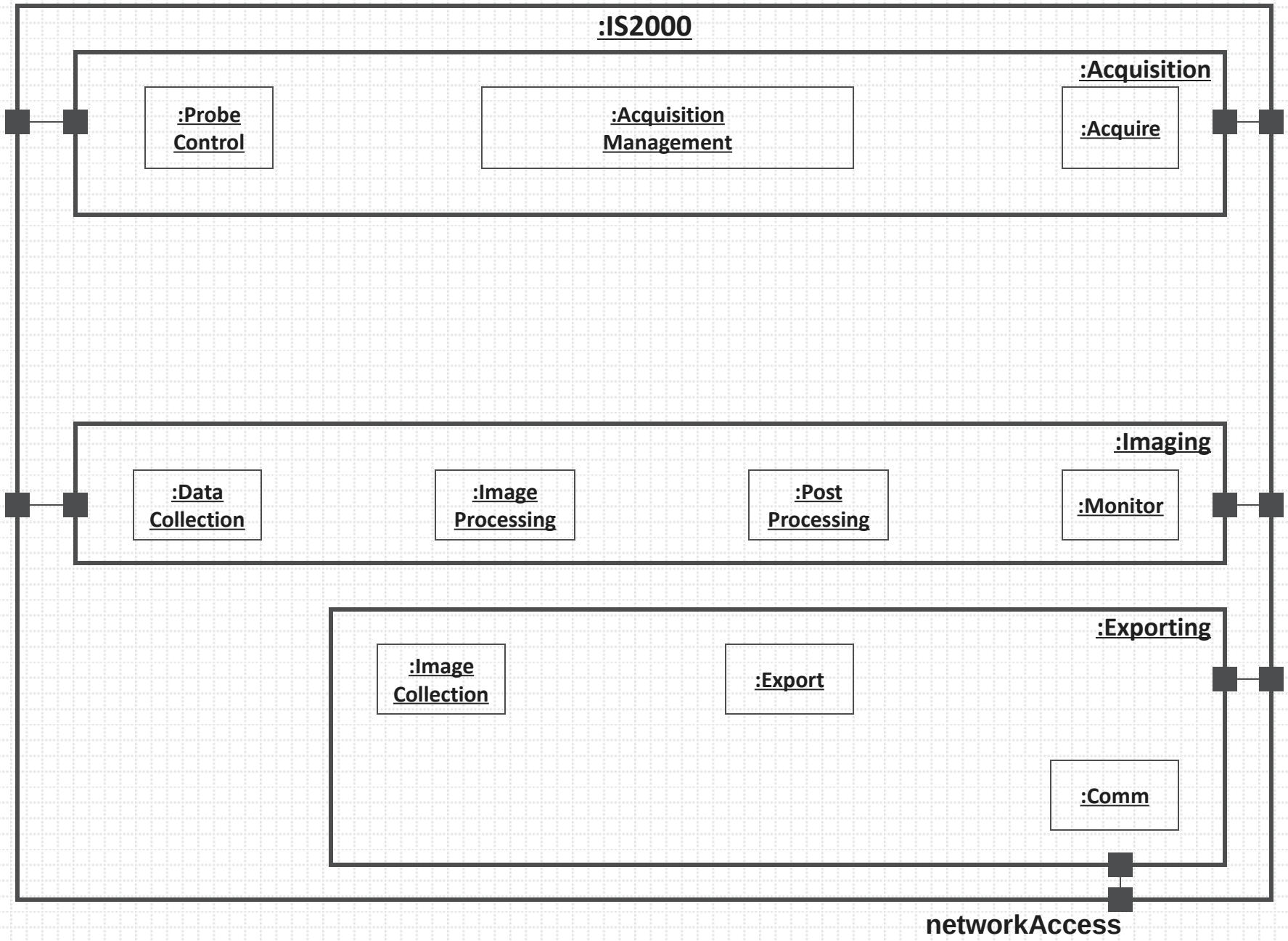
Step 1: Central Design – Component :*Acquisition* Decomposition



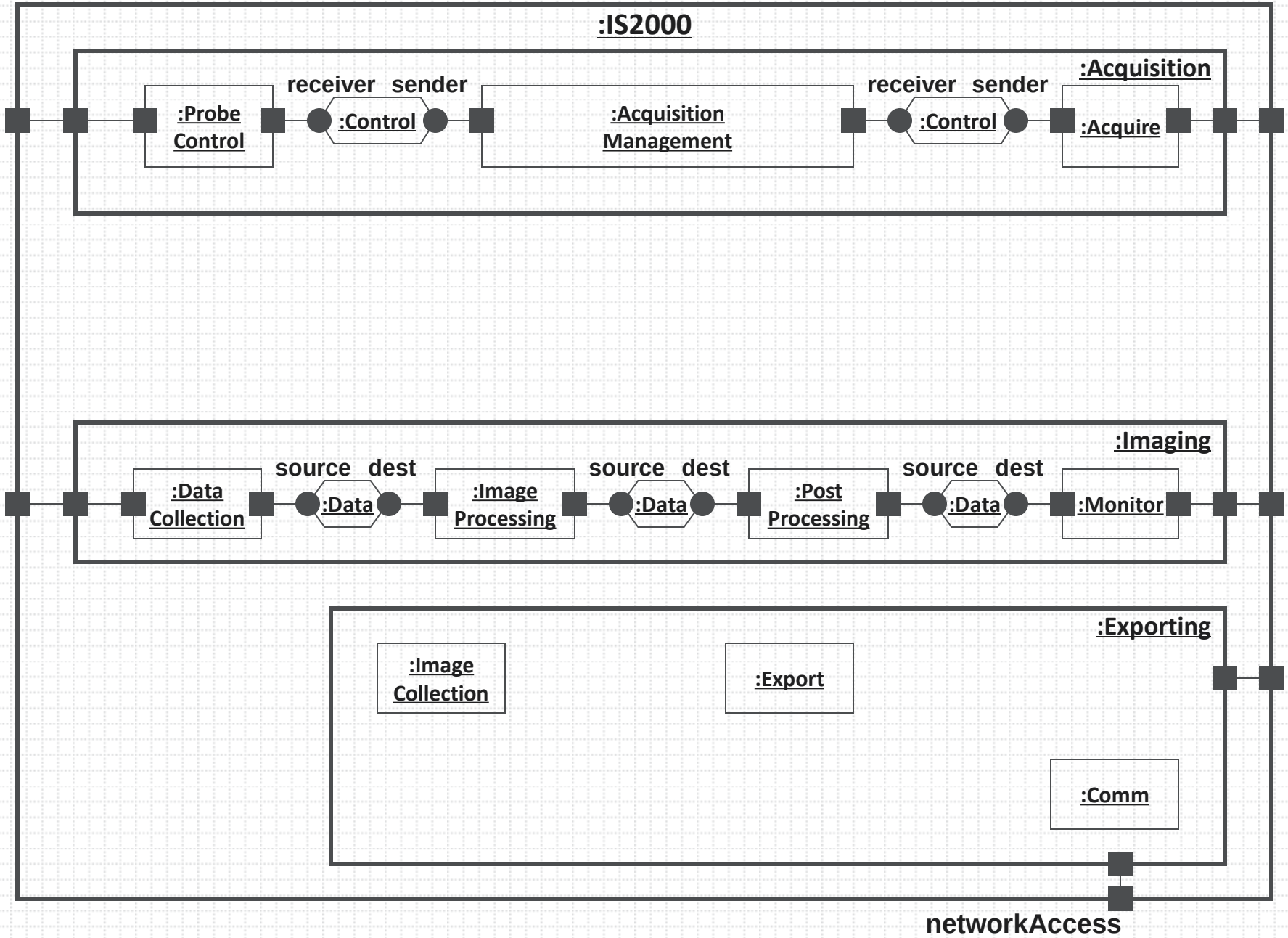
Step 1: Central Design – Component :*Imaging* Decomposition



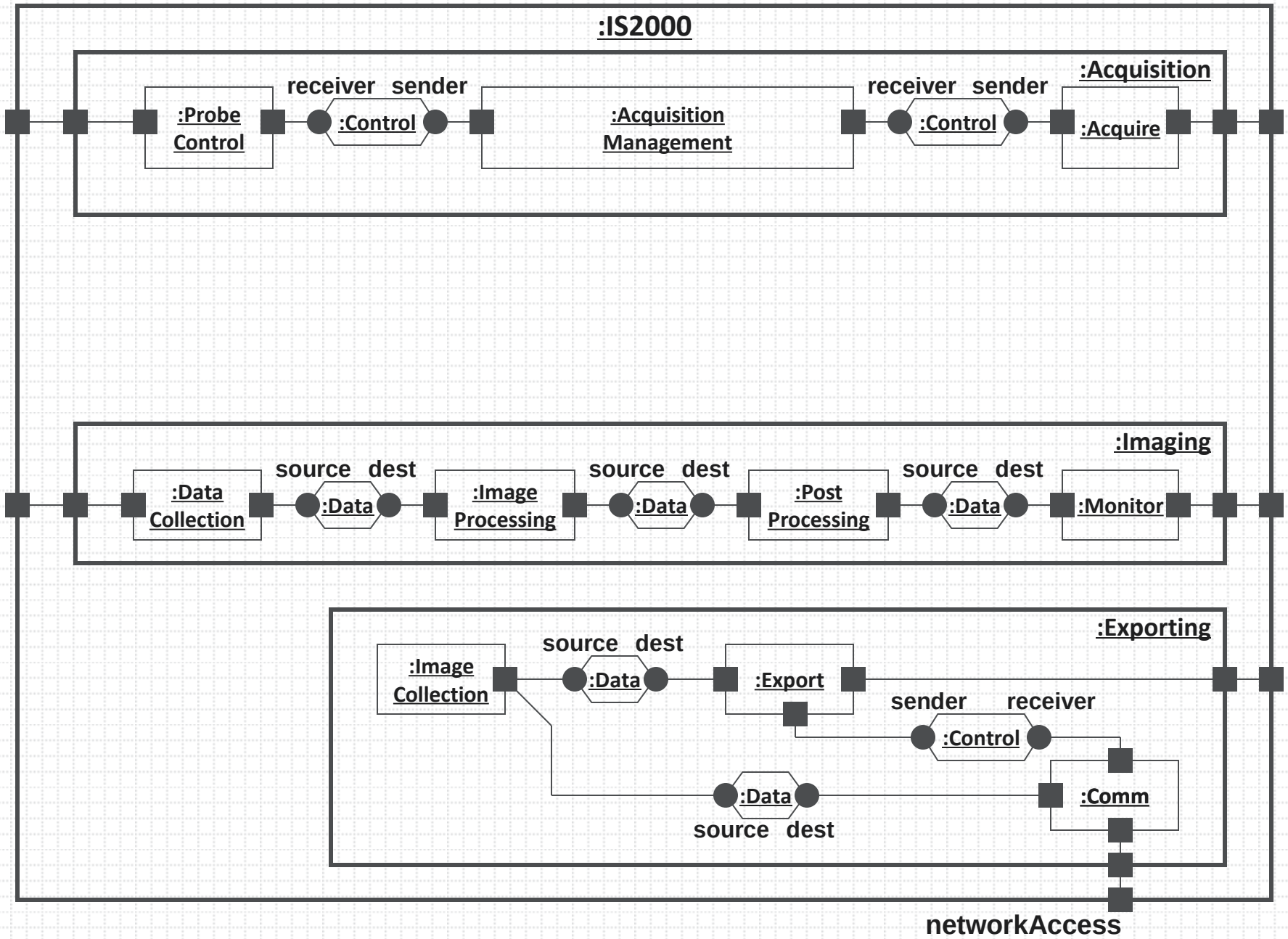
Step 1: Central Design – Component :*Exporting* Decomposition



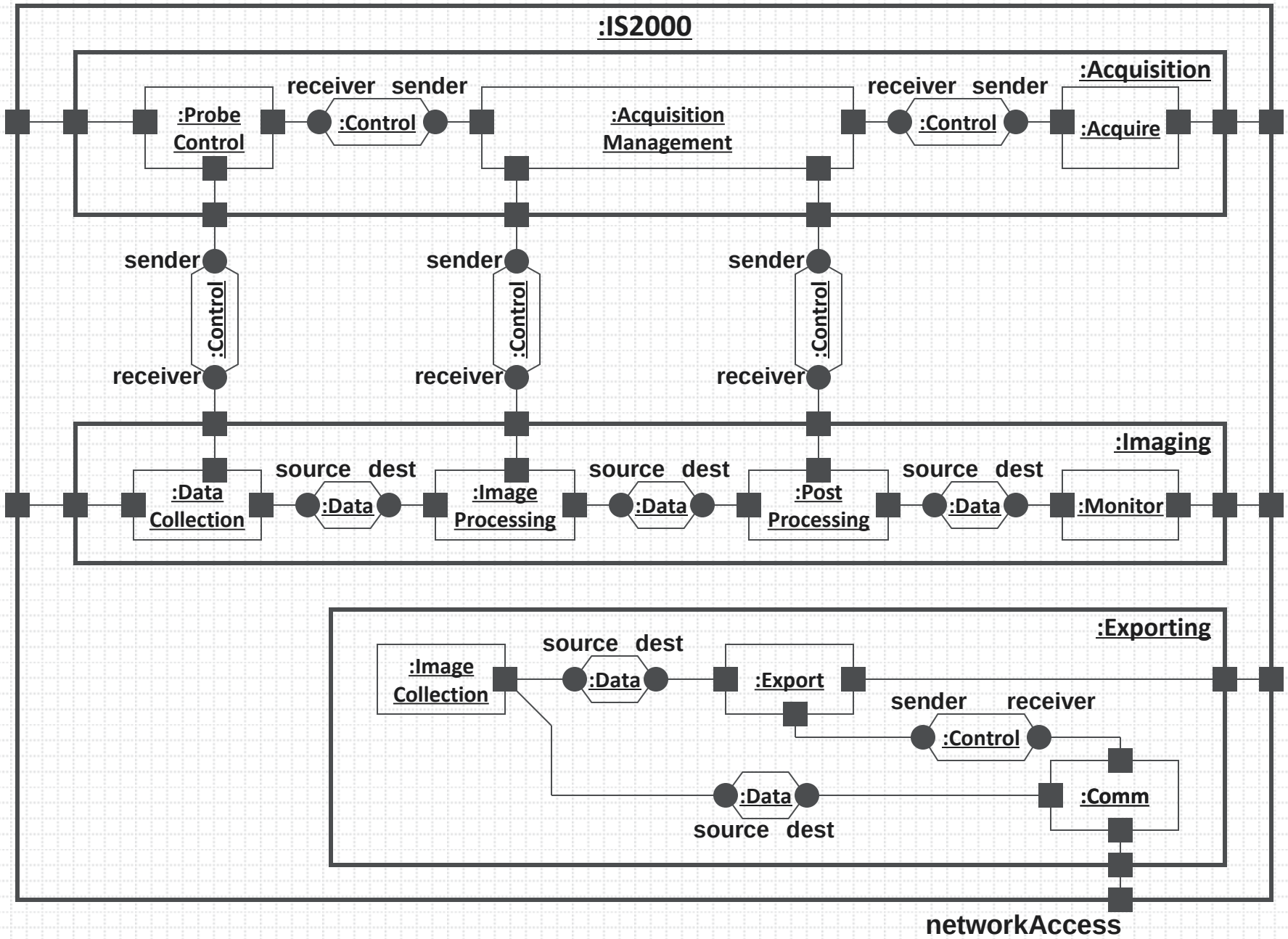
Step 1:Central Design – Connector of :*Acquisition, :Imaging*



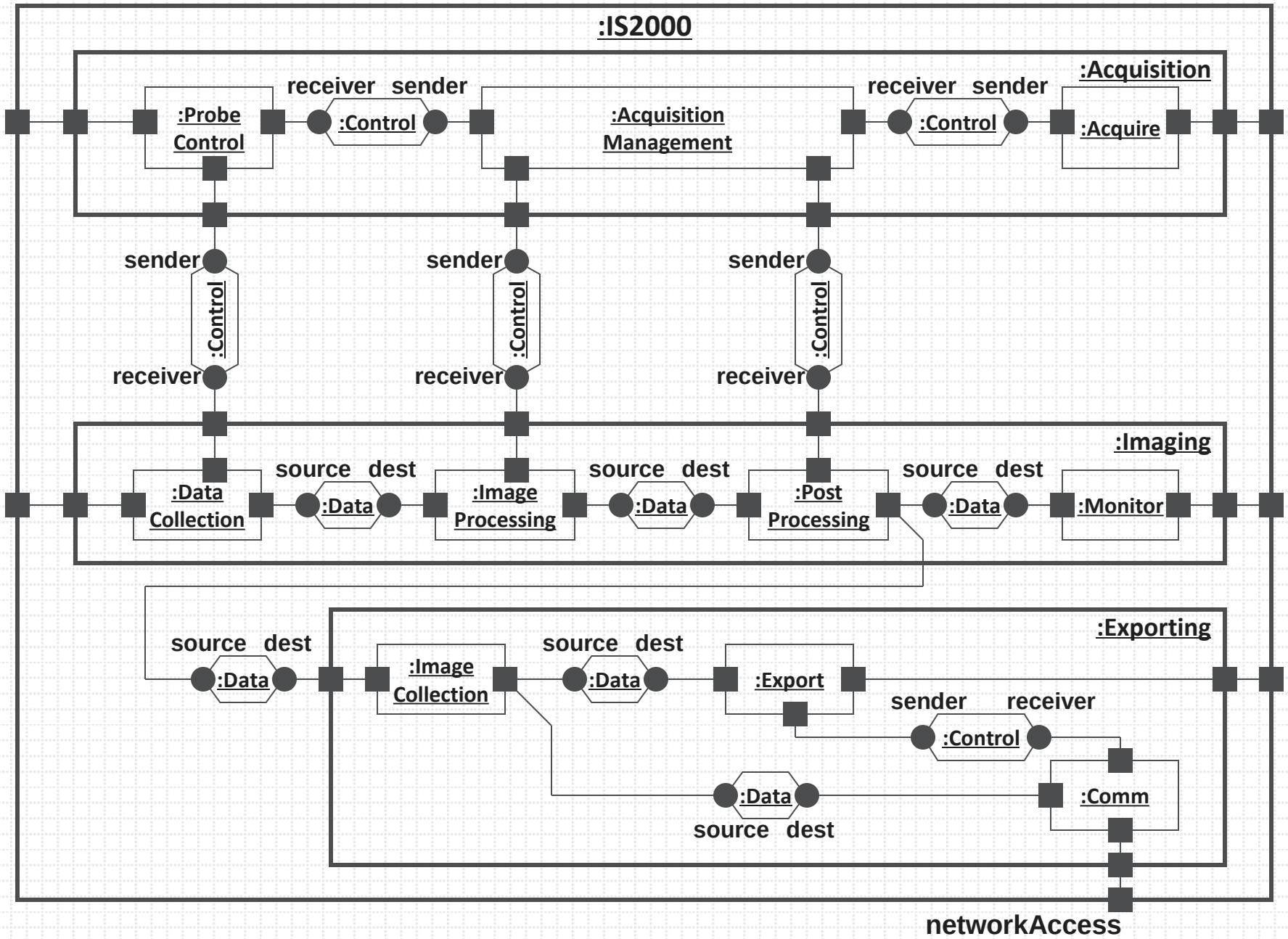
Step 1:Central Design – Connector of :*Exporting*



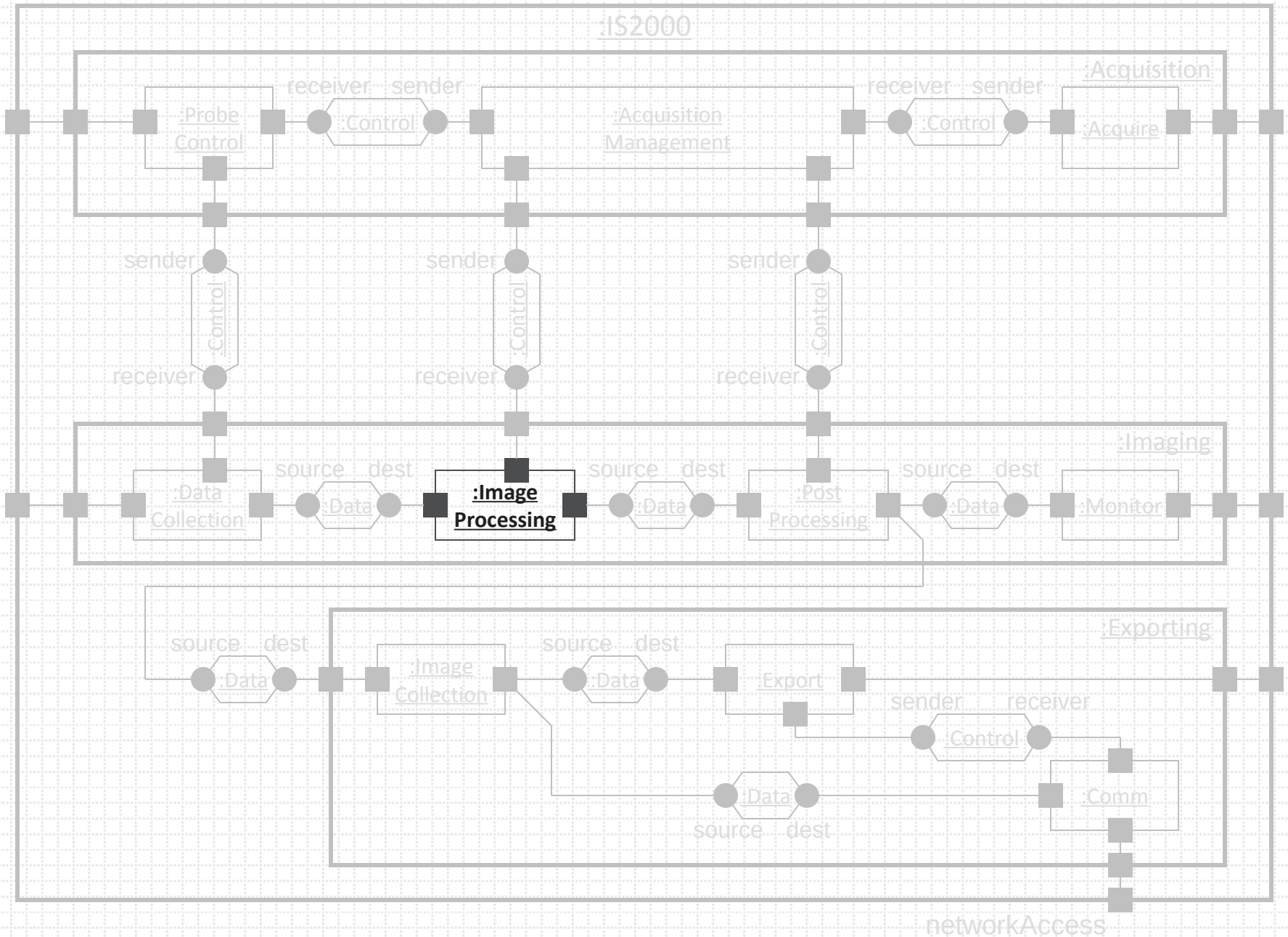
Step 1:Central Design – Conceptual Model(细化的)



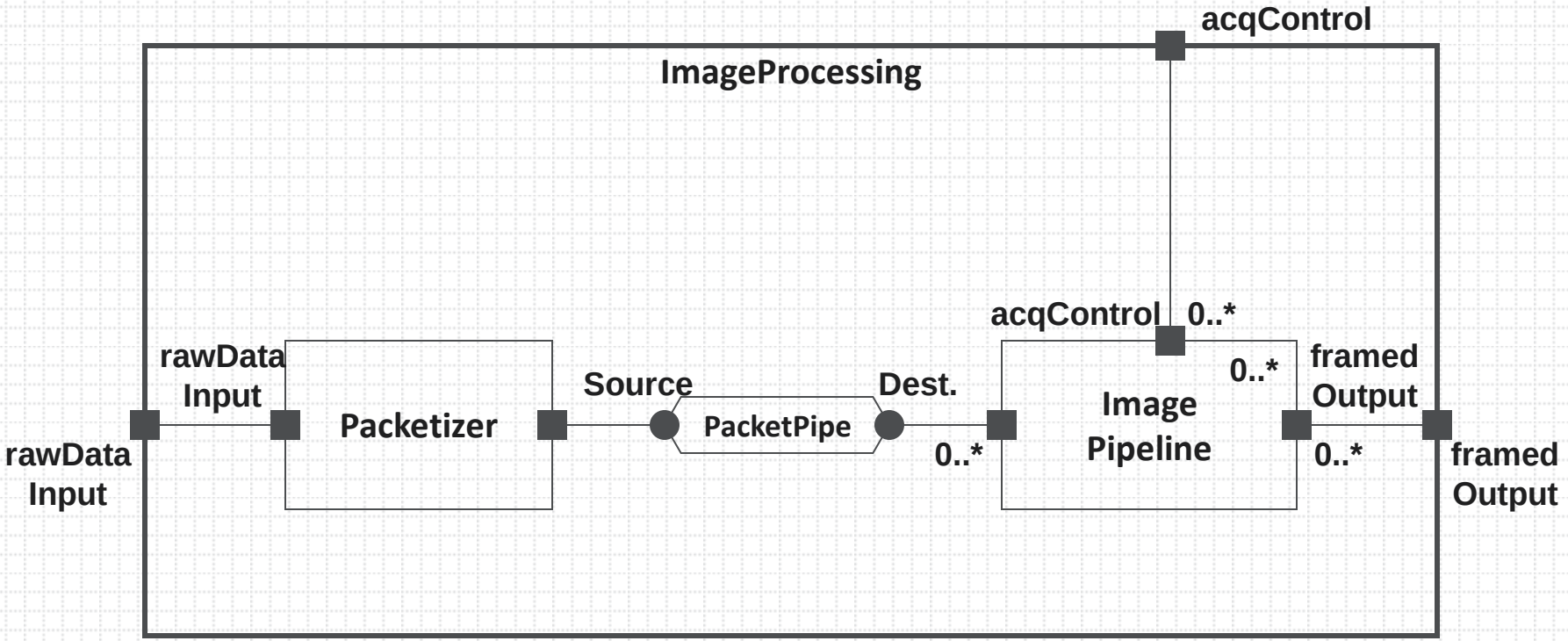
Step 1:Central Design – Conceptual Model(细化的)



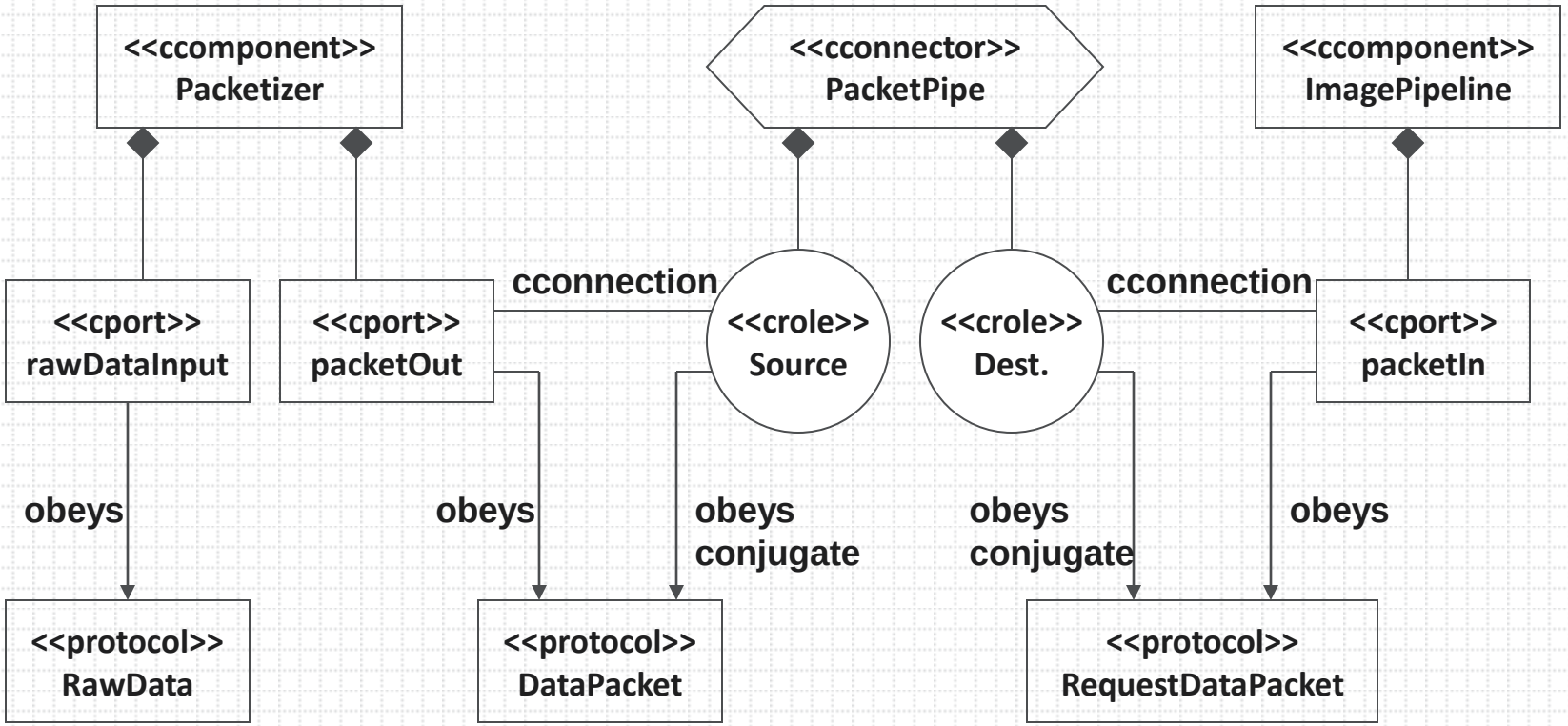
Step 1:Central Design – Conceptual Model(细化的)



Step 1: Central Design —进一步细化
Image processing decomposition

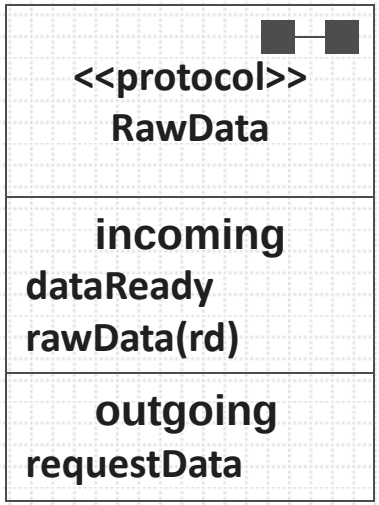


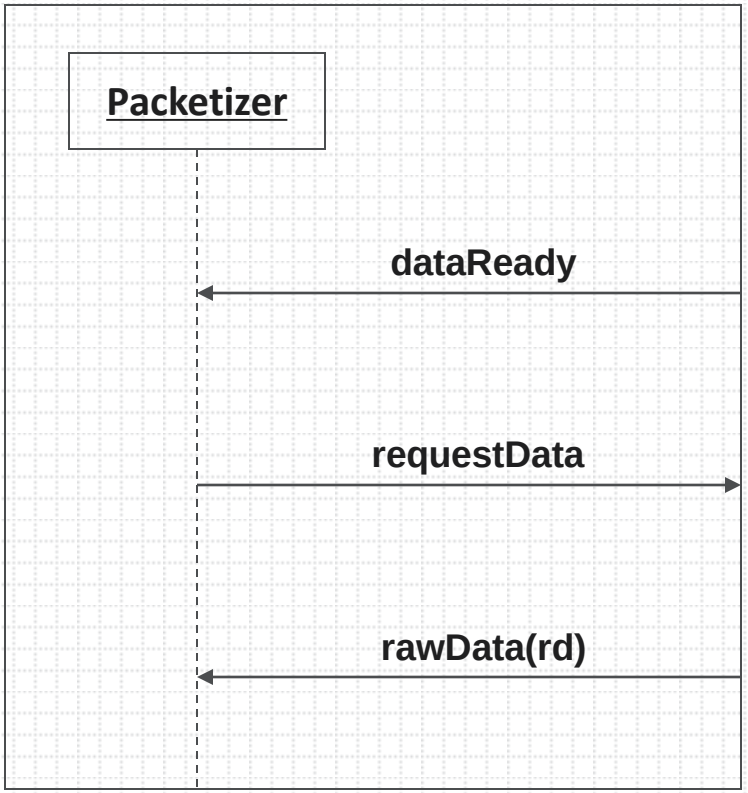
Step 1: Central Design – Configuration Design
Protocols for Packetizer and PacketPipe



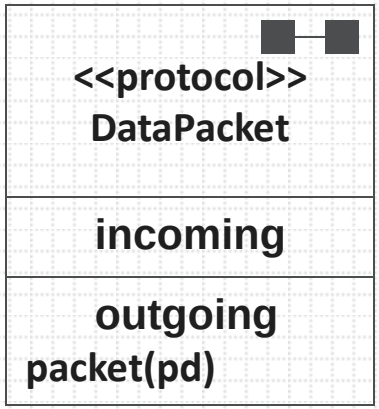
Step 1: Central Design – Configuration Design

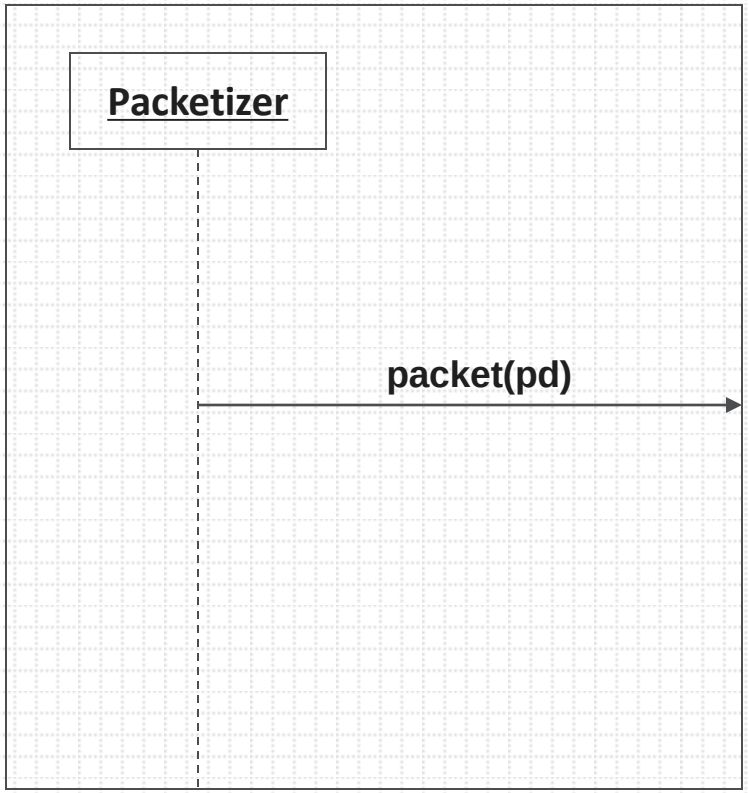
RawData protocol



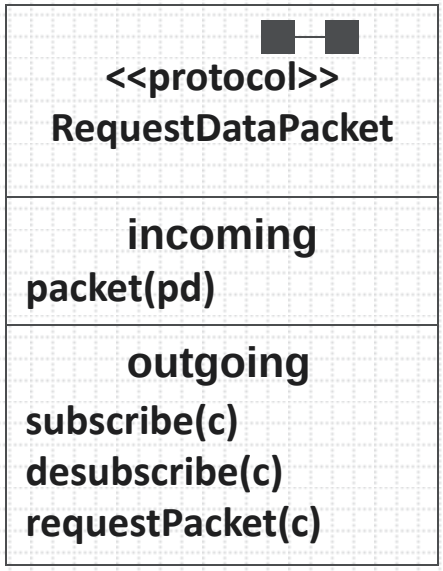


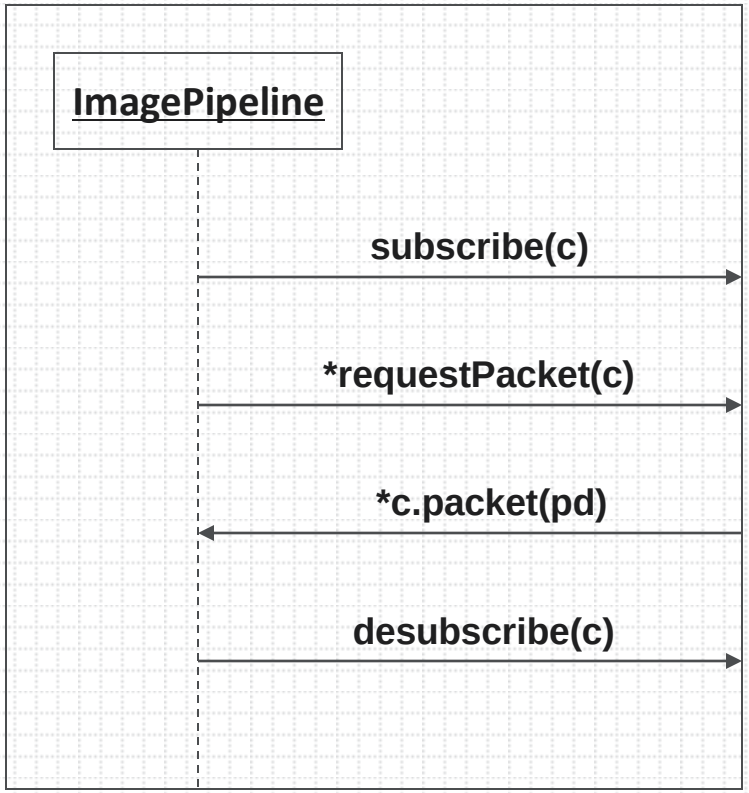
Step 1: Central Design– Configuration Design
DataPacket protocol





Step 1: Central Design – Configuration Design
RequestDataPacket protocol

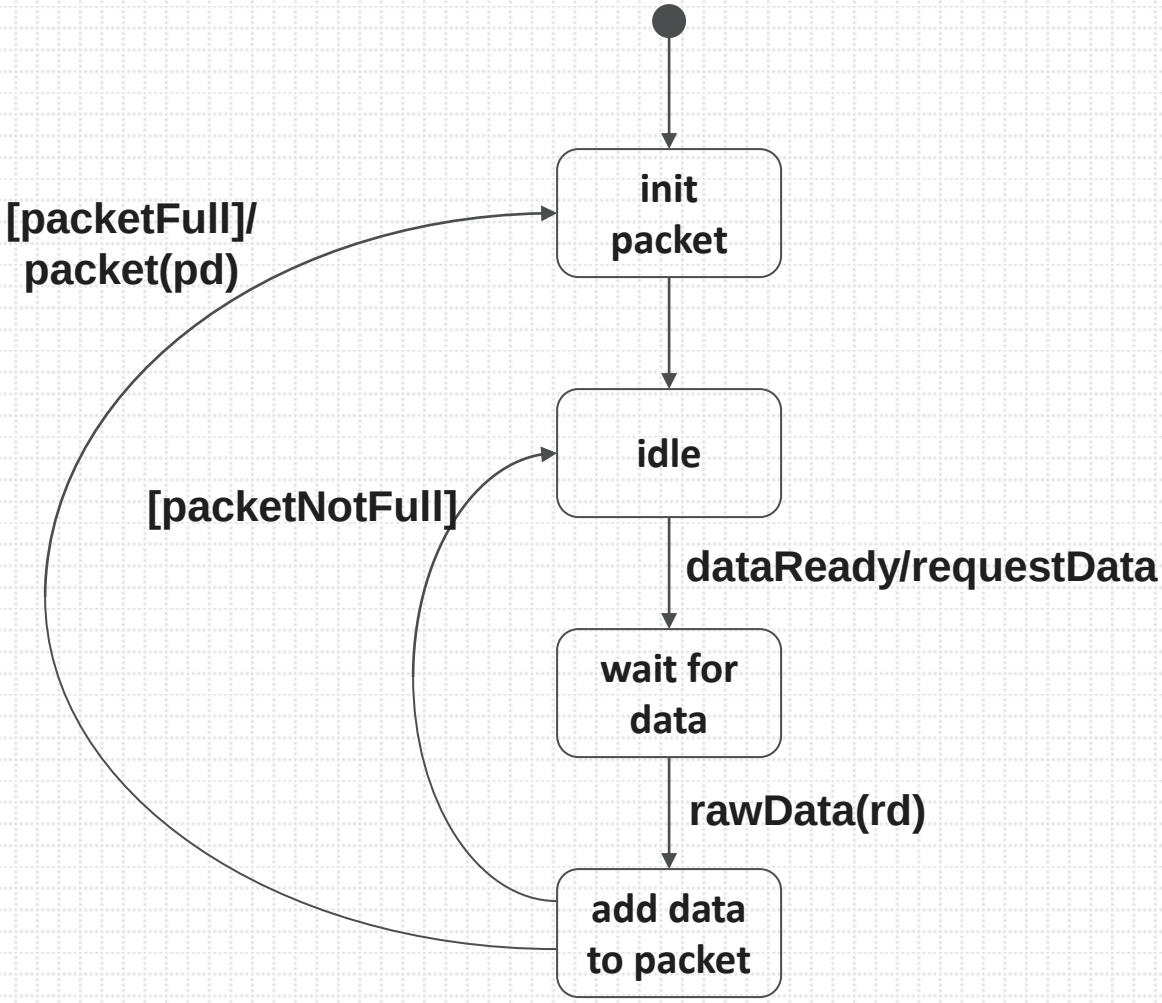




{The iteration is over one requestPacket followed by one c.packet.}

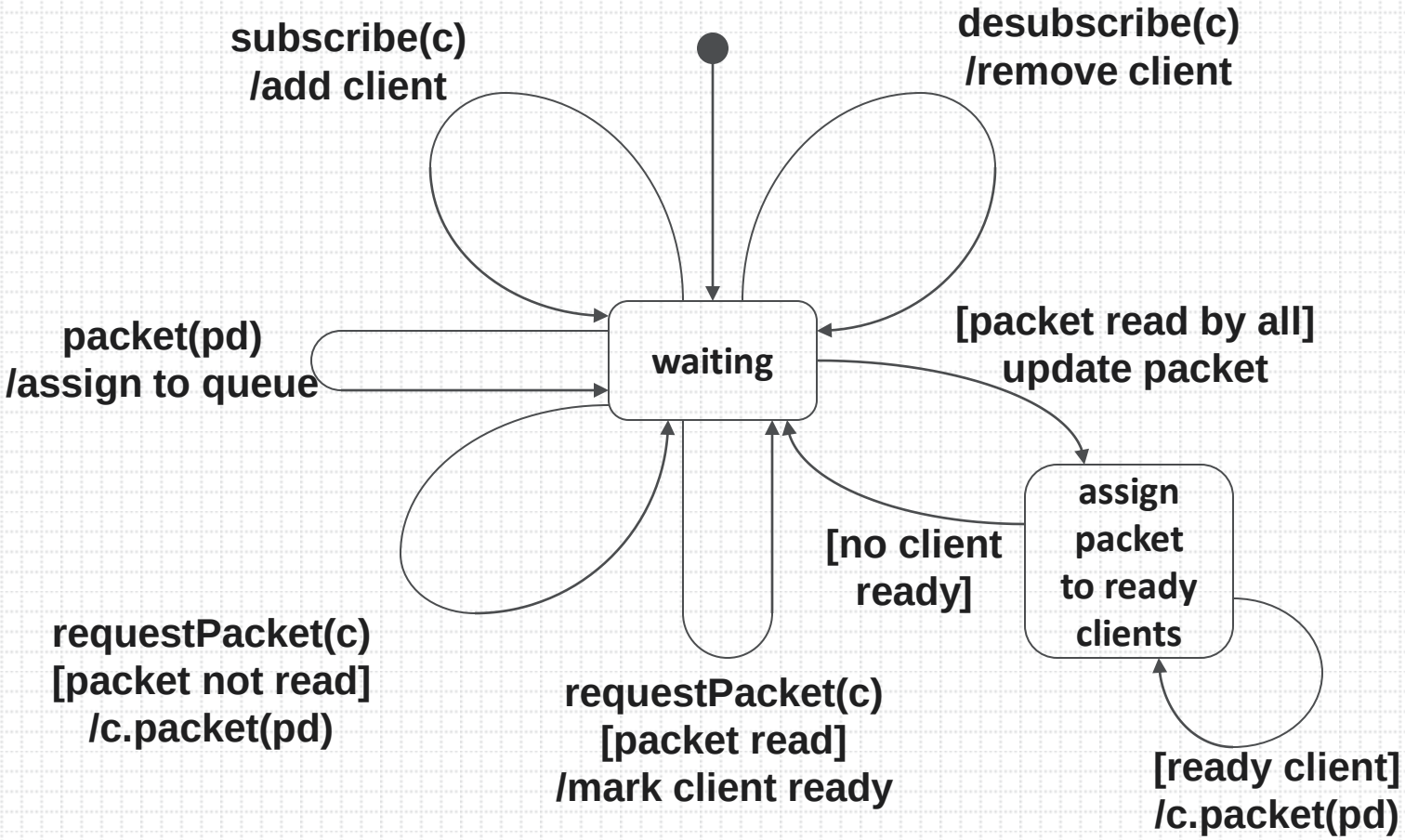
Step 1: Central Design – Configuration Design

Packetizer behavior

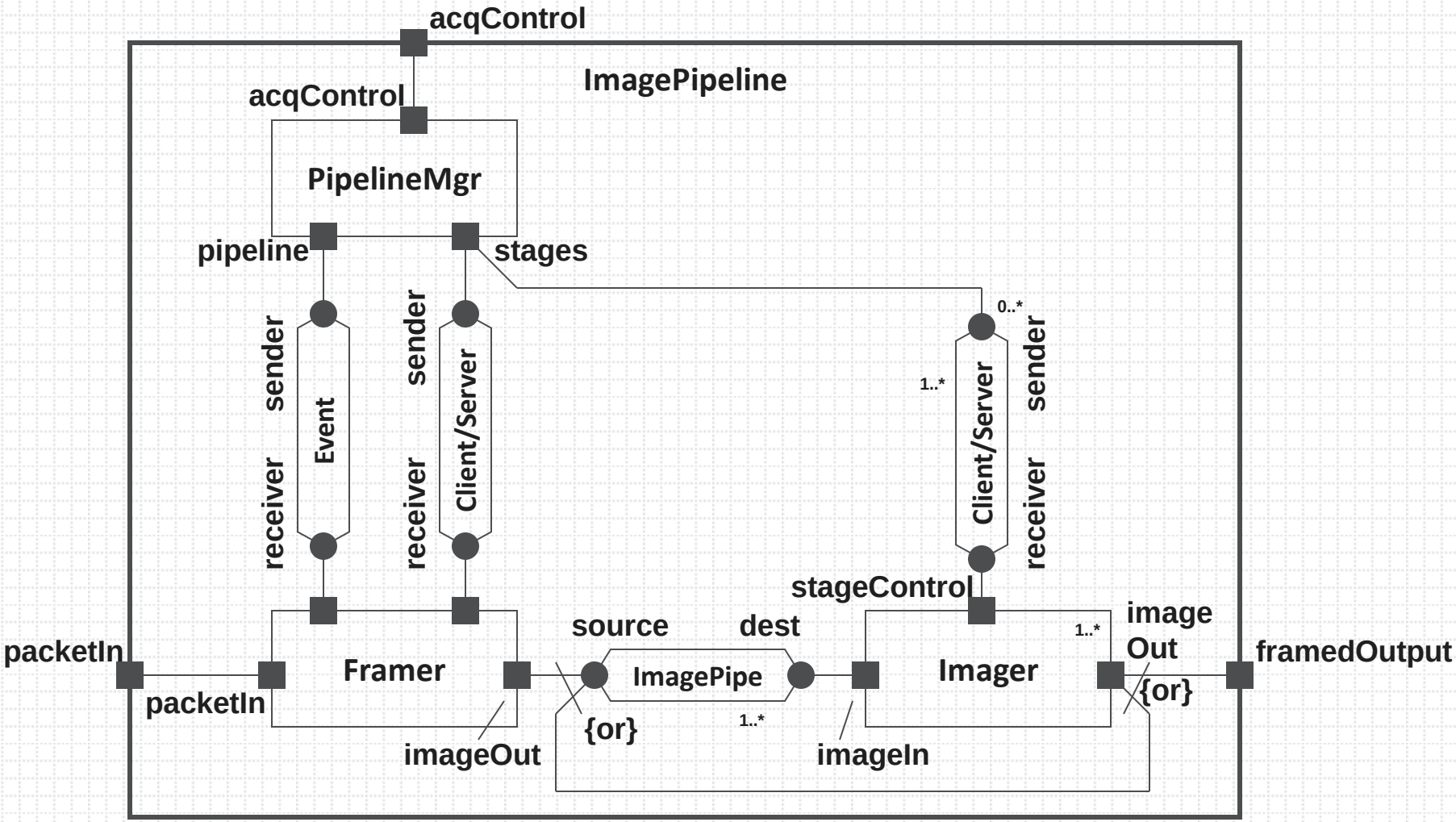


Step 1: Central Design – Configuration Design

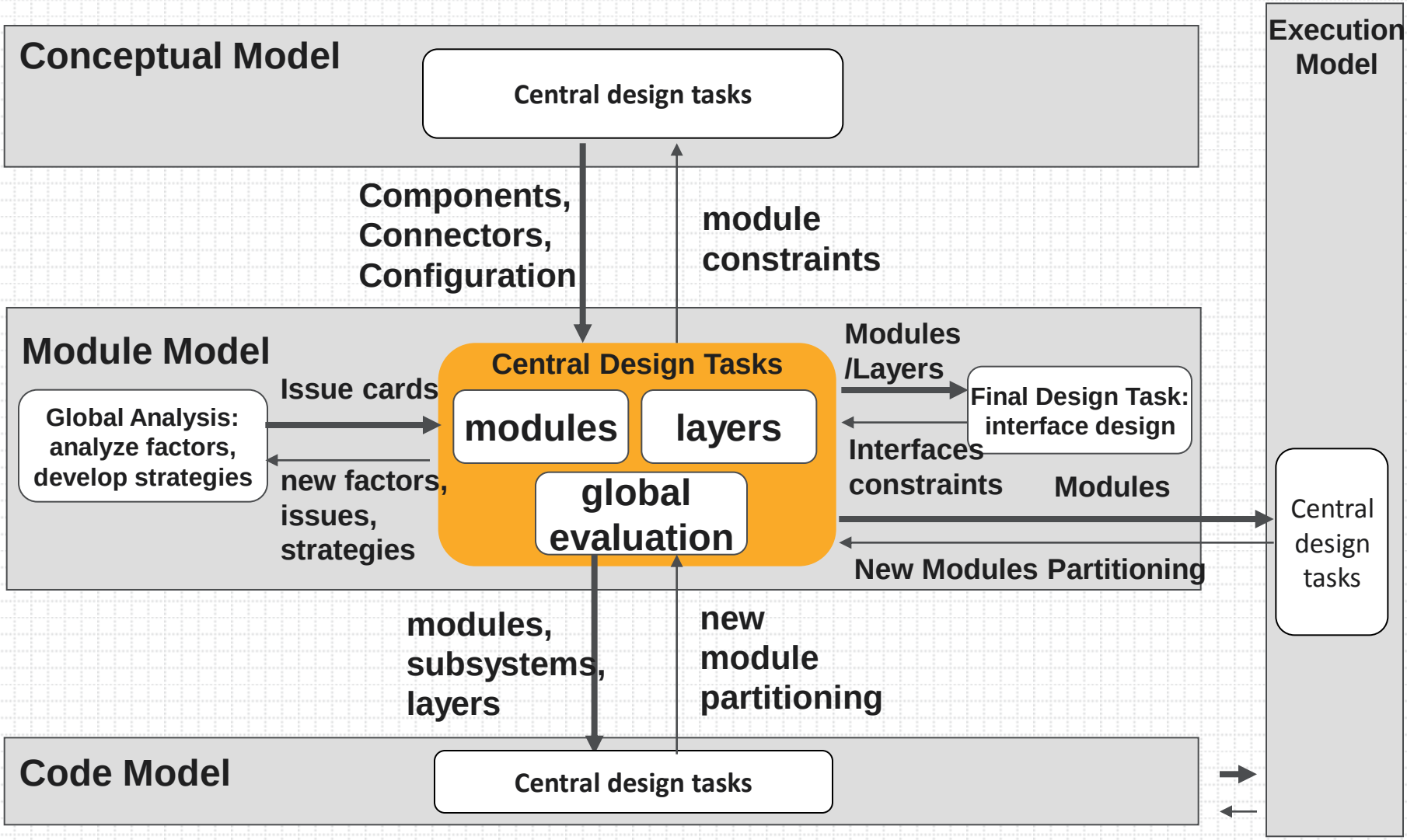
PacketPipe connector behavior



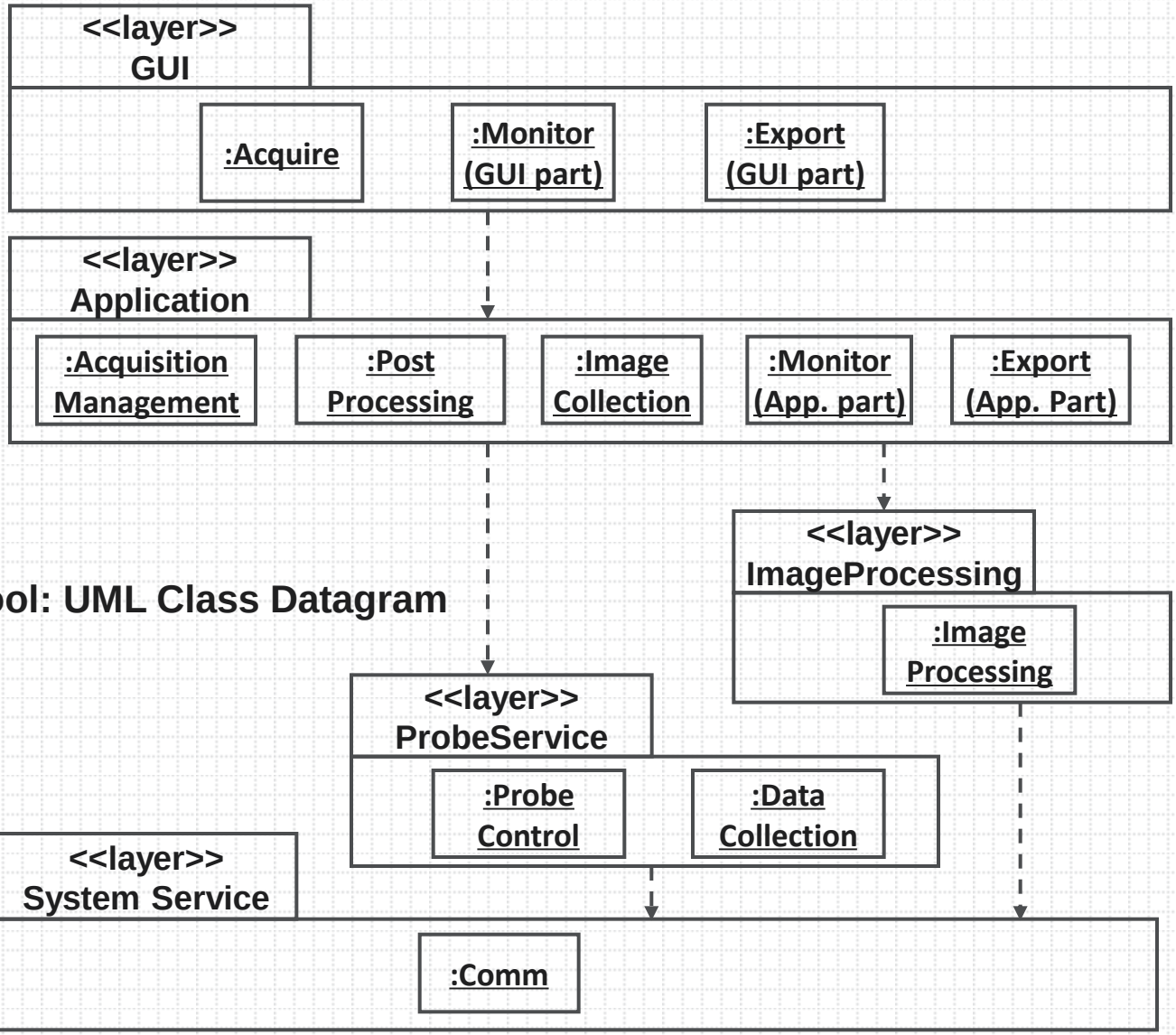
Step 1: Central Design —进一步细化
ImagePipeline decomposition



Step2 : Central Design – Module Model

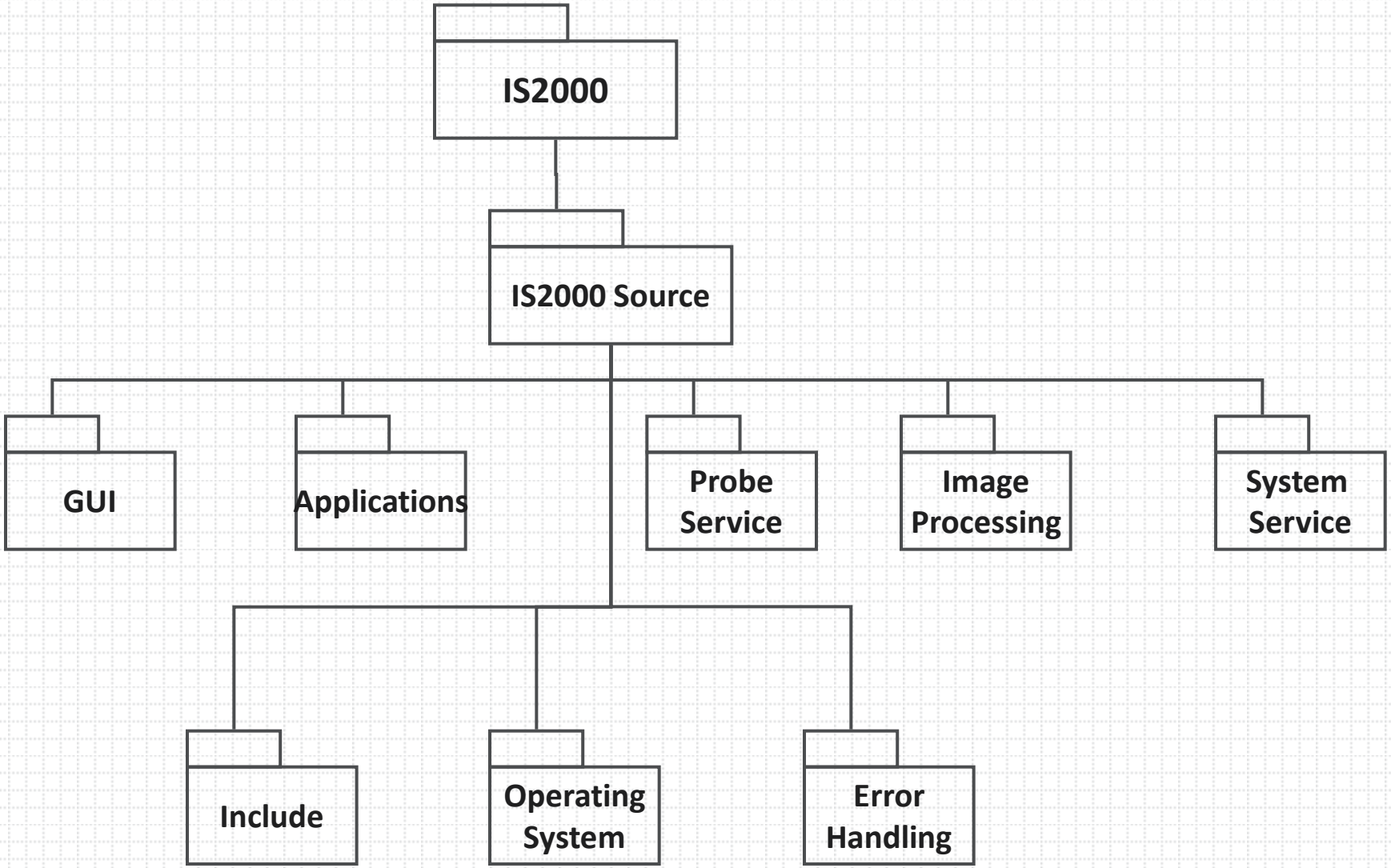


Step2：Central Design – Module Model

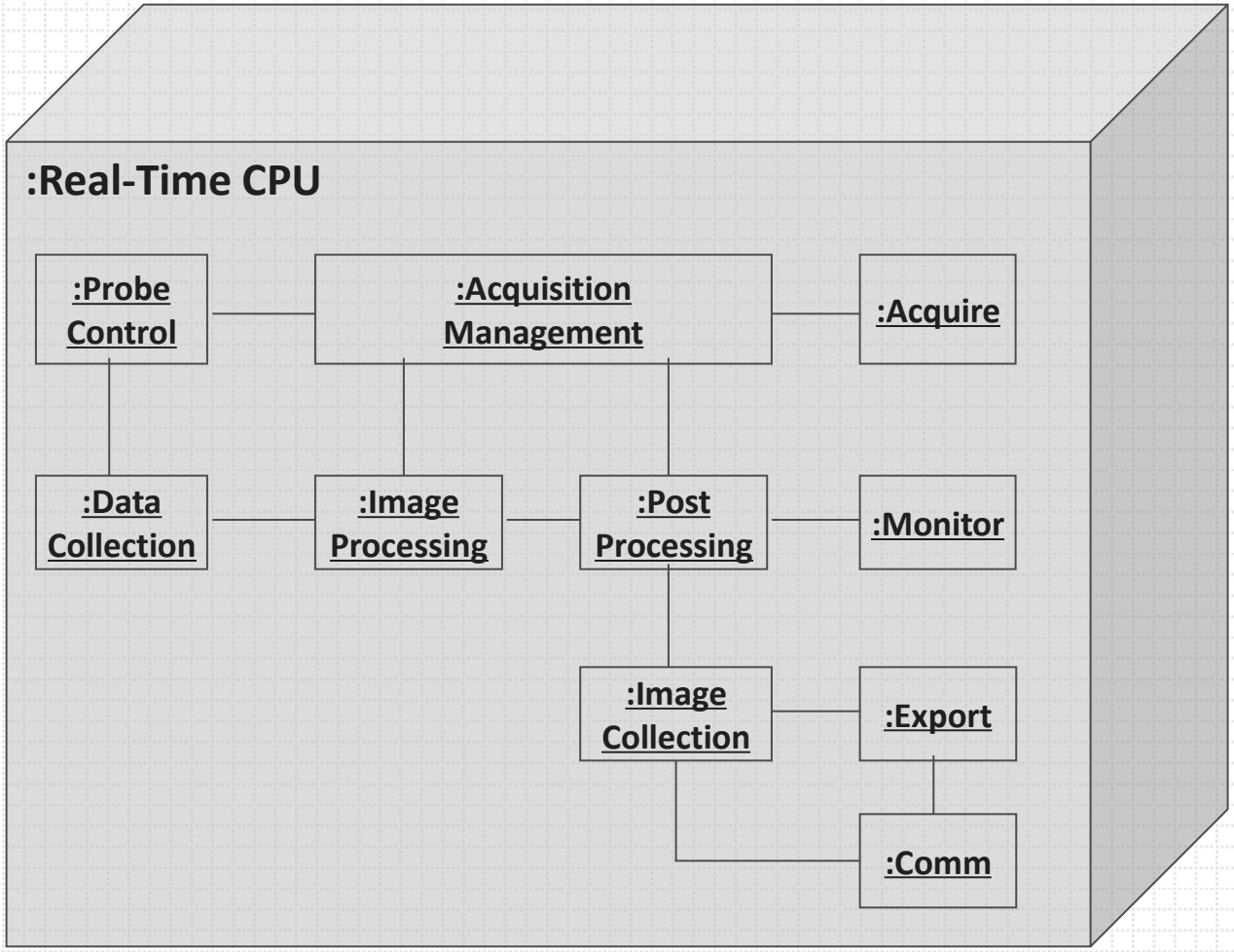


Modeling Tool: UML Class Datagram

Step3 : Central Design – Code Model

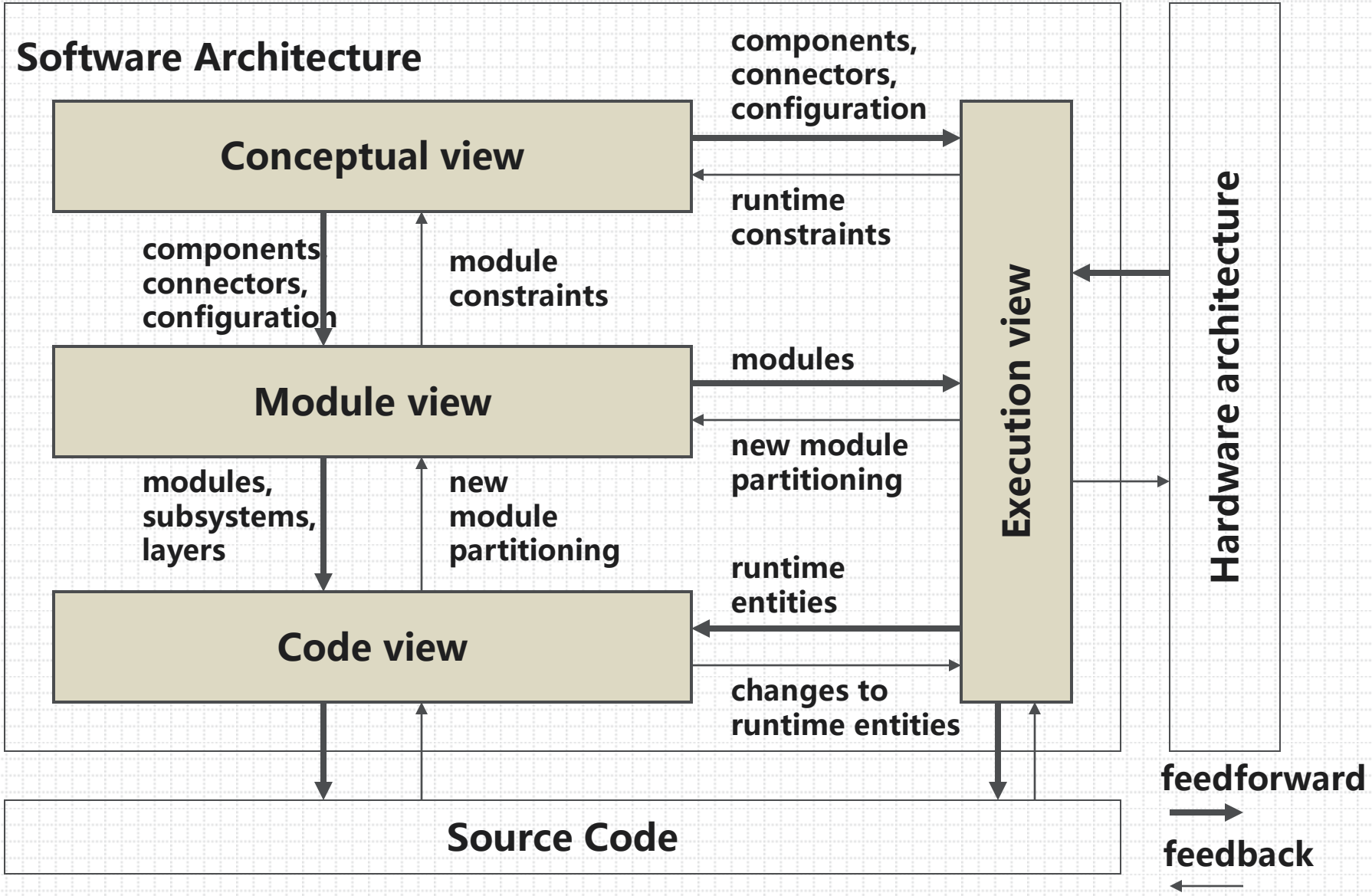


Step4：Central Design – Execution Model

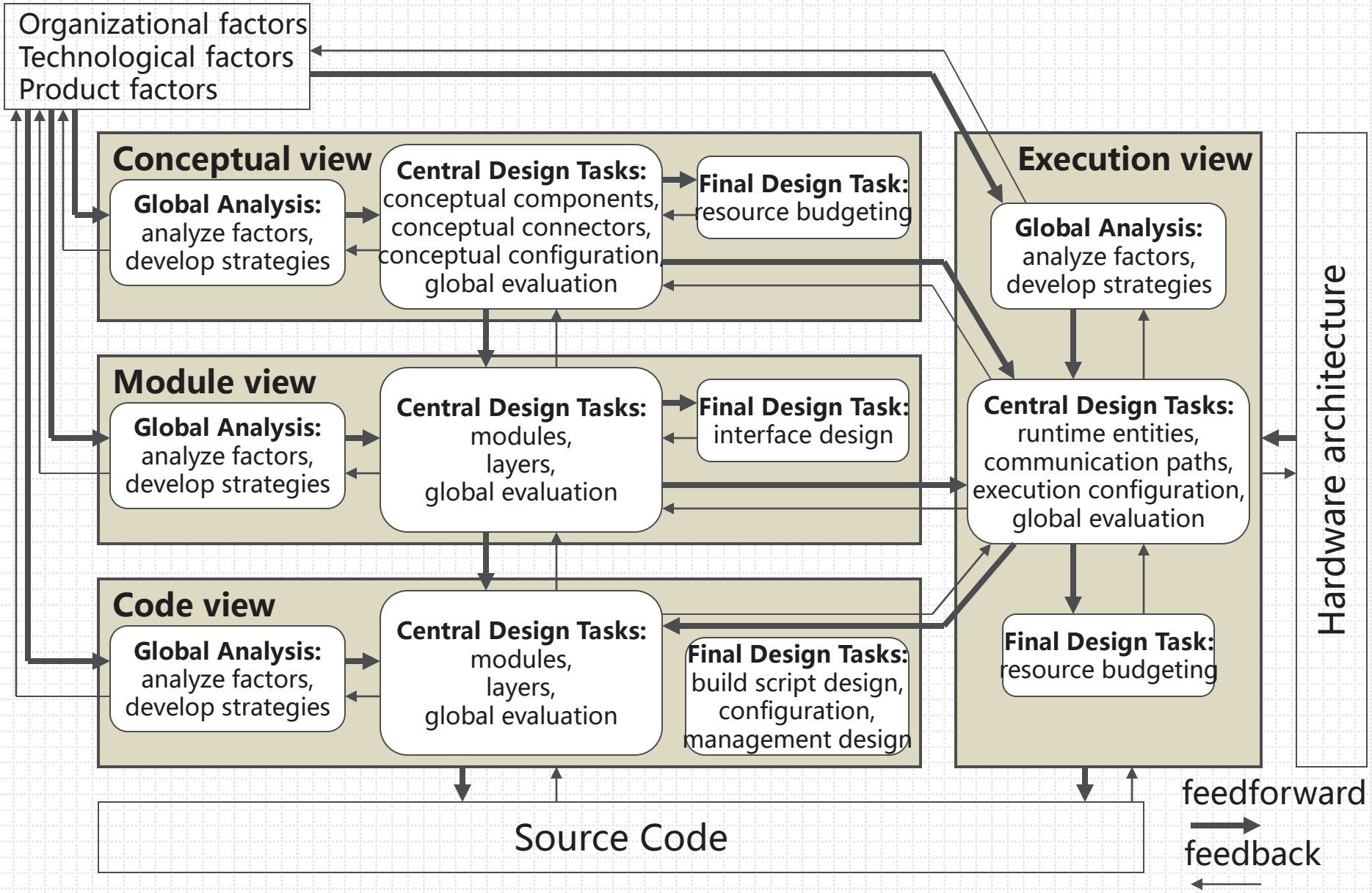


Modeling Tool: UML Deploy Datagram

Four views of a system



Process Review



2016

End