A02-2. *Conceptual View*

2014

Sungwon Kang

A. Purpose

- Closest to application domain
  <= least constrained by the software and the hardware platforms
- Model the system as a collection of decomposable, interconnected components and connectors
  - Components are independently executing peers
    => must unbundle the communication and control aspects and putting them in connectors
    => Conceptual view should keep the control aspects of the components simple and to isolate control in the connectors

Abstraction from control

- Quality Attributes to consider
  - Yes: performance (but must deal with it again in execution view)
  - No: portability => module view issue
- When system's behavior is captured by use cases and scenarios, the conceptual view should handle all of them satisfactorily.
  (<– Compare with our 8 viewpoints framework)

B. Context

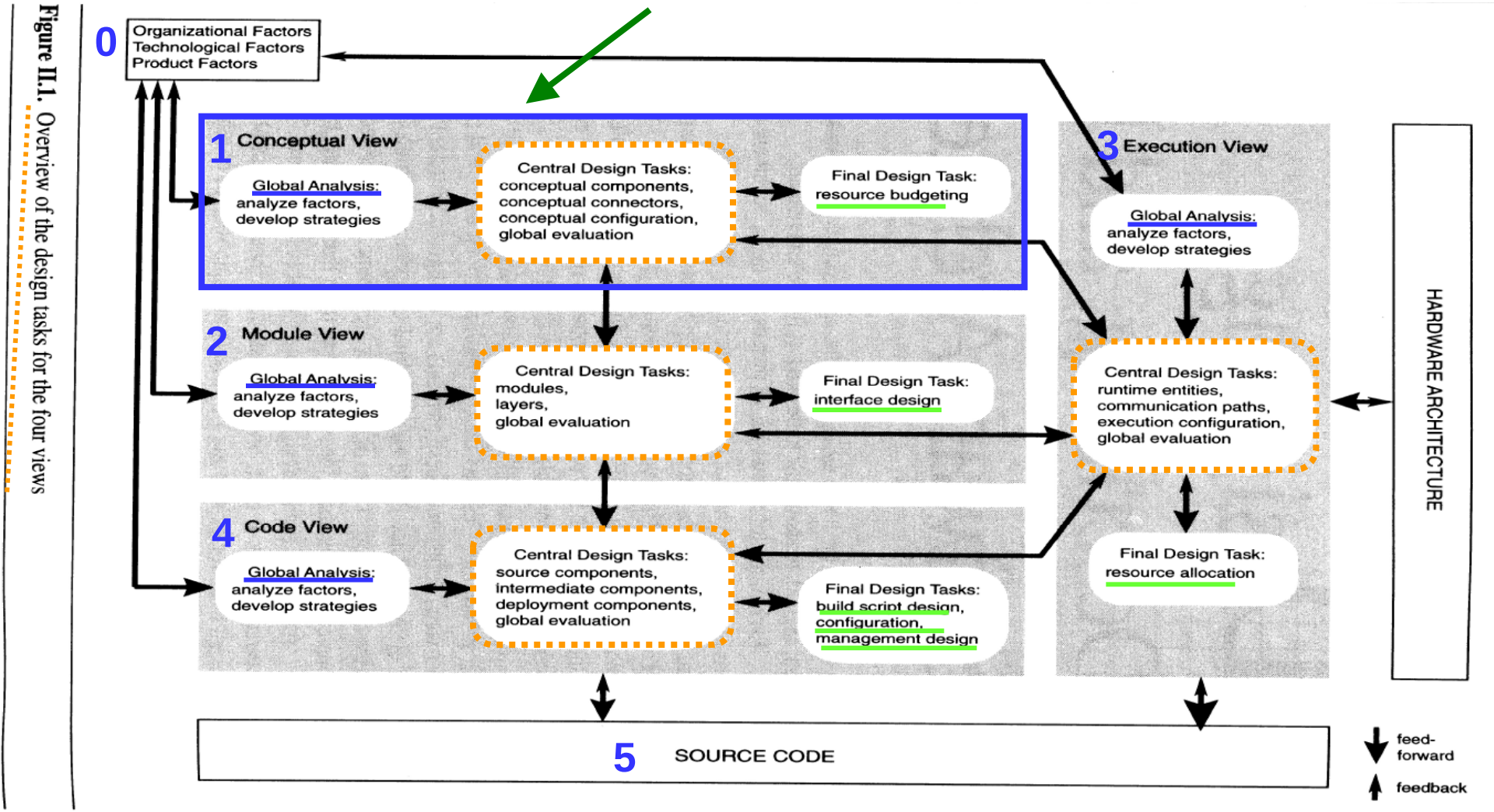


Figure II.1. Overview of the design tasks for the four views

C. Global Analysis

- Product factors: look at all of them
- Technological factors:
  - domain-specific hardware
  - architecture technology
  - domain-specific standards
- Organizational factors:
  - management
  - schedule
  - budget
- New factors may come up during the central design tasks.

D. Central Design Tasks

1. Define component and connector types. **Why?** For reusability
2. Define how component and connector types interconnect.
3. Map system functionality to components and connectors **Why?**
   - Concentrate functional behavior in the components
   - Concentrate control aspects in the connectors

   For buildability
   reusability
   evolvability
4. Define the instances of the component and connector types that exist in the product, and how they are interconnected.

"*The order in which you do these things is not fixed.*

. . .

You will likely repeat many of theses activities as you refine the design."

E. Final Design Tasks

- Budget Resources for the components and connectors instances
  - Critical application-level resources that are limited or to be shared

  (Can be called "frame conformance". So far focus was on solution, not on whether that solution fits the frame.)

  E.g. - memory to components involved in data-intensive activities
  - communication bandwidth
  - computation time budgets to time-critical components

[1] Global Analysis

Which issues would you like to deal with first?

I1. Aggressive schedule

I3. Changes in General Purpose and Domain Specific Hardware

I6. Easy Addition and Removal of Features

The following one is a specialization of I6.

Issue 7

Easy Addition and Removal of Acquisition Procedures and Processing Algorithms

There are many acquisition procedures and processing algorithms. Implementation of each of these features is quite complex and time-consuming. There is a need to reduce complexity and effort in implementing such features.

Influencing Factors

O4.1: Time-to-market is short.

O4.2: Delivery of features is negotiable.

P1.1: New acquisition procedures can be added every three years.

P1.2: New image-processing algorithms can be added on a regular basis.

P1.3: New types of image and signal data may be required on a regular basis as the probe hardware changes.

Solution

Define domain-specific abstractions to facilitate the task of implementing acquisition and processing applications.

Strategy: *Use a flexible pipeline model for image processing.* S7A

Develop a flexible pipeline model for implementing image processing. Use processing components as stages in the pipeline. This allows the ability to introduce new acquisition procedures quickly by constructing pipelines using both old and new components.

Strategy: *Introduce components for acquisition and image processing.* S7B

Develop components for domain-specific acquisition and image processing to minimize the effects of any changes to the application domain. Developing a component model for domain-specific processing makes it easier to add or to upgrade components that are more efficient, and to offer new features.

Strategy: *Encapsulate domain-specific image data.* S7C

To add or to upgrade to improved image-processing techniques, introduce components for domain-specific processing. Thus we can take advantage of industry standards and vendor solutions for transporting data over the network.

Related Strategies

See also *Encapsulate domain-specific hardware* (issue, Changes in General-Purpose and Domain-Specific Hardware).

Issue 9

Real-Time Acquisition Performance

Meeting real-time performance requirements is critical to the success of the product. There is no separate source code for meeting the real-time performance requirements directly. The source code that implements functional processing must also meet the performance constraints.

Influencing Factors

T1, T3: A single standard processor running a real-time operating system is expected to provide sufficient computing resources.

P3.1, T2.1: The maximum signal data rate changes with changes in the probe hardware.

P3.2: Acquisition performance requirements are slightly flexible.

Solution

Partition the system into separate components for algorithms, communication, and control to provide the flexibility to implement several different strategies.

Strategy: ***Separate time-critical components from nontime-critical components.***

To isolate the effects of change in the performance requirements, partition the system into components (and modules) that participate in time-critical processing and those that do not. This requires careful consideration at the interface between the real-time and nonreal-time sides of the system.

Related Strategies

See also *Separate components and modules along dimensions of concern* (issue, Easy Addition and Removal of Features).

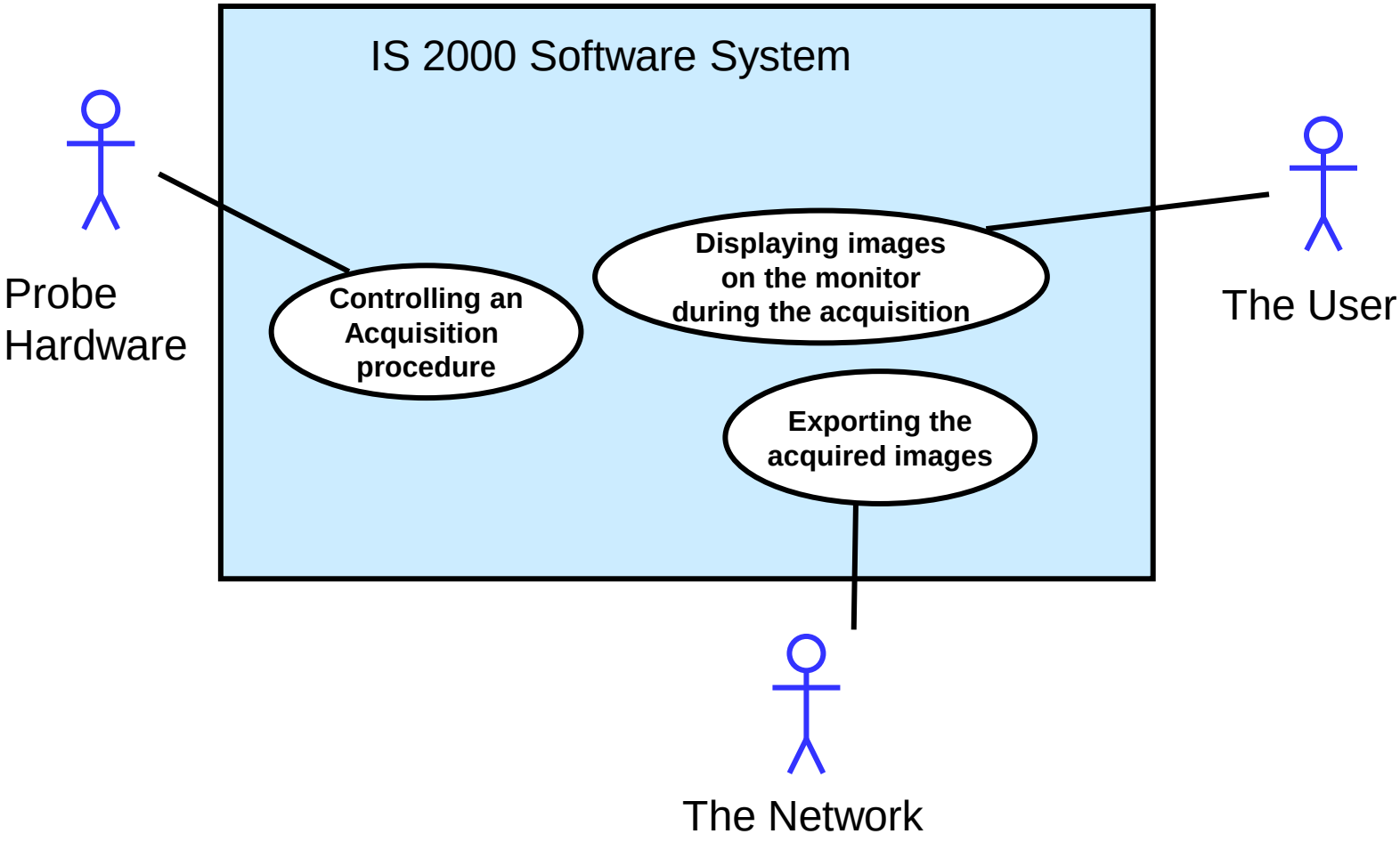
[2] Central Design Tasks

- The **context** of the system
  - **User**: controlling and acquisition procedure
  - **probe hardware**: displaying images on the monitor during acquisition, exporting the acquired images
  - **Network**: network access

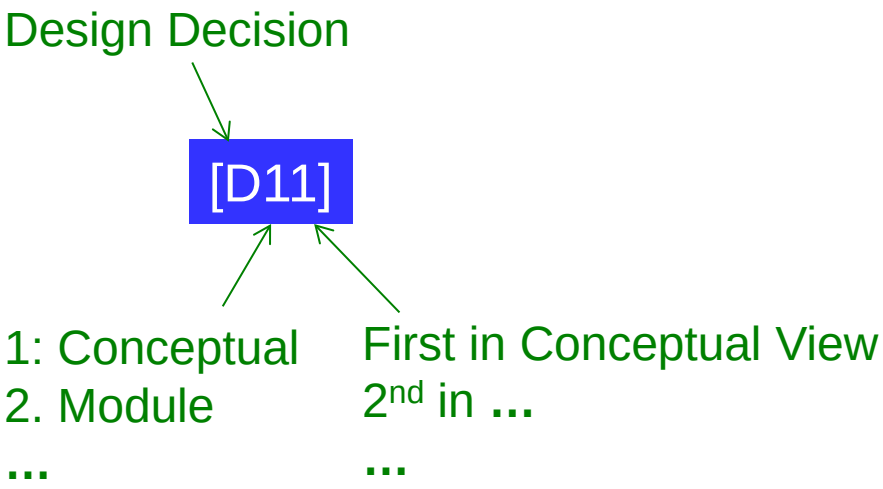
  => 3 ports

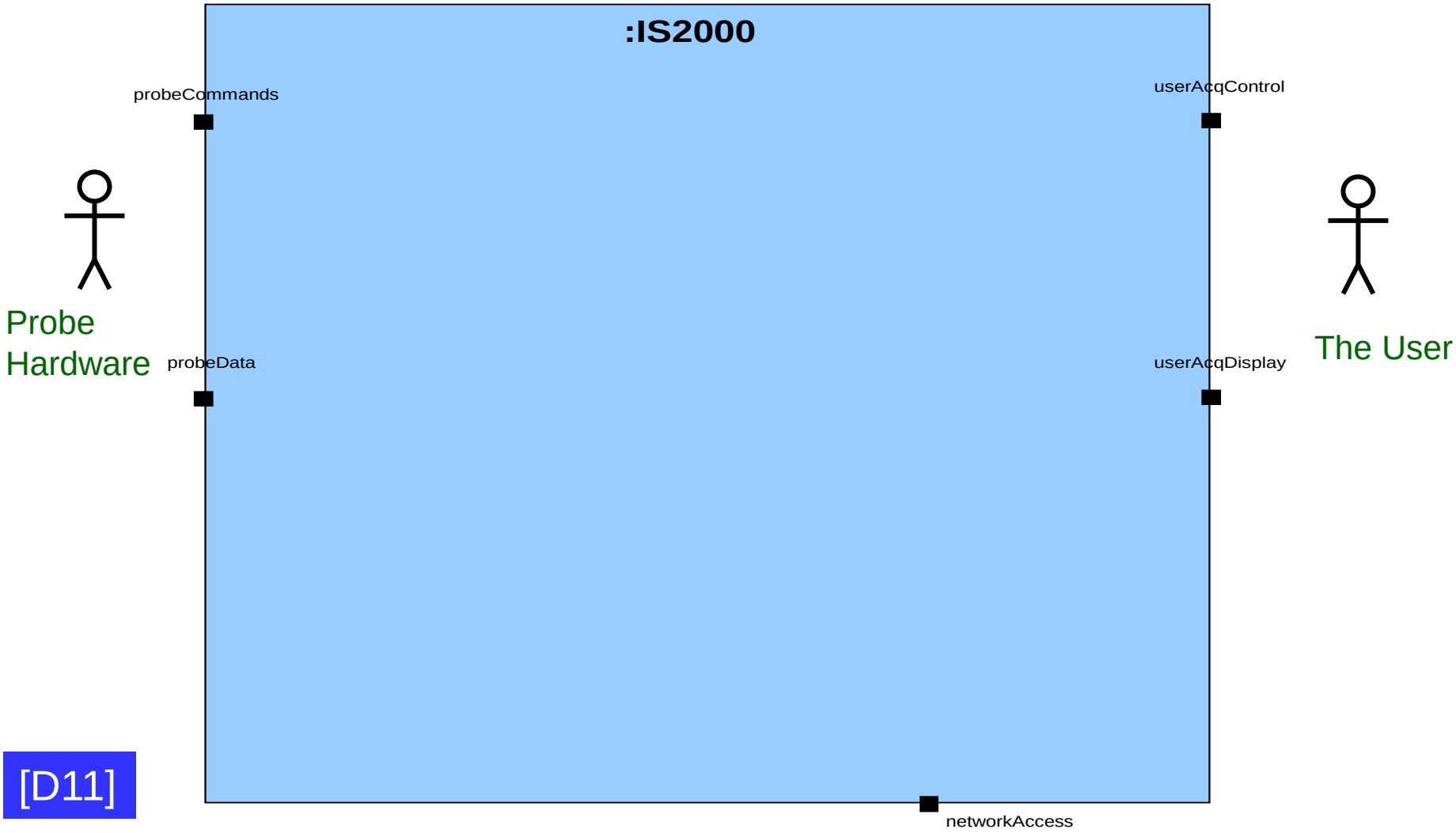
- *"S7B. Introduce components for acquisitions and image processing"*

Context Diagram



Design Decision Notation

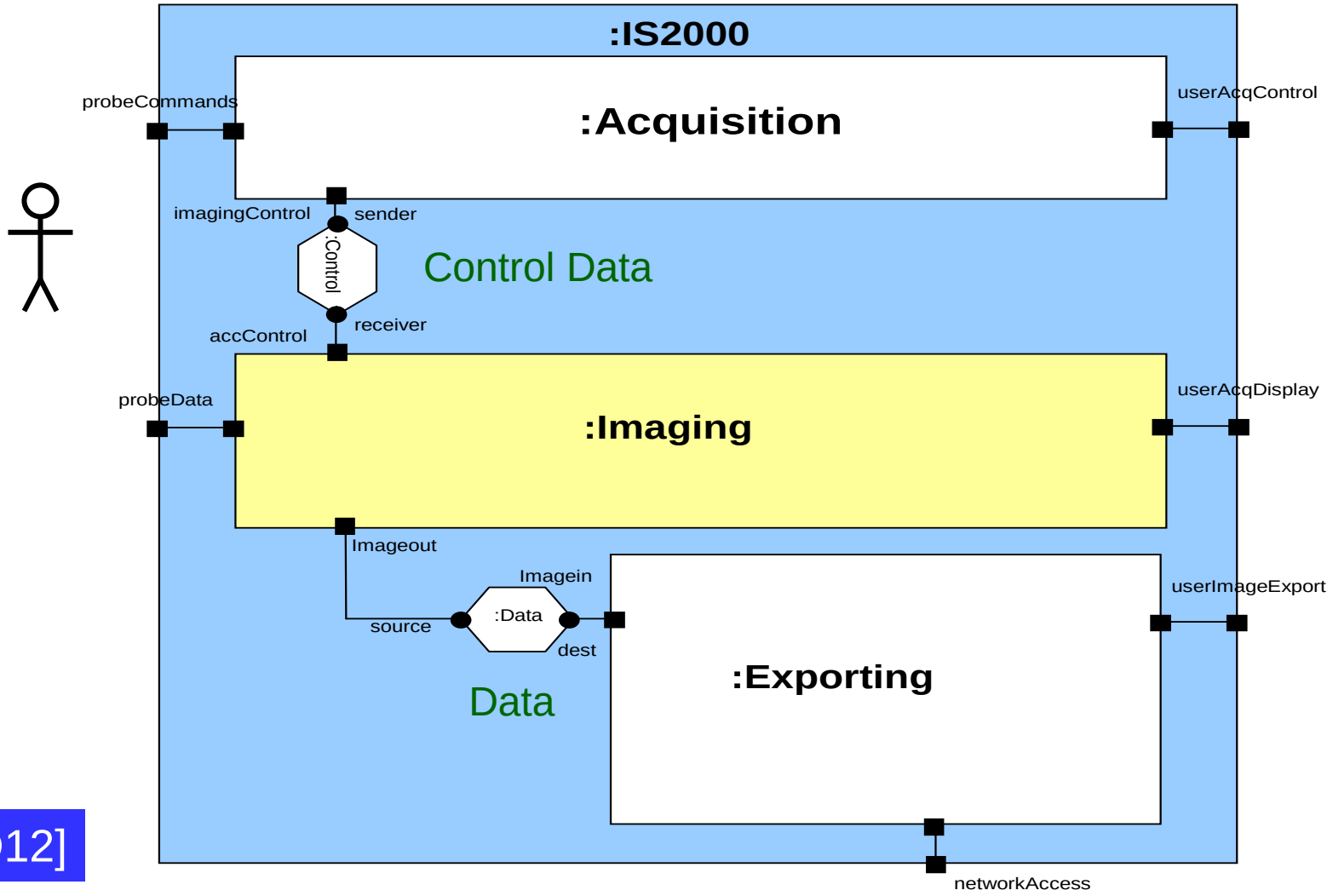




[D11]

Represent the IS 2000 as a component with ports for interactions with the user, the probe hardware and the network.

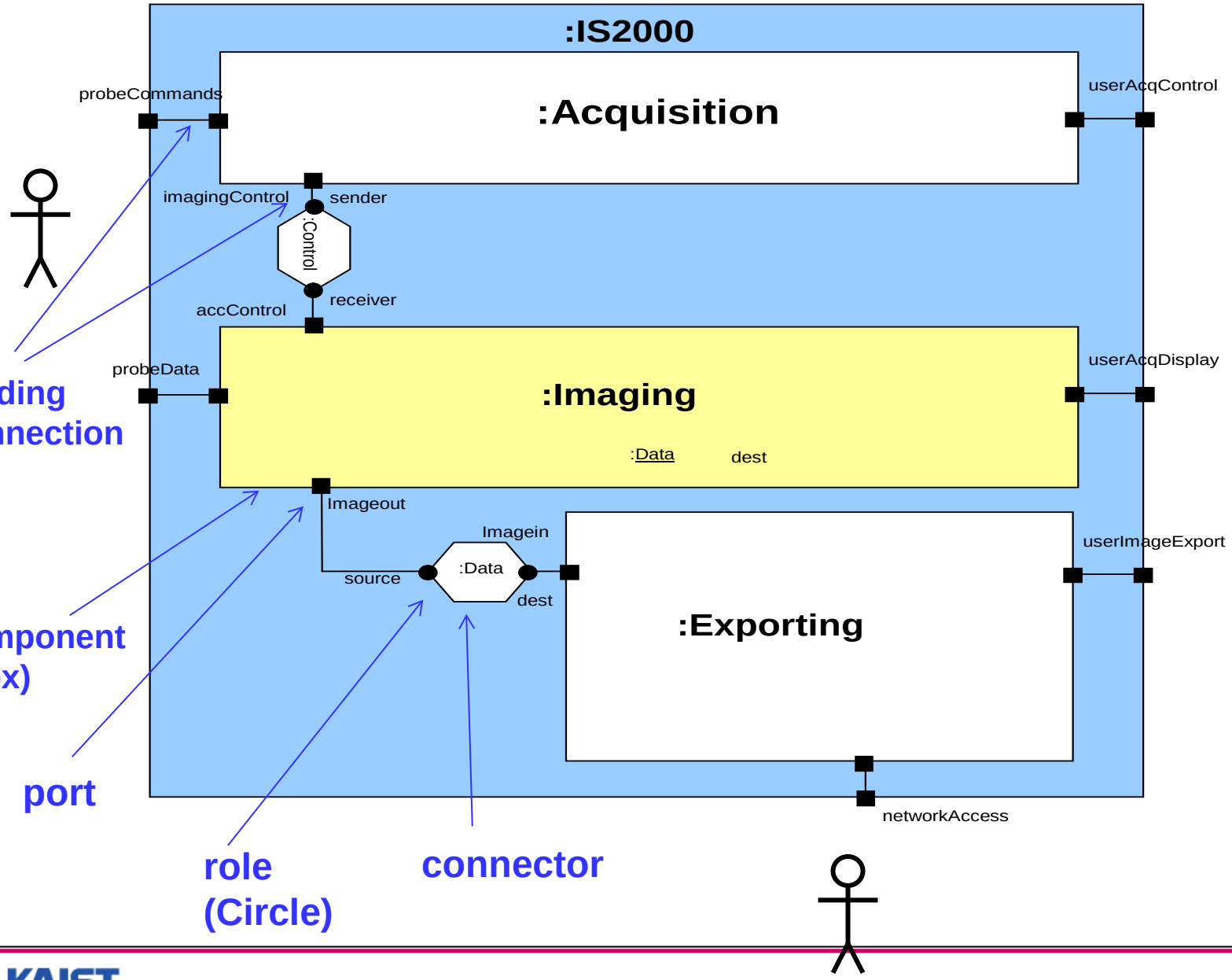
Fig 4.5 Initial high-level configuration of IS2000



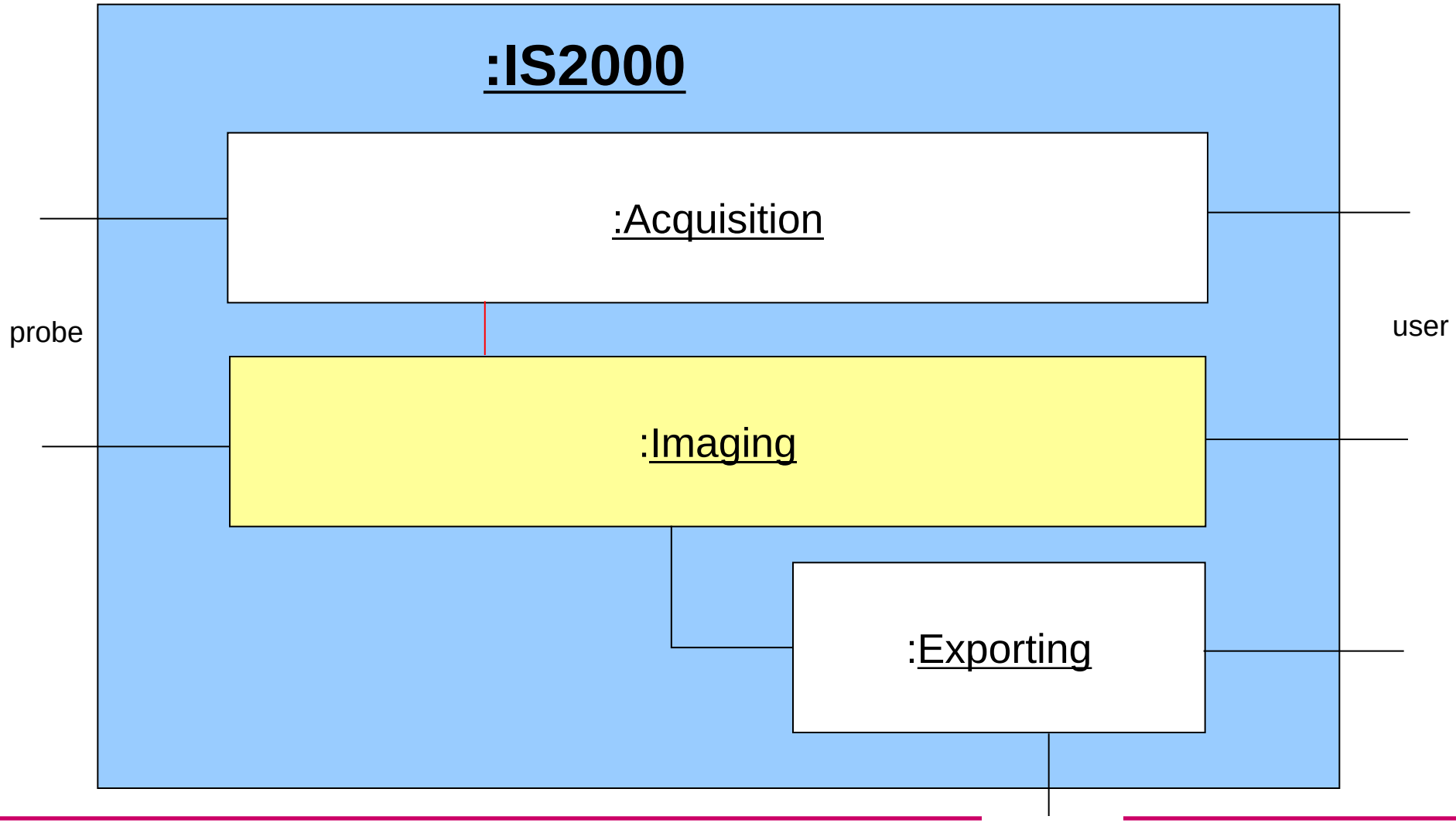
[D12]

S7B "Introduce components for acquisition and image processing"

Architecture Description Notation



Is this better than the previous one?



[D13]

- **S3A**: "*Encapsulate domain-specific hardware*"

  => Add ProbeControl, DataCollection components

  ☞ insulate the rest of the system for hardware changes and more reuse opportunities

  ⇒ Add connector from ProbeControl to DataCollection

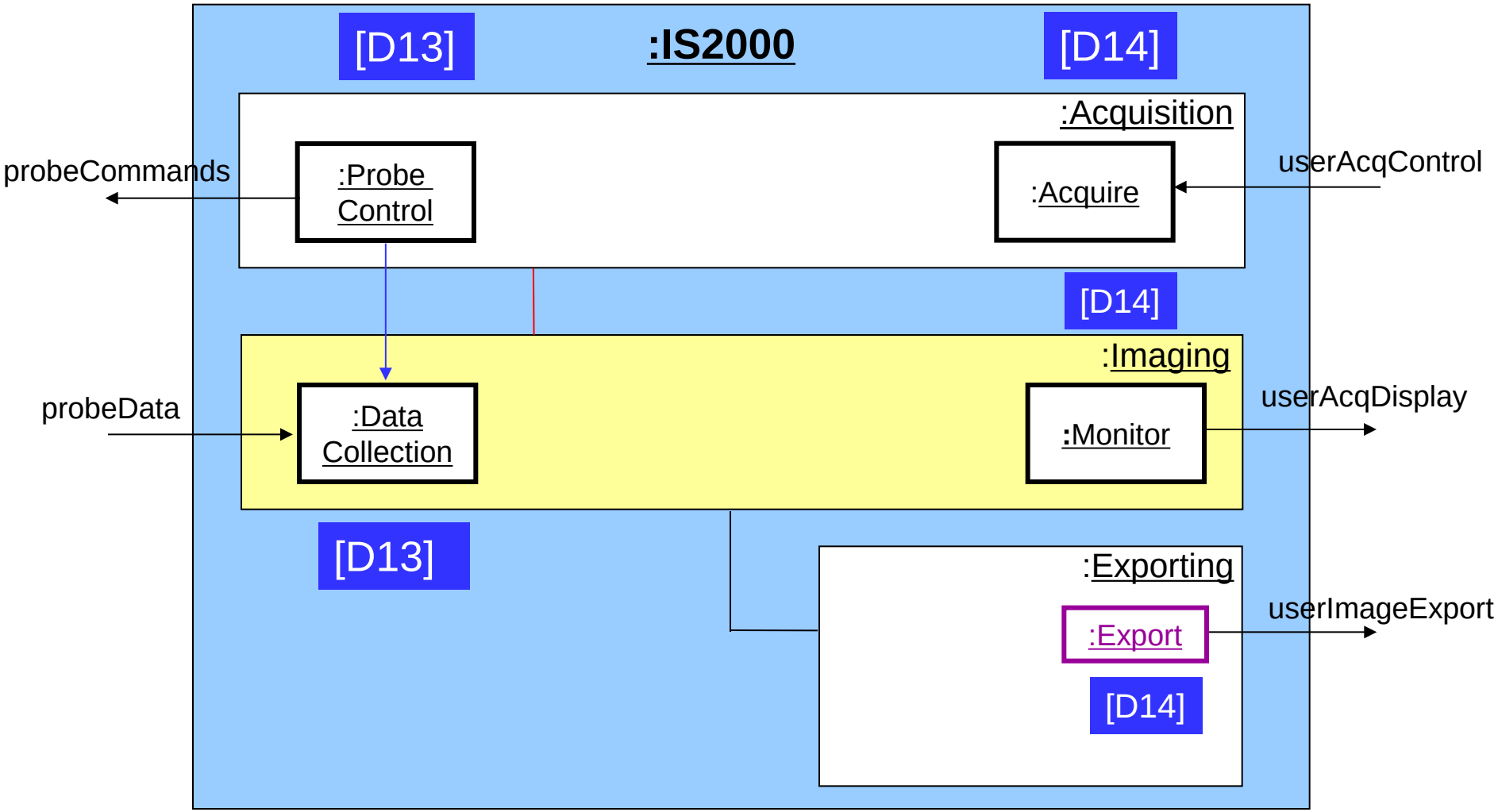
  (<= DataCollection receives formatting information from ProbeControl and uses it to accept the image data from the probe)

  - In module view, we will put modules for both in the same layer

[D14]

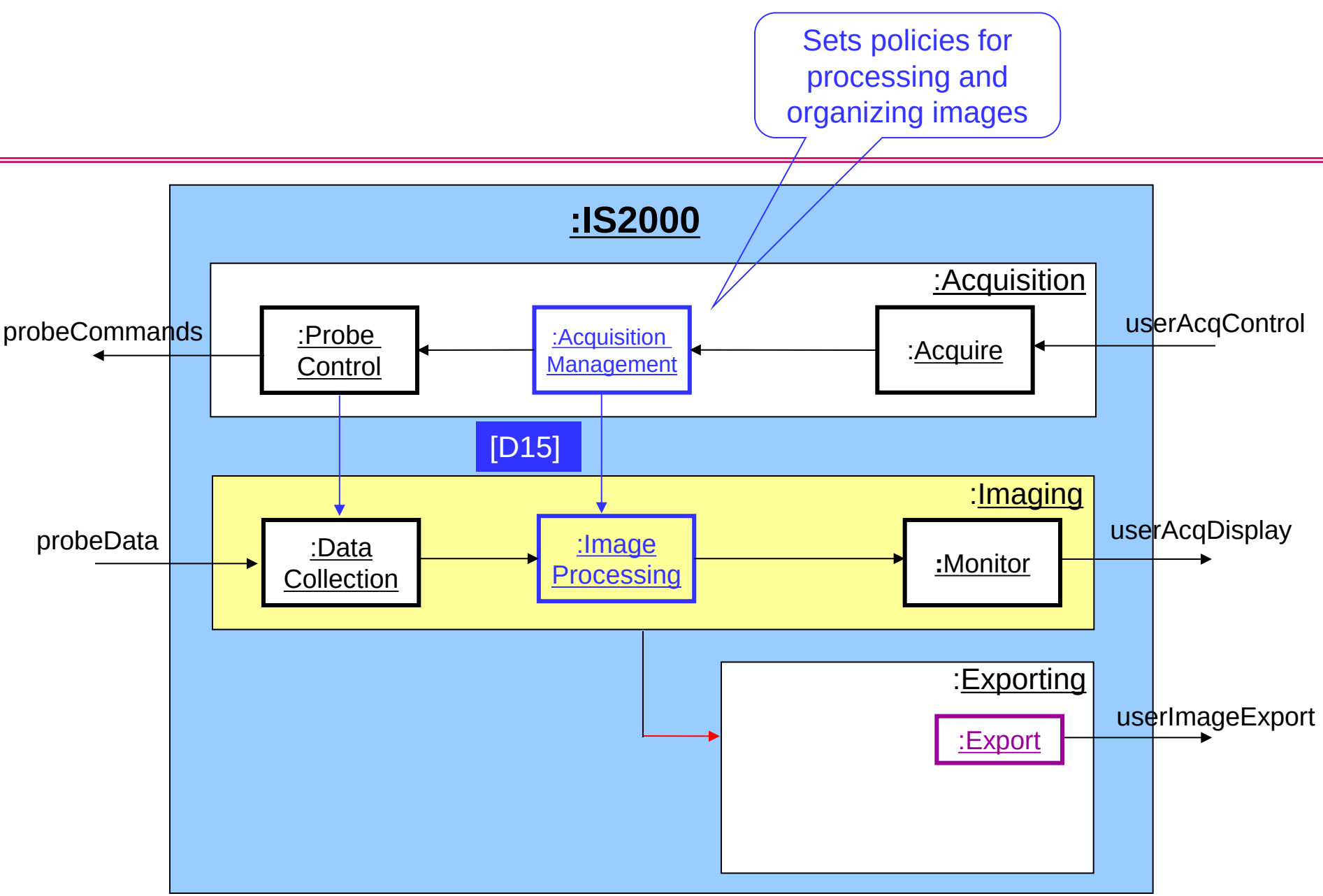
- **S6C**: "*Decouple the user interaction model*"

  => Add Acquire, Monitor, Export components


[D13]
:IS2000
[D14]
:Acquisition
probeCommands
:Probe Control
:Acquire
userAcqControl
[D14]
:Imaging
probeData
:Data Collection
:Monitor
userAcqDisplay
[D13]
:Exporting
:Export
userImageExport
[D14]

[D15]

- **S6A**: "*Separate components and modules along dimensions of concern*"
  - ⇒ AcquisitionManagement created
  - ⇒ Imaging component must abide by the policies set by AcquisitionManagement
    - ⇒ Add connector (a) from AcquisitionManagement to Imaging


Sets policies for processing and organizing images
:IS2000
:Acquisition
probeCommands
:Probe Control
:Acquisition Management
:Acquire
userAcqControl
[D15]
:Imaging
probeData
:Data Collection
:Image Processing
:Monitor
userAcqDisplay
:Exporting
:Export
userImageExport
networkAccess

[D16]

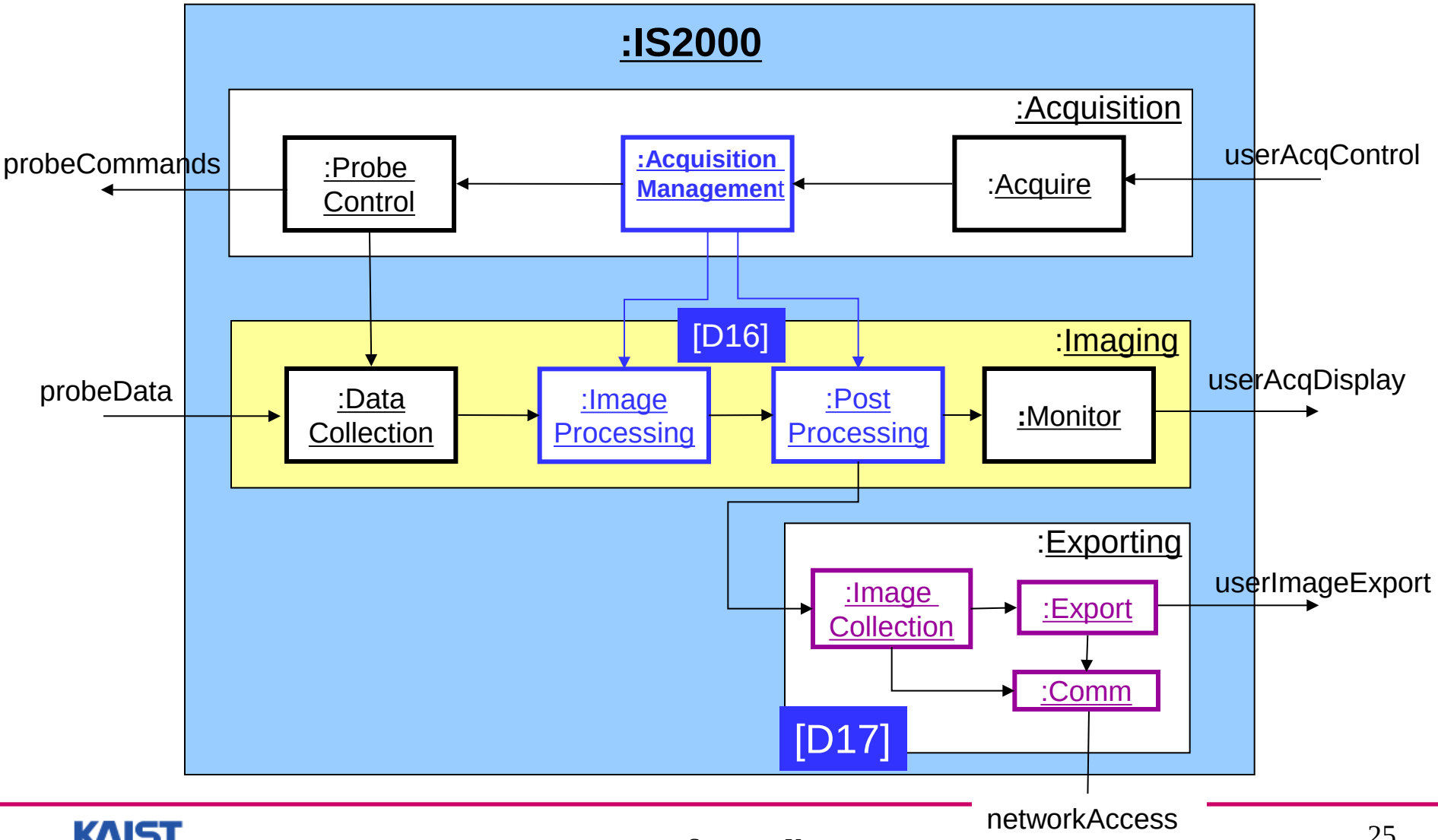
- **S9A**: "*Separate time-critical from non-time-critical components*"
  ⇒ ImageProcessing should contain only time-critical initial processing of raw data

[D17]

- **S7C**: "Encapsulate domain-specific image data"
  ⇒ The processed images are retained for as long as 24 hours
    - ImageCollection stores processed images

  ⇒ Export moves data to other systems

  ⇒ Comm is responsible for domain specific communication of the image data


:IS2000
:Acquisition
probeCommands
:Probe Control
:Acquisition Management
:Acquire
userAcqControl
[D16]
:Imaging
probeData
:Data Collection
:Image Processing
:Post Processing
:Monitor
userAcqDisplay
:Exporting
:Image Collection
:Export
:Comm
userImageExport
[D17]
networkAccess

For what kind of reasoning are ports and roles useful here?

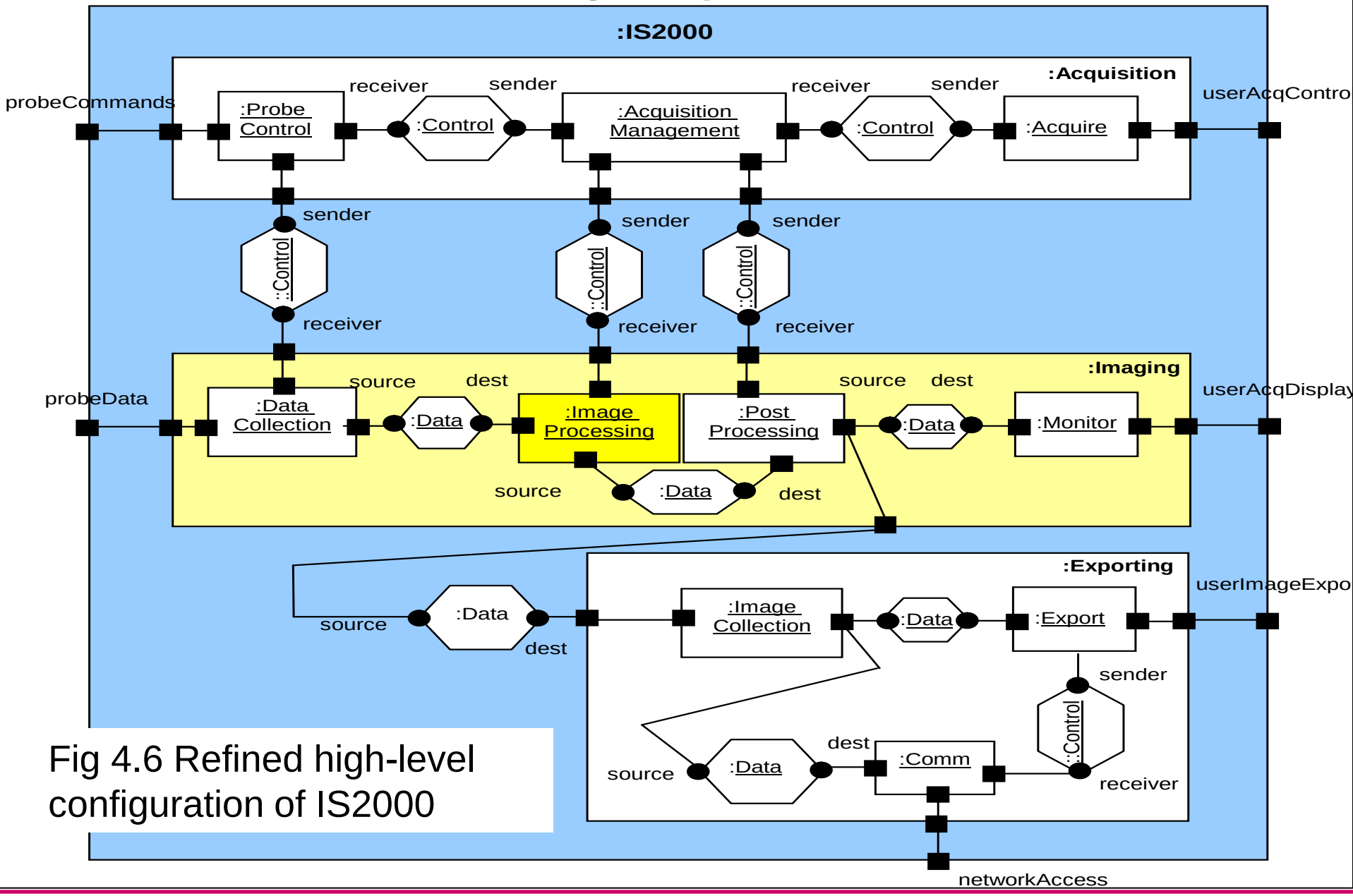
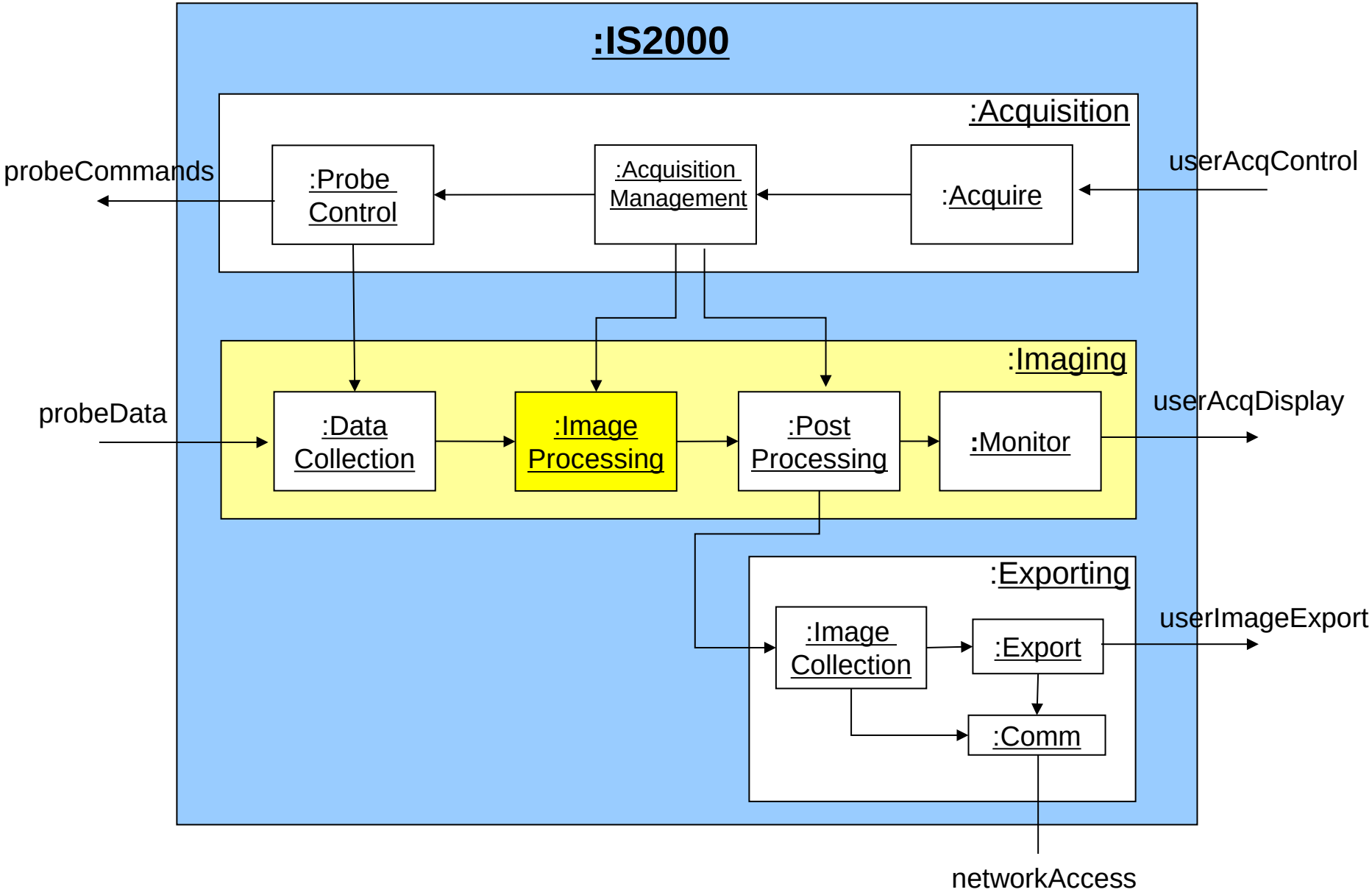


Fig 4.6 Refined high-level configuration of IS2000

Which one is better for “conceptual” view?



We now focus on ImageProcessing Component

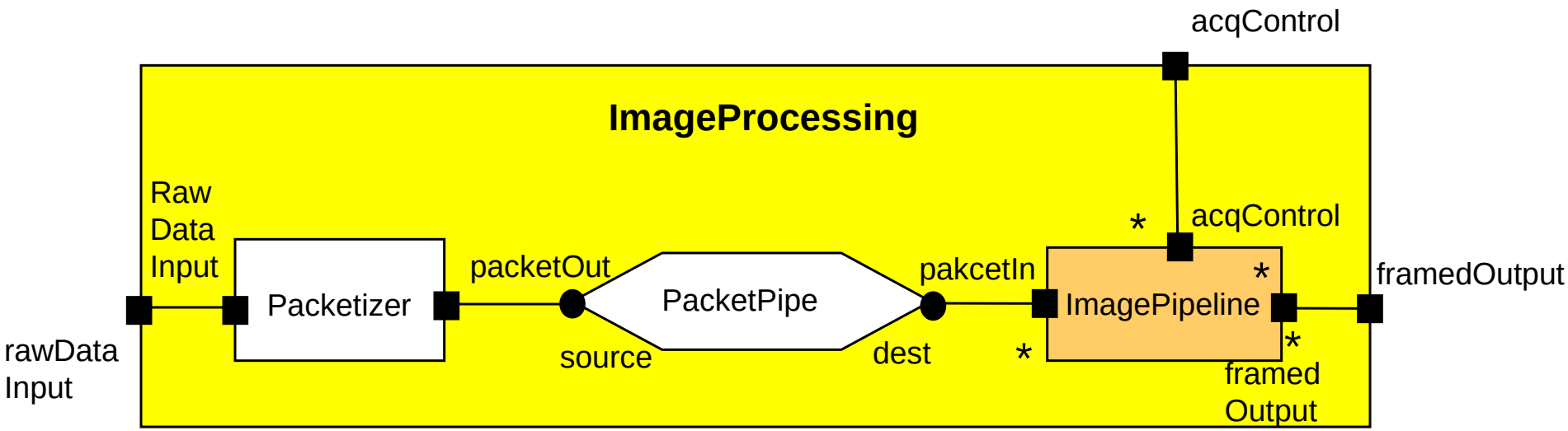


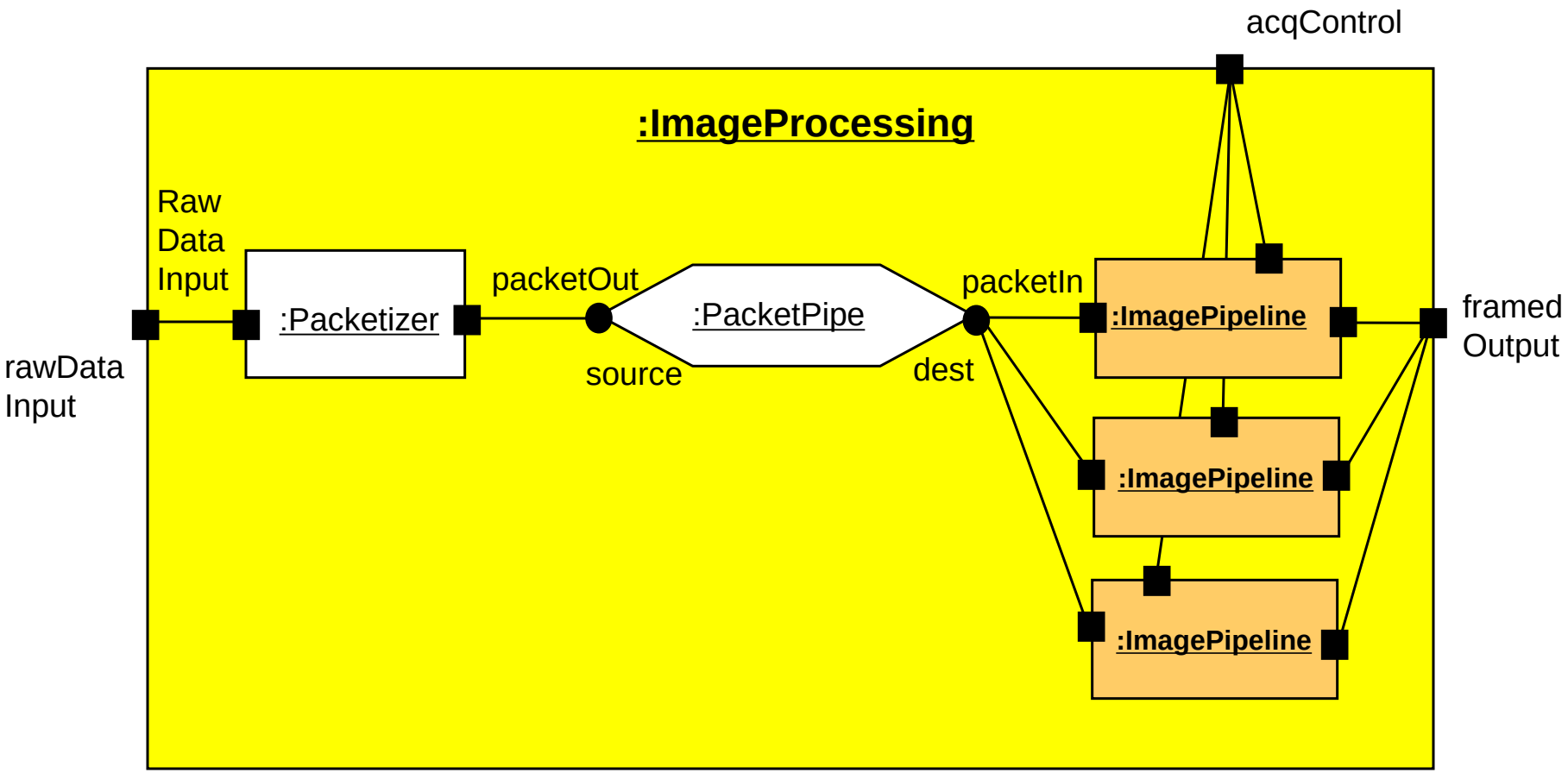
Figure 4.7. ImageProcessing decomposition

[D18]

S7A. "Use a flexible pipeline model for image processing"

• All pipelines receive the same data from the Packetizer and filter out what they don't need.

An Instance of Figure 4.7



What is one image pipeline?

Image Processing Pipeline Example

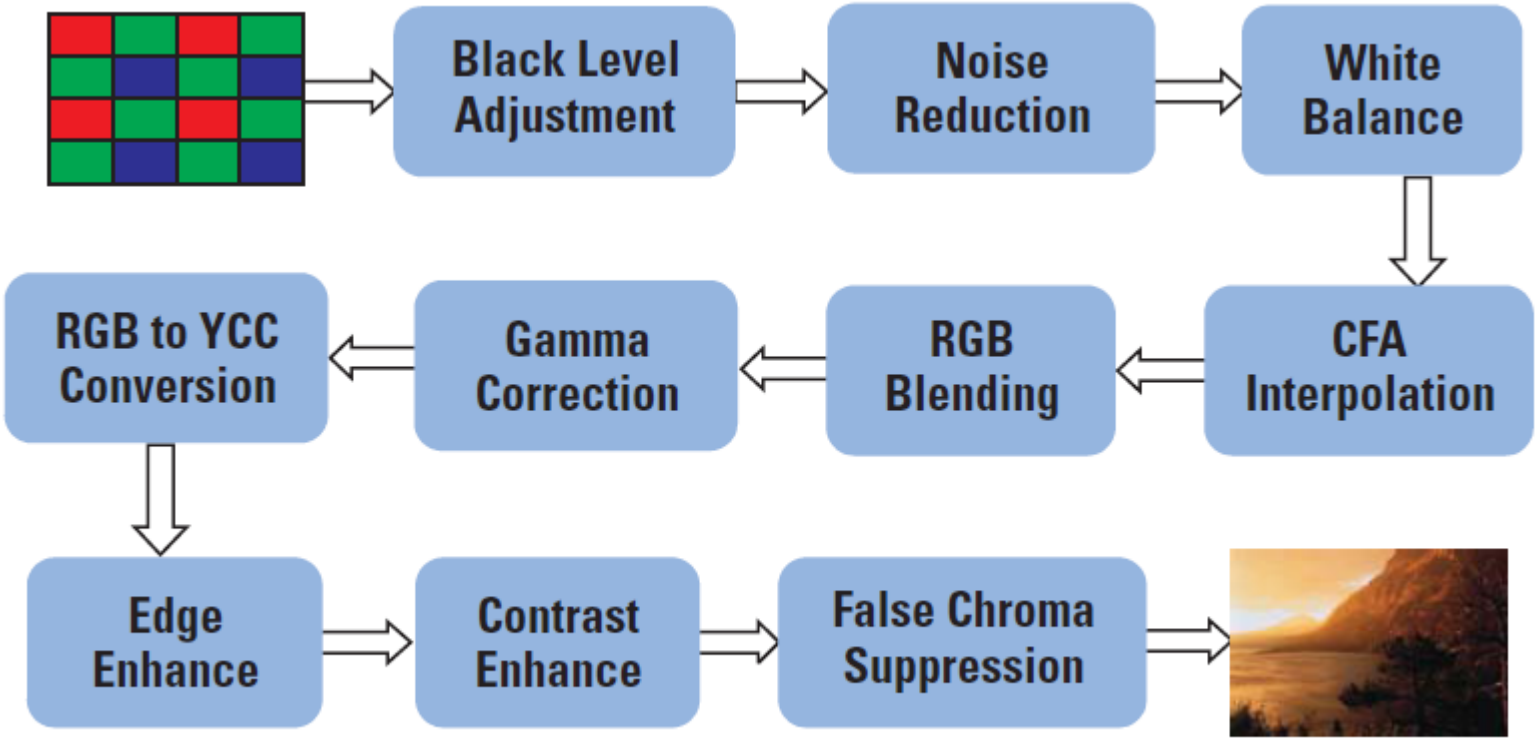


Figure 1. DaVinci imaging-processing pipeline.

Jianping Zhou, "Getting the Most Out of Your Image-Processing Pipeline", Texax Instruments White Paper, Oct. 2007.

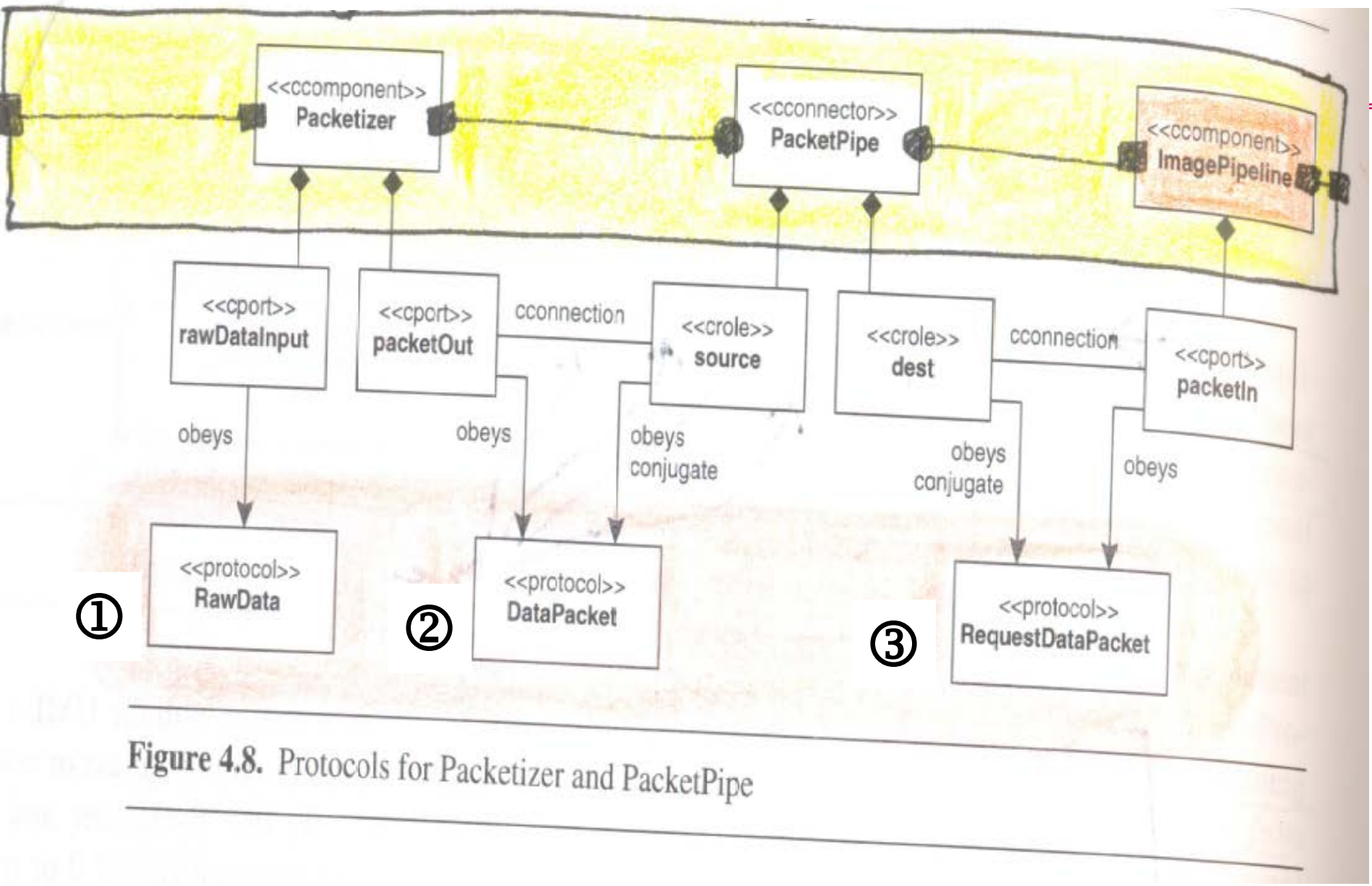
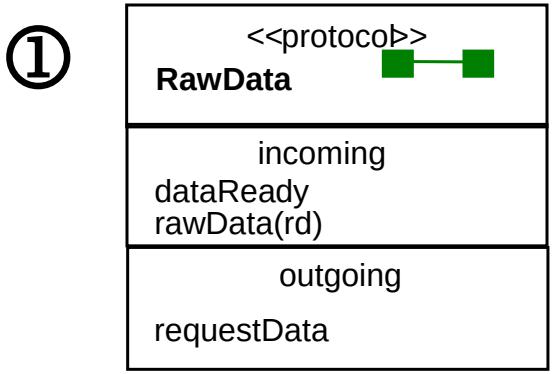
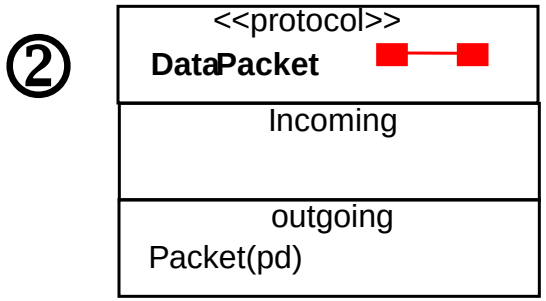
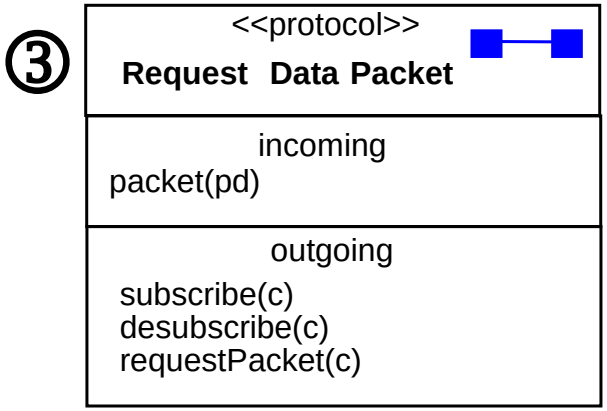


Figure 4.8. Protocols for Packetizer and PacketPipe







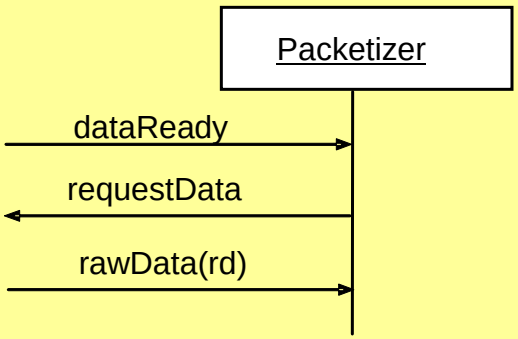


Figure 4.9.
RawData protocol

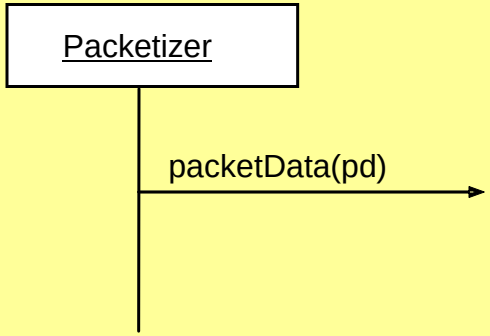


Figure 4.10.
DataPacket protocol

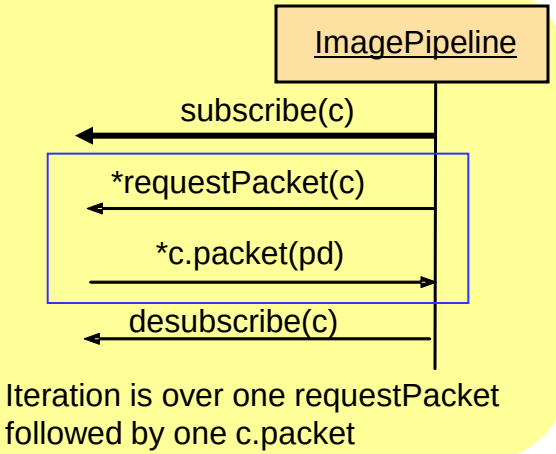
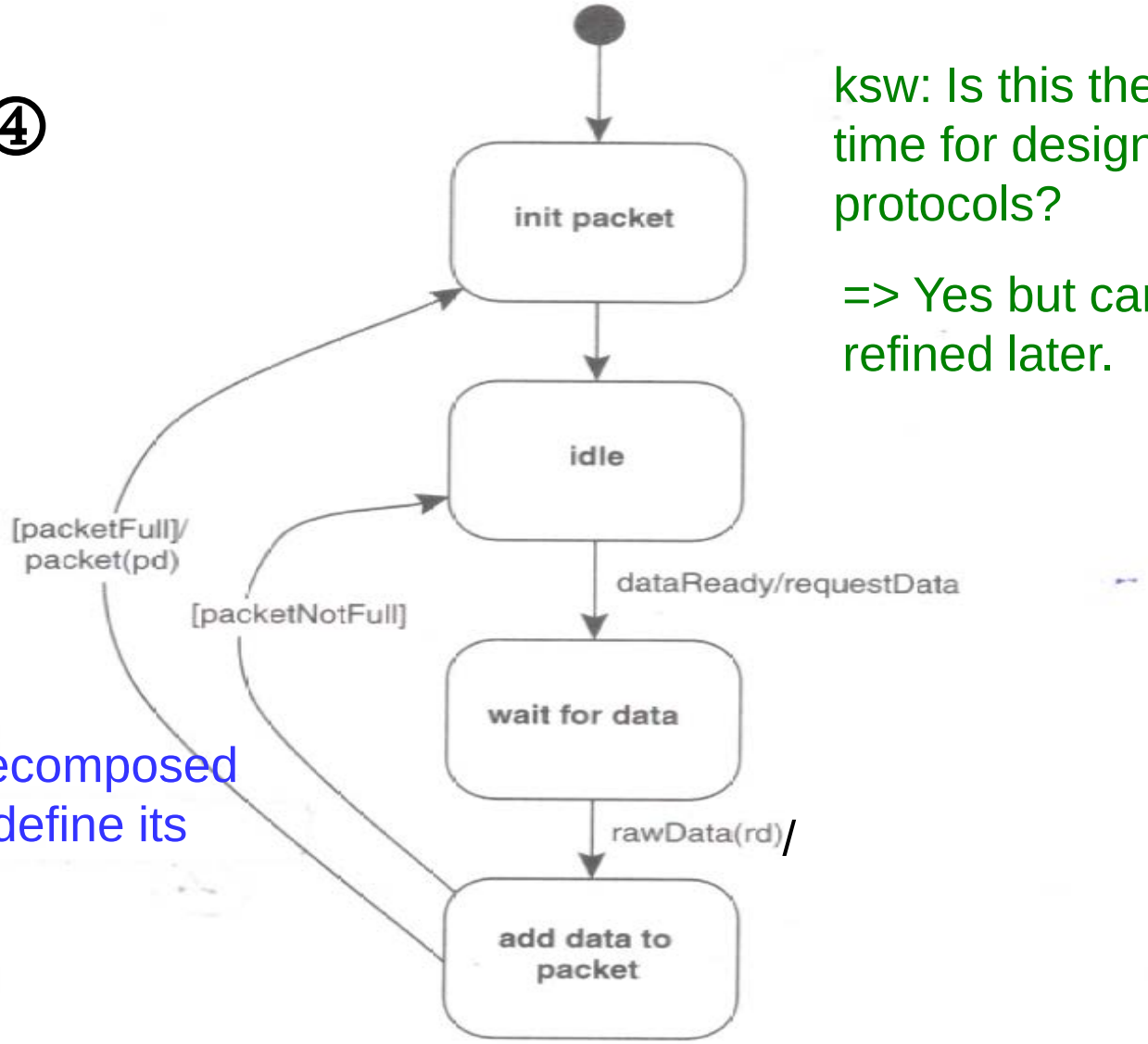


Figure 4.12. Request
DataPacket protocol

④

ksw: Is this the right time for designing protocols?

=> Yes but can be refined later.



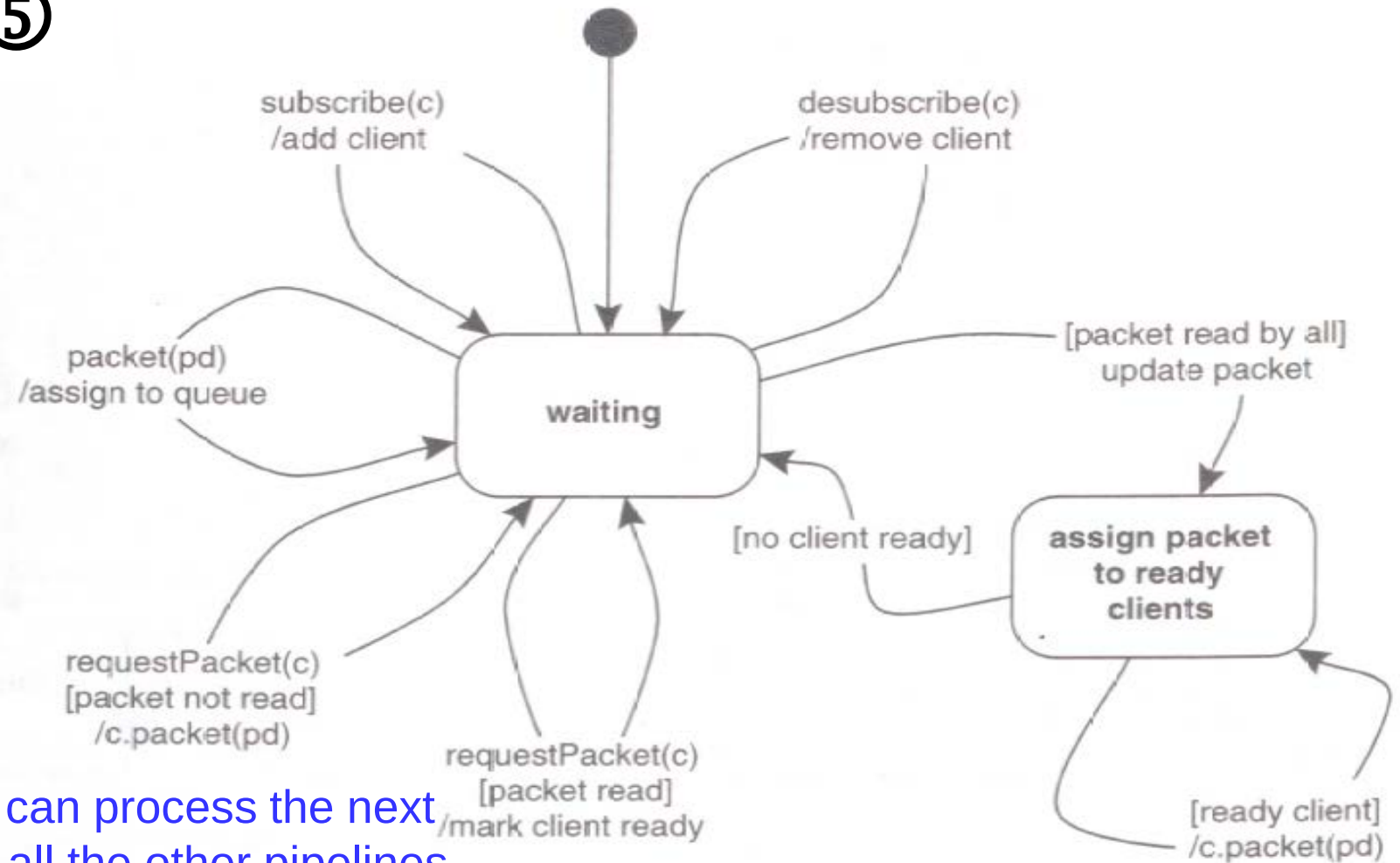
Packetizer is not decomposed any further. So we define its behavior now.

rd: raw data

pd: processed data

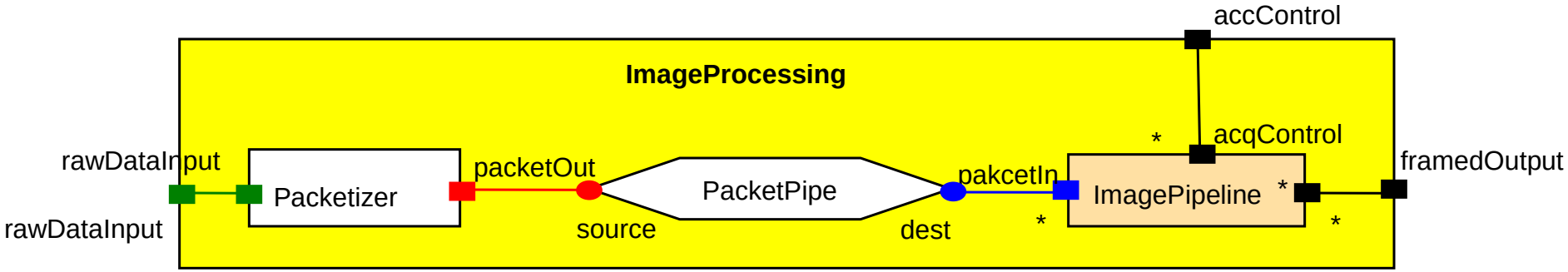
Figure 4.11. Packetizer behavior

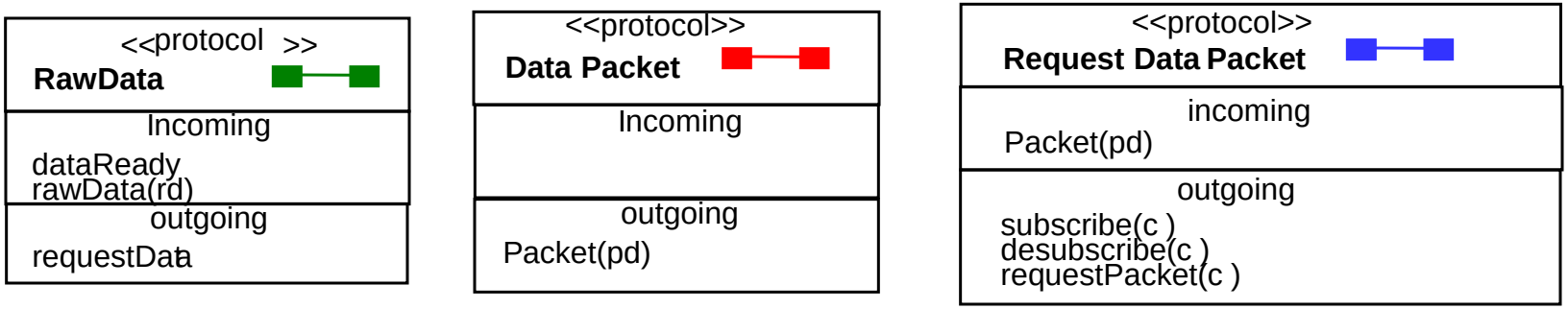
⑤

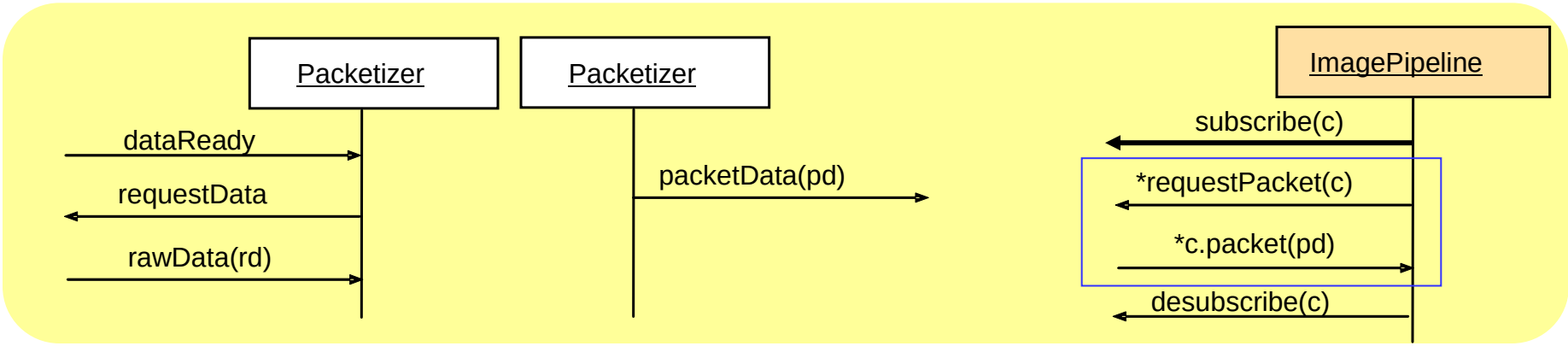


No pipeline can process the next packet until all the other pipelines are also ready to receive it.

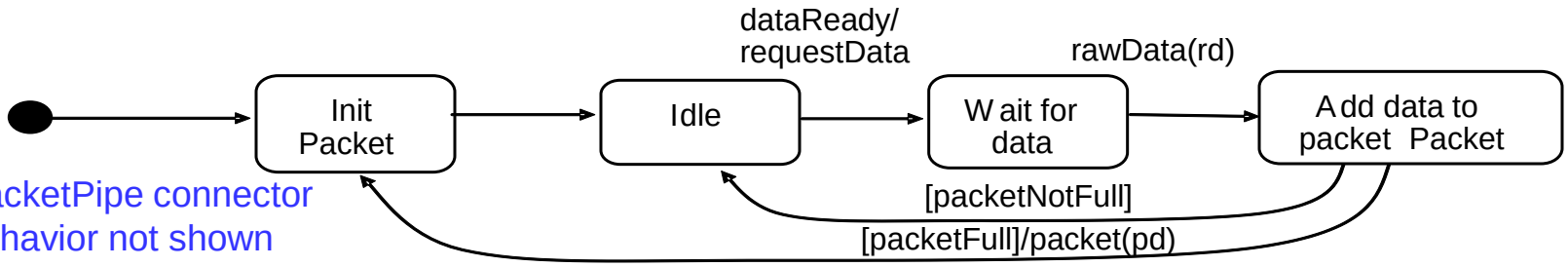
Figure 4.13. PacketPipe connector behavior







Packetizer Behavior



PacketPipe connector behavior not shown

Design of ImagePipeline

[D19]

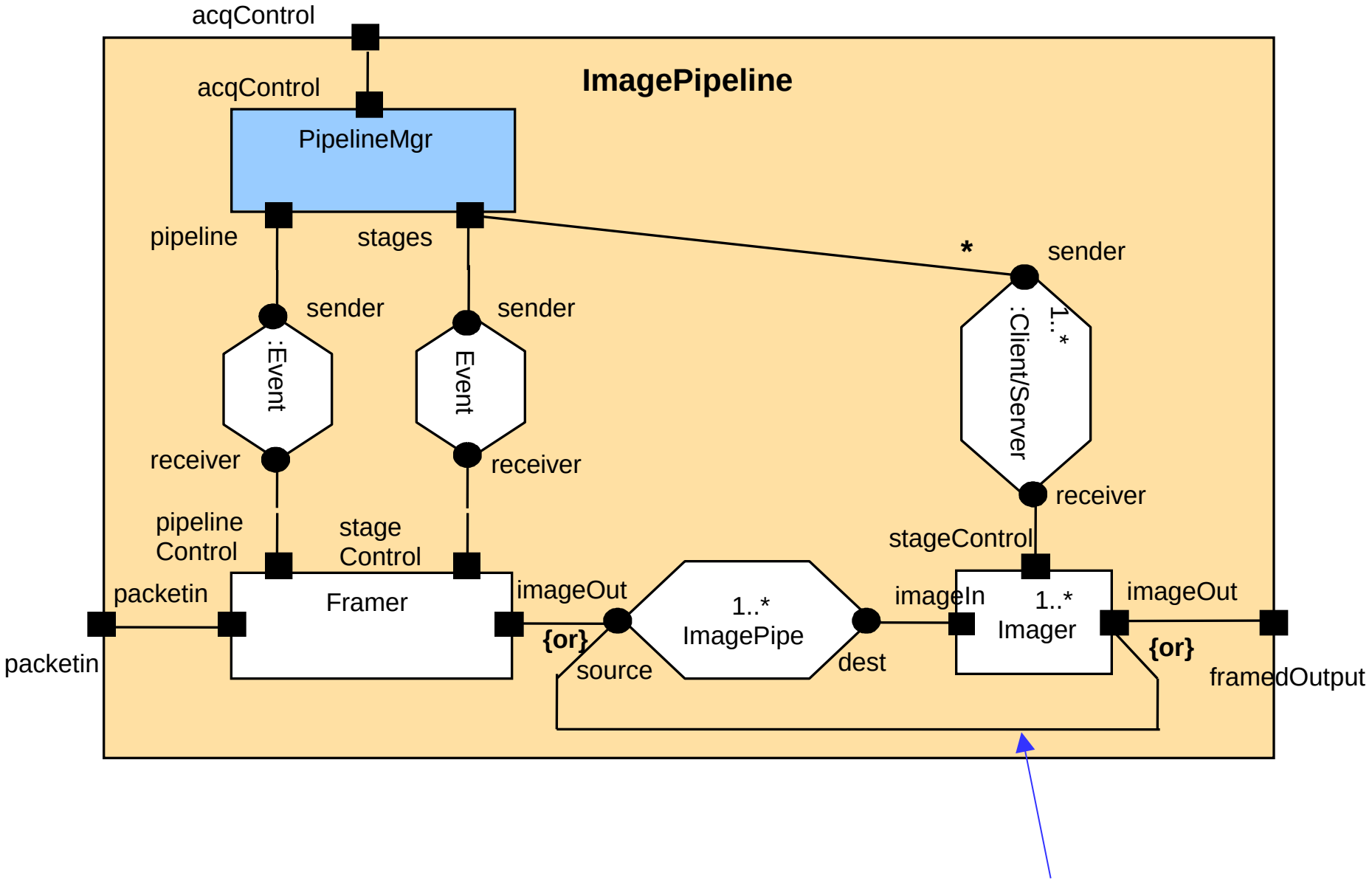
- *S7A. "Use a flexible pipeline model for image processing"*

  => Decompose ImagePipeline into a number of stages and a PipelineManger

  => Makes it easy to

  - reuse a stage across multiple pipelines
  - add new stages to a pipeline
  - create new pipelines for new kinds of acquisition procedure

  (One ImagePipeline per one acquisition procedure.)



ImageOut port is either connected to another ImagePipe or when it is the last stage in the pipeline, is bound to framedOutput.

[D19]

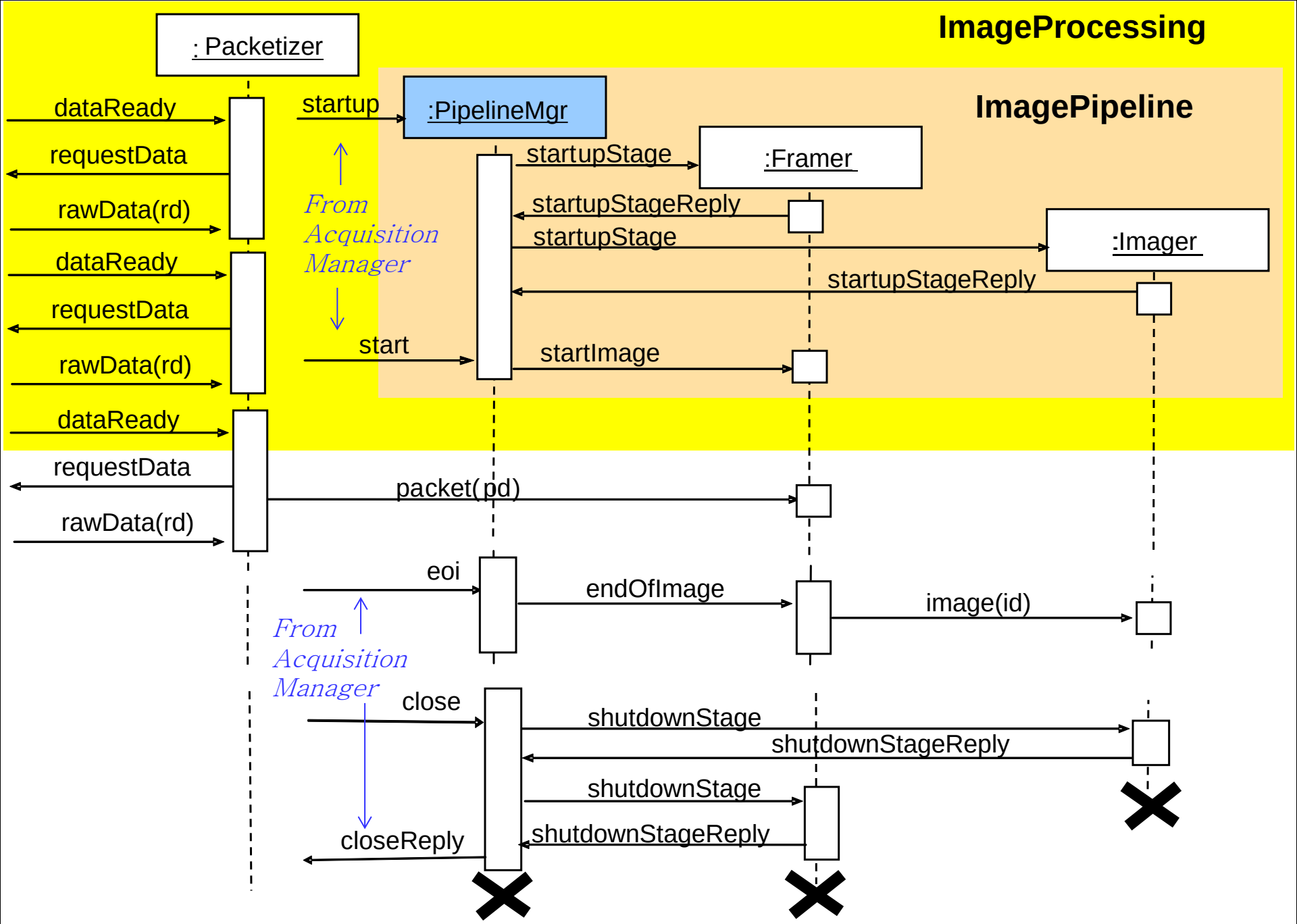
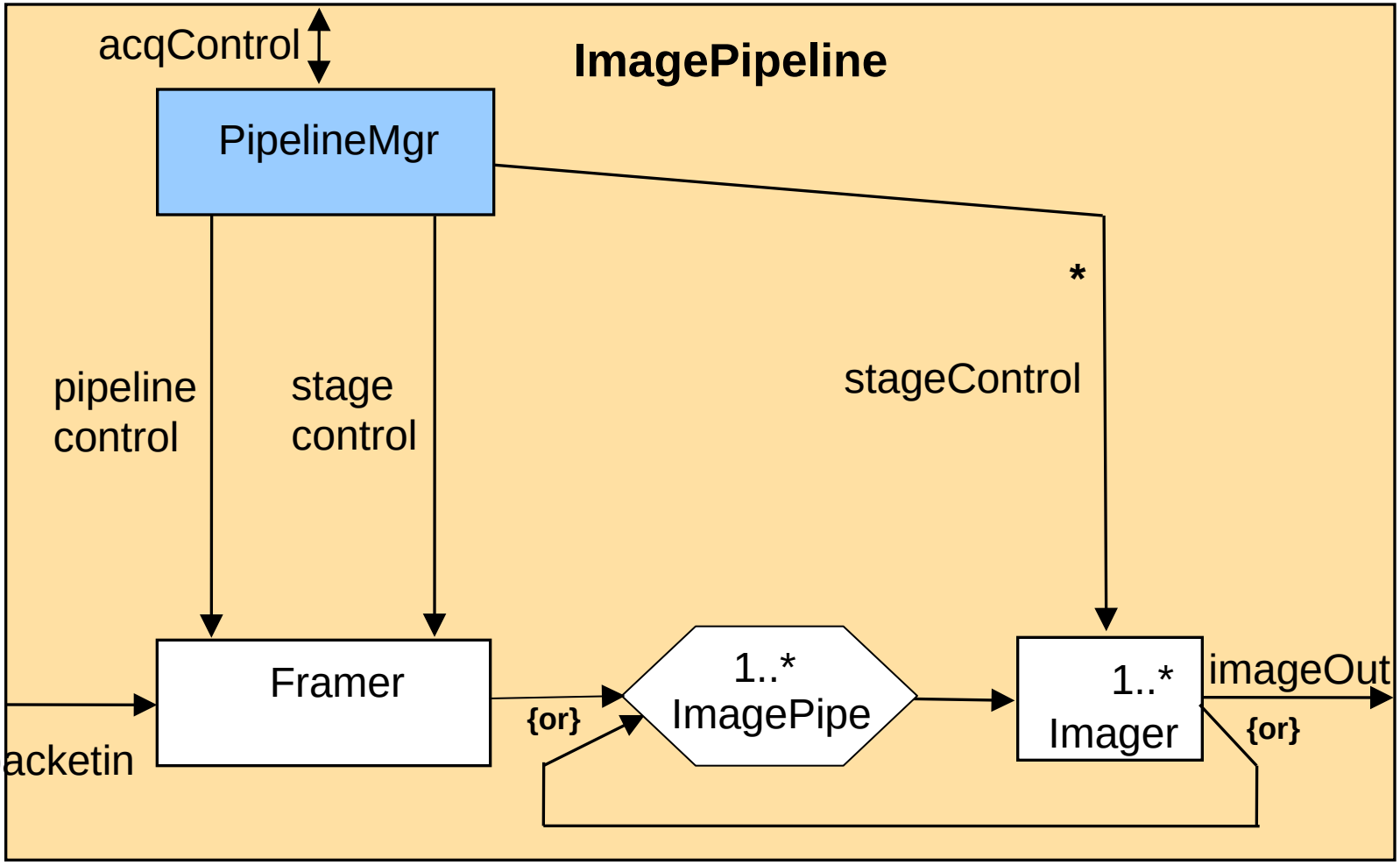


Figure 4.15. Sequence Diagram for the ImageProcessing component

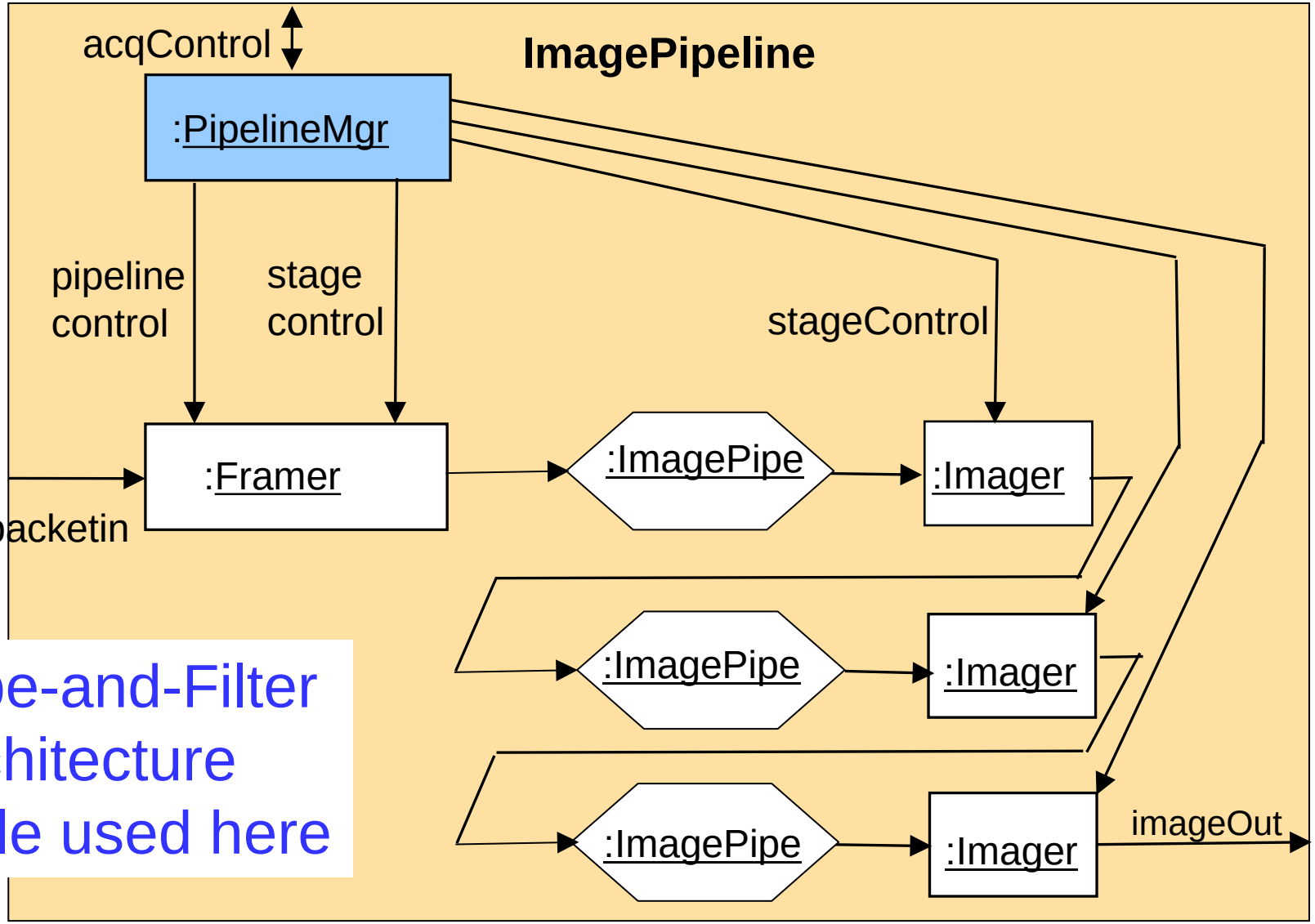
Is this better than the previous one for conceptual view?



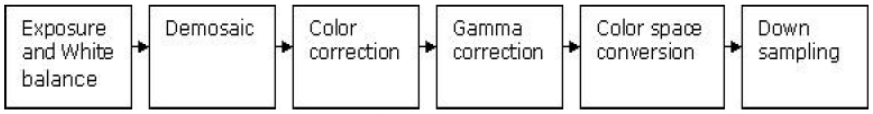
Conceptual Architecture for image processing component

- Data flow explicit with minimal notation
- Role and multiplicity is structural. So it is explicit.
- Connector mapping to runtime entities to be determined
- Connection component explicit

An Instance of ImagePipeline



Pipe-and-Filter architecture style used here



Typical image processing chain after acquisition [Zaharia 11]

- So far the focus was on introducing components to support the functional features (except for the real-time processing issue).

  => ksw: Actually, *functional cohesion* promotes understandability, buildability and evolvability

- Let's consider additional global properties on failure detection.

  (We could have considered this during the initial global analysis.)

Newly Added Factors

Product Factor	Flexibility and Changeability	Impact
P5: Failure detection, reporting, recovery		
P5.1: Error classification		
System errors are classified according to their type and severity.	Error classification is likely to change during development. It is affected by functional features, the user interaction model, and the probe model.	There is a moderate impact on all components. Error logging is affected.
P5.2: Error logging		
Error logging is used to capture diagnostic, clinical, and software trace events.	Error logging is affected by error classification, error-handling policy, and the hardware platform. It is likely to change during development.	There is a moderate impact on all components. Acquisition performance may be affected.
P5.3: Support for use of error logs		
Error logs can be retrieved and viewed for the purpose of error tracking and diagnosis.	Requirements for the use of an error log are expected to be quite stable.	There is a minimal impact on components.
P5.4: Recovery		
Acquired imaged data must be recoverable in the event of a system failure.	Recovery requirements are stable. They are flexible to adapt to changes in acquisition size, data rate, format, and recovery support by the data storage system.	There is a moderate to large impact on all components involved in recovery.

Table 4.2. Factors Added During Design of Conceptual View

Issue 10

Implementation of Recovery

Recovery is the responsibility of many components. We need to ensure that support for recovery is uniformly implemented by all relevant components.

Influencing Factors

P5.4: Acquired imaged data must be recoverable in the event of a system failure. Recovery requirements are stable and can be flexible to reduce effort of implementation.

Solution

Apply the principle of separation of concerns to introduce a separate recovery mode of operation for each component. Ensure that the data to be recovered is accessible to all components.

Strategy: *Introduce a recovery mode of operation.*

Introduce a recovery mode to localize change. This means introducing components for cleanup, shutdown, and crash recovery. The scope of impact can be reduced cost-effectively by introducing a recovery mode for processing the recovered data. Using a separate recovery model means that we don't have to worry about affecting the system's performance during an acquisition. However, users may wish to execute recovery in parallel with other activities in the future. Existing components involved in processing will now have an additional responsibility to provide an interface to run in recovery mode. We also want to encapsulate file system and database recovery support.

Strategy: *Make all data at the point of recovery persistent and accessible.*

Make data at the point of recovery persistent and accessible through an interface at the architecture level for easier implementation of a recovery mechanism. If it is not accessible, then recovery may require snapshots of the internal state.

Issue 11

Implementation of Diagnostics

Error logging will be used to support diagnostics. Responsibility for diagnostics is shared by all components. We need to ensure that support for diagnostics is uniformly implemented by all components.

Influencing Factors

P5.1: System errors are classified according to their type and severity. Error classification is likely to change.

P5.2: Error logging is affected by error classification, error-handling policy, and the hardware platform. It is likely to change.

P5.3: Requirements for the use of error logs are expected to be stable.

Solution

Support for diagnostics is needed during development, for tracing, and at runtime. The design of the diagnostics has many elements. It includes the log itself (which is a repository of the error information), the types of errors being reported, and the kinds of information captured (for example, component, error, location). Components need to log information, and the user needs to be able to read the logs.

Strategy: *Define an error-handling policy.*

Define a policy to avoid, detect, and handle different classes of errors. This includes how to define and use exceptions, and how to select an exception mechanism. Provide guidelines on how to avoid errors, as well as how to detect and report them.

Strategy: *Reduce effort for error handling.*

Error handling is tedious, and many developers dislike it. Make their task easier by using tools to support the error-handling policy. For example, there are tools that take an error classification scheme as input and generate code templates for error reporting.

Strategy: *Encapsulate diagnostics components.*

Localize the impact of error logging by encapsulating one component for error logging and another for the use of logs. Define product-specific interfaces so that their implementation can be easily replaced.

Strategy: *Use standard logging services.*

To meet the schedule and to take advantage of industry standards, use standard services such as message catalogs and the file system to read and review log files.

[3] Final Design Tasks

Resource Budgeting - Memory Allocation

- FIFO queues for the pipes between the pipeline stages
  <– Based on predicted image size

- PersistentDataPipe (Connector between ImageProcessing and PostProcessing)
  <– Based on the number of images that will accumulate between the two
  <– which in turn is based on their relative speed of processing an image

Resource Budgeting - Time

- Time between *detection of a realtime event* and *its effect on event data* must be $<$ 1 msec.

  $\Rightarrow$ Affects budgets for DataCollection, ImageProcessing, and PostProcessing

  $\Rightarrow$ Recursively, budgets are assigned to subcomponents

- Must support a sustained data rate of 2MB/sec generated by the probe hardware.

  => Affects the time budget and the buffer size for the Data Collector component

Questions?