Exploiting Constraints to Build a Flexible and Extensible Data Stream Processing Middleware

NEXUS

Workshop on Scalable Stream Processing Systems (SSPS)

IPDPS 2010, Atlanta/Georgia, USA

19.04.2010

Nazario Cipriani, Carlos Lübbe, Alexander Moosbrugger

Data Stream Processing Systems – Some Foundations

- **Context-aware applications need context data to be processed**
  - Context-aware systems utterly heterogeneous
  - Nearby sensors produce context data streams that needs to be processed online
  - Not convenient to first store and then process offline due to high data volume

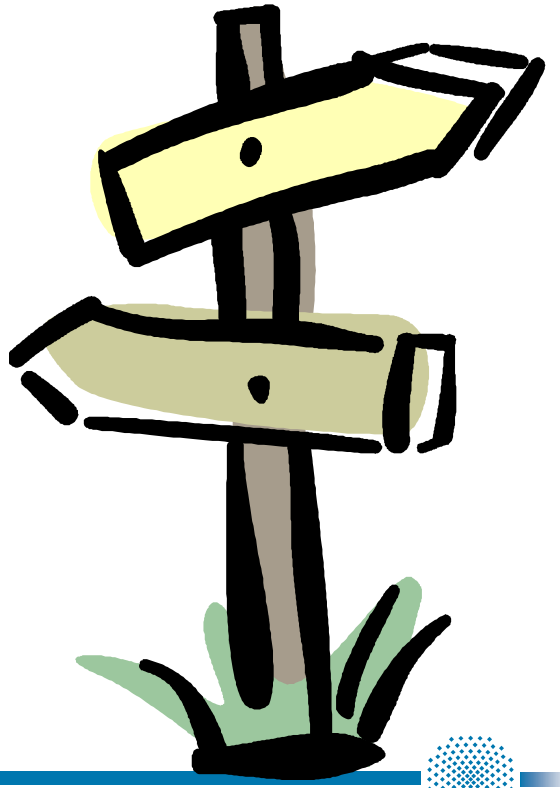
    ➔ **Use *Data Stream Processing Systems (DSPS)*!**

- **Data streams are handled by Data Stream Processing Systems**
  - A.k.a. Data Stream Management Systems (***DSMS***)
  - Data streams are processed according to some processing definition
  - Push-based processing paradigm instead of pull-based

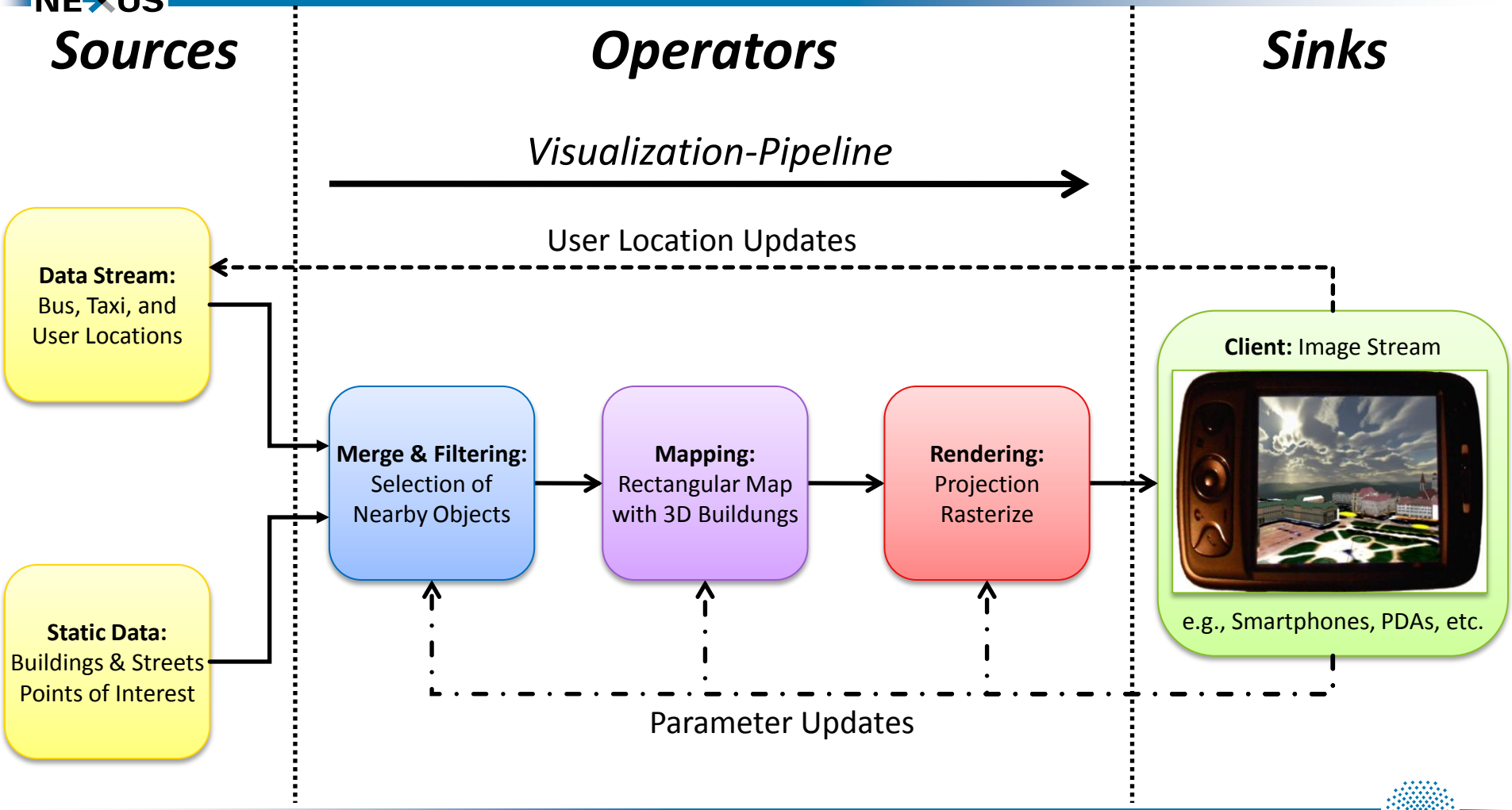
- **Today's DSPS offer a broad range of sophisticated and efficient processing schemes for online data stream processing**
  - Well suited for general purpose data stream processing

Agenda

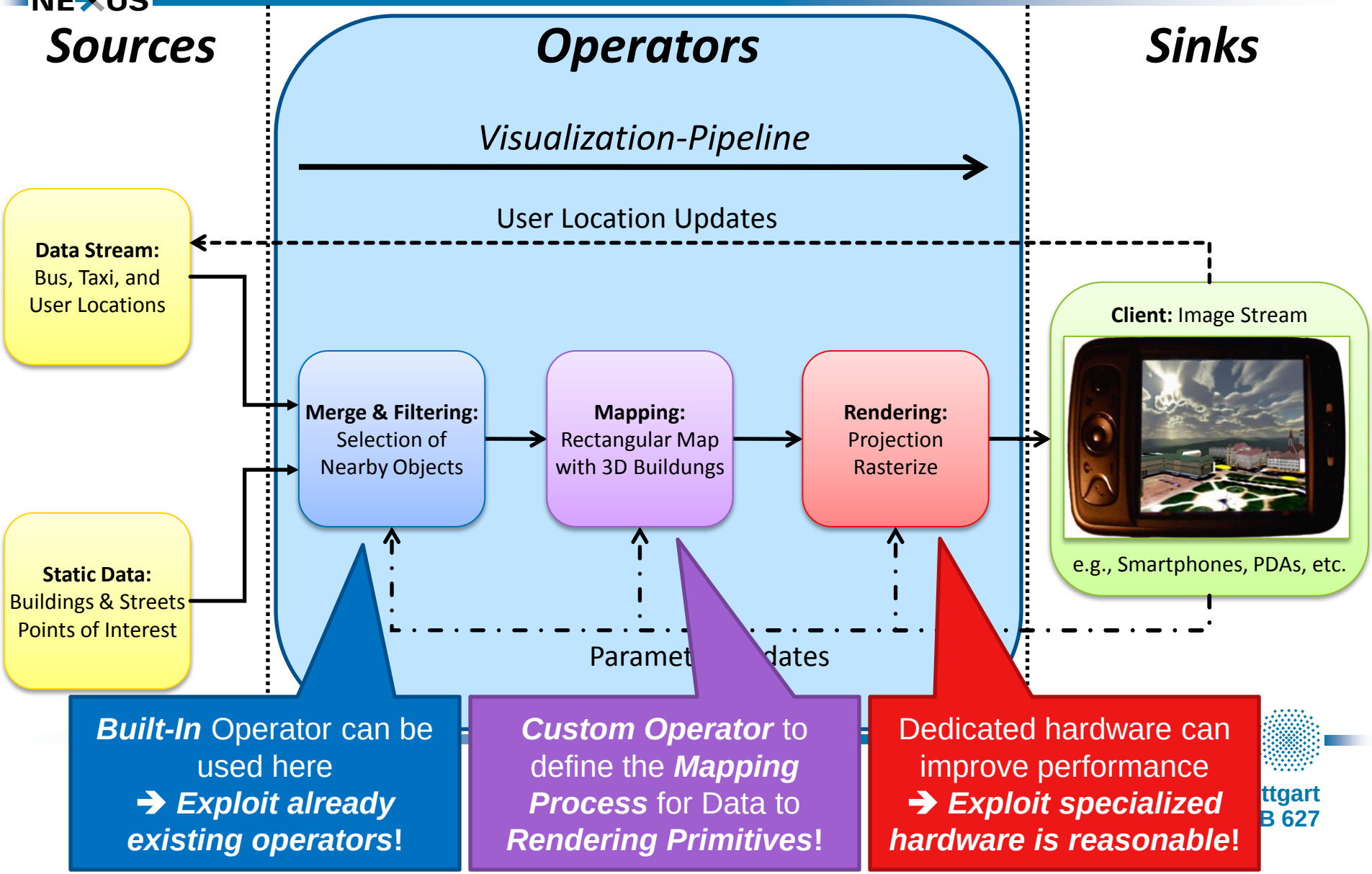
- **Non-Trivial Application Scenario of a Distributed Visualization Pipeline**
- **Adaptation Problem for non-trivial Applications**
  - Gap Between Application's and System's Interests
- **Constraints Classification**
- **Enhancing the NexusDS Platform by Constraints**
  - NexusDS Platform Overview
  - Network Groups and Operator Model
  - Constraint-Based Data Flow Graphs
  - Enhanced Processing Model Supporting Constraints
- **Conclusion and Future Work**



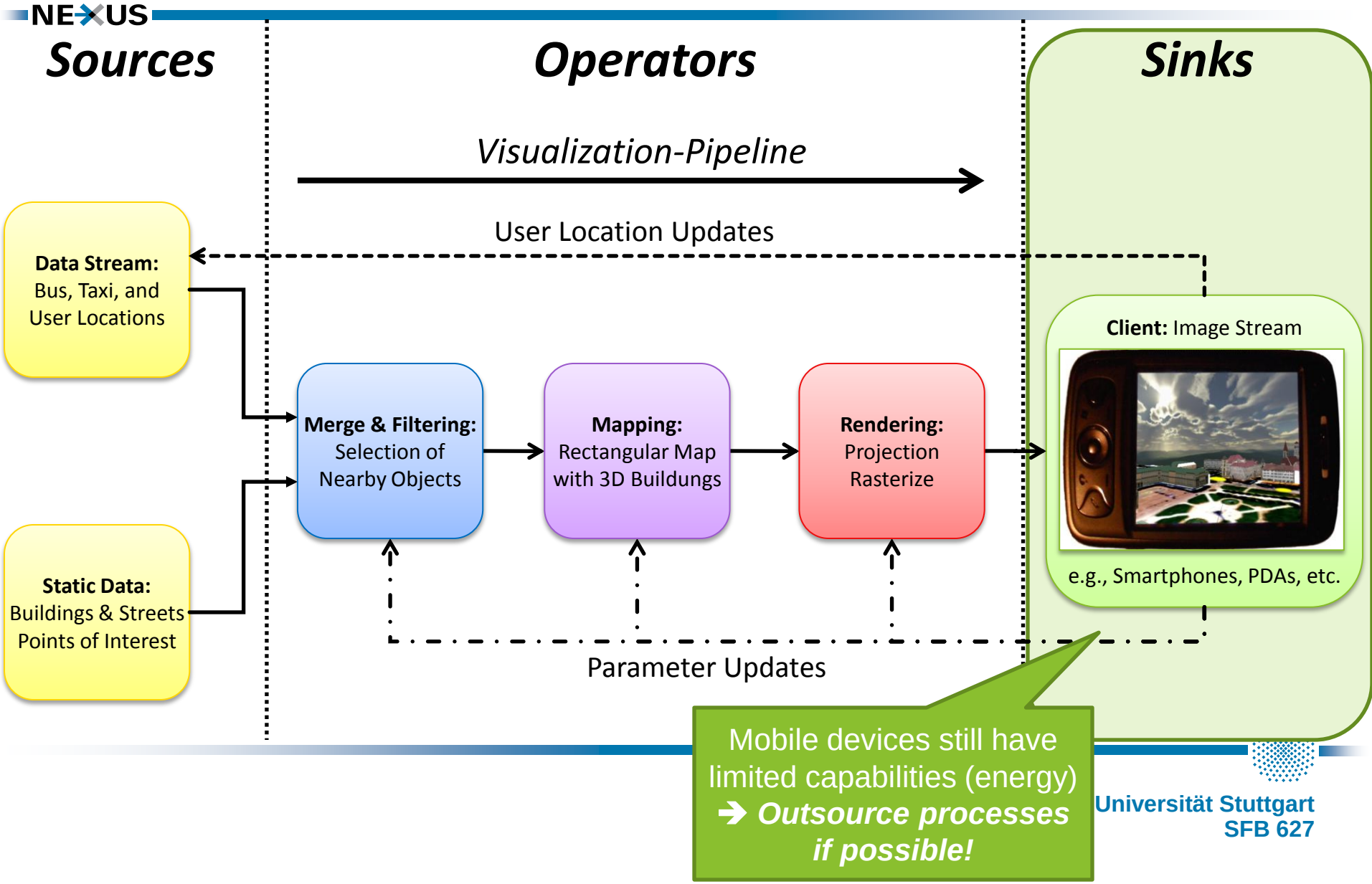
Sample Scenario – Distributed Visualization Pipeline



Sample Scenario – Distributed Visualization Pipeline



Sample Scenario – Distributed Visualization Pipeline



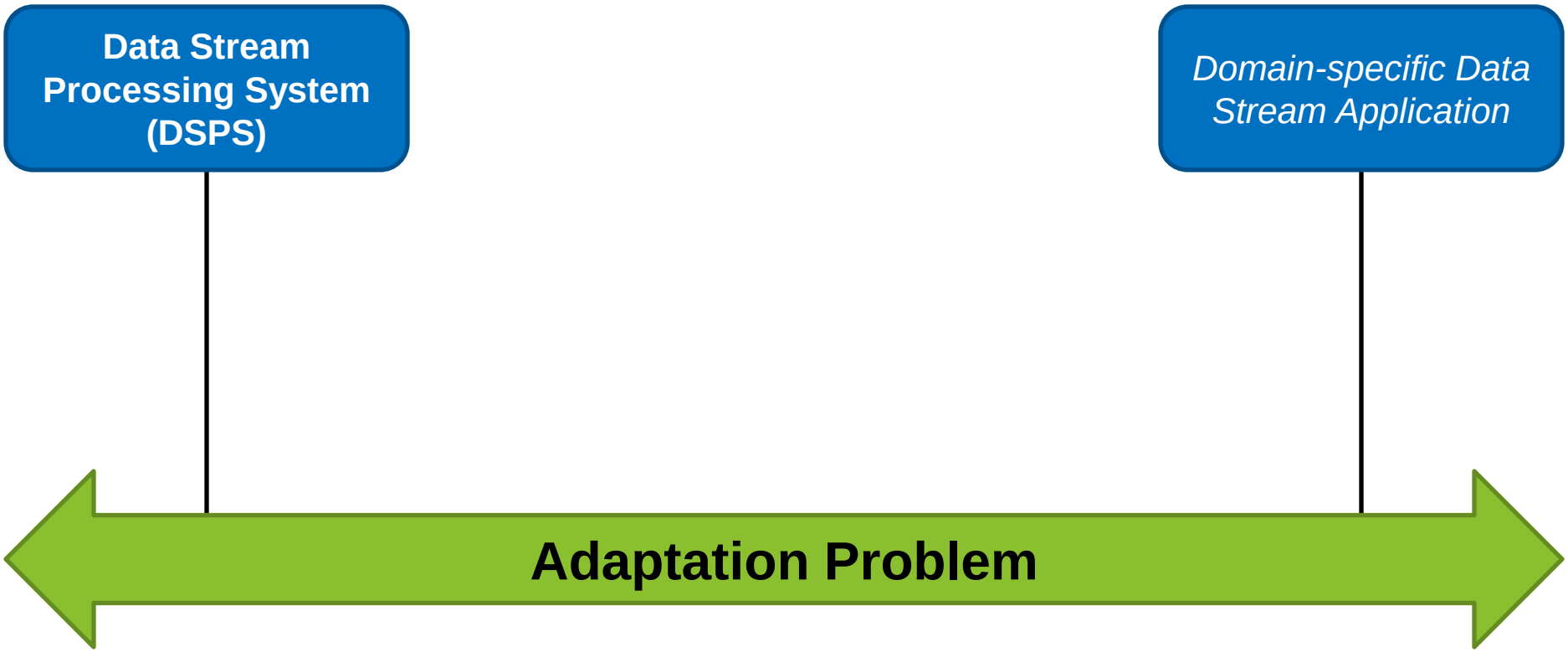
Requirements for DSPS to Support Non-Trivial Applications?

- **Extensible operator base**
  - Complex domain specific operators, may require dedicated hardware
  - Support for structured and unstructured data: “Tuples” vs. images, videos
- **Heterogeneous system topology**
  - System with wide range of devices in mind – powerful computing servers vs. mobile devices
  - Exploit available hardware for efficient execution
- **Operator has deployment and runtime restrictions**
  - Influence deployment and runtime of operators in processing pipelines

Adaptation Gap between Applications and DSPS

- **Many different (distributed) context data streams ➔ Use distributed DSPS**
- **Applications or domains may ask for specific functionality**
  - But still depending on general stream processing principles

    ➔ **Reuse existing components, "do not reinvent the wheel"!**

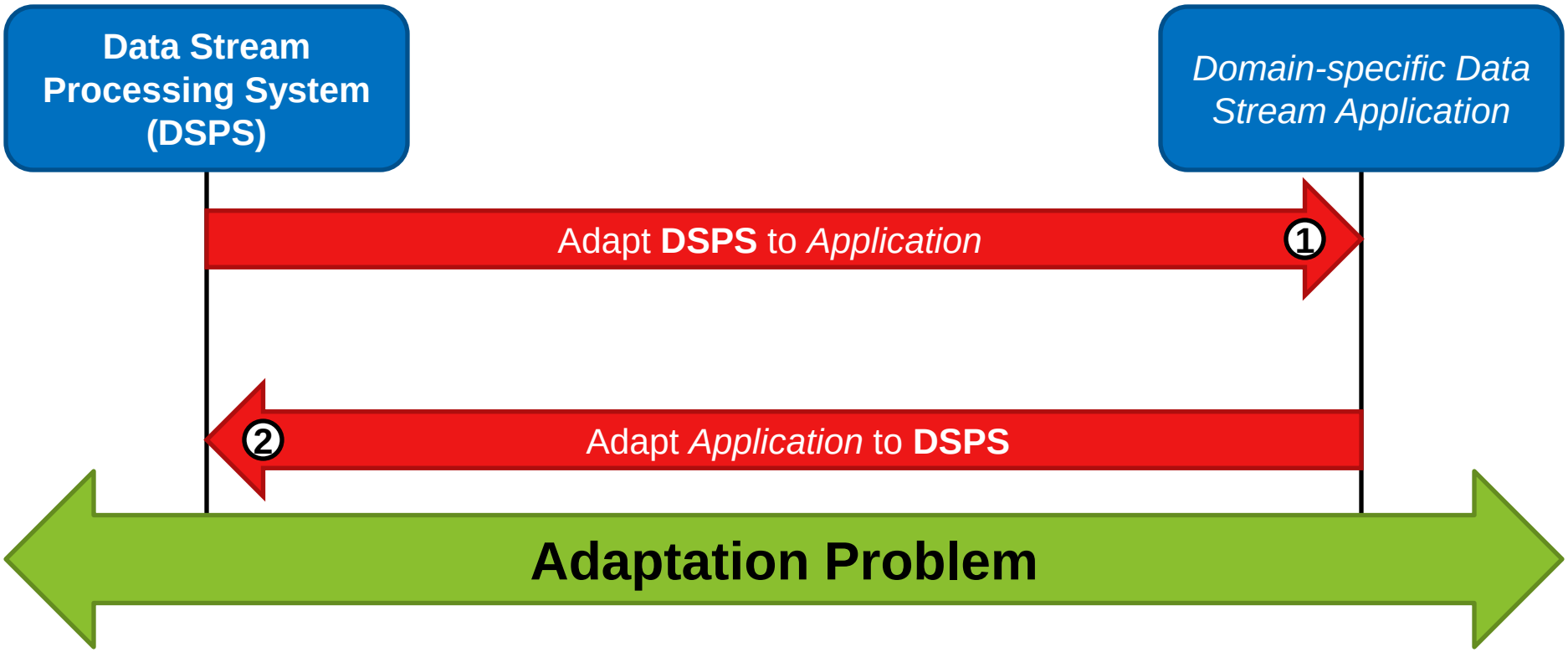


Adaptation Gap between Applications and DSPS

- **Many different (distributed) context data streams ➔ Use distributed DSPS**

- **Applications or domains may ask for specific functionality**
  - But still depending on general stream processing principles

    ➔ **Reuse existing components, "do not reinvent the wheel"!**

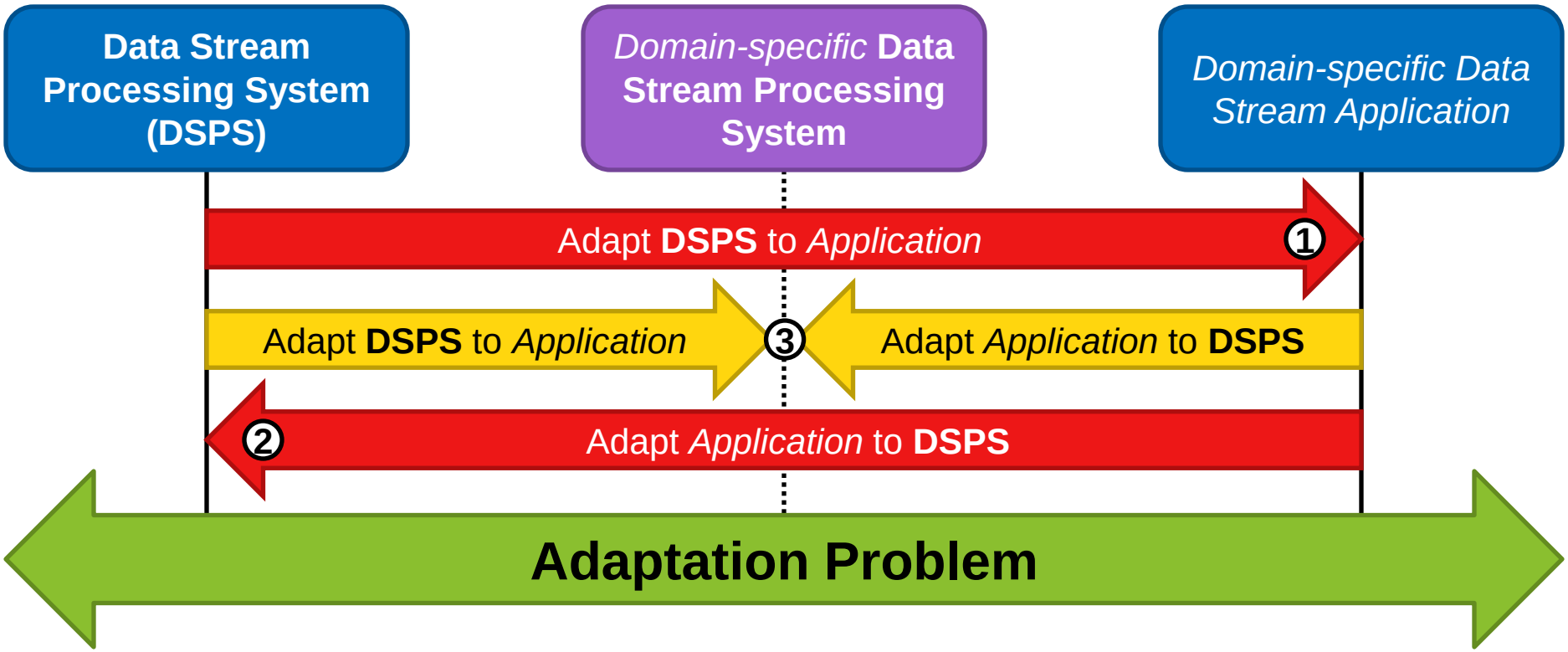


Adaptation Gap between Applications and DSPS

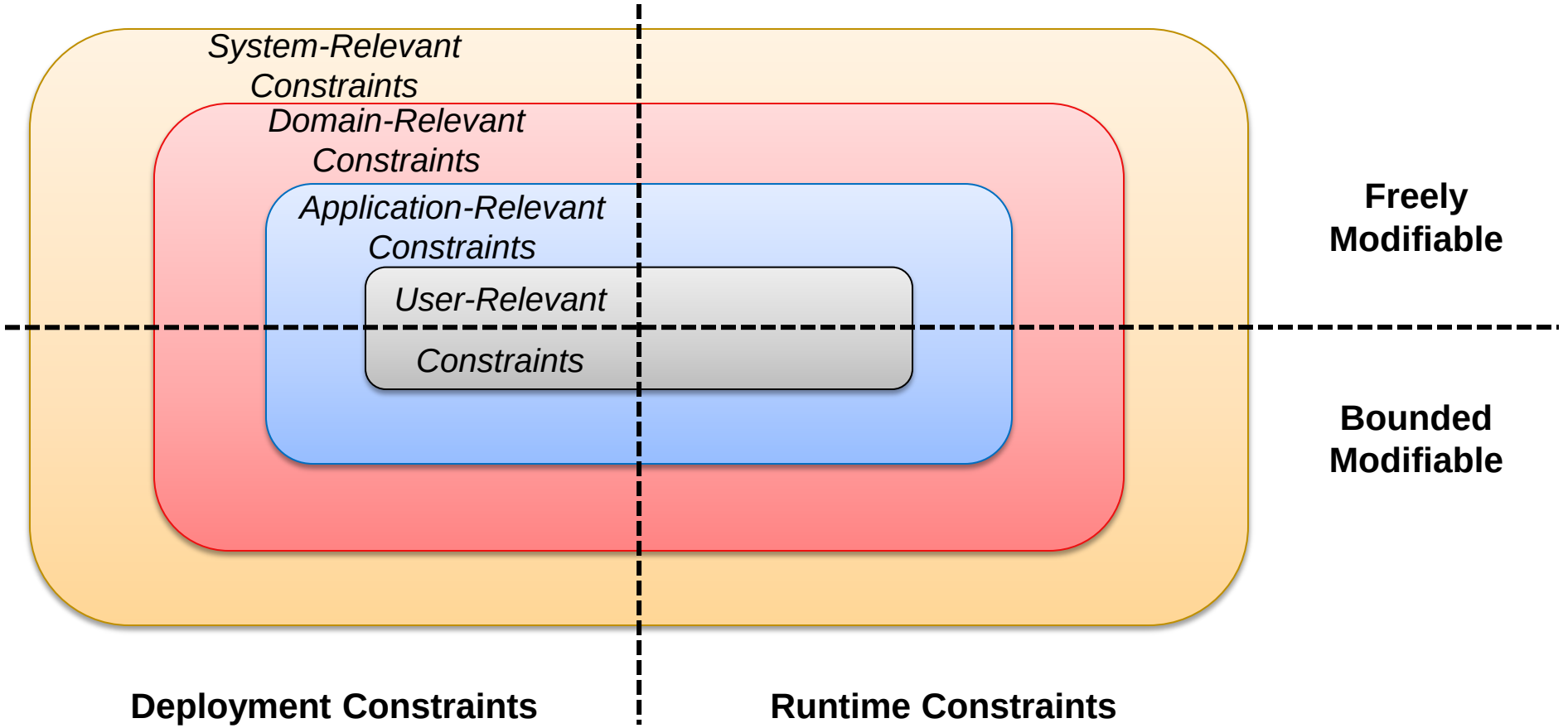
- **Many different (distributed) context data streams ➔ Use distributed DSPS**

- **Applications or domains may ask for specific functionality**
  - But still depending on general stream processing principles

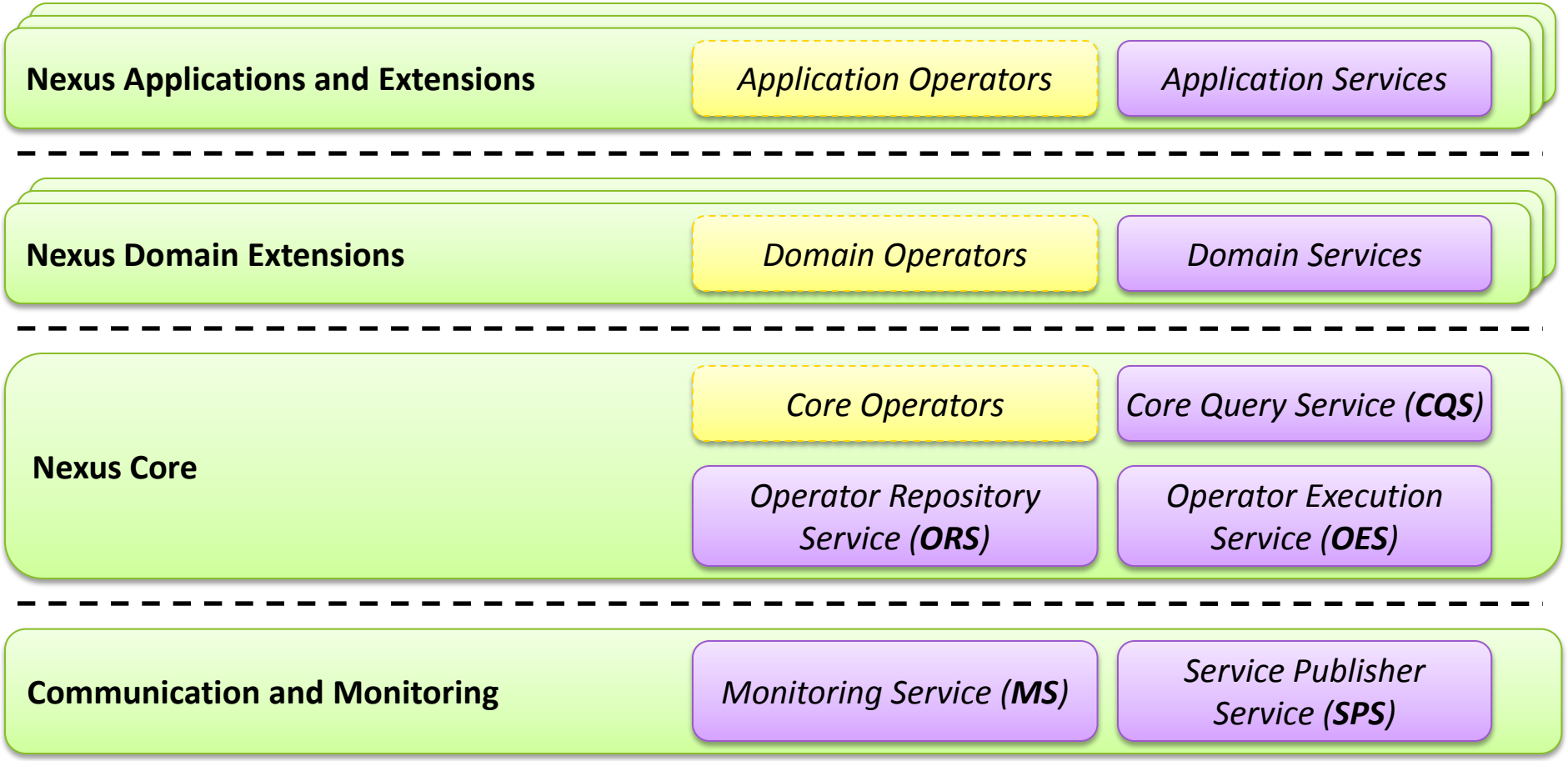
    ➔ **Reuse existing components, "do not reinvent the wheel"!**



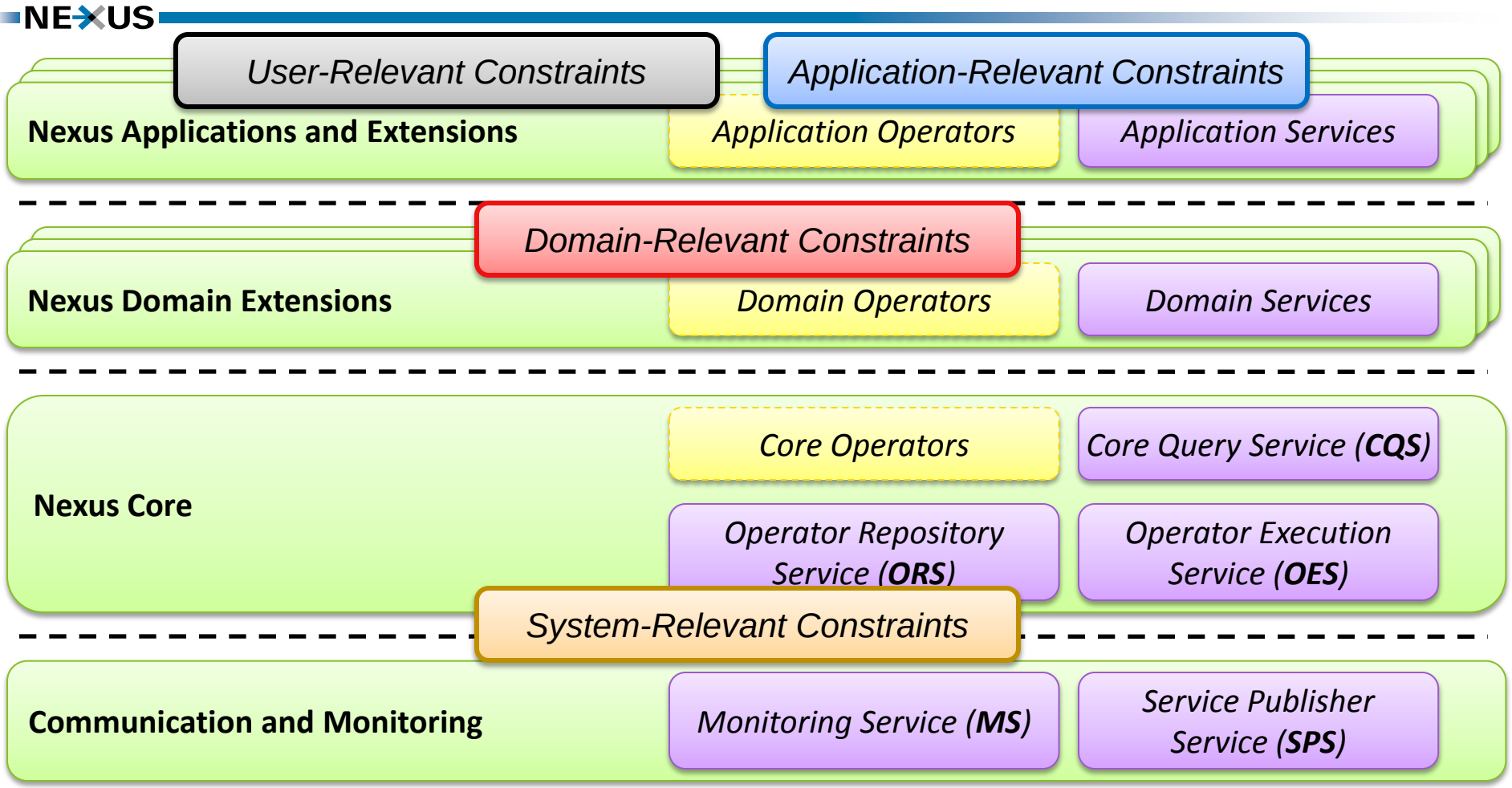
Constraints Classification



NexusDS Platform Overview



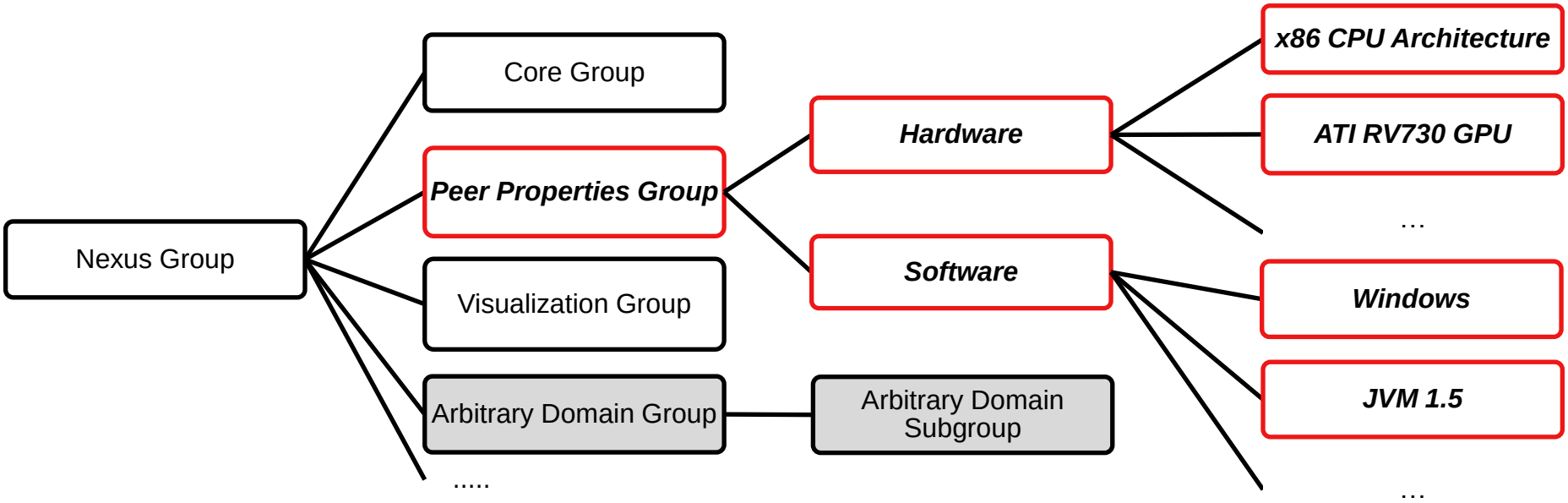
NexusDS Platform Overview



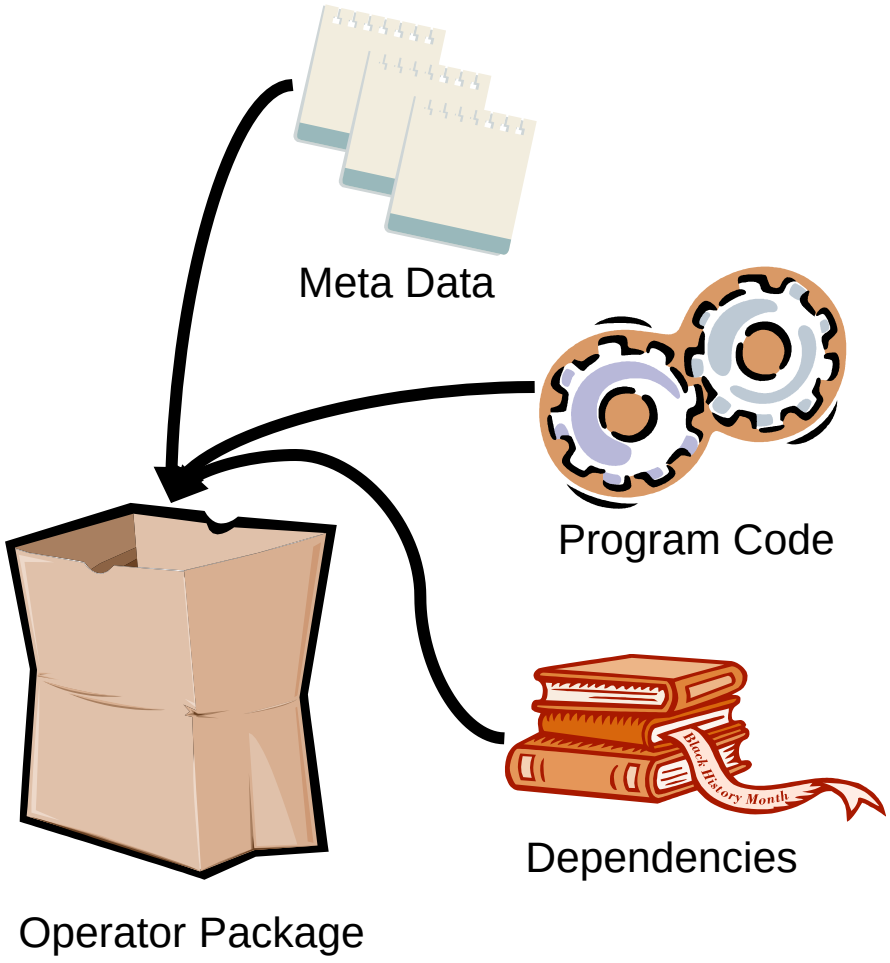
Classification of Runtime Environments

NEXUS

- Organization scheme of available environments to reduce deployment complexity
- Access corresponding groups to get available execution environments



Operator Packages – Overview



- **Meta Data**
  - Defines operator's properties
  - Descriptor
  - Requirements → ***Deployment***
  - Presets → ***Runtime***
- **Program Code**
  - Actual operator implementation
- **Dependencies**
  - Third party libraries

Constraint-Based Data Flow Graphs

User-Relevant
Application-Relevant
Domain-Relevant
System-Relevant

NEXUS

Deployment

***Name** = Mapping*
***Author** = Me*

Deployment

NodeID = 007…009
OperatorID = 123
Hardware= GPU

Data Stream:
Bus, Taxi, and
User Locations

Static Data:
Buildings & Streets
Points of Interest

Merge & Filtering:
Selection of
Nearby Objects

Mapping:
Rectangular Map
with 3D Buildungs

Rendering:
Projection
Rasterize

Client: Image Stream

e.g., Smartphones, PDAs, etc.

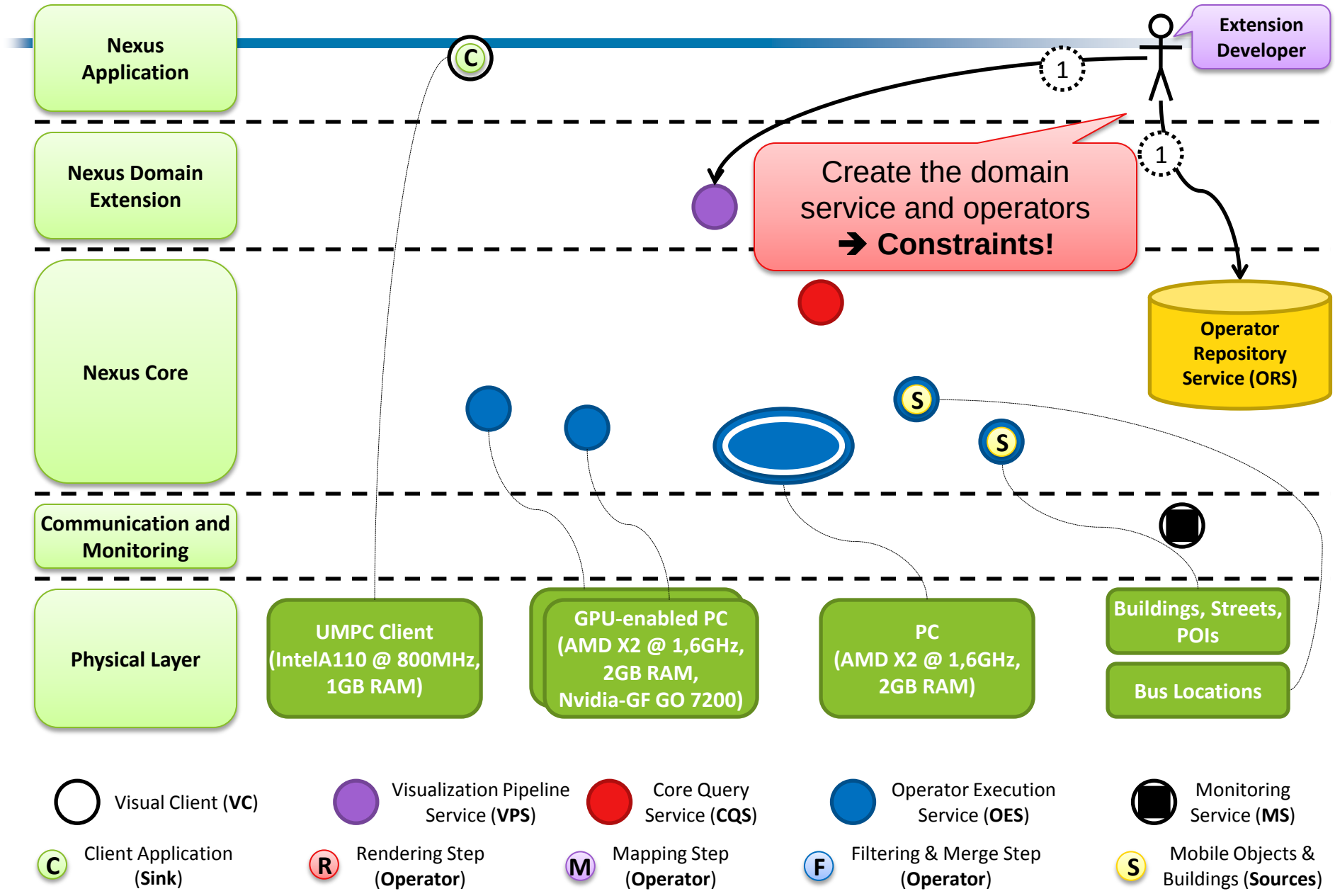
Runtime

Buffer = 100
Model_Detail = Medium

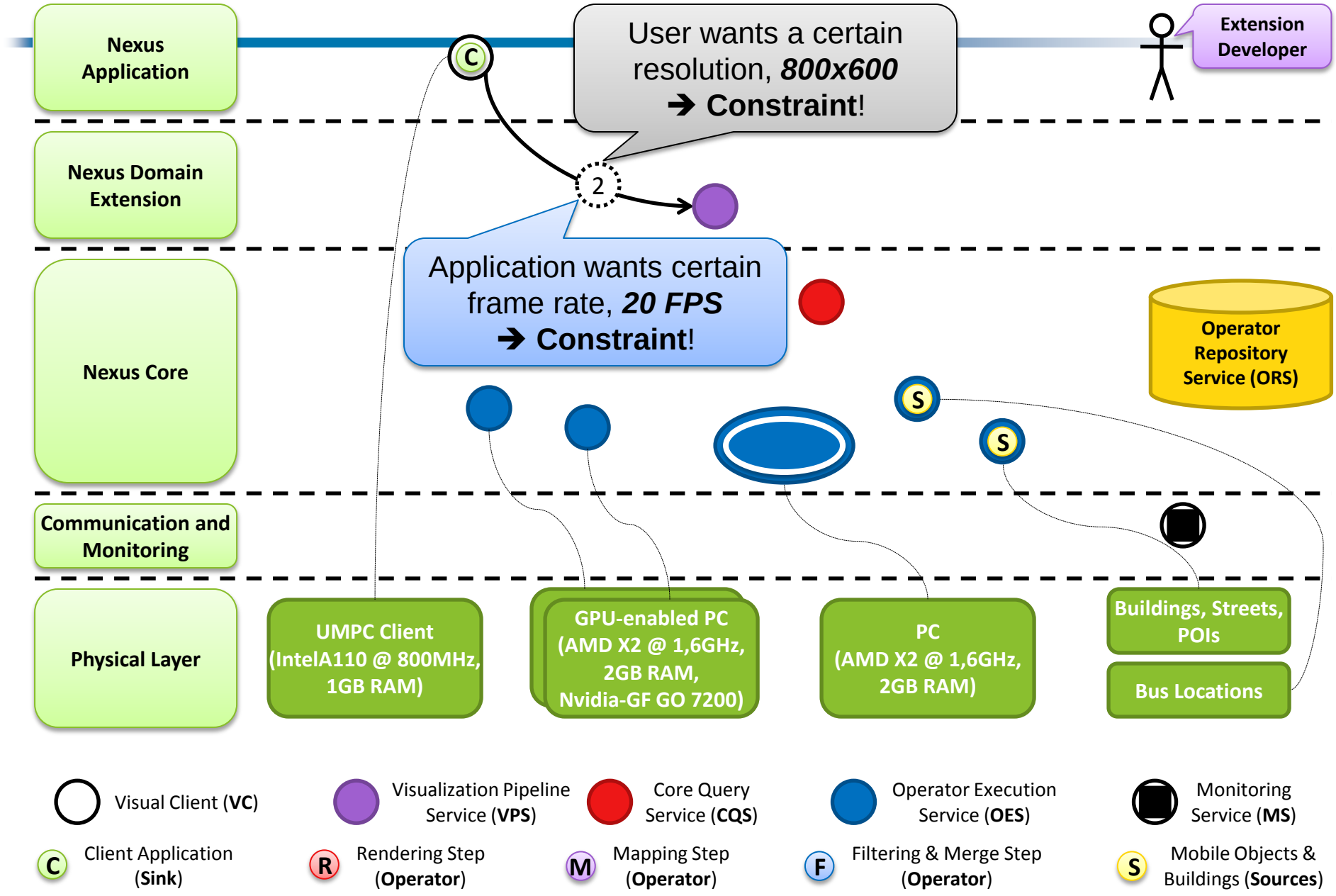
Runtime

Buffer = 10
Resolution = 800x600
FPS = 20

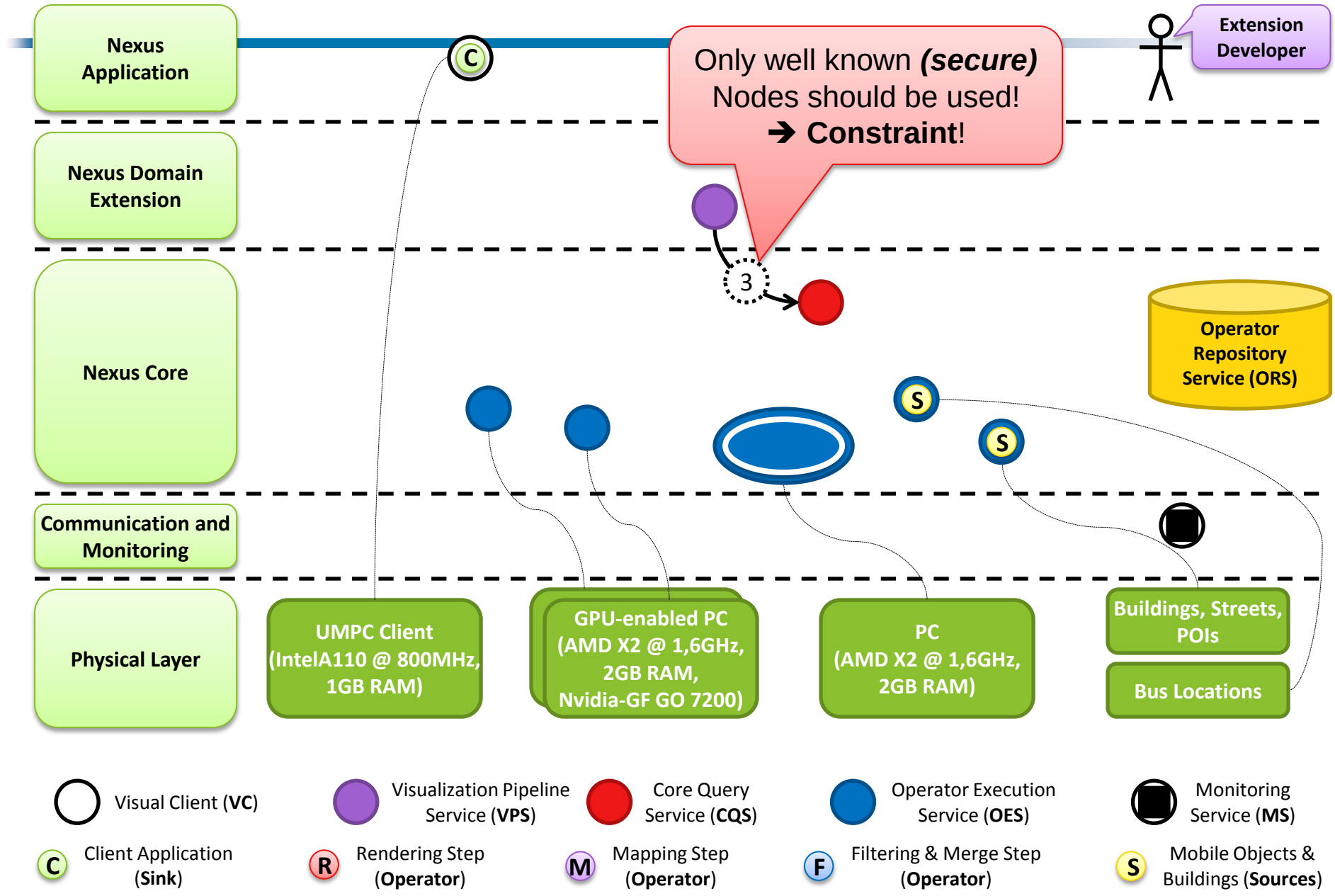
Deployment Model – Visualization Pipeline Sample Scenario



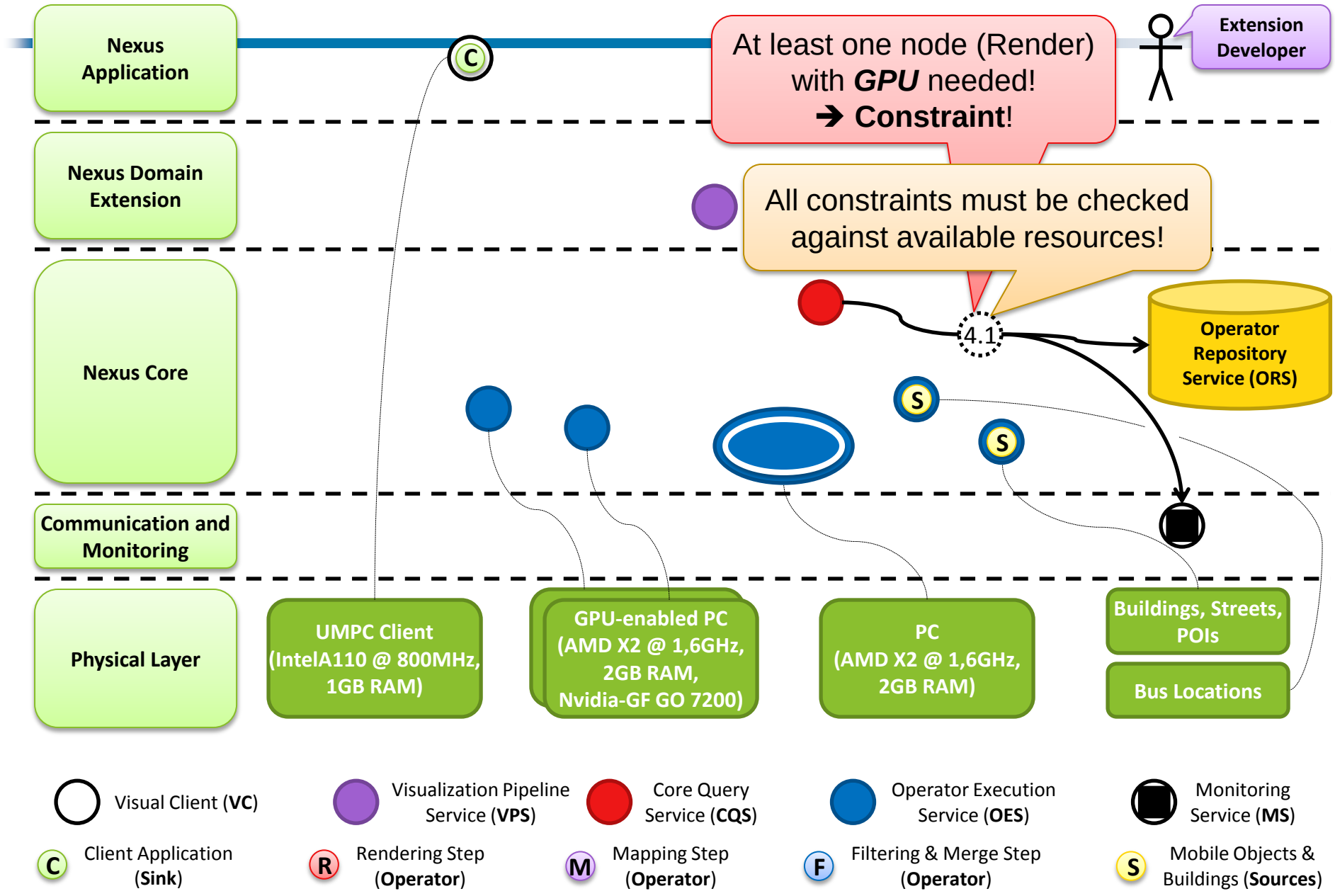
Deployment Model – Visualization Pipeline Sample Scenario



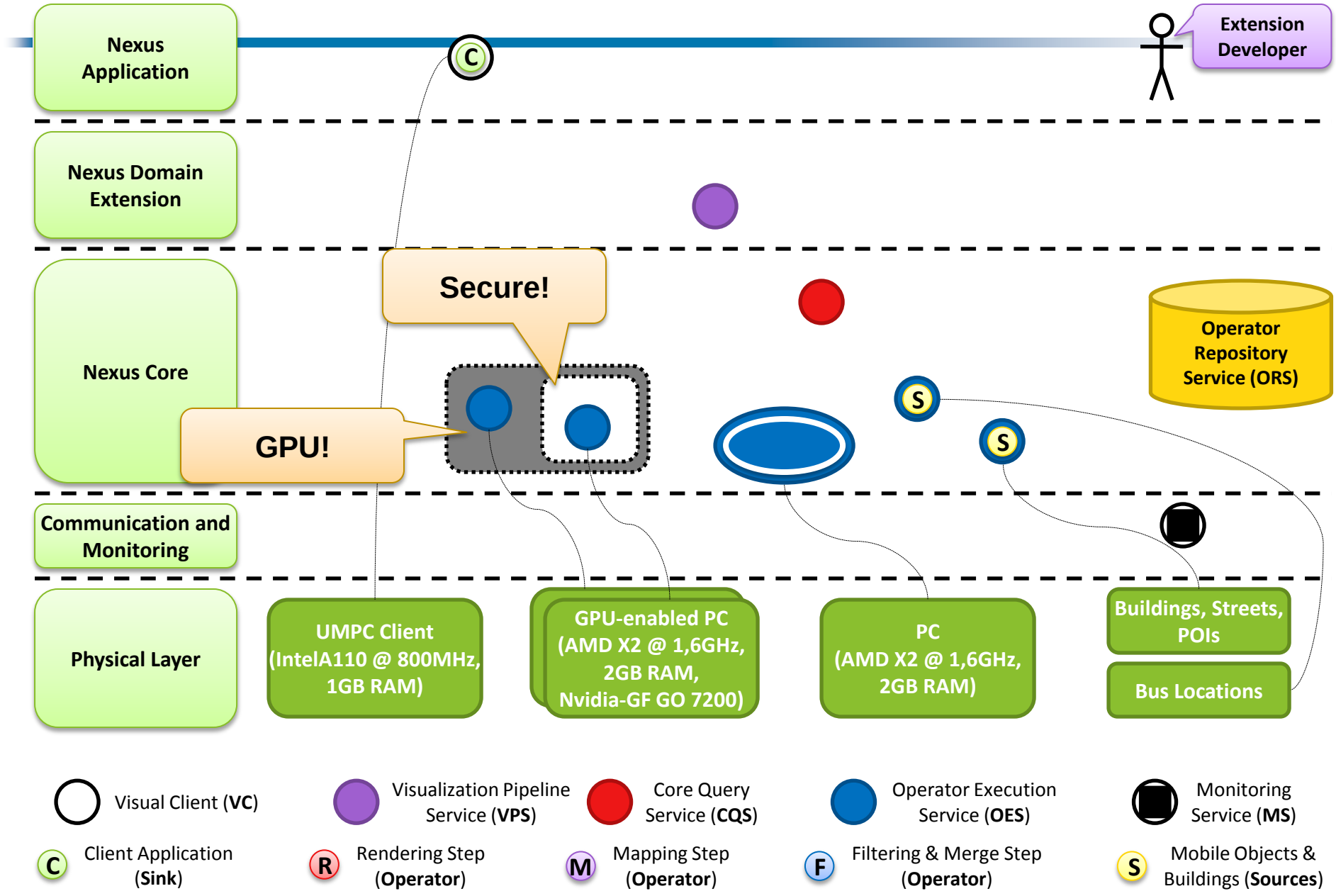
Deployment Model – Visualization Pipeline Sample Scenario



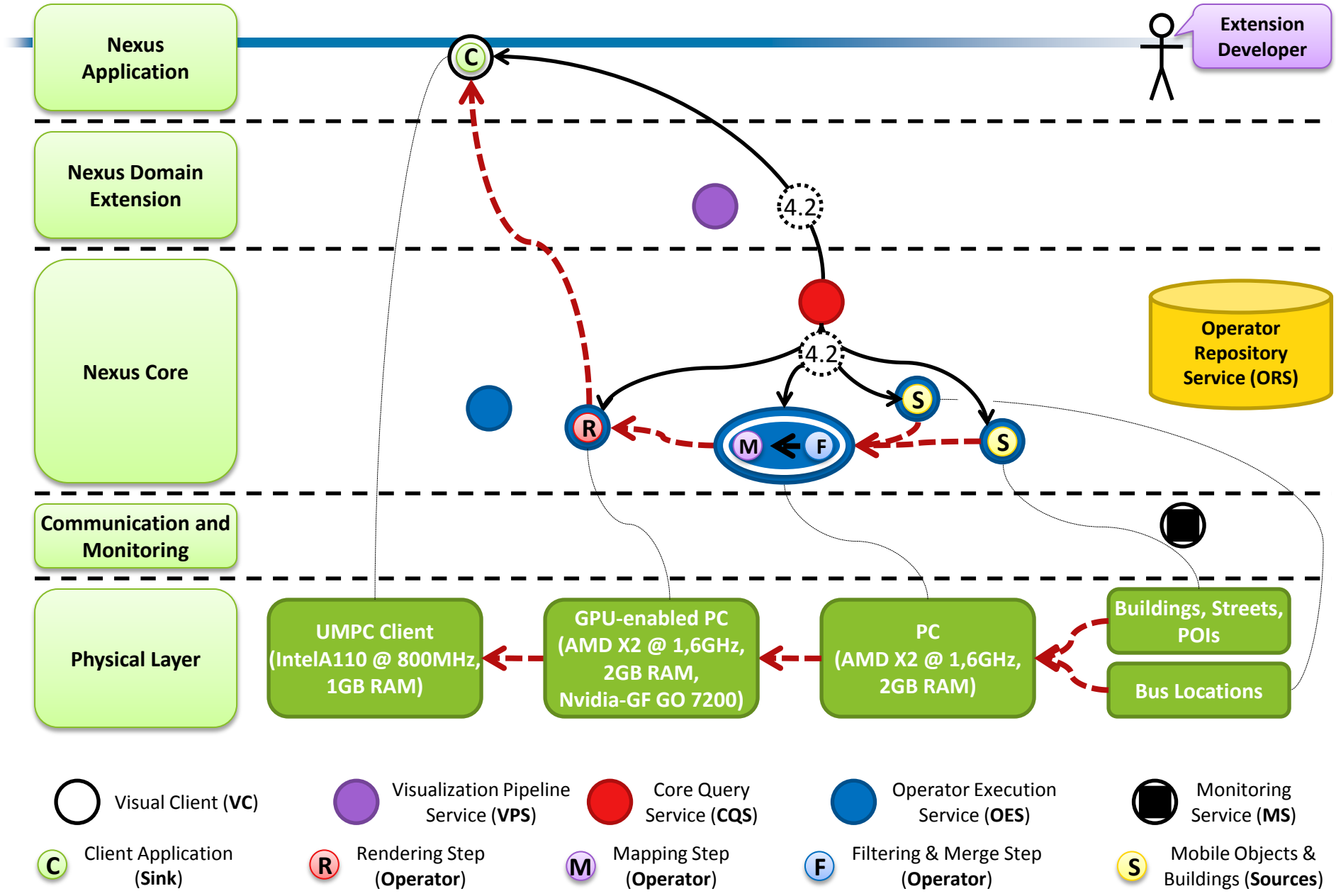
Deployment Model – Visualization Pipeline Sample Scenario



Deployment Model – Visualization Pipeline Sample Scenario



Deployment Model – Visualization Pipeline Sample Scenario



Conclusion and Future Work

- **Adaptation problem for non-trivial applications**

- **Identified requirements to satisfy needs of specific applications**
  - Based on non-trivial example of distributed visualization pipeline
  - Heterogeneous system topologies, highly domain-specific operators, operators connect their execution to explicit runtime and deployment restrictions

    ➔ **Constraints!**

- **Deployment is done according to pre-defined requirements on different levels (constraints)**

- **Future things to do**
  - Create suitable constraint-definition language to express complex constraint-links
  - Optimize deployment and execution process in terms of costs

Finish… And Flowers for You!