

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

NE➤US

Workshop on Scalable Stream Processing Systems (SSPS)

IPDPS 2010, Atlanta/Georgia, USA

19.04.2010

Nazario Cipriani, Carlos Lübbe, Alexander Moosbrugger

2 Data Stream Processing Systems – Some Foundations

NEUS

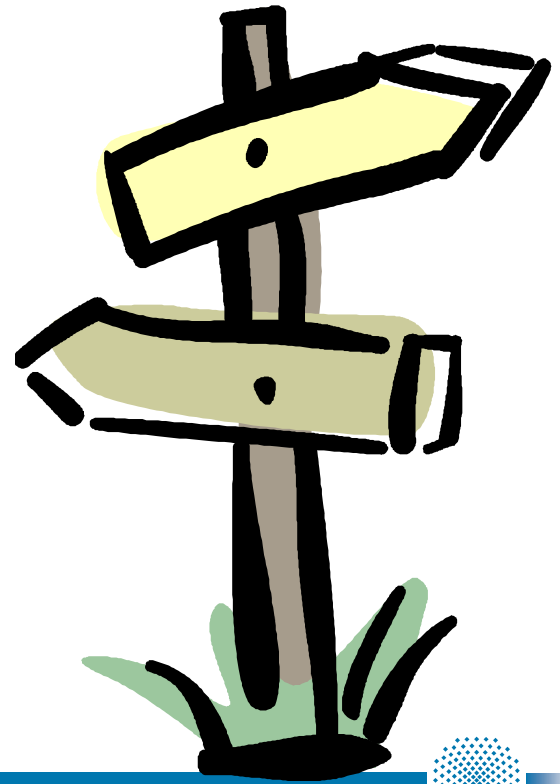
- **Context-aware applications need context data to be processed**
 - Context-aware systems utterly heterogeneous
 - Nearby sensors produce context data streams that needs to be processed online
 - Not convenient to first store and then process offline due to high data volume
 - ➔ **Use *Data Stream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



3 Agenda

NE~~X~~US

- **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**



4 Sample Scenario – Distributed Visualization Pipeline

NEUS

Sources

Operators

Sinks

Visualization-Pipeline

User Location Updates

Data Stream:
Bus, Taxi, and
User Locations

Merge & Filtering:
Selection of
Nearby Objects

Mapping:
Rectangular Map
with 3D Buildings

Rendering:
Projection
Rasterize

Client: Image Stream



e.g., Smartphones, PDAs, etc.

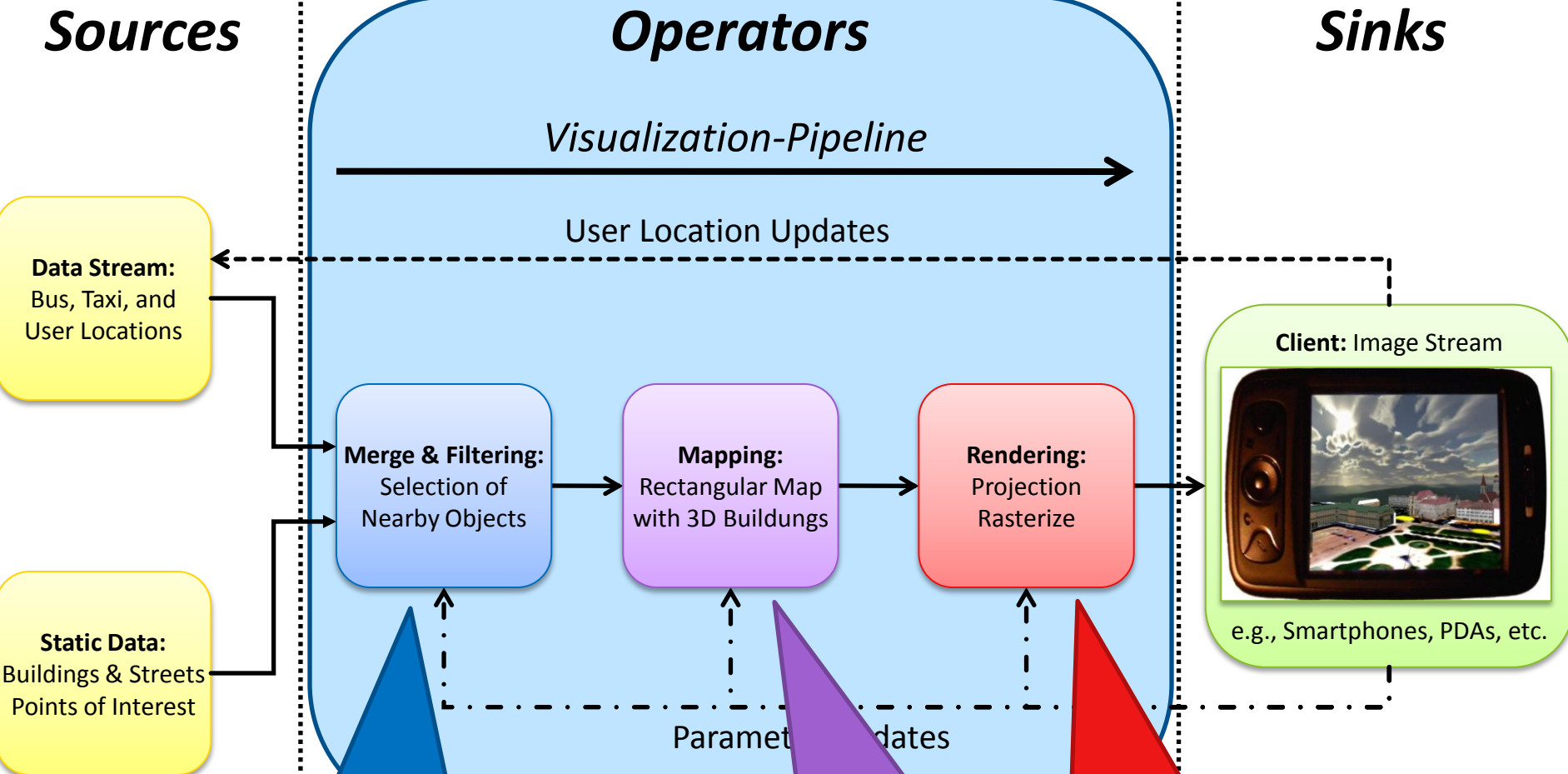
Static Data:
Buildings & Streets
Points of Interest

Parameter Updates



5 Sample Scenario – Distributed Visualization Pipeline

NEUS



Built-In Operator can be used here
→ *Exploit already existing operators!*

Custom Operator to define the *Mapping Process* for Data to *Rendering Primitives!*

Dedicated hardware can improve performance
→ *Exploit specialized hardware is reasonable!*

6 Sample Scenario – Distributed Visualization Pipeline

NEUS

Sources

Operators

Sinks

Visualization-Pipeline

User Location Updates

Data Stream:
Bus, Taxi, and
User Locations

Merge & Filtering:
Selection of
Nearby Objects

Mapping:
Rectangular Map
with 3D Buildings

Rendering:
Projection
Rasterize

Client: Image Stream



e.g., Smartphones, PDAs, etc.

Static Data:
Buildings & Streets
Points of Interest

Parameter Updates

Mobile devices still have
limited capabilities (energy)
→ *Outsource processes
if possible!*

7 Requirements for DSPS to Support Non-Trivial Applications?

NE~~X~~US

■ 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



8 Adaptation Gap between Applications and DSPS

NEUS

- 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”!

Data Stream
Processing System
(DSPS)

*Domain-specific Data
Stream Application*

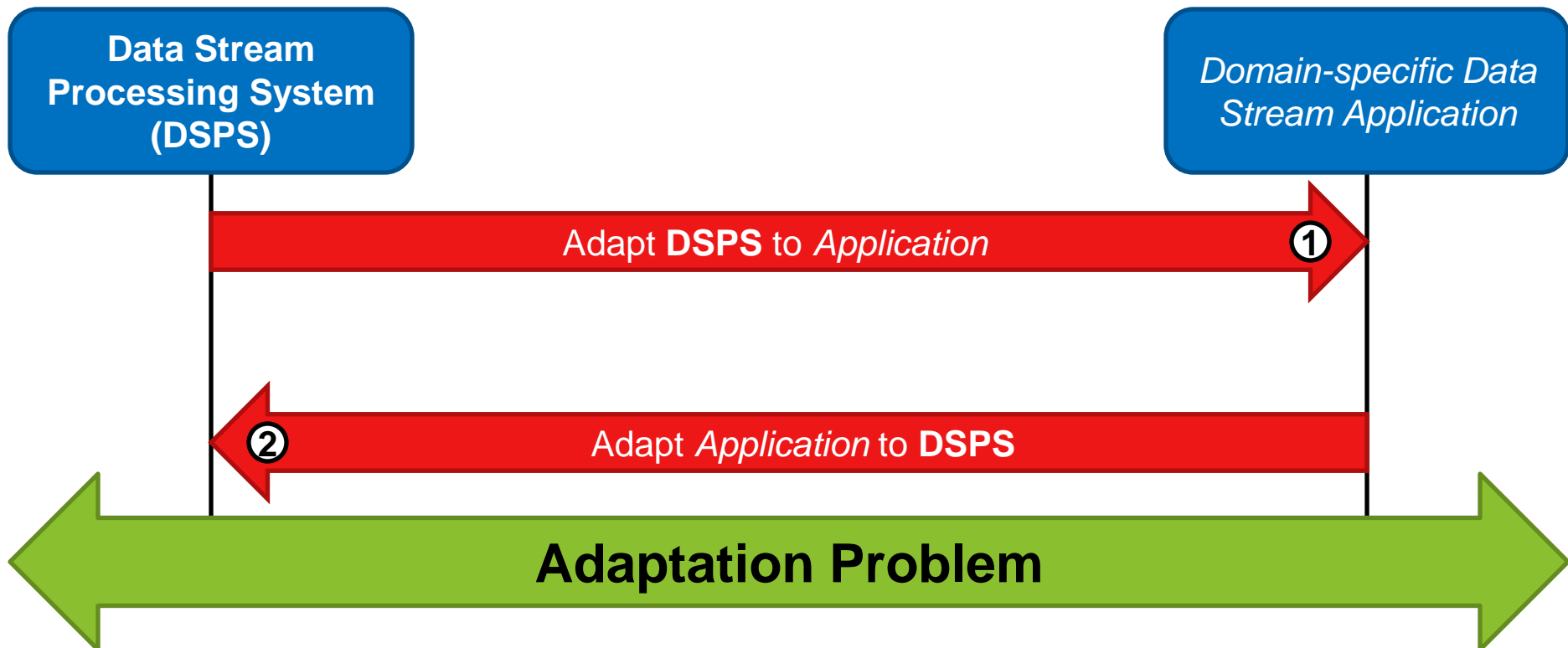
Adaptation Problem

The diagram shows two blue rounded rectangular boxes at the top. The left box contains the text 'Data Stream Processing System (DSPS)' and the right box contains the text 'Domain-specific Data Stream Application'. Two vertical black lines extend downwards from the bottom center of each box. These lines meet a large, horizontal green double-headed arrow at the bottom. The text 'Adaptation Problem' is centered within this green arrow.

9 Adaptation Gap between Applications and DSPS

NEUS

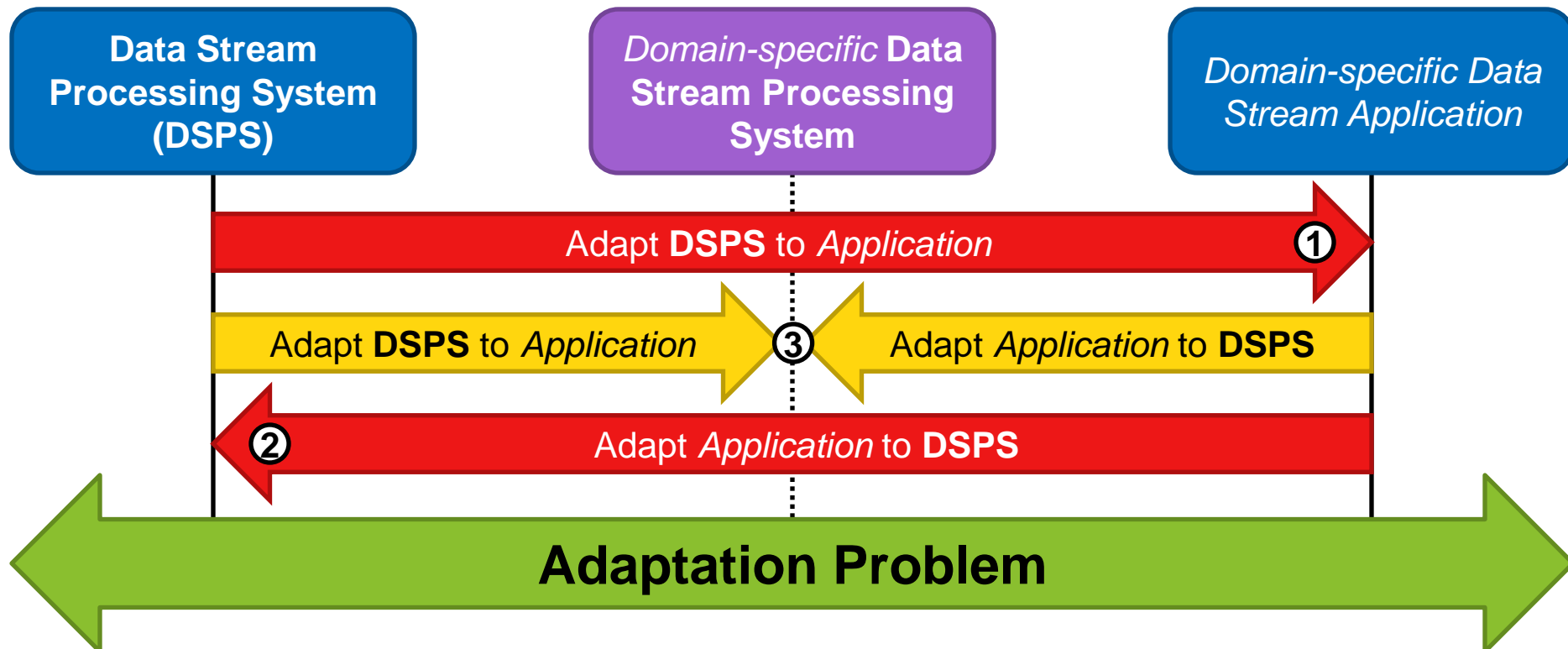
- Many different (distributed) context data streams → Use distributed DSPS
- Applications or domains may ask for specific functionality
 - But still depending on general stream processing principles
 - Reuse existing components, “do not reinvent the wheel”!



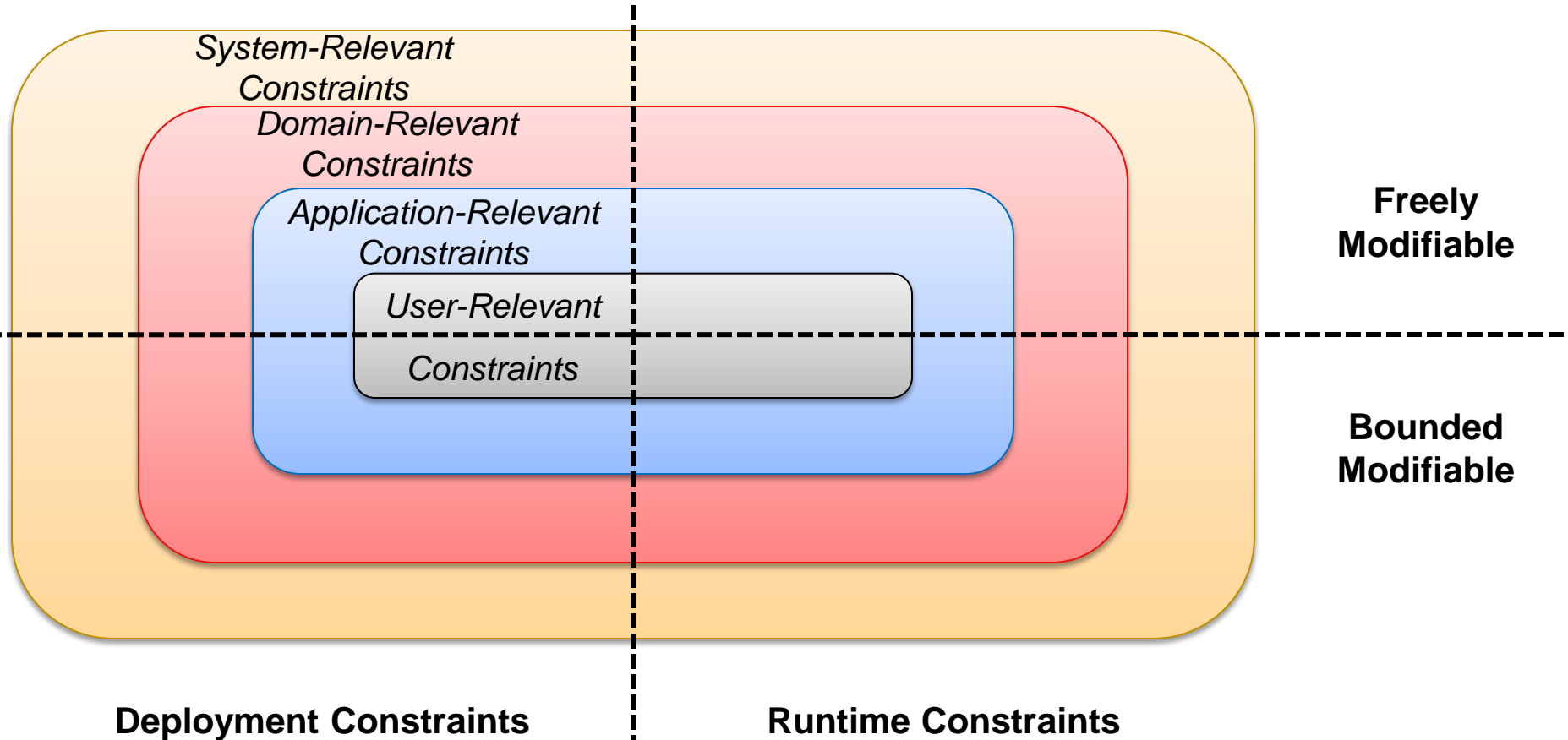
10 Adaptation Gap between Applications and DSPS

NEUS

- 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”!



11 Constraints Classification



12 NexusDS Platform Overview

NE~~X~~US

Nexus Applications and Extensions

Application Operators

Application Services

Nexus Domain Extensions

Domain Operators

Domain Services

Nexus Core

Core Operators

Core Query Service (CQS)

*Operator Repository
Service (ORS)*

*Operator Execution
Service (OES)*

Communication and Monitoring

Monitoring Service (MS)

*Service Publisher
Service (SPS)*



13 NexusDS Platform Overview

NE~~X~~US

User-Relevant Constraints

Application-Relevant Constraints

Nexus Applications and Extensions

Application Operators

Application Services

Domain-Relevant Constraints

Nexus Domain Extensions

Domain Operators

Domain Services

Nexus Core

Core Operators

Core Query Service (CQS)

Operator Repository Service (ORS)

Operator Execution Service (OES)

System-Relevant Constraints

Communication and Monitoring

Monitoring Service (MS)

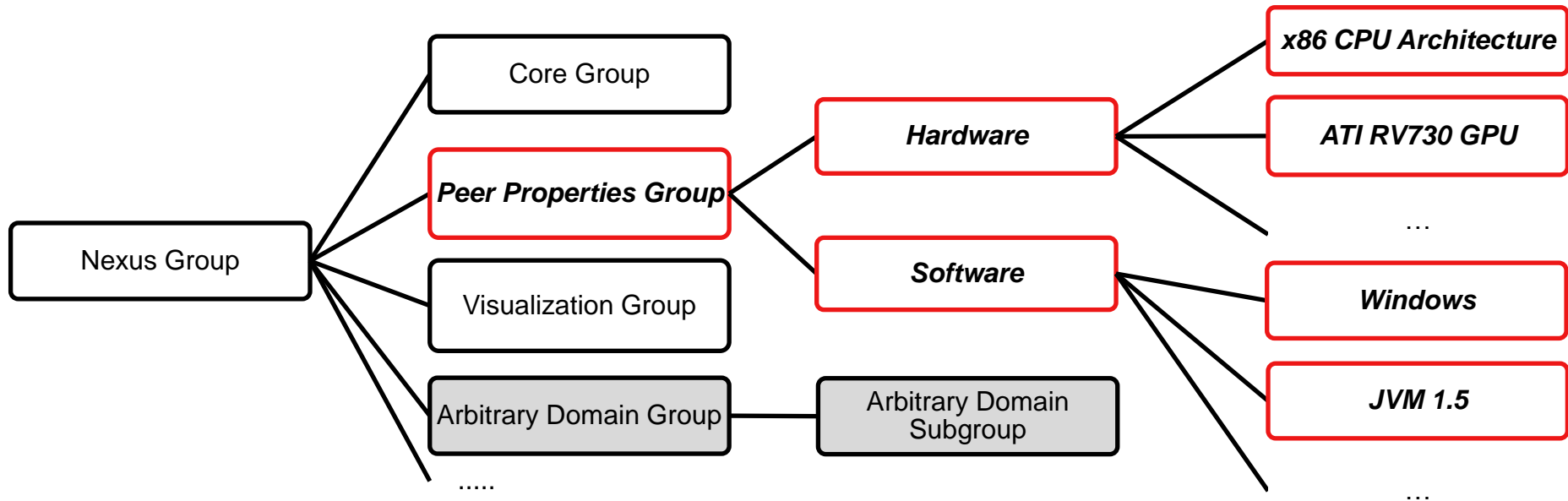
Service Publisher Service (SPS)

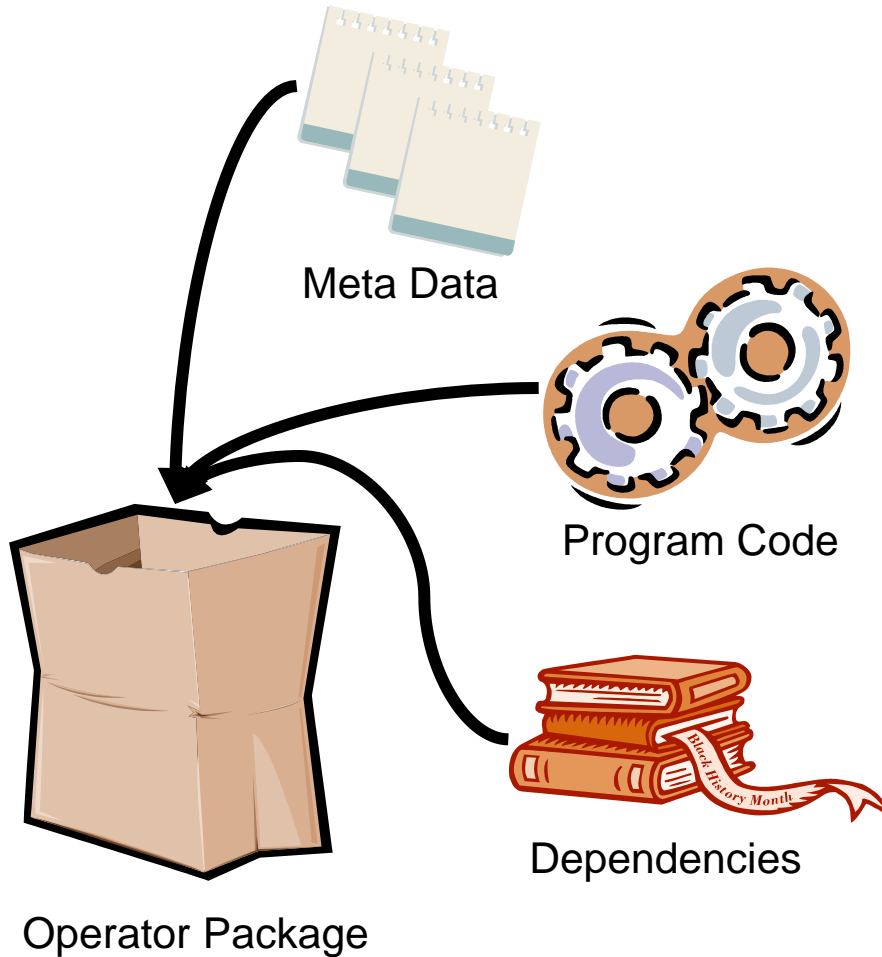


14 Classification of Runtime Environments

NE~~X~~US

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





■ Meta Data

- Defines operator's properties
- Descriptor
- Requirements → **Deployment**
- Presets → **Runtime**

■ Program Code

- Actual operator implementation

■ Dependencies

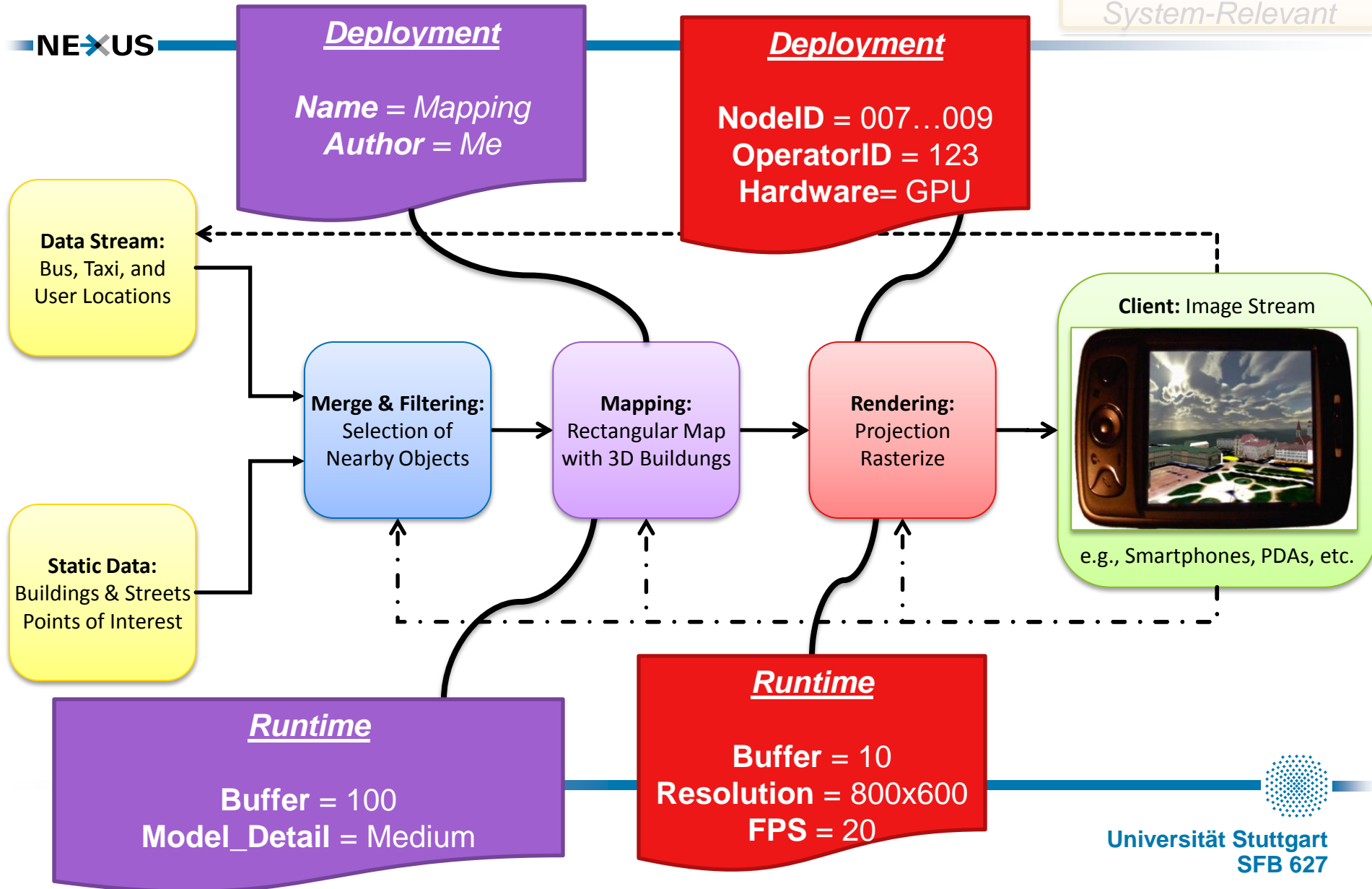
- Third party libraries



16 Constraint-Based Data Flow Graphs

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

NE~~X~~US



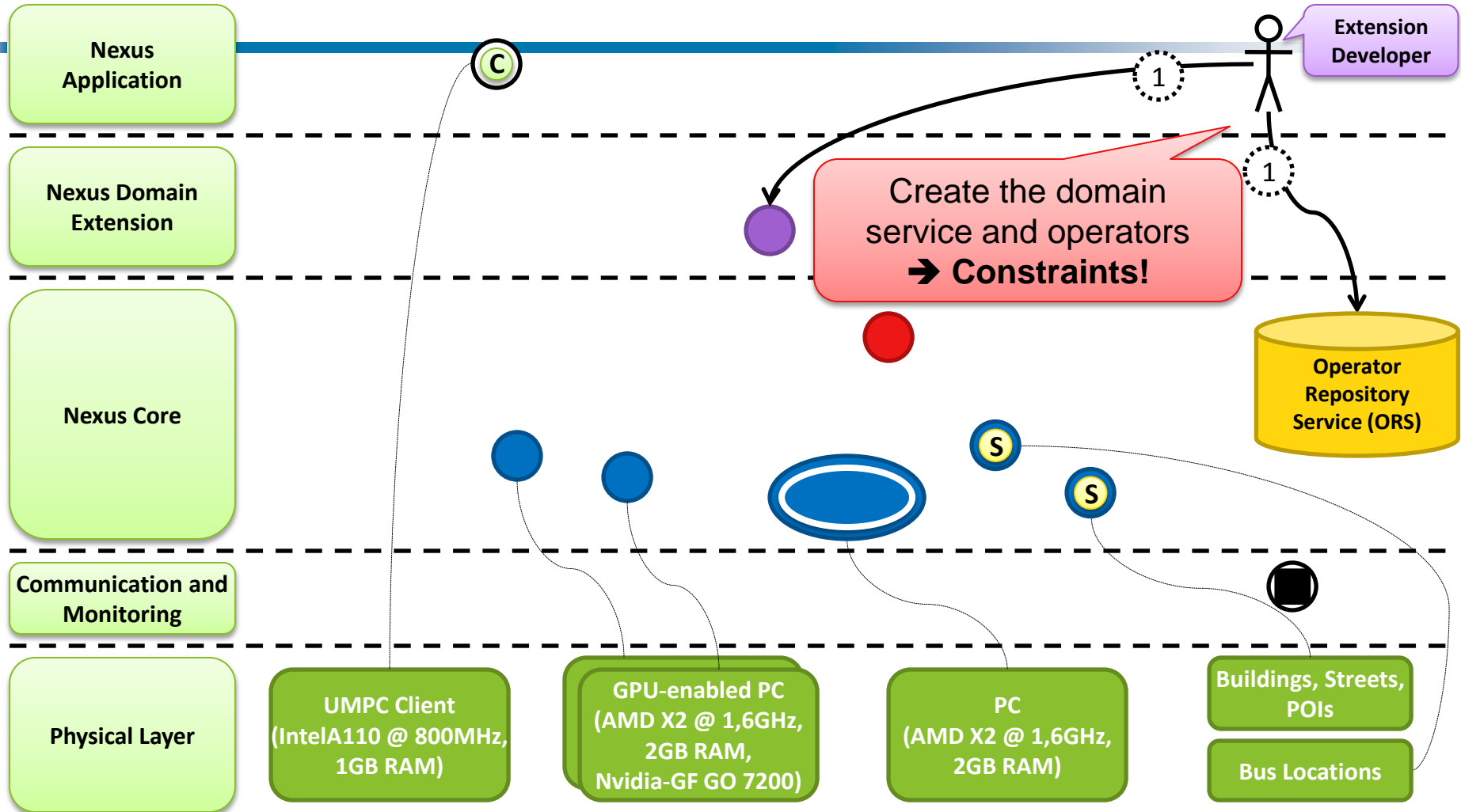
User-Relevant

Application-Relevant

Domain-Relevant

System-Relevant

17 Deployment Model – Visualization Pipeline Sample Scenario



○ Visual Client (VC)

● Visualization Pipeline Service (VPS)

● Core Query Service (CQS)

● Operator Execution Service (OES)

● Monitoring Service (MS)

○ Client Application (Sink)

○ Rendering Step (Operator)

○ Mapping Step (Operator)

○ Filtering & Merge Step (Operator)

○ Mobile Objects & Buildings (Sources)

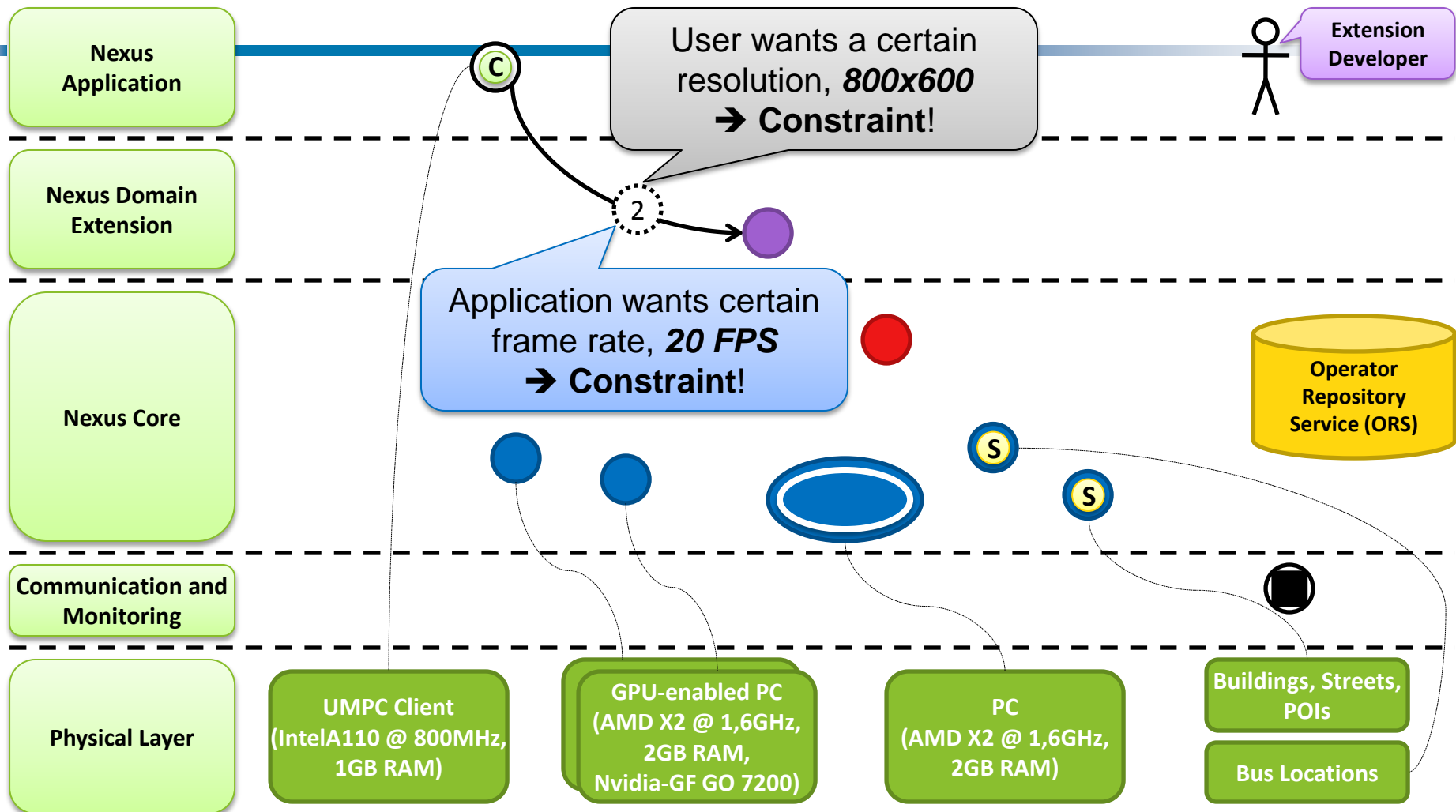
User-Relevant

Application-Relevant

Domain-Relevant

System-Relevant

18 Deployment Model – Visualization Pipeline Sample Scenario



○ Visual Client (VC)

● Visualization Pipeline Service (VPS)

● Core Query Service (CQS)

● Operator Execution Service (OES)

● Monitoring Service (MS)

○ Client Application (Sink)

○ Rendering Step (Operator)

○ Mapping Step (Operator)

○ Filtering & Merge Step (Operator)

○ Mobile Objects & Buildings (Sources)

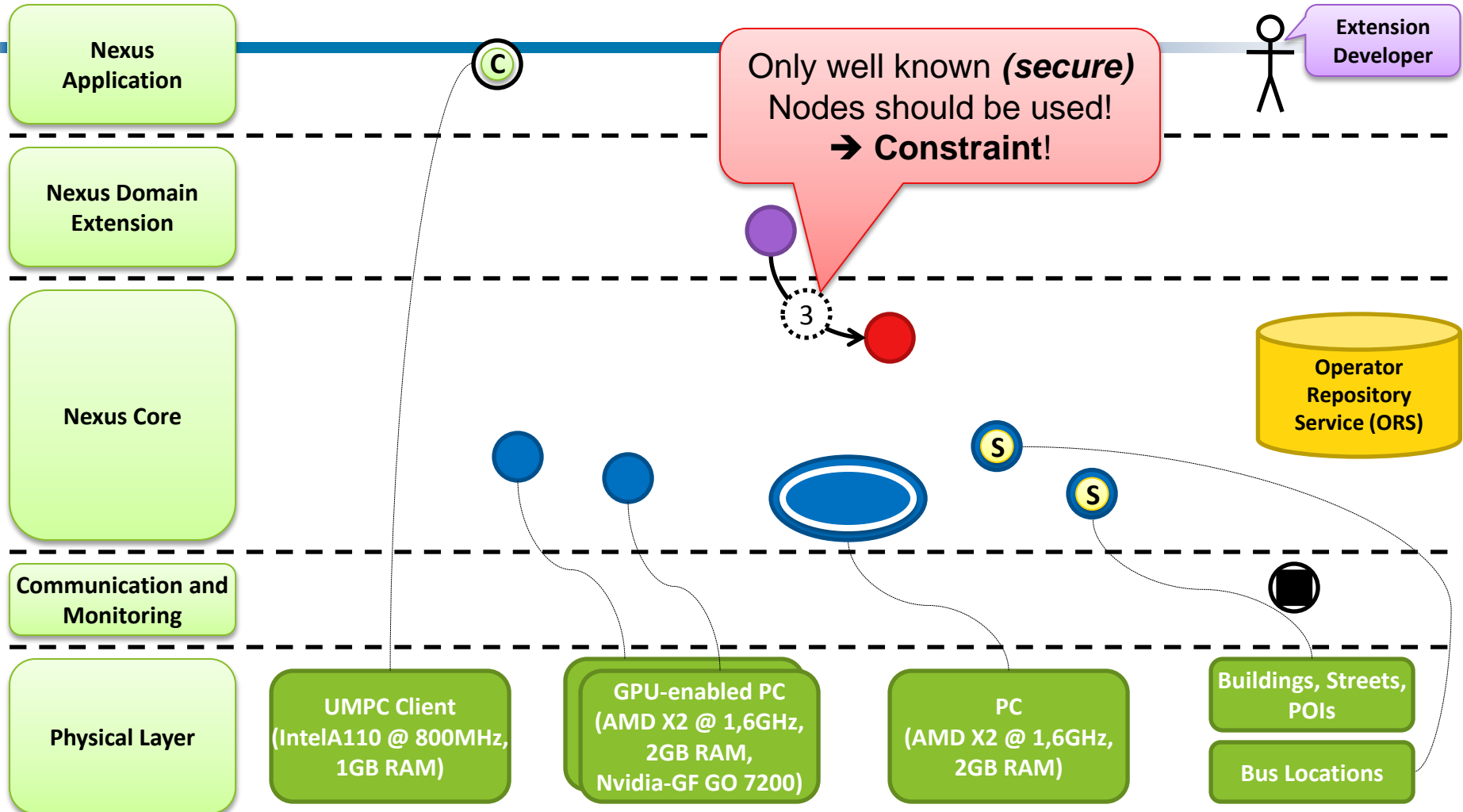
User-Relevant

Application-Relevant

Domain-Relevant

System-Relevant

19 Deployment Model – Visualization Pipeline Sample Scenario



Only well known (*secure*)
Nodes should be used!
→ **Constraint!**

Extension Developer

Operator Repository Service (ORS)

Communication and Monitoring

Physical Layer

UMPC Client
(IntelA110 @ 800MHz,
1GB RAM)

GPU-enabled PC
(AMD X2 @ 1,6GHz,
2GB RAM,
Nvidia-GF GO 7200)

PC
(AMD X2 @ 1,6GHz,
2GB RAM)

Buildings, Streets,
POIs
Bus Locations

Visual Client (VC)

Visualization Pipeline Service (VPS)

Core Query Service (CQS)

Operator Execution Service (OES)

Monitoring Service (MS)

Client Application (Sink)

Rendering Step (Operator)

Mapping Step (Operator)

Filtering & Merge Step (Operator)

Mobile Objects & Buildings (Sources)

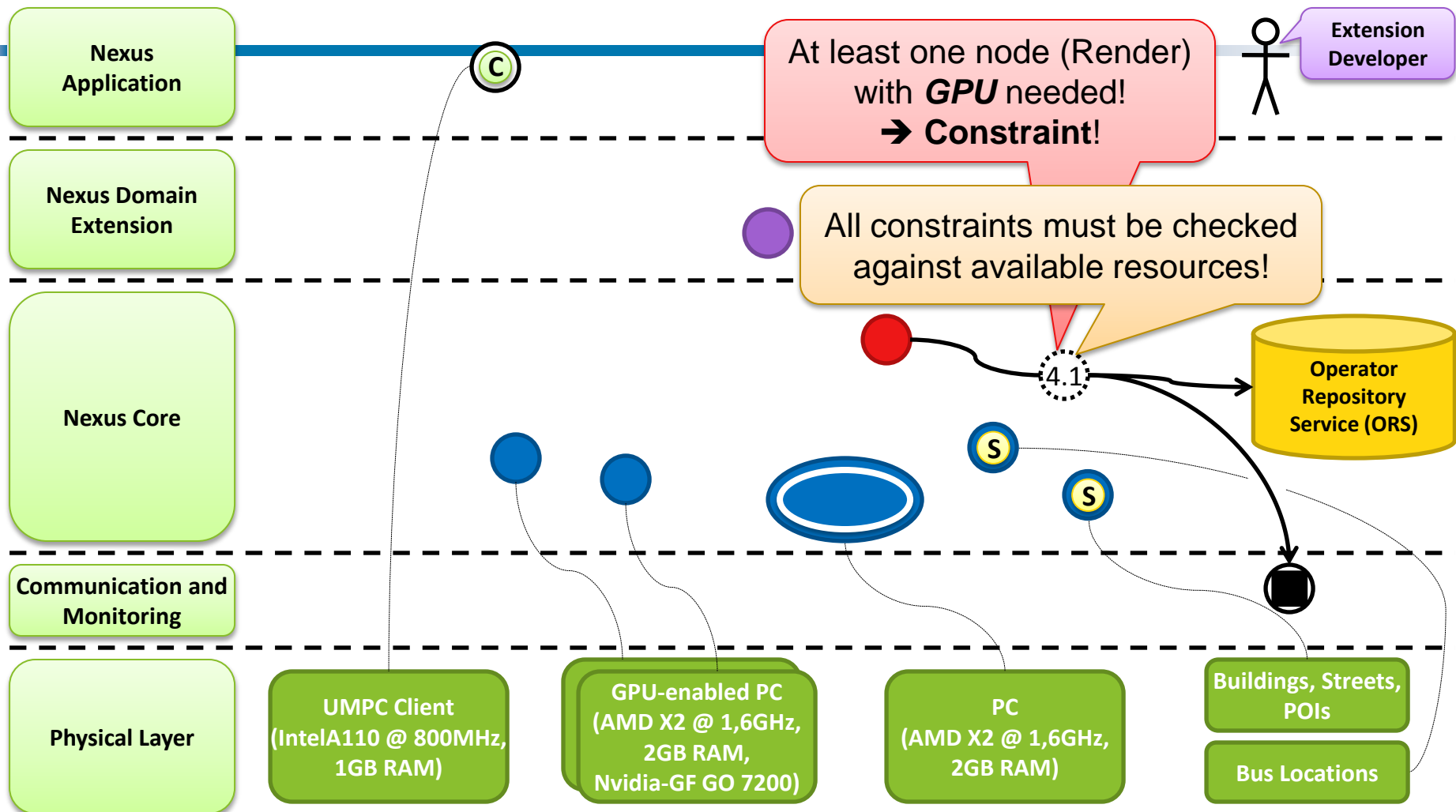
User-Relevant

Application-Relevant

Domain-Relevant

System-Relevant

Deployment Model – Visualization Pipeline Sample Scenario



Visual Client (VC)

Visualization Pipeline Service (VPS)

Core Query Service (CQS)

Operator Execution Service (OES)

Monitoring Service (MS)

Client Application (Sink)

Rendering Step (Operator)

Mapping Step (Operator)

Filtering & Merge Step (Operator)

Mobile Objects & Buildings (Sources)

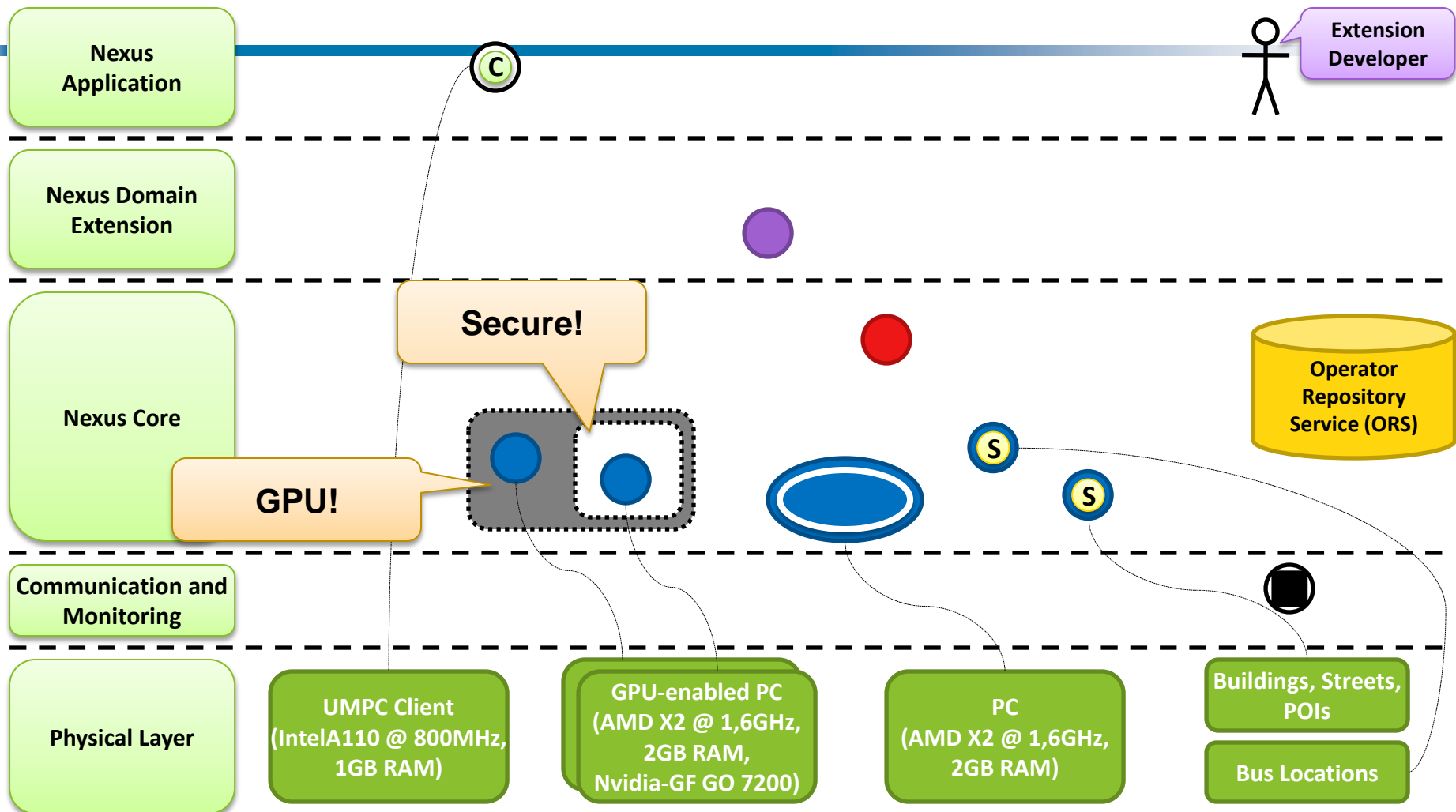
User-Relevant

Application-Relevant

Domain-Relevant

System-Relevant

21 Deployment Model – Visualization Pipeline Sample Scenario



○ Visual Client (VC)

● Visualization Pipeline Service (VPS)

● Core Query Service (CQS)

● Operator Execution Service (OES)

● Monitoring Service (MS)

○ Client Application (Sink)

○ Rendering Step (Operator)

○ Mapping Step (Operator)

○ Filtering & Merge Step (Operator)

○ Mobile Objects & Buildings (Sources)

- **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



24 Finish... And Flowers for You!

NEXUS

