Integrated Clinical Environment Use Case

Smart Healthcare

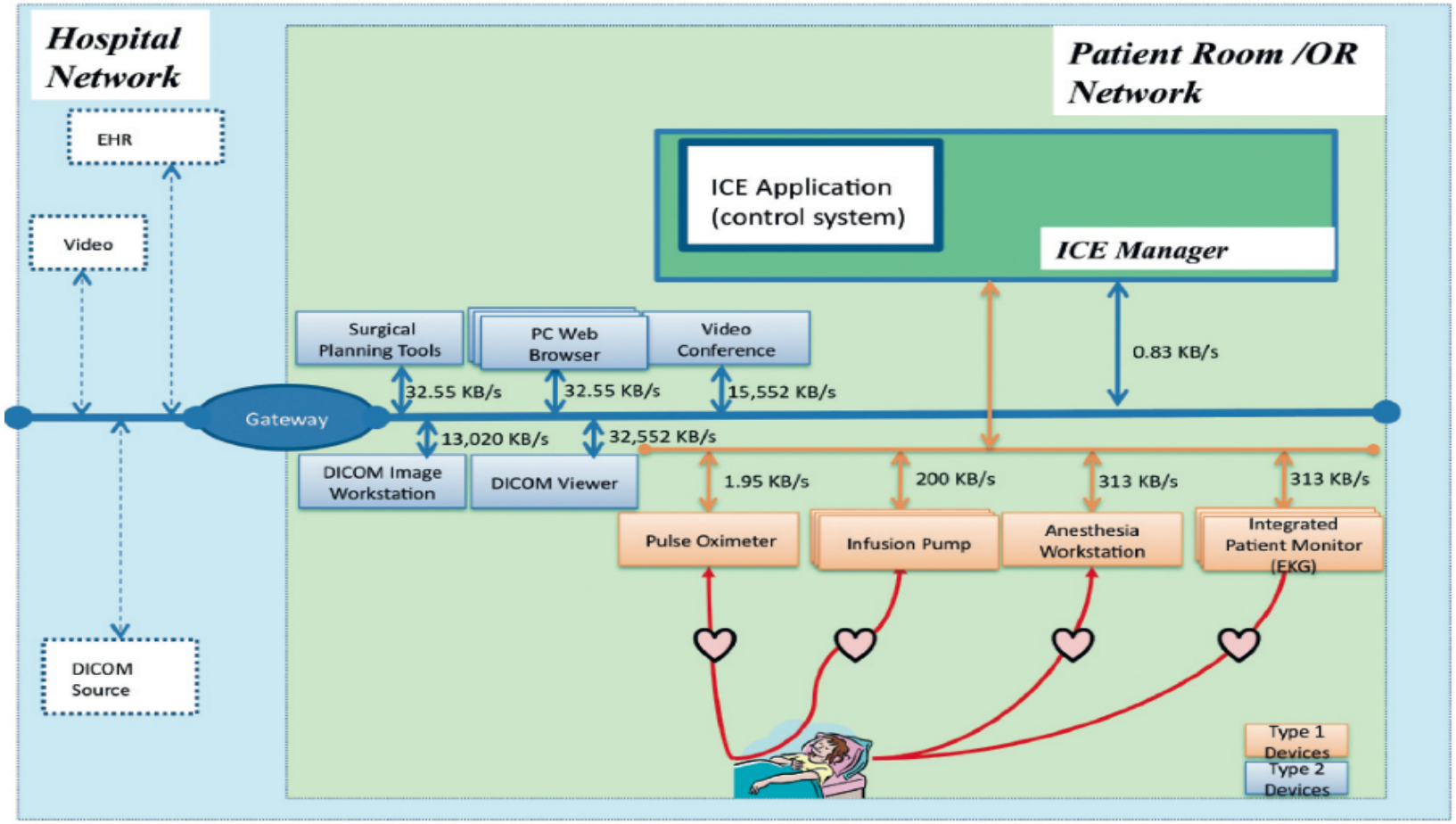
Real Time Innovations

Agenda

- OpenICE project
- How Open Standard DDS can help
- Technical Details of DDS
- Tutorial

What is an Integrated Clinical Environment?

An open medical device and data integration platform @ Mass General Hospital

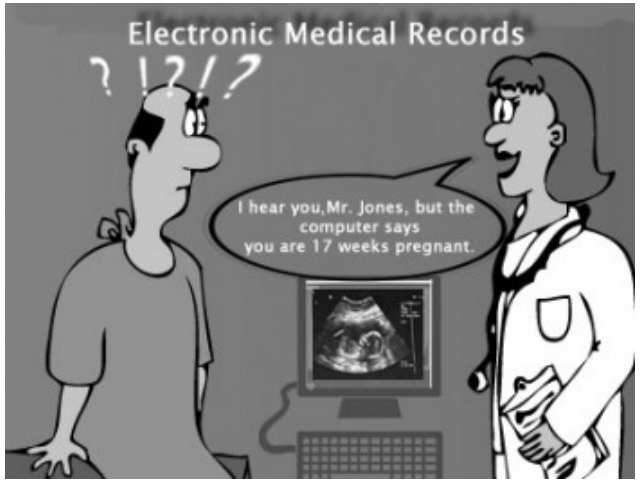


Motivation

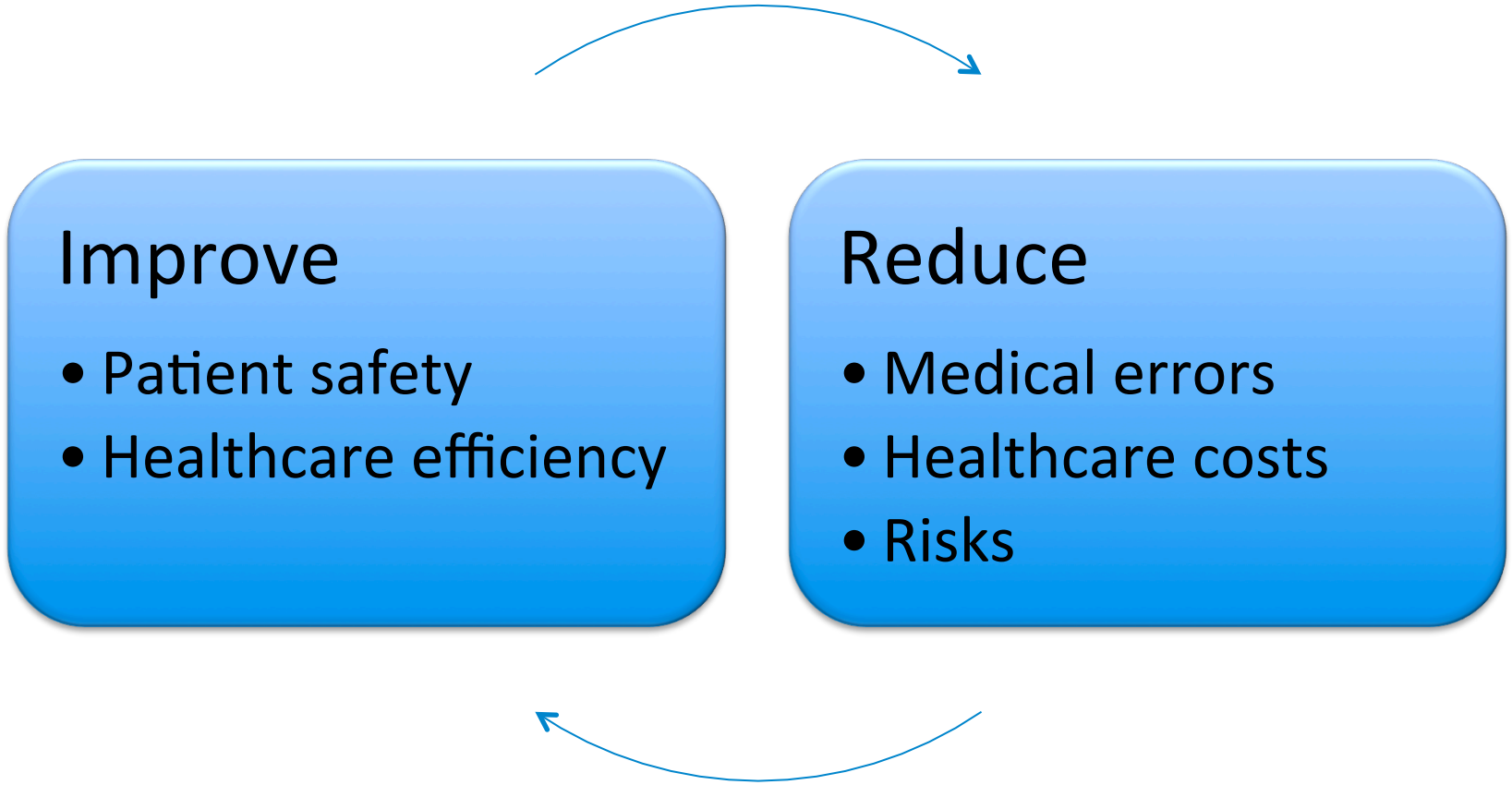
Hospital error is the 3th leading cause of preventable death

Currently, patient monitoring and care-delivery systems provide no cross-correlation of data from different devices

Existing EMR data is not comprehensive and accurate



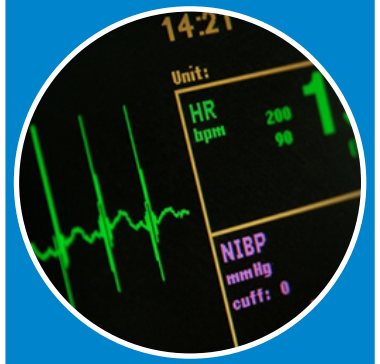
Value: Improving People, Clinical Culture and Practices



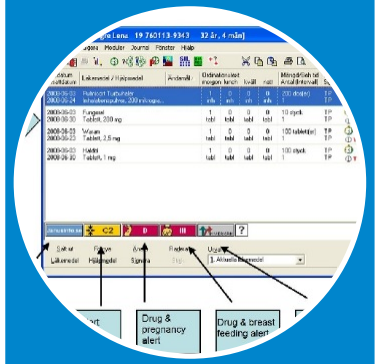
Key Clinical Scenarios

Patient Controlled Analgesia(PCA)

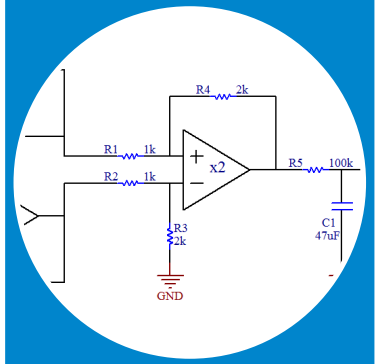
Operating Room to ICU Transfer



Smart Alarm Generation



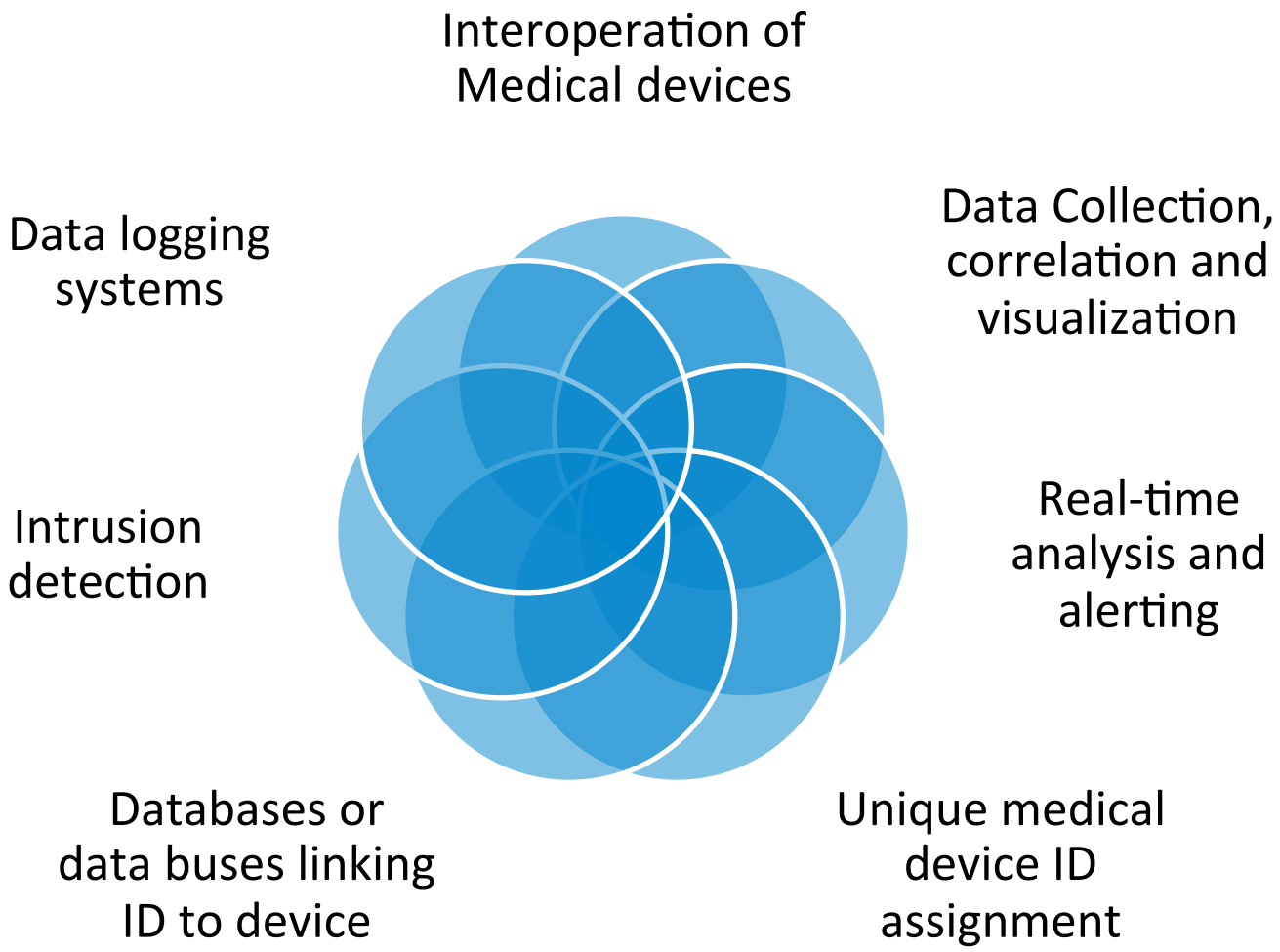
Clinical Decision Support



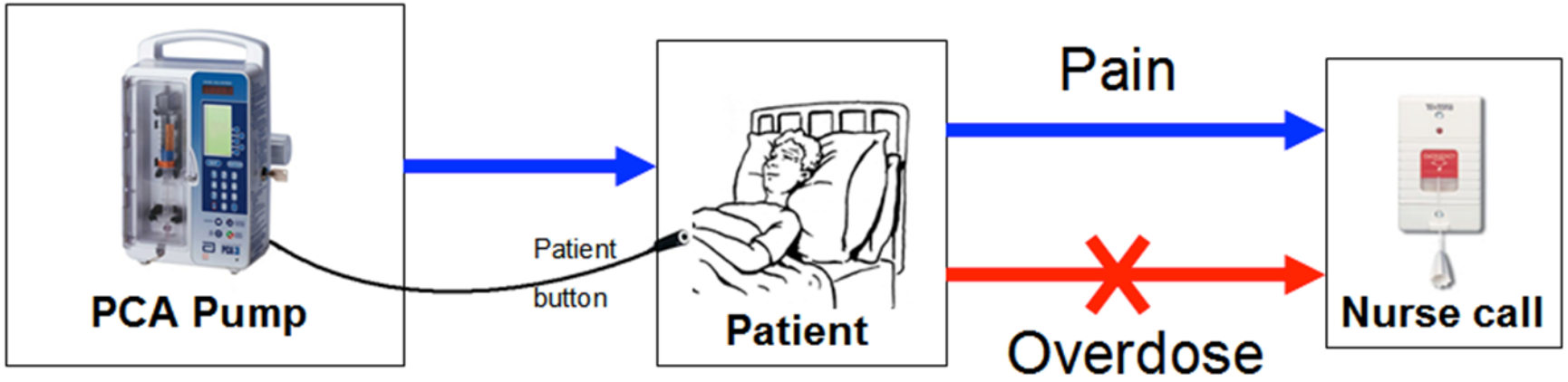
Physiological Closed Loop Control

Plug-n-Play Vendor Agnostic Interoperability

Key Capabilities of an Integrated Clinical Environment

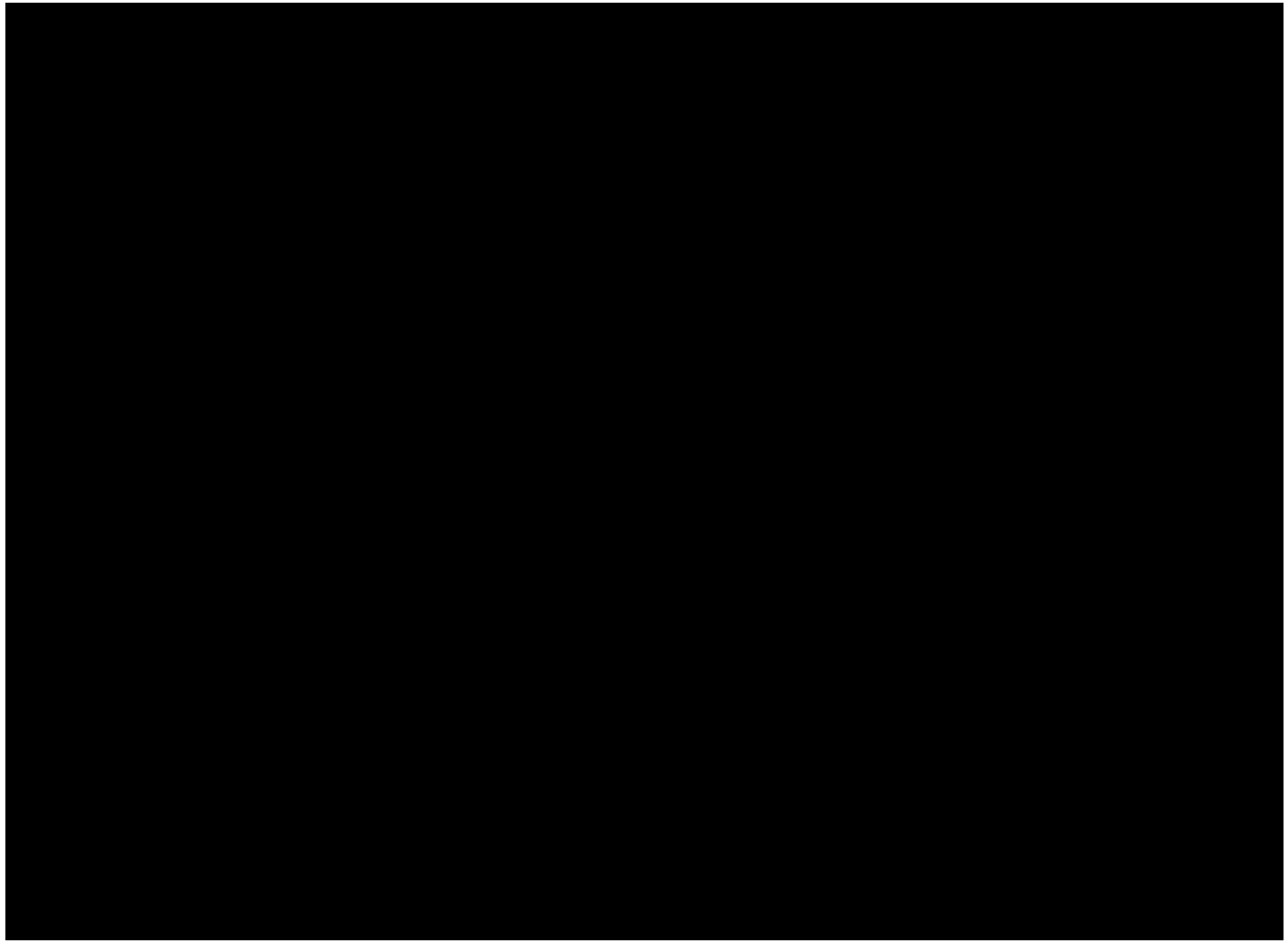


Scenario 1: Patient Controlled Analgesia (PCA)



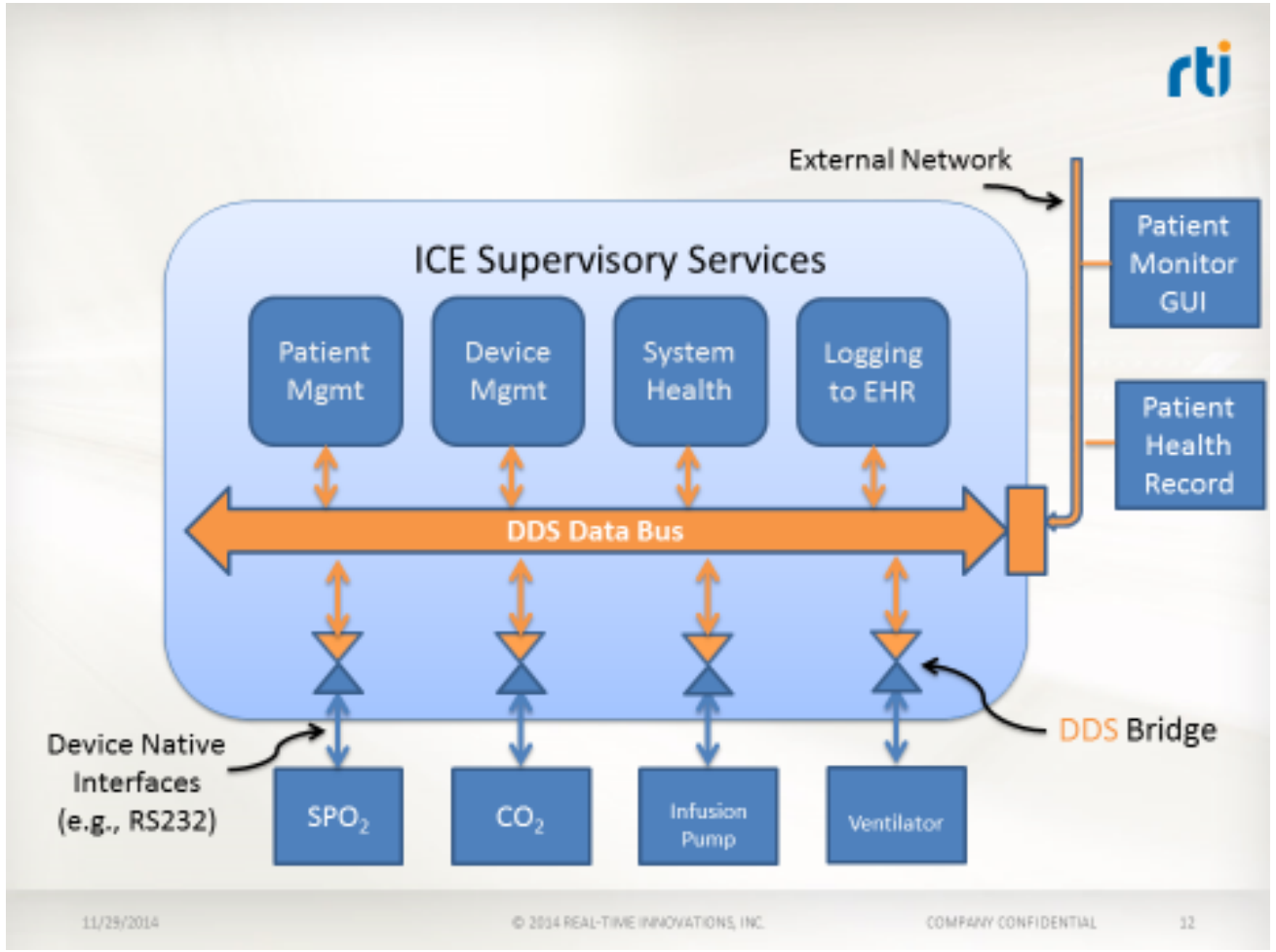
- PCA is simple, effective, & in widespread post-operative use
- Alarm fatigue prevents automated monitoring; 80% alarms turned off
- PCA kills ~3 patients a day in the US alone

Today's Problem



PCA - People and Systems

- Actors
  - Patients
  - Doctors and Nurses
  - Devices: Infusion pump, pulse oximeter, respiratory rate monitor(CO_2)
  - Distributed Software Applications
    - Supervisory services and applications
    - Device Adaptors



PCA – Improved Workflow

A hospitalized patient receives IV pain medication by PCA pump

The patient is connected to, and monitored with, a pulse oximeter and a respiratory CO_2 monitor (capnograph)

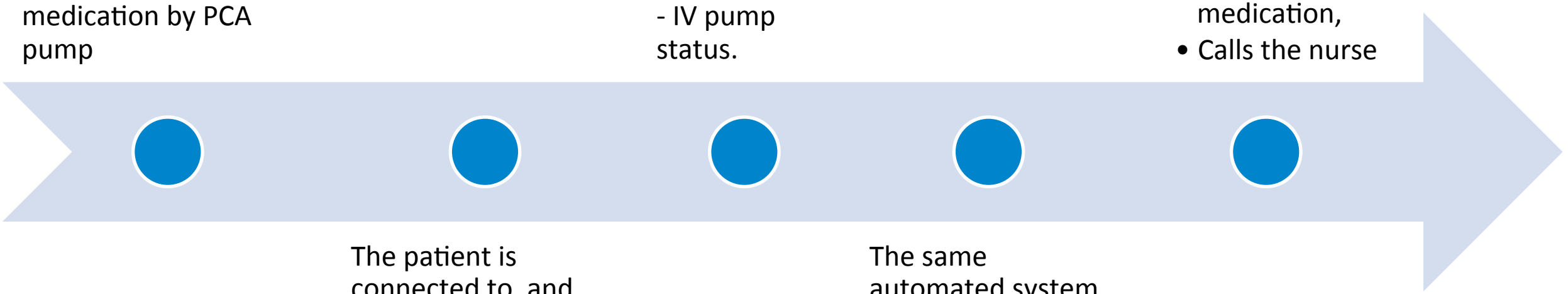
Automated system monitors:

- patient data,
- CO_2 data,
- IV pump status.

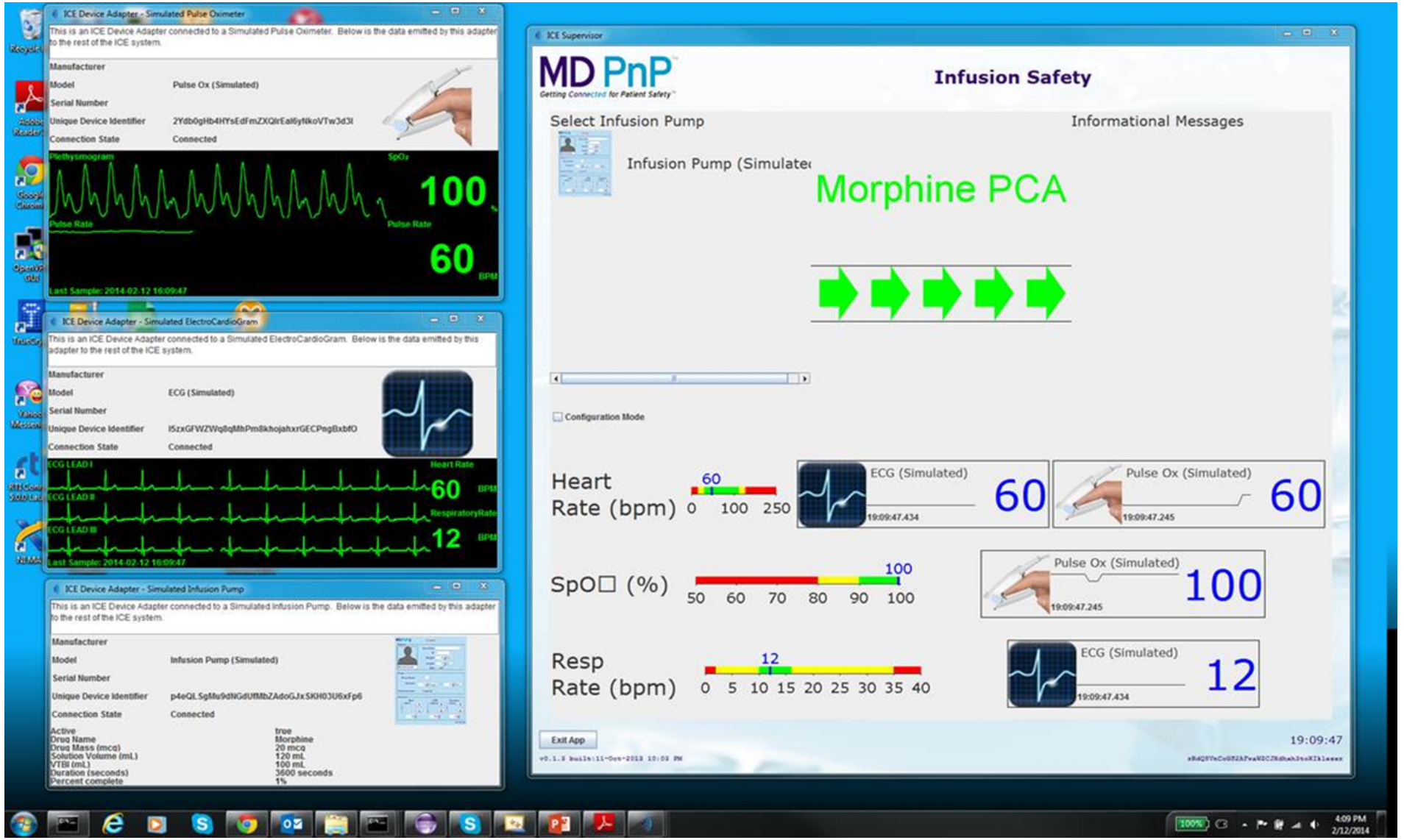
The same automated system correlates data

If unsafe level, Supervisory "app":

- Detects the problem,
- Stops the medication,
- Calls the nurse



Example PCA safety application



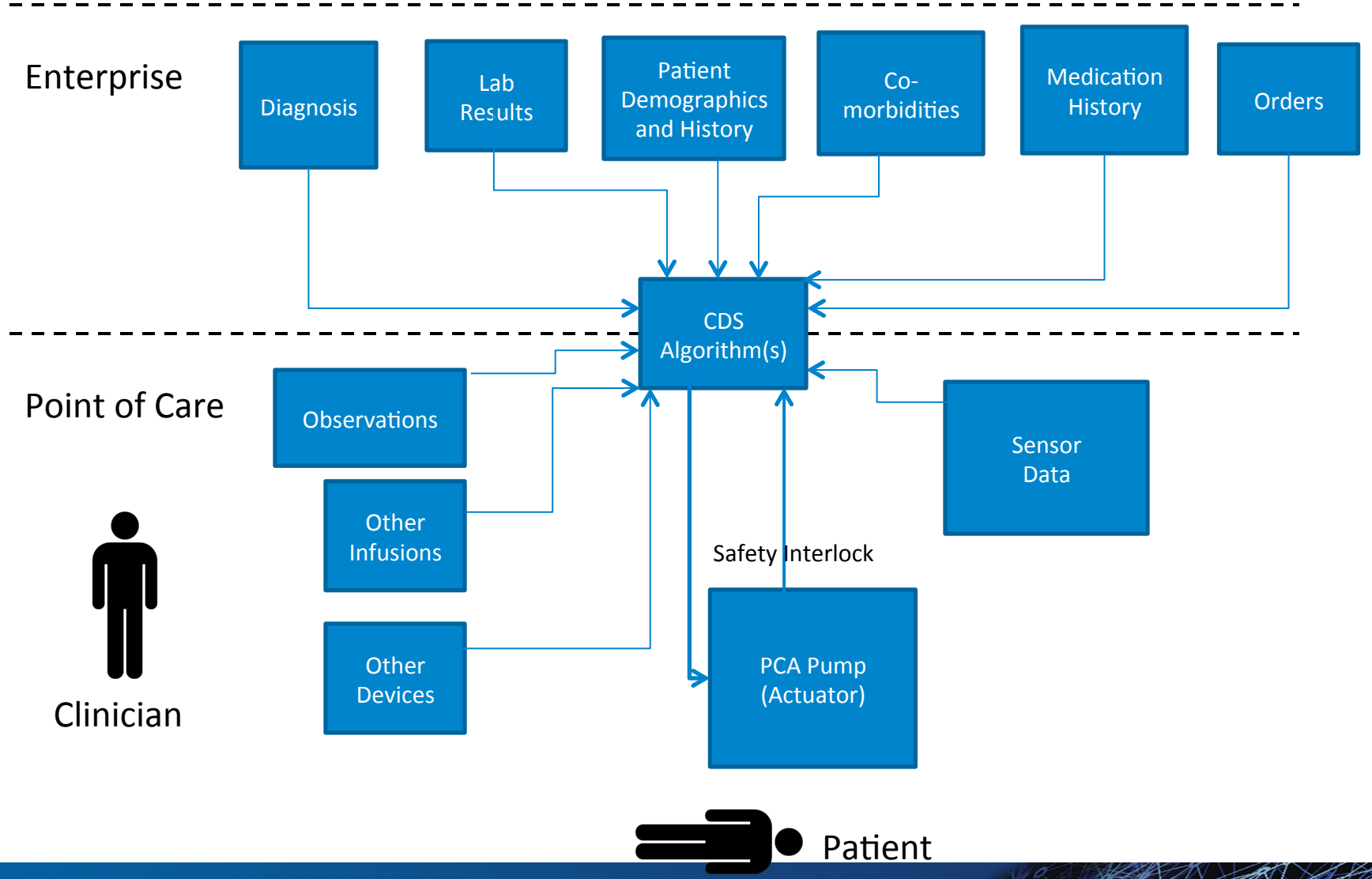
PCA - Successful End Condition

Respiratory problem detected

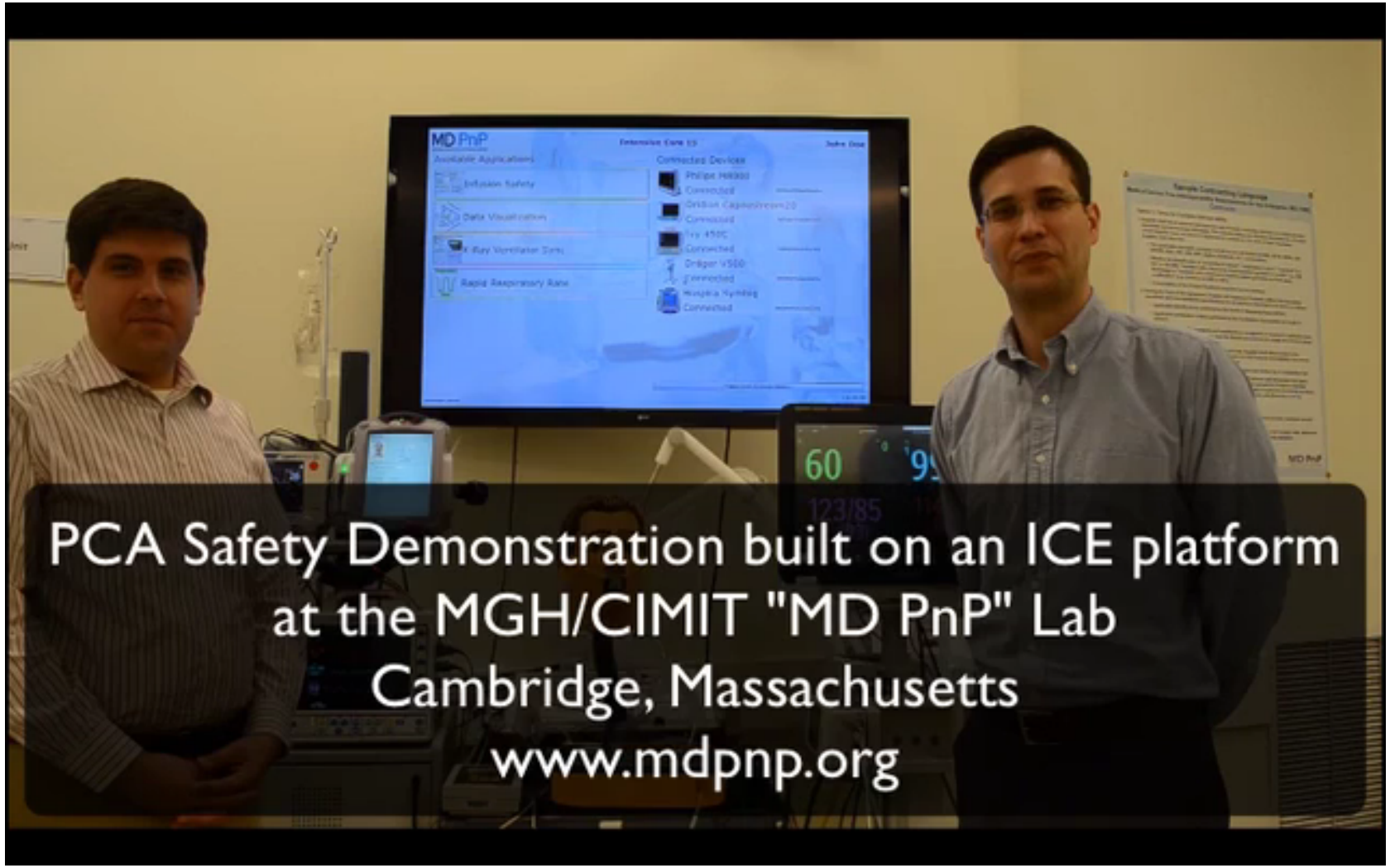
False alarms reduced

Respiratory or cardiac arrest avoided

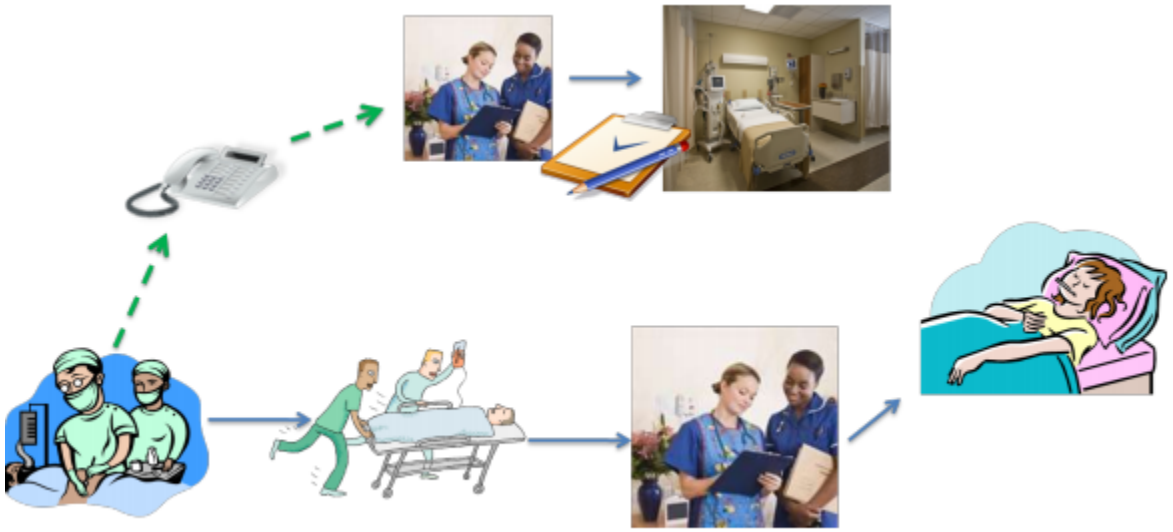
Smart PCA System



Improved PCA Demo



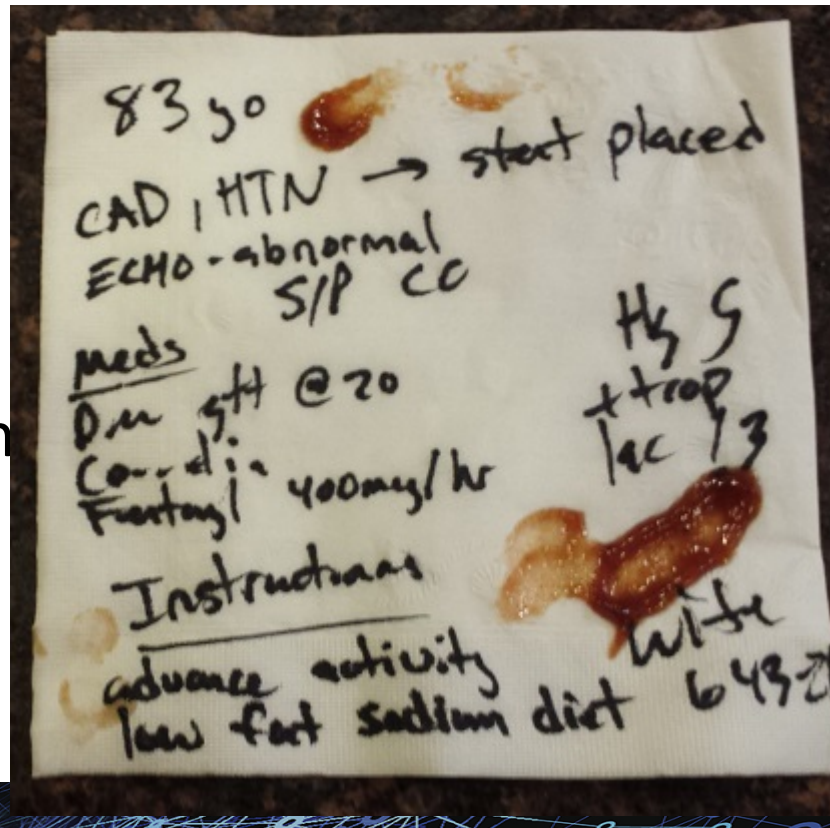
Scenario 2: Operating Room to Intensive Care Unit



OR to ICU - Today's Problem

- Transferring patients from the OR to the ICU is a complex manual process => major source of adverse events
  - An average of 5.42 supply and device assembly errors per handoff
  - 2.09 information omissions per handoff
  - 39% of handoffs involved errors of both types

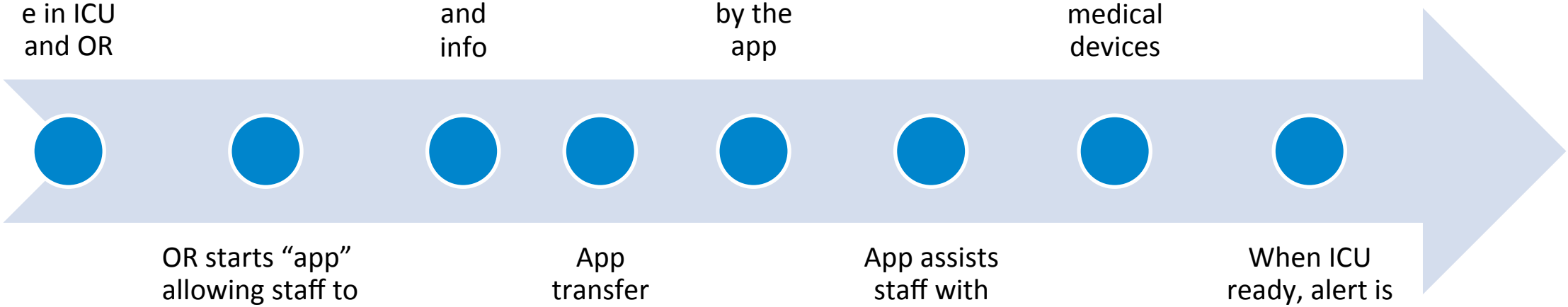
- Checklists have been shown to reduce errors:
  - They are updated manually, requiring caregivers to integrate large amounts of distributed information
  - Significant additional time may be required
  - Information loss is frequent
  - Medication errors are frequent



OR to ICU - People and Systems

- Actors
  - Patients
  - Anesthesiologist
  - Circulating Nurse
  - Surgical Team
  - Infusion pumps, ventilator,
  - bedside patient monitor
  - Distributed Software Applications
    - Decision support system

OR to ICU – Improved Workflow



"app" availabl e in ICU and OR

OR starts "app" allowing staff to input:

- Patient's status
- Progress of operation
- Drugs or more units of blood

Staff inputs orders and info

App transfer s data to the ICU

Checklist is created by the app

App assists staff with keeping data complete and current.

App determines

which OR device settings must be carried over => programs medical devices

When ICU ready, alert is transmitted to OR notifying staff for transport

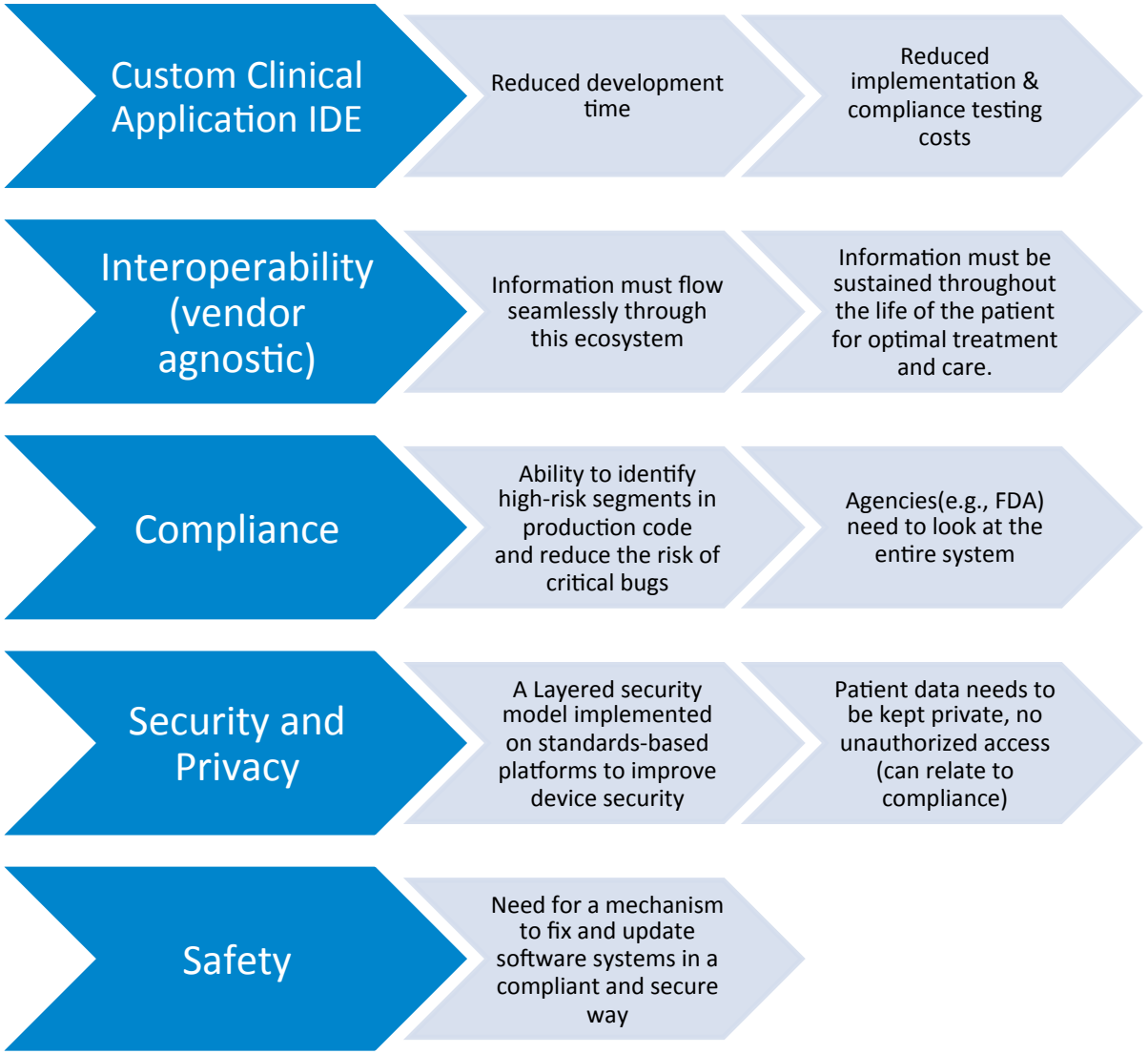
OR to ICU - Success End Condition

Right set of properly configured medical devices and correct data for the right patient is available at time of arrival to ICU

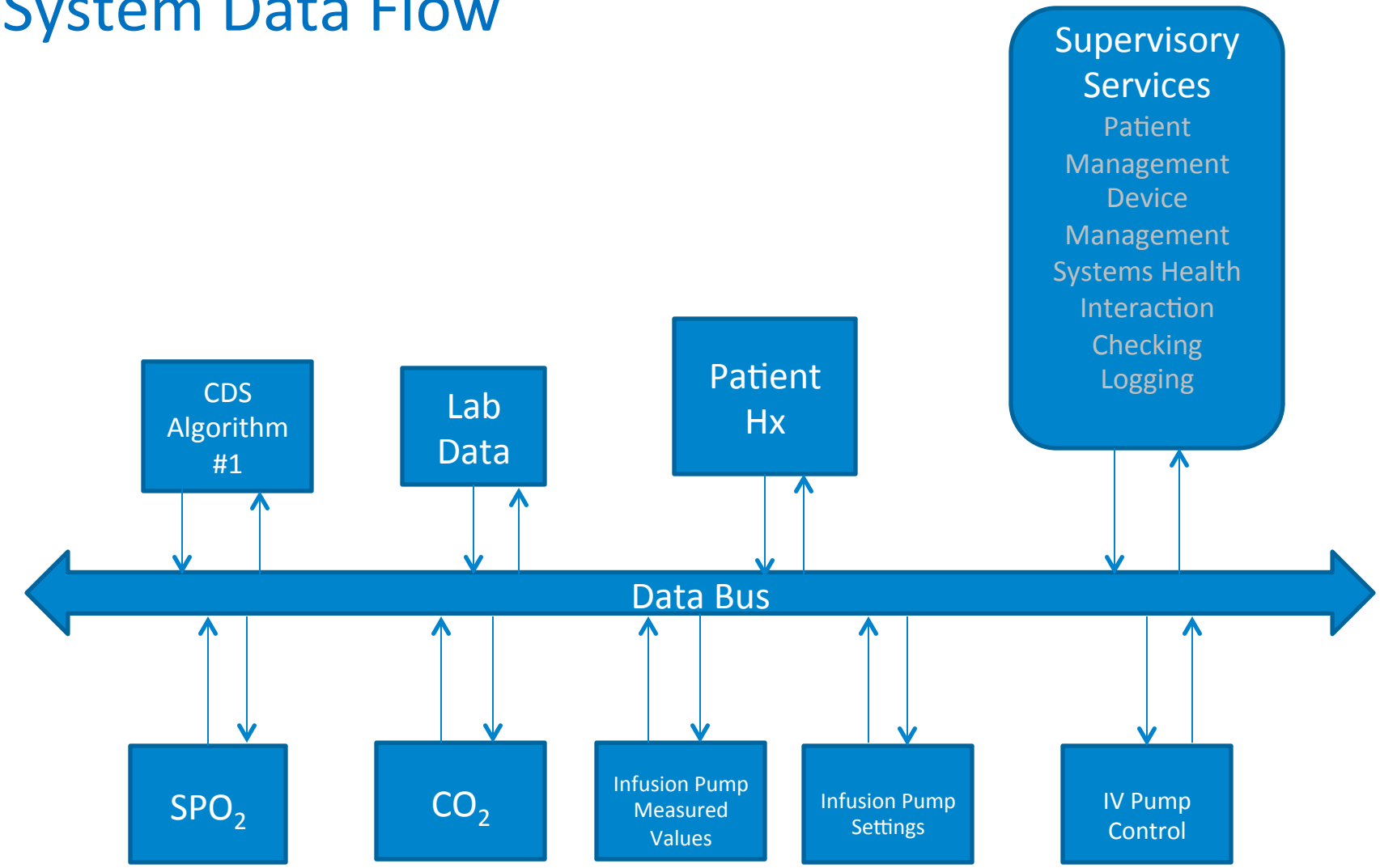
Data-Flow Requirements

- Controlled real-time Quality of Service (QoS)
  - Ensure data timing, reliability
  - Guaranteed knowledge of remote system status
  - Reduce application burden
- Facile interoperability between vendor devices
  - Support industry-standard data models
- Available
  - Manage redundant sources/sinks/ networks
  - Enforce clear failover behavior
  - Lower system brittleness
- Extensible
  - Integrate with unit (nurse's) station
  - Support whole-hospital system
  - Interact with cloud analytics
- Evolvable
  - Connect device and system legacy versions
  - Support data-type evolution
- Secure
  - Discover and connect patients and care teams; enforce access control & privacy
- Low compliance testing costs
  - Reusable communications module
  - IEEE 60601 class III device support

System Requirements



Required System Data Flow



Business and Technical Challenges

Acceptance

- From medical device manufacturers to interoperate
- Training of hospital staff

Legal implications

- In case of problem, whose fault? Which device of which vendor?

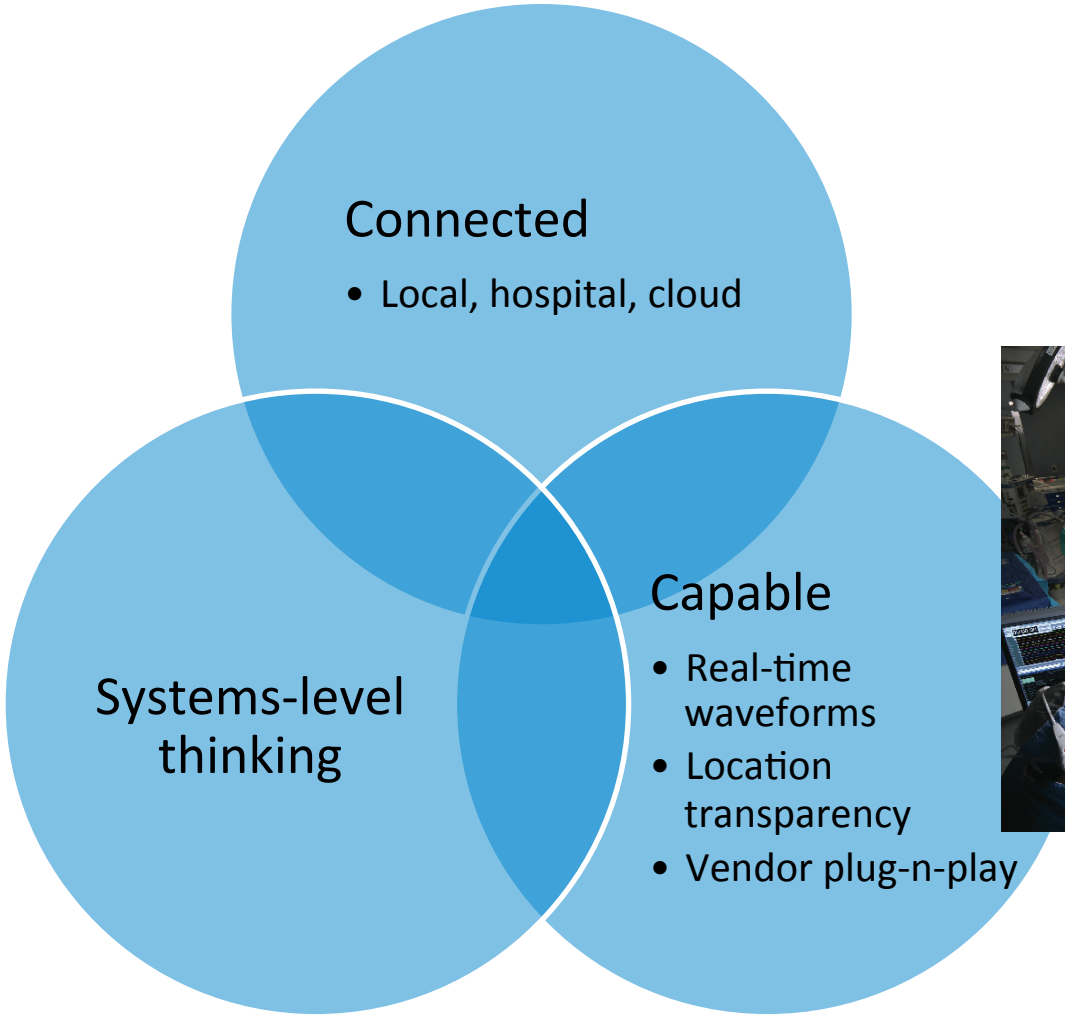
Regulatory

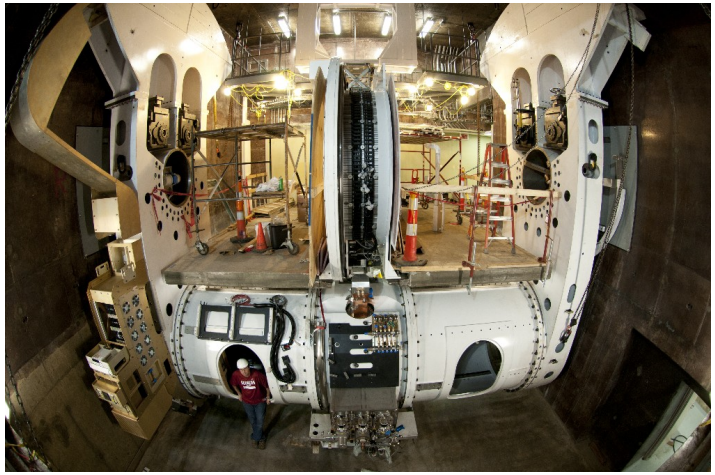
- Standards need to be FDA approved

Interoperability

- Mix of different data types and sources

Future of Medicine





THANK YOU!

Improved OR to ICU handoff

Deep Dive into DDS and Code

Overview

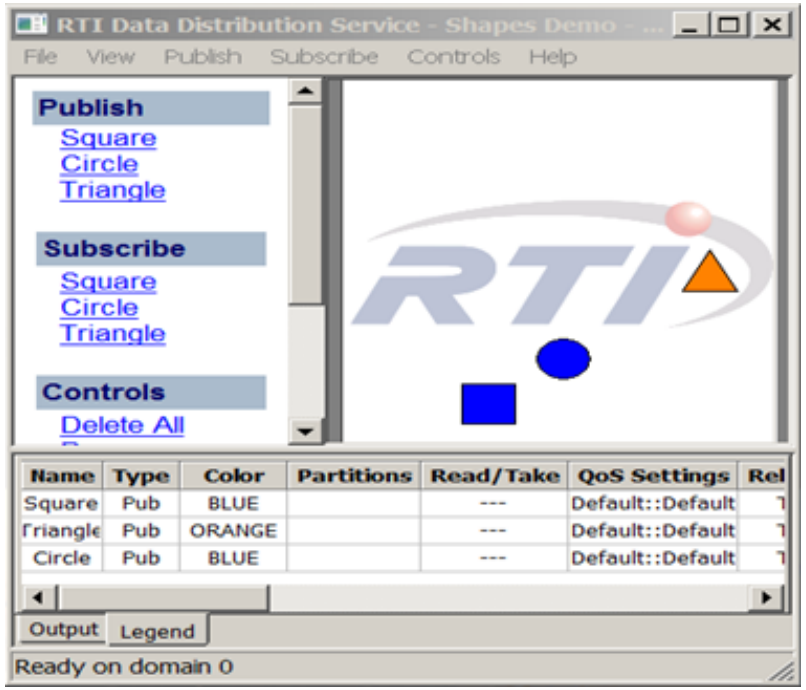
- Technical details of DDS
- Why DDS in medical systems
- Data model for medical systems
- Design patterns and data flow
- Java application code
- Where to learn more

DDS Technical Details

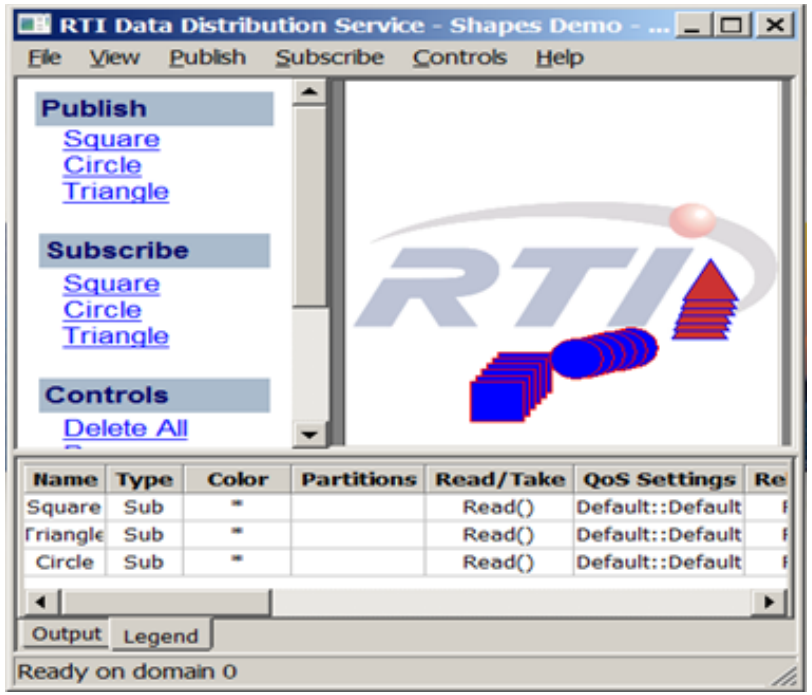
Fundamentals

- Publish-Subscribe Model
- Data-Centric Interfaces
- API Terminology
- Quality of Service

Publish-Subscribe Model

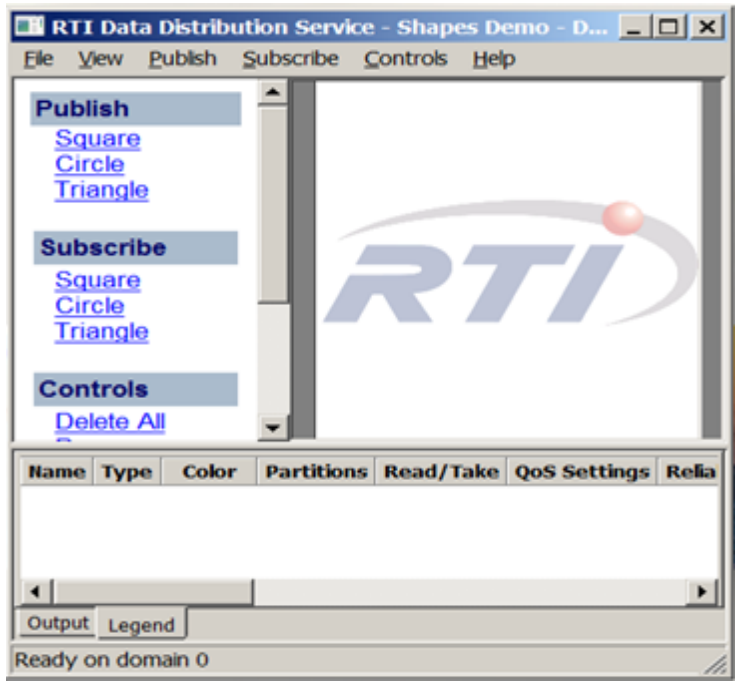


Application publishes shape data

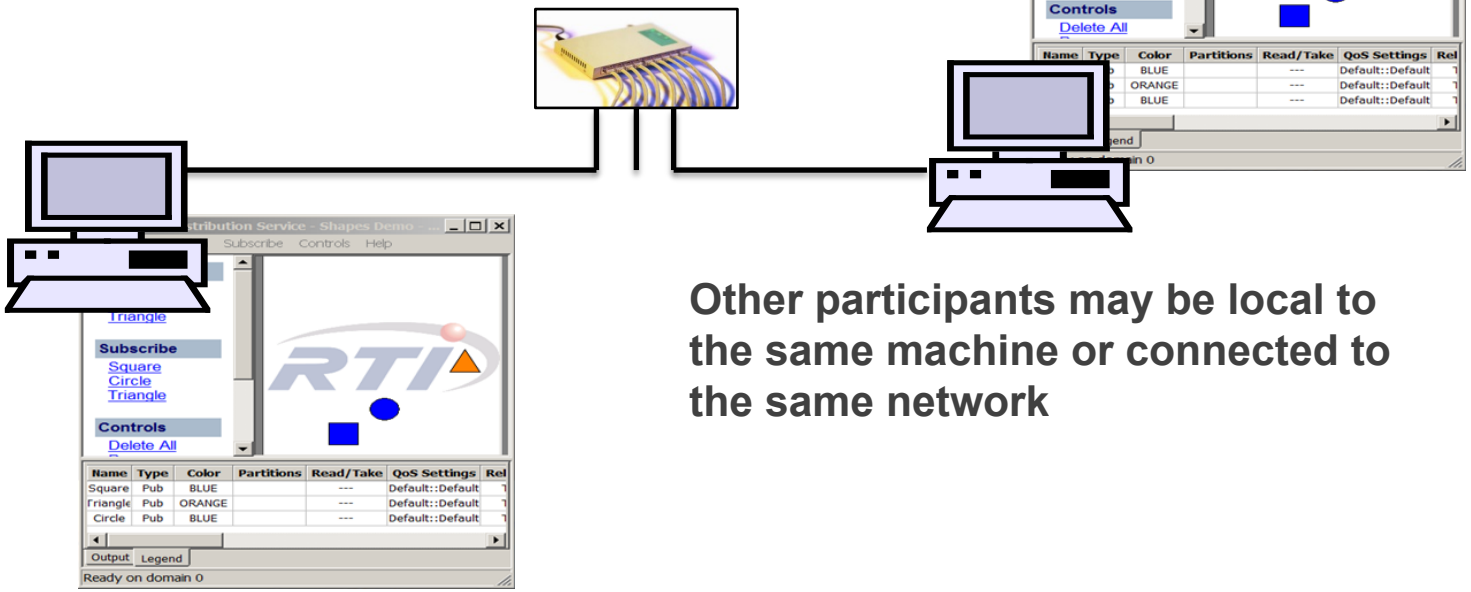


Application subscribes to shape data

Peer-to-Peer Discovery



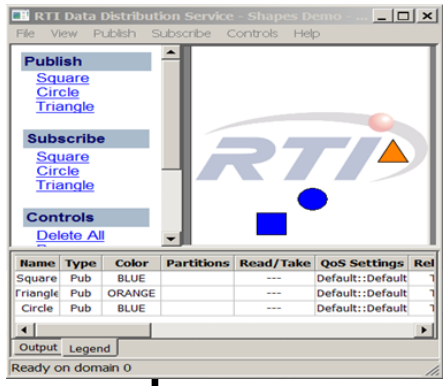
New application sends announcements looking for other participants

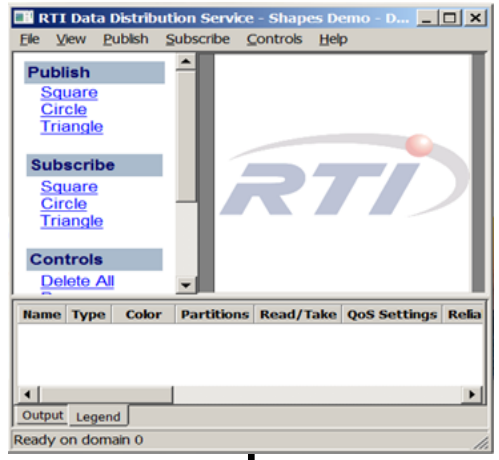


Other participants may be local to the same machine or connected to the same network

Endpoint Matching

During discovery, participants exchange publication and subscription intents, but not all discovered endpoints communicate.



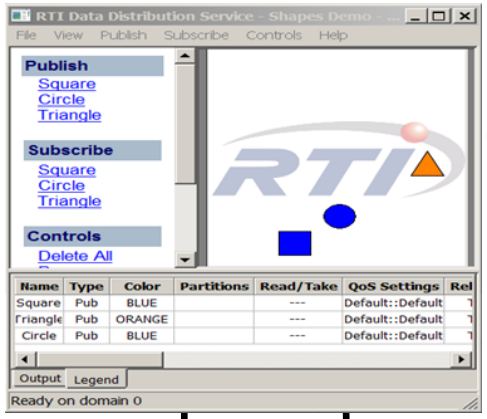


Equal topic names

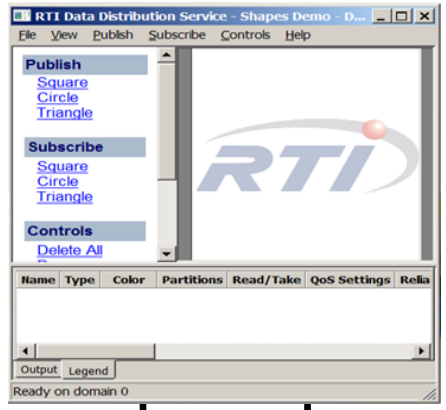
Topic: “Squares”

Topic: “Circles”

Two More Rules for Matching



Compatible Data Types



Type:

"ShapeType"

Type:

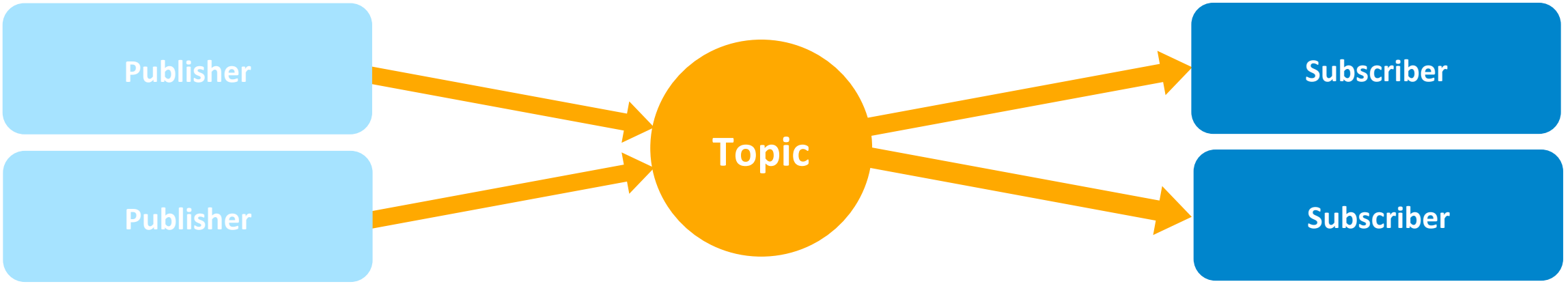
"PositionType"

Compatible
Quality of Service

"Best-Effort" QoS

"Reliable" QoS

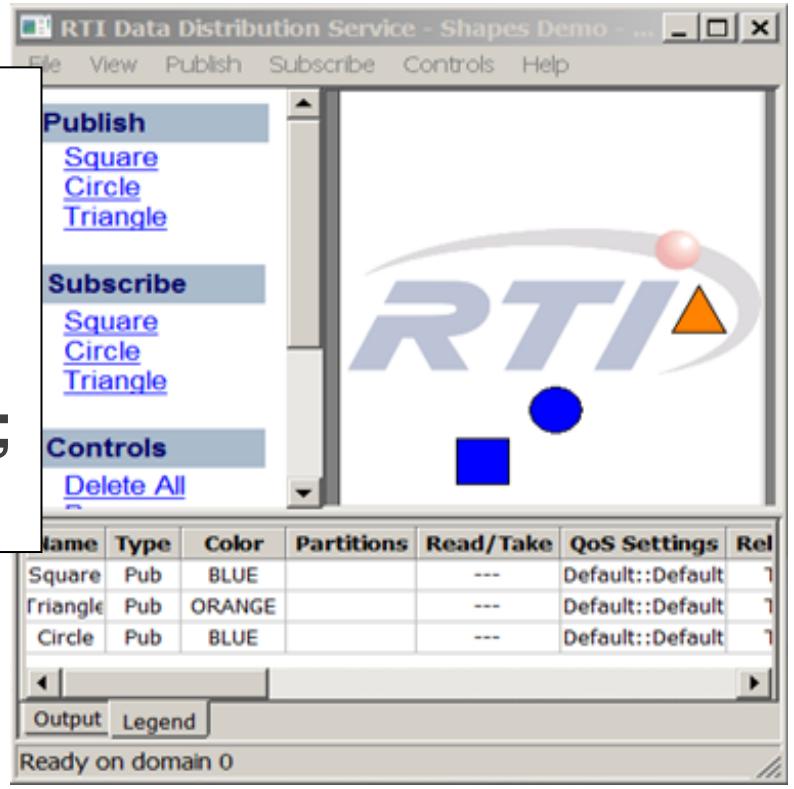
Decoupled Applications



- Applications declare intents to publish or subscribe to a topic
- No a priori knowledge about each other
- Standard, automatic **discovery** process
- Matched subscribers get data when it is published

Data-Centric Interfaces

```
struct ShapeType {
  string<128>        color;
  long                    x;
  long                    y;
  long                    shapesize;
};
```

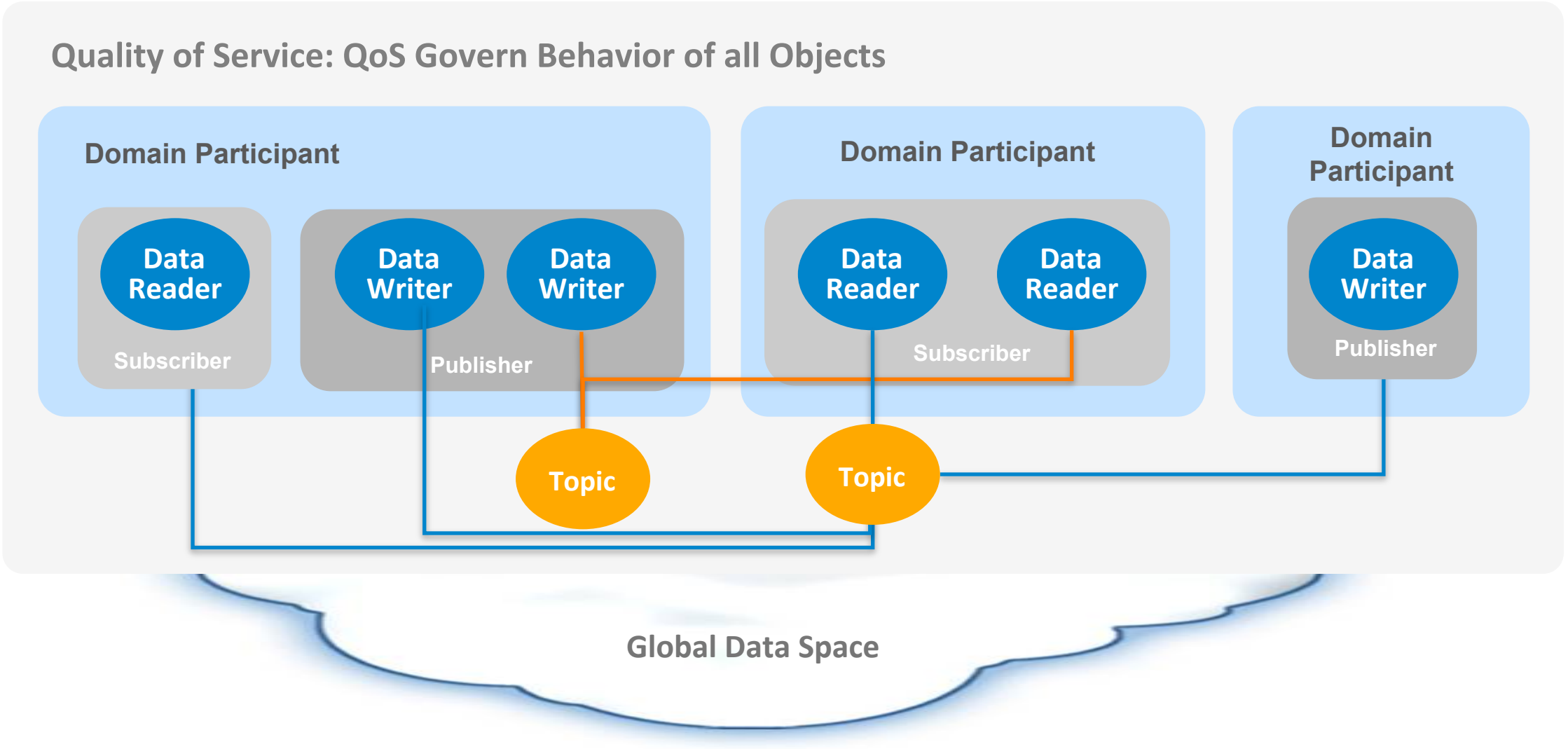


Applications can interact directly with data like a C structure. In addition to grouping related information, DDS types should group data with similar distribution characteristics.

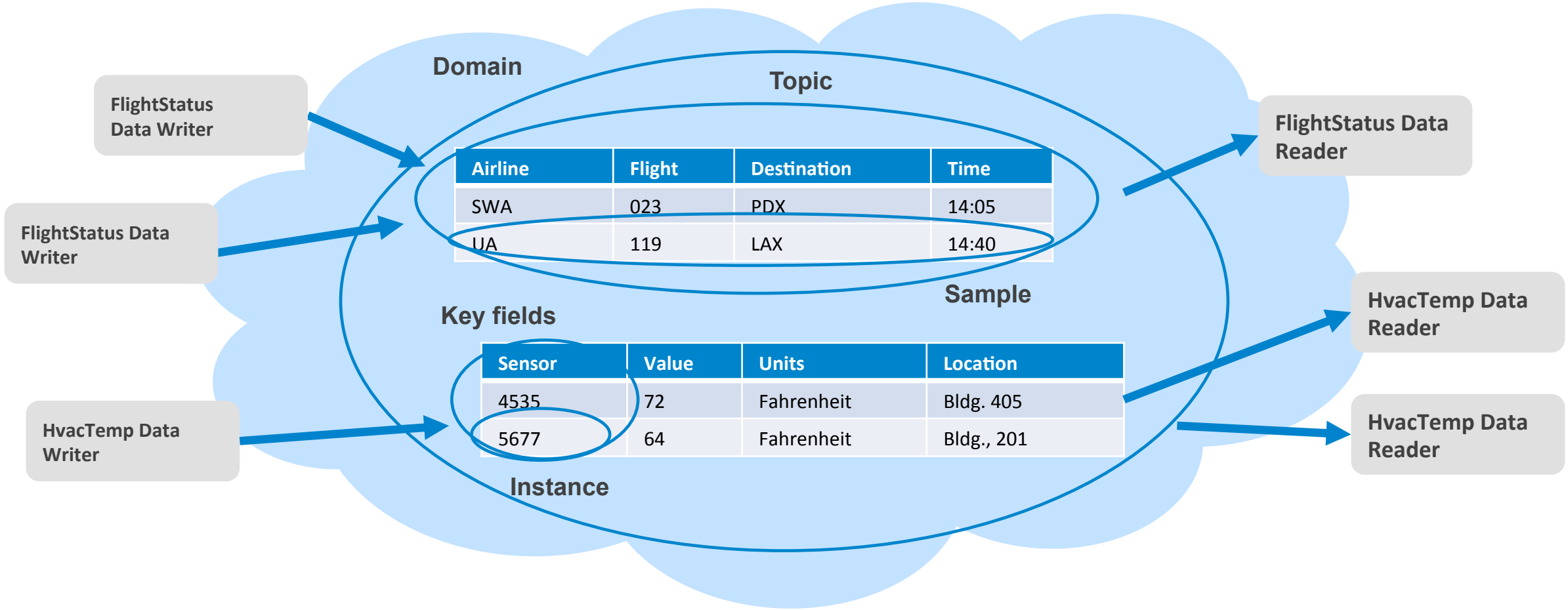
APIs

- The OMG Data Distribution Service standard provides bindings for
  - Java
  - C/C++
- This allows a single data model to communicate in a language-independent way

API Terminology



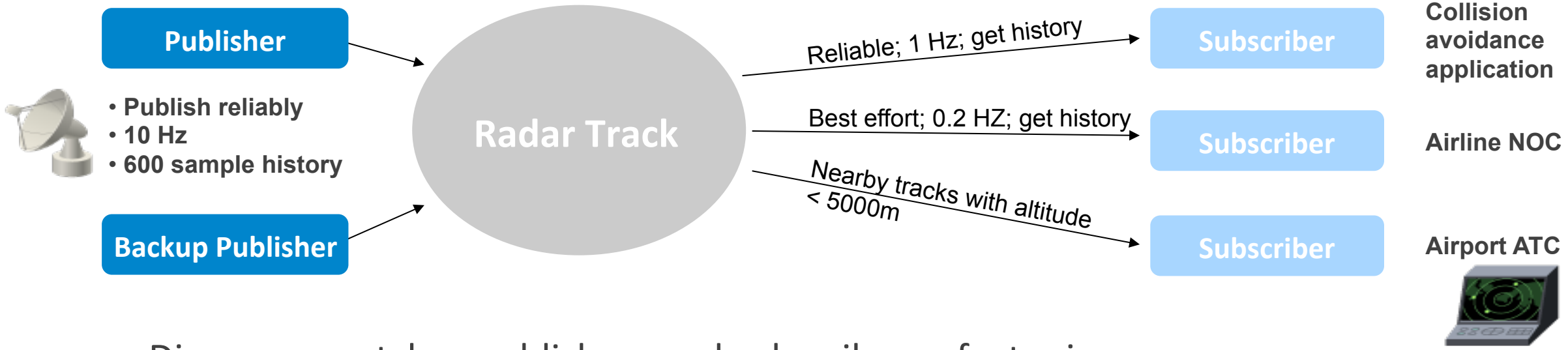
Data Space Entities



Quality of Service

- Quality of Service (QoS) settings provide a shared vocabulary to characterize data flow
  - Senders and receivers define the QoS they respectively offer and request
  - QoS exchange process ensures compatibility before allowing communication
  - Runtime behavior is tied to associated QoS
    - Should lost samples be repaired?
    - How many received samples should be queued?

Summary



- Discovery matches publishers and subscribers of a topic
- Topics tied to Data type, allowing smarter bus
- Quality of Service describing application data needs must be compatible to communicate
- Specifying and relying only on these data properties enables decoupled, resilient systems

Hands-On: Hello World

- Make sure you have the Connext DDS Evaluation installed
- Create a data type in IDL
- Run a code generator
- Modify QoS
- View and undersatnd the APIs

Why DDS in Medical Systems?

Data-Flow Requirements

- Controlled real-time Quality of Service (QoS)
  - Ensure data timing, reliability
  - Guaranteed knowledge of remote system status
  - Reduce application burden
- Facile interoperability between vendor devices
  - Support industry-standard data models
- Available
  - Manage redundant sources/sinks/ networks
  - Enforce clear failover behavior
  - Lower system brittleness
- Extensible
  - Integrate with unit (nurse's) station
  - Support whole-hospital system
  - Interact with cloud analytics
- Evolvable
  - Connect device and system legacy versions
  - Support data-type evolution
- Secure
  - Discover and connect patients and care teams; enforce access control & privacy
- Low compliance testing costs
  - Reusable communications module
  - IEEE 60601 class III device support

DDS in Medical Systems

Representing patient and sensor data

DDS in Medical Systems

- Medical system data flows
  - Medical devices
  - Patient data
  - Alarms
- Each data flow has its own data model and QoS

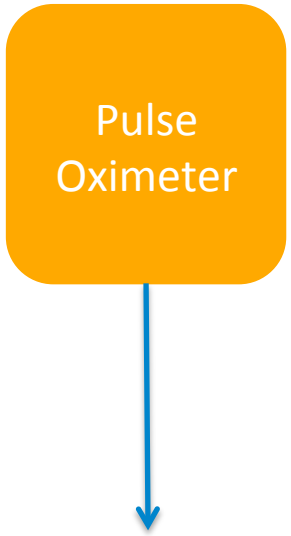
DDS in Medical Systems

[Demo Tutorial]

RTI Case + Code: Tutorial System

- Single application replays simulated data from
  - Pulse oximeter
  - ECG
- Device-patient mapping app
  - Associates device data with patient ID
- Bedside supervisor
  - Monitors all devices associated with each patient
  - Sends Alarm if multiple devices have alarm conditions
- Alarms HMI
  - Displays alarms

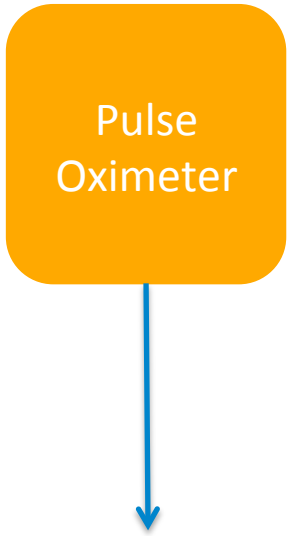
Device Data Modeling: Measured Data



Numeric: Periodic data data representing a measured sensor value

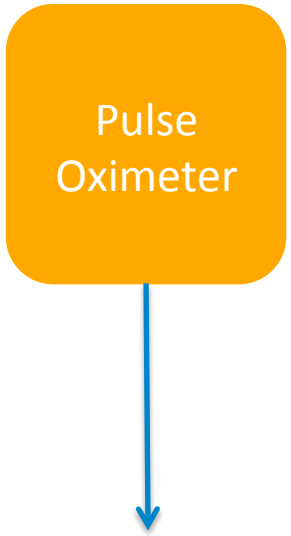
```
struct Numeric {
    UniqueDeviceIdentifier unique_device_identifier; //@key
    MetricIdentifier metric_id; //@key
    VendorMetricIdentifier vendor_metric_id; //@key
    InstanceIdentifier instance_id; //@key
    UnitIdentifier unit_id; //@key
    float value;
    Time_t device_time;
};
```

Numeric Quality of Service



- Numeric data represents a reading taken periodically from a patient
  - Oxygen level in a patient's blood
- Numeric data is modeled as periodic/sensor data
  - QoS for periodic data
    - Best-effort reliability
    - Liveliness to detect writer disconnection

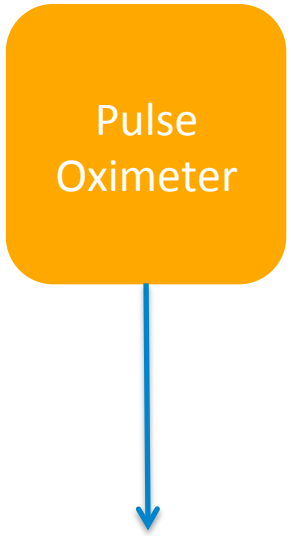
Device Data Modeling: Device Identity



DeviceIdentity: State data data representing device information

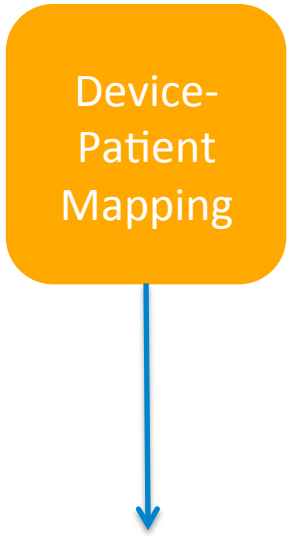
```
struct DeviceIdentity {
    UniqueDeviceIdentifier unique_device_identifier; //@key
    LongString manufacturer;
    LongString model;
    LongString serial_number;
    Image icon;
};
```

Device Identity Quality of Service



- Device Identity data is state data
  - Published only once
  - Interested subscribers can receive this at startup
- Device Identity QoS:
  - Reliable: sent only once, must arrive
  - Durable: late-joining subscribers will receive the state
  - History depth 1: Only need one identity update for each device

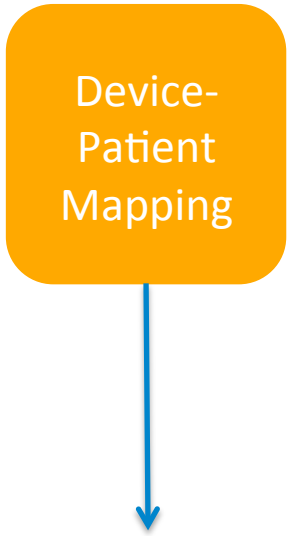
Device-Patient Mapping Data Model



DevicePatientMapping: State data data representing which device is monitoring which patient

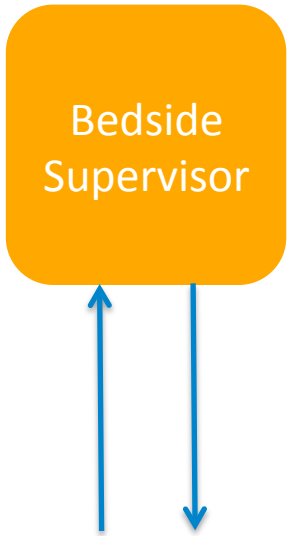
```
// Patient being monitored by a device
struct DevicePatientMapping
{
    // Unique ID of a device
    ice::UniqueDeviceIdentifier device_id; //@key

    PatientId patient_id;
};
```

Device-Patient Mapping Quality of Service



- Device Patient mapping data is state data
  - Published only when a device is associated/disassociated with a patient
  - Interested subscribers can receive this mapping between devices monitoring patients at startup
- Device Identity QoS:
  - Reliable: sent only when state changes, must arrive
  - Durable: late-joining subscribers will receive the current state
  - History depth 1: Only need one identity update for each device

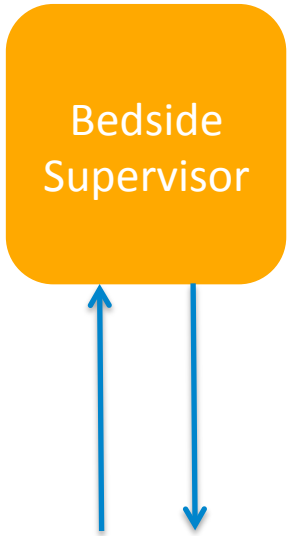
Bedside Supervisor: Data Modeling



- Receives Values, DevicePatientMapping, DeviceIdentity
- Sends Alarms

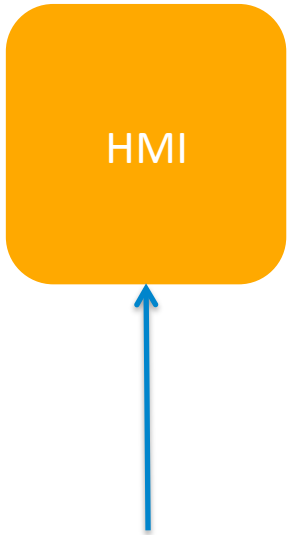
```
struct Alarm
{
    // The patient being monitored
    PatientId patient_id; //@key

    // The alarm kind (such as high pulse)
    AlarmKind alarmKind;

    // The values causing the alarm
    sequence<ice::Numeric, MAX_PATIENT_DEVICES>
        device_alarm_values;
};
```

Alarm Quality of Service



- Alarm data is state data
  - Published only when an alarm turns on or off
  - Interested subscribers can receive the current alarm state at startup
- Alarm QoS:
  - Reliable: sent only when state changes, must arrive
  - Durable: late-joining subscribers will receive the current alarm state
  - History depth 1: Only need one alarm state update for each patient (in this example)

HMI



- Receives Alarm state data
- Displays Alarms

Demo: Building the Medical App

- IDL-to-Java generation
- A look at DDS APIs inside the demo application

Where Can I Learn More?

Resources

- go.rti.com/JavaOne
- Information on:
  - The DDS standard
  - The OpenICE project for connected medical devices
  - RTI Case+Code tutorials