



UNIVERSITA' DEGLI STUDI DI BERGAMO
Facoltà di Ingegneria

Informatica Industriale

Prof. Davide Brugali

2.7 – OLE for Process Control (OPC)

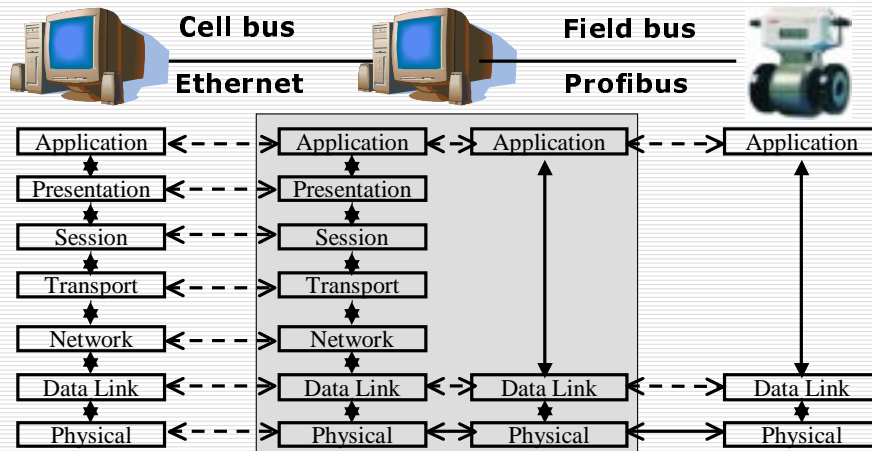
OLE for Process Control

□ **Motivation:**

A standard mechanism for communicating to numerous data sources, either devices on the factory floor, or a database in a control room.



Layers of standard functionalities



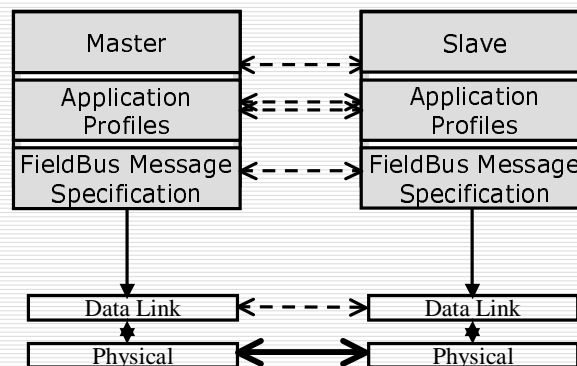
2.0 - Integrazione

Informatica Industriale - Prof. Davide Bruggali

3/47

Layers of standard functionalities

PROFIBUS



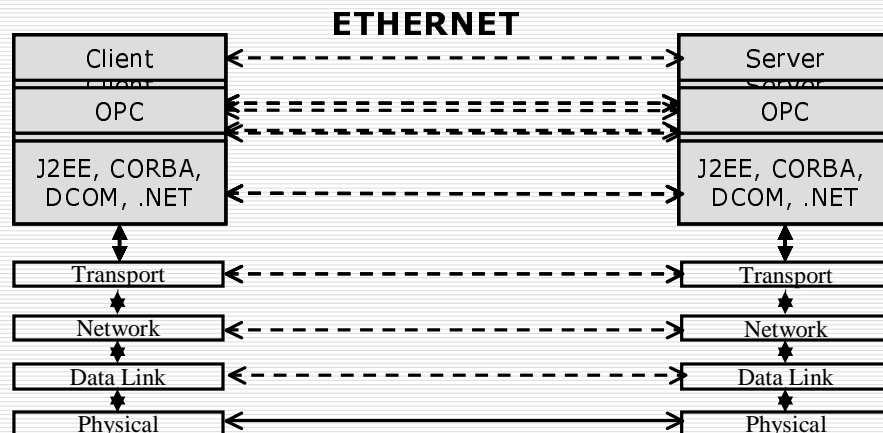
2.0 - Integrazione

Informatica Industriale - Prof. Davide Bruggali

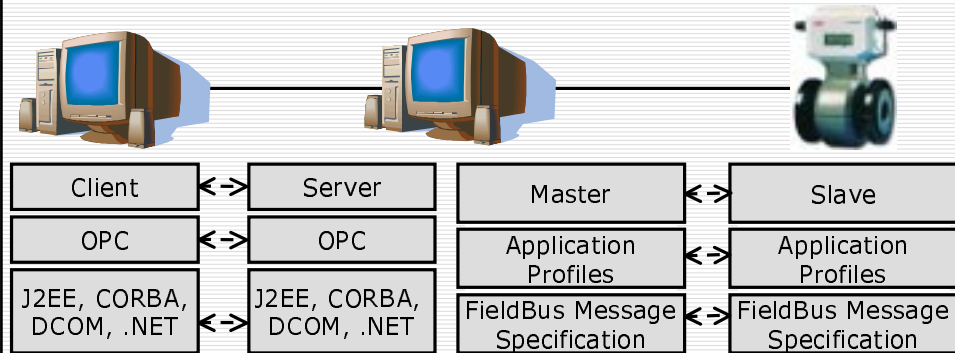
4/47

Designation	Profile contents	Current status of PNO guideline
PROFIdrive	The profile specifies the behavior of devices and the access procedure to data for variable-speed electrical drives on PROFIBUS.	V2 3.072 V3 3.172
PA devices	The profile specifies the characteristics of devices of process engineering in process automation on PROFIBUS.	V3.0 3.042
Robots/INC	The profile describes how handling and assembly robots are controlled over PROFIBUS.	V1.0 3.052
Panel devices	The profile describes the interfacing of simple human machine interface devices (HMI) to higher-level automation components.	V1.00 3.082
Encoders	The profile describes the interfacing of rotary, angle and linear encoders with single-turn or multi-turn resolution.	V1.1 3.062
Fluid power	The profile describes the control of hydraulic drives over PROFIBUS, in cooperation with VDMA.	V1.5 3.112
SEMI	The profile describes characteristics of devices for semiconductor manufacture on PROFIBUS (SEMI standard).	3.152
Low-voltage switchgear	The profile defines data exchange for low-voltage switchgear (switch-disconnectors, motor starters, etc.) on PROFIBUS DP.	3.122
Dosage/weighing	The profile describes the implementation of weighing and dosage systems on PROFIBUS DP.	3.162
Ident systems	The profile describes the communications between devices for identification purposes (bar codes, transponders).	3.142
Liquid pumps	The profile defines the implementation of liquid pumps on PROFIBUS DP, in cooperation with VDMA.	3.172
Remote I/O for PA devices	Due to their special place in bus operations, a different device model and data types are applied to the remote I/Os compared to the PROFIBUS PA devices.	3.132

Layers of standard functionalities



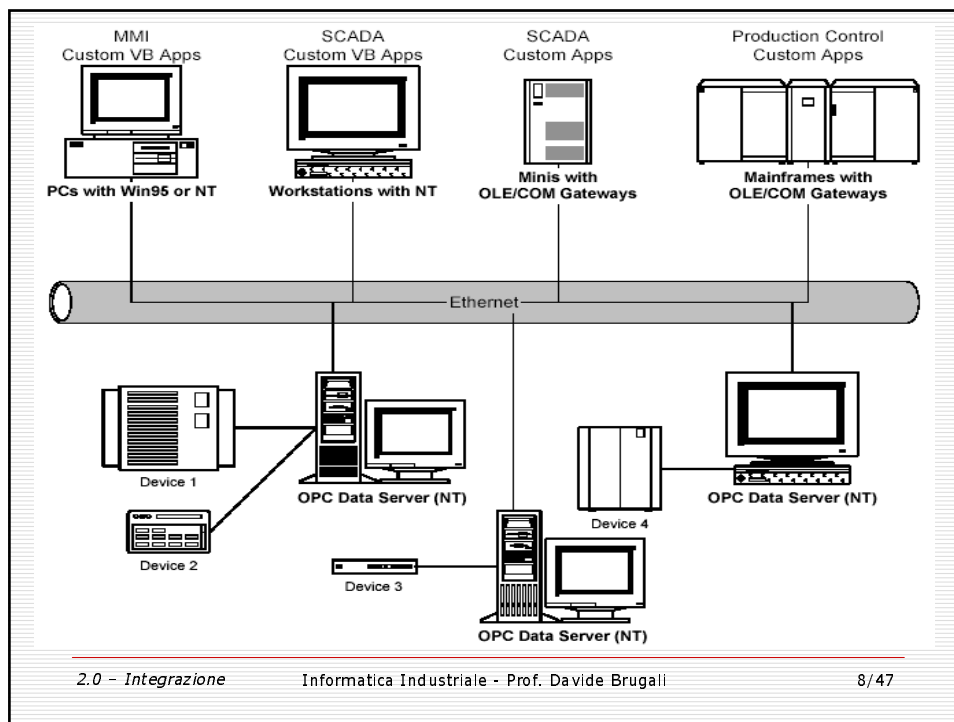
Layers of standard functionalities



2.0 - Integrazione

Informatica Industriale - Prof. Davide Brugali

7/47

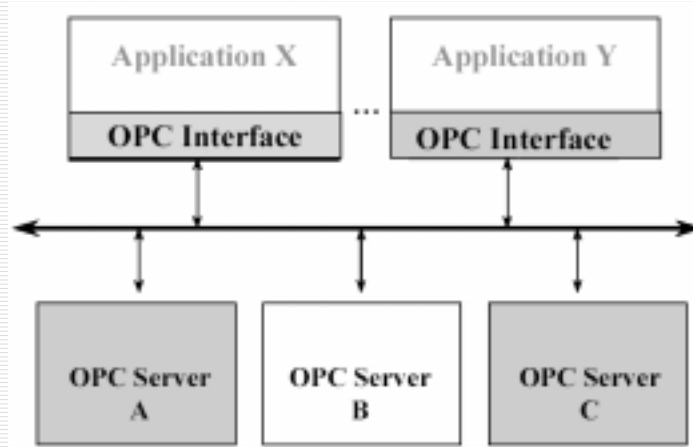


2.0 - Integrazione

Informatica Industriale - Prof. Davide Brugali

8/47

Layers of standard functionalities



Need of standards

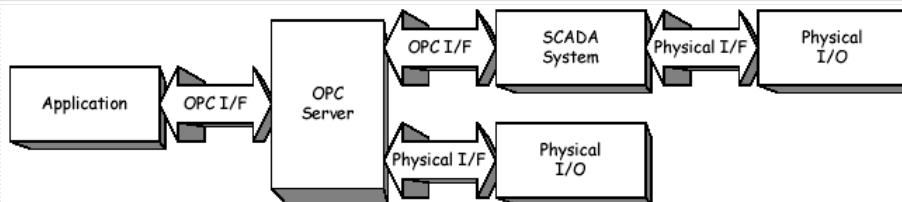
- ❑ **Much duplication of effort**
Everyone must write a driver for a particular vendor's hardware.
- ❑ **Inconsistencies between vendors drivers**
Hardware features not supported by all driver developers.
- ❑ **Support for hardware feature changes**
A change in the hardware's capabilities may break some drivers

OPC

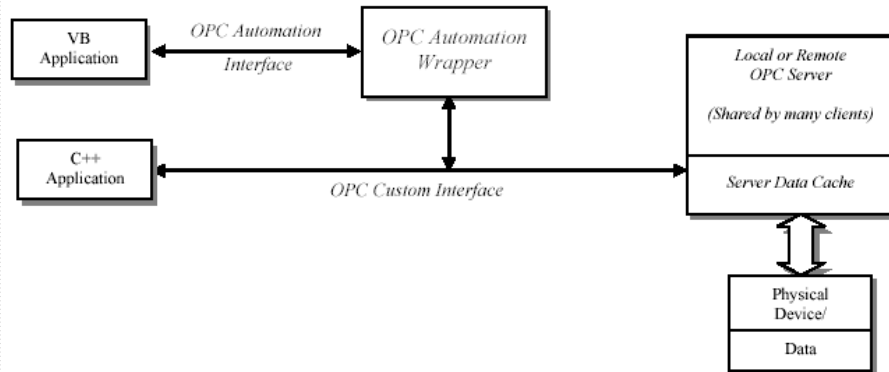
- ❑ OLE for Process Control (OPC) draws a line between hardware providers and software developers.
- ❑ A vendor can now develop a reusable, highly optimized server to communicate to the data source, and maintain the mechanism to access data from the data source/device efficiently.
- ❑ Providing the server with an OPC interface allows any client to access their devices.

Where OPC Fits

- ❑ Although OPC is primarily designed for accessing data from a networked server, OPC interfaces can be used in many places within an application. At the lowest level they can get raw data from the physical devices into a SCADA or DCS, or from the SCADA or DCS system into the application. The architecture and design makes it possible to construct an OPC Server which allows a client application to access data from many OPC Servers provided by many different OPC vendors running on different nodes via a single object.



General OPC Architecture and Components



OPC Server Browser

- ❑ The issue is, "How does a client program show the user what OPC Servers are available on a particular machine?".
- ❑ The OPC Foundation supplied Server Browser OPCENUM.EXE can reside on any machine, will access the local Component Categories Manager and provides a new interface IOPCServerList which can be marshaled and used by remote clients.
- ❑ This server has a published classid (see below) and can be installed once on any machine which hosts OPC servers.
- ❑ The client still needs to know the nodename of the target machine however he can now create this object remotely and use it's IOPCServerList interface to determine what types and brands of servers are available on that machine.

OPC First Releases

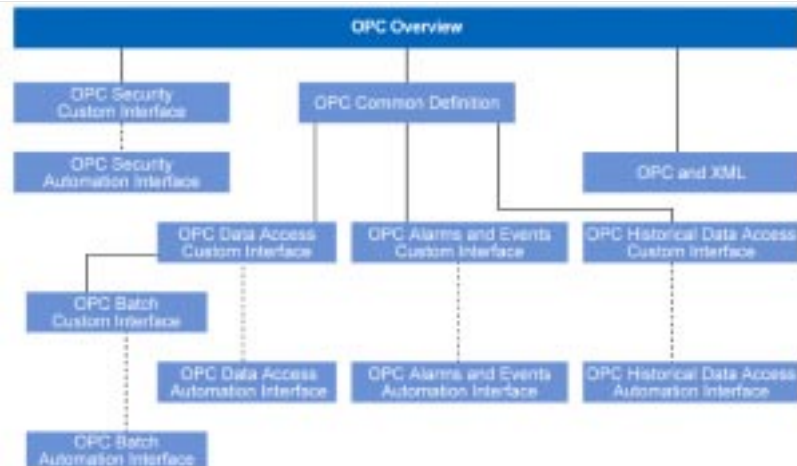
- ❑ A primary goal for OPC is to deliver specifications to the industry as quickly as possible. With this in mind, the scope of the first document releases is limited to areas common to all vendors. Therefore, the first releases focus on
- ❑ **Online DataAccess**, i.e., the efficient reading and writing of data between an application and a process control device flexibly and efficiently;
- ❑ **Alarm and Event Handling**, i.e., the mechanisms for OPC Clients to be notified of the occurrence of specified events and alarm conditions, and
- ❑ **Historical Data Access**, i.e., the reading, processing and editing of data of a historian engine.

OPC Subsequent Releases

- ❑ Security,
- ❑ Batch and historical alarm and
- ❑ Event data access

- ❑ XML interfaces

OPC Interface Overview



OPC Data Access Specification

- ❑ Defines an interface between client and server programs to access process data.
- ❑ Transparent access to different data sources (e.g. temperature sensors) and data sinks (e.g. controllers)
- ❑ Data sources and data sinks can be located on I/O cards directly plugged into the PC.
- ❑ They can also be located on devices such as controllers and input/output modules, connected by means of serial links or via field buses.

OPC Data Access Clients and Servers

- **DAClients** can be:
 - simple Excell sheet
 - complex Visual Basic GUI or
 - components of larger programs.
- **DAServers** can be:
 - simple programs providing access to a PLC
 - complex programs enabling access to a large number of variables and devices
 - components of larger programs (e.g. PC based Control)

OPC Data Access : Namespace

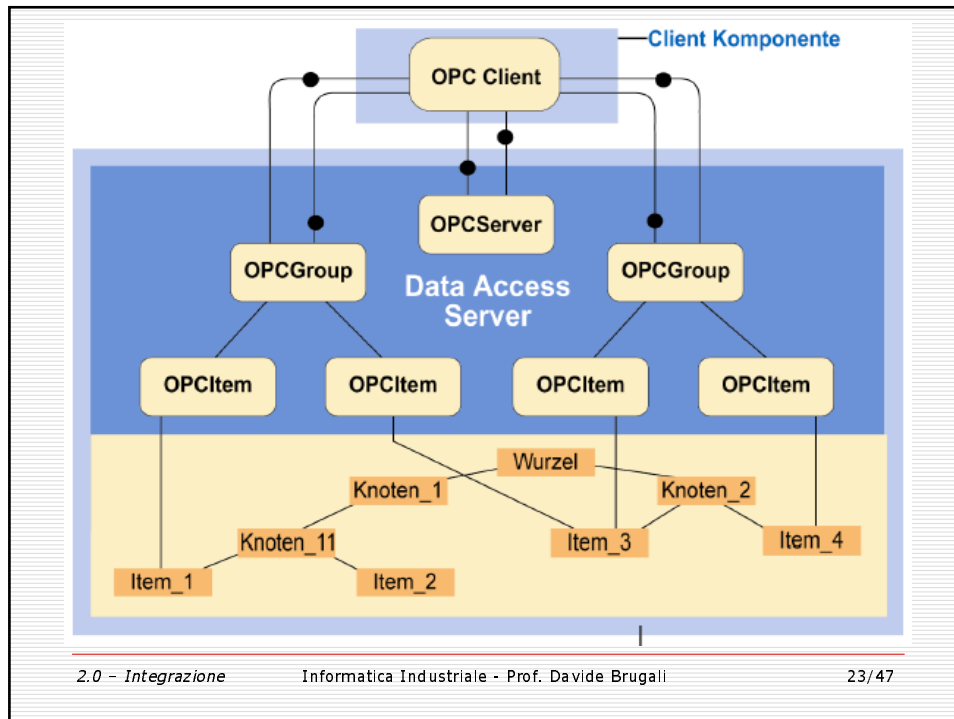
- The **Namespace** contains all the data sources and data sinks made available by a server.
- It can be a tree structure of any depth.
- **Nodes** can represent devices in an installation.
- **Leaves** are available at the nodes, representing data sinks and data sources.
- These can be setpoints and measured variables of a device.
- The leaves are also called **Items**.

OPC Data Access : Properties

- ❑ In an application, not only are data sources and data sinks important, but also the information on these variables and devices.
- ❑ Copying this information on items unnecessarily expands the namespace.
- ❑ Therefore, attributes, so called *Properties*, are allocated to nodes and leaves.
- ❑ The name of the manufacturer of a device, for example, can be mapped to a Property.

OPC Data Access : OPC Objects

- ❑ A DAClient can create several OPC objects in a DAServer to define its view of the process:
- ❑ OPCServer
- ❑ OPCGroup
- ❑ OPCItem



OPC Data Access : OPC Objects

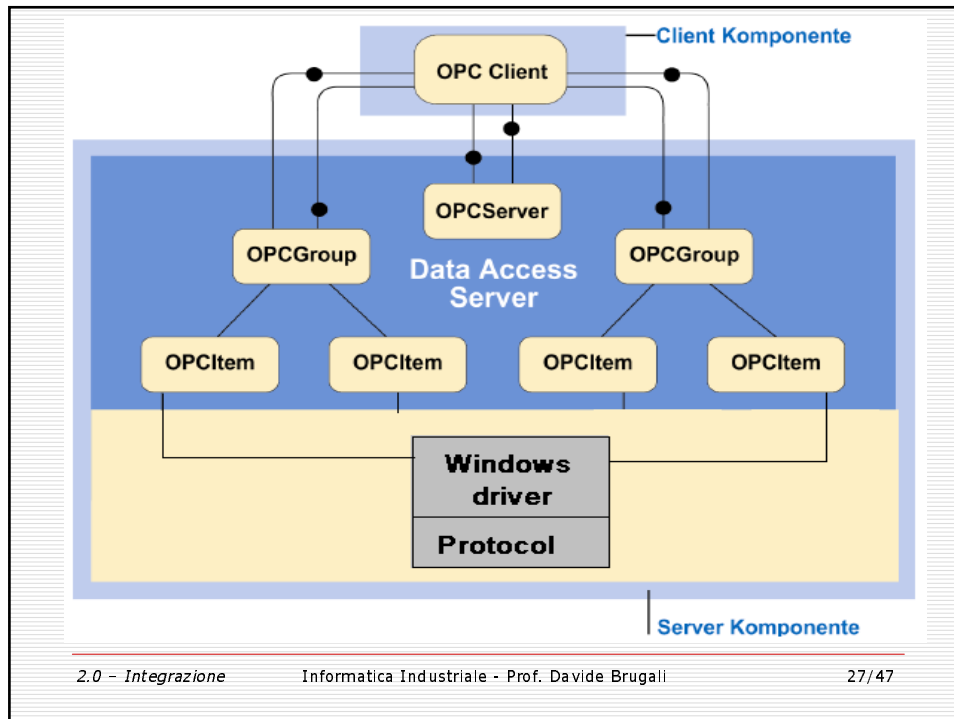
- ❑ **OPCItem Objects** represent leaves or properties of leaves and nodes of the namespace.
- ❑ **OPCGroup Objects** are used to structure OPCItem Objects. This can be done for logical aspects (e.g. all variables in one process section) or dynamic aspects (e.g. process variables with the same or similar time behavior) or as the user like.
- ❑ All the OPCGroup Objects are managed by the **OPCServer object**. Within one OPC Server component there can be several of these object hierarchies, one for each DAClient that has access to this component.

OPC Data Access : DCOM Objects

- ❑ Only OPCServer Objects and OPCGroup Objects are DCOM objects.
- ❑ For them custom interfaces, methods and parameters have been defined.
- ❑ OPCItem Objects have no custom interfaces because it is more efficient to access several OPCItem objects with a single call via methods and interfaces of the OPCGroup object.

OPC Data Access : data format

- ❑ The **actual data**. For exchanging data between client and server (and vice versa), all OPC specifications use DCOM data types called VARIANT data types. Application-specific data types are mapped to VARIANT data types (e.g. Bit to Boolean).
- ❑ The **time stamp**. There are devices that supply the time stamp with the value itself. In this case, it is used; otherwise it is generated by the server.
- ❑ The **status information**. The value can be Good, Bad (e.g. no link to the device), or Uncertain (deviation of the value from the measuring range) .



Browsing the namespace

- ❑ After starting the server, the Data Access Client accesses the top-level object OPCServer.
- ❑ The client will first query for the structure of the namespace.
 - Only nodes below the current node
 - Only leaves below the current node
 - Only leaves with specific data type
 - Only leaves with specific symbolic name

Creation of additional objects

- ❑ After the Data Access Client has browsed the namespace, it will create OPCGroup and OPCItem objects
- ❑ Parameters for creating OPCGroup objects:
 - Symbolic name
 - RequestedUpdateRate (scanning rate)
 - PercentDeadband (transmission rate)
 - ActiveState (automatic transmission)
- ❑ Parameters for creating OPCItem objects:
 - ItemId (namespace path)
 - ActiveState
 - RequestedDatatype
 - ClientHandle ↔ ServerHandle

Access to properties

- ❑ Static information associated to nodes and leaves:
 - Numerical identifier (PropertyID)
 - Data type
 - Textual description
- ❑ PropertyID
 - OPC-specific Properties (e.g. access rights)
 - Recommended Properties (server specific)
 - Vendor-specific Properties (e.g. calibration)

Reading and Writing of values

- After the creation of the object hierarchy, the Data Access Client can read and write values or get information about the change of values.
- The values can be taken from the server's internal cache of the process data (**Cache**) or directly from the data source (**Device**).
- Three types of data reading:
 - **Synchronous read** is only to be used if data access takes place quickly (e.g. values are available in the PC)
 - With **asynchronous reading**, the client calls the method of the server and immediately gets feedback. After a certain interval of time, which depends on the kind of data access, the client gets the desired value.
 - With a **refresh**, the client reads all active OPCItem objects of an active OPCGroup object.
 - With **subscription** mode, the server reads the values in cycles determined by the set update rate and passes them to the client if a change in the values or the status has taken place.
- Writing actions are always effected directly in the device. Writing to a cache and automatically transferring the values at a later date is not possible.

Status information (OPCServer)

- **StartTime** : the time when the server was started
- **CurrentTime** : the time in the server when the method call was made
- **LastUpdateTime** : the time when the last data were sent to the client by callback
- **ServerState** : running, failed, noconfig, suspended, test
- **GroupCount**: the number of OPCGroup objects created by the client.

Status information (OPCGroup)

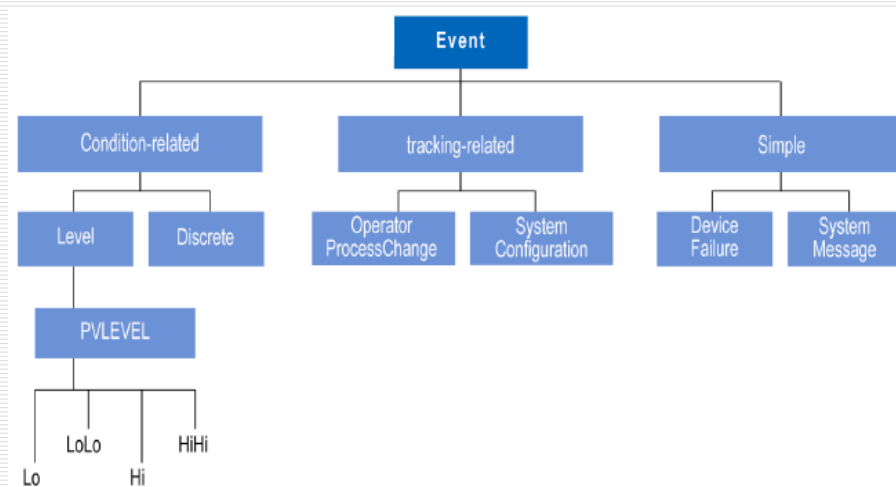
- ❑ **UpdateRate**: the number of times the values for the active OPCItem objects have to be read
- ❑ **ActiveState** : when *true* the values of the OPCItem objects are read
- ❑ **Name** : the symbolic name
- ❑ **PercentDeadband** : defines when values are automatically sent to the client

OPC Alarms and Events Specification

- ❑ Defines an interface between client and server programs for ***monitoring and acknowledging events and alarms***.
- ❑ An Event is a detectable occurrence (e.g. failure of a device, excess of a limit value)
- ❑ Several events can be allocated to specific variables. For example, for one temperature four limit values can be defined. This means that four events can occur.
- ❑ Data Access Servers make available a relatively continuous data flow.
- ❑ *Alarm and Event Servers do not send any values to the client, but send information that something has happened (e.g. a temperature has exceeded a limit value)*
- ❑ The criteria used by the server to decide that an event has occurred have to be set.

Event types

- ❑ **Simple Events** are used to notify the occurrence of events like the failure of a device to the client.
- ❑ **Tracking-related Events** provide information on the occurrence of an event which has to be tracked.
- ❑ **Condition-related events** allow to activate/deactivate the event notification.



Event description

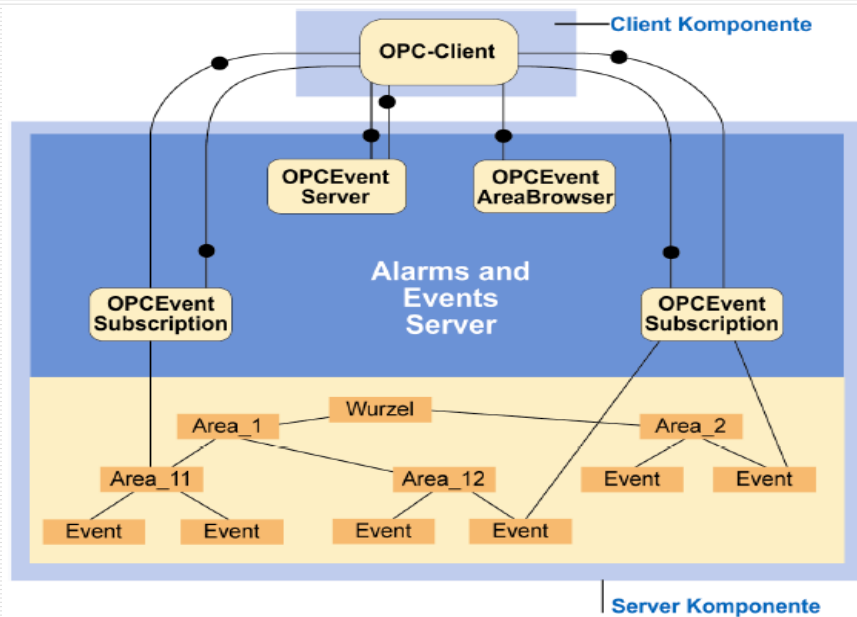
- Simple Events
 - Description of the source
 - The time at which the event happened
 - The event type
 - The event category
 - The event severity
 - A message
- Tracking-related Events (in addition)
 - ActorID indicates the origin of the event
- Condition-related Events
 - ConditionName
 - SubConditionName
 - ConditionQuality
 - ActiveTime

Structuring Events

- The **Eventarea** is used for *topographic* organization of events and can be compared with the namespace of the Data Access Specification.
 - The nodes of the eventarea describe areas in which events may occur (e.g. a workcell).
 - The leaves describe event sources.
 - Several events can be available at one event source
- The **Filter Space** is used for the *logical* structuring of the events. The filter space marks the structure of the events according to their attributes (type, category, severity)

Event and Alarm Objects

- ❑ The **OPCEventServer** object represents the top level object in the hierarchy.
- ❑ The **OPCEventAreaBrowser** object is used for browsing the event area.
- ❑ The **OPCEventSubscription** objects form the lowest level.
- ❑ There are no explicit objects available for events.



Filter criteria

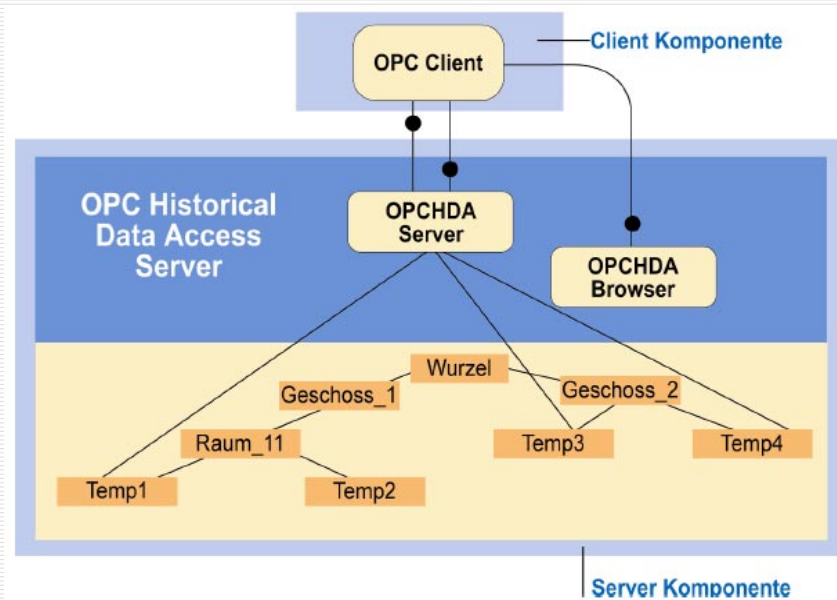
- ❑ Assignment of events to OPCEventSubscription objects is based on filters.
- ❑ Filters consist of information that characterizes the events (type, category, severity) and of information describing the position of the events in the eventarea (area, source).
- ❑ A server can support filtering based on the following values:
 - Event type
 - Event category
 - Severity
 - Area (nodes in the eventarea)
 - Source (leaves in the eventarea)

OPCEventSubscription objects

- ❑ OPCEventSubscription objects are used to monitor the event sources in the server and inform the client of the events which have occurred.
- ❑ Events are transferred to the client by means of callbacks. The BufferTime value determines the time interval at which the events are to be sent.
- ❑ The client assigns events to the OPCEventSubscription object by using filters.

OPC Historical Data Access Specification

- ❑ Historical Data Access Servers enable clients to access historical data. Two basic types of data:
 - Raw data, i.e. the historical data stored
 - Aggregated data, i.e. data extracted from the raw data and only created on request of the client.
- ❑ Access to the data can be
 - Write access
 - Read access
 - Change access
- ❑ The source can be a database



Differences to Data Access

- ❑ There are no objects comparable to the OPCGroup and OPCItem objects
- ❑ The client directly addresses data items by means of an handle without creating objects in the server.
- ❑ A **DAServer** allows permanent access to a limited number of process variables (100-1000)
- ❑ A **HDAServer** allows clients to read a large number of variables (10.000) or aggregates only once a day.

OPC Batch Specification

- ❑ The specification defines supplements to the Data Access Specification for the case of batch processing.
- ❑ A batch consists of different recipes describing the making of a product.
- ❑ When executing a batch, a data exchange with the devices is carried out. Recipe data are sent and report data are received.

OPC Batch Specification

- Four types of batch information:
 - Equipment capabilities
 - Current operating conditions
 - Historical contents
 - Recipe contents
- An OPC Batch Server defines a recipe consisting of the following information:
 - Header
 - Formula
 - Device requirement
 - Procedure
- In the namespace this information is mapped on properties of the nodes and leaves, which represent Procedural Control Modules