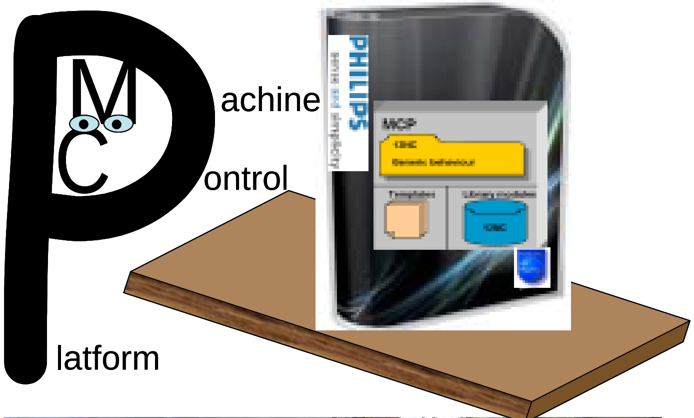
PHILIPS



MCP

Best of both worlds (PLC ⇔ PC)

Authors: Wim Bor, John van der Heijde, Frank Mertens, Ronald Korting
Philips Applied Technologies
February 11th, 2008

Answers to the following questions

Introduction

- ☑ What is MCP
- ☑ Why do we need it
- ☑ How did we create it

Technical

- ☑ What did we create
- ☑ What makes it unique

☑What is MCP

- What MS-Windows is for a PC, is MCP for industrial Products / Equipment

- Like MS-Windows, MCP
  - is a toolbox
  - by itself has limited functionality

- It is a software environment which enables you to
  - create an industrial application.
  - introduce quality into the equipment- and product- development

☑What is MCP

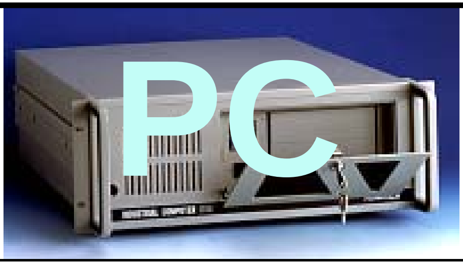
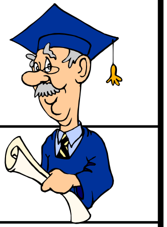
For clarification purposes a conformity check between MS-Windows and MCP

- Comprises a basic functionality like Internet browsing
- Supports content navigation (Windows based)
- MS-Word is not available by default

- Comprises a basic functionality like MachinePart browsing
- Supports content navigation (Object based)
- Practical application components are not available by default

☑What is MCP

A toolbox that combines Best Practices of Both Worlds

		
Tools	**Vendor specific** Siemens S7, CodeSys, Sigmatek	**Open Systems** C++, C#, .NET
Way of working	**Company dependent** 	**Standardized** Methods, Processes, Languages • **O**bject **O**rientation • **R**ational **U**nified **P**rocess • UML, **C++, C#,** 
Personnel	**Practically skilled**	**Theoretical educated**
Architecture	**Cyclic Scheduling**	**Event Driven Scheduling**

☑Why do we need it

Who are we

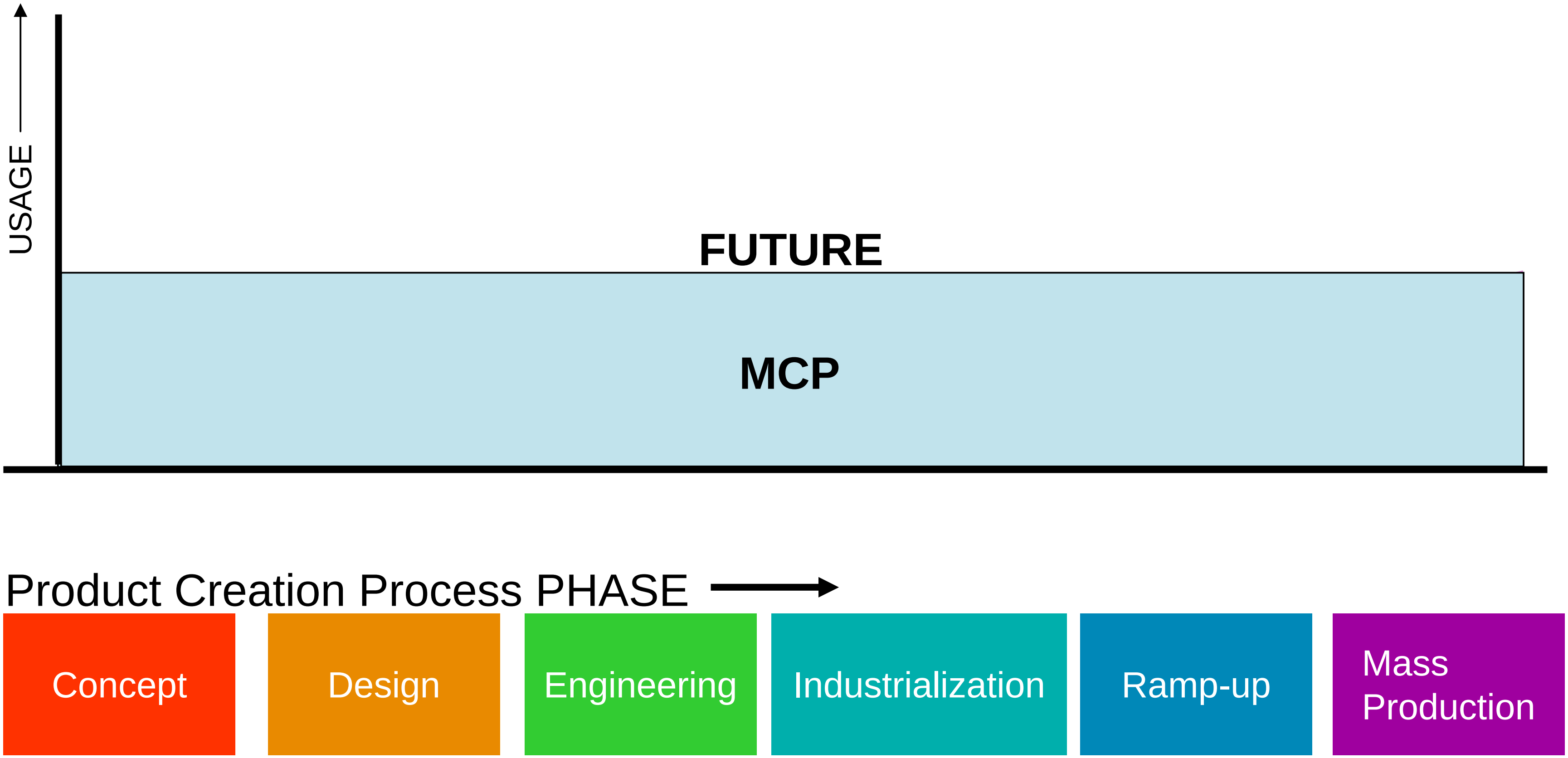
- We are software developers who are responsible for creating industrial products and equipment

We need it because

- The hardware platform of customers often a PC.
- The current toolbox is linked to hardware suppliers
- We need of the shelve solutions
- Customers do not want to pay for the development of a toolbox (comparable to MS-Windows)

☑Why do we need it

We need evolutionary development.


USAGE

FUTURE

MCP

Product Creation Process PHASE →

Concept

Design

Engineering

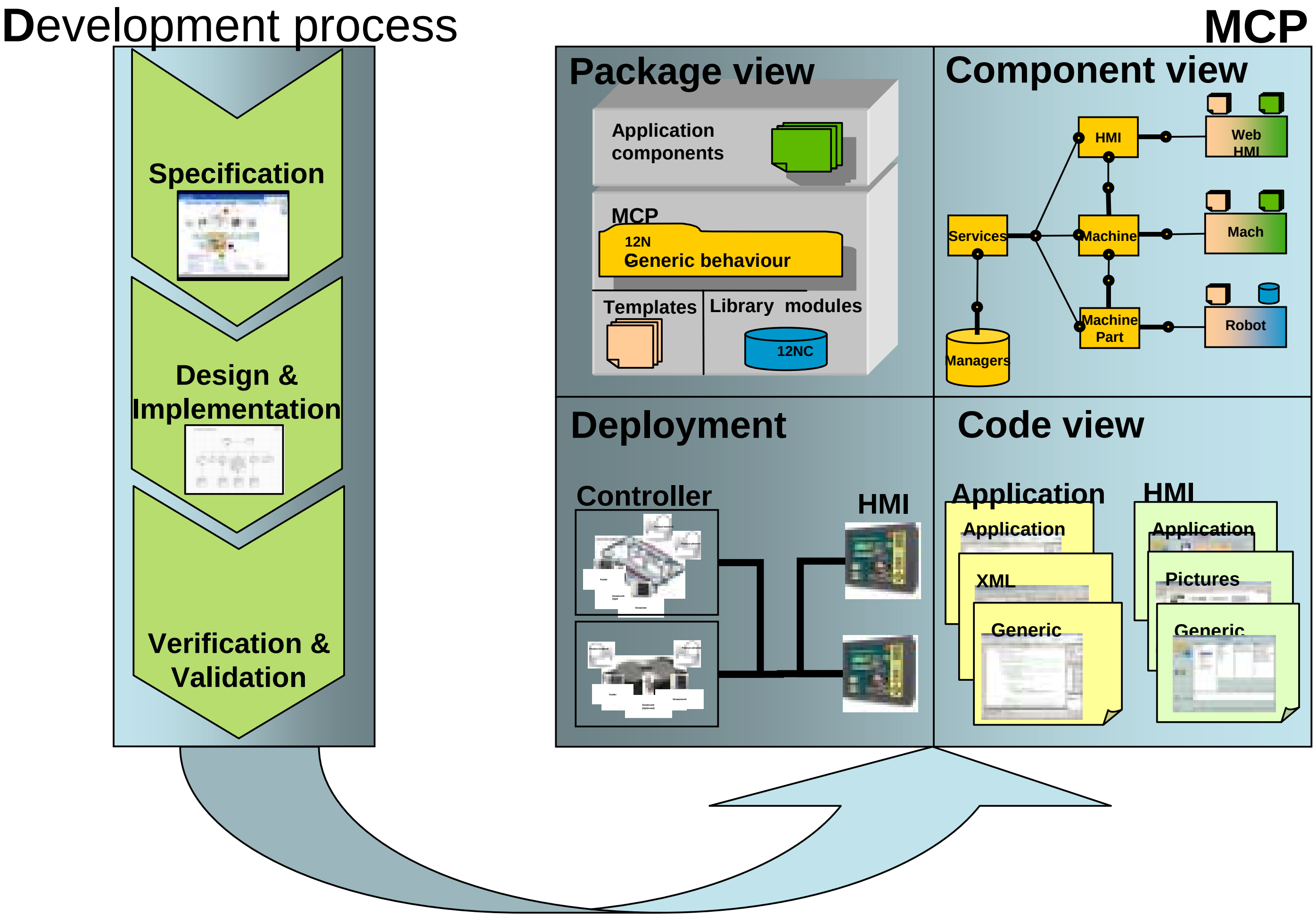
Industrialization

Ramp-up

Mass Production

☑Why do we need it

To close the gap between software developers and other disciplines.



Development process

Specification

Design & Implementation

Verification & Validation

MCP

Package view

Application components

MCP

12N

Generic behaviour

Templates

Library modules

12NC

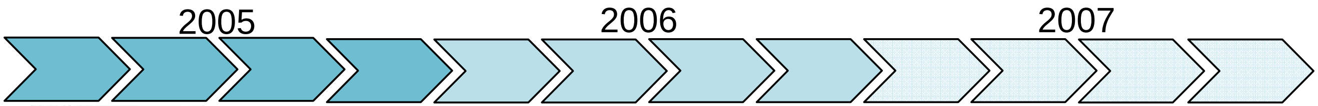
Component view

Deployment

Code view

☑How did we create it

- We as Apptech introduced the MCP ideas into Production equipment
- Philips Lighting did adopt this approach
- Together with Philips Lighting the MCP toolbox was developed for Siemens hardware (funded by Philips Lighting)
- Within different customer projects the platform was converted into
  - Bachmann
  - Beckhoff



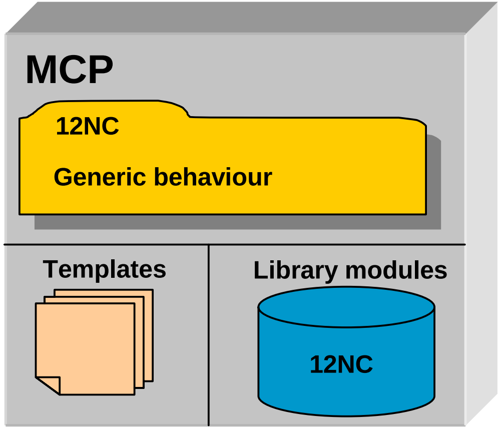
MCP
Siemens

MCP
Bachmann

MCP
Beckhoff

MCP
C# .Net

☑What did we create



The MCP toolbox contains

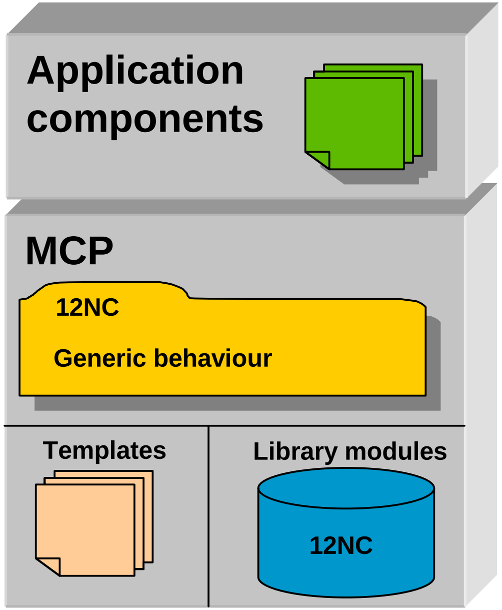
- A Generic behaviour component
  Realisation of a defined infrastructure.

- Templates which can operate with the component generic behaviour
  Application programmers can add functionality by using pre-defined templates.
  Templates can be plugged-in into the generic behaviour component.

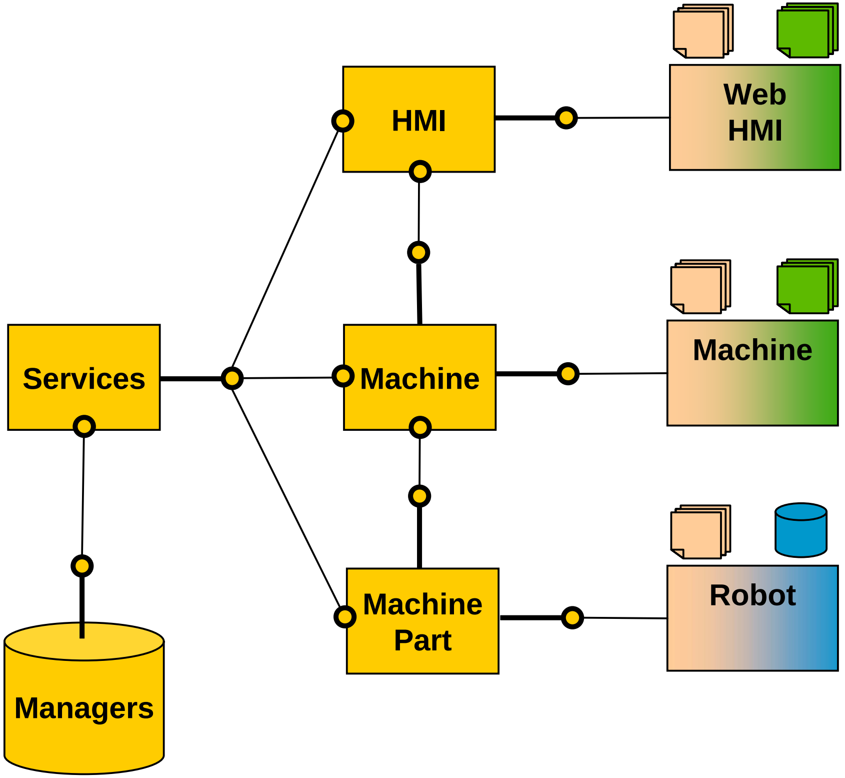
- Library modules
  Realisation of practical examples that are operational together with the generic behaviour component.

☑What did we create

An overview of an application based on MCP

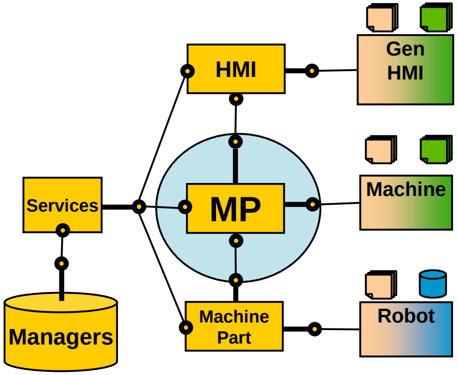


Package view



Component view

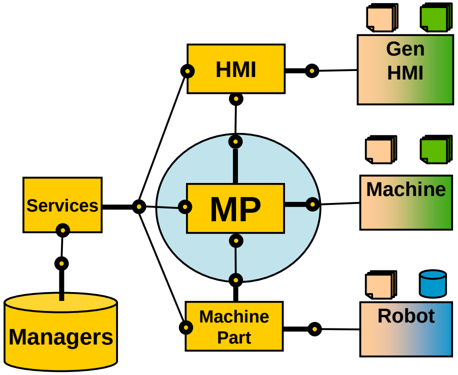
☑What makes it unique



This is the MachinePart pattern.
It is described by the following subjects:

- What is the goal
- What problems do we tackle
- What are the benefits using it
- How is it described (Structure, Roles, Collaborations)
- Examples

☑What makes it unique



- The patterns goal is:

  To define a generic behavior and interface of individual “Machine Parts”

- The following problems are tackled:

  The diversification of application solutions

  The scheduling of the Machine part framework

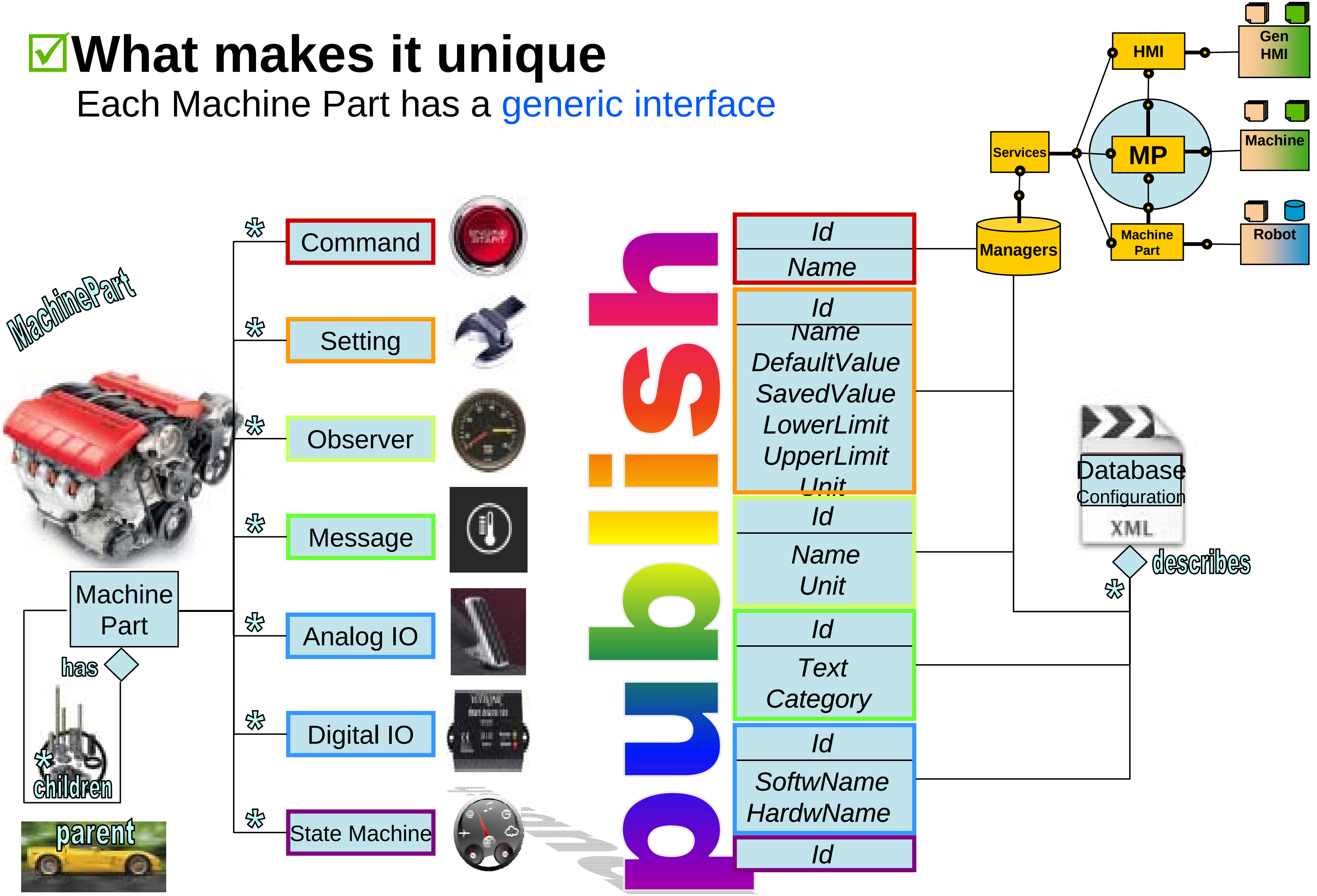
- The following benifists can de described:

  The application developer can focus on the “essence”

  It enables automatic data generation on HMI level

☑What makes it unique

Each Machine Part has a generic interface



MachinePart

Machine Part

has

* children

parent

* Command

* Setting

* Observer

* Message

* Analog IO

* Digital IO

* State Machine

publish

Id
Name

Id
Name
DefaultValue
SavedValue
LowerLimit
UpperLimit
Unit

Id
Name
Unit

Id
Text
Category

Id
SoftwName
HardwName

Id

Managers

Services

HMI

MP

Machine Part

Gen HMI

Machine

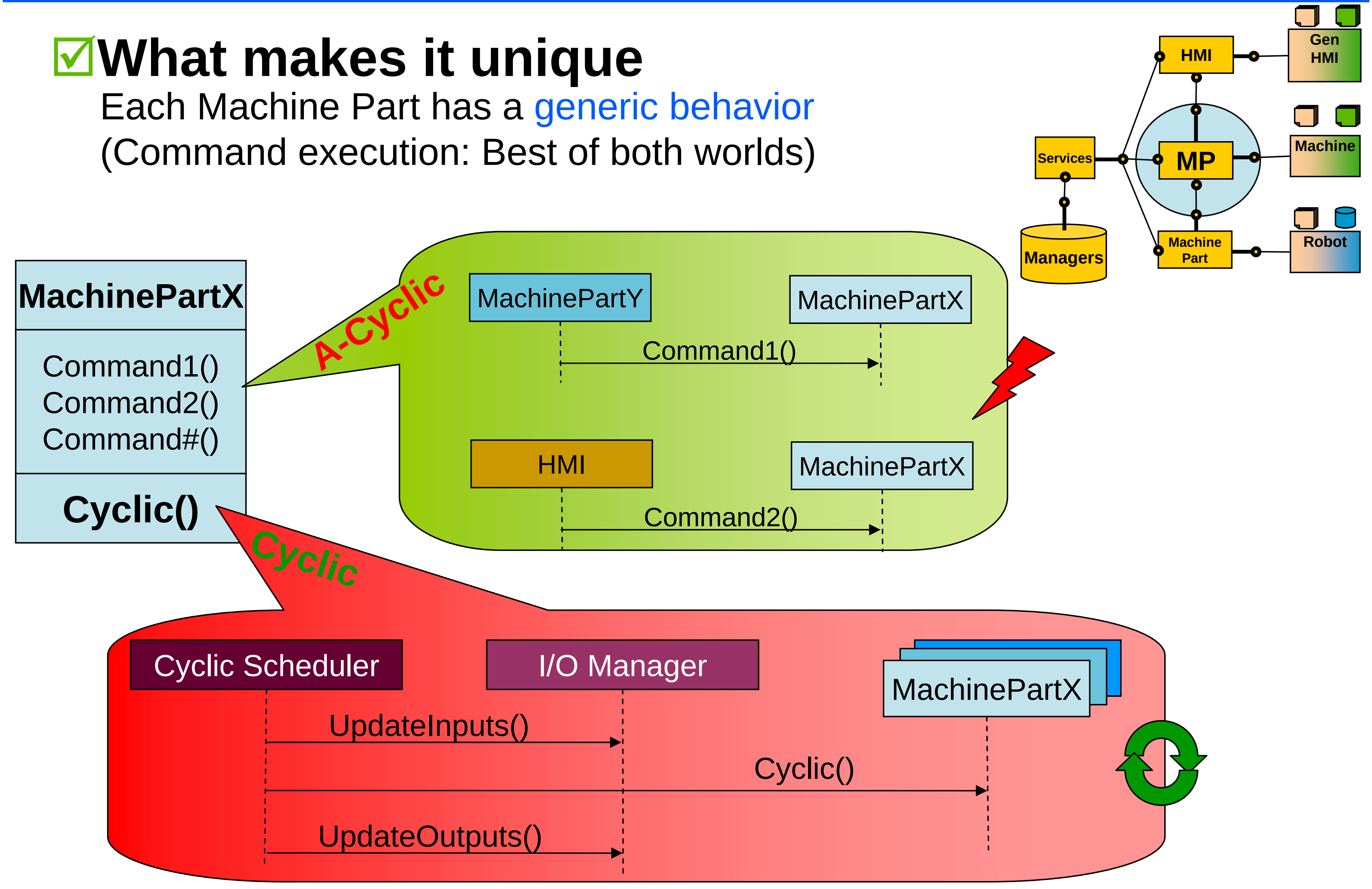
Robot

Database
Configuration
XML

describes
*

☑What makes it unique

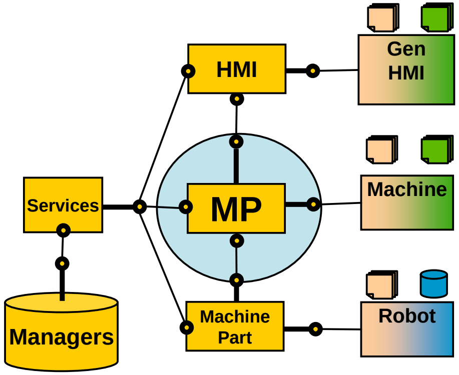
Each Machine Part has a generic behavior
(Command execution: Best of both worlds)

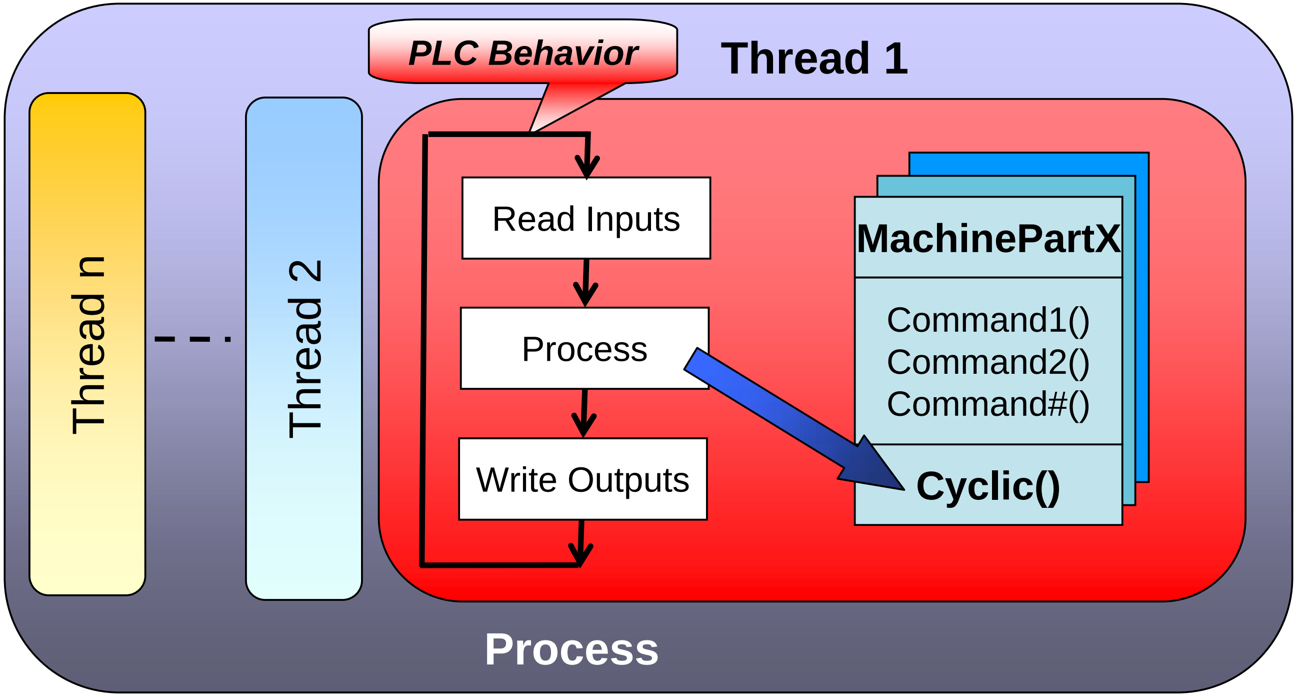


PHILIPS

☑What makes it unique

Each Machine Part has a generic behavior
(Command execution: Best of both worlds)





The scheduling of the machine part framework

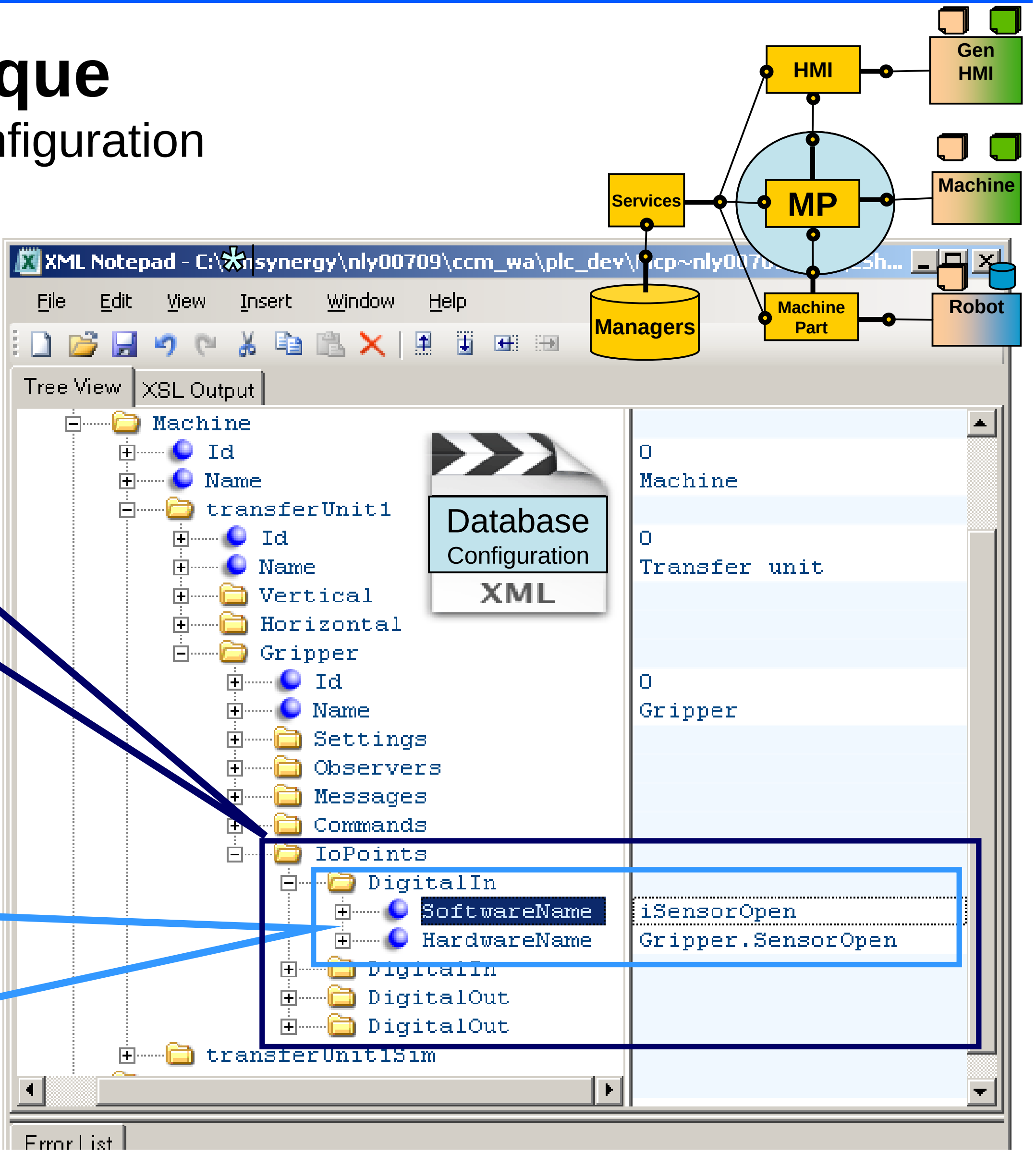
☑What makes it unique

A flexible way of machine configuration

Each Machine Part Contains a list of its I/O points

Software name is used internally

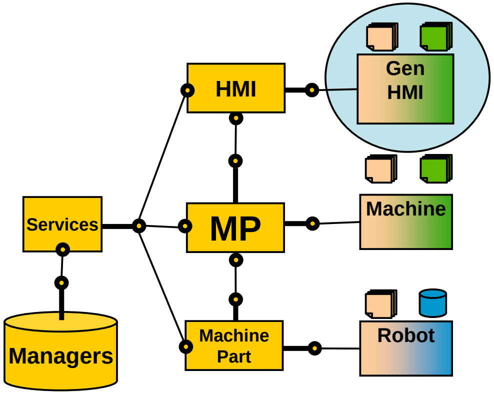
Hardware name makes the coupling to the I/O configuration

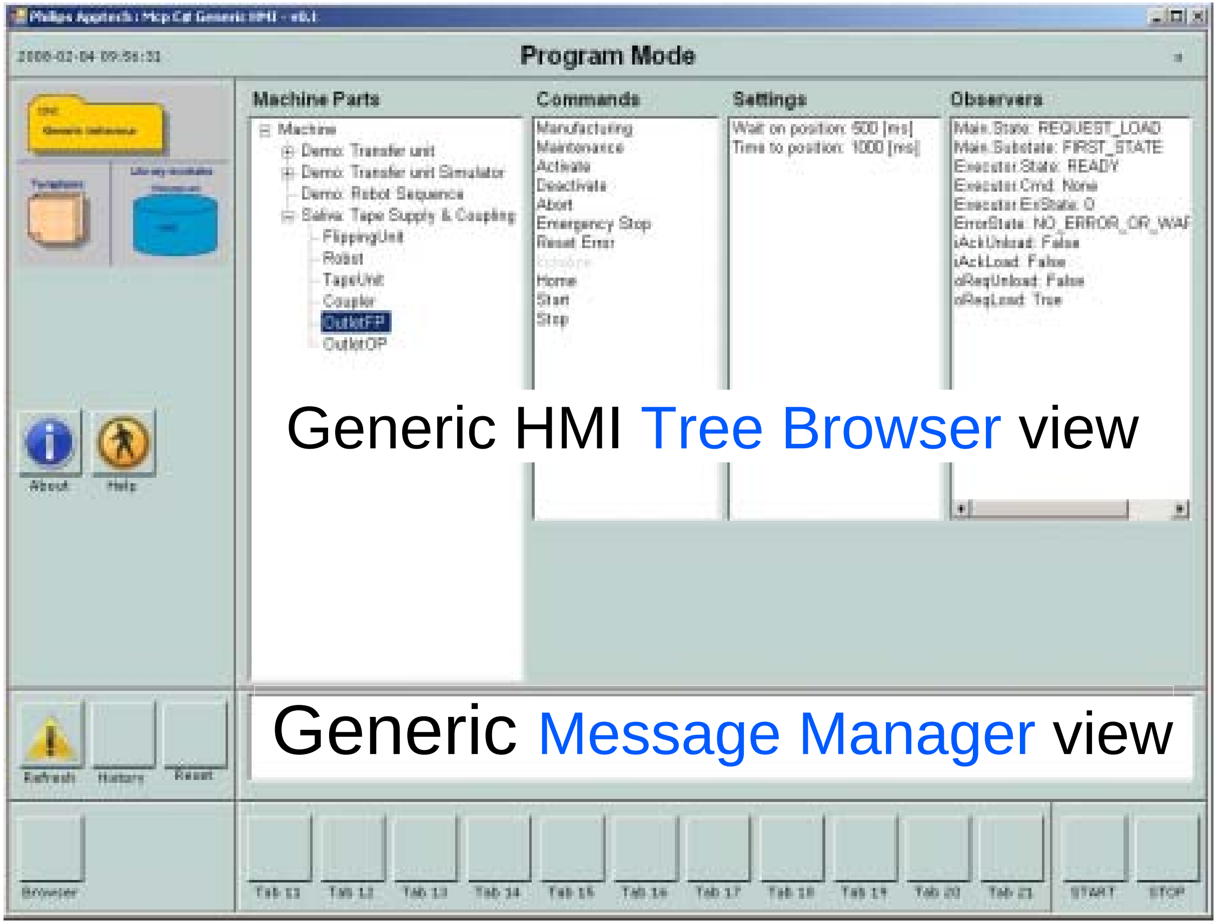


```
Machine
    Id                              0
    Name                            Machine
    transferUnit1
        Id                          0
        Name                        Transfer unit
        Vertical
        Horizontal
        Gripper
            Id                      0
            Name                    Gripper
            Settings
            Observers
            Messages
            Commands
            IoPoints
                DigitalIn
                    SoftwareName    iSensorOpen
                    HardwareName    Gripper.SensorOpen
                DigitalIn
                DigitalOut
                DigitalOut
    transferUnit1Sim
```

☑What makes it unique

The MachinePart pattern enables a generic HMI tree browser view




Philips Apptech : Mcp Cat Generic HMI - v0.1

2008-02-04 09:56:31

Program Mode

Machine Parts

- Machine
  - Demo: Transfer unit
  - Demo: Transfer unit Simulator
  - Demo: Robot Sequence
  - Salvia: Tape Supply & Coupling
    - FlippingUnit
    - Robot
    - TapeUnit
    - Coupler
    - OutletFP
    - OutletOP

Commands

Manufacturing
Maintenance
Activate
Deactivate
Abort
Emergency Stop
Reset Error
Home
Start
Stop

Settings

Wait on position: 500 [ms]
Time to position: 1000 [ms]

Observers

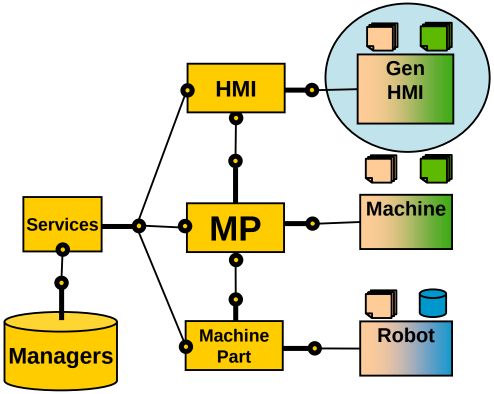
Main.State: REQUEST_LOAD
Main.Substate: FIRST_STATE
Executor.State: READY
Executor.Cmd: None
Executor.ErState: 0
ErrorState: NO_ERROR_OR_WAI
iAckUnload: False
iAckLoad: False
oReqUnload: False
oReqLoad: True

Generic HMI Tree Browser view

Generic Message Manager view

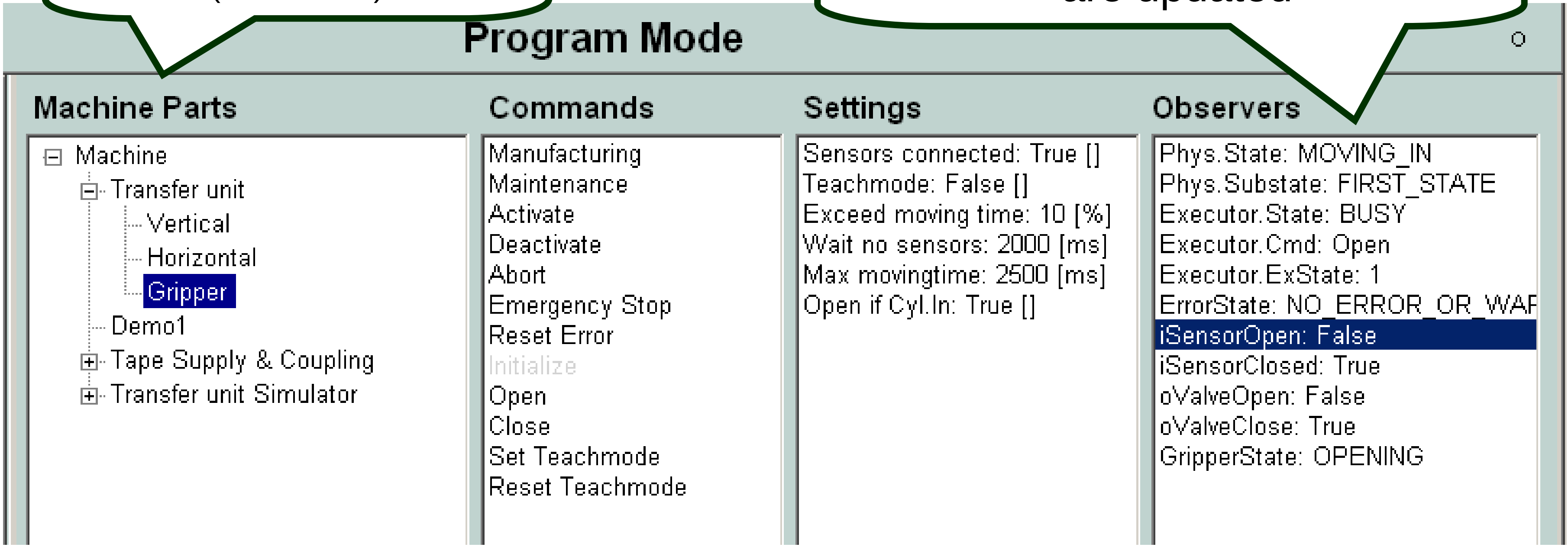
☑What makes it unique

Within the generic HMI tree browser view All Observers and I/O points of each MachinePart are automatically observed



Selected machine part
(in tree view)

Software name and actual value are updated



Program Mode

○

Machine Parts

- ⊟ Machine
  - ⊟ Transfer unit
    - Vertical
    - Horizontal
    - Gripper
  - Demo1
  - ⊞ Tape Supply & Coupling
  - ⊞ Transfer unit Simulator

Commands

Manufacturing
Maintenance
Activate
Deactivate
Abort
Emergency Stop
Reset Error
Initialize
Open
Close
Set Teachmode
Reset Teachmode

Settings

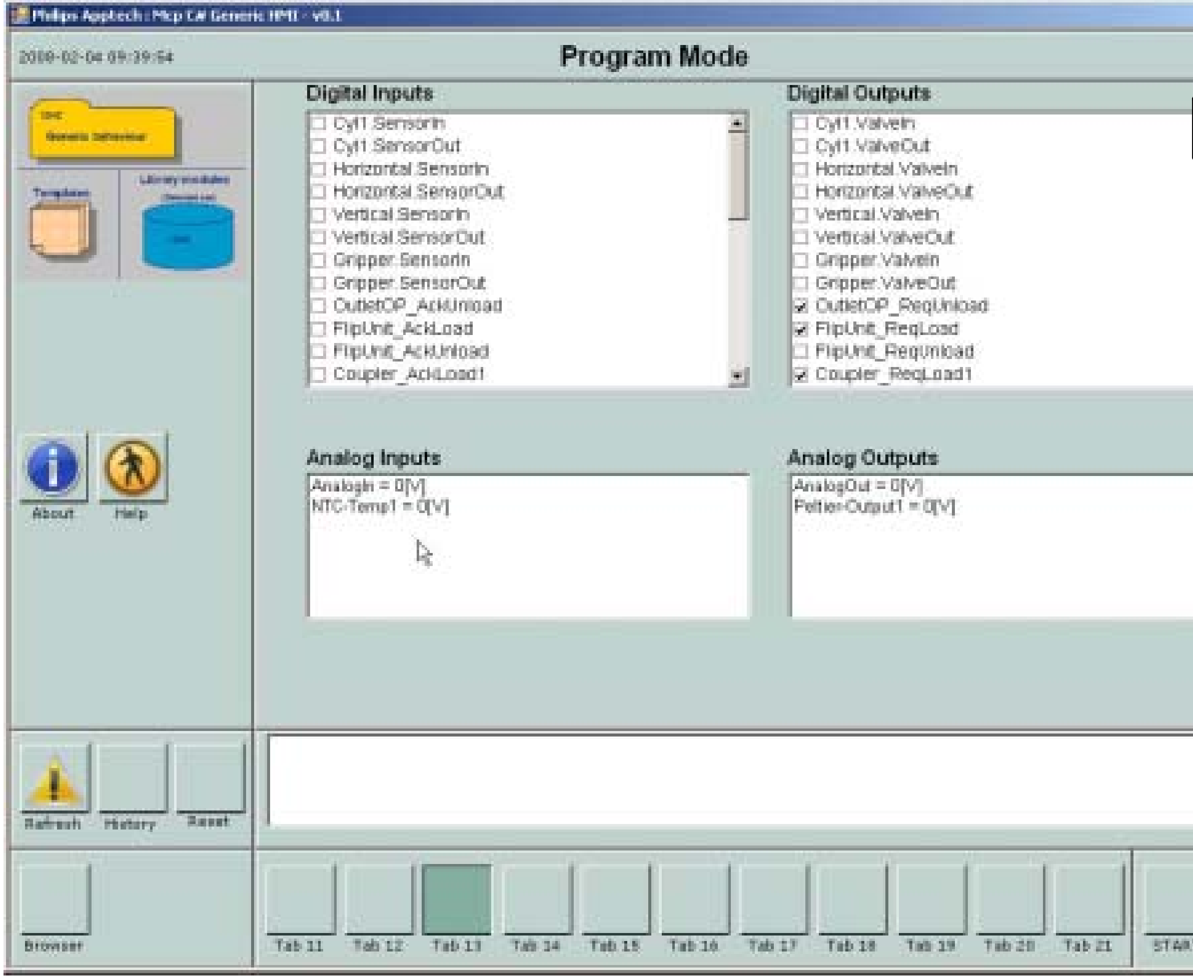
Sensors connected: True []
Teachmode: False []
Exceed moving time: 10 [%]
Wait no sensors: 2000 [ms]
Max movingtime: 2500 [ms]
Open if Cyl.In: True []

Observers

Phys.State: MOVING_IN
Phys.Substate: FIRST_STATE
Executor.State: BUSY
Executor.Cmd: Open
Executor.ExState: 1
ErrorState: NO_ERROR_OR_WAF
iSensorOpen: False
iSensorClosed: True
oValveOpen: False
oValveClose: True
GripperState: OPENING

☑What makes it unique

The MachinePart pattern enables a generic HMI IO-point browser view



Philips Apptech : Mcp C# Generic HMI - v0.1

2008-02-04 09:39:54

Program Mode

Digital Inputs

☐ Cyl1.SensorIn
☐ Cyl1.SensorOut
☐ Horizontal.SensorIn
☐ Horizontal.SensorOut
☐ Vertical.SensorIn
☐ Vertical.SensorOut
☐ Gripper.SensorIn
☐ Gripper.SensorOut
☐ OutletOP_AckUnload
☐ FlipUnit_AckLoad
☐ FlipUnit_AckUnload
☐ Coupler_AckLoad1

Digital Outputs

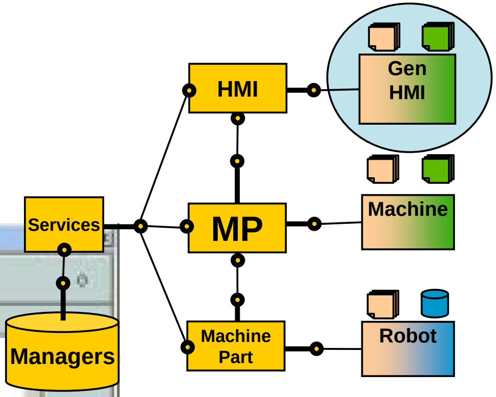
☐ Cyl1.ValveIn
☐ Cyl1.ValveOut
☐ Horizontal.ValveIn
☐ Horizontal.ValveOut
☐ Vertical.ValveIn
☐ Vertical.ValveOut
☐ Gripper.ValveIn
☐ Gripper.ValveOut
☑ OutletOP_ReqUnload
☑ FlipUnit_ReqLoad
☐ FlipUnit_ReqUnload
☑ Coupler_ReqLoad1

Analog Inputs

AnalogIn = 0[V]
NTC-Temp1 = 0[V]

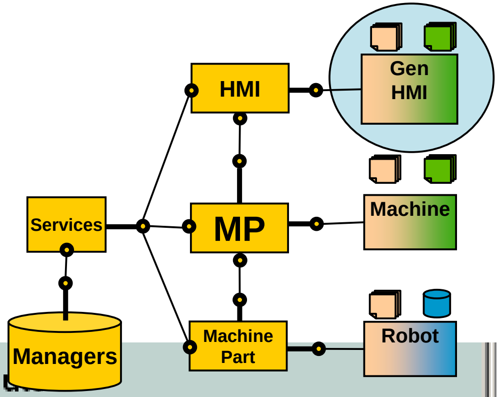
Analog Outputs

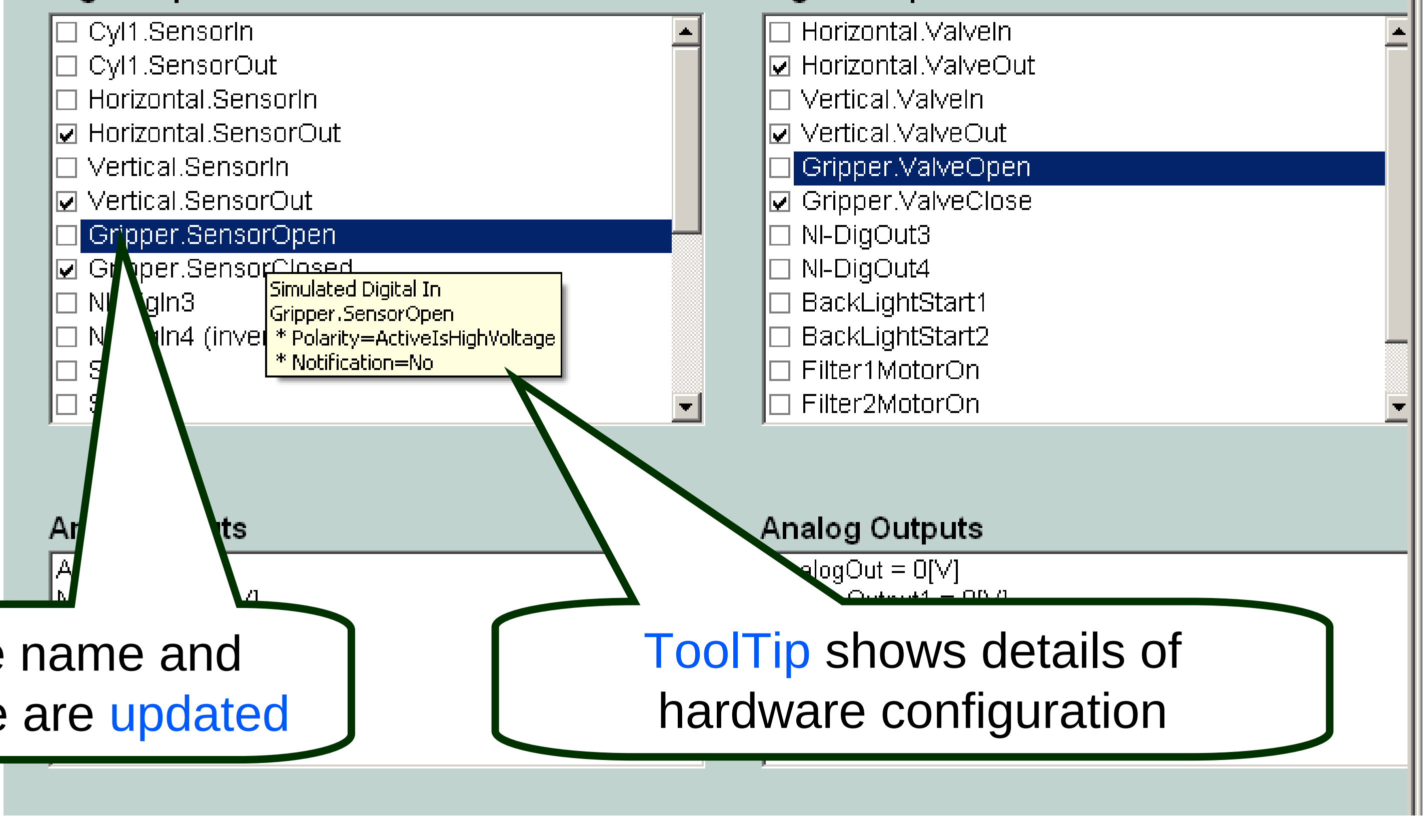
AnalogOut = 0[V]
Peltier-Output1 = 0[V]



☑What makes it unique

- Within the generic HMI IO-point view the whole I/O configuration is automatically observed





Digital Inputs

- ☐ Cyl1.SensorIn
- ☐ Cyl1.SensorOut
- ☐ Horizontal.SensorIn
- ☑ Horizontal.SensorOut
- ☐ Vertical.SensorIn
- ☑ Vertical.SensorOut
- ☐ Gripper.SensorOpen
- ☑ Gripper.SensorClosed
- ☐ NI-DigIn3
- ☐ NI-DigIn4 (inver
- ☐ S
- ☐ S

Simulated Digital In
Gripper.SensorOpen
* Polarity=ActiveIsHighVoltage
* Notification=No

Digital Outputs

- ☐ Horizontal.ValveIn
- ☑ Horizontal.ValveOut
- ☐ Vertical.ValveIn
- ☑ Vertical.ValveOut
- ☐ Gripper.ValveOpen
- ☑ Gripper.ValveClose
- ☐ NI-DigOut3
- ☐ NI-DigOut4
- ☐ BackLightStart1
- ☐ BackLightStart2
- ☐ Filter1MotorOn
- ☐ Filter2MotorOn

Analog Inputs

Analog Outputs

AnalogOut = 0[V]

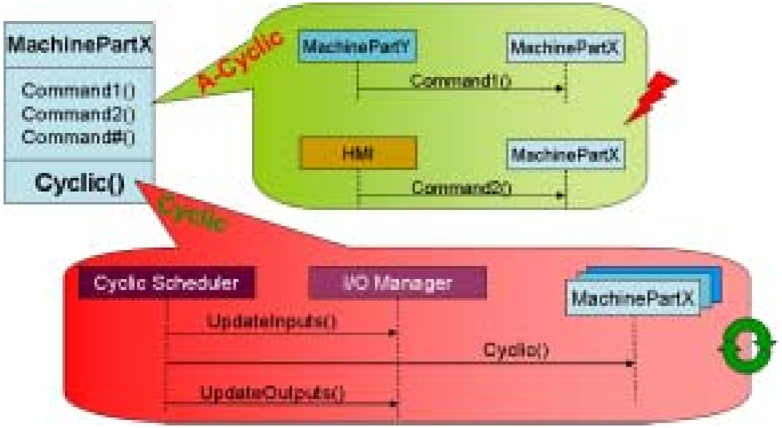
Output1 = 0[V]

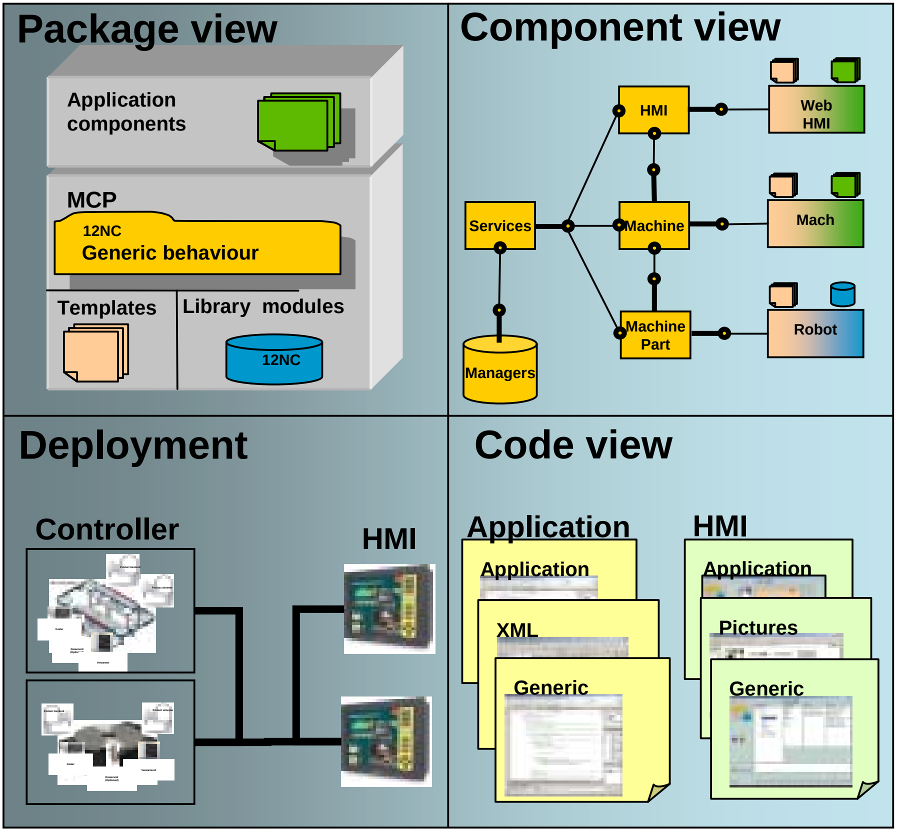
Hardware name and actual value are updated

ToolTip shows details of hardware configuration

☑Summary

The *best* *approach in Product and Equipment development is a combination of* *Cyclic* *(PLC) and* *Event-driven* *(PC) behaviour !!!!!*





PHILIPS