

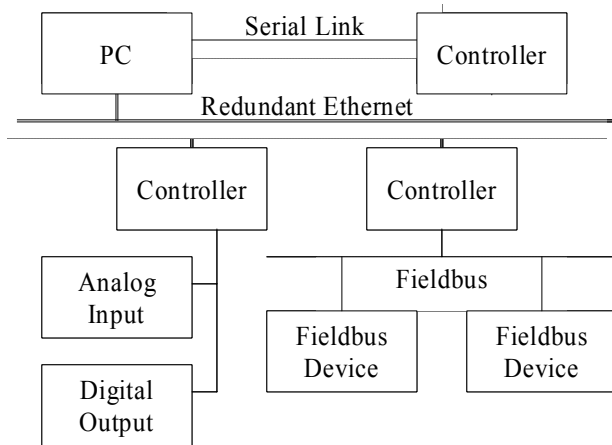
---

# Using Prediction-Enabled Technologies for Embedded Product Line Architectures

Magnus Larsson  
Anders Wall  
Christer Norström

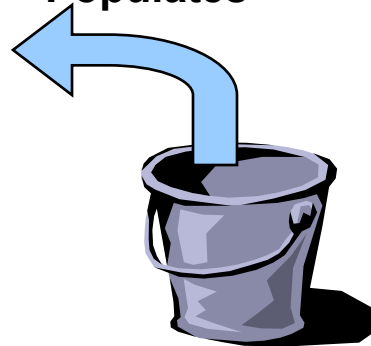
# Product lines

- ❑ Are built on a component technology
- ❑ Using existing components that fits into the architecture



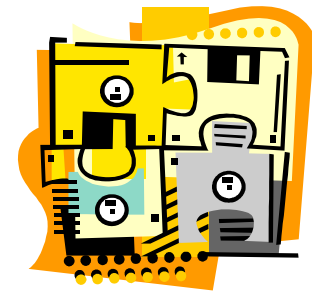
**Architecture**

**Populates**



**Components**

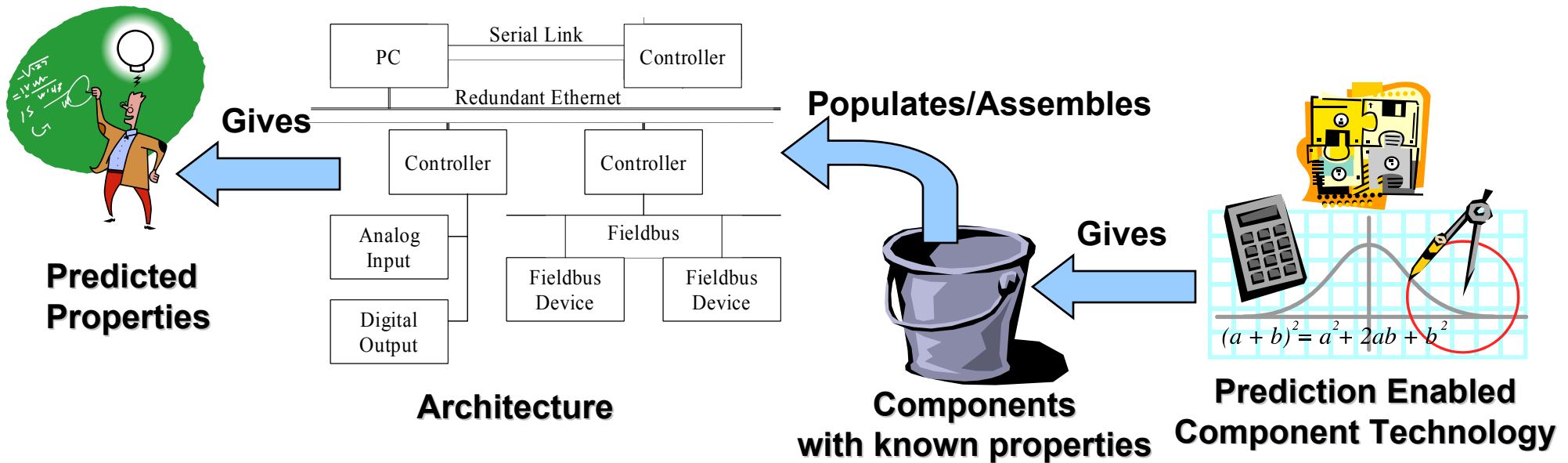
**Gives**



**Component Technology**

# Product lines with prediction

- Are built on a prediction enabled component technology
- Possible to predict properties of the new product



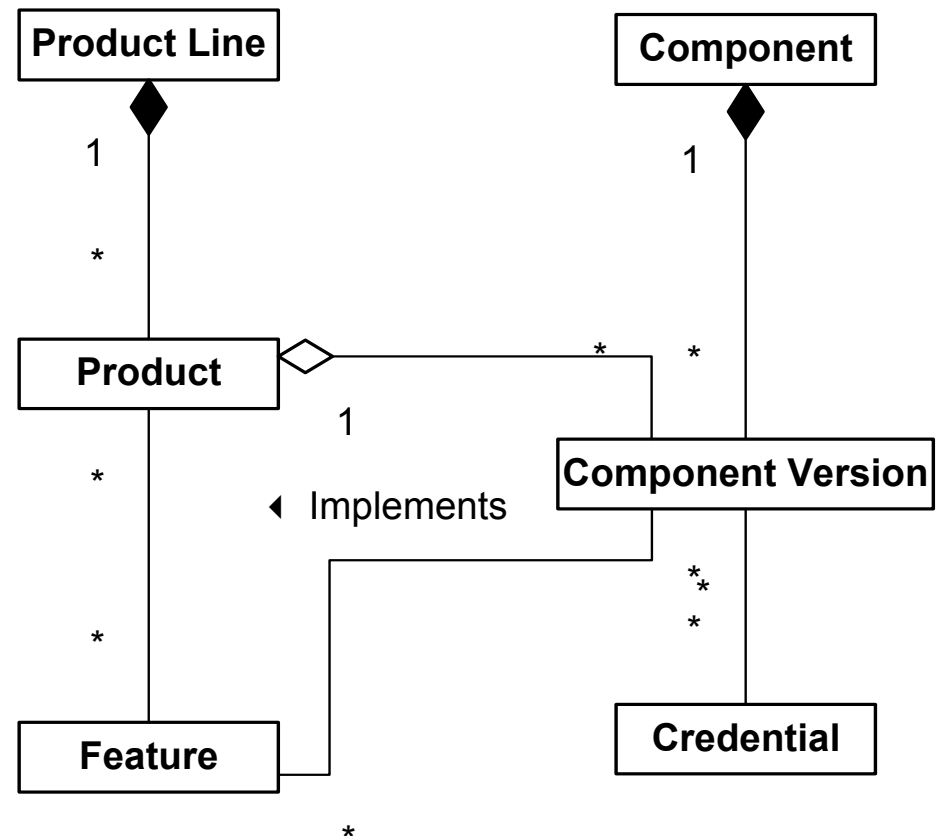
# Outline

---

- Product Lines
- Motivation for a Prediction enabled technology
- How can it be done

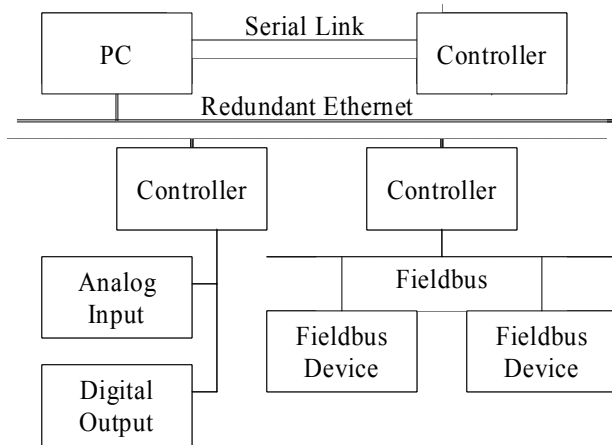
# Product Lines

- ❑ Reusing the architecture
- ❑ Usually build on a component technology
- ❑ Component reuse
- ❑ Products have features that are implemented by components.
- ❑ The implementation of a feature may vary between products
- ❑ The set of features may vary between products



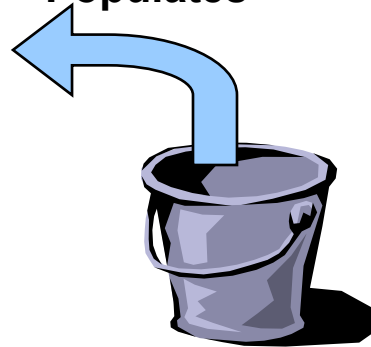
# Product lines

- ❑ Are built on a component technology
- ❑ Using existing components that fits into the architecture



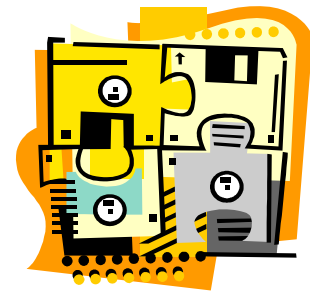
**Architecture**

**Populates**



**Components**

**Gives**



**Component Technology**

# Prediction-Enabled Component Technology

---

- ❑ A component technology (PECT) which has prediction capabilities built into it
- ❑ Contains both analytical and component model
- ❑ The component model defines functional interfaces and the execution strategy
- ❑ Analytical model defines the attributes needed for predicting a property of a set of composed components, i.e. an Assembly

# Why PECT in Product-lines?

---

## Built using reusable assets:

- An architecture
- A set of components
  - Different version or variant for different products
  - Interfaces that has to be implemented for each product
  - Customization of an existing component parameterization

## Answer certain statements

- Have we composed a valid product?
- Is the product temporal correct?

# Why (cont)

---

## ❑ What is the impact of evolving an existing product-line?

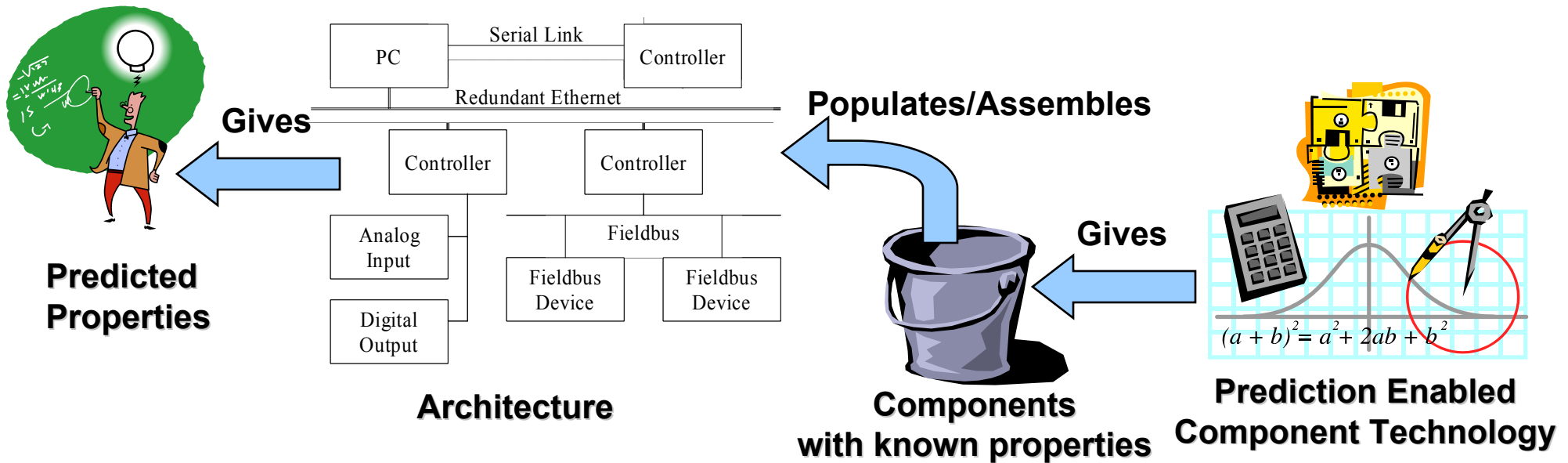
- PLA is an investment that must be used a long time in order to payoff
  - Is the product still temporal correct
  - Are new dependencies introduced that violate the validity of a product?

## ❑ Product lines based on proprietary component model or/and components

- Makes it possible to create a tailored PECT for a particular Product line

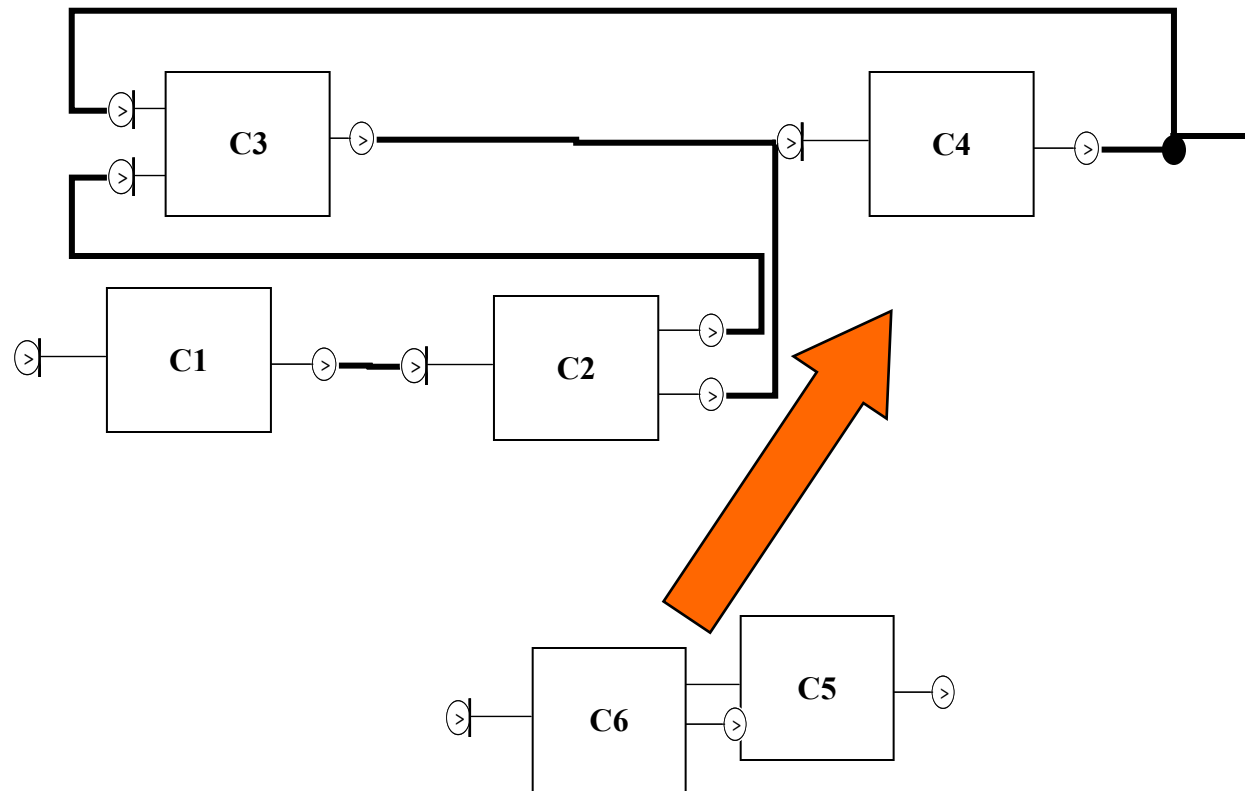
# Product lines with prediction

- ❑ Are built on a prediction enabled component technology
- ❑ Possible to predict properties of the new product



# Scenario1: New features

- A set of components are added to an existing product
- Might cause incompatible problems
- Maintenance can have the same effect

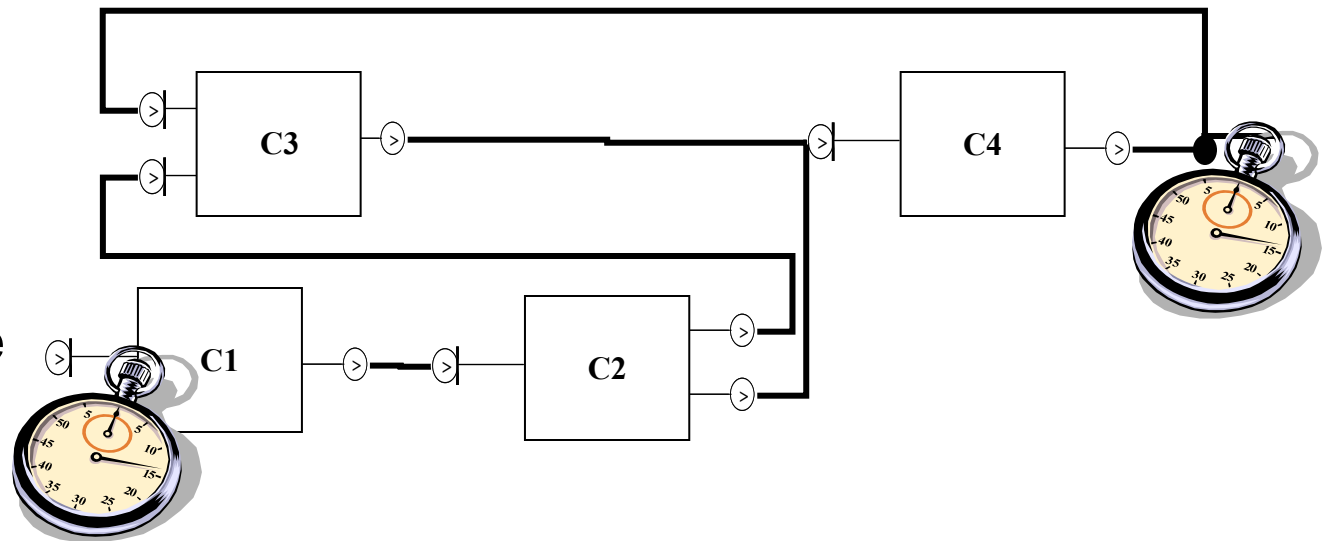


# Scenario 2: Predict the temporal behavior

## □ Analyze

- End to end deadlines
- Response time

## □ Existing techniques as RMA can be used



# How

---

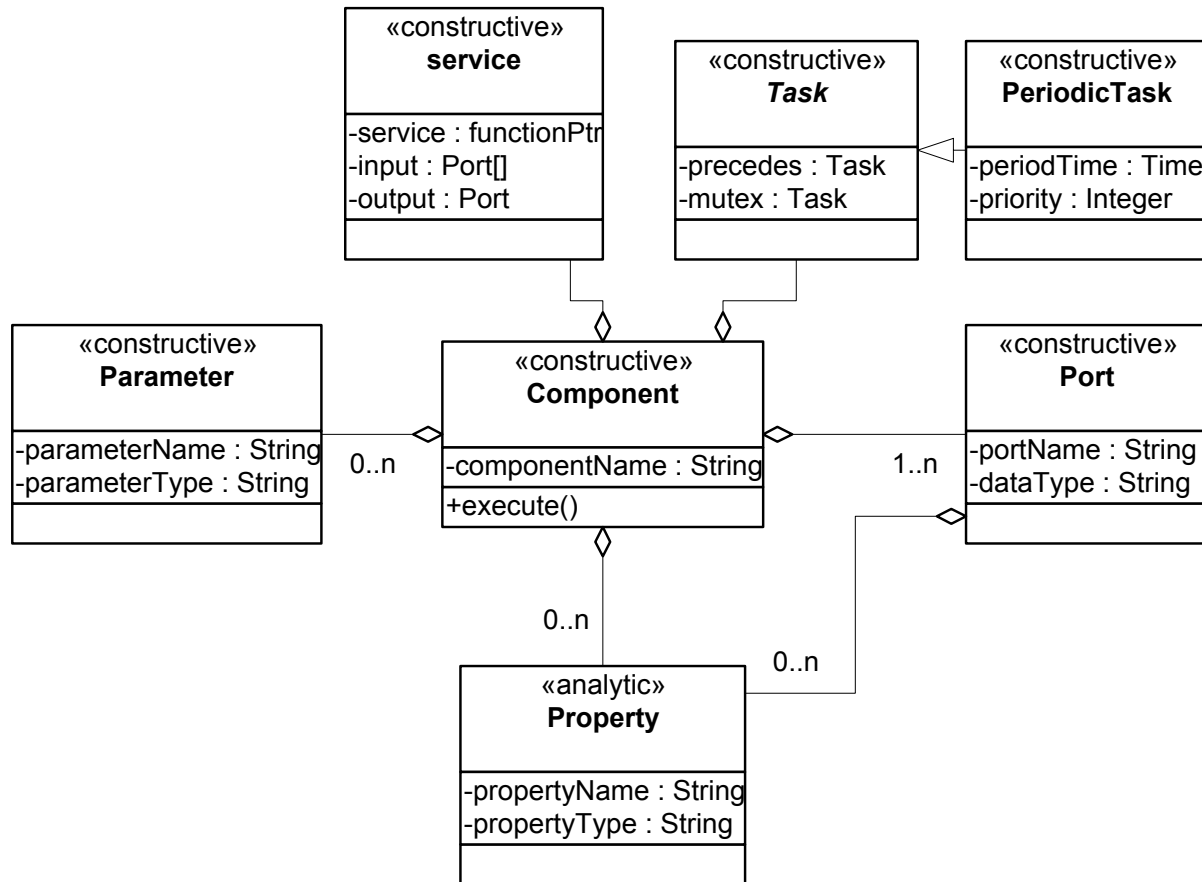
- ❑ **Create a PECT that support**
  - Component model
  - Analytic model
- ❑ **Create a set of components in the PECT**
- ❑ **Define Prediction theory and validate it**

# Analytic Study of PECT

---

1. Define the goal of the study
2. Define the property of interest
3. Define the process of how inferences are to be drawn
4. Analytic study
5. Define the test environment
6. Evaluate relevance of existing process for desired inferences
7. Define sampling procedure
8. Select sample
9. Perform analyses

# Our Component Model



# Component Model Definitions

---

## □ We define

- Component
- Assembly
- Precedence
- Mutual exclusion
- Data flow

# Component

---

A *component*  $c$  is a tuple  $\langle f, P, I, O, \tau, s_c \rangle$ , where  $f$  is the service encapsulated by  $c$ ,  $P$  is the set of parameters,  $I$  is the set of in-ports,  $O$  is the set of out-ports,  $\tau$  is the task that controls the execution of  $c$ , and  $s_c$  is the state of component  $c$ .

- A component provide a service
- The parameters are set during deployment
- A Component task execute if and only if when the associated task executes

# Assembly

---

An *assembly*  $A$  is a tuple  $\langle C(A), R^* \rangle$ , where  $C(A) \subseteq C$  is the set of components in  $A$ , and  $R^*$  is the set of relations valid between  $C(A)$  in  $A$ , and  $C$  is a set of all components encapsulated in the product

- ❑ An assembly is a set of composed components
- ❑ The components in a assembly are interconnected by the relations  $R^*$ , specified in an assembly:
  - Data
  - Precedence
  - Mutual exclusion

# Precedence

---

A *precedence* relation,  $\rightarrow$ , is a binary, transitive relation among a pair of tasks  $\langle \tau_i, \tau_j \rangle \in T \times T$ , such that if  $\tau_i \rightarrow \tau_j$ , then  $\tau_j$  may start its execution earliest at the end of  $\tau_i$ 's execution and  $i \neq j$

- ❑ Tasks can execute in a defined order
- ❑  $C1 \rightarrow C2$  means that C1 executes before C2

# Mutual exclusion

---

A *mutual exclusion* relation,  $\otimes$ , is a binary, symmetric relation among pair of tasks  $\langle \tau_i, \tau_j \rangle \in T \times T$ , such that if  $\tau_i \otimes \tau_j$  then neither  $\tau_i$  nor  $\tau_j$  is permitted to execute while the corresponding party, or a transitively related party is executing and  $i \neq j$

- Tasks may not access the same resource at the same time

# Data flow

---

A *data flow* connection =, is a binary, anti-symmetric relation among pair of ports on components,  $\langle c_i \cdot i_x, c_j \cdot o_y \rangle \in C.I \times C.O$ , such that if  $c_i \cdot i_x = c_j \cdot o_y$  then  $c_i$ 's in port  $i_x$  is connected to  $c_j$ 's out port  $o_x$

- Out-ports are connected to In-ports

# Rate Monotonic Analysis

---

$$R^{n+1}(c_i) = c_i.wcet + B(c_i) + \sum_{\forall c_j \in hp(c_i)} \left\lceil \frac{R^n(c_i)}{c_j.T} \right\rceil c_j.wcet$$

**Where:**

**R = the response time for a component**

**C.wcet = the worst case execution time for C**

**B(C) = Blocking time for component C**

**C.T = Period time for C**

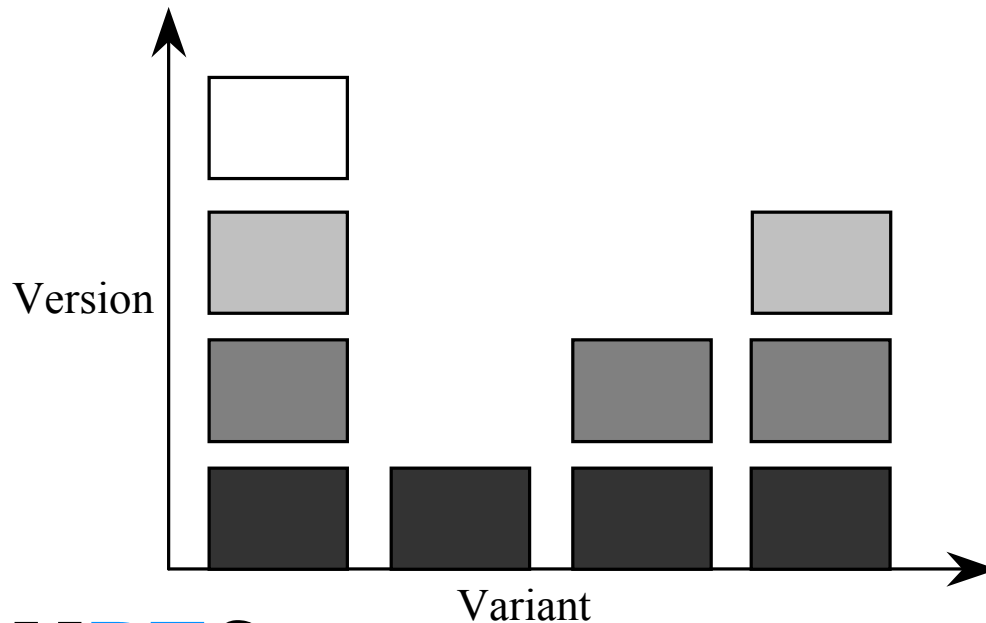
# Variant- and version consistent

An assembly  $A$  is variant- and version consistent,  $A$ .consistent if:

$A$ .consistent =  $\forall \langle \langle c_i, \text{variant}, x \rangle, \langle c_i, \text{variant}, y \rangle \rangle \in V \times V: x=y$

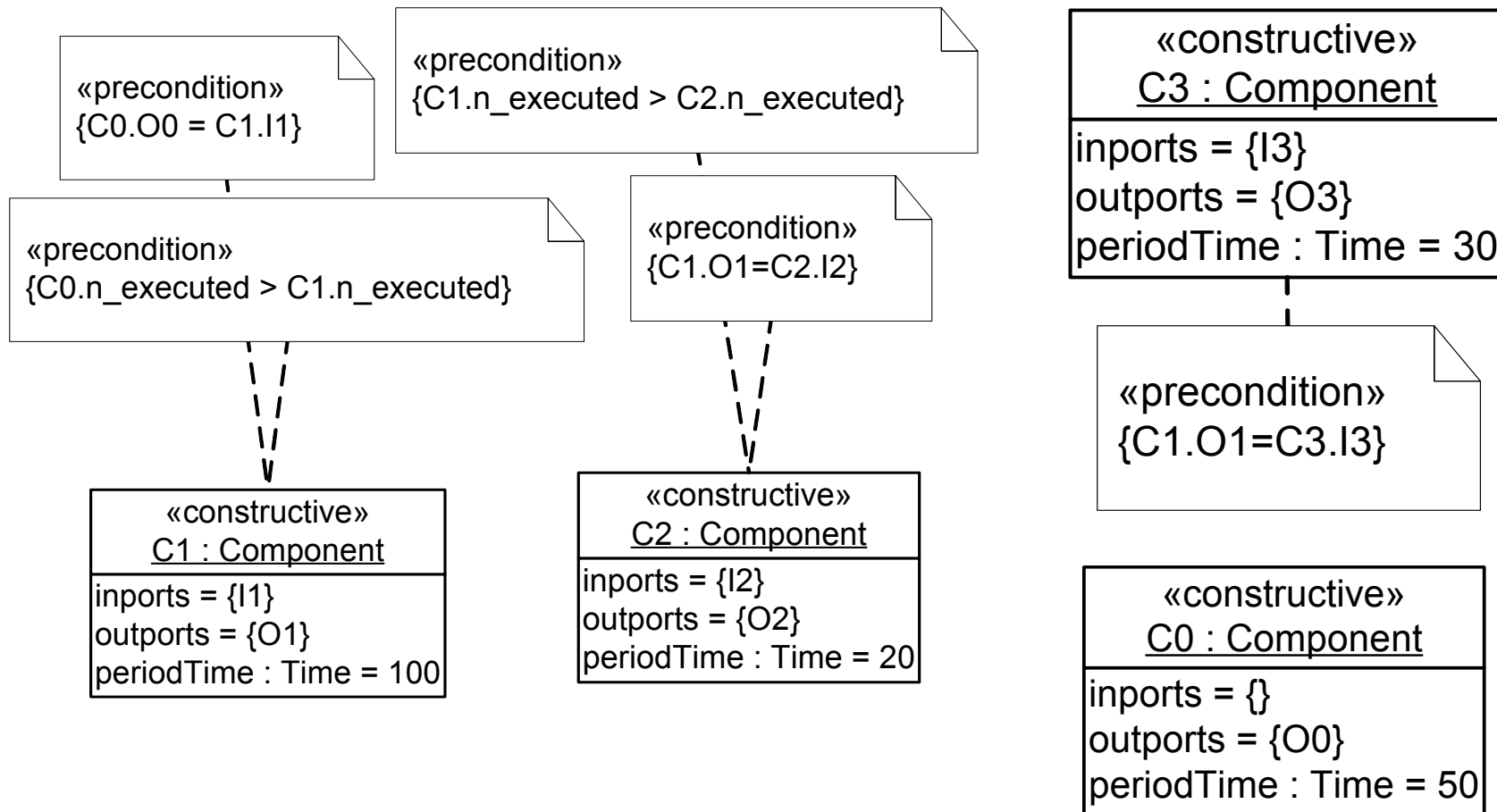
$$V = \bigcup_{c_i \in C(A)} c_i.\text{depends}$$

,where ,  $c_i \in C(A)$ , *variant* is a component variant and  $x,y$  are versions.  $\square$

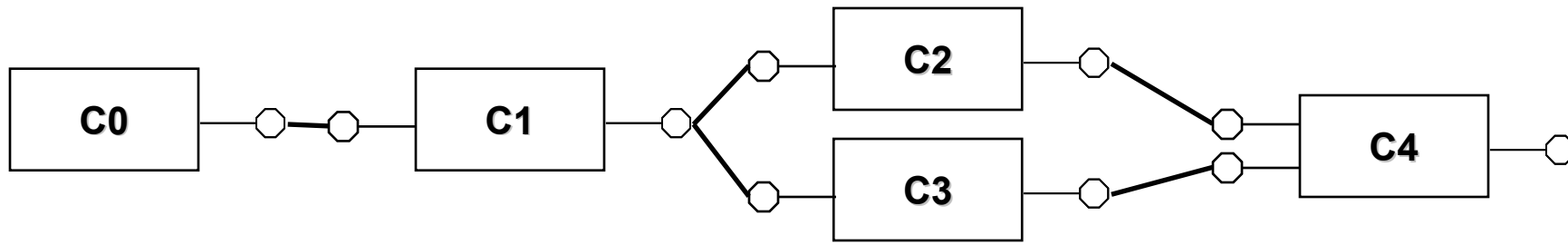


The assembly is consistent if a component does not appear twice with different version in the union set of all dependencies.

# Example in UML



# An Example Assembly



«precondition»  
 {C3.n\_executed > C4.n\_executed}

«constructive»  
 C4 : Component

---

inports = {I4, I5}  
 outports = {}  
 periodTime : Time = 40

«precondition»  
 {C3.O3 = C4.I4,  
 C2.O2 = C4.I5 }

«precondition»  
 {C4.depends\_on.includes(C3.version) and  
 C4.depends\_on.includes(C2.version)}

# Conclusion: Product lines with prediction

- ❑ Are built on a prediction enabled component technology
- ❑ Possible to predict properties of the new product
  - Such as temporal correctness, consistency or reliability
- ❑ Design the PECT so it can support many different properties

