

# Real Time Systems Architecting with PPOOA

Presentation at Unterschleißheim  
April 28, 2006

PPOOA, an Architectural Style  
PPOOA\_AP, an architecting Process  
PPOOA-Visio, a CASE tool

José L. Fernández  
Independent Consultant  
Associate Professor at Madrid Technical University (UPM)  
[fernandezjl@acm.org](mailto:fernandezjl@acm.org)

## The issue: Development effort in embedded systems projects

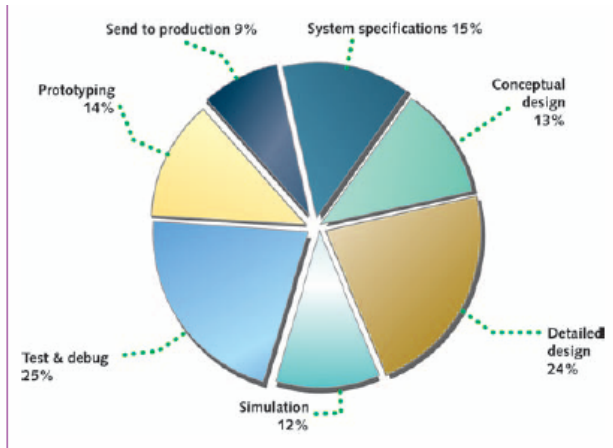


Figure illustrates that an important effort (25%) is allocated to test & debug by industry.

When professional engineers spend more time fixing their own mistakes than they do making them, there is something fundamentally wrong<sup>1</sup>.

<sup>1</sup> Jim Turley. "Development teams are getting bigger and richer". ESP Oct 2005

A wide ranging survey of developers from across North America and Europe performed by Embedded Systems magazine, revealed some interesting information about how embedded systems developers are working.

# Content

---

- **Architecture and Architectural Styles**
- **Real Time Systems**
- **Real Time Architectural Styles**
- **PPOOA Metamodel**
- **PPOOA Architecting Process (PPOOA\_AP)**
- **PPOOA\_AP Guidelines**
- **Experiences of using PPOOA and PPOOA\_AP**
- **PPOOA implementation on Visio CASE tool**
- **Next PPOOA tool features**
- **To conclude**

## Software Architecture

---

- A software architecture describes the structural properties of the software product, particularly its components and their dependencies, rationale and guidelines about their use.

## Architectural Style

---

- Shaw defines an architectural style as a family of architectures constrained by component/connector vocabulary, semantic constraints and analysis methods supported<sup>1</sup>.

<sup>1</sup>Shaw, M. Comparing Architectural Design Styles. IEEE Software November 1995.

## Real Time Systems

---

- A real-time system is one in which correctness depends on meeting time constraints
- Real-time systems often have to deal with multiple independent streams of input events and produce multiple outputs. These events have arrival rates often unpredictable, although they should be processed meeting timing constraints specified by software requirements.

# Real Time Architectural Styles<sup>1</sup>

---

- **Timeline.** It uses fixed time frames within each all the application procedures are executed in some predetermined sequence. It is also called a cyclic executive architecture.
- **Event-driven.** It consists of a set of tasks, each awaiting an event. Each task is provided with a priority.
- **Pipeline.** It also consists of a set of tasks waiting for events but they send remaining processing and the final response to events to other scheduled entities elsewhere in the system.
- **Client-Server.** Although the event still arrives at an initial schedulable entity, the successive processing stages are invoked using procedure calls from the initial task, rather than been invoked using a one-way message.

<sup>1</sup>Locke C.D. An Architectural Perspective of Real-Time Ada Applications. Reliable Software Technologies- Ada Europe 99. Lecture Notes in Computer Science 1622. Springer Verlag, 1999.

## The Designer's problem



How to find the best architecture that fits the customer demands?

- Common building elements, and a rigorous and easy to be adopted architecting process (PPOOA and PPOOA\_AP)



How to document and modify the architecture?

- A CASE tool implementing the building elements and the architecting process of PPOOA (PPOOA-Visio)

28 April 2006

©José Luis Fernández-Sánchez

8

The important issue regarding methods is to implement best practices in industry. It is not only a matter of tools, it requires motivated and well trained engineers.

Typically novice designers have problems to identify domain concepts besides things or physical entities. A good domain model is a prerequisite for the identification of system conceptual components and system executable components. Good architecting methods can help in component identification and description.

The improvement in productivity and quality is the benefit.

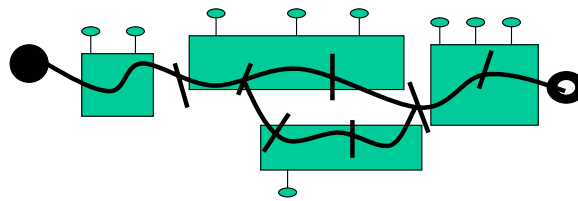
# PPOOA

---

- A model based approach for architecting real-time systems
  - Based on [UML](#) notation
  - Supports a diversity of components and coordination mechanisms (for synchronization and communication) not found in UML.
  - Improves [behavior modelling](#) (“causal flow of activities” modelling) and allows performance assessment.
  - An [architecting process \(PPOOA\\_AP\)](#), defining the steps to build the architecture
  - A [CASE tool \(PPOOA-Visio\)](#)

## PPOOA, A Real-Time Architectural Style

- PPOOA, Processes Pipelines in Object Oriented Architectures is an architectural style for concurrent object oriented architectures. It can be used when individual paths of execution are required to be concurrent and several processes may be positioned along the path to control the action.



28 April 2006

©José Luis Fernández-Sánchez

10

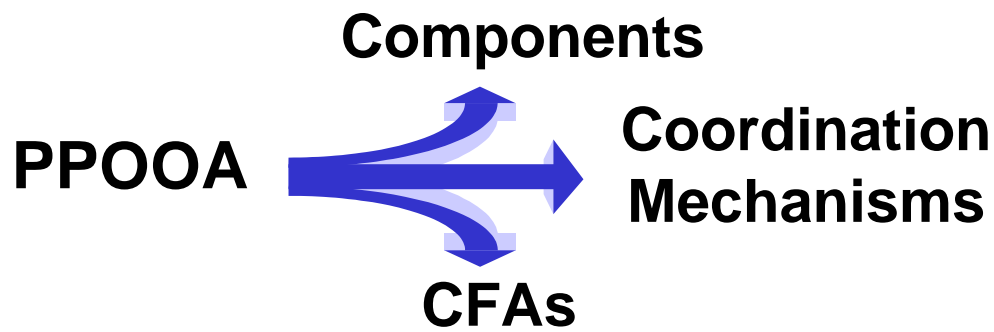
It works similar to a “conveyance system” where diverse machines and workers do some work on the “processed item”, while the conveyor moves it.

This architecture style is recommended in systems acquiring and processing bunch of raw data. Typically Supervisory Control and Data Acquisition Systems (SCADA), Air Traffic Control Systems (ATC), robotics and some avionics subsystems.

The style may be applied to a part of the complete architecture, when other part is developed using another architectural style.

## PPOOA Building Elements

---



28 April 2006

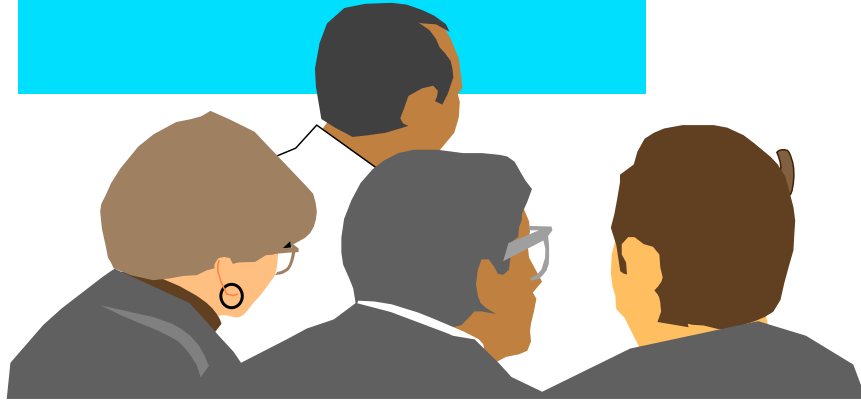
©José Luis Fernández-Sánchez

11

The vocabulary for building elements for real-time systems proposed in the PPOOA style consists of: Component, Coordination Mechanism and CFA.

Each of the basic elements is specialized in several types of elements as described later.

**UML METAMODEL EXTENSION TO  
INCLUDE PPOOA BUILDING  
ELEMENTS**



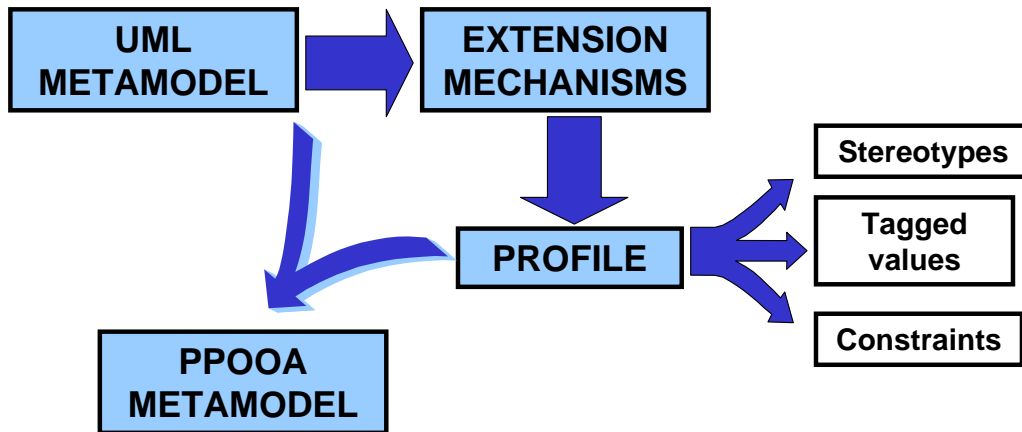
28 April 2006

©José Luis Fernández-Sánchez

12

An important issue of UML is its capacity to be extended by using the profiling concept that allows the compatibility with other UML modeling issues. A UML profile is defined as a stereotype of package that groups model elements that have been customized for a specific domain. It allows one to develop variants of the UML suited for such specific domain.

## PPOOA Metamodel Creation



28 April 2006

©José Luis Fernández-Sánchez

13

A stereotype is a semantic feature attached to an existing model element. Stereotypes not only classify model elements at the object model level, but they facilitate the addition of “virtual” UML metaclasses with new metaattributes and semantics.

The other extension mechanisms allow users to attach additional properties and semantics directly to individual model elements, as well as to model elements classified by a stereotype.

It is important to make clear that stereotypes may extend the semantics, but not the structure of preexisting elements.

Any modeling element may have arbitrary information attached to it, such as management information, code generation information or additional semantic information, in the form of a property list consisting of tag-value pairs.

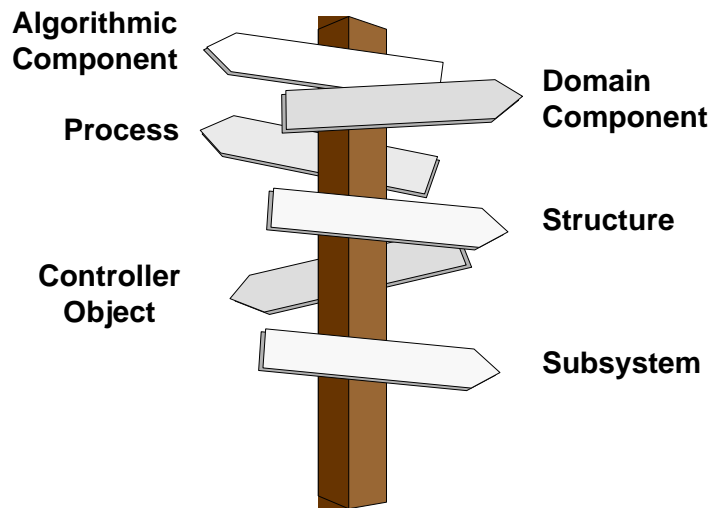
A tag represents the name of an arbitrary property with the given value. For example, preconditions and postconditions attached to operations in a class are examples of standard tagged values.

Constraints, as well as tagged values, are used to extend the semantics of model elements classified by that stereotype.

Constraints can be attached to classes or objects, restricting in this way the semantics of these elements.

The constraints must be observed by all model elements marked with that stereotype.

## Components (I)



28 April 2006

©José Luis Fernández-Sánchez

14

•**Algorithmic Component.** Algorithmic components or utilities are elements of the architecture that perform calculations or transform data from one type to another but separated from its structural abstraction . Data classification components, Unix filters or data processing algorithms typically represent them.

•**Domain Component.** The domain component is an element of the architecture that responds directly to the modeled problem. This component does not depend on hardware or user interface . Its instances are similar to the classic object in object oriented design.

•**Process.** The process is a building element of the architecture that implements an activity or group of activities that can be executed at the same time as other processes. Its execution can be scheduled. The PPOOA vocabulary supports two different types of processes: periodic and aperiodic. Each of them is described in PPOOA profile and their real-time attributes are also considered .

•**Structure.** A structure is a component that denotes an object or class of objects characterized as an abstract state machine or an abstract data type. Typical examples are: stack, queue, list, ring and others.

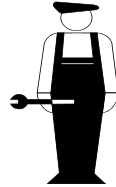
•**Controller Object.** The controller object is responsible for initiating and managing directly a group of activities that can be repetitive, alternative or parallel. These activities can be executed depending on a set of events or conditions. Typically, a controller object receives an event. An event is something that can happen, like an OS signal, a hardware interrupt or it represents a computed event, like an airplane entering a conflict volume or an alarm. Where one of these events occurs, the system schedules associated event handlers.

## Components (II)

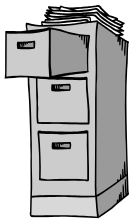
---



**Controller :**  
Manages external  
events



**Domain component/  
Algorithmic component:**  
Performs operations



**Structure:**  
Maintains relations  
between objects



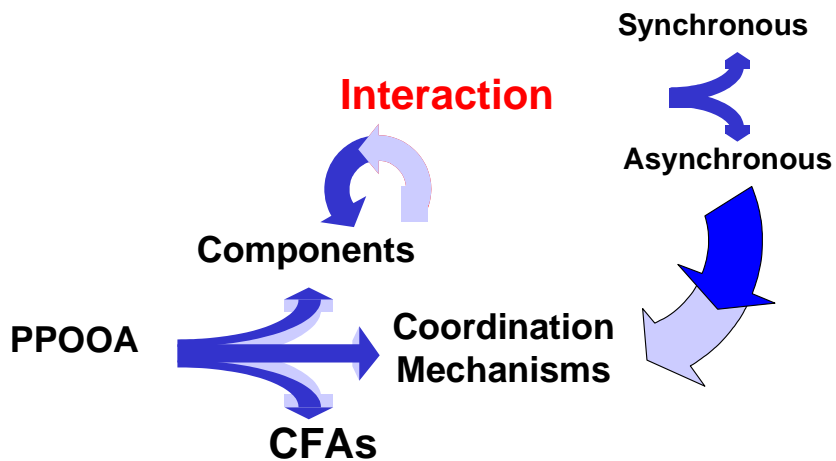
**Process:**  
Coordinates work  
to others

28 April 2006

©José Luis Fernández-Sánchez

15

## Interaction between Components (I)



28 April 2006

©José Luis Fernández-Sánchez

16

Component instances interact through messages. A message can be implemented by a simple operation call, or as a signal object that can be put in a coordination mechanism (bounded buffer, mailbox...)

A synchronous message between component instances indicates wait semantics.

An asynchronous message between component instances indicates no-wait semantics.

## Interaction between Components (II)

---

•The synchronous interaction is supported by operations modeled with interfaces.

### INTERFACE IS

-NAME name of the operation

-TYPE operation type

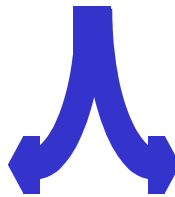
-SIGNATURE (parameters, type of parameters)

•The asynchronous interaction is supported by the PPOOA Coordination Mechanisms.

# Coordination Mechanisms

Coordination Mechanisms support two major issues of real-time systems: *synchronization* of flows of activities and *asynchronous communication* between the components of the system.

***Synchronization*** is the blocking of a process until a certain condition is met.



***Communication*** is the transfer of information between components.

28 April 2006

©José Luis Fernández-Sánchez

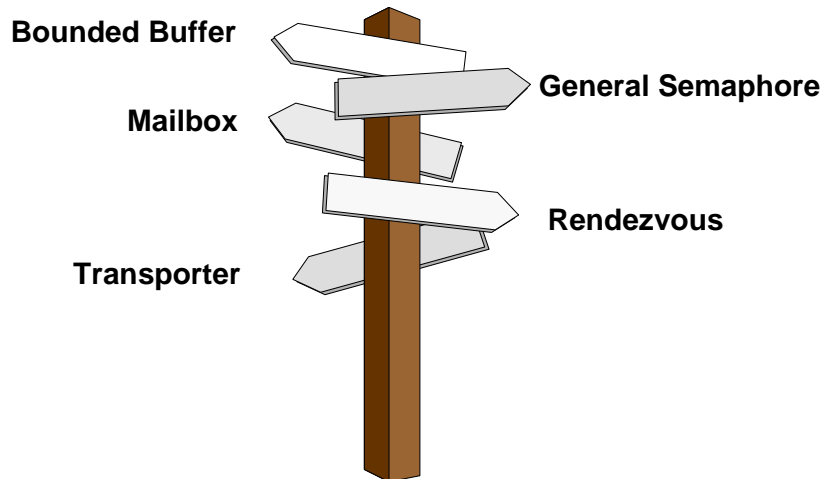
18

One of the contributions of the PPOOA style is to model these properties accordingly to the FODA (Feature-Oriented Domain Analysis) methodology of Domain Analysis . FODA was applied as a novelty because it was the first time that it was used in an architectural level in the analysis and classification of a coordination mechanisms taxonomy [Fernandez 93].

Features are classified into:

- Identification features deal with those features that describe how coordinating processes refer to each other.
- Synchronization features describe those features related to the exclusion and ordering constraints of the coordination mechanisms.
- Communication features describe the communication capabilities of the coordination mechanism.

## Coordination Mechanisms included in PPOOA vocabulary



28 April 2006

©José Luis Fernández-Sánchez

19

•**Bounded-Buffer:** It is a temporary and capacity limited holding area, from which data producers and consumers can asynchronously get or send data to.

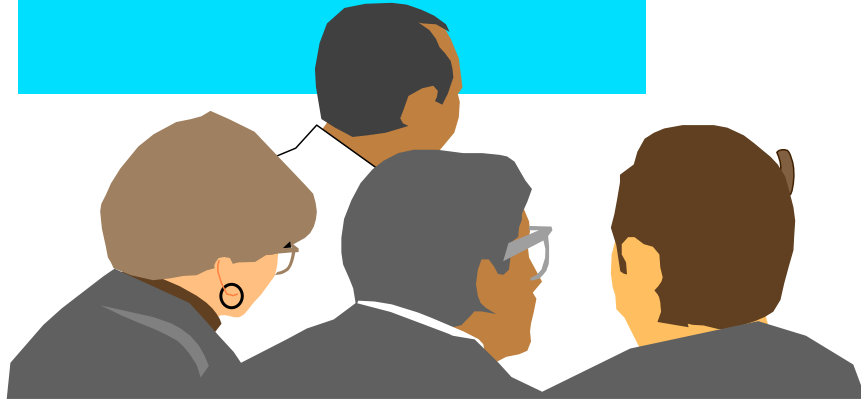
•**General-Semaphore:** It is a pure synchronization mechanism implemented as a non-negative integer value that is used to synchronize processes or control the access to a critical region.

•**Mailbox:** It is a coordination mechanism used to pass messages asynchronously between processes. The mailbox capacity is for one message but the messages may have a variable length.

•**Rendezvous:** It is a synchronous and unbuffered coordination mechanism that allows two processes to communicate bidirectionally.

•**Transporter:** It is a mechanism that makes calls to get and send messages from and to the coordinating partners (producer and consumer).

## UML METAMODEL EXTENSION TO INCLUDE PPOOA CFAs



28 April 2006

©José Luis Fernández-Sánchez

20

A CFA (Causal Flow of Activities) is a cause-effect chain of activities that cuts across different building elements of the architecture. This chain progresses through time and executes the activities provided by the system components until it gets to an ending point.

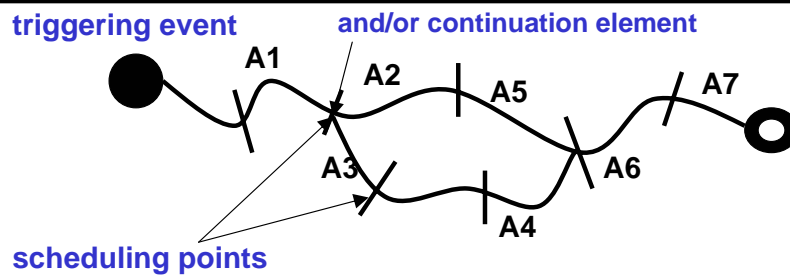
As we can see in the CFA metamodel [Fernandez 01], a CFA consists of one or more Path Elements. These Path Elements are mapped to Classifiers.

Therefore, Classifiers implement the Path Elements. ■

## CFA (I)

### DEFINITION

**CFA means Causal Flow of Activities. Therefore, a CFA is a chain of activities that is triggered by an event.**



28 April 2006

©José Luis Fernández-Sánchez

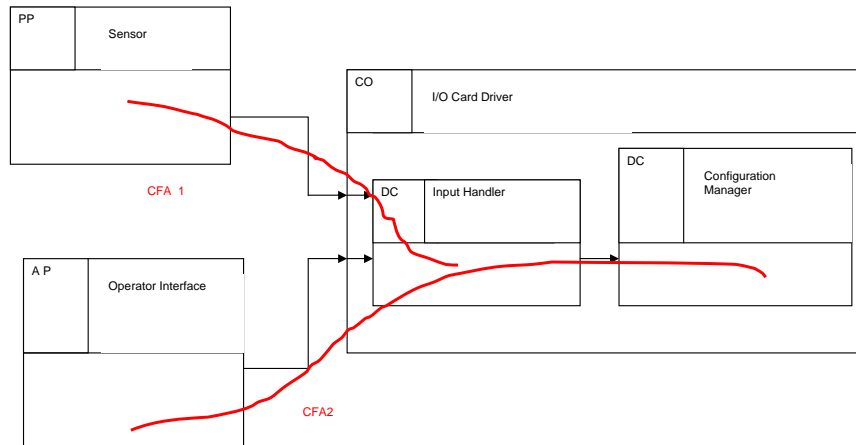
21

CFAs are used in PPOOA to describe a system response. A CFA is a causality flow of activities triggered by an event that can be external, internal or timer.

The chain of activities implemented by the architecture components progresses through time until it gets to an ending point. Several active CFAs can exist at the same time and may even interact with each other. CFAs can also stop at certain waiting points, where coordination mechanisms or objects with concurrent access are located.

CFAs can split into parallel flow of activities, and flows of activities can also merge together. Activities are the smallest division of a response. The computation that takes place in an activity cannot cause changes in the system resources allocation. The system resources allocation is done at scheduling points.

## CFA (II)



28 April 2006

©José Luis Fernández-Sánchez

22

In the example we identify two CFAs (red lines): One is representing the sample sensor response. The other CFA is representing the change sensor parameters response.

An important issue that can be seen in the slide, is that some system components participate in both CFAs (system responses)

# PPOOA\_AP: PPOOA Architecting Process

28 April 2006

©José Luis Fernández-Sánchez

23

PPOOA Architecting Process is focused on the development of a software architecture for a system conforming the principles inherent to PPOOA architectural style and using the vocabulary of components and coordination mechanisms proposed by PPOOA.

The scope of PPOOA architecting process is the architectural or preliminary software development phase of the software life cycle. For this reason, the architecting phase is described in more detail than in a full cycle software development methodology .

## Why a new architecting process?

- Traditional CBD and OO architecting approaches focus upon producing encapsulations and abstractions for system componentry.
- The effort of the resulting architecture on the ability of a system to meet its time responsiveness expectations requires additional understanding well beyond functionalities and their combined computational timing requirements.
- Concurrency modeling and synchronization behavior become a dominant concern early in the architecture development whenever time is a critical factor
- Object definition and collaboration strategies should reflect meaningful timing constraints.

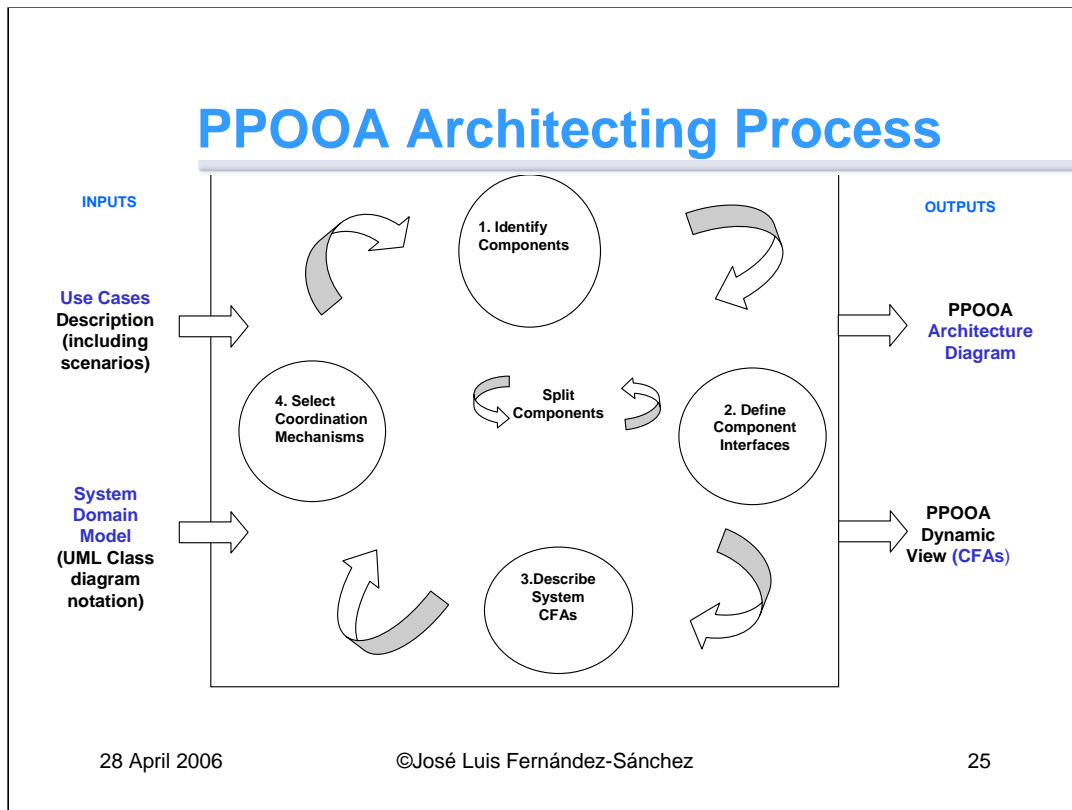
28 April 2006

©José Luis Fernández-Sánchez

24

Architecture is critical to the realization of many system qualities, and these qualities should be designed and evaluated at the architecture level.

Responsiveness is a critical quality attribute of real-time systems. It is related to the response time to stimuli (events) either external or internal to the system or timed.



PPOOA architecting process is essentially an iterative process that is split into major steps and minor steps. Architecting process major steps are those that follow the identification of the major components of the system, their interfaces and the main flows of activities that the system will support.

These major steps are namely:

1. Identify Components
2. Define Component Interfaces
3. Describe System CFAs
4. Select Coordination Mechanisms

The process shown in the slide above, is repeated iteratively as we split main Components into subcomponents in top-down decomposition and responsibilities allocation process.

The main inputs to the PPOOA architecting process are the use cases and the system domain model described by either one or several UML class diagrams.

Although UML sequence diagrams are not essential as an input to PPOOA architecting process, they will also be very helpful.

## **PPOOA\_AP Guidelines**

- **Architecture design principles and component selection criteria are described as guidelines**
- **25 guidelines are proposed in PPOOA\_AP**

28 April 2006

©José Luis Fernández-Sánchez

26

The recommendations or guidelines that form the “design space” represent the design principles embedded in PPOOA architectural style. This “design space” has a double utility.

- On one hand it is possible to know the principles that are inherent to the architecture and that must be met when working with multiple style architectures.
- The guidelines are also very useful for guiding the novice software architects that are not experienced in using PPOOA vocabulary and diagrams.

## **Groups of Guidelines**

---

- **Guidelines relative to PPOOA Architectural Style**
- **Guidelines relative to the Use of PPOOA Components**
- **Guidelines relative to the use of PPOOA Coordination Mechanisms**
- **Guidelines for CFA Construction**
- **Guidelines for process Scheduling**

# Guideline Example

## SCHEDULING GUIDELINE

**Allocation policies of other shared resources, different from the CPU, are considered when these resources are shared between activities of different CFAs.**

28 April 2006

©José Luis Fernández-Sánchez

28

### Guideline explanation (Extracted from PPOOA AP Documentation):

In the Architecture the shared resources are those components that encapsulate shared data, coordination mechanisms and devices, for example data acquisition boards. Other resources such as buses or local area networks are not considered here.

Allocation policies for those components that encapsulate shared resources may be:

- FIFO (First In First Out).
- Request ordering regarding arrival time
- Request ordering regarding priority
- Priority Inheritance Protocol. The priority of a process that uses a resource, if there are others waiting, is temporally increased to the requester priority.
- “Highest Locker”. The priority of the process that uses the resource is the highest of all priorities of those processes that can access the resource.
- Priority Ceiling Protocol. It is based on a priority current ceiling of all the used resources.

A process can access the resource only if its priority is higher than the current ceiling. If the access to a resource is denied to a certain process, then the process that has the resource at that moment inherits the blocked process priority.

In the PPOOA architectural style, the “Highest Locker” allocation policy is recommended because it is very simple and the execution platform in this way does not require modifications.

Devices such as disks, terminals, I/O boards normally follow FIFO allocation policies, although in some cases they can be ordered regarding priority.

## Experiences of Using PPOOA and PPOOA\_AP

---

- Diverse systems were developed using PPOOA and PPOOA\_AP architecting approach
  - SCADA (Supervisory Control and Data Acquisition) System developed at UPM (Spain).
  - Underwater Autonomous Robot developed by Qinetiq (UK) and presented at ICSSEA 2002
  - Space System developed by ARTAL (France) as part of CARTS, 5<sup>th</sup> Framework Programme, IST funded project (IST-1999-2068).
  - Airbus A400, some avionics functionalities developed by CASA-EADS (Spain) (Since May 2005)

# Implementing PPOOA in Microsoft Visio

## **Antecedents:**

The PPOOA profile was implemented in a specific prototype CASE tool as part of the research work of the CARTS project funded by the European Union, IST initiative. Developing mature functionalities for such prototype tools requires significant effort. As a consequence the architecting method seems immature or awkward to use in research tool prototypes. Such a problems can impede industrial adoption of the PPOOA software architecting method.

So, we chose to implement PPOOA in a commercial CASE tool. This effort take more than 3 years and 20000 LOC to implement PPOOA features on the top of Visio.

## Why use a commercial CASE tool?

---

- Commercial CASE tools already support UML notation
- CASE tools offer general functionality that is expected of visual languages
- Developers are already familiar with CASE tools

28 April 2006

©José Luis Fernández-Sánchez

31

Commercial CASE tools already support UML notation and offer general functionality that is expected of visual languages. Visual elements can be created and deleted; manipulated; connected; copied and pasted; and saved and loaded.

Developers are already familiar with commercial CASE tools. Thus a particular software development method as PPOOA can be crafted on the top of the existing general functionality of the commercial visual CASE tool.

We implemented PPOOA on the top of Microsoft Visio, considering the extension mechanisms of the tool, and the PPOOA metamodel previously developed as an extension of UML metamodel

## Microsoft Visio (I)

---

- Whether an engineer creates flowcharts, electrical schematics, facilities management drawings, network diagrams, architectural plans, software designs or maps to the company picnic, the Visio tool have the support he or she needs.

28 April 2006

©José Luis Fernández-Sánchez

32

We decided to choose Microsoft Visio because it is highly customizable and offers a robust user interface to build upon . Furthermore, it has a large user base and is commonly found in universities and industry. Visio can be easily customized for different domains as nicely illustrated by the applications that Visio already offers: UML and other software methods diagrams, network architectures diagrams, ER diagrams to model databases, electrical engineering diagrams, pipes and instruments diagrams for industrial processes etc.

## Microsoft Visio (II)

---

- A Visio solution is a combination of Visio shapes and programs that model the real world and solve specific modelling problems
- A Visio solution usually includes:
  - **Stencils** of master shapes
  - **Templates** that provide stencils of specific shapes

28 April 2006

©José Luis Fernández-Sánchez

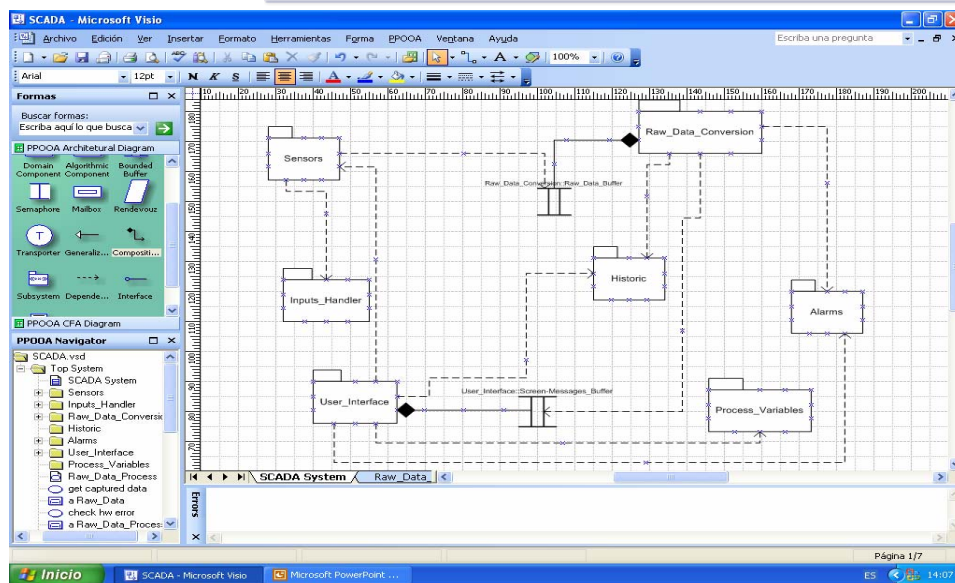
33

A Visio customized application for a specific domain can, for example, offers customized stencils that contain the visual elements of the domain, additional toolbars, accelerators and menus, additional menu entries in a visual element's context menu, custom properties for visual elements, windows that contain hierarchical views ("model explorer"), domain-specific error messages, and help features as an extension to the Visio help systems.



# PPOOA - Visio

## ( Modelling a System Architecture Diagram)



28 April 2006

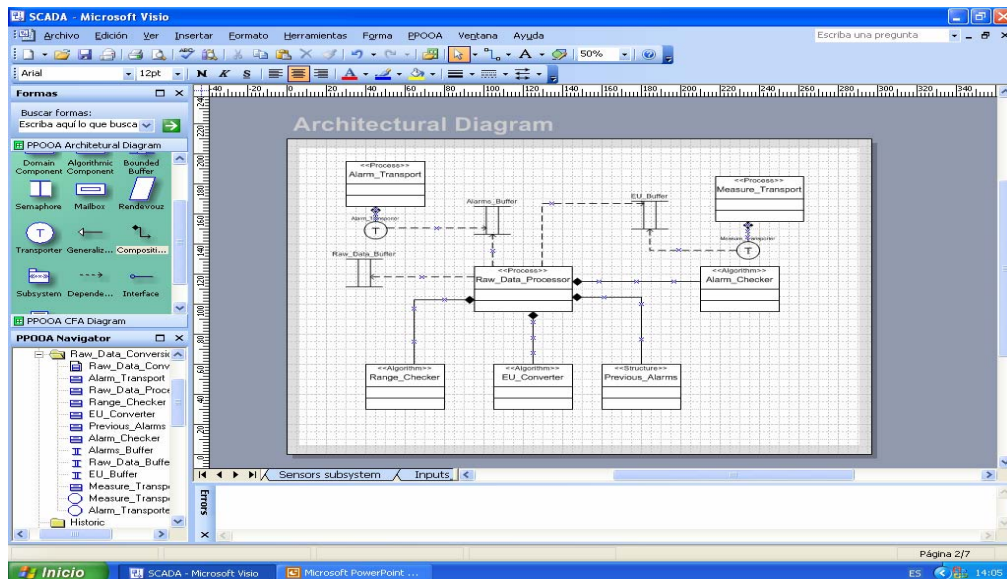
©José Luis Fernández-Sánchez

35

An example of a PPOOA-Visio window is shown in the slide above . It represents an architecture diagram for the data acquisition system architected using PPOOA-UML building elements. For PPOOA we could reuse several shapes (masters in Visio terminology) that were already part of Visio UML template. Other shapes as coordination mechanisms (semaphore, bounded buffer, transporter and rendezvous) are specific of PPOOA building elements. In the system architecture diagram, subsystems and buffers are independently tagged, and their attributes defined and stored in the tool database.

# PPOOA - Visio

## ( Modelling a Subsystem Architecture Diagram)



28 April 2006

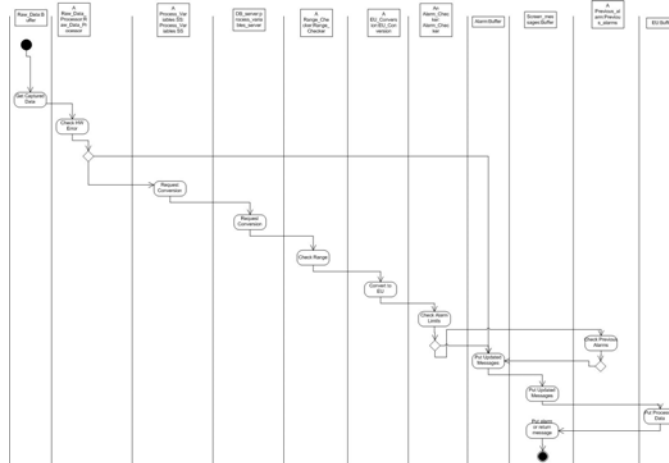
©José Luis Fernández-Sánchez

36

An example of a PPOOA-Visio window is shown in slide above. It represents the architecture diagram for the "raw\_data\_conversion" subsystem architected using PPOOA-UML building elements.

# PPOOA – Visio

## (Modelling a system response, “CFA Diagram”)



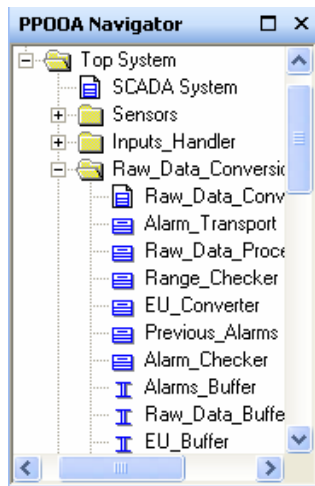
28 April 2006

©José Luis Fernández-Sánchez

37

The PPOOA dynamic view supports the "Causal Flow of Activities" representation. A CFA represents a system internal view of the activities performed by one system response to an event. The building elements of a CFA are: Triggering event, activity(ies) and continuation elements. These elements are described in the PPOOA metamodel. For PPOOA dynamic view, we use the UML activity diagram notation extended to implement CFA building elements and PPOOA constraints. The slide above, is an example describing the response that models the activities of the “processing of raw data” to convert it to engineering units and check its alarm condition. Different instances of the components represented in the architecture diagram can participate in the CFA. In a complex and large system we need several CFAs to model the complete system behavior.

## PPOOA-Visio Navigator



- Works similar to the Visio UML Navigator
- Shows the “PPOOA building elements” used in the document (“System architecture”)
- Handles the Visio tool events.

28 April 2006

©José Luis Fernández-Sánchez

38

PPOOA Navigator is used to see the entire structure of a diagram or to jump to a specific building element (component or coordination mechanism).

This window contains the references to architecture diagrams or views of the developed system, and the instances of the PPOOA building elements used in this particular software architecture.

## PPOOA-Visio (Tool Features)

---

- Automatic generation of documentation
- Automatic checking of PPOOA building guidelines
- Contextual help

## Automatic generation of architecture documentation

---

- The PPOOA Visio Tool **generates automatically the documentation** of the system architecture developed
  - Description of each building element, its domain attributes, interface and real time attributes
  - Generated in HTML format (easy to convert to Microsoft Word and Adobe Acrobat pdf)

## Example of automatic generation of architecture documentation

<b>Component Name</b>	<b>Raw_Data_Processor</b>
<b>Component Type</b>	Periodic Process
<b>Responsibilities</b>	This component is responsible of alarms status checking and alarms creation
<b>Activities</b>	<ol style="list-style-type: none"> <li>1. get captured data from buffer</li> <li>2. send query to process variables SS to obtain conversion function, range limits and alarm limits</li> <li>3. check range limits</li> <li>4. send alarm violation message or alarm return to alarm buffer</li> <li>5. conver data to engineering units</li> <li>6. check alarm limits</li> <li>7. send alarm message or alarm return to alarm buffer</li> <li>8. send update message</li> <li>9. send the processed dataa to engineering units buffer</li> <li>10. send alarm message or alarm return to screen messages buffer</li> </ol>
<b>Interface provided</b>	Deactivate
<b>Collaborators</b>	Raw_Data_buffer, Engineering_Units_Buffer and Alarm_Buffer.
<b>Subcomponents</b>	Range_Verification,EU_Conversion,Alarm_Verification
<b>Real-Time attributes (depend on the kind of component used)</b>	Period = 50 miliseconds
<b>Open Issues</b>	None

28 April 2006

©José Luis Fernández-Sánchez

41

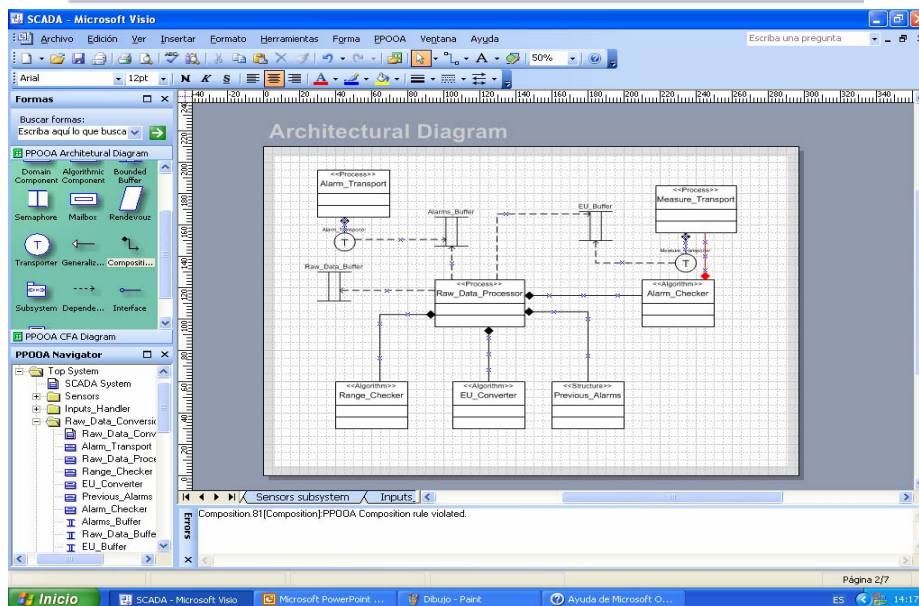
The PPOOA-Visio implementation can generate an architecture document of the designed architecture by a simple click of the mouse right button. This feature simplifies the tedious documentation writing activity of the software architect. The tool collects data from the PPOOA building elements represented in the diagrams, to automatically generate the documentation for each component, coordination mechanism and CFA.

## Automatic Checking of PPOOA building rules

---

- The PPOOA-Visio tool supports automatic **checking of PPOOA building guidelines**
  - Composition relations between two building elements
  - Dependency (“usage”) relations between two building elements

# Checking PPOOA Rules



28 April 2006

©José Luis Fernández-Sánchez

43

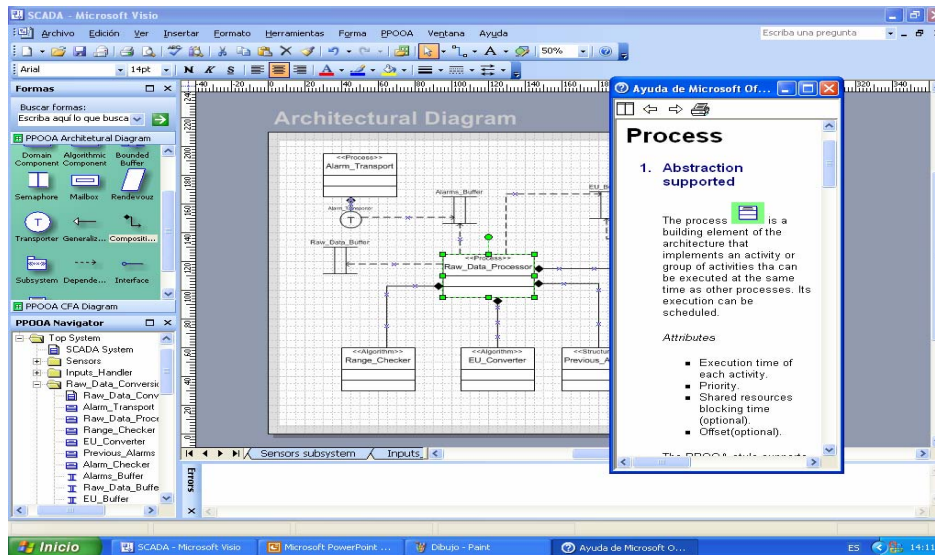
In the example there is a violation of a composition rule because an algorithm component can not contain a process component (red line and the error message: composition rule violated)

## PPOOA-Visio Help

---

- PPOOA architecting method PPOOA\_AP is implemented as part of the Visio help and in a contextual form
  - Architecting process steps
  - Description of the PPOOA building elements
  - Composition rules
  - Dependency rules
  - References to other documents and papers

# Contextual help



28 April 2006

©José Luis Fernández-Sánchez

45

The user can obtain help regarding the PPOOA building elements (here is the example of Process) or PPOOA composition and usage rules.

## Next tool features

---

- Implement performance engineering and “time response analysis” (RMA) capabilities
- Implement import and export mechanisms (XMI) to other CASE tools and Integrated Development Environments
- Improve automatic evaluation of models
- Create and implement a “developer assistant” that will help novice users to apply the PPOOA method and tool.

## To Conclude (I)

---

- **PPOOA is an architectural style that solves some [UML limitations](#).**
  - better support for CBD (Taxonomy of Components)
  - emphasizes coordination mechanisms usage
  - allows time responsiveness assessment ( by using CFAs)
- **PPOOA is complemented by an architecting process called PPOOA\_AP.**
  - major and minor steps to be followed
  - guidelines or strategies to be applied
- **PPOOA is already implemented in a well known commercial CASE tool ([Microsoft Visio 2003](#))**

## To Conclude (II)

---

PPOOA supports the **easy adoption by industry** of Real Time and Embedded Systems architecting techniques:

- A free trial version of PPOOA tool is offered by request
- Free publications and reports may be downloaded from ppoa web page ([www.ppoa.com.es](http://www.ppoa.com.es))
- On-site training dealing with UML and PPOOA may be negotiated per customer
- Maintenance of PPOOA tool may be negotiated per customer
- Consultancy dealing with PPOOA adoption (pilot project + next steps) may be negotiated per customer

## PPOOA publications

---

- "A Taxonomy of Coordination Mechanisms Used in Real-Time Software Based on Domain Analysis". CMU/SEI-93-TR-34. Software Engineering Institute. Carnegie Mellon University. December 1993.
- "A Taxonomy of Coordination Mechanisms Used by Real-Time Processes". ACM Ada Letters. March 1997.
- "An Architectural Style for Object-Oriented Real-Time Systems", 5th International Conference on Software Reuse. Victoria (Canada). IEEE 1998.
- "Architecture Modeling at Preliminary Design of Real-Time Systems". European Reuse Workshop. Madrid. European Software Institute, November 1998.
- "Extending UML for Real-Time Component Based Architectures", 14 th International Conference on Software and Systems Engineering (ICSSEA), Paris, December 2001.
- Chapter 12 of the book: "Business Component- Based Software Engineering", Frank Barbier (ed), Kluwer Academic Publishers, Boston, October 2002
- "A Process for Architecting Real-Time Systems", 15 th International Conference on Software and Systems Engineering (ICSSEA), Paris, December 2002.
- "Implementing A Real Time Architecting Method in a Commercial CASE Tool", 16 th International Conference on Software and Systems Engineering (ICSSEA), Paris, December 2003.
- "Supporting Functional Allocation in Component-Based Architectures", 18 th International Conference on Software and Systems Engineering (ICSSEA), Paris, December 2005.

28 April 2006

©José Luis Fernández-Sánchez

49

## Contact us

---

- [www.ppooa.com.es](http://www.ppooa.com.es)
- [ppooa\\_visio@telefonica.net](mailto:ppooa_visio@telefonica.net)
- José L. Fernández  
C/ Federico Moreno Torroba 11  
28007 Madrid (Spain)  
phone/fax: 34 915011202