Implementing a Real-Time Architecting Method in a Commercial CASE Tool

José L. Fernández-Sánchez

Escuela Técnica Superior de Ingenieros Industriales

C/ José Gutiérrez Abascal 2

28006 Madrid - Spain

Phone: 34 91 3363146

e-mail: jlfdez@ingor.etsii.upm.es

Juan C. Martínez-Charro

Escuela Técnica Superior de Ingenieros Industriales

C/ José Gutiérrez Abascal 2

28006 Madrid - Spain

e-mail: jcmartinez@alum.etsii.upm.es

Abstract: Late nineties, the first author proposed a new style for concurrent object oriented architectures. The style can be used in object-oriented systems when individual paths of execution are required to be concurrent and multiple processes may be positioned along the path to control the action. Therefore each process controls a segment of the path sequentially. So he called the style PPOOA, "Pipelines of Processes in Object Oriented Architectures" [1].

PPOOA promotes that architectural decisions related to concurrency in real-time systems should be made at the preliminary design phase.

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.

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 illustrate the approach with the implementation of PPOOA on the top of Microsoft Visio, considering the extension mechanisms of the tool, and the PPOOA metamodel previously created as an extension of UML metamodel [2].

Keywords: Methodologies, CASE tools, software architecture, real-time systems and UML

1. INTRODUCTION

Late nineties, the first author proposed a new style for concurrent object oriented architectures. The style can be used in object-oriented systems when individual paths of execution are required to be concurrent and multiple processes may be positioned along the path to control the action. Therefore each process controls a segment of the path sequentially. So he called the style PPOOA, 'Pipelines of Processes in Object Oriented Architectures [1].

PPOOA promotes that architectural decisions related to concurrency in real-time systems should be made at the preliminary design phase.

Currently UML [5] is the standard notation for component based software development. 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.

The PPOOA profile is compatible with UML and improves its use in real-time architectures. The UML stereotypes are extended with the elements of the style (periodic and aperiodic processes, controller object, and coordination mechanisms).

The modeling of event responses with enhanced UML activity diagrams is also critical for the analysis of the time responsiveness in real-time systems. This representation allows a seamless application of real-time behavior analysis techniques such as Rate Monotonic Analysis (RMA) [3]

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 a commercial visual CASE tool.

We illustrate the approach with the implementation of PPOOA on the top of Microsoft Visio, considering the extension mechanisms of the tool, and the PPOOA metamodel previously created as an extension of UML metamodel [2]. The PPOOA profile is compatible with UML and improves its use in real-time architectures. The UML stereotypes are extended with the elements of the style (periodic and aperiodic processes, controller object, and coordination mechanisms). UML Activity diagrams are also adapted for PPOOA requirements, specifically modeling resources and considering scheduling points.

The paper is organized as follows: Section 2 presents the main characteristics of PPOOA architectural style for real-time component based systems. Section 3 describes briefly Microsoft Visio, the commercial CASE tool used. The Visio extension mechanisms, that allow the implementation of new methodologies on the top of the tool, are presented. Section 4 describes the approach followed to implement PPOOA on the top of Visio. Some figures represent the look and feel of the diagrams supported. Section 5 gives the principal conclusions of this work and its perspectives.

2. PPOOA, AN ARCHITECTURAL STYLE FOR REAL-TIME SYSTEMS

A real-time system is one in which correctness depends on meeting time constraints. In these systems, correctness arguments must reason about functional requirements and response time requirements.

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.

Object oriented methodologies and early architectural decomposition efforts traditionally focus on domain analysis and functional cohesion to derive software components.

The experiences in defining time constrained architectures, let us to consider that timing concerns must play an important role in the choice of system components and objects. Concurrency modelling and its accompanying coordination (synchronization and/or communication) behaviour become a dominant concern surprisingly early in the architecture development process whenever response time is a critical factor.

Pipelines of processes can be used in object oriented systems when individual paths of execution are required to be concurrent, and multiple processes may be positioned along the path to control the action (Figure 1). Therefore each process controls a segment of the path sequentially. An example of this architecture style is PPOOA, "Pipelines of Processes in Object Oriented Architectures" [1].

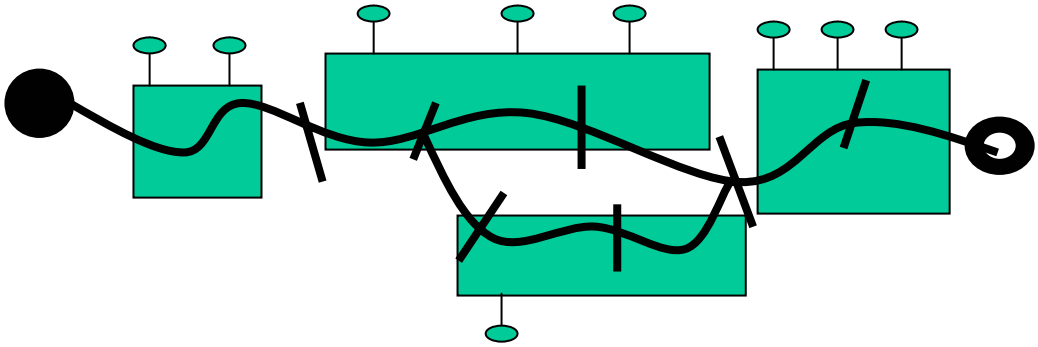

Figure 1. Pipelines of Processes

PPOOA promotes that architectural decisions related to concurrency in real-time systems should be made at the preliminary design or architecture phase of the development cycle.

Currently UML is the standard notation for object oriented development. 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.

The PPOOA profile is compatible with UML and improves its use in real-time architectures. The UML stereotypes are extended with the elements of the style (periodic and aperiodic processes, controller object, and coordination mechanisms). Activity diagrams are also adapted for Rate Monotonic Analysis (RMA) [3] requirements, specifically modeling resources and considering scheduling points. This adaptation allows seamless application of time responsiveness assessment of the software architecture represented in PPOOA.

We describe a software architecture using two views; one is a static representation and the other, the dynamic view of the system, represented by the modeling of the responses to external, timer or internal events.

The PPOOA architecture diagram is used instead of the UML component diagram to describe the static view of the architecture. The PPOOA architecture diagram focuses on "design or conceptual components" representation and the composition and usage relationships between them. PPOOA coordination mechanisms are also represented.

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 [2]. For PPOOA dynamic view, we recommend the UML activity diagram notation extended to implement CFA building elements.

3. VISIO, AN ADAPTABLE DESIGN TOOL

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.

We decided to choose Microsoft Visio because it is highly customizable and offers a robust user interface to build upon [4]. Furthermore, it has a large user base and is commonly found in 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.

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.

The Visio object model represents the objects, properties, methods and events that the Visio engine exposes through automation. Most objects in the model correspond to items, the user can see and select in the Visio user interface. Typical objects in the Visio model are: application/global object, document object, page object, master object, selection object, shape object and window object. For example, a shape object can represent anything on a Visio drawing page that the user can select with a pointer or an object from another application that is linked, embedded, or imported into a Visio drawing [4].

Some objects represent collections of other objects. For example, a Document object represents one open document in a Visio instance.

Visio exposes a Visual Basic API to access and analyze a document. All GUI elements (e.g., window, page, shape, and selection objects) are represented in the Visio object model [4].

A Visio event is an action or occurrence, often generated by a tool user. Events can result from actions such as opening or closing documents, dropping or deleting shapes on a drawing page, editing the text of shapes, and altering shape formulas. Events can be handled programmatically or by using Visio formulas.

4. IMPLEMENTING PPOOA IN VISIO

We followed an approach to implement PPOOA in Visio that is consistent with the tool builder recommendation:

- Identify PPOOA building elements currently not supported by Visio UML template.
- Develop and implement PPOOA´s shapes and stencils in Visio.
- Once we were satisfied with the initial set of shapes, we developed the drawing assistants or dialog boxes that the user might need to construct PPOOA diagrams (architecture diagrams and CFA´s).
- Finally, since PPOOA´s shapes and dialog boxes interact with the tool database we determined how to design shapes and set their attributes and behavior accordingly.

PPOOA Visio implementation was developed following the object-oriented paradigm to allow easy modifiability and evolution. We decided to use Microsoft Visual C++ as programming language. We used Microsoft Foundation Library to create the dialog boxes and windows we need to develop the PPOOA-Visio user interface.

The developing process was incremental. We developed two prototypes, we called them version 1 and version 2. The project team developed more than 8000 LOC.

PPOOA-Visio tool version 1 prototype was developed to validate the viability of using Visio to support PPOOA building elements and PPOOA-UML metamodel.

This version offers a template with two stencils: the architecture diagram stencil and the dynamic view stencil. The Visio library, including the dialog boxes the user needs to add PPOOA building information in the Visio document, is also supported.

Tool version 2 prototype is an enhancement of version 1 that has the following features:

- Drawing explorer window. Called here PPOOA Navigator (Figure 1). 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.
- Handling Visio events. In Visio, events can result from user actions such as opening or closing documents, dropping or deleting shapes on the drawing page, editing the text of shapes and altering shape attributes.
- Implementing PPOOA document structure. That is, pages representing architecture diagrams, and pages representing the dynamic view or responses model represented by the CFAs.

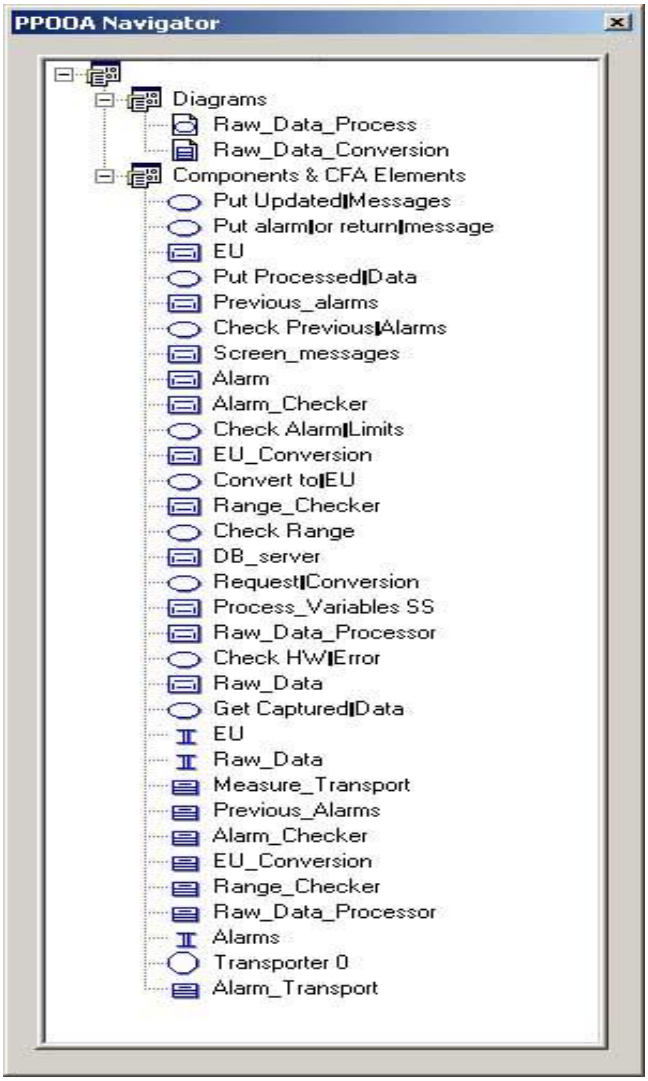

Figure 1. PPOOA Navigator Window

An example of a PPOOA-Visio window is shown in Figure 3. It represents an incomplete 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 architecture diagram, components and coordination mechanisms are independently tagged, and their attributes defined and stored in the tool database.

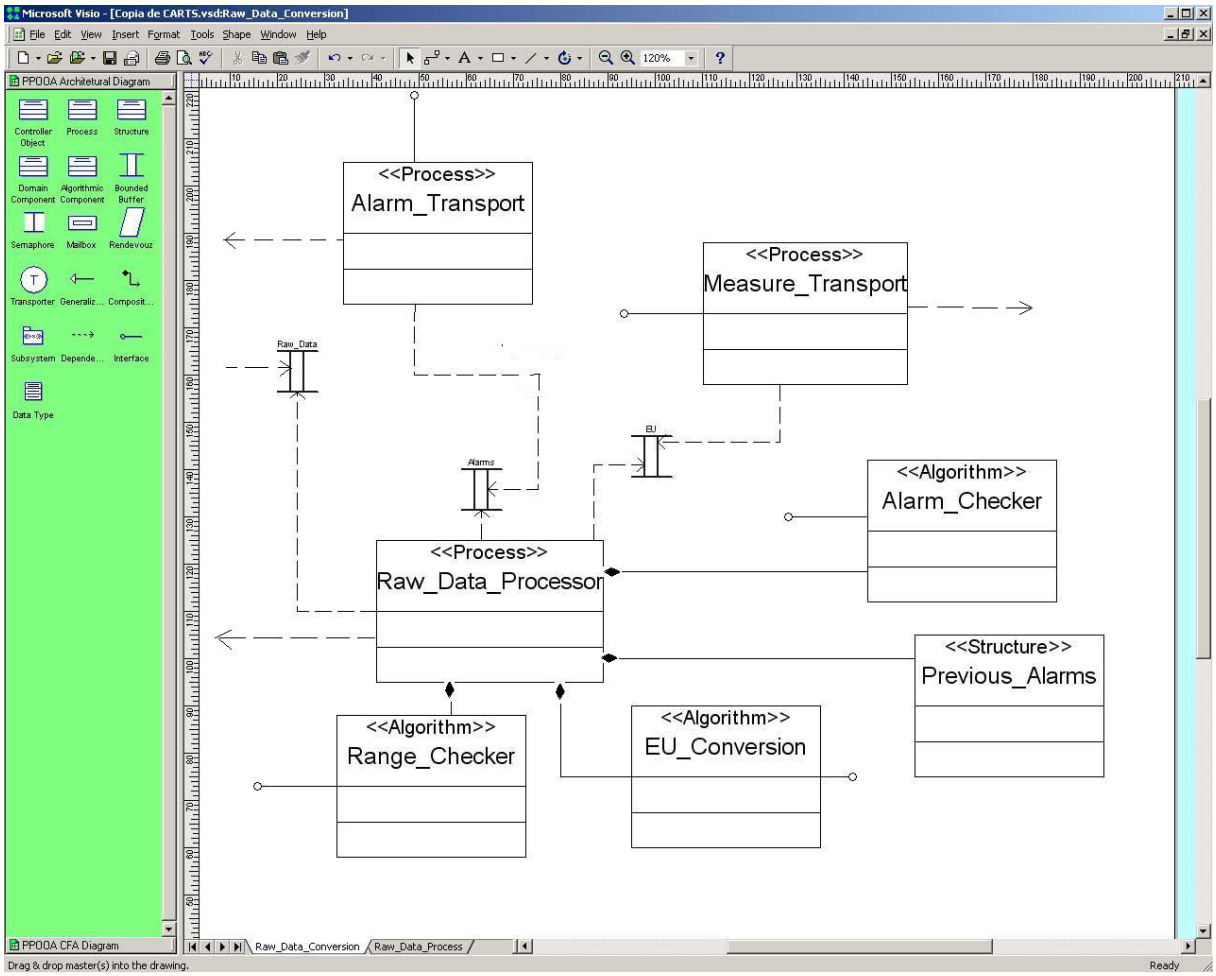

Figure 3. PPOOA Architecture Diagram

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 [2]. For PPOOA dynamic view, we recommend the UML activity diagram notation extended to implement CFA building elements and PPOOA constraints. Figure 4 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.

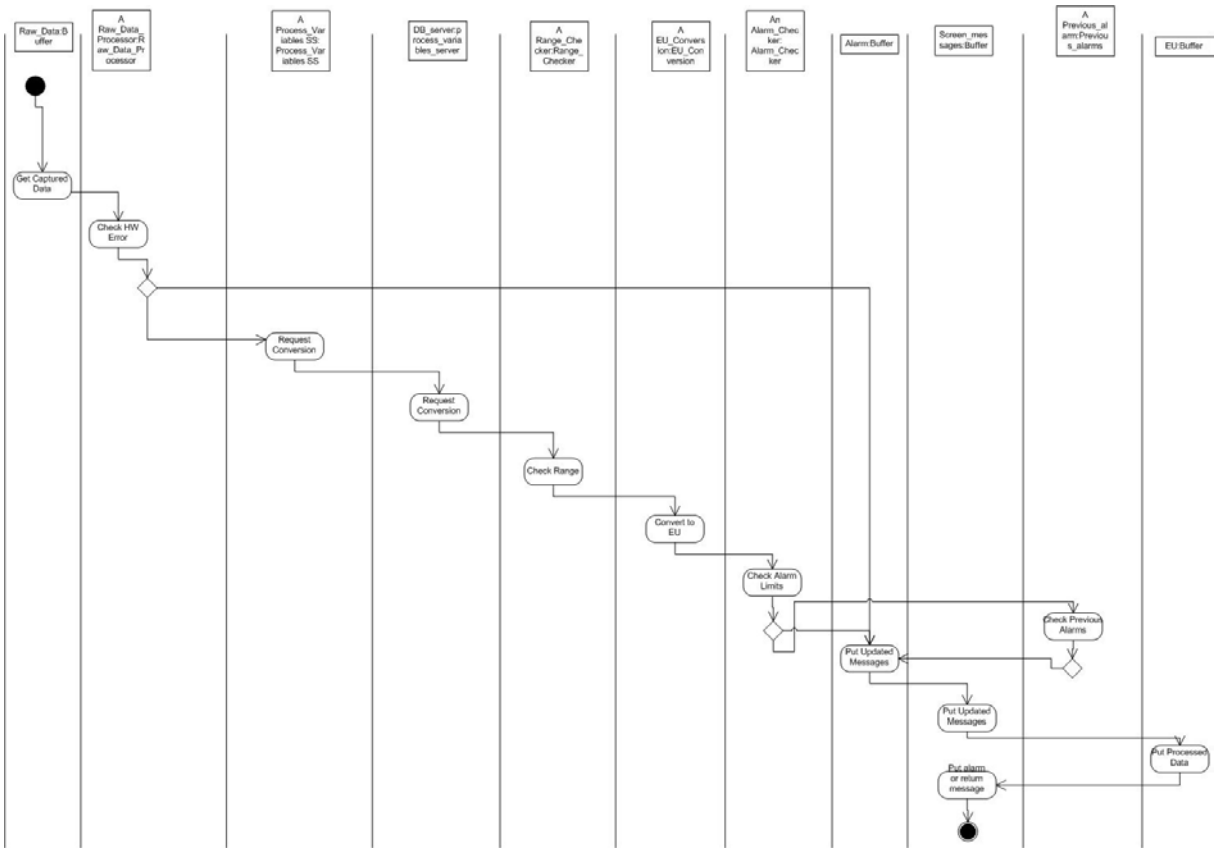

Figure 4. Raw Data Processing CFA

The PPOOA-Visio implementation can generate an architecture document of the designed architecture by a simple mouse right button click. 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 architecture document.

5. CONCLUSIONS AND PERSPECTIVES

Visio-PPOOA has many desirable features from the user and adoption point of view. Users will make likely architect real-time systems using visual tools that are integrated in an environment that they are familiar with.

Help Features and extensions will be added to the tool to instruct novice users about the PPOOA methodological issues, tactics and guidelines.

We are improving the PPOOA-Visio implementation to allow the automatic model checking. Typically, PPOOA components composition constraints, and usage dependencies between components and coordination mechanisms shall be checked to avoid incorrect static architecture models.

Future research is the integration of PPOOA Visio with a Rate Monotonic Analysis tool to allow automatic analysis of the time responsiveness properties of the designed real-time system architecture.

References

[1] Fernandez J.L. An Architectural Style for Object Oriented Real-Time Systems. Proceedings of the Fifth International Conference on Software Reuse. Victoria(Canada). IEEE. 1998.

[2] Fernandez-Sanchez, J.L. and Monzon, A. Extending UML for Real-Time Component-Based Architectures. 14th International Conference on Software and Systems Engineering and their Applications. Paris (France). December 2001.

[3] Klein, M. Ralya, T. Pollak, B. Obenza, R. and Gonzalez-Harbour, M. A Practitioner's Handbook for Real-Time Analysis. Guide to Rate Monotonic Analysis for Real-Time Systems. Kluwer Academic Publishers, Norwell, MA.1993.

[4] Microsoft. Developing Microsoft Visio Solutions. Microsoft Press. Redmond, WA.2001.

[5] Object Management Group. Unified Modeling Language. Standard Version 1.5.March 2003.