Supporting Functional Allocation in Component-Based Architectures

José L. Fernández Sánchez and Eduardo Esteban Betegón

E.T.S. Ingenieros Industriales
C/José Gutierrez Abascal 2
28006 Madrid (Spain)
+34 913363145

fernandezjl@acm.org, eaesteban@yahoo.es

ABSTRACT

The systems engineering community describes the systems engineering process as an iterative approach to technical management, acquisition and supply, system design, product realization, and technical evaluation at each level of the system, beginning at the top and propagating them through a series of steps which eventually lead to a preferred system solution [8].

There is a consensus defining the systems engineering approach as a complex mix of processes dealing with management and technical issues.

For large systems, systems engineers are primarily concerned with the functional and system architectures that lead to the detailed design and implementation of large systems of all types.

The goal of functional analysis is to create a functional architecture that can be the foundation for defining the system architecture through the allocation of functions to the system architecture components.

Systems architecture is developed during the system synthesis phase of the systems engineering process.

Large systems are often made up of subsystems that are logically and physically partitioned into parts or components. Aggregation allows us to consider the thing as a unit, ignoring its parts, a simplification in thought. Alternatively, hierarchy allows us to consider a component as an assembly of parts to think how it is built.

It is recognised that functional allocation relates to the system engineering approach segregating form from function [6]. This approach requires independent models of function (functional) and form (structure), and a deliberate mapping between elements in both models.

It is known that functional allocation does not support a strict object oriented paradigm, nor is always even desirable for object-oriented systems. But large and complex system engineering problems have proven that segregation of form and function is a valuable approach.

This paper offers a modeling solution named PPOOA (Pipelines of Processes in Object Oriented architectures) extending UML notation. PPOOA is implemented in a CASE tool developed for supporting functional allocation to the system architecture components. PPOOA tool is currently being applied in some aerospace real-time systems architecting process.

Keywords

Systems engineering, systems architecture, CASE tools, functional allocation.

1. INTRODUCTION

Behavior of a system involves one or more actions taking place. These actions are various called responses, operations, functions, or activities. In a more standard way a function is described as a transformation of inputs to outputs that may include the creation, monitoring, modification or destruction of elements, or a null transformation [6].

The structural view of the system must be consistent with the behavioral view of it. Allocation is the design relationship, which maps one set of behavior elements to another set of structural elements (architecture building blocks).

Allocation is the way for achieving the functional completeness of the system and the requirements fulfillment.

Currently, object oriented software development methodologies, more oriented to a conceptual paradigm than a functional decomposition, do not solve completely the functional allocation problem and thus, the CASE tools supporting these methodologies do not implement the functional allocation relationship adequately.

In this paper we propose a solution for functional allocation of system responses activities to system components. An architecting process and the PPOOA-Visio CASE tool we developed support this solution. The solution presented is consistent with UML[7] and SysML[6] standards.

The structure of the paper is as follows: Firstly, in section 2 we describe briefly the systems engineering process and when in the lifecycle, functional allocation is performed. In Section 3, we describe how functional allocation is specified in UML and SysML standards. Section 4 describes the functional allocation approach adopted in the PPOOA (Pipelines of Processes in Object Oriented Architectures) architectural style for real-time systems [1]. Section 5 describes the implementation of the functional allocation as a capability of the PPOOA-Visio CASE tool. Finally, section 6 summarizes our findings and tool users feedback.

2. THE SYSTEMS ENGINEERING PROCESS

The systems engineering community describes the systems engineering process as an iterative approach to technical management, acquisition and supply, system design, product realization, and technical evaluation at each level of the system, beginning at the top and propagating them through a series of steps which eventually lead to a preferred system solution [8]. So the systems engineering approach is a complex mix of processes dealing with management and technical issues.

The system development process will necessarily include three logical steps as described by Sage [5]:

- Formulation of the problem under consideration

- Analysis to determine the impact of alternatives

- Interpretation of this impact in accordance with the system goals and quality criteria established by the decision maker, and selection of an appropriate plan of action to continue the effort.

The expansion of these three basic steps depends on the lifecycle adopted and the system engineering standards followed. For the sake of understandability we expand the previous three logical steps process in the seven steps process represented in Figure 1 and based on a well-known process first established by Hall [5].

1. This step involves the establishment of the needs, requirements and constraints.

2. This step involves the selection of objectives and quality criteria that guides the search for alternatives.

3. This step involves the modeling of solution alternatives (architectures) with sufficient detail to permit analysis of the impacts of implementation and subsequent assessment with respect the objectives and quality criteria.

An architecture is chosen to implement the functions and satisfy the requirements and constraints. This step can be viewed as a search through a highly non-linear design space of very large dimension [8].

4. This step involves the evaluation of consequences of the alternatives (architectures) that allocating the specified functional requirements are relevant to the issue under consideration by the value system established in step 2. There are a variety of simulation and quantitative methods that are of potential value here.

5. This step involves the selection by the engineers of the system solution parameters that best meet system objectives satisfying requirements and constraints.

6. This step involves evaluating impacts of the alternatives and interprets the alternatives in terms to which they meet system goals.

7. This step involves communicating the results and scheduling subsequent efforts.

1. Problem Definition

2. Selection of goals and criteria

3. System synthesis

4.Analysis of alternatives

5.Refinement of alternatives

6.Decision making
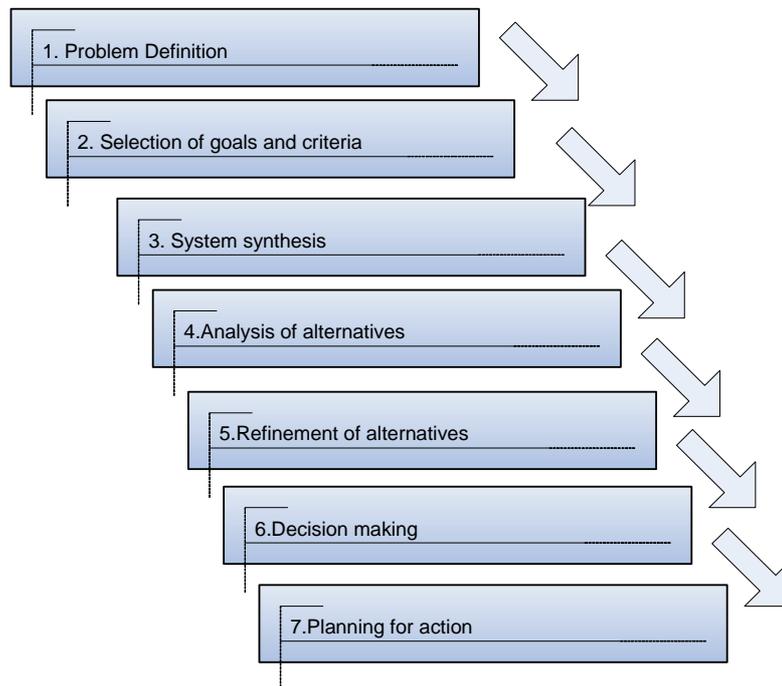
7.Planning for action

Figure 1. Seven steps of systems engineering

The approach proposed in this paper is especially useful for the system synthesis step, since it allows modeling architectural solutions with two views, one structural and the other for behavior.

The models used in the views support functional allocation to the system components and make easy the evaluation of alternatives based on efficiency criteria that are of paramount importance for real-time systems.

3. BEHAVIOR ALLOCATION IN VISUAL MODELING LANGUAGES
3.1 UML

The Unified Modeling Language (UML) is an industry standard managed by the Object Management Group. UML provides a notation and semantics for modeling, constructing and documenting software intensive systems.

UML is based on the object-oriented paradigm for software development. A primary goal of UML was to put an end to the variety of object oriented development methods within the object oriented community used in the earlier nineties. UML has evolved since its first version produced in 1997. The current version of UML is UML 2.0. Tool builders are adapting the CASE tools to meet the UML 2.0 requirements.

It is not the purpose of the paper to describe UML 2.0. The reader can consult the standard and related literature for that purpose [7].

Here we will describe the solution given in UML 2.0 for the functional allocation problem, that is the focal point of the paper.

The term allocation is used in UML 2.0 to describe the assignment of artifacts to nodes in the deployment diagram according to the deployment defined between them [7].

Additionally, the engineer can use activity partitions, grouping activities with respect the system-building element that is responsible for them or where they reside in the system. This approach is useful but incomplete to solve the functional allocation issue, since it is unidirectional, depending on the CASE tool implementation, that a system component knows the activities it implements.

3.2 SysML

The multiplicity of building elements of a complex system is greater than those pertaining to a software system. This multiplicity of building elements and concerns requires diverse views to represent system developing aspects.

A good set of system views should be complete (cover all aspects of the system) and orthogonal (capture different pieces of information). Rechtin[4] proposes a set of views to be used in systems architectures, represented in Figure 2.
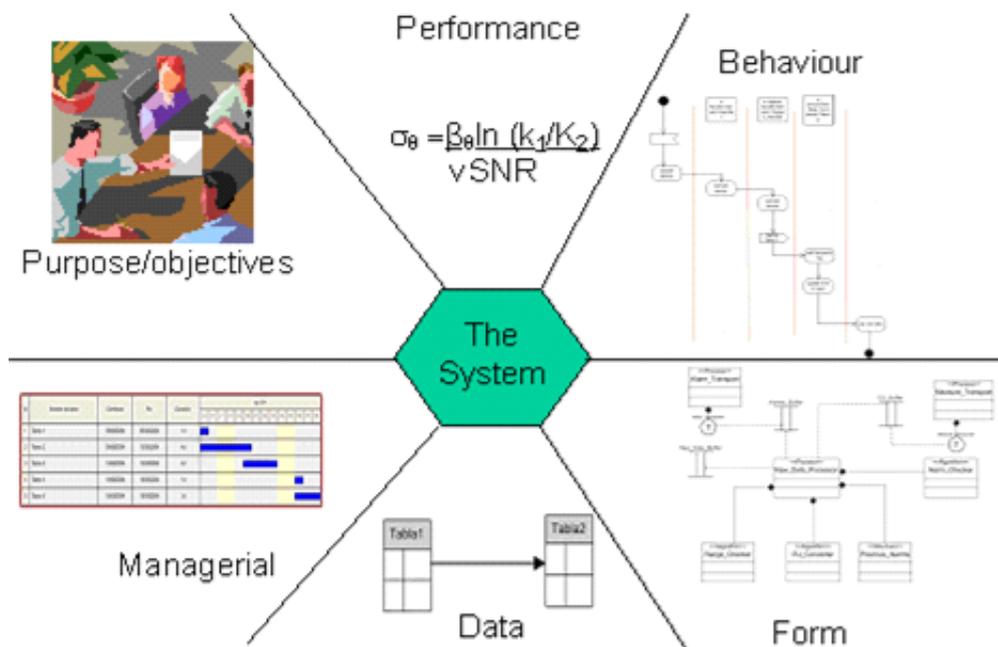

Figure 2. Schematic representation of the six views of system architecture

The systems engineering community still lacks a standard modeling language that may be used for describing the different system views, and that is widely accepted and used.

SysML [6] is a general purpose modeling language for systems engineering applications. The origins of SysML initiative can be traced to a decision by the International Council on Systems Engineering´s Model Driven Systems Design workgroup in January 2001 to customize UML for systems engineering applications. SysML is defined as a UML 2.0 profile.

SysML is intended to support modeling of a broad range of systems, which may include physical equipment, software, data, personnel, procedures and facilities.

Diagrams supported in SysML are:

- Use cases diagrams describing system usage.

- Class diagram defining items of interest and relationships between them.

- Assembly diagrams describing the system as a collection of parts, which fill specific roles within a large whole.

- Parametric diagram providing mechanisms for integrating engineering analysis with the system assemblies. Parametric constraints depict a network of constraints among properties of a system.

- Activity Diagram that is intended to be able to represent what is expressed by the Enhanced Functional Flow Block Diagram (EFFBD).

- Interaction diagrams include the sequence diagram, interaction overview diagram and timing diagram, and are unchanged from UML 2.0, except for some minor notational changes for timing diagrams.

- State machines used for modeling discrete behavior through finite-state transition diagrams.

Due to its goal of supporting systems engineering, SysML deals specifically with allocation. SysML gives an overview of the types of allocation supported: requirements allocation, functional allocation, flow allocation, structural allocation, deployment allocation and property allocation. SysML requires a generalization of allocation which is not possible with UML 2.0 deploy relationship.

Allocation is used to relate model elements in different kinds of diagrams. SysML supports allocation of both of the generic definition of a model element, and of the specific usage of a model element. Class diagrams depict the definition of model elements whereas activity and assembly diagrams depict the usage of model elements.

SysML specifies that functional allocation and flow allocation may be depicted together using activity diagrams with activity partitions (swimlanes) and allocation reference. Also, allocation may be shown in tabular views.

SysML is in the process of approval and adoption by tool builders and industry.

4. SUPPORTING FUNCTIONAL ALLOCATION IN PPOOA

PPOOA, Pipelines of Processes in Object oriented Architectures, is a software architecture style for concurrent real-time systems. PPOOA proposes a vocabulary of components and coordination mechanisms to be used in the architectural design of software intensive systems [1].

UML metamodel was extended to include PPOOA building elements: processes, coordination mechanisms, domain components and others [2].

The PPOOA architecting process, named also PPOOA_AP, is part of the general system development process. It takes some inputs from the analysis phase and produces some outputs to the detailed design phase. PPOOA architecting process may be applied in either a waterfall development process or iterative development process.

PPOOA describes an architecture using two views; one is a structural representation and the other, the behavior 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 structural view of the architecture. The system architecture diagram focuses on "design or conceptual components" representation and the composition and usage relationships between them. Coordination mechanisms used as connectors, are also represented. Figure 3 describes the Planner subsystem of an industrial robot using PPOOA, architectural diagram. Additionally to the subsystem components represented as processes, a semaphore for protecting access to feedback data, and messages queues are also depicted.
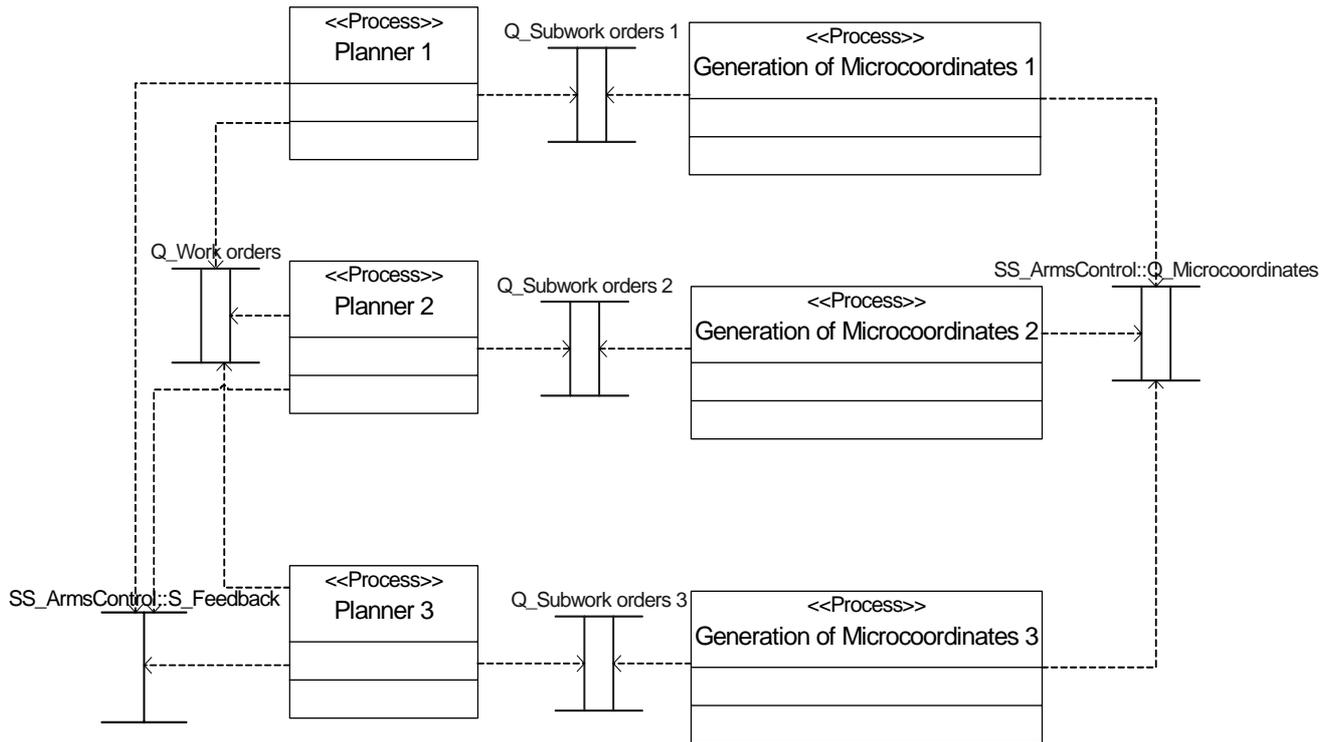
Figure 3. Planner Subsystem

The system behavior view is supported by the "Causal Flow of Activities (CFA)" visual representation. A CFA represents a system internal view of the flow of activities performed by the 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 the system behavioral view, PPOOA uses the UML activity diagram with its notation extended to implement CFA building elements.

Figure 4 represents the CFA "Create Subwork Order" describing that each plan produced by the Planner process (see Figure 3), results in a sequence of subwork orders that are asynchronously passed to the Generation of microcoordinates process.

Figure 4 shows that functional allocation is solved by using UML activity diagrams with activities in partitions (swimlanes), and consequently, naming the partitions with the system or subsystem building elements, that allocate the activity. In the case that

different instances of the same system component allocate diverse activities these instances are represented in different partitions.

In the next section we will describe the implementation solution we developed to support functional allocation in the PPOOA CASE tool. In the case, the systems engineer is not able to use the PPOOA CASE tool. He or she can represent functional allocation in a tabular form.

The representation of system functionality using CFAs for modeling the system responses to events is quite useful for applying schedulability analysis techniques, such as Rate Monotonic Analysis [3]. RMA allows the analysis of systems responses to determine the fulfillment of temporal requirements. CFAs may be considered as inputs for the RMA process that may be done manually or supported by specific tools.

5. PPOOA-VISIO CASE TOOL

We implemented PPOOA on the of Microsoft Visio, considering the extension mechanisms of the tool, and the PPOOA metamodel previously created as an extension of the UML metamodel [2].

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 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, etc.

A Visio customized application such as PPOOA, offers:

- Customized stencils that contain the visual elements of PPOOA
- Additional toolbars, accelerators and menus
- Additional menu entries in a visual element's context menu
- Real-time properties for PPOOA building elements
- Navigator window that represents the hierarchical views of the repository of the system architecture building elements
- PPOOA specific error messages (violation of PPOOA building rules)
- And the PPOOA help as an extension to the Visio help.

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.
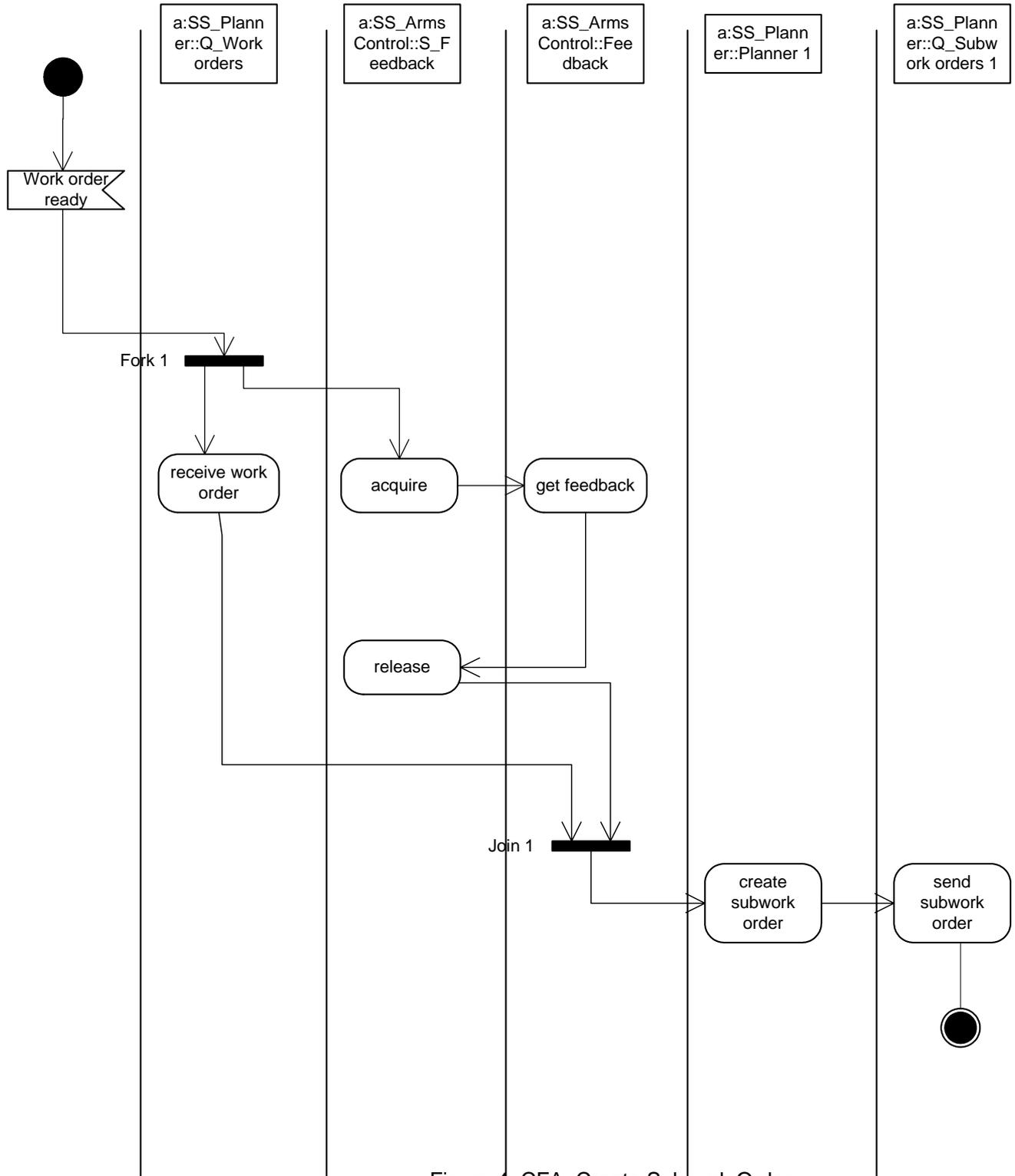
Figure 4. CFA, Create Subwork Order

5.1 Functional Allocation Implementation

Activities allocation to system components is one relevant feature of PPOOA-Visio tool not found in other CASE tools

When the system architect drags an activity to a CFA drawing page, it triggers the dialog box that can be seen in Figure 5. In this dialog the system architect can introduce the component that will implement this activity, its instance and the estimated or budgeted execution time of the activity. Additionally, a more detailed description of the activity is possible.

If the system architect opens the "Properties" dialog box of a component or coordination mechanism, he or she can see the activities the component allocates and their respective execution time (if the architect estimated it) in the Real-Time Attributes tab.

Figure 6 shows the tool window that allows system architect to check traceability among system responses (CFA) activities and the system components that should implement them.

To support this allocation we implement a bidirectional relationship between activity and component in the tool database. When a new activity is added to a CFA or system response, a new object is created in the tool database. The user ("systems engineer") then selects the component to which the activity is allocated. The tool extracts the component unique identifier (tool implementation dependant), and associates it to the activity. Additionally, one of the attributes stored with the activity is the activity-estimated execution time introduced by the systems enginneer.

The steps implemented in the tool for functional allocation can be described more precisely as follows:

1. The tool user (systems architect) creates a new system response activity and allocates it to a system component already defined in the architecture
2. The tool obtains the UID (Unique Identifier) of the selected system component
3. The UID of the selected component is stored in the tool database element representing the allocated activity
4. The UID of the allocated activity is stored in the tool database element representing the selected system component.

Figure 7 represents the sequence diagram in UML of the above steps, representing how functional allocation is implemented in the PPOOA-Visio CASE tool.

The implementation of functional allocation by PPOOA-Visio tool ensures that any system response activity and the system component that performs it, know each other. So when the tool generates the documentation of the developed architecture of a real-time system, allocation information is included in the documentation. Traceability of functional requirements to the system architecture components can be checked and verified.
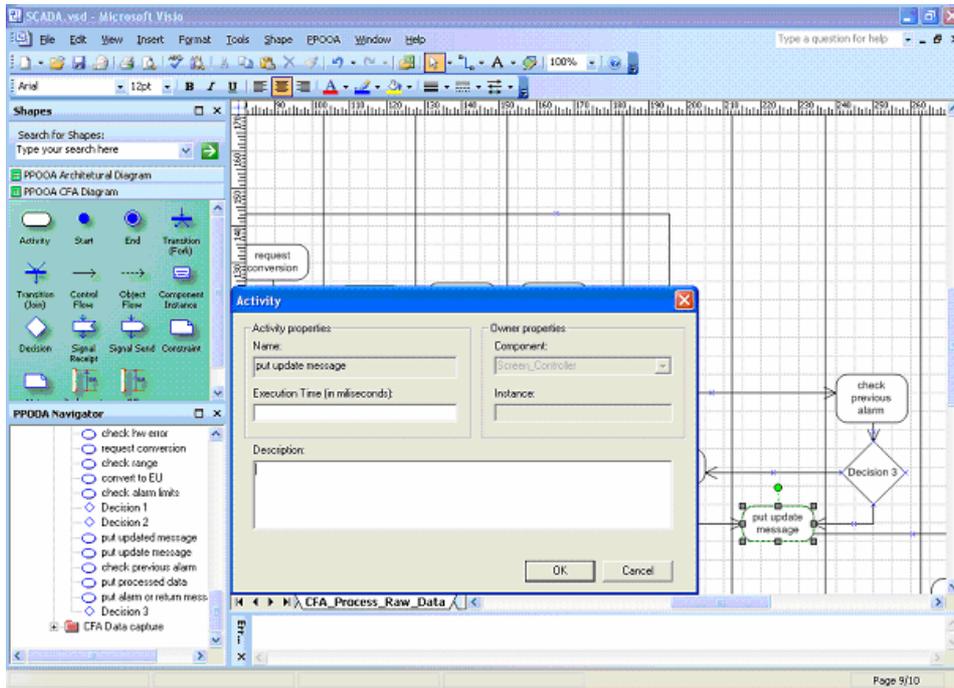
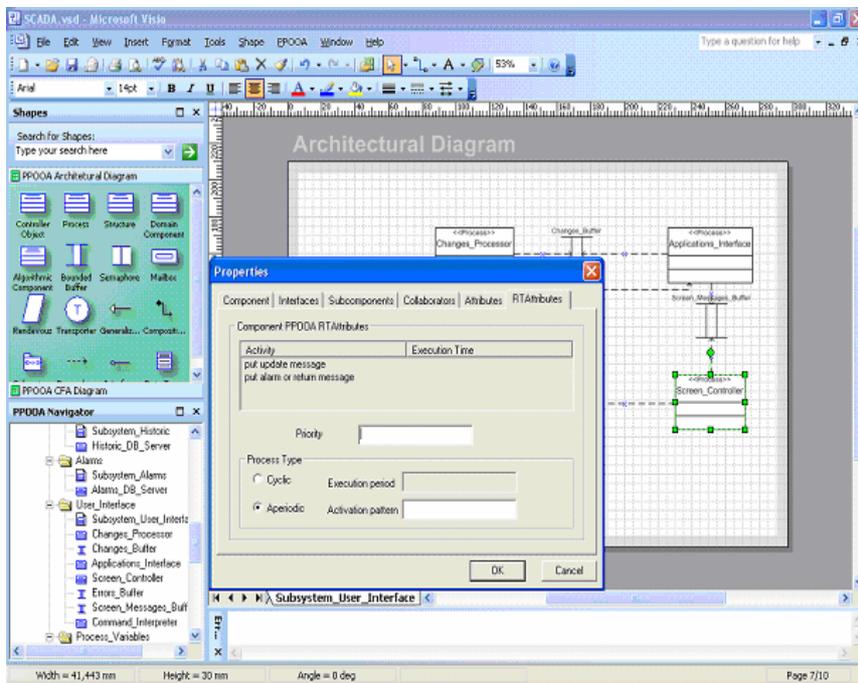Figure 5. Activity allocation to component in PPOOA-Visio

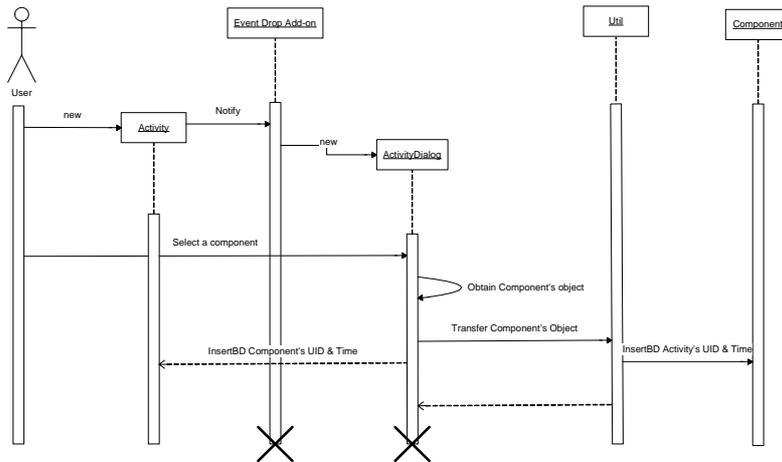Figure 6. A system component and the activities it allocates

Figure 7. Functional allocation implementation in PPOOA-Visio tool

6. CONCLUSIONS

Functional allocation relates to the system engineering discipline separating structure from function. System engineering requires independent models of function and structure, and a deliberate mapping or trace between elements in each of these models. This mapping does not support a standard object oriented paradigm.

The development of large and complex systems engineering problems has proven, however that the segregation of structure and function is a valuable approach.

An implementation of the segregation of structure and function is offered in PPOOA architectural style for component based systems architectures. Structure is represented by architectural diagrams. Function is represented by CFAs or activity diagrams. Functional allocation is implemented by named partitions in activity diagrams. The implementation in PPOOA-Visio CASE tool allows a bidirectional functional allocation, so activities are allocated to components and components know the activities they allocate.

The main benefit experienced by tool users is the documentation of the functional traceability to the system components.

Future research considers the description of system responses and the activities of a CFA with a more precise semantics.

7. REFERENCES

[1]   Fernandez J.L.: An Architectural Style for Object-Oriented Real-Time Systems; 5th International Conference on Software Reuse. Victoria (Canada). IEEE 1998.

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

[3]   Klein, M.H., 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, Boston, MA, 1993.

[4]   Rechtin, E. and Maier, W.M.: The Art of Systems Architecting; CRC Press, Boca Raton, FL, 1997.

[5]   Sage A. P. and Armstrong J.E.: Introduction to Systems Engineering. John Willey & Sons, New York, 2000.

[6]   Sys Ml Partners: SysML Specification, V 0.9; Object Management Group, January 2005.

[7]   U2 Partners: Unified Modeling Language: Superstructure. V 2.0; Object Management Group, July 2003.

[8]   Technical Board: Systems Engineering Handbook. A What To Guide for all SE Practitioners; International Council on Systems Engineering., June 2004.