Robotics Controller System.

Architecture developed using the PPOOA architectural style and PPOOA_Visio tool based on UML notation.

José L. Fernández-Sánchez
Industrial Enginering School (ETSII)
Madrid Technical University (UPM)
Madrid-Spain
e-mail: fernandezjl@acm.org

Technical Report : PPOOA-TR-01_2005, January 2005

Abstract

This example is based on a real robotic platform developed by ABB Robotics [Hissam 04].

The modelling problem focuses on the interaction between processes in the robotics platform main computer. That computer is responsible for running programs (written in a specific robot programming language) that generate work orders. These orders are decomposed into subwork orders that ultimately result in communicating microcoordinates to an axis computer that contains the device drivers responsible for actual movement of the robotic arm (or arms).

The light processes or threads that execute on the main computer are a mix of periodic and aperiodic processes performing a mix of synchronous and asynchronous interprocess communication. Much of the asynchronous type of communication is conducted through First in First out (FIFO) queues.

We illustrate the strength of PPOOA architectural style [Fernandez 98] and PPOOA-Visio tool [Fernandez 03], to represent the problem from the point of view of the real-time concerns. The PPOOA architecting process [Fernandez 02a], allows the system architect to make decisions related to the time requirements of the responses and the resources used by the architecture building elements.

PPOOA building elements allow a complete representation of the static view of the architecture including processes, components and the queues used to communicate processes.

PPOOA offers the representation of a “system response” as an UML activity diagram named here CFA (Causal Flow of Activities). A CFA is decomposed in activities that can be allocated to the different components of the architecture making easy the systems engineering synthesis process. An event that can be external, internal or timer normally triggers a CFA.

For the example we represent the static diagrams for each subsystems and the CFAs representing the architecture dynamic view encompassing the system responses modelled in PPOOA-Visio tool.

The PPOOA-Visio CASE tool has generated part of the architecture description showed in this report.

The CFAs modelled may be the input to a schedulability assessment of the architecture using Rate Monotonic Analysis techniques [Klein 93].

Contents

1. Description of the problem scope

The model problem focuses on the interaction between processes in the robotics platform main computer. That computer is responsible for running programs (written in a specific robot programming language) that generate work orders. These orders are decomposed into subwork orders that ultimately result in communicating microcoordinates to an axis computer that contains the device drivers responsible for actual movement of the robotic arm (or arms) [Hissam 04].

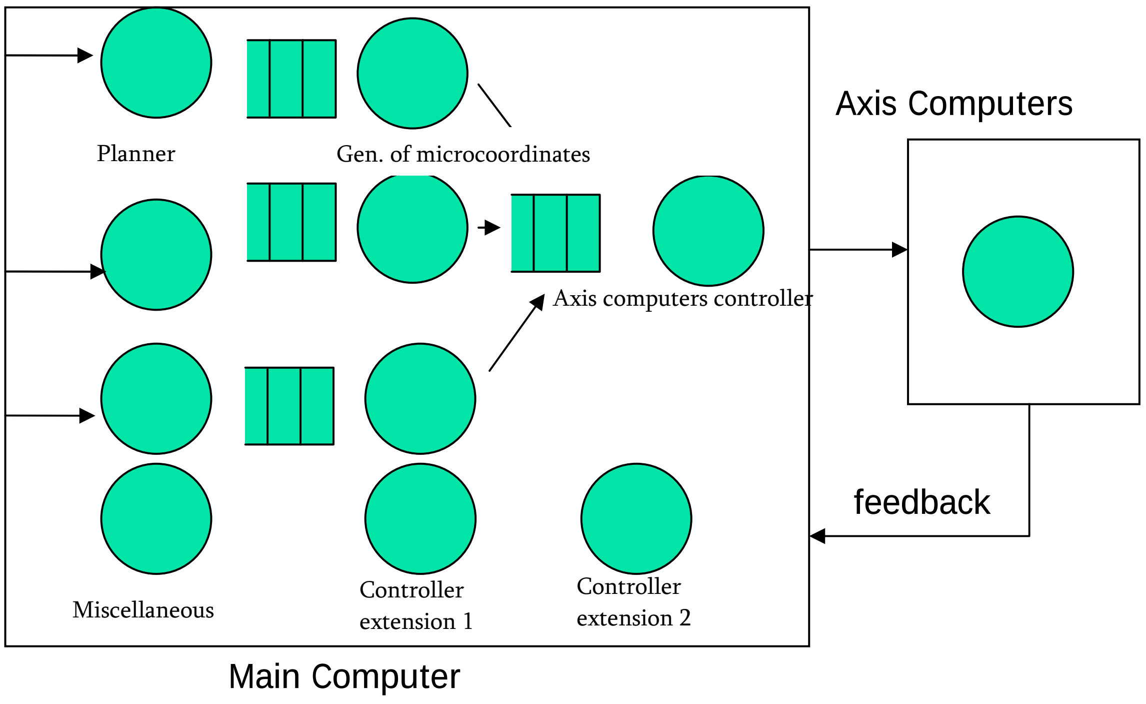


Figure 1. Processes on the Main Computer

There are six main light processes, each described below.

Planner. Three planner processes, carry out planning activities: primarily, they receive work orders and create plans. Each plan produced by the Planner process, results in a sequence of subwork orders that are asynchronously passed to the Generation of microcoordinates process. Typically, the queue size for subwork orders is set to 30 in the core controller platform.

When Planner process produces subwork orders and attempts to place one into a full queue, Planner becomes blocked until Generation of microcoordinates removes an item from that queue, thereby making room for the new subwork order. This system behavior is considered normal.

The axis computer sends feedback to Planner indirectly—that is, it delivers feedback first synchronously to Axis computers controller (described below) in the main computer, and then the Axis computers controller “publishes” it to those processes subscribed to that feedback (Process Planner being one of them). Because this feedback informs the planning process, planning can’t be done entirely in advance.

The arrival rates of work orders to Planner vary ranging from 10 Hz (i.e., one every 100 ms) to 15 Hz (i.e., one every 66.7 ms). For this model problem, we assume that the arrivals are describable via Exponential (75) (i.e., exponential random distribution with a mean of 75 ms).

The execution time of Planner—the time needed to decompose a work order into a sequence of subwork orders—is also highly variable. For this model problem, we assume that Planner’s execution times are describable using Exponential (9).

Currently, Planners execute at a relatively low priority because they have a relatively long execution time (in the aggregate). That is, although the execution time to decompose one work order is fairly short [describable using Exponential (9)], given the relatively high interarrival interval of the work orders to Planners, a sudden large number of work orders could cause those processes to monopolize the central processing unit (CPU), if the processes are given a high priority [Hissam 04].

The only timing requirement for the Planners is to ensure that the queue between a Planner and its respective Generation of microcoordinates does not become empty during the processing of a work order.

Generation of microcoordinates. The Generation of microcoordinates processes also participate in planning. A queue between the process Planner and the process Generation of microcoordinates contains the subwork orders placed in that queue by the process Planner.

The Generation of microcoordinates processes have an execution time of between 1 and 2 ms and execute periodically with a period of 24 ms. Once, each 24 ms period, process Generation of microcoordinates will remove one subwork order from this queue, generate six individual microcoordinates, and place them in the process Axis Computers Controller queue. Process Generation of microcoordinates’s execution time includes the time it takes to transform a subwork order to six microcoordinates and place them in the process Axis computers controller queue.

The Generation of microcoordinates processes execute at a much higher priority than the Planner processes do. The Generation of microcoordinates processes must complete their work before the end of their period.

Axis computers controller. The single process executes with a period of 4 ms. Process Axis computers controller receives microcoordinates (i.e., robot movement command) from one or more Generation of microcoordinates processes and sends them regularly to the various axis computers controlling the various robot arms—in this case, to three different axis computers controlling three different robot arms. Each Planner-Generation of microcoordinates pair is associated with its own robot arm. Process axis computers controller will read only one microcoordinate from the queue during its 4 ms period. The execution time of process Axis computers controller is 0.5 to 1 ms, and its priority is very high. The process’s deadline is the end of its period.

The queue between process Generation of microcoordinates and process axis computers controller must never become empty while the robot is turned on. If it does, the robot's controller will consider it to be an unsafe condition and abnormally halt the robot.

Controller extension. These processes are medium-priority processes that represent the controller extensions developed by customers. Controller extension processes are important because, on one hand, they have timing requirements of their own that must be satisfied; on the other hand, they can delay the execution of Planner processes and therefore interfere with that process's ability to keep at least one work order in the queue.

For this modelling problem, two types of Controller extension processes are considered: one with stochastic characteristics and one with deterministic characteristics. The characteristics of two Controller extension processes are listed below. We assume that the deterministic Controller extension process has a deadline at the end of its period, while the stochastic Controller extension process has a soft deadline.

Process Miscellaneous represents miscellaneous work that is carried out at the lowest priority in the system. We assume that this work arrives periodically with a period of 5 seconds and executes for 250 ms.

Additionally, high-priority OS functions that occur rarely are executed for a short amount of time.

2. System Architecture described using PPOOA

2.1Static View of the Architecture

The static view is represented by one diagram for the system and one diagram for each of its subsystems.

The notation used is UML with some stereotypes of PPOOA architectural style specifically for coordination mechanisms [Fernandez 02].

The dependencies represented are UML usage dependencies.

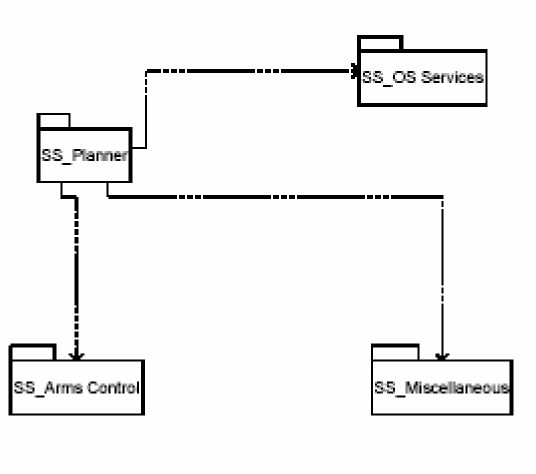


Figure 2 Robotics Controller System

Figure 3 shows the Planner Subsystem components. A brief description of its components generated by PPOOA-Visio tool, is given below. Time is measured in milliseconds.

Component Name	Planner
Component Type	Process
Responsibilities	Carry out planning activities. It receives work orders and creates plans
Activities	o create subwork order:
Interface provided	TBD
Collaborators	Q_Work order, Feedback, Q_subwork order
Subcomponents	TBD
Real-Time attributes (depend on the kind of component used)	Priority= low, execution time= exponential (9), arrivals exponential (75)
Open Issues	None

Component Name	Generation of microcoordinates
Component Type	Periodic Process
Responsibilities	Works on subwork orders. This process will remove one subwork order from the queue, generate six individual microcoordinates, and place them in the microcoordinates queue. When not in motion it only produces stand still microcoordinates
Activities	o check subwork orders: o generate stand still microcoordinates: o generate microcoordinates:
Interface provided	TBD
Collaborators	Q_Subwork orders, Q_Microcoordinates
Subcomponents	TBD
Real-Time attributes (depend on the kind of component used)	Priority= high ,Period= 24 ms, 1<execution time<2
Open Issues	None

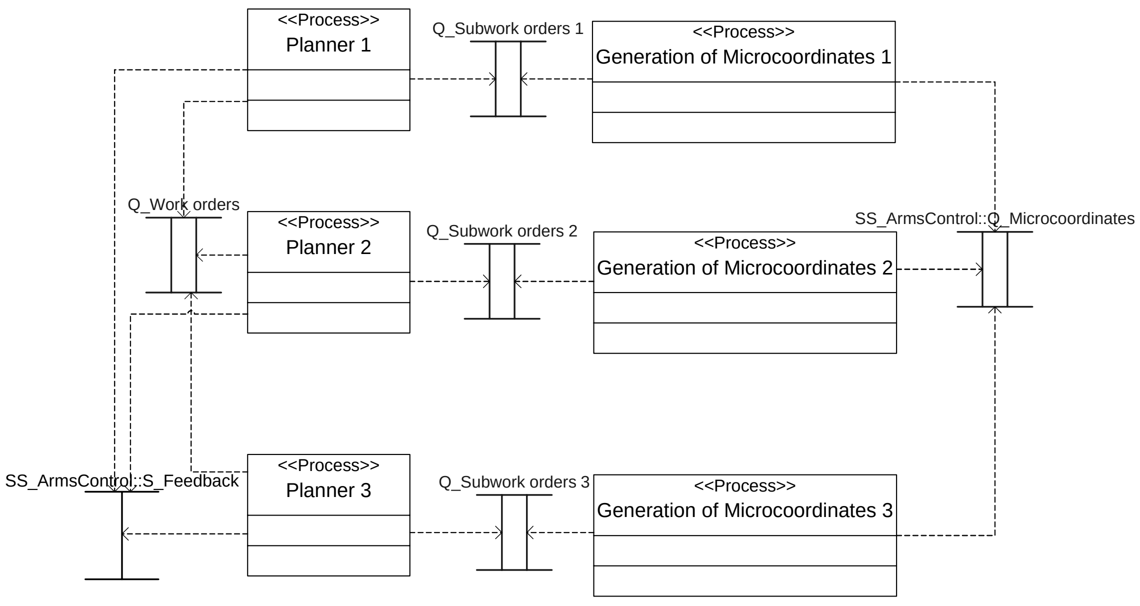

<<Process>> Planner 1
Q_Subwork orders 1
<<Process>> Generation of Microcoordinates 1
Q_Work orders
<<Process>> Planner 2
Q_Subwork orders 2
<<Process>> Generation of Microcoordinates 2
SS_ArmsControl::Q_Microcoordinates
SS_ArmsControl::S_Feedback
<<Process>> Planner 3
Q_Subwork orders 3
<<Process>> Generation of Microcoordinates 3

Figure 3. Subsystem Planner

Each process Planner- Generation of microcoordinates pair is associated with its own robot arm.

Figure 4 shows the Axis controller subsystem and its main building elements.

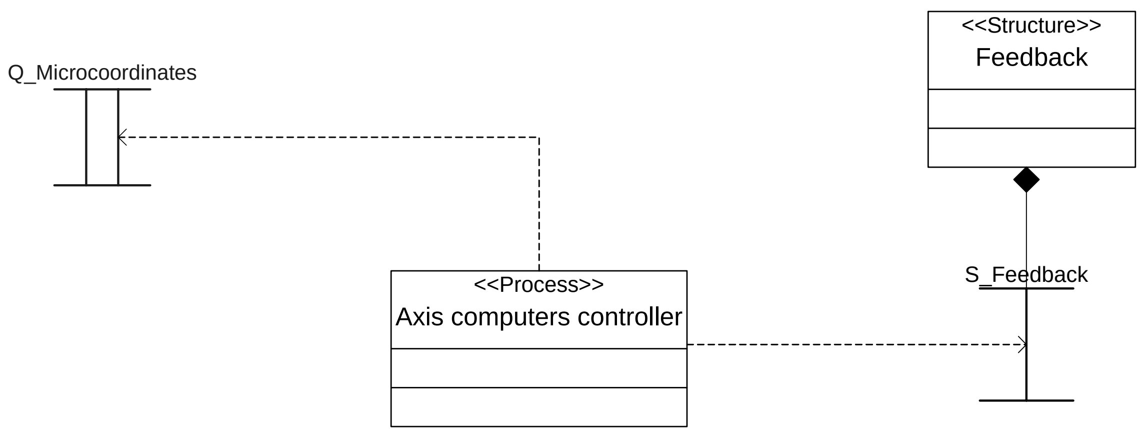


Figure 4. Subsystem Arms Control

Component Name	**Axis computers controller**
Component Type	Periodic Process
Responsibilities	Receives microcoordinates (i.e. robot movement command) and sends them regularly to the various axis computers controlling the various robot arms- in this case, to three different axis computers controlling three different robot arms. Updates feedback information to be used by the Planner process so planning can´t be done entirely in advance.
Activities	o process microcoordinates:
Interface provided	
Collaborators	Q_Microcoordinates, Feedback
Subcomponents	
Real-Time attributes (depend on the kind of component used)	Priority = Very high, Period = 4
Open Issues	None

The axis computer sends feedback to Planner indirectly. That is, it delivers feedback first synchronously to Axis computer controller, and the Axis computer controller updates the feedback component. Planner can read it. A semaphore is used to guarantee mutual exclusion. The semaphore may be part of feedback component and implement the feedback structure as a concurrent form structure [Booch 87]

Component Name	**Feedback**
Component Type	Structure
Responsibilities	Stores current feedback information provided by the axis computers.
Activities	o get feedback: o update feedback:
Interface provided	o get feedback: Reading o update feedback: Writing
Collaborators	Axis computers controller, Planner
Subcomponents	• S_Feedback: semaphore
Real-Time attributes (depend on the kind of component used)	
Open Issues	Implement Feedback as a concurrent form of a Structure [Booch 87]

Component Name	**S_Feedback**
Component Type	Semaphore
Responsibilities	
Activities	o acquire: o release: o acquire: o release:
Interface provided	o Acquire: Signal o Release: Signal o Get status: Read_only
Collaborators	Axis computers controller, Planner
Subcomponents	
Real-Time attributes (depend on the kind of component used)	
Open Issues	It is recommended to implement it as a mutex

Figure 5 shows the Operating System services subsystem.

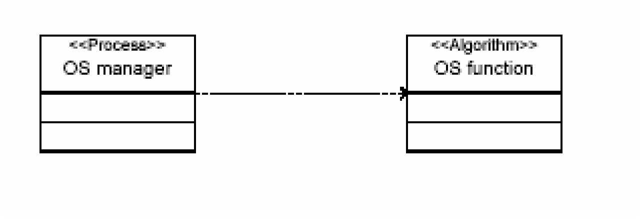


Figure 5. Operating System services subsystem

Component Name	**OS manager**
Component Type	Aperiodic Process
Responsibilities	Handles OS invocations
Activities	o handle OS invocation:
Interface provided	TBD
Collaborators	OS Function
Subcomponents	TBD
Real-Time attributes (depend on the kind of component used)	Priority=High, Arrivals exponential (500)
Open Issues	None

Component Name	**OS function**
Component Type	Algorithmic Component
Responsibilities	An OS funtion
Activities	o process OS function:1
Interface provided	TBD
Collaborators	OS manager
Subcomponents	TBD
Real-Time attributes (depend on the kind of component used)	Execution=1
Open Issues	None

Figure 6 shows the Subsystem Miscellaneous that contains components performing background work and the customer added controller extensions that are augmentations to the core controller platform. Such extensions to the platform could introduce schedulability problems that need to be modelled and assessed. No usage dependencies are known at this moment of the development.

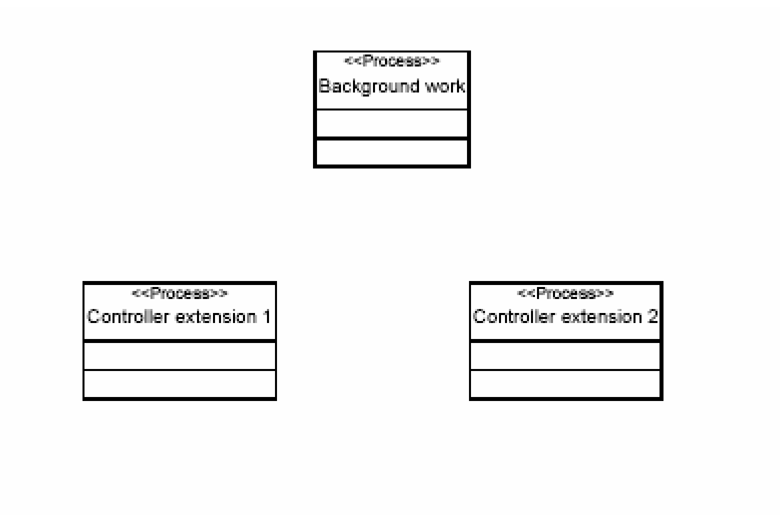


Figure 6. Subsystem Miscellaneous

Component Name	**Background work**
Component Type	Periodic Process
Responsibilities	Miscellaneous work that is carried out at the lowest priority in the system.
Activities	o process miscellaneous work:250
Interface provided	TBD
Collaborators	TBD
Subcomponents	TBD
Real-Time attributes (depend on the kind of component used)	Priority= very low ,Period= 5000 , execution time=250
Open Issues	None

Component Name	Controller extension 1
Component Type	Periodic Process
Responsibilities	Controller extension developed by a customer
Activities	TBD
Interface provided	TBD
Collaborators	TBD
Subcomponents	TBD
Real-Time attributes (depend on the kind of component used)	Priority= medium, Period= 100
Open Issues	TBD

Component Name	Controller extension 2
Component Type	Aperiodic Process
Responsibilities	Controller extension developed by a customer
Activities	TBD
Interface provided	TBD
Collaborators	TBD
Subcomponents	TBD
Real-Time attributes (depend on the kind of component used)	Priority= medium , Arrivals exponential (100)
Open Issues	TBD

2.2Dynamic view of the architecture

The dynamic view of the architecture is represented by modelling the system responses to a stimulus or event that may be either external, either internal or timer.

A response is modelled using the PPOOA Causal Flow of Activities technique and represented using UML activity diagrams.

A CFA is a causality flow of activities triggered by an event. The term causality flow means cause-effect chains that cut across many components of the system architecture. This chain progresses through time and executes the activities provided by the system components until it gets to an ending point.

In general, there can exist several active CFAs at a moment and they can also interact with each other. It is also possible that two instances of the same CFA coexist. When a new instance of the CFA starts, it continues through a series of cause-effect elements until it finishes and it ignores other external stimuli at the starting point. Generally, a new stimulus will create a new instance of the CFA.

The CFA instances can stop at certain waiting points where there is coordination with other instances. At these points, coordination mechanisms or objects with concurrent access are located.

The following event arrival patterns can be considered [Klein 93]:

- Periodic: Interval [$\pm \Delta$t] [phase].
- Irregular: Interval, Interval, [Interval].
- Bounded: Minimum_Interval [Maximum_Interval].
- Bursty: Number of Events, Interval.
- Unbounded: Distribution Function.

In a real-time system software architecture, each of the CFA instances is associated to some type of temporal requirement, that is, a certain period of time in which the associated effect or response should occur. These requirements are classified in: hard, firm or soft in the same way as it is done at a real-time system level.

- A hard temporal requirement CFA is considered to have failed if the effect associated to an instance needs more time than the specified time requirement.
- A soft temporal requirement CFA meets time requirement in average for its instances.
- A firm temporal requirement CFA compromises the last two cases, being able in this way to use more time for some CFA instance.

The response associated to a triggering event corresponds to the activities achieved inside the triggered CFA.

It is important to notice that the response includes all the data processing, independently of where the code is located.

Many times, parts of the response are considered overheads. Aspects such as interruption management, device management, context switching, remote servers and system calls, are considered as part of the response.

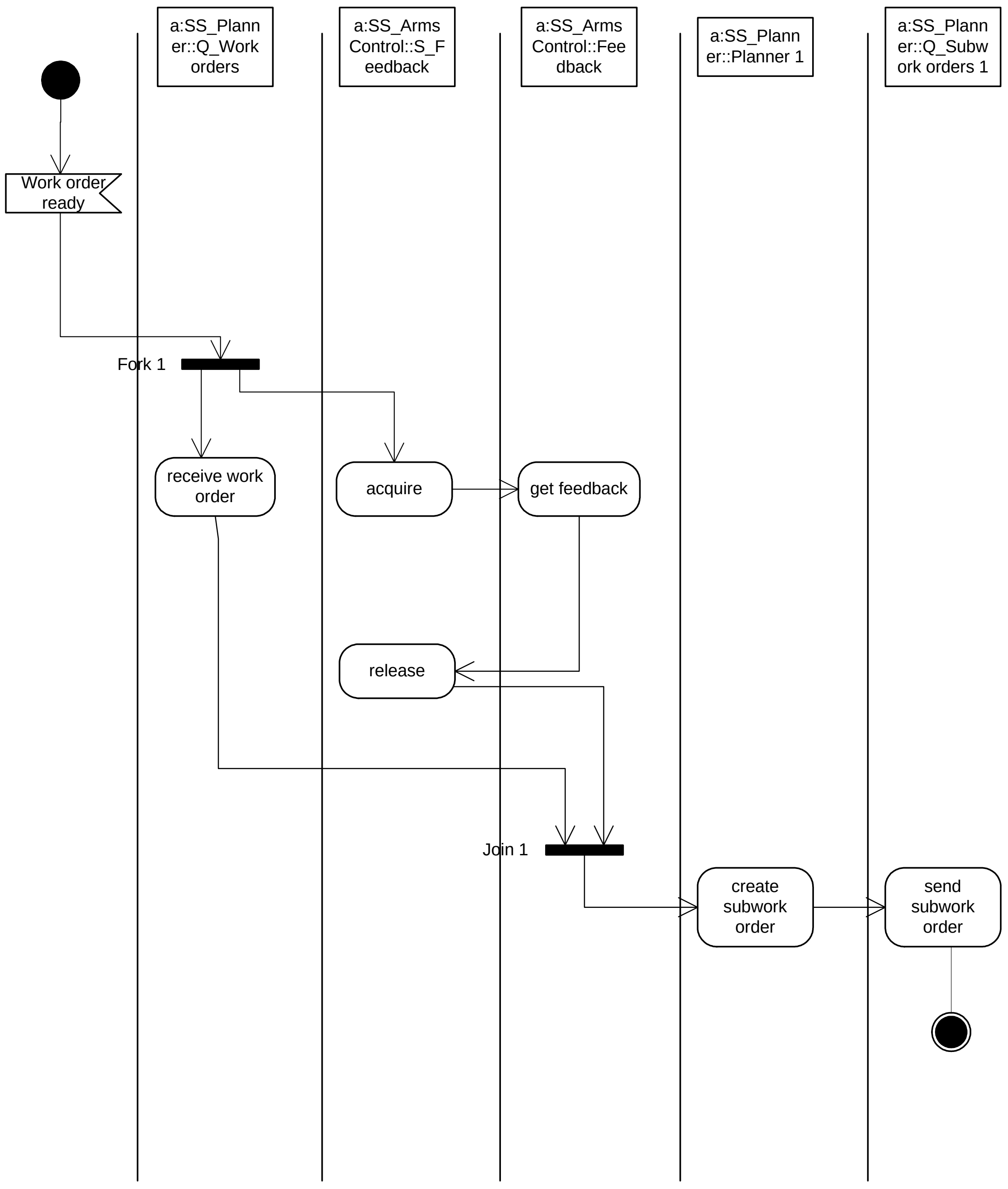
The smallest division of a response is an activity, or action in terms of RMA. Due to scheduling requirements, the computation that takes place in an activity cannot cause changes in the system resources allocation. The scheduling points are each of the time instants where decisions relative to system resources allocation are made. Scheduling points are considered when:

- The priority of a process changes.
- An Input/Output operation starts or finishes.
- Coordination is done with another CFA instance.
- An interruption from the system clock is received.

The activities allocation using the UML concept of partition (“swimlanes”) is an important contribution of the PPOOA architectural style to the representation of the mapping of the CFA activities to the components and coordination mechanisms of the system architecture. This allocation is a critical issue in the systems engineering process since it permits the assessment of the different design decisions.

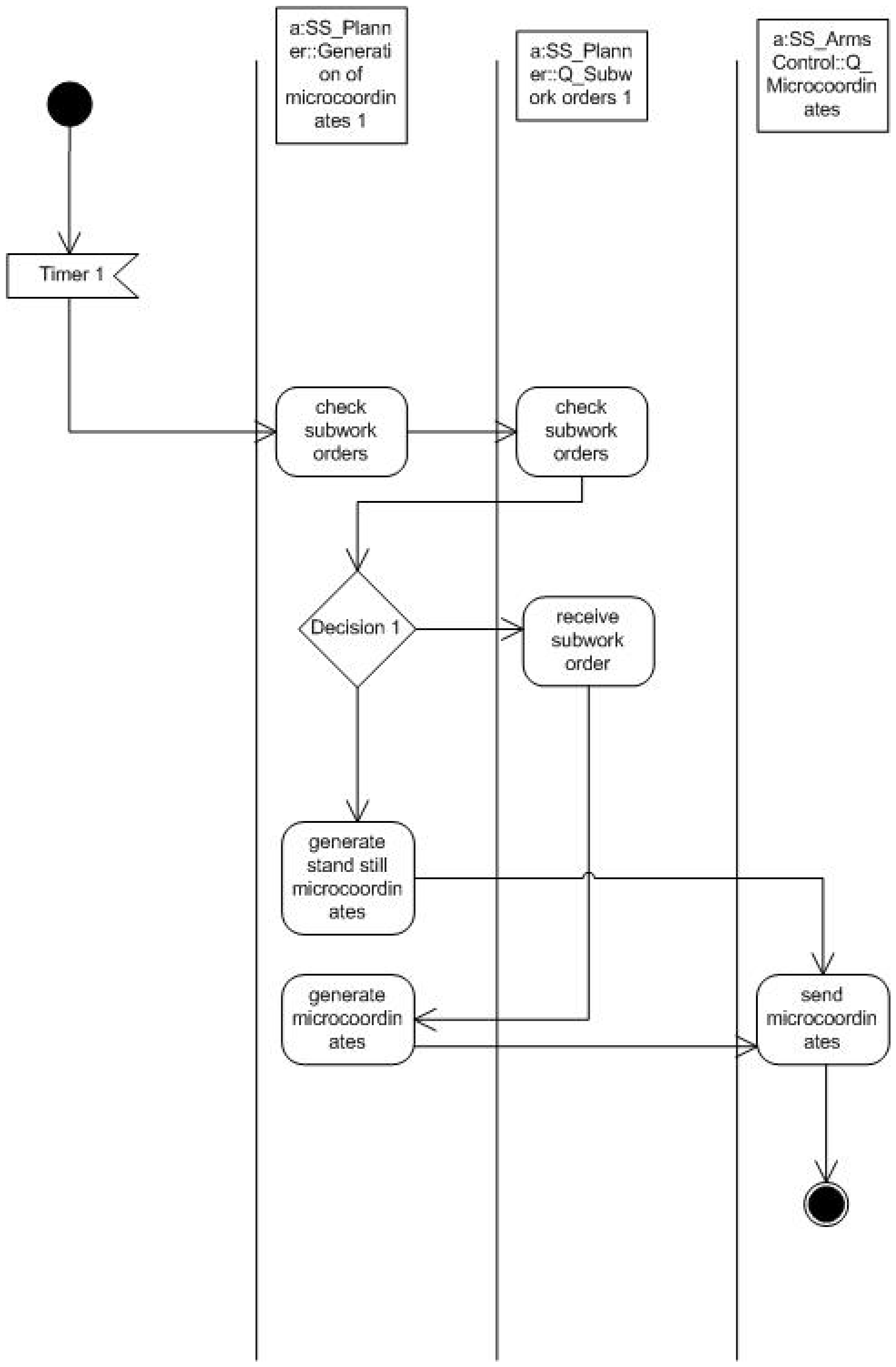
An activity diagram is used to represent each CFA of the robotics systems. As we mentioned previously, several instances of the same CFA or response can coexist simultaneously, for example in the execute subwork order CFA.

CFA: Create subwork order



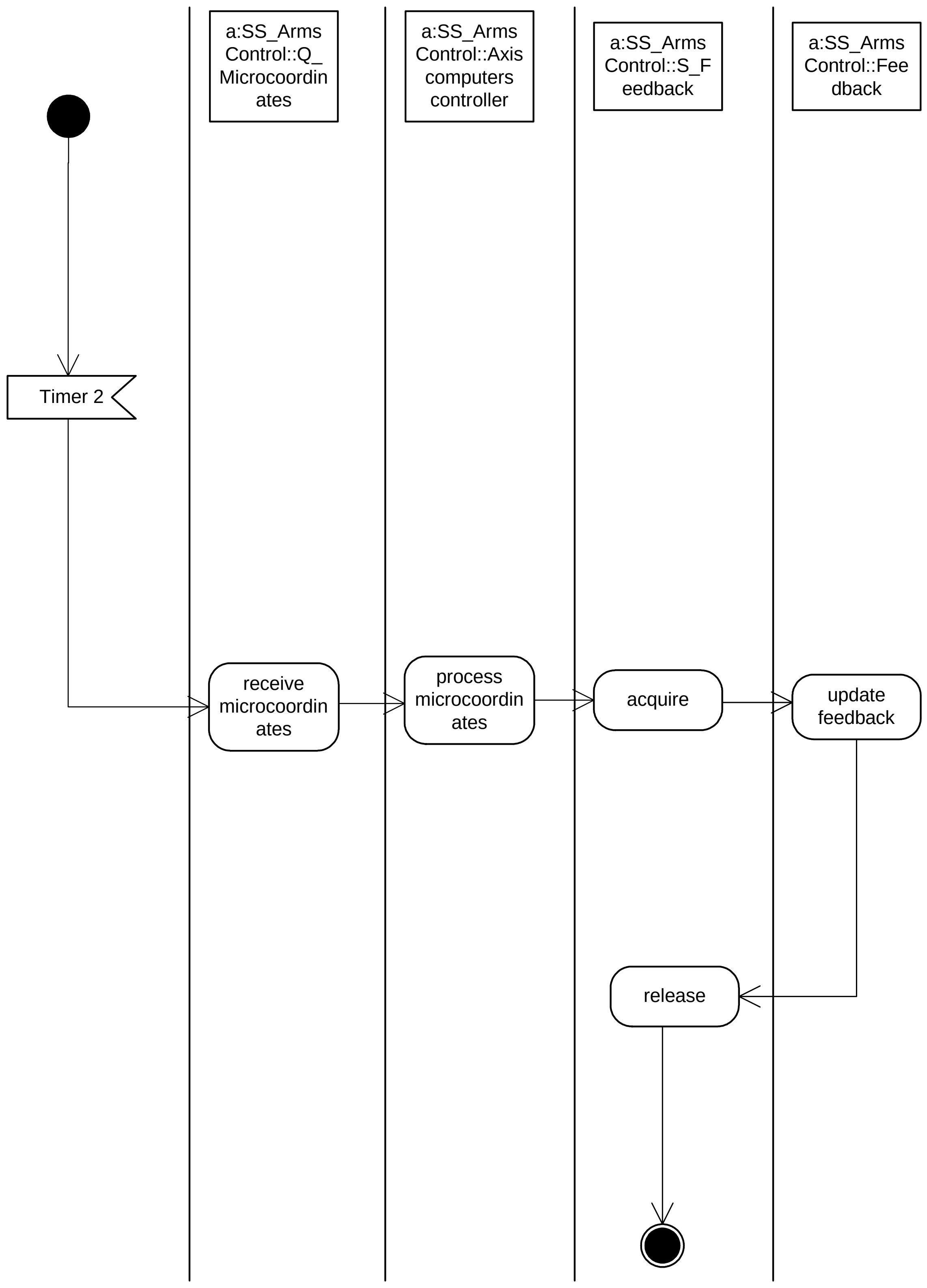
Starting Event	work order ready
Event type	External_event
Arrival pattern	Unbounded
Associated temporal requirement	
Associated activities	receive work order= get feedback= release= acquire= create subwork order= send subwork order=

CFA:Execute subwork order


a:SS_Planner::Generation of microcoordinates 1
a:SS_Planner::Q_Subwork orders 1
a:SS_Arms Control::Q_Microcoordinates
Timer 1
check subwork orders
check subwork orders
Decision 1
receive subwork order
generate stand still microcoordinates
generate microcoordinates
send microcoordinates

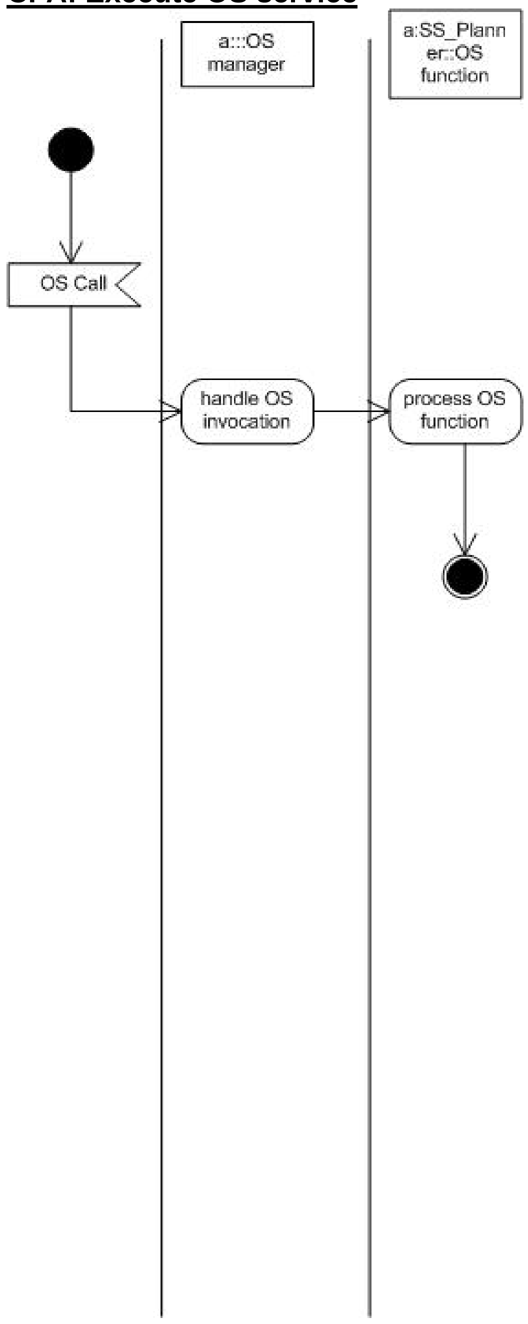
Starting Event	Timer 1
Event type	Timer_event
Arrival pattern	Periodic
Associated temporal requirement	
Associated activities	check subwork orders= check subwork orders= generate stand still microcoordinates= generate microcoordinates= receive subwork order= send microcoordinates=

CFA: Execute arm control


a:SS_Arms Control::Q_ Microcoordin ates
a:SS_Arms Control::Axis computers controller
a:SS_Arms Control::S_F eedback
a:SS_Arms Control::Fee dback
Timer 2
receive microcoordin ates
process microcoordin ates
acquire
update feedback
release

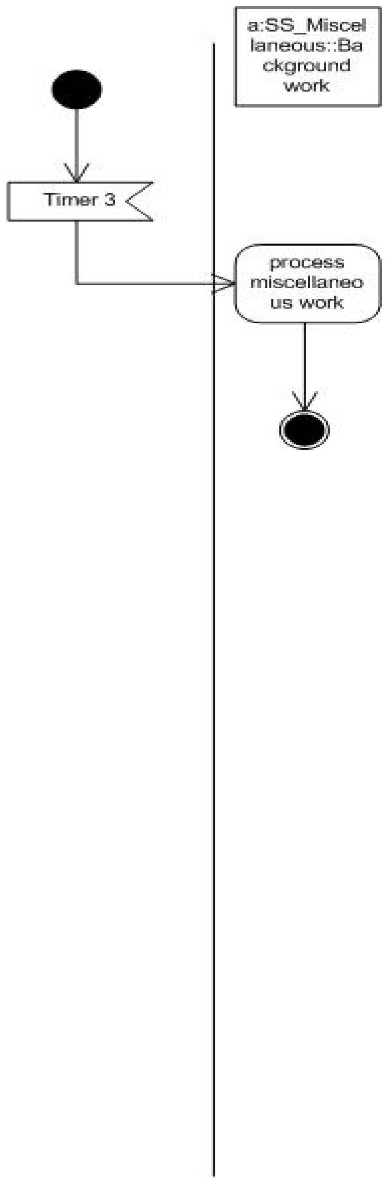
Starting Event	Timer 2
Event type	Timer_event
Arrival pattern	Periodic
Associated temporal requirement	4 milliseconds
Associated activities	receive microcoordinates= process microcoordinates= acquire= update feedback= release=

CFA: Execute OS service



Starting Event	OS Call
Event type	Internal_event
Arrival pattern	Unbounded
Associated temporal requirement	
Associated activities	handle OS invocation= process OS function=

CFA: Execute miscellaneous work


a:SS_Miscellaneous::Background work
Timer 3
process miscellaneous work

Starting Event	Timer 3
Event type	Timer_event
Arrival pattern	Periodic
Associated temporal requirement	5000 milliseconds
Associated activities	process miscellaneous work=

3. Conclusion

The modelling problem approach offered by PPOOA architectural style is used here to illustratre how architecture static and dynamic representations may be useful for assessing design decission related to the real-time properties of the system. The modelling of system responses and the allocation of responses activities to system components is a critical issue in the system synthesis and later tradeoff analysis.

The application of analytical assessment techniques such as RMA to these models is staightforward.

References

[Booch 87] Booch G.,"Software Components with Ada, Structures, Tools and Subsystems", The Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.

[Fernandez 98] Fernandez J.L.,"An Architectural Style for Object-Oriented Real-Time Systems", 5th International Conference on Software Reuse. Victoria (Canada). IEEE 1998.

[Fernandez 02] Fernandez J. L. Chapter 12 of the book: "Business Component- Based Software Engineering", Frank Barbier (ed), Kluwer Academic Publishers, Boston, October 2002

[Fernandez 02 a] Fernandez J.L. and Masson W.A. "A Process for Architecting Real-Time Systems", 15th International Conference on Software and Systems Engineering (ICSSEA), Paris, December 2002.

[Fernandez 03] Fernandez J.L. and Martinez J.C."Implementing A Real Time Architecting Method in a Commercial CASE Tool", 16th International Conference on Software and Systems Engineering (ICSSEA), Paris, December 2003.

[Hissam 04] Hissam S.A. and Klein M. "A Model Problem for an Open Robotics Controller". CMU/SEI-2004-TN-030.Software Engineering Institute.Carnegie Mellon University. July 2004.`

[Klein 93] M.H. Klein, T. Ralya, B. Pollak, R. Obenza, M. González Harbour. A Practitioner´s Handbook for Real-time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems. Kluwer Academic Publishers, Dordrecht, 1993.