Estimation Process of Performance Constraints during the design of Real-Time & Embedded Systems

by

Ramón PUIGJANER
Univ. de les Illes Balears
Departament de Ciències
Matemàtiques i Informàtica
Cr. de Valldemossa km 7.6
E-07071 Palma

Abdelmalek BENZEKRI
IRIT - SIERA
Univ. Paul Sabatier
118, Rte. de Narbonne
F-31062 Toulouse Cedex
phone: (+33).61.55.67.68
e-mail: benzekri@irit.fr

Sandra AYACHE *et al*[1]
Matra Marconi Space
Space Branch
Rue des Cosmonautes
Z.I. du Palays
F-31400 Toulouse

Abstract

Real Time and Embedded systems always have performance constraints either on response time or on throughput or on device utilisation rate. With conventional design methods these constraints are verified until the testing phase. No estimation of the variables on which the designer has specified some performance constraint is available before.

In the frame of the Esprit II project COMPLEMENT[2], whose aim is to detect and fulfil the gaps between the different methods and tools that cover the whole design process of a real time and embedded system, an important effort is invested in including the use of performance modelling techniques very early in the development life-cycle in order to compute estimations of the performance criteria.

The goal of this paper is to present some of the results obtained around the integration of performance evaluation modelling within design methods:

- the taxonomy of performance requirements to classify the types of performance constraints,
- the performance information model to bridge the gap between the design entities and the performance models, by introducing annotations to be able to run the derived models in order to provide the efficiency estimation of the system.

---

1. Eric CONQUET, Delphine GAZAL, Omar HJIEJ, Alberto VALDERRUTEN and Yves RAYNAUD, members of the COMPLEMENT Performance Task Force team.

2. COMPLEMENT is partially funded by the Commission of the European Communities as an ESPRIT II project (project number 5409)

1. Introduction

There are many reasons why a designer may wish to calculate the performance of real time and embedded systems, as their functionality may be influenced by performance requirements. If the quality of service is often specified, it is of the responsibility of the designer to build a system that meets the non-behavioural requirements. Response times, throughput of the system or its utilisation rate are some of the non functional criteria that measure the efficiency of a system. Performance evaluation modelling is an activity that is driven by quantitative specifications of the system separate functions. This activity leads to compute the value of the performance variables in order to verify if the performance constraints are met. With conventional design methods, this activity is often held at the testing phase. However a bad design decision can lead to a non acceptable efficiency of the implemented system and hence oblige the designer to go back to the design phase. To avoid such a costly revision, the integration of performance evaluation techniques within design methods is an issue to predict the non behavioural criteria, early in the development life-cycle. This hence can lead to a better understanding of the system and moreover, both quantitative and qualitative validation of the system may be conducted.

Section 2 of this paper is devoted to the taxonomy of performance constraints versus the types of systems.

Section 3 deals with the performance information model which aim is to introduce the set of annotations required to specify quantitatively the design entities attributes related to performance.

Section 4 describes these annotations.

Finally section 5 summarizes the work achieved up to now and points out ongoing activities.

2. Taxonomy of Performance Requirements versus Types of Systems

2.1 Taxonomy of Performance Requirements

The meaning of the term "performance requirement" complies with the PSS-05-0 definition [ESA 91]:

Performance requirements: These specify numerical values for measurable variables (e.g. rate, frequency, capacity and speed). Performance requirements may be incorporated in the quantitative specification of each function, or stated as separate requirements. Qualitative statements are unacceptable (e.g. replace "quick response" with "response time must be less than x seconds for y% of the cases with an average response time of less than z seconds"). The performance attributes may be presented as a range of values, for example the:

- worst case that is acceptable,
- nominal value to be used for planning,
- best case value, to indicate where growth potential is needed.

Performance requirements should express static or dynamic requirements. Static performance requirements are generally described as capacity requirements, whereas dynamic performance requirements are denominated as speed requirements.

Capacity is easy to quantify: specify how many end users, nodes in networks... Since capacity requirements often increase in the future, it may also be helpful to indicate how volatile the stated capacity is likely to be in the future [DAV90].

Dynamic performance requirements are often more difficult to satisfy and to evaluate. **Timing constraints** and **Probabilistic constraints** are the two main dynamic performance requirements classes.

Timing constraints define response time requirements for systems and/or their environment. A special attention is needed to express correctly such requirements. Four types of timing constraints can be defined [DAS85]:

- *Stimulus-response* also called responsive output constraint which denotes typically a response time of a transaction. This constraint can be expressed by a mean value or by some type of statistical distribution related characteristics.
- *Response-response* also called timed output constraint where the constraint is given by the interval between successive outputs. This interval is frequently constrained to be between a minimum and a maximum values but can also be defined by some type of statistical distribution related characteristics.
- *Stimulus-stimulus* also called throughput input constraint which denotes the interval between two successive inputs.
- *Response-stimulus* also called interactive input constraint where the constraint is given by the interval between an output and the following input.

The latter two types of timing constraints imply that the system may be able to detect a violation of timing constraints by the environment or the users, and then perform an alternative action (generate warnings, error messages...).

The former two types of timing constraints define timing requirement on the system being specified. They imply that designers of the software may involve any architectural solution they choose but the software components must be implemented rapidly or slowly enough to meet the timing requirement.

If the approach is taken to specify the environment and the system as two cooperating processes, then the timing constraints can be reduced to the two former types. Each process then

controls the inputs - or even ignores them - but has to take care of the constraints upon the responses.To simplify, these constraints can be seen as either duration constraints (time between two events of same type) or throughput constraints (time between two events of different types).

An example of a duration constraint is the one where a SYSTEM X test node shall send a TC or generate a stimulus to SAS as a result of a monitoring exception within 10ms (time between the detection of the exception and the TC packet/command to SAS leaving the test node). Mandatory attributes expressing the duration constraint are:

- Identification of the constrained object (SYSTEM X).
- Description of the constrained transaction (send a TC or generate a stimulus to SAS as a result of a monitoring exception).
- Duration limit (absolute value or average value, here 10ms).
- Relation between the duration of the transaction and the duration limit (less than, greater than or between).

Other recommended attributes are:

- Events delimitating the constrained transaction (detection of the exception, TC packet/ command to SAS leaving the test node).
- Context (specific, any).

An example of a throughput constraint is the one where a SYSTEM X test node shall generate a maximum number of 64 data delivery messages per second, independent of the other processing on the node. Mandatory attributes expressing the throughput constraint are:

- Identification of the constrained object (SYSTEM X test node).
- Kind of throughput (interactive input or timed output, here timed output).
- Identification of the kind of "flowing" item (data delivery message).
- Description of the constrained transaction (generate data delivery message).
- Throughput limit (64 data delivery messages / second).
- Kind of throughput constraint (minimum input, exact input, maximum output, minimum output, exact output).

Other recommended attributes are:

- Context (specific, any).

Probabilistic constraints include both **resource utilisation rate** and **throughput absorption**. Resource utilisation rate denotes the use of a system component per time unit. Minimum or maximum values can be given in order to express the limitations on the resources. An example of a resource utilisation rate constraint is the one where for each SYSTEM Y computer, the set of processes (applicative processes or system processes) needed for the ex-

ecution of crucial and nominal functionalities, excepted tendency analysis, shall not use more that 50%of the CPU in nominal mode and 70% of the CPU in peak mode, in order to insure a sufficient evolutivity. Mandatory attributes expressing the resource utilisation rate constraint are:

- Identification of the resource (SYSTEM Y computer).
- Rate limit (absolute or average value, interval).
- Relation between resource utilization rate and rate limit (less than or greater than).

Other recommended attributes are:

- Context (specific, any).
- Constraint origin (rigorous level of satisfaction: evolutivity, or performance). In the case of evolutivity origin, it is necessary to take into account that for some resources the service time is a function of the workload (e.g. the disks have a load dependent service time due to the time used to solve the contention in the control unit); in consequence, in these cases the capacity of growth is not directly proportional to the increase of the workload.

Throughput absorption constraint is used to express the capacity of a system to absorb the throughput input and is measured as the allowed loss rate. An example of a throughput absorption constraint is the one where an ATM switch (Broadband ISDN) is to be designed so that the throughput of 1 million cells per second be allowed to have a maximum cell loss of 10^{-9}.Mandatory attributes expressing the throughput absorption constraint are:

- Identification of the constrained object (ATM switch).
- Description of the constrained transaction (ATM cell processing).
- Throughput absorption limit (upper limit expressing the constraint tolerance) (the loss of one cell over a billion in average).

Other recommended attributes are:

- Starting event of the transaction (the front wave of the cell).
- Context (specific, any).

Our purpose elaborating this taxonomy of performance requirements was to reach two different goals:

- The first one is to provide guidelines for the production of "good" performance requirements.
- The second one is to provide a starting point for the study of traceability of performance requirements onto design elements: it is easier to identify elements of the design concerned with performance when information on the kind of performance requirement is available.

2.2 Types of System Transactions

The specification of a system's performance defines the temporal behaviour that the system must satisfy under a certain workload scenario. Such specification is compound of a description of the environment where the system is to be run -also called the scenario- plus a set of timing and probabilistic constraints on the behaviour of the system. The scenario should describe both the workload of the system plus the hardware constraints, if any, imposed to the designer, whereas timing and probabilistic constraints specify both responsiveness of the system and its resources utilisation rate [UIB77].

System transactions form part of system requirements. They constitute a valuable means of description of intended behaviour of the system. It is usual to classify system requirements into two groups: functional and non-functional. One of the major problems with functional system requirement is that they tend to define system actions only locally. On the other hand, non-functional requirements (the abilities and concrete constraints) are frequently widespread in their effect. System transactions represent the way of bringing these two disparate views into unison and provide a better focus to ensure that the user requirements are properly taken into account by the requirements concerning the system under development [TSG11].

The usual way of obtaining the description of overall system behaviour consists of the definition of a series of input events together with the system actions related to those events. A characterisation of system transactions depending on the constraints upon these events may guide the designer tackling the performance requirements.

Throughput input transaction systems (e.g. a system processing inputs coming from an external source) must take the inputs at the rate the environment provides them. The control is outside of the scope of the target system. The rate of the information flow may be absolutely determined or be stochastic, with a known or assumed distribution of arrival times. The most usual arrival patterns are at constant intervals or as a Poisson distribution, although other distribution laws sometimes occur.

Interactive input transaction systems (e.g. polling systems), oblige the environment to wait for a system output before generating a new input. Thus after an input, there will be no subsequent input until the appropriate output has been given by the target system, allowing it to control the rate at which transactions arrive. The interval between the previous output and the subsequent input may be absolutely determined or stochastic.

Timed output transaction systems (e.g. unsolicited and periodical actions), must send events to the environment at a pre-established time intervals. The output rate should be specified with a tolerance margin.

Responsive output transaction systems (e.g. interactive terminal system), must send events to the environment at times within a specified period or delay after some input (the time necessary to process the input) and the environment sends a new input some time (thinking time) after the reception of the output. The bounds to the delay are not usually absolute but are determined by a distribution law.

3. The Performance Information Model

3.1 The COMPLEMENT approach

The COMPLEMENT approach to the investigation of real-time development methods is, firstly, to determine the information which must be created (and recorded) for RT&E systems and secondly to compare this with the information which can be created and recorded by the methods and tools under investigation [TSG22].

The information model is the total sum of the information which must be generated and/or collected during the life of any project to develop a computer based system. The requirement model contains all information about what the system is intended to do. The design model contains all the information about how the design is intended to satisfy the requirements. The implementation contains all the information about how the implementation achieves the intentions of the design. The technical management model contains all the information on the processes used to develop the requirement, the design and the implementation.

The information model in COMPLEMENT has to express various desired characteristics of both the target RT&E system and the process by which it is developed. In order to achieve this goal, and to facilitate the integration of the performance modelling activity within the life-cycle, the Performance Information Model has been elaborated. The aims are to represent entities used in the different kinds of activities related to performance:

- the performance constraints expression,
- the modelling and design assessment,
- the performance variable measurement.

3.2 Conventions

As we want to show information entities and relationships, we describe the information models using an entity-relationship approach, with our own conventions.

Note all examples given below are taken from Figure 1.

Convention 1: *Entity A consists of two or more sub-entities B, C...* e.g. the Performance Constraint entity consists of Duration Constraint, Throughput Constraint and Utilisation Rate Constraint entities.

Convention 2: *Entity A has B as an attribute;* e.g. Design Component entity has Performance Annotations as attribute.
Convention 3: *Entity A "Xes" the entity B;* e.g. Performance Model produces Model Result.
Convention 4: Two (or more) relationships having the same label and relating the same entity A to other entities B and C can be described by joining or splitting arrows; e.g. Measurement Result validates Performance Model and Model Result (which is produced by the Performance Model).
Convention 5: *Relationship X has A as an attribute;* e.g. Scenario Includes System Transaction with Transaction Timing Parameter as an attribute.

3.3 Overview of the Performance Information Model

A top level view of the model is represented in Figure1.
The specification of **Performance Requirements** requires the specification of:

- System utilization **Scenarios**
  We consider that a Scenario is a list of pairs **<System Transaction, Transaction Timing Parameter>** where the transaction timing parameter is a "user thinking time" for a responsive output system or an activation rate (of transaction start events) for throughput input systems.
- **Performance Constraints**
  The two types of dynamic performance constraints **Timing Constraints** and **Utilization Rate Constraints** identified in the taxonomy of performance requirements are represented. The relationship **refers to** between a Performance Constraint and a Scenario recalls that it is important to specify the context of applicability of a performance constraint.
- **Performance Measurement Requirements** and **Performance Measurement Means** (for taking into account the need for performance measurements since system specification)
  Performance Measurement Means express *how* performance measurements will be done. A Performance Measurement Requirement expresses a performance variable to be measured and how it will be measured. As this performance variable is expressed in a performance constraint, we say that a Performance Constraint **defines** a Performance Measurement Requirement. Such a requirement **specifies** a Measurement Check to be done after system implementation and **refers to** Performance Measurement Means.

Requirements Elements to be considered for the specification of performance aspects are:

- **System Transactions**

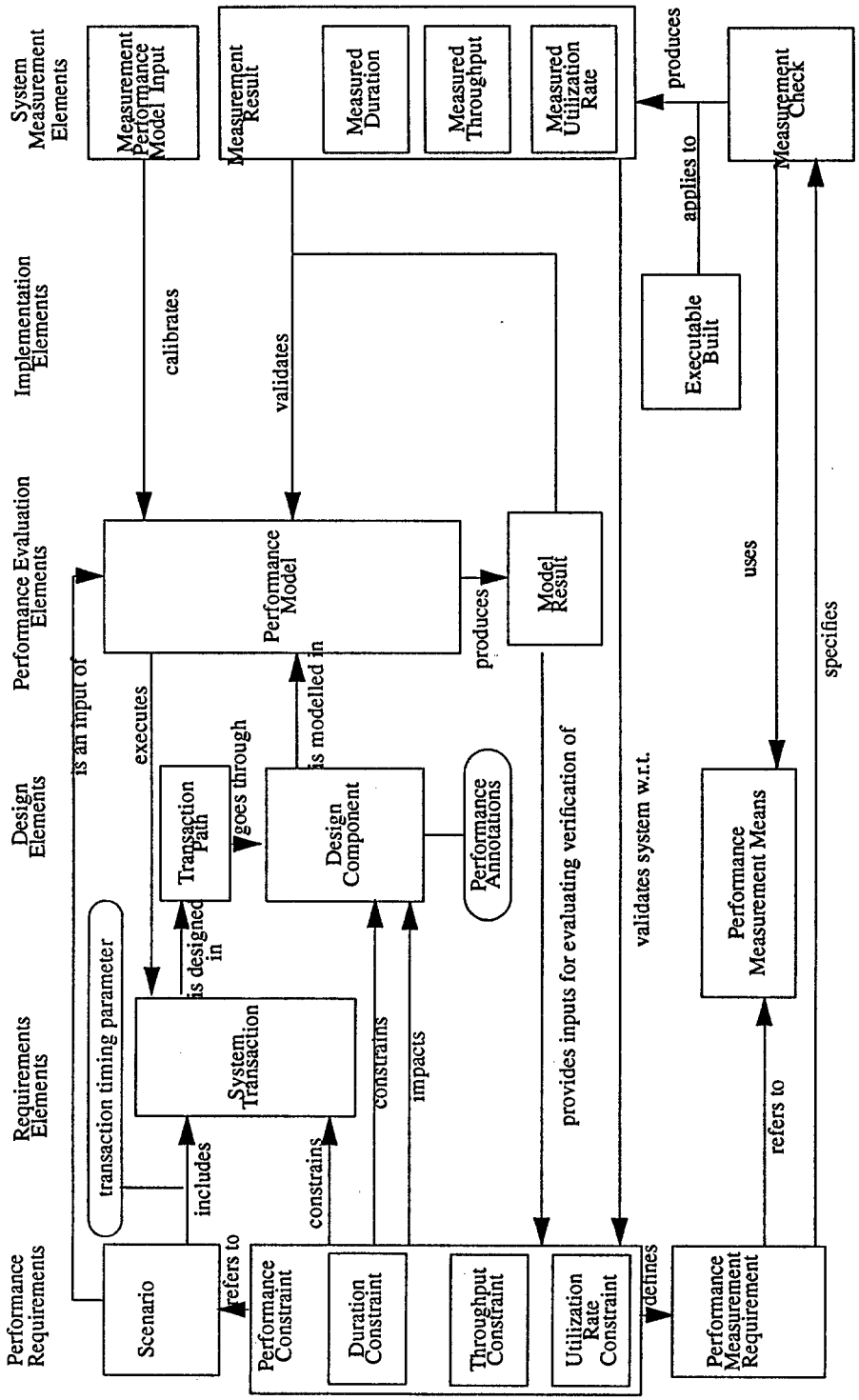

FIGURE1: COMPLEMENT performance information model (top level view)

The relationship **includes** (Scenario ---> System Transaction) and its attribute has been introduced in order to allow the definition of scenarios as a list of pairs <System Transaction, Transaction Timing Parameter>. The relationship **constrains** (Performance Constraint ---> System Transaction) recalls that it is important to express the constrained transaction when specifying a Duration or a Throughput Constraint.

Design Elements to be considered for performance evaluation are:

- **Design Components**

  We have represented **Performance Annotations** to be associated with design components as attributes of Design Components. These annotations are introduced in section 4.

  The relationship **constrains** establishes links between Performance Constraints and sets of design components belonging to the same hierarchical level in a design hierarchy. It aims to trace during design the software or hardware design components which are constrained by the constraint.

  The relationship **impacts** establishes links between Performance Constraints and software atomic components of the lowest level of a hierarchical design. It aims to determine precisely the software atomic components which will be impacted by the constraint. That means that a change in the design of such a component may impact on the fact that the constraint is verified (or not verified). The main interest of **impacts** links is the identification of software components to be modelled.

  To illustrate the difference between the two relationships, we can say that an Utilization Rate Constraint constrains a hardware component and impacts software components.

- **Transaction Paths**

  This entity represents how design components *implement* system transactions. We can say that a Transaction Path **goes through** Design Components. As a transaction path is a design element, we have chosen to say that a System Transaction **is designed in** a Transaction Path.

Entities related to **Performance Evaluation** are:

- The **Performance Model** itself in which some Design Components (considered as *sensible* from a performance point of view) **are modelled**. The relationship **is an input of** (Scenario ---> Performance Model) expresses that a Performance Model has to be generated for each system utilization scenario. The relationship **executes** (Performance Model ---> System Transaction) expresses that during model solving, transactions belonging to the input scenario are executed.
- **Model Result**

This entity aims to represent results **produced by** the resolution of a Performance Model. Model Results **provide inputs for evaluating verification of** Performance Constraints.

The **Implementation Element** to be considered for performance measurement is the **Executable Built.**

Entities related to **System Measurement** are:

- **Measurement Check**

  It aims to represent quantitative checks. A Measurement Check **uses** Performance Measurement Means.

- **Measurement Result**

  It aims to represent results **produced** by Measurement Checks. The three kinds of Measurement Results, **Measured Duration, Measured Throughput** and **Measured Utilization Rate** are represented.

  The relationship **validates system w.r.t.** (Measurement Result ---> Performance Constraint) expresses that some measurement results corresponding to measurements of variables expressed in performance constraints are useful for system validation.

  The relationship **validates** (Measurement Result, Model Result ---> Performance Model) expresses that the comparison between a measurement result (measuring a set of performance variables) and a model result (estimating the same variables) is useful for model validation.

- **Measurement Performance Model Inputs**

  They are measurements of model inputs (e.g. Transaction Timing Parameters, operation durations, hardware component characteristics...). These measurements are useful for performance model **calibration.** Calibration consists in replacing values of performance models inputs (which are estimated before system implementation) by values measured during and after system implementation. Exceptionally, part of the calibration process can be done before system implementation when some systems components are reused. The calibration activity is a cornerstone of a good modelling process because it allows reuse of models which are modelling reusable components.

4. Performance Information Annotation

4.1 Levels of Performance Models

During the design, as the system is not yet available, the way to estimate whether or not the performance constraints are met is to build performance models.

The results of these models should allow the designers to select among several design options taking into account the estimated performance of each one.

Two levels of performance models are being explored:

- the *logical performance model* built from the system logical design that should allow estimations of response times and throughputs to be obtained.
- the *physical performance model* built from the system physical design (the implementation of the previous one taking into account the geographical distribution of the hardware resources and the physical implementation of software components in hardware systems) that should allow estimations of response times, throughputs and utilization rates to be obtained.

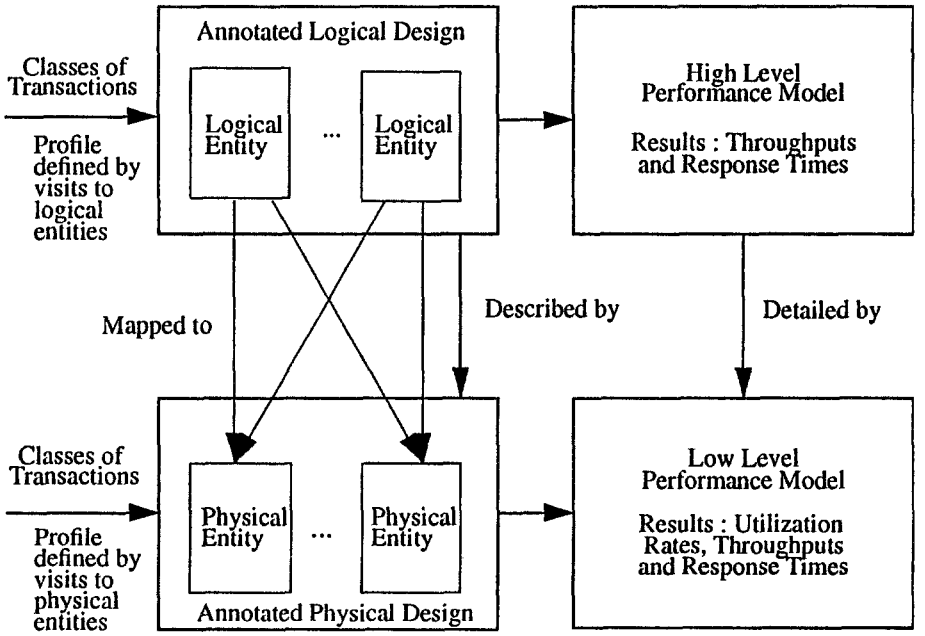

FIGURE 2: Levels of Performance Models

From the design information it is possible to deduce the model but to run it, it is necessary to annotate the design with supplementary information concerning the estimations of the elementary times to execute the access to the logical and/or physical design entities. The performance model allows the computation of the delays generated by the logical lockings and by the resource use conflicts by means of analytical or simulation techniques.

4.2 Definitions

We assume that the design of a RT & E System is an iterative process. In this process (see the Figure 2), we identify two levels:

- the logical design,
- the physical design.

In the first one a set of logical entities (like access to a Data Base, send a message, etc.), that we shall call operations, is defined.

From the other hand we shall define a set of classes of active entities, that we shall call transactions, that are executed by a sequential access to different operations. During this execution some kind of synchronization among several transactions can appear (locking the use of an operation, fork and join operations, etc.) and this synchronization will be the main source of execution delays.

To execute these operations, they must be mapped on to physical entities or devices (CPU, disks, specific types of networks, etc.) taking into account their capacity and their geographical distribution; this physical implementation constitutes the physical design of the RT & E System. Once this implementation has been done, each one of the transaction classes can be described by a sequence of accesses to these physical entities. When we look at the system at this level, the delays are mainly due to the waiting queues when several transactions try to use simultaneously the same device.

From the transaction profile at the logical level and the mapping of the logical design onto the physical design, it is possible to deduce the transaction profile for the physical level. However, in order to have clearer models it can be useful to collapse several successive operations in one if there are no synchronization relations among them or with other transactions and/or operations.

A set of different transactions can be included in the same class if they have a similar path through the entities constituting the design. A transaction is bounded by its starting and ending events.

The scenarios defining the RT & E System workload will be characterized by different mixtures of transaction classes.

From both levels of design, performance models can be derived in order to estimate, whether the performance constraints established in the requirements analysis, are met. From the Logical Design can be derived the High Level Performance Model and from the Physical Design, the Low Level Performance Model. However, with the strict information contained in both designs it is just possible to build the model but not to run it because of lack of numerical data. This information must be included in the design by means of annotations in-

cluding the necessary data.

In all cases it is assumed that from the design description it is possible to build a model describing the behaviour of the system. However, to run it to obtain performance estimations, it is necessary to include by annotation some supplementary information in the design.

As the performance constraints are always related to time (response time; utilization rate: proportion of time that a device is used; throughput: quantity of load executed per unit time), the information to be annotated will be either directly time or other magnitudes allowing the computation of times.

4.3 Information to be Annotated

In this section the information necessary to run the model will be described. To run a performance model there are three kinds of informations necessary describing:

- each one of the logical or physical entities constituting the system,
- each one of the transaction classes to be executed in the system,
- the workload scenario.

Obviously there will be some differences on the information to annotate depending on the level of design considered.

4.3.1 Entities

4.3.1.1. *Logical Design*

It is assumed that the logical design gives information concerning the use of each operation and that this use can be:

- by just one transaction,
- by a finite number of transactions simultaneously,
- by as many simultaneous transactions as necessary.

In all cases, it is necessary to annotate the time to execute the operation or the rate at which the quantity of service asked by the transaction (e. g. the length of a message in a communication system) allowing to estimate the service time. Also it is necessary to annotate all other informations concerning the internal behaviour of the logical entity as buffer size, algorithm used by the logical entity, internal synchronization procedures, etc.

In addition, in cases 2 and 3, the execution time or the rate can be load dependent, that is either the execution time or the rate can vary according to the total number of transactions executing simultaneously the operation.

4.3.1.2. *Physical Design*

In this level of design, the physical entities to be considered are CPU's, disks, communication systems, etc. However the quantity of information to be annotated is depending on how critical is the entity considered in the system performance. In consequence, alternatives in the information and default hypothesis are proposed for some devices.
The information to be annotated concerns the hardware and software characteristics of the components of the system:
CPU: Number of processing units; Scheduling discipline; Capacity of each processor; etc.
Disks: Policy of the control unit to solve the access conflicts. Rotational speed. If the disks are sectorized, number of sectors per track, otherwise statistical distribution of the number of registers per track. If the service time of each disk is considered independent of the load, service time distribution. If it is necessary to consider the physical structure and connection of the disk subsystems, it is necessary to annotate: Number of disks in each subsystem and connection path. Statistical distribution and/or characteristics of the seek time. If the control unit has a subsidiary cache memory, read and write policies and hit rates.
Memory: In present systems this element is not normally a critical one and frequently can be ignored in the performance model. If it is necessary to consider the memory, the information to be annotated concerns mainly the memory management policy.
Operating System: It is necessary to define the scheduling policies used by it in managing all the queues organized in the system. From the quantitative point of view, it is necessary to describe the overhead resource consumption.
Communication System: For this device it is very difficult to standardize the information to be annotated due to the large variety of configurations and solutions that this element can have. In consequence the information to be annotated can be very different depending on the type of solution adopted in this subsystem. Informations to be annotated are, for example: Type of communication media (private line or leased line or switched line). Transfer rate and transfer policy. Topology of the network and the terminals connection. Fixed or adaptive routing. If the communication media is a public packet switched network: the agreed response time. If the communication media is of ethernet type: cable length, nominal rate, number of nodes connected and retransmission policy. If the communication media is of token ring type: the cable length, nominal rate, number of stations connected and delay in bits per station.

4.3.2 Transactions

Although most of the informations needed to build the performance model come from the

strict design information, there are some supplementary informations to be annotated in order to clarify the model building and to run the model.

As it has been said, to model the performance of a RT & E System it is convenient to group the transactions in classes with similar characteristics from the point of view of its behaviour (similar sequence of accesses to logical entities or similar resource consumption). Also it is not necessary to consider all the transactions but only those being representatives of the system workload.

4.3.2.1. *Logical Design*

For each class of transactions it is necessary to annotate:

- The start event and how these events are generated (throughput input or interactive input coming from the environment; responsive output or timed output of another system component).
- The estimated logical profile, that is the sequence of visits to the different operations to complete the execution. The service asked to each one can be depending on some parameter associated to the transaction or not. In the first case it is necessary to annotate the corresponding parameter measured in terms of time or any other unit able to estimate the service time. This service time (or parameter allowing its computation) can be defined by a fixed quantity or by a statistical distribution with its characteristics. In order to reduce the number of transaction classes the logical profile can be described by means of routing probabilities, that are the probabilities to access an operation after exiting from another one.
- The synchronization relations between each transaction class with other transaction classes or events in the system (locking the use of an operation, fork and join operations, etc.).
- The end event.

4.3.2.2. *Physical Design*

For each class of transactions it is necessary to annotate:

- The start event and how these events are generated (throughput input or interactive input coming from the environment; responsive output or timed output of another system component).
- The estimated physical profile, that is the sequence of visits to the different devices to complete the execution. The service asked to each one can be depending on some parameter associated to the transaction or not. In the first case it is necessary to annotate the corresponding parameter measured in terms of time or any other unit able to

estimate the service time. This service time (or parameter allowing its computation) can be defined by a fixed quantity or by an statistical distribution with its characteristics. In order to reduce the number of transaction classes the physical profile can be described by means of routing probabilities, that are the probabilities to access a physical entity after exiting from another one.

- The synchronization relations between each transaction class with other transaction classes or events in the system (locking the use of a resource, fork and join operations, etc.).
- The end event.

4.3.3 Working Scenarios

The goal of this annotation is to define the workload submitted to the system establishing the mix of transaction classes and their timing.

For each class of throughput input class of transactions it is necessary to annotate the statistical distribution of the interarrival time and its characteristics.

For each class of interactive input class of transactions it is necessary to annotate the statistical distribution of the time between successive outputs and inputs and its characteristics.

For each class of responsive output class of transactions it is necessary to annotate the statistical distribution of the time between successive outputs and inputs (thinking time) and its characteristics. Also it is necessary to annotate the number of simultaneous users generating each class of transactions.

For each class of timed output class of transactions it is necessary to annotate the statistical distribution of the time between successive events starting the transaction to be output and the characteristics of the statistical distribution.

5. Guidelines for building Performance models

When a performance model of a design is to be built, two kinds of problems appear:

- the model itself to be used,
- the data needed to run the model to obtain the performance estimations.

In the following sections several rules and recommendations are given in order to ease the model building process.

5.1 For the Model Selection and Construction

Recommendations:

To have a library of generic parameterized models of frequently used devices or subsystems:

- For each device or subsystem it is convenient to have several models with different level of detail and oriented to either analytical or simulation modelling.
- The parameters are to be chosen as close as possible to physical quantities describing the device and that can be found in its description.
- These generic models can represent either elementary devices (CPU, disk, ethernet, etc.) or subsystems (disk subsystem with its control unit) or computer systems (elements of computer series); obviously the models of each level can be built with the models of the previous levels.
- It is convenient that the models of all levels have been tested in order to debug them as well as to match their correctness with the corresponding devices or subsystems.

To have a library of basic parameterized models of design elements:

- For each device or subsystem is convenient to have several models with different level of detail and oriented to either analytical or simulation modelling.
- The parameters are to be chosen as close as possible of physical quantities describing it and that can be found in its description.
- Associated with the library it is convenient to have a set of rules to help in the simplification of models obtained from the juxtaposition of the basic models of design elements (collapsing of several elements, replacing by simpler models, etc.) in order to do not get too huge models.
- It is convenient that the models of all levels have been tested in order to debug them as well as to match their correctness with the corresponding design elements.

To have a library of generic parameterized models of frequently used operating systems:

- For each operating system it is convenient to have several models with different level of detail and oriented to either analytical or simulation modelling.
- The parameters are to be chosen as close as possible to the accessible parameters of the operating system and that can be found in its description.
- These generic models can represent the operating systems either as an increment of the resource consumption of the workload programs or as an independent task using the system resources according to the instantaneous system load.
- It is convenient that the models of all levels have been tested in order to debug them as well as to match their correctness with the corresponding devices or subsystems.

Rules:

- The accuracy of the model results is more sensible to the data than to the model accuracy.
- For a first exploration of the system behaviour it is convenient to use an analytical model even if it is necessary to force some hypothesis.

- For equivalent accuracy models it is more preferable an analytical one than a simulation one due to the shorter computing and debugging times.

5.2 For the Design Annotation

Recommendations:

To have a data base of reasonable values (resource consumption, probabilities, distribution laws, characteristics of operating systems, etc.) used in previous models of real systems duly matched with the measurements of the implemented system; in this way the designer has some help in the process of design annotation.

- For each filed value it is convenient to have its history, say the series of estimated values and the measured one as well as some indication of the environment in which the value has been estimated or measured.
- The data base must include some intelligent rules or accessing paths in order to ease the selection of the corresponding value.

To have a tool allowing the measurement of data corresponding to the implemented elements to substitute the estimated values and to have a better performance estimation as long as the different elements are implemented.

Rules:

- If no previous estimation exists, it is convenient to explore the performance of the system when the variable gets a reasonable range of values.
- It is better a bad estimation than no estimation.

6. Conclusion

Performance constraints are of high criticism in RT&E systems world. The performance evaluation modelling activity is a mean by which systems designers validate quantitatively their work. However this activity is often taken very late in the software life-cycle and may lead to costly revisions if the performance constraints are not met.

The COMPLEMENT Performance Task Force team is integrating this activity within conventional design methods in order to tackle both functional and performance requirements, in an efficient manner. This integration needed to characterise performance constraints to make them familiar to the designers.

The Performance Information Model introduces annotations of design entities. A new discipline is required from the designers: the estimation of the likely performance criteria. However, in return performance evaluation modelling and analysis provide a great help when validating designs.

Feasibility studies have been realised in the COMPLEMENT framework:

- integration of annotated LOTOS and Queueing Networks [VHBG92] [IRI37] [IRI39],
- derivation of performance models from HOOD annotated descriptions [MAT33],
- building generic performance models for the MASCOT design method [JLP92] [PLJ92].

The estimation process of performance constraints during the design of RT&E systems is only the first step in the integration approach. The next step is to define transformation rules from annotated method specific designs to model building languages, in order to automate as much as possible the modelling activity.

7. Bibliography

[DAS85] B. Dasarathy, *Timing Constraints of Real-Time Systems: Constructs for Expressing them, Methods of Validating them*, IEEE Trans. on Software Engineering, Vol. 11, No. 1 (January 1985), pp. 80-86.

[DAV90] A. M. Davis, *Software Requirements - Analysis and Specifications*, Prentice-Hall, 1990, 516 pp.

[ESA91] ESA PSS-05-0 Standard Issue 2 (February 1991).

[IRI37] *First Approach to annotate METAIS LOTOS Specifications*, by Y. Douma, O. Hjiej, A. Valderruten, COMPLEMENT report, April 1992.

[IRI39] *Queueing Network Models derived from annotated METAIS LOTOS Specifications*, by A. Valderruten, O. Hjiej, COMPLEMENT report, April 1992.

[JLP92] K. Jackson, A. Llamosì, R. Puigjaner, Performance Models under the Mascot Method. Proceedings of the Software Engineering Research Forum SERF'92. Melbourne Florida 1992. Editor: R. Rodriguez.

[MAT33] *Feasibility Study of HOOD annotations*, by S. Ayache, E. Conquet, COMPLEMENT report, Sept. 1991.

[PLJ92] R. Puigjaner, A. Llamosì, K. Jackson, *Generic Performance Models of the Basic Structures of the MASCOT Design Method*, University of the Balearic Islands, Spain, 1992.

[TSG11] *COMPLEMENT Glossary*, COMPLEMENT members, January 1992.

[UIB77] R.Puigjaner, A. Llamosì, Performance Annotations for MASCOT, COMPLEMENT report, Dec. 1992.

[TSG22] *Information Model for RT&E Systems*, COMPLEMENT Task 1A team, December 1991.

[VHBG92] A. Valderruten, O. Hjiej, A. Benzekri, D. Gazal, *Deriving Queueing Networks Performance Models from Annotated LOTOS Specifications*, 6th Int. Conf. on Modelling Techniques and Tools for Performance Evaluation, Edinburgh, 16th to 18th September, 1992.