TOWARDS A PERFORMANCE MODELLING ENVIRONMENT:

NEWS ON HIT

H. Beilner, J. Mäter, and N. Weißenberg

Informatik IV
Universität Dortmund

ABSTRACT

HIT is a comprehensive software tool supporting the model-based evaluation of computing system performance. HIT models exhibit a highly structured view of the systems to be assessed, based on (vertical) functional hierarchies and (horizontal) modularization as employed in modern software engineering and hardware architecture approaches. Analysis of HIT models is provided by analytic-algebraical, analytic-numerical, exact and approximate techniques and by discrete-event simulation. Both model description and model analysis utilize the model structure for convenient problem specification and efficient evaluation, respectively. Particular emphasis is placed on decomposition and aggregation options and on a mixed (heterogeneous) use of different analysis techniques. Great care is also employed with respect to tool handling aspects. This paper describes recent extensions of the HIT modelling environment and illustrates it by way of an extended office model example.

1. INTRODUCTION

In an age of continuously shrinking development and usage cycles of computing systems' hardware and software, computer support during system design, realization and operations appears a must. The type of desirable support should (ideally) relate to all domains of requirements for these systems, to their functionality and correctness, their performance, their reliability and availability, etc. It seems that considerable effort will still have to be spent before such an ideal, integrated support can become operational: At present, a satisfactory level of (say: design-) support has not even been attained within the various, individual requirement domains (hardware and software correctness, performance, reliability), and sizeable R&D activities aiming at improved, domain-specific support are underway. On the other hand, it is certainly not too soon to acknowledge the integrated point of view as a major guideline and to try and move towards this target, from within the various requirement domains.

With these ideas in mind, and with the particular emphasis on performance modelling as one, necessary type of design support, a corresponding project was launched in Dortmund, in 1983, which has since focussed on the development of the modelling tool, HIT. HIT adheres to the sketched guideline by exhibiting a system/model description interface as tailored upon the prevalent structural view of computer architecture and software engineering (functional/virtual machine hierarchies, modularization, ADT/object-oriented fashions of thinking). The present status of HIT proves that the basic evaluation power of a performance modelling tool is not hampered by this approach: Simulative, analytic-algebraical, analytic-numerical, exact and approximate analysis techniques can in fact be fed from that structured description interface. Quite to the contrary, the description structure can be taken advantage of by structuring model evalua-
tion processes accordingly, i.e. by using problem-specific description structures as the basis for decomposition/aggregation steps and corresponding structured (homogeneous or heterogeneous) evaluations displaying considerably improved, total evaluation efficiency.

HIT has of yet been made available in two major versions, a 0- and a 1-version, dated 1985 and 1987, respectively. Externally available references to HIT-0 include Beilner and Scholten (1985, in German), presenting the initial model world, and Wolf (1986, in French), concentrating in particular on the compilation techniques used. Special emphasis on simulative evaluation is placed in the HIT-1 reference, Beilner and Stewing (1987, in English). The present contribution is intended to highlight recent progress in both model description and evaluation techniques incorporated in HIT, and to also focus on various handling aspects, which have been considerably enhanced by the implementation of an object management facility, OMA. The latter facility introduces systematic storage and retrieval options for (partial) model descriptions and results and eases automatic (total) model configuration.

Right at the beginning, acknowledgements are due to NIXDORF Computer and to the German Federal Department of Research and Technology, BMFT, without whose major support the development of HIT would not have been possible. Thanks are also extended to the whole HIT development team of which only a few members will be mentioned in the body of this paper. Finally, reference must be given to other performance modelling tools such as RESQ (Sauer and McNair, 1984), QNAP (Veran and Potier, 1984), MAOS (Jobmann, 1985), etc., which have not gone unnoticed while developing HIT. Also, the experience from our own, earlier tools, COPE (Beilner and Mäter, 1984) and NUMAS (Müller, 1984) has most certainly been exploited.

2. MODEL WORLD AND MODEL SPECIFICATION

For the purpose of describing a computing system, in other words: of specifying a corresponding model, HIT employs the familiar view of a hierarchy of functional levels and function-realizing layers. A level is characterized by a set of functions, called SERVICEs, which it PROVIDEs to its environment; the environment (a layer to be constructed on top of this level) may USE these functions within certain algorithmic patterns; naming these patterns, calling them services and providing them for use to some yet "higher" environment obviously establishes a next higher level (or a part of it). This scheme serves for constructing almost arbitrary vertical structures. To avoid misunderstandings: Levels are not necessarily system-wide; the hierarchy needs not be strict; recursion across the level structure is not allowed; the total structure is not necessarily a tree.

The sketched vertical level/layer structuring scheme is complemented in HIT by means for horizontal structuring, employing the equally familiar modularization view. A level (a set of services provided for use) is regarded as export interface of some machinery which is to realize this level's services, is regarded as export interface of a "machine", in HIT terminology. That machine can be (horizontally) structured into a set of modules, called COMPONENTs, where every component is responsible for the realization of a subset of the machine-provided services, thereby partitioning the level-specific service set accordingly. Starting from the other end: Every component provides specific services; a machine includes some set of components and provides the union of services provided by these components.

Returning to the services (introduced above as patterns of using some lower "level"; recognizable now as patterns of using the services of a machine): A set of different services can, of course, be described for an individual machine interface, where such a service set is called a "load pattern", in HIT. Combining a machine with a machine-consistent load pattern and declaring particular services as provided (externally visible, exported) is equivalent with constructing a component, in the above sense. For this construction process, HIT insists on an explicit linking (REFERring) of names of USEd services (called upon in the load pattern) to names of PROVIDEd services (established by the machine) in order to introduce a certain naming independence and to thereby facilitate team efforts at constructing large models.

In total, the specification of a HIT model will consist of a series of more local, mutually fairly independent activities, each concerned with the declaration of a machine, with the pro-