Configuring Complex Multi-Sensor Test Bed Systems

Dietmar Prisching
dietmar.prisching@avl.com
AVL List GmbH
Graz, AUSTRIA

Michael Paulweber
michael.paulweber@avl.com
AVL List GmbH
Graz, AUSTRIA

Bernhard Rinner
rinner@iti.tu-graz.ac.at
Institute for Technical Informatics
TU Graz, AUSTRIA

Abstract – Test bed systems are important tools for research and development. They are connected to the physical system under test via numerous sensors and actuators. Due to the ever increasing requirements in performance and functionality automated support in the configuration of the test bed's hardware and software components is desired. This paper presents our approach to evaluate and predict the performance of highly-configurable embedded software and presents a case study for a configuration of the PUMA Open test bed system.

Keywords: Test bed systems; real-time; performance estimation; PUMA Open; embedded system

I. INTRODUCTION

Test bed systems are nowadays essential tools for research and development especially in the automotive domain. Functionality and complexity of such tools have been significantly raised over the last years in order to fulfill the ever increasing requirements. A typical test bed system is connected to the physical system under test via numerous sensors and actuators and automatically performs various measurements and test procedures. The most challenging requirements for such a test bed system are, therefore, the integration of various hardware and software components, the real-time data processing and the configuration of the overall system. Due to the complexity of modern AS support in the configuration of the hardware and software components is required. The determination of performance characteristic is an important precondition for a configuration support.

In this paper we present our approach to model, evaluate and predict the performance of highly-configurable embedded software (HCES) in an automotive test bed system. The determination of performance parameters such as computation time is important in order to check whether the (real-time) requirements have been satisfied. Our performance model is based on specific scenarios (use cases) that are most relevant for a performance evaluation. The timing properties of the HCES are computed using response time analysis. Performance data of the test bed system configuration is acquired by measurement and stored in a database for reuse.

We have evaluated our approach on the PUMA Open test bed system. PUMA Open is targeted for the design and test of engines, transmissions and power trains. As the experimental results demonstrate we are able to model and evaluate the performance of the numerous PUMA Open configurations. This performance estimation is now used in the design and implementation of new configurations and helps to reduce their development time.

The remainder of this paper is organized as follows: Section 2 introduces the PUMA Open automation system. Section 3 briefly discusses related work. Section 4 presents our implemented performance evaluation and prediction method. Section 5 presents results from a case study and Section 6 concludes this paper.

II. PUMA OPEN AUTOMATION SYSTEM

PUMA Open is an automation system (AS) (Figure 1) for the development and test of engines, transmissions and power trains. PUMA Open has been designed as an open platform in the sense that it is based on standardized interfaces for data acquisition and communication as well as modular hardware and software components. This supports the extension and configuration of the AS.

Figure 1: The PUMA Open Instrumentation and Test System for Engines, Transmissions and Power Trains

Figure 2 presents a part of the PUMA Open instrumentation interface. It supports various bus systems such as IEEE1394, CAN, Profibus, RS232, T-Link(RS485) and Ethernet to connect sensors (multi sensor system), actuators and several measurement devices with the computer system. In a typical configuration about 50-

60 sensors and actors are attached to the engine under test. Sensor connected via at the IEEE1394 include: PT100, several high temperature sensors NiCrNi, DMS measurement, current, voltage, pressure and speed sensors.

More complex physical parameters are determined with measurement devices such as:

- fuel consumption measurement
  S733 (gravimetric), PLU (density, volume),
  S735 (mass)
- oil consumption measurement
  O403 (level)
- diesel measurement
  S415 smokemeter
  O439 Opacimeter
  472 Smart Samples
- Emission measurement
  CVS devices (Concentration)
  Fast response devices
- Sensiflow air consumption measurement
- BlowBy compression bypass amount

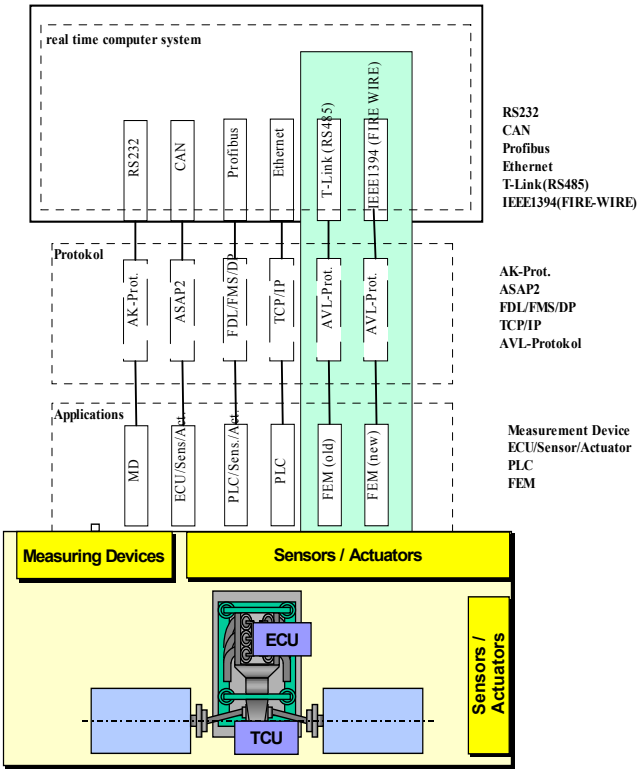

Figure 2: Part of the PUMA Open instrumentation interface

The PUMA Open is a complex object-oriented system and it combines both real-time (RT) and non real-time (NRT) computing on the same platform (PC solutions). The NRT part is based on the operating system Microsoft® Windows NT/2000 and the RT part is based on the Windows real-time extension INtime. Its main components are:

- PUMA Operating System
- Control and Automation Functions
- Data Acquisition and Storage
- Multi-Level Safety Monitoring
- Graphical User Interface

The PUMA Open real-time computer system (RTCS) is based on a layered architecture. At the bottom lies the real-time operating system (RTOS) INtime. The next layer is ARTE (AVL Real-Time Environment), and on top of the architecture are the various PUMA sub-systems. ARTE provides all real-time services that are required by the other components. The ARTE services can be used via a standardized interface. All real-time tasks have priority over any non real-time processing in PUMA.

ARTE can also be seen as a wrapper over the INtime RTOS. ARTE simplifies the development of real-time software components for the developers by providing customized real-time services as a library. Currently, several processes of the real-time operating system INtime realize ARTE. ARTE's main features are:

- A task system with cyclic and acyclic tasks with up to 256 priorities.
- Fast data transfer via system variables allocated in shared memory.
- Non real-time access via ARTE system variables.
- Analysis and diagnosis functions to support the development process.

We distinguish between two types of configurations of the PUMA Open AS. The PUMA Open AS can be assembled using a (sub)set of all possible sub-systems and interfaces. Since this set of components does not change during operation of the AS, this is referred to as static configuration. During operation of the AS, several components may be activated and shut down, i.e., the AS is operated in different modes of operation. This is referred to dynamic configuration of the AS.

At a high-level view the PUMA Open AS has three different modes of operation. In the monitoring mode, only the PUMA operating system and the graphical user interface are activated. In this mode, the system is initialized, the system parameter are checked and loaded as well as the I/O sub-systems are booted. In the manual mode, the data acquisition and storage as well as the multi-level safety monitoring sub-systems are also activated. In this mode, the test and the engine (technical process) parameter are checked and loaded; the engine monitoring, the data acquisition, the limit monitoring and the post processing are activated. Finally, in the automatic mode all sub-systems are activated and an automatic test-run is executed.

III. STATE OF THE ART

At the market multiple modeling techniques for computer system exist. Since a modeling techniques must reflect the properties of the modeled systems we emphasize on two up-to-date approaches that apprehend our requirements in the best case. These are the software performance models from [SMI97] (SPE*ED) and from [HIG01].

Both models are starting with an analytic performance model. The analytic performance model from [SMI97] is based on queuing network models [JAI91]. The analytic

performance model of [HIG01] contains process service time, process dispatch time, queue waiting time for each process and I/O as function of transaction rate.

In a second step it is necessary to model or describe system specifics. In [SMI97] the users' views of the system model are scenarios. Software scenarios are assigned to the facilities that execute the processing steps. Performance data are provided from the user, which specify software resource requirements for each processing step. In order to determine the performance of the overall system the following procedure can be used: First, the major functional scenarios (focus scenarios), which are important from a performance perspective, are identified. Second, a model of a system workload is established using several focus scenarios. From that, the different modes of operation can be constructed. Finally, the focus scenarios are evaluated resulting in performance statements.

[HIG01] works with a similar approach as [SMI97] on the topic of scenarios. The scenarios are described as transactions, and the user must identify and characterize the primary transactions from a performance perspective. However, the Performance data of the system are acquired through performance measurements and stored in a performance database.

Finally the analytic performance models and the models of the system specifics deliver the model results. At [HIG01] the results predicted by the model can be compared to those actually measured. SPE*ED [SMI97] produces analytic results for the software models, and an approximate, analytic MVA solution of the generated queuing network model. The results reported by SPE*ED are the end-to-end response time, the elapsed time for each processing step, the device utilization, and the amount to time spent at each computer device for each processing step. SPE*ED is intended to model software systems under development.

IV. PERFORMANCE EVALUATION AND PREDICTION

The ideas of both methods are practical for our approach. In comparison to [SMI97] and [HIG01] we start also with an analytic performance model. But our analytic performance is not based on a queuing network models. With queuing network models approximate solutions can be calculated. But for our project we need results as precise as achievable. Our analytic performance model is based on a *response time analysis (RTA)* derived from Burns [BUR93], [BUR01] and Bernat [BER02]. Response time analysis is an effective, simple and flexible technique that allows the modeling of most aspects of fixed priority real-time systems. [BUR94] applies an engineering approach to calculate the *worst case response time* (R_i) for each task (τ_i,) at the *critical instant* 0^2 [LIU73] (all tasks are released together). Our RTA model is based on the approach from [BER02] that considers multiple invocations of tasks, idle time at each priority level, task offsets and sporadic task invocations. We have extended this approach to quantify the interoperability of RT (INtime) and NRT (Windows) processing. Therefore, we include the operating system context switching (f_{OSCS}) of RT and NRT into the RTA formulation. In [PR03] we have evaluated the Windows-INtime interoperability. For a PENTIUM II (434 MHz) configuration the OS context switch time (C_{OSCS}) was determined as 5 µs. Thus, a high context switch rate has a significant influence on the system performance. With an extension of the formulation from [BUT97], the CPU utilization for a specific RT-processing (task set Π) is given by:

$$\text{Equation 1: } U_{\Pi} = \sum \frac{C_j}{T_j} + f_{OSCS} \cdot C_{OSCS}$$

Furthermore, RTA can be used to retrieve the NRT computation distribution in dependency on a RT processing load.

Concerning the system specifics we use the same approach with scenarios. Our case study PUMA Open is completely designed with the unified model language (UML) notation. Thus, the PUMA Open requirements at the development time were realized using UseCases. So we can derive the important scenarios from a performance perspective with the PUMA Open UML model. With extensive performance system measurements we will establish a performance database. One the one hand, we chose the same approach as [HIG01] to obtain performance data. One the other hand, only if the input parameters for the model are as precise as possible the calculated output is useful. Therefore, it is necessary to determine the worst-case execution time WCET of tasks at a high precision.

One approach is to measure the worst-case execution time like [HIG01]. Several aspects like processor properties (cache, pipelining) must be considered. Another approach relies on the calculation of the worst-case execution time (e.g. from C code). The dynamic setting of hybrid systems requires a research focus on an improved determination of the WCET, possibly by combining the mentioned approaches. Important aspects of WCET are run-time analysis (calculation methods, compiler integration and measurement procedure), research at the hardware system architecture and optimization of executed source code.

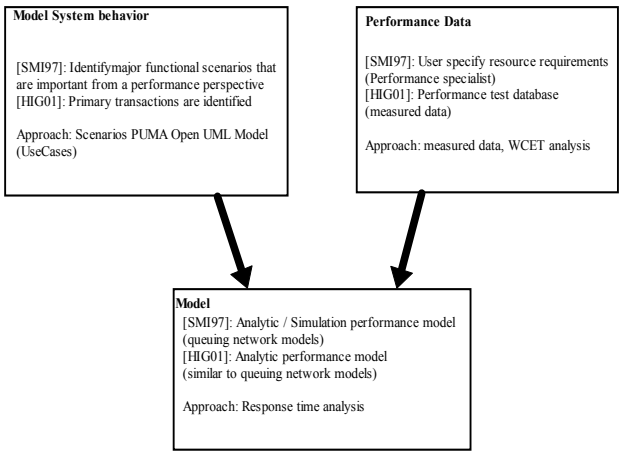

Figure 3: Overview about the software performance models from [SMI97] and [HIG01] vs. own approach.

Figure 3 illustrates an overview of the software performance models from [SMI97] and [HIG01] in comparison to our own approach.

V. RESULTS

Table 1 shows the task set for a PUMA Open scenario called 'Monitor – Cyclic Calculation'. This scenario declares that the PUMA Open is ready and it is important from a performance perspective. The PUMA Open configuration includes the I/O subsystems EMCON, CAN and 1394 (F-FEM). Performance data of the system tasks (Cyclic time T_i, computation time C_i and offset O_i) are measured with the *ProfileAnalyzer*. The ProfileAnalyzer allows a thread-based analysis of applications with RT (INtime) and NRT (Windows) processing on a single-processor platform. In [PR03] the ProfileAnalyzer is introduced in detail. Due to the task offsets a critical instant [BUR94] of the tasks never occurs [BER02]. The OS context switch rate and the utilization are derived as f_{OSCS} = 3.12 kHz, U_{Π}= 30.52 % ($U_{\Pi\ measured}$ = 31.92 %). The RTA was calculated for a hyperperiod of 40ms and the calculated worst-case response times of the tasks are listed in the rightmost column of Table 1. The total computation time of the Windows within the hyper-period of is ~35 ms and the maximum suspension time of windows is computed with 1320 µs. Given these numbers we can conclude that the RT processing works well and the Windows performance is not affected.

Table 1:

Name	T / ms	C / µs	Priority	O / µs	R / µs
Intr	1	34	50	0	39
dspt	40	34	132	0	34
tmr	10	7	134	0	80
RxTx	1	61	135	150	61
Read0	10	214	136	3500	214
DEB	8	20	137	0	105
ECAInterf	2	13	138	0	118
CFRecorder	4	15	139	0	133
CMyProtocol	1	7	140	0	140
TrgTask	1	23	141	0	224
CT1000	1	18	142	0	242
ECAControl	2	38	143	0	280
CT2000	2	25	144	0	305
IohSbc01	2	35	145	200	140
DAalive	50	29	146	0	369
DAWatchDEx	4	23	147	0	292
ADBDataEval	32	27	148	0	419
CCE	8	17	149	0	436
Sender	10	7	150	0	443
ABXC	32	18	151	0	461
CI	8	20	152	0	481
CT4000	4	28	154	0	509
CT5000	5	9	156	0	518
CT8000	8	10	158	0	528
CT10000	10	213	160	0	741
CA2CAnProto	10	98	161	0	839
CT16000	16	9	162	0	848
CT20000	20	13	164	0	861
CT32000	32	12	166	0	873
CT50000	50	271	168	0	1287
CDemoProto	50	33	169	0	1320

VI. DISCUSSION

In this paper we have presented our approach to evaluate and predict the performance in a complex test bed system. Performance evaluation and prediction is an important part for the configuration support not only of complex test bed systems but also for general computing systems. Although our performance model has been targeted to the PUMA Open test bed system, a general approach to model the performance of complex computing systems can be derived.

VI. REFERENCES

[BER02] BERNAT. G., 2002 Response Time Analysis of Asynchronous Real-Time Systems, University of York

[BUR94] BURNS, A., 1994: Preemptive Priority-Based Scheduling: An Appropriate Engineering Approach, Advances in RealTime Systems, 1994 225-248.

[BUR93] AUDSLEY, N., C.; BURNS, A., 1993 Applying New Scheduling Theory to Static Priority Pre.emptive Scheduling. Software Engineering Journal

[BUR01] BURNS, A., WELLINGS. A., 2001: Real-Time systems and programming languages. Addision Wesly 3rd edition, 2001

[BUT97] BUTTAZZO, G. C., 1997: HARD REAL_TIME COMPUTING SYSTEMS, Predictable Scheduling Algorithms and Applications, London, Kluwer Academic Publishers

[HIG01] HIGHLEYMAN, B., 2001: PERFORMANCE ANALYSIS - The Sombers Way http://www.sombers.com

[JAI91] JAIN, R.,1991: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling, John Wiley and Sons, Inc.

[LIU73] LIU, C. L., LAYLAND, J. W., 1973: Scheduling algorithms for multiprogramming in a hard-real-time environment, Journal of the Association for Computing Machinery, 20(1), pp. 46-61

[PR03] PRISCHING, D., RINNER, B., 2002: Thread-based analysis of embedded applications with real-time and non real-time processing on a single-processor platform, embedded world 2003 Congress, Nürnberg

[SMI97] SMITH, C., WILLIAMS, L., 1997 Performance Engineering Evaluation of Object-Oriented Systems with SPE*EDTM