PARTES: Performance Analysis of Real-Time Embedded Systems

Benjamin Gorry  Andrew Ireland  Peter King
School of Mathematical and Computer Sciences,
Heriot-Watt University, Riccarton, Edinburgh, EH14 4AS, Scotland
{beng, air, pjbk}@macs.hw.ac.uk

Abstract
The development of performance models from real-time system source code can be a time consuming process. It can be difficult to capture actual system timing values to be used in these models. The tool PARTES is introduced. PARTES facilitates the extraction of CSPL (C-based Stochastic Petri net Language) models from an ANSI-C program which has been annotated by the user. These models are then subjected to sensitivity analysis via the SPNP (Stochastic Petri Net Package). The results from analysis can be used to directly identify any potentially problematic timing areas in the original source code.

1. Introduction
Distributed real-time systems are everywhere in modern day society. These systems are often complex as time as a continuous factor can be difficult to account for. This can make verification of these systems an extremely difficult task. Testing techniques have traditionally been used to prove correctness of real-time software systems but are regarded as being very time consuming and laborious. This paper discusses PARTES (Performance Analysis of Real-Time Embedded Systems), a mechanized approach where model-checking technology is combined with performance modelling technology to facilitate the verification of real-time software systems.

PARTES builds on existing model-extraction technology. By using an annotation-driven approach to model development, models which contain actual system execution times are extracted from ANSI-C source code. Part of model verification and sensitivity analysis is then performed on models of the system. Sensitivity analysis provides an approach to focusing our attention on potential areas of concern within a real-time system. In this paper we describe the model development process and provide results from example case studies.

2. Model Development and Analysis
FeaVer (Feature Verification) [2] is a model-extraction framework which is used for the verification of distributed systems code written in ANSI-standard C. SPNP (Stochastic Petri Net Package) [5] is a tool which can be used to analyse CSPL (C-based Stochastic Petri Net language) models. Testing often proves to be expensive in terms of time and manpower. To eradicate this problem we can use performance modelling techniques; by representing the intrinsic characteristics of the timing elements of a system we can evaluate how the system will perform under a range of conditions [3].

SPNP allows us to perform sensitivity analysis [4] of a CSPL model. By changing the parameters of a model we can view the effect the changes have on the system we are analysing. We can thus tell how “sensitive” a system is to certain parameters. A common sensitivity measure is the MTTA (Mean Time To Absorption). In PARTES, the MTTA provides a measure of when an absorbing state will be reached. If an absorbing state is reached the model will progress no further.

The partial derivatives of parametrised areas are evaluated and the absolute value of a partial derivative indicates the magnitude of the variation [1]. If the measure of the partial derivative increases over time the parameter becomes more likely to contribute to the system reaching a state of absorption.

3. PARTES
PARTES (Performance Analysis of Real-Time Embedded Systems) builds on the approach taken by FeaVer by providing further guidance for abstraction choices. PARTES links model checking and performance modelling techniques to the source code level. By using a set of annotations, which are placed directly in the ANSI-C source code, a set of structured formal models which represent faithful abstractions of the real-time system are extracted. Figure 1 shows the structure of the methodology. Firstly,
a FeaVer test harness is populated from the annotated sections of source code. Using this test harness, a Promela model which can be analysed for functional correctness using SPIN is developed. Secondly, a performance test harness representing the structure of the annotated source code is extracted. This structure is used by PARTES in the development of CSPL models. However, models in other formal notations could be developed from the test harness structure. Thirdly, timing analysis is performed on the original source code. From the annotated source code, PARTES annotations act as timing hooks. These hooks are replaced with additional sections of source code which are used to gather mean timing values within 95% confidence limits. These values are used to represent the actual mean execution times of the source code when it is run on the target system hardware. Finally, by combining the details of the performance test harness with the results from timing analysis, timed Petri net models in CSPL form are developed. By using a number of Petri net sub-components, the structure of each model is developed. Areas of the model which contain timed transitions are assigned firing rates which correspond to the mean values gathered during timing analysis. The models developed using the PARTES-extracted test harness with FeaVer are used to identify any functional untimed areas of concern in the source code. For real-time issues, the CSPL models developed by PARTES are subjected to sensitivity analysis using SPNP. The partial derivatives generated during sensitivity analysis provide a measure of how the system will behave in relation to any timing constraints which have been imposed on it. If a model reaches an absorbing state it will deadlock. This indicates that a timing constraint in the model has been violated. Any area of concern which is identified can then be related directly back to the corresponding area of source code which has been annotated. This allows the engineer to focus testing efforts on any potentially problematic areas. PARTES has been evaluated using a number of case studies which exhibit a variety of system characteristics, the details of which are listed in Table 1. These vary in the number of system components, the number of distributed shared resources, the number of possible areas of timing concern, whether or not a system timing constraint has been violated, and whether PARTES has identified which area of timing concern is responsible. Results so far have been encouraging.

### References