ALL-TIMES - a European Project on Integrating Timing Technology

Jan Gustafsson,[1] Björn Lisper,[1] Markus Schordan,[2] Christian Ferdinand,[3] Peter Gliwa,[4] Marek Jersak,[5] Guillem Bernat[6]

[1] School of Innovation, Design, and Engineering, Mälardalen University, 721 23 Västerås, Sweden
[2] Vienna University of Technology, Argentinierstrasse 8/4/185.1, A-1040 Vienna, Austria
[3] AbsInt Angewandte Informatik GmbH, Science Park 1, 66123 Saarbruecken Germany
[4] Gliwa GmbH, Dollmannstr. 4, D-81541 München, Germany
[5] Symtavision GmbH, Frankfurter Str. 3 B, 38122 Braunschweig, Germany
[6] Rapita Systems Ltd., IT Centre, York Science Park, York, YO10 5DG United Kingdom

Abstract. ALL-TIMES is a research project within the EC 7th Framework Programme. The project concerns embedded systems that are subject to safety, availability, reliability, and performance requirements. Increasingly, these requirements relate to correct timing. Consequently, the need for appropriate timing analysis methods and tools is growing rapidly. An increasing number of sophisticated and technically mature timing analysis tools and methods are becoming available commercially and in academia. However, tools and methods have historically been developed in isolation, and the potential users are missing a process-related and continuous tool- and methodology-support. Due to this fragmentation, the timing analysis tool landscape does not yet fully exploit its potential.
The ALL-TIMES project aims at: combining independent research results into a consistent methodology, integrating available timing tools into a single framework, and developing new timing analysis methods and tools where appropriate.
ALL-TIMES will enable interoperability of the various tools from leading commercial vendors and universities alike, and develop integrated tool chains using as well as creating open tool frameworks and interfaces. In order to evaluate the tool integrations, a number of industrial case studies will be performed.
This paper describes the aims of the ALL-TIMES project, the partners, and the planned work.

This work was supported by the EU FP7 project ALL-TIMES (Integrating European Timing Analysis Technology, grant agreement no. 215068).

1 Introduction

ALL-TIMES (Integrating European Timing Analysis Technology) [1] is a research project within the EC 7th Framework Programme, with focus on correct timing of real-time embedded systems. The project started in December 2007 and will go on for 2 years and 3 months. The subject of ALL-TIMES has wide industrial relevance and there is a significant body of European research and experience in this area including a number of hi-tech SMEs. Timing measurement/analysis is vital for improving the reliability, performance, and efficiency of embedded systems. It helps to reduce the overall system costs by validating timing requirements, reducing the cost of development, and reducing unit costs in production.

Existing tools (commercial and academic) provide a set of powerful analysis techniques. Nevertheless there is a growing need, addressed in the ALL-TIMES project, for the integration of existing timing measurement/analysis techniques with the latest academic results in this area.

1.1 Concept, and General Objectives

A large class of embedded systems have safety, availability, reliability and performance requirements. This class spans across several areas, including automotive, avionics, telecom, and space systems. Common for these systems is the need to guarantee their correct behaviour as well as the satisfaction of non-functional requirements, in particular regarding timing.

The cost for delivering products with latent errors is staggering. For example, warranty costs in the automotive industry run 2% - 5% of sales [2]. Such levels have tremendous impact on the profitability. Timing is an essential dimension, notably one of the most difficult to analyze, and one that is generally only addressed late in the design cycle. In the automotive industry 50% of warranty costs today can be traced to software and electronics problems [2], for which about one third are reported to be directly related to timing issues. Current trends in industry of increased size and complexity of systems make the timing problem much more difficult, and in the future, consequences could be more dire.

Engineering timing correctness into a system requires treating timing as a first-class citizen throughout the software development process, including early stages: not as a property that is only addressed at the latest stage of this process. Early stage means before all code is available and all system design decisions have been made.

1.2 Timing Analysis

Timing analysis can be divided into *code-level analysis* and *system-level analysis*. *Worst-case execution time* (WCET) *analysis* and *scheduling analysis* are two exemplary techniques for the respective levels. WCET analysis computes an upper bound to the longest execution time that a fragment of code (e.g., a task) takes to execute in the worst case. Scheduling analysis determines the end-to-end execution time of a set of tasks. Code-level analysis thus assumes an isolated view of a fragment of code, whereas system-level analysis takes the complete system (one Embedded Control Unit (ECU), or even several ECUs including their communication interfaces) into consideration.

1.3 The Problem

Industry faces a difficult task to improve the reliability, safety, performance and resource efficiency of systems with regard to timing. The take up by industry of research and development in timing analysis is still low. There are several aspects to this problem; we classify them in the following themes:

– interoperation, scale and automation;
– integration into build process;
– education, and dissemination of knowledge.

Interoperation, scale and automation. Current technologies lack strong interoperability with other timing tools and compilers, making the adoption effort much more significant. Standard formats for representation and interoperability with tools are needed. Early efforts in the ARTIST2 [3] and INTEREST [4] projects indicate promising technologies.

Furthermore, some of the current analysis techniques have serious scalability issues relating to the size of the programs to be analyzed. Finally, a major issue is the full automation of the analysis process (the magic one-button solution) that is a pre-requisite for integration of timing analysis tools into current build processes.

Integration into build process. Industries that do need timing analysis tend to be conservative by nature. The adoption of a new technology implies change, which demands clear demonstrable benefits in perspective, and may be costly. Thus, new technology may be slow to deploy and integrate into the end-customer process. Especially difficult is the integration of a new technology into a project in progress. Academic prototypes are not usable by large companies that require commercial quality tools with long-term support guarantees. This results in large lead times to get these technologies to market, and consequently in the loss of market opportunity. A second issue is the large number of evolved procedures and constraints, which can make it difficult to exploit a technology simply because required input data cannot be obtained, or because parameters yielding the biggest improvements cannot be changed. This project will address these issues by targeted pilot studies to demonstrate the added value that investing in timing analysis tools brings to customers.

Education, and dissemination of knowledge. The timing analysis expertise is fragmented over universities and small companies, and its ability to reach a wider audience is limited. Large companies know that they need timing analysis, and are aware of the risks and consequences of timing errors in their products. However, few have knowledge of the available solutions, and even fewer have the capacity and will to take up the technology.

The current level of engineers, as regards knowledge of timing issues, is not sufficient. A recent example is the failure to establish a timing model in the current AUTOSAR [5] standard. An ALL-TIMES partner (Symtavision) is heavily involved in this activity. On the other hand there have been a good number of success stories on companies adopting these technologies. Unfortunately, the dissemination of these results is slow, partially due to restrictive corporate publishing policies.

The rest of this article is organized as follows: We present the main objectives of the ALL-TIMES project in Section 2 and its expected results in Section 3. In Section 4, we list the ALL-TIMES partners and their respective roles in the project. In Section 5, we describe the work packages. Finally, in Section 6, we draw some conclusions.

2 Main Project Objectives

The two principal project objectives are:

- to integrate different timing measurement/analysis tools using an open tool framework, and
- to achieve 25% improvement in the design time pertaining to timing issues.

One of the overall objectives of the project is the provision of new integrated tool sets for timing measurement and analysis targeted at the embedded real-time systems market. This relates to the advancement of new analysis techniques for integrated scheduling analysis, WCET analysis and timing measurement. In particular, the project will deliver new methods for timing measurement and analysis at both the system level and the code level in an open framework. An important aspect is a precise characterization of industrial requirements regarding timing. The project will provide a detailed requirements study in the first project phase.

A demonstrable 25% improvement in design time of embedded systems development can be achieved by enabling a quick, safe, automatic and efficient mechanism for deriving timing data instead of conventional manual and laborious approaches. The tool integration and analysis development work in ALL-TIMES aims at this. The fulfilment of the objective will be estimated by interviewing engineers participating in case studies on the efforts required to obtain timing estimates, and the quality of the results.

3 Expected Results

The ALL-TIMES project will:

- interface the different analysis techniques (three code-level and two system-level techniques) that are represented in the project;
- provide an open interface to integrate additional timing measurement/analysis techniques and tools, aiming at becoming a de-facto standard;
- provide solutions for timing analysis/estimation in early design phases as part of design-space exploration and architecture optimization.

4 Partners, their Tools, and their Roles

Two partners in ALL-TIMES are university groups:

- The WCET group at Mälardalen University - MDH (coordinator) [6]
- The SATIrE group at Vienna University of Technology - TUV [7]

The other four partners are SMEs. They will contribute to the development, evaluation and exploitation of different parts of the project:

- AbsInt Angewandte Informatik GmbH - ABS [8]
- Gliwa GmbH - GLI [9]
- Symtavision GmbH - SYM [10]
- Rapita Systems Ltd - RPT [11]

The partners, and the tools they contribute to the project, are shortly described below. We also briefly indicate the tool integrations that may be considered.

4.1 Mälardalen University

The WCET group at Mälardalen University works with methods and tools for WCET analysis for real-time systems. Specifically, they have developed a tool (SWEET) for analysis of programs written in C. Its modular tool architecture consists of three major phases:

1. A flow analysis phase, where bounds on the number of times instructions can be executed are derived, given the program code and possible input values.
2. A low-level analysis phase, where bounds on time it might take to execute instructions are derived, given the program object code and the architectural features of the target hardware.
3. A WCET estimate calculation phase, where the costliest program execution path is found using information from the first two phases.

The analysis phases communicate their results through well-defined data structures. The current research topic of the Mälardalen group is flow analysis. An annotation language can be used to constrain input data values.

4.2 Vienna University of Technology

The researchers involved in this project in the compilers and languages group at TU Vienna are concerned with the design, implementation, and application of programming languages, program analysis and optimization, and tools for embedded systems. Specifically they have developed the Static Analysis Tool Integration Engine (SATIrE) and integrated components of different program analysis tools. The design philosophy of SATIrE is the integration of analysis tools such that the results can always be used by all other integrated tools, enabling the composition of arbitrary tool chains. A plug-in mechanism for user-defined components enables connections to other external tools. SATIrE offers the following integrated base components:

– EDG C/C++ Front End
– LLNL-ROSE C/C++ intermediate representation
– ROSE C++ Unparser
– Program Analysis Generator (PAG) from AbsInt
– Annotation Parser & Mapper
– Annotation Generator
– Generator & Parser for external representation of AST
– Loop Optimizer (part of LLNL-ROSE, ported from Fortran D)

SATIrE currently allows to address all features of C++ with Exceptions being the only open issue. It supports all features of EC++. For C most features including some dialects are supported. The mapping of analysis information through different intermediate levels is supported by user-defined analysis-information transformers.

Based on SATIrE, the WCET tool TuBound [12] is being developed for computing the worst-case execution time of C programs by static analysis. The TuBound approach combines source-level analysis and code-level analysis in the presence of compiler optimizations.

4.3 AbsInt Angewandte Informatik GmbH

aiT is AbsInt's family of WCET analyzer tools. aiT WCET Analyzers statically compute tight upper bounds for the worst-case execution times (WCET) of tasks in real-time systems. They directly analyze binary executables without any need for instrumentation, and take the intrinsic cache and pipeline behavior into account.

The analyzers employ abstract interpretation to determine estimations for the WCETs of basic blocks. Integer linear programming (ILP) is used to derive a worst-case program path and an overall WCET estimation from the basic block WCET estimations. A graphical user interface supports the visualization of the worst-case path and the interactive inspection of all pipeline and cache states at arbitrary program points.

aiT's results are valid for all inputs and each execution of a task. aiT can be run interactively via a graphical user interface (GUI). The fields in the GUI can be filled with appropriate values, which may be stored in a project file. Alternatively, an existing project file can be loaded. aiT can also be started in simple batch mode with a project file.

4.4 Gliwa GmbH

Gliwa develops debugGURU, which is a framework for measuring and debugging timing related aspects of embedded software. The target code gets instrumented to gather timing information at run-time. This information is either processed "on the fly" by the target or transferred to and interpreted/visualized by a PC. Since debugGURU supports various target interfaces such as CAN, Nexus or KWP2000, measuring is possible not only in a development environment but

also “on the road”.
There are several plug-ins available that are easy to integrate into a system which supports debugGURU, for example:

timeGURU measures reliable run-time information about tasks, interrupts, processes, and/or any piece of code.

memGURU monitor memory accesses and consumption

delayGURU examines how much time is left in a task or an interrupt for additional functionality, useful for example during the development of embedded systems.

4.5 Symtavision GmbH

SymTA/S, developed by Symtavision, stands for Symbolic Timing Analysis for Systems. SymTA/S focuses exclusively on system timing and performance. Detailed functionality is abstracted, and only those properties that impact timing are modeled. The main advantages of this approach are: efficient modeling; unrivalled analysis speed; applicability in early design phases (when functions have not even been implemented); flexibility and independence of specific hardware and software.

SymTA/S is not a single, monolithic tool but rather a flexible and extensible tool suite. SymTA/S performs scheduling analysis for CPUs with RTOSes, buses with arbitrating protocols, and systems consisting of multiple resources (CPUs and buses). SymTA/S calculates resource loads, worst-case response times for tasks scheduled on CPUs, worst-case transmission times for messages sent via shared buses, end-to-end latencies and compares these values against user-specified constraints, e.g., deadlines. Additionally, SymTA/S has powerful exploration and sensitivity analysis modules for optimization of electronic architectures and scheduling. SymTA/S can be used early on in architecture definition and contracting phases, and continuously throughout the design until timing is verified as part of sign-off.

The core package is the SymTA/S analysis engine. The analysis engine provides all the basic functions to design and analyze the timing in a system, regardless of the internal implementation of the resources. It focuses on the interfaces between resources, where input-output timing and buffering are of central concern.

For the analysis of individual resources, SymTA/S has an interface to component libraries. The analysis engine integrates these local resource performance models into a global, system-level analysis model, and solves it. The analysis engine together with one or more component libraries allows quick modeling, configuration, and analysis of the performance and timing – from a single resource all the way to a distributed system including complex functional and architectural dependencies.

4.6 Rapita Systems Ltd

RapiTime is a software toolkit that provides a unique solution to the problem of worst-case execution time analysis and performance profiling, a solution that works for complex software running on advanced embedded microprocessors. RapiTime is a comprehensive performance analysis and WCET tool. It supports software written in C and Ada. It is compatible with industrial-scale programs from a few KBytes to millions of lines of code, and works with virtually every 8, 16, and 32-bit embedded microprocessor on the market, including those with advanced hardware features.

RapiTime contains the following main functions:

Performance Profiling. View high and low water marks, examine how different functions contribute to the average, longest, and shortest execution times, and locate performance bottlenecks at the root of throughput problems.

Code Coverage Analysis. Identify code coverage omissions, assess the coverage necessary for WCET analysis, check if the worst-case path has been followed during testing, and more.

Worst-Case Execution Time Analysis. Determine accurate worst case execution times, visualize the contribution of each function to the overall worst-case, examine worst-case execution frequencies, identify code on the worst-case path, and explore the variability in execution times due to hardware effects.

Targeted Optimization. Identify worst-case hotspots, select the best opportunities for optimization via advanced code metrics, see the difference between code that contributes the most on average, and code that contributes the most to the worst-case. Assess the headroom available to add new functionality.

Report Viewer. Eclipse-based, interactive access to data. Configurable views of worst-case, high water mark, and average-case behavior. Code metrics and comparisons. Search and sort facilities to highlight hotspots. Call-tree views of program structure and worst-case path. Graphical analysis of execution time distributions.

4.7 Possible Tool Integrations

Figure 1 indicates the possible integrations between timing analysis tools that may be considered within ALL-TIMES.

5 Work Packages

The project is divided into four work packages. We now briefly describe these.

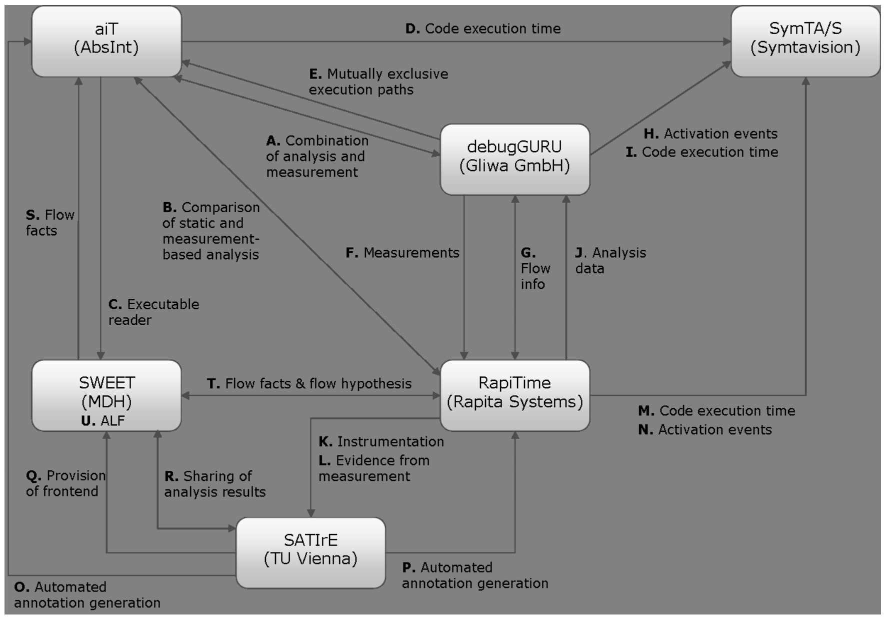
aiT (AbsInt)
SymTA/S (Symtavision)
debugGURU (Gliwa GmbH)
SWEET (MDH)
U. ALF
RapiTime (Rapita Systems)
SATIrE (TU Vienna)
D. Code execution time
E. Mutually exclusive execution paths
A. Combination of analysis and measurement
H. Activation events
I. Code execution time
B. Comparison of static and measurement-based analysis
S. Flow facts
F. Measurements
G. Flow info
J. Analysis data
C. Executable reader
T. Flow facts & flow hypothesis
M. Code execution time
N. Activation events
K. Instrumentation
L. Evidence from measurement
Q. Provision of frontend
R. Sharing of analysis results
P. Automated annotation generation
O. Automated annotation generation

Fig. 1. Possible integrations between timing analysis tools.

5.1 Work Package 1: Requirements

The aim of this work package is to identify the particular requirements for ALL-TIMES. These requirements appear on two different levels:

- requirements on timing analysis tools in general, and
- requirements on interfaces between tools.

The first level requires an identification of relevant use cases. These will come from (potential) end-users of timing analysis technology, in particular in the avionics and automotive industry. The SME partners all have customers in these areas, and representative use cases will be collected from some of these. In this process, it is possible to apply the "Mälardalen model" (see Section 5.4), and involve M.Sc. students in the collection and analysis of use cases.

The second level is of a more technical nature, and requires close interaction between tool experts.

Identification of use cases. The project needs to focus on most promising use cases. Project partners already have visions and concepts for use cases during different design stages. These use cases can serve as a starting point for discussion with end users. In this process, those use cases that are both valuable from the end users' perspective, and realistic from the project partners' perspective, will be identified, elaborated, and prepared for implementation.

Along with the use cases, an initial evaluation of timing analysis tools will be conducted. This investigation will elaborate the strengths and weaknesses of the respective tools, for different use cases, leading to a methodology to decide on the right tool for a given development phase.

General requirements on timing analysis tools. An initial estimate of the most important factors influencing the choice/combination of timing analysis/performance verification tools includes:

- Criticality of timing constraints
- Design stage (early estimation vs. late verification)
- Established design flow and hence availability / type / quality of input data

This needs to be verified and refined based on the identified use cases. The identified use cases need to be refined into technical requirements and design steps. On system level, one crucial aspect is the availability of input data required for scheduling analysis. This data will come from the different tools and techniques present in this project. The combination of test-based, tracing, semi-formal and formal approaches will enable to identify and demonstrate best fits for each of these techniques. Aspects to be considered:

- Accuracy of analysis
- Ease of obtaining the required input data
- Refinement from early, abstract models to later, more detailed models
- Roundtrip engineering / product lines / product evolution
- Architecture alternatives
- Different contexts and corresponding system behavior
- Tool interface requirements

The requirements on the code- and system level tools, in order to communicate with each other, need to be examined. For code level tools, the various tool characteristics result in specific input requirements and possible output. Broad room in this examination will for example be given to the import of measurement data in analysis tools and the communication of the results of source level analysis to analyzers working on binary code, to list but a few. The requirements on the communication of timing estimates from code level to system level will also be examined.

As a starting point, existing tool integration technologies will be reviewed (e.g., the XML timing cookies developed in INTEREST [4] or AIR, the ARTIST2 Intermediate program Representation for WCET analysis tools [3]) to assess these w.r.t. possible adaptation/extension for ALL-TIMES purposes. An appropriate solution will be specified.

5.2 Work Package 2: System-level Integration

This work package addresses reliable integration of multiple functions sharing a processor in a real-time system. For this, system-level analysis takes the complete system into consideration (whereas code-level analysis in Work Package 3 assumes an isolated view of a piece of code). The key to system-level integration is to assure schedulability of the system under all relevant conditions. The different timing analysis techniques in ALL-TIMES will be combined to determine system schedulability, with the goal to exploit the strengths of the different techniques, to avoid their weak points, and to overcome their limitations. The work package consists of three parts: interface, early-stage methodology and late-stage methodology. Here, early-stage and late-stage refer to design stages of a system that the user of an integrated tool chain is designing.

Development of the system-level tool interface will start in parallel with development of the early-stage system-level analysis methodology. The interface will then be refined and extended together with the development of the late-stage system-level analysis methodology. The rationale for this ordering (early stage before late stage) is the lack of detailed system information at an early stage. Therefore, a relatively simple interface will be sufficient, and the emphasis should be on speed and flexibility of the integrated tool chain. In the second step, the interface will be enriched to allow exchanging a larger variety of detailed system data available in later design stages.

System-level interface specification for timing analysis techniques. An open interface to combine different timing analysis techniques (scheduling analysis, WCET analysis, simulation, test, tracing, . . .) will be specified. The interface specification will be rich and flexible enough to allow combining timing analysis techniques in different ways depending on a specific design situation, to exchange data between tools at different levels of granularity and detail, and to iterate between different techniques for refinement of analysis results. Specifically, the interface must be suited for both early design stages and late design stages.

Early-stage system-level timing analysis methodology. A methodology for system-level timing analysis will be developed that exploits the strengths of different timing analysis techniques (scheduling analysis, WCET analysis, simulation, test, tracing, . . .) during early design stages. The goal is to enable a user of an integrated-tool chain-specific solution.

The methodology will include execution time estimation for software components on alternative processors as well as performance estimation using scheduling analysis and sensitivity analysis. The latter will allow a user to assess how much room there is for estimation errors and how much flexibility remains for later changes.

Late-stage system-level timing analysis methodology. A methodology for system-level timing analysis will be developed that exploits the strengths of different timing analysis techniques (scheduling analysis, WCET analysis, simulation, test, tracing, . . .) during late design stages. The goal is to enable a user of an integrated-tool chain to verify system timing on a level of quality and reliability not achievable by any single technique.

The methodology will combine the various techniques:

– to determine worst-case response times and response jitter, response time distributions and other important performance measures of a system
– to obtain tight analysis bounds by considering correlations between functions and events in different system contexts and scenarios.

5.3 Work Package 3: Code-Level Tool Integration

Code-level analysis assumes an isolated view of a piece of code whereas system-level analysis takes the complete system into consideration. The ALL-TIMES project will consider three different approaches to code-level analysis: Measurement of execution time (GLI), measurement-based (or hybrid) analysis (RPT), and static analysis on binary level (ABS) and source/intermediate level (MDH, TUV). The focus of this work package is to combine these approaches in an optimal manner, to exploit the strengths of the different methods, to avoid their weak points, and to overcome their limitations.

Incorporating time measurement data. The purpose of this work is to improve static analyzers by using the results of time measurements. This will be done in the following directions:

– Comparison of the statically computed longest path with measured data to identify the unwanted inclusion of error cases in the statically computed longest path. Once identified, the error cases can usually be excluded by a manual user annotation.
– Adaptation of the different tools to measurement methods with different number and position of measurements points in a program. For example, AbsInt's aiT can directly handle basic block measurements, but the results of less fine-grained measurement require some extensions. The result of this work will include a common format to specify all kinds of timing measurement results.

Source-level analyses. The micro-architecture analysis has to consider the very details of a processor implementation and therefore works on the binary program representation. A tool like AbsInt's aiT also tries to determine auxiliary information such as upper bounds of loop iterations on the binary level. Yet better results usually can be expected from source code analyses. One of the goals of the project is to overcome the limitations of considering only one level.

Examples of analyses that can be promising on source level involve the determination of loop bounds and recursion depths, possible values of function pointers, (non-)accessed variables of a function/task, and path exclusions. Measurement-based WCET analyses can usually do without such analyses. Yet an important aspect is quality of the measured data. Analyses like the ones enumerated above can give hints on the reached coverage of measurements. The purpose of this work is to create analyses using an industrial strength front-end for C/C++, to integrate the results of source code analyses as performed by

TUV's and MDH's tools into binary-level and measurement-based tools, and to develop a worst-case execution time estimator for programs in C/C++ source code. This estimator will use some parameters to configure a virtual processor so that it resembles real processors.

Code-level timing analysis in early design stages. Code-level timing analysis currently requires executable code as well as a detailed model of the target processor for static analysis or actual hardware for measurements. This means that all current code-level techniques can be applied only relatively late in the design, when code and hardware (models) are already far developed. Yet timing problems becoming apparent only in late design stages may require a costly re-iteration through earlier stages. Thus, we are striving for the possibility to perform code-level timing analysis already in early design stages.

Choosing a suitable processor configuration (core, memory, peripherals, . . .) for an automotive project at the beginning of the development is a challenge. In the high-volume market, choosing a too powerful configuration can lead to a serious waste of money. Choosing a configuration not powerful enough leads to severe changes late in the development cycle and might delay the delivery.

Currently, to a great extent this risky decision is taken based on gut feeling and previous experience. Our goal is to provide a family of tools to assist in the exploration of alternative system configurations before committing to a specific solution. Our approach requires that (representative) source code of (representative) parts of the application is available. This code can come from previous releases of a product or can be generated from a model within a rapid prototyping development environment.

To achieve the task, we will extend AbsInt's family of aiT WCET analyzers. aiT WCET analyzers statically compute tight upper bounds of worst-case execution times (WCET) of tasks in real-time systems, taking into account the cache and pipeline behaviour. They operate on binary executables and may take additional information in the form of user annotations. Through annotations the user provides information the tool needs to successfully carry out the analysis (e.g., loop bounds and recursion bounds that cannot be determined automatically, targets of computed calls or branches, etc.) or to improve precision.

aiT supports various cores and support is extended constantly. To be applicable in early design phases, the tool will be extended to make the cache and memory mapping completely parameterizable so that the user can experiment with different configurations. Furthermore, since performance guarantees at that stage are not as important as later in the development process, some precision will be traded against ease of use, speed and reduced resource needs. For example, source-code analysis will be integrated to enable certain information like unknown loop bounds to be determined from the source code, instead of asking the user for annotations.

Our early-phase code-level analysis will be integrated into the SymTA/S system-level architecture exploration analysis. The combination of code-level and system-level architecture exploration will lead to informed decisions with respect to which architectures are appropriate for an application.

5.4 Work Package 4: Validation and Dissemination

System- and code-level validation. The main purpose of the validation work is to compare and evaluate the different approaches to timing analysis subproblems and to develop a methodology for selecting the optimal method and tool for the work at hand. One important way to validate the methods developed within ALL-TIMES is to perform case studies together with industrial partners on selected problems. In this way, the ALL-TIMES methods and tools will be tested on industrial-strength systems, valuable feed-back from early users will be conveyed back to the developers, and end-user awareness will be raised early to solutions in the area of timing analysis.

The "Mälardalen model". The main performers of the case studies will be students on MSc level, supervised by experts at the company and from the ALL-TIMES project.

During a number of years, Mälardalen University has been performing case studies to evaluate timing tools in industrial settings. The purposes of the case studies have been, amongst other things, to evaluate the tools and methods on "real" code to get feedback for research and development. Results and evaluations made in the reports have resulted in research reports [13] and spawned new research and development activities. An additional advantage of the model is that it brings timing analysis into education; the M.Sc. students themselves become proficient with the latest timing analysis technology. It also helps disseminating the technology: both directly, to companies participating in the case studies, and indirectly by the students bringing their competence into their respective workplaces after graduation.

The time spent by the MSc student is typically used in the following way:

- introduction to timing analysis (university) – one week
- study of state of the art (MSc student) – a few weeks
- education in the used tool(s) (tool vendor) – one week
- introduction to the company and its code (company) – one week
- applying timing analysis to the code (MSc student) – 2–3 months
- writing report and presenting results to the other partners (MSc student) – one month

The ALL-TIMES project partners have an extensive network of industrial partners that will be enrolled during the case studies.

Dissemination. Dissemination and exploitation of the results from ALL-TIMES is aiming at spreading awareness of timing analysis and knowledge of the solutions (tools and methods) proposed by ALL-TIMES. The targets of dissemination are professionals working in the area of embedded and real-time computer systems, the research community (including PhD students), undergraduate students at universities, and the interested public.

6 Conclusions

The ALL-TIMES project is an ambitious effort to enable interoperability of timing tools from leading commercial vendors and universities in the EC. The project will develop tool chains using open tool frameworks and interfaces. These integrated tool chains will be evaluated using case studies performed towards industrial end-user companies. The main goal will be a demonstrable improvement in the design time of embedded systems development.

References

1. ALL-TIMES: Homepage (2008) `www.all-times.org`.
2. IBM: News Web page. `http://www.ibm.com/news/be/en/2005/05/3102.html` (April 2005)
3. ARTIST2: Timing-Analysis Cluster homepage (2008) `http://www.artist-embedded.org/artist`.
4. INTEREST: INTEREST (2008) `http://www.interest-strep.eu/`.
5. AUTOSAR: Homepage (2008) `http://www.autosar.org/`.
6. Mälardalen University: WCET project homepage (2008) `www.mrtc.mdh.se/projects/wcet`.
7. SATIrE: SATIrE homepage (2008) `http://www.complang.tuwien.ac.at/markus/satire`.
8. AbsInt: aiT tool homepage (2008) `www.absint.com/ait`.
9. Gliwa: homepage (2008) `http://www.gliwa.com/e/home.html`.
10. Symtavision: homepage (2008) `http://www.symtavision.com/`.
11. Rapita: RapiTime WCET tool homepage (2006) `www.rapitasystems.com`.
12. Prantl, A., Schordan, M., Knoop, J.: TuBound - a conceptually new tool for worst-case execution time analysis. In: Proceedings of the 8th International Workshop on Worst-Case Execution Time Analysis. (July 2008)
13. Gustafsson, J., Ermedahl, A.: Experiences from applying WCET analysis in industrial settings. In: Proc. 10th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC2007), Santorini Island, Greece (May 2007)