IST-510255
EmBounded
Automatic Prediction of Resource Bounds for Embedded Systems

Specific Targeted Research Project (STReP)
FET Open

D39 (WP1): Summary Management Report (Year 3)

Due date of deliverable: 1st. March 2008
Actual submission date: 30th April 2008

Start date of project: 1st March 2005 Duration: 36 months

Lead contractor: St Andrews University Revision: 1.8

Purpose: Provide an overall summary of the status of the project at the end of Year 3.

Results: This deliverable is not intended to provide research results.

Conclusion: This deliverable has no conclusions.

Project co-funded by the European Commission within the 6^{th} Framework Programme (2002-06)			
Dissemination Level			
PU	Public		⁕
PP	Restricted to other programme participants	(including the Commission Services)	
RE	Restricted to a group specified by the consortium	(including the Commission Services)	
CO	Confidential only for members of the consortium	(including the Commission Services)	

Summary Management Report (Year 3)

Kevin Hammond <kh@dcs.st-and.ac.uk>

1 Executive Summary

The project has continued to progress well in year 3, with a number of key deliverables produced, and the major technical objectives achieved. The project is running slightly behind the original planned timeline, mainly as a result of late starts at some sites, and the need to invest additional effort in the key cost modelling and analysis technology that the project is developing, especially for real-time. To allow us to fully complete the objectives of the project and to produce the remaining deliverables in a relaxed manner, we have submitted a revised Description of Work (Annex 1) that extends the project to Month 42, and this is currently under consideration by the Commission. The deadlines reported in this summary management report are consistent with this revised description of work.

Minor changes were made to the workplan to take account of changes requested by the project reviewers, notably in Workpackages 8 and 9, which are now, respectively, focused more towards evaluation, and towards exploitation of our results in contexts other than Hume. Effort was also redirected in the short term from Workpackage 6 (resource certification) towards real-time analysis (Workpackage 3), and cost modelling (Workpackage 2). This has been successful in allowing us to make more rapid progress in these key areas. Work on certification is now progressing, and we anticipate that all technical objectives of this workpackage will be met as planned. Overall the project team has made very good progress on some hard technical problems, and is now in an excellent position to meet all the project objectives by Month 42.

2 Work Performed on the Project

In year 1, we initially undertook our requirements analysis (M1). We subsequently developed high-level formal models of resource consumption (M2), and completed work on the semantics-based translation (M6). We also developed testbed applications for real-time systems (M4).

In year 2, we produced prototype stack- and heap- analyses (M3 and M5). We also produced both prototype and refined WCET analyses (M7). We enhanced the initial testbed applications and developed new computer vision algorithms (leading towards M11), commenced work on resource certification (leading towards M9), and continued work on the compiler and model checker (leading towards M10). All partners contributed to the revised requirements analysis (D2,WP5).

In year 3, we completed work on worst-case execution time (WCET) case studies (D16), the assertion language for certification (D17), the Hume to HAM compiler (D23), machine code generators (D24), analyses integrated into the compiler (D25), translator to support model checking (D26), and real-time computer vision algorithms (D27).

We are now working on Deliverables D28 (validation of the cost model), D29 (refined machine level analysis), D32 (application to traditional language expressions), D33 (evaluation of Hume), and D34 (problem-solving environment).

St Andrews. In year 3, St Andrews worked jointly with LMU in improving the formal analyses for time, heap, and stack consumption from WP3/4, worked with AbsInt on improving timing information as part of the integration of high-/low-level analyses in WP5, worked with LASMEA and HWU on the design methodology for Hume, and has assisted LMU with validation of the cost model, as part of WP2.

Work at St Andrews in WP3 has concentrated on obtaining improved time cost information for a number of HAM instructions, on improving analysis quality so that tighter bounds on time and space usage are obtained, on enhancing the coverage of the analyses so that they may be applied to more complex examples (Deliverable D16), on improving robustness of the analysis, on significantly enhancing performance of the analyses, and on proving the formal soundness of the analyses.

As part of WP5, St Andrews undertook further timing work to obtain better information for a number of HAM instructions on the M32C, especially those exploiting floating-point arithmetic. St Andrews also incorporated type information into HAM instructions, and used this to refine the cost information we obtained. In collaboration with AbsInt, St Andrews is now working on the tighter integration between the high-level and low-level analysis to be reported in Deliverable D29.

Finally, St Andrews assisted LMU with the validation of the cost model (Deliverable D28) in WP2.

Heriot-Watt. Heriot-Watt continued to refine the Hume compiler, as part of WP7. In particular, they introduced further performance gains through using unboxed representations of data and by fully integrating type checking into code generation and other compiler phases. They have also consolidated a uniform release of the compiler. The compiler has benefited from a growing community of compiler users and we have noted a marked decline in bug reports. At the same time we have extended the reference interpreter to incorporate an experimental implementation of hierarchical Hume, enabling the nesting of boxes. Heriot-Watt further developed their formalisation of the box calculus for Hume and used it to perform a number of simple program proofs. To enable investigation of the applicability of Hume analyses to traditional languages (WP9), we have defined a small imperative language termed mini-C, defined translations from mini-C to both the expression and coordination layers of Hume, and built corresponding translators. Heriot-Watt has worked jointly with LASMEA on developing the Hume programming methodology as part of WP8. Finally, Heriot-Watt have completed their treatment of HW-Hume in Isabelle, and of coordination aspects of Hume, in particular cross-level transformation, in TLA.

LMU. In year 3, LMU worked on WP2, WP6, WP3 and incidentally on WP4. Work on WP2 focused on validating the formal cost model against actual execution costs. Work on WP3/WP4 focused on improving and enhancing the analysis produced as Deliverable D15 and on validating this against a series of examples, including ones taken from WP8. Finally work on WP6 has focused on the definition of assertion languages to allow certification for both Hume expressions and Hume boxes.

In WP2, LMU worked on the validation of the cost model (Deliverable D28). LMU instrumented the Hume to native code generator with cost-counting instructions, according to the previously defined cost model (Deliverable D4). Results from cost-counting were compared with measured costs for heap space, stack space and time consumption. Both simple expression-level programs and box-level applications were used as test examples. A comparison of counted and measured costs is currently in preparation as Deliverable D28.

Work on WP6 (led by LMU) focused on the definition of assertion languages for both the expression and the box level of Hume (Deliverable D17). A resource-aware operational semantics for Core-Hume has been encoded in Isabelle/HOL. For the assertion language, we use a shallow embedding of program properties on the expression level and the Temporal Logic of Actions (TLA) on the box level. In making progress towards the deliverable on certificates (Deliverable D21), a VDM-style program logic has been formalised and proven sound in Isabelle/HOL. Currently a specialised logic for reasoning

about resources is being defined on top of this general program logic. Certificates will then take the form of proofs in this specialised logic.

Finally, LMU worked jointly with St Andrews on improving the formal analyses for time, heap and stack consumption as parts of WP3/WP4. Specifically, LMU provided the translation module that transforms a Hume program into the intermediate code, which is used as input by the analyses. The analyses have now been integrated into the Hume compiler (Deliverable D25). In the course of validating the analysis results on realistic applications (Deliverable D16), this transformation process was extended to cover the full Hume language.

LASMEA. In year 3, LASMEA undertook two main sets of activities.

Firstly, the CyCab control system has been completed. This involved adapting the lane-tracking simulation program to use an actual camera mounted on the vehicle. Then the communications library for the CyCab was translated from C++ into Hume. This was tested in a manual system whereby a graphical interface was used to control the vehicle. Some timing problems and other real-time aspects of the system had to be resolved prior to incorporating this code into the control system. Finally, the entire system was built adding a single master command Hume box to tie the individual components together and route the data to the relevant parts of the program. A simple guidance system was implemented which uses the vehicle position estimates deduced by the lane-tracking to steer the vehicle according to the lane edges. This system has undergone preliminary testing around the campus at LASMEA.

Secondly, the evaluation of the Hume language has been started by developing the Hume programming methodology. This methodology is refinement-based and shows how to incorporate the resource analysis into the development of embedded and real-time systems. The technique has been retrospectively applied to components in the CyCab control system.

AbsInt. AbsInt budgeted only limited effort (0.5 month) for year 3 of the project. Work at AbsInt has therefore primarily focused on supporting St Andrews and LMU in deriving the low-level timing information that is required for use in WP5 and WP3. AbsInt has also looked at the problem to select some more powerful target architecture and is currently trying to set up a suitable hardware platform. This effort is funded mainly outside EmBounded.

3 Project Objectives and Major Achievements

3.1 Project Objectives

The primary *objectives* of the project are:

Objective 1 production of *formal models of resource consumption* in real-time embedded systems for very high-level programming language constructs (*achieved by:* M2, Month 7);

Objective 2 development of *static analyses* of upper bounds for these resources based on the formal models of resource consumption (*intermediate milestones:* M3, M5, M6, M7 *achieved by:* M13, Month 33);

Objective 3 provision of independently and cheaply verifiable automatically generated *resource certificates* for the space and time behaviour of software/firmware components that can be used to construct embedded software/firmware in a compositional manner (*intermediate milestone:* M9 *achieved by:* M14, Month 34);

Objective 4 validation of our analyses against complex real-time embedded *applications* taken from computer vision systems for autonomous vehicle control (*intermediate milestones:* M8, M11, M12 *achieved by:* M15, Month 34);

Objective 5 investigation of how these technologies can be applied in the short-to-medium term in more *conventional language frameworks* for embedded systems (*achieved by* M17, Month 36);

Objective 6 development of underpinning specification, implementation and support environment for the Hume language (*intermediate milestone:* M10, *achieved by:* M16, Month 36).

3.2 Recommendations from Previous Reviews

The reviewers made four recommendations:

- REC 1: clarify the real-time semantics and hardware interaction model.
- REC 2: WP8 to focus on evaluation aspects.
- REC 3: WP9 to include assessment aspects.
- REC 4: D7 to be updated.

We have dealt with these recommendations as follows:

- REC 1: we have produced a detailed response to the questions raised by the reviewers.
- REC 2: we have taken this on board, making changes to the WP8 workplan and enhancing the role of Deliverable D33. We also provided preliminary input on evaluation criteria, as requested (also in the form of a more substantial research paper that is in preparation).
- REC 3: we have also taken this on board: D32/D35 will explain how Hume technology relates to existing approaches.
- REC 4: we have revised D7 and resubmitted it.

4 Workpackage Progress

WP2: Resource Modelling

Workpackage Objectives

Objectives

- to develop an accurate *cost model* for the HAM (and/or for MPC 5xx assembler code), covering execution time, heap consumption and maximum stack size;
- to *validate the cost model* by measuring the resources consumed in HAM (or assembler) programs on the target hardware and comparing it to the consumption predicted by the cost model.

Starting Point for WP2

In year 1, we produced deliverable D4 (HAM Cost Model). This deliverable provides a formal specification of both the behaviour and the resource consumption of the HAM abstract machine. We extended the promised deliverable by providing a formal specification of the behaviour in addition to resource consumption information. In year 2, having completed our initial WP2 tasks, we switched attention to other workpackages, as planned.

Progress on WP2 in Year 3

We commenced and have now almost completed our validation of the cost model (D28). In order to achieve this validation, we have constructed a tool that allows us to derive cost information based on the theoretical cost model. We instrumented the Hume to native-code generator with cost-counting instructions, according to the cost model defined in Deliverable D4. We have compared information derived from this tool, against actual execution data for heap consumption, stack usage and time consumption. As example programs, we used both simple expression-level programs and box-level applications. A detailed comparison of counted and measured costs is currently in preparation as part of Deliverable D28.

The HAM cost model has been successfully validated against expression-level programs (exact on space; 13% – 38% over-prediction on time); box-level applications (exact on space; 18% – 38% over-prediction on time). We have measured all resources of interest: heap space consumption, stack space consumption and execution time, and found empirical evidence that the costs described by the cost model are within 50% of the measured costs on the M32C/85 for significant portions of the control algorithms developed in WP8.

Deviations from project workprogramme for WP2

In order to concentrate on key issues in WP3 and WP4, we have dropped the optional deliverable D22 from the revised Description of Work.

List of Deliverables (WP2)

Del. No.	Deliverable Name	WP No.	Date due	Actual/ Forecast del. date	Est. Person Months	Used Person Months	Lead Contractor
D4	HAM Cost Model	WP2a	7	12	6	6	LMU
D22*	Report on Abstract Hardware Description Language	WP2b	–	–	2	–	LMU
D28	Validation of the Cost Model	WP2c	37	37	4	8	LMU

*Optional Deliverable D22 has been deleted from the list of deliverables in the revised workplan.

List of Milestones (WP2)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M2	HAM Cost Model	WP2a	7	12	LMU
M12	HAM Cost Model Validated	WP2c	37	37	LMU

WP3: Real-Time Analysis

Workpackage Objectives

Objectives

- to develop a *worst-case execution time (WCET) analysis* for real-time embedded programs written in Hume;
- to phrase and validate the results of this analysis in terms of a formal cost model for Hume;
- to validate the results of the analysis using realistic examples.

Starting Point for WP3

By the end of year 2 of the project, we had made substantial progress on WP3. We have defined a formal analysis for worst-case execution time based on analysis of source programs, have undertaken significant real-time measurements on both the PowerPC and Renesas M32C/85 platforms that will be used to provide concrete of our formal analysis, have constructed a prototype worst-case execution time analysis, and have provided sample results for a number of benchmark programs. This work is described in deliverables D14 (WCET Analysis Report) and D15 (WCET Analysis Implementation).

Progress on WP3 in Year 3

In year 3, we developed a number of case studies for our worst-case execution time analysis, included in deliverable D16 (Case Studies for WCET Analysis), and which served as a basis for validating our real-time analysis. We made a number of improvements to the WCET analysis aimed at improving the quality of the timing information produced by the analysis, increasing the analysis performance, and improving coverage to allow us to deal with large-scale case studies from WP8. Our target was that the implementation of our WCET analysis should accurately predict upper bounds on the time

requirements for at least one of the programs provided by WP8. For simple programs, results should be within 50% of the real costs, provided their time consumption is linear in the input size of the program.

Deviations from project workprogramme

We spent slightly more time than originally planned on improving the analysis that was delivered in deliverable D15. Results from these improvements are reported in D16.

List of Deliverables (WP3)

Del. No.	Deliverable Name	WP No.	Date due	Actual/ Forecast del. date	Est. Person Months	Used Person Months	Lead Contractor
D14	WCET Analysis Report	WP3a	20	24	1	0.5	SA
D15	WCET Analysis Implementation	WP3b	20	24	16	20	SA
D16	Case Studies for WCET	WP3c	24	32	3	3	SA

List of Milestones (WP3)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M7	Prototype WCET Analysis	WP3a	20	24	SA
M8	Validated WCET Analysis	WP3b	24	32	SA

WP4: Bounded Space Analysis

Workpackage Objectives

Objectives

- to develop a *static analysis* to conservatively predict the maximum stack size of real-time embedded programs written in Hume;
- to develop a *static analysis* to conservatively predict the maximum heap usage for real-time embedded programs written in Hume;
- to phrase and validate the results of these analyses in terms of a formal cost model;
- to validate the results of these analyses with realistic examples.

Starting Point for WP4

As a precursor to work on bounded space analysis, we concentrated on producing deliverables D3 (Hume-HAM translation) and D12 (Hume Semantics) from WP7 (see below). We also produced an initial implementation of stack- and heap-analysis based on sized types which we intended to extend to form deliverable D13 by incorporating work from Jost's PhD thesis. We also made further progress towards deliverables D13 (and thus also D5 and D11) in the form of work on a heap, stack and time

cost model for the Hume source level derived from the Hume formal semantics (deliverable D12). This high-level cost model was provided as deliverable D12b.

We then constructed formal static analyses for determining heap- and stack-space usage for Hume (deliverables D11 and D5), based on the space costs defined in the formal Hume Semantics (deliverable D12). Our analysis consists of an elaborate type system, whose typings yield strict upper bounds on the Heap-space consumption of the typed terms, and builds on advanced work on amortised cost analysis, developed as part of Jost's PhD thesis.

We have completed prototype analyses for stack space usage and heap space usage. These have been combined into a single implementation (described in deliverable D13). We have confirmed the accuracy of this analysis against some simple test cases, and must now validate the analysis, both formally and informally, as part of deliverable D30.

Progress on WP4 in Year 3

As planned, work in year 3 has mainly focused on real-time issues in WP3. As a result of improvements in the underlying generic analysis for both space and time, some benefits have accrued in terms of accuracy of the analysis, performance and reliability.

Deviations from project workprogramme for WP4

The original project plan showed deliverables D5 and D11 both being produced in year 1. In order to provide a strong base for the work on the formal analyses, and in order to properly exploit the technical capabilities of the researchers employed on the project, we decided, however to concentrate initially on deliverables D3 (Hume-HAM translation) and D12 (Hume Semantics) from WP7. This meant that production of deliverables D5, D11 and D13 was postponed until year 2. Work on deliverable D13 took slightly longer than originally planned, reflecting increasing maturity of understanding of the analysis problem, and a desire to produce a single prototype implementation incorporating stack, heap and time metrics in a generic framework. This has, however, been partially compensated by reduced effort on the implementation of the time analysis (D15, WP3).

List of Deliverables (WP4)

Del. No.	**Deliverable Name**	**WP No.**	**Date due**	**Actual/ Forecast del. date**	**Est. Person Months**	**Used Person Months**	**Lead Contractor**
D5	Stack Size Analysis Report	WP4a	20	20	0.5	0.5	SA
D11	Heap Space Analysis Report	WP4b	14	23	0.5	0.5	SA
D13	Space Analysis Prototypes	WP4a/b	14	23	9	14	SA
D30	Validation of Space Analyses	WP4c	34	38	3	–	SA

List of Milestones (WP4)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M3	Prototype implementation of Stack Analysis	WP4a	24	24	SA
M5	Prototype implementation of Heap Analysis	WP4b	7	7	SA
M15	Validated heap and stack analyses	WP4c	42	42	SA

WP5: Adapting Low-Level Hardware Behaviour Analyses

Workpackage Objectives

Objectives

- integration of low- and high-level analyses, accounting for hardware details of standard microcontrollers

Starting Point for WP5

In year 1, we produced deliverables D5 (project requirements analysis) and D6 (Extended AIS design). Based on the requirements analysis, we decided to focus on the Renesas M32C/85 as our concrete target architecture.

In year 2, we commenced work on the major activity in this workpackage (WP5c), namely refining the machine-level analysis. Researchers from St Andrews and Ludwig-Maximilians-Universität travelled to Saarbrücken to obtain training on the AbsInt tools and to discuss how the analyses needed to be adapted to allow integration of high- and low-level information. As a result of these discussions, AbsInt extended its timing analyzer aiT for M32C to support the analysis of executables generated by the GCC compiler (previously only the IAR compiler was supported), so permitting the validation of alternative compiler technologies. We then commenced work on integrating high- and low-level analysis information. Finally, we performed a series of practical experiments using AbsInt's **aiT** tool to determine worst-case execution time analysis for Hume Abstract Machine (HAM) instructions. The results of this work have been published in an international research paper [BFHH07].

Progress on WP5 in Year 3

In year 3, we undertook further timing work to obtain better information for a number of HAM instructions on the M32C, especially those exploiting floating-point arithmetic. We also incorporated type information into HAM instructions, and used this to refine the cost information we obtained. We are now working on the tighter integration between the high-level and low-level analysis to be reported in Deliverable D29.

Deviations from project workprogramme for WP5

There have been no deviations from the planned workprogramme.

List of Deliverables (WP5)

Del. No.	Deliverable Name	WP No.	Date due	Actual/ Forecast del. date	Est. Person Months	Used Person Months	Lead Contractor
D2	Requirements Analysis	WP5a	1	6	4.5	2.5	AbsInt
D6	Extended AIS Definition	WP5b	9	9	4	4	AbsInt
D29	Refined Machine-Level Analysis	WP5c	42	42	15	13.5	AbsInt

List of Milestones (WP5)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M1	Completed Requirements Analysis	WP5a	1	6	AbsInt
M13	Working Real-Time Analysis	WP5b/5c	42	42	AbsInt

WP6: Resource Certification

Workpackage Objectives

The goal of WP6 is to provide formally verifiable certificates of bounded resource consumption for Hume source programs. To achieve this objective, a resource-aware program logic needs to be constructed on top of the formal semantics for Hume (which must reflect the costs of the execution on the HAM level).

Objectives

- to produce formally-verifiable *certificates guaranteeing bounded time and space usage* for HAM programs.
- to integrate these certificates with the time and space cost models for Hume developed in WP3 and WP4.

Starting Point for WP6

Work on WP6 commenced at the end of year 2. Building on experience of our work undertaken in the Framework V MRG project, which developed a framework for resource certification in a somewhat simpler, but mobile code setting, we decided to use a multi-layered-logic approach as we had done before.

Progress on WP6 in Year 3

The main step in the certification process involves mapping the results of the static analyses down to assertions in a specialised logic of the HAM, which can be proven by taking a syntax-oriented proof strategy. This mapping and a suitable definition of the specialised logic are the main components for milestone M9. In order to focus on the novel aspects in Hume, we first defined a sublanguage that exposes named higher-order functions and algebraic data structures. We then formalised resource properties for concrete Hume programs, using these to develop the assertion language for resource properties that was produced in Deliverable D17.

Deviations from project workprogramme for WP6

In order to allow focus on the key novel issues of real-time analysis for Hume, in year 3 of the project we have expended less effort than originally planned on the certification tasks leading to Deliverables D21 (Resource Certification) and D31 (Certification Case Studies). Although, as a consequence, we will have lost some technical depth, we do not anticipate that this will have an impact on the achievement of the general objectives of the project, or on the specific objectives of the workpackage.

List of Deliverables (WP6)

Del. No.	Deliverable Name	WP No.	Date due	Actual/ Forecast del. date	Est. Person Months	Used Person Months	Lead Contractor
D17	Assertion Language Report	WP6a	24	32	4	4.5	LMU
D21	Resource Certificate Report	WP6b	40	40	4	–	LMU
D31	Certification Case Studies	WP6c	42	42	6	–	LMU

List of Milestones (WP6)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M9	Prototype Resource Certificates	WP6a/b	28	40	LMU
M14	Resource Certification Examples	WP6c	34	42	LMU

WP7: Compiler and Systems

In WP7, we continued to refine the Hume compiler. In particular, we introduced further performance gains through using unboxed representations of data and by fully integrating type checking into code generation and other compiler phases. We have also consolidated a uniform release of the compiler. The compiler has benefited from a growing community of compiler users and we have noted a marked decline in bug reports. At the same time we have extended the reference interpreter to incorporate an experimental implementation of hierarchical Hume, enabling the nesting of boxes. Heriot-Watt further developed their formalisation of the box calculus for Hume and used it to perform a number of simple program proofs. Finally, Heriot-Watt have completed their treatment of HW-Hume in Isabelle, and of coordination aspects of Hume, in particular cross-level transformation, in TLA.

Workpackage Objectives

Objectives

- To develop the Hume language as a sound basis for building formally-verifiable models of resource usage;
- To provide a suite of research-quality compilers and tools for Hume.

Starting Point for WP7

In year 1, we produced deliverables D3 (formal translation) and D12 (formal semantics). We also refactored the prototype HAM implementation to improve performance and increase reliability, and have added a number of features and improvements to the prototype Hume to HAM compiler, including investigation of the HW-Hume level, targetting hardware/software codesign.

Finally, we had started work on a native code compiler for Hume with a view to significantly improving performance over the current HAM implementation. The compiler supported around 95% of the HAM instructions and could successfully compile the majority of the Hume benchmark application suite both for Linux and for stand-alone operation on the *Renesas M32C* development board recommended by LASMEA.

In year 2, we made major progress on the Hume to C compiler, delivering substantial increases in performance and widening coverage to almost all of Hume. We also made good progress on developing the Hume IDE to support interactive graphical program construction. In addition, we deployed TLA+ in Isabelle to model coordination behaviour, used TLA/TLC to prove a fundamental recursion to iteration transformation, elaborated a box calculus for reasoning more generally about coordination level transformation, and explored a hierarchical extension to Hume. Finally, we developed a formal model of HW-Hume in Isabelle, enabling machine-assisted correctness proof.

Progress on WP7 in Year 3

In year 3, Heriot-Watt continued to refine the Hume compiler, as part of WP7. In particular, they introduced further performance gains through using unboxed representations of data and by fully integrating type checking into code generation and other compiler phases. They have also consolidated a uniform release of the compiler. At the same time we have extended the reference interpreter to incorporate an experimental implementation of hierarchical Hume, enabling the nesting of boxes. Heriot-Watt further developed their formalisation of the box calculus for Hume and used it to perform a number of simple program proofs. Finally, Heriot-Watt have completed their treatment of HW-Hume in TLA+/Isabelle.

Deviations from project workprogramme for WP7

By reallocating deliverables D3/D12 to researchers at SA/LMU in year 1, we were able to gain approximately eight months additional development effort at HWU. This has allowed us to focus on improving the quality of the Hume to HAM compiler and other tools, notably a high-performance Hume to C compiler. These enhancements have included the ongoing construction of an "unboxed" version of the compiler, and other enhancements that have significantly improved the performance of the compiled code. We feel that this considerably improves the credibility of our research, and provides us with a strong base for longer-term exploitation. In our revised workplan, we have confirmed optional Deliverable D34 (Hume PSE).

List of Deliverables (WP7)

Del. No.	Deliverable Name	WP No.	Date due	Actual/ Forecast del. date	Est. Person Months	Used Person Months	Lead Contractor
D3	Formal Translation from Hume to HAM	WP7a	5	12	4	2.5	HWU
D12	Formal Semantics for Hume	WP7b	14	12	4	5	HWU
D23	Hume to HAM Compiler	WP7c	35	35	10	14.5	HWU
D24	Machine Code Generators	WP7c	35	35	2	6.5	HWU
D25	Analysis in Compiler	WP7c	35	35	2	2	HWU
D26	Translator supporting model checker	WP7d	35	35	10	5.5	HWU
D34	Hume PSE	WP7e	42	42	5	2	HWU

List of Milestones (WP7)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M6	Completed Semantics-Based Translations	WP7a/b	14	12	HWU
M10	Completed Hume Compiler and Model Checker	WP7c	35	35	HWU
M16	Completed Hume PSE	WP7d	42	42	HWU

WP8: Applications

Workpackage Objectives

Objectives

- to develop realistic embedded applications as cost modelling testbeds;
- to develop computer vision algorithms for controlling autonomous vehicles using Hume;
- to investigate the benefits and limitations of Hume as a means for programming real-time embedded systems.

Starting Point for WP8

In year 1, we produced deliverable D7 (real-time testbed applications). This deliverable was subsequently revised, and has been provided as a living document. We produced a simple lane-tracking algorithm, plus control system demonstrators.

In year 2, the simple lane-tracking application was completed. This involved upgrading to the new `humec` compiler, substantial debugging work and verification on the vehicle simulation program written for the purpose. In addition, several simpler versions of the code were written to allow the use of the analysis tools on individual code snippets. Mostly, this involved replacing calls to library functions

written in C with native (but less efficient) Hume code. We also undertook a real-time control project for the M32C microcontroller board. This was an inverted pendulum experiment for which the objective is to control the pendulum in the vertical position using an arm actuated by an electric motor using input from angular sensors attached to the pivots. The `humec` compiler output was modified and the Renesas C compiler suite used to generate a simple state-space control program with optimal gains computed offline. Controlling this device requires a maximum sample time of 10ms so this experiment constitutes a severe test of Hume's execution performance. Additionally, the stringent memory requirements of the board (32k RAM) required careful implementation and placement of memory. Finally, we started implementation of our lane-tracking application on the CyCab. Deliverable D7 has been substantially updated to reflect the new work.

Progress on WP8 in Year 3

In year 3, we first completed a large-scale application in the form of the CyCab control system. This involved adapting the lane-tracking simulation program to use an actual camera mounted on the vehicle. Then the communications library for the CyCab was translated from C++ into Hume. This was tested in a manual system whereby a graphical interface was used to control the vehicle. Some timing problems and other real-time aspects of the system had to be resolved prior to incorporating this code into the control system. Finally, the entire system was built adding a single master command Hume box to tie the individual components together and route the data to the relevant parts of the program. A simple guidance system was implemented which uses the vehicle position estimates deduced by the lane-tracking to steer the vehicle according to the lane edges. This system has undergone preliminary testing around the campus at LASMEA. This work was reported in Deliverable D27.

Secondly, we began our evaluation of the Hume language by expounding the Hume programming methodology. This methodology is refinement-based and shows how to incorporate the resource analysis into the development of embedded and real-time systems. The technique has been retrospectively applied to components in the CyCab control system. This work will be incorporated into Deliverable D33.

Deviations from project workprogramme for WP8

Due to the need to recruit suitable staff, work commenced in month 5 of the project. We were then, however, able to make rapid progress in year 1 on the testbed applications required as deliverable D7. Some effort was devoted in year 2 to producing revised versions of these algorithms; the remaining effort has been put into the development of real-time computer vision algorithms. In line with the recommendations made by the reviewers, in year 3 we have focused effort on providing large-scale testbed demonstrators to allow validation of our cost models and analyses in WP3, WP2 and shortly WP4. Also in line with the recommendations of the reviewers, we have refocused the workpackage to place more emphasis on evaluation of Hume rather than being principally concerned with applications.

List of Deliverables (WP8)

Del. No.	Deliverable Name	WP No.	Date due	Actual/ Forecast del. date	Est. Person Months	Used Person Months	Lead Contractor
D7	Real-time testbed implementations	WP8a	10	12	10	10	LASMEA
D27	Real-time computer vision algorithms	WP8b	29	32	30	21	LASMEA
D33	Hume Evaluation	WP8c	36	42	7	3	LASMEA

List of Milestones (WP8)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M4	Completed Real-Time Testbed Applications	WP8a	10	12	LASMEA
M11	Completed Computer Vision Algorithms	WP8b	29	35	LASMEA
M17	Completed Hume Evaluation	WP8c	36	42	LASMEA

WP9: Application to other Language Settings

Workpackage Objectives

Objectives

- to determine how our resource models and analyses may be applied to traditional programming languages for embedded systems.

Starting Point for WP9

Work on this workpackage commenced in year 3 of the project.

Progress on WP9 in Year 3

To enable investigation of the applicability of Hume analyses to traditional languages, we have defined a small imperative language termed mini-C, defined translations from mini-C to both the expression and coordination layers of Hume, and built corresponding translators.

Deviations from project workprogramme for WP9

There were no deviations from the planned workprogramme.

List of Deliverables (WP9)

Del. No.	Deliverable Name	WP No.	Date due	Actual/ Forecast del. date	Est. Person Months	Used Person Months	Lead Contractor
D32	Application to Traditional Settings Report (Expression forms)	WP9a	34	40	2	1.5	HWU
D35	Application to Traditional Settings Report	WP9b	36	42	4	–	HWU

List of Milestones (WP9)

Milestone No.	Milestone Name	WP No.	Date due	Actual/Forecast del. date	Lead Contractor
M17	Completed Hume Evaluation	WP8c	36	42	HWU

WP10: Dissemination and Public Awareness

Workpackage Objectives

Objectives

- to ensure proper dissemination of research results;
- to raise public awareness of the EmBounded project.

Starting Point for WP10

In year 1, we ran three project workshops: one kickoff technical/management workshop in March 2005 (Edinburgh); one technical workshop in November 2005 (Frankfurt); and one management workshop in February (Edinburgh). We produced two research papers: one describing the project, its aims and objectives [HDF+06]; and one outlining the application of linear heap analysis technology to object-oriented languages [HJ06b]. We constructed a web site (`http://www.embounded.org`) that is used to hold project documents and deliverables as described in the plan for use and dissemination of knowledge. We also delivered talks on Hume at over fifteen venues. In year 2, we ran three project workshops: one technical workshop in June 2006 (Saarbrücken); one open workshop in September 2006 (colocated with IFL 2006 in Budapest) and one technical/management meeting in Edinburgh in January 2007. In year 3, we ran one project workshop in Scotland in July 2007.

Publications We have produced twenty-eight research papers and articles [BCH+07a, FHF07, GCH+07, GMI07, GPMI08, LM07, HBH+07, BFHH07, HGMI07, HFH+06b, Ham05, HFH+06a, HDF+06, BCH+07b, MWH+06, HMV06, BH06b, BH06c, HvS06, HF06, FH06a, FH06b, DHL06, BKHB06, BH06a, SHFV07, HDF+06, HJ06a], plus the two general articles detailed below.

Web site We have constructed a web site (`http://www.embounded.org`) that is used to hold project documents and deliverables as described in the plan for use and dissemination of knowledge. Implementations of the Hume language, and prototype implementations of our analyses can also be accessed through this site.

Raising Public Awareness We have produced articles for our internal publicity departments describing the project in general terms. We have targeted industrially-based conferences and events in embedded systems. We published general articles in ERCIM News No 67 (special issue on "Embedded Intelligence", organised by DECOS) and in the 30th Anniversary issue of House Magazine, a specialist publication targeting UK MPs and government workers. In the coming year, we propose to take advantage of opportunities to engage further with the public, including events such as the UK's National Science Week.

Other Dissemination Activities We have given talks and presentations on the project work in various places, including: the International Symposium on Trends in Functional Programming (TFP 2007), New York; the International Symposium on Implementation and Applications of Functional Languages (IFL 2007), Freiburg, Germany; the ACM Symposium on Applied Computing (SAC '07), Seoul, Korea; the International Conference on Intelligent Robotics and Manufacturing Automation (IRMA '07), Venice, Italy; the European Symposium on Verification and Validation of Software Systems (VVSS '07); the International Workshop on Worst-Case Execution Time (WCET '07), Pisa, Italy; EuroTRUSTAmi 2007, Sophia-Antipolis, France; the European Symposium on Programming (ESOP 2006), Germany; the International Symposium on Trends in Functional Programming (TFP 2006), Nottingham UK;

the International Symposium on Implementation and Application of Functional Languages (IFL 2006), Budapest Hungary; the International Conference on Generative Programming and Component Engineering (GPCE 2006), Portland, Oregon, USA; the RTAS Workshop on Innovative Techniques for Certification of Embedded Systems (ITCES 2006), San Jose, California, USA; the International Workshop on Worst-Case Execution Time Analysis (WCET 2006), Dresden, Germany; the International Conference on Foundations of Software Technology and Theoretical Computer Science, Kolkata, India, 2006; the Asian Symposium on Programming Languages and Systems (APLAS, Australia); the SAE World Congress, Michigan, USA, 2006; the International Conference on Types (TYPES 2006), Nottingham, UK; the Workshop on Implicit Computational Complexity (GEOCAL06), Marseilles (France); the Scottish Programming Language Seminar Series, Glasgow (UK); the 2006 and 2007 EMRS/SEAS DTC Conferences, Edinburgh, UK; Université Paul Sabatier, Toulouse (France); Universidade do Porto (Portugal); The ASTReNet Workshop on Formal Aspects of Source Code Analysis and Manipulation (London, UK); Harvard University (USA); IFIP Working Group 2.11 (USA and Denmark); Laboratoire VERIMAG, Grenoble (France); INRIA-Futurs (France); Waterfall Solutions Ltd (UK); Kings College London (UK); International Workshop on MicroGrids 2006 (UK); Universitaet van Amsterdam (the Netherlands); the University of Linz (Austria); and RISC-Linz (Austria).

Del. No.	**Deliverable Name**	**WP No.**	**Date due**	**Actual/ Forecast del. date**	**Est. Person Months**	**Used Person Months**	**Lead Contractor**
D1	Project Presentation	WP10	6	6	1	0.5	SA
D8	First Project Workshop	WP10	12	12	–	–	SA
D9	Second Project Workshop	WP10	12	12	–	–	SA
D18	Third Project Workshop	WP10	24	24	–	–	SA
D19	Fourth Project Workshop	WP10	24	24	–	–	SA
D36	Fifth Project Workshop	WP10	36	36	–	–	SA
D37	Sixth Project Workshop	WP10	36	42	–	–	SA
D38	Project Web Site	WP10	36	42	8	2.5	SA
D40	Plan for Use and Dissemination of Knowledge	WP10	36	42	0	0	SA

Milestone No.	**Milestone Name**	**WP No.**	**Date due**	**Actual/Forecast del. date**	**Lead Contractor**
M18	End of Project	WP10	36	42	SA

5 Consortium Management

5.1 Project Status

Having formed a good basis to achieve the key project objectives by the end of year 2, in year 3, we capitalised on this, investing significant effort in enhancing, improving and validating our key cost models and static analyses. In order to account for the slightly staggered start to the project, and in order to allow additional effort to be deployed on real-time analysis in year 3, as suggested by the reviewers, we have submitted a revised description of work that extends the project into month 42. We believe we will have no difficulty in meeting all the stated objectives of the project in this timescale.

5.1.1 Project Timetable

Figure 1 is a front-lined barchart showing the current status of the project.

5.1.2 List of Participants

Partic. Role	Partic. No.	Participant name	Participant short name	Country
CO	1	The University of St Andrews	USTAN	UK
CR	2	Heriot-Watt University Edinburgh	HWU	UK
CR	3	Ludwig-Maximilians Universität, München	LMU	Germany
CR	4	Laboratoire des Sciences et Matériaux pour l'Electronique et d'Automatique Univ. Blaise-Pascal, Clermont-Ferrand	LASMEA	France
CR	5	AbsInt Angewandte Informatik GmbH, Saarbrücken	AbsInt	Germany

List of Participants

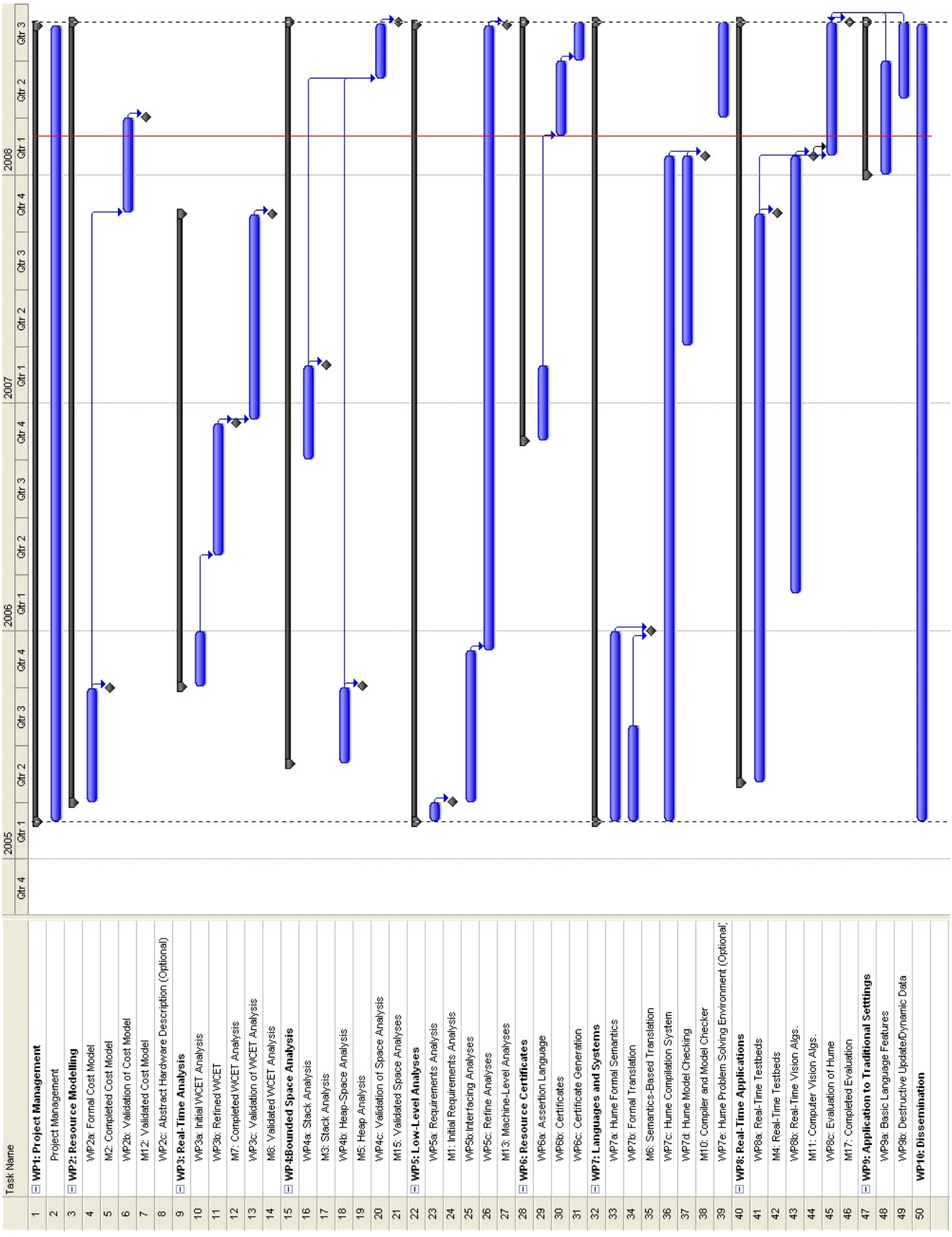

Figure 1: Frontlined Barchart (Year 3), Revised Project Plan

5.1.3 Budgeted and Actual Person-Months

Person-Month Status Table

Contract N°: 510255
Acronym: EmBounded
Period: Year 3

			Partner - Person-month per Workpackage						AC - own staff			
			Tot.	1	2	3	4	5	AC Tot.	1	2	3
WP1:	Project management and Monitoring	Actual WP total	5.5	5.5	0	0	0	0	3	3	0	0
		Planned WP total	9	9	0	0	0	0	3	3	0	0
WP2:	Resource Modelling	Actual WP total	13	4	0	9	0	0	3	1	0	2
		Planned WP total	12	4	0	8	0	0	3	1	0	2
WP3:	Real-Time Analysis	Actual WP total	25	14	0	11	0	0	3	3	0	0
		Planned WP total	20	13	0	7	0	0	3	2	0	1
WP4:	Bounded space analysis	Actual WP total	15	9	0	6	0	0	3	2	0	1
		Planned WP total	13	9	0	4	0	0	3	2	0	1
WP5:	Adaptation of low-level hardware behaviour analyses	Actual WP total	19.5	1.5	0.5	0.5	0.5	16.5	0.5	0	0	0
		Planned WP total	23.5	4	1	1	1	16.5	1	1	0	0
WP6:	Certification	Actual WP total	3	0	0	3	0	0	1	0	0	1
		Planned WP total	14	0	0	14	0	0	2	0	0	2
WP7:	Languages and Systems	Actual WP total	38	5	30.5	2.5	0	0	4.5	1	3	0.5
		Planned WP total	37	4	33	0	0	0	3.5	1	2.5	0
WP8:	Embedded Applications	Actual WP total	33	0	12.5	0	20.5	0	3	0	3	0
		Planned WP total	47	0	12	0	35	0	3	0	3	0
WP9:	Application to more traditional settings	Actual WP total	2.5	0	1.5	1	0	0	1	0	1	0
		Planned WP total	6	2	2	2	0	0	1.5	0.5	0.5	0.5
WP10:	Dissemination and public awareness	Actual WP total	5.5	5.5	0	0	0	0	2	2	0	0
		Planned WP total	9	9	0	0	0	0	1	1	0	0
		Actual total	160	44.5	45	33	21	16.5	23.5	12	7	4.5
Total Project Person-months		Planned total	190.5	54	48	36	36	16.5	21	11.5	6	6.5

5.1.4 Overall Status of Deliverables

The deliverables produced to date are shown below.

Deliverable	WP	Title	Owner	Due	Delivered
D1	WP10	Project Presentation	SA	6	6
D2	WP5	Requirements Analysis	LMU	1	6
D3	WP7	Hume-Ham Translation	LMU	5	12
D4	WP2	Cost Model (HAM)	LMU/SA	9	12
D5	WP4	Stack Size Report	SA	20	20
D6	WP5	Extended Ais	AbsInt	9	9
D7	WP8	Real-time testbed Apps	LASMEA	10	12
D8	WP10	Workshop 1	SA/LMU	12	12
D9	WP10	Workshop 2	SA/LMU	12	12
D10	WP1	Management Report 1	SA	12	12
D11	WP4	Heap Space Report	SA	7	23
D12	WP7	Hume Formal Semantics	SA	9	12
D13	WP4	Space Analysis Prototype	SA	20	23
D14	WP3	WCET report	SA	24	24
D15	WP3	Prototype WCET	SA	24	24
D16	WP3	WCET case studies	SA	24	32
D17	WP6	Assertion Language	LMU	24	32
D18	WP10	Workshop 3	SA/LMU	24	24
D19	WP10	Workshop 4	SA/LMU	24	24
D20	WP1	Management Report 2	SA	24	24
D23	WP7	Hume to HAM Compiler	HWU	35	35
D24	WP7	Machine Code Generators	HWU	35	35
D25	WP7	Analysis in Compiler	HWU	35	35
D26	WP7	Translator supporting model checker	HWU	35	35
D27	WP8	Real-time computer vision algorithms	LASMEA	35	35
D36	WP10	Workshop 5	SA	36	36
D39	WP1	Management Report 3	SA	36	38

5.1.5 Progress towards Final Objectives and Deliverables

The planned Year 3 deliverables are shown below. Our immediate targets are the production of case studies for the worst-case execution time analysis and the construction of the assertion language for formal certification. We must also work on validation of the cost model and space analyses, on constructing real-time computer vision algorithms, on properly integrating the high- and low-level analyses, and on providing support for model checking as part of the Hume compilation system.

Deliverable	WP	Title	Owner	Due
D21	WP6	Resource Certificate Report	LMU	40
D28	WP2	Validation of the Cost Model	LMU	37
D29	WP5	Refined Machine-Level Analysis	AbsInt	42
D30	WP4	Validation of Space Analyses	SA	42
D31	WP6	Resource Certification Case Studies	LMU	42
D32	WP9	Application to Traditional Settings Report (Expression forms)	HWU	40
D33	WP8	Hume Evaluation	LASMEA	42
D34	WP7	Hume PSE	HWU	42
D35	WP9	Application to Traditional Settings Report	HWU	42
D37	WP10	Workshop 6	SA	42
D38	WP10	Project Web Site	SA	42
D40	WP10	Plan for Use and Dissemination of Knowledge	SA	42

5.2 Coordination Activities

Project members have been cooperating closely, as required by the project. Representatives from all five partners have attended all project workshops and meetings. All partners are in regular email contact both on a case-to-case basis and through the project mailing list. In addition, HWU and USTAN are in regular phone and skype contact, and hold joint project meetings on a roughly monthly basis. USTAN and LMU are also in regular phone contact. There have also been a number of research visits: Scaife visited HWU on several occasions; Loidl visited HWU and USTAN on several occasions; Jost visited LMU on several occasions; Hammond visited LASMEA; and Hammond, Loidl, and Jost visited AbsInt.

We are investigating possible links with the following Framework VI projects: Mobius project (Mobility, Ubiquity and Security) at LMU and Edinburgh University, funded under the FET Global Computing Initiative; Aether (Self-Adaptive Embedded Technologies) through the University of Hertfordshire and the Universitateit van Amsterdam, funded under the FET Advanced Computer Architecture initiative; DECOS (Dependable Embedded Components and Systems); and the Artist2 Network of Excellence on Embedded Systems Design.

References

[BCH+07a] Armelle Bonenfant, Zenzhi Chen, Kevin Hammond, Greg Michaelson, Andy Wallace, and Iain Wallace. Towards resource-certified software: A formal cost model for time and its application to an image-processing example. In *ACM Symposium on Applied Computing (SAC '07), Seoul, Korea, March 11-15*, 2007.

[BCH+07b] Armelle Bonenfant, Zezhi Chen, Kevin Hammond, Greg Michaelson, Andy Wallace, and Iain Wallace. Towards resource-certified software: A formal cost model for time and its application to an image-processing example. In *ACM Symposium on Applied Computing (SAC '07), Seoul, Korea, March 11-15*, 2007.

[BFHH07] A. Bonenfant, C. Ferdinand, K. Hammond, and R. Heckmann. Worst-Case Execution Times for a Purely Functional Language. In *Proc. 2006Intl. Symp. on Impl. and Appl. of Functional Langs. (IFL 2006), to appear*. Springer-Verlag, 2007.

[BH06a] L. Beringer and M. Hofmann. Reading, Writing and Relations. In *Proc. Fourth ASIAN Symposium on Programming Languages and Systems (APLAS 2006), Sydney, Australia*, 2006.

[BH06b] Edwin Brady and Kevin Hammond. A dependently typed framework for static analysis of program execution costs. In *Implementation of Functional Languages (IFL) 2005*, volume 4015 of *Lecture Notes in Computer Science*, pages 74–90, Berlin/Heidelberg, 2006. Springer.

[BH06c] Edwin Brady and Kevin Hammond. A verified staged interpreter is a verified compiler: Multi-stage programming with dependent types. In *Proc. Conf. Generative Programming and Component Engineering (GPCE '06), Portland, Oregon*, Lecture Notes in Computer Science. Springer, 2006.

[BKHB06] N. Benton, A. Kennedy, M. Hofmann, and L. Beringer. Reading, Writing and Relations. In *Proc. Fourth ASIAN Symposium on Programming Languages and Systems (APLAS 2006), Sydney, Australia*, 2006.

[DHL06] C. Dax, M. Hofmann, and M. Lange. A Proof System for the Linear Time μ-Calculus. In *Proc. Conf. on Foundations of Software Technology and Theoretical Computer Science, Kolkata, India*, 2006.

[FH06a] Christian Ferdinand and Reinhold Heckmann. Improved Worst-Case Execution Time Prediction via Execution Contexts. In *Proc. Embedded World 2006 Conference,Nürnberg, Germany*, 2006.

[FH06b] Christian Ferdinand and Reinhold Heckmann. Verifying Timing Properties of Safety-Critical Embedded Software by Abstract Interpretation, 2006.

[FHF07] Christian Ferdinand, Reinhold Heckmann, and Bärbel Franzen. Static memory and timing analysis of embedded systems code. In Perry Groot, editor, *Proceedings of VVSS2007 - 3rd European Symposium on Verification and Validation of Software Systems, 23rd of March 2007, Eindhoven*, number TUE Computer Science Reports 07-04, 2007.

[GCH+07] Benjamin Gorry, Zezhi Chen, Kevin Hammond, Andy Wallace, and Greg Michaelson. Using mean-shift tracking algorithms for real-time tracking of moving images on an autonomous vehicle testbed platform. In *IRMA 2007, International Conference on Intelligent Robotics and Manufacturing Automation, Venice, Italy, November 23-25, 2007, 2007*, volume 25. World Academy of Science, Engineering and Technology (PWASET), 2007.

[GMI07] Gudmund Grov, Greg Michaelson, and Andrew Ireland. Formal verification of concurrent scheduling strategies using tla. In *3rd IEEE International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems*. IEEE, 2007.

[GPMI08] Gudmund Grov, Robert Pointon, Greg Michaelson, and Andrew Ireland. Preserving coordination properties when transforming concurrent system components. In *Coordination Models, Languages and Applications Track of the 23rd Annual ACM Symposium on Applied Computing*, 2008. To appear.

[Ham05] Kevin Hammond. Exploiting Purely Functional Programming to Obtain Bounded Resource Behaviour: the Hume Approach. In *Central European Functional Programming School (CEFP05)*, number 4164 in LNCS, pages 100–134, Budapest, Hungary, July 4–15,, 2005. Springer-Verlag.

[HBH+07] Christoph A. Herrmann, Armelle Bonenfant, Kevin Hammond, Steffen Jost, Hans-Wolfgang Loidl, and Robert Pointon. Automatic amortised worst-case execution time analysis. In *7th Int'l Workshop on Worst-Case Execution Time (WCET) Analysis, Proceedings*, pages 13–18, 2007.

[HDF+06] Kevin Hammond, Roy Dyckhoff, Christian Ferdinand, Reinhold Heckmann, Martin Hofmann, Steffen Jost, Hans-Wolfgang Loidl, Greg Michaelson, Jocelyn Sérot, and Andy Wallace. The embounded project (project paper). In *Proc. 6th Symposium on Trends in Functional Programming (TFP 2005), Tallinn, Estonia, 23-24 September 2005*, volume 6 of *Trends in Functional Programming*. Intellect, 2006.

[HF06] Reinhold Heckmann and Christian Ferdinand. Static Memory and Execution Time Analysis of Embedded Code. In *Proc. SAE World Congress, Michigan, USA*, 2006.

[HFH+06a] Kevin Hammond, Christian Ferdinand, Reinhold Heckmann, Roy Dyckhoff, Martin Hoffmann, Steffen Jost, Hans-Wolfgang Loidl, Greg Michaelson, Robert Pointon, Norman Scaife, Jocelyn Sérot, and Andy Wallace. Towards formally verifiable resource bounds for real-time embedded systems. In *Proc. Workshop on Innovative Techniques for Certification of Embedded Systems*, 2006.

[HFH+06b] Kevin Hammond, Christian Ferdinand, Reinhold Heckmann, Roy Dyckhoff, Martin Hofman, Steffen Jost, Hans-Wolfgang Loidl, Greg Michaelson, Robert Pointon, Norman Scaife, Jocelyn Sérot, and Andy Wallace. Towards formally verifiable WCET analysis for a functional programming language. In Frank Mueller, editor, *6th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis*, number 06902 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.

[HGMI07] Kevin Hammond, Gudmund Grov, Greg Michaelson, and Andrew Ireland. Low-level programming in Hume: an exploration of the HW-Hume level. In *Proc. Implementation of Functional Languages (IFL 2006)*, volume 4449 of *Lecture Notes in Computer Science*. Springer, 2007. To appear.

[HJ06a] Martin Hofmann and Steffen Jost. Type-based amortised heap-space analysis. In Peter Sestoft, editor, *Programming Languages and Systems : 15th European Symposium on Programming, ESOP 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 27-28, 2006*, volume 3924 of *Lecture Notes in Computer Science*, pages 22–37. Springer-Verlag, 2006.

[HJ06b] Martin Hofmann and Steffen Jost. Type-based amortised heap-space analysis (for an object-oriented language). In Peter Sestoft, editor, *Proceedings of the 15th European Symposium on Programming (ESOP)*, volume 3924 of *Lecture Notes in Computer Science (LNCS)*. Springer-Verlag, March 2006.

[HMV06] Kevin Hammond, Greg Michaelson, and Pedro Vasconcelos. Bounded space programming using finite state machines and recursive functions: the hume approach. *ACM Transactions on Software Engineering Methodology*, 2006. Submitted.

[HvS06] M. Hofmann, J. van Oosten, and T. Streicher. Well-foundedness in Realizability. *Archive for Mathematical Logic*, 45(7):795–805, 2006.

[LM07] Chunxu Liu and Greg Michaelson. Hw-hume in isabelle. In O. Chitil, editor, *Draft Proceedings of 19th International Symposium on Implementation and Application of Functional Languages*, number TR 12-07. Computing Laboratory, University of Kent, 2007.

[MWH+06] Greg Michaelson, Andy Wallace, Kevin Hammond, Iain Wallace, Armelle Bonenfant, and Zezhi Chen. Toward resource certified image processing software. In *Systems Engineering for Autonomous Systems Defence Technology Centre (SEAS DTC), Annual Technical Conference, July 2006, Edinburgh, Conference Proceedings*, page A15. UK MoD, 2006.

[SHFV07] Hugo R. Simoes, Kevin Hammond, Mário Florido, and Pedro Vasconcelos. Intersection Types for Cost-analysis of Functional Programs. In *Proc. 2006 International Conf. on Types (TYPES 2006) (to appear)*. Springer-Verlag LNCS, 2007.