1 Purpose and aims

Embedded Systems (ES) constitute the vast majority of computer systems today, and virtually every product includes ES. Traditionally, the software (SW) of ES consisted of small pieces of code controlling particular electromechanical devices, but in recent years software has become the dominant component of products in providing functions, connectivity and human-machine interfaces. The platforms of such systems were processors with a small amount of memory with capacity usually driven by the production price. The continuous development of hardware (HW) following Moore’s law makes it possible to significantly increase the software complexity, reaching high performance levels. In addition to the support for developing a computer platforms (including CPU, memory, etc), different specific HW devices dramatically increased performance figures and decreased the production costs. These dedicated computational resources start to be used in general-purpose computation. Predictions are that the next generation of high performance ES will be build on heterogeneous platforms, i.e. platforms consisting of different types of computational units and technologies: Field-Programmable Gate Arrays (FPGAs), Graphics Processing Units (GPUs) and multi-core Central Processing Units (CPUs). Such platforms will be possible to adapt to a wide range of application areas, giving the impression of ”dedicated” characteristics. With such dedicated platforms, the system performance can be significantly increased, supporting, for instance, processing of massive amounts of data in real-time - a strong requirement in different types of autonomous systems. For example, the vehicle industry (Volvo CE and Volvo Technology) is currently looking for a 3D-capturing sensor and related processing system to identify other vehicles, objects and people.

The current general practice is to use GPU integrated with CPU in personal computers. This trend is also extending now in the ES areas, with Intel’s Atom Pineview/Cedarview, or AMD’s Embedded G-Series multi-core processors. Consider adding to this the Xilinx/ARM FPGA - Ziq, providing both multiprocessing and actual hardware power. This makes it possible to build systems able to efficiently process e.g. very large amount of data streams, significantly more than it is possible today with the conventional solutions. The SW companies have started research for producing software for heterogeneous platforms (for example Liquid Metal project at IBM, based on the Lime language that extends Java with special concurrency constructs [4], and the Kiwi project at the University of Cambridge and Microsoft Research Cambridge that update .NET programming languages for parallel execution [14]. In a recent publication, S. Singh from Microsoft Research claims ”The future is heterogeneous” [39].

However there is in general a little support in software development running on a heterogeneous platform. At the same time ES software is becoming increasingly complex, and new approaches are needed that can model and design it in an efficient way. Examples of such approaches are Model-driven Engineering (MDE) and Component-based Software Engineering (CBSE), which aim to raise the design abstraction level and reuse already designed and implemented solutions.

The challenges in using these approaches is how to ensure quality requirements characteristic to ES: resource constraints, performance, real-time requirements and dependability requirements (such as reliability, safety, availability).
The HW part of an embedded system benefits from two major technological developments, both rooted in the ever decreasing silicon figures (under 10nm in the forthcoming years [17]): (i) the multi-core processing solutions [25]; (ii) the ever increasing logical capacities of FPGAs and the large improvements in performance and processing aspects of FPGA soft/hard-cores [45]. This makes the heterogeneous platforms an increasingly attractive solution for ES, given the flexibility, versatility and large capacity for a wide range of applications and domains.

However, this also lead to a small group of challenging questions: (i) How to partition the implementation in components ending either in the SW (processor) or in the HW part (FPGA)? (ii) How to partition further the SW parts over the stand alone processor(s) and the FPGA soft/hard-cores? (iii) How to allocate the SW over the existing multi-core processors? (iv) How to integrate the legacy/pre-designed components (wither HW or SW)? (v) How to effectively develop ES on heterogeneous platforms ensuring quality requirements? (v) How to enable a seamless transformation of data and control between different computational resources?

An answer to some of the design questions is offered by a co-design approach. With a peak of the research activities in the '90s and possibly best illustrated by the efforts in the creation of the POLIS framework [6], the HW/SW co-design traditionally lacked the support of high abstraction tools. At the present, given the multi-core systems, the approach should also be complemented by a SW/SW co-design. In addition, today there are several tool-based approaches that may support a better implementation of a co-design methodology. However, without a proper methodology, and without additional customized design solutions, such tools do not a-priori offer design solutions.

The overall goal of HELPING is to improve of embedded systems development and their performance on heterogeneous platforms. The aim of the project is to advance state of the art in model- and component-based development for ES by improving prerequisites for efficient reuse of existing SW components by early system verification: It will formulate and implement an appropriate component model for computation-intensive embedded real-time systems, and provide a support for distribution of SW to the heterogeneous platform to achieve optimal results in performance, system response, and to meet other important ES requirements. The component model will be used for (i)modelling systems and reasoning about extra-functional-properties (EFPs) that depend on the underlying platforms, and for (ii) defining deployment configurations. Further, a set of data transformation patterns in form of connectors will be developed which will significantly increase the development process effectiveness.

2 Survey of the field

The fields related to the research belong to: (i) component models for embedded system end specification of EFPs, (ii) system (i.e. SW and HW) modelling, and (iii) SW deployment (that includes SW allocation and synthesis).

In recent years, the component-based approach has steadily penetrated the ES domain. There exists several component models dedicated to ES [12], including the following: AUTOSAR [5], the new standard in automotive industry using a simple component model; BIP [7] a framework for modelling heterogeneous real-time components and composing component behaviour; BlueArX [21] a simple component model used by Bosch for the automotive control domain; Koala [42] a component model used in consumer electronics at Philips; PECOS [44] developed by ABB and Bern University, and used for small devices; Pin [26] developed at Carnegie Mellon Software Engineering Institute with Mälardalen University involvement, used as a basis in prediction-enabled component technologies; Robocop [22] a component model developed by the consortium of the Robocop ITEA project; Rubus [2] developed by Arcticus Systems AB and Mälardalen University and used in hard real-time systems; OSGi [32],
a service-oriented framework with an open specifications for the delivery of multiple services over wide area networks to local networks and devices; SaveCCM [1], developed within the SAVE project by several Swedish universities and ProCom [10, 37], developed at Mälardalen University.

Specification of EFPs is in focus of CBSE and MDE. In CBSE the EFPs specification and composition have been the main focus in series of CBSE symposia. “Predictable assembly from certificated components” (PACC) [43] defines a framework, and a method to extend the component specifications with EFPs and their analysis using “analytical interfaces” is elaborated. In the MDE approach systems are modelled at (at least) two levels with Platform Independent modeling (PIM) and Platform Specific Model, (PSM) [35] in which the PIM part can be reused for different platforms specified by PSM. This is a widely used approach, but it has a problem with reusability on component level, and by this a problem with scalability. For modelling systems there exosts several modeling languages, some of the most prominent are AADL and SysML.

Hardware design has traditionally been an issue of VHDL or Verilog coding. Recently, higher level abstraction frameworks have been successfully tested in this area: SystemC [3], the UML (and especially through the extension Marte [31], or by using Domain Specific Languages [27]) and the Mathworks tool complex [24, 23]. UML tools, but especially the Mathworks toolboxes allow for a full and correct automated code generation for SW implementations (C, C++, etc) or for HW ones (VHDL, Verilog), thus providing a very helpful support for HW design. In addition, especially by providing a long path of high level abstraction development, and by accurate simulations, the tools bring a strong support to co-design. However, a co-design methodology (HW/SW, SW/SW for multi-core) is missing, and, for the HW solutions appropriate data types are required. Moreover, in order to apply the automated code-generations functionality, a clear distinction of the HW and SW elements must be observed. This does not relax the synthesis issues, especially considering the complex modern designs, where elements with the same implementation target (either SW or HW) may be placed at different levels of the hierarchy, in an intricate mixture of HW or SW architecture specification. Partitioning of work to CPUs and GPUs has been addressed by, e.g., Joselli et al. and Greve et al. [15, 19]. Detailed optimization, including memory registers and number of threads, has been investigated by Ryoo et al. [34]. This work in not related to component models and connections between them, but rather are focused on a code level.

The considered dynamic parallel execution environment also imposes special requirements on the functional or algorithmic side of the SW solution. As the trend of using graphics HW to speed up general purpose computations emerged, several interesting techniques to utilize the inherited parallelism of GPUs have been explored [33]. With the introduction of the CUDA programming model, GPU computing became more flexible and powerful. Massive parallelism is now provided by hundreds of processor cores and thousands of concurrent threads hierarchically arranged to offer both coarse-grained and fine-grained task-based and data-parallel parallelism [30]. OpenCl [20] is a programming language for heterogeneous platforms using CPU and GPU and enabling a fast parallelism. Accelerator system from Microsoft allows certain kinds of data parallel transformation to be written once and then executed on three different targets: GPUs, multicore CPUs and FPGA [40]. However, in general mapping the same computation onto several different computing elements typically involves rewriting algorithms to achieve an efficient execution on each architecture using a different programming language and a very different execution model compared to the original algorithm developed for a CPU [39].

HELPING focuses on the modelling part. Using CBSE approach we can assume that the

---

components being modelled are either new (i.e. are not yet implemented), or reused (i.e. already implemented, even for different computational units). We also allow modelling without code, using estimated or calculated values of EFPs. In an iterative development process, the modelling can start with a small number of the exiting components, where this number increases in each iteration. Already in an early phase we can reason about software allocation, and we can refine (or redo) the analysis in each iteration.

3 Project description

In HELPING project we will develop theories and demonstration tools that support design and development of ES that process large amount of data in real-time using heterogeneous HW platform. Our main objective is to provide methods and tools for optimal distribution of SW on platforms that include multiore CPUs, GPUs and FPGAs, with respect to important requirements of ES like performance, real-time requirements, and resource utilization. Our objective is to model and analyse the solutions in an early phase of the design, to make it possible to take proper design decision early, but postpone as much as possible the implementation phase, which has to benefit of the automated code generation tools available for both SW and HW.

3.1 Project Challenges

The main challenges in development of heterogeneous real-time ES are:

- How can we model ES performing on a heterogeneous platform which will enable to specify the system architecture with necessary information related to extra-functional properties of architectural elements and of the entire system?
- How can an embedded system SW be allocated on different types of computational resources such as CPUs, GPUs and FPGAs utilizing the platform in best the possible way with regards to performance, real-time and resource utilisation?
- How can we achieve a seamless interaction between data and SW deployed on different computational resources?

Our approach in meeting these challenges is applying component-based development. The component-based approach is important for efficient management of complex SW, for its reuse, and easy configuration of the entire systems. In this approach we will develop a modelling formalism for SW components and for underlying heterogeneous platform. The modelling of components will include not only functional modelling, but also specifications on extra-functional properties (EFPs) such as performance, timing properties and resource utilization, valid for different computational units. This formalism will be used in modelling of SW and the underlying platforms, and in the specifications of component allocation in the platform. The component and system EFPs will be obtained by a combination of static analysis and measurements.²

Using this approach we will address the following research challenges.

1. **Component model.** Design of a component model that runs on combination of different platforms (CPU, GPU, FPGA); Building up a formalism for specifications of EFPs important in heterogeneous ES and applicable for different contexts (different types of the underlying platforms).

2. **Heterogeneous Platform Modelling.** Design or adaptation of a modelling language that will be able to describe different computational resource of a heterogeneous platform.

²EFP analysis, analysis and measurements is not the scope of this project, but the results will be used the from related project (See Section Other Grants.)
3. **Component allocation.** Components allocation to processing units their transformation to the real-time execution units following execution model. Among other things, this requires means to obtain relevant system-level quality properties for a particular allocation, from the EFPs of individual components.

4. **Component compositions.** Component compositions deployed on different computational resources (CPU and GPU for example).

The approach and the challenges are presented on the figure below, in which a typical ES application containing of sensors, actuators, and a heterogeneous platform, is shown.

![Figure 1: Software components mapping to an ES application with a heterogeneous platform.](image)

### 3.2 Project Work Packages

The work will be organised in three work packages, as described below.

**WP1: Component specifications in heterogeneous systems**

**Motivation.** To provide an analysis of SW deployment on heterogeneous platform, we need a model of SW system and a model of the underlying platform, as well as ability to specify extrafunctional properties (EFPs). While there are standard ways on how to specify FPs (e.g defining interfaces as component signatures using modelling or programming languages), there is no standard way how to specify EFPs, in particular not for heterogeneous systems. EFPs should include specifications of SW/performance parameters, but also platform-specific parameters. This requires novel methods for EFPs specification as parts of component models. In addition to the EFPs specification, for an efficient development, it is necessary to provide support for management of EFPs (the ways how to ensure them, for example how to ensure and justify a minimal throughput), as well as compositions of them.

**Overall Goals and Outcomes.** The overall goal of this WP is to develop formalism and methods for component specifications that include EFPs specification for heterogeneous systems, and develop or adopt existing formalisms for specification of the underlying platform. The concrete outcomes will include:

**O1.1** Specification of a component model that supports interoperability between components deployed on different computational units.

**O1.2** Platform modelling that enables specification of platform architecture and EFPs as attributes of the platform architectural elements.

**O1.3** A framework for a formal EFP specification and management for components and systems.
The results will include formalisms, methods, technologies, and implementation of a tool prototype for specification of component-based embedded system including the system architecture and their properties. They constitute a foundation for SW deployment (i.e. allocation and synthesis) analysis and input to WP2 and WP3.

**WP2: Component allocation to hardware**

**Motivation.** The proposed approach enables an analysis of SW run-time properties (i.e. EFPs) in relation to its deployment independently of the functional implementation of SW components. This enables performing the analysis in an early development phase (when components are not necessary implemented), to come to some important decisions (such as designing the underlying platform), or in a later phase when an optimal or a trade-off solution in respects to given EFP requirements.

**Overall Goals and Outcomes.** The goal of this WP is to investigate how the available information about EFPs can be exploited to automate parts of the development process, and to provide necessary support for the developer to make rational allocation decisions. As a first step, we will develop means to derive system properties, for a given allocation mapping, from the EFPs of individual components and a specification of the HW platform. Next, this analysis will be used to guide an automated search of suitable allocation mappings.

- **O2.1** Analysis methods for deriving relevant system-level properties from component properties that that are platform-related, and from the system architecture.
- **O2.2** Theories and tools for automated generation of allocation mappings, with respect to multiple optimization criteria.

**WP3: Component Synthesis on Heterogeneous Platforms**

**Motivation.** The synthesis part of the development process includes several steps: *(i)* transformation of a component model to an execution model (for example, components that are design and resalable units can be translated to real-time tasks, possibly running in different threads. In addition to provide an optimal transformation with a respect to requirements and (for example solving a question how to break down components into execution chunks and the merge them in executable units in order to achieve the best performance or save resources), important challenge is to preserve the execution semantics that is defined in the modelling phase using component models; *(ii)* generation of a glue code that enables interoperability between the execution units. In particular, a generation of a glue code between the computational units running on different platforms is important for an efficient transfer of data and control between the units - a part that today is very troublesome, time consuming and often a source of design and implementation errors. *(iii)* Finally, compilation of component source code.

**Overall Goals and Outcomes.** The overall goal of this WP is to provide an optimal (or near to optimal) solution for code generation from the specified deployment model and existing source code on a heterogeneous platform. The synthesis process will enable several iterations in which an trade-off analysis of the transformation to the execution model can be performed, and the transformation will include automatic building of connectors between the platforms. We will develop a set of pre-defined connectors that follow different design connection patterns. The concrete outcome form this WP will be:

- **O3.1** Transformation rules from SW architecture including deployment specification to an executable model.
- **O3.2** A library of predefined patterns for interoperability for data and control between different computational units, or within the same platform.
3.3 The iteration synthesis process with a trade-off analysis based on performance, timing, and resource constraints.

The work will include a development of theories, models and methods, and prototype tools implementation, integrated with the results from WP1 and WP2.

3.3 Significance

The project results are expected to have a large impact in the research community and in industrial contexts. In our discussions with several Swedish companies (between others ABB, Volvo, Hectronic) all of them claimed the importance of having an efficient process in development of heterogeneous ES. The research significance is many-fold: (1) components allocation on heterogeneous platform which requires adequate modelling of different computational resources; (2) Code synthesis over the different computational resources; (3) Design and implementation of connectors on a heterogeneous platform. All this will require generalization of principles of state of the art in CBSE, and MDE for ES.

3.4 Preliminary Results

Components and system modelling for heterogeneous platform. Our work will be a further advance and specialization of our previous work on The Foundation of Strategic Research (SSF) Research Center PROGRESS in which we have developed a component model for ES ProCom [38], with a framework for management of EFPs [12, 37], and a prototype tool PRIDE [8]. We will further develop these elements for the heterogeneous systems, provide a more generalized and formalized framework. Further, we will use the practical results from SSF RALF3 (Software for Embedded High Performance Architectures) project which is working on specifications and analysis of EFP on different computational resources, and which will provide a physical platform used for evaluation of the results. In particular we will use the case - an underwater robot that will be developed in RALF3, for modelling of its control and 3D video systems that will process a large amount of data in the real-time. We will also use the results obtained in behaviour and resource modelling: REMES - Resource modelling Language [18]. The component model will have to also provide characteristics for (possible) HW implementations. A continuation of previous studies [27, 28, 41] is to be extended to cover more generic platforms, with a larger set of properties.

Components allocation. Our work on developing ProCom [38], a component model for distributed ES, supports a deployment process where components, physical nodes are specified and an allocation algorithm has been developed [11]. This provides a good basis for the work on automating the allocation. The related problem of automating the allocation of components to real-time execution units has been investigated for the predecessor of ProCom, SaveCCM [13].

Component Synthesis. In the framework of PROGRESS, we have developed synthesis methods with emphasis of preserving the execution semantics [9] defined by ProCom component model. The basic ideas from this and related references can be further expanded in a case of parallelism and building up connectors between different computational resources. Mathematics and emulation based approaches used for computing best solutions in the case of multi-processing HW systems [29, 36] can be further extended to provide support for co-design (HW/SW, SW/SW). The implementation of such methods will be represented by algorithmic methodology scripts in tools like Mathworks (Simulink). In addition, methods will be provided to automatically generate respective code (for instance C for SW and VHDL for HW) regardless of the respective module locations.

The preliminary results makes a good bases for extension of the principles on different computational resources.
3.5 Time- and Realization Plan

The HELPING project will be active during 5 years, and the group will consist of a professor 25% (Ivica Crnković - with expertise in SW modelling and component-based development for ES), an adjunct professor 20% (Tiberiu Seceleanu - expert in HW/SW co-design) and a PhD student 100% (to be recruited).

The project will be performed in two iterations, first three years and second two years. The work will start with WP1 (Components and system modelling) and it is expected to be completed in 18 months. WP2 (components allocation) is planned to start in the second year and is expected to be finished by the half of the third year. WP3 is planned to start in the second half of the second year and will be run in parallel with WP1 and later with WP2 until the end of third year. Each WP will have a validation phase during its last six months of their phase, where the validation will be performed on simpler case studies, including some parts of the underwater robot video system. The first iteration will finished with the achieved goals: the system modelling, allocation and connectors theories, methods and the first prototype. The second iteration will will repeat the WPs, providing the refinements based on the validations from the first iteration. A validation of the results will be performed on more complex case studies, including the underwater robot and possibly an industrial case in cooperation with industry. A control and optimisation system for a wind mill developed at ABB Research is a potential candidate. The expected final result will include several leading conferences (ICSE, CBSE, DATE, CODES+ISSS, ASE, etc.) and journal papers (IEEE TSE, ACM TOSEM, Elsevier JSS, IEEE D & ToC, Eslevier JSA), the prototype tools, and a completed PhD thesis.

3.6 Part of project cost

The proposed project would be fully financed (100%) by the grant requested from the Swedish Research Council.

3.7 International and national collaboration

The main applicant of this proposal is a professor at Mälardalen University since 2002, the scientific research leader of Industrial Software Engineering group. The group is internationally recognized for its excellence in CBSE and is leading organisation of several major international events in CBSE (ACM CompArch federated event, including CBSE symposium, Euromicro SEAA conference, etc.). We have developed component models for ES used in research communities (SaveCCM, ProCom), and used in industry (Rubus). We have also analysed best practices in component-based development processes and collaborated in their deployment in industry (in cooperation with CC Systems, ABB, Volvo CE, Ericsson). In particular the cooperation with ABB Corporate Research is extensive - several senior researchers and two adj. professors are active in the cooperation.

The international cooperation includes activities on different levels: common research and education projects, common organisation of research events such as conferences and workshops or networks, common research, exchange of students and researchers and similar. The group has continuous cooperation with, among others, Carlo Ghezzi and Raffaela Mirandola (Politecnico di Milano), Henry Muccini, Vittorio Cortellessa and Paola Inverardi (L’Aquila University), Antonia Bertolino (Italian National Research Council), Paul Grünbacher (Linz University), František Plašil (Charles University Prague), Ralf Reussner (Karlsruhe University), Wilhelm Schäfer (Paderborn University), Michael Gonzalez (University of Santander), Franck Barbier (Pau University), Michel Chaudron (Leiden University), Hans van Vliet and Patricia Lago (Vrije Universiteit Amsterdam), Kung-Kiu Lau (Manchester University), Pekka Abrahamsson (Helsinki University), Panagiotis Katsaros and Ioannis Stamelos (Thessaloniki...
University), Grace Lewis and Kurt Wallnau (SEI/Carnegie Mellon University), Ian Gorton (Pacific Northwest National Laboratory Richland), Clemens Szyperski (Microsoft), Judith Stafford (TUFTS Boston), George Heineman (Worcester Polytechnic Institute), Daniel Gajski (Irvine University), Ralf Resussner (Karlsruhe University).

The group has national collaboration with, among others, Claes Wohlin (Blekinge Institute of Technology), occasional cooperation with Martin Törngren (Royal Institute of Technology) and recently with SICS.

Tiberiu Seceleanu benefited over the years from a select cooperation group. Common activities with all the individuals in the group - academic researchers with full international recognition, or industrial researchers with high knowledge of their specific domains, have been concretized either in academic publications or high quality, or in European level research initiatives with acknowledged completion. A brief description of the group would include: David Garlan (Carnegie Mellon University), Axel Jantsh, Martin Törngren (Royal Institute of Technology), Hannu Tenhunen (European Institute of Technology / Royal Institute of Technology), Tughrul Arslan (Edinburgh University), Laila Gide (Thales / Artemis), Koen Bertels (Delft University of Technology), Luca Benini (University of Bologna), Gilbert Edelin, (Thales), Rudi Lauwereins (IMEC), Jan Madsen (Technical University of Denmark), Stamatis Karnouskos (SAP), Armando Colombo (Schneider), Jan Bosch (Chalmers University), Jerker Delsing (Luleå University of Technology), Ivan Ring Nielsen (Technoconsult), Roberto Zafalon (ST Microelectronics), Ahmed Jerraya (Commissariat à l’ Energie Atomique).

3.8 Other grants

Ivica Crnković is leading the VR multi-project grant CONTESSE – Contract-Based Components for Embedded Software. The project has different objectives than the current proposal. The main objective of CONTESSE is component contract specifications and contract compositions, while HELPING is dealing with deployment on heterogeneous platform.

Ivica Crnković is leading the SSF-funded project RALF3 – Software for Embedded High Performance Architectures. HELPING has complementary objectives – while RALF3 project is focused on implementations of heterogeneous systems and analysis of EFPs in related to different computational resources, HELPING will provide a foundational theory for SW modelling and deployment that will be used in RALF3. On other side HELPING will use demonstrators from RALF3 for the validation, and results from EFPs analysis (e.g. execution time).

Ivica Crnković also holds a grant OPEN-SME – Open-source software reuse service for SMEs in EU FP7. In this project research results are deployed and validated in industrial context.

Tiberiu Seceleanu is the leader at ABB of the iFEST – industrial Framework for ES Tools, an Artemis project (grant: 100203), 2010-2013 [16]. iFEST addresses ES life cycle aspects, with a main focus on tool integration and HW/SW co-design. The identified design flow process (following a V-model), inter-stage interactions and the selection of tools are well suited for analysis and application in HELPING.

References


