Information and Communication Technologies (ICT) Programme

Project Nº: H2020-ICT-2016-1-732105

D4.3: CERBERO Multi-Layer Runtime Adaptation Strategies (ver. 1)

Lead Beneficiary:	UPM
Workpackage:	WP4
Date:	March 2018
Distribution - Confidentiality:	[Public]

Abstract:

As stated in the amended DoW, this deliverable is the first version (it will be followed by an updated version in M28, D4.1) and "contains detailed technical information on the different strategies to support adaptivity in CERBERO compliant systems. Different chapters for HW, SW and sensor-driven adaptation strategies/algorithms/components are envisioned. Along with the reports, also the HW monitors, SW agents and specific algorithms are going to be delivered. Please note that all the technical providers, cooperating to the above listed tasks, will be responsible of their own physical components".

Disclaimer

The CERBERO Consortium is the following:

Num.	Beneficiary name	Acronym	Country
1 (Coord.)	IBM Israel – Science and Technology LTD	IBM	IL
2	Università degli Studi di Sassari	UniSS	IT
3	Thales Alenia Space España, SA	TASE	ES
4	Università degli Studi di Cagliari	UniCA	IT
5	Institut National des Sciences Appliquées de Rennes	INSA	FR
6	Universidad Politécnica de Madrid	UPM	ES
7	Università della Svizzera Italiana	USI	CH
8	Abinsula SRL	AI	IT
9	AmbieSense LTD	AS	UK
10	Nederlandse Organisatie Voor Toegepast Natuurwetenschappelijk Ondeerzoek TNO	TNO	NL
11	Science and Technology	S&T	NL
12	Centro Ricerche FIAT	CRF	IT

For the CERBERO Consortium, please see the http://cerbero-h2020.eu web-site.

Document Authors

The following list of authors reflects the major contribution to the writing of the document.

Name(s)	Organization Acronym
Eduardo de la Torre, Alfonso Rodríguez, Leonardo Suriano, Daniel Madroñal, Eduardo Juarez	UPM
Joost Adrianse	TNO
Francesca Palumbo	UNISS
Tiziana Fanni	UNICA
Maxime Pelcat	INSA
Ayse Goker, Leszek Kaliciak, Stuart Watt, Hans Myrhaug	AS
Sebastián Tardón; Manuel Sánchez	TASE
Pablo Muñoz Martínez	S&T

Document Revision History

Date	Ver.	Contributor (Beneficiary)	Summary of main changes
2018/01/11	0.1	UPM	ToC
2018/01/30	0.2	UPM	Updated ToC and added section on Overall Adaptation Strategies
2018/03/08	0.3	TNO, UNICA, INSA, UPM	Added sections from TNO, UNICA, UPM, INSA
2018/03/19	0.4	TASE	Added TASE contributions in sensors section and UPM in HW section intro. Added AS contribution to sensors section.
2018/03/30	0,5	UPM, TNO, TASE	Updates on Section 6, use case linked to components
2018/04/06	0.6	UPM, TNO, TASE	Last updates on SoA, changed to individual sections. Additions and corrections in sections 3, 4 and 5. Intro section partially added.

2018/04/09	0.7	UPM, AS	Executive Summary added, section 1 finished, SoA additions in section 6.
2018/04/16		UNISS	Internal Review
2018/04/30	1.1	UPM, UNISS	Final Review
2018/05/14	1.2	UPM, AS, TNO	Changes to revisions' comments, last updates

Table of contents

1. Executive Summary

This deliverable document presents the overall adaptation scenario foreseen in the CERBERO project, and presents its main components. The CERBERO adaptation approach identifies three basic parts that benefit from adaptation: Hardware (HW), Software (SW), and Sensors. The three main sections (3 to 5) provide an insight to these three main parts.

Adaptation requires a series of actions (sense, derive measures, take decisions, and perform the required changes), which must be tailored according to the nature of the fabric being adapted: the SW subsystem, the HW subsystem, and the sensors layer. The components of the framework that make up the CERBERO adaptation framework have been identified and categorized. Heterogeneity is one of the challenges of CERBERO, given the fact that an adaptation decision (i.e., a trigger for adaptation) might affect to more than one component at a time and, also, being motivated from different reasons, and with different functional and non-functional objectives.

The document presents also the specific adaptation strategies and components that will be used in the different CERBERO use cases.

1.1. Structure of the Document

Section 2 provides an overall view of the runtime adaptation environment, identifying its different components. Next three sections are the core of the document, namely, Section 3 and 4 deal with HW and SW adaptation techniques, respectively, while Section 5 concerns sensor adaptation. Finally, Section 6 states, among the previously mentioned techniques, which ones will be employed in the different use cases.

1.2. Relation with CERBERO Requirements

Deliverable D2.7 of the CERBERO project defines a list of CERBERO Technical Requirements (CTRs) the project should achieve. The CERBERO adaptation strategy, and its related components and techniques, described in this document contribute to the fulfilment of the mentioned requirements in the following aspects:

CTR id	CTR Description	Link with the D5.6 document on CERBERO framework components
0001	CERBERO framework SHOULD increase the level of abstraction at least by one for HW/SW co-design and for System Level Design.	The support provided by PREESM for the abstraction of SW and HW tasks, the capability of SPIDER to decide, at runtime, task migration between fabrics, the unified PAPI access scheme to monitors for HW and SW are the key contributions to this requirement.
0003	CERBERO framework SHOULD provide incremental prototyping capabilities for HW/SW co-design.	Incremental prototyping capabilities are envisioned at the tools/components level: • MDC will be provided with an enhanced HLS support; • DPR features will be improved thanks to JIT HW implementation and composition tool; • the runtime monitoring of ARTICo³, JIT HW and MDC

		reconfigurable hardware accelerators will be enabled thanks to PAPIFY integration; • PREESM, SPIDER and PAPI will be used to drive ARTICo³, JIT HW and MDC prototyping features.
0006	CERBERO framework SHOULD ensure energy efficient and dependable HW/SW co-design using cross-layer runtime adaptation of reconfigurable HW.	Energy is a main KPI, and it is addressed in most of the techniques described in this deliverable, including all HW fabric types, SW agents and sensor infrastructure. Monitors and adaptation techniques for energy efficiency and dependability are foreseen in the three kinds of components, also.
0009	CERBERO SHALL develop integration methodology and framework.	The adaptation infrastructure and the associated tools are part of the CERBERO framework.
0014	CERBERO WP and task leaders SHALL organize scheduled face to face and remote meetings.	WP4 periodic management meetings have been organised in order to track progress, deviations and risks.
0016	CERBERO tools SHOULD be tested vs state-of-the-art	Section 6 in this deliverable contains information about the use of the various components and technologies in the three use cases.
0018	CERBERO technology providers SHALL prepare face to face or online tutorials / education for use case engineers.	Tutorials on HW and SW adaptation have been prepared for the Summer School 2017 and CPS Week 2018. Academic engineers have received these courses in order to have feedback.
0019	CERBERO technology providers SHALL coordinate technical support for their tools with use case engineers.	A preliminary version of some of WP4-related tools (PREESM, ARTICo3, MDC) has been delivered for the Summer School 2017.
0020	CERBERO framework SHALL provide methodology and tools for development of adaptive applications.	This deliverable provides information about the components (mainly) and tools (partially) involved in the adaptivity support.

1.3. Related Documents

- D2.7. – CERBERO Technical Requirements: The activities behind D4.3 are driven by the CERBERO Technical Requirements that have been described in the previous subsection.
- D4.4. – CERBERO Self-adaptation Manager: D4.3 represents an input for D4.4. The CERBERO Self-adaptation Manager is meant to orchestrate the ensemble of strategies presented here.
- D5.6. – CERBERO Framework Components: Adaptivity, to be successfully and easily mastered, has to be supported by the design framework. Techniques and components described in D4.3 are bounded (as pointed out in the text where necessary) by specific Framework components detailed in D5.6.

2. Overall Adaptation Strategies

The need for adaptation in complex CPS and CPSoS is easy to justify, given the fact that this type of systems must be set to operate under changing conditions and fulfill changing requirements along their lifetime.

2.1. Types of adaptation

There are three types of adaptations [Rohr06] which are differentiated by the moment in which the adaptation is initiated:

- **Reactive**: adaptation is consequence of external events or internal changes. This happens when the performance of the system is degraded, or a safety constraint is not satisfied, which are analysed by a monitor that triggers the adaptation.
- **Predictive**: adaptation is internally triggered to avoid future off-nominal states based on predictive models of the system. The adaptation engine can have prediction algorithms of how the execution will behave based on the context, anticipating off-nominal states and correcting them before a monitor triggers a reactive behaviour.
- **Proactive**: adaptation happens during the nominal operations. In this case, adaptation is related to optimization; while previous types attempt to hold nominal states and safety constraints, proactive adaptation is related to *self-optimization* and improvements of the system's performance while it is operating in nominal conditions.

2.2. Type of adaptation triggers

The triggers that could initiate adaptation behaviours can be divided in three main categories:

- **System** (*self-awareness*): the system must monitor its internal status to keep the application safety constraints. If a constraint is not satisfied, the system will trigger an adaptive behaviour to correct the situation. In the literature this is also called *self-healing*.
- **Environment** (*environment-awareness*): the system can be influenced by the environment; for such reason, changes in the environment could trigger adaptation behaviours to avoid potential risks. For this kind of adaptation trigger it is required sensory capabilities, for instance hardware sensors or internet data gathering.
- **Human** (*user-commanded*): the interaction with the user is another way to trigger adaptive behaviours to cope with the user desires or actions. In this direction, the user can be seen as another layer of the system, so a proper interface with the user is required.

2.3. The adaptation loop

Most adaptive systems follow a loop process to achieve adaptation, no matter the type or the trigger for adaptation. There is always, someway, a continuous sequence of the following steps:

- Sense/monitor the environment (physical part) or the system itself (cyber part), making the system context aware and self-aware, respectively.
- From the above measurements, derive the key parameters that will facilitate the decision to adapt or reconfigure. This can be done via models of both the physical and cyber part.
- Make a decision to change the configuration by means of some criteria. For instance, if performance goes under a certain threshold, perform modifications to increase performance. These criteria may be application specific.
- Provide the means to perform system adaptation.
- To have one or more computing fabrics with sufficient flexibility to hold the commanded adaptation.

All these functions in the adaptation loop must be done by the adaptation components. Offline adaptation does not require all components to be embedded in the system, but self-adaptation does.

The identification of the components that perform the adaptation tasks is not always straightforward. However, in the case of CERBERO, where one of the goals is to generalise the concept of CPS adaptation, an effort has been made to provide a coherent adaptation infrastructure, and to provide common means to interact between these adaptation components.

It is important to highlight that, in the CERBERO approach, computing fabrics of different nature and a variety of sensors require this effort of generalization, given the fact that, for instance, every fabric might require different means for adapting, and the way the key performance metrics are provided can be different, also.

In the following subsection, the overall adaptation infrastructure and its components is described.

2.4. The CERBERO adaptation components

The main components of the CERBERO (Self-)Adaptation Infrastructure, which is shown in Figure 1, are:

- **Adaptation Fabrics:** computing/sensing resources.
- **Adaptation Monitors:** hardware/software components to track the state of the *Adaptation Fabrics*.
- **Adaptation Engines:** hardware/software components to change the configuration of the *Adaptation Fabrics*.
- **Adaptation Manager:** high-level (i.e., system layer) entity with runtime decision-making capabilities. It uses the information provided by the *Adaptation Monitors* to decide whether to change the configuration of the *Adaptation Fabrics* using the *Adaptation Engines*.

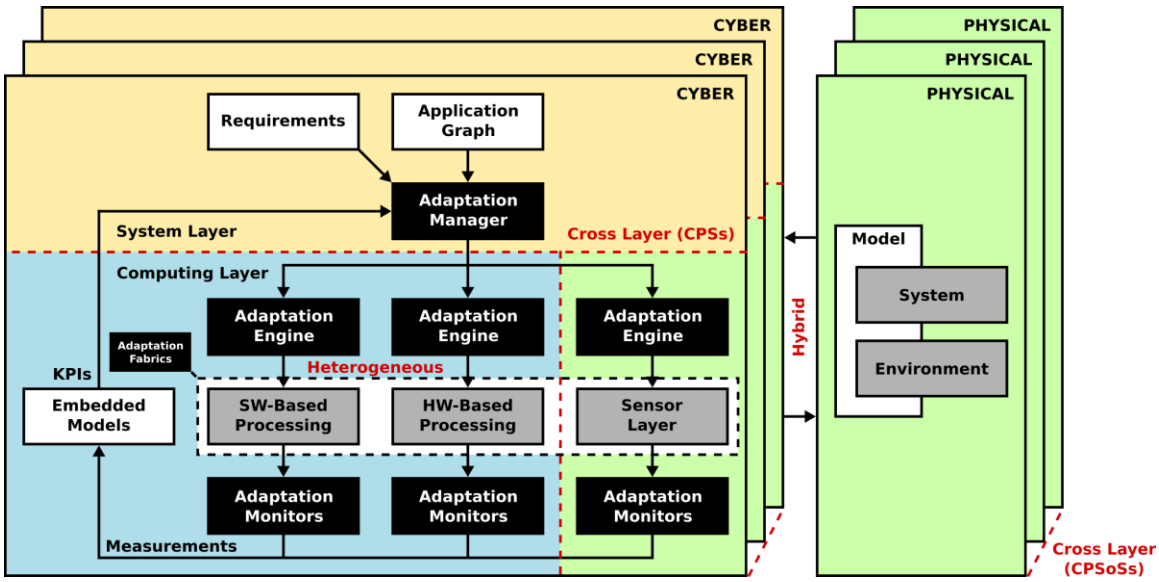

Figure 1 - CERBERO (Self-)Adaptation Infrastructure

The core adaptation process as shown in Figure 1, therefore, consists of composed blocks of functionality, each containing an adaptation manager, overseeing monitors to track system, environment, and user. These monitors are then used, in combination with an embedded model, to drive adaptation through an adaptation engine. Importantly, these units may be organized hierarchically: the adaptation may involve sensing and manipulating a set of subordinate systems, each of which may also have its own adaptation behaviours, as well as signalling requirements to super-ordinate systems. This allows, for example, a software-developed high-level adaptation manager to coordinate adaptation between several lower-level subsystems.

3. Hardware Adaptation

In the context of HW, three different adaptation fabrics will be used within CERBERO: ARTICo^3, MDC and Just-in-Time HW composition. Furthermore, ARTICo^3 acts as a container for the other two possibilities and, hence, ARTICo^3 + MDC or ARTICo^3 * JIT composition are possible.

ARTICo^3 addresses module-level reconfiguration (typically, a HW accelerator), providing acceleration scalability and/or fault tolerance in DMR or TMR (Double or Triple Modular Redundancy) for enhanced reliability. It can hold modules designed using Register-Transfer-Level (RTL) design entry, modules designed by using High Level Synthesis (HLS), or, as mentioned, modules built with MDC and HLS, or with JIT HW composition.

MDC provides coarse-grain reconfiguration support (CGR). The associated tool flow provides circuit merging and fast reconfiguration switching for a finite set of predefined circuits.

Finally, JIT HW composition addresses fine-grain reconfiguration, providing a way to map circuits at runtime by composing small HW components laid on an overlay architecture. This mapping can be deterministic (starting from an intermediate SW representation) or based on bio-inspired techniques, namely, Evolvable HW.

Section 3.1 provides a summary of related work on the adaptation possibilities and approaches derived from the use of FPGA technology. It contains details on how reconfiguration overlays can be set on top of them, and how the combination of different granularity levels can provide benefits in terms of higher adaptation levels, as it is planned to provide with the CERBERO HW adaptation strategy.

Section 3.2 provides insights to the three basic techniques (ARTICo^3, MDC and Just-in-Time HW composition). Later, Section 3.3 addresses the two required reconfiguration engines: CGR (for MDC) and DPR (for ARTICo^3 and JIT HW composition).

Finally, Section 3.4 deals with HW monitors. An effort has been made to provide consistent monitoring interfaces for all HW fabric types, as well as for runtime SW execution at CPS level.

3.1. State of the Art on HW adaptation

Although there are many research works on the topic of HW adaptation, this section focuses on the most challenging issues within CERBERO to this respect, namely, the provision of HW overlays on FPGAs to easy and dynamise their reconfiguration capabilities, and the use of combined granularities in order to combine the advantages of different types of HW adaptation. They are shown below.

3.1.1. State of the Art and Advances on FPGA overlays

Today applications require huge computing power and standard monolithic general-purpose processors have often left the place to more efficient Graphic Processing Units (GPUs) or Massively Parallel Processor Arrays (MPPAs) [Dinechin14]. However, such kind of devices impose strong challenges when power constraints have to be met, e.g. in

CPS or Internet of Things (IoT) contexts. Here, Application Specific Integrated Circuits (ASICs) can better fit with the computing and power efficiency demand. However, ASICs development is an extremely long and costly process, and often FPGAs can be chosen as an alternative. FPGAs can be pushed to provide efficiency close to the ASICs, while offering also programmability [Trimberger15]. However, compared to the typical software development, FPGAs require depth hardware expertise and longer implementation phases. HLS [Liang12] [Najjar13] made it simpler by introducing the possibility of programming FPGAs directly from high level languages, such as C, C++ or OpenCL. Nevertheless, long compilation times are still necessary to generate FPGAs programming files, limiting the usage of HLS only for static reconfigurable systems [Stitt11].

The latest FPGAs opened to the possibility of tightly coupling them with an operating system running on general purpose processors [Ahmad16]. In such architectures, the operating system could manage hardware tasks in the same way it deals with software ones, thus exploiting some kind of hardware abstraction that hides the underlying implementation details [Bergmann13]. Such a hardware abstraction is commonly referred to as FPGA overlay [Cong14]. FPGA overlays are basically programmable hardware abstraction layers on top of FPGAs obtained by means of pre-implemented programmable components mapped on the available FPGA resources and serving both computing and routing functionalities. The concept of FPGA overlays consists of using FPGAs as programmable accelerators instead of wired application specific ones. In this way, the problem of designing accelerators is replaced by the one of programming ALUs/processors [Polig15]. The benefits of using FPGA overlays are: better application management, portability, easy and fast code compilation and FPGA programming, and massive design reuse.

FPGA overlays are not conceived for replacing HLS tools or vendor specific design flows, rather they aid developers when programmability, resource sharing and design time are strictly constrained. They abstract FPGA strengths, like scalability, reliability and isolation, from implementation details, delivering them to the developer that can also be not aware of the particular hardware substrate. Two main kinds of FPGA overlays have been presented in literature:

- fine-grained FPGA overlays – they provide basically an FPGA-on-an-FPGA, thus realizing a non vendor specific FPGA whose bitstream is portable to other FPGA devices thanks to the small granularity of the configured components (LUTs, switch boxes, communication boxes), at the price of long compilation time and big configuration, area and performance overheads [Brant12];
- coarse-grained FPGA overlays – they implement reconfigurability at the data/word level, guaranteeing simpler design and faster compilation phase. They can differ in terms of complexity and number of computational units, interconnect and configuration strategy [Laforest17]. The main trends are [Kapre06]: pushing performance with a spatially constrained overlay where computation and interconnecting units remain unchanged during the execution of a certain kernel [Capalija13]; save resources leveraging on time multiplexed computing and interconnecting units whose behaviour is changed during kernel execution, cycle-by-cycle [Liu15].

Despite several FPGA overlay architectures have been presented so far, only few of them have been effectively validated in the practice [Bergmann13] [Capalija13]. This is mainly due to area and performance overheads, meaning that overlays have been often designed with no care about the underlying hardware substrate, leaving to the synthesis tool the possibility of inferring operations within coarse grained blocks, such as DSP, and leading to sub-optimal results [Ronak14]. Recently, with the increasing interest of the scientific community on FPGA overlays, some works tried to overcome these issues by better shaping the overlay depending on the available resources of the specific FPGA target, such as DSPs [Jain2 2015] or BRAMs [Kapre17].

In the CERBERO project we intend to exploit the benefits of FPGA overlays (by enabling efficient and effective hardware software adaptation strategies), while mitigating their limits (by the adoption of proper compilation and design infrastructures). The bottom part of the CERBERO framework (see D5.6) will be mainly exploited; hardware abstractions will allow the usage of advanced and combined hardware reconfiguration approaches (as coarse-grained or dynamic partial reconfiguration) in a lightweight manner, hiding such complexity to the designer. The purpose is providing different forms of adaptivity, both functional and non-functional, taking into account system, environment and users triggers.

3.1.2. State of the Art and Advances in Multi-Grain Reconfigurable Approaches

Reconfigurable systems offer high performance and flexibility, filling the gap between general purpose processors and ASIC systems. A reconfigurable system is generally composed by a network of configurable processing elements (PEs) with configurable interconnections that can compute simple or more complex functions according to their granularity. Reconfigurable systems can be divided in two main granularity classes: Fine-Grained (FG) and Coarse-Grained (CG).

FG reconfigurable systems involve PEs that can compute simple functions at the single bit level, they can offer high flexibility being able to compute any kind of functionality, but involving a large amount of PEs. This implies the need of large configuration bitstreams and long configuration phases. The most common example of FG architectures is FPGAs, usually exhibiting substantial configuration memory footprint and time. Recently, dynamic partial reconfiguration has been proposed to mitigate a bit those limits, allowing for the dynamic reconfiguration of predefined (at design time) regions within an active design [Vivado17].

CG reconfigurable systems involve PEs at the level of the data/word, thus being able to compute more complex functionalities. For implementing a given functionality, they achieve higher area efficiency and imply less configuration overhead (data and time) with respect to FG systems [Hartenst12]. Nevertheless, they offer more limited flexibility.

Different works explored the adoption of both kinds of reconfigurable solutions, FG and CG, on the same substrate to combine their benefits together. Modern FPGAs themselves are actually multi-grain platforms, since they include CG reconfigurable blocks as BRAMs and DSPs. Amagasaki et al. [Amaga08] propose a variable grain logic cell (VGLC) architecture that can change the computational granularity corresponding to the application. The VGLC has four units, each one involving a two input 1-bit full adder and a two input LUT sharing some common logic. The HoneyComb architecture [Thomas12]

is an adaptable dynamically reconfigurable cell array. Every cell consists of a routing unit and a functional module; the routing units of all cells are connected to their neighbours and compose the reconfigurable communication network. The specification of every component within the array can be enabled, disabled, or modified, thus also partial reconfiguration is possible. The DeSyRe framework [Sourdis13], leveraging on a mixed grain texture, can manifest adaptivity and fault tolerance features. A DeSyRe RISC component is divided in smaller sub-components surrounded by reconfigurable interconnects. In case of fault of one sub-component, it can be replaced, using the reconfigurable interconnects, by an identical unused close sub-component or by a functionally equivalent instance implemented in FG reconfigurable hardware. Diniz et al. [Diniz14] propose a runtime accelerator-binding scheme for tile-based mixed-grained reconfigurable processors. The mixed-grained reconfigurable processor is composed of multiple tiles. Each tile consists of multiple CG and FG reconfigurable elements. The number of reconfigurable elements inside each tile and in the whole architecture is a design time decision. Given an architectural configuration, a communication-minimizing binding for datapaths of custom instructions is determined at runtime, employing datapath reusing and inter-tile communication cost estimation. Yuan at al. [Yuan15] present a multi-grain FPGA aimed for mobile computing and focus on two key steps towards higher efficiency: interconnect networks, and coarse-grained reconfigurable digital signal processors. The chip incorporates FG configurable logic blocks, medium-grained digital signal processors along with reconfigurable block RAMs, and two CG kernels.

The abovementioned approaches are limited to partially reconfigurable CG arrays, where the PEs are identical. Moreover they do not provide the designer proper instruments to partition functionalities between FG and CG substrates, neither to trigger reconfiguration. In the CERBERO project the multi-grain adaptive support will combine ARTICo^3 and MDC approaches, where different partially reconfigurable slots of the FPGA (ARTICo^3 compliant) are filled in with heterogeneous application specific CG datapaths (MDC compliant). The idea is not providing just functional adaptivity, but also being able to support non-functional (i.e. redundancy for fault tolerance or parallelization to improve throughput) driven ones, combining the benefits of both FG and CG. The goal of CERBERO in this perspective is not simply providing multi-grain reconfigurable accelerators, rather to build proper hardware abstractions, capable of facilitating FPGA overlays together with a framework for the design of the different parts of the system, their deployment and runtime management.

3.2. Adaptation Fabrics

3.2.1. ARTICo^3

The use of SRAM-based FPGAs has merged the best of two worlds (i.e. hardware and software), enabling systems with software-like flexibility while keeping the high-performance benefits of dedicated hardware-based processing. The specific technology that supports this is Dynamic and Partial Reconfiguration (DPR), a procedure that basically consists of writing in a configuration memory to change part of the circuits implemented in the FPGA device while the rest of the system is still working. This hardware *copy & paste* methodology is illustrated in Figure 2.

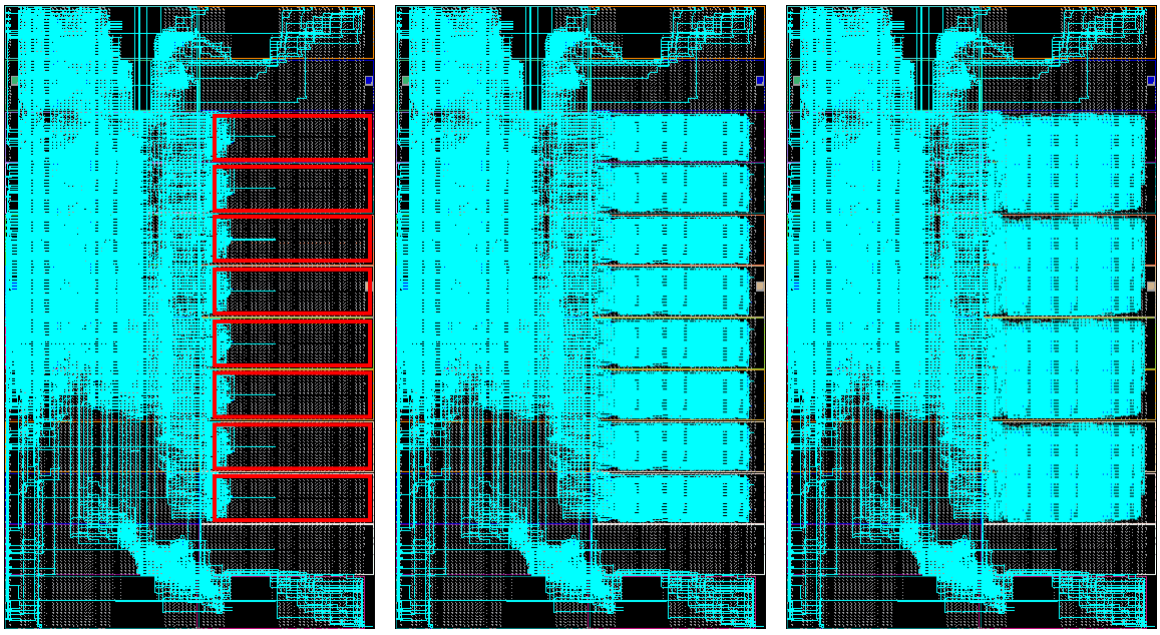

Figure 2 – DPR-compliant layout of the ARTICo^3 architecture

The ARTICo^3 (please, refer to D5.6 for more information) architecture exploits DPR in high-performance embedded systems that use a processor-coprocessor approach. However, instead of relying only on one application-specific hardware accelerator for each task as it has been traditionally done, the computing fabric supports a multi-accelerator based computing scheme. Similarly to embedded GPUs with support for general purpose computing, the ARTICo^3 computing fabric can operate in SIMD-like fashion (Single Instruction Multiple Data), where each copy of a given hardware accelerator works with a different set of input data. In this regard, it is important to highlight that ARTICo^3-based computing requires both the processor-coprocessor approach and a hardware/software partitioning that only selects computing-intensive data-parallel tasks to be implemented as hardware accelerators. The execution model of the architecture can be seen in Figure 3.

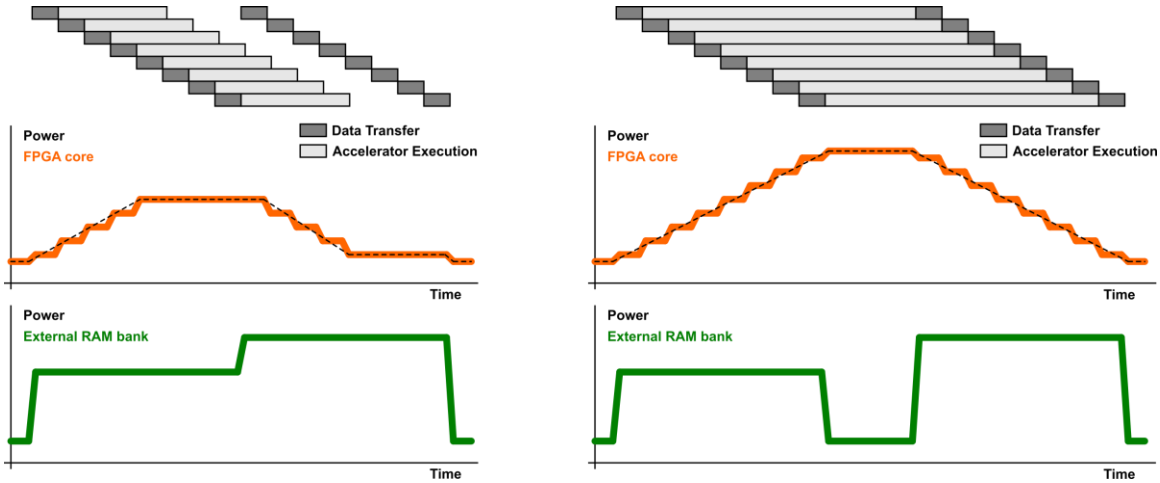

Figure 3 – ARTICo^3 execution model for SIMD-like execution (top) and its impact on power consumption (left: memory-bounded; right: computing-bounded)

However, module replication using DPR can be used to increase fault tolerance in the reconfigurable partitions too. The configurable datapath of the architecture can switch the data delivery/collection patterns to either target computing performance (see previous paragraph), or redundant execution, with two or even three copies of an accelerator performing the same computation over the same input data where the results are retrieved through a voting unit to mask possible errors.

3.2.2. MDC Accelerators

Coarse-grained reconfigurable (CGR) architectures demonstrated to be a viable possibility to achieve adaptivity in cyber physical systems [ESL-HEVC]. Contrary to FGR/DPR architectures, CGR applies reconfiguration at the word/data level. Computing and interconnecting resources are configured word by word, rather than considering single bits. A computing resource could be, for instance, a whole multiply and accumulate unit managing 8 to 16 bit-width data, as depicted in Figure 4. In such a context, reconfiguration can be performed extremely quickly due to the limited bitstream size, as described below.

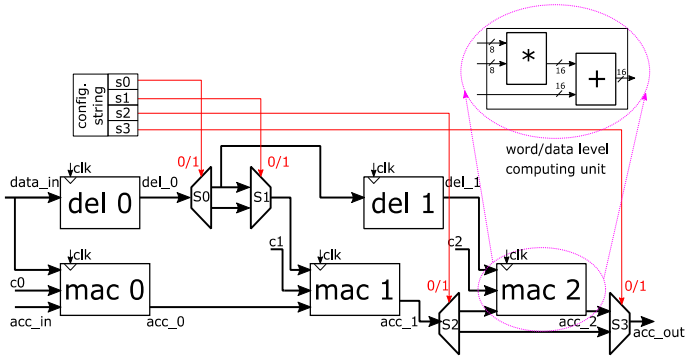

Figure 4 – Example of CGR Architecture

The most common approach in designing CGR systems is from architecture to applications: a generic architecture of potentially configurable, homogeneous or heterogeneous Processing Elements (PEs) linked with potentially configurable, interconnecting structures is made available for mapping different kernels. Such substrates are usually very generic. In such a way, flexibility is favoured, at the price of execution efficiency of the single applications.

The CGR design approach exploited by the MDC tool (please refer to deliverable D5.6 for details on MDC) is the opposite, going from applications to the architecture. The CGR substrate is shaped according to a set of desired applications, resulting in an application specific substrate capable of achieving strong execution efficiency, but limited to the fixed implementable applications set. Moreover, the cost of reconfiguration is minimized, both in terms of time (up to one clock cycle to configure the interconnection logic, when composed of combinatorial elements) and power (no need of downloading a new big bitstream through dedicated channels). Modular high level representations of the applications can simplify the design of such CGR architectures since each module can be mapped directly to one different PE. The original functionality of the applications can be guaranteed by the insertion of crossbar switches that drive the data according to configuration patterns. The dimension of these patterns depends on the

number of crossbar switches placed throughout the substrate. For instance, in the example of Figure 4, a 4-bit configuration pattern is necessary.

3.2.3. Just-In-Time Composable Hardware

JIT hardware composition refers to the ability to implement, at runtime, hardware accelerators on FPGAs without a pre-synthesized design. The components that make up this fabric are three different ones: a virtual architecture, a library of PEs, and an algorithm guiding the mapping of the PEs on the virtual architecture.

Virtual architecture

The virtual architecture consists of a 1D or 2D regular overlay that abstracts the FPGA resources from upper layers. An overlay is made of different blocks that contain specific resources of an FPGA. The blocks of the overlay need to have predefined communication interfaces between them.

It is possible to have layout blocks that are common to multiple FPGA models. Therefore, upper layers do not need to take into account the distribution of logic elements within the FPGA fabric that is being used, but the type of block.

The virtual architecture should support scalability in order to dynamically change the size of the circuit and adapt itself to different computing requirements.

Library of PEs

In order to compose hardware, it is necessary to have a set of different PEs synthesized and mapped to fit into blocks of the virtual architecture. The PEs can be different circuits: multiplexers, adders, etc. These elements will be stored as relocatable bitstreams so they can be accomodated into any suitable block of the virtual architecture.

One of the main challenges in this kind of fabric is how to select the PEs that allow the implementation of the widest possible range of accelerators.

Runtime composition algorithms

Finally, it is necessary to have algorithms that map the different PEs into the different blocks of the virtual architecture in order to obtain the desired functionality. Different approaches will be explored:

1) *Deterministic approach*

The idea behind this approach is based on runtime software compilation where one high-level programming language is compiled into an intermediate representation (IR), and then the runtime engine specific to this fabric compiles this IR for the processor where the SW runs. After this, the deterministic hardware composition, with the help of a runtime mapper algorithm, can map different PEs from a predefined library onto the virtual architecture getting the desired accelerator.

These runtime adaptation libraries will abstract hardware details from the user making very easy for people without hardware skills to implement hardware accelerators, thus making FPGAs more accessible to people with no hardware background. FPGA vendors are currently making a great effort to achieve this purpose with HLS. Although HLS makes the design of HW accelerators much easier, hardware skills are still needed in order to get efficient solutions. In contrast, for just-in-time hardware composition option,

once the PE library is created, no hardware knowledge is needed. Another advantage over HLS is that synthesis and implementation time is greatly reduced.

In the context of CPSs with deeply heterogeneous systems where different HW and SW fabrics can be used, this adaptation strategy facilitate the task of the designers because only one software implementation will be needed. From this implementation, an IR is generated and can be implemented in both a HW fabric or a SW fabric.

2) *Iterative/evolutionary approach*

The idea behind this approach is based on the ability of nature to evolve to adapt itself to changing environments. In this case, the algorithms guiding the hardware composition are called evolutionary algorithms (EAs). The application of these EAs to synthesize electronic circuits is called evolvable hardware (EH).

Evolutionary algorithms are a set of optimization algorithms used as problem solvers in cases where there is little knowledge of the physic equations underneath the problem to solve or where external conditions are expected to change often. EH works with a set of solutions at a time. Each solution consists of a mapping of the PEs in different blocks of the virtual architecture. It is necessary to have a fitness function that indicates how good a solution is. Based on this fitness function, the best solutions (individuals) of the set of solutions (population) are selected. Genetically inspired operators are applied to the selected solutions, and a new set of solutions will be created. This process will be repeated until a solution with a good enough fitness function is found.

This approach will be used to solve different problems in CPS, for example to create adaptive controllers. Reinforcement algorithms will be also explored.

This approach has really good advantages for CPS, some of them are:

- Very good adaptability in changing environments.
- No need to know the physics underneath the problem to be solved. Thus, if new problems arise on the field, it is possible to try to apply EAs to see if the problem can be tackled autonomously by the system.
- Very good for autonomous systems where there is little or no human interaction.
- Self-healing capabilities: If a problem arises on the FPGA fabric, the EA will detect that the previous accepted solution is no longer acceptable, and it will evolve to find a new solution that avoids using the damaged section of the fabric.

3.3. HW Adaptation Engines

As mentioned before, the adaptation engines are embedded components that provide the means to adapt the fabrics according to the decisions taken by the adaptation manager. In CERBERO's HW-related fabrics, there is a combination of coarse-grain and dynamic partial reconfiguration mechanisms, and so, each one has an associated adaptation engine.

The first one (CGR) provides fast reconfiguration between merged kernels produced from the MDC tool. The second one (DPR) provides exchangeability of ARTICo^3 slots, as well as a fine-grain approach, which addresses FPGA reconfiguration at PE level or frame-level for JIT composition of HW.

The combination of either CGR blocks or JIT-composed fabrics within ARTICo^3 modules provide a mixed-grained adaptation approach which is identified as a big

scientific contribution of the CERBERO adaptation framework, since the combination of their properties provides fast adaptation time, fault tolerance and performance and energy scalability within a dynamic HW adaptation context.

3.3.1. Coarse-Grain Reconfiguration Engine

The MDC tool is an automated framework that generates CGR architectures by describing application kernels with modular high-level models of computation: the Dataflow Process Networks (for more details on the model please refer to D3.5). As described in D5.6, MDC adopts an iterative datapath-merging algorithm that is capable of sharing dataflow actors among the different input models. To access shared resources Switching-Boxes (SBoxes) are inserted in the combined model. MDC handles programmability, keeping trace of the SBoxes configuration patterns for each kernel to be executed, saving them into dedicated Look-Up Tables. Once all the input dataflow models are combined together, MDC generates the corresponding HDL description embedding in the top-module also a configuration module that properly set the SBoxes selectors according to a given network ID (each application kernel corresponds to a different network ID). Figure 5 depicts an example with 2 input dataflow models whose combination is achieved by the insertion of four SBoxes, requiring an overall configuration pattern size of 4 bits.

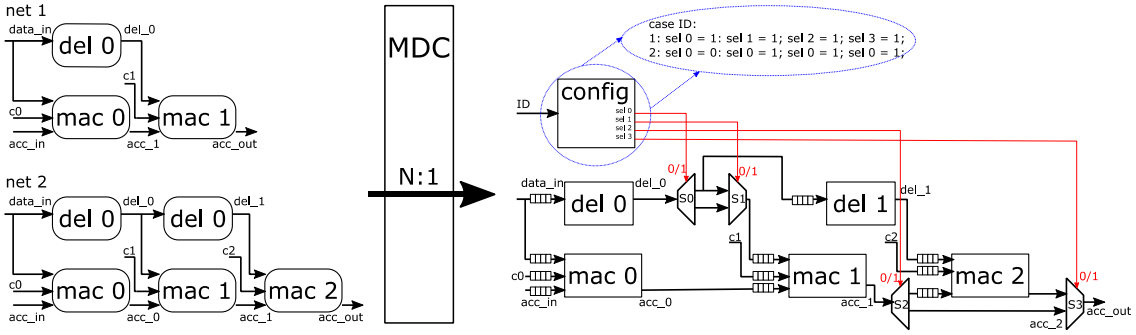

Figure 5 – MDC-compliant CGR architecture: N input networks merged and mapped over a unique CGR substrate.

Figure 6 highlights the operation of a CGR architecture generated by MDC. When the network ID = 1 *net 1* in Figure 2 is executed, and actors *del 1* and *mac 2* are excluded from the computation.

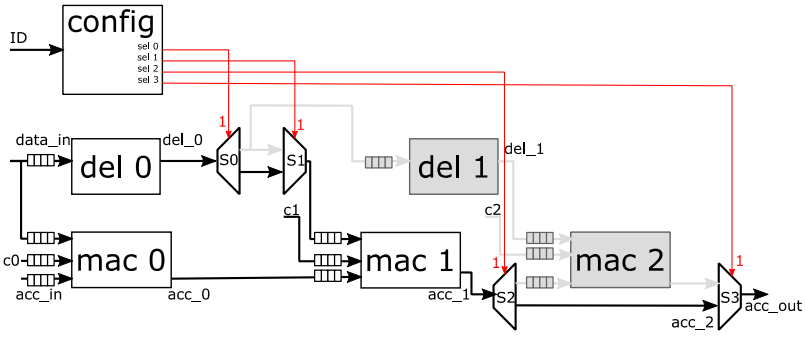

Figure 6 – MDC-compliant CGR architecture: running example

Such a CGR architecture can target both ASIC and FPGA technologies. In case FPGA is selected, thus dealing with an application specific CGR FPGA overlay, MDC offers also the possibility of seamlessly integrating the CGR architecture as an adaptive hardware accelerator in a processor-coprocessor system (see Figure 7).

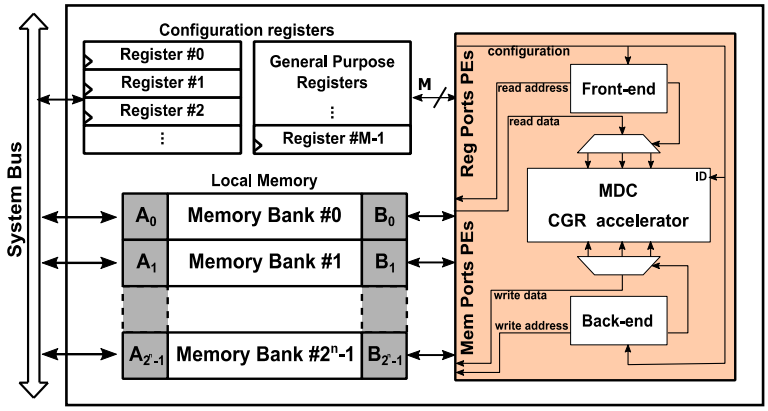

Figure 7 – MDC generated accelerator for host-coprocessor environment

3.3.2. Dynamic and Partial Reconfiguration Engine

DPR on FPGAs refers to the capacity of these devices to dynamically change part of its circuitry (while the other parts continue with their normal operation) to implement other functionalities that were not present before the reconfiguration process. DPR is achieved by modifying some sections of the configuration memory in real time.

The design flow of DPR-capable systems is based on the definition of a static system that should never change during execution (fixed digital circuit) and one or multiple Reconfigurable Partitions (RP), where different accelerators can be allocated at runtime.

The main difference with is that DPR does not need to have all the functionalities implemented at the same time. Only one accelerator is implemented, and the rest of the functions are stored in an external memory. Therefore, the main advantage of DPR over coarse-grain reconfiguration is FPGA resource utilization in designs where multiple accelerators need to be reconfigurable. This leads to designs that use smaller FPGAs or, in the event of having larger devices, to designs where accelerators can be specifically tailored to intensive data-parallel computation. The main disadvantage is reconfiguration time. CGR is almost instantaneous, whereas DPR takes more time to finish.

Both ARTICo^3 and hardware composition tools need DPR for their operation. ARTICo^3 uses DPR to reconfigure the available slots with different accelerators to exploit task-level parallelism, and/or with one or more copies of the same accelerator in order to speed up a specific data-parallel task or to obtain hardware modular redundancy for enhanced fault tolerance. Hardware reconfiguration tools need DPR to reconfigure the PEs in the blocks of the virtual architecture.

Relocatable DPR consists in reconfiguring the same bitstream in different (but with equivalent resources) reconfigurable partitions. In order to achieve relocatable DPR, two things are necessary: relocatable partial bitstreams and a modified reconfiguration engine.

When implementing an accelerator in an RP, in order to make it relocatable, it is important to ensure that the design in the RP is completely isolated from the static system, and the only connection between them happens through a predetermined interface. Xilinx commercial tools do not generate relocatable partial bitstreams and therefore, it is necessary to develop a custom tool to obtain them.

The reconfiguration engine is the component in charge of the reconfiguration process. It is a HW or SW component that has access to special resources of the FPGA to modify the configuration memory at runtime. The inputs of the reconfiguration engine are the location of the memory that stores the partial bitstream, and the area of the FPGA where it has to load the bitstream.

The resources of the FPGA that reconfigure Xilinx FPGAs are designed to reconfigure one column of resources of a clock region at a time. Thus, if there are multiple RP in the same clock region in a vertical position, it is necessary to compose the partial bitstream also at runtime. Therefore, a readback-modify-write approach is required for bitstream composition at sub-clock region level.

3.4. Adaptation Monitors

Figure 8 presents the overall CERBERO self-adaptation Infrastructure. This section focuses more on the HW adaptation monitors that are registers placed inside the reconfigurable fabric. In particular, Figure 8 shows how monitors are interfaced to PAPIFY and the upper layers of the CERBERO (self-)Adaptation Infrastructure.

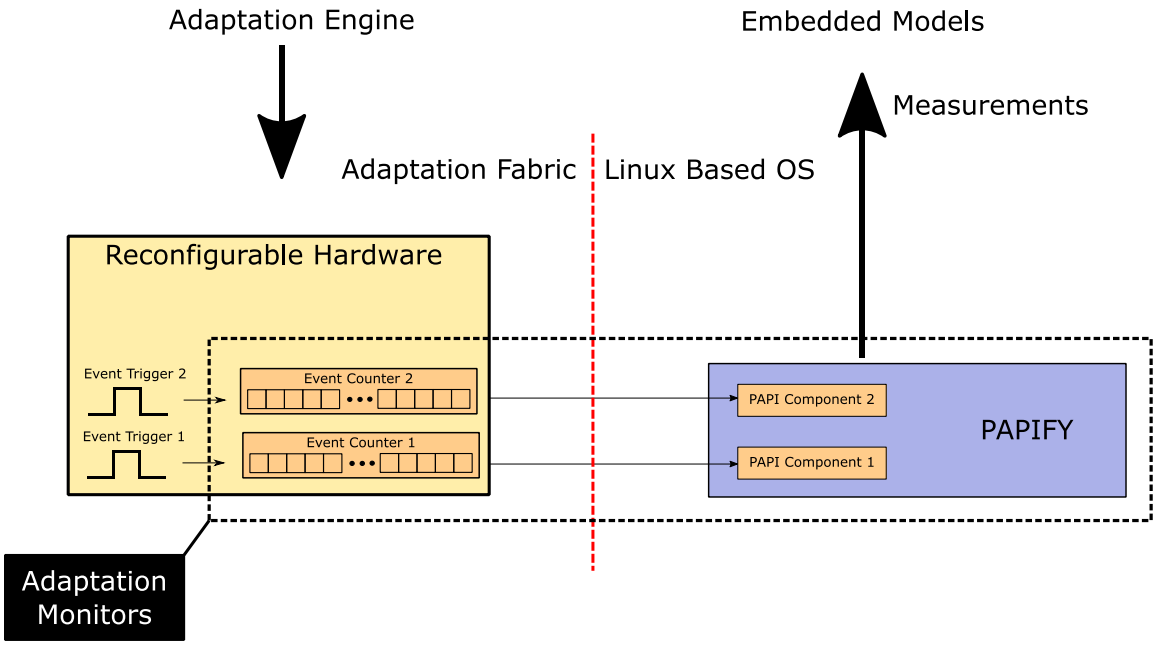

Figure 8 – HW-level monitors' interfaces

In reconfigurable hardware, one or more monitors can be placed to count the occurrences of specific events that are important for the computation, thus they are also called *Event Counters*. Measurements from the monitors are read by PAPIFY, an application based on the open source standard library Performance API (PAPI). PAPIFY interfaces the

monitors with the *Embedded Models* to quantify/estimate the *KPIs*. As described in D4.4, communication between the monitors and PAPIFY is managed through the definition of software PAPI components.

Figure 9 describes in detail how the communication is managed: HW monitors write/read functions are defined in PAPI components and are capable of directly mapping the user-space virtual addresses to reconfigurable HW accelerators physical addresses using mmap(…); in such a way, the SW application can use PAPI function calls to access the monitors. The example of Figure 9 shows a PAPI component that manages communication with multiple HW monitors.

In the ARTICo^3 architecture, there are the following monitors:

- fault monitors, to increase reliability while monitoring multiple slots carrying out the same functionality;
- latency monitors, to obtain the execution time of any accelerator in clock cycles.

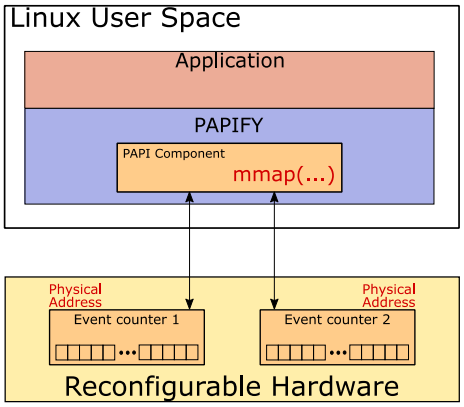

Figure 9 – HW monitor access via PAPI components

Moreover, there is the possibility of monitoring events in accelerators placed inside the ARTICo^3 slots. Considering MDC generated dataflow-based accelerators they can be:

- *FIFO read/write rates:*
  - to estimate system consumption, by exploiting a priori characterization of the single actors consumption, in order to opt for less consuming configurations (both with coarse- or fine-grain reconfiguration) if needed;
  - to know the current computation nature if multiple dataflow branches are available on data-dependent applications: fine- and/or coarse-grain reconfiguration can improve parallelization for the most stressed branches in place of the others, thus improving the system execution efficiency.
- *FIFO occupancy:*
  - to detect communication bottlenecks within the accelerator and decide whether or not a specific actor can be parallelized.
  - to resize the FIFOs according to their utilization: if a FIFO is always almost full and another one is always almost empty, it can mean that their dimensions are not properly set. Hence, the design could be reconfigured achieving a resizing of the FIFOs.

4. Software Adaptation

SW adaptation can be defined as "changing a software system during its execution." [Oreizy08] That is, adaptation is the capability of a software system to modify its behaviour based on internal or external changes during runtime. However, from the IEEE systems and software engineering vocabulary [IEEE10], adaptation is "the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed." Then, we can see adaptation from two perspectives: *static* and *dynamic*. By one hand, static adaptation takes place at design phase; in the other hand, dynamic (which is often called adaptability) happens at runtime:

- *Static:* "Software Adaptation focuses on the problems related to reusing existing software entities when constructing a new application and promotes the use of adaptors —specific computational entities for solving these problems. The main goal of software adaptors is to guarantee that software components will interact in the right way not only at the signature level, but also at the protocol and semantic levels." [Canal04] In this sense, "Software Adaptation can be considered as a new generation of Coordination Models." [Brogi06]
- *Dynamic:* "Adaptability means changing some aspect of a system's detailed behaviour while keeping the gross behaviour of the system consistent. From a contextual systems perspective, adaptability means matching behaviour to changes in environment, task, user population, preferences or some other factor; from a component systems perspective, it means selecting and/or configuring the component set to provide the optimum behaviour." [Dobson04].

This section is focused on dynamic adaptation as the different components of the CERBERO toolchain must adapt its behaviour during execution based on the current context. So, the next section will present more details on adaptation types and techniques for dynamic adaptation in the context of the self-adaptation infrastructure. A complete survey on static adaptation was written by Kell [Kell08].

4.1. SW agents and the self-adaptation manager

A SW adaptive agent is an entity that is able to gather contextual information (through adaptation monitors) and, based on that data, adjust its execution based on a behavioural model (adaptation fabrics), following a monitoring-plan-adapt cycle (coordinated by an adaptation manager), being the plan a two steps process (using an adaptation engine) in which the information is interpreted and then an adaptive action is selected [Garlan02].

Functionally, a SW agent to provide self-adaptation can be derived from the robotics architectures such as the three-layers (3T) [Gat97] architectures, goal-driven systems that follow the sense-plan-act paradigm. Software architectures for robotics control based on the 3T schema demonstrated considerable flexibility and adaptability, so advances in modern robotics controllers can be exploited as a base for self-adaptation engines **[**Kramer07].

Based on that approach, the layers are sequenced from bottom (functional) to top (deliberative) layers, being the intermediate layer the executive one. The execution flow is from bottom to top, being the sensor and data acquisition components placed in the functional layer. Typically, in such layer appear reflexive behaviours, which are reactive adaptations with a low latency, which correspond to critical safety constraints. The executive layer often provides task modelling capabilities, including hierarchical tasks decomposition and what-if simulation techniques for selecting the adaptation behaviour. The top layer is often related to Artificial Intelligence (AI) techniques, implementing planning & scheduling or machine learning mechanisms. Its objective is to perform both predictive and proactive adaptation, based on high-level models of the system. Functionally, the information flows between the lower level to the higher one, triggering adaptive behaviours in function of the context of each layer. As well, the deliberative layer can generate a plan (based on predictive models) to achieve the goals (goal-driven) that is to be executed from top to bottom.

The 3T schema is often related to a single agent, i.e., it can be coupled with a CPS. For providing collaboration required for CPSoS, these agents often provide coordination primitives, and, commonly, it is defined an agent hierarchy that can be centralized o decentralized. For the first one the classic schema is to share a database in which all agents store contextual information, so the other agents trigger adaptive behaviours based on that data. While this schema is easier to implement, the dependency of a central agent has several flags such as potential bottlenecks or unavailability of the system in case of failure of the central agent. Instead, the decentralized schema entails that the information is distributed among various agents, which provides more autonomy of each individual agent. However, the implementation is complex due to the synchronization between agents, while also it is required to properly define the role of each agent.

The contribution of CERBERO respect to the SW adaptation will be conducted toward supporting distributed dynamic adaptation for CPSoS by means of synchronizing behaviours through different components. To achieve this goal, a key factor will be the CERBERO intermediate format layer, which will allow different components to share information and perform adaptation based not only in its actual status but also considering others component's state. Moreover, a data fusion logging service will enhance SW adaptation providing a coherent log that can be exploited to correct and to enhance the system's behaviours.

4.2. Adaptation techniques and strategies

There are several adaptation techniques and strategies that enables self-adaptation engines to select the adaptive behaviour based in the current context. In the following enumeration, we will briefly present some of the techniques that are applicable at CPSoS level:

- **What-if simulation:** support the formulation and management of high-level what-if scenarios. These can be initiated by the system or environment (system-driven) or by the user (user-driven). A relevant aspect of the last is that requires research into user-interface aspects of formulating these cases and presenting the results, as well as developing the rules for dealing with what-if scenarios.

- **Task modelling:** different actions modelling languages enables adaptation based on context. The modelling of different agents' actions (and the user if required), e.g., as a hierarchical sequence of operational procedure steps, or using the Abstract Task Hierarchy (as described in [Bosse17]) enable behaviour selection based on the current context, providing mainly reactive adaptation schemas.
- **Planning & scheduling:** these techniques can be applied to perform both predictive and proactive adaptation based on high-level models of the system under control. In this regard, planning & scheduling techniques often provides long term plans and resource usage predictions that allow the engine to act in a proactive way. Generally, these techniques are goal-oriented.
- **Simulation in the loop:** if a simulation model of the system is available, it is possible to perform simulation in the loop, i.e., execute the simulation on-line. This enables predictive adaptation based on the comparison between the physical system and the software simulation. Applying simulation on the loop for real-time applications can lead to a better predictive adaptation, as it is possible to anticipate off-nominal conditions based on a high-coupled model.
- **Explanation and sense-making**: allow comparison of results within different scenarios, ranking possible action plan according to some criteria. In general, this technique requires to aggregate, interpret and present results coming in from various other components in a way that allows the user to be supported in their decision-making. In other words, what is the best way to convert the information flowing around the system into actionable information, supporting an explanation of the reasoning that properly enables user-commanded adaptation.
- **User modelling:** include a user model based around task load and emotional valence could be useful to provide adaptation when the human is considered as a layer of the system. The user model could be extended with additional dimensions and fields, ranging from simple user preferences, to past behaviours, or even derived parameters observed from the system behaviour.
- **Machine learning:** applying machine learning processes could enhance proactive adaptation based on the system history, providing support for self-optimization for various components of the system. Furthermore, it can be applied to improve the user model which will lead to better user-commanded adaptations.
- **Collaboration:** in a CPSoS application, collaboration techniques can be deployed to enable adaptive behaviours in coordinated environments. In this regard, it is desirable that heterogeneous CPSs can coordinate their activities to achieve a common goal, which typically entails a component of distributed optimization.

Each adaptation technique can be exploited by itself, providing SW adaptation at various levels of abstraction. However, within the CERBERO framework toolchain we will adopt what-if simulation, planning & scheduling, simulation in the loop and user modelling. In this regard, the objective is to provide adaptation not only based in an isolated technique but to coordinate them to adapt the system in the context of CPSoS, which commonly requires to adapt the system simultaneously at various abstraction levels. For instance, an adaptation on the system level (e.g., triggered from the simulation in the loop) could require adaptation on the user level, which is related to user modelling techniques. Our aim is to synchronize these adaptation techniques through the CERBERO intermediate format layer.

Focusing on CPS level, SW adaptation relies on the fact of having the system specified as a graph of intercommunicated actors. At this actor level, software adaptation will be triggered by the modification of parameters in a dataflow application representation conforming the PiSDF model [Desnos13]. These parameter modifications have different potential sources, compatible with CERBERO use case requirements:

- Modifications in sensed cyber (self-awareness) KPIs,
- Modifications in sensed physical KPIs,
- Modifications in user-triggered commands.

Using parameterized dataflow has for advantages to let the processing be triggered by the arrival of data, helping distributed execution, while keeping reconfiguration capabilities to the system based on external events. Dataflow parameters also homogenize the modeling of HW and SW reconfigurations, offering a way to master heterogeneity support. In order to manage highly varying workload and parameters, extensions of the JIT-MS [Heulot14] adaptive system management will be studied. In particular, extensions will be made for both CGR and DPR reconfigurations.

5. Sensor Adaptation

5.1. Introduction

Sensor-based adaptation can range from basic adaptation with using simple sensors to identify thresholds for action (e.g. measuring volts, temperature etc.) up to information fusion techniques. Sensor-based adaptation therefore includes information fusion techniques needed to enable different notions of adaptivity, and the different adaptation strategies based on the sensed data. It focuses on information-based adaptivity. Information from same or different sources needs to be combined for system adaptivity strategies in order to e.g.: enhance images/videos for the user and computer vision, make complex decisions, retrieve the necessary information from information storage, enable self-healing and energy aware adaptation of the robotic arm, or enable decision support for electric car simulator.

Sensor fusion also known as multi-sensor data fusion is a subset of information fusion. It combines sensory data or data from disparate sources in such a way as to reduce the uncertainty.

The uncertainty reduction can mean more accurate or more dependable information, or refer to the result of an emerging view, such as stereoscopic vision - calculation of depth information by combining two-dimensional images from two cameras at slightly different viewpoints.

Many data fusion approaches are problem specific; however some common methods are based on:

- Central limit theorem, e.g. [Katenka08]
- Kalman filter, e.g. [Barbosa16]
- Bayesian networks, e.g. [Jianzhong16]
- Dempster-Shafer, e.g. [Zhang15]
- Convolutional neural network, e.g. [Liu17]

Global Positioning System (GPS) is an example application of sensor fusion where data needs to be fused using various different methods.

5.2. Adaptation strategies for sensors and detectors

Figure 10 presents the generic view of information processing and strategies for adaptation. The data from sensors needs to be combined in order to make decisions, for example. Both the data from individual sensors as well as the fused data will be stored in order to be used in e.g. adaptation strategies. The stored data needs to be retrievable in order to be usable.

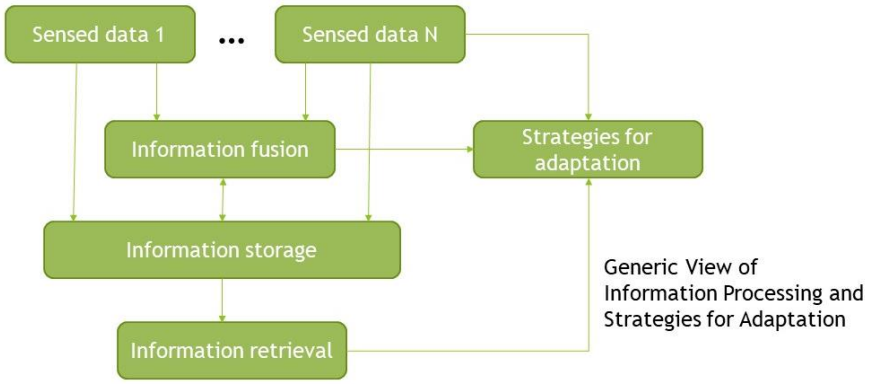

Figure 10 – Generic view of information processing and strategies for adaptation from sensed data

Here, the fused data and data from individual sensors is stored so that it can be later retrieved. The strategies for adaptation can make use of data coming directly from sensors, the fused data, and also the relevant data that was retrieved from the storage – which can be considered as prior knowledge for example.

The adaptation strategies considered in this section are summarised in Table 1.

Table 1 – Summary of adaptation strategies

Adaptation Strategy	Description Summary	Triggers for adaptation
Adaptation of Enhanced Vision System	Adaptation of the image quality to varying visibility conditions based on the number of active cameras. Can be also triggered by user requesting specific image quality. Synchronous cameras will de-noise images, asynchronous cameras will see through moving objects	Environment, User
	Adaptation of the image quality to poor visibility conditions based on new image enhancement methods	Environment, User
Adaptation of the fusion model for hybrid image retrieval	The weights associated with query and its context are adaptively updated based on the measured levels of their relationship	System, User
Computation precision adaptation	The computation precision will adapt to the current needs based on the sensed KPIs	System
Decision support adaptation	The decision support function will adapt its recommendations based on combined information such as traffic, temperature, battery level, etc.	

5.2.1. Enhanced Vision System

The purpose of enhanced vision system is to improve the user's situational awareness (can be related to Augmented Reality) and computer vision methods.

The enhanced vision system consists of multiple cameras, various image enhancement methods, and the adaptation to visibility conditions capability.

Adaptation to visibility conditions may involve automatic assessment of environmental visibility. The visibility can be divided into absolute and relative. For absolute visibility, the distance from the object should be known plus some object's characteristics. The relative visibility is a deviation of the current conditions from the ideal subjective visibility - no clouds, noise, good illumination. Relative visibility would be based on characteristics such as luminance, contrast, object clarity (quality of edges, blurriness, etc.).

A physical way to measure visibility could be based on a light receiver and a transmitter in a fixed position in the robot. This type of visibility measurement would measure the contrast between a target and its background. Camera would be located within a specific distance from the robot's component. The assumption is that the component's contrast in ideal visibility conditions is known.

For example, two cameras can be used to determine the night-time and day-time visibility conditions based on the contrast information and the estimated distance from foreground objects [Du13].

The method proposed in [Graves14] estimates the visibility based on the Sobel filters taking into consideration the fact that the reduced visibility results in an image with less detail, especially in the distance.

Sutter et al [Sutter16] automatically estimate the visibility from panoramic images. The algorithm is based on Koschmieder's law, which relates apparent contrast of an object to its distance from the observer. Local contrast information is computed from image patches using a standard measure for human contrast perception.

Multi-camera systems are increasingly used in both consumer and industrial applications. One example is the mobile phones market [Dual1]. The images from different cameras are combined in different ways in order to: generate the depth map to blur the image background (e.g. HTC), capture more light and reduce noise (e.g. Huawei), create wide field of view and reduce distortions (LG), provide the zoom-in capability (iPhone).

Another example of commercially available multi-camera system is the super resolution camera array which allows to capture images of moving subjects at a very high level of details [Eoptis2].

Multi-camera arrays are also used for depth estimation [Javidnia17]. In this particular example the proposed framework utilizes analysis of the local Epipolar Plane Image to initiate the depth estimation process. The estimated depth map is then processed using Total Variation minimization based on the Fenchel-Rockafellar duality.

Multi-camera systems can be also used for atmospheric visibility estimation [Du13] and high dynamic range microscopy [Javidnia17].

Multi-purpose multi-camera array was made by Stanford Computer Graphics Laboratory and consists of 100 CMOS-based cameras [Stanford3]. Their multi-camera system can function in many ways, depending on the arrangement and aiming of the cameras. The arrangement needs to be physically changed.

We intend to use a multi-camera system to enhance images in a new adaptive way e.g. to adaptively select a subset of cameras to obtain the required image quality.

Regarding the image enhancement approaches for visibility improvement, apart from the image processing techniques based on the standard filters for noise removal, contrast correction, histogram equalization, etc., there are also more sophisticated methods. The system developed for NASA fuses images from standard and infrared cameras [Hines05]. In [Kalkofen07] some visual key features such as edges are overlaid on top of original image to enhance the depth perception of the focus objects. Focus objects are not only overlaid on top of the video image, but they are partially occluded by key features from context objects. Another approach that overlays an image of detected edges on the original in order to augment the Google Glass user's perception is presented in [Hwang14]. Others try to correct optical defects of human eyes, especially defocus, by overlaying a compensation image on the user's actual view so that the filter cancels the aberration [Itoh15].

We intend to develop and combine new image enhancement approaches that can be applied to adaptively enhance the image based on the different visibility levels. In addition, the enhancement will utilize the information from more than one camera (in different ways) for further improvement.

The existing single image sensors struggle to capture image data in low light conditions. This in turn makes it difficult to track, detect, and identify targets, for example.

In order to capture sufficient light in low light conditions, the exposure time or the sensor sensitivity can be increased. However, the increased exposure time would introduce motion blurs for moving objects and lead to significant degradation of image quality while increasing the sensor sensitivity would exaggerate the ambient random noise fluctuations.

Hence, all existing types of camera sensors have their advantages and disadvantages (see Table 2).

Table 2 – Comparisons of camera sensors. Adapted from [Li11].

Type	Advantages	Disadvantages
High Speed	Fast shutter	Low image resolution Require large data bandwidth
High Resolution	Rich spatial details	Blurs for fast motion
Multispectral	High contrast High dynamic range	Require special equipment

Below we will discuss adaptation strategies for visibility conditions in a system of cameras, for a design-time model of an adaptive camera system and then for visibility conditions in a physical prototype i.e. runtime.

Adaptation strategy and enhanced vision in design-time

We to use an array of high resolution cameras with relatively short exposure time to alleviate the aforementioned problem. The images taken by the synchronized camera system will be fused after their registration in order to de-noise the resultant image, see [Zhang09] for example of image de-noising from multiple views.

The cameras can be connected to each other using the binary tree topology, for example [Wilburn04]. One camera can then be designated as root camera which would generate clocks and triggers that would be propagated to other cameras in order to synchronise them.

We would like to have the control of the synchronisation levels. For example, a very good synchronisation would be necessary for de-noising of images with moving objects, and slightly asynchronous cameras can be used for de-noising and moving object removal (when taking pictures of the ocean floor, for instance).

We intend to use tightly packed high-definition cameras focusing on the same scene arranged into an array grid. One of the simple but effective de-noising algorithm which can be also considered as a super-resolution method is the averaging of registered images.

One of the potential novel problems to solve would be the automatic adaptation of the image resolution (noise-level) to the lighting conditions which could be due to the weather change, for example. Thus, based on the light sensor readings and the experimentally predetermined image qualities corresponding to specific lighting conditions, the required number of camera sensors would be activated. Alternatively, a user may request a specific image quality.

Regarding the potential optimisation problem to solve, one can try to minimise the throughput or response time of the camera system, while maximising the image quality (signal to noise ratio). The image quality improves with the higher number of activated cameras which on the other hand will affect throughput and response time.

One type of the trade-off could be related to the performances of computer vision algorithms e.g. edge detection, object detection and tracking and the image quality and response time of live video streaming. The image quality improves with the higher number of activated cameras, but at the expense of increased response time.

As aforementioned, different number of active cameras requires different throughput levels and will produce images of different quality. The more cameras active, the better the image quality can generally be obtained, however resulting in higher throughput burdens. Second, different compression algorithms and compression ratios will also result in different throughput, image quality (if a lossy compression is used), and different times required to compress. CERBERO technologies *DynAA Simulation Model* and *AOW Optimizer* will be used to simulate different camera configurations in terms of different number of cameras activated and different compression algorithms and compression ratios. The results from different DynAA simulations will be fed into the AOW Optimizer in order to find the optimal configurations with respect to the key performance

indicators, such as image quality, response time, and throughput. Figure 11 represents the design-time model for the adaptive camera system.

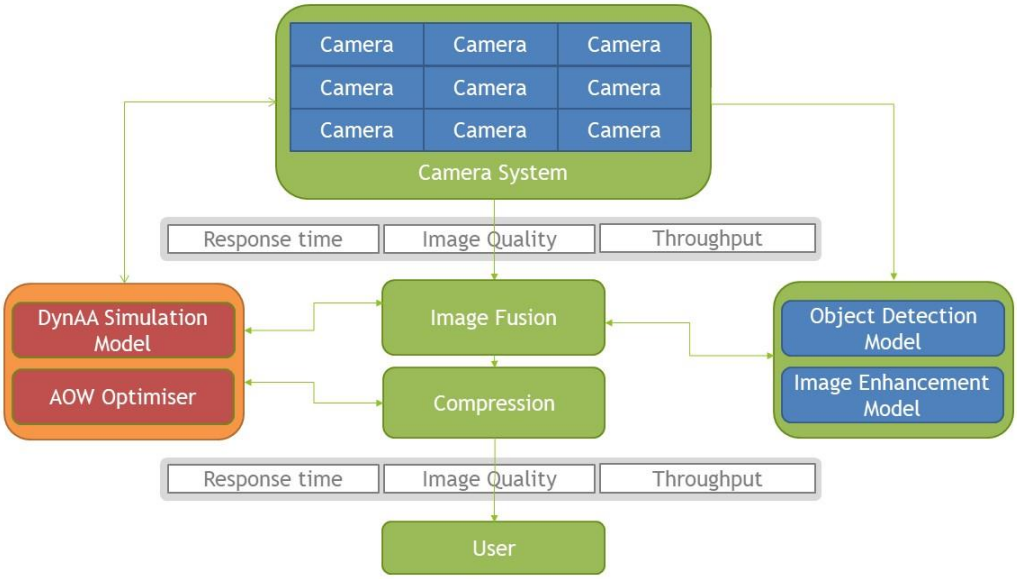

Figure 11 – Enhanced vision – design-time model.

Adaptation strategy and enhanced vision in a physical prototype

The physical prototype of the camera system will adapt its footage to visibility conditions in real time by

- Overlaying the edges detected by Canny edge detector [Canny1986] on top of the original image
- Fusing the novel edge detector with the original image

Adaptation to visibility conditions may also involve automatic assessment of environmental visibility. In the simplest case this could be based on the measurements from illumination sensor. The adaptation would then take the form of automatic adjustment of brightness of camera lights, or the colour histogram equalisation.

Initially, there could be two HD cameras within the adaptive camera system physical prototype. The use of two cameras as opposed to one is mainly for improved edge detection, and noise removal. Two cameras can also provide additional depth information to help measure the visibility conditions. Both cameras, after the calibration process, will monitor the same area of the environment from two different perspectives. Figure 12 presents the adaptation strategy for the aforementioned camera system.

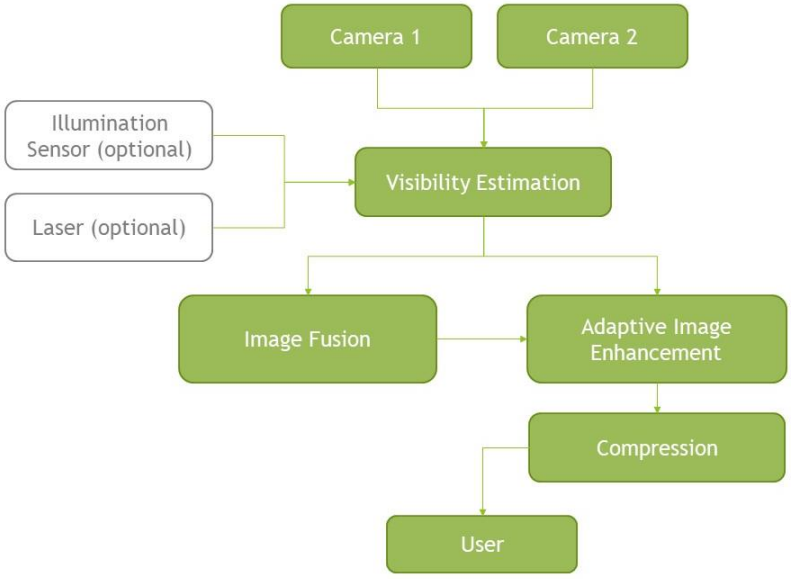

Figure 12 – Enhanced vision – physical prototype.

The adaptivity of the camera system can be based on the automatic detection of the visibility conditions. Some indications of the poorer visibility conditions are related to the noise and blurriness of certain areas in the image. The blurriness levels can be measured by transforming an image into the spectral space (FFT) and investigating the high and low signal frequencies. Another method for blurriness measurement could be based on the variation of Laplacian, for example. We can distinguish local and global blur. When considering local blur, an image can be divided into sub-images and the levels of blurriness estimated automatically for each region.

The influence of the edge detected image and the original one in the fusion of both can be adjusted to correspond to different visibility levels, thus de-blurring and de-noising the image in an adaptive manner.

The novel edge detector is based on the bilateral filter [Tomasi1998] and eight directional derivatives. As opposed to the Gaussian filter, the bilateral filter takes into account the variation of pixel intensities to preserve edges when removing the noise.

5.2.2. Adaptive hybrid image retrieval model

The purpose of the adaptive hybrid image retrieval model is to enable the image collection search based on the combination of different types of information about an image and to further improve the performance by fusing the features in an adaptive manner.

Search engine developers have realized that the same standard search method cannot be used for all queries. Users can enter an infinite number of queries representing a wide range of information needs. Good information retrieval engines should be able to interpret users' queries and predict their intentions. It should then apply an appropriate search strategy and return relevant results. In summary, a successful search mechanism needs to adapt to each individual user's query. To adapt to individual needs, context

information can be used, such as time, location, interest [Goker04]. Context information can include sensor-based information covering aspects of the physical environment and be used both for user queries and in mobile sensing in general. Approaches to adaptation can include having a context middleware [Myrhaug04, Yürür16] where the context-aware middleware combines a representation of context and sensing of the environment and user – in other words is akin to the adaptation manager. Just as the "decision support system" is an implementation of adaptation manager, we can use context-aware middleware in the same kind of way.

The hybrid image retrieval model is used to retrieve relevant images from the data collected by the cameras based on the combination of various features, in this example text and visual features. There is not much point in storing collected data if it cannot be retrieved – this can be compared to throwing things into a black hole. In the field of image retrieval, combination of various features related to the same image usually improves the retrieval mainly due to the correlation and complementarity of the different types of information. An image is an information object that can be described by different visual features, textual features, metadata, etc.

In the early stage of image retrieval research, librarians had to attach keywords to each image in order to retrieve relevant images with text retrieval techniques. Nowadays, however, manual labelling becomes infeasible due to the increasing size of the image collections. To circumvent such obstacle, content-based image retrieval (CBIR) which uses visual features to measure the content similarity between images, has emerged. Typical visual features include colour histogram, texture and shape, etc. An image is represented as a vector in a feature space. For example, each dimension in a colour histogram space corresponds to a colour bin along channels R-G-B or H-S-V, and the value of an image on each dimension is the normalized number of pixels in the image falling into the corresponding bin. The similarity between two images can be measured based on how close their corresponding vectors are in the feature space, e.g. through the Cosine function. Nevertheless, even the start-of-art CBIR techniques can only achieve a limited performance because of the semantic gap between the content and its high level semantics. Given that more and more images and multimedia documents contain both visual content and certain amount of text annotations (e.g. tags, metadata, text descriptions, etc.), combining the textual and visual features of images for image retrieval has recently attracted increasing attention.

Global approaches find it hard to capture all the properties of an image; therefore, the implemented local features are based on the “bag of visual words” approach. The first step in the “bag of features” method is to localize the points of interest (point-like, region-like) by using corner or blob detectors. Other sampling techniques include random and dense sampling. The second step involves the representation of regions around the sample points in a form of multidimensional vectors. There are various existing descriptors, the SIFT (Scale Invariant Feature Transform) [Lowe99] being one of the most widely used. The initial extraction is performed on a training set of images and the K-means clustering is applied to it. Each cluster will correspond to one “visual word”, a local pattern. Finally, each image in a data collection can be characterized by a histogram of “visual words” counts.

The relevance feedback is the user feedback related to retrieval results that is used to adaptively improve or narrow down the search.

It has been shown that the query can be correlated with its context to a different extent [Teevan05, Goker09]. Our fusion model uses an adaptive weighting scheme where the respective weights associated with the query and its context are automatically modified, depending on the relationship strength between visual query and its visual context and textual query and its textual context; the number of terms or visual terms (mid-level visual features) co-occurring between current query and its context represented as relevance feedback.

Wu et al. [Wu12] implement an adaptive data fusion method with dynamically adjustable weights. Two methods for the weight updating are investigated, namely "performance square" updating and a mixture of the aforementioned and linear regression analysis. The model combines evidence from different sources but do not incorporate any user feedback.

Wang et al. [Wang12] proposed an adaptive weighting approach to improve the current statistical context-sensitive retrieval model. First, the so-called "potential for adaptability" is investigated, the performance gap between the context-sensitive model with fixed weights and the one with adaptive weights, to show that the system can really benefit from having query-specific weights. Support vector regression is then applied to build a weight-prediction model, which enables a more flexible combination of current query and its context.

Most approaches that try to adapt the weights corresponding to query and its context have the linear combination of the relevance scores at their core. There are many different approaches to adjust the weights in a linear model. Machine learning can then be used to dynamically change these weights. For example, [Xia16] address the issue of search results diversification by data fusion. The authors assessed using differential evolution to learn weights for the linear combination method. Experiments with three groups of data show that differential evolution performs better than heuristic-based weighting schemes.

Our adaptive weighting approach differs from the above in that it represents a hybrid approach; it is not mono-modal. Moreover, we use a hybrid approach that takes into account inter- and intra-correlations between feature spaces and combines them in the context of user feedback, which is different from simply combining them in an ad-hoc manner. We use two notions of user feedback, visual and textual.

We will measure the strength of the relationship between the query and its context by computing the similarity between co-occurrence matrices corresponding to the query and its context (feedback images). The higher the number of terms or visual terms (mid-level features) co-occurring between current query and the context, the stronger the relationship and vice versa.

Let us assume that the relevance feedback is given after the first round retrieval to refine the query. The adaptation of the fusion model can be interpreted in a following way:

1. weak relationship between query and its context, context becomes important. We adjust the probability of the original query terms; the adjustment will significantly modify the original query.
2. strong relationship (similarity) between query and its context, context will not help much. The original query terms will tend to dominate the whole term distribution in the modified model. The adjustment will not significantly modify the original query.

5.2.3. Adaptive computation precision

This activity uses the concept of parameterized dataflow to build systems with a native support for computation precision adaptation. Precision adaptation is an emerging research topic included within the domain of approximate computing [Nogues16]. It consists in changing the quality of a processing (for instance the length of a filter, the number of bits in data representations, the sensor sampling rate) to match the minimal required QoS and minimize the processing cost. Within this activity, processing is adapted at runtime for respecting the applicative Quality of Service while minimizing resources. For this purpose, moldable parameters are studied. They define a set of potential values that the runtime manager can explore to adapt precision.

5.3. Information fusion methods to enable adaptivity

Information fusion is a combination of different types of information in order to obtain more reliable and accurate information.

In computer vision, for example, image fusion is the process of combining relevant information from two or more images into a single image. The resulting image should be more informative than any of the input images. Different combination methods will also produce different effects like noise removal, panoramic view, edge enhancement, etc.

Information fusion utilizes all available information at multiple abstraction levels (measurements, features, decisions) to maximize an expert system's performance. Table 3 presents the summary of all the fusion strategies to be used in the CERBERO project, including the novel fusion strategies.

Table 3 – Summary of fusion strategies

Fusion strategy	Description Summary
Image fusion for image enhancement	The new fusion method combines novel edge detector with the original image to enhance the camera footage
Novel image fusion model for depth information	The images from different cameras could be fused in order to create the depth map of the environment. The fused data could be used for image enhancement by progressively blurring the background for example, and for enhanced navigational capabilities
Image fusion for image de-noising	The images from multiple cameras can be fused to remove noise (synchronised cameras) or see through obstacles (asynchronous cameras)
Novel fusion method for hybrid image retrieval	Different types of information about an image needs to be fused in order to retrieve relevant images from the collection. The proposed fusion incorporates inter and intra feature correlations for further improvement
Novel fusion of the frame difference based and feature-based methods for detecting and tracking moving objects	Different object detection and tracking methods could be fused to overcome the limitations of individual approaches, e.g. frame difference with feature-based methods. Can be used for video augmentation, for example.
Fusion for computation precision	Fusion/combination of different parameters influencing

adaptation	computation is necessary to enable computation precision adaptation

5.3.1. Image fusion

Image fusion is needed for the proposed adaptation strategies and image enhancement approaches. It consists of different strategies for combining images from camera sensors.

Image fusion for image enhancement

We combine the images from the novel edge detector with the original image in order to enhance the underwater visibility for both the robot operator and computer vision algorithms. The data fusion takes the form of a linear combination method.

Here, the weight *w* could represent the illumination measurement coming from the illumination sensor which is scaled to a numerical value from interval [0,1]. This type of weighting would represent a continuous form of adaptation of the camera footage to the different visibility conditions levels; the bigger the weight, the higher the level of image enhancement to counter the poor visibility conditions.

The aforementioned data fusion process affects the entire image because no thresholding operation is used on the edge detected image. This combination method results in the sharpening, noise reduction, and the edge enhancement of an image.

Image fusion for depth information

Two cameras positioned in a straight line within some distance from each other can be used to measure the atmospheric visibility based on the contrast between the target and its sky background, and the distance of the target. For example, the visibility can be calculated by determining the contrasts of a target with its sky background in the two digital photographs, as well as the distance between the locations where the photos were taken [Buades10] (see Figure 13).

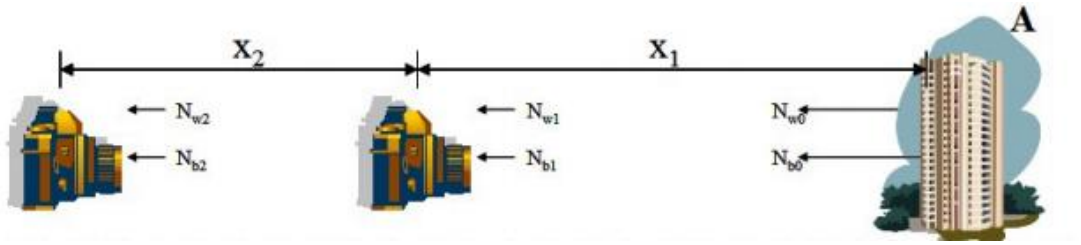

Figure 13 – Visibility estimation based on the dual camera system. Adapted from [Buades10].

Combination of images from two cameras can be used to create the depth map of a scene. Figure 14 presents the mathematical foundations of the idea.

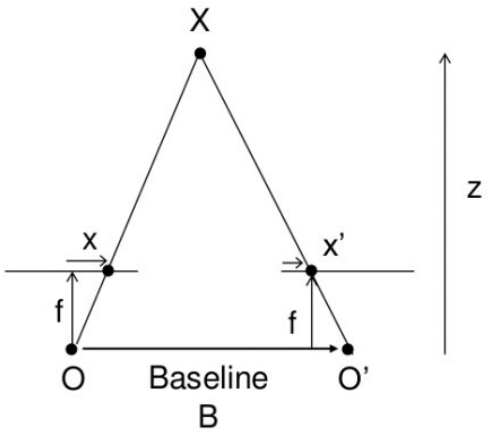

Figure 14 – The depth information from two cameras. Adapted from [Li11].

The depth information can be used to adapt imagery by progressively blurring the background for example, and for enhanced navigational capabilities of the robot.

Image fusion for image de-noising

The image fusion model used for removing noise from images is based on image averaging. The images are generated by multiple synchronised or slightly asynchronous cameras. They are all registered in order to depict the same scene and were taken simultaneously.

The averaging algorithm of registered images would allow us to reduce the noise by the factor of square root n where n denotes the number of cameras [Buades10]. Thus, for four cameras the factor would be 2, and for nine – 3.

The registration (calibration) process of the images taken by multiple cameras can be performed from the software perspective by detecting a number of (e.g. 50) the most characteristic points in the images (keypoints) using one of the corner detectors (e.g. Harris) [Kim04]. Next, the areas around the keypoints can be described in a vector form using a SIFT (Scale Invariant Feature Transform) descriptor which is based on directional image derivatives. The search for the corresponding images patches can be performed by similarity/dissimilarity measurement between the descriptors – e.g. Euclidean, Manhattan distance, cosine of the angle etc. Because matches may be inaccurate, common homography algorithms use a Random Sample Consensus (RANSAC) to remove outliers from the list of matches. The homography is considered a success if sufficient number of inliers is found, e.g. 25 [LiKamWa13]. The displacement vector for each feature (descriptor) can then be used to find the transformation (e.g. affine transformation in the form of 3x3 matrix using least squares method – rotation, scaling, translation) so that the images can be represented in the same coordinate system.

After the registration process, the images can be combined within the camera system by the averaging algorithm (super-resolution), for example [Buades10, IMAV]. In the case of a very good synchronisation the algorithm would de-noise the images, while for the slightly asynchronous cameras it can also remove moving objects (when taking the images of the ocean floor, for instance).

5.3.2. Fusion for hybrid image retrieval

Different types of information about an image need to be combined to retrieve relevant images from the database. In order to utilize the correlated and complimentary information, we also incorporate inter and intra correlations between feature spaces.

The existing fusion approaches can be divided into early and late fusion strategies. Early fusion is the combination at representation level while the late fusion is the combination at decision level.

The most common early fusion technique is concatenation of visual and textual representations. Some models incorporate the tensor product to combine the systems [Wang10]. Tensor product captures the relationships between all dimensions of different feature spaces.

In the case of late fusion, the most widely used method is the arithmetic mean of the scores, their sum (referred to as CombSUM [Csurka12], [Ballas14]), or their weighted linear combination. One of the best performing systems on the ImageCLEF data collection, XRCE [Mensink10], utilizes both (for comparison purposes) early (concatenation of features) and late (an average of scores) fusion approaches. Another common combination method, referred to as CombPROD in the literature ([Csurka12], [Ballas14]), is the square of the geometric mean of the scores - their product. It has been argued that the major drawback of the late fusion approaches is their inability to capture the correlation between different modalities [Mensink11].

It has been discovered, however, that specific early and late fusion strategies can be interchangeable [Kaliciak14].

Other combination methods involve a combination of late fusion and image re-ranking [Clinchant11]. Because the first stage is based on the pre-filtering of the collection by text, the model is referred to as the semantic combination.

Some fusion strategies can be also classified as intermediate fusion [Bhowmik14]. They simultaneously learn individual classifier and combination classifier weights [Zhang11], and this process happens at various levels of learning.

The fusion approach that can be easily modified to incorporate the user feedback is based on the so-called transmedia pseudo-relevance mechanism [Csurka12]. This is a feedback query expansion, usually based on textual query expansion (in most papers, e.g. [Depeursinge10]). Typically, textual annotations from the top visually-ranked images (or from a mixed run) are used to expand a textual query.

The hybrid relevance feedback model with adaptive weights was inspired by Quantum Mechanics, where the combined system is represented as a tensor product of density matrices.

The hybrid relevance feedback model is defined on a Hilbert space which can be thought of as a natural extension of the standard vector space model, with its useful notions of subspaces and projections. The model is based on the notion of co-occurrence and the tensor operators, and incorporate different types of correlation between feature spaces. Inter correlations are captured by the tensor operator and the intra correlations are modelled by co-occurrence matrices.

5.3.3. Fusion of the frame difference based and feature-based methods for detecting and tracking moving objects

In general, approaches to detecting and tracking objects are often based on frame-difference, background subtraction (e.g. colour based image segmentation), and methods based on the optical flow.

The colour based methods work well if the colour of an object is known and is relatively uniform. If these assumptions are met then the colour-based object detection and tracking can be also invariant to camera movement. The frame difference based methods on the other hand do not rely on object colour but are quite sensitive to camera movements.

Both approaches can be associated with low to medium computational cost. We would like to fuse these low cost approaches in order to alleviate some of the drawbacks of individual methods and combine the strengths of both.

Additionally, we would like to investigate the use of multiple camera system in object detection and tracking.

5.3.4. Fusion of time-series sensor data

Most CPS systems will contain a large set of different sensors. Some of this sensors will provide continuous stream of data (like cameras or microphones) but most sensors will produces sensor readings at often very different frequencies (which could be once every few seconds but also many times per second, often referred to as sample rate in Hz). In case data from multiple sensors is to be used to perform complex data analysis processes, it is important that the available data from the different sensors is pre-processed before it can be fused into input data for the analysis. This mainly applies to time series data and not so much for continuous data stream (where sample rate is often determined by the media protocol used). The required pre-processing will depend on the complexity of performed data analysis algorithms in the adaptation process. Some algorithms will not work properly if they directly receive raw data from sensors, especially not if sample rate between multiple signals differs or if the data contains noise (e.g. sudden high frequency glitches) or gaps (e.g. periods without reading because value did not change).

For many data analysis algorithms (like anomaly detection or trend analysis), the different input signals need to be synchronized in time and frequency (so the algorithms can use evenly distributed input data). For this fixed time intervals need to be defined based on which input data is re-sampled. It could be that some sensors provide data at much higher frequency than others or some sensors can provide sudden bursts of data. Missing data points will need to be interpolated and for other sets of data points an average will need to be calculated to replace the given set. The pre-processed data from multiple sensors can simply be fused into a single and clean data set with evenly sequenced data points for all input sensors. This will simplify data analysis process. Based on type of analysis required, it could be that some of the raw data will also need to be saved (e.g. for analyzing the high frequency bursts of data which are filtered out of the clean data set).

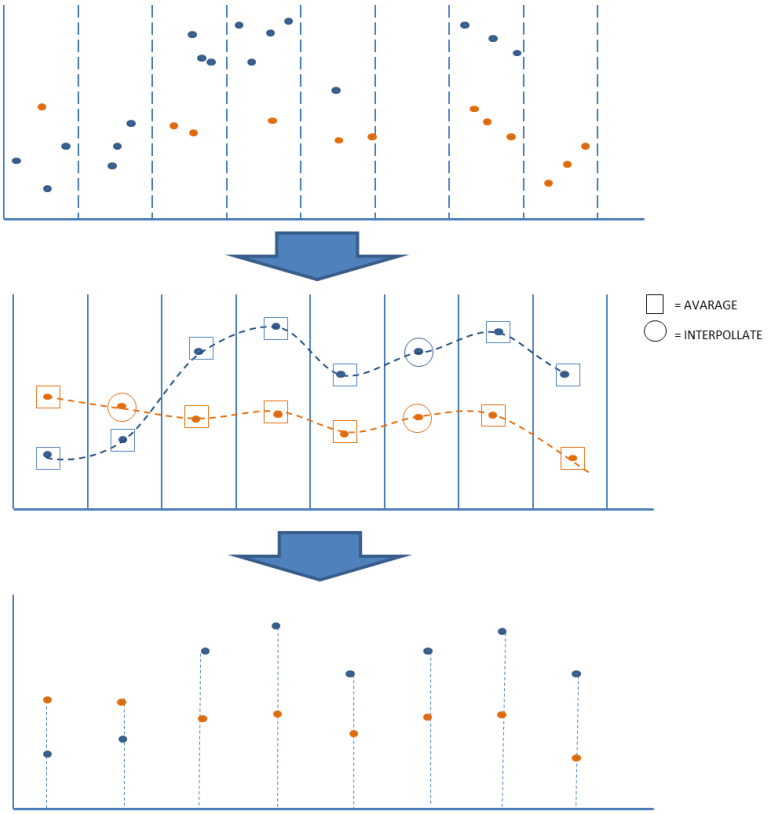

Figure 15 – Fusion process of two time series sensor data

The fusion steps of integrating time series sensor data would be:

1. Determine time intervals (based on required precision and algorithms);
2. Generate values, which could be accomplished by:
    - Generation of average within time interval for each sensor data set in case multiple values are available, or
    - Interpolating a value in case value is missing in given interval (based on values in previous and next interval(s));
3. Fuse data into single equally sampled time sequenced data set.

5.3.5. Fusion for computation precision adaptation

In the CERBERO cyber-physical context, fusion/combination of different parameters influencing computation is necessary. This fusion is performed as part of the processing management. It partially relies on PiSDF *configuration actors*, i.e. pieces of computation that transform a set of data into a parameter capable of influencing future execution.

The CERBERO adaptation manager is aware of several KPIs from both cyber and physical worlds and takes decision based on these two sets of information. Sensed KPIs may come from e.g. a hardware timer connected to a quartz-generated clock, an energy sensor (shunt resistor with current and voltage measurement), cameras observing the environment, an Inertial Measurement Unit (IMU), etc. All these sensed KPIs have the

capability to make the application QoS-aware and to drive previously evoked computation precision adaptation.

This adaptation support is suited for streaming applications, such as the video processing pipeline from CERBERO Ocean Monitoring use case (high throughput) and the robotic arm control loop from CERBERO Planetary Exploration use case (low latency).

5.4. Time synchronization

Time synchronization is the first step in adaptive sensor fusion, and is not always straightforward to accomplish. The system observes the world through a sensor, and as such is dependent on proper time synchronized observations to build an accurate state; whether the observing system is a real-time system or a simulation. A simulation may be considered as a special type of sensor, in which the information is not derived from a physical phenomenon, as is usual for a sensor, but from a simulated phenomenon instead. In Table 4 the different combinations for sensor / system combinations are shown with the corresponding types of required synchronization mechanisms.

Table 4 Type of synchronization required for sensor/system combination

		Sensor	
		Physical	Simulation
System	Real-time	Synchronization not required	One-way synchronization
	Simulation	One-way synchronization	Two-way synchronization

One-way synchronization is usually a matter of halting the simulated system until the real-time or physical data is available when the simulation is faster than real-time, whereas the information needs to be buffered in the simulation is slower than real-time. When both systems are simulated, and two way synchronization is required, it is slightly more difficult since the speed of the faster simulator needs to be adjusted to the slower simulator.

Generally, two types of simulation environments can be defined. Firstly, in *continuous simulators* the models are continuously updated and integrated according to a global time representation, which increments in fixed time steps. Secondly, in *discrete event simulators* time advances in discrete steps and the system is considered to be static in between two events. If a combination of a discrete event simulator and continuous time simulator are used, special care needs to be taken to synchronize the time between both environments. Gheorghe *et al.* [Gheorg06] propose a formal method to design hybrid continuous/discrete co-simulation systems, which was further refined by Nicolescu *et al.* [Nicol07].. A graphical representation of the proposed time synchronization is shown Figure 16. Both simulation events need to determine what events are relevant as synchronization data, i.e. events that contains information that should be included in the other simulation. These *synchronization events* then determine when and how the simulations should synchronize.

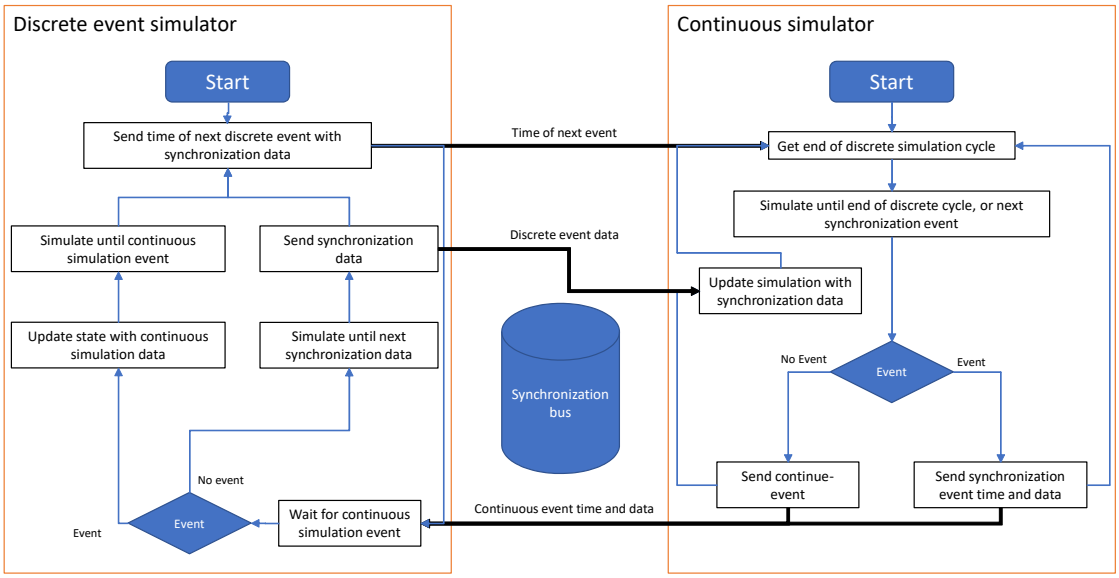

Figure 16. A schematic representation of the process of synchronization and the interfaces of discrete event and continuous time simulators

When using sensor adaptation in a scenario in which time-synchronization is required, usually the sensor is a continuous system. When the observing system is also a continuous simulation, the co-simulation can be achieved by simply setting the sensor sampling period to the simulation time step size, and every sample is provided the simulation. When the simulated system uses a discrete event simulator, the synchronization events are the sensor measurements and possible sensor control actions such as querying an active sensor, or switching a passive sensor on or off.

6. Intended Support for Adaptation Strategies in Use Cases

6.1. Adaptation in the Space Exploration use case

Robotic arm adaptivity for space exploration applications is a very challenging problem, especially in such dynamic, uncertain and harsh environments. For this reason, adaptation methods must be provided to guarantee the success of the robotic arm movement.

Final end effector position is provided by inverse kinematics equations to determine the joint parameters to reach the desired position. WidowX robotic arm, which is selected for demonstration purposes, includes six actuators that act as sensors, providing information about position, speed, torque, etc. Controllers that track the trajectory by reading every actuator position are able to detect deviations between reached and desired position. These deviations may be due to obstacles, maximum torque for the current load, circuit malfunction due to radiation effects and loss of communication with one or more actuators.

CERBERO technologies provide Self-Adaptation manager support for adaptive motion planning and Self-Healing. These technologies are the following:

- ARTICo^3: Inverse kinematics equations have multiple solutions to reach the desired end effector position. ARTICo^3 provides parallel methods of calculation for inverse kinematics equations and selects which one best fits the environment, based on joint parameters measurements.
- ARTICo^3/MDC: Various reconfiguration types are needed to adapt to harsh environments. These tools provide fault monitors and hardware reconfiguration.
- PAPIFY: Power measurement and estimation are required for an autonomous system. Inverse kinematics solutions could be selected to minimize power consumption, e.g. minimizing angle variation of each joint.
- Preesm/Spider: Reinforcement learning provides decision methods to solve a variety of problems based on inputs and previous experiences. If communication with a joint has been lost, reinforcement learning provides a solution to adapt to this problem, taking into account before solving inverse kinematics equations.

In the context of sensor fusion, the robotic arm use case has the following characteristics:

The way on how to combine internal and external sensors for position, velocity and acceleration estimation in real time is crucial for space exploration robotics arms. Methods based on joint position or Inertial Measurement Unit (IMU) are very useful.

Robotic arm measurements are limited by joint position and speed; based on the kinematics of the robotic arm, it is possible to extract data with 20 Hz rate. Each joint actuator is able to provide information about position, speed, torque or current load.

Most of IMUs combine accelerometer, gyroscopic and magnetic sensors, which provides useful information about the robotic arm movement as attitude, angular rates, linear velocity and position relatively to a reference.

By the combination of these measurements, constant speed movements can be performed. A control algorithm must analyze these measurements and produce a joint position/speed correction in order to achieve a final end effector constant speed movement.

6.2. Adaptation in the Smart Travelling use case

In the Smart Travelling use case the following adaptation strategies will be applied:

- **Self-adaptation through the decision support function:** The decision support function in the Smart Travelling use case acts as the Self-Adaptation Manager, responsible for any required adaptation relevant for the driver. Like a software agent the manager will follow the 3T model and follow the sense, plan and act paradigm. By monitoring the status of the car, the environment and the user preferences and goals, the decision support function will need to determine if adaptation is required. In case, for example, the GPS location of the car indicate that the car no longer following the planned route, the decision support function could initiate a new route planning. In case monitored battery power is reduced more than predicted, the decision support function could adapt by recalculating the prediction based on new information received (e.g. more intense traffic, higher speed of travelling, lower or higher outside temperature, which will influence heating/cooling and thus battery consumption) and decide to propose the driver alternative routes or charging options.
- **Time synchronisation:** For adapting DynAA to work as a system in the loop, adaptation is required for connecting the real time sensors to the simulations run inside DynAA. The most important problem to tackle is synchronization of time between simulation run in DyNAA and the sensors running outside of DynAA (in the use case Smart Travelling the sensors simulated inside SCANeR). Within the CERBERO project the planned adaptation of DynAA will use the time synchronisation solution as described in paragraph 5.4.
- **Fusion of time-series sensor data:** In order to analyse the results of the simulation runs, data analysis processes will be executed, which will need pre-processed fused input data from all the sensors (and simulation modules). As CRF uses the simulator to interpret events related to the behaviour of the driver during the simulation, it is important that events form all the different simulation modules and sensors can be correlated. To interpret specific events in the simulation (e.g. sudden eye movements), data is required in given time intervals and the data from the different sources needs to be synchronised in time to be able to draw conclusions on for example causes of specific events.

6.3. Adaptation in the Ocean Monitoring use case

In the Ocean Monitoring use case the following adaptation strategies will be applied:

- **Self-adaptation through hierarchical organization of adaptation processes.** Instead of a single adaptation manager, adaptation will be organized as modular aspects, similar to NIST's 4D/RCS-4 [Albus02]. The adaptive camera, for example, will contain its own adaptation manager, model, engine, and monitors. The power system will contain a separate manager, model (using DynAA in this case), engine, and monitors. If power is running low, the power component can signal the camera component to reduce its energy requirements. Similarly, if the camera detects a sensor failure, it can signal the system as a whole to end tasks early.

- **Context-aware middleware as an adaptation manager**. At an information level, there is a close connection between context-aware applications and adaptation [Yürür16], and – as a context-aware system component – a context-aware middleware will play the role of the adaptation manager for optimizing image relevance. This will enable the ocean monitoring case to integrate information adaptation into a cyber-physical system.
- **Time synchronization.** Just as in the smart travelling use case, the ocean monitoring use case depends on a time-synchronized DynAA as the embedded model in the adaptation system. So, while the time data sources may not be identical, there is the same need for synchronization between DynAA model time and external environment and system time.
- **Adaptation of the underwater camera system.** The camera systems in the ocean monitoring case involve several aspects of adaptation, as outlined in sections 5.2 and 5.3. For example, the colour balance could be based on the water depth measurement or the measurement of the amount of red colour in the colour histogram (a sensor). The red colour at different depths gradually disappears (this can be modelled), therefore the underwater photographers need to use camera filters to compensate for the loss of colour. Traditionally, the filters are physically changed at different depths. We are going to automatically compensate (adapt) for the loss of the red colour by measuring the water depth or the amount of red colour in the histogram, and using multicoloured lights (controlled by an adaptation engine) or histogram manipulation techniques..

7. References

[Amaga08] M. Amagasaki, R. Yamaguchi, M. Koga, M. Iida, and T. Sueyoshi. An embedded reconfigurable IP core with variable grain logic cell architecture. Int. J. Reconfig. Comp., 2008:180216:1–180216:14, 2008.

[Ballas14] Ballas N, Labbé B, Le Borgne H, Gosselin P, Picard D, Redi M, Merialdo B, Mansencal B, Benois-Pineau J, Ayache S, Hamadi A. Irim at TRECVID 2014: Semantic indexing and instance search. InProceedings of TRECVID 2014 Nov 10.

[Barbosa16] D. Barbosa, A. Lopes and R. E. Araújo, Sensor fusion algorithm based on Extended Kalman Filter for estimation of ground vehicle dynamics. *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 1049-1054, 2016.

[Bergmann13] N.W. Bergmann, S. Shukla and J. Becker, QUKU: a dual-layer reconfigurable architecture, ACM Transactions on Embedded Computing Systems, 2013.

[Bosse17] T. Bosse, L. Breebaart, J. Van Diggelen, M.A. Neerincx, J. Rosa and N.J. Smets Developing ePartners for human-robot teams in space based on ontologies and formal abstraction hierarchies, Int. J. Agent-Oriented Software Engineering, Vol. 5, No. 4, 2017.

[Brant12] A. Brant and G.G.F. Lemieux, ZUMA: an open FPGA overlay architecture, IEEE Symposium on Field-Programmable Custom Computing Machines, 2012.

[Brogi06] Brogi, A., Canal, C. and Pimentel, A. On the semantics of software adaptation. Journal of Science of Computer Programing, vol. 61, pp. 136–151, 2006.

[Canal04] Canal, C., Murillo, J.M. and Poizat, P. Coordination and Adaptation Techniques for Software Entities. In: Malenfant J., Østvold B.M. (eds) Object-Oriented Technology. ECOOP 2004 Workshop Reader. ECOOP 2004. Lecture Notes in Computer Science, vol. 3344 pp. 133–147. Springer, Berlin, Heidelberg.

[Capalija13] D. Capalija and T.S. Abdelrahman, A high-performance overlay architecture for pipelined execution of dataflow graphs, International Conference on Field Programmable Logic and Applications, 2013.

[Cong14] J. Cong, H. Huang, C. Ma, B. Xiao and P.Zhou, A fully pipelined and dynamically composable architecture of CGRA, IEEE symposium on FPGAs for Custom Computing Machines, 2014.

[Csurka12] Csurka G, Clinchant S. An empirical study of fusion operators for multimodal image retrieval. InContent-Based Multimedia Indexing (CBMI), 2012 10th International Workshop on 2012 Jun 27 (pp. 1-6). IEEE.

[Desnos13] K. Desnos, M. Pelcat, J. F. Nezan, S. S. Bhattacharyya and S. Aridhi, "PiMM: Parameterized and Interfaced dataflow Meta-Model for MPSoCs runtime reconfiguration," 2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), Ag. Konstantinos, 2013, pp. 41-48.

[Dinechin14] B.D. de Dinechin, D.V. Amstel, M. Poulhies and G. Lager, Time-critical computing on single-chip massively parallel processor, Design, Automation and Test in Europe Conference, 2014.

[Diniz14] C. M. Diniz, M. Shafique, S. Bampi and J. Henkel, "Run-time accelerator

binding for tile-based mixed-grained reconfigurable architectures," 2014 24th International Conference on Field Programmable Logic and Applications (FPL), Munich, 2014, pp. 1-4. doi: 10.1109/FPL.2014.6927392

[Dobson04] Dobson, S. Component-Oriented Approaches to Context-Aware Computing. In: Malenfant J., Østvold B.M. (eds) Object-Oriented Technology. ECOOP 2004 Workshop Reader. ECOOP 2004. Lecture Notes in Computer Science, vol. 3344 pp. 84–93. Springer, Berlin, Heidelberg.

[Du13] K. Du, K. Wang, P. Shi, Y. Wang. Quantification of atmospheric visibility with dual digital cameras during daytime and nighttime. *Atmospheric Measurement Techniques*, 1;6(8):2121, 2013.

[Dual1] https://www.gsmarena.com/understanding_the_dual_camera_systems_on_smartphones-news-27516.php

[Eoptis2] https://www.eoptis.com/en/products/custom/gigapixel-camera-array

[Garlan02] Garlan, D. and Schmerl, B. Model-based adaptation for self-healing systems. In Procs. of the 1st workshop on Self-healing systems, Charleston, SC, USA Nov. 2002.

[Gat97] Gat, E. Three-layer Architectures, Artificial Intelligence and Mobile Robots, MIT/AAAI Press, 1997.

[Gheorg06] Gheorghe, L., Bouchhima, F., Nicolescu, G., & Boucheneb, H. (2006, June). Formal definitions of simulation interfaces in a continuous/discrete co-simulation tool. In Rapid System Prototyping, 2006. Seventeenth IEEE Int. Workshop on (pp. 186-192). IEEE.

[Graves14] N. Graves, S. Newsam, Camera-based visibility estimation: Incorporating multiple regions and unlabeled observations. *Ecological informatics*, 1;23:62-8, 2014.

[Hartenst01] R.W. Hartenstein. Coarse grain reconfigurable architecture (embedded tutorial). In Proceedings of ASP-DAC 2001, Asia and South Pacific Design Automation Conference 2001, January 30-February 2, 2001, Yokohama, Japan, pages 564–570, 2001.

[Heulot14] Julien Heulot, Maxime Pelcat, Jean-François Nezan, Yaset Oliva, Slaheddine Aridhi, et al.. Just-In-Time Scheduling Techniques for Multicore Signal Processing Systems. GlobalSIP14, Dec 2014, Atlanta, United States.

[Hines05] G. D. Hines, Z. U. Rahman, D. J. Jobson, G. A. Woodell, S. D. Harrah. Real-time enhanced vision system, *In Enhanced and Synthetic Vision*, vol. 5802, pp. 127-135, 2005.

[Hwang14] A.D. Hwang, E. Peli. An augmented-reality edge enhancement application for Google Glass. *Optometry and vision science: official publication of the American Academy of Optometry*, 91(8):1021, 2014.

[IEEE10] ISO/IEC/IEEE International Standard. Systems and software engineering – Vocabulary. In ISO/IEC/IEEE 24765:2010(E), pp.1-418, Dec. 15, 2010.

[Itoh15] Y. Itoh, G. Klinker. Vision enhancement: defocus correction via optical see-through head-mounted displays. *In Proceedings of the 6th Augmented Human International Conference*, pp. 1-8, 2015.

[Jain215] Adapting the DySER architecture with DSP blocks as an overlay for the Xilinx Zynq, ACM SIGARCH Computer Architecture News, 43(4), pp. 28-33. 2015.

[Javidnia17] H. Javidnia, P. Corcoran. Total Variation-Based Dense Depth from Multi-Camera Array. *Computer Vision and Pattern Recognition* ,2017.

[Jianzhong16] Jianzhong Sun, Hongfu Zuo, Kun Liang, and Zhixiong Chen, Bayesian

network-based multiple sources information fusion mechanism for gas path analysis", Journal of Propulsion and Power, Vol. 32, No. 3, pp. 611-619, 2016.

[Kalkofen07] D. Kalkofen, E. Mendez, D. Schmalstieg. Interactive focus and context visualization for augmented reality. *In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1-10, 2007.

[Kaliciak14] L. Kaliciak, H. Myrhaug, A. Goker, D. Song. On the duality of specific early and late fusion strategies, Information Fusion (FUSION), 17th International Conference on, 1--8, 2014.

[Kapre06] N. Kapre, N. Mehta, M. de Lorimier, R. Rubin, H. Barnor, M.J. Wilson, M. Wrighton and A. DeHon, Packet switched vs. Time multiplexed FPGA overlay networks, IEEE Symposium on Field-Programmable Custom Computing Machines, 2006.

[Kapre17] N. Kapre, Implementing FPGA overlay NoCs using the Xilinx UltraScale memory cascades, IEEE Symposium on Field-Programmable Custom Computing Machines, 2017.

[Katenka08] N. Katenka, E. Levina and G. Michailidis, Local vote decision fusion for target detection in wireless sensor networks. *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 329-338, 2008.

[Kell08] Kell, S. A Survey of Practical Software Adaptation Techniques. Journal of Universal Computer Science, vol. 14(13), pp. 2110–2157, 2008.

[Kramer07] Kramer, J. and Magee, J. Self-Managed Systems: an Architectural Challenge. In Procs. of the IEEE Future of Software Engineering conference, Minneapolis, MN, USA, May 2007.

[Laforest17] C.E. Laforest and J.H. Anderson, Microarchitectural Comparison of the MXP and Octavo Soft-Processor FPGA Overlays, ACM Transactions on Reconfigurable Technology and Systemsn, 10(3), 2017.

[Liang12] Y. Liang, K. Rupnow, Y. Li. D. Min, M.N. Do and D. Chen, High-level synthesis: productivity, performance, and software constraints, Journal of Electrical and Computer Engineering, 2012.

[Liu17] Y. Liu, X. Chen, H. Peng, Z. Wang, Multi-focus image fusion with a deep convolutional neural network. *Information Fusion*, 1;36:191-207, 2017.

[Lowe99] Lowe, David G. "Object recognition from local scale-invariant features." Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. IEEE, 1999.

[Mensink10] T. Mensink, G. Csurka, F. Perronnin. LEAR and XRCE's participation to visual concept detection task - ImageCLEF 2010, Proceedings of the 14th Annual ACM International Conference on Multimedia, 77--80, 2010.

[Mensink11] T. Mensink, J. Verbeek, G. Csurkay. Weighted transmedia relevance feedback for image retrieval and auto-annotation, Technical Report Number 0415, 2011.

[Myrhaug04] Myrhaug H., Whitehead N., Goker A., Faegri T.E., and Lech T.C. (2004). AmbieSense – a system and reference architecture for personalised and context-sensitive information services for mobile users. Second International Symposium on Ambient Intelligence, November 2004, Eindhoven, Netherlands, Springer Verlag. pp 327-338.

[Nicol07] Nicolescu, G., Boucheneb, H., Gheorghe, L., & Bouchhima, F. (2007).

Methodology for efficient design of continuous/discrete-events co-simulation tools. High Level Simulation Languages and Applications-HLSLA. SCS, San Diego, CA, 172-179.

[Nogues16] Erwan Nogues, Daniel Menard, Maxime Pelcat. Algorithmic-level Approximate Computing Applied to Energy Efficient HEVC Decoding. IEEE Transactions on Emerging Topics in Computing, Institute of Electrical and Electronics Engineers, 2016.

[Oreizy08] Oreizy, P., Medvidovic, N. and Taylor R.N. Runtime Software Adaptation: Framework, Approaches, and Styles. In Procs. of the 30th International Conference on Software Engineering, Leipzig, Germany, May 2008.

[Rohr06] Rohr, M., Giesecke, S., Hiel, M., Heuvel, W.J., Weigand, H. and Hasselbring, W. A classification scheme for self-adaptation research. In Procs. of the International Conference on Self-Organization and Autonomous Systems in Computing and Communications. Germany, Jan. 2006.

[Sourdis13] I. Sourdis, C. Strydis, A. Armato, C.S. Bouganis, B. Falsafi, G.N. Gaydadjiev, S. Isaza, A. Malek, R. Mariani, D.N. Pnevmatikatos, D.K. Pradhan, G.K. Rauwerda, R.M. Seepers, R.A. Shafik, K. Sunesen, D. Theodoropoulos, S. Tzilis, and M. Vavouras. Desyre: Ondemand system reliability. Microprocessors and Microsystems - Embedded Hardware Design, 37(8-C):981–1001, 2013.

[Stanford3] http://graphics.stanford.edu/projects/array/ [Last accessed 30/3/2018].33

[Sutter16] T. Sutter, F. Nater, C. Sigg. Camera Based Visibility Estimation. *Technical Conference on Meteorological and Environmental Instruments and Methods of Observation (TECO)*, 2016.

[Thomas12] Alexander Thomas, Michael Rückauer, and Jürgen Becker, "HoneyComb: An Application-Driven Online Adaptive Reconfigurable Hardware Architecture," International Journal of Reconfigurable Computing, vol. 2012, Article ID 832531, 17 pages, 2012. doi:10.1155/2012/832531

[Vivado17] Vivado Design Suite User Guide – Partial Reconfiguration UG909 (v2017.1) April 5, 2017

[Wang10] J. Wang, D. Song, L. Kaliciak.Tensor product of correlated text and visual features: a quantum theory inspired image retrieval framework, AAAI-Fall 2010 Symposium on Quantum Information for Cognitive, Social, and Semantic Processes, 109--116, 2010.

[Wang12] X. Wang, M. Yang, H. Qi, S. Li, and T. Zhao. Adaptive weighting approach to context-sensitive retrieval model, the 8th Asia Information Retrieval Societies Conference, 7675: 417-426, 2012.

[Wu12] S. Wu, Y. Xing, J. Li, and J. Bi. Adaptive data fusion methods for dynamic search environments, the 8th Asia Information Retrieval Societies Conference, 7675: 336-345, 2012.

[Xia16] Xia J, Xu C, Wu S. Differential Evolution-Based Fusion and Its Properties for Web Search. In Web Information Systems and Applications Conference, 2016 13th 2016 Sep 23 (pp. 67-70). IEEE.

[Yuan15] F. L. Yuan, C. C. Wang, T. H. Yu and D. Marković, "A Multi-Granularity FPGA With Hierarchical Interconnects for Efficient and Flexible Mobile Computing," in IEEE Journal of Solid-State Circuits, vol. 50, no. 1, pp. 137-149, Jan. 2015. doi: 10.1109/JSSC.2014.2372034

[Yürür16] Ö. Yürür, C. H. Liu, Z. Sheng, V. C. M. Leung, W. Moreno and K. K. Leung, "Context-Awareness for Mobile Sensing: A Survey and Future Directions," in IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 68-93, Firstquarter 2016. doi: 10.1109/COMST.2014.2381246

[Zhang11] W. Zhang, Z. Qin, and T. Wan. Image scene categorization using Multi-Bag-of-Features, Proceedings of International Conference on Machine Learning and Cybernetics, 4:1804--1808, 2011.

[Zhang15] Y. Zhang, QA. Zeng, Y. Liu, B. Shen, Integrated data fusion using dempster-shafer theory. *In Computational Intelligence Theory, Systems and Applications (CCITSA), First International Conference on*, pp. 98-103, 2015.

[Zhou17] Zhou, Y. and Chen, T. Software Adaptation in an Open Environment: A Software Architecture Perspective. 1st ed. CRC Press, 2017.