Multi-Partner Project: Key Enabling Technologies for Cognitive Computing Continuum - MYRTUS Project Perspective

Francesca Palumbo[1], Francesco Ratto[1], Claudio Rubattu[2], Maria Katiuscia Zedda[3], Tiziana Fanni[3], Veena Rao[4], Bart Driessen[5], Jeronimo Castrillon[6]
[1]*Università degli Studi di Cagliari,* [2]*Università degli Studi di Sassari,* [3]*Abinsula S.r.l.,* [4]*HIRO-MicroDataCenters B.V.,* [5]*TNO,* [6]*Technische Universität Dresden*

***Abstract*—The MYRTUS Horizon Europe project embraces the principles of the EU CloudEdgeIoT Initiative, integrating edge, fog, and cloud in a continuum of computing resources. MYRTUS intends to deliver abstractions, cognitive orchestration mechanisms, and a whole design environment to build and operate collaborative, distributed, heterogeneous systems. The goal is to provide high performance and play a crucial role in enabling energy efficiency and trustworthiness in nowadays systems.**

***Index Terms*—Computer hardware and architecture, Design environment, dynamic orchestration, Computing continuum, Interoperability, AI.**

I. Overview, Challenges and Objectives

The concept of *compute continuum* has been recently brought into the field to describe systems that are *more than the sum of cloud, edge, and Internet of Thing functionalities*, where computing breaks boundaries among layers *moving the computation* from the device to the farthest data center, and vice versa, *according to application needs and availability of resources* [1]. Several challenges have to be addressed:

CH1 Cloud computing solutions offer high computing and storage, but struggle to address low-energy and low-latency application scenarios and privacy concerns. In contrast, edge computing improves privacy and energy efficiency, by processing data closer to the source, but with limited computation and storage capabilities. Integrating these paradigms in a continuum of computing resources requires the definition of a Hardware (HW) and Software (SW) architecture that allows for horizontal (intra-layer) and vertical (inter-layer) orchestration on heterogeneous computing components.

CH2 Cloud, edge, and end-devices are typically handled as isolated silos, preventing applications from being seamlessly deployed and dynamically updated for continuous optimization, strategic to foster efficiency across continuum.

CH3 The more a system is heterogeneous, complex, and required to be adaptive, the more it is likely for designers to rely on partially integrated toolchains/methodologies tuned for specific aspects. Frameworks and tools exist, but effective interoperability is still to be reached.

MYRTUS is funded by the European Union, by grant No. 101135183. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

The *"Multi-layer 360° dYnamic orchestration and interopeRable design environmenT for compute-continUum Systems"* (MYRTUS) Horizon Europe project [2] aims at:

OBJ1 Defining a reference infrastructure where a diversity of fog and edge devices converge with the cloud to form a computing continuum capable of addressing the needs of complex and dynamic systems.

OBJ2 Featuring a runtime orchestration scheme, embodied within the *"Multi-layer 360° dynamIc RunTime Orchestration"* (MIRTO) Artificial Intelligence (AI)-powered cognitive engine, to guarantee high performance and energy efficiency, preserving security and trust.

OBJ3 Providing a reference Design and Programming Environment (DPE) for continuum computing systems, featuring interoperable support for cross-layer modelling, threat analysis, Design Space Exploration (DSE), application modelling, components synthesis, and code generation.

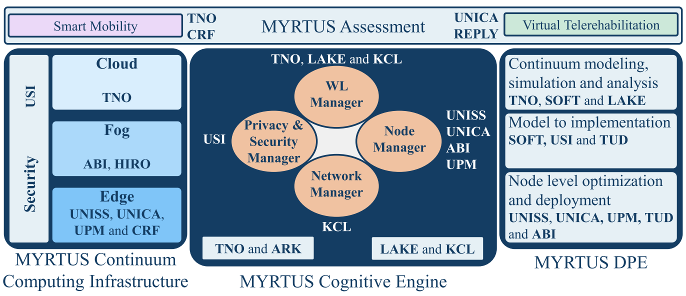

Fig. 1. Pillars and consortium.

The consortium counts fourteen participants from eight countries aiming to tackle the aforementioned objectives developing technologies grouped under three main technical pillars, as shown in Figure 1.

The **MYRTUS Continuum Computing Infrastructure** (*MYRTUS technical pillar 1*, see Section III) provides the key enabling technologies to realize horizontal and vertical composition for seamless execution of complex Workloads (WLs). *Universidad Politécnica de Madrid (UPM)*, *Università degli Studi di Cagliari (UNICA)*, *Università degli Studi di Sassari (UNISS)*, and *Canon Research Centre France (CRF)* are providing advances at the edge, and working together with *HIRO-MicroDataCenters B.V. (HIRO)*, *Abinsula S.r.l. (ABI)*, *TNO*, and *Università della Svizzera Italiana (USI)* in the integration of a secure, scalable, distributed and heterogeneous computing continuum compatible with the Gaia-X Trust Framework.

The **MIRTO Cognitive Engine** (*MYRTUS technical pillar 2*, see Section IV) is the core of the project and focuses on high-level orchestration for continuous optimizations, to maximize performance and energy efficiency across the continuum. *TNO*, in close cooperation with *ArubaKube S.R.L. (ARK)*, for the automation and deployment aspects, is leading the definition of the cognitive engine architecture that leverages swarm intelligence and Federated Learning (FL) technologies, brought respectively by *Lakeside Labs GMBH (LAKE)* and *King's College London (KCL)*, to provide different orchestration goals: WL management (*TNO*, *LAKE*, *KCL*), node management (*UNISS*, *UNICA*, *ABI*, *UPM*), network management (*KCL*), and privacy and security management (*USI*).

The **MYRTUS DPE** (*MYRTUS technical pillar 3*, see Section V) integrates strategies and tools for modelling, design and programming in an interoperable framework to ensure solution uptake. Model-based strategies are foreseen for 1) system-level analysis/characterization (*Softeam (SOFT)*, *LAKE*, *TNO*), 2) high-level application definition (*USI*, *SOFT*, *Technische Universität Dresden (TUD)*), and 3) DSE support and device specialization (*TUD*, *UPM*, *UNISS*, *UNICA*, *ABI*).

MYRTUS technologies will be assessed in two different application scenarios: **Smart Mobility** (developed jointly by *TNO* and *CRF*) and **Virtual Telerehabilitation**, (developed jointly by *UNICA* and *Forge Reply S.r.l. (REPLY)*) ensuring that the project delivers real-world benefits to users.

The rest of this paper is organized as follows. Section II describes the undertaken steps towards the compliance with EUCloudEdgeIoT.eu Initiative (EU-CEI). Sections III to V provide a snapshot of the status of MYRTUS technical pillars. Section VI concludes with the current ongoing activities.

II. Towards Standardization: Alignment with the EUCloudEdgeIoT.eu Initiative

The EU-CEI initiative defines a reference architecture for the continuum to be promoted as a standard. EU-CEI has identified eight categories, the Building Blocks (BBs) of a computing continuum infrastructure, representing the technical processes to operate applications along the continuum [1]. Its motivations and goals are compatible with the MYRTUS technologies and objectives presented in Section I and in these first months of the project, we made the effort to frame the MYRTUS technologies in the context of the EU-CEI reference architecture.

The goal of this activity is, on one hand to make a first step towards the standardization of the MYRTUS technologies. On the other hand, MYRTUS aims at feeding/contributing to EU-CEI by providing 1) concrete implementation examples on real test cases for all the BBs, and 2) an additional BB.

Table I shows a summary of the framing effort. The infrastructure and its management are strongly interleaved and there cannot be a complete distinction between EU-CEI BBs pertaining just to *MYRTUS technical pillar 1* or to *MYRTUS technical pillar 2*. For example, in terms of resource management and orchestration, specific support at the infrastructure level will be provided by Kubernetes[1], while at the Cognitive Engine level, in combination with the AI BB, decisions for orchestrating the tasks over resources will be made.

The EU-CEI reference architecture does not address the problem of turning applications into executable implementations. This is non-trivial for architectures that rely on heterogeneous families of CPUs and it becomes progressively more challenging as HW accelerators are introduced in the architecture. As the MYRTUS-compliant continuum infrastructures comprise heterogeneous and reconfigurable computing components the need for a DPE, *MYRTUS technical pillar 3*, emerged as a fundamental BB to enable the use of the continuum architecture.

III. MYRTUS Computing Continuum Infrastructure - Technical Pillar 1

The MYRTUS reference infrastructure is a composable layered cloud-fog-edge continuum, integrating heterogeneous, federated, and collaborative computing components, whose generic architecture is drafted in Figure 2. The *Edge Layer* consists of commercial multicores, Heterogeneous Multi-Processor Systems-on-Chip (HMPSoCs) Field Programmable Gate Array (FPGA)-based accelerators [3], and adaptive RISC-V processors with custom computing units [4]. The *Fog Layer* consists of Fog Micro Data Centers (FMDCs) and smart gateways [5] to provide analytics services on medium to long-term data and to extend the capabilities of edge devices. The *Cloud Layer* provides intensive computing, long-term storage, subsystem monitoring, coordination, data mining, and historical analysis.

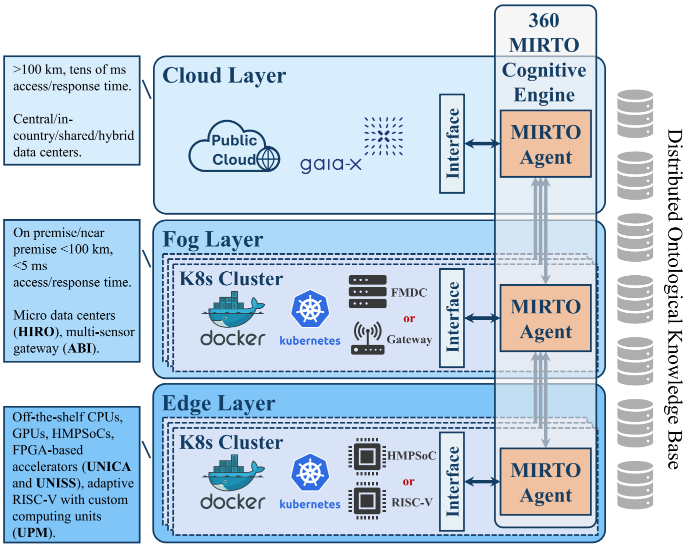

Fig. 2. MYRTUS layered computing continuum infrastructure.

To establish a continuum of resources and to dynamically adjust the computation load over them: 1) all the components at each layer communicate with their layer-/component-specific MIRTO agent which, in turn, communicates with the other layer-/component-specific agents, and 2) all layers support

[1]https://kubernetes.io/

TABLE I

EUCLOUDEDGEIOT.EU INITIATIVE FRAMING: EU-CEI BBS VERSUS THEIR MYRTUS ENVISIONED IMPLEMENTATION

EU-CEI BUILDING BLOCKS	MYRTUS ENVISIONED IMPLEMENTATION
Security and Privacy - Mechanisms for secure data and transactions between different components. **Trust and Reputation** - Models for allowing users of a continuum platform to generate trust in providers or increase their reputation (mainly in federated models).	Built-in infrastructure mechanisms, design and runtime strategies are envisioned, including: 1) authorization and authentication mechanisms of users/resources, 2) support for data integrity and availability, based on trustable, accessible, and coherent data exchange, 3) implementation of secure communication schemes, and 4) support for system integrity leveraging design time threat analysis and exploiting trust-related KPIs to implement trust and reputation schemes at runtime.
Data management - It includes collection, storage, computation, and actions performed over data.	Functionalities, storage, and processing capabilities are layer-/component-dependent. The envisioned resources heterogeneity allows capturing a wide variety of requirements challenging the DPE.
Resource management - It entails management of physical infrastructures and individual devices. **Orchestration** - Distributions of workloads, data or resources for executing a given action.	Kubernetes is used as a low-level orchestrator; while, the MIRTO Cognitive Engine covers the high-level orchestrator role, handling scalability without compromising QoS and heterogeneity. Aligned with EU-CEI vision, MIRTO optimization goals include latency reduction, throughput increase, and improved reliability without sacrificing security, privacy and trust. In addition, MIRTO aims also to reduce energy consumption.
Network - Connectivity considerations, including private networks and activities such as network slicing.	MYRTUS, by construction, aims to define a multi-layer infrastructure set-up. To foster seamless WL balance at runtime, MYRTUS computing components will embed identical interfaces and support the same protocols. Moreover, optimal network resource management to balance, where possible, load and latency is one of the drivers for runtime optimization.
Monitoring and Observability - It is intended at infrastructure level, including telemetry, and service/application level.	In line with EU-CEI monitors classification, we foresee: 1) Application monitoring (status of the application to identify underperformance issues not related to network/devices), 2) Telemetry monitoring (connectivity status and information loss), and 3) Infrastructure and Resource monitoring (status of the components). Observability will be achieved by the MIRTO Cognitive Engine leveraging a distributed KB to make smart decisions.
Artificial Intelligence (AI) - It is expected to be embedded in most of the activities performed.	MIRTO Cognitive Engine implements a plethora of different intelligence strategies to master WLs and resources orchestration at runtime.

Kubernetes as low-level orchestrator. To create a continuum of information, all layers will share one ontological KB (logical view), which can be distributed in different layers (implementation view). This way, data remains close to their consumers but can be shared between layers for analysis and decision-making.

The EU-CEI BBs will be implemented across the proposed infrastructure in a target-dependent manner.

- **Security and Privacy** and **Trust and Reputation** - Mechanisms for protecting data, securing communication, and components authentication, will be implemented. Three security levels are envisioned, as reported in Table II. Moreover, on the cloud side, adherence to the Gaia-X trust model will be guaranteed[2].
- **Data Management** - Storage and functional capabilities are layer-/component-dependent. At the edge, local storage in main memory and ad-hoc, flexible, coprocessing is enabled. The two fog layer components differ in storage and processing capabilities, but both serve as edge-cloud bridge. The smart gateway acts as a hub for data exchange among a diversity of actors at the edge (e.g., sensors, actuators, HW accelerators, etc.) and the cloud, and supports light local processing; whereas, the FMDC provides edge services SW stack to support for big data processing. Concerning the HW, the FMDC provides disaggregated, heterogeneous, hyper-converged servers, that are high-performing and energy efficient.
- **Resource Management** and **Orchestration** - All the infrastructure components will support low-level orchestration to enable WLs offloading/management. For all the edge and fog components a Linux-based Operating System and support for Kubernetes have been implemented, including offloading support to accelerators at the edge.
- **Monitoring and Observability** - FPGA-based edge devices are already instrumented to support basic runtime monitoring through performance monitoring counters and information, like latency and energy, are retrieved. Moreover, at minimum, execution-relevant metrics such as processing, communication latency, and energy consumption will be retrieved at the Fog Layer. Finally, the definition of trust indicators to be computed and made available locally at runtime is envisioned. Observability will be enabled by leveraging a shared KB[3], which will include the Resource Registry/Status (providing a snapshot of the components availability and their status) along with other historical information.
- **Network** - Edge components are expected to connect to the continuum with standard protocols (through Linux libraries). As an example, the HMPSoC accelerators are already capable of establishing secure connections via HTTP with the smart gateway exchanging JSON packets. The gateway itself is extremely flexible in terms of connectivity interfaces, it is customizable with ad-hoc user-defined interfaces, and natively supports several protocols (e.g. HTTP, MQTT, etc.). The FMDC is designed to seamlessly integrate with various edge components, leveraging standard protocols (e.g. HTTP, MQTT, CoAP etc.) to maintain secure and efficient communication channels.
- **Artificial Intelligence (AI)** - computing components in all layers are already capable of running AI models, which is mandatory to support the MIRTO agents executed on the computing continuum infrastructure to support the 360° orchestration.

[2]https://docs.gaia-x.eu/policy-rules-committee/trust-framework/22.10/

[3]The use of ETCD, https://etcd.io/, has been considered, being a strongly consistent, distributed key-value store for data that needs to be accessed by a distributed system or cluster of machines and already part of the Kubernetes environment.

TABLE II
MYRTUS ENVISIONED SECURITY LEVELS.

	High - *PQC resistant*	**Medium** - *Non-PQC resistant but suitable for current threats*	**Low** - *Lightweight non-PQC considering components capabilities*
Encryption	Symmetric encryption primitives as AES-256 [6].	Symmetric encryption primitives as AES-128 [6].	Symmetric encryption primitives as ASCON-128 [7].
Authentication	Digital signature schemes, following the NIST standard, e.g. CRYSTALS-Dilithium [8], FALCON [9].	Digital signature schemes, e.g. RSA [10], ECDSA [11].	Digital signature schemes as ECDSA [11].
Key exchange	Key encapsulation mechanisms, following the NIST standard as CRYSTALS-KYBER [12].	Key encapsulation mechanisms as RSA [10].	Key encapsulation mechanisms as ECDSA [11].
Hashing	At least 512 size hash as SHA-512 [13].	At least 256 size hash as SHA-256 [13].	Lightweight algorithms, e.g. ASCON-Hash, QUARK [14], spongent [15], photon [16].

IV. MIRTO COGNITIVE ENGINE - TECHNICAL PILLAR 2

MIRTO cognitive engine is responsible for high-level continuum orchestration both at deployment time (when a computation request is issued) and at execution time (while tasks are already running). This dynamic orchestration entails four steps executed in loops [17], [18]: 1) *sensing* of internal and external triggers for the orchestration; 2) *evaluation* of aggregated local and global information; 3) *decision* for resource allocation/configuration to improve KPIs; and 4) *reconfiguration/reallocation*. Four primary drivers for optimization are there: *optimal workload execution* (e.g. improving throughput and/or latency), *optimal network usage* (e.g. reducing network congestion, while guaranteeing adequate computing power), *optimal node configuration* (e.g. trading-off QoS to minimize energy consumption in specific components), and *privacy and security guarantees* (e.g. changing the adopted set of components according to the requirements of a newly incoming task).

The preliminary architecture of a MIRTO Cognitive Engine agent is depicted in Figure 3. At all layers, the MIRTO agents communicate with each other to negotiate the usage of resources and interoperability of services over multiple layers. At this stage, this is the initial architecture proposal:

- **MIRTO Agent -** Creates a *MIRTO Application Programming Interface (API) Daemon* defining the MIRTO agent as a (web-)service with a specification for its API. This REST-like API establishes how users will request orchestration activities to the MIRTO agent using a *Topology and Orchestration Specification for Cloud Applications (TOSCA)*[4] *Object Model*. It also provides a security module for user authentication (*Authentication Module*) and TOSCA description validation (*TOSCA Validation Processor*).
- **MIRTO Manager -** Unifies the four optimization drivers into the *MIRTO Manager* (whose internal architecture is currently under definition) that is responsible for deciding on the allocation of resources managed by the agent and/or on the configuration of the specific target chosen for execution.
- **Proxies -** Proposes interface points (*proxies*) to the KB and the deployment mechanism. This latter embodies the MYRTUS continuum life-cycle controlling strategy based on LIQO[5]. LIQO allows for clustering and resource virtualization. I constitutes the interface among MIRTO agents and Kubernetes-based orchestration achieving seamless virtualization of the underlying infrastructure.

[4]https://docs.oasis-open.org/tosca/TOSCA/v2.0/TOSCA-v2.0.html
[5]https://liqo.io/

To foster interoperability and portability, the interfaces between *MYRTUS technical pillar 1* and *MYRTUS technical pillar 2* are defined in an implementation-agnostic and target-independent manner. Nevertheless, for demonstration purposes, the implementation of the interfaces will be done according to selected technologies, which are currently under evaluation.

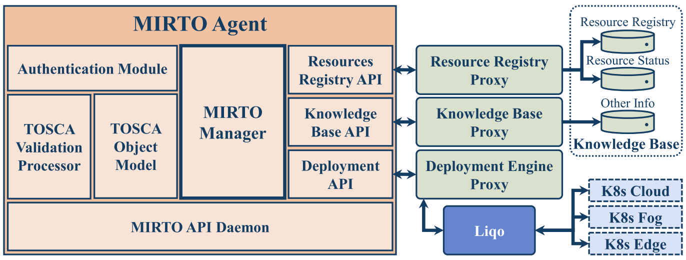

Fig. 3. MIRTO Cognitive Engine Agent.

Here follow the list of EU-CEI BBs supported by the MIRTO Cognitive Engine.

- **Security and Privacy**, **Trust and Reputation** and **Data Management** - The MIRTO Manager has four specific drivers, which are captured through execution requirements, including security, trust and reputation, or data-related ones. The TOSCA language offers mechanisms to describe application requirements for these aspects. For example, a deployment request may indicate that some of the SW containers should only run within a certain security level among those in Table II. Also, requirements can be placed over the storage of data, e.g. that they should happen in an encrypted manner. Such requirements are part of the constraints to be solved by the *MIRTO Manager* when making decisions for (re-)allocating, optimizing, and (re-)configuring execution over the continuum infrastructure.
- **Resource Management**, **Orchestration**, and **AI** - The *MIRTO Manager* is the cognitive block within MIRTO. Several concurrent and complementary approaches are meant to be put in place. At edge, on FPGA-based accelerators, MIRTO agents will use Machine Learning (ML)-based models to estimate the best operating point of a workload and, given the current status, change configuration accordingly (if needed). The possibility

of combining learned models from different agents using FL techniques, allowing MIRTO edge agents to evolve based on each other's experiences, is currently under consideration. At the Fog Layer, within FMDC units, the MIRTO agent will monitor system data, and learn from previous events and interactions making informed decisions to maintain optimal system conditions in terms of performance, resilience and efficiency. In general, both at the cloud and Fog Layer, variants of MIRTO agents will be developed using strategies based on swarm-like intelligence, FL, and distributed optimization. The goal is to have different flavors of MIRTO agents, capable of operating under different AI-based algorithms, suitable to address various contexts of applications and orchestration challenges.

V. MYRTUS Design and Programming Environment - Technical Pillar 3

The MYRTUS DPE is responsible for creating the deployment specification for the continuum, including all the executables and configuration files to program the heterogeneous components. Moreover, it exports meta-information with non-functional properties of the applications to aid the MIRTO Cognitive Engine in runtime decision-making. To foster interoperability among different end devices and tools, the DPE leverages open-source tools and formats to describe and exchange applications, i.e. TOSCA and Multi-Level Intermediate Representation (MLIR). As shown in Figure 4, the DPE is composed of three steps: 1) a step for high-level modeling, simulation, and analysis; 2) a step for turning the model into a concrete implementation; and 3) a step on node-level optimization and deployment of key computational kernels of the application.

- **Continuum modeling, simulation and analysis** - This step extensively leverages Modelio[6] to: i) model the functional partitioning of the overall scenario; ii) generate the Attack Defence Tree (ADT) for the analysis of the threats to which the system is exposed; iii) provide functional-level requirements, such as the expected end-to-end latency and fault conditions, leveraging its internal model-based KPIs estimation capabilities. A major Modelio extension to create the Cloud Service Archive (.csar) package, which will contain relevant TOSCA templates, scripts and files to allow workload deployment and management in Kubernetes-based environments is ongoing. This extension leverages Modelio CAMEL Designer[7] to specify multiple aspects/domains related to multi-/cross-cloud applications. FREVO[8] generates the local rules for the swarm agents to be used within the MIRTO Cognitive Engine and exploits DynAA[9] as a simulation engine to test the generated rules and evaluate the KPIs on simulated scenarios.
- **Model to Implementation** - Going from the modeling level to deployment implies defining the Program Code. Modelio can extract parts of the applications (*Portioned App*) that require acceleration (e.g., DSP kernels) and can be used directly to synthesize code. Predefined interoperability mechanisms guarantee that implementations can be derived also from external *Design Specific Languages (DSLs)* and/or *ML frameworks* or taken from existing hand-optimized *C/C++ components*. The Program Code is then passed to Step 3 for compilation, optimization, and, in the case of FPGA-based computing components, also for accelerator synthesis. This step also connects the DPE to the MIRTO Cognitive Engine. First, the component-level view of the application is fed to the MIRTO Cognitive Engine (defining the interface from design time to runtime, aka from Pillar 3 to Pillar 2) completing the deployment specification model with all the needed .tosca/.csar files. Second, Modelio is used to synthesize the swarm agents to be included in the MIRTO Manager of the Cognitive Engine from the local rules and *Threat Counter Measures Snippets* to mitigate the security threats associated with the ADT.
- **Node Level Optimisation and Deployment** - This step results in the executables and bitstreams for running and/or configuring the different computing components. A common interoperability framework based on MLIR, built atop the MLIR infrastructure of the EVEREST project [20], is adopted to allow i) importing third-party codes (from DSLs like [21], [22] or ML models as Pytorch[10] or ONNX[11] like [23]), ii) having access to third-party tools (like polyhedral compilers for optimization purposes), and iii) compiling code for different targets (as reconfigurable accelerators [24], CPUs, or customizable RISC-V cores). Mocasin[12] [25], a high-level Python-based DSE tool for heterogeneous many-cores, will be extended to support Coarse-Grain Reconfigurable Architecture (CGRA) architectures. Main MLIR dialects to be adopted have been defined. `dfg-mlir`[13] and `cgra-mlir` dialects will be used to model applications as dataflows and generate CGRA configurations, respectively. Numerical kernels can be described with an extension of the `teil` [26] dialect, with a NumPy-like front end, with support for custom data types using the `base2` dialect [27].

Existing dialects/tools will be leveraged from the MLIR ecosystem for accepting inputs in different languages (i.e., *torch-MLIR* and *Polygeist*) and target i) FPGA, producing Verilog (through High-Level Synthesis (HLS)) to feed MDC[14] for the generation of runtime reconfigurable accelerators, and ii) CPU/GPU, through the LLVM Intermediate Representation. For the HLS step, CIRCT-hls[15], already used by `dfg-mlir`, and Vitis-HLS, already supported by MDC, are currently under consideration. The deployment specification will be passed from Modelio to `dfg-mlir` in TOSCA format (i.e., YAML). The rest of the application is compiled with standard compilers, ensuring it can interoperate with the accelerated portions.

The DPE, while representing an important addition to the EU-CEI BBs, contributes to their implementation as follows.

[6] https://www.modelio.org/index.htm

[7] CAMEL Designer, https://github.com/Modelio-R-D/CamelDesigner/wiki, is an open-source Modelio extension based on the Cloud Application Modelling and Execution Language (CAMEL) language.

[8] FRamework for EVOlutionary design, https://frevo.sourceforge.net/

[9] To be released open-source by the end of the project [19]

[10] https://pytorch.org/

[11] https://onnx.ai/

[12] https://github.com/tud-ccc/mocasin

[13] https://github.com/Feliix42/dfg-mlir

[14] Multi-Dataflow Composer tool, https://mdc-suite.github.io/

[15] Circuit IR Compilers and Tools, https://github.com/llvm/circt/tree/main

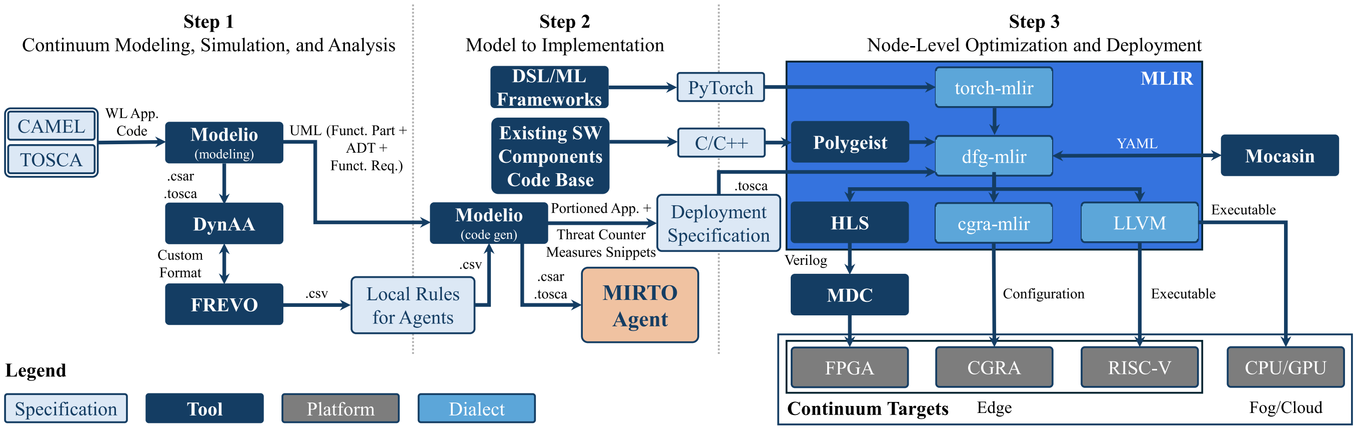

Fig. 4. MYRTUS Design and Programming Environment.

- **Security and Privacy** are considered from the very beginning as Modelio provides the ADT of the system and will synthesize the countermeasures snippets.
- **Orchestration** and **AI** are assisted by the initial deployment specification defined at design time, and by the initial local rules defined and synthesized for swarm agents.
- **Data management** on heterogeneous devices, particularly focusing on the computation over data, is enabled by the DPE node-level optimization and deployment step.

VI. Next Steps and Final Remarks

The main characteristics/capabilities of the technical pillars have been specified in the first months of the project. For each of them, a preliminary architectural specification and a first set of requirements, to drive the subsequent implementation and integration steps, have been drafted and released at M8.

The list below identifies all the ongoing consortium-level activities, in order of priority.

- **MIRTO Manager architecture (Pillar 2)**: Each MIRTO Manager handles data and information of various types according to the layer/component it operates into. As multiple drivers are there, different cooperating elements within the Manager will be there. We are currently modulating responsibility among those elements by identifying specific requirements they respond to [28]. For instance, a *Node Manager* will put in place directives coming from the *WL Manager* to run applications in container form on HMPSoC FPGA-based accelerators and, depending on the optimization goal and KPI-based characterization of HW functionalities, it will also select the configuration for HW acceleration that is most suitable. To establish deployment or reallocation directives, the *WL Manager* will gather information related to i) the state of resource utilization from the Resource Registry, ii) historical data and/or AI models from the KB, iii) application orchestration costs from a *Network Manager*, and iv) trust and security constraints from the *Privacy and Security Manager*. We expect to finalize MIRTO Manager architecture by M12.
- **Knowledge Base (interface between Pillar 1 to 2)**: MYRTUS infrastructure and MIRTO Cognitive Engine "share" information. Application, telemetry, and infrastructure and resource monitoring are intended to be made observable, where and when needed, by MIRTO Cognitive Engine agents. Even though we are considering the use of ETCD from the Kubernetes environment, the specific technology to be adopted has not been confirmed yet since the definition of the MIRTO Manager architecture will set the requirements for it. For sure the KB is expected to keep track of the current status of every single component (e.g. supportable security level and actual security configuration, type of computing node and their availability, etc.) in the Resource Registry, as well as of the historical batch data needed to implement, for example, Reinforcement Learning-based strategy within the Network Manager. We expect to finalize the adopted KB, how data is stored there, and how MIRTO Cognitive Engine is assessing them as soon as the MIRTO Manager architecture is finalized.
- **Container Image Registry and Repository (Pillar 1)**: The possible technologies to implement them are still under discussion. Candidate solutions should be easily accessible by all layers and expose security guarantees (e.g. access controls, image scanning, etc.). We expect to have a list of possible solutions for evaluation by M12.
- **Deployment Specification (interface between Pillar 3 to 2)**: The MYRTUS DPE creates the deployment specification for the MIRTO Cognitive Engine to orchestrate WLs and resources in MYRTUS-compliant computing continuum systems. Modelio creates the .csar package to allow WLs deployment in Kubernetes-based environments. This package will include also relevant metadata information to manage edge-nodes operating points. As meta-information, we envision a setup similar to the one presented in [29], [30] where different operating points for applications are described and leveraged at runtime to improve energy efficiency. The complete package specification and how to include all the design-time to run-time information is still under discussion and will be available after the M18 review.

In parallel to these activities, MYRTUS consortium is also starting partial integration of all the pillars' technologies to be assessed within the mobility and the rehabilitation use cases already at M18.

REFERENCES

[1] Task Force 3: Architecture, "Developing a Reference Architecture for the Continuum - Concept, Taxonomy and Building Blocks," Oct. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8403593

[2] F. Palumbo, M. K. Zedda, T. Fanni, A. Bagnato, L. Castello, J. Castrillón, R. D. Ponte, Y. Deng, B. Driessen, M. Fadda, T. H. du Fretay, J. de Oliveira Filho, V. Rao, F. Regazzoni, A. Rodríguez, M. Schranz, and G. Sedda, "MYRTUS: multi-layer 360° dynamic orchestration and interoperable design environment for compute-continuum systems," in *Proceedings of the 21st ACM International Conference on Computing Frontiers*, 2024.

[3] F. Ratto, S. Esposito, C. Sau, L. Raffo, and F. Palumbo, "Multithread Accelerators on FPGAs: A Dataflow-Based Approach," in *Proceeding of the 11th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM 2022)*, 2022.

[4] D. Vázquez, A. Rodríguez, A. Otero, and E. de la Torre, "Extending RISC-V processor datapaths with multi-grain reconfigurable overlays," in *Proceedings of the 37th Conference on Design of Circuits and Integrated Systems*, 2022.

[5] T. Fanni, G. Meloni, M. Melis, A. Solinas, and M. K. Zedda, "The multi-sensor gateway, a unified communication scheme and orchestration actor for heterogeneous systems," in *CPS Workshop 2022*, 2022.

[6] National Institute of Standards and Technology, "Advanced encryption standard," *NIST FIPS PUB 197*, 2001.

[7] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon," *NIST Lightweight Cryptography Project*, 2019.

[8] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS – dilithium: Digital signatures from module lattices," Cryptology ePrint Archive, Paper 2017/633, 2017. [Online]. Available: https://eprint.iacr.org/2017/633

[9] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon," *NIST Post Quantum Cryptography Project*, 2022.

[10] E. Milanov, "The rsa algorithm," *RSA laboratories*, pp. 1–11, 2009.

[11] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[12] P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, D. Stehle, and J. Ding, "Crystals-kyber," *NIST Post Quantum Cryptography Project*, 2022.

[13] "SECURE HASH STANDARD," 2001. [Online]. Available: https://csrc.nist.gov/files/pubs/fips/180-2/upd1/final/docs/fips180-2withchangenotice.pdf

[14] J. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark: A lightweight hash," *J. Cryptol.*, vol. 26, no. 2, pp. 313–339, 2013.

[15] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "spongent: A lightweight hash function," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop*, B. Preneel and T. Takagi, Eds., 2011.

[16] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON family of lightweight hash functions," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, P. Rogaway, Ed., 2011.

[17] R. et al., "Feedback control as mape-k loop in autonomic computing," in *Software Engineering for Self-Adaptive Systems III. Assurances*. Springer International Publishing, 2017.

[18] P. et al., "Hardware/Software Self-adaptation in CPS: The CERBERO Project approach," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer International Publishing, 2019.

[19] J. O. de Filho, T. Vogel, and J. de Gier, *Runtime Services and Tooling for Reconfiguration*. Singapore: Springer Singapore, 2016, pp. 69–92. [Online]. Available: https://doi.org/10.1007/978-981-10-0715-6_3

[20] C. Pilato, S. Banik, J. Beránek, F. Brocheton, J. Castrillon, R. Cevasco, R. Cmar, S. Curzel, F. Ferrandi, K. F. A. Friebel, A. Galizia, M. Grasso, P. Silva, J. Martinovic, G. Palermo, M. Paolino, A. Parodi, A. Parodi, F. Pintus, R. Polig, D. Poulet, F. Regazzoni, B. Ringlein, R. Rocco, K. Slaninova, T. Slooff, S. Soldavini, F. Suchert, M. Tibaldi, B. Weiss, and C. Hagleitner, "A system development kit for big data applications on FPGA-based clusters: The EVEREST approach," in *Design, Automation and Test in Europe Conf.*, 2024.

[21] S. Karol, T. Nett, J. Castrillon, and I. F. Sbalzarini, "A domain-specific language and editor for parallel particle methods," *ACM Transactions on Mathematical Software (TOMS)*, vol. 44, no. 3, pp. 34:1–34:32, Mar. 2018.

[22] A. Susungi, N. A. Rink, A. Cohen, J. Castrillon, and C. Tadonki, "Meta-programming for cross-domain tensor optimizations," in *Proceedings of 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE'18)*, ser. GPCE 2018. New York, NY, USA: ACM, Nov. 2018, pp. 79–92. [Online]. Available: http://doi.acm.org/10.1145/3278122.3278131

[23] F. Manca, F. Ratto, and F. Palumbo, "ONNX-to-Hardware Design Flow for Adaptive Neural-Network Inference on FPGAs," in *Proceedings of the XXIV International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*. Springer Nature Switzerland, 2024.

[24] S. Soldavini, K. F. A. Friebel, M. Tibaldi, G. Hempel, J. Castrillon, and C. Pilato, "Automatic creation of high-bandwidth memory architectures from domain-specific languages: The case of computational fluid dynamics," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 16, no. 2, Mar. 2023. [Online]. Available: https://doi.org/10.1145/3563553

[25] C. Menard, A. Goens, G. Hempel, R. Khasanov, J. Robledo, F. Teweleitt, and J. Castrillon, "Mocasin – rapid prototyping of rapid prototyping tools: A framework for exploring new approaches in mapping software to heterogeneous multi-cores," in *Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools, co-located with 16th International Conference on High-Performance and Embedded Architectures and Compilers (HiPEAC)*, ser. DroneSE and RAPIDO '21. New York, NY, USA: Association for Computing Machinery, Jan. 2021, pp. 66–73. [Online]. Available: https://doi.org/10.1145/3444950.3447285

[26] N. A. Rink and J. Castrillon, "TeIL: a type-safe imperative Tensor Intermediate Language," in *Proceedings of the 6th ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming (ARRAY)*, ser. ARRAY 2019. New York, NY, USA: ACM, Jun. 2019, pp. 57–68. [Online]. Available: http://doi.acm.org/10.1145/3315454.3329959

[27] K. F. A. Friebel, J. Bi, and J. Castrillon, "BASE2: An IR for binary numeral types," in *13th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART 2023)*, ser. HEART2023. New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 19–26. [Online]. Available: https://doi.org/10.1145/3597031.3597048

[28] P. Ruiu, C. Rubattu, M. Anedda, A. Lagorio, V. Popescu, N. Amarie, D. D. Giusto, and M. Fadda, "Edge-to-cloud continuum orchestrator for distributed video applications," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2024.

[29] R. Khasanov and J. Castrillon, "Energy-efficient runtime resource management for adaptable multi-application mapping," in *Proceedings of the 2020 Design, Automation and Test in Europe Conference (DATE)*, ser. DATE '20. IEEE, Mar. 2020, pp. 909–914. [Online]. Available: https://ieeexplore.ieee.org/document/9116381

[30] T. Smejkal, R. Khasanov, J. Castrillon, and H. Härtig, "E-Mapper: Energy-efficient resource allocation for traditional operating systems on heterogeneous processors," 2024. [Online]. Available: https://arxiv.org/abs/2406.18980