Designing Swarms of Cyber-Physical Systems: the H2020 CPSwarm Project

Invited Paper

Alessandra Bagnato
Softeam R&D Department
Avenue Victor Hugo 21
Paris, France 75016
alessandra.bagnato@softeam.fr

Regina Krisztina Bíró
Search-Lab
Szeruskert Utca 19
Budapest, Hungary 1033
regina.biro@search-lab.hu

Dario Bonino
Istituto Superiore Mario Boella
Via Pier Carlo Boggio, 61
Torino, Italy 10138
dario.bonino@ismb.it

Claudio Pastrone
Istituto Superiore Mario Boella
Via Pier Carlo Boggio, 61
Torino, Italy 10138
claudio.pastrone@ismb.it

Wilfried Elmenreich
Universitaet Klagenfurt
Universitaetsstrasse 65-67
Klagenfurt, Austria 9020
wilfried.elmenreich@aau.at

René Reiners
Fraunhofer Institute for Applied Information Technologies
Sankt Augustin, Germany
rene.reiners@fit.fraunhofer.de

Melanie Schranz
Lakeside Labs
Klagenfurt, Austria 9020
schranz@lakeside-labs.com

Edin Arnautovic
TTTech Computertechnik AG
Schoenbrunner Strasse 7
Vienna, Austria 1040
edin.arnautovic@tttech.com

ABSTRACT

Cyber-Physical Systems (CPS) find applications in a number of large-scale, safety-critical domains e.g. transportation, smart cities, etc. As a matter of fact, the increasing interactions amongst different CPS are starting to generate unpredicted behaviors and emerging properties, often leading to unforeseen and/or undesired results. Rather than being an unwanted byproduct, these interactions could, however, become an advantage if they were explicitly managed, and accounted, since the early design stages. The CPSwarm project, presented in this paper, aims at tackling these kinds of challenges by easing development and integration of complex herds of heterogeneous CPS. Thanks to CPSwarm, systems designed through a combination of existing and emerging tools, will collaborate on the basis of local policies and exhibit a collective behavior capable of solving complex, real-world, problems. Three real-world use cases will demonstrate the validity of foundational assumptions of the presented approach as well as the viability of the developed tools and methodologies.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; **Robotics**; • **Software and its engineering → Unified Modeling Language (UML)**; **System modeling languages**; *Development frameworks and environments*; *Integrated and visual development environments*;

KEYWORDS

cyber-physical systems, cyber-physical system of systems, model based design, swarm computing, emergent behaviour, evolutionary algorithms, development tools, deployment tools, CPS integration

ACM Reference format:
Alessandra Bagnato, Regina Krisztina Bíró, Dario Bonino, Claudio Pastrone, Wilfried Elmenreich, René Reiners, Melanie Schranz, and Edin Arnautovic. 2017. Designing Swarms of Cyber-Physical Systems: the H2020 CPSwarm Project. In *Proceedings of ACM International Conference on Computing Frontiers, Siena, Tuscany, Italy, May 2017 (ComputingFrontiers'17),* 8 pages.
DOI: XX.XXX/XXXX

1 INTRODUCTION

According to the NIST definition[1], "Cyber-physical systems (CPS) are engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components". One of the distinctive characteristics of CPS is, therefore, that they are more focused on control activities rather than on pure software services. Although CPS are often simplistically defined as embedded systems able to run software locally, their principal mission is different from pure computation, and involves interaction with the physical world, like e.g., in cars, medical devices, scientific instruments, etc. The dynamics of such systems can evolve very fast and are dependent on variations from the surrounding environment or from other interacting systems.

[1] In the "SYNOPSIS" provided at https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286, *last visited on March 07, 2017.*

ComputingFrontiers'17, Siena, Tuscany, Italy

DOI: XX.XXX/XXXX

CPS and Cyber-Physical Systems of Systems (CPSoS) are increasingly playing the role of foundational building blocks for bringing adaptive intelligence to processes and environments [30], in several application domains ranging from Smart Mobility, to Smart Health [34], Smart Cities and Smart Production [21]. Due to the increasing pervasiveness of CPS, issues related to effective design of solutions able to reach predefined goals flexibly, reliably and adapting to changing surrounding conditions, become challenging and worth of further investigation. While increasing the CPS adoption results in increasingly mature solutions for their development, a single, consistent, science of system integration for CPS has not yet been consolidated. According to the ECSEL Strategic Innovation Agenda and to the NSF Steering committee for Innovations in Cyber-Physical Systems, the design of CPS is currently hampered by the limited ability to design at a system-level. There are many factors currently lacking in system-level design, such as:

(a) the near absence of formalized high fidelity models for large systems,
(b) insufficient ways of measuring performance and
(c) inadequate scientific foundations (e.g., see [17] for an introduction on limitations on CPS modeling).

The "Cyber-Physical European Roadmap & Strategy" (CyPhERS) EU project, which compiled the "Research Agenda and Recommendations for Action" [30] in the CPS domain, identified similar challenges on the technical and scientific standpoints. In particular, the absence of an integrated design approach, at the system level, which would enable a better design of these systems is seen as one of the biggest limitations to overcome in current CPS solutions.

From a foundational standpoint, individual theories allowing formal description of the different aspects of Cyber Physical System design including physical, technical, organizational and human-system interaction are available, at different degrees of maturity. However, such disciplines are not fully integrated in a common systems theory. In other words, while methodologies, representations and tools exist for addressing single aspects of CPS design, supporting to the whole design lifecycle is still an open challenge.

This challenge becomes particularly intriguing when involved systems are modular, autonomous CPS, which given a bare hardware can perform several different tasks by locally collaborating to reach an "emerging" behavior, which is much more than the simple sum of single CPS functionalities.

In this paper, we introduce the CPSwarm approach for designing complex herds of heterogeneous CPS systems that interact and collaborate based on local policies and that collectively exhibit a behavior capable of solving complex, real-world, problems. The project, funded under the H2020 ICT-01 program of the European Commission aims at defining a complete tool-chain, which starts from models of CPS basic components, functions and prototype behaviors, and enables the designer to: (a) set-up collaborative autonomous CPSs; (b) test the swarm performance with respect to the design goal (i.e., to evaluate the solution fitness against the design requirements); (c) deploy solutions towards "reconfigurable" CPS devices and Cyber Physical Systems of Systems (CPSoS). The remainder of the paper is organized as follows: Section 2 provides the context in which the project is deployed, describing state-of-the-art approaches and similar initiatives both at the European level and worldwide. Section 3 introduces the CPSwarm project, with its aims and goals while, Section 4 better details the project approach to design large-scale CPS. Section 5 describes the use cases in which the solutions developed within the project will be tested and the expected innovations in the respective domains. Eventually, Section 6 draws conclusions and highlights next steps and expected outcomes of the project.

2 RELATED WORK

Relations to the current state-of-the-art at the academic research level, and possible innovations, are manifold. They, in particular, encompass modeling and design methods, and tools, deployment tools and code generation support. The following sections better detail the relevant related works in each of the cited areas and define the CPSwarm contribution in such domains.

2.1 Modeling methods and tools

Current research on modeling CPS shows a very active domain, in which modeling is seen as one of the foundational approaches for dealing with natural heterogeneity in CPS. By modeling a CPS it is possible to effectively capture domain specific concepts and requirements. Many modeling techniques have been proposed in the CPS design community, including: meta-modeling and meta-programming techniques, formal semantics such as denotational, axiomatic or operational semantics, actor-oriented design approaches, etc.

Model-Based Design (MBD), as an example, has been identified as a powerful design technique for CPS [14], [15]. In MBD, models pass through all stages of: (a) design, (b) analysis, (c) verificaton, and (d) validation. Specifications of systems and their underlying components are defined in form of models reflecting the evolution of the system. These models can be used for early design analysis. They can help in separation of concerns, traceability, trace generation, impact analysis, formal verification, simulation and synthesis.

A large number of modeling languages have been adopted for CPS modeling, addressing the underlying aspects, i.e., modeling of physical processes and requirements management. A good survey was presented by the Columbus project [4], with the aim of defining an interchange format for CPS design, covering languages and tools like Stateflow/Simulink[2], Modelica[3], Checkmate [32] and Masaccio [12]. These languages/tools enable CPS modeling for all design phases, including simulation and verification.

More recently, high level languages such as UML [24], SysML [25] and MARTE [23] have been exploited to model these complex distributed systems. However, none of these languages can singularly address all the challenges related to CPS modeling. More specifically, the UML language, traditionally used to represent software systems, defines the syntax of model diagrams; but it does not offer any specific semantics for CPS modeling. The SysML standard limits its applicability to aspects of requirements management [18]. Eventually, MARTE enables designers to define non-functional constraints, only. In general, the complexity of CPS design demands

[2]Simulink - Simulation and Model-based Design. The MathWorks, https://de.mathworks.com/products/simulink.html,*last visited on march 09, 2017*
[3]Modelica and the Modelica association, https://www.modelica.org/, *last visited on March 09, 2017*

for extended system models, analytical tools and simulation tools along with suitable modeling languages [2, 20].

Among current solutions to model-based CPS design, few approaches exploit formal methods to assess the correctness of CPS design and provide guarantees for specified CPS properties. This is particularly true in case of groups of CPS interacting together to reach a predefined, high-level goal. As an example, Sun et al. [37] use model checking to verify the correctness of composition in a power grid CPS, while assuming that the individual components work correctly. Using a decomposition approach, their system is logically divided into smaller modules, which can be efficiently checked. Even at the single CPS design-level, formal methods might be exploited to ensure consistency between designed architecture and derived solutions. In this context, Bhave et al. [3] propose a method for defining and evaluating consistency between architectural views derived from different heterogeneous models and the base architecture. They, in particular, formulate the problem of consistency checking as a typed graph matching problem between the connectivity graphs of different architectural views and the base architecture of the system.

2.2 Design methods and tools

Traditional system design methods typically follow a top-down approach. Designers usually start with a high-level specification. Then, they iteratively refine and enrich such models to finally reach a representation describing all components and interactions within the system. This top-down design process is the basis for hierarchical system control structures. However, the design of distributed non-hierarchical CPS requires a different approach. One way to design these kind of systems is to imitate the structure, and behavioral patterns adopted in nature to face similar problems. The work of Reynolds [28], for example, represents a prominent approach to the problem, by creating a nature-inspired algorithm based on the swarming behavior of birds.

Self-organizing systems, call for a different kind of design methodology able to capture bottom-up processes, the emergent behaviors and to somehow combine the micro-macro view of self-organizing systems. Creating a library of emergent behaviors can be useful, but limits the system design to a number of predefined templates. To the authors' knowledge, currently there are few, or none, straightforward methodologies to find the appropriate micro level rules that result in the desired macro level behavior, although some steps were already taken in this direction. Many proposed techniques stem from the field of multi-agent systems and agent-oriented software engineering[8, 10, 26, 35]. However, as explained above, a formal design methodology must contain processes that revisit and iteratively refine the micro level behaviors in order to create a bridge between local and global level behaviors [6]. The great majority of works in this field describe swarming mechanisms, but do not give answers how to apply them to achieve an intended technical effect. Yet, one can collect existing, published, experiences as a pool of design patterns in order to adopt them to engineer the desired system. Such pattern collections have already been formulated, for example by De Wolf and Holvoet [5], or by Sudeikat and Renz [36].

When designing large, parameter-heavy complex systems, manual trial-and-error solutions[9] are often not efficient or sometimes not feasible. In this case, automated processes that systematically test micro-level rules driven by some evaluation function based on the desired global behavior become crucial. Since the search space expands heavily with the number of possible local states and interaction rules, an exhaustive search is simply not possible. One viable approach for automated simulation-based search and design of self-organizing systems would, for example, be using evolutionary methods. The main advantage of such algorithms is that they can be applied to any problem where the quality of a candidate solution can be directly measured, thus there is no need for any internal knowledge about the simulated system. Moreover, such algorithms gracefully support parallel execution and can cope with the uncertainty of stochastic simulation models.

There are several examples of evolving the local rules of a self-organizing system. Quinn et al. demonstrate the usefulness of evolutionary algorithms to evolve cooperative behavior [27] by evolving teamwork strategies among a set of robots. In the field of swarm robotics, Nelson describes the evolution of cooperative systems with ANN controllers for a team that plays "Capture the flag" against an opponent team of robots [22]. Fehervari and Elmenreich evolve neural network controllers for a team of self-organizing soccer robots [7]. Trianni presents several experiments where the controllers of a group of robots were evolved to achieve a certain team behavior [38]. Examples are not limited to robotic applications; Arteconi [1] applies evolutionary methods for designing self-organizing cooperation in peer-to-peer networks.

2.3 Deployment tools and code generation

CPS modeling is not limited to representing and simulating CPS and CPS behavior under operational conditions. It also involves aspects related to modularity and scalability of actual CPS programming and deployment. Indeed, CPS are located at the boundary between the physical world, with its processes, qualities and measures, and the immaterial world, which permits to monitor, control and direct physical phenomena involving the CPS solution. Lee [16] discusses two complementary approaches for designing CPSs under this stand point. The first approach, which is called "cyberizing the physical" refers to wrapping software abstractions around physical subsystems. The second approach is called "physicalizing the cyber" refers to endowing software and networking components with abstractions, which are suitable for physical subsystems. As the complexity of CPS increases, the challenges of modeling the interaction between cyber and physical systems increase as well. Hence, using these approaches helps in bridging their differences for achieving more realistic modeling.

A number of tools and frameworks have been developed to ease correct implementation and deployment of CPS. One of the most diffused approaches is code generation, or synthesis. The tool converts some high-level specification of a CPS into an (intermediate) implementation easier to run on the target platforms. Among the relevant approaches Roy et al. [29] introduce a CPS design tool named PESSOA for synthesizing controllers for CPS. It accepts a CPS represented in form of a set of differential equations and automata. The output is a controller for the system that enforces the given specification. On the other hand, Martin and Egerstedt[19]

discuss system tools for translating high-level CPS models into executable code on physical devices.

Another very active area of research regards the automated deployment of CPS programs. Hnat et al. [13] present a macro-programming framework, which is called MacroLab, for programming CPS. Using MacroLab, a user can write a single program for the entire CPS. The program is then decomposed into a set of micro-programs, which are loaded into each node.

3 THE CPSWARM PROJECT

The CPSwarm project positions itself in the domain of CPS system design and engineering, and aims at providing tools and methodologies that pave the way towards well-established, model-based and predictive engineering design methodologies and tool chains for next generation CPS systems. It has been funded under the ICT01 H2020 program of the European Commission on "Smart Cyber-Physical Systems" and it builds upon state of the art in CPS and IoT. Among the several issues related to CPS design, CPSwarm aims at: (1) bridging the gaps between currently available approaches and methodologies, (2) providing a relevant subset of the glue toolchains and layers which are currently missing in CPS design [11]. CPSwarm tackles the challenges identified in Section 2 by establishing a science of system integration in the domain of swarms of CPS, i.e., of complex herds of heterogeneous CPS systems that interact and collaborate based on local policies and that collectively exhibit a behavior capable of solving complex, industrial-driven, real-world problems. The CPSwarm consortium includes partners from 6 different EU countries and involves large, small and medium enterprises and several research institutions. Due to the participation of enterprises, the main outcomes of the project are expected to be readily usable in the CPS design process. Engineered versions of tools and solutions developed within the project will likely be exploited to improve the current market positioning of the involved partners. Nevertheless, attention will particularly be devoted to the open-source community and an transparent give-back approach to result exploitation is already planned, including open source release of core components and solutions.

3.1 Goals

The project research objectives are manifold and try to achieve a good coverage of issues currently hampering the CPS design domain. With particular reference to the previously discussed issues and state-of-the-art, CPSwarm aims at providing the following contributions.

Improved support of CPS design. CPSwarm aims at offering a fully-fledged design and simulation environment, namely the *CPSwarm Workbench*, natively supporting iterative, computer-aided design of complex, autonomous CPSs focusing on swarms of heterogeneous systems.

Extendable library of re-usable models for CPS. The CPSwarm Workbench will feature a shared library, namely the *CPSwarm Library*, specifying models of CPS subsystems, functions, algorithms and communication systems. It encompasses hardware-independent models (thus providing native support to interoperability) for single CPS subsystems, behavioral / functional models of both internal CPS subsystems and external services, and a library of existing HW representations enabling deployment of designed CPS on a selected set of heterogeneous systems. Mechanisms and guidelines for extending and integrating the CPSwarm library in CPS systems will be part of the project outcomes.

Reduced complexity of design work-flow. By exploiting the CPSwarm Workbench and Library, CPSwarm will enable design and integration methodologies devised to reduce development and integration effort and maximize re-use. This includes techniques for easier, block-based design of CPS and methodologies supporting iterative refinement of design choices through simulation and comparison against the target performances. Code generation tools will enable bulk and over-the-air programming and update of CPS systems thus reducing dramatically the time and effort necessary for CPS deployment, configuration and monitoring. The considered Models library also includes models describing the interaction between humans and CPS.

Extendable library of swarming algorithms. In order to support development of concrete CPS solutions, the CPSwarm Workbench and Library will not just focus on descriptive modeling aspects, but will include a set of reference, reusable algorithm solutions. Two main aspects will be considered: (a) evolutionary design of local CPS behavior controllers (algorithms, rules) which optimize the CPS behavior with respect to predefined goals. This also includes meta-heuristic design and applications, where needed; (b) algorithms and tools for building swarms of CPS systems that collaborate to reach higher-complexity goals.

Toolchains for CPS integration. Support for interoperability and predictive-engineering inherently built within CPS models is a necessary aspect to enable estimation and prediction of the overall CPS Swarm behavior and performance, as well as reliable integration with third-party CPSoS. At the same time, hardware abstraction issues must be tackled to enable cross-platform CPS integration and design patterns. At this purpose the CPSwarm Model Library is complemented by a CPSwarm Abstraction Layer, "isolating" generated artifacts from real CPS hardware.

3.2 Methodology

CPSwarm faces the challenge of effectively capturing the industrial needs and the innovation demands about tool-chains to set-up collaborative and autonomous CPS. In order to cope with such a challenge, CPSwarm follows the hybrid research and innovation methodology depicted in Figure 1.

The proposed methodology is built upon an iterative approach tailored to better bridge the gap between requirement elicitation and development activities. According to the methodology, the process starts by collecting insight from several sources including project-specific requirements from EU roadmaps and initiatives and research achievements in the field of CPS and Swarm core technologies, e.g. linked research activities, relevant standards, etc.

Inputs are collected through ad-hoc stakeholder workshops and literacy research (market studies, EU and international research and innovation road-maps, state-of-the-art surveys, etc.) and translated into an high-level analysis of the system and applications requirements. The resulting knowledge, recorded in form of descriptive reports is further analyzed through requirements engineering techniques.

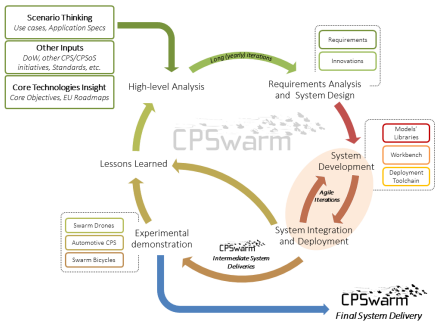

Figure 1: The CPSwarm project methodology.

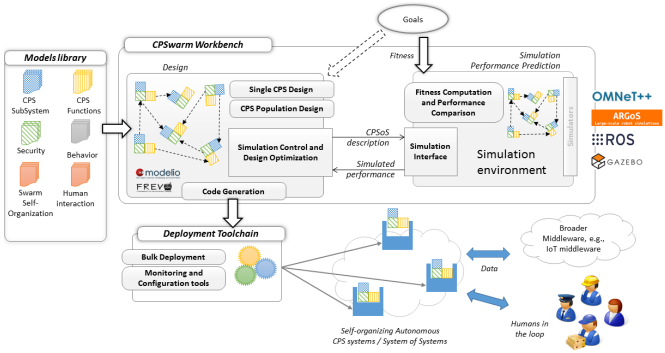

Figure 2: The CPSwarm high-level architecture and concept.

The *Requirements Analysis and System Design* step follows, producing as outcome a first system design specification together with the corresponding set of requirements and "innovations". As soon as they are available, system requirements are translated and linked into functions and fed into the *CPSwarm Requirements and Innovation Tracker*.

The tracker is used to provide transparent access to requirements to all analysts and developers involved in the projects both on the status of the requirements (open, rejected or solved) and functions (defined, developed or validated). When a sufficient number of features and requirements are known, the *System Development* step begins, using a distributed SCRUM [31] development methodology and short agile iterations as shown in Figure 1, where features of the CPSwarm system are identified and selected to be implemented at each cycle. Each short iteration produces internal releases of SW/HW components, which are continuously integrated and deployed in the *System Integration and Deployment* step.

Once the overall macro-components of the platform are available and integrated together, an intermediate release of the CPSwarm system is produced and the *Experimental Demonstration* step starts. This is the final part in each long (yearly) iteration, named "phase", where the CPSwarm system is deployed in real-life settings to demonstrate the identified use cases.

4 DESIGNING SWARMS OF CPS

CPSwarm is deployed around the conceptual architecture shown in Figure 2. A Model library collects formal representations of CPS subsystems, CPS base functions, security recipes, behavior routines, swarm and self-organization algorithms, human-to-CPS interaction patterns, and forms the cornerstone upon which the project is founded. The library, which for example could be based on, and extend, the Modelio store[4], will be one of the first, publicly available, model libraries for designing swarms of CPS. Building upon models in the Library, the CPSwarm Workbench will enable CPS engineers to collaborate and assemble model-based descriptions of CPS systems, with humans in the loop.

4.1 Workflow

More specifically, the CPS design work-flow will be deployed in three main phases, as follows.

[4]http://store.modelio.org, *last visited on February 21, 2017.*

4.1.1 Phase1: Model-Based Design. After identifying the core CPS components needed for a solution, CPS engineers will exploit a library of customizable behavioral routines and self-organization algorithms (e.g., bird flock, ant colonies, etc.) to set-up swarms of CPS that self-organize to solve the target problem. The resulting ensemble is functionally acting as a unique Cyber Physical System of Systems.

Depending on the selected organization mechanisms, different inter-CPS communication means will be available, and connectivity to higher-level middle-ware solutions will be supported, thus enabling integration in higher-level systems of systems. For example, such an integration will provide means for setting-up the swarm free parameters depending on the specific application e.g., changing the area to be covered by a population of sensors, etc.

Throughout the CPS design, interaction patterns and guidelines available in the model library will support engineers in identifying and applying the most suitable interaction mechanisms between humans and CPS system, fueling higher level of trust between users and CPS. Safety and security concerns will also be addressed, both under the standpoint of formal verification of designed solutions (e.g., by verifying generated SysML [18] artifacts) and under the standpoint of threat modeling and countermeasures identification.

The outcome of the first, model-based, design phase will be a swarm (either homogeneous or heterogeneous) of completely specified CPS and their behavioral models.

4.1.2 Phase2: Predictive engineering. In the second phase, the CPSwarm Workbench will enable advanced simulation of the overall emerging system behavior and evaluation of achieved performances against goals and design constraints. This, in turn, will enable engineers to check the degree of adaptation and flexibility of devised solutions with respect to changing operating context (simulated). According to the predictive engineering principles, simulation will include complex real data, and or hardware/human in the loop solutions where needed. Aspects related to local (short-range, between components of the same swarm) and global communication, security, etc., are just examples of the complex scenarios analysis tackled through the CPSwarm Workbench. Co-simulation will be considered, where needed, by integrating a set of selected simulation engines, which will allow simulating the different CPS facets,

properly. The FREVO (Framework for Evolutionary Design) [33][5], OMNeT++[6], ARGOS[7], ROS[8] and Gazebo[9] are samples of engines that will be taken in consideration.

A dedicated Simulators Integration Layer (SIL) will be designed and implemented to decouple the *CPSwarm Simulation and Performance Prediction Workbench* from engine-specific representations, control commands and data models. Simulation results will be then analyzed by a *Fitness Computation and Performance Comparison* component to better evaluate the fitness of the solution with respect to the foreseen goals.

As the design of swarm intelligence might be a challenging and somewhat crafty process, several iterative cycles might occur before reaching the best solution satisfying the overall goals, the initial constraints, or effectively trading off conflicting requirements. The CPSwarm Workbench will be designed to support this iterative refinement with semi-automated tools exploiting simulation results. The Simulation Control and Design Optimization component will be in charge of controlling the runs of the simulation environment, through the Simulation Interface, and will support automatic reconfiguration of the overall CPSoS description by changing algorithms, behaviors or configuration parameters. Multi-criteria optimization techniques will also be considered for evaluating different design alternatives and iteratively progress towards the final optimized solution.

4.1.3 Phase3: Bulk deployment and monitoring. The CPSwarm Workbench does not limit itself to design time usage. On the contrary, it aims at positively affecting the overall CPS engineering process by reducing the development time and total cost of ownership of designed CPS solutions. To reach this goal, CPSwarm promotes a modular structure for CPS exploiting an abstraction of lower level functionality to enable automatic, model-based, code generation and deployment of the designed systems. The adoption of automatic code generation approaches also helps avoiding the introduction of errors in manual implementation of the software components, thus removing inefficiencies and reducing debugging time.

CPSwarm defines a concept of modular CPS that is in line with the approaches exploited in Eclipse Papyrus[10], Modelio[11] and other similar IDEs. In this concept (depicted in Figure 3), a plain embedded system is preloaded with a so-called CPSwarm Abstraction Layer[12], which provides the primitives (and run-time) needed to activate/deactivate/control device-independent cyber-physical routines. Such routines have direct mappings to the characteristics modeled at design phase. This direct correspondence permits to generate code executable on the target CPS hardware and that, at the same time, can be easily traced back to simulation and performance evaluation outcomes. In other words, for each component explicitly modeled during the design phase, a corresponding software artifact will be generated and installed on target platforms,

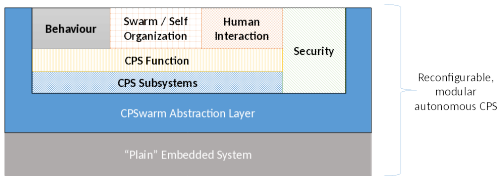

Figure 3: The CPSwarm modular CPS system concept.

exploiting the CPSwarm Abstraction Layer, and proper instrumentation will be provided to measure the system performance during real operations.

All the components hosted by the CPSwarm Abstraction Layer are abstracted from the actual embedded system used to run the CPS and might possibly be hot-swapped / loaded on CPSwarm compliant devices. When it comes to deployment, the components need to be translated to platform dependent libraries - ideally completely automated. A small subset of target platforms[13] will be addressed in project demo scenarios (see Section 5), and its complexity might vary depending on the available computational power, memory and hardware resources.

The joint operation of CPS-independent modules and CPS-dependent abstraction provides an unprecedented flexibility in CPS deployment, as the same system can be easily re-purposed. Moreover, it dramatically reduces deployment and field-experimentation time and costs, as the CPS swarm programming of different systems can happen almost in parallel and unsupervised. CPSwarm pushes these capabilities to the limit by explicitly defining a Deployment Tool-chain subsystem that handles automatic, over-the-air, bulk update of heterogeneous CPS populations. Such update can involve either single components, or the full CPS logic, even including some core parts of the CPS Abstraction Layer (in this case we speak of upgrades). Updates and upgrades can happen both offline, before starting operations, or during mission execution, depending on some external factors (e.g., security policies, norms, safety procedures, etc.). This allows better reacting to changes in the swarm surrounding environment and in contour conditions. Moreover, the CPSwarm deployment tool-chain will incorporate components for continuously monitoring deployed CPS, allowing for on-line evaluation of the swarm performance in real scenarios and supporting early detection of failures.

5 USE CASES

The actual usefulness of tools and methodologies developed in CPSwarm shall be proven in real-world scenarios, with variability and hidden factors that are almost impossible to capture in purely simulated environments. To this purpose, CPSwarm will demonstrate the viability of the proposed approach on 3 complimentary, yet different, use cases targeted at: (a) swarms of (mixed) robotic vehicles (e.g. Unmanned Aerial Vehicles (UAV) and rovers), (b) automotive CPS systems for freight vehicles and (c) swarm logistics. All scenarios are characterized by the presence of heterogeneous CPS interacting together and showing emerging behaviors difficult to predict with traditional approaches.

[5] http://frevo.sourceforge.net, *last visited on February 21, 2017.*
[6] https://omnetpp.org, *last visited on February 21, 2017.*
[7] http://www.argos-sim.info, *last visited on February 21, 2017.*
[8] http://www.ros.org, *last visited on February 21, 2017.*
[9] http://gazebosim.org, *last visited on February 21, 2017.*
[10] https://eclipse.org/papyrus/,*last visited on February 21, 2017.*
[11] https://www.modelio.org,*last visited on February 21, 2017*
[12] Functionally similar to the SIL.
[13] prominently based on ROS, but not limited to.

5.1 Surveillance, Search and Rescue

The first CPSwarm use case considers heterogeneous swarms of ground robots/rovers and UAVs carrying surveillance taks for critical infrastructures like, e.g., industrial or power plants as well as Search and Rescue (SAR) tasks. For surveillance, the consortium partners envision applications of swarms for: (a) intrusion detection (e.g., detection of unauthorized persons entering a restricted plant area) or (b) follow and observe actions of unauthorized persons in the plant. Whereas, in SAR applications, swarms can be exploited for: (c) generating a situation overview of the disaster scene in case of an industrial plant accident including real-time images (VIS, IR), toxic and explosive gas leakage detection or (d) finding of human casualties or persons trapped in the disaster area. The gathered information is used to help security personnel, first responders as well as rescue teams to conduct their mission efficiently.

The two application scenarios share common requirements: a vast spatial area has to be inspected and information has to be provided to the stakeholders (security personnel, rescue teams, etc.) in real-time, especially in case of an incident. Swarms can reduce the inspection/detection times compared to, e.g., single UAV/rover applications. Especially in SAR a single minute can decide between death and life. On the other hand, the inspection cycle time for surveillance can be reduced considerably by enabling denser inspection (multiple drones).

The core of this first use case is a set of UAVs and rovers that can act autonomously. They typically carry different sensors (VIS or IR cameras, microphones, gas sensors, etc.) and can communicate among each other (via WiFi, 4G or other communication means). The whole ensemble operates as a self-organizing mixed team where particular tasks for vehicles are not predefined at mission start but negotiated during mission execution. The swarm needs to be highly adaptive to changes in the environment and can act dynamically. Moreover, in contrast to fully centralized control, the swarm can still operate even if the connectivity among vehicles or with a base station is sparse.

The swarm mission itself is defined in a central operation center in the beginning of the mission with a dedicated swarm definition tool (mission planner) that defines the goals and behavior of the swarm, thanks to the *CPSwarm Deployment Toolchain*. The central station can additionally collect the sensor data and perform sensor fusion and analysis in real time.

5.2 Autonomous Driving and Platooning

The second use case lies in the area of cooperating autonomous driving vehicles connected over a kind of an electronic draw bar. The leading vehicle acquires and calculates different kind of information for its own automated operation (e.g., from computations or sensors) and sends such information to other vehicles in the *swarm*. We can imagine a scenario where the leading vehicle prescribes the actions and decisions (i.e. navigation, decision on take-over maneuvers, sequencing maneuvers, lane change etc.) for the follow-up vehicle(s) that will make use of the leading vehicle actions. The follow-up vehicle will need full autonomous driving capability and environmental awareness as well, however they will follow the leading vehicle in a preset distance even when they have to make decisions, i.e. in case of lane change maneuver on their own. The follow-up vehicles will also take over full control in case the lane change needs to be interrupted for the complete swarm due to other traffic prohibiting to change lanes. This use case shall pave the way concerning the first technical challenges to solve. The research in the project will focus on the deterministic wireless networking needed for such scenarios and how to model this real-time communication aspects on the swarm level. Another challenge is to create and deploy applications developed using CPSwarm approach on real-world automotive platforms such as AUTOSAR[14].

5.3 Logistics Assistant

The SWARM Logistics Assistant use case involves robots, rovers and drones that collaboratively perform opportunistic scanning of a given warehouse. The involved robots are intended to assist humans in a logistics domain. These assistive tasks include, e.g., joining forces to move a heavy obstacle from one place to another. Robots involved in the scenario are designed to scan the entire area of the warehouse and share the acquired information to update a knowledge base (e.g., map) on the go. In addition to collecting information about the maps of the entire area, the connected robots will also be used for collecting additional information implicitly e.g. room temperature, presence of humans, detection of in-path obstacles, etc. Since the information is acquired collaboratively by all the connected robots of the swarm, the current status of the area is always up to date and the effort is always divided among all members. As a starting point, each connected robot will be fed with some default information e.g. map of the area. This information will then be updated opportunistically on the go as the robots perform their main tasks. The swarm logistics scenario is enabled by the CPSwarm toolchain, and it is aimed at demonstrating the viability and effectiveness of the model-based and predictive design approach promoted by CPSwarm. In the scenario, the CPSwarm Workbench will be exploited to design and test both the single behaviors of involved robots, the foreseen interactions with humans and the overall emerging features of involved robots acting in swarms. The CPSwarm Simulation toolkit will, particularly, be exploited to test reactions of the swarm to situations not encoded and/or known at design time, also involving humans. This will enable proof testing of the logistic assistant policies and behaviors before deployment on the shop floor. Thanks to the bulk deployment tool developed in the project, any discrepancy or any additional feature that might emerge during the experimentation will be easily incorporated in the system design and re-programming of robots will be performed seamlessly, with virtually no down times.

6 CONCLUSIONS & FUTURE WORKS

This paper introduced the CPSwarm project, highlighting the addressed challenges in the Cyber Physical Systems engineering domain. The key points of the proposed approach and the methodology adopted to reach the project goals have been described in details, offering the reader a precise overview of the main project goals and expected outcomes. Within the next 3 years CPSwarm aims at achieving sensible gains in terms of better efficiency of design and deployment of CPS swarms able to collectively address

[14] https://www.autosar.org *last visited on February 21, 2017.*

complex tasks in several domains, from search and rescue to autonomous driving.

7 ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 731946.

REFERENCES

[1] Stefano Arteconi. 2008. *Evolutionary methods for self-organizing cooperation in peer-to-peer networks.* Ph.D. Dissertation. alma. http://amsdottorato.unibo.it/918/
[2] S. Baldi, I. Michailidis, H. Jula, E. B. Kosmatopoulos, and P. A. Ioannou. 2013. A "plug-n-play" computationally efficient approach for control design of large-scale nonlinear systems using co-simulation. In *52nd IEEE Conference on Decision and Control*. 436–441.
[3] A. Bhave, B. H. Krogh, D. Garlan, and B. Schmerl. 2011. View Consistency in Architectures for Cyber-Physical Systems. In *2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*. 151–160.
[4] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Angiovanni-Vincentelli. 2006. Languages and Tools for Hybrid Systems Design. *Found. Trends Electron. Des. Autom.* 1, 1/2 (Jan. 2006), 1–193.
[5] Tom De Wolf and Tom Holvoet. 2007. Design Patterns for Decentralised Coordination in Self-organising Emergent Systems. In *Proceedings of the 4th International Conference on Engineering Self-organising Systems (ESOA'06)*. Springer-Verlag, Berlin, Heidelberg, 28–49.
[6] B. Edmonds and J. J. Bryson. 2004. The insufficiency of formal design methods - the necessity of an experimental approach for the understanding and control of complex MAS. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.* 938–945.
[7] I. Fehervari and W. Elmenreich. 2010. Evolving Neural Network Controllers for a Team of Self-organizing Robots. *Journal of Robotics* (2010).
[8] J. M. E. Gabbai, Hujun Yin, W. A. Wright, and N. M. Allinson. 2005. Self-Organization, Emergence and Multi-Agent Systems. In *2005 International Conference on Neural Networks and Brain*, Vol. 3. nil24–1863.
[9] Carlos Gershenson. 2007. Design and Control of Self-organizing Systems. (2007). http://cogprints.org/5442/
[10] Marie-Pierre Gleizes, Valérie Camps, Jean-Pierre Georgé, and Davy Capera. 2008. *Engineering Systems Which Generate Emergent Functionalities.* Springer Berlin Heidelberg, Berlin, Heidelberg, 58–75.
[11] Volkan Gunes, Steffen Peter, Tony Givargis, and Frank Vahid. 2014. A Survey on Concepts, Applications, and Challenges in Cyber-Physical Systems. *KSII Transactions on Internet and Information Systems (TIIS)* 12, 12 (Dec 2014).
[12] Thomas A. Henzinger. 2000. *Masaccio: A Formal Model for Embedded Components.* Springer Berlin Heidelberg, Berlin, Heidelberg, 549–563.
[13] Timothy W. Hnat, Tamim I. Sookoor, Pieter Hooimeijer, Westley Weimer, and Kamin Whitehouse. 2008. MacroLab: A Vector-based Macroprogramming Framework for Cyber-physical Systems. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys '08)*. ACM, New York, NY, USA, 225–238.
[14] Etienne Brosse Imran Quadri, Alessandra Bagnato and Andrey Sadovykh. 2015. Modeling methodologies for Cyber-Physical Systems: Research field study on inherent and future challenges. *Ada User Journal* 36, 4 (2015), 1666–1671.
[15] J. C. Jensen, D. H. Chang, and E. A. Lee. 2011. A model-based design methodology for cyber-physical systems. In *7th International Wireless Communications and Mobile Computing Conference*. 1666–1671.
[16] E. A. Lee. 2010. CPS foundations. In *Design Automation Conference*. 737–742.
[17] Edward A. Lee. 2016. Fundamental Limits of Cyber-Physical Systems Modeling. *ACM Trans. Cyber-Phys. Syst.* 1, 1, Article 3 (Nov. 2016), 26 pages.
[18] Edward A. Lee and Haiyang Zheng. 2005. *Operational Semantics of Hybrid Systems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 25–53.
[19] Patrick Martin and Magnus B. Egerstedt. 2012. Hybrid systems tools for compiling controllers for cyber-physical systems. *Discrete Event Dynamic Systems* 22, 1 (2012), 101–119.
[20] Iakovos Michailidis, Martin Felix Pichler, and Elias B. Kosmatopoulos. 2013. Multi-Linear State Space Model Identification for Large Scale Building Systems. In *Sustainable Building Conference*.
[21] László Monostori. 2014. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP* 17 (2014), 9 – 13.
[22] A.L. Nelson, E. Grant, and T.C. Henderson. 2004. Evolution of neural controllers for competitive game playing with teams of mobile robots. *Robotics and Autonomous Systems* 46, 3 (2004), 135 – 150.
[23] Object Management Group (OMG). 2008. The UML MARTE Standardized Profile. In *Proceedings of the 17th IFAC World Congress.*
[24] Object Management Group (OMG). 2011. UML 2.4.1 Superstructure Specification. (2011).
[25] OMG. 2012. OMG Systems Modeling Language (OMG SysML), Version 1.3. (2012). http://www.omg.org/spec/SysML/1.3/
[26] Mariachiara Puviani, Giovanna Di Marzo Serugendo, Regina Frei, and Giacomo Cabri. 2012. A Method Fragments Approach to Methodologies for Engineering Self-organizing Systems. *ACM Trans. Auton. Adapt. Syst.* 7, 3, Article 33 (Oct. 2012), 25 pages.
[27] Matt Quinn, Lincoln Smith, Giles Mayley, and Phil Husbands. 2003. Evolving Teamwork and Role-allocation with Real Robots. In *Proceedings of the Eighth International Conference on Artificial Life (ICAL 2003)*. MIT Press, Cambridge, MA, USA, 302–311. 4
[28] Craig W. Reynolds. 1987. Flocks, Herds and Schools: A Distributed Behavioral Model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. ACM, New York, NY, USA, 25–34.
[29] Pritam Roy, Paulo Tabuada, and Rupak Majumdar. 2011. Pessoa 2.0: A Controller Synthesis Tool for Cyber-physical Systems. In *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control (HSCC '11)*. ACM, New York, NY, USA, 315–316.
[30] B. Schätz, M. Törngren, S. Bensalem, M. V. Cengarle, H. Pfeifer, J. McDermid, R. Passerone, and A. Sangiovanni-Vincentelli. 2015. *Research Agenda and Recommendations for Action.* Technical Report. CyPhERS âĂŞ Cyber-Physical European Roadmap & Strategy.
[31] Ken Schwaber. 1997. *SCRUM Development Process.* Springer London, London. 117–134 pages.
[32] B. Silva, K. Richeson, B. Krogh, and A. Chutinan. 2000. Modeling and verifying hybrid dynamic systems using CheckMate. In *Proc. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems*. 323–328.
[33] A. Sobe, I. Fehérvári, and W. Elmenreich. 2012. FREVO: A Tool for Evolving and Evaluating Self-organizing Systems. In *Proceedings of the 1st International Workshop on Evaluation for Self-Adaptive and Self-Organizing Systems*. Lyon, France.
[34] John A. Stankovic. 2016. Research Directions for Cyber Physical Systems in Wireless and Mobile Healthcare. *ACM Trans. Cyber-Phys. Syst.* 1, 1, Article 1 (Nov. 2016), 12 pages.
[35] Jan Sudeikat, Lars Braubach, Alexander Pokahr, Wolfgang Renz, and Winfried Lamersdorf. 2009. Systematically Engineering Self-Organizing Systems: The SodekoVS Approach. *ECEASST* 17 (2009).
[36] Jan Sudeikat and Wolfgang Renz. 2008. *Toward Systemic MAS Development: Enforcing Decentralized Self–organization by Composition and Refinement of Archetype Dynamics.* Springer Berlin Heidelberg, Berlin, Heidelberg, 39–57.
[37] Y. Sun, B. McMillin, X. Liu, and D. Cape. 2007. Verifying Noninterference in a Cyber-Physical System The Advanced Electric Power Grid. In *Seventh International Conference on Quality Software (QSIC 2007)*. 363–369.
[38] V. Trianni and S. Nolfi. 2011. Engineering the Evolution of Self-Organizing Behaviors in Swarm Robotics: A Case Study. *Artificial Life* 17, 3 (July 2011), 183–202.