Six Software Engineering Principles for Smarter Cyber-Physical Systems

Danny Weyns
KU Leuven, Linnaeus University
danny.weyns@kuleuven.be

Tomas Bures
Charles University
bures@d3s.mff.cuni.cz

Radu Calinescu
University of York, UK
radu.calinescu@york.ac.uk

Barnaby Craggs
University of Bristol
barney.craggs@bristol.ac.uk

John Fitzgerald
Newcastle University
john.fitzgerald@ncl.ac.uk

David Garlan
Carnegie Mellon University
garlan@cs.cmu.edu

Bashar Nuseibeh
Open University UK, Lero
bashar.nuseibeh@open.ac.uk

Liliana Pasquale
University College Dublin
liliana.pasquale@ucd.ie

Awais Rashid
University of Bristol
awais.rashid@bristol.ac.uk

Ivan Ruchkin
University of Pennsylvania
iruchkin@cis.upenn.edu

Bradley Schmerl
Carnegie Mellon University
schmerl@cs.cmu.edu

***Abstract*—Cyber-Physical Systems (CPS) integrate computational and physical components. With the digitisation of society and industry and the progressing integration of systems, CPS need to become "smarter" in the sense that they can adapt and learn to handle new and unexpected conditions, and improve over time. Smarter CPS present a combination of challenges that existing engineering methods have difficulties addressing: intertwined digital, physical and social spaces, need for heterogeneous modelling formalisms, demand for context-tied cooperation to achieve system goals, widespread uncertainty and disruptions in changing contexts, inherent human constituents, and continuous encounter with new situations. While approaches have been put forward to deal with some of these challenges, a coherent perspective on engineering smarter CPS is lacking. In this paper, we present six engineering principles for addressing the challenges of smarter CPS. As smarter CPS are software-intensive systems, we approach them from a software engineering perspective with the angle of self-adaptation that offers an effective approach to deal with run-time change. The six principles create an integrated landscape for the engineering and operation of smarter CPS.**

I. Introduction

Cyber-Physical Systems (CPS) are systems that integrate computational and physical processes [1], [2]. The recent progress in digitising the society and integrating computing systems puts pressure on CPS to become "smarter" [2]–[4]. By "smarter" we mean: the systems that perpetually adapt and evolve themselves by improving their existing capabilities and learning new ones to deal with uncertainty, handle unexpected conditions, and accommodate the change they face throughout their lifetime [5]. The trend towards smarter CPS leads to a significant increase in the complexity of these systems, to the extent that software largely determines the design, deployment, and evolution of the CPS. Software takes a central role in realising (system-wide) supervisory control, managing safety and security, and delivering resilience. Examples include smart applications in areas such as mobility, manufacturing, power distribution, water management, and safety in cities. Such systems are complex socio-technical fabric where humans, devices, and software interact in a myriad of ways [6]. In this paper, we use the term "smarter CPS" to mark this qualitative change step in complexity and smartness that emerging and future CPS are exhibiting compared to traditional, more rigid and more isolated CPS. We use the concept of "component" to refer to the constituent parts of these systems, including cyber, physical, and social (human) components, which may themselves be independently owned and managed. Due to their close connection to the physical and social operating context, smarter CPS bring in the systems and systems-of-systems engineering perspective into the software engineering process. Whereas the focus of systems-of-systems is mostly on integrating computing systems, smarter CPS put the emphasis on integrating humans, software and physical components that dynamically aggregate, adapt and evolve to realise and improve their goals in the face of continuous change [7].

Recent discussions within the international research community (e.g., CPS Week, Dagstuhl and Shonan meetings on CPS) have led to insights suggesting that the development of future CPS requires a paradigm shift. This is also reflected in several surveys, see e.g., [8]–[10], and recent funding programs of EU and US NSF. The required shift relates to what CPS engineering methods and techniques need to be brought together in order to achieve the required smartness while dealing with the complexity arising from the scale, connectivity, and inherent uncertainty of these systems. The inherent blend of cyber, physical and social components, demands a novel truly multi-disciplinary engineering approach, as smarter CPS present a combination of challenges that existing modelling and development methods have difficulties addressing: (1) intertwined digital, physical and social spaces, (2) use of heterogeneous models that vary in concepts, formalism, and level of abstraction, (3) need for context-tied cooperation to achieve system goals, (4) widespread uncertainty and disruptions in a broad variety of changing contexts, (5) inherent human constituents, and (6) continuous encounter with new situations.

A number of approaches have been put forward to deal with some of these challenges. Lee [2] argues for model-based design of systems comprising physical and cyber elements, embracing temporal dynamics of CPS. Wang et al. [11] highlight approaches that deal with uncertainties in manufacturing systems, and Pierce et al. [12] present DESTECS that enables system and software engineers to collaboratively produce models that unite a system with controllers. Yet, there is a need for a coherent perspective on engineering smarter CPS.

To that end, this paper contributes a set of six basic principles to guide the engineering of smarter CPS: (1) crossing boundaries, (2) leveraging the human, (3) fluid modelling, (4) on the fly coalitions, (5) dynamically assured resilience, and (6) learn novel tasks. By "principles" we mean ideas, concepts, and mechanisms that offer a basis for solving an engineering problem. Each of the principles takes a particular viewpoint on the engineering of smarter systems, yet the principles are connected. As smarter CPS are software-intensive systems, we approach them mainly from a software engineering perspective, although the presented approaches align with the physical and social aspects of smarter CPS. Our angle is self-adaptation [13], [14] that offers an effective approach to deal with run-time uncertainties that may occur within the system itself, its environment or requirements [15].

We derived the six principles from discussions at the SEsCPS workshops, e.g., [16], and our own experiences with engineering CPS, e.g., [17]–[19]. The complementary set of principles cover the main stages of engineering software-intensive systems: domain engineering, design, operation, and evolution. As such, the engineering principles create a new landscape for the design and operation of smarter CPS.

II. Running Example

To clarify the challenges of smarter CPS and illustrate the six engineering principles, we use a running example of a CPS for rescue coordination after an earthquake. The overall goal of the CPS is to rescue victims from an area with damaged buildings and buildings on fire. The CPS comprises different rescue teams that dynamically inter-operate to coordinate the rescue operations. Rescue teams span multiple rescue agencies, comprising a mix of firefighters and paramedics equipped with sensors in their clothes and smart helmets (through which they can communicate and also receive information and instructions), as well as autonomous robots and unmanned aerial vehicles (UAVs) that can scout areas outside and inside buildings in the hazard area.

III. Software Engineering Principles for Smarter Cyber-Physical Systems

A. Crossing Boundaries

Smarter CPS operate in complex environments that straddle cyber, physical, and social spaces. Crossing boundaries refers to tight interaction between these spaces. Cyber spaces involve applications, networking configurations and infrastructure (e.g., the software components running on robots and sensors that communicate via a network). Physical spaces refer to the elements that are governed by the laws of physics (e.g., buildings and the robots and people that can move through them). Social spaces cover humans roles, knowledge and organisational structure (e.g., the role of firefighters and paramedics involved in the rescue operation, and the organisation they belong to). Cyber, physical and social spaces are highly intertwined, with eroding boundaries and systems often operating across them. For example, robots and rescue workers may only move to parts of the buildings that have not been damaged. This allows forming transient communication networks to exchange relevant information and to coordinate the rescue operations. Figure 1 sketches the different spaces. Components and their evolution are often modelled as graphs. Properties can be modelled by annotating edges with weights, for instance, to represent physical distance or degree of social interaction. However, existing modelling formalisms and tools only partially consider the heterogeneous nature and tight interaction of cyber, physical, and social spaces, see e.g., [20].

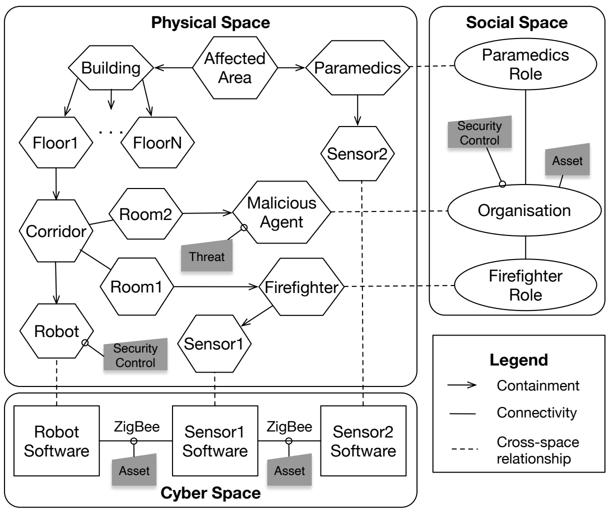


Fig. 1: Modelling example of crossing boundaries.

The principle "crossing boundaries" centres the engineering on the characteristics of the different spaces and their integration. A key characteristic is the structure of spaces in terms of components and their relationships [21]. In a physical sense, a structure denotes the physical characteristics of a space, such as containment (e.g., robots and people inside a building), distance between objects, connectivity (e.g., two areas connected through a corridor), and coverage (e.g., area covered by the network). In a cyber sense, structure can denote the topology and type of the network used by two or more software components to communicate. In a social sense, structure can represent ad-hoc formed organisations of rescue workers that operate in the area affected by the earthquake.

One effect of the tight interaction between cyber, physical, and social spaces is that smarter CPS are more vulnerable to attacks than conventional software. We argue that the structure of different spaces can provide awareness about security concerns such as assets, threats, and the extended attack surface brought by the interplay between cyber, physical and social spaces. This can help identifying more effective security adaptations that can be enacted in the three spaces.

Example. Robots and sensors worn by rescue teams exchange valuable information about the areas of the building that need to be explored during the rescue operation (asset). Since robots and sensors communicate using a protocol (Zigbee here) and may have limited battery, they cannot enable strong security mechanisms to check the authenticity of the information received. A malicious actor can exploit his/her physical proximity to the robot (physical space) to send tampered sensor data (cyber space), possibly slowing down the rescue operation and harming the reputation (asset) of the rescue organisation (social space). Although the assets to be protected belong to the cyber and social space, a security adaptation can be enacted in the physical space (e.g., a robot should identify unauthorized individuals and force them to leave the building). Alternatively, a malicious agent can take advantage of his/her affiliation with the rescue organisation (social space) to access and disclose sensitive information about the identity of the people that have been rescued (asset), which is exchanged between firefighters and paramedics. In this situation, stronger authentication and encryption mechanisms can be enabled when sensitive information is transmitted over the network (cyber space). The rescue organisation can also restrict access to sensitive information only to the firefighters and paramedics directly involved in the rescue operations (social space).

B. *Leveraging the Human*

As we have argued above, smarter CPS are built and used by humans whose behaviours intersect with the physical and cyber components in a range of complex ways [22]. Humans are not merely developers or users of CPS but an integral functional part of the information flow [23] (e.g., firefighters and paramedics equipped with wear-able technology operating in a hazardous area form ad-hoc collaborations and information flows to realise the rescue mission). Humans can play an adversarial role too (as illustrated above). The inherent uncertainty of humans in their interaction with technologies may result in latent failures with profound impact (e.g., human decision-making in the rescue CPS made under extreme conditions may jeopardise dynamic coalition formation of rescue personnel, robots and UAVs). Research has shown that leaving humans out of the loop can lead to failures that impact key properties of a CPS such as safety and security, see e.g., [19]. Hence, engineering smarter CPS must systematically account for the integral role of humans throughout the CPS life-cycle and not be left as a bolt-on or additional feature. This underpins the need to reason about adaptation across the boundaries of cyber, physical and social spaces.

The principle "leveraging the human" integrates humans in the design and operation as first-class citizens, in particular in runtime models of the system as "beyond-control" components. Treating humans as inherent components in the overall modelling of a CPS enables incorporating the behaviour of people when reasoning about the CPS and its adaptations. The "beyond-control" property expresses that these components cannot be controlled directly (contrary to traditional system components) and that the true state of these components is unknown, but has to be inferred from observations and the humans' actions influenced and guided through information delivered to them. Explicitly considering and modelling humans makes it possible to create interesting information flows that become key links in the CPS (e.g., expertise of individuals in crisis management captured in probabilistic models). On the other hand, this makes it also possible to account for potentially unexpected behaviour of humans and a potential gap in understanding between HW/SW components and humans, which relates to the question about how technologies may impact human behaviours and where the intersections may create latent failure. First-class integration of humans in CPS can help ensuring that known human (active) errors do not happen at the same time as system (latent) failures occur, which, if not anticipated, may compromise the integrity of the CPS and those of the (often critical) services they deliver.

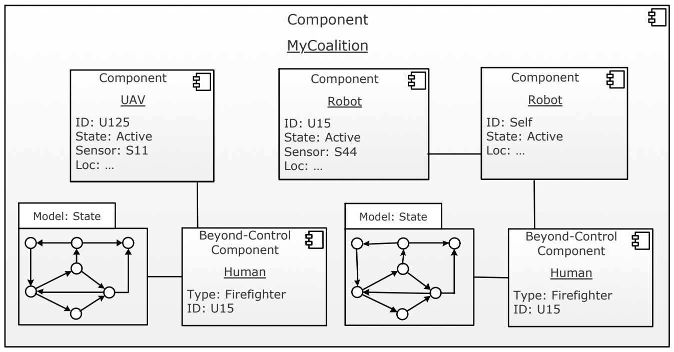


Fig. 2: Modelling example of leveraging the human.

Example. The component MyCoalition deployed on each robot represents knowledge of a robot's current coalition, see Figure 2. The state of firefighters (that are modelled as beyond-control components) is captured in probabilistic models that are kept up to data using sensors (from other devices). Given the probabilistic nature of the state of beyond-control components and the state of the other components of the coalition, the decisions of the robot can take into account modelled expertise of human actors to make effective decisions, while ensuring the integrity of the rescue team and its mission at all times.

C. *Fluid Modelling*

Complex CPS are typically built with multidisciplinary teams that use heterogeneous models that vary in formalisms, concepts, and levels of abstraction. The outputs and guarantees of these models are combined in the deployed system. Such fluid modelling is a natural outcome of the different specification and verification approaches. Since the models are designed separately, inter-dependencies naturally occur. This often results in mismatches between models, introducing errors [24] (e.g., the movements of UAVs may considered successful in the planning model, yet, they may fail in the real deployment due to differences in the assumed and actual consumption of energy). To prevent such errors, inconsistencies between models need to be detected by integrating the heterogeneous models with proper semantics [25]. Adopting a single, all-encompassing homogeneous modelling language denies the fact that domain-specific formalisms are better at verifying properties for their domain, and that there is usually a well-established body of knowledge and expertise built up around these formalisms. On the other hand, creating simplified abstractions of each model and relating them often sacrifices more sophisticated interactions (e.g., related to dynamic behaviour) that make such properties unverifiable.

The principle "fluid modelling" integrates heterogeneous models by explicitly stating the relationships between them as an *integration property* — a logical assertion over several models [26]. Such an approach prescribes three steps:

1) Create integration abstractions by abstracting structural and behavioural elements of models that are important for stating desired integration properties.
2) Specify multi-model integration properties to express relationships between the abstractions of models; the verify the properties by combining constraint satisfaction problem solvers and verifiers for individual models.
3) Execute domain-specific analyses on consistent models (facilitated e.g. by analysis contracts [18]). This ensures that effects of analyses update models without introducing errors or violating model consistency.

This approach can be used across physical, software, and human models required for specifying and verifying CPS.

Example. As explained above, one integration property that needs to be checked for the earthquake rescue system is that the energy estimates used in path planning agree with those used in a power model, within a given error threshold. Suppose that path planning uses an environment model of UAV movements, with their timings and energy estimates. The behaviours in this model can be checked with properties in, say, Probabilistic Computation Tree Logic (PCTL) [27]. The power model has detailed descriptions of battery dynamics and hardware-specific movement operations (e.g., finer-grained motor actuation). Applying the approach outlined above gives:

1) Abstract the power-related plans using PCTL, and movement operations — as a collection of annotated objects, also known as a *view* (rectangles in Figure 3).
2) Specify the integration property that the energy estimates agree between the models for the abstractions defined in Step 1, and verifies it with the help of an SMT solver and a probabilistic model checker (star in Figure 3).
3) Execute analyses and re-verification according to their dependencies. E.g., if some analysis changes control gains for motors, engineers need to update both models and re-verify the integration property (parallelograms).

This approach is fluid in three ways: it can handle changes of models throughout their lifetime, incorporate new models by developing abstractions for them, and adjust to uncertainty in models by relaxing the bounds in integration properties.

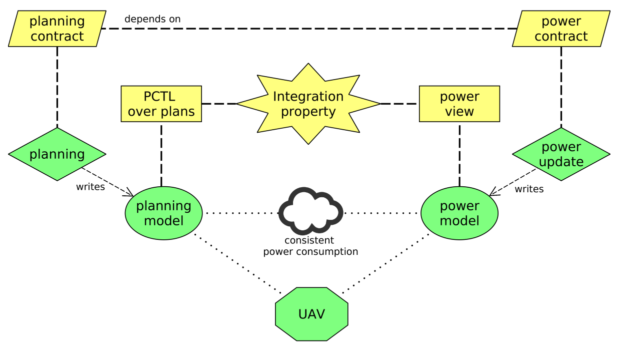


Fig. 3: Fluid approach for dynamic adaptation and integration of models and analysis. Green (dark) elements represent existing artefacts. Yellow (bright) elements are dynamically added.

D. *On the Fly Coalitions*

A cornerstone of smarter CPS to address complex problems is cooperation across entities within the CPS and with external systems. This cooperation, in the form of data exchange and coordinated activities, enables the constituting components of smarter CPS to work together as a multi-agent system [28], [29] (e.g., coordinated flight patterns of UAVs to coordinate scouting, and coordination of firefighters and paramedics in rescue operations). The common denominator of such cooperation is that it is tied to the operating context (e.g., a coalition of firefighters and paramedics from different rescue agencies would be dynamically formed if injured victims are discovered in a collapsed building; yet, once the victims are saved or if an exceptional event happens, such as a gas leak is detected close-by, the coalition is dynamically dissolved or rearranged with another objective or strategy to achieve its current objective). This leads to a highly adaptive system architecture, which opportunistically reshapes at runtime to address potentially new situations at hand. This level of adaptability is difficult to tackle with traditional engineering approaches that either put the focus on individual entities in the system or lack first-class support for truly open collaborations between entities [29].

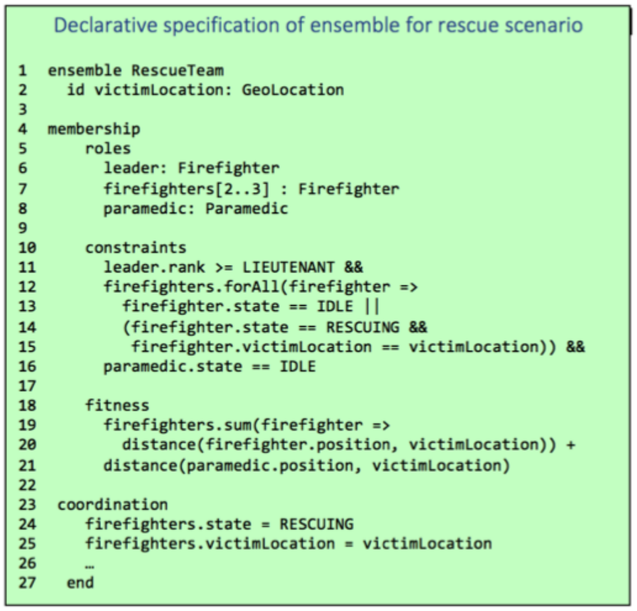
The principle "on-the-fly coalitions" makes such dynamic collaborations first-class concepts. The DEECo component model [30] is one such realisation. Here a specification of a coalition is captured by the concept of an ensemble, which is a first-class modelling and runtime concept, along with so called autonomic components that represent individual entities (e.g. a firefighter or paramedic). The ensemble embodies the coordination needed among multiple entities for the completion of a specific goal (e.g. to rescue a victim). The ensemble acts as a type, which is instantiated for a particular matching situation (e.g. to form a team when a victim is identified).

Example. Figure 4 show an ensemble type that declaratively prescribes what types of entities it needs and what properties should hold about the actual state of the entities – e.g. that one firefighter needs to act as a leader (line 6) with a minimum rank of lieutenant (line 11), and two or three additional firefighters and one paramedic are needed (lines 7, 8). These workers should be idle or be at the victim location (lines 12 to 16). Should there be more options for choosing the available firefighters and paramedic, they should be chosen based on their state and proximity to the victim location (lines 18 to 21). Once an ensemble is established, it assigns tasks to the autonomic components in the ensemble (from line 23).

The selection of autonomic components to form an ensemble happens dynamically at runtime and reflects the current state and availability of the firefighters and paramedics. Thanks to the declarative specification of the ensemble type, the ensemble instance can also seamlessly and automatically reshape at runtime in case of exceptional situations (e.g. in case a firefighter is injured, she is automatically removed from the ensemble and another available firefighter joins the ensemble).

E. *Dynamically Assured Resilience*

Smarter CPS will play key roles in domains of significant economic and societal importance. These CPS must withstand uncertainties and disruptions with acceptable impact on their


Declarative specification of ensemble for rescue scenario

```
ensemble RescueTeam
  id victimLocation: GeoLocation

membership
    roles
      leader: Firefighter
      firefighters[2..3] : Firefighter
      paramedic: Paramedic

    constraints
      leader.rank >= LIEUTENANT &&
      firefighters.forAll(firefighter =>
        firefighter.state == IDLE ||
        (firefighter.state == RESCUING &&
         firefighter.victimLocation == victimLocation)) &&
      paramedic.state == IDLE

    fitness
      firefighters.sum(firefighter =>
        distance(firefighter.position, victimLocation)) +
      distance(paramedic.position, victimLocation)

 coordination
    firefighters.state = RESCUING
    firefighters.victimLocation = victimLocation
    …
  end
```

Fig. 4: Excerpt of an ensemble specification.

provided service, i.e., they must be resilient. In critical applications, this resilience must be guaranteed by assurance evidence enabling stakeholders to trust the CPS [31]. Since the wide variety of changing contexts that smarter CPS operate in cannot be fully anticipated before operation, assuring their resilience must include activities performed throughout the CPS lifetime. First, requirements analysis, design and implementation must devise the functionalities for the CPS components, the architecture with variability points, and the actual code (e.g., functionality for navigation, victim detection, and communication capabilities of robots and UAVs, and the ways in which components of different rescue agencies could be integrated). Next, verification and validation must generate preliminary assurance evidence, and a partial assurance argument with placeholders for the evidence unavailable until runtime (e.g., a combination of formal verification, simulation, and testing can be used to gather preliminary assurance evidence for the rescue case). Finally, fully-fledged CPS configurations and plans, with dynamic adaptations of configurations and online re-planning, and the additional assurance evidence required to complete the assurance argument, must be synthesised at runtime, when the operation context is known and its uncertainty can be quantified and resolved through adaptation (e.g., this may involve allocating rescue tasks to CPS components based on their capabilities and the characteristics and actual conditions of the buildings and the terrain affected by the earthquake).

The principle of "dynamically assured resilience" achieves this dynamically assured resilience through the use of runtime mathematically based modelling, analysis and synthesis techniques such as stochastic and statistical model checking [32] and runtime verification [33]. As these techniques are computationally intensive and adopting new CPS configurations and plans can consume considerable time and resources, system-level runtime reconfiguration should only be required infrequently, after major disruptions. Accordingly, each new CPS configuration or plan synthesised at runtime must be robust, i.e., capable of handling a range of context changes.

Example. Figure 5 illustrates how the rescue CPS synthesises resilient configurations and plans when a disruptive change occurs, i.e., a collapsed building catches fire during the mission.

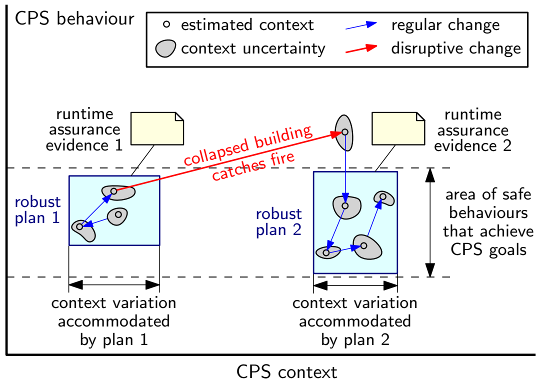

CPS behaviour
estimated context
context uncertainty
regular change
disruptive change
runtime assurance evidence 1
runtime assurance evidence 2
collapsed building catches fire
robust plan 1
robust plan 2
area of safe behaviours that achieve CPS goals
context variation accommodated by plan 1
context variation accommodated by plan 2
CPS context

Fig. 5: Robust plan synthesised after a major disruption, with assurance evidence for the intended application and context.

F. *Learn Novel Tasks*

Smarter CPS inevitably will encounter novel scenarios during their lifetime (e.g., robots may need to find paths in environments that they have not encountered before). Reacting to novel situations via manual interventions would require tremendous efforts and may be too slow in critical situations. Dealing with these intrinsic challenges requires smarter CPS to preserve knowledge from the past and utilise this knowledge efficiently when performing novel tasks in the future.

The principle of "learn novel tasks" leverages on recent progress in the field of machine learning and lifelong learning [34] that enables a learner to deal with new learning tasks from many related tasks. To that end, a CPS needs to be enhanced with a meta-learning system with facilities to store knowledge and exploit this knowledge to evolve a learner from experiences of executed tasks. The basic modules are:

- A concept generator extracts high-level concepts from executing many concrete tasks or historical data of executing tasks. This generator can for instance be realised using a deep neural network [35].
- The concept discriminator predicts labels for the concepts. The concept generator can be realised using a learning method such as a classifier [36].
- A meta-learner exploits the extracted knowledge to teach a learner how to deal with new situations by gradually updating the learner while tasks are performed. Meta-learners exist for different types of learning tasks [37].

Example. Figure 6 illustrates how the "learn novel tasks" can teach a path finder of a robot to find paths in buildings when the robot encounters new conditions. The concept generator extracts high-level concepts from executing concrete rescue tasks such as "pattern for team composition" and "delegation of function." From these concepts, the concept discriminator predicts labels such as "master-slave" and "on the fly planning by team member" that can, for instance, be applied to compose or dynamically adapt a team of robots to rescue victims and find paths in buildings that were not encountered before.

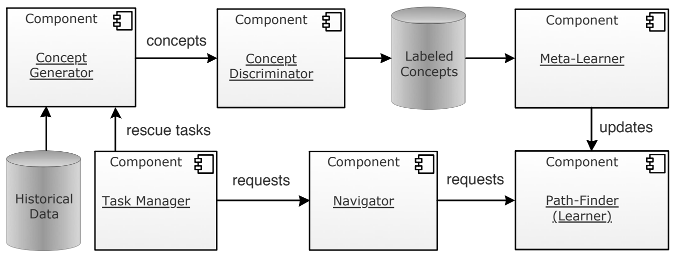


Fig. 6: Learning a path finder learner to deal with novel tasks.

Robots and UAVs equipped with such a lifelong learning system will be able to quickly learn to behave efficiently in new situations based on knowledge learned over time.

IV. New Landscape for Engineering Smarter CPS

The six principles for engineering smarter CPS cover the main stages of the lifetime of CPS:

Domain engineering: "crossing boundaries" and "leveraging the human" tackle the eroding and dynamic boundaries of digital, physical, and social spaces, and the impact of humans as inherent constituents of next-generation CPS.

Design: "fluid modelling" and "on the fly coalitions" tackle the need for aligning abstractions and the demand for context-tied adaptation between cooperating CPS components.

Operation: "dynamic assured resilience" tackles the need for continuous adaptations and guaranteed resilience under widespread uncertainty and disruptions of CPS at runtime.

Evolution: "learn novel tasks" tackles the need for automated evolution of CPS when facing new situations.

The six principles suggest that much is expected of future CPSs and their engineering. We need new tool chains but also engineers with a deep rooted understanding of how to develop software that is able to adapt and evolve under continuous change. As our rescue example shows, future CPSs are composed through interleaving computational, physical and social components and their spaces, linking independent constituent systems, many of them with sources of uncertainty that require on-the-fly adaptation and learning to evolve.

The emerging message is that CPS engineering challenges established boundaries in two main dimensions: boundaries between engineering disciplines on the one hand — and boundaries between development and operation on the other hand. We offer the six principles presented in this paper as a basis for engineers to address the challenges that come with those fading boundaries when building smarter CPS.

References

[1] R. Baheti and H. Gill, "Cyber-physical systems," in *2019 IEEE International Conference on Mechatronics (ICM)*, vol. 1, pp. 430–432, 2019.

[2] E. A. Lee, "Cyber physical systems: Design challenges," in *Object and Component-Oriented Real-Time Distributed Computing*, 2008.

[3] M. Bujorianu and H. Barringer, "Cyber-physical systems," *The Impact of Control Technology*, vol. 12, no. 1, pp. 161–166, 2011.

[4] J. Tavčar and I. Horváth, "A review of the principles of designing smart cyber-physical systems for run-time adaptation: Learned lessons and open issues," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 145–158, 2019.

[5] D. Weyns *et al.*, "A research agenda for smarter cyber-physical systems," *Journal of Integrated Design and Process Science*, 2021 (in print).

[6] R. Calinescu, J. Cámara, and C. Paterson, "Socio-cyber-physical systems: Models, opportunities, open challenges," in *Software Engineering for Smart Cyber-Physical Systems*, 2019.

[7] K. L. Bellman *et al.*, "Special issue on "self-improving self integration"," *Future Generation Computer Systems*, vol. 119, 2021.

[8] J. A. Stankovic, "Research directions for cps in wireless and mobile healthcare," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 1, 2016.

[9] V. Gunes *et al.*, "A survey on concepts, applications, and challenges in cyber-physical systems," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 12, pp. 4242–4268, 2014.

[10] K. Zhou, T. Liu, and L. Zhou, "Industry 4.0: Towards future industrial opportunities and challenges," in *International Conference on Fuzzy Systems and Knowledge Discovery*, 2015.

[11] L. Wang, M. Törngren, and M. Onori, "Current status and advancement of cyber-physical systems in manufacturing," *Journal of Manufacturing Systems*, vol. 37, pp. 517 – 527, 2015.

[12] K. Pierce *et al.*, "Collaborative modelling and co-simulation with DESTECS: A pilot study," in *21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2012.

[13] D. Garlan *et al.*, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, 2004.

[14] D. Weyns, *Introduction to Self-Adaptive Systems, A Contemporary Software Engineering Perspective*. Wiley - IEEE, 2020.

[15] R. Calinescu *et al.*, "Understanding uncertainty in self-adaptive systems," in *International Conference on Autonomic Computing and Self-Organizing Systems*, 2020.

[16] T. Bures *et al.*, "Software Engineering for Smart Cyber-Physical Systems (SEsCPS 2018)," *ACM SIGSOFT Softw. Eng. Notes*, vol. 44, no. 4, 2019.

[17] C. Tsigkanos *et al.*, "Ariadne: Topology aware adaptive security for cps," in *International Conference on Software Engineering*, 2015.

[18] I. Ruchkin *et al.*, "Contract-based integration of cyber-physical analyses," in *International Conference on Embedded Software*, 2014.

[19] B. Craggs and A. Rashid, "Smart cyber-physical systems: Beyond usable security to security ergonomics by design," in *International Workshop on Software Engineering for Smart Cyber-Physical Systems*, 2017.

[20] P. G. Larsen *et al.*, "Integrated tool chain for model-based design of cyber-physical systems: The into-cps project," in *CPS Data*, 2016.

[21] B. A. Yilma, H. Panetto, and Y. Naudet, "Systemic formalisation of Cyber-Physical-Social System (CPSS): A systematic literature review," *Computers in Industry*, vol. 129, p. 103458, 2021.

[22] D. S. Nunes, P. Zhang, and J. Sá Silva, "A survey on human-in-the-loop applications towards an internet of all," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 944–965, 2015.

[23] B. A. Yilma, H. Panetto, and Y. Naudet, "A Meta-Model of Cyber-Physical-Social System: The CPSS Paradigm to Support Human-Machine Collaboration in Industry 4.0," in *Collaborative Networks and Digital Transformation*, Springer, 2019.

[24] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling cyber–physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[25] Y. Liu *et al.*, "Review on cyber-physical systems," *Journal of Automatica Sinica*, vol. 4, no. 1, pp. 27–40, 2017.

[26] I. Ruchkin, *Integration of Modeling Methods for Cyber-Physical Systems*. PhD Thesis, Carnegie Mellon University, Mar. 2019.

[27] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, pp. 512–535, 1994.

[28] M. Z. Candra and H.-L. Truong, "Reliable coordination patterns in cyber-physical-social systems," in *2016 International Conference on Data and Software Engineering*, pp. 1–6, 2016.

[29] Y. Rizk, M. Awad, and E. W. Tunstel, "Decision making in multiagent systems: A survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 514–529, 2018.

[30] T. Bures *et al.*, "Deeco: An ensemble-based component system," in *ACM Sigsoft Symposium on Component-Based Software Engineering*, 2013.

[31] de la Vara *et al.*, "Amass: A large-scale european project to improve the assurance and certification of cyber-physical systems," in *Product-Focused Software Process Improvement*, Springer, 2019.

[32] D. Weyns and M. U. Iftikhar, "Activforms: A model-based approach to engineer self-adaptive systems," *CoRR*, vol. abs/1908.11179, 2019.

[33] R. Calinescu *et al.*, "Engineering trustworthy self-adaptive software with dynamic assurance cases," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1039–1069, 2018.

[34] Z. Chen and B. Liu, *Lifelong Machine Learning, Second Edition*. Morgan and Claypool, 2018.

[35] W. Liu *et al.*, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11 – 26, 2017.

[36] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," in *Emerging Artificial Intelligence Applications in Computer Engineering*, IOS, 2007.

[37] J. Vanschoren, "Meta-learning: A survey," in *arXiv, 1810.03548, cs.LG*, https://arxiv.org/abs/1810.03548, 2018.