DEECo: An Ecosystem for Cyber-Physical Systems

Rima Al Ali, Tomas Bures, Ilias Gerostathopoulos, Petr Hnetynka,
Jaroslav Keznikl, Michal Kit, Frantisek Plasil
Charles University in Prague, Faculty of Mathematics and Physics
Department of Distributed and Dependable Systems
Malostranske namesti 25, Prague 1, 11800, Czech Republic
{alali,bures,iliasg,hnetynka,keznikl,kit,plasil}@d3s.mff.cuni.cz

ABSTRACT

In this work we tackle the problem of designing and developing software-intensive cyber-physical systems (CPS), which are large distributed systems of collaborating elements that closely interact with the physical world, such as intelligent transportation systems and crowdsourcing applications. Due to their specific constraints, such as extreme dynamism and continuous evolution of the physical substratum, and requirements, such us open-endedness and adaptability, CPS introduce many new challenges for software engineering. In response, we present a tailored ecosystem of software engineering models, methods, and tools. This ecosystem is centered on the DEECo component model, which we have proposed specifically for architecting software-intensive CPS.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *Distributed applications*; D.2.6 [**Software Engineering**]: Programming Environments – *Integrated environments*; D.2.11: Software Architectures.

General Terms

Design, Verification

Keywords

Cyber-physical systems; software engineering; component-based systems; ensembles of components

1. INTRODUCTION

Cyber-physical systems (CPS) are large distributed systems consisting of mobile computing devices (embedded computers, smartphones), which closely interact with their physical environment by sensing and actuating. Examples are numerous: intelligent transportation systems, emergency coordination systems, smart grids, etc. CPS are typically highly dynamic and open-ended, and, at the same time, are required to be resilient and self-adaptive to avoid contingency situations. Due to the specific requirements and constraints of CPS, many of the assumptions which typically hold when building traditional systems (reachability, availability of global state, controlled dynamism, etc.) are violated, making the software engineering of CPS a distinct challenge.

The CPS requirements and constraints are well manifested in the e-mobility case study – a case study proposed in scope of the EU FP7 project ASCENS. The case study deals with collaborative route planning of intelligent electric cars (e-cars). Each e-car has to plan and execute a route schedule, given a set of points-of-interest (POIs) to be reached within particular time and energy constraints. The actual schedule has to take into account also the necessity to recharge the vehicle during stop-overs at parking-lot/charging-stations (PLCS). While each e-car is able to monitor and predict its battery level, energy consumption, and the traffic density, it also needs to communicate with the PLCS along the route and with other e-cars to exchange information about traffic jams, road closings, PLCS availability, etc.

Implementing such a system using traditional ways, relying on explicit communication using messages and/or method calls, would be difficult since the system architecture is subject to frequent changes (different cars within direct communication range, closed PLCS, etc.) and has to be decentralized for the system to scale.

In our research, we aim at finding an efficient way to systematically design and develop such CPS via a synergy and adaptation of well-known techniques, centered on component-based software engineering [4]. In this context, we strive for a comprehensive ecosystem of software models and software engineering methods. Specifically, we focus on:

(i) architectural modeling, building on the concept of distributed software components [3] with clear execution and interaction semantics;
(ii) execution environment implementing the semantics of (i) in a way that is suitable for distributed and decentralized execution;
(iii) high-level (requirements-oriented) design process [6] on top of (i), with special focus on autonomous behavior and distributed collaboration;
(iv) design-time and runtime analysis (e.g., functional verification, timing analysis), which would provide the necessary dependability bounds and control the emergent behavior of CPS.

2. THE DEECo ECOSYSTEM

Common denominator of the methods in the proposed ecosystem is the DEECo component model [3] (Dependable Emergent Ensembles of Components), which we have designed specifically for architecting software-intensive CPS.

From the perspective of the DEECo component model, each CPS entity (e.g., e-car and PLCS from the case study) is modeled as a *component* (see). Each component constitutes state, referred to as *knowledge*, and behavior, expressed in terms of a set of *processes*. Each process is executed cyclically in a soft real-time fashion, where each execution consists of atomic read of inputs, execution of the process body, and atomic write of outputs. Specifically, a process can access only local knowledge of the corresponding component. To achieve component interaction, components are dynamically composed into groups called

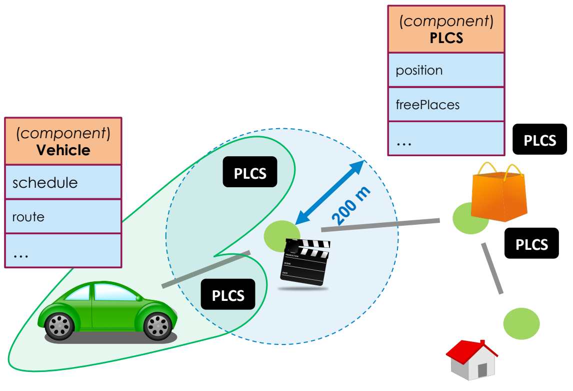

Figure 1: E-mobility: components and ensembles.

ensembles. Membership of a component in an ensemble is declaratively expressed in terms of the *membership condition*, defined upon component *interfaces*, which provide a partial view on component knowledge. Members of an ensemble interact in terms of implicit *knowledge exchange*, which is handled by the execution environment. Thus, a component operates in autonomy, based solely upon its own knowledge, which is implicitly updated depending on the ensembles the component belongs to at a particular moment. Note that a single component can be involved in several ensembles at the same time.

In order to bring the DEECo concepts close to regular software development for evaluation and experimentation, we have implemented the DEECo execution environment in Java [5]. The DEECo concepts (components and ensembles) are mapped to Java via an internal domain-specific language realized by Java annotations. The core responsibility of the runtime environment – execution of knowledge exchange – is implemented in two ways: (i) via distributed tuple-space middleware, and (ii) via periodic broadcast, supporting deployment on mobile ad-hoc networks (MANETs).

Complementary to the component model of DEECo, we have proposed a requirements-oriented design method called *Invariant Refinement Method* (IRM) [6]. The main idea of IRM is to capture the high-level goals and requirements in terms of *invariants*, which describe the desired state of the system-to-be at every time instant. Invariants are to be maintained by the coordination of the different system components. As a design decision, top-level invariants are iteratively decomposed into more concrete sub-invariants, forming a decomposition graph with traceability of design decisions. The decomposition process ends when the (leaf) invariants represent a detailed design of the system implementation – either in terms of local component behavior, corresponding to a component process, or in terms of component interaction, corresponding to an ensemble.

To enable design-time analysis, DEECo is further equipped with a well-defined computational model which allows for formal reasoning [2]. A DEECo-based application is essentially modeled as a finite-state transition system, whose states capture the knowledge of the system's components and the transitions correspond to execution of component processes or ensemble knowledge exchange. Such formal model is then analyzed via standard model-checking means; i.e., by verification of properties in temporal logic.

3. CONTEXT AND EXPERIENCE

Individually, there are a number of approaches from different areas that partially tackle the challenges of CPS. Component-based approaches provide the concepts of encapsulation and well-defined software architectures, useful in complex software systems in general. Agent-based approaches provide conceptual autonomy to the loosely coupled system parts, useful in dynamic and unpredictable environments. The emerging concept of attribute-based communication models the communication as best effort and localized to dynamically changing groups (ensembles) of components. Finally, real-time and control engineering feature cyclic execution that maintains the operational normalcy of a system. Intuitively, our approach tries to combine all the advantages from these approaches into a comprehensive framework for building CPS.

Several case studies from different application domains (including e-cars) have been implemented in jDEECo. Our experience so far shows that DEECo concepts combine well the encapsulation and modularity brought by components with the needs of autonomic behavior and highly dynamic architecture. The overall ecosystem and associated design and analysis methods also complement well the DEECo concepts.

4. CONCLUSION AND CURRENT FOCUS

In our work, we have so far created the basic building blocks of the DEECo ecosystem – i.e., the component model, a Java-based execution environment, a tailored design method, and formal analysis based on formal semantics. Our current focus spans: all the four above-mentioned parts of the ecosystem:

- On the architecture side, we are working on providing ensemble patterns, whose instantiation corresponds to well-known protocols, following our observation that protocol-based communication is challenging in CPS.
- On the implementation side, we are currently combining execution environment with a network simulation environment, to test the scalability of our approach.
- On the design side, we are extending IRM to support elaboration and combination of alternative designs, each focusing on different operational situations.
- On the analysis side, we are currently focusing on estimating impact of communication delays on accuracy of exchanged data at design time and runtime [1], in order to facilitate system robustness..

5. ACKNOWLEDGMENTS

This work has been supported by the European Union Seventh Framework Programme FP7-PEOPLE-2010-ITN under grant agreement n°264840, and by the EU project ASCENS 257414.

6. REFERENCES

[1] Al Ali, R., Bures, T., Gerostathopoulos, I., Keznikl, J. and Plasil, F. 2014. Architecture Adaptation Based on Belief Inaccuracy Estimation. *To appear in Proc. of WICSA'14*.

[2] Barnat, J., Benes, N., Bures, T., Cerna, I., Keznikl, J. and Plasil, F. 2013. Towards Verification of Ensemble-Based Component Systems. *To appear in Proc. of FACS'13*.

[3] Bures, T., Gerostathopoulos, I., Hnetynka, P., Keznikl, J., Kit, M. and Plasil, F. 2013. DEECo – an Ensemble-Based Component System. *Proc. of CBSE'13*, ACM, 81–90.

[4] Crnkovic, I. 2002. *Building Reliable Component-Based Software Systems*. Artech House, Inc., Norwood, MA.

[5] jDEECo Website. *https://github.com/d3scomp/JDEECo*.

[6] Keznikl, J., Bures, T., Plasil, F., Gerostathopoulos, I., Hnetynka, P. and Hoch, N. 2013. Design of Ensemble-Based Component Systems by Invariant Refinement. *Proc. of CBSE'13*, ACM, 91–100.