DEECo: an Ecosystem for Cyber-Physical Systems

Rima Al Ali, Tomas Bures, Ilias Gerostathopoulos, Petr Hnetynka, Jaroslav Keznikl, Michal Kit, and Frantisek Plasil

Charles University in Prague, Faculty of Mathematics and Physics, Department of Distributed and Dependable Systems

DEECo Ecosystem is ...
Software Engineering for Future CPS

A collection of models, methods, and tools for engineering future Cyber-Physical Systems (CPS).

Future CPS typically are
- highly distributed & decentralized
- comprised of mobile computing devices
- operating opportunistically (think of utility/volunteer computing)
- inherently dynamic (physical environment can be "wild")

We want them to be
- autonomous: subsystems operate independently, even detached from peers
- autonomic: subsystems operate optimally with small (or no) supervision
- open-ended: plug & play of new devices/functionalities, large scale
- self-aware: self-managing, self-healing, self-adaptive, self-organizing...

... centered on an Architectural Model
Component Model

Main DEECo concepts
- Component
- Ensemble

A single component comprises
- knowledge (state)
- processes (behavior)

Each process execution consists of atomic read of inputs, execution of the process body, and atomic write of outputs. A process can access only local knowledge of the component.

Dynamic component composition into Ensembles
- attribute-based communication
- declarative membership condition upon component interfaces
- members interact via implicit knowledge exchange handled by execution environment

A component operates in autonomy, based solely upon its own knowledge, which is implicitly updated as the component enters different ensembles.

A single component can be involved in several ensembles at the same time.

... prepared for Design
Invariant-based Design

Invariant Refinement Method
- Invariants capture operational normativity
- Iterative decomposition results into contractual design
- Traceability between requirements and architecture

"INV (Invariant Refinement Method): is a requirements-oriented design method for DEECo-based systems. Its main idea is to capture the high-level goals and requirements in terms of invariants describing the desired state of the system to be at every time instant. Invariants are to be maintained by the coordination of the different components in the system. Top-level invariants are hierarchically decomposed with the leaf invariants corresponding to either a single component process, or an ensemble.

... validated on a Case Study
E-Mobility

[Diagram of E-Mobility]

Collaborative route planning of intelligent electric cars (i.e., cars), where each i-car:
- plans its route according to a set of points-of-interest (POIs) to be reached within particular time and energy constraints
- may stop and/or recharge on parking lots/charging stations (PLCS)
- needs also to communicate with the PLCS along the route and with other i-cars
- reserves PLCS close to its POIs

... implemented in Java
Execution Environment

Mapping the concepts (components and ensembles) to Java
- via an internal domain-specific language
- realized by Java annotations

[jDEECo] provides a fully distributed execution environment for DEECo components and ensembles. It is responsible for process scheduling and handling knowledge exchange. It manages necessary network traffic, relying on low-level communication protocols (e.g., Gossip for MANETs).

[jDEECo] is available at: [https://github.com/d3scomp/jDEECo.git](https://github.com/d3scomp/jDEECo.git)

... deployable on MANETs
Different Deployments

Support for both IP-based and broadcast-based networks
- possible deployment in MANETs

[Diagram of MANETs]

A system is modeled as a finite state transition system, whose states constitute the component knowledge and the transitions correspond to execution of component processes or ensemble knowledge exchange. The model is validated against standard models in a formal manner, i.e., by verification through traces and checking properties.

... featuring Formal Semantics

Expressed by finite state transition system
- Allows for formal analysis

[Diagram of Formal Semantics]

A system is modeled as a finite state transition system, whose states constitute the component knowledge and the transitions correspond to execution of component processes or ensemble knowledge exchange. The model is validated against standard models in a formal manner, i.e., by verification through traces and checking properties.

... looking into the future. Current Focus

Designing ensemble patterns, whose instantiation corresponds to well-known protocols
- Enabling new protocols
- Combining the execution environment with a network simulation environment (e.g., NS-3)
- Extending JVM to support elaboration and combination of alternative designs, each focusing on different situations the system can encounter
- Extending impact of communication delays or accuracy of exchanged data in design-time and runtime

[Diagram of Future Focus]

Increase system trust and robustness

References


Contact

Department of Dependable and Dependable Systems
Charles University in Prague
h3s.mff.cuni.cz