awareness magazine
SELF-AWARENESS IN AUTONOMIC SYSTEMS
10.2417/3201209.004415

Autonomous components in dynamic environments

Tomáš Bureš, Ilias Gerostathopoulos, Petr Hnětynka, Jaroslav Keznikl, Michal Kit, and František Plášil

A new component model provides abstractions for efficient development of autonomous, self-adaptive systems operating in distributed, open-ended dynamic environments.

Distributed systems such as cloud computation, swarm robotics, and intelligent navigation of electric vehicles are gaining popularity. The common characteristic of most distributed systems is that they need a significant level of self-awareness and autonomy, meaning that they can dynamically adapt to changes in their operation environment and recover from potential failures.

To build such systems, the Autonomous Service-Component Ensembles (ASCENS) project features components as autonomous and adaptive entities which form dynamic groups (called ensembles) to achieve common goals.[1] Ensembles aggregate the properties of their members and provide means for communication and collaboration, essential to pursuing group objectives (see Figure 1). These ideas are formalized in the Service-Component Ensemble Language (SCEL) developed under the ASCENS project.[2] In order to provide software engineering constructs embodying the ASCENS concepts and SCEL abstractions, we have created a new component model called Dependable Emergent Ensembles of Components (DEECo).[3]

Typically, component models allow individual agents to proactively communicate via procedure call or message exchange. However, this leads to sustained interdependence among components, which in turn prevents true autonomy. To overcome this, a component in DEECo is described by its knowledge (which can be perceived as a local database for a component) and its processes, which rely only on data derived from knowledge of their own component (as opposed to being allowed to directly communicate with other components) (see Figure 2). The obliviousness of the components to each other makes each one inherently autonomous.

Of course, components need to exchange data to collaborate. This is addressed by ensembles. An ensemble prescribes how a set of components may exchange data. The ensemble defines both under which circumstances data exchange should be performed, and the actual data exchange. The circumstances are

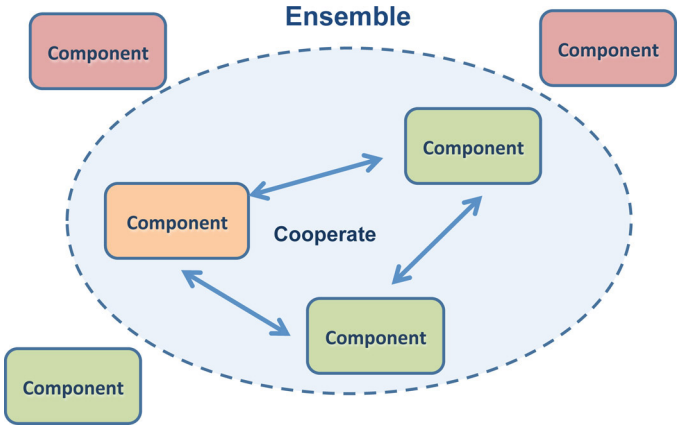

Figure 1. *Autonomous components form groups (ensembles) in order to cooperate.*

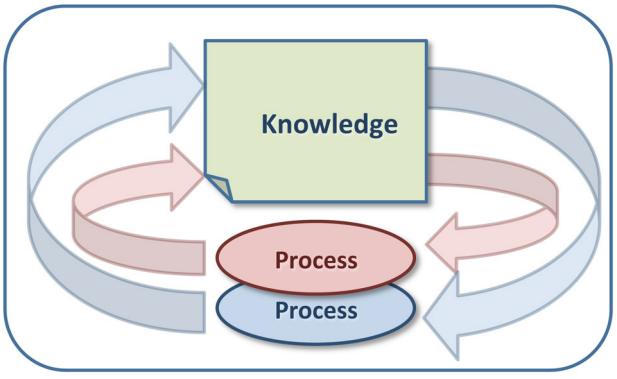

Figure 2. *Processes execute based on their component's local knowledge and possibly manipulate its contents as a result.*

expressed by a logical predicate (the membership predicate) over the states of components which may potentially become members of the ensemble. The actual data exchange (the mapping function) defines allowed knowledge exchanges between particular components within the ensemble (see Figure 3). Interfaces mask data structures and filter out irrelevant parts of component knowledge, allowing a component to have both shared and private knowledge, and ensuring data relevance.

Continued on next page

We elaborate on basic DEECo concepts using a simple example: autonomous robots whose objective is to drive safely through an intersection (see Figure 4). The components must give priority to other robots according to usual traffic rules. Each robot is modelled as a component (see Figure 5) with knowledge. The knowledge contains properties such as remaining battery charge, path to travel, current position, and belief about the other

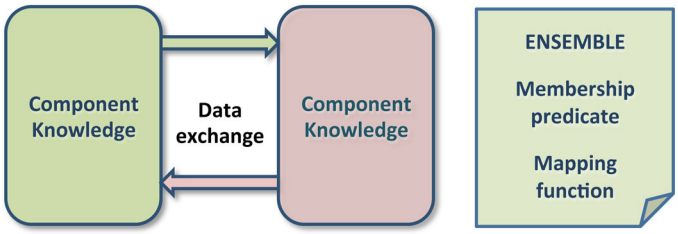

Figure 3. *Once the membership predicate holds, an ensemble is created and data exchange is performed according to the mapping function.*

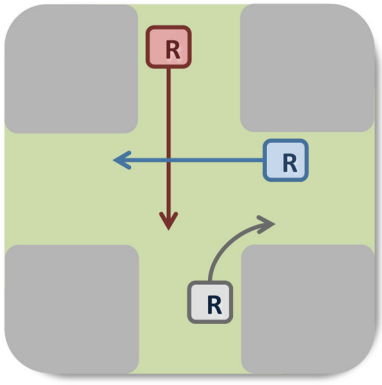

Figure 4. *Three autonomous robots (R) moving through a crossing, exchanging their positions and intentions in order to avoid collision.*

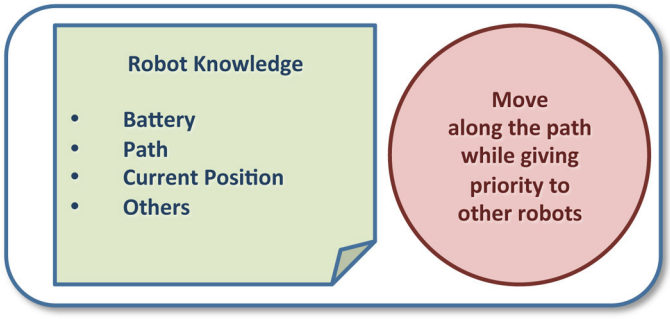

Figure 5. *A robot component in our example consists of four fields (knowledge constructs) and one process: move along the path while giving priority to other robots.*

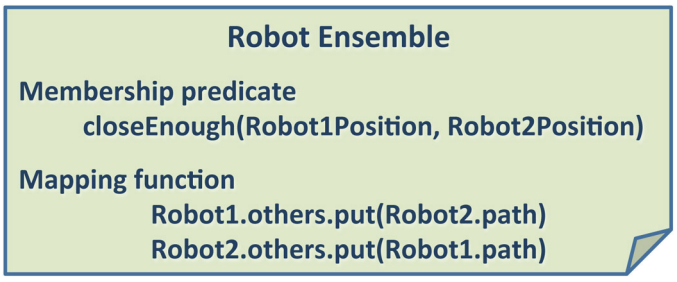

Figure 6. *The robot ensemble prescription in our example specifies that when two robots get close enough (membership predicate), the member's 'path' knowledge element is exchanged between the robots (mapping function).*

robots in the vicinity. The knowledge also has one process, which is responsible for updating the robot's current position according to data available from the robot's knowledge. Knowledge of the robots' intended paths is exchanged when two robots get close enough to each other. This is prescribed by an ensemble (see Figure 6).

The actual data exchange among robots (as prescribed by the ensemble) is driven by distributed component runtime, which interprets ensemble prescriptions and acts as the container for components. By separating the local operation from the data exchange, the DEECo component model allows for intuitive modelling of autonomous components and for efficient management of a component's knowledge of its operating environment, which is important in highly distributed open-ended systems with heterogeneous and ad hoc communication infrastructures.

The explicit notion of a component's knowledge also provides a basis for the component's self-awareness by including self-reflective information in its knowledge, and by separating the means of collecting this information. A good example is the inclusion of performance indicators, which are collected over time by the component runtime environment, and drive component adaptation subject to a set of rules.[4,5]

In the future we intend to develop a systematic approach to performance-based self-awareness using the DEECo component model. We will also work on raising the DEECo concepts to system-level requirements engineering, which will allow efficient handling of autonomy during the entire software development life cycle.

Continued on next page

Author Information

Tomáš Bureš, Ilias Gerostathopoulos, Petr Hnětynka, Jaroslav Keznikl, Michal Kit, and František Plášil
Department of Distributed and Dependable Systems
Faculty of Mathematics and Physics
Charles University of Prague
Prague, Czech Republic

References

1. N. Serbedzija, S. Reiter, M. Ahrens, J. Velasco, C. Pinciroli, N. Hoch, and B. Werther, *Requirement specification and scenario description*, **ASCENS**, 2011. http://www.ascens-ist.eu/deliverables/
2. R. De Nicola, G. Ferrari, M. Loreti, and R. Pugliese, *Languages primitives for coordination, resource negotiation and task description*, **ASCENS**, 2011. http://rap.dsi.unifi.it/scel/
3. J. Keznikl, T. Bureš, F. Plášil, and M. Kit, *Towards dependable emergent ensembles of components: the DEECo component model*, **Joint 10th Working IEEE/IFIP Conf. Software Architecture/6th Eur. Conf. Software Architecture (WICSA/ECSA)**, 2012.
4. L. Bulej, T. Bureš, V. Horký, J. Keznikl, and P. Tůma, *Performance awareness in component systems: vision paper*, **IEEE Conf. Comput. Software Appl. (COMPSAC)**, 2012.
5. L. Bulej, T. Bureš, J. Keznikl, A. Koubková, A. Podzimek, and P. Tůma, *Capturing performance assumptions using stochastic performance logic*, **3rd Joint ACM/SPEC Int'l Conf. Perform. Eng. (ICPE)**, pp. 311–322, 2012.