Department of Distributed and Dependable Systems
Technical Report no. D3S-TR-2013-01
February 18, 2013

DEECo Computational Model – I

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

Abstract: This report provides the formalization of DEECo concepts and operational semantic of DEECo systems. Further, it establishes the refinement relation and shows the centralized tuple space semantics (as featured by local and Apache River-based knowledge providers in jDEECo).

Version: March 2nd, 2013

This work was partially supported by the Grant Agency of the Czech Republic project P202/11/0312.
This work was partially supported by the EU project ASCENS 257414.

Contents

1 Introduction

This report formalizes the DEECo component model. It provides definition of core DEECo concepts (in Section 2) and gives the operational semantic (in Section 3). The semantics is defined by constructing an automaton of a DEECo system composed of components and ensembles. Further the report associates time and timing constraints with transitions in the DEECo system automaton (in Section 4). This way, it specializes the model by introducing periodic and triggered processes as well as periodic and triggered ensembles.

The DEECo operational semantics is intentionally defined as relatively general. This is to allow for refinement by different implementation – e.g. the using a centralized tuple space (exemplified in Section 5).

Section 6 gives the conclusion and outlines the specializations still to be covered. In particular, this document assumes relatively simple model of binary ensembles with no mutual exclusion of ensembles.

2 Definition of core DEECo concepts

This section provides formal definitions of core DEECo concepts. For their intuitive meaning and examples of their use, we refer the reader to [1][2].

Component is a tuple $C = (K_C, P_C, I_C)$, where K_C is a knowledge of the component, P_C is a set of processes and I_C is the initial knowledge valuation.

Knowledge K_C is a set of all potential knowledge fields of the component. Note that knowledge K_C comprises all potential and future knowledge fields of component C.

Knowledge valuation V_C is a partial function $V_C: K_C \rightharpoonup D$, where D is the domain of knowledge field values. We denote $\mathbb{V}_C = 2^{K_C \rightharpoonup D}$ as the set of all valuations over knowledge K_C of component C.

Knowledge valuation changeset U_C is a partial function $U_C: K_C \rightharpoonup (D \cup \{\textbf{undef}\})$, where D is the domain of knowledge field values, and **undef** represents a special value which signifies that a knowledge field valuation should be removed from component's knowledge valuation. We denote $\mathbb{U}_C = 2^{K_C \rightharpoonup (D \cup \{\textbf{undef}\})}$ as the set of all knowledge valuation updates for a component C.

We further define the knowledge valuation update operator $\leftarrow$ with the following semantics: Let $V_C \in \mathbb{V}_C$ be a knowledge valuation, let $U_C \in \mathbb{U}_C$ be a knowledge valuation changeset, then $(k, v) \in V_C \leftarrow U_C$ iff one of the following holds

- $(k, v) \in U_C \;\&\; v \neq \textbf{undef}$; or
- $(k, v) \in V_C \;\&\; \neg\exists u \in D \cup \{\textbf{undef}\}: (k, u) \in U_C$.

Initial knowledge valuation $I_C \in \mathbb{V}_C$ is the valuation of the components knowledge at the time of start of the component.

Process of a component C is a function $p: \mathbb{V}_C \times \Omega_p \to \mathbb{U}_C$, where Ω_p denotes probabilistic sample space of the process p.

The process p is thus a function that computes a new valuation of some knowledge fields of the component and preserves the knowledge valuation of the remaining fields. Ω_p is used here to capture non-determinism stemming from using systems services and from reading data from sensors.

The role of Ω_p, can be illustrated by assuming a process that internally queries a sensor to measure the distance between a robot and a wall. The observations are read by the process and used in process computation. The observations may differ each time the process executes – as a result the process may return different valuation even with the same input knowledge valuation. To capture this non-determining when the process is modeled as a function, we extract the non-determinism as a random observation (from domain of Ω_p), which taken every time the process executes and given to the process function as a parameter.

Ensemble definition is a tuple $E = (B_E, M_E)$, where

$B_E \subseteq \mathbb{V}_{C_i} \times \mathbb{V}_{C_j}$ denotes the membership predicate over valuations of two components;

$M_E: \mathbb{V}_{C_i} \times \mathbb{V}_{C_j} \to \mathbb{U}_{C_i} \times \mathbb{U}_{C_j}$ denotes the mapping function, which computes the update of the components' knowledges as the effect of the knowledge exchange implied by the ensemble.

System is a tuple $S = (\mathbb{C}, \mathbb{E})$, where $\mathbb{C}$ denotes the set of all components and $\mathbb{E}$ denotes the set of all ensemble definitions.

3 Semantics of DEECo concepts

In this section, we define the semantics of a DEECo system $S = (\mathbb{C}, \mathbb{E})$ by constructing a transition system. We construct the transition system as a Cartesian product of a number of automata (described below). Each automaton in the product is non-deterministic automaton where each transition is associated with an action $a \in \mathbb{A}$.

3.1 Components

Each component $C = (K_C, P_C, I_C) \in \mathbb{C}$ is associated with current valuation V_C of its knowledge K_C and with current belief of valuation of knowledge of other components – we denote $V_C^{C_i}$ the component's C belief of valuation of the knowledge of C_i. For the sake of easier construction of the semantics, we define V_C^C as V_C. The initial valuation of V_C equals to I_C and the initial valuation of $V_C^{C_i} = \emptyset$ for each C_i.

Each component $C = (K_C, P_C, I_C)$ is further associated with a set of queues $\{Q_C^{C_i} | \forall\, C_i \in \mathbb{C}: C_i \neq C\}$. Each queue $Q_C^{C_i}$ in set the represents the updates of C_i's knowledge valuation being sent to C. The queue $Q_C^{C_i}$ itself is a structure over $\mathbb{V}_{C_i}$ having operations enqueue (denoted as $Q_C^{C_i}.enqueue(item)$) and dequeue (denoted as $Q_C^{C_i}.dequeue$). The operations have the classical semantics – i.e. enqueue puts an item to the back of the queue while dequeue removes the item from the head of the queue.

We associate each queue $Q_C^{C_i}$ of the component C with an automaton $A(Q_C^{C_i})$ (see Figure 1). The automaton has one state and two loops representing the two actions it may perform:

- $Q_C^{C_i}.enqueue(V_{C_i})$ – i.e. inserting the current valuation of component's C_i knowledge into the queue.
- $V_C^{C_i} \coloneqq Q_C^{C_i}.dequeue$ – i. e. removing the C_i's knowledge valuation from the queue and assigning it as the current C's belief of the C_i's knowledge valuation.

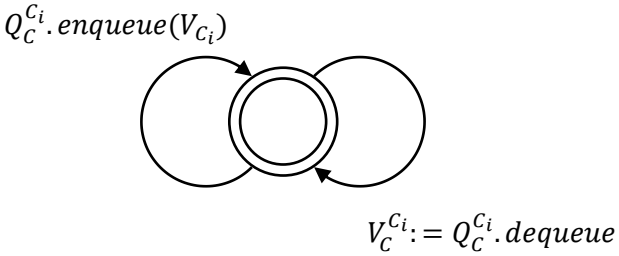


Figure 1: Knowledge valuations queue automaton

We associate each process $p \in P_C$ of a component C with an automaton $A(p)$ – depicted in Figure 2 . The initial state of the automaton is the *idle* state. The automaton associates the following actions with its transitions:

- $idle \rightarrow running : acc \coloneqq V_C$, where V_C is the input of the process, acc is a variable local to the automaton and $\coloneqq$ is the assignment operator (i.e. the process remembers the current valuation V_C as acc).
- $running \rightarrow idle : V_C \leftarrow p(acc, r)$, where $r \in \Omega$ is a random event representing the non-determinism and $\leftarrow$ denotes the knowledge valuation update as defined in Section 2.

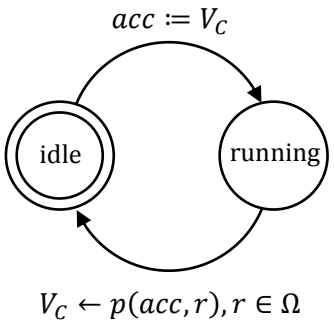


Figure 2: Component process automaton

The overall automaton $A(C)$ for a component C is the Cartesian product

$$A(C) = \prod_{C_i \in \mathbb{C}} A(Q_C^{C_i}) \times \prod_{p \in P_C} A(p)$$

3.2 Ensembles

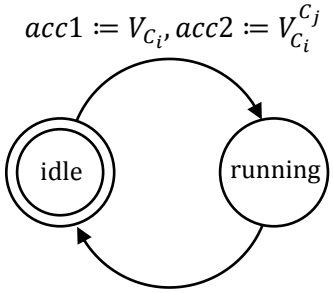
We associate each ensemble definition $E = (B_E, M_E) \in \mathbb{E}$ with a set of automata $\left\{A\left(E^{(C_i,C_j)}\right) \middle| \forall\, C_i, C_j \in \mathbb{C}\right\}$. Each automaton $A\left(E^{(C_i,C_j)}\right)$ represents by a Cartesian product of two automata, which are:

1. The automaton of coordinator's side $A(E_{co}^{(C_i,C_j)})$
2. The automaton of member's side $A(E_{mem}^{(C_i,C_j)})$

Similarly to the process automaton, both automatons $A(E_{co}^{(C_i,C_j)})$ and $A(E_{mem}^{(C_i,C_j)})$ have two states – idle and running. Idle state is the initial state.

In the automaton of the coordinator's side $(E_{co}^{(C_i,C_j)})$, we associate the following actions with its transitions (see Figure 3):

- $idle \rightarrow running : acc1 := V_{C_i}, acc2 := V_{C_i}^{C_j}$, where $acc1$, and $acc2$ are variables local to the automaton, and $:=$ is the assignment operator.
- $running \rightarrow idle : V_{C_i} \leftarrow if\ B_E\left(V_{C_i}, V_{C_i}^{C_j}\right) then\ proj_1(M_E(acc1, acc2))\ else\ \emptyset$, where $proj_1$ denotes the projection that for a tuple returns the first item (i.e. $proj_1\big((x_1, x_2)\big) = x_1$), and $\leftarrow$ denotes the knowledge valuation update as defined in Section 2, and $if \ldots then \ldots else \ldots$ is used in the traditional functional programming sense as a statement that returns the value of the $then$ branch if the if condition holds, and the value of the $else$ branch otherwise.



$$acc1 := V_{C_i}, acc2 := V_{C_i}^{C_j}$$

$$V_{C_i} \leftarrow if\ B_E\left(V_{C_i}, V_{C_i}^{C_j}\right) then\ proj_1\big(M_E(acc1, acc2)\big)\ else\ \emptyset$$

Figure 3: Coordinator's side of an ensemble automaton

Similarly, in the automaton of the member's side $A(E_{mem}^{(C_i,C_j)})$, we associate the following actions with its transitions (see Figure 4):

- $idle \rightarrow running : acc1 := V_{C_j}^{C_i}, acc2 = V_{C_j}$.
- $running \rightarrow idle : V_{C_j} \leftarrow if\ B_E\left(V_{C_j}^{C_i}, V_{C_j}\right) then\ proj_2\big(M_E(acc1, acc2)\big)\ else\ \emptyset$, where $proj_2$ denotes the projection that for a tuple returns the first item (i.e. $proj_2\big((x_1, x_2)\big) = x_2$).

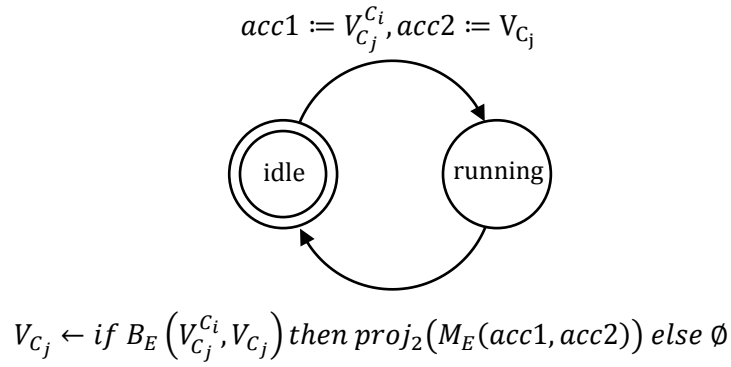


Figure 4: Member's side of an ensemble automaton

The automaton $A\left(E^{(C_i,C_j)}\right)$ thus corresponds to a potential ensemble (based on the ensemble definition E) in which C_i takes the role of the coordinator and C_j takes the role of the member. In this respect, $A\left(E^{(C_i,C_j)}\right)$ has the following definition:

$$A\left(E^{(C_i,C_j)}\right) = A(E_{co}^{(C_i,C_j)}) \times A(E_{mem}^{(C_i,C_j)})$$

The overall automaton $A(E)$ for an ensemble definition E is a Cartesian product

$$A(E) = \prod_{\forall C_i,C_j \in \mathbb{C}} A\left(E^{(C_i,C_j)}\right)$$

3.3 System

The transition system for a DEECo system $S = (\mathbb{C}, \mathbb{E})$ is given by the Cartesian product

$$A(S) = \prod_{\forall C \in \mathbb{C}} A(C) \times \prod_{\forall E \in \mathbb{E}} A(E)$$

4 Timing aspects

To introduce the notion of real-time operation to a DEECo system $S = (\mathbb{C}, \mathbb{E})$ we associate a timestamp with each action in an execution trace generated by transitions in automaton $A(S)$.

Formally, we define the real-time execution trace T of automaton $A(S)$ as an infinite sequence of action-timestamp pairs $T = (a_1, t_1), (a_2, t_2), (a_3, t_3), \ldots$, where each $a_i \in \mathbb{A}$ and for each $t_i \in Time$ it holds that $t_i \leq t_{i+1}$. (In this context, we use $Time$ to denote a set of timestamps.)

We denote $\mathbb{T}(S)$ as set of all real-time execution traces of $A(S)$.

Further, to allow for reasoning about the valuation of the knowledge at a particular time instant, we introduce the function $V_T^C: Time \rightarrow \mathbb{V}_C$. For a given real-time execution trace $T = (a_1, t_1), (a_2, t_2), (a_3, t_3), \ldots$ and component $C = (K_C, P_C, I_C)$ the function is defined as follows:

- $t < t_1$: $V_T^C(t) = I_C$;
- otherwise: $V_T^C(t)$ equals to V_C after all actions $a_1, \ldots, a_m$, where $m = \max(i | t_i \leq t)$.

4.1 Periodic processes

Having associated the time with execution of $A(S)$, we can now define the semantics of DEECo's periodic processes. A periodic process is (in addition to its function) associated with a period R. Further, its execution may be delayed in the execution trace by a fixed offset $o \geq 0$.

Thus, a component process p in a system S is periodic with period R, if for each $T \in \mathbb{T}(S)$ it holds that there exists $o \geq 0$ such that for all $i \in \mathbb{N}^0$:

- There is exactly one $(a_s, t_s) \in T$, such that a_s corresponds to $idle \rightarrow running$ transition of $A(p)$ and $o + Ri \leq t_s \leq o + R(i+1)$;
- there is exactly one $(a_f, t_f) \in T$, such that a_f corresponds to $running \rightarrow idle$ transition of $A(p)$ and $o + Ri \leq t_f \leq o + R(i+1)$;
- $s < f$.

4.2 Triggered processes

A triggered process p of component C is associated with a set of knowledge fields $G_p \subseteq K_C$ upon whose change it is triggered. Further, the triggered process is associated with a relative deadline D, which serves as the upper time bound between the knowledge change and completion of the corresponding process iteration.

Formally, a component process p in a system S is triggered, if for each $T \in \mathbb{T}(S)$ and for each $(a_h, t_h) \in T$ such that a_h corresponds to an action that changes the value (i.e. assigns a new different value) of some $k \in G_p$, there exists $(a_s, t_s) \in T$ and $(a_f, t_f) \in T$ such that $h < s < f$ and $t_f - t_h \leq D$, where a_s corresponds to $idle \rightarrow running$ transition of $A(p)$, and a_f corresponds to $running \rightarrow idle$ transition of $A(p)$.

4.3 Periodic ensembles

A periodic ensemble is a concept similar to a periodic process. Essentially it means that the membership predicate and mapping function are executed periodically – however independently on the side of each component involved in the ensemble.

An ensemble definition E in a system S defines periodic ensembles with period R, if for each $T \in \mathbb{T}(S)$ and each $A\left(E^{(C_{co}, C_{mem})}\right) \in A(E)$ exist offsets $0 \leq o_{co} < R$ and $0 \leq o_{mem} < R$ such that for all $i \in \mathbb{N}^0$:

- There is exactly one $\left(a_{s_{co}}, t_{s_{co}}\right) \in T$, such that $a_{s_{co}}$ corresponds to $idle \rightarrow running$ transition of $A\left(E_{co}^{(C_{co}, C_{mem})}\right)$ and $o_{co} + Ri \leq t_{s_{co}} \leq o_{co} + R(i+1)$;
- there is exactly one $\left(a_{f_{co}}, t_{f_{co}}\right) \in T$, such that $a_{f_{co}}$ corresponds to $running \rightarrow idle$ transition of $A\left(E_{co}^{(C_{co}, C_{mem})}\right)$ and $o_{co} + Ri \leq t_{f_{co}} \leq o_{co} + R(i+1)$;
- $s_{co} < f_{co}$;
- there is exactly one $\left(a_{s_{mem}}, t_{s_{mem}}\right) \in T$, such that $a_{s_{mem}}$ corresponds to $idle \rightarrow running$ transition of $A\left(E_{mem}^{(C_{co}, C_{mem})}\right)$ and $o_{mem} + Ri \leq t_{s_{mem}} \leq o_{mem} + R(i+1)$;
- there is exactly one $\left(a_{f_{mem}}, t_{f_{mem}}\right) \in T$, such that $a_{f_{mem}}$ corresponds to $running \rightarrow idle$ transition of $A\left(E_{mem}^{(C_{co}, C_{mem})}\right)$ and $o_{mem} + Ri \leq t_{f_{mem}} \leq o_{mem} + R(i+1)$;
- $s_{mem} < f_{mem}$.

4.4 Triggered ensembles

Similarly to triggered processes a triggered ensemble executes whenever any knowledge (or belief) it triggers on changes. We thus associate the definition of a triggered ensemble $E = (B_E, M_E)$, where $B_E \subseteq \mathbb{V}_{C_{co}} \times \mathbb{V}_{C_{mem}}$ with a set of knowledge fields $G_{co} \subseteq K_{C_{co}}$ and $G_{mem} \subseteq K_{C_{mem}}$ upon whose change it is triggered. Further, we associate the ensemble with a relative deadline D, which serves as the upper time bound between the knowledge change and of completion of the corresponding ensemble knowledge exchange.

Formally, an ensemble definition E in a system S defines triggered ensembles, if for each $T \in \mathbb{T}(S)$ and each $A\left(E^{(C_{co},C_{mem})}\right) \in A(E)$ the following two statements hold:

- For each $\left(a_{h_{co}}, t_{h_{co}}\right) \in T$ such that $a_{h_{co}}$ corresponds to an action that changes the value (i.e. assigns a new different value) of some $k_{co} \in G_{co}$ in $V_{C_{co}}$ or the value of some $k_{mem} \in G_{mem}$ in $V_{C_{co}}^{C_{mem}}$, there exists $\left(a_{s_{co}}, t_{s_{co}}\right) \in T$ and $\left(a_{f_{co}}, t_{f_{co}}\right) \in T$ such that $h_{co} < s_{co} < f_{co}$ and $t_{f_{co}} - t_{h_{co}} < D$, where $a_{s_{co}}$ corresponds to $idle \rightarrow running$ transition of $A\left(E_{co}^{(C_{co},C_{mem})}\right)$, and $a_{f_{co}}$ corresponds to $running \rightarrow idle$ transition of $A\left(E_{co}^{(C_{co},C_{mem})}\right)$;
- for each $\left(a_{h_{mem}}, t_{h_{mem}}\right) \in T$ such that $a_{h_{mem}}$ corresponds to an action that changes the value (i.e. assigns a new different value) of some $k_{mem} \in G_{mem}$ in $V_{C_{mem}}^{C_{co}}$ or the value of some $k_{mem} \in G_{mem}$ in $V_{C_{mem}}$, there exists $\left(a_{s_{mem}}, t_{s_{mem}}\right) \in T$ and $\left(a_{f_{mem}}, t_{f_{mem}}\right) \in T$ such that $h_{mem} < s_{mem} < f_{mem}$ and $t_{f_{mem}} - t_{h_{mem}} < D$, where $a_{s_{mem}}$ corresponds to $idle \rightarrow running$ transition of $A\left(E_{mem}^{(C_{co},C_{mem})}\right)$, and $a_{f_{mem}}$ corresponds to $running \rightarrow idle$ transition of $A\left(E_{mem}^{(C_{co},C_{mem})}\right)$.

4.5 Belief propagation

The DEECo operational semantics accounts for distributed execution of components by featuring communication queues $Q_C^{C_i}$, which model the latency connected with belief propagation. This latency is essentially contributed by three factors: (i) time between knowledge valuation change and handing over the changed valuation the network stack, (ii) network latency, (iii) time to retrieve the knowledge valuation from the network stack and updating the belief accordingly.

We assume that under normal operation the belief propagation latency is bound by assuming time bounds for all the three contributing factors (i)-(iii). From the perspective of the operational semantics, factor (i) is connected with the *enqueue* transition, factors (ii) and (iii) are jointly accounted for in the *dequeue* transition.

5 Refinement of the semantics

The operational semantics given in this report defines all possible traces of a system S. It is intentionally defined as relatively general so as it can be implemented using variety of approaches (e.g. using a centralized tuple space, using an agent-based middleware, etc.). Each of such implementation should however refine the semantics only in such a way that it does not generate trace with a knowledge valuation not obtainable by the by the semantics featured in this report.

Formally, we say that a semantics $\mathcal{A}$ refines $\mathcal{B}$ (denoted as $\mathcal{A} \preccurlyeq \mathcal{B}$) if it holds that for any system $S = (\mathbb{C}, \mathbb{E})$ and each real-time execution trace $T_{\mathcal{A}} \in \mathbb{T}_{\mathcal{A}}(S)$ featured by the semantics $\mathcal{A}$, there exists a trace $T_{\mathcal{B}} \in \mathbb{T}_{\mathcal{B}}(S)$ featured by semantics $\mathcal{B}$ such that for each $C \in \mathbb{C}$ it holds that $V^C_{T_{\mathcal{A}}} = V^C_{T_{\mathcal{B}}}$.

Thus, denoting the basic semantics described in Section 1 and 4 as $\mathcal{D}$, we require every DEECo implementation to feature some semantics $\mathcal{A}$, for which it holds that $\mathcal{A} \preccurlyeq \mathcal{D}$.

5.1 Centralized tuple-space semantics

To illustrate the refinement of the semantics, we describe the operational semantics currently featured by the local knowledge provider and the Apache River-based knowledge provider found in jDEECo. Both the knowledge providers feature the same semantics (denoted here as $\mathcal{T}$), which is based on having a centralized tuple space with the possibility of atomic operations (either by employing locks or by employing optimistic transactions).

The semantics $\mathcal{T}$ specializes $\mathcal{D}$ in the following way:

1. For each pair of components $C, C_i \in \mathbb{C}$ the belief $V_C^{C_i}$ always equals the current valuation V_{C_i}.
2. The ensemble evaluation on a coordinator component and a member component is performed simultaneously.

To prove that the refined semantics (denoted as $\mathcal{T}$) refines $\mathcal{D}$ (i.e. $\mathcal{T} \preccurlyeq \mathcal{D}$) we show how it can be derived by only constraining traces produced by $\mathcal{D}$.

In particular, point (1) can be modeled as allowing only traces where for each component $C_i \in \mathbb{C}$ all modifications of V_{C_i} are immediately followed by actions $Q_C^{C_i}.enqueue(V_{C_i})$ and $V_C^{C_i} := Q_C^{C_i}.dequeue$ performed for all $C \in \mathbb{C}$.

Point (2) can be modeled by allowing only traces T such that any transition of the automaton $A\left(E^{(C_i,C_j)}\right)$, for any $E \in \mathbb{E}$, must appear as a part of some direct sub-sequence $(a_{s_{co}}, t_{s_{co}})(a_{f_{co}}, t_{f_{co}})(a_{s_{mem}}, t_{s_{mem}})(a_{f_{mem}}, t_{f_{mem}})$ of T. (The notation is used with the same meaning as in Section 4.3.)

6 Conclusion

This report defines basic operational semantics $\mathcal{D}$ intended for DEECo-based implementations. It is intentionally relatively general so as to allow for different implementations. For the sake of simplicity, the semantics omits some of the more advanced concepts. These are in particular (i) ensembles with 1:n mapping function; and (ii) mutual exclusion over ensembles with a potential cost function, which would allow expressing optimization problems in terms of ensemble membership. These advanced concepts are to be introduced later once practical experiments will provide better understanding of what should be their proper semantics that simultaneously (a) would be suitable for development of distributed autonomic systems, (b) would be easily tractable by developer, and (c) could be efficiently realized on existing communication middleware.

References

[1] Keznikl, J. et al. 2012. Towards Dependable Emergent Ensembles of Components: The DEECo Component Model. WICSA–ECSA '12, 249–252.

[2] Bures, T. et al. 2012. Language Extensions for Implementation- Level Conformance Checking. ASCENS Deliverable 1.5.