Towards a comprehensive approach to spontaneous self-composition in pervasive ecosystems

Sara Montagna Mirko Viroli Danilo Pianini Jose Luis Fernandez-Marquez

Email: sara.montagna@unibo.it

ALMA MATER STUDIORUM—Università di Bologna a Cesena
University of Geneva, Switzerland

Undicesimo Workshop Nazionale “Dagli Oggetti agli Agenti” (WOA’12)

Milano-Bicocca, Italy, 17-19 September 2012

1. A comprehensive approach for pervasive ecosystems
2. The self-composition issue in pervasive service ecosystems
3. Gradient self-compositions
4. Towards simulation of gradient self-compositions

Outline

Pervasive service ecosystems [VPMS12]

SAPERE Vision

- Mobile devices, people, software services, data, events
  - Individuals
- Self-organisation enacted at the system level
- High degrees of
  - scale
  - openness
  - adaptivity
  - toleration of long-term evolution

Abstract Architecture

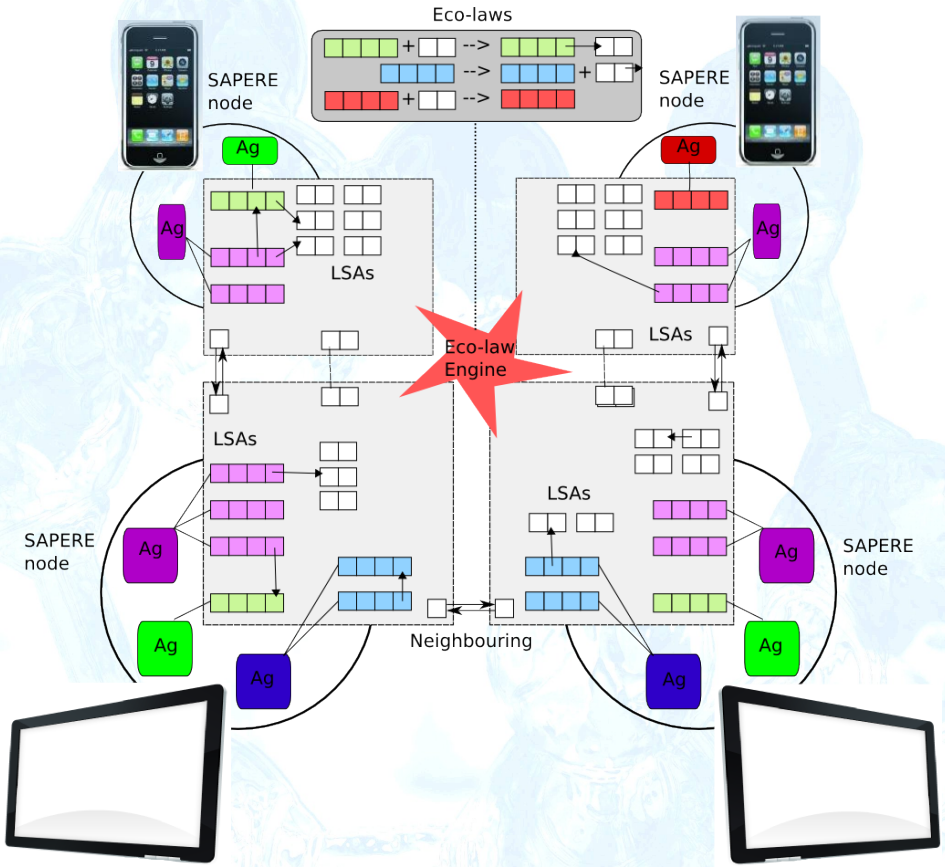

Figure : An architectural view of a pervasive ecosystem.

Live semantic annotations

Basic block of semantic chemistry

- A unified description for every entity
- A unique LSA-id plus a semantic description (SD)
- RDF-inspired set of multi-valued properties
- Contains everything is needed for describing the entity

Example: gradient source annotation

```
:id314 mid:#loc :loc117;   sos:type sos:source;
       sos:step "0";       sos:sourceid "341AB2"
       sos:aggr_prop       sos:sourceid;
       sos:r_diff "10";    sos:r_ctx "100"
```

Eco-Laws

Language of semantic chemistry

- Chemical rules over LSA templates
  `P+...+P --r--> Q+...+Q`
  - Constrained variables written `?V(`*`filter`*`)`
  - Check for presence “+”, absence “-” or unique existence “=”
- They can diffuse an LSA in the neighborhood
- They can aggregate LSAs like in chemical bonding

Example: source pump

```
?SOURCE sos:type sos:source; sos:aggr_prop ?P; sos:step ?T; sos:r_diff ?R
--?R-->
?SOURCE sos:step =(?T+1) + ?GRAD(?GRAD clones ?SOURCE)
sos:type -sos:source sos:diff sos:aggr; sos:dist "0"; sos:orientation "here"
```

Outline

Self-Composition

Key issue

- Patterns of behaviour emerge without any supervision
- Example: fully-spontaneous composition of services, possibly at multiple levels

Some self-composition issues

- Composition of services not explicitly designed to coordinate
- Composition of “compatible” services
- Creation of “meaningful” services
- Context awareness
- Multi-level composition

Self-composition in service ecosystems

Composition of services in literature

1. Service Composition in SOA – advanced semantic matching
2. Evolutionary techniques
3. Competition-based approaches

All the above, altogether

1. Choice of the services to compose
2. Pre-selection of “promising” compositions
3. Fine parameter tuning
4. Service evaluation metrics
5. Best services must be promoted

Outline

Paradigmatic Example: Crowd Steering

Goal and requirements

- Guide people towards POIs
  - POIs chosen with respect to people's interests
  - Avoiding obstacles (incl. crowds)
  - no supervision

Scenario

- A museum with a dense network of sensor nodes
  - Sensing of the presence of nearby visitors
  - Computation abilities
- Visitors own smartphone devices holding their preferences

Services available

- Gradient service
- Those provided by sensors (e.g., crowd detection service)

A Prototype Solution for Gradient Composition

Composition "composition recommender" agents computing all the available compositions

Contextualisation Gradients are contextualised

Feedback Users public their "satisfaction" once they used the service

Choice Users tend to prefer lower distance and higher satisfaction

Evaporation Satisfaction fades with time

Evolution Parameters tuning by agents using evolutionary techniques

Eco-Laws for Gradient

```
[PUMP]: An annotation of type source continuously injects the initial gradient annotation
?SOURCE sos:type sos:source; sos:aggr_prop ?P; sos:step ?T; sos:r_diff ?R; sos:r_ctx ?RC
--?R-->
?SOURCE sos:step =(?T+1) +
?GRAD(?GRAD clones ?SOURCE) sos:type -sos:source sos:diff sos:aggr;  sos:dist "0"; sos:orientation "here"

[DIFF] A gradient annotation is cloned in a neighbour, with distance increased and updated orientation
?GRAD sos:type sos:diff; sos:dist ?D; sos:r_diff ?R +
?NEIGH mid:type mid:#neigh; mid:remote ?L; mid:orientation ?O; mid:distance ?D2
--?R-->
?GRAD + ?NEIGH +
?GRAD1(?GRAD1 clones ?GRAD)  sos:type -sos:diff sos:ctx; sos:dist =(?D+?D2); sos:orient =?O; mid:#loc ?L

[CTX] A contextualising annotation is transformed back into an annotation to be diffused
?GRAD sos:type sos:ctx; sos:r_ctx ?RC --?RC-> ?GRAD sos:type sos:-ctx sos:diff;

[YOUNGEST]  Of two annotations the one with newest information is kept
?ANN1 sos:type sos:aggr; sos:aggr_prop ?P; ?P =[?C]; sos:step ?T1 +
?ANN2 sos:type sos:aggr; sos:aggr_prop ?P2; ?P2 =[?C]; sos:step ?T2(?T2<?T1)
--->
?ANN1
[SHORTEST]  Of two annotations the one with shortest distance from source is  kept
?ANN1 sos:type sos:aggr; sos:aggr_prop ?P; ?P =[?C]; sos:dist ?D1; sos:step ?T  +
?ANN2 sos:type sos:aggr; sos:aggr_prop ?P2; ?P2 =[?C]; sos:dist ?D2(?D2>=?D1); sos:step ?T
--->
?ANN1

[DECAY] An annotation decays
?GRA sos:type sos:diff; sos:r_dec ?RD --?RD-> 0
```

Eco-Laws for Gradient Composition

```
[COMPOSITION] The gradient source is composed with the crowd service
?SOURCE sos:type sos:source; scm:satisfaction ?S + ?CROWD scm:type crowd; crowd:level ?CL
--->
?SOURCE + ?CSOURCE(?CSOURCE clones ?SOURCE) scm:property sos:dist;
scm:parameters scm:crowd_op  ?CF;  scm:crowd_op  ?CF*?CL

[CONTEXTUALISATION] If sensors perceive crowd, the gradient distance is augmented
?GRAD sos:type sos ctx; sos:dist ?D;  scm:property sos:dist;
scm:parameters scm:crowd_op scm:crowd_factor;  scm:crowd_factor ?CF; scm:crowd_op  ?CF*?CL +
?CROWD scm:type crowd;  crowd:level ?CL
--->
?CROWD + ?GRAD  sos:type -sos:ctx sos:diff; sos:dist =(?D+?CF*?CL)

[FEEDBACK] Feedbacks are used to update the satisfaction values
?FEEDBACK scm:parameters scm:crowd_op; scm:feedback  scm:velocity; scm:velocity ?V +
?GRAD scm:satisfaction ?S; scm:parameters scm:crowd_op
--->
?GRAD scm:satisfaction =(?S+?V)

[EVAPORATION] The gradient satisfaction value gets decreased
?GRAD scm:satisfaction ?S; scm:factor_ev ?FE; scm:r_ev ?RE
--?RE->
?GRAD scm:satisfaction =(?FE*?S)

[DECAY] If the gradient satisfaction value becomes zero that composition is removed
?GRAD scm:satisfaction "0";
--->
```

Outline

Simulation as a proof of concepts

- Conducted using ALCHEMIST [PMV11]
- Early experiments on gradient composition with crowd level
- Different compositions with different crowd relevance
  - different composite gradients
- Satisfaction value measures the time to POI
- Users choose one gradient considering distance and satisfaction

Simulation Results I

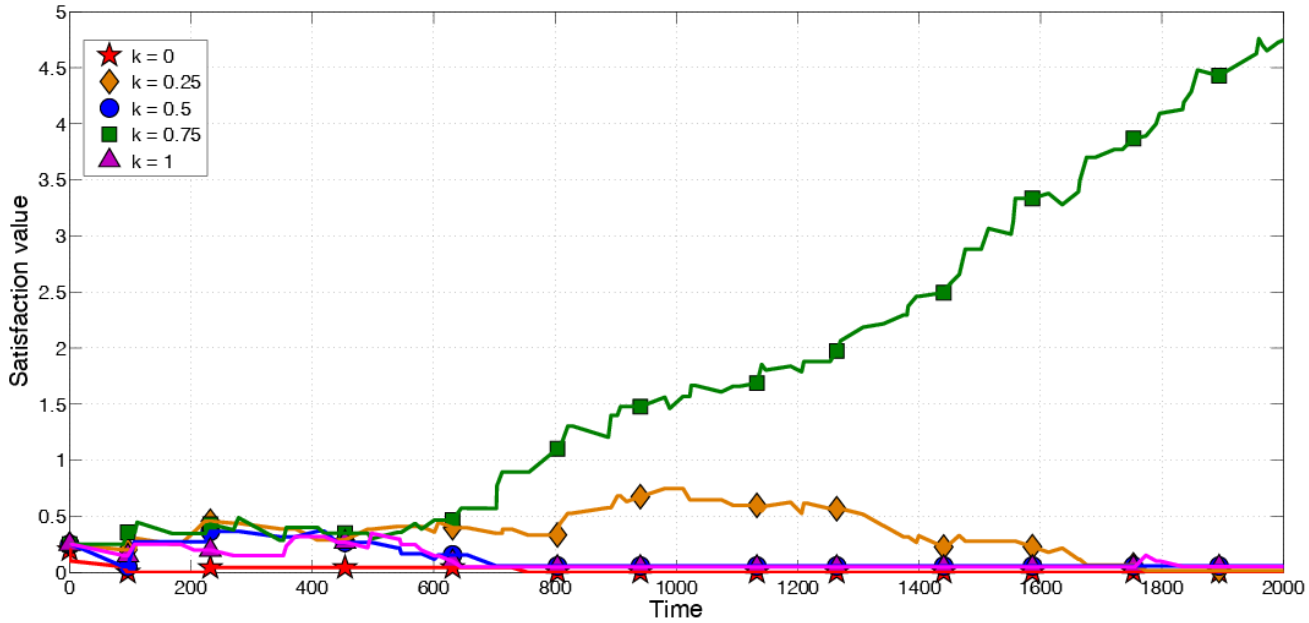

Figure : Satisfaction values for different compositions changing over time.

Simulation Results II

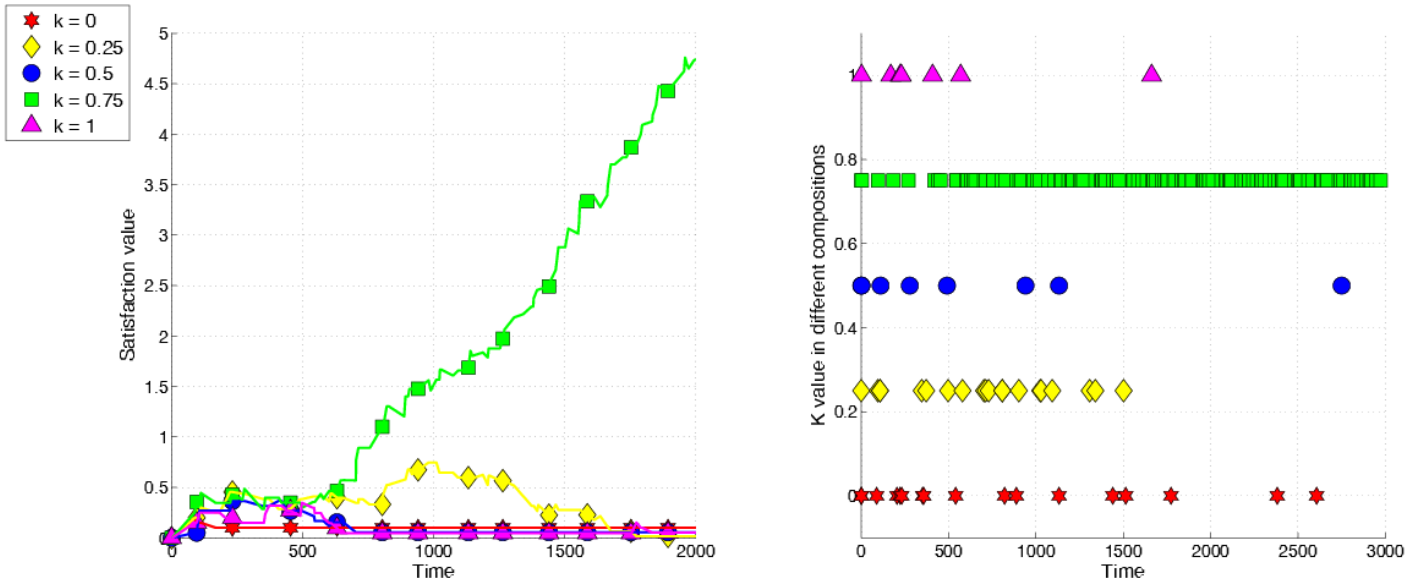

Figure : Satisfaction values for different compositions changing over time.

References I

Danilo Pianini, Sara Montagna, and Mirko Viroli.
A chemical inspired simulation framework for pervasive services ecosystems.
In *Proceedings of the Federated Conference on Computer Science and Information Systems*, pages 675–682. IEEE Computer Society Press, 2011.

Mirko Viroli, Danilo Pianini, Sara Montagna, and Graeme Stevenson.
Pervasive ecosystems: a coordination model based on semantic chemistry.
In *27th Annual ACM Symposium on Applied Computing (SAC 2012)*, pages 295–302. ACM, 2012.

Towards a comprehensive approach to spontaneous self-composition in pervasive ecosystems

Sara Montagna Mirko Viroli Danilo Pianini Jose Luis Fernandez-Marquez

Email: sara.montagna@unibo.it

ALMA MATER STUDIORUM—Università di Bologna a Cesena
University of Geneva, Switzerland

Undicesimo Workshop Nazionale “Dagli Oggetti agli Agenti” (WOA’12)

Milano-Bicocca, Italy, 17-19 September 2012