Ensemble-oriented programming of self-adaptive systems

Michele Loreti

Dipartimento di Statistica, Informatica, Applicazioni
Università degli Studi di Firenze

AWASS 2013, Lucca, June 24-28, 2013
Ensemble-oriented programming of self-adaptive systems: the E-Vehicle Case Study

Michele Loreti

Dipartimento di Statistica, Informatica, Applicazioni
Università degli Studi di Firenze

AWASS 2013, Lucca, June 24-28, 2013
Outline...
Motivations;
Motivations;

*E-Vehicle Case Study: a short overview*;
Motivations;

*E-Vehicle Case Study*: a short overview;

Basic ingredients of:
Outline...

- Motivations;
- *E-Vehicle Case Study*: a short overview;
- Basic ingredients of:
  - SCEL, a *Service Component Ensemble Language*;
Motivations;

E-Vehicle Case Study: a short overview;

Basic ingredients of:
- SCEL, a Service Component Ensemble Language;
- jRESP, a framework for executing SCEL applications in Java;
Motivations;

*E-Vehicle Case Study*: a short overview;

Basic ingredients of:
- SCEL, a *Service Component Ensemble Language*;
- jRESP, a framework for executing SCEL applications in Java;

*E-Vehicle Case Study* in SCEL and jRESP: a tentative *roadmap*.
Challenges of Ensembles Programming

Ensembles are software-intensive systems featuring massive numbers of components, complex interactions among components, and with other systems operating in open and non-deterministic environments dynamically adapting to new requirements, technologies and environmental conditions.

From the final report of: IST Coordinated Action InterLink [2007].

Challenges for software development for ensembles

- the dimension of the systems
- the need to adapt to changing environments and requirements
- the emergent behaviour resulting from complex interactions
- the uncertainty during design-time and run-time

Michele Loreti
Challenges of Ensembles Programming

Ensembles are software-intensive systems featuring

- massive numbers of components
- complex interactions among components, and with other systems
- operating in open and non-deterministic environments
- dynamically adapting to new requirements, technologies and environmental conditions

From the final report of: IST Coordinated Action InterLink [2007].
Challenges of Ensembles Programming

Ensembles are software-intensive systems featuring

- **massive numbers** of components
- **complex interactions** among components, and with other systems
- operating in **open and non-deterministic environments**
- **dynamically adapting** to new requirements, technologies and environmental conditions

From the final report of: IST Coordinated Action InterLink [2007].

Challenges for software development for ensembles
Challenges of Ensembles Programming

Ensembles are software-intensive systems featuring

- massive numbers of components
- complex interactions among components, and with other systems
- operating in open and non-deterministic environments
- dynamically adapting to new requirements, technologies and environmental conditions

From the final report of: IST Coordinated Action InterLink [2007].

Challenges for software development for ensembles

- the dimension of the systems
Challenges of Ensembles Programming

Ensembles are software-intensive systems featuring:

- **massive numbers** of components
- **complex interactions** among components, and with other systems
- operating in **open and non-deterministic environments**
- **dynamically adapting** to new requirements, technologies and environmental conditions

From the final report of: IST Coordinated Action InterLink [2007].

Challenges for software development for ensembles:

- the **dimension** of the systems
- the **need to adapt** to changing environments and requirements
Challenges of Ensembles Programming

Ensembles are software-intensive systems featuring:

- massive numbers of components
- complex interactions among components, and with other systems
- operating in open and non-deterministic environments
- dynamically adapting to new requirements, technologies and environmental conditions

From the final report of: IST Coordinated Action InterLink [2007].

Challenges for software development for ensembles:

- the dimension of the systems
- the need to adapt to changing environments and requirements
- the emergent behaviour resulting from complex interactions
Challenges of Ensembles Programming

Ensembles are software-intensive systems featuring

- massive numbers of components
- complex interactions among components, and with other systems
- operating in open and non-deterministic environments
- dynamically adapting to new requirements, technologies and environmental conditions

From the final report of: IST Coordinated Action InterLink [2007].

Challenges for software development for ensembles

- the dimension of the systems
- the need to adapt to changing environments and requirements
- the emergent behaviour resulting from complex interactions
- the uncertainty during design-time and run-time
Component Ensembles and Awareness

Service components (SCs) and service-component ensembles (SCEs) permit to dynamically structure independent, distributed entities that can cooperate, with different roles, in open and non-deterministic environments.

Awareness

Awareness of Service Components is achieved by equipping SCs with information about their own state enabling SCs to get information on their working environment allowing SCs to use this information for restructuring and adapting.

Awareness makes SCs adaptable, connectable and composeable.
Component Ensembles and Awareness

Components and Ensembles

Service components (SCs) and service-component ensembles (SCEs) permit to dynamically structure independent, distributed entities that can cooperate, with different roles, in open and non-deterministic environments.

Awareness

Awareness of Service Components is achieved by equipping SCs with information about their own state, enabling SCs to get information on their working environment, allowing SCs to use this information for restructuring and adapting. Awareness makes SCs adaptable, connectable and composeable.
Component Ensembles and Awareness

Components and Ensembles

Service components (SCs) and service-component ensembles (SCEs) permit to dynamically structure independent, distributed entities that can cooperate, with different roles, in open and non-deterministic environments.

Awareness

Awareness of Service Components is achieved by
Component Ensembles and Awareness

Components and Ensembles
Service components (SCs) and service-component ensembles (SCEs) permit to dynamically structure independent, distributed entities that can cooperate, with different roles, in open and non-deterministic environments.

Awareness
Awareness of Service Components is achieved by
- equipping SCs with information about their own state
Components and Ensembles

Service components (SCs) and service-component ensembles (SCEs) permit to dynamically structure independent, distributed entities that can cooperate, with different roles, in open and non-deterministic environments.

Awareness

Awareness of Service Components is achieved by

- equipping SCs with information about their own state
- enabling SCs to get information on their working environment
Component Ensembles and Awareness

Components and Ensembles

Service components (SCs) and service-component ensembles (SCEs) permit to dynamically structure independent, distributed entities that can cooperate, with different roles, in open and non-deterministic environments.

Awareness

Awareness of Service Components is achieved by

- equipping SCs with information about their own state
- enabling SCs to get information on their working environment
- allowing SCs to use this information for restructuring and adapting
**Components and Ensembles**

Service components (SCs) and service-component ensembles (SCEs) permit to dynamically structure independent, distributed entities that can cooperate, with different roles, in open and non-deterministic environments.

**Awareness**

Awareness of Service Components is achieved by

- equipping SCs with information about their own state
- enabling SCs to get information on their working environment
- allowing SCs to use this information for restructuring and adapting

Awareness makes SCs adaptable, connectable and composeable.
We consider a car pooling system where motorized mobility for users and privately owned vehicles have to be coordinated in order to satisfy user requirements and to guarantee an (possibly) optimal allocation of resources.

We will use SCEL to specify system behaviour while jRESP, its runtime environment, will be used to develop a prototype implementation in Java.
We consider a car pooling system where motorized mobility for users and privately owned vehicles have to be coordinated in order to satisfy user requirements and to guarantee an (possibly) optimal allocation of resources.
We consider a car pooling system where motorized mobility for users and privately owned vehicles have to be coordinated in order to satisfy user requirements and to guarantee an (possibly) optimal allocation of resources.

We will use SCEL to specify system behaviour while jRESP, its runtime environment, will be used to develop a prototype implementation in Java.
In the E-Vehicle Case Study the following entities cooperate:

- Users
- Vehicles
- Parking lots
- Charging stations

Michele Loreti
In the E-Vehicle Case Study the following *entities* cooperate:
In the E-Vehicle Case Study the following *entities* cooperate:
- users;
In the E-Vehicle Case Study the following *entities* cooperate:

- users;

**User:**

A user is an active unit of the system. Given its daily calendar of activities a user:

- plans its journey
- books parking lots/charging stations
In the E-Vehicle Case Study the following *entities* cooperate:

- **users**;

**User:**

A user is an active unit of the system. Given its daily calendar of activities a user:

- plans its journey
- books parking lots/charging stations
  - this task is performed in collaboration with the associated vehicle.
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;

**Vehicle:**

A vehicle can:

- move towards destinations,
- use the vehicle travel planner to plan the journey,
- book parking lots and charging stations.
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;

A vehicle also monitors:

- the current traffic and road feasibility,
- fuel consumption,
- and parking lot/charging station availability.
In the E-Vehicle Case Study the following entities cooperate:

- users;
- vehicles;
- parking lots;
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
- parking lots;

**Parking lots**

A parking lot is an infrastructure unit of the system:
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
- parking lots;

**Parking lots**

A parking lot is an infrastructure unit of the system:

- It can be booked (and unbooked) by vehicles.
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
- parking lots;

**Parking lots**

A parking lot is an infrastructure unit of the system:

- It can be booked (and unbooked) by vehicles.
- A single parking lot can be booked by a single vehicle at a time.
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
- parking lots;
- charging stations.
In the E-Vehicle Case Study the following entities cooperate:

- users;
- vehicles;
- parking lots;
- charging stations.

**Charging stations:**

A charging station is an infrastructure unit of the system:
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
- parking lots;
- charging stations.

**Charging stations:**

A charging station is an infrastructure unit of the system:

- It is used by electric vehicles in order to recharge their batteries.
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
- parking lots;
- charging stations.

**Charging stations:**

A charging station is an infrastructure unit of the system:

- It is used by electric vehicles in order to recharge their batteries.
- A single charging station can be booked by a single vehicle at a time.
In the E-Vehicle Case Study the following *entities* cooperate:

- users;
- vehicles;
- parking lots;
- charging stations.

**Charging stations:**

A charging station is an infrastructure unit of the system:

- It is used by electric vehicles in order to recharge their batteries.
- A single charging station can be booked by a single vehicle at a time.
  - Similar to the booking of parking lots, it is usually possible for vehicles to book a charging station in advance.
Main Success Scenario

**e-Mobility in SCEL (and jRESP)**

- **P1**
- **P2**
- **P3**
- **P4**

**POI**
- **POI1**
- **POI2**
- **POI3**
- **POI4**

**Calendar**

<table>
<thead>
<tr>
<th>Time</th>
<th>POI</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td>POI2</td>
</tr>
<tr>
<td>10:00</td>
<td>POI1</td>
</tr>
<tr>
<td>12:00</td>
<td>POI3</td>
</tr>
<tr>
<td>14:00</td>
<td>POI2</td>
</tr>
<tr>
<td>16:00</td>
<td>P1</td>
</tr>
</tbody>
</table>

**POIj = j-th point of interest**

**Pi = i-th parking lot**
Programming Ensembles...
SCEL is a specification/programming language equipped with...
SCEL is a specification/programming language equipped with

- **programming abstractions** necessary for
  - directly representing Knowledge, Behaviors and Aggregations according to specific Policies
  - naturally programming interaction, adaptation and self- and context-awareness
SCEL is a specification/programming language equipped with

- **programming abstractions** necessary for
  - directly representing Knowledge, Behaviors and Aggregations according to specific Policies
  - naturally programming interaction, adaptation and self- and context-awareness

- **linguistic primitives** with *solid semantic grounds*
  - To develop logics, tools and methodologies for *formal reasoning* on systems behavior
  - to establish *qualitative and quantitative properties* of both the individual components and the ensembles
The Service-Component Ensemble Language (SCEL) currently provides primitives and constructs for dealing with:

1. Knowledge: to describe how data, information and (local and global) knowledge is managed.
2. Behaviours: to describe how systems of components progress.
3. Aggregations: to describe how different entities are brought together to form components, systems, and ensembles.
4. Policies: to model and enforce the wanted evolutions of computations.
SCEL: programming abstractions

The Service-Component Ensemble Language (SCEL) currently provides primitives and constructs for dealing with:

1. **Knowledge**: to describe how data, information and (local and global) knowledge is managed
The Service-Component Ensemble Language (SCEL) currently provides primitives and constructs for dealing with:

1. **Knowledge**: to describe how data, information and (local and global) knowledge is managed
2. **Behaviours**: to describe how systems of components progress
The Service-Component Ensemble Language (SCEL) currently provides primitives and constructs for dealing with:

1. **Knowledge**: to describe how data, information and (local and global) knowledge is managed

2. **Behaviours**: to describe how systems of components progress

3. **Aggregations**: to describe how different entities are brought together to form *components, systems and ensembles*
The Service-Component Ensemble Language (SCEL) currently provides primitives and constructs for dealing with:

1. **Knowledge**: to describe how data, information and (local and global) knowledge is managed
2. **Behaviours**: to describe how systems of components progress
3. **Aggregations**: to describe how different entities are brought together to form *components, systems* and *ensembles*
4. **Policies**: to model and enforce the wanted evolutions of computations.
A SCEL component

Component

Knowledge $K$

Processes $\Pi$

Interface $I$

Policies $P$

This information is rendered as a set of attributes whose values can be derived from the knowledge.
Component *interface* provides information about the component itself. This information is rendered as a set of *attributes* whose values can be derived from the *knowledge*.
In order to guarantee the maximum degree of flexibility, ensembles are rendered in terms of predicate-based communication primitives that select the targets among those enjoying specific properties.
In order to guarantee the maximum degree of flexibility, ensembles are rendered in terms of predicate-based communication primitives that select the targets among those enjoying specific properties.
Parking Lots close to POIs as Ensembles

Calendar

9:00
10:00
12:00
14:00
16:00

P1
P2
P3

e-Mobility in SCEL (and jRESP)

ATTRIBUTES

● type: parking lot component
● position: position of the park
● ...

Michele Loreti
E-Vehicle Case Study (in SCEL)...

Parking Lots close to POIs as Ensembles

<table>
<thead>
<tr>
<th>Calendar</th>
<th>POI1</th>
<th>POI2</th>
<th>POI3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ATTRIBUTES

- **type**: parking lot component
- **position**: position of the park
- ...

Michele Loreti
E-Vehicle Case Study (in SCEL)...

Calendar

<table>
<thead>
<tr>
<th>Time</th>
<th>POI</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td>POI2</td>
</tr>
<tr>
<td>10:00</td>
<td>POI1</td>
</tr>
<tr>
<td>12:00</td>
<td></td>
</tr>
<tr>
<td>14:00</td>
<td>POI3</td>
</tr>
<tr>
<td>16:00</td>
<td></td>
</tr>
</tbody>
</table>

ENSEMBLE FOR POI2
Group of components with type parking lot and position at walking distance from POI2
Basic design principles...

1. **no centralized control**
2. **heavy use of recurrent patterns** to simply the development of specific knowledge
   - a single interface that contains basic methods to interact with knowledge
   - policies
     - based on the pattern *composite* (policies are structured as a stack)
   - ...
3. **use of open technologies** to support the integration with other tools/frameworks or with alternative implementations of SCEL
SCEL component

- SCEL Processes (Threads)
- Policies
- Knowledge
- Hardware/Virtual Machine
- Input devices/Sensors (GPS, Temperature, Battery level, CPU load...)
- Output devices/Actuators

Michele Loreti
E-Vehicle Case Study in SCEL
A possible roadmap...

1. Identify basic information elements (attributes) to use for coordinating the interactions among components;
2. Design component behaviours in terms of SCEL processes;
3. Develop a jRESP application for supporting/controlling activities of involved agents;
4. Present the obtained results in a simulation environment.
1 Identify basic information elements (attributes) to use for coordinating the interactions among components;
1. Identify basic information elements (attributes) to use for coordinating the interactions among components;

2. Design component behaviours in terms of SCEL processes;
Identify basic information elements (attributes) to use for coordinating the interactions among components;

Design component behaviours in terms of SCEL processes;

Develop a jRESP application for supporting/controlling activities of involved agents;
Identify basic information elements (attributes) to use for coordinating the interactions among components;

Design component behaviours in terms of SCEL processes;

Develop a jRESP application for supporting/controlling activities of involved agents;

Present the obtained results in a simulation environment.
Good work!