ContextWare

Context Awareness in Mobile Ad-Hoc Communication Scenarios

Deptartment of Computer Science
University of Linz, Austria
Alois FERSCHA
Volker CHRISTIAN
Wolfgang BEER
{ferscha,voc}@soft.uni-linz.ac.at
beer@ssw.uni-linz.ac.at

Corporate Technology
Software & Engineering
Siemens, Munich
Lothar BORRMANN
Lars MEHRMANN
Björn SCHIEMANN
{lothar.borrmann,
lars.mehrmann,
bjoern.schiemann}
@mchp.siemens.de

The SILICON Framework provides an abstract software layer for modeling of interactions in complex ad-hoc communication scenarios, among heterogenous objects (entities).

The framework increasingly speeds up the software design for interacting sensor and actuator scenarios. With a platform independent configuration set, it is possible to launch complete ad-hoc entity interaction scenarios. The configuration includes interpreted context rules, which control the control and data flow inside a context aware scenario. These context rules are written in a specific syntax and can also be deployed while scenario runtime.

Entities, within a context aware scenario, are handled in a fully distributed, independent way. They contain their own logic module, tranport module and a set of attributes in order to accomplish the requirements needed in peer-to-peer environments.

Dynamic entity and attribute loading or delivery, enables dynamic upgrade of object capabilities and extension of running context aware scenarios.

Framework Architecture

The figure gives an overview over the framework architecture and provides a logical separation in different modules. Along with the graphics, we provide an architecture description of the whole system and a detail view of the major backend modules.
The example XML documents provide insights into the principal communication interfaces involved. Follow the numbers to get an idea how a context sceanrio works.

Functionality Illustration

1) When an entity starts up, its container loads all attributes, specified in its inventory configuration file:

```
- <Inventory>
  - <AttributeSequence>
    - <Attribute Name="AlarmButton" Host="localhost" Class="uni.linz.context.attributes.singlebutton.SingleButton">
        <AttributeSpecific Red="255" Green="0" Blue="0" Label="Alarm" />
      </Attribute>
    - <Attribute Name="RFID" Host="localhost" Class="uni.linz.context.attributes.rfid.Rfid">
      - <AttributeSpecific FakeJNI="false" AutoStart="true">
          <JNIInterface Class="uni.linz.context.attributes.jni.rfid.inside.RfidReaderJNI" Port="1" />
        </AttributeSpecific>
      </Attribute>
    - <Attribute Name="Name" Host="localhost" Class="uni.linz.context.attributes.name.Name">
        <AttributeSpecific FirstName="Jonathen" LastName="Heart" />
      </Attribute>
      <Attribute Name="IsIn" Host="localhost" Class="uni.linz.context.attributes.isin.IsIn" />
    - <Attribute Name="Map" Host="localhost" Class="uni.linz.context.attributes.map.Map">
      - <AttributeSpecific>
          <Map File="Munic_Map.jpg" Width="220" Height="210" />
        </AttributeSpecific>
      </Attribute>
    </AttributeSequence>
  </Inventory>
```

2) The context container itselve uses a configuration file to check which transport layer should be used and which data format:

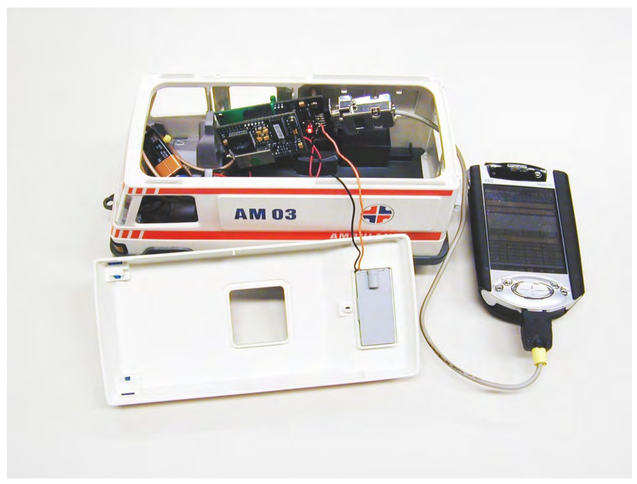
```
<?xml version="1.0" ?>
- <Container Title="LooxContainer" Width="240" Height="295">
    <Logging Type="http" Port="8080" LogStdOut="No" LogStdErr="Yes" />
  - <Transport Port="8079">
      <Encoder Class="uni.linz.context.framework.encode.CFXmlEventEncoder" />
      <Decoder Class="uni.linz.context.framework.decode.CFXmlEventDecoder" />
    </Transport>
  - <Entity Name="LooxContainer" GridWidth="1" GridHeight="1" ControlPanel="Off" Choice="On">
      <Logo File="loox.gif" XPos="Center" YPos="Center" XWide="100%" YWide="100%" />
    - <Layout>
        <Show Name="Jonathen_Heart" PanelNumber="1" />
      </Layout>
    </Entity>
  </Container>
```

3) GUI representation of entity “Ambulance_Munic_00” after start up:

GUI representation of attribute Map on entity Ambulance_Munic_00:

4) Every entity contains its own interpreter, which is responsible for accepting and processing of incoming and outgoing context events. Therefore a set of context rules provide the basic functionality of the context scenario. Additional rules can be loaded at runtime, to change the scenario behaviour.

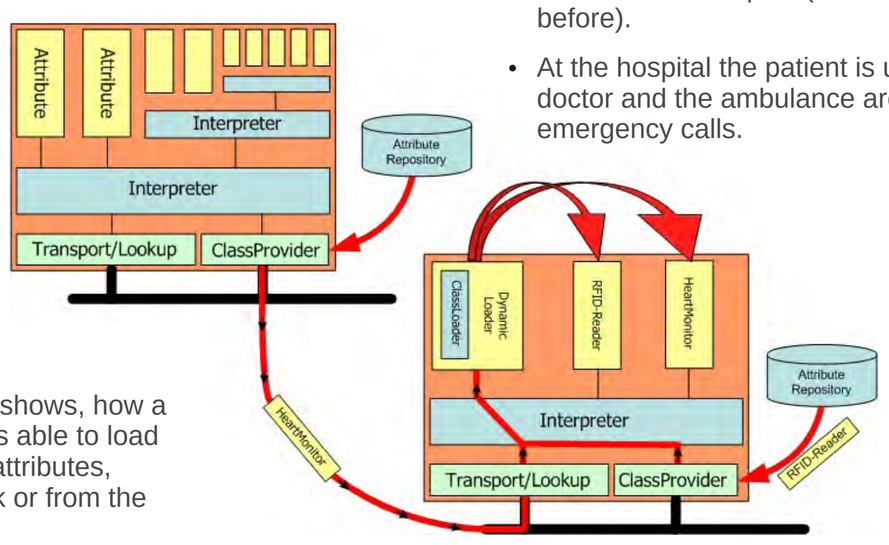
```
rules {
        long gx = 40;
        long gy = 40;

        on Jonathen_Heart.AlarmButton:Alarm () {
                Dispatcher.Map:drawObject ( "Jonathen_Heart", "heart.gif", gx, gy );
                Dispatcher.Map:findObject("Jonathen_Heart");
        }

        on Jonathen_Heart.RFID:TagAppeared ( string id ) {
                LaptopContainer.RFIDResolver:resolveRfid ( id );
        }

        on Jonathen_Heart.RFID:positionResolved ( string ident, long x, long y ) {
                gx = x;
                gy = y;
                Jonathen_Heart.Map:drawObject ( ident, "patient.gif", x, y );
        }
}
```



5) Munich Emergency Scenario

With the modeling of a complex emergency scenario, we try to show the flexibility of our SILICON Framework. Many different roles inside the emergency scenario where modeled (Dispatcher, Hospital, Patient, Ambulances, Doctors, Specialists, ...).
The application flow was designed by context rules as follows:

- Ambulance cars resolve their position on the scenario map.
- Doctors resolve their position on the scenario map.
- The emergency dispatcher is able to see all ambulance cars and doctors on the map.
- The heart patient resolves its position on the map, but does not share this information with anyone else.
- The patients heart monitor encounters a problem and calls the emergency dispatcher.
- The emergency dispatcher gets the alarm call from the heart patient and his position on the map.
- At the same time the emergency dispatcher gets a suggestion who is the nearest doctor and which is the nearest emergency car (assumption they are not busy).
- The dispatcher calls a doctor and an ambulance car from its list of entities.
- Both, the doctor and the ambulance have to accept or deny the dispatchers call.
- The ambulance car gets the patient and the doctor and travels to the hospital (which has been informed before).
- At the hospital the patient is unloaded and both, the doctor and the ambulance are now free for future emergency calls.



1), 2)

The picture above shows, how a context container is able to load all its entities and attributes, either over network or from the local file system:

SIEMENS
Siemens AG
CT SE 2
Otto-Hahn-Ring 6
81739 München
Tel. +49(89) 6 36-02
Fax. +49(89) 6 36-3 50 75
http://www.siemens.de/ct/

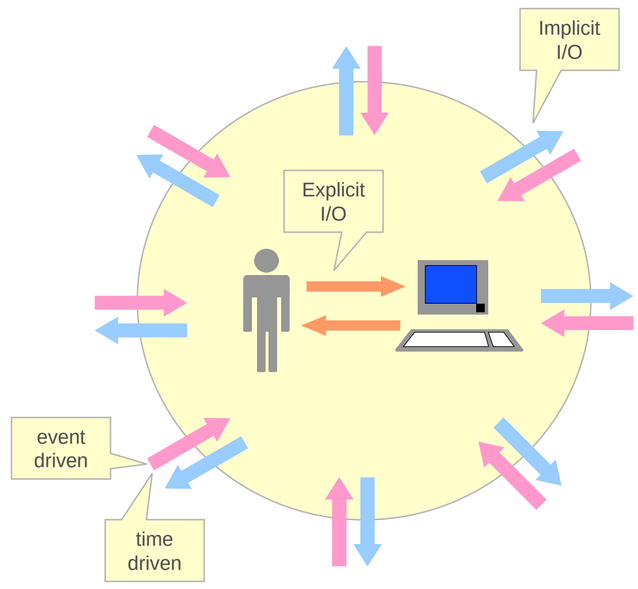
Institut für Praktische Informatik
Johannes Kepler Universität Linz
Univ. Prof. Alois Ferscha
Altenberger Straße 69
A-4040 Linz
Tel.: +43 (732) 2468 8556
Fax: +43 (732) 2468 8426
http://www.soft.uni-linz.ac.at

ContextWare

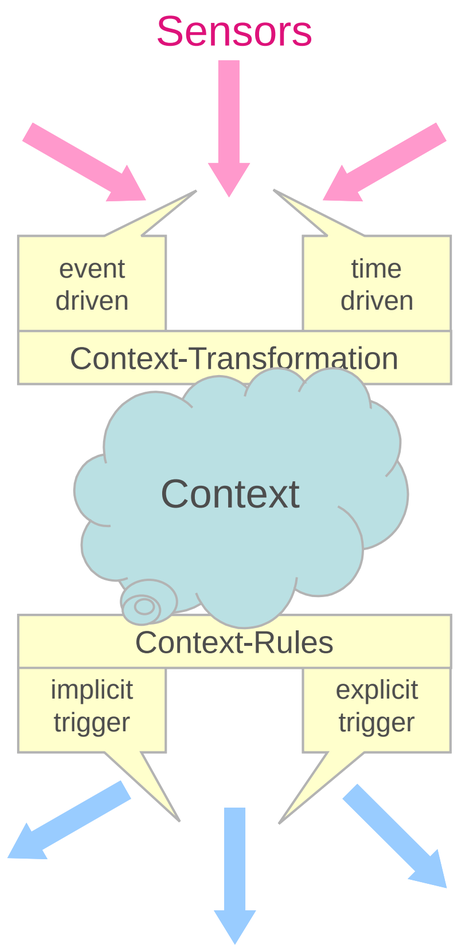
Context Awareness in Mobile Ad-Hoc Communication Scenarios

Deptartment of Computer Science
University of Linz, Austria
Alois FERSCHA
Volker CHRISTIAN
Wolfgang BEER
{ferscha,voc}@soft.uni-linz.ac.at
beer@ssw.uni-linz.ac.at

Corporate Technology
Software & Engineering
Siemens, Munich
Lothar BORRMANN
Lars MEHRMANN
Björn SCHIEMANN
{lothar.borrmann,
lars.mehrmann,
bjoern.schiemann}
@mchp.siemens.de



ContextWare Concepts



Context Sensing
acquire low level context information

Context Transformation
transform / aggregate / interpret low level context information

Context Representation
data structures for context information
centralized / decentralized?

Context Triggering
implicit / explicit event triggering

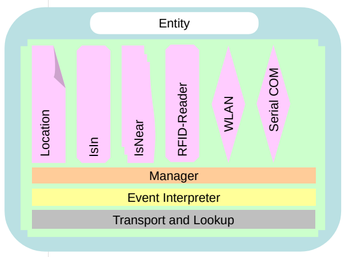
Controlling Actuators
control the environment

Awareness ...

"... an understanding of the activities of others, which provides a context for your own activities." [Dourish, Bellotti]

Context ...

" ... is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application..." [Dey 2000]

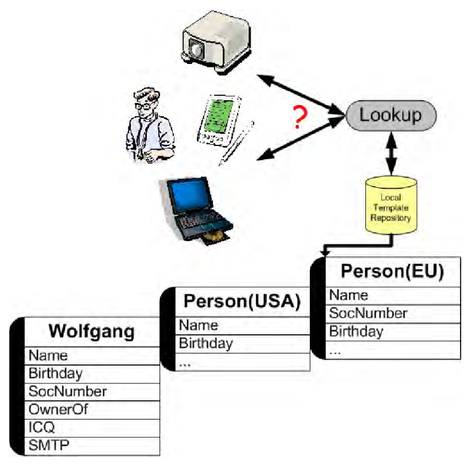


Entity Representation

A generic representation of context aware entities, assures that all kinds of objects can be involved in a context aware scenario. Entities are containers that are able to contain a set of entity specific attributes.

Dynamic Interaction

To enable the dynamic interaction between entities, actually among the attributes inside the entities, it is necessary to deploy certain interaction rules.These rules can be added or changed dynamicly at runtime. Context rules implicitly control the basic interaction in a context aware scenario.

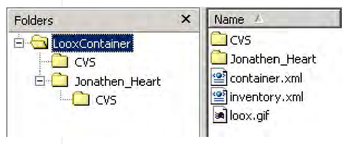


Entity Lookup and Classification

Unknown entities, that appear in a context aware scenario, can be classified with roles by the use of templates. The lookup attribute identifies new entities and compares them with the available set of templates. Therefore an entity is able to act in more than one role.
For context aware applications it is necessary to know what role an unknown entity plays in a scenario. Is it a person thing or place, or a combination of more than one role (e.g. a car is a place and a thing)

Routing of context information

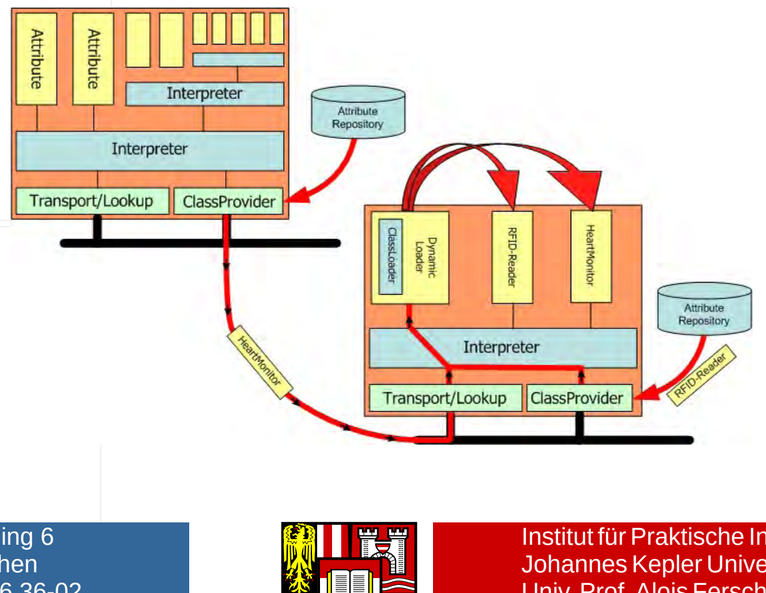
An effective use of context information is possible, if a highly dynamic routing of this information between different applications is possible. For example the use of spoken commands on a handheld, to control environmental settings (light, beamers, ...)



XML Configuration

With a XML based configuration syntax, it is possible to configure context aware scenarios without writing a single line of compiled program code. Furthermore predefined attributes can easily reused in individual scenarios.

```
- <Inventory>
  - <AttributeSequence>
    - <Attribute Name="AlarmButton" Host="localhost" Class="uni.linz.context.attributes.singlebutton.SingleButton">
        <AttributeSpecific Red="255" Green="0" Blue="0" Label="Alarm" />
      </Attribute>
    - <Attribute Name="RFID" Host="localhost" Class="uni.linz.context.attributes.rfid.Rfid">
      - <AttributeSpecific FakeJNI="false" AutoStart="true">
          <JNIInterface Class="uni.linz.context.attributes.jni.rfid.inside.RfidReaderJNI" Port="1" />
        </AttributeSpecific>
      </Attribute>
    - <Attribute Name="Name" Host="localhost" Class="uni.linz.context.attributes.name.Name">
        <AttributeSpecific FirstName="Jonathen" LastName="Heart" />
      </Attribute>
      <Attribute Name="IsIn" Host="localhost" Class="uni.linz.context.attributes.isin.IsIn" />
    - <Attribute Name="Map" Host="localhost" Class="uni.linz.context.attributes.map.Map">
      - <AttributeSpecific>
          <Map File="Munic_Map.jpg" Width="220" Height="210" />
        </AttributeSpecific>
      </Attribute>
  </AttributeSequence>
</Inventory>
```



Dynamic Attribute and Entity Loading

Dynamic attribute and entity loading provides a flexible way, to host non digital entities on digital devices. It is also possible to remotely load new attributes in an entity container, to extend the given functionality.
I could, for example, load an ICQAttribute to my colleques container in order to communicate with him over ICQ.

SIEMENS
Siemens AG
CT SE 2
Otto-Hahn-Ring 6
81739 München
Tel. +49(89) 6 36-02
Fax. +49(89) 6 36-3 50 75
http://www.siemens.de/ct/

Institut für Praktische Informatik
Johannes Kepler Universität Linz
Univ. Prof. Alois Ferscha
Altenberger Straße 69
A-4040 Linz
Tel.: +43 (732) 2468 8556
Fax: +43 (732) 2468 8426
http://www.soft.uni-linz.ac.at