Utilising the Event Calculus for Policy Driven Adaptation in Mobile Systems

C. Efstratiou
Lancaster University
Overview

- Adaptation in Mobile Systems
- Requirement for a Policy Based approach
- The Coordinated Adaptation Platform
- Event Calculus as a Policy Language
- Examples
- Open Issues
- Conclusions
Adaptation

- Current systems dealing with a single type of adaptation (i.e. network QoS, power).
- Need for applications capable of adapting to multiple types of adaptation triggers
  - Network QoS
  - Power availability
  - Service availability
  - User context
- Multiple applications
Problems and Restrictions of Current Systems

- Conflicting adaptation
- Un-coordinated adaptation
- No user awareness
  - Understanding of system behaviour
  - Support for customisable adaptation
The cause: Tight coupling of adaptation policies and mechanisms

- Current systems: hard coded adaptation policies within the adaptive applications.

- Requirement for:
  - Decoupling policies and mechanisms
  - Allow modification of policies
  - Allow dynamic user involvement in the adaptation cycle
The Coordinated Adaptation Platform

- Application Registration
  - Mechanisms
  - State variables
- Policy evaluation
  - State variables as events
  - Adaptation mechanisms as actions
The Event Calculus

- **Event**
  - \( e \) at \( t \)

- **Fluent**
  - \( e_i \) \( f \) \( e_j \)

- **Happens**\((e, t)\)
- **HoldsAt**\((f, t)\)
- **Initiates**\((e, f, t)\)
- **Terminates**\((e, f, t)\)
- **Clipped**\((f, t_1, t_2)\)
- **Declipped**\((f, t_1, t_2)\)
- \( t_1 < t_2 \)
The Event Calculus Policy Language

\[\begin{align*}
\text{event definition}_1 \\
\ldots \\
\text{event definition}_n \\
\text{fluent definition}_1 \\
\ldots \\
\text{fluent definition}_m \\
\text{condition} \{ \text{condition} \} \\
\text{action} \{ \\
\quad \text{action}_1 \\
\quad \ldots \\
\quad \text{action}_k \\
\} \\
\end{align*}\]
Policy Rules: Example 1

\[
\begin{align*}
\text{event} & \quad \text{lowBand} : - \quad \text{NetworkInterface.availableBandwidth} < 19200 \\
\text{event} & \quad \text{normBand} : - \quad \text{NetworkInterface.availableBandwidth} \geq 19200 \\
\text{fluent} & \quad \text{inLowBand} \{ \\
\text{  } & \quad \text{initiates} \text{(lowBand)} \\
\text{  } & \quad \text{terminates} \text{(normBand)} \\
\} \\
\text{condition} \{ \\
\text{  } & \quad \text{initiates} \text{(lowBand, inLowBand, t1) and} \\
\text{  } & \quad \text{not clipped} \text{(inLowBand, t1, t2) and} \\
\text{  } & \quad t2 = t1 + 30 \\
\} \\
\text{action} \{ \\
\text{  } & \quad \text{WebBrowser.LowBand()}}
\end{align*}
\]
Policy Rules: Example 2

\begin{verbatim}
\textbf{event} lowPower :- Battery.Percent < 10
\textbf{event} normPower :- Battery.Percent >= 10

\textbf{fluent} inLowPower {
    \textbf{initiates}(lowPower)
    \textbf{terminates}(normPower)
}

\textbf{condition} {
    \textbf{initiates}(lowPower, inLowPower, t1)
}

\textbf{action} {
    WebBrowser.LowBand()
}
\end{verbatim}
Policy Rules: Example 3

event lowPower :- Battery.Percent < 10
event normPower :- Battery.Percent >= 10
event webHighPriority :- Priorities.getPriority("WebBrowser") = 1
event webNormPriority :- Priorities getPriority("WebBrowser") != 1

fluent inLowPower {
  initiates(lowPower)
  terminates(normPower)
}
fluent atWebPriority {
  initiates(webHighPriority)
  terminates(webNormPriority)
}
condition {
  ( initiates(lowPower, inLowPower, t1) and
    not holdsat(atWebPriority, t1) ) or
  ( terminates(atWebPriority, t2) and
    holdsat(inLowPower, t2) )
}
action {
  WebBrowser.LowBand()
}
Open Issues

- **Efficient Policy Evaluation**
  - Model event calculus predicates as FSMs.
- **Policy Specification Conflicts**
- **Adaptation Conflicts**
  - Sequence of adaptation actions aiming at conflicting goals
  - Not always possible to determine what is the primary goal
  - User involvement may be necessary to resolve unclear situations
Conclusions

- Supporting multiple adaptive applications triggered by a variety of adaptation attributes.
- Decouple adaptation mechanisms and adaptation policies.
- Utilise an event based policy language that allows the explicit specification of time dependencies.