Ambient-Oriented Exception Handling

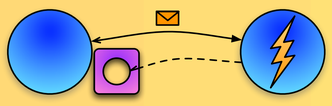
Stijn Mostinckx Tom Van Cutsem Elisa Gonzalez Boix
Jessie Dedecker Theo D'Hondt Wolfgang De Meuter

Programming Technology Laboratory
Vrije Universiteit Brussel, Belgium

Problems

Asynchrony

```
editor#merge(myLocalChanges);
```
- message scheduled for transmission
- future (placeholder for result) created



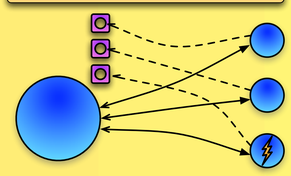
no need to wait for a result

```
try {
    editor#merge(myLocalChanges);
} catch(MergeX exc) {

}
```
Context is left before exceptions may be reported

```
try {
    editor#merge(myLocalChanges);
} catch(TimeOutX exc) {

}
```
Future may never get resolved

Concurrency

```
for editor in participants
    editor#merge(myLocalChanges);
```
each message is executed in parallel



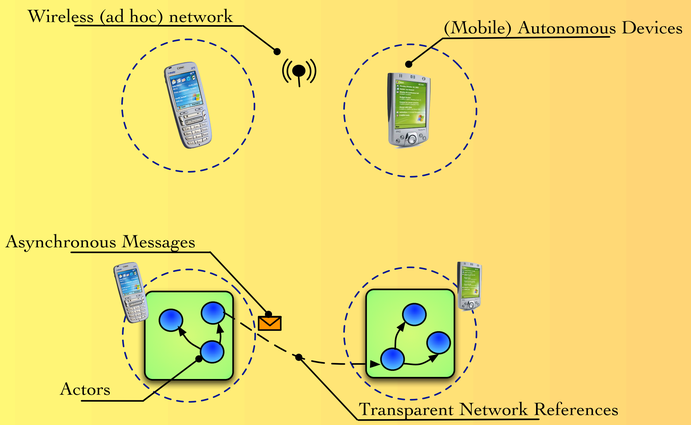
independent futures

```
try {
    for editor in participants
        editor#merge(myLocalChanges);
} catch(MergeX exc) {

}
```
Multiple exceptions may be reported concurrently

```
try {
    for editor in participants
        editor#merge(myLocalChanges);
} catch(MergeX exc) {

}
```
Implicit dependencies

Context

Pervasive Computing Mobile Networks

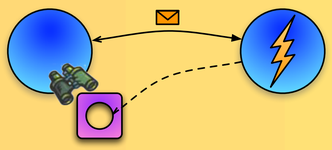


Example: Collaborative Editor

Primitives

Future-based Propagation

```
future = editor#merge(myLocalChanges);
when(future) { /* post-processing code */ }
```
- scheduled upon resolving the future
- access to lexical environment



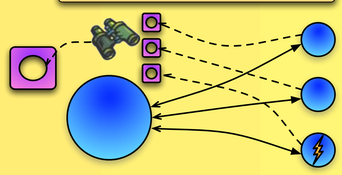
reconciling events with "sequential" code

```
when(future) {
    /* post-processing code */
} catch(MergeX exc) {

}
```
Context is captured and bound to a future rather than the stack

```
when(future) {
    /* post-processing code */
} due( 60 seconds ) {

}
```
Attribute deadlines to future resolution

Concerted Exceptions

```
group {
    for editor in participants
        editor#merge(myLocalChanges); }
```
Observes all futures within a block and resolves a future of its own



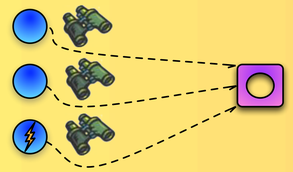
clustering for futures

```
group {
    for editor in participants
        editor#merge(myLocalChanges);
} catch(MergeX exc) {

}
```
Immediate handling given a log of previously raised exceptions

```
group {
    for editor in participants
        editor#merge(myLocalChanges);
} resolve(Exception[] exc) {

}
```
Wait for all futures to be resolved Treat all raised exceptions jointly

Collaborative Handling

```
conversation( editors )
```
Participants observe the conversation and will propagate exceptions to it.



clustering for actors

```
startConversation(conv) {
    when(conv) { /* when finished */
    } catch(MergeX exc) {
        /* handler */ }
}
```
Hook to install handlers

```
catch(MergeX exc) {
    become(recoveryBehaviour);
    thisActor#mergeConflict(exc)
}
```
Collaborative handling by switching in dedicated failure handling behaviour

Further Reading:
[1] Stijn Mostinckx, Jessie Dedecker, Elisa Gonzalez Boix, Tom Van Cutsem, Theo D'Hondt, Wolfgang De Meuter, "Ambient-Oriented Exception Handling", *in Advanced Topics in Exception Handling Techniques", eds. C. Dony, J. L. Knudsen, A. Romanovsky, A. Tripathi.* LNCS 4119, p. 141-160, Springer-Verlag, 2006.

http://prog.vub.ac.be/amop