Architectural Reconfiguration using Coordinated Atomic Actions

Rogério de Lemos
University of Kent, UK

- Motivation;
- Coordinated atomic actions (CA actions);
- CA actions applied to reconfiguration;
- Conclusions;
System reconfiguration is not that simple:
- it’s hard!

In dependable system where assurances are necessary:
- it’s even harder!!

What happens if something goes wrong?
- atomic actions applied to reconfiguration - nothing new!
  - ensures consistency in the presence of failures and concurrent access;
- coordinated atomic actions (CA actions);
  - Newcastle Univ. (B. Randell group), 15 years ago;
Where are the ‘self’ properties?

- a general mechanism that can be used in self-reconfigurable systems;
- systems react to “unexpected” situations through “predictable” means;
In the context of fault tolerance:

- **fault handling during system recovery:**
  - addition, removal, or replacement of components and connectors;
  - modifications to the configuration or parameters of components and connectors;
  - alterations in the component/connector network’s topology.

- **the outcome of reconfiguration should be a safe (stable and useful) state in the system configuration:**
  - sequence of atomic actions has been widely advocated:
Coordinated Atomic Actions (CA Actions)

CA actions is a unified approach that deals with both competitive and cooperative concurrency:

- **transactions** - maintain the consistency of shared resources in the presence of failures and concurrency;
- **conversations** - control cooperative concurrency, and implement coordinated and disciplined error recovery;

CA actions have applied to:

- structuring complex and concurrent activities for error confinement;
- supporting the provision of error detection and handling.
CA Action

Activity T1
start transaction

Activity T2

exception handler H1
abnormal control flow
exit after successful handling

exception handler H2
abnormal control flow

start transaction

commit transaction

exit points

entry points

accesses
recovery

external objects

exception handling context

Borrowed from A. Romanovsky
Existing applications of CA actions:

- it has focused on error handling – application dependent;
- fault handling are considered in the context of the application;
- there is no explicit separation of concerns between application and reconfiguration services;
Separation of Concerns

At the level of the architectural element:

- internal structuring for the purpose of error confinement;
- more attention to each part, and their interaction;
- promotes reuse on configuration services since they are similar across architectural elements.
Peer-to-peer architectural style;

Architectural elements:
- stereotyped UML2.0 components;
- Provided and required interfaces;
Replacing Components

Rogério de Lemos
Replacing Components

FrontEnd_C

Single_C

BridgeYahoo_C

BridgeLycos_C

time

connect_Config_Single

disconnect_Config_Single

connect_Config_Yahoo

connect_Config_Lycos

connect_FE

connect_BY

disconnect_FE

connect_FE

connect_BL
Replacing Connectors

- <<Component>> FrontEnd
- <<Connector>> Single
- <<Component>> BridgeYahoo
- <<Component>> BridgeLycos

- <<Component>> FrontEnd
- <<Connector>> Single
- <<Component>> BridgeYahoo
- <<Component>> BridgeLycos
Replacing Connectors

FrontEnd_C
Single_C
BridgeYahoo_C
BridgeLycos_C
BridgeMS_C
Vote_C

connect_Config_Voter
connect_FE
connect_TMR
connect_BY
connect_BL
connect_BMS

disconnect_Config_Voter
disconnect_FE
disconnect_TMR
disconnect_BY
disconnect_BL
Separation of Concerns

*Computation, coordination and configuration* (CCC) system architecture [Fiadeiro, Wermelinger, Andrade, etc.]
Conclusions

- Applying CA actions as a mechanism for supporting dynamic architectural configuration;

- Separation of concerns between:
  - error detection and recovery, which is application dependent;
  - fault handling, which can be incorporated into the middleware;

- For self-adaptive systems, structural flexibility is obtained by:
  - small increments in the system configuration that can be undone in case of failure;
Conclusions

 Fault tolerance at the architectural level

- error detection and handling:
  - application dependent;
  - idealised Fault Tolerant Architectural Elements (iFTE);

- fault handling:
  - not application dependent;
  - reconfiguration support by CA action;