Entities and dynamic reconfigurations

DISTRIBUTED SYSTEMS RESEARCH GROUP
http://dsrg.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
Motivation
Observation:

These “entities” are invisible in both architectural and behavioral view.

OOP view

Component view

FileManager

FileManager

File

App

File

App

File

File

App

File

File

App

File

File

App

File

File
FileManager – Example assignment

• Component candidates
  ▪ Editor
    • Accesses multiple files (entities)
      ▪ open, read, write, close
  ▪ FileManager
    • Provides the abstraction of files
    • Accesses storage
      ▪ readBlock, writeBlock
  ▪ Storage
    • Actual HW access as an internal action
  ▪ ...

How should such an architecture look?
FileManager – Options

A) Entities *not* modeled at all
B) Entities ~ separate components
C) Entities ~ separate interfaces
FileManager – Summary

A) Intuitive solution
☺ Supported in component models
☺ Entities not modeled

B) Entities modeled
☺ Runtime arch. changes (unsupported)
☺ Internals of FileManager exposed

C) Entities modeled
☺ Runtime arch. changes (unsupported)

This presentation is about how to overcome this
Our proposal
Dynamic reconfiguration – Bindings

**Note**

- *Let’s consider situation C)*
- *Let FileManager be a primitive component*
Dynamic reconfiguration – Bindings

Key idea

• Dynamically created/destroyed interfaces
  ▪ Interface reference can be passed among components
• Dynamic binding/unbinding of these interfaces

```
FileManager
  FileManager
  Editor
    IFile
      open()
      read() / write()
      close()
```

FileManager

Editor

IFile

open()

read() / write()

close()
Taxonomy (1/2)

- Interfaces
  - Single / Collection of
    - Element of an interface collection ~ interface

- Bindings
  - “Classical” bindings (Static / Dynamic)
  - Protobindings
Reconfiguration actions (1/2)

• Reconfiguration actions
  ▪ Triggered by method calls
  ▪ Specified as annotations
    • Property of a frame!

• Create & Destroy
  ▪ Property of the entity owner’s frame

• Link & Unlink
  ▪ Property of the entity user’s frame

Note: Binding can only be created if there is a path of protobindings between the two components
Dynamic reconfiguration – Examples

**Scenario:** Client & Server (entity provider)

**Note:** Return value is a “proxy”

Recall: Reconfiguration action is a property of a frame
**Scenario:** Worker with callback

```
performWith(@create @destroy IEntity)
```

```
performWith(@link @unlink IEntity)
```

Client

use()

Worker

use()
**Scenario:** Passing interface references

- **Client**
  - Security
  - Worker

- **Server**

**Code Examples:**

```plaintext
@create IEntity open()

@link IEntity open()

use()
@unlink close()

@create IEntity open()

use()
@destroy close()
```
**Scenario**: Passing interface references

- Client
  - Security
  - Worker

- Server
  - IEntity open()
  - performWith(IEntity)
  - performWith(@link IEntity)
  - use()
  - @unlink close()
Dynamic reconfiguration – Examples

**Scenario:** Multiple inner workers
Dynamic reconfiguration – Adding components

Note

- *We already have “dynamic” bindings*
- *Let’s consider situation C) again*
- *Let FileManager be a composite component*

![Diagram showing FileManager with IFile open, read, write, close methods, FileManagerLogic, File1, and File2 with init method.](image-url)
**Key idea**

- A *protocomponent* (component template) as a part of architecture
  - Associated with its deployment plan

![Diagram](image)

**Code Snippet**

```plaintext
FileManager

FileManagerLogic

IFile open()

init()

newFile(out IFile, out IInit)

read()

write()

close()

File1

File
```
Taxonomy (2/2)

- Components
  - “Classical” components (Static / Dynamic)
  - Protocomponents
Reconfiguration actions (2/2)

• Specified as annotations
  ▪ Property of an architecture!

• New
  ▪ Annotation of an unbound required interface
    • Restricted signature
    • Calls are accepted by the runtime
  → Instantiation of a frame, bindings, and protobindings
    • Protocomponent as a template

• Delete
  ▪ Annotation of a method(s) on protocomponent’s interface(s)
  → Deletion of the dynamic component
**Dynamic reconfiguration – Example**

**Note:** FileManagerLogic *doesn’t know, whether newFile() creates a dynamic component or only 2 dynamic interfaces.*
Dynamic reconfiguration – Example

**Note:** *Outside FileManager everything looks as in the previous case. Only creation of dynamic interfaces is observable.*
Summary

• Architectural (template) elements
  ▪ Protobinding
  ▪ Protocomponent

• 6 reconfiguration actions
  ▪ Binding reconfiguration
    • Property of a frame
    • Link & Unlink, Create & Destroy
  ▪ Component reconfiguration
    • Property of an architecture
    • New & Delete
Conclusion

Architectural benefits

• Reconfiguration captured by the architecture description

• Architecture as an initial snapshot vs. a pattern

• Fits smoothly into the micro-component view of runtime

• Research challenges with implementation
Conclusion

**Verification benefits**

- Possibility to model this kind of dynamic reconfiguration
  - The need is already here
    - Since FT demo
  - Interesting properties to be checked
    - Protocol of a single File instance
    - Reconfiguration correctness
Conclusion

Affects also

• Performance
  ▪ Entities need resources (e.g., memory)
  ▪ Time complexity related to the number of existing entities
  ▪ Synchronization for shared entities

• Security
  ▪ Interface reference passing
Future work

• Discussion
  ▪ Inexpressible patterns

• Word the idea
  ▪ Publication / Technical report

• Implementation
  ▪ Protocols / SOFA