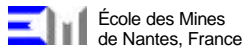


Integrating Behavior Protocols in Enterprise Java Beans

Andrés Farías
Yann-Gaël Guéhéneuc

Speaker: Jacques Noyé



Behavior protocols as behavior specifications of components.



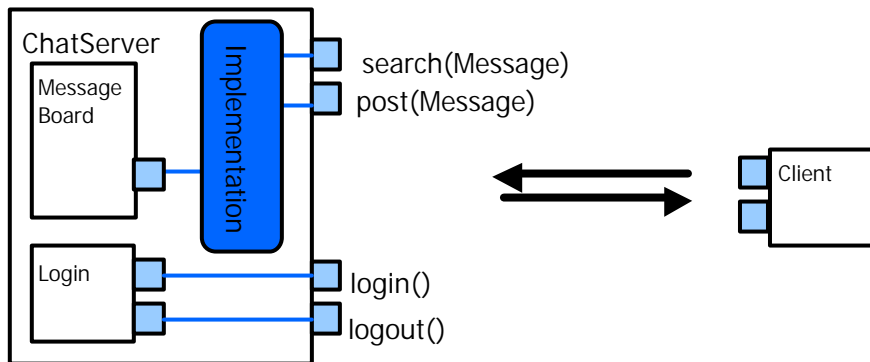
Plan

- Overview of components models with explicit protocols.
- A notion of coherence:
 - Protocols at the implementation and specification levels.
 - Coherence verification and compliance.
- Integrating explicit protocols in the EJB component model.

There are mainly 3 points:

- You are going to explain what's a component is with the example of the chat server AND in the same example you will explain what protocols are.
- Then, as the “contribution” of the paper you will introduce the notion of coherence between the protocol specified and the real protocol followed by a component. You will also talk about how to verify this notion and how to make two non compliant protocols compliant.
- Finally, you will offer a little discussion about how integrating protocols in the EJB component model.

Example: A *chat server*



As an introductory example you will present the example of the Chat server that you know already.

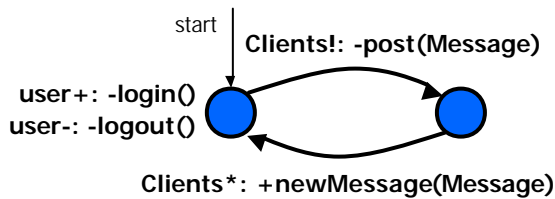
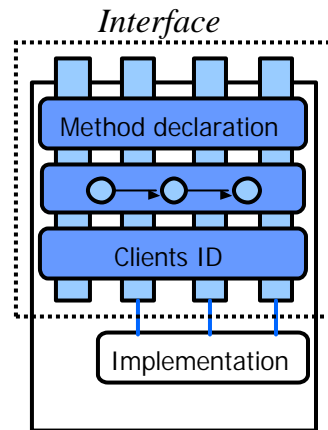
The main points to show is that:

- A component offers a set of services.
- A component use collaborators to solve implementation issues. Some of them provide directly services to be offered by the component (Login). Others are used in the component implementation (MessageBoard).

Then you can pass directly to the next slide where you can talk about the interaction between this component and its collaborator: services are not always available OR you can say that their availability is non-uniform.

What are protocols?

- Sequential constraints over service requests/receptions.



So in fact, you say that services may be implemented in such a way that they must be used following a given pattern. You can show that protocols can serve to specify the interactions that must be followed by the component and its collaborators

Then, you can say that protocols should ideally be at the interface level in which other information such as the identities of the collaborators should be specified too.

Next slide is going to talk about the natural questions that appear when dealing with this kind of component models, so you can say this and show the following slide!



Dealing with protocols...

- Programmers writing down a protocol for every component is not realistic.
- Does a component implementation follows a given protocol already?
- How to determine the protocol followed by a given component?
- How to make a protocol compliant with respect to a given protocol?

There are several “natural” questions that must arise in such a component model:

- It is not realistic that programmers are going not only to write the implementation but also to write a protocol for some components here the protocol is in fact simply described by the implementation. Ohhh that lazy people... but in fact you can say that it is a problem for them... (but we will offer them a good solution: CwEP, but this, you don't have to say it right now.
- You can wonder whether a given component is compliant or not with a given protocol.
- How to know what is the protocol followed by a given component?
- If my component does not follows a given protocol, what do I have to modify in order to turn it compliant?

That's are enough questions for a single workshop, so we stop here and we start proposing some concrete solution to those questions.

Next slide we will start defining the notions necessary to answer the questions.

A notion of coherence

(1/2)

- The protocol a component follows is called *Implementation-level Protocol*.

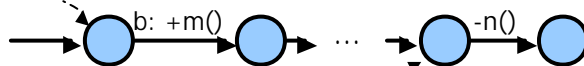
```
public class A {  
    B b;  
    public void n(){ ... }  
    public void foo(){
```

• b.m();

...

• this.n();

...



So in this slide you introduce the notion of a protocol at the level of the implementation. You can say that this protocol is obtained from the component source code after being analysed.

As shown in the picture every instruction is preceded by a point representing a state in the protocol. One instruction is represented by only one state, by one state may represent several instructions.

There is also a little algorithm for analyse well defined collaborators and then we are able to detect them and make them explicit in the component protocol.

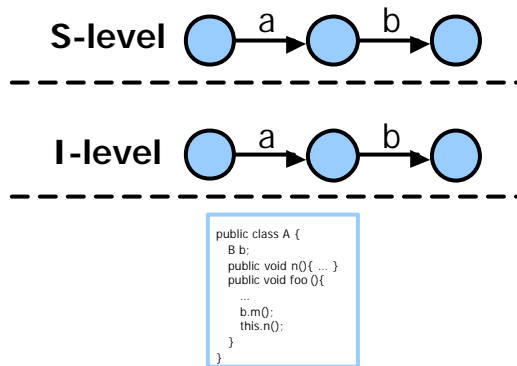
The pictures shows two method calls. Remarc that a call TO a component is marked with a "+", while auto calls are marked with "-" representing a self service request.

Now that we have defined the protocol at the implementation level, we will define the component at the specification level (in the next slide) and then the notion of coherence.

A notion of coherence

(2/2)

- The protocol a component must follow is called *Design-level Protocol*.
- Coherence between protocols:



The protocol at the specification level is the protocol that designers may define for a component.

At the moment where we wrote the article, we didn't really care about how comparing two protocols. The worst case is simply using structural comparison, but today we are using an equivalence relation (two protocols will be coherent if one can substitute (under Nierstrasz definition) the other and viceversa).

So Coherence is just about saying if protocols are equivalents.

Now that we know what means two protocols being coherent we will attack the problem of "how to".



Verification, compliance, and prototype

- Coherence verification:
 - Dynamic.
 - Static.
- Modifying component implementation:
 - Dynamically.
 - Statically.
- Prototype CwEP:
 - Based on the Eclipse parser.
 - Build on an implementation of protocol.

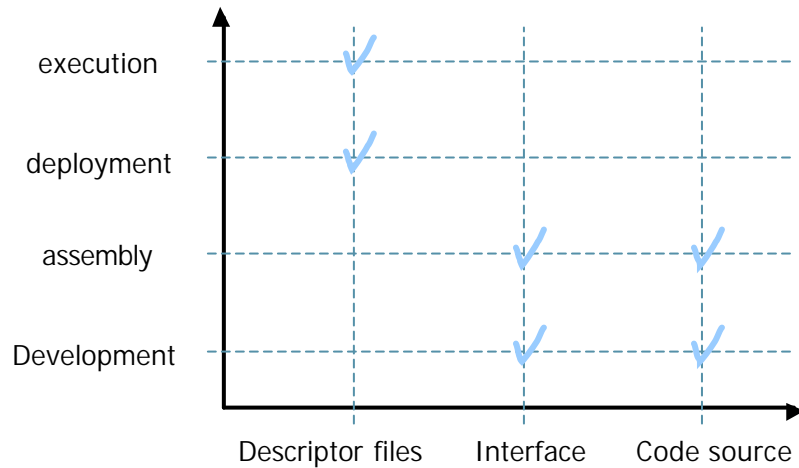
Coherence could be verified statically and dynamically.
See the article.

If you have the two protocols and they are not equivalent
we could modify the implementation protocol in order
to make it compliant. (see LMO 2002 =)

Finally, we ARE building a prototype capable of doing
all of that. Now, the prototype can just do the protocol
extraction.

Next slide is about integrating this model in the EJB.
Then, a good transition could be to say that all of this
could be done in a concrete component model, so
we have studied how to proceed and there are several
alternatives. NOW you change the slide...

Explicit protocols in the EJB



9/10

In this graphic we can see that in fact the integration is less “possible” when you try to do it in the last phases of the life cycle. Moreover, it seems that doing it at the interface level with help of code to ensure that the protocol is respected is the best solution.

See more details in the paper...



Conclusions and Future work

■ Conclusions:

- Components with explicit protocols.
- Coherence between specification protocols and implementation protocols.
- Discussion of the integration of explicit protocols in the EJB component model.

■ Future Work:

- Integration of protocol specifications in an open EJB server, such as Jonas.
- Extension of the prototype for protocol generation and coherence detection.

Ok, for the conclusions you think a lot.

Future work is also natural: try to finish the prototype and do real experimentation in the EJB component model.