Distributed Object Inheritance to Structure Distributed Applications

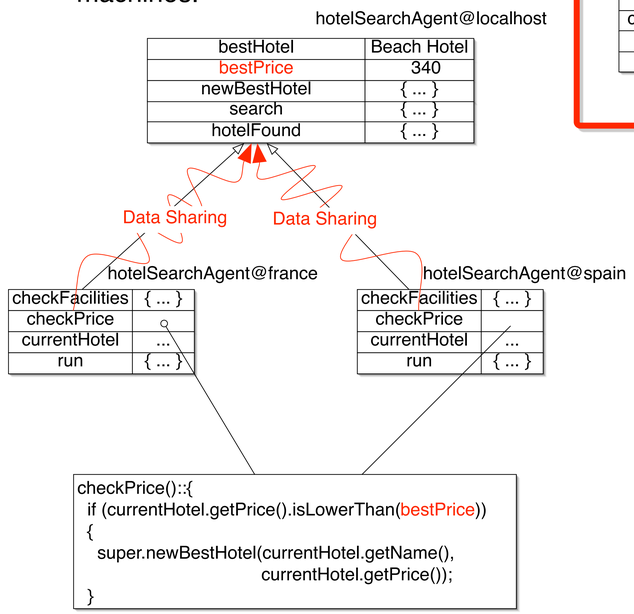
Vrije Universiteit Brussel -- Programming Technology Laboratory -- http://prog.vub.ac.be
Jessie Dedecker -- Thomas Cleenewerck -- Wolfgang De Meuter
(jededeck | tcleenew | wdmeuter)@vub.ac.be

Introduction

Prototype-based languages (PBLs) are good at sharing information between objects, while sharing is a ubiquitous problem in distributed application programming (due to concurrency and partial failures). New language concepts can exploit the advantages of PBLs to ease the distribution problems.

Distributed Object Inheritance

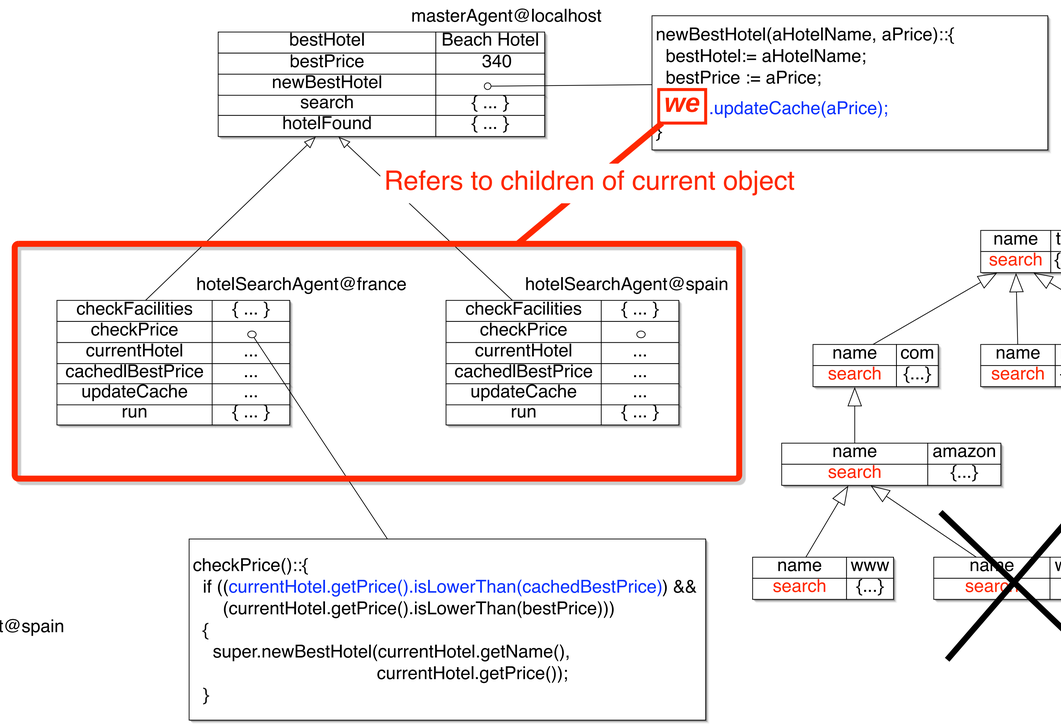
Objects can share information with each other through the use of *object inheritance*. When an object receives a message that it does not know it can *delegate* that message to its object parent, which then executes that message in the context of original receiver. The child - parent object relation may be distributed over multiple machines.



Communication Patterns

Object inheritance raises the split object problem. The split object problem can be solved by transparent multicast messages, depending on the required semantics of the split object.
Multicast messages are introduced via:

- WE keyword
- Multivalue references
- Per Object



Multivalues are a more general form of the WE keyword. Multivalues can hold references to multiple objects. When a message is sent to a multivalue, then the message is sent to all objects referred to by the multivalue. The result of such a message is then again a multivalue reference.

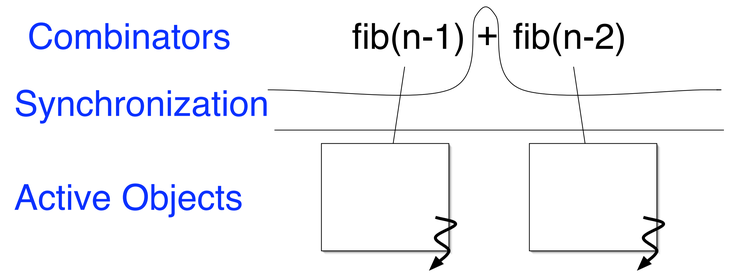
Dynamic Object Hierarchies

Object hierarchies are very dynamic. This way the hierarchies can match the dynamicity of the distributed applications (e.g. computers are continuously connecting and disconnecting to the network). For example if we want to build a dynamic domain name server we can create it very naturally with distributed object inheritance.

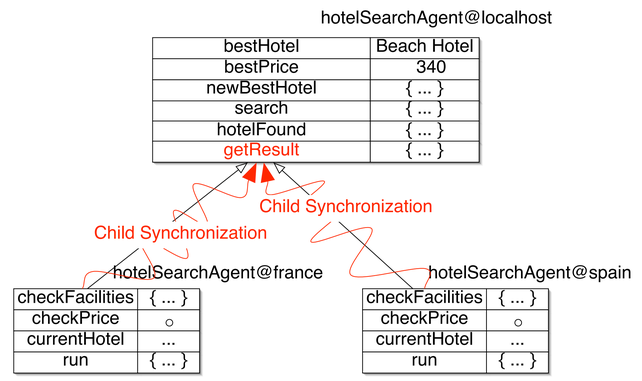
Synchronization

When objects are concurrently cooperating to achieve some goal there is a need for coordination between different concurrent objects. Concurrent processes are synchronized using different kinds of method synchronizers:

1) Combinator methods



2) Child synchronization is used when a parent object is waiting for its child objects to have reached a certain point in their execution.



References

[1] Robert Tolksdorf and Kai Knubben. Programming Distributed Systems with the Delegation-based Object-Oriented Language dSelf , In Proceedings of SAC'02 - ACM 2002 SYMPOSIUM ON APPLIED COMPUTING, Technical Track on "Programming Languages and Object Technologies". 2002.

[2] Jean-Pierre Briot and Akinori Yonezawa.
Inheritance and synchronization in concurrent OOP.
European Conference on Object-Oriented Programming (ECOOP'87), Paris, France, number 276 in Lecture Notes in Computer Science, pages 32-40, Springer-Verlag, 1987.