Discussion on some Challenges and Evolutions for Exception Handling

Christophe Dony

Montpellier-II University - LIRMM

http://www.lirmm.fr/~dony

Presentation at :

Workshop on Exception Handling in Contemporary Software Systems - LACD'11

A Programming language point of view

…

Giving programmers control structures

to manage those situations

in which standard execution is blocked

Some challenges for Exception Handling

- C1 : Toward usages, best practices and patterns.
  - Convince that EH is necessary and useful
  - Improve today mainframe EHS languages (Java, …?)
    - Problems? misuses? solutions?
- C2 : Abstraction, Efficiency, Reuse : rediscovering lost ideas
  - Architecture level handlers
  - Exception handling as a dialog (resumption, restarts)
- C3 : Build new EHS for the new world and using the new world
  - components, aspects, services, ambient, ubiquitous, concurrent
  - Example of MAS
- C4 : Orthogonal dimensions
  - Cover the life cycle
  - Combination of tools and techniques
    - Example of exception handling and replication

Some challenges for Exception Handling

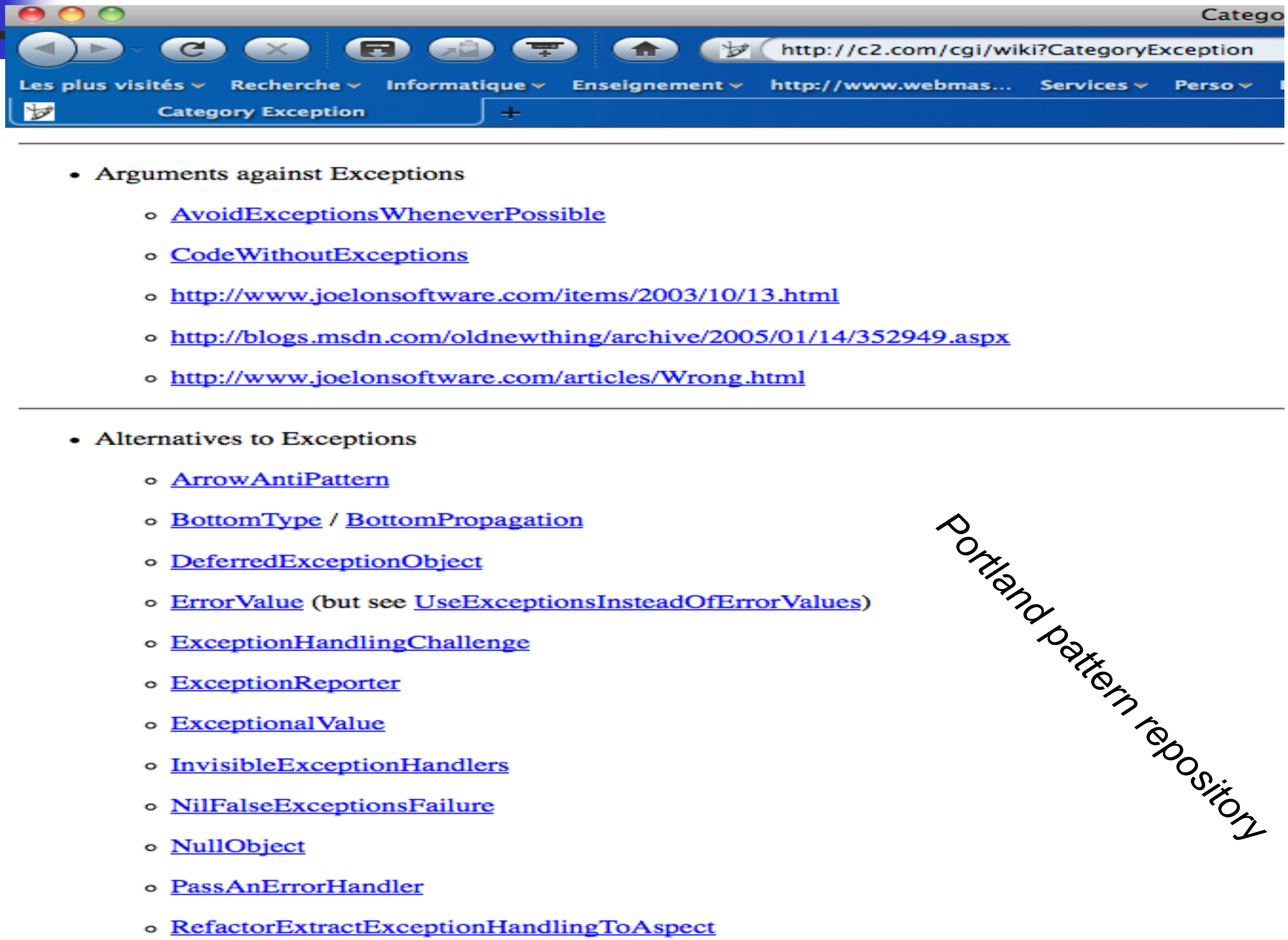
- C1 : Toward usages, best practices and patterns.
  - Convince that EH is necessary and useful
  - Improve today mainframe EHS languages (Java, …?)
    - Problems? misuses? solutions?
- C2 : Abstraction, Efficiency, Reuse : rediscovering lost ideas
  - Architecture level handlers
  - Exception handling as a dialog (resumption, restarts)
- C3 : Build new EHS for the new world and using the new world
  - components, aspects, services, ambient, ubiquitous, concurrent
  - Example of MAS
- C4 : Orthogonal dimensions
  - Cover the life cycle
  - Combination of tools and techniques
    - Example of exception handling and replication

Are exception handling systems useful? Many people are not convinced

- Alternative opinion do make sense
    - *[Bla82] Andrew P. Black. Exception Handling: The Case Against. Phd dissertation, University of Oxford, January 1982.*
    - *See also ECOOP 2005 EH workshop reader.*

- *"what should be specifically handled is not what is exceptional but what is unexpected i.e. deviation from specifications" …*
- *"EHS is undesirable because exception handling constructs introduce difficulties with programming languages semantics and use,"*
- "*EHS are unnecessary because exception handling specific constructs could be provided or subsumed by less specific ones*"
    - Passing handlers as parameters …
    - Exceptional values ….

Running discussions (papers, web) …



- Arguments against Exceptions
  - AvoidExceptionsWheneverPossible
  - CodeWithoutExceptions
  - http://www.joelonsoftware.com/items/2003/10/13.html
  - http://blogs.msdn.com/oldnewthing/archive/2005/01/14/352949.aspx
  - http://www.joelonsoftware.com/articles/Wrong.html

- Alternatives to Exceptions
  - ArrowAntiPattern
  - BottomType / BottomPropagation
  - DeferredExceptionObject
  - ErrorValue (but see UseExceptionsInsteadOfErrorValues)
  - ExceptionHandlingChallenge
  - ExceptionReporter
  - ExceptionalValue
  - InvisibleExceptionHandlers
  - NilFalseExceptionsFailure
  - NullObject
  - PassAnErrorHandler
  - RefactorExtractExceptionHandlingToAspect

Portland pattern repository

What can we do?

- P1 : Discuss the term “exception” …
- P2 : Convince developers that built-in solutions are less powerful than EHS
- P3 : Write patterns.
  - Unified set of constructs?
  - Mainframe languages?
- P4 : Suggest improvements to Java seen as a mainframe language for EHS

What can we do?

- P1 : Discuss the term “exception” …
- P2 : Convince developers that built-in solutions are less powerful than EHS
- P3 : Propose patterns.
  - Unified set of constructs?
  - Mainframe languages?
- P4 : Suggestions to improve Java as a mainframe language for EHS

Exceptions are not exceptional !!!

- Exceptions (in our computer science context)
    - does not denote in whole generality
  - exceptional situations
    - but
  - **situations that prevent standard executions to pursue**

- Some are rare (exceptional :-( )
    - VirtualMachineError
    - Eyjakjallajokul eruption
- Some are frequent
    - *IOExceptions*
    - printer out of paper or inc

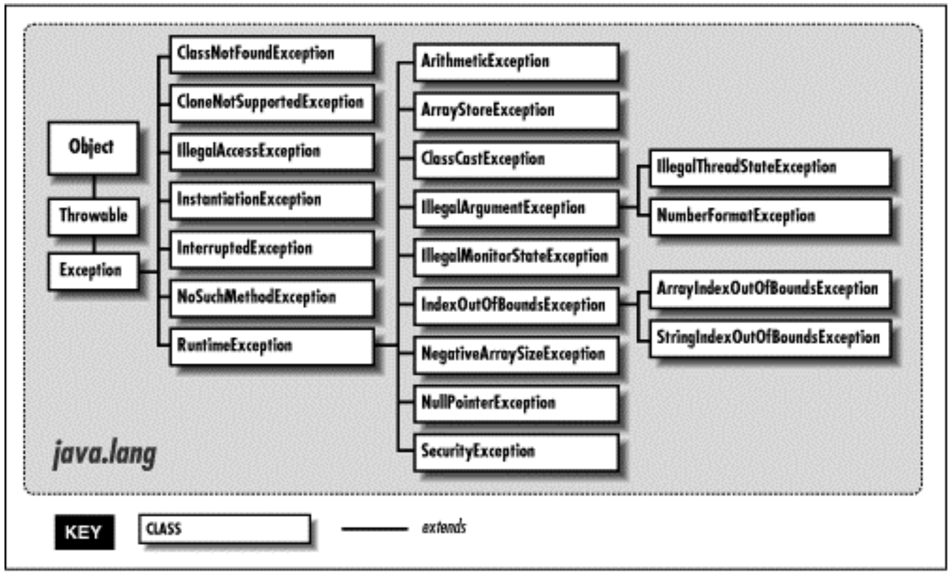
Exception are not exceptional !!!

- This « lapsus linguae » is a bigger issue :
  - Induces recurrent new suggestions :
    - *fault, failure, condition, alarm, signal, emergency*, …
  - Induces recurrent discussions :
    - *“what should be specifically handled is not what is exceptional but what is unexpected …”*
    - …

    of which this discussion is another example

Another term?

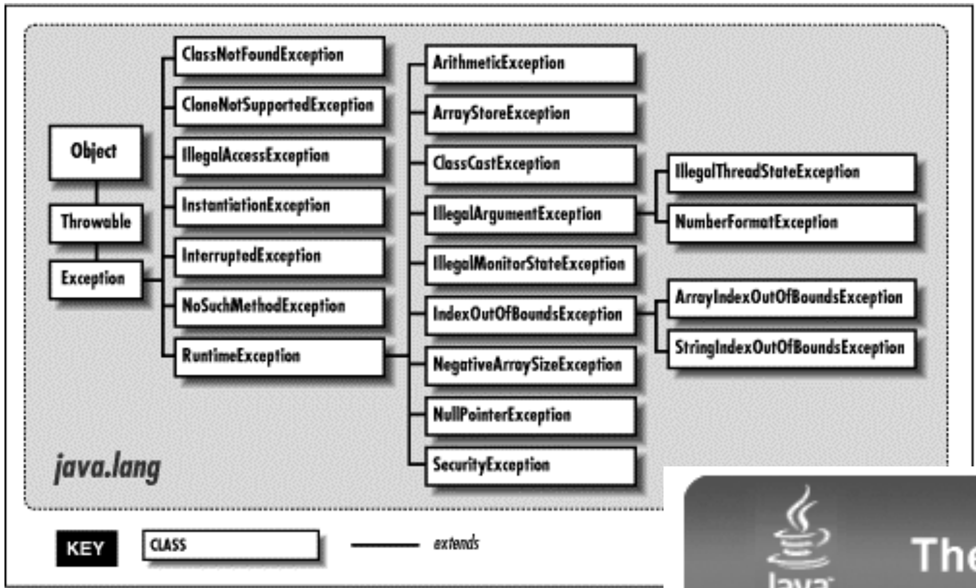
- Is it reasonable ?
- If yes, which one?
  - "unexpected"? No, we know such situations do happen
  - "unpredictable"? No, what is unpredictable is "when", not "what"
    - We have lists of "what"



  - "Uncontinuable" … why not?
  - "Throwable" … quite good but …

Problem gets even more complicated with classifications

- A classification (Java's one) of "exceptions" in which "Exception" is one of the categories



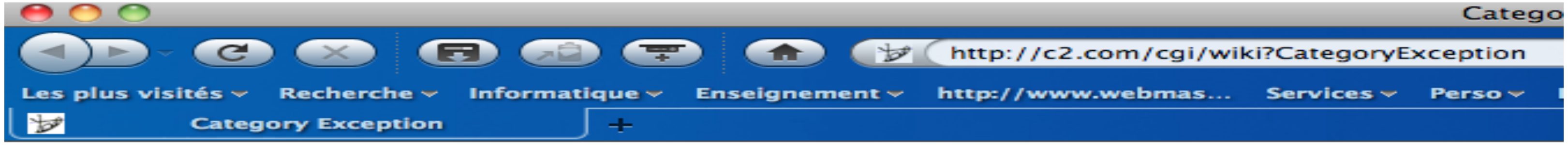
I’ll continue to use the term “exception” in this talk anyhow …

Imposing an appropriate and definitive term is a true challenge …

What can we do?

- P1 : Discuss the term “exception” …
- P2 : Convince developers that built-in solutions are less powerful than EHS
- P3 : Propose patterns.
  - Unified set of constructs?
  - Mainframe languages?
- P4 : Suggestions to improve Java as a mainframe language for EHS

Convince developers that built-in solutions are less powerful than EHS

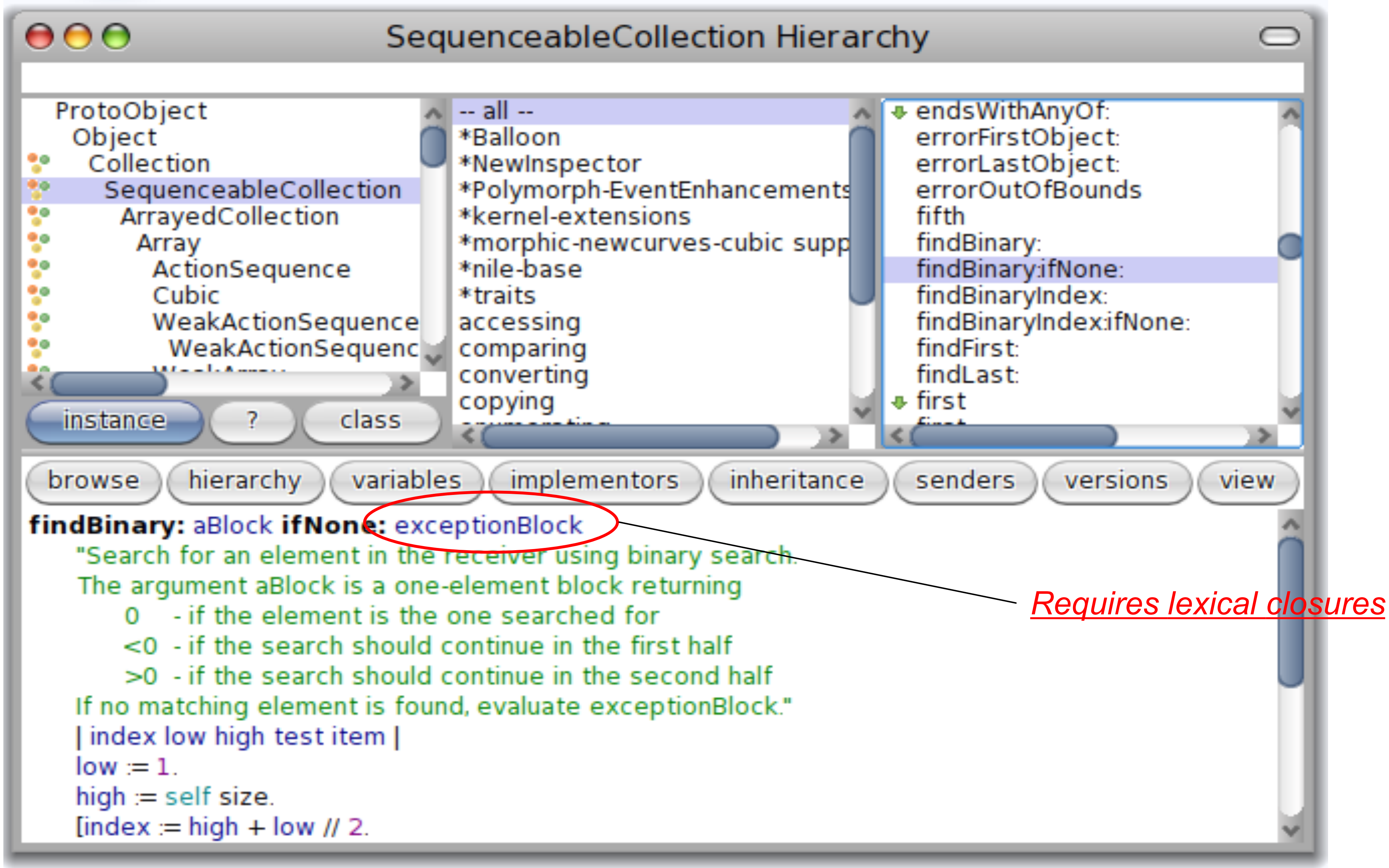


- Arguments against Exceptions
  - AvoidExceptionsWheneverPossible
  - CodeWithoutExceptions
  - http://www.joelonsoftware.com/items/2003/10/13.html
  - http://blogs.msdn.com/oldnewthing/archive/2005/01/14/352949.aspx
  - http://www.joelonsoftware.com/articles/Wrong.html

- Alternatives to Exceptions
  - ArrowAntiPattern
  - BottomType / BottomPropagation
  - DeferredExceptionObject
  - ErrorValue (but see UseExceptionsInsteadOfErrorValues)
  - ExceptionHandlingChallenge
  - ExceptionReporter
  - ExceptionalValue
  - InvisibleExceptionHandlers
  - NilFalseExceptionsFailure
  - NullObject
  - PassAnErrorHandler
  - RefactorExtractExceptionHandlingToAspect

Example

The case for higher order functions (1)



The case for higher order functions (2)

```
findBinary:
findBinary:ifNone:
findBinaryIndex:
findBinaryIndex:ifNone:
findFirst:
findLast:
```

Many functions are Lacking :
findFirst:ifNone:
findLast:ifNone:
…

- To write them all would be painful
- *try{call anyFindFunction}*
  catch (ItemNotFound e) {…}

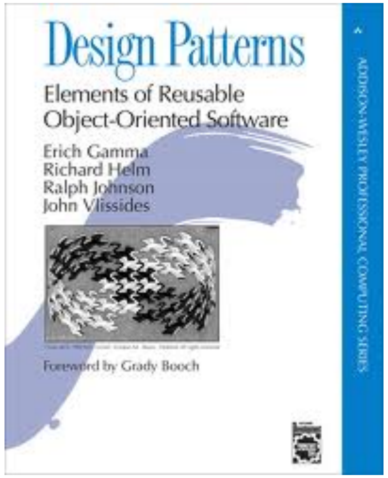
  is better
- …

What can we do?

- P1 : Discuss the term “exception” …
- P2 : Convince developers that built-in solutions are less powerful than EHS
- P3 : Propose patterns.
  - Unified set of constructs?
  - Mainframe languages?
- P4 : Suggestions to improve Java as a mainframe language for EHS

Design Pattern need : unified basic constructs AND operational languages

- Classes, Composition
- Inheritance
- Message Sending
- C++, Smalltalk



- *[WB06] R.J. Wirfs-Brock. Toward exception-handling best practices and patterns.*

- No unified construct set
  - See next slide
- Which mainframe language?

? Exception Handling Design Patterns ?

Do we have operational unified construct set?

- We do not !

- [GRRX01] Garcia, Rubira, Romanovsky, Xu. A comparative study of exception handling mechanisms for building dependable object-oriented software.

Taxonomy Aspects	Design Decisions \ Exception Mechanisms	Ada 95	Lore	Smalltalk	Eiffel	Modula3	C++	Java	Delphi	Guide	Ext. Ada	BETA	Arche
A1. Exception Representation	Symbols	-1		-1	-1	-1				-1	-1		
	Data Objects						+1	+1	+1				+1
	Full Objects		+1									+1	
A2. External Exceptions in Signatures	Unsupported	-1		-1	-1				-1			-1	
	Optional		0				0						0
	Compulsory					+1				+1	+1		
	Hybrid							+1					
A3. Separation between Internal and External Exceptions	Unsupported	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	Supported												
A4. Attachment of Handlers (+)	Statement	+1	+1			+1	+1	+1	+1	+1		+1	+1
	Block	-1				-1	-1	-1	-1				-1
	Method				+1					+1		+1	
	Object										+1	+1	
	Class	+1	+1	+1	+1					+1	+1	+1	
	Exception		+1									+1	
A5. Handler Binding	Static			-1							-1	-1	
	Dynamic												
	Semi-Dynamic	+1	+1		+1	+1	+1	+1	+1	+1			+1
A6. Propagation of Exceptions (+)	Unsupported			-1							-1	-1	
	Automatic	-1	-1		-1	-1	-1	-1	-1				
	Explicit	+1	+1			+1	+1	+1	+1	+1			+1
A7. Continuation of Control Flow (+)	Resumption		-1	-1								-1	
	Termination	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1
A8. Cleanup Actions	Unsupported										-1		
	Use of Explicit Propagation	0			0		0						0
	Specific Construct		+1	+1		+1		+1	+1	+1		+1	
	Automatic Cleanup												
A9. Reliability Checks (+)	Dynamic Checks	+1	+1	+1	+1	+1	+1	+1	+1	+1			+1
	Static Checks				+1	+1	+1	+1	+1	+1	+1	+1	+1
A10. Concurrent Exception Handling	Unsupported		-1	-1	-1	-1	-1		-1	-1	-1	-1	
	Limited	0						0					
	Complete												+1
	Final Score	1	5	-3	1	3	3	6	3	7	-1	3	6

Challenges

- Continue to write language independent patterns (various proposals - papers, web)
  - http://c2.com/cgi/wiki?ExceptionPatterns
  - Impact somehow low without unified construct set

- Establish next mainframe languages integrating complete and well-designed EHS …
  - Difficult

- Suggest adaptations to today's mainframe language,
- Influence the next mainframe languages …

What can we do?

- P1 : Discuss the term “exception” …
- P2 : Convince developers that built-in solutions are less powerful than EHS
- P3 : Propose patterns.
  - Unified set of constructs?
  - Mainframe languages?
- P4 : Suggestions to improve Java as a mainframe language for EHS

Problems with Java EHS as a mainframe for EH

- Good news : with JAVA people do use EHS
  - With benefit in many cases
  - Java EHS globally sound, simple to use and efficient
- But, various issues
  - Misuses related to “checked exceptions”
  - Lack of control structures
    - At least a “retry”
  - Classification problematic

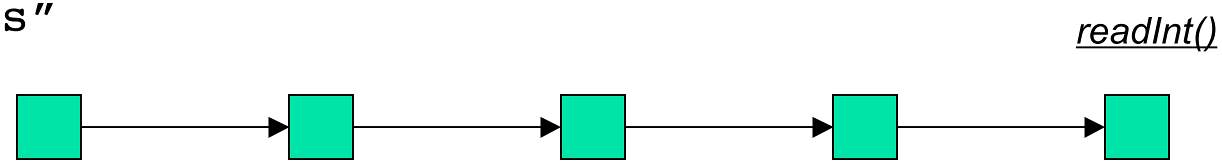
Standard misuses

- *[RS03] D. Reimer and H. Srinivasan. Analyzing exception usage in large java applications - ECOOP03 workshop on EHS*

- swallowed exceptions

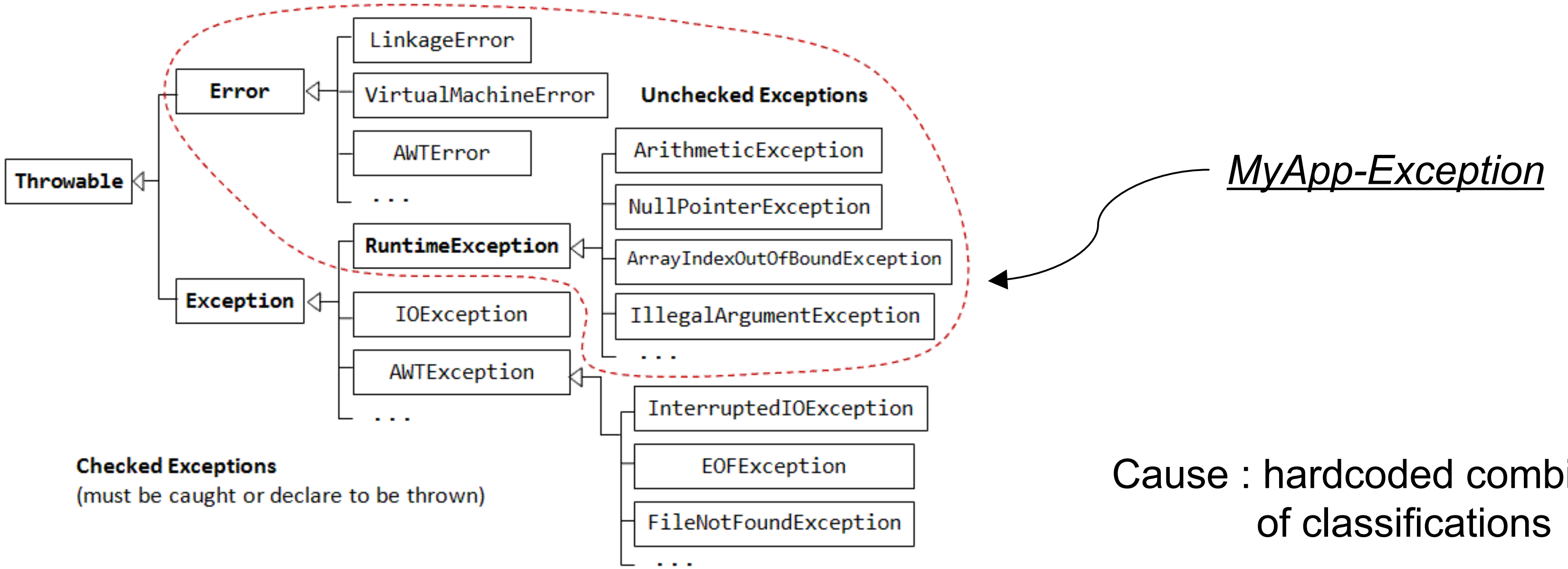
```java
public int readInt() {
    BufferedReader keyboard;
    try {
        keyboard = new BufferedReader(new FileReader("truc"));
    } catch (FileNotFoundException e1) {}
    // ...
```

  - Handler that neither log, rethrow nor handle exceptions (simplest version : empty catch blocks)
  - Standard reason : stop writing “throws clauses”



Standard Misuses …

- And by extension :
  - Dev. tend not to use libraries that throw exceptions
  - Dev. tend to badly classify their own exception kinds



Cause : hardcoded combination of classifications

Solutions to the “swallowed exceptions” misuse?

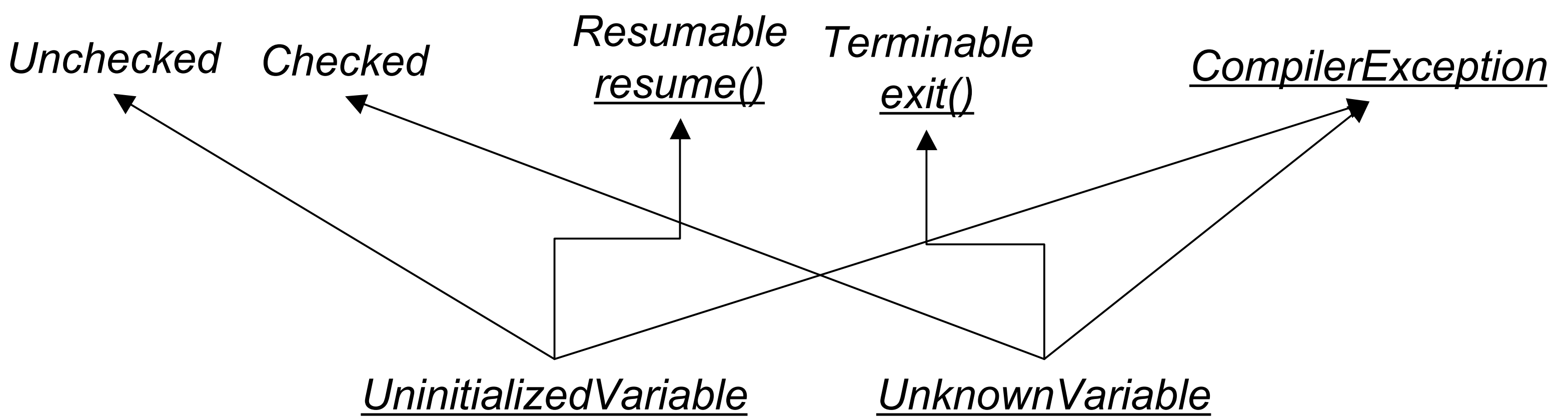
- Monitor programmers?
- Relax checking rules?
  - *“Unhandled exception type FileNotFoundException”*
    - *Could be a warning*
    - Could be considered at package level instead of method level

- *“throws clauses”* could be automatically generated
    - By the compiler
    - By the IDE (see workshop presentations on exception flow tools -)

- Provide for decoupling classifications …

Decouping classifications
… one of the most challenging issue

- Use whatever known technique
  - Meta-classes, aspects, annotations, multiple-inheritance, mixins, MDE, …
- To decouple and combine all necessary classifications
  - Ontological, reuse-based, properties-based …



P4 : Adding more control structures

- At least a “retry”.

```
int ilu = 0;
boolean succes = false;
while (! succes) {
    try {ilu = Integer.parseInt(keyboard.readLine());
         succes = true;}
    catch (NumberFormatException e){
        System.out.println("Error : " + e.getMessage());
        System.out.println("Please try again! ");}
} // end while
return ilu;
```

```
int ilu = 0;
    try {ilu = Integer.parseInt(keyboard.readLine());}
    catch (NumberFormatException e)
        { System.out.println("Error : " + e.getMessage());
          System.out.println("Please try again! ");
          retry(); // or e.retry();}
return ilu;
```

Some challenges for Exception Handling

- C1 : Toward usages, best practices and patterns.
  - Convince that EH is necessary and useful
  - Improve today mainframe EHS languages (Java, …?)
    - Problems? misuses? solutions?
- C2 : Abstraction, Efficiency, Reuse : rediscovering lost ideas
  - Architecture level handlers
  - Exception handling as a dialog (resumption, restarts)
- C3 : Build new EHS for the new world and using the new world
  - components, aspects, services, ambient, ubiquitous, concurrent
  - Example of MAS
- C4 : Orthogonal dimensions
  - Cover the life cycle
  - Combination of tools and techniques
    - Example of exception handling and replication

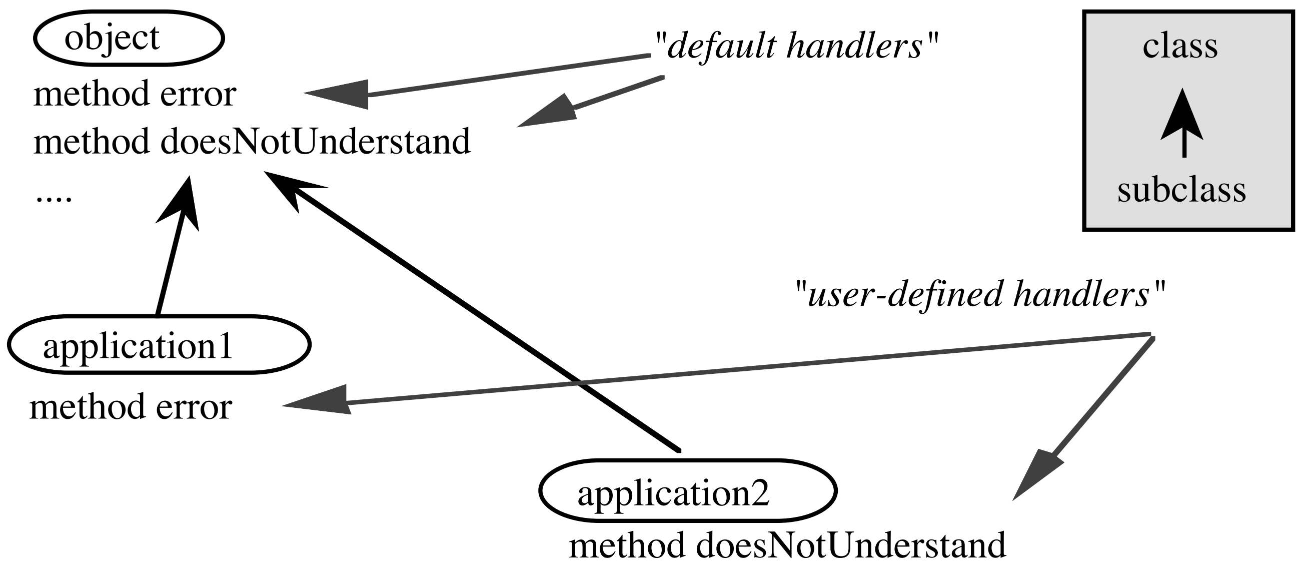
Architecture level handlers …

- Specify Exception Handling policies at any place within a software architecture
  - Now a common research subject
    - *[IB01] Issarny, Banatre. Architecture-based exception handling.*
    - *Filho, Brito, Rubira. Specification of exception flow in software architectures.*
- <u>Lost</u> idea : Class-level handlers
  - Simple and Useful solution for basic OOP

- Key issue was …
  - to combine class-level with block-level handlers

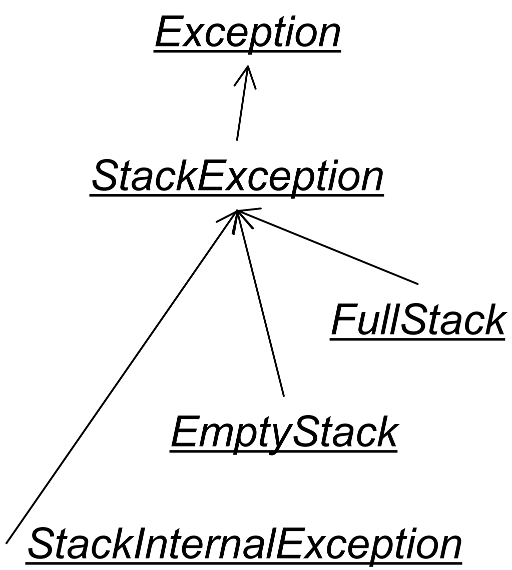
Architecture level handlers :
the case for class-level handlers

- Inspired from historical Smalltalk lexical scope class-level handlers



Class-level handlers

- A short application example




```
Stack
    when: #(FullStack EmptyStack)
    do: ‘:e | e signal’
    when: Exception
    do: 'e: | StackInternalException signal’.
```

Java syntax simulation

```
Class Stack extends Object{
    catch(FullStack e) {throw e;}
    catch(EmptyStack e) {throw e;}
    catch (Exception e) {throw new StackInternalException()}
    …
```

Class-level handlers

- Work in combination with block-level thanks to
  - A <u>dynamic-scope</u> policy
  - a “callee-caller based” handler search

- Simple to implement (could be done with annotations, aspects)
- Manageable at design time (UML classdiagrams)
- Introduce exception-based reuse schemes

```
Class GrowingStack extends Stack{
   catch(FullStack e) {this.grow(). e.retry();}
   public void grow() {…}
```

Resumption and Restarts

- Resumption policy …
- Restarts : solution for dialogs between signallers and handlers in order not to restart exception halted computation from scratch
  - *[Moon86] Moon. Object-oriented programming with flavors.*
  - *[Pit01] K. Pitman. Condition handling in the Lisp language family.*
  - *Ported to Smalltalk [Don01] C. Dony. A fully object-oriented exception handling system: rationale and smalltalk implementation.*
  - Everyday-life : most problems solved by dialog
  - Interests
    - Interactive applications
    - Task collaborative applications
    - Dialog based (web) client-server applications
    - Ubiquitous computing

Signalers can establish restarts cases

```
(defun my-symbol-value (name)
  (if (boundp name)
      (symbol-value name)
      (restart-case (error 'unbound-variable :name name)
        (use-value (value)
          :report "Specify a value to use."
          :interactive (lambda ()
                         (format t "~&Value to use: ")
                         (list (eval (read))))
          value)
        (store-value (value)
          ….
          )))
```

signaling

one "restart"

another "restart"

Argument returned by the handler

If to be used interactively

Handlers can choose restarts

- Resume the execution at a restart point

A handler for "unbound-variable" exception

Send control back to the signaler

```
(handler-bind ((unbound-variable
                #'(lambda (e)
                    (let ((restart (find-restart use-value e)))
                      (if restart
                        (invoke-restart restart 2)
                        (throw e))))))
  (* 4 (+ 3 x)))
= 20
```

application to interactive applications

- If not handled, the exception restarts can be use by a debugger

```
(+ x 3)
 Error: The variable THIS-SYMBOL-HAS-NO-VALUE is unbound.
 Please select a restart option:
   1 - Specify a value to use.
   2 - Specify a value to use and store.
   3 - Return to Lisp toplevel.
   4 - Exit from Lisp.
 Option: 1
 Value to use: 2
 => 5
```

Some challenges for Exception Handling

- C1 : Toward usages, best practices and patterns.
  - Convince that EH is necessary and useful
  - Improve today mainframe EHS languages (Java, …?)
    - Problems? misuses? solutions?

- C2 : Abstraction, Efficiency, Reuse : rediscovering lost ideas
  - Architecture level handlers
  - Exception handling as a dialog (resumption, restarts)

- C3 : Build new EHS for the new world and using the new world
  - components, aspects, services, ambient, ubiquitous, concurrent
  - Example of MAS

- C4 : Cover orthogonal dimensions
  - Cover the life cycle
  - Combination of tools and techniques
    - Example of exception handling and replication

Building new EHSs for the new world
Many researches and results

Examples

- Agents
  - *[SDUV04] Souchon, Dony, Urtado, Vauttier. Improving exception handling in multi-agent systems.*
- Components
  - *[RdLFF05] Rubira, De Lemos, Filho. Exception handling in the development of dependable component-based systems.*
- Services
  - *[TIRL03] Tartanoglu, Issarny, Romanovsky, Levy. Dependability in the Web services architecture.*
- Aspects
  - *[CCF+ 09] Castor, Cacho, Figueiredo, Garcia, Rubira, de Amorim, da Silva. On the modularization and reuse of exception handling with aspects.*
- …

Building new EHSs for the new world
Many researches and results …

Examples

- Exceptions at the software architecture level
  - *[IB01] Issarny, Banatre. Architecture-based exception handling.*
  - *Filho, Brito, Rubira. Specification of exception flow in software architectures.*
- Ambient systems
  - *[MDB+06] Mostinckx, Dedecker, Boix, Van Cutsem,De Meuter. Ambient-oriented exception handling.*
- Pervasive systems
  - *[MECL10] Mercadal, Enard, Consel, Loriant. A domain-specific approach to ar- chitecturing error handling in pervasive computing.*
- Product Lines Architectures
  - *[BDBR08] Bertoncello, Dias, Brito, Rubira. Explicit exception handling variability in component-based product line architectures.*
- …

C3 : Build new EHS using the new world Abstraction, Modularization, Reuse

- Advances coming with the component world … an example

Extracted from : [MECL10]

```
context SmokeDetected as Boolean
  indexed by location as Location {
  source smoke from SmokeDetector [skipped catch];
}
context AverageTemperature as Temperature
  indexed by location as Location {
  source temperature from TemperatureSensor [mandatory catch];
}
context FireState as Boolean
  indexed by location as Location {
  context SmokeDetected [mandatory catch];
  context AverageTemperature [no catch];
}
```

C3 : Build new EHS using the new world
Simplification, abstraction, modularization, reuse

- Advances coming with the aspect world … an example

Extracted ffom : [CCF+ 09]

```
public aspect GOHandler { // another source file
  pointcut crsHandler() :
      execution(public static boolean closeResultSet(..));
  boolean around(ResultSet rs) : crsHandler() && args(rs){ //
      advice
    try { return proceed(rs);
    } catch (SQLException e) { System.out.println(e.toString());
      return false;
    }
  }
  declare soft : SQLException : crsHandler();
}
```

Some suggested meta-rules to build new EHS in new contexts

- Provide for "propagation of locally unhandled exceptions to callers"
  - If any "software contract" [meyer 88] broken, tell the caller.
- Execute caller handlers in the caller environment

Caller contextualization

All kind of lexical scope handlers are unused

- Consider software architectures
  - mix block-level handlers and architecture level handlers

- …

Some suggested meta-rules to adapt EHS to new contexts … contd.

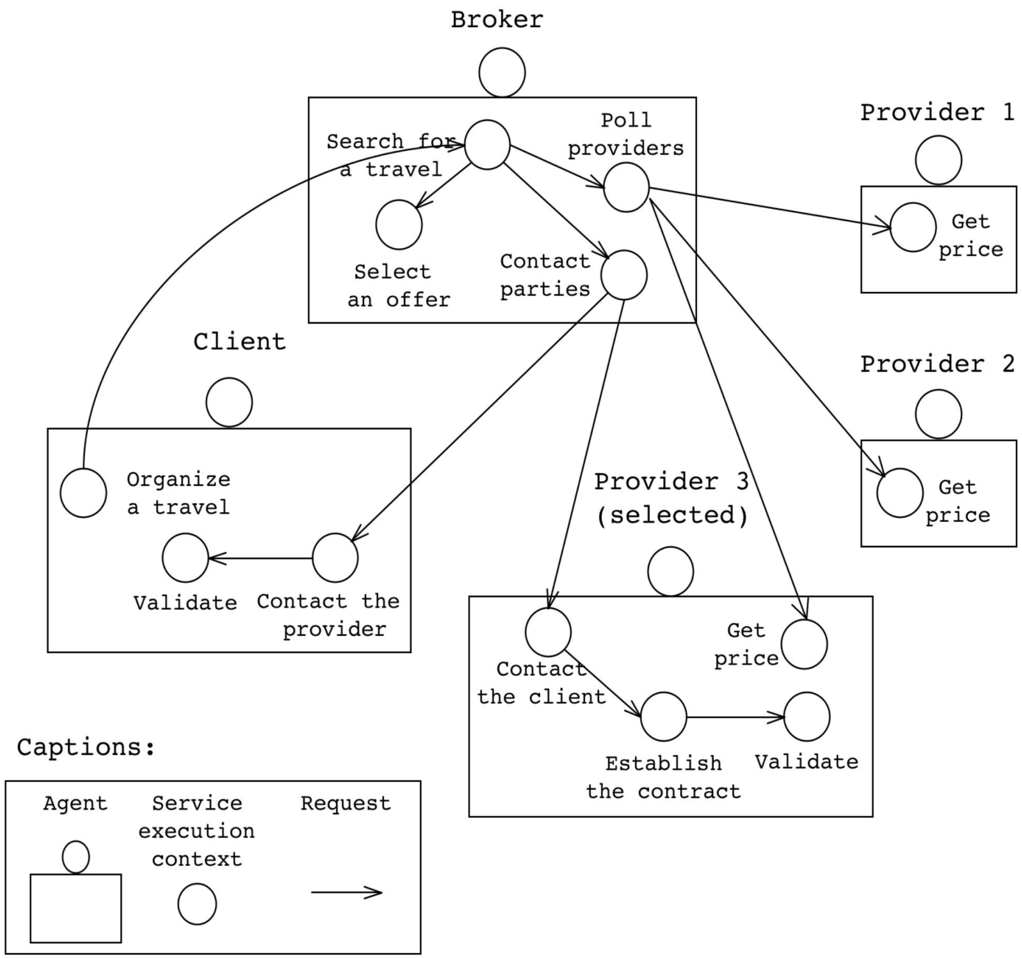
- …
- Provide for a simple to use mode
  - Complex features are rarely used
- Respect the philosophy of the destination paradigm
- Reuse appropriate existing works

Discussing the above meta-rules on an EHS for MAS programmers

- Work with Christelle Urtado and Sylvain Vauttier (LGI2P EMA)
  - *[DUV06] Christophe Dony, Christelle Urtado, and Sylvain Vauttier. Exception handling and asynchronous active objects: Issues and proposal.*

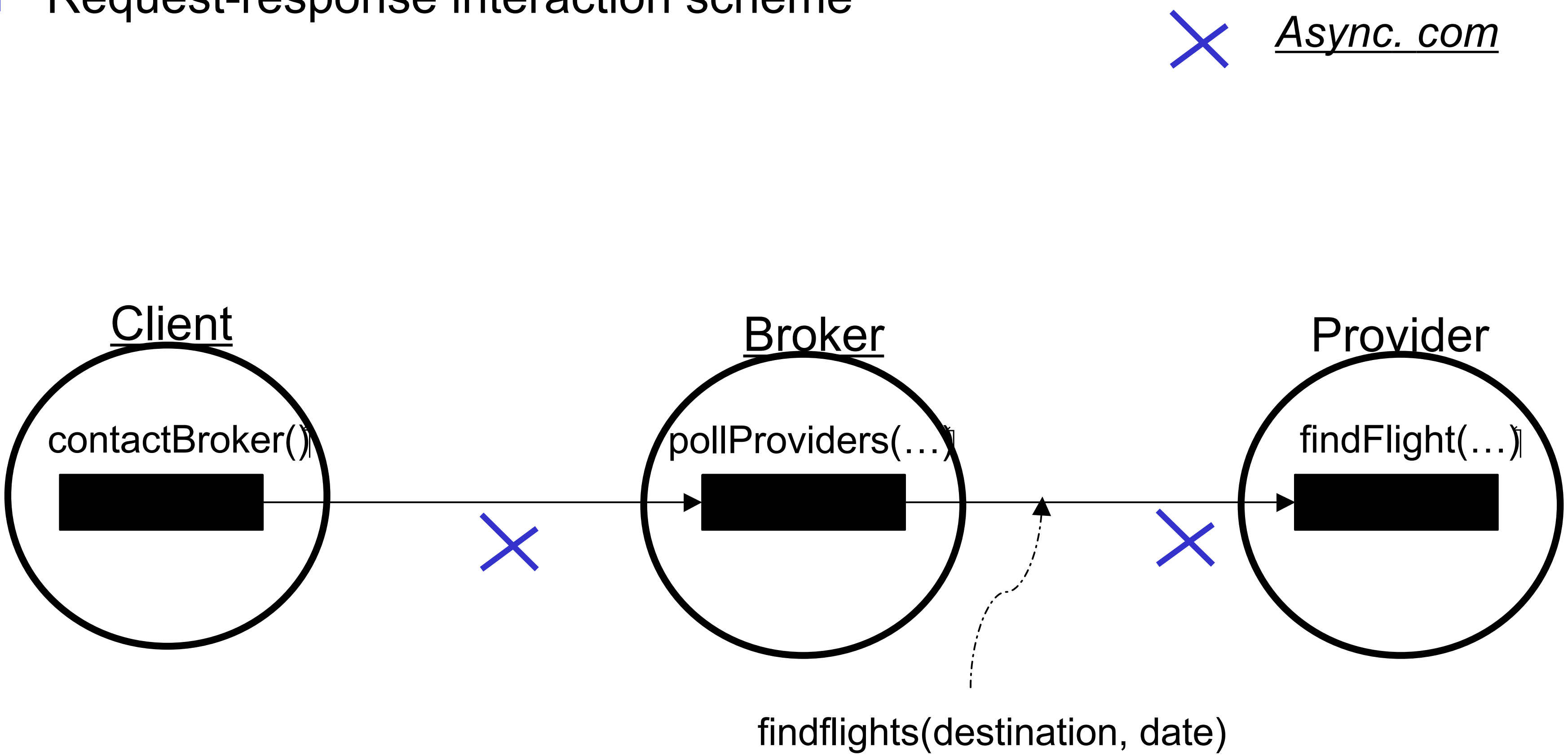
- Agents :
  - Reactive
  - Autonomous
  - Collaborate through Asynchronous request-response interaction protocol
    - Middleware independent
  - External and Internal concurrency
    - One thread to read the mbox
    - One thread for each service method) execution

A running example …



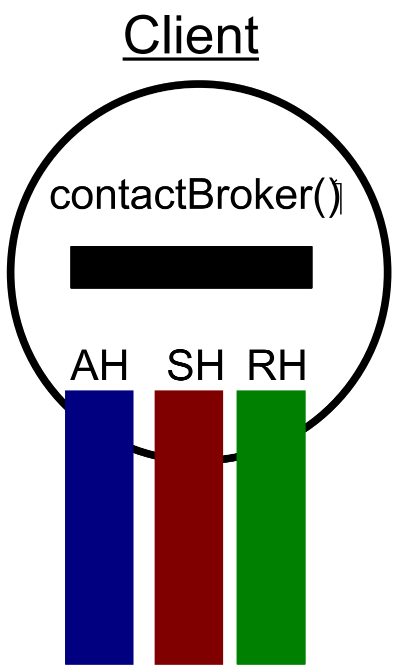
Agent autonomy and reactivity

- Request-response interaction scheme

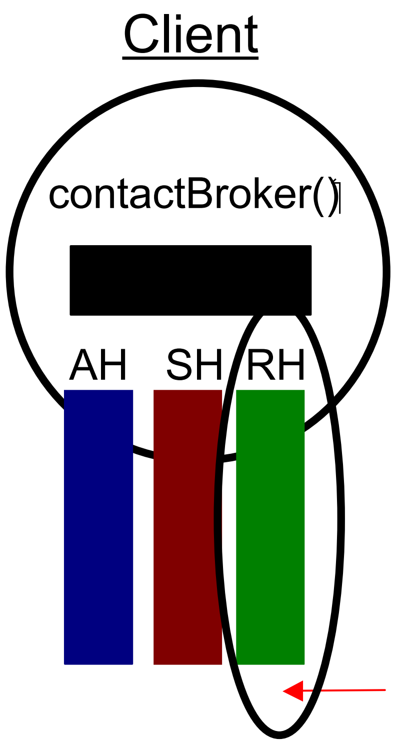


… Think to architecture

- Agents level (AH), Services level (SH), Request level (RH) handlers



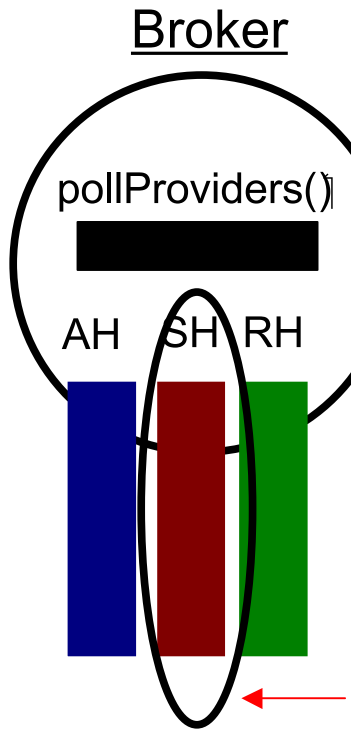
Example of a request level handler …




```
public class Client extends X-SaGEAgent

    @service
    public void contactBroker (…) {
        ...
        sendMessage
            (new RequestMessage
                (aBrokerAgent,
                ”PollProviders”,
                destination,
                date)
            {@requestHandler
                public void handle (NoAvailablePlaces exc){
                    date = date +- 1;
                    retry();}
        });
        ...
    }
```

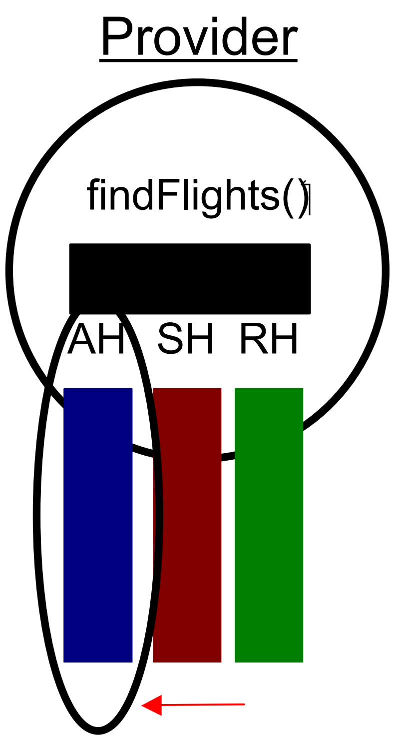
Example of a service level handler

```
public class Broker extends X_SaGEAgent  {

    @service
    public void pollProviders (destination date) {
        ... }

    @serviceHandler(servicename=pollProviders)
    public void handle (NoAirportForDestination exc) {
        signal(exc);}
```



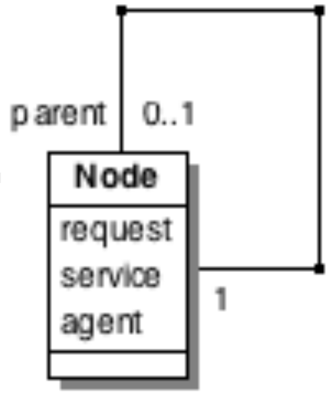
Example of agent level handlers

```
//Trap all low-level technical exceptions
// signals a higher-level one

@handler
   public void handle(NetworkConnectionException e){
              signal(new TemporaryTechnicalProblem(…));}
@handler
     public void handle (DatabaseConnectionException e){
              signal(new TemporaryTechnicalProblem(…));}

public class Provider extends X_SaGEAgent  {
   …

     }
```

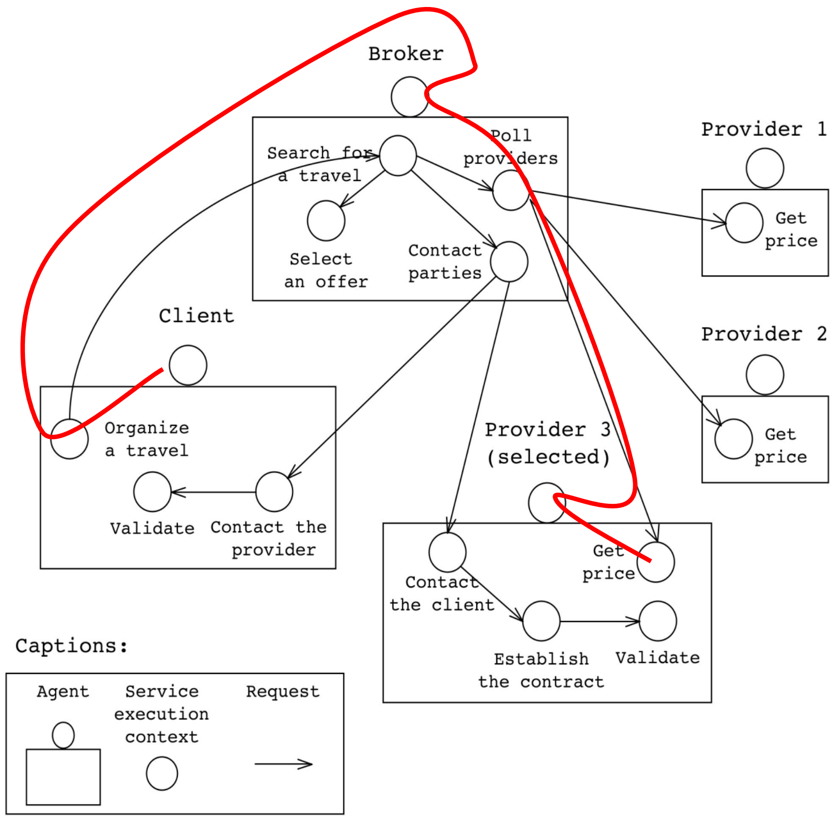


… Provide for “caller contextualization”

- Dynamically maintain the request tree



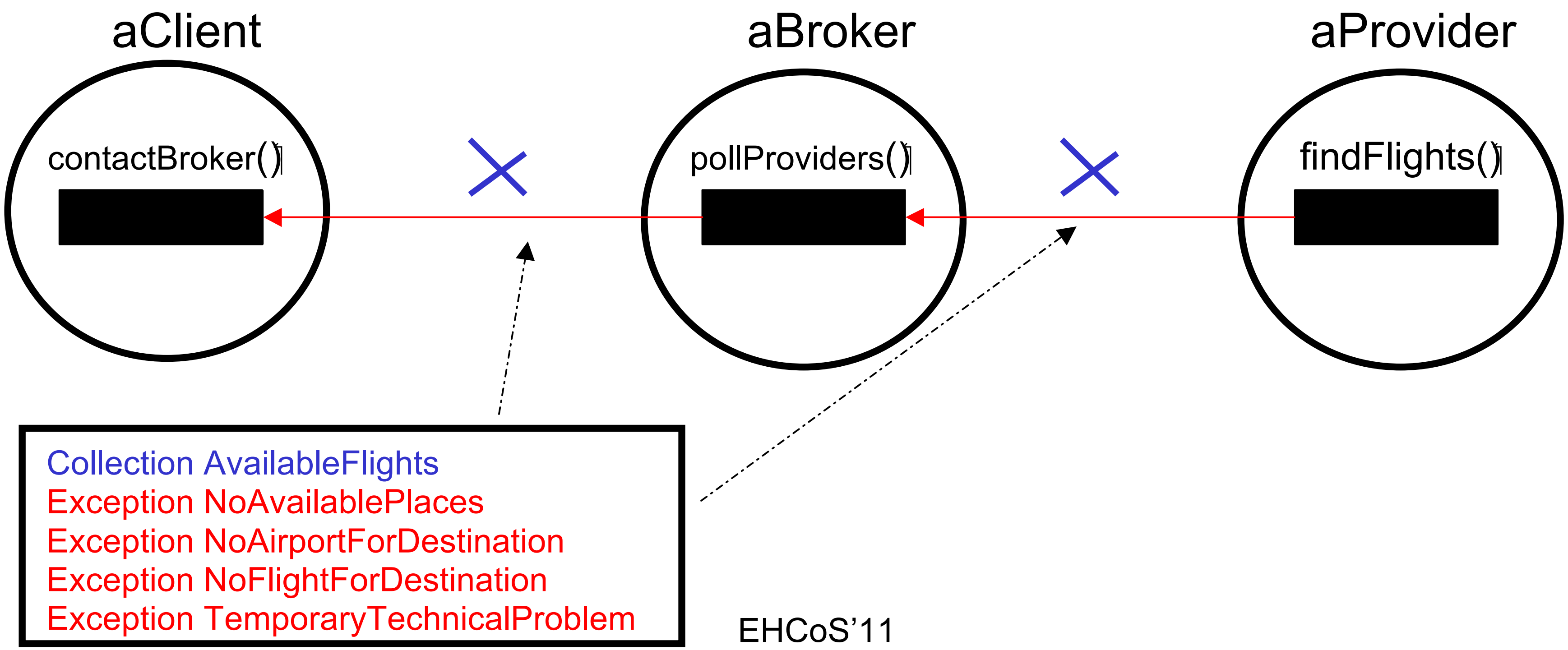
- Do not follow the idea of independent “exceptions supervisors”
  - [Klein, Dellarocas 99] : **Supervisors**
- Propagate exceptions through the call chain
- Take into account all kind of handlers



… Respect the philosophy of the destination paradigm

- Maintain agent autonomy and reactivity
  - By using the standard asynchronous response mechanism
  - For normal or exceptional responses

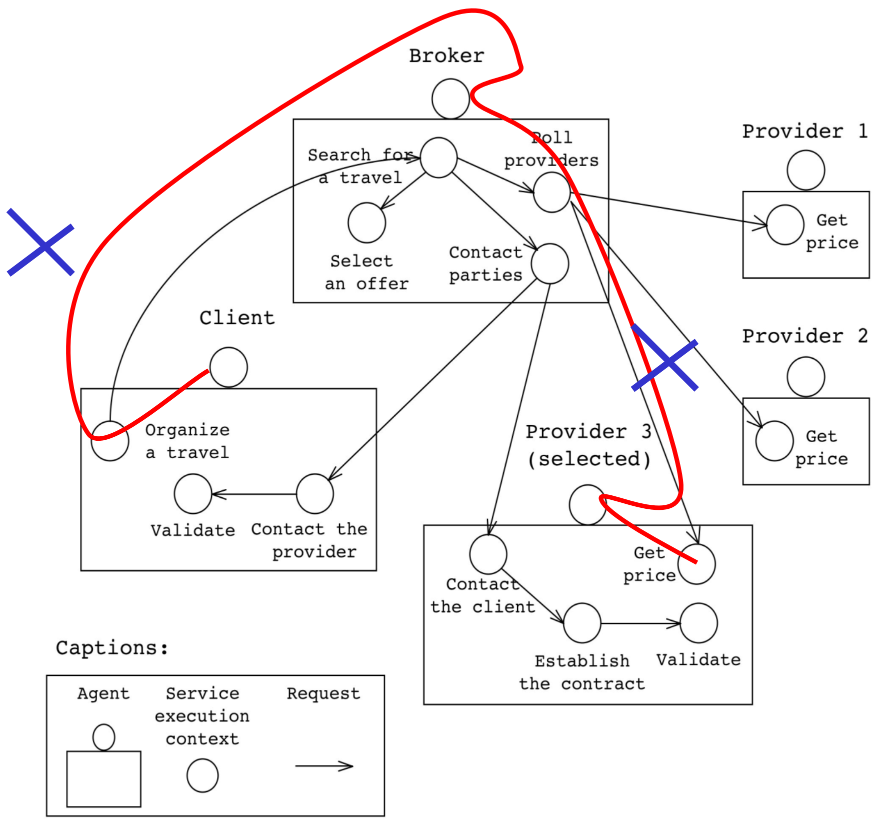
async. com.



… Respect the philosophy of the destination paradigm

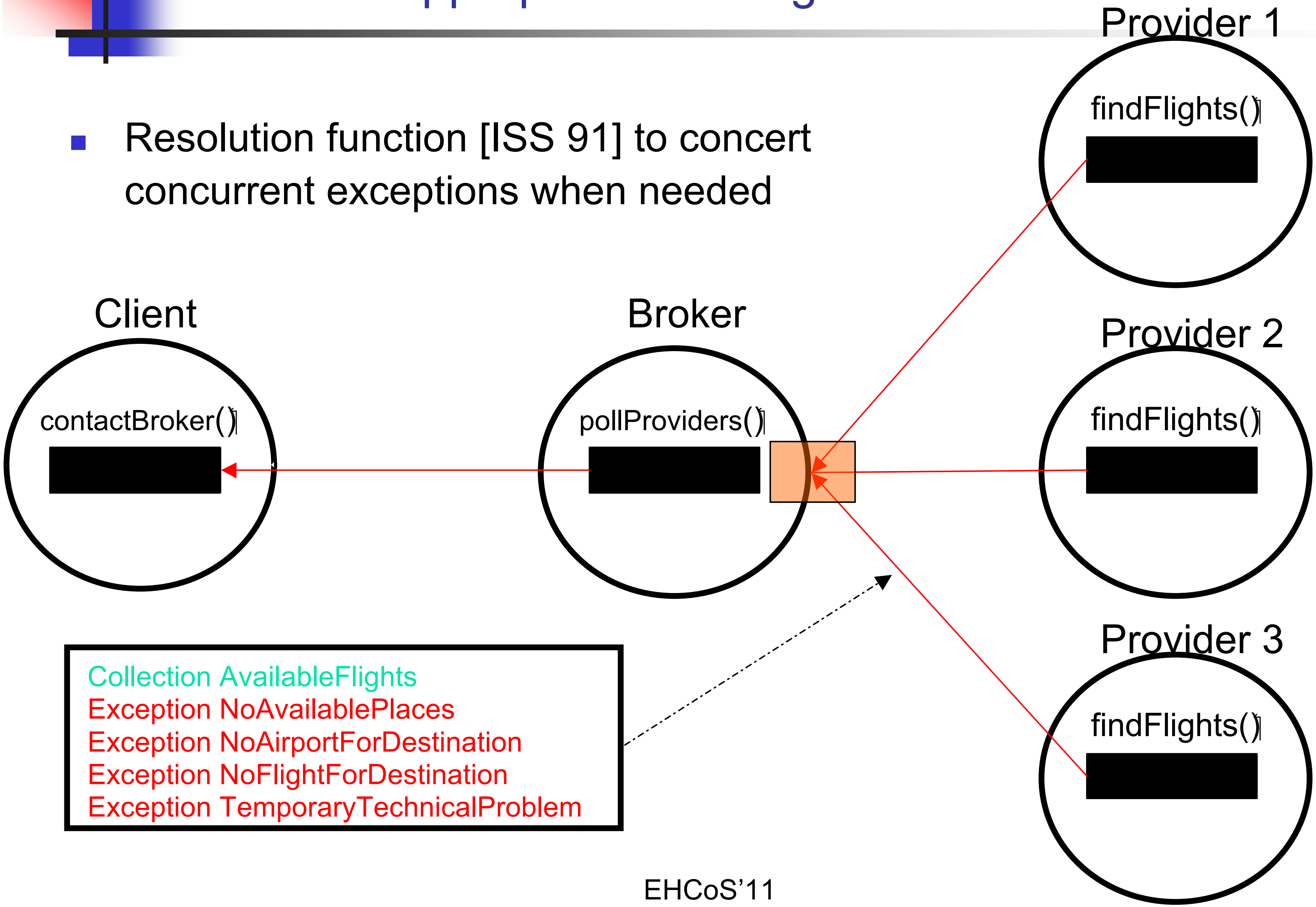
- maintain agent autonomy and reactivity
  - asynchronous progagation of exceptions between agents

async. com.



… Reuse appropriate existing works

- Resolution function [ISS 91] to concert concurrent exceptions when needed



An example of a resolution function

- Resolution function
  - invoked each time an exception signaling reaches a complex service, before invoking a potential handler
  - In this example, used to control n-versions providers

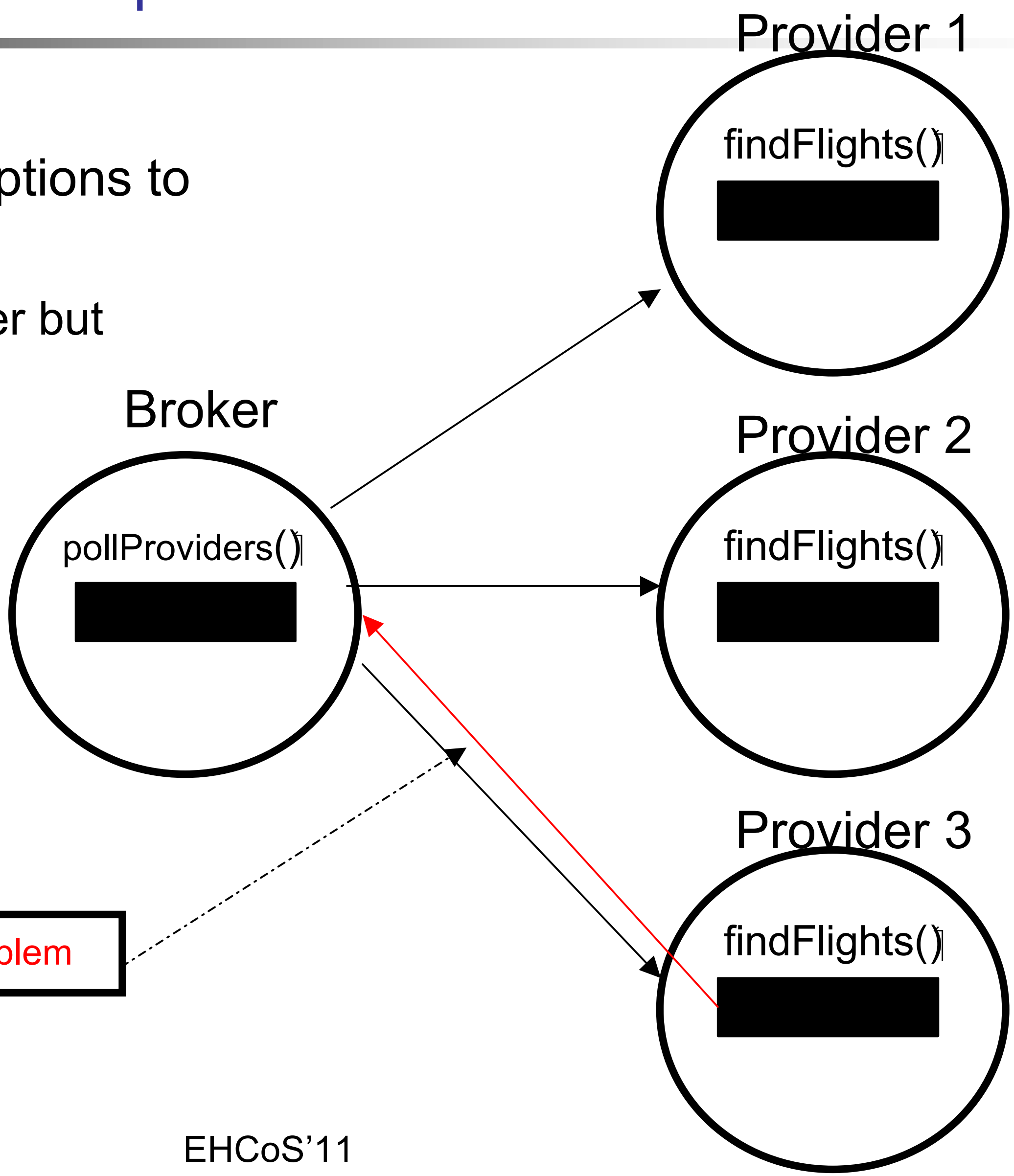
```
@serviceResolutionFunction(servicename=pollProviders)
public Exception concert(Exception e) {
  //log e
  //log current failing sub-service

  //example of decision
  if ((numberOf(NoAvailablePlaces)
     >=
     0.8 * numberOf(subServices))
    return e;
    else
    return null;
}
```

Signaling continue

Signaling stops

… Provide for a simple to use mode

- Do not propagate exceptions to "brother" agents
  - Less expressive power but simpler to use



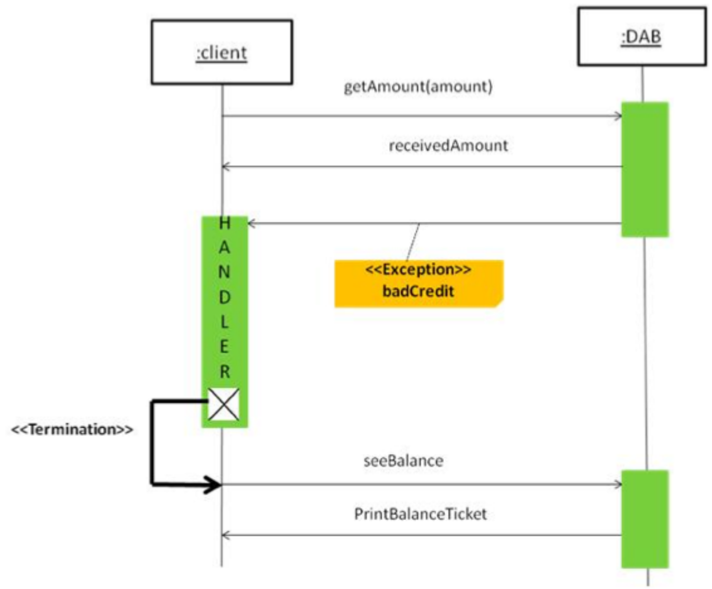
Exception TemporaryTechnicalProblem

Some challenges for Exception Handling

- C1 : Toward usages, best practices and patterns.
  - Convince that EH is necessary and useful
  - Improve today mainframe EHS languages (Java, …?)
    - Problems? misuses? solutions?
- C2 : Abstraction, Efficiency, Reuse : rediscovering lost ideas
  - Architecture level handlers
  - Exception handling as a dialog (resumption, restarts)
- C3 : Build new EHS for the new world and using the new world
  - components, aspects, services, ambient, ubiquitous, concurrent
  - Example of MAS
- C4 : Orthogonal dimensions
  - Cover the life cycle
  - Combination of tools and techniques
    - Example of exception handling and replication

Covering the life cycle
various researches and results …

- [dLR01] de Lemos, Romanovsky. Exception handling in the software lifecycle.
- [SMKD07] Shui, Mustafiz, Kienzle, Dony. Exceptional Use Cases.
- [HH06] Halvorsen, Haugen. Proposed notation for exception handling in uml 2 sequence diagrams.
- …



Combination of tools and techniques an example : exception handling and replication

- Collaboration
  - Paris VI University- LIP6 - INRIA-REGAL
    - Jean-Pierre Briot, Zahia Guessoum, Olivier Marin, Jean-François Perrot
      - Dima agent Framework (guessoum&al 06)
      - DarX replication system (Marin&al 03-06)
  - Montpellier-II University - LIRMM
    - Christophe Dony, Chouki Tibermacine
  - Ecole des Mines d'Ales - LGI2P
    - Christelle Urtado, Sylvain Vauttier

Combination of exception handling and replication

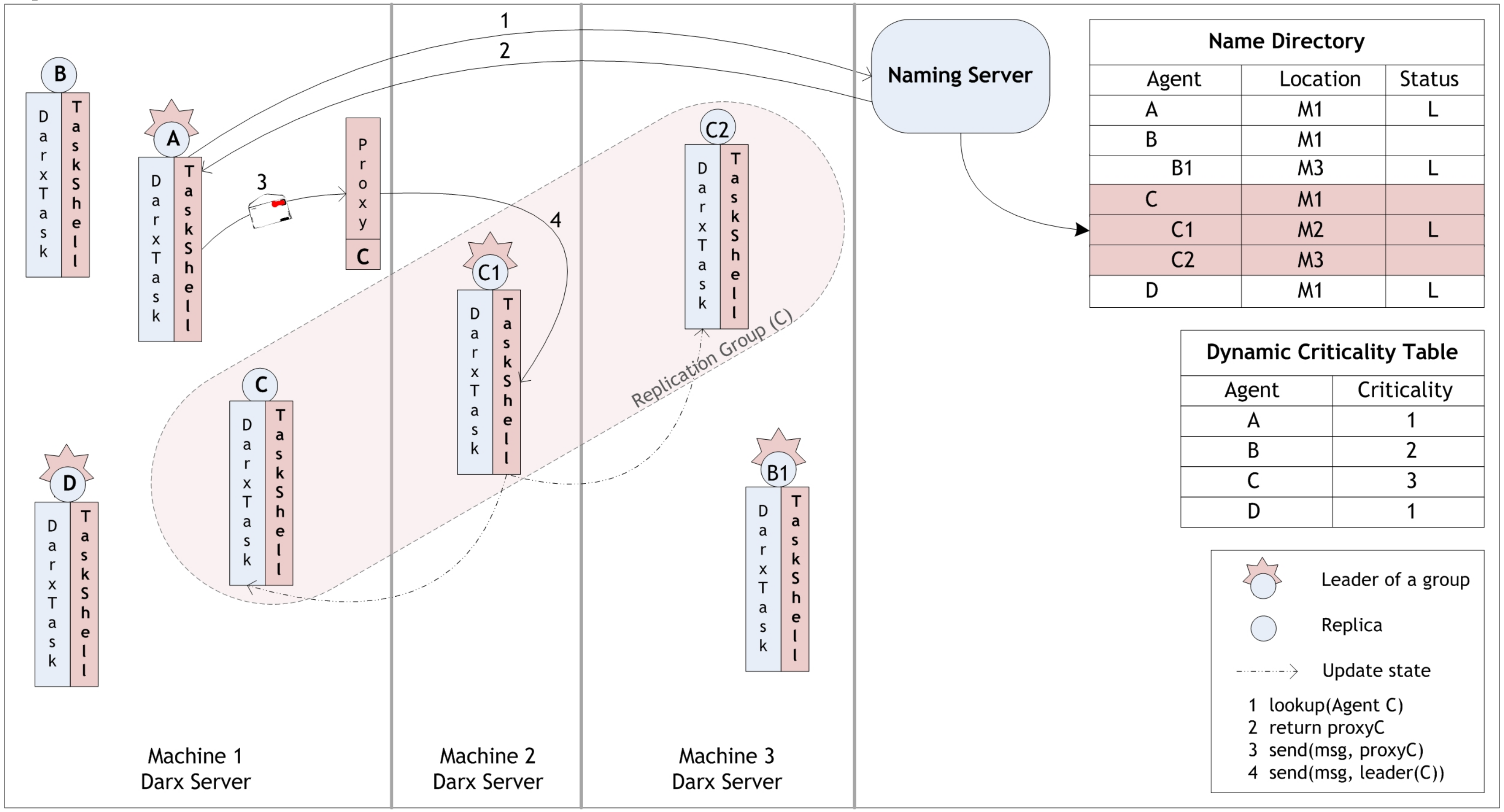
- Simple idea :

- 1) A transparent replication systems
  - handles
    - System (replica-spefic) fault or exception
      - ▶ E.g. *NetworkConnectionException*

- 2) A combined EHS :
  - allow programmers to deal with
    - Business (replica-independent) exceptions
  - Improves the efficiency of the replication level

Yet another classification!

A replication system (DARX)



Name Directory

Agent	Location	Status
A	M1	L
B	M1	
B1	M3	L
C	M1	
C1	M2	L
C2	M3	
D	M1	L

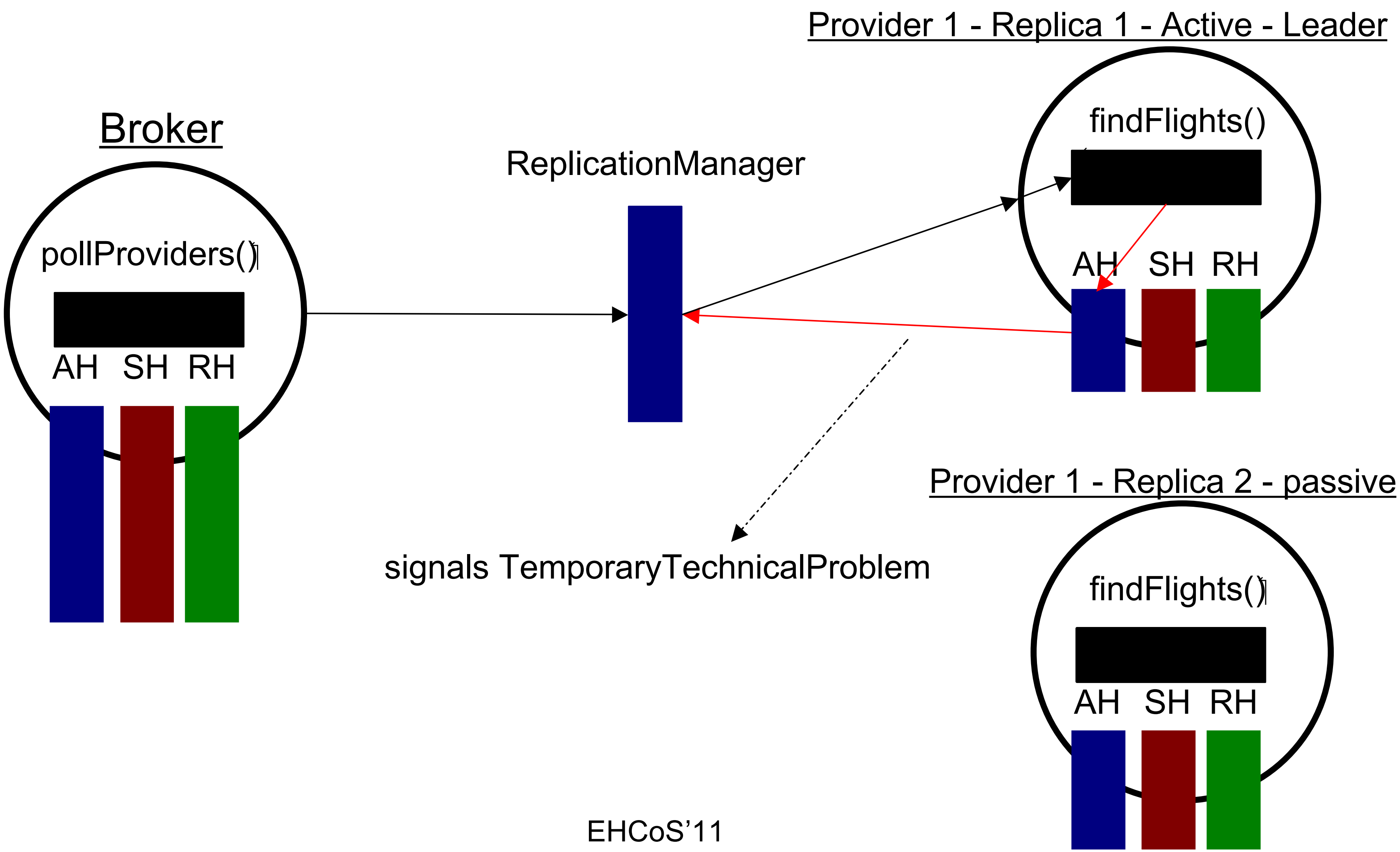
Dynamic Criticality Table

Agent	Criticality
A	1
B	2
C	3
D	1

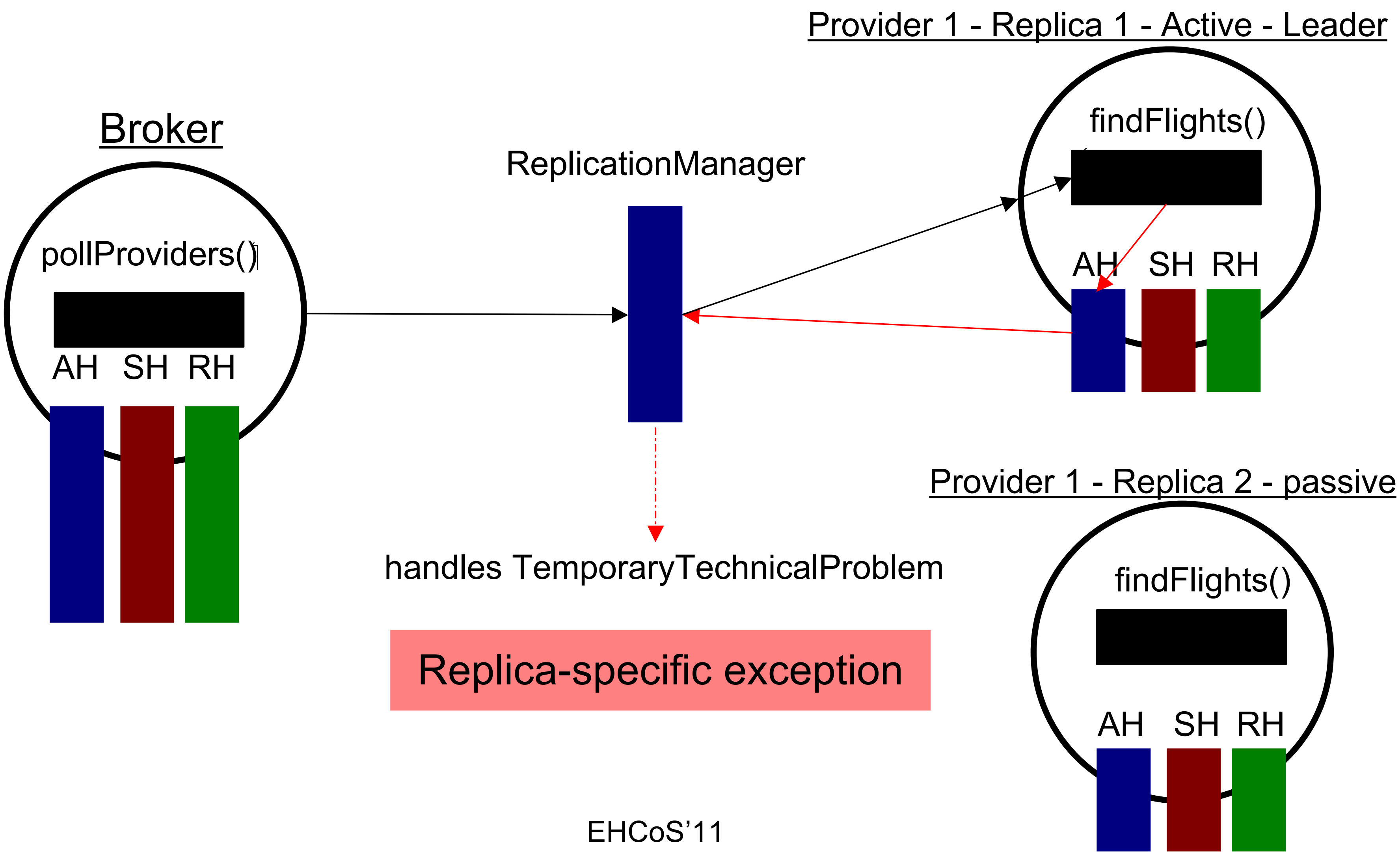
Schema by Zeine Azmeh

Criticality, replication group, leader, active and passive replicas …

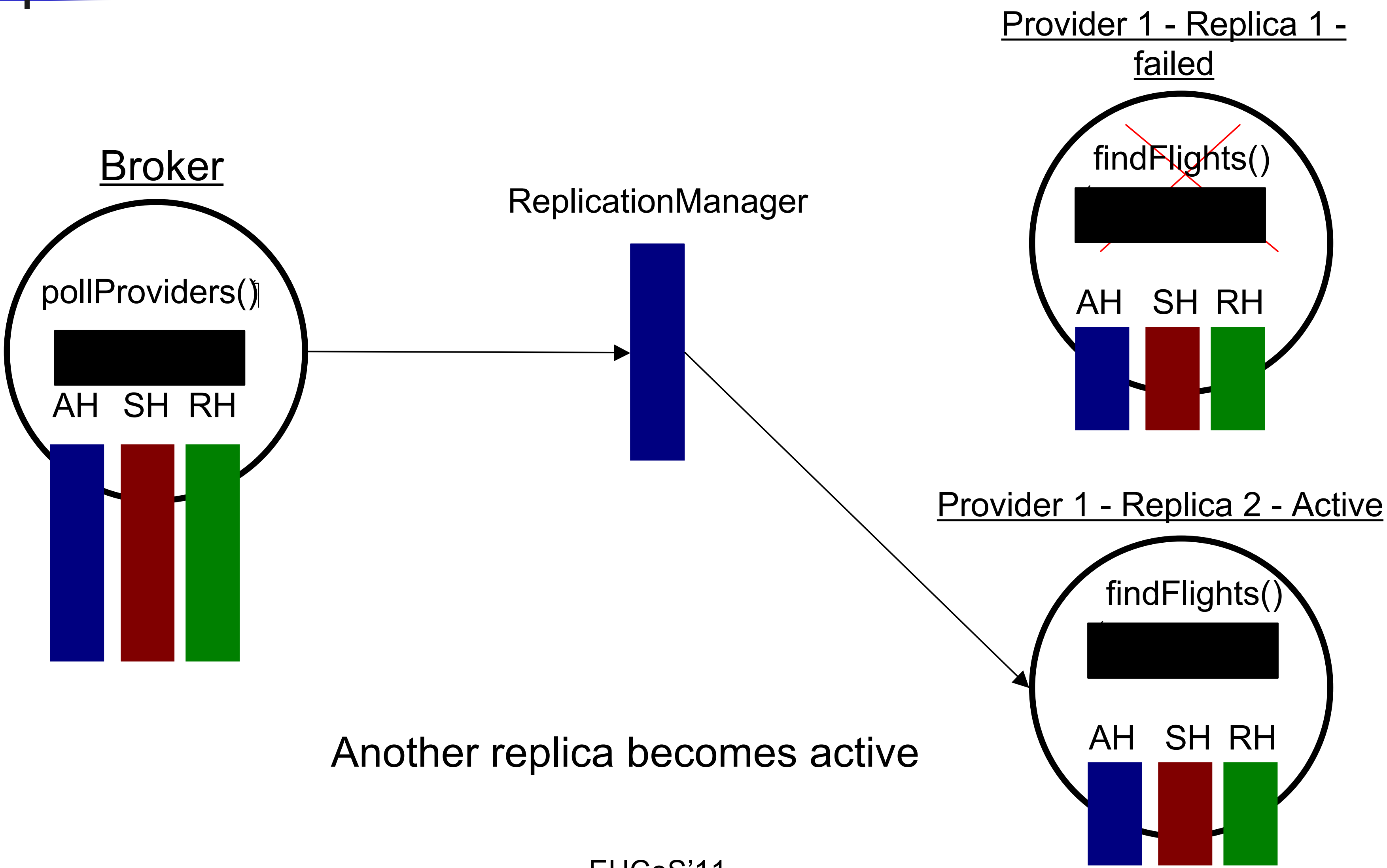
Controlling replicated agents : Replica-specific exception (1)



Controlling replicated agents : Replica-specific exception (2)

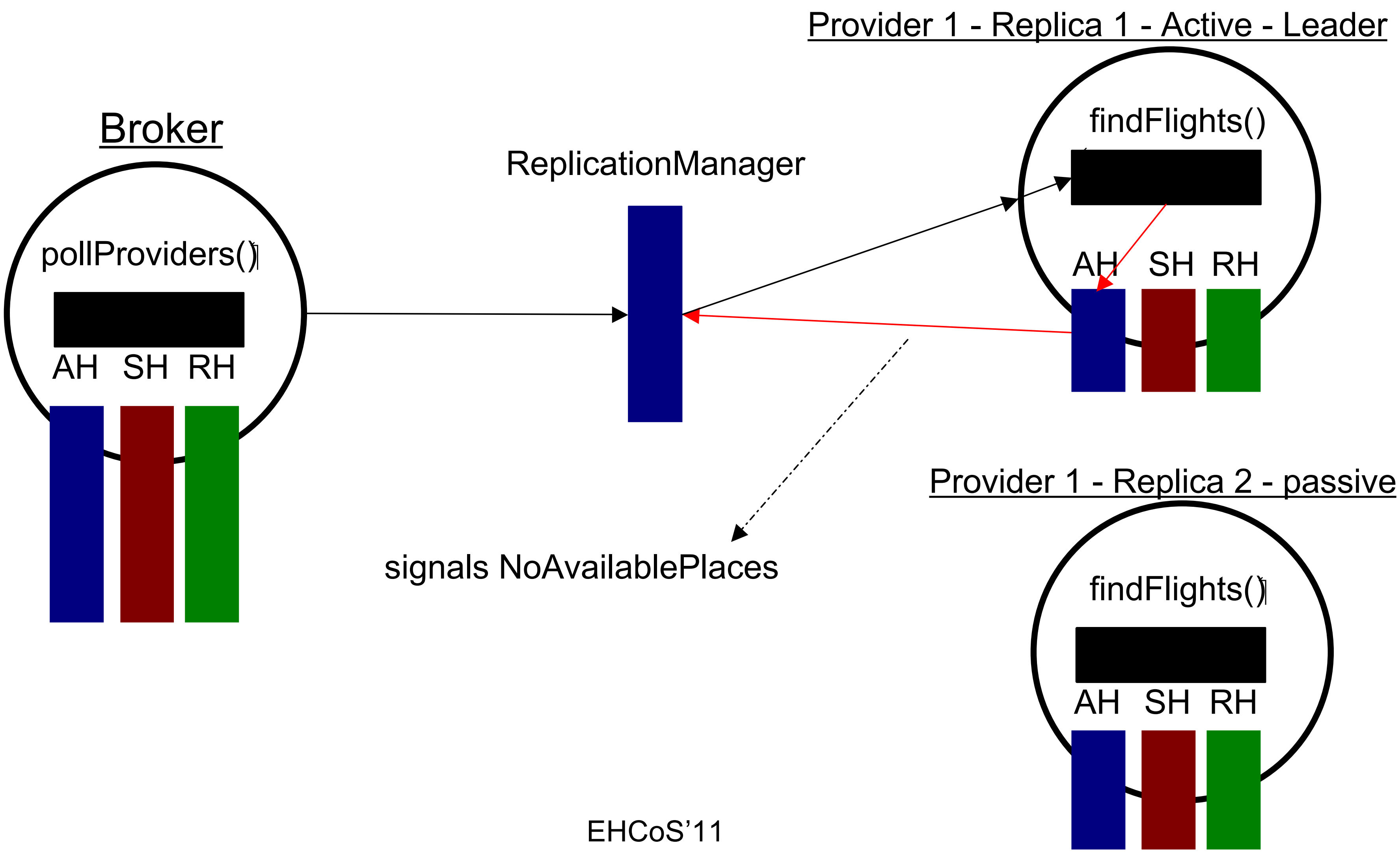


Controlling replicated agents : Replica-specific exception (3)

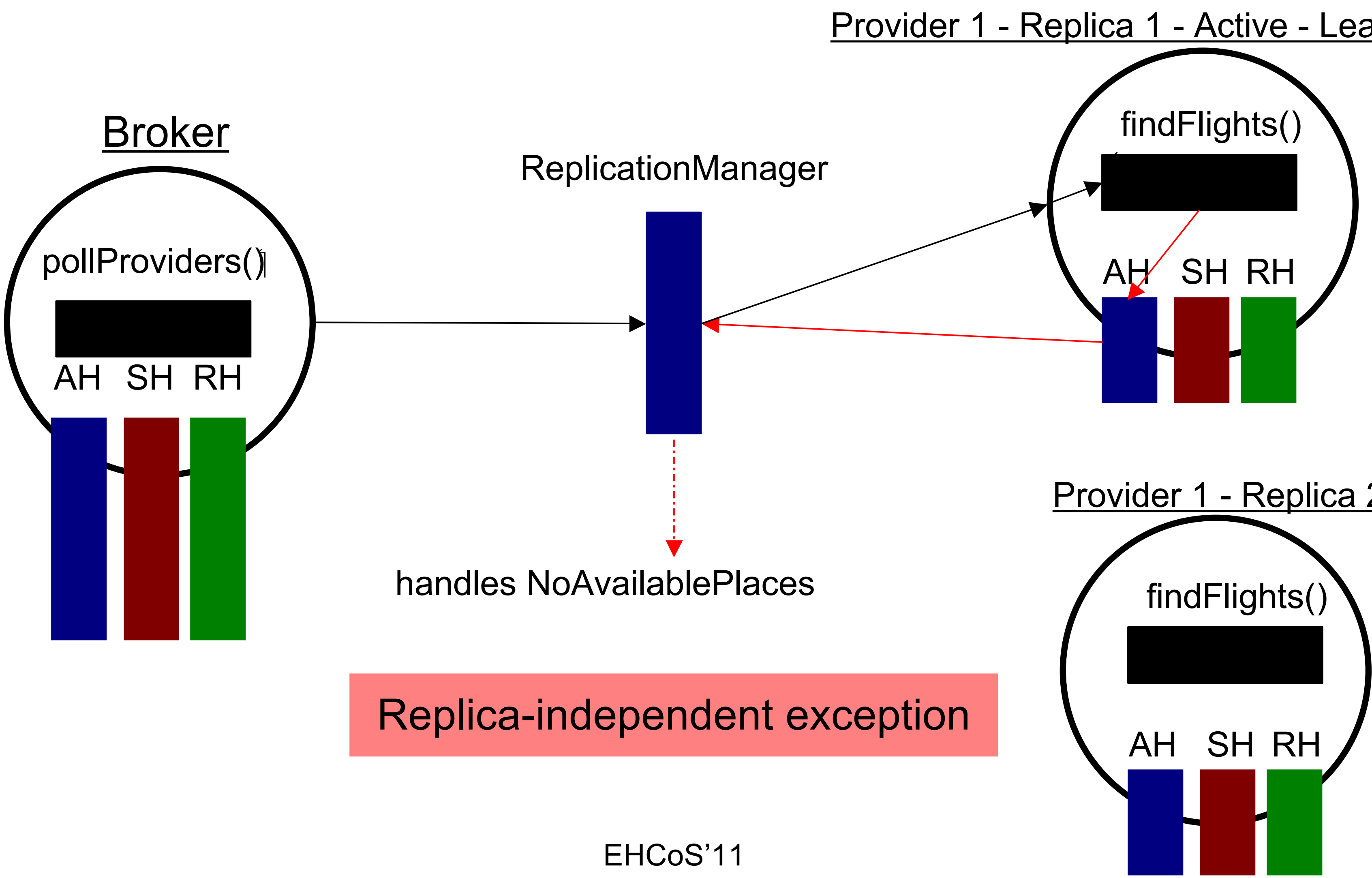


Another replica becomes active

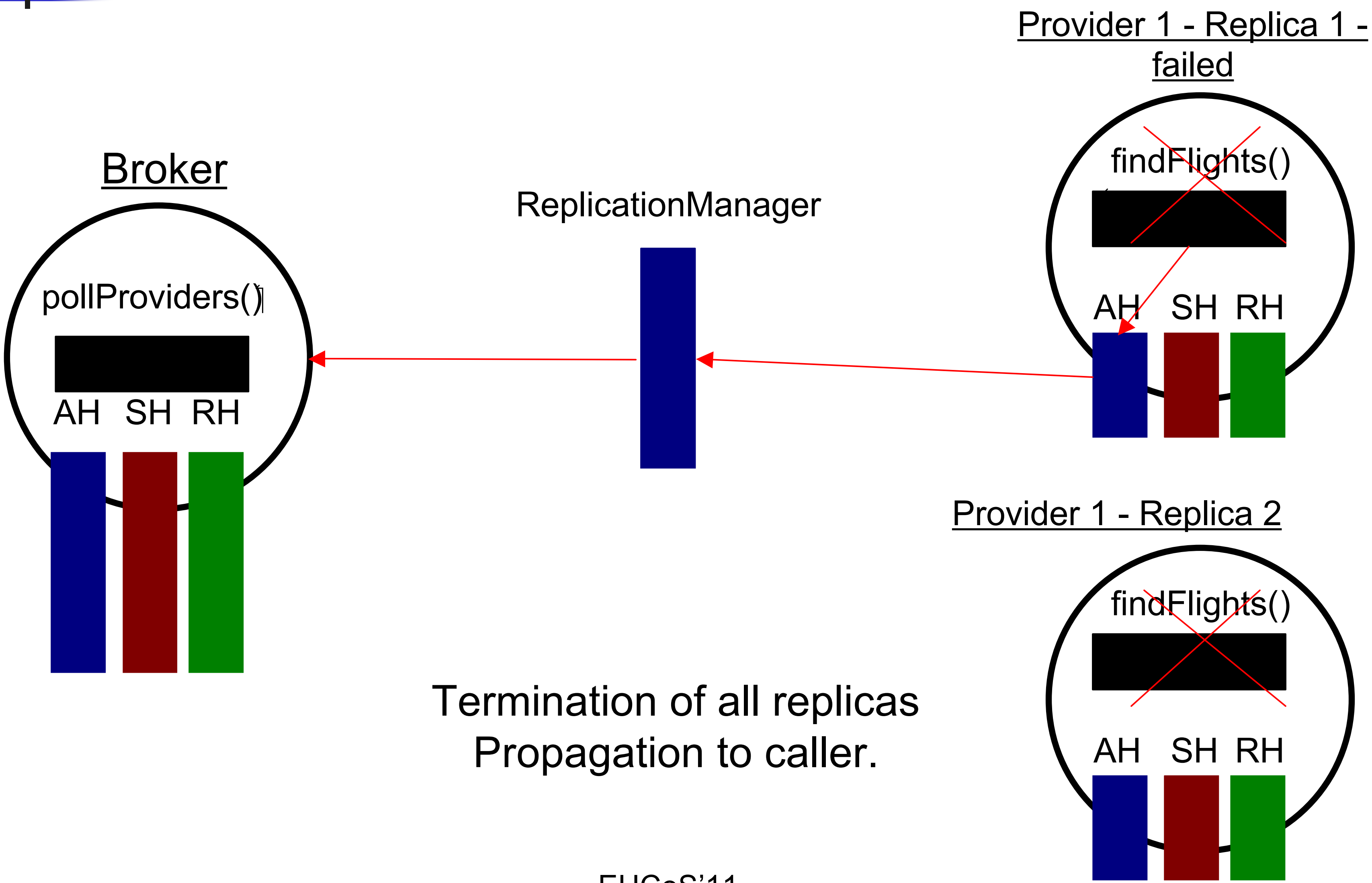
Controlling replicated agents : Replica-independent exception (1)



Controlling replicated agents : Replica-independent exception (2)



Controlling replicated agents : Replica-independent exception (3)



Termination of all replicas
Propagation to caller.

Conclusion
and opening discussion …

- "Uncontinuable" as a new name for what was called "exception"
- Usages, best practices, patterns
  - Need more mainframe languages
- More Modularity, Reuse, Expressive power, high-level abstractions
  - Too many research ideas left unexploited
  - New solutions come with new paradigms
    - components,
    - Aspects … annotations
    - Models
- Also …
  - Adaptability (domain specific EH ?)
  - Check, prove, reason on programs that handle exceptions

References

- See the associated abstract paper :
  http://www.lirmm.fr/~dony/postscript/exc-AbstractEHCOS.pdf