Focussed

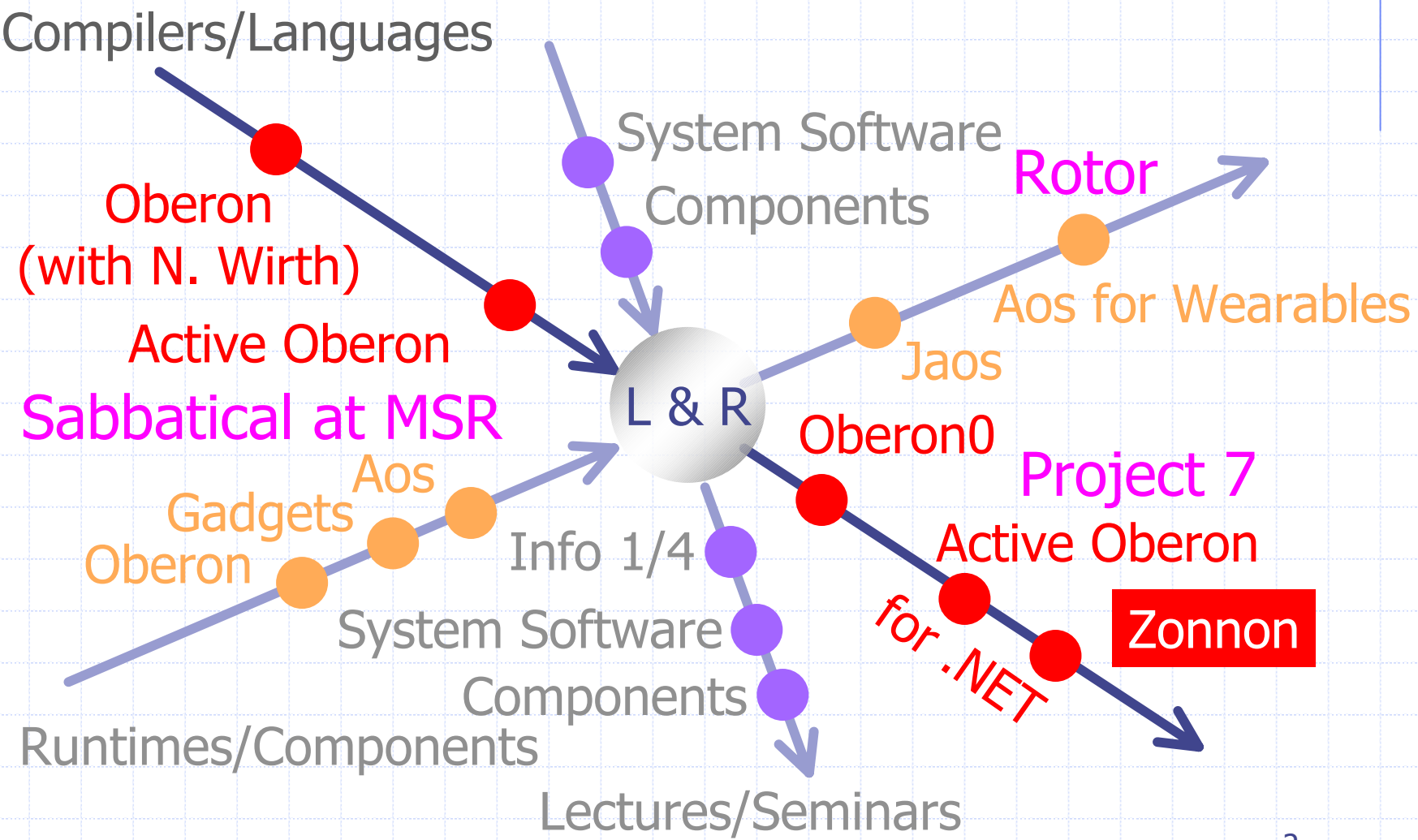
Zonnon: A Language & Compiler Experiment on the Basis of .NET

Jürg Gutknecht & Eugene Zueff
ETH Zürich

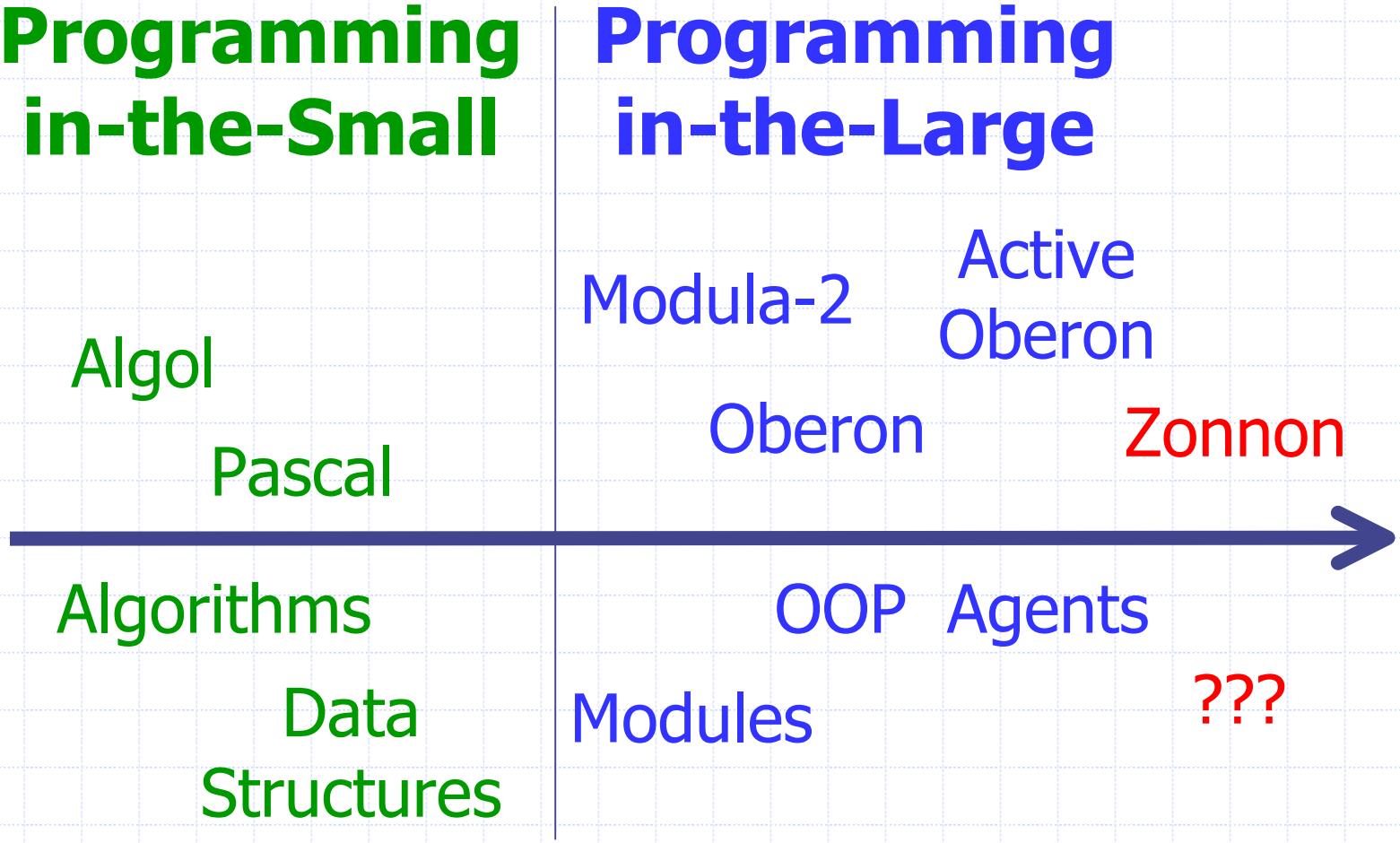
Compiler

JMLC 2003, August 25-27

Language & Runtime Projects



Spectrum of Language Evolution



The Zonnon Language Project

- **Goals**
  - **Improve and upgrade the ordinary OO model**
    - Seamlessly integrate additional "Programming in the Large" issues
    - Conceptually support remote objects and distributed systems
  - **Provide a Pascal family language for .NET**
    - Enable teaching of algorithms & data structures without OO corset
    - Bridge the gap from the Pascal Age to 3rd Millenium technology

A Sample Zonnon Program

```
MODULE RandomNumbers;
  VAR z: INTEGER { 32 }; (*global variable*)
  PROCEDURE { PUBLIC } Next(): REAL;
    CONST a = 16807; m = 2147483647;
      q = m DIV a; r = m MOD a;
    VAR g: INTEGER { 32 };
  BEGIN
    g := a*(z MOD q) - r*(z DIV q);
    IF g > 0 THEN z := g ELSE z := g + m END;
    RETURN z*(1.0/m)
  END Next;
BEGIN z := 31459
END RandomNumbers.
```

- Modula-2 ?
- Oberon ?
- Zonnon ?

The same considering Programming in the Small

A .NET Interoperability Sample

```
OBJECT SmoothSort IMPLEMENTS SortControls.SortPanel;
   (*superfast sorting algorithm, invented by Dijkstra*)
   PROCEDURE { PUBLIC } Sort (VAR a: ARRAY OF REAL)
    IMPLEMENTS SortControls.SortPanel.sort;
      PROCEDURE Up(VAR b, c: INTEGER);
      PROCEDURE Down(VAR b, c: INTEGER);
      PROCEDURE Sift(r, b, c: INTEGER);
      PROCEDURE Trinkle(r, p, b, c: INTEGER);
      PROCEDURE SemiTrinkle(r, p, b, c: INTEGER);
   BEGIN ...
   END Sort
END SmoothSort;
```

C# is a Good Language, but ...

- What can be improved?
  - Composability
  - Concurrency
  - Communicativity

C# is a Good Language, but ...

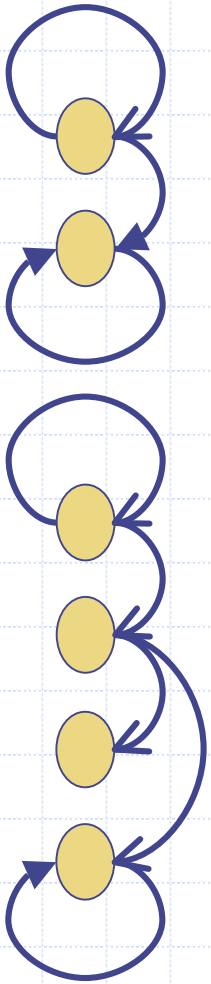
- What can be improved?
  - Composability
  - Concurrency
  - Communicativity

Building Blocks & Relations

- **C#**
  - Interfaces
  - Classes
- **Zonnon**
  - Definitions
  - Implementations
  - Object Types
  - Modules

compile

run



extends :n

implements :n

inherits :1

refines :1

implements :1

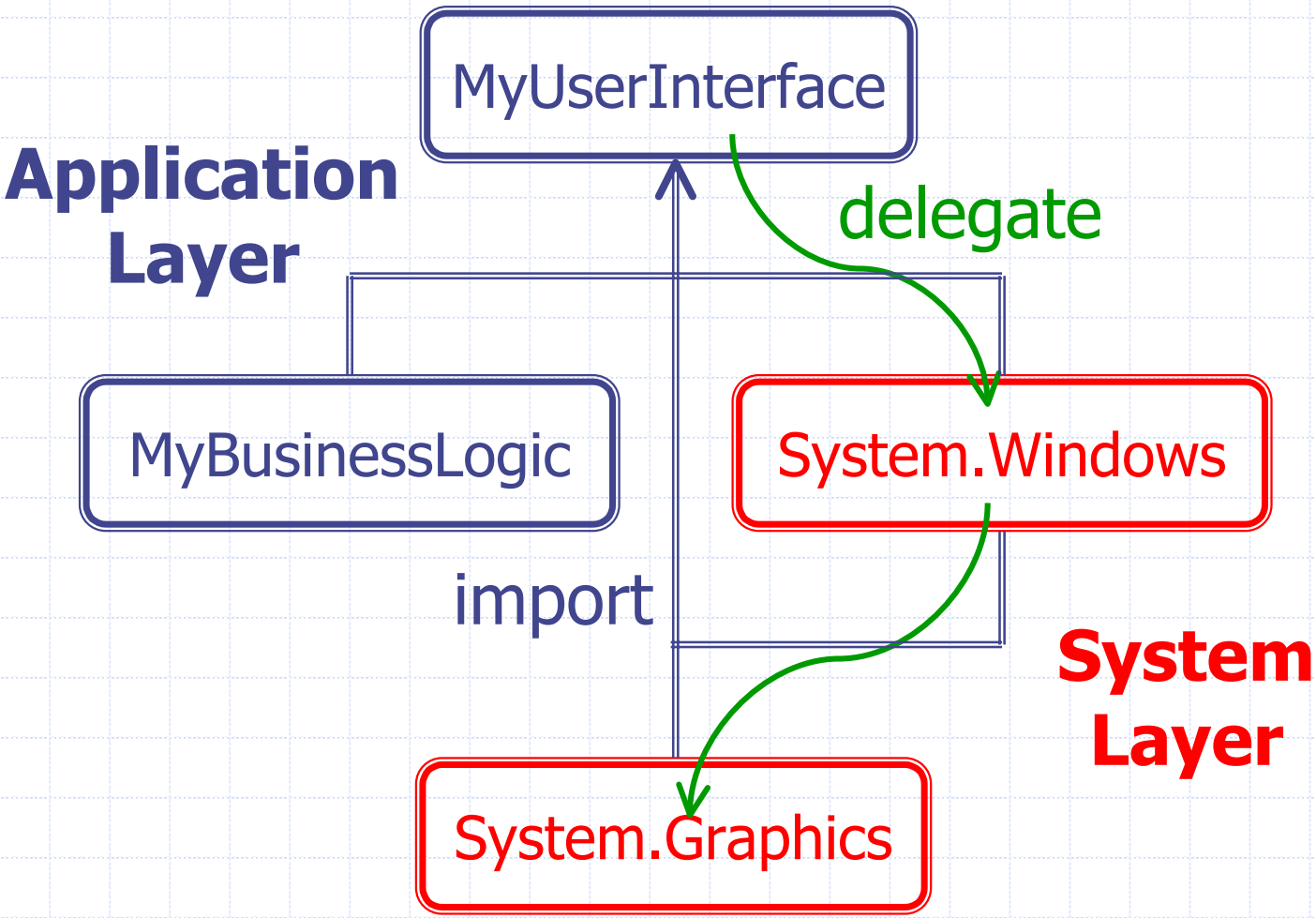
aggregates :n

imports :n

Modules as Structural Units

- **What is a Module?**
  - A container for logically related object types
  - A static object managed by the system
    - Loaded on demand by the runtime
- **Why are Modules Important?**
  - Provide a simple tool for
    - Encapsulating separate concerns
    - Static decomposition of a system
  - Narrow down and make explicit mutual dependencies via the IMPORT relation
  - Uniform system and application levels

Sample Module Hierarchy



Abstractions: What is it Primarily?

JukeBox: Player or Store?

class JukeBox: Player, Store
{ ...
}

Truck: Container or Vehicle?

class Truck: Vehicle, Container
{ ...
}

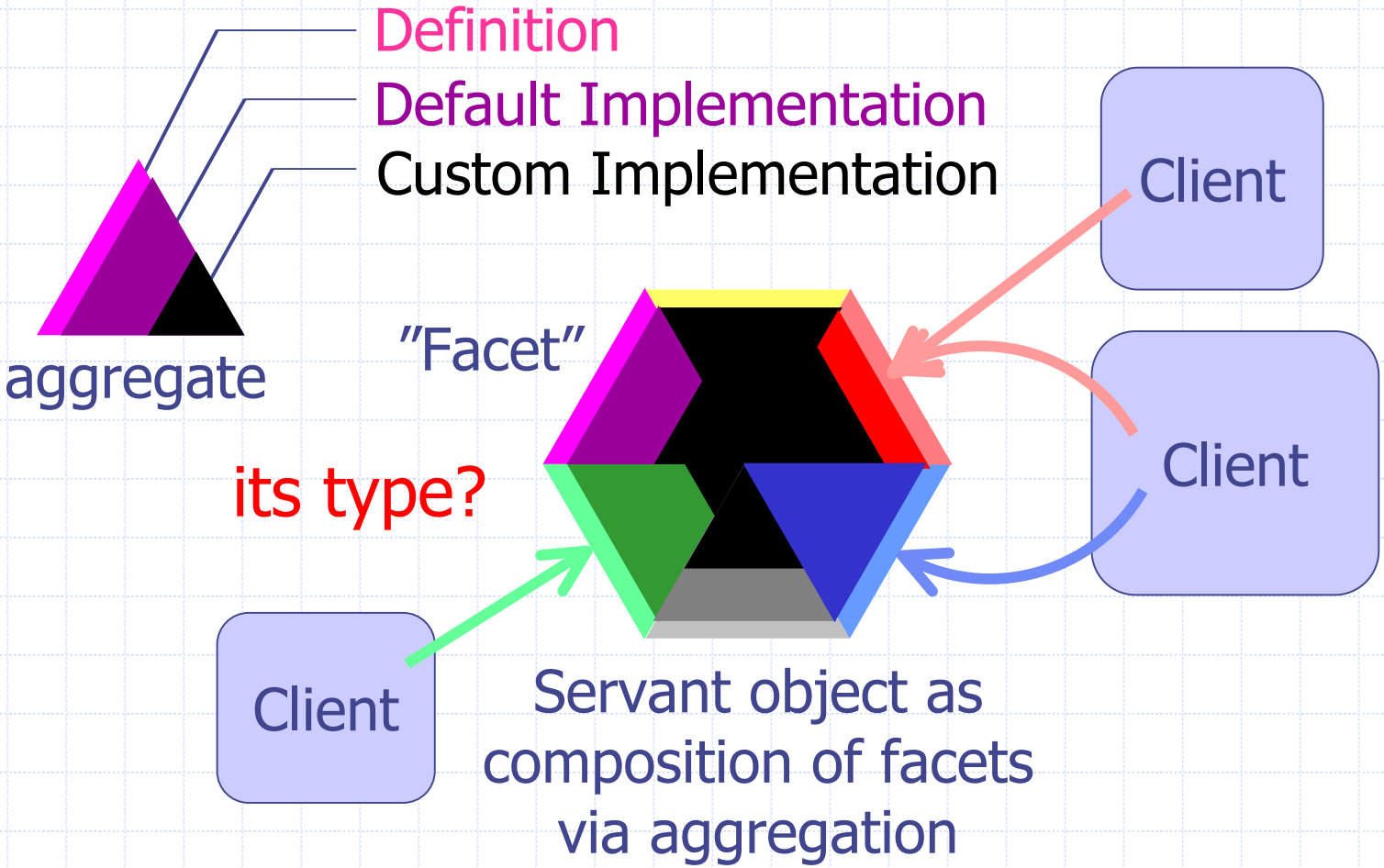
Computer: Calculator or DataBase or Browser?

class Computer: Calculator, DataBase, Browser { ...
}

Base class

Interface

A Uniform Abstraction Concept



An Example

Namespace

```
DEFINITION Music.Player;
  VAR cur: Song;
  PROCEDURE Play (s: Song);
  PROCEDURE Stop;
END Player.
```

```
DEFINITION Music.Store;
  PROCEDURE Clear;
  PROCEDURE Add (s: Song);
END Store.

IMPLEMENTATION Music.Store;
  VAR rep: Lib.Song;
  PROCEDURE Clear;
    BEGIN loop := NIL
  END Clear;
  PROCEDURE Add (s: Song);
    BEGIN s.next := rep; rep := s
  END Add;
BEGIN Clear
END Store.
```

```
OBJECT Music.JukeBox IMPLEMENTS Player, Store;
  IMPORT Store; (* aggregate *)
  PROCEDURE Play (s: Song); IMPLEMENTS Player.Play;
  PROCEDURE Stop; IMPLEMENTS Player.Stop;
END JukeBox.
```

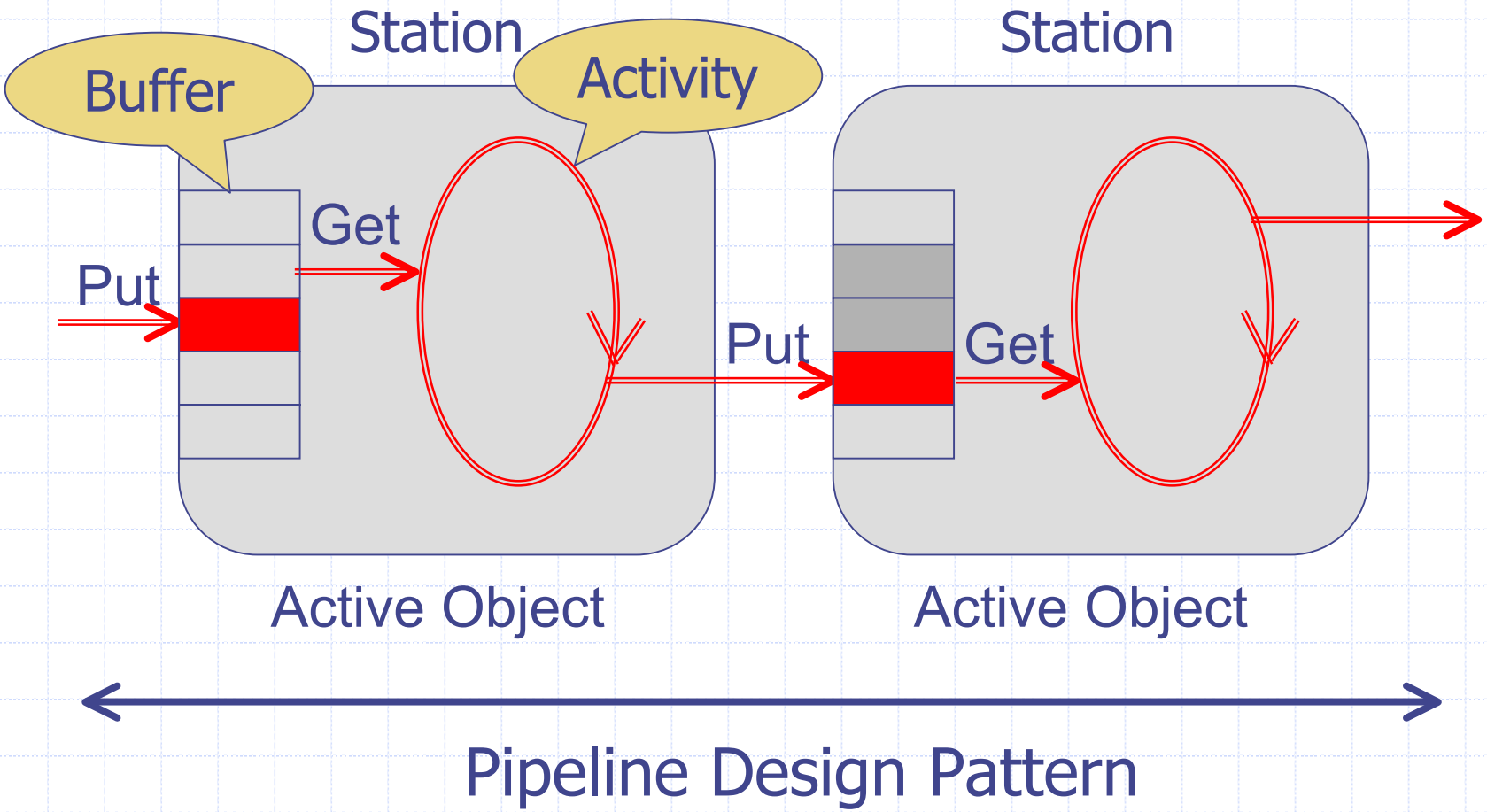
C# is a Good Language, but ...

- What can be improved?
  - Composability
  - Concurrency
  - Communicativity

Concurrency

- **C#**
  - System.Threading library
  - General lock objects
  - Wait/Pulse for threads self-management
- **Zonnon**
  - Activities built into objects
  - Object monitor locks
  - Thread management by system via AWAIT

Pipeline with Active Objects



Active Objects in Zonnon

```
OBJECT Station (next: Station);
   VAR { PRIVATE } n, in, out: INTEGER;
      buf: ARRAY N OF OBJECT;
   PROCEDURE  { PRIVATE }  Get (VAR x: OBJECT);
   BEGIN { LOCKED } AWAIT (n # 0);
      DEC(n); x := buf[out]; out := (out + 1) MOD N
   END Get;
   PROCEDURE { PUBLIC } Put (x: OBJECT);
   BEGIN { LOCKED } AWAIT (n # N);
      INC(b); buf[in] := x; in := (in + 1) MOD N
   END Put;
   ACTIVITY Process; VAR x: OBJECT;
   BEGIN LOOP Get(x); (*process x;*) next.Put(x) END
   END
BEGIN n := 0; in := 0; out := 0; NEW(Process)
END Station;
```

Separate thread

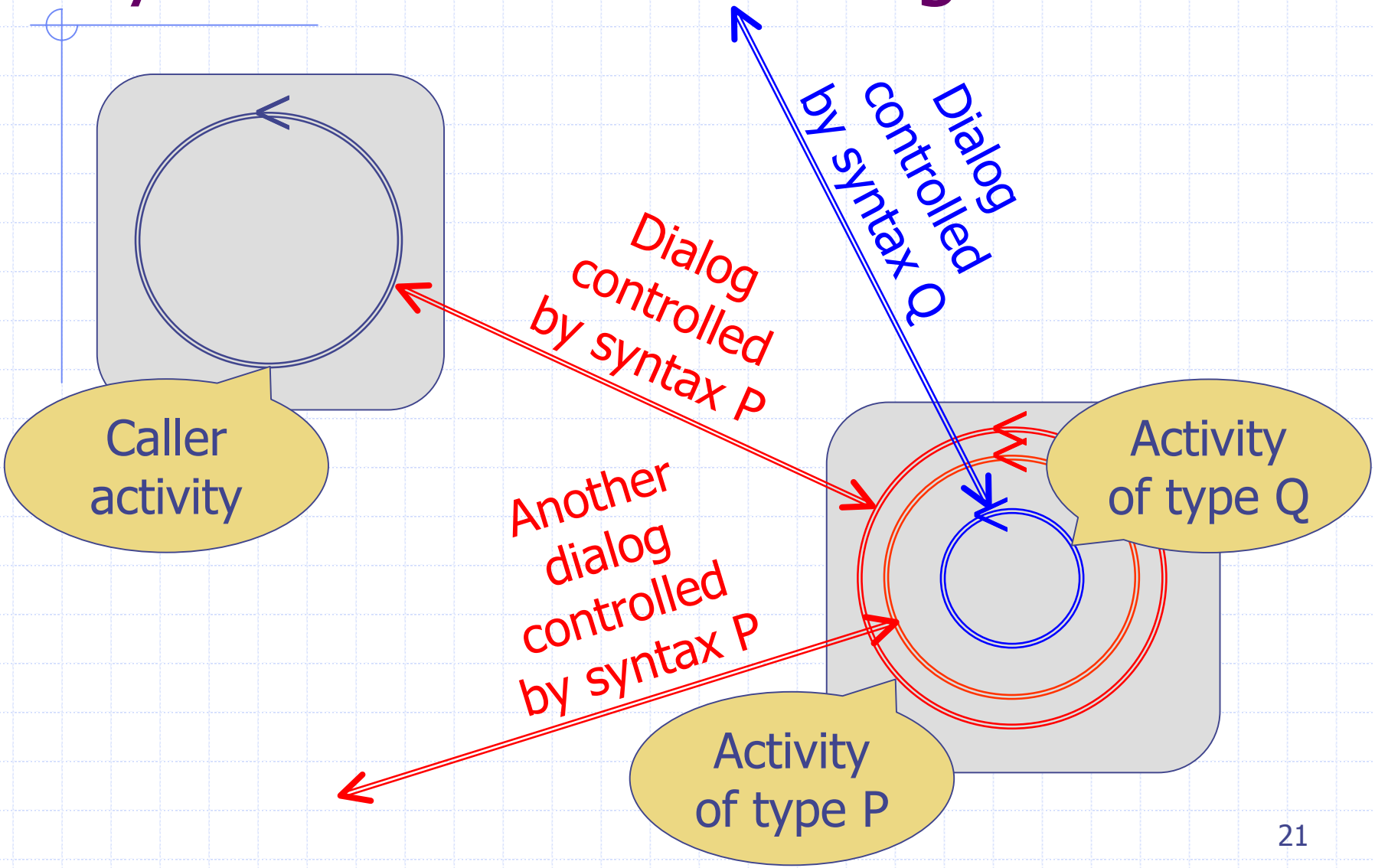
C# is a Good Language, but ...

- What can be improved?
  - Concurrency
  - Composability
  - Communicativity

Object Communication

- **C#**
  - Local method invocation
  - Remote method invocation
    - Proxy objects & marshalling
    - SOAP & remote channels
- **Zonnon**
  - Local method invocation
  - Syntax controlled dialogs

Syntax Controlled Dialogs



Syntax Protocols as Contracts

```
DEFINITION ETicketing;
   ACTIVITY TravelService =
      CHECKPRICE Destination [ TicketType ]  Price |
      BUYTICKET Destination  [ TicketType ]
         AccountID TicketID;
      Destination = CharString;
      TicketType = (FULL | REDUCED) [TWOWAY];
      Price = Number;
      AccountID = CharString ":" CharString "." CharString;
      TicketID = CharString "." CharString
   END TravelService;
END ETicketing.
```

EBNF

Keywords

Dialogs as Parser Activities

- **Servant Code**

```
ACTIVITY MyService;
  VAR t: TOKEN;
BEGIN … ?t; …; !t; …
END MyService;
```

Parser run as separate thread

- **Client Code**

```
ACTIVITY You;
  VAR t: TOKEN; s: Servant; p: S.P;
BEGIN NEW(p, s);
  … p!t; … p?t; …
END You;
```

Where are We?

http://bluebottle.ethz.ch/Zonnon/

- **Zonnon Report**
  - Authored by Brian Kirk & David Lightfoot
  - Beta release "fresh from press"
- **Zonnon Compiler**
  - Using leading edge MS integration technology
  - First compilation results available
  - Extensive test suite available
- **Zonnon Text Books**
  - "A Short Course in Programming-in-the-Small" planned for end of 2003