



Ist die Programmiersprachen Entwicklung zu Ende?

**Wir präsentieren
Oberon Sprachevolutionen
Jürg Gutknecht, ETH Zürich
Oktober 2000**

Die *Pascal* Sprachenfamilie

ETH Zürich

Algorithmen &
Datenstrukturen

1960 **ALGOL** Prozeduren, Lokalität

1970 **Pascal** Datentypen

Systembau

1980 **Modula-2** Module

1990 **Oberon** Typenerweiterung

2000 **Active Oberon** Aktive Objekte

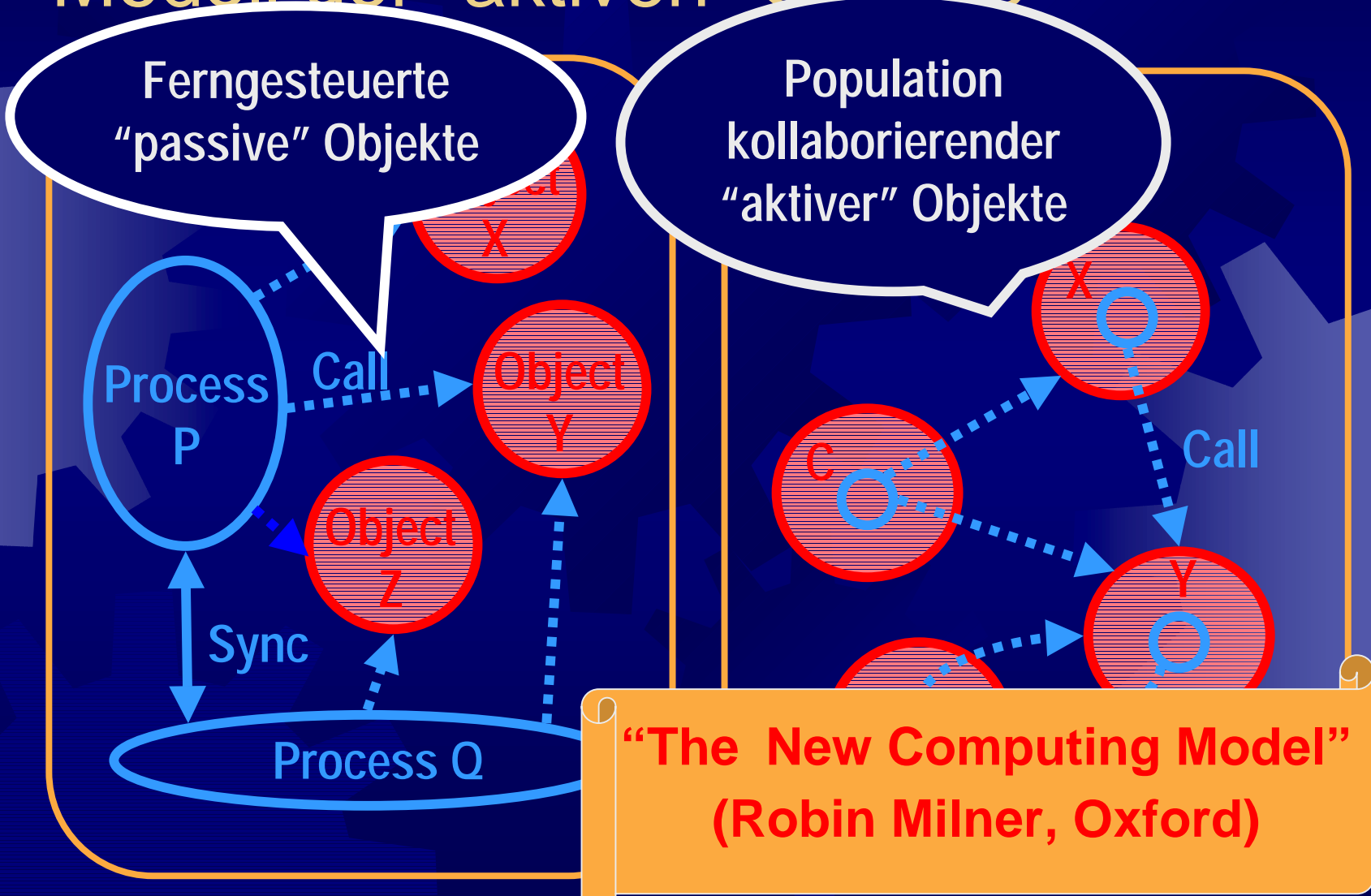
2000 **Lightning Oberon** (mit MSR)

Interoperabilität

Globale
Programmierung

Evolution *Active Oberon*

Modell der "aktiven" Objekte



Evolution Active Oberon

Aufgewertete Objekttypen

```
Z = OBJECT
```

```
VAR t: T;
```

```
PROCEDURE p (u: U;  
BEGIN { EXCLUSIVE } ...  
END p;
```

```
PROCEDURE q;  
BEGIN assertion := TRUE  
END q;
```

```
BEGIN ...
```

```
BEGIN { EXCLUSIVE } ...  
PASSIVATE assertion; ...
```

```
END
```

```
END Z;
```

Zustand

Methoden

Separater Thread

Monitor Objekt

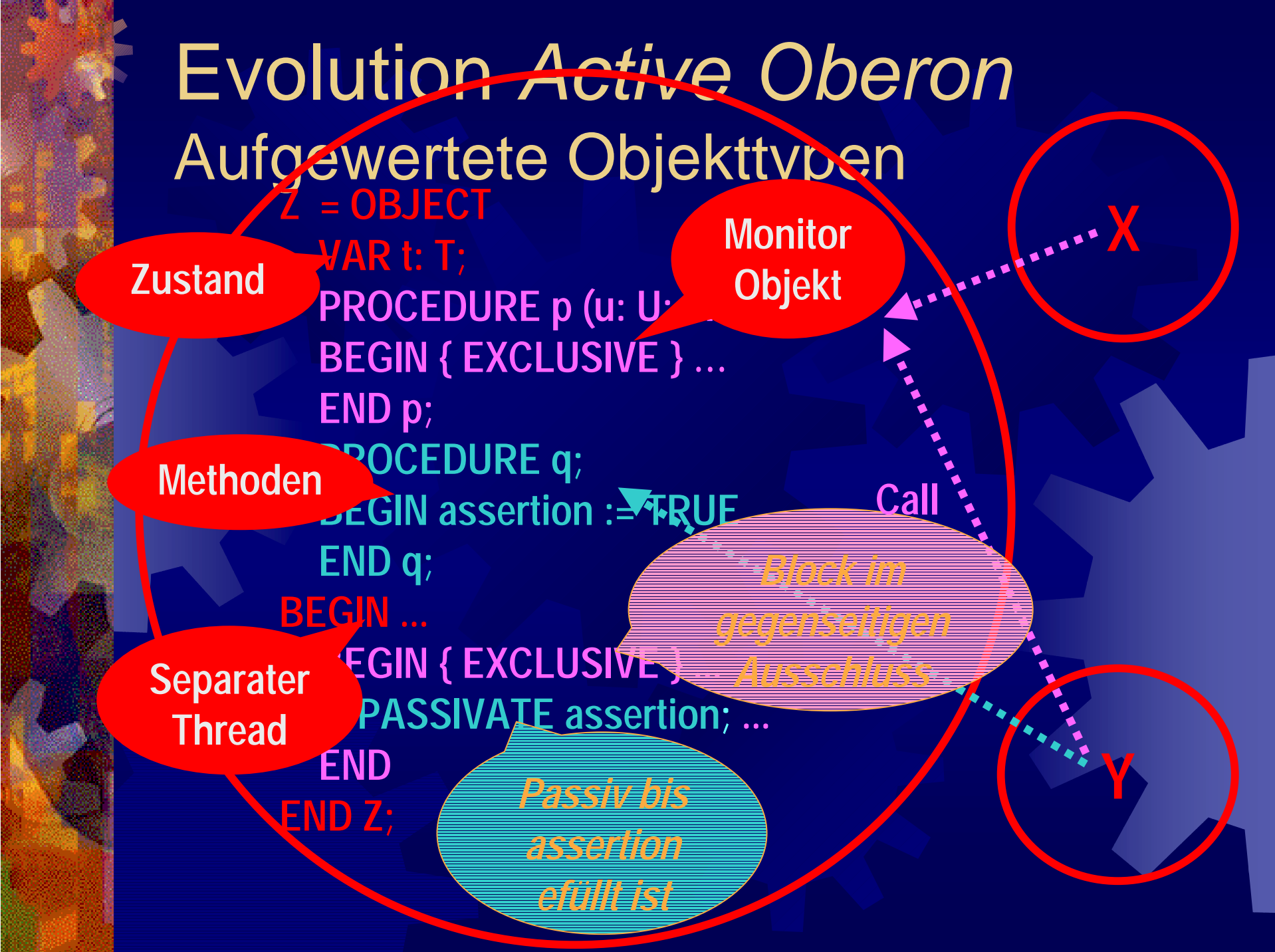
Block im gegenseitigen Ausschluss

Passiv bis assertion erfüllt ist

Call

X

Y

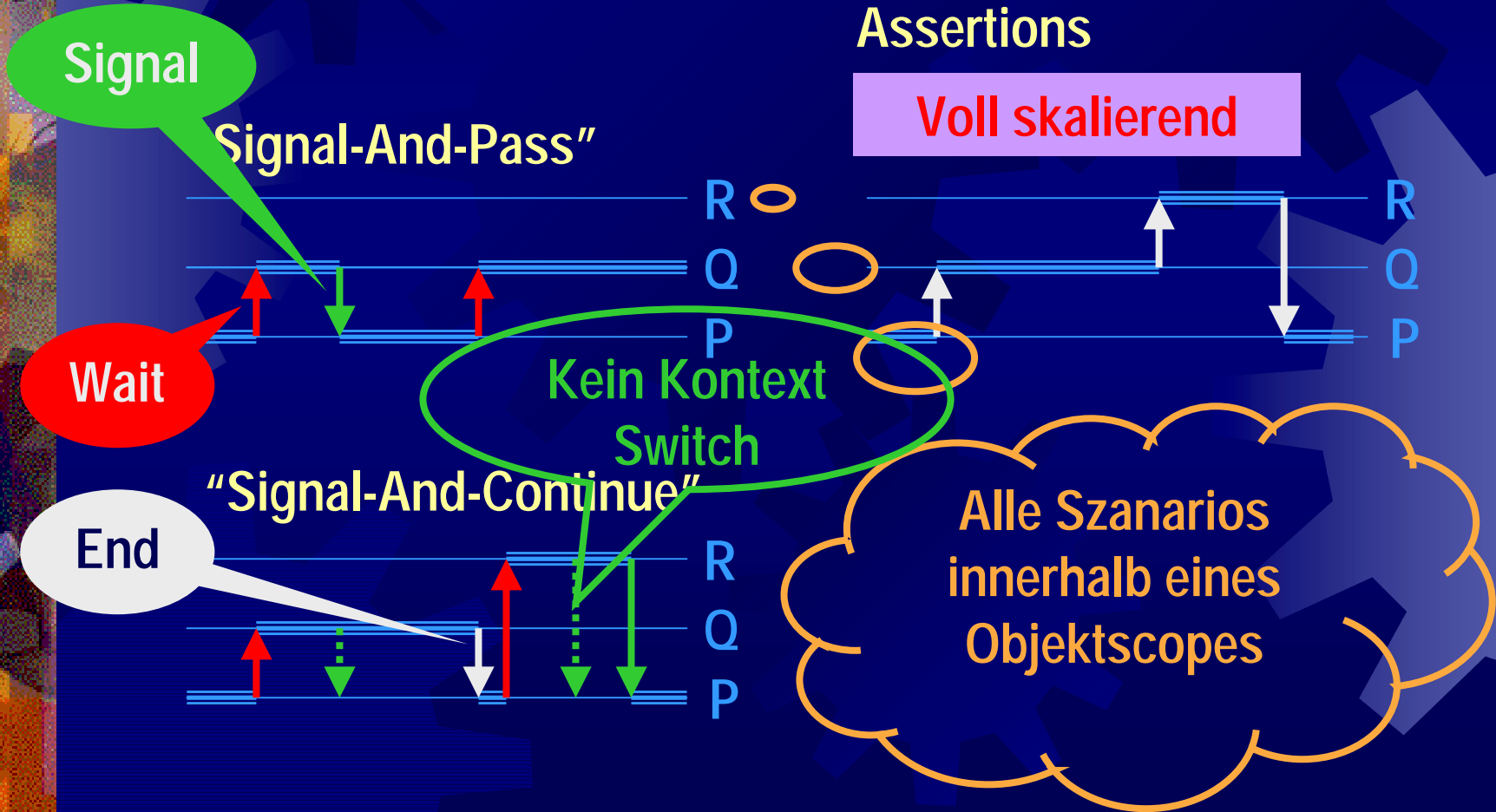


Evolution *Active Oberon*

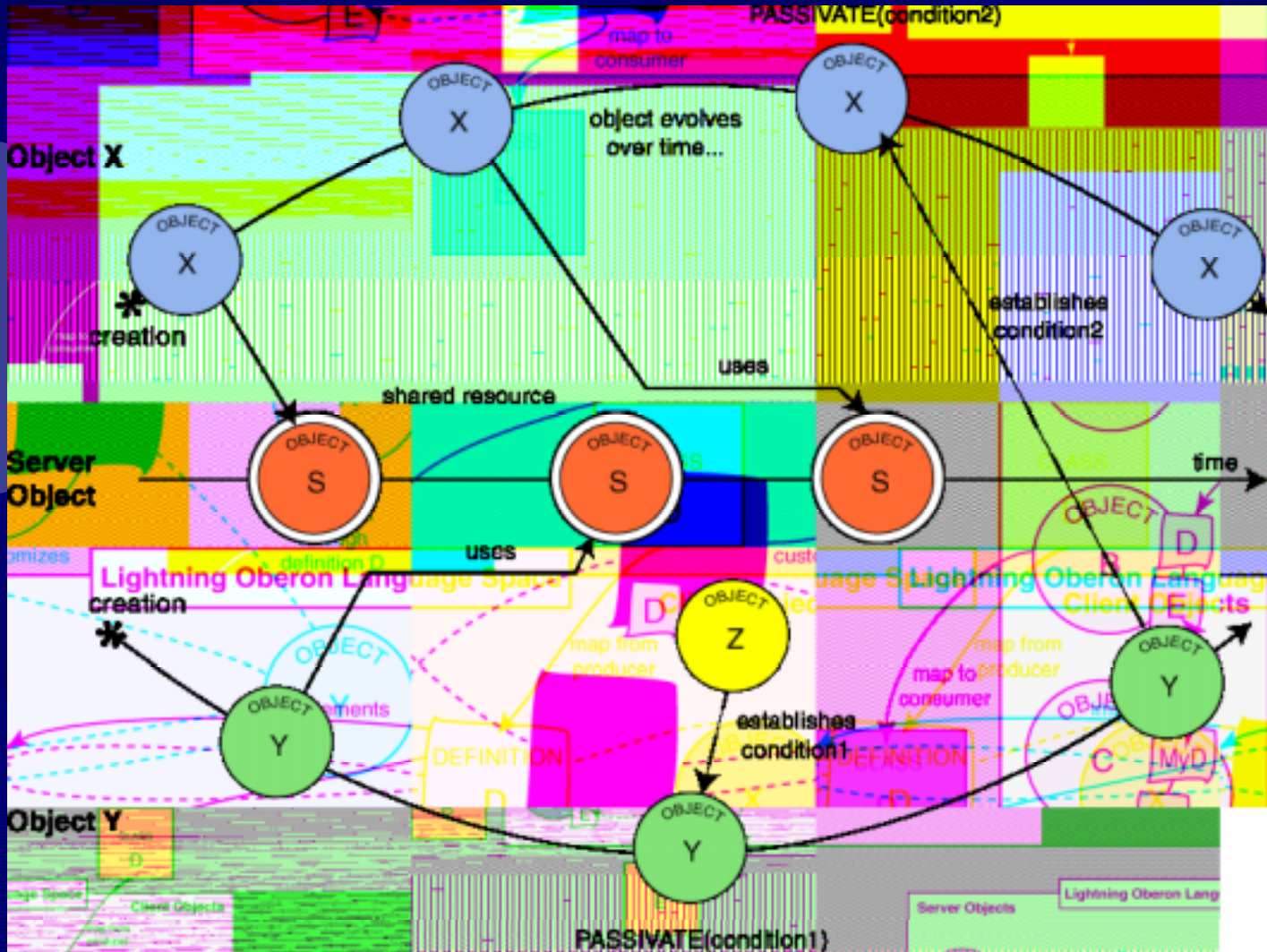
Synchronisierung mit *Assertions*

Basiert auf kooperativen Partnerprozessen

Basiert auf systemverwalteten Assertions



Evolution *Active Oberon* Laufzeitszenarium



Evolution *Active Oberon*

Ein Beispiel

TYPE

```
Movie = OBJECT
```

```
VAR first, cur: Figure; X, Y: REAL; stopped: BOOLEAN;
```

```
PROCEDURE NEW (orgx, orgy: REAL; video: Figure);
```

```
BEGIN X := orgx; Y := orgy; first := video
```

```
END NEW;
```

```
PROCEDURE Stop;
```

```
BEGIN stopped := TRUE
```

```
END Stop;
```

```
BEGIN { active }
```

```
stopped := FALSE; cur := first;
```

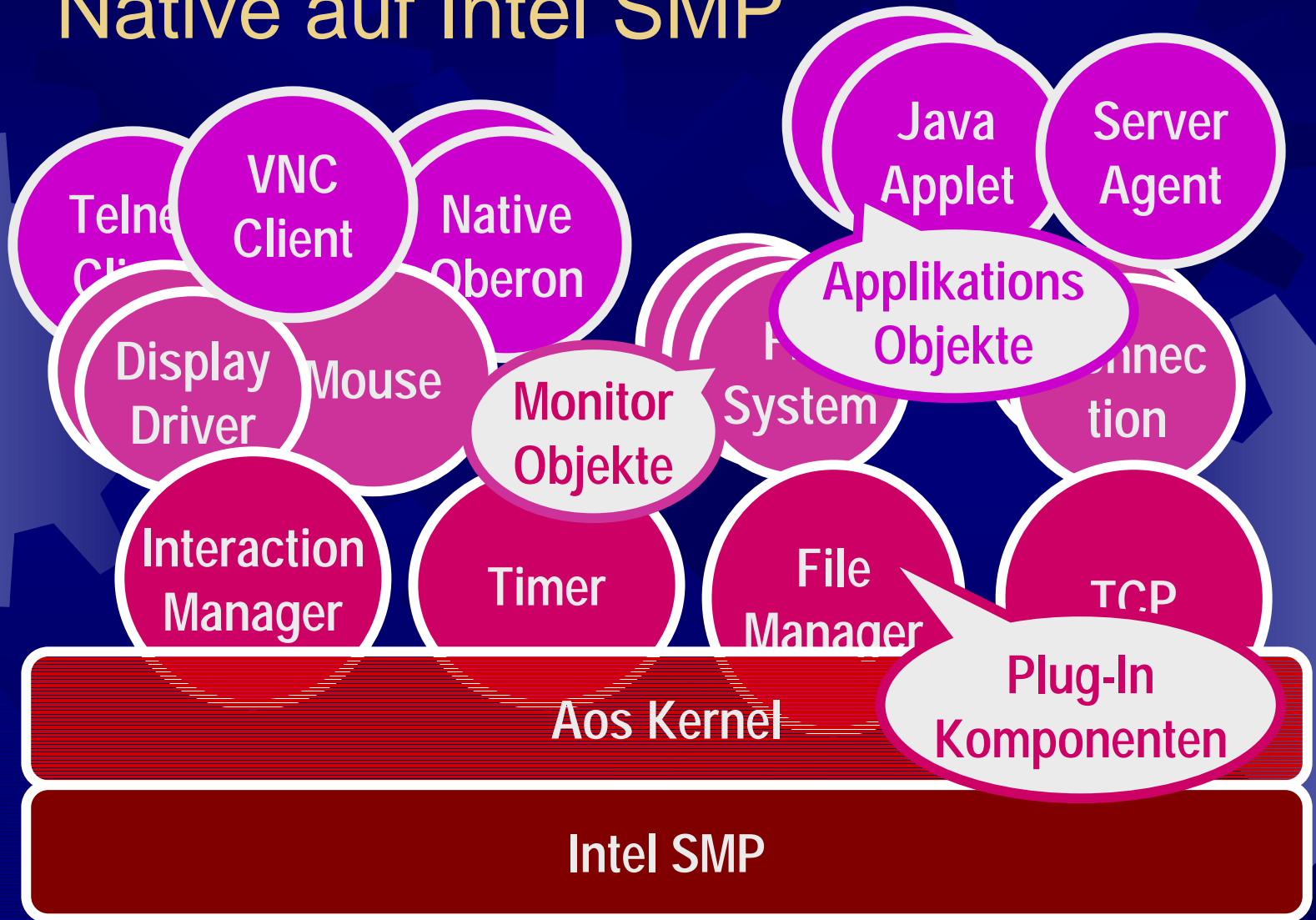
```
WHILE ~stopped DO
```

```
cur.Display(X, Y); cur := cur.next (*cyclic list*)
```

```
END
```

```
END Movie;
```

Evolution *Active Oberon AOS* Native auf Intel SMP



Evolution *Lightning Oberon* MS *.NET* Interoperabilitätsplattform

➤ **IL Intermediate Language**

- o Virtuelle Stackmaschine
- o Objektmodell mit Metadaten
- o *CLR* Common Language Runtime
- o Hochoptimierender JIT Compiler

➤ **Microsoft Research Project7** Mit ETHZ

- o Research Languages
 - ❖ Scheme, Haskell, Oberon, Oz, Mercury, ...
- o Kommerzielle Sprachen
 - ❖ C#, Eiffel, Cobol, ...

Evolution *Lightning Oberon* OOP im neuen Kleid (1)

➤ **Traditionelle OOP**

○ **Klassenhierarchie als Mehrzweck Tool**

❖ Unterklassen

- zur Klassifizierung von Objekten
- zur Objektbenutzung via Polymorphie

❖ Vererbung zur Wiederverwendung von Code

○ **Modifiers und Patterns zur Differenzierung**

❖ virtual, final, abstract, static, ...

○ **Passive Objekte**

❖ Reaktiv und ferngesteuert

Evolution *Lightning Oberon*

OOP im neuen Kleid (2)

➤ **Lightning Oberon**

Definitions

- o Implementationseinheiten
- o Einheiten der Benutzung
- o Wiederverwendung von Code

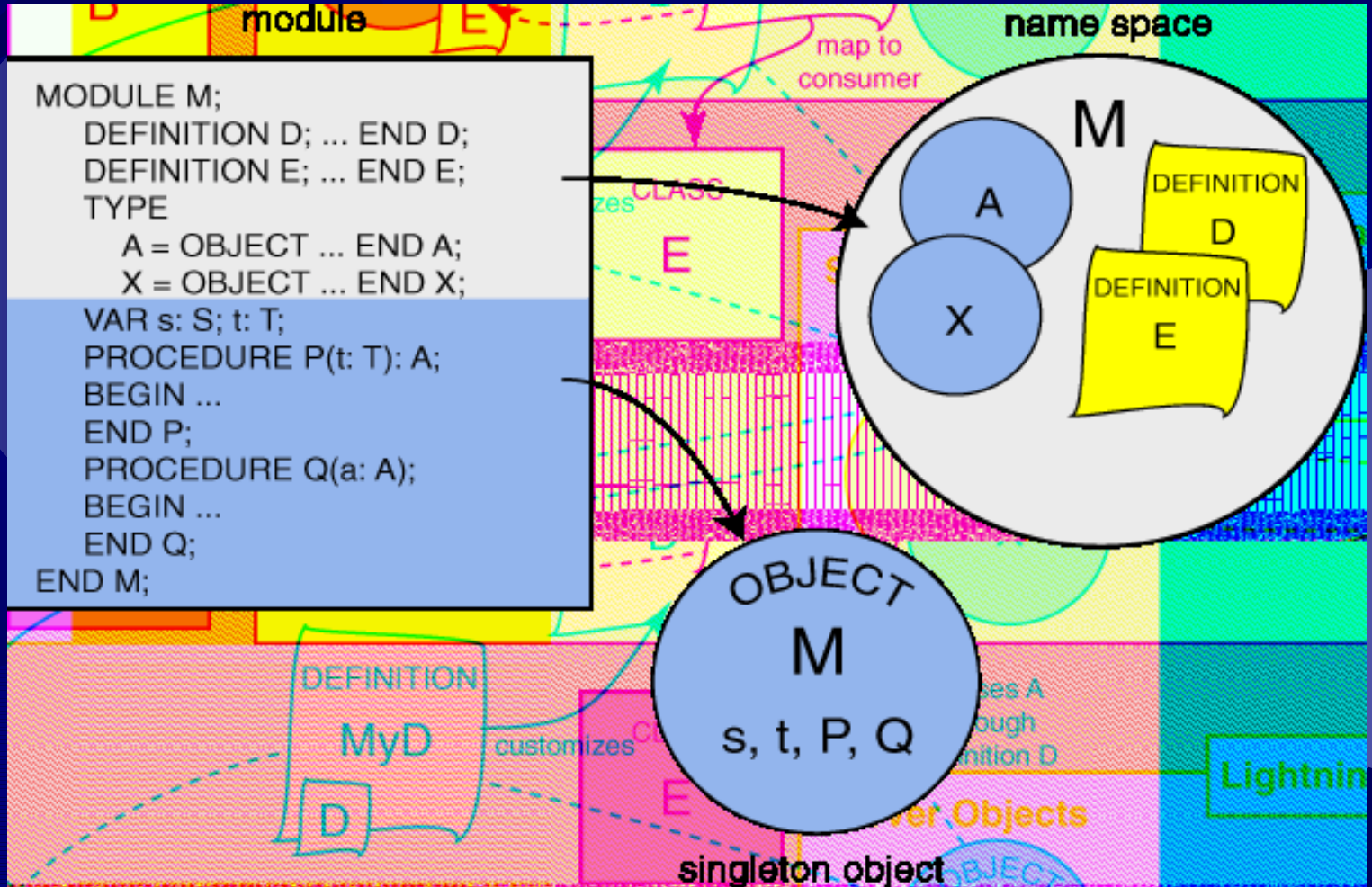
Module

- o Namespaces
- o Singleton Objekte

Aktive Objekte

- o Aktiv und selbststeuernd

Evolution *Lightning* Oberon Modulkonzept



Evolution *Lightning Oberon*

Definitionskonzept (2)

➤ Relationen

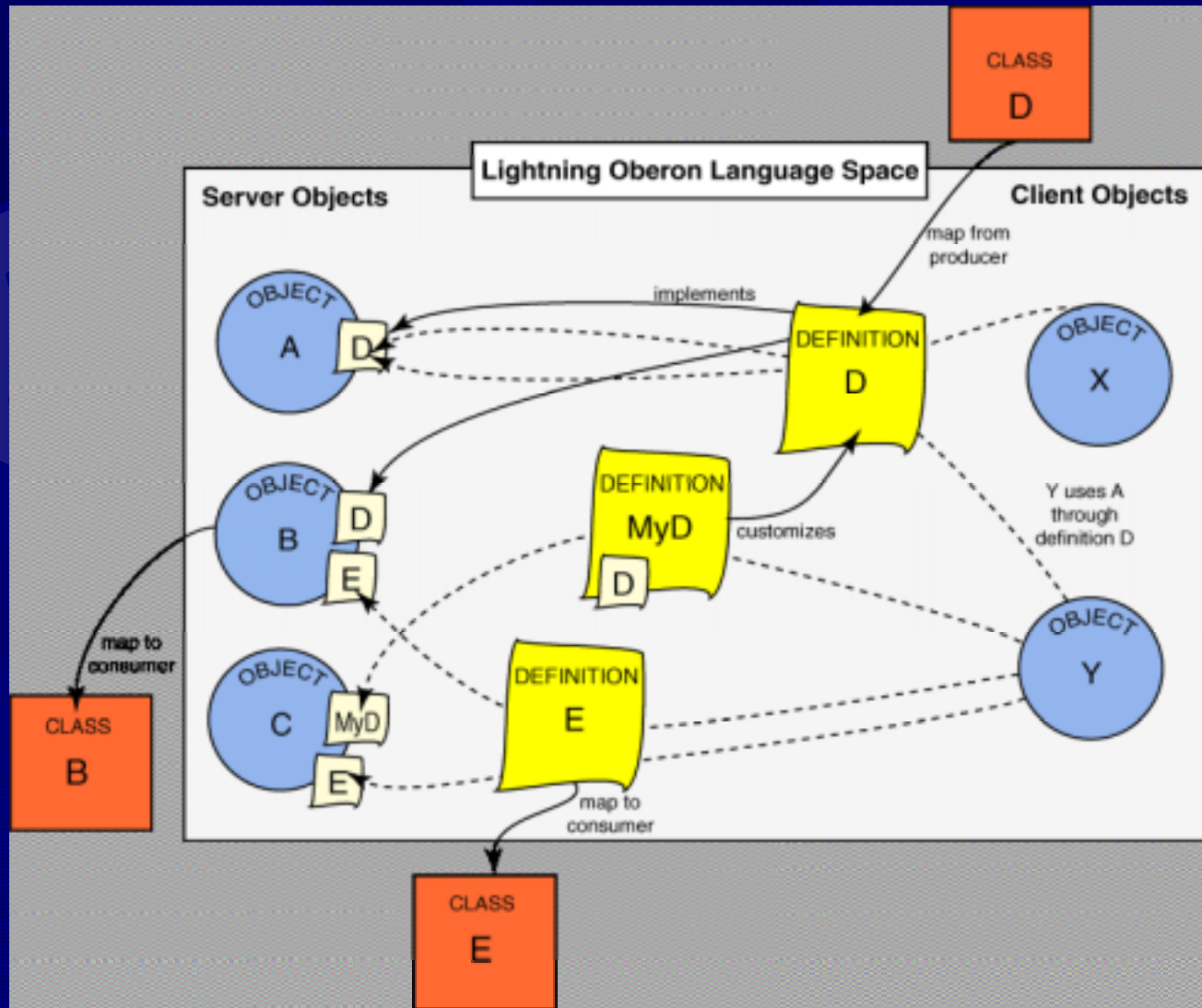
o Verfeinerung

- ❖ Hinzufügen von Variablen und Methoden
- ❖ Implementieren abstrakter Methoden

o Implementierung

- ❖ Adjungieren des Zustandsraumes
- ❖ Implementieren aller abstrakten Methoden

Evolution *Lightning Oberon* Definitions-konzept (3)



Evolution *Lightning Oberon* Codemuster

Implementation

```
DEFINITION I; (* pure interface
  VAR s: S; t, u: T;
  PROCEDURE p (...);
  PROCEDURE q (...);
END I;
```

```
DEFINITION D;
  VAR x: X; y: Y;
  PROCEDURE f (...);
    VAR u, v: U;
  BEGIN ..
  END f;
  PROCEDURE g (...);
  PROCEDURE h (...);
  BEGIN .. f(...);
  END h;
END D;
```

```
TYPE
  S = OBJECT
    IMPLEMENTS I, D
  END;
VAR s: S;
```

Benutzung

```
IF s IMPLEMENTS D THEN
  D(s).f(...)
END
```

```
DEFINITION MyD REFINES D;
  VAR z: Z;
  PROCEDURE g (...);
  BEGIN .. RETURN ..
  END g;
  PROCEDURE k (...);
END MyD;
```

Verfeinerung

Evolution *Lightning Oberon* Blockanweisung

ACTIVE
EXCLUSIVE
CONCURRENT

```
BEGIN
```

```
... BEGIN { options }
```

```
END ... BEGIN { options }
```

```
END ...
```

```
ON EXCEPTION DO
```

```
...
```

```
END
```