

**Combined Application of UML and SDL-92/SDL-96**

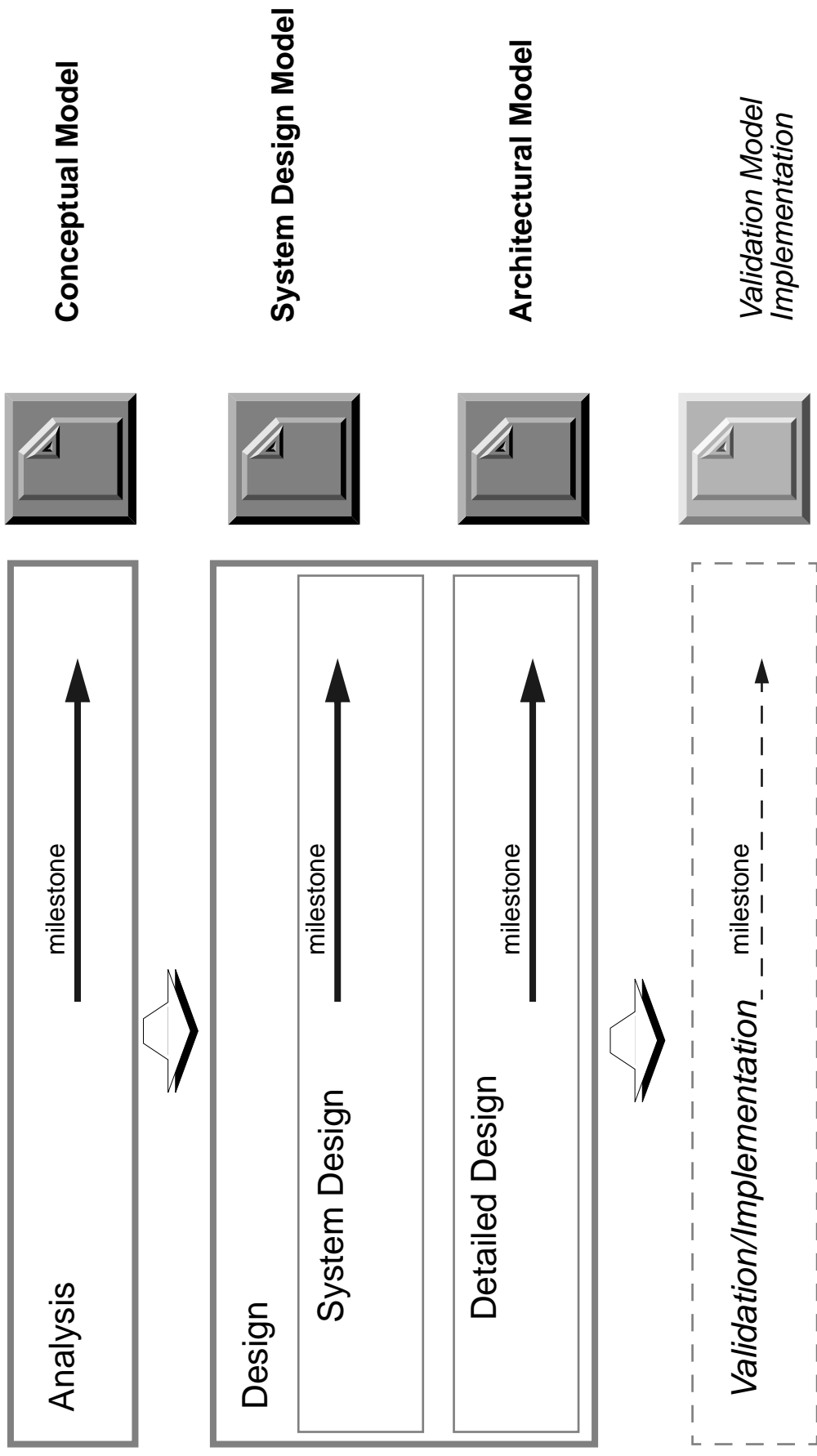
Eckhardt Holz  
Humboldt-University Berlin  
Dept. of Computer Science  
Berlin - Germany

holz@informatik.hu-berlin.de

## Application of UML for Telecommunication Systems Design

- UML as ODP-Viewpoint-language
  - Enterprise Viewpoint
  - Computational Viewpoint
  - Information Viewpoint
  - Engineering Viewpoint
- Relation to other languages and techniques
  - **UML — SDL**
  - UML — ODL
  - UML — MSC
- ODP-based design process for UML

- Mapping of UML
  - analogous to OMT-SDL mappings (e.g. INSYDE, BNR)
    - Static Structure + State/Activity Diagrams
    - should be simpler through more precise semantics
  - exploitation of extension mechanisms
    - Stereotypes for SDL-language elements (BLOCK, SYSTEM etc.)
    - OCL-definitions and properties
    - Static Structure, Collaboration, State/Activity Diagrams
    - SDL in UML specified (?)
- Mapping SDL to UML
  - Round-Trip-Engineering
  - Re-use of existing SDL specifications
- Linking
  - data types, signals in UML - structure/behaviour in SDL
  - analogous SOMET - Telelogic, ObjectGeode

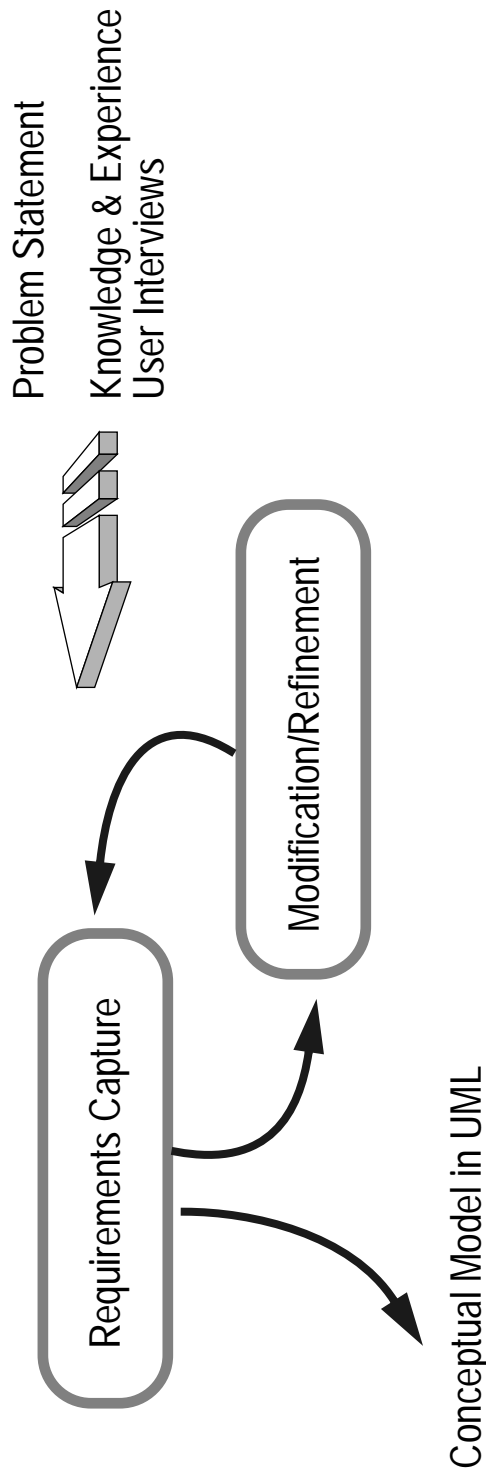


## Analysis

The analysis is performed using UML

Model components:

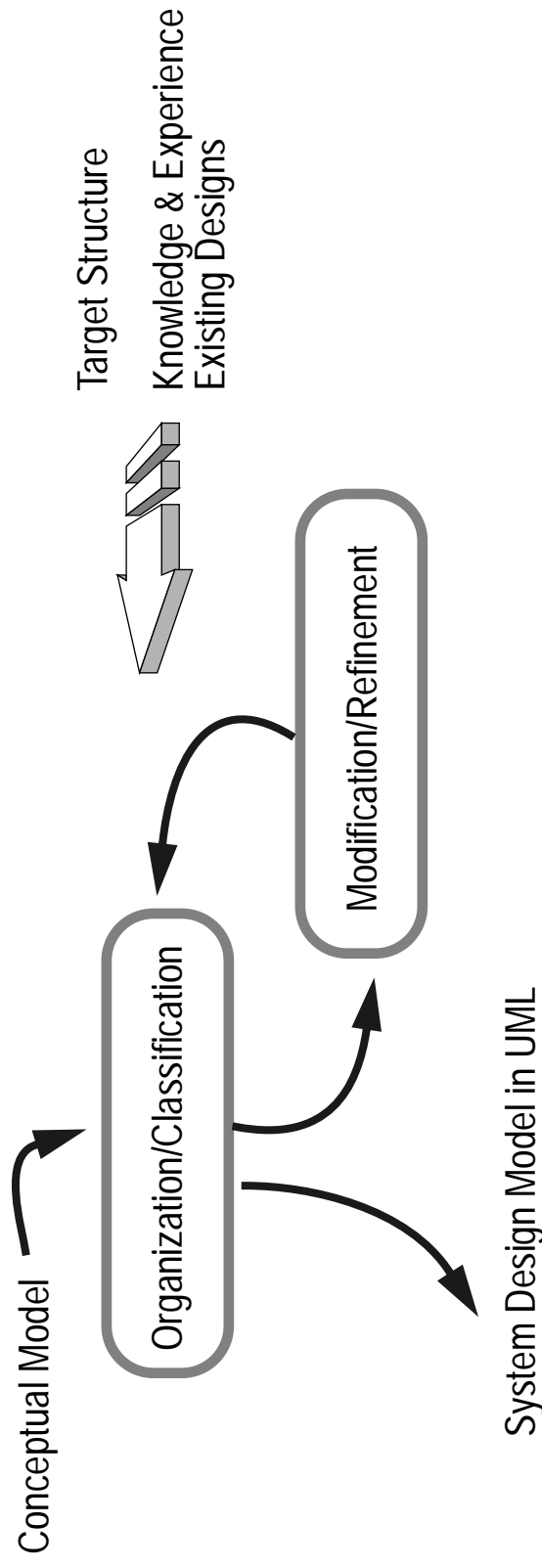
- Static Structure Model
- Use Cases
- State Charts/Activity Diagrams



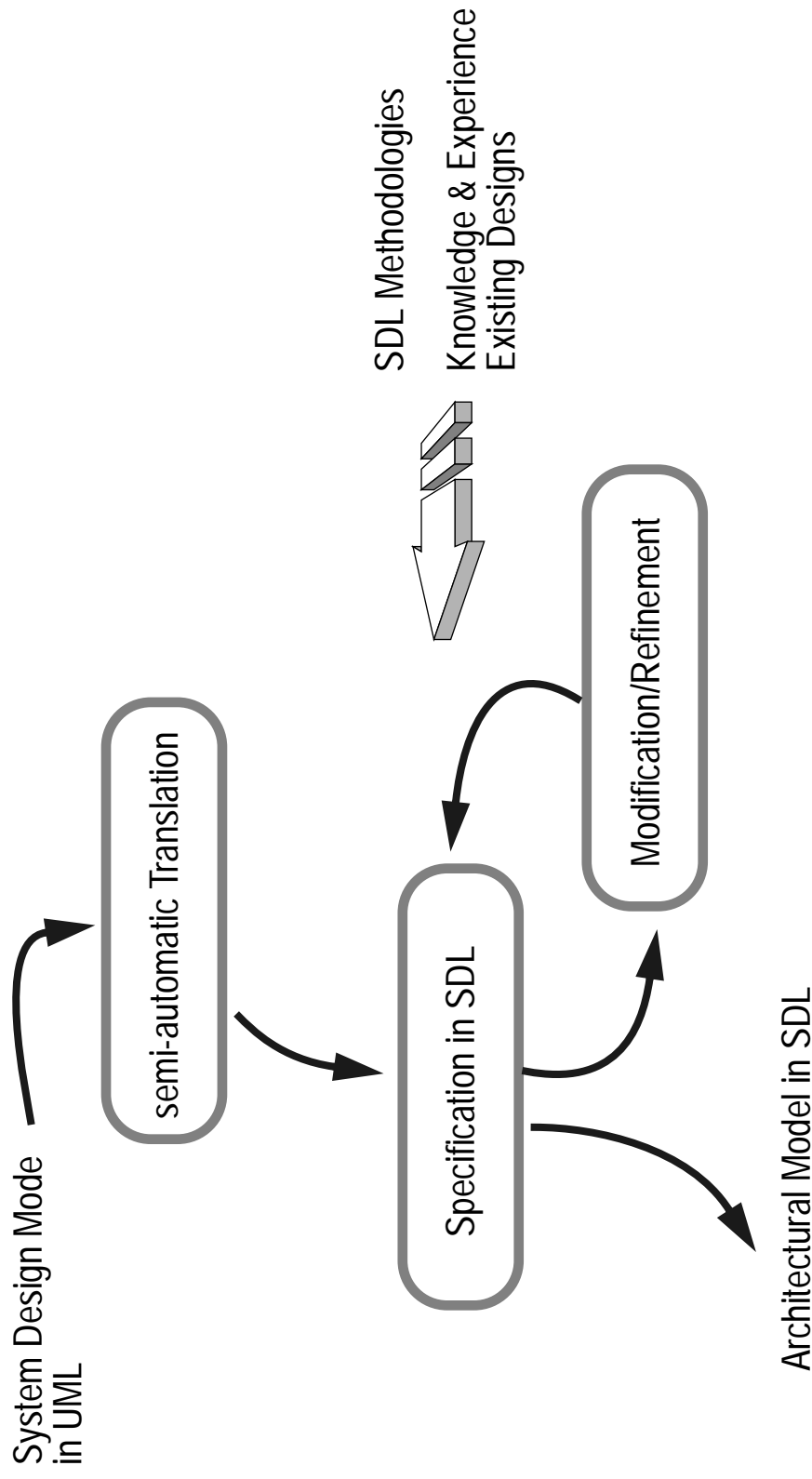
## System Design

The following decisions have to be made during system design:

- organize the system into subsystems
- mark subsystems as *software*, *environment*, *hardware* etc.
- Static Structure, Collaboration Diagrams
- State Charts/Activity Diagrams



## Detailed Design



## Classes

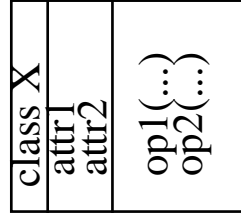
- Top class of aggregation hierarchy
  - System Type
- Classes with state diagram
  - Process Type
- Aggregates of active classes
  - Block Type

# Classes

- Passive classes
  - SDL Data type declaration



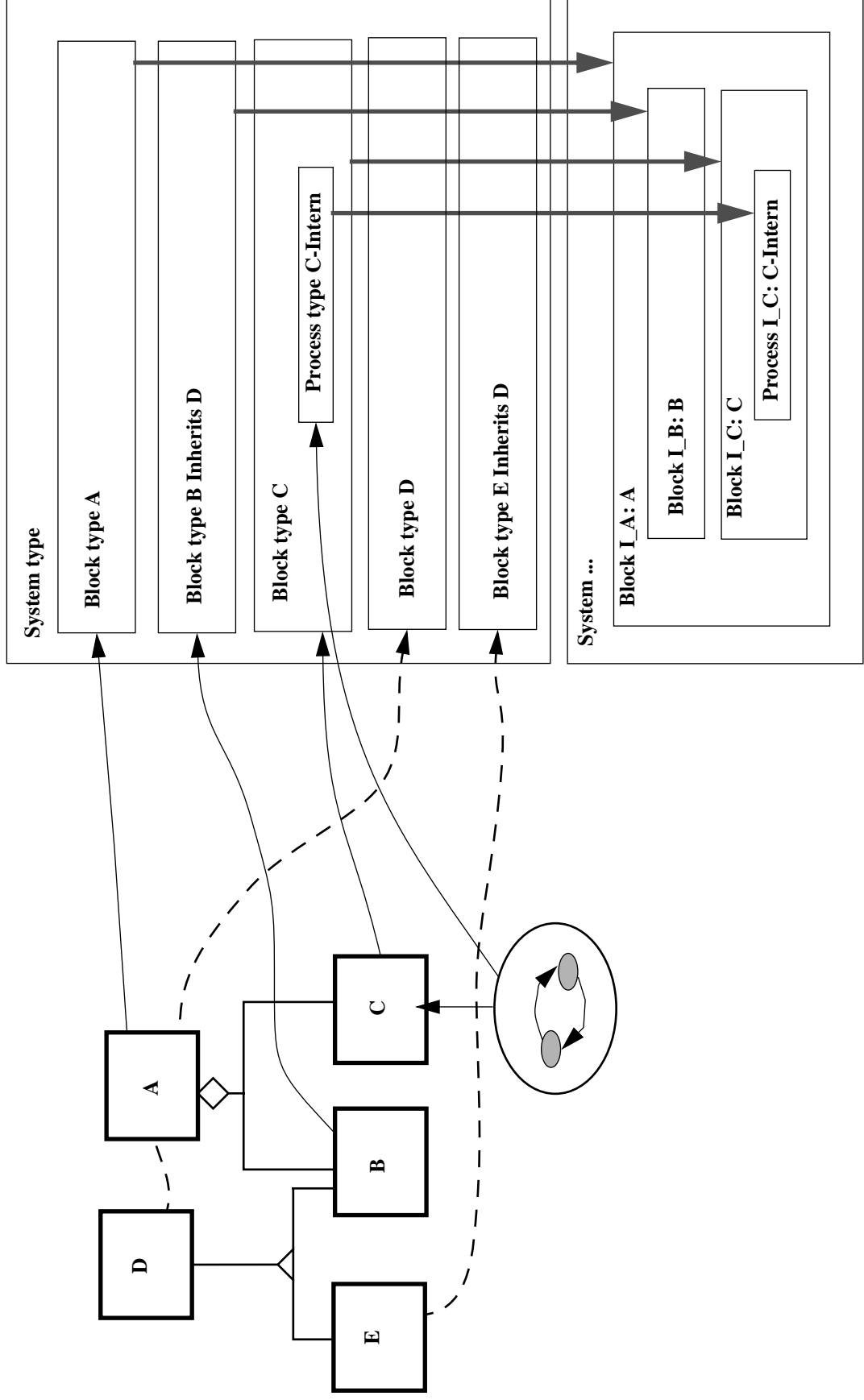
**newtype X;**  
**operators**  
 getAttr1, setAttr1,  
 getAttr2, setAttr2,  
 op1: ...; op2: ...;  
**endnewtype X;**



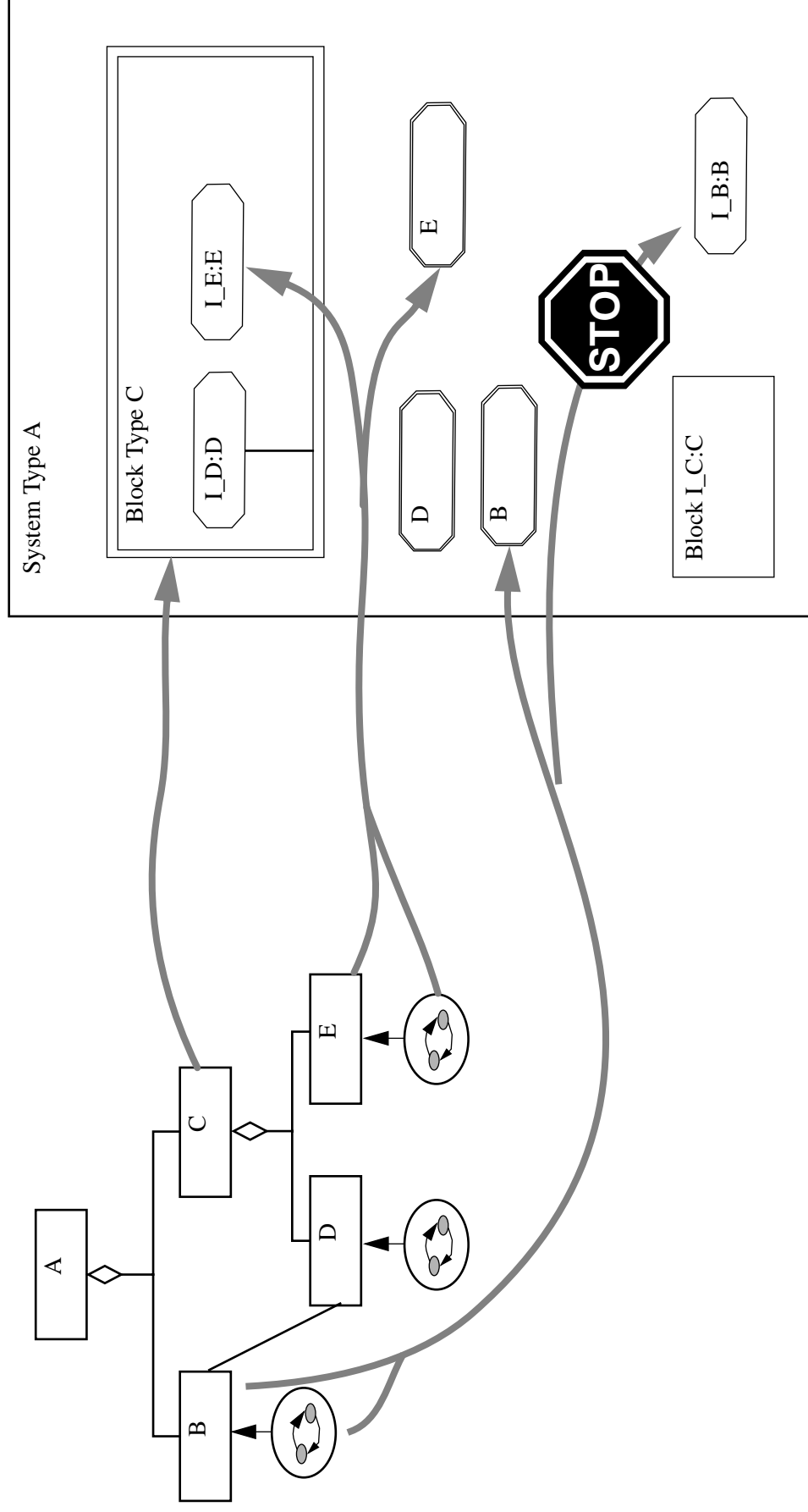
**newtype X;**  
**struct**  
 attr1...; attr2 ...;  
**operators**  
 op1: ...; op2: ...;  
**endnewtype X;**

- Active classes
  - System Type
  - Block Type
  - Process Type

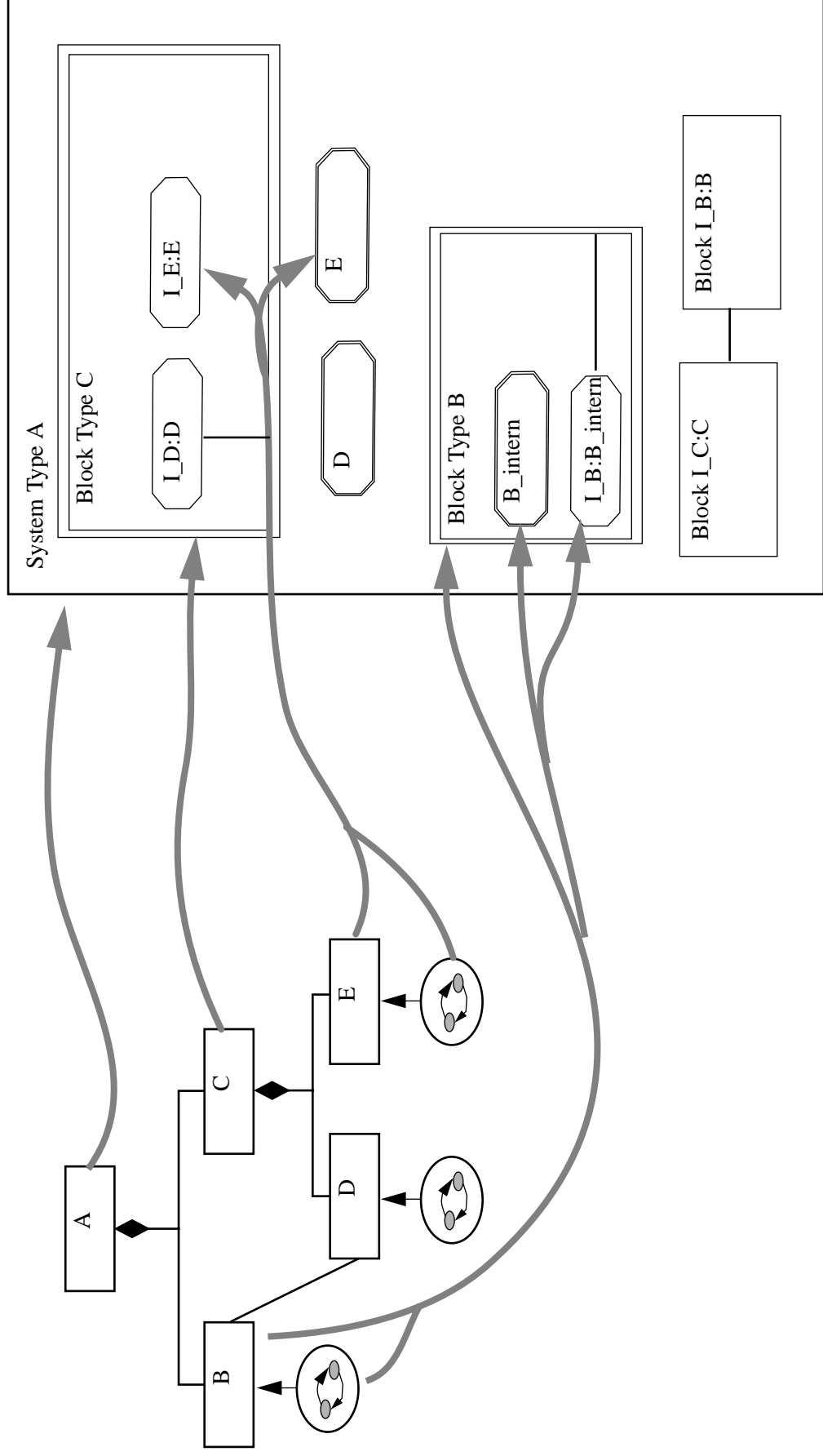
# Transformation to SDL-92



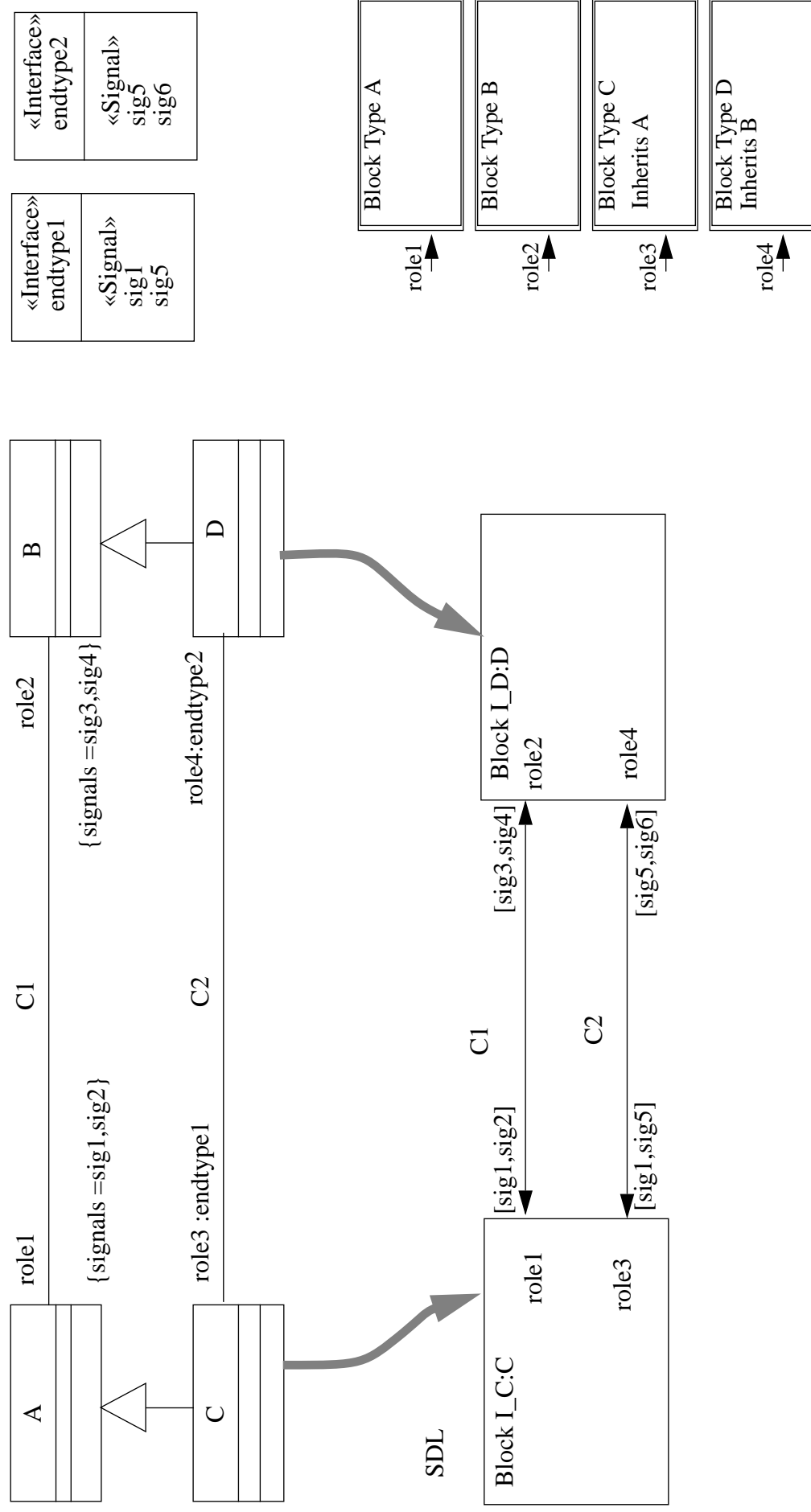
# Balancing



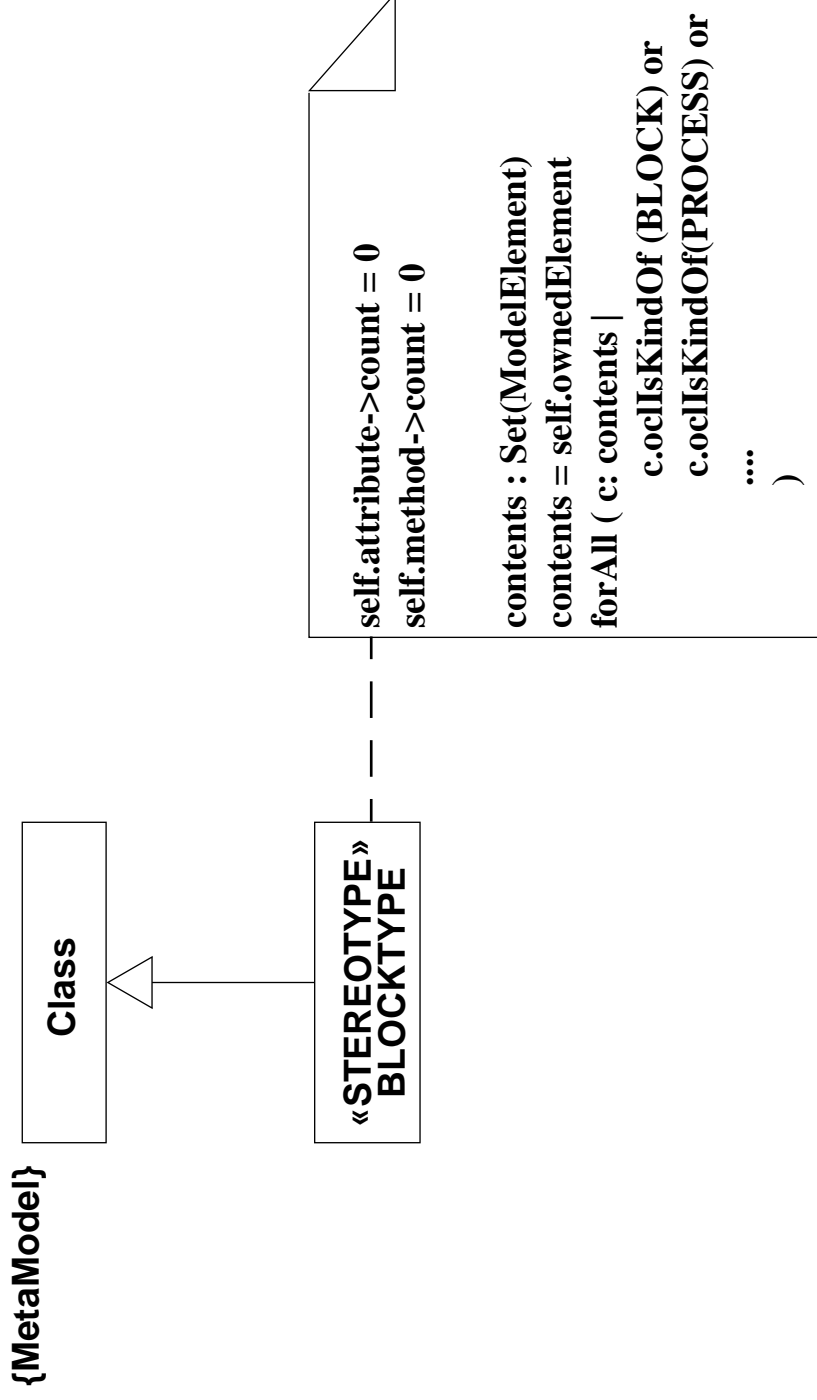
# Balancing

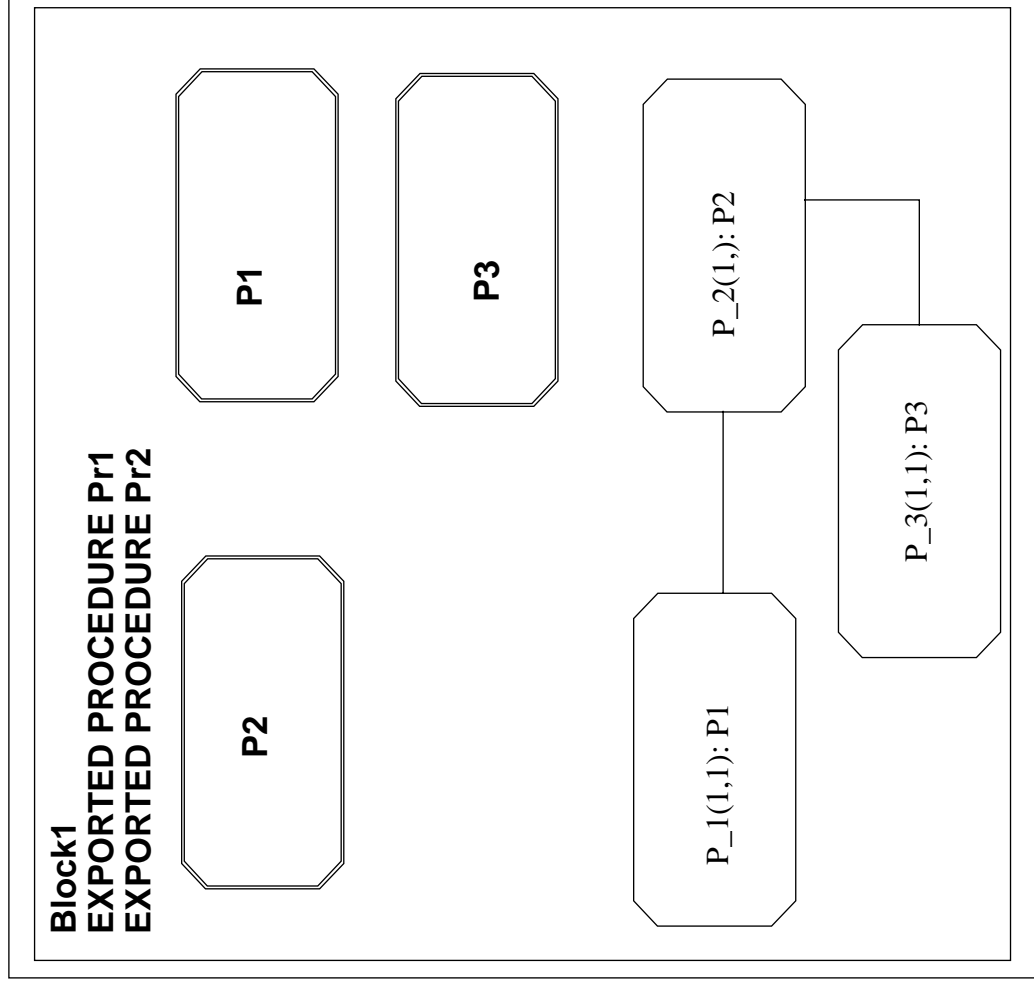
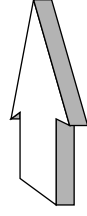
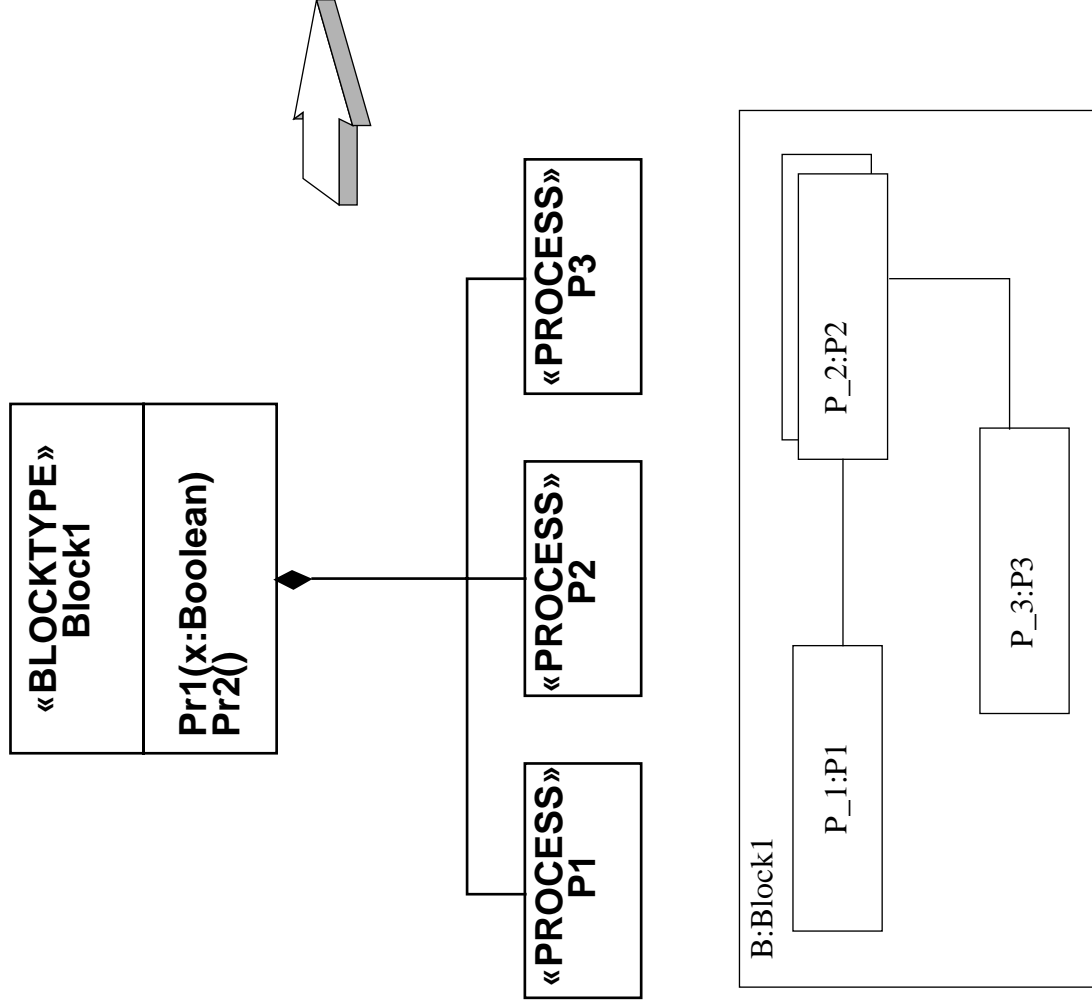


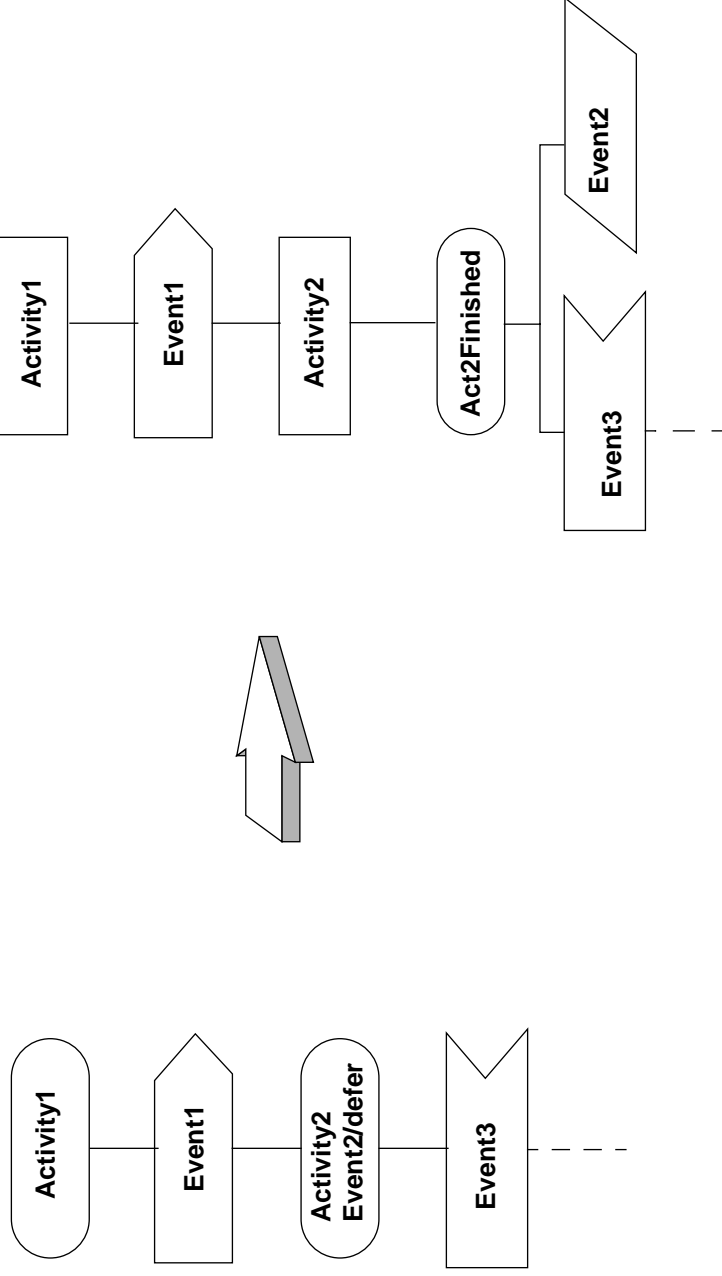
# Connections











## Further Concepts

- Interfaces
  - gates, gate-types
  - RPC declarations
- Collaborations
  - system and block structure diagrams, channels
  - interactions/message flow??
- Parameterized Classes
  - types with context parameters
- Packages
  - packages, use-clauses

## Conclusion

- UML is a suitable analysis and design technique for the development of SDL specifications
- UML models of large SDL specification help for a better understanding
- UML is much less ambiguous as OMT, which makes transition to SDL less restrictive
- two ways to go from UML to SDL:
  - put rules in translation process
    - no extensions/restrictions on UML notation, but possibly incomplete translation
  - put rules in design process
    - stereotypes and properties to reflect SDL concepts
    - reverse mapping
- IDL specification of UML is a good starting point for the specification of mapping rules and the tool development