UML-based tool

for Constructing Component Systems

via

Component Behaviour Inheritance

Ella Roubtsova

The Nederlands

Serguei Roubtsov

Finland

ERCIM fellowship

Jump to first page

Outline

- Component and Component Specification;
- The need of a tool support for component specifications reuse.
- Tool idea. Process semantics of component specification with process inheritance is the basis of reuse .
- Reuse of behavioral patterns and design debugging .

Jump to first page

Component and Component specification

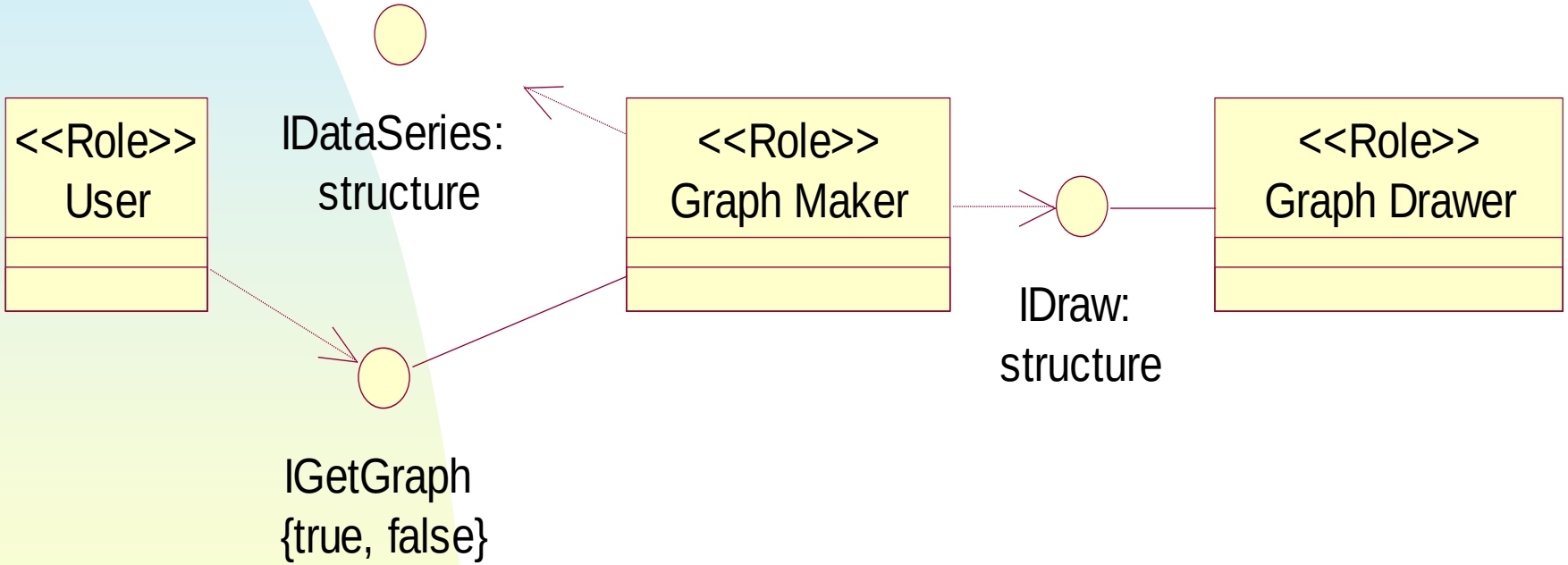
Component specification is an abstraction from component implementation keeping :

- Contractually specified interfaces and explicit dependencies
- Independent deployment
- Composition by third parties

Jump to first page

Behavioural pattern $P = (A, z, z^*, z_F, T)$

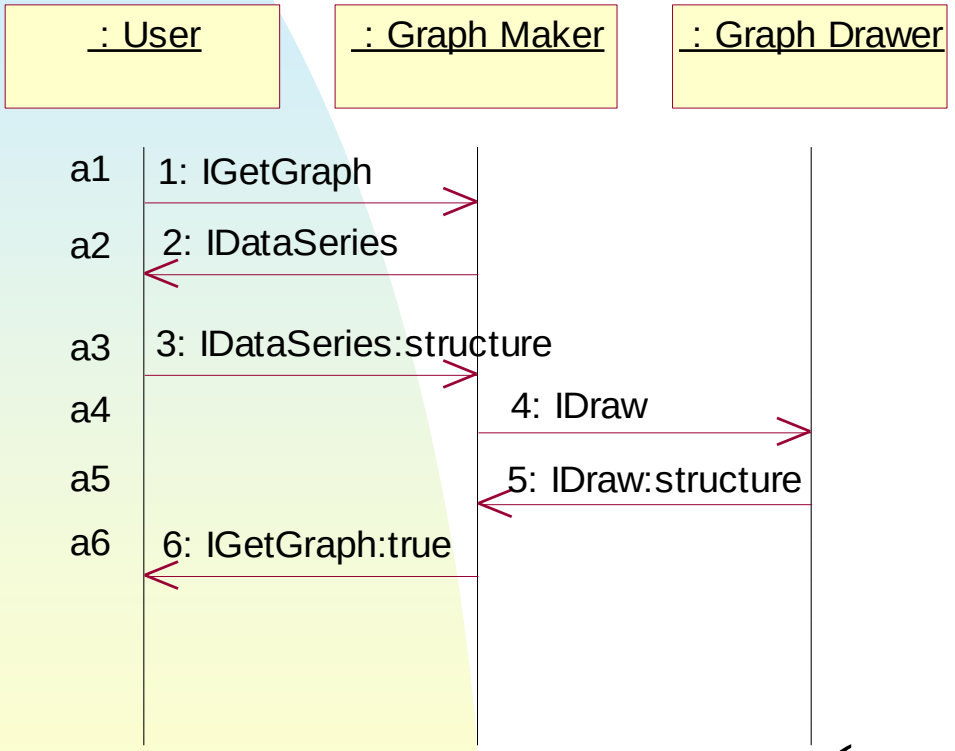
Interface-role diagram

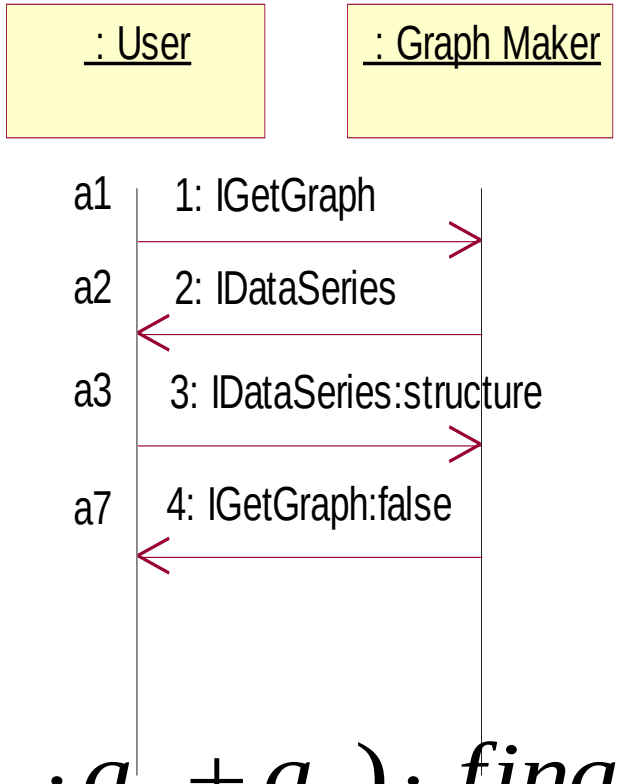


a = User.GraphMaker.IGetGraph : true

Jump to first page

Set of sequence diagrams specifies the structure of the behavioural pattern

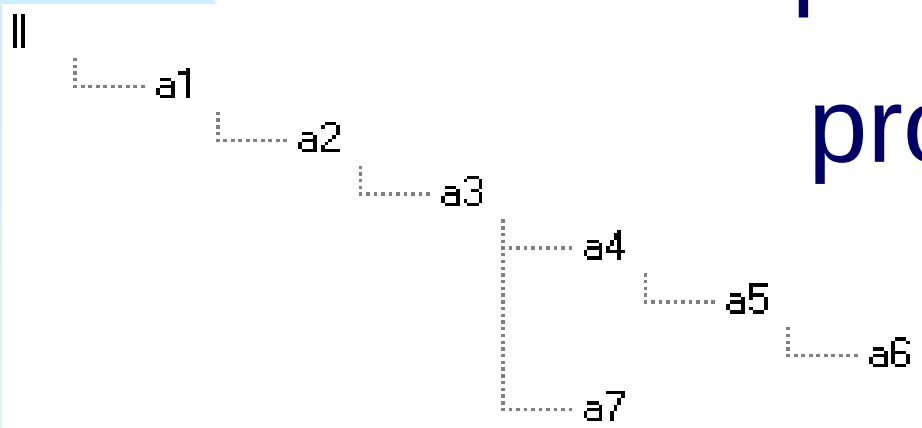




$$p = start \cdot a_1 \cdot a_2 \cdot a_3 \cdot (a_4 \cdot a_5 \cdot a_6 + a_7) \cdot final$$

Jump to first page

Behavioural pattern = process term = process graph



$$p = start \cdot a_1 \cdot a_2 \cdot a_3 \cdot (a_4 \cdot a_5 \cdot a_6 + a_7) \cdot final$$

a_1= User. Graph Maker. IGetGraph

Jump to first page

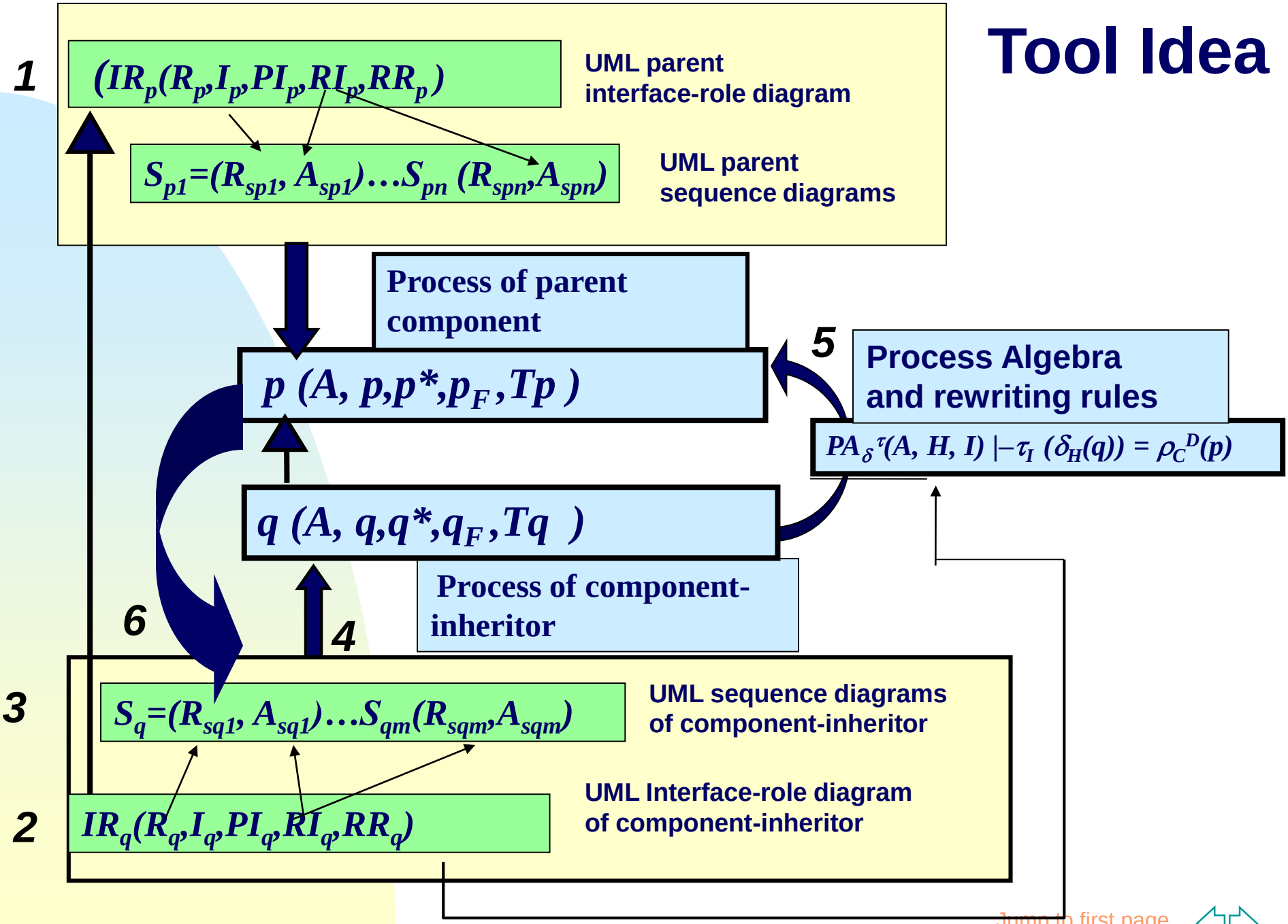
Motivation

Why do we need a UML based tool supporting constructing component systems?

- Consistent specification of a component is a time consuming task.
- Composition of component specifications should not spoil specified behaviour of components .

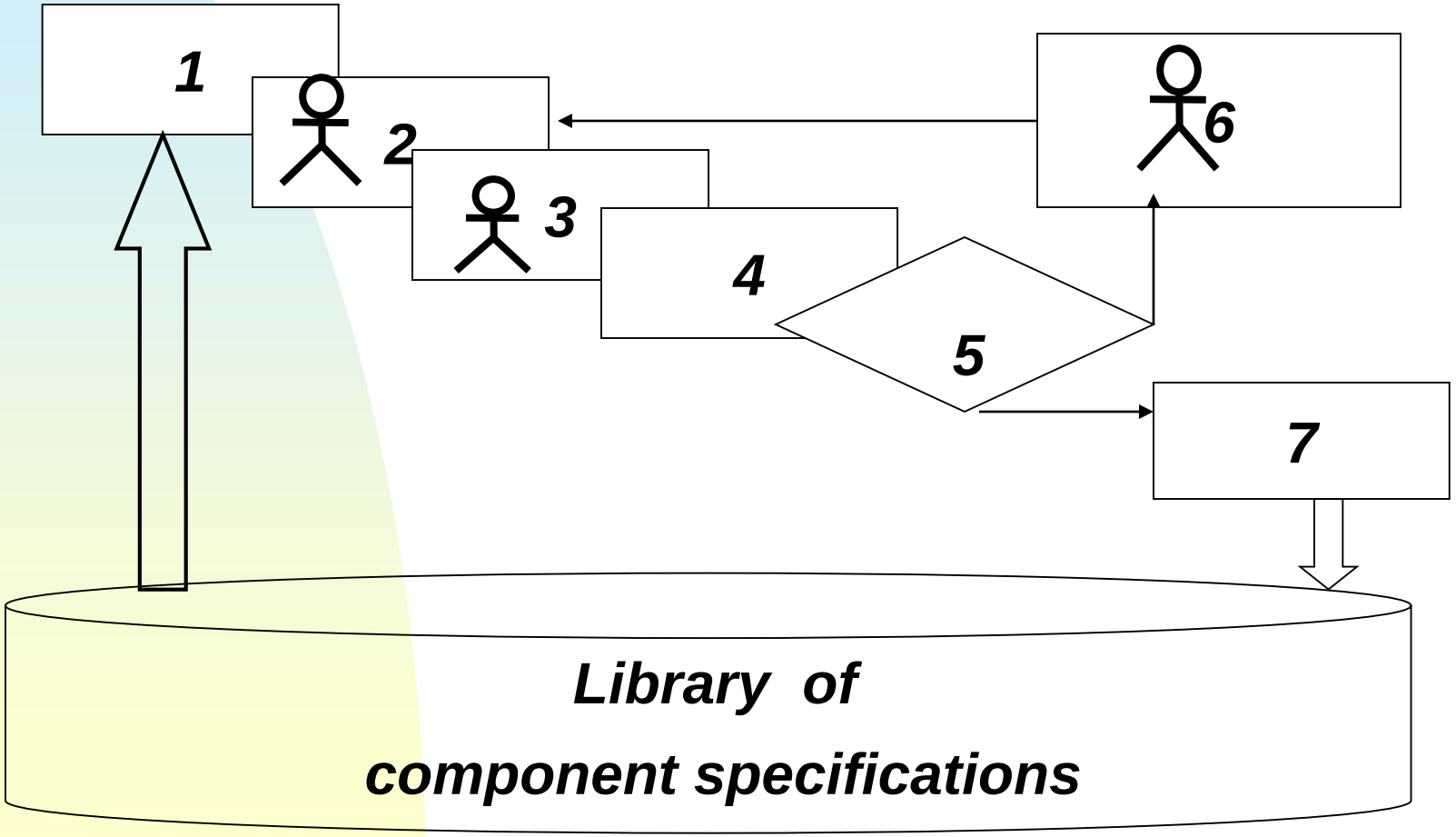
Jump to first page

Tool Idea



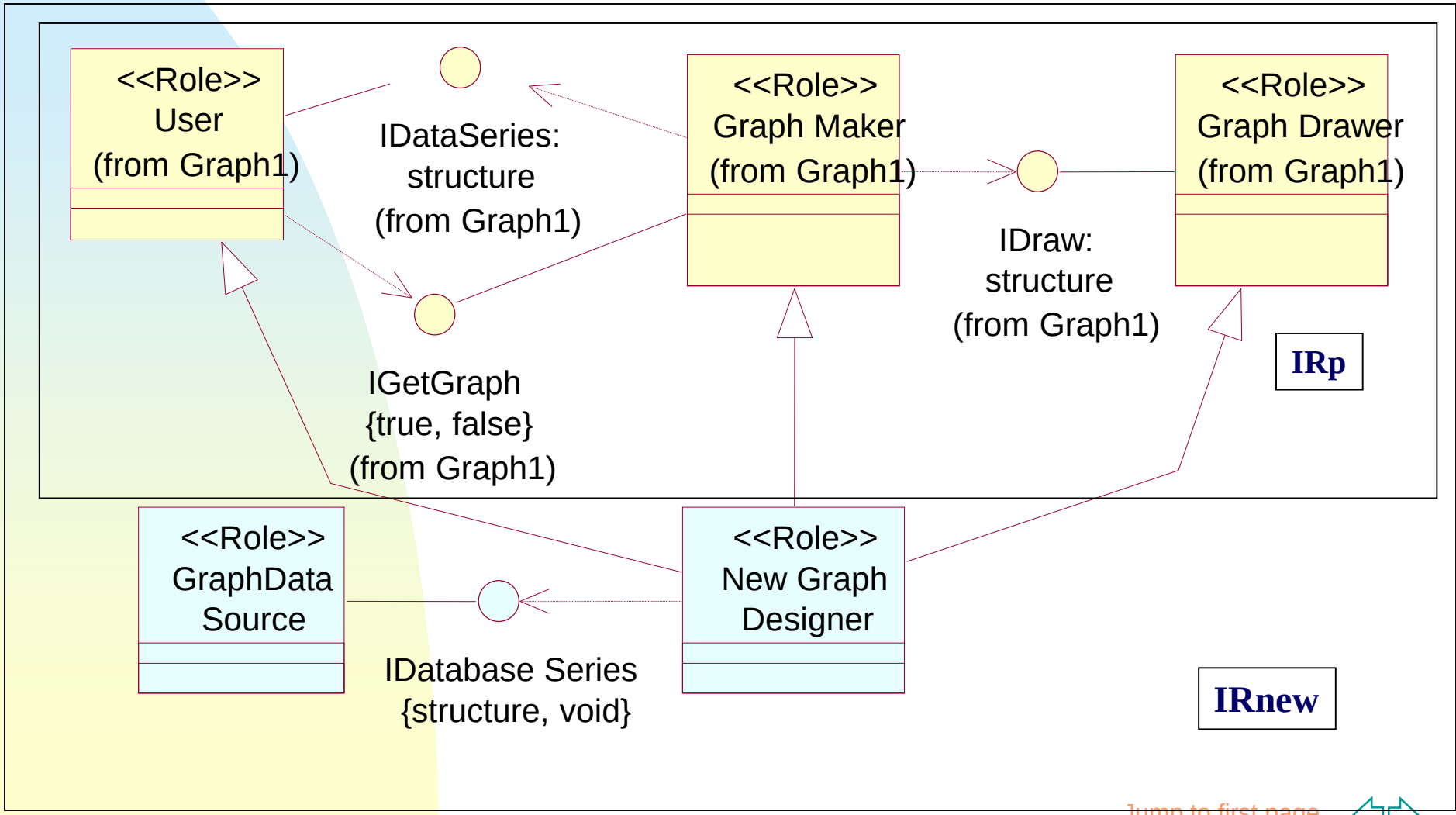
Jump to first page

Constructing component systems as a workflow



Jump to first page

Reuse of role-views (1,2)



Jump to first page

Rules for Interface-Role Diagram Reuse

1. Create new roles $Rp \cap Rnew = \varnothing$

2. To inherit an element of communication, inherit the role which provides the interface and the role which requires the interface

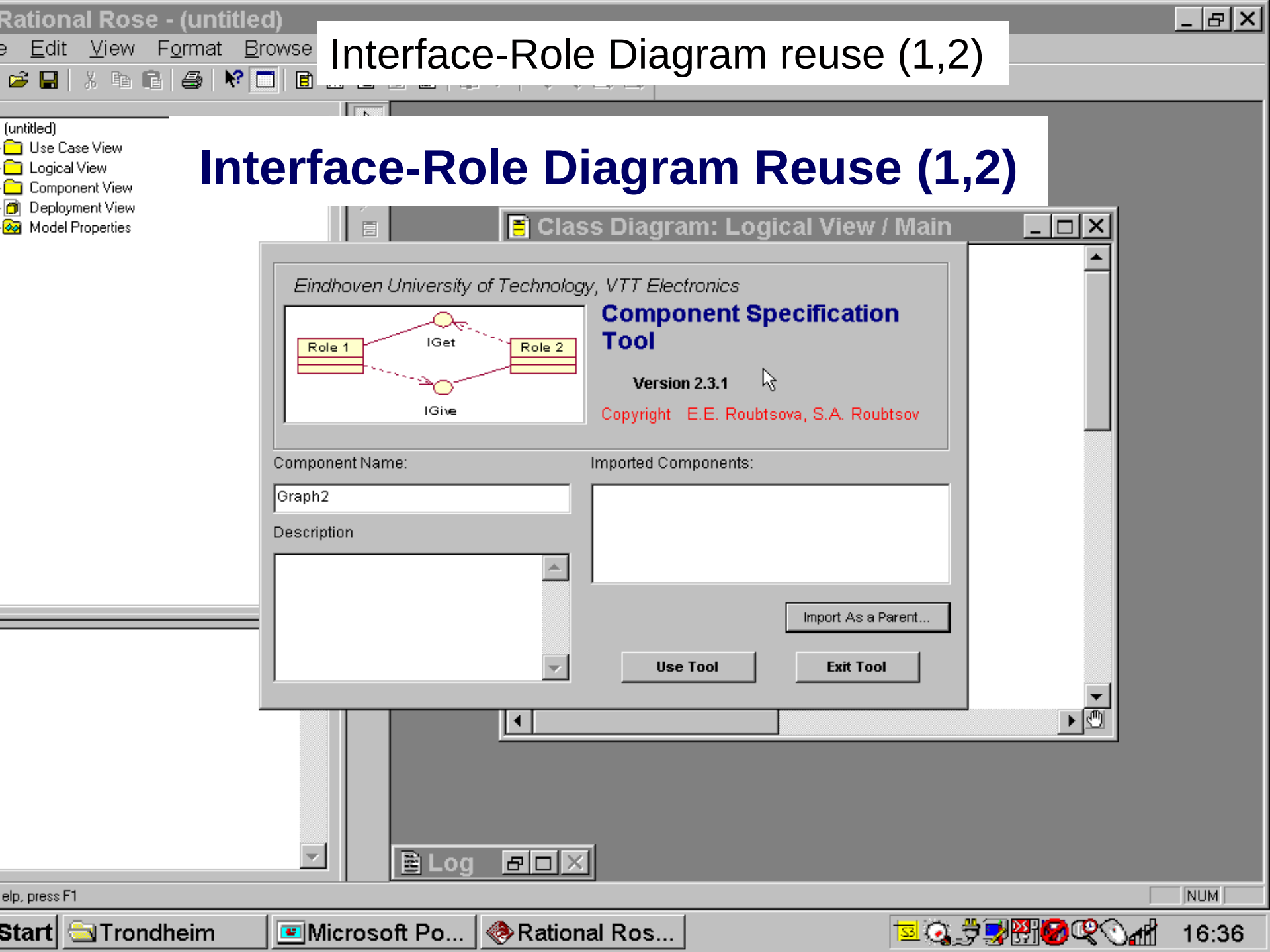
$RRnew \neq \varnothing$, $RIq = RIp \cup RInew \cup RId$;

3. Create new interfaces provided and required by new roles: $Ip \cap Inew = \varnothing$,

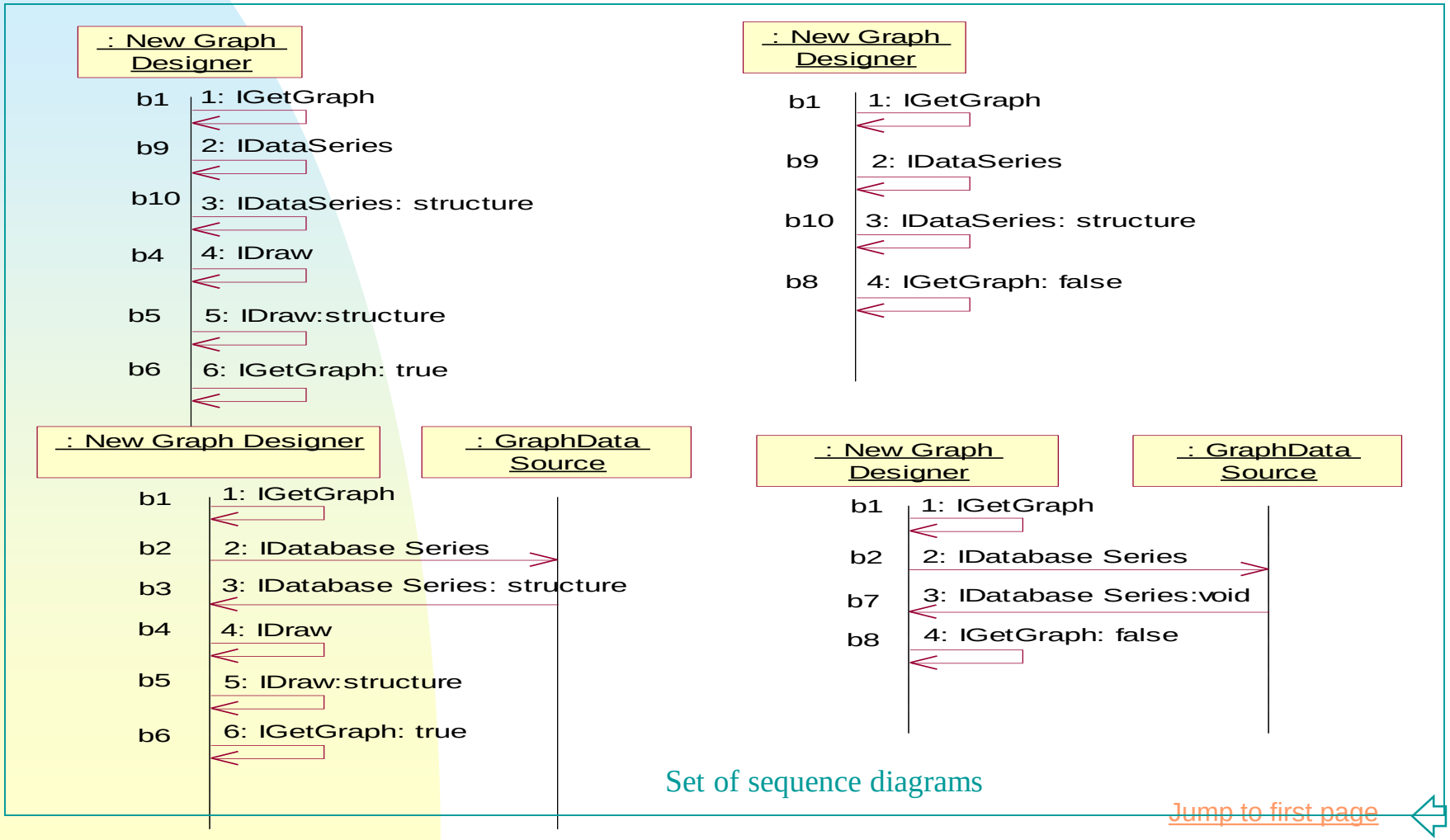
Jump to first page

Interface-Role Diagram reuse (1,2)

Interface-Role Diagram Reuse (1,2)



Reuse of sequence diagram sets (3)



Set of sequence diagrams

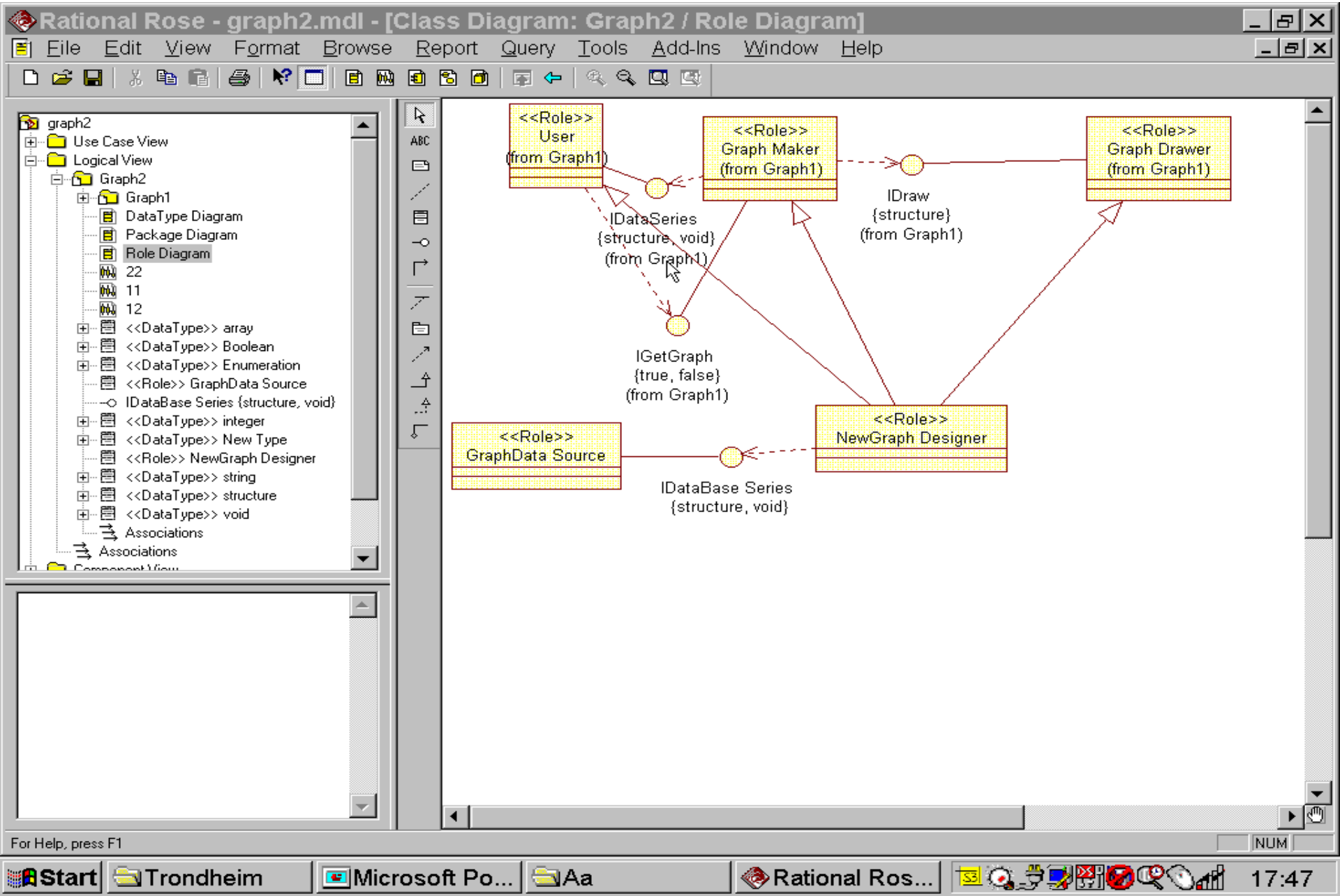
Jump to first page

Rules for Sequence Diagram Set Reuse

1. Design a sequence diagram set from actions specified by interface-role diagram ;
2. Tool constructs the process corresponding the sequence diagram set ;
3. Choose the parent from the set of parents;
4. Tool checks the inheritance of the process of this parent;

Jump to first page

Design a sequence diagram (3)



Practical use of process inheritance in the UML

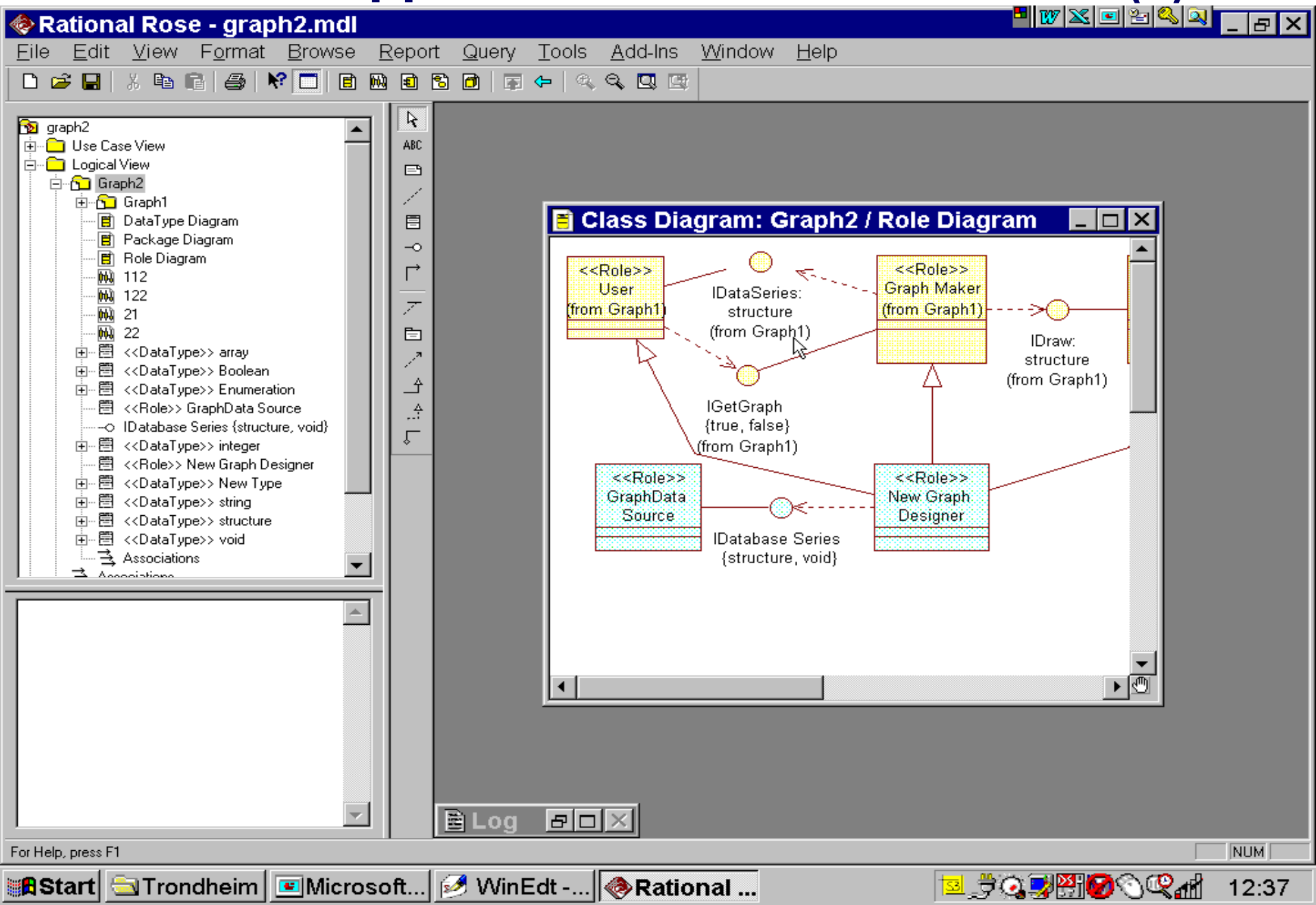
$$PA(A,H,I+RN) \;/\!\!-\!\!- \;\tau_I(\delta_H(q)) = \rho_p{}^d(p)$$

How to find from the interface-role diagram

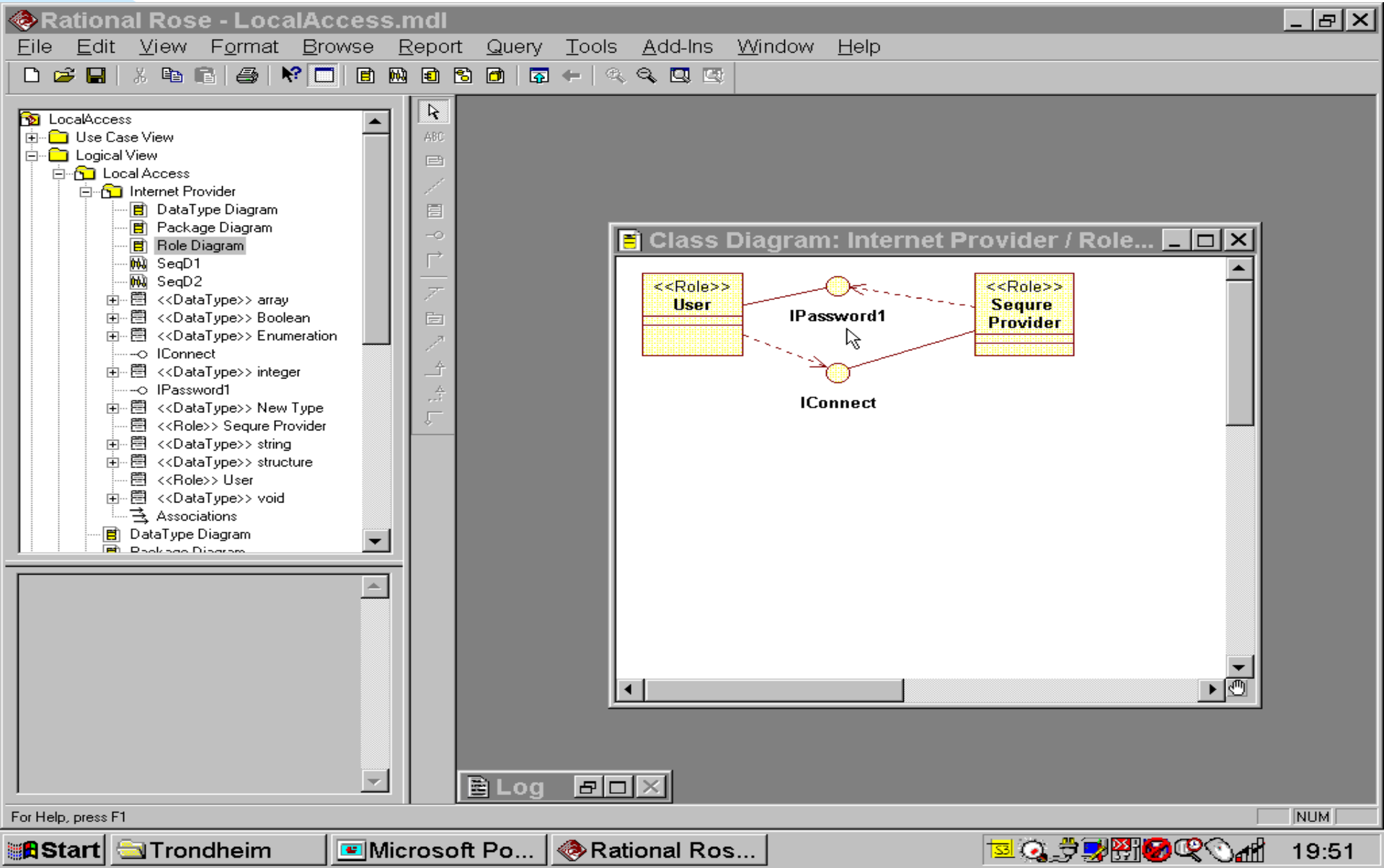
- the renaming function for the parent process
- the set of all actions *A* for the process algebra
- the set of actions *H* to be blocked;
- the set of actions *I* to be hidden;

Jump to first page

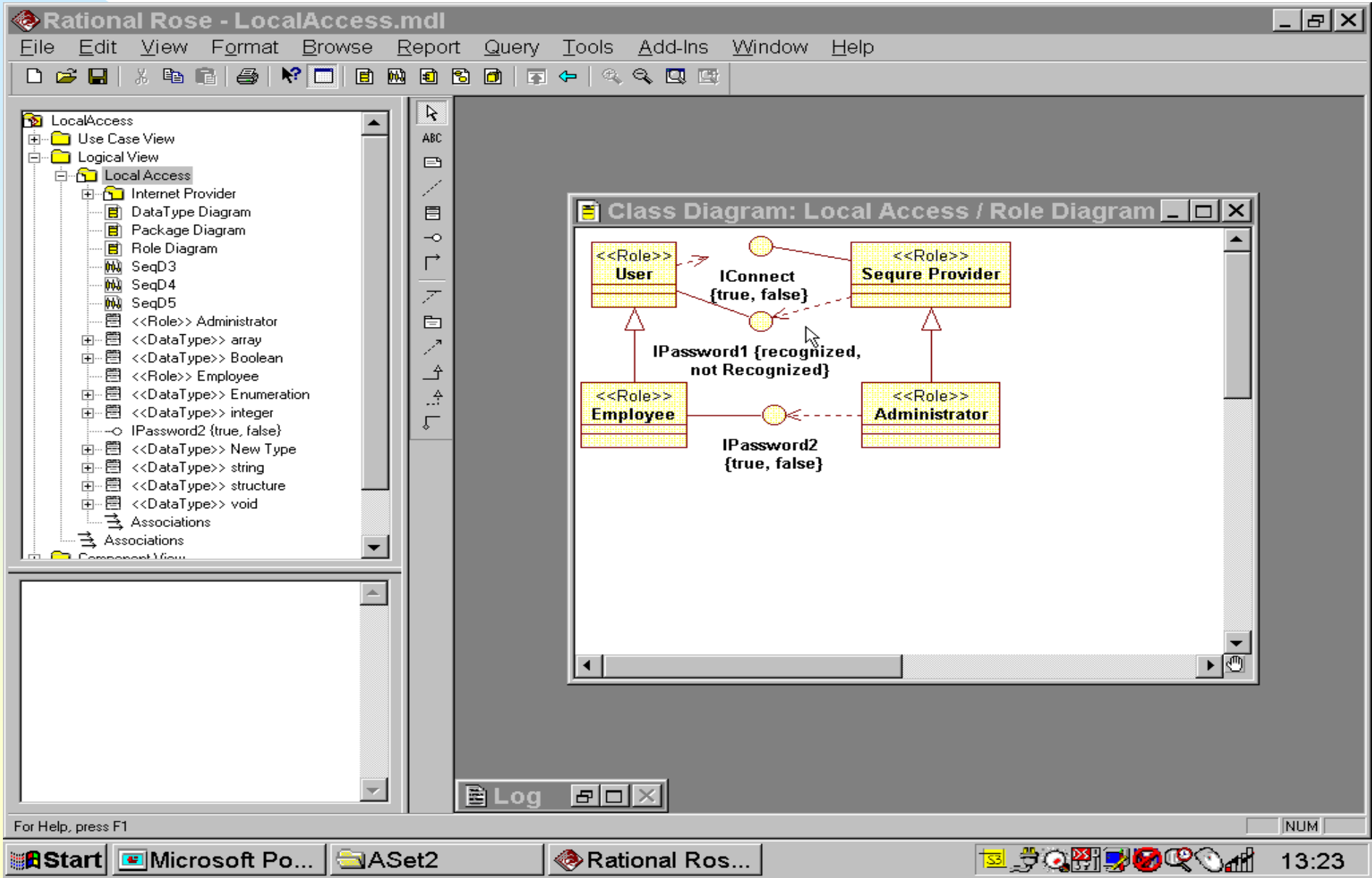
Tool support for inheritance checks (4)



Debugging of Specifications (5)



Debugging of Specifications (6)



Conclusion

• Tool does not change the current practice of component specification, it does not demand any specific skills from designers (formal model and methods are built into the tool).

• Advantages and results of the tool methodology:

- Consistent component specifications in the UML;
- Correct reuse of component specifications;
- Decreasing of the design time;

E.Roubtsova@tue.nl

ext-Serguey.Roubtsov@vtt.fi

Jump to first page