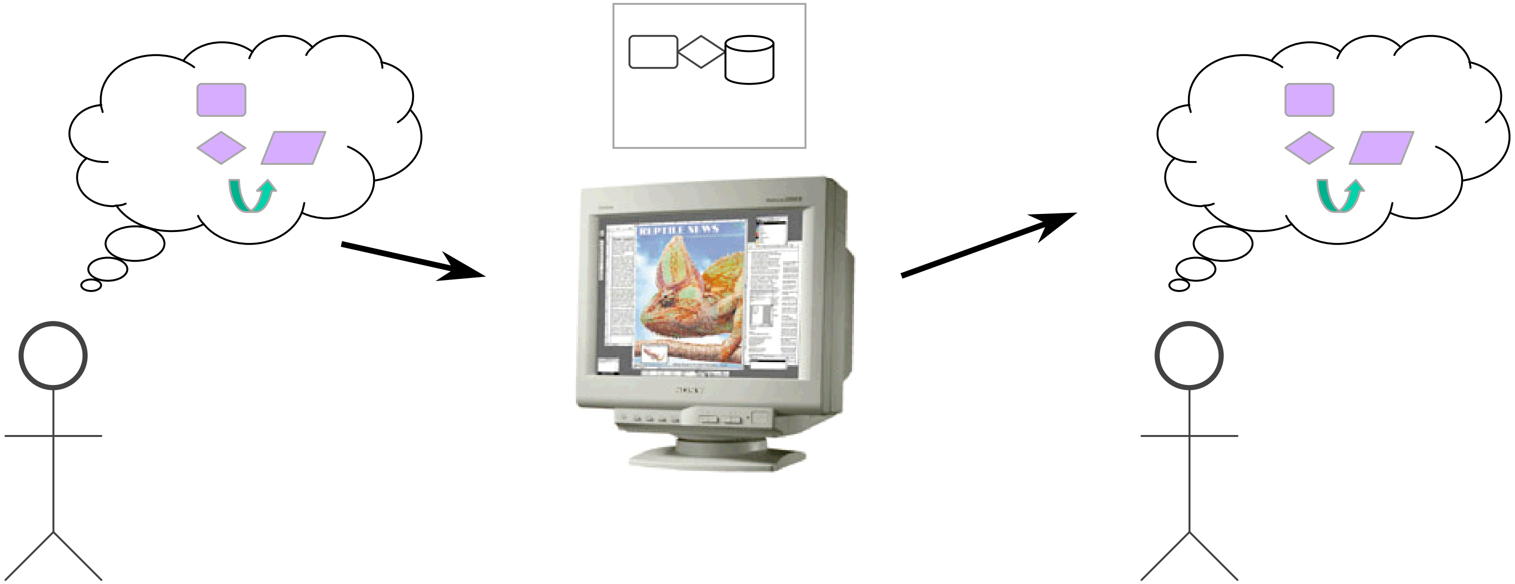
Stockholms universitet

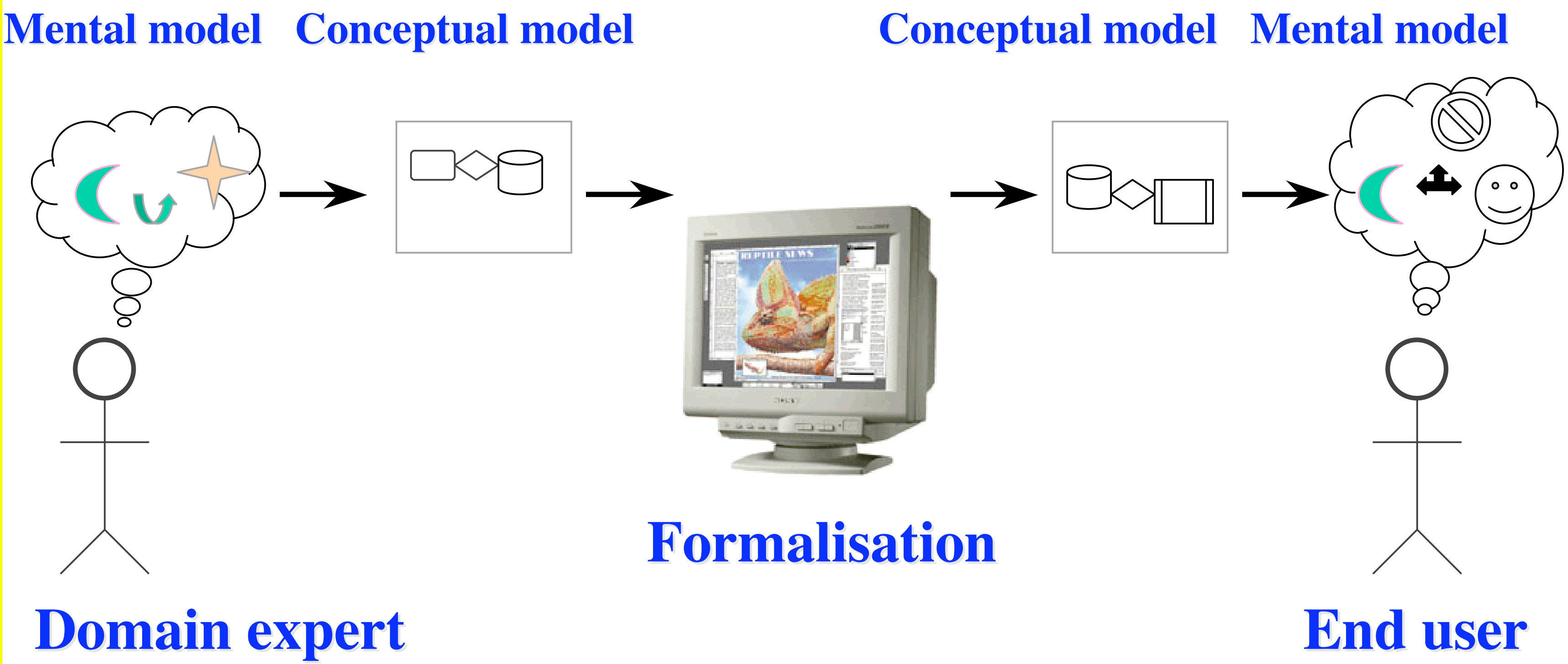
Using User-centred Knowledge Model (t-UCK) as a Modelling Support

Anne Håkansson
Stockholm University
Anne.Hakansson@dsv.su.se

The ultimate solution



Reality



Mental model

Conceptual model

Conceptual model

Mental model

Formalisation

Domain expert

End user

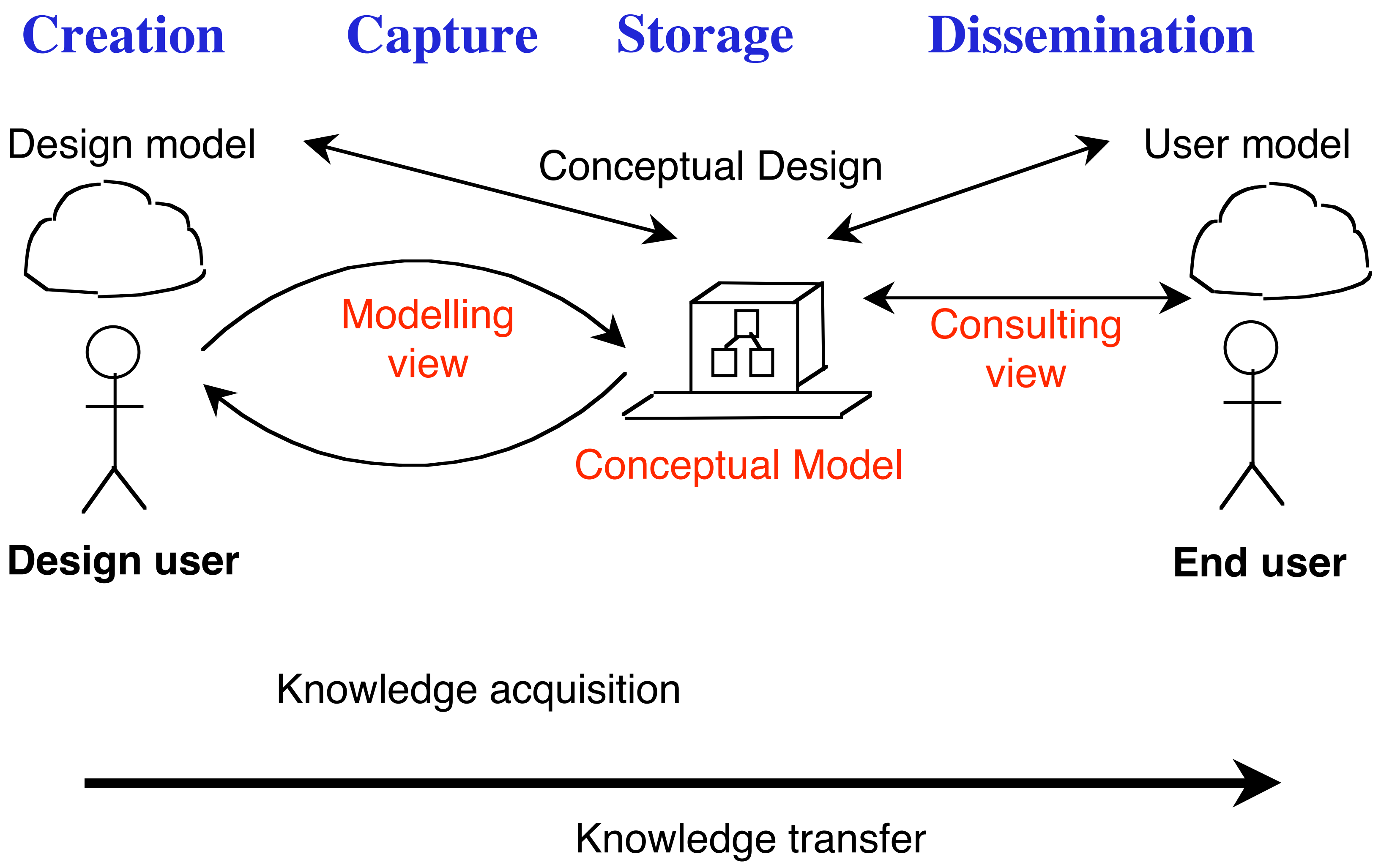
Need for t-UCK

- Modelling knowledge-intensive systems
  - *Disseminate knowledge within organisations*
  - *Support using knowledge in new/other situations*
- Incorporate the users
  - *Supply/ use required knowledge, change/ understand reasoning strategy, provide, utilse additional functionality*
- Support using and understanding the contents of the system
  - *Provide same views of the contents*

The User-centred knowledge model

- Knowledge transfer
  - *Design user(s) - System - End users*
  - *Transfer domain specific knowledge*
  - *Involves knowledge acquisition / elicitation*
- Conceptual design
  - *Bridge the gap between design model - user model*
- Conceptual model
  - *Modelling view and Consulting view*

The User-centred knowledge model



Conceptual model

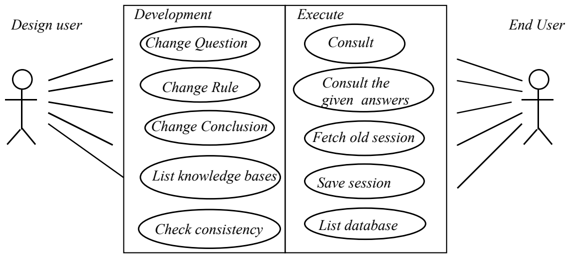
- Framework for Developing and Consulting
- Clarify different terms and support applying these

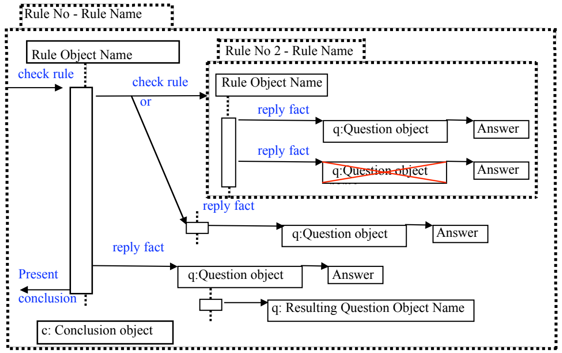
  (questions - rules - conclusions)
- Constitutes the Graphical User Interface
- Transparent and reflects the contents

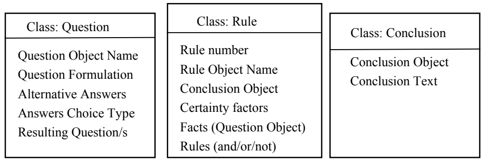
Modelling view and Consulting view

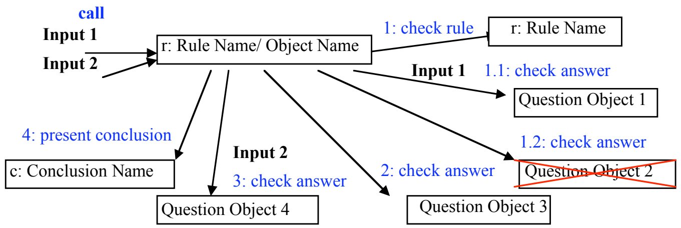
- Support a number of tasks and users
- *Design user* views for learning and designing
  - Evaluate contents and reasoning
  - Find lack of knowledge and faults
- *End user* views for learning and operating
  - Understand conclusions and reasoning
  - Find values, rules and understand reasoning

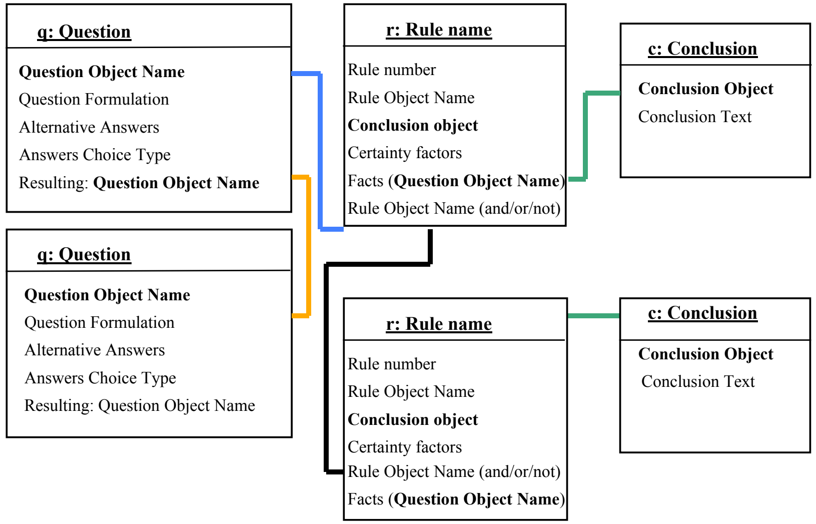
User Interface

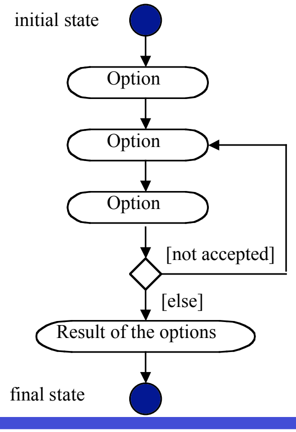


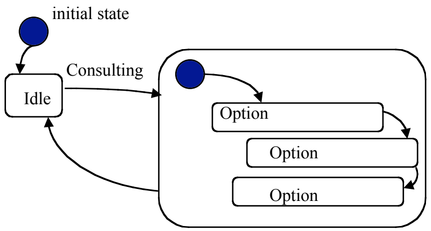




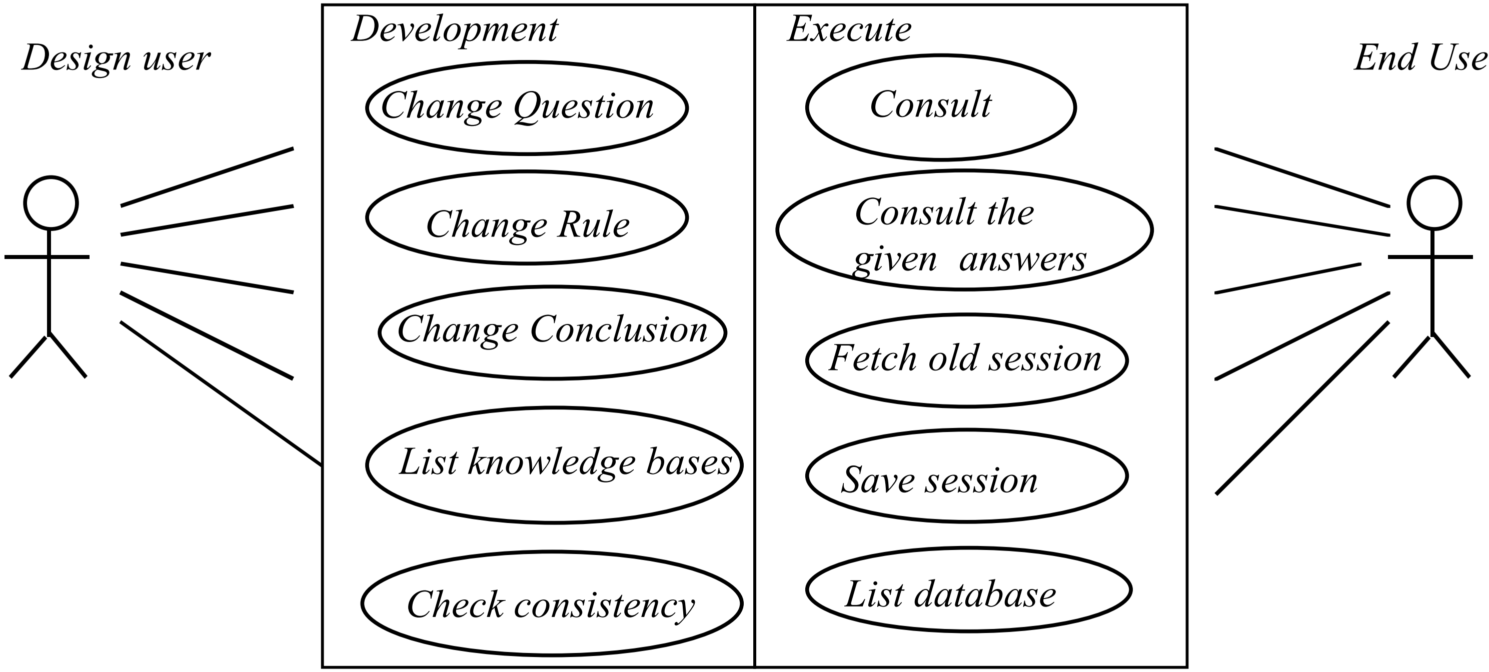






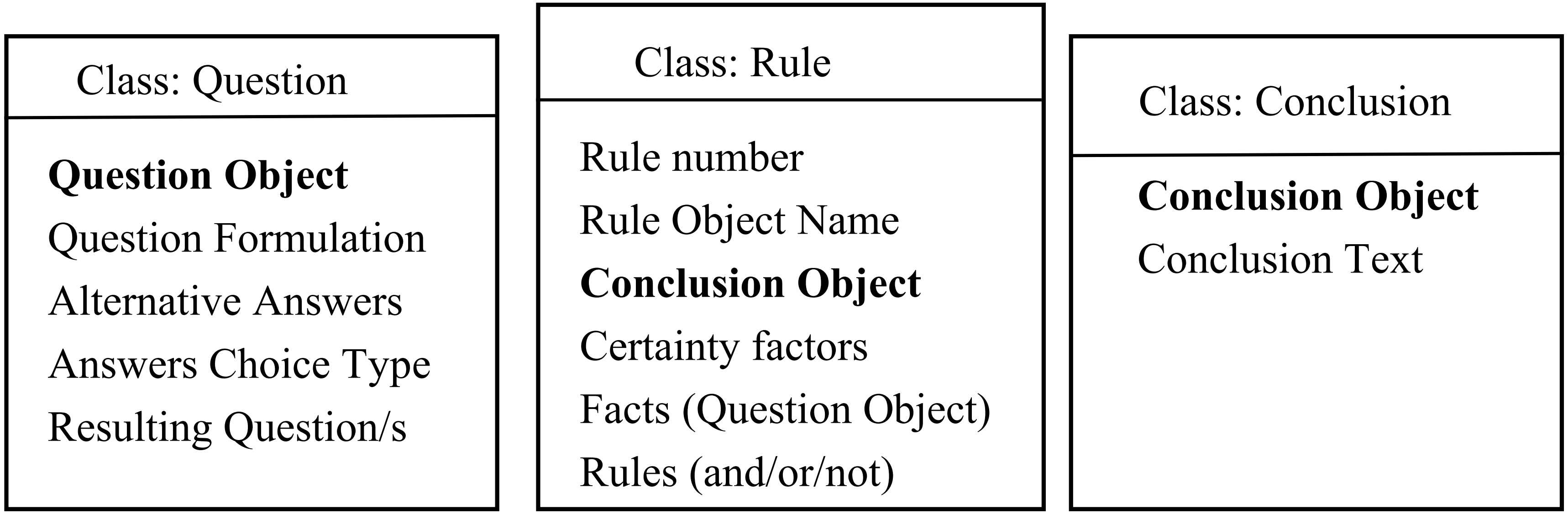


UML Use Case diagram



- Describes the options to be performed and executed
  - Design users use the system for developing
  - End users use the system for advices / conclusions

UML Class diagram



Class: Question

Question Object
Question Formulation
Alternative Answers
Answers Choice Type
Resulting Question/s

Class: Rule

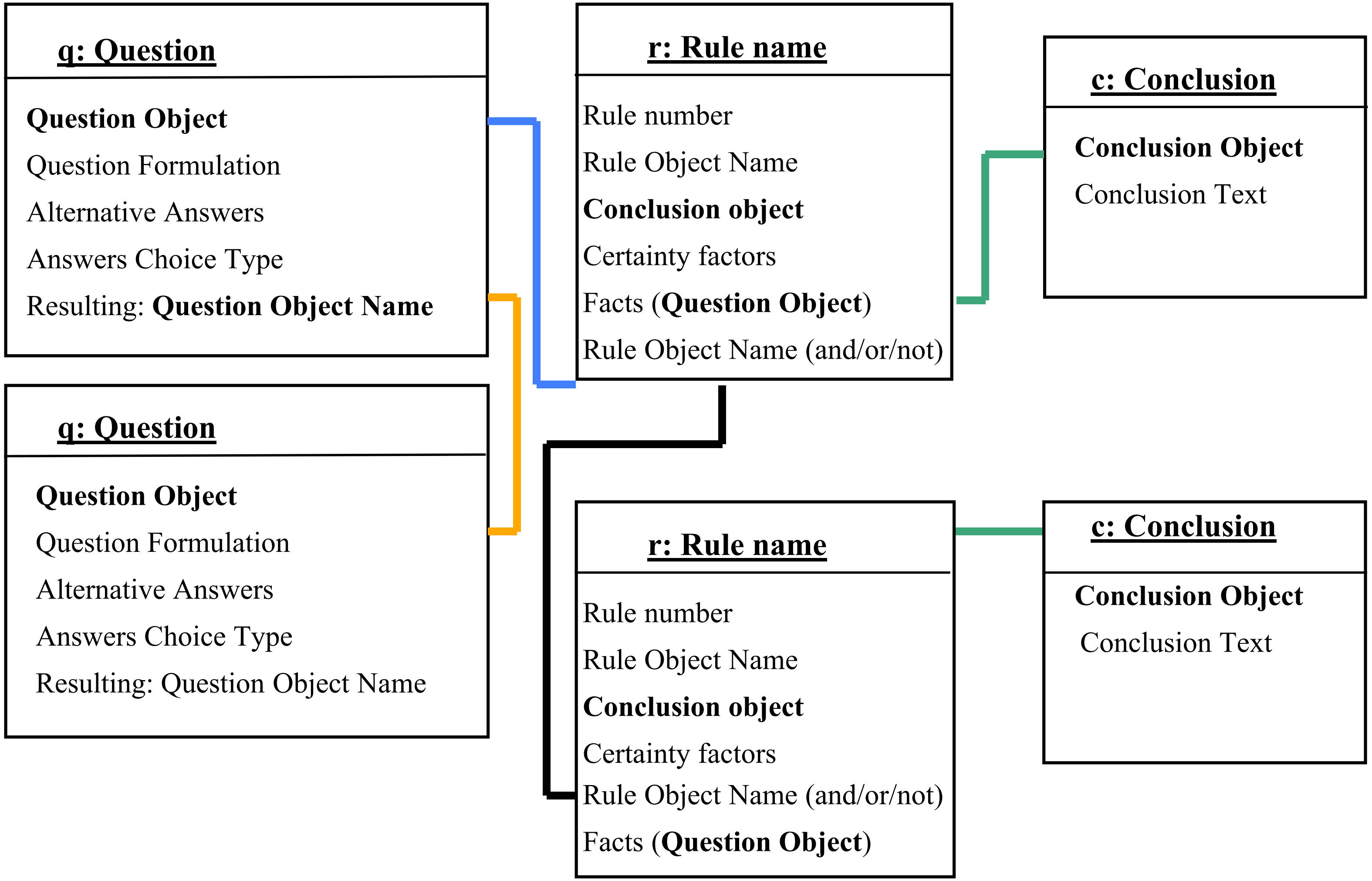
Rule number
Rule Object Name
Conclusion Object
Certainty factors
Facts (Question Object)
Rules (and/or/not)

Class: Conclusion

Conclusion Object
Conclusion Text

- Templates for questions, rules and conclusions
  - Design users insert new question / rule / conclusion
  - End users use these objects during consultation

UML Object diagram



q: Question

Question Object
Question Formulation
Alternative Answers
Answers Choice Type
Resulting: **Question Object Name**

q: Question

Question Object
Question Formulation
Alternative Answers
Answers Choice Type
Resulting: Question Object Name

r: Rule name

Rule number
Rule Object Name
Conclusion object
Certainty factors
Facts (**Question Object**)
Rule Object Name (and/or/not)

r: Rule name

Rule number
Rule Object Name
Conclusion object
Certainty factors
Rule Object Name (and/or/not)
Facts (**Question Object**)

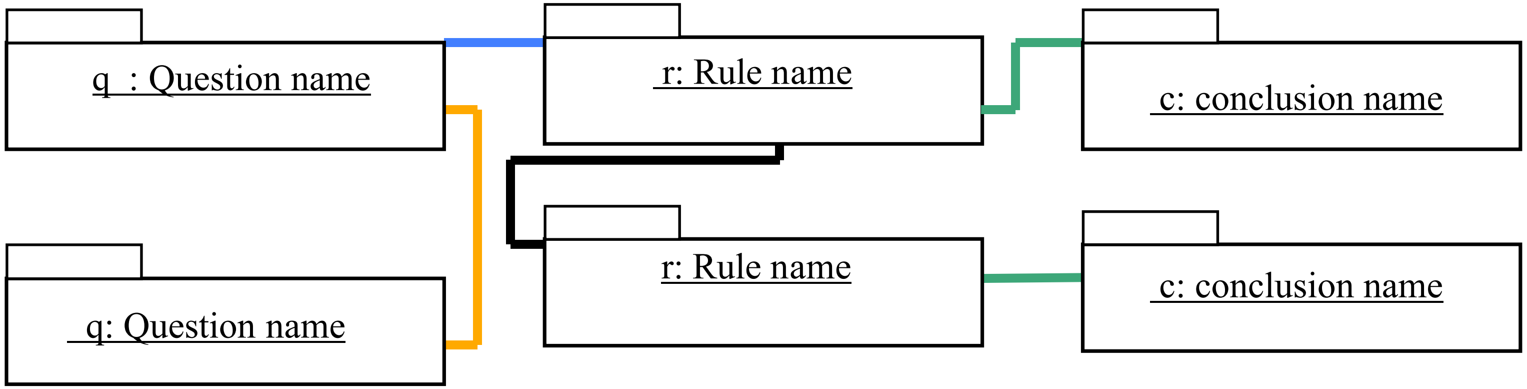
c: Conclusion

Conclusion Object
Conclusion Text

c: Conclusion

Conclusion Object
Conclusion Text

UML Packages

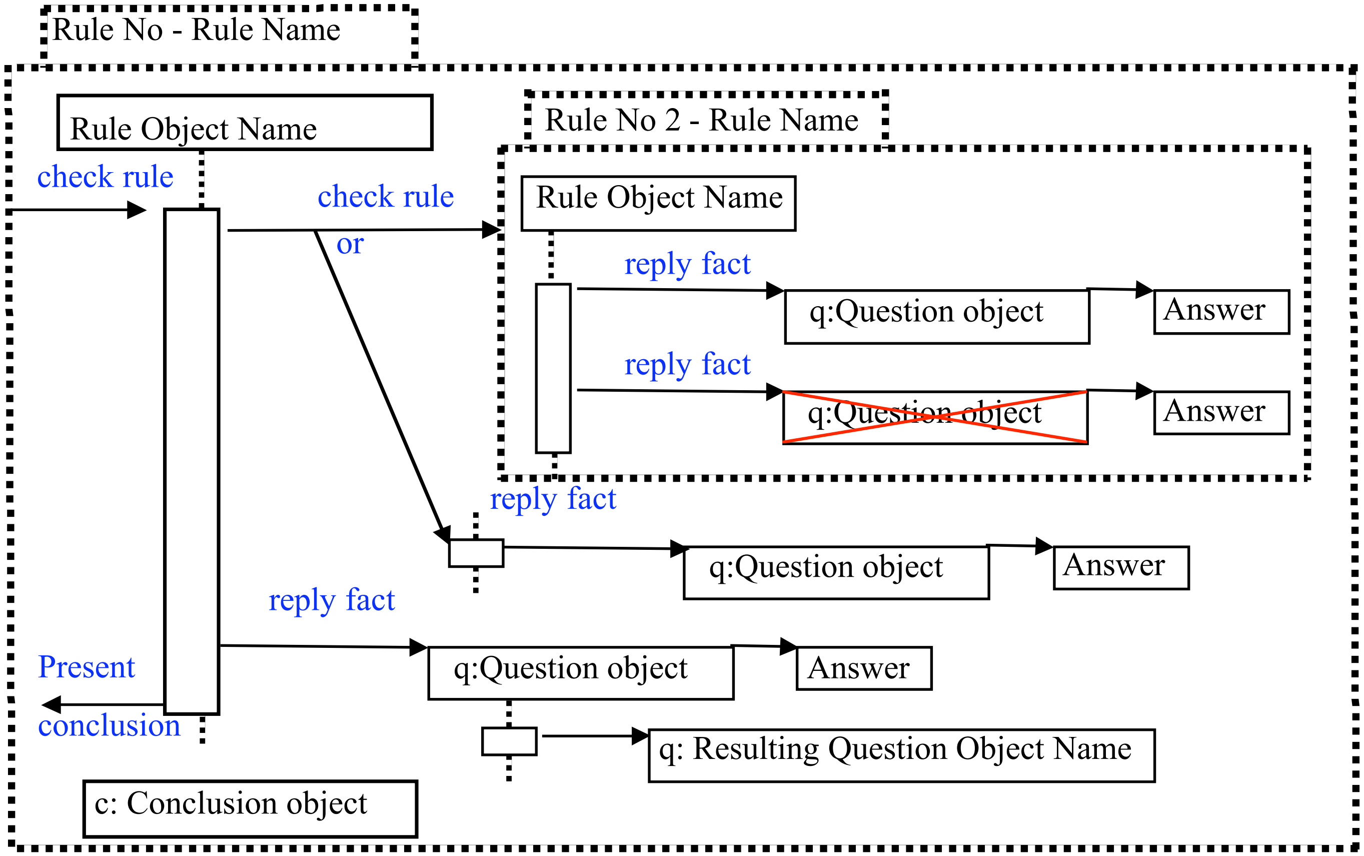


Packages encapsulate data

– decrease the complexity of the system.

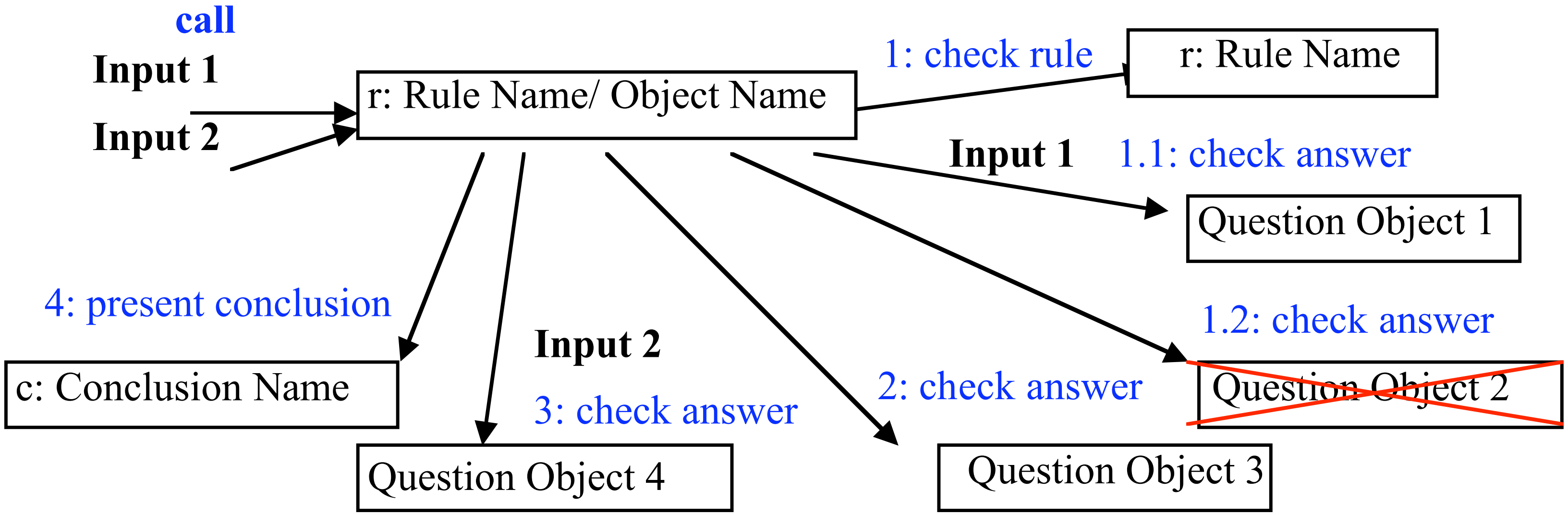
– tightly connected rules can be put into a package

Sequence diagram



Design users and end users can observe dependencies between rules and facts (developing / explanations)

Collaboration diagram



call

Input 1

Input 2

r: Rule Name/ Object Name

1: check rule

r: Rule Name

Input 1

1.1: check answer

Question Object 1

1.2: check answer

~~Question Object 2~~

2: check answer

Question Object 3

4: present conclusion

c: Conclusion Name

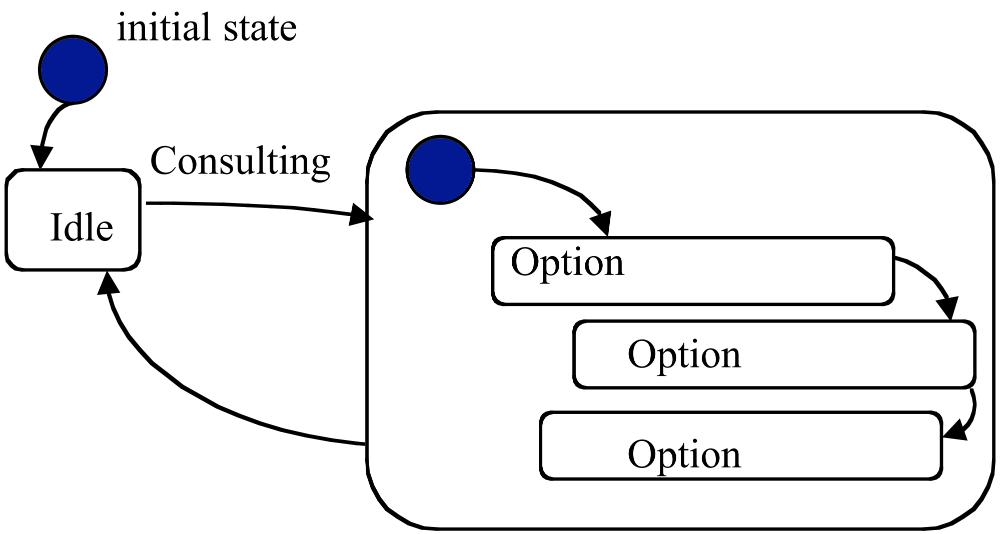
Input 2

3: check answer

Question Object 4

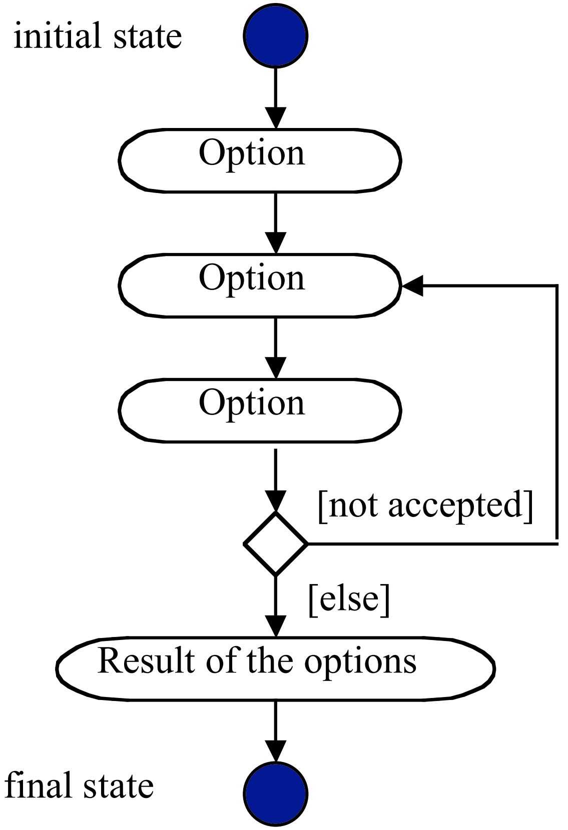
User provides inputs. Design users and end users can observe input effects on rules and conclusions.

UML State Chart diagram



State chart diagram describes the options in the code and dependencies of these options. Develop / Use other functionalities (static).

UML Activity diagram



Activity diagram shows procedural behaviour of a declarative representation. Develop / Use (dynamic)

Examples of KMS

- Visual knowledge modelling of information logistics processes
  - Sending e-invoice between companies:
    - *A SME sends an invoice to another SME that use the same enterprise system*
    - *A SME sends an invoice to an organization using a well established enterprise system (e g SAP)*
  - Automatic configuration:
    - *Building rules from ARIS*

Examples of KMS

- Communication protocol between supplier (sender) - customer (receiver)
- Several different rules

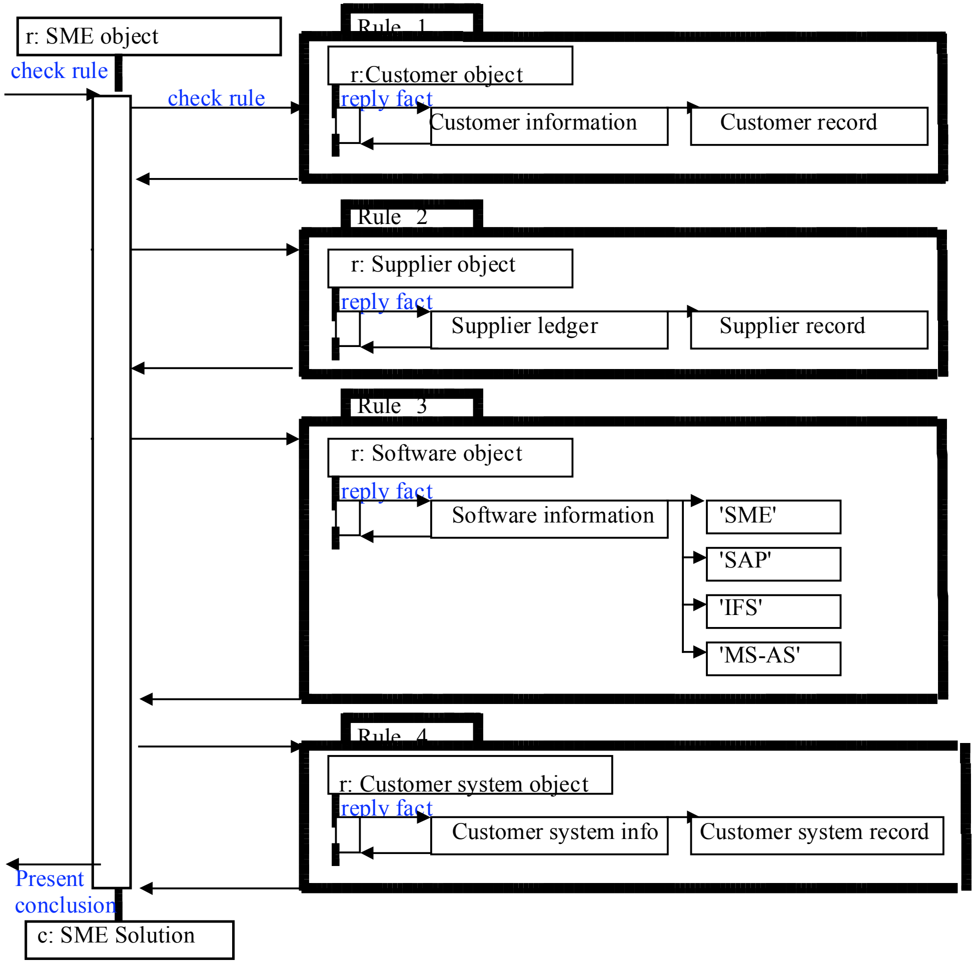
Rule1 - facts about the company

Rule2 - information about supplier ledger

Rule3 - kind of system the company uses: SME, SAP, IFS, or MS-AS

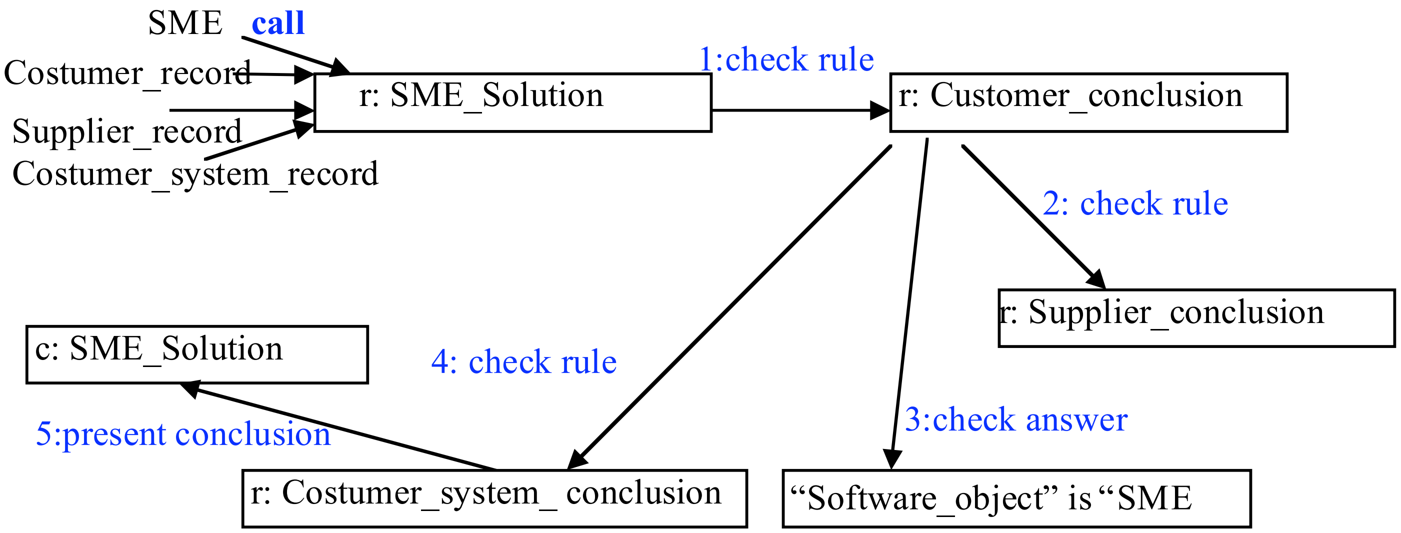
Rule 4 - current state of the system

Examples of KMS



A sequence diagram including rules in a knowledge base.

Examples of KMS



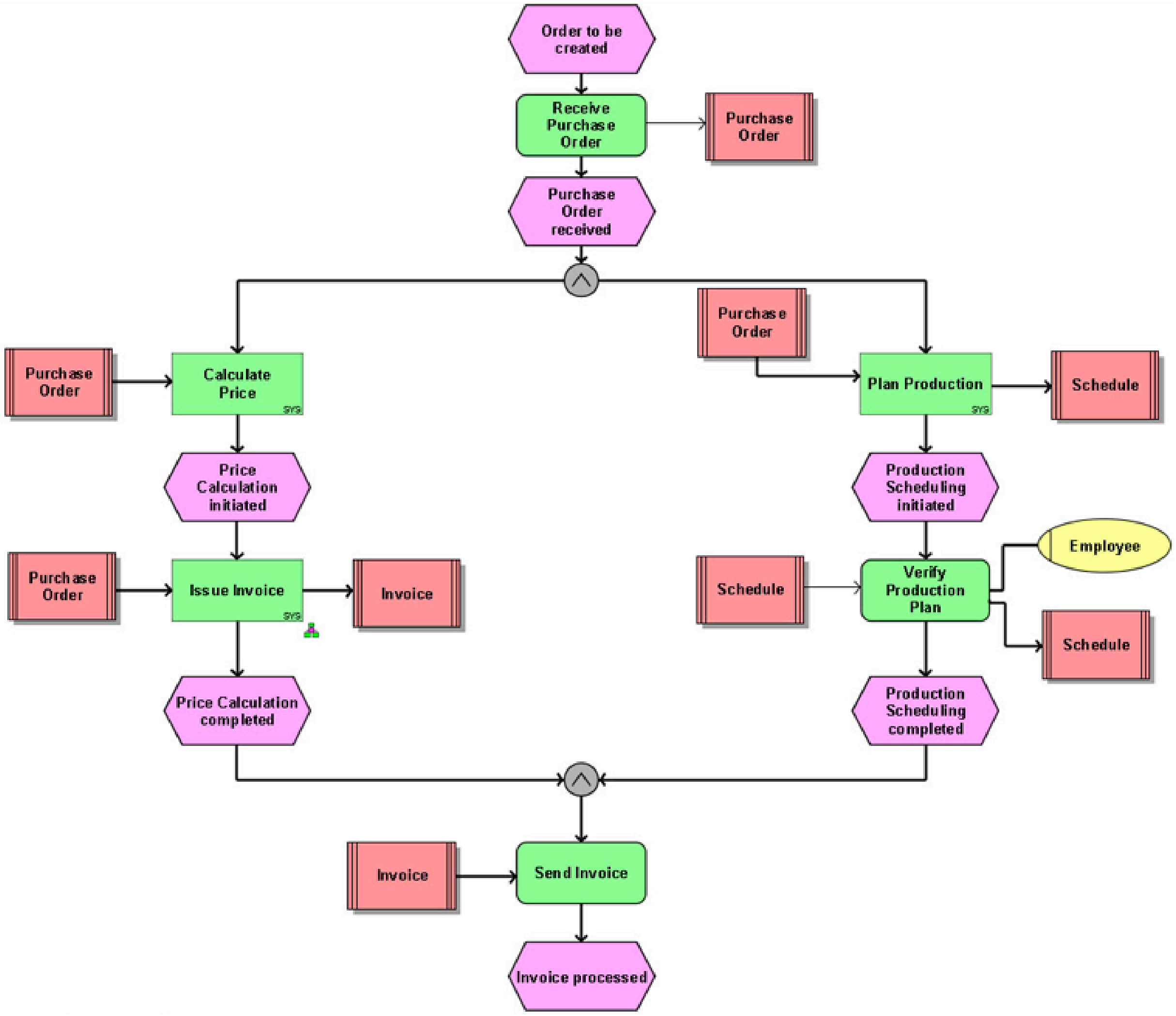
A dynamic presentation of rules for a conclusion.

Examples of KMS 2

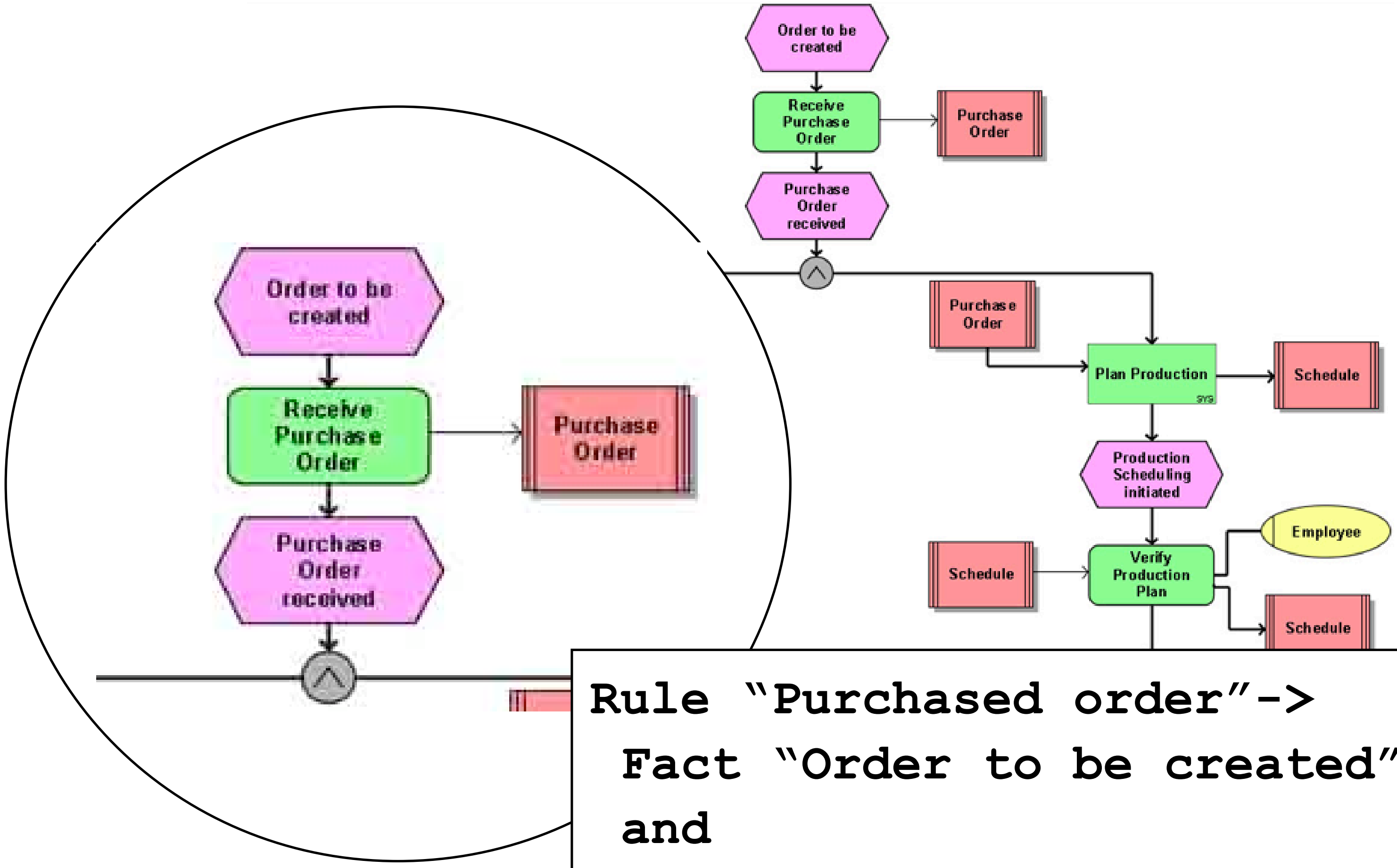
- Enterprise system configuration
  - Configure automatically
  - User specifies the contents
    - Standard packages
      - Modules

        General Ledger, Fixed Assets, Sales & Receivables, Purchase & Payables, Inventory, Manufacturing, Capital Requirements Planning, Human resources
      - Parameters
  - Adjust system from the specification
  - Building rules from requirements

Examples of KMS



Examples of KMS



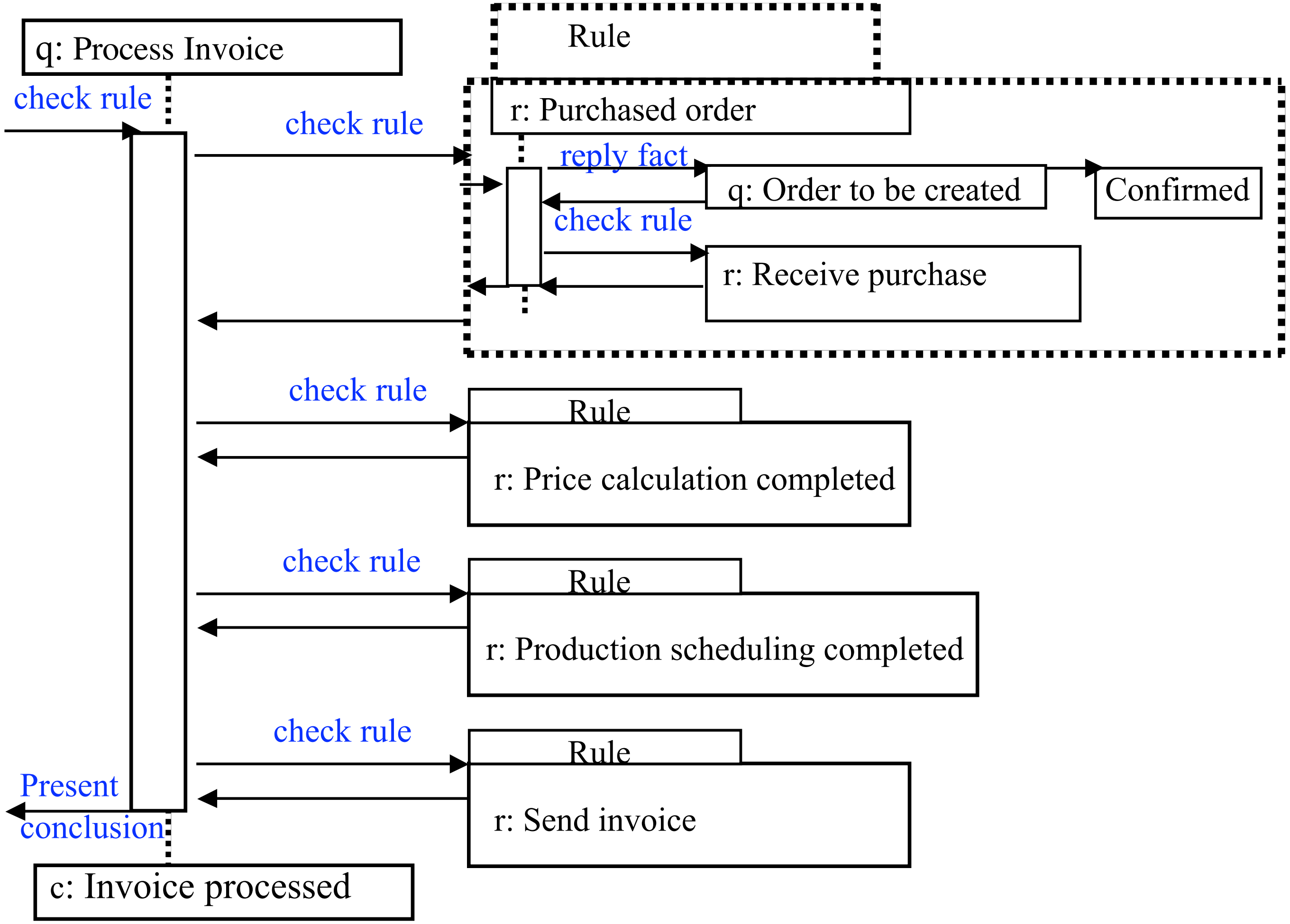
Rule "Purchased order"->

Fact "Order to be created"

and

Rule "Receive purchase order".

Examples of KMS



Questions ?