

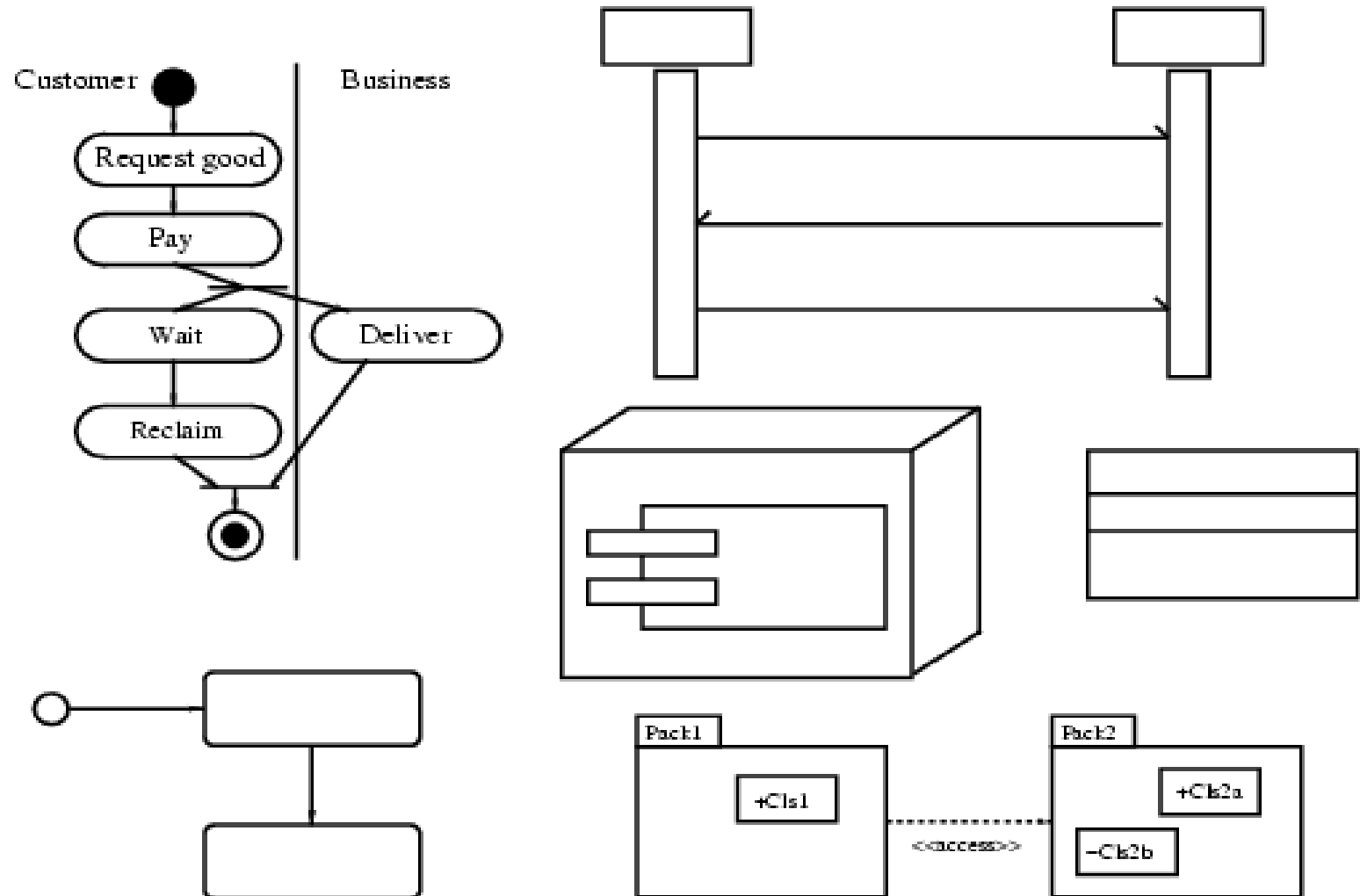
3 Modell-basierte Sicherheit mit UML

UML

Unified Modeling Language (UML):

- **visual** modelling for OO systems
- different **views** on a system
- high degree of **abstraction** possible
- de-facto industry **standard** (OMG)
- standard **extension** mechanisms

A Glimpse at UML



Used Fragment of UML

Use case diagram: discuss **requirements** of the system

Class diagram: data **structure** of the system

Statechart diagram: **dynamic** component behaviour

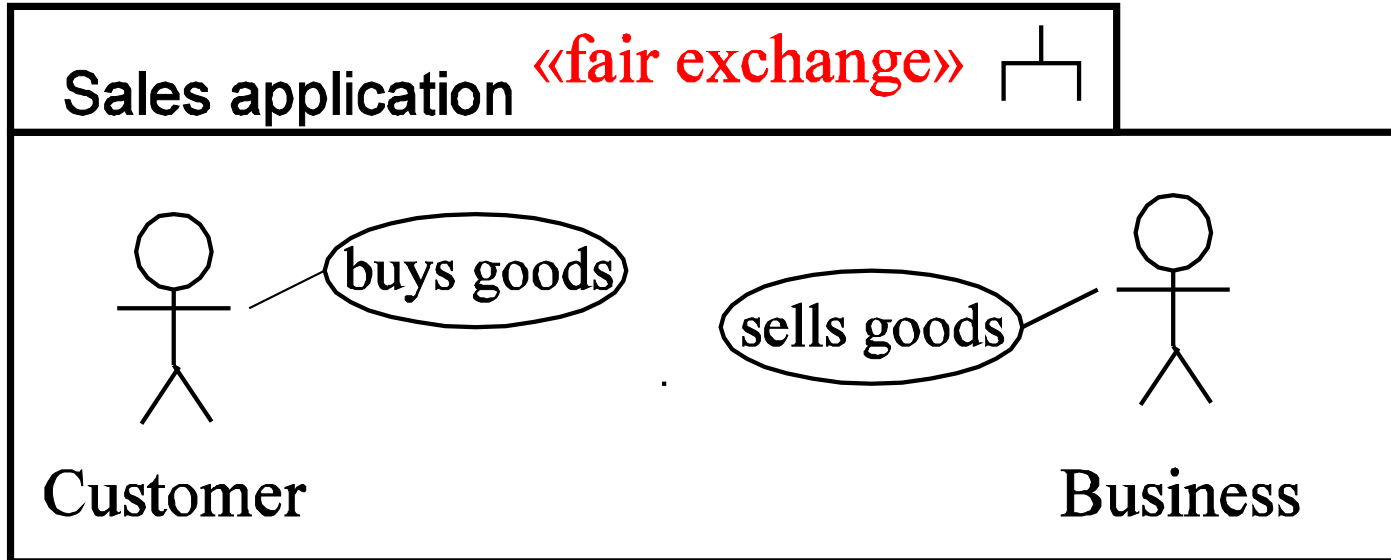
Activity diagram: flow of **control** between components

Sequence diagram: **interaction** by message exchange

Deployment diagram: physical **environment**

Package/Subsystem: **collect** diagrams for system part

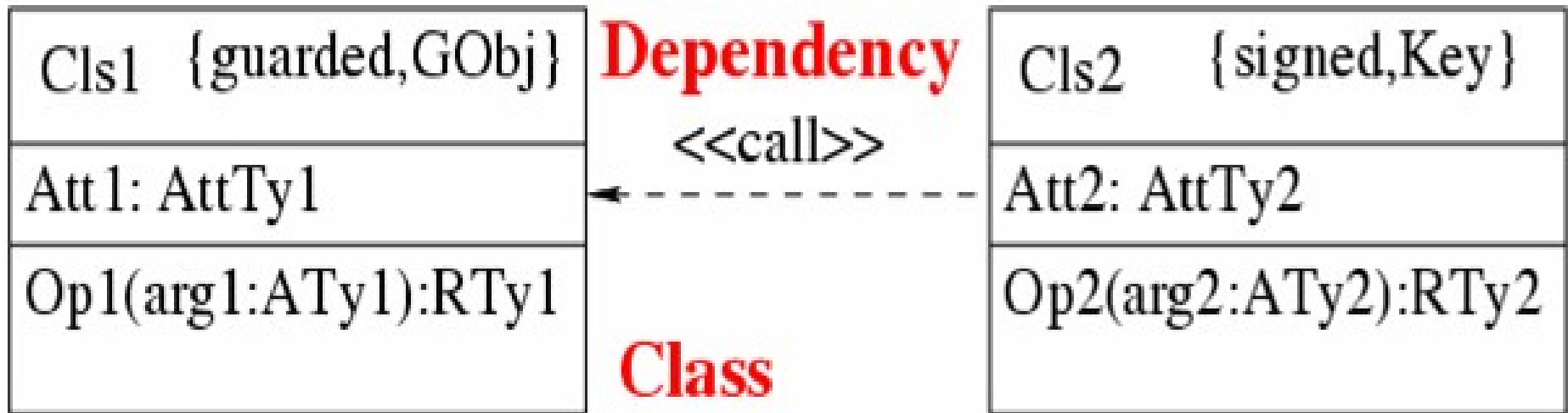
UML Run-through: Use Case Diagrams



Specify intended use cases of the system:
scenarios on the functionality offered to the
user or other systems.

Actors, linked to activities.

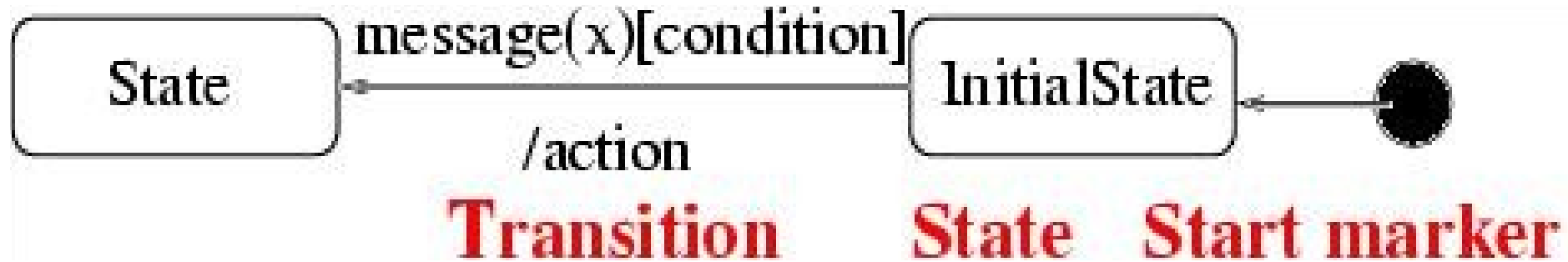
UML Run-through: Class diagram



Class structure of system.

Classes with attributes and operations/signals;
relationships between classes.

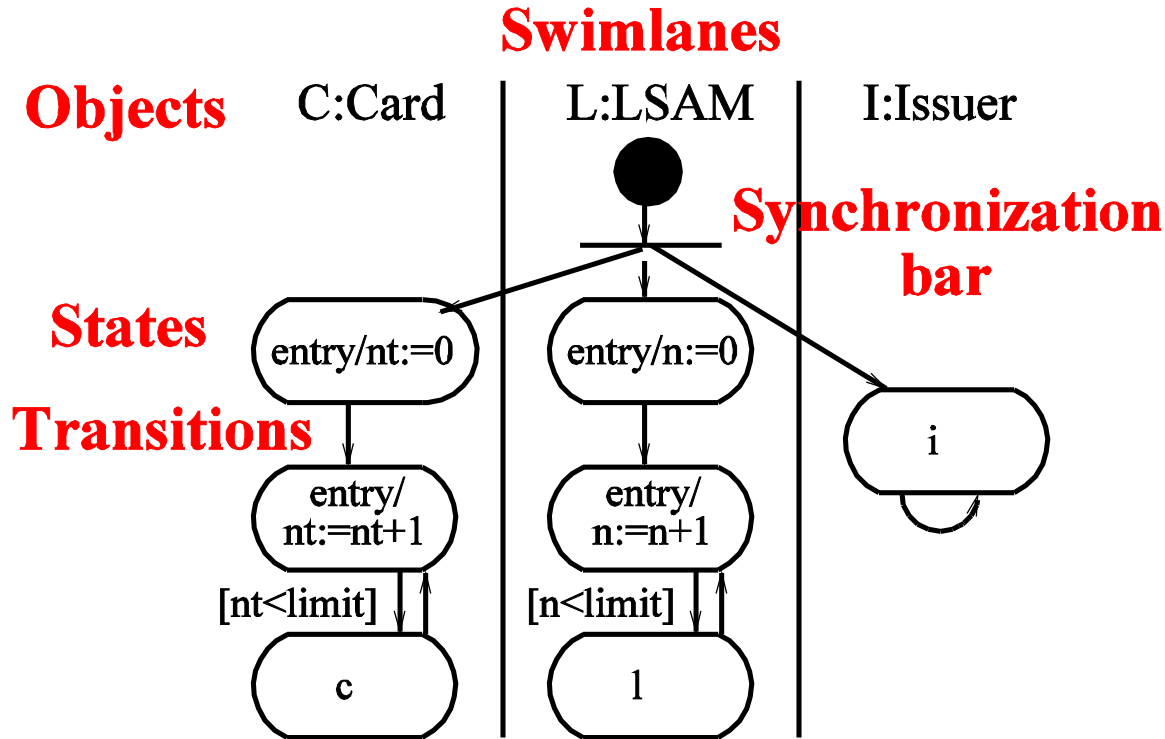
UML Run-through: Statecharts



Dynamic behaviour of individual component.

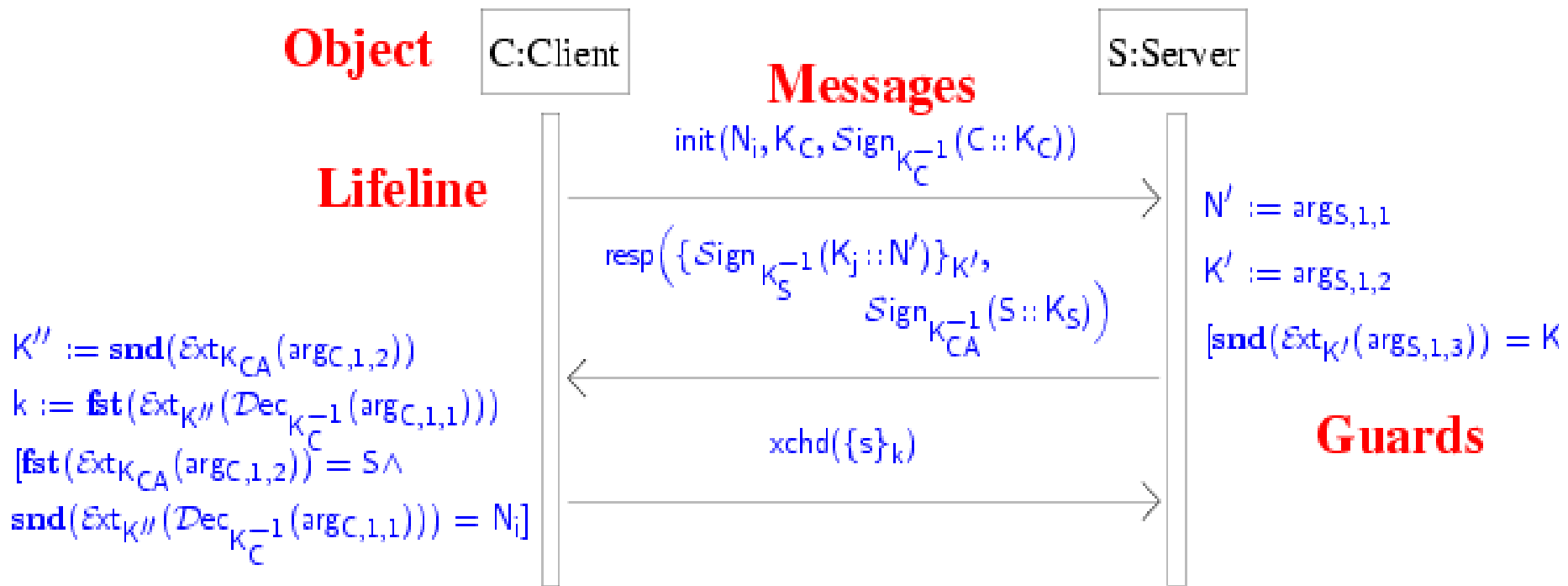
Input events cause state change and output actions.

UML Run-through: Activity Diagram



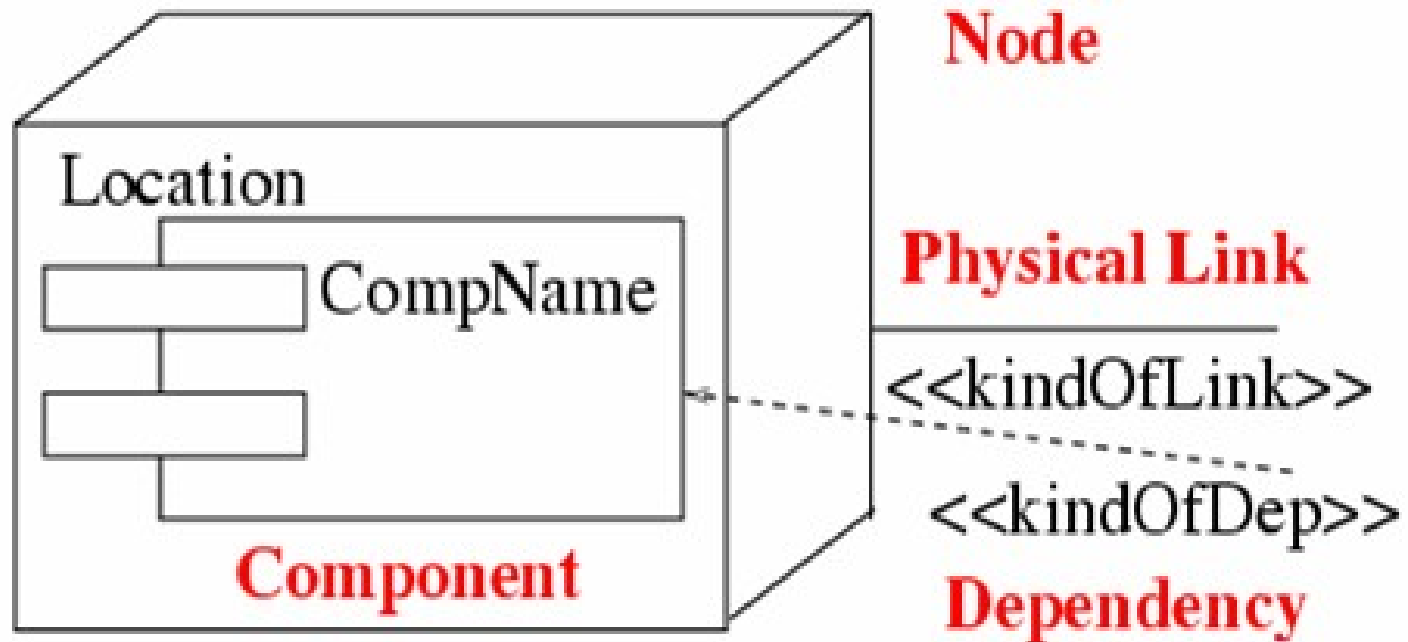
Specify the **control flow** between components within the system, at higher degree of abstraction than statecharts and sequence diagrams.

UML Run-through: Sequence Diagram



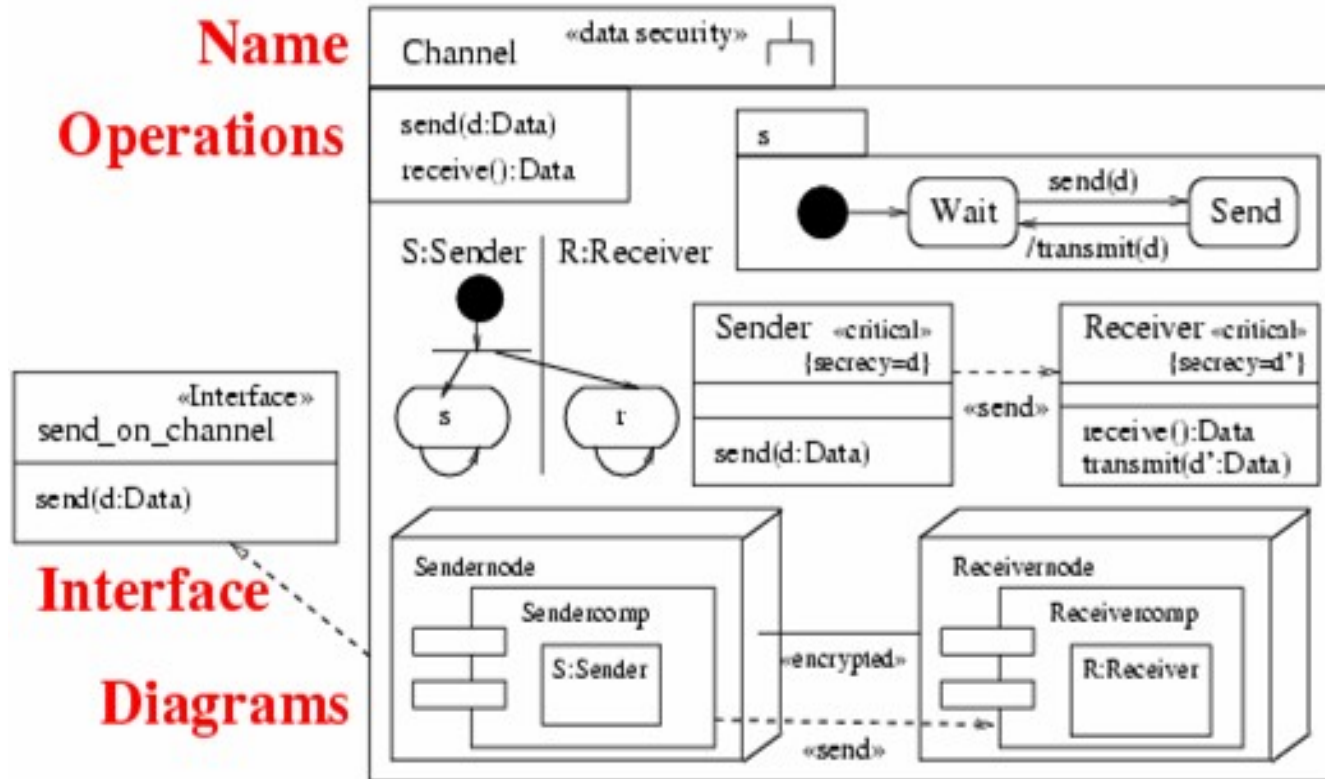
Describe **interaction** between objects or components via **message exchange**.

UML Run-through: Deployment Diagram



Describe the **physical layer** on which the system is to be implemented.

UML Run-through: Package



May be used to organize model elements into groups.

UML Extension Mechanisms

Stereotype: **specialize** model element using `<<label>>`.

Tagged value: **attach** `{tag=value}` pair to stereotyped element.

Constraint: **refine** semantics of stereotyped element.

Profile: **gather** above information.

Requirements on UML Extension for Security I

Provide basic **security requirements** such as secrecy, integrity, authenticity.

Allow considering different **threat scenarios** depending on adversary strengths.

Allow including important **security concepts** (e.g. *tamper-resistant hardware*).

Allow incorporating **security mechanisms** (e.g. access control).

Requirements on UML Extension for Security II

Provide **security primitives** (e.g. (a)symmetric encryption).

Allow considering underlying **physical security**.

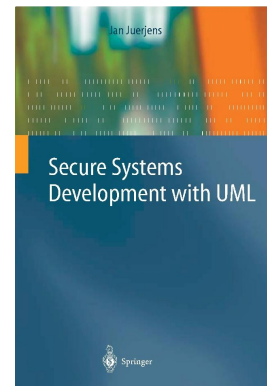
Allow addressing **security management** (e.g. secure workflow).

Also: Include **domain-specific** security knowledge (Java, smart cards, CORBA, ...).

Model-based Security with UMLsec

Extension of the Unified Modeling Language (UML) for **secure systems** development.

- evaluate UML models for security
- encapsulate **established rules** of prudent secure engineering
- make available to developers **not specialized** in secure systems
- consider security requirements from **early** design phases, in system **context**
- can use in certification

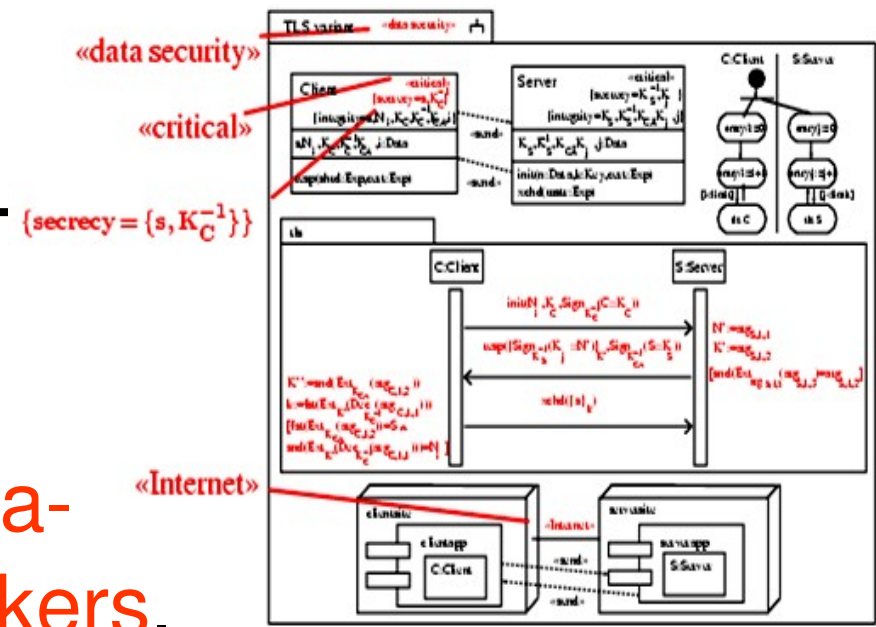


UMLsec

Insert recurring security requirements, adversary scenarios, security mechanisms as predefined markers.

Use associated logical constraints to verify specifications using model checkers and ATPs based on formal semantics.

Ensures that UML specification enforces the relevant security requirements wrt Dolev-Yao type adversaries. [\[FASE01,UML02,FOSAD05,ICSE05\]](#)



What Does UMLsec Cover ?

Security requirements: <<secrecy>>,...

Threat scenarios: Use `Threatsadv(ster)`.

Security concepts: For example <<smart card>>.

Security mechanisms: E.g. <<guarded access>>.

Security primitives: Encryption built in.

Physical security: Given in deployment diagrams.

Security management: Use activity diagrams.

Technology specific: Java, CORBA security.

UMLsec: General Ideas

Activity diagram: secure control flow, coordination

Class diagram: exchange of data
preserves security levels

Sequence diagram: security-critical interaction

Statechart diagram: security preserved
within object

Deployment diagram: physical security
requirements

Package: holistic view on security

UMLsec Profile (excerpt)

| Stereotype | Base class | Tags | Constraints | Description |
|----------------------|------------|----------------|---|---|
| Internet | link | | | Internet connection |
| secure links | subsystem | | dependency security matched by links | enforces secure communication links |
| secrecy | dependency | | | assumes secrecy |
| secure dependency | subsystem | | call, send respect data security | structural interaction data security |
| no down-flow | subsystem | high | prevents down-flow | information flow |
| data security | subsystem | | provides secrecy, integrity | basic datasec requirements |
| fair exchange | package | start, stop | after start eventually reach stop | enforce fair exchange |
| guarded access | Subsystem | | guarded objects acc. through guards. | access control using guard objects |

UMLsec as Integrating Formal Framework

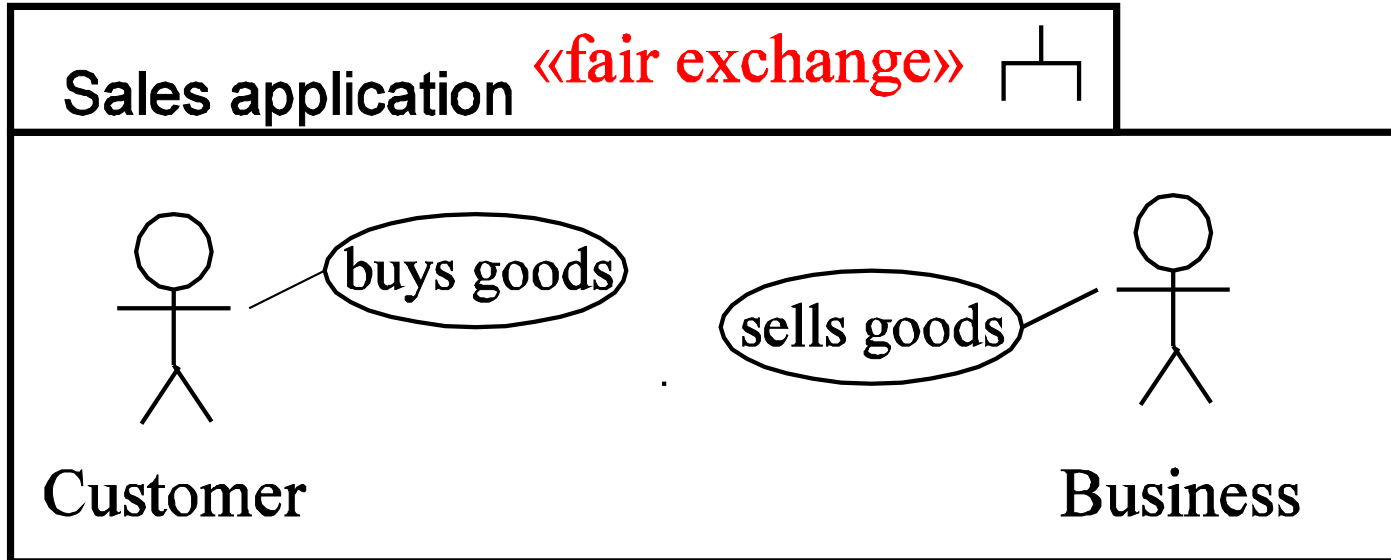
Have formalizations of major security requirements in one integrated notation.

Want to **relate** / **combine** requirements; get **modularity** / **composability**, hierarchical **decomposition**, **refinement**, ... :

For example:

- If system satisfies **<<secure links>>** and subsystems satisfy **<<data security>>** then system satisfies **<<data security>>**.

Requirements with Use Case Diagrams



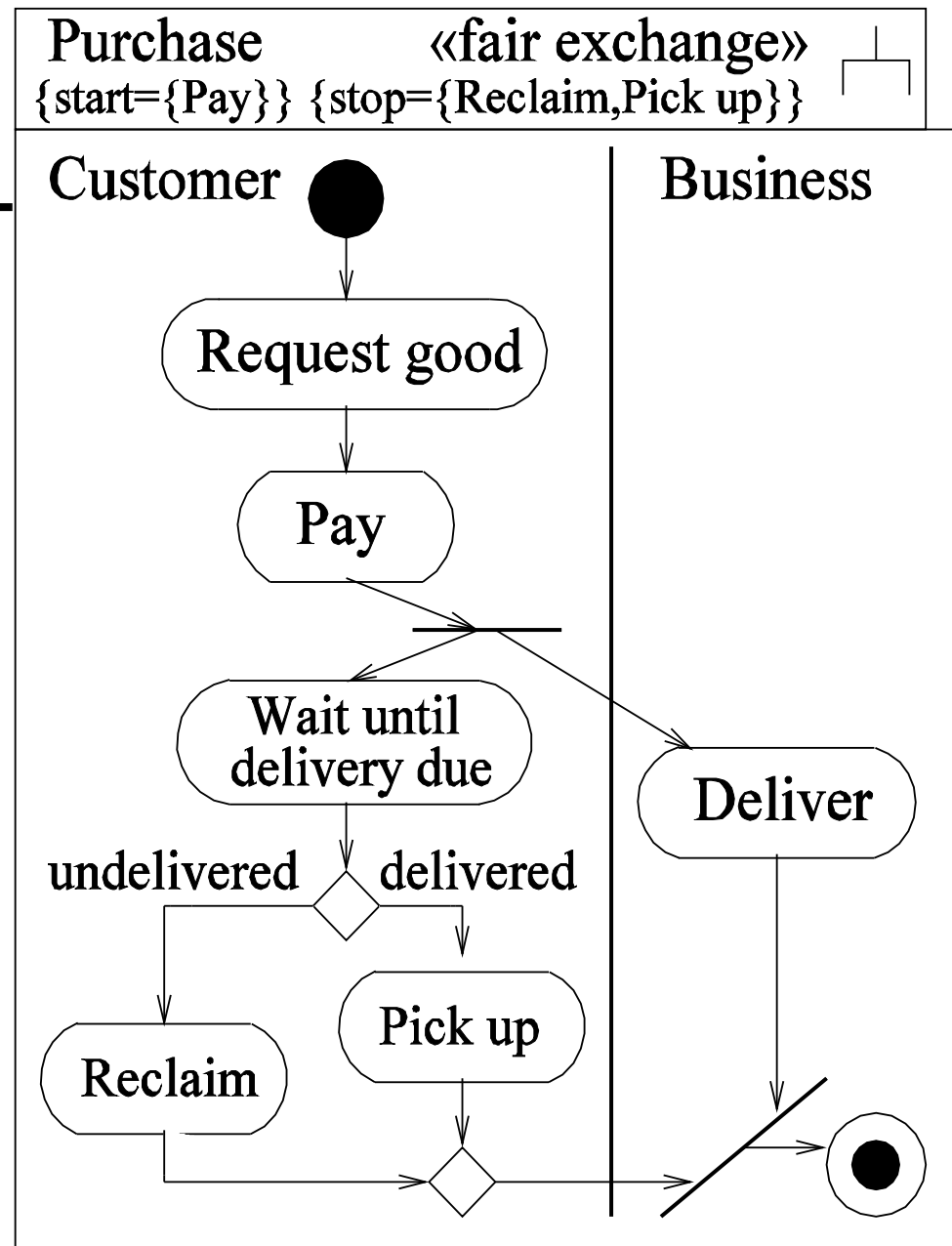
Capture security requirements
in use case diagrams.

Constraint: need to appear in corresponding
activity diagram.

Fair Exchange

Customer buys good from a business.

How can enforce fair exchange:
After payment, customer is eventually either **delivered** good or able to **reclaim** payment (or vc.vs.).



<<fair exchange>>

Ensures generic **fair exchange** condition.

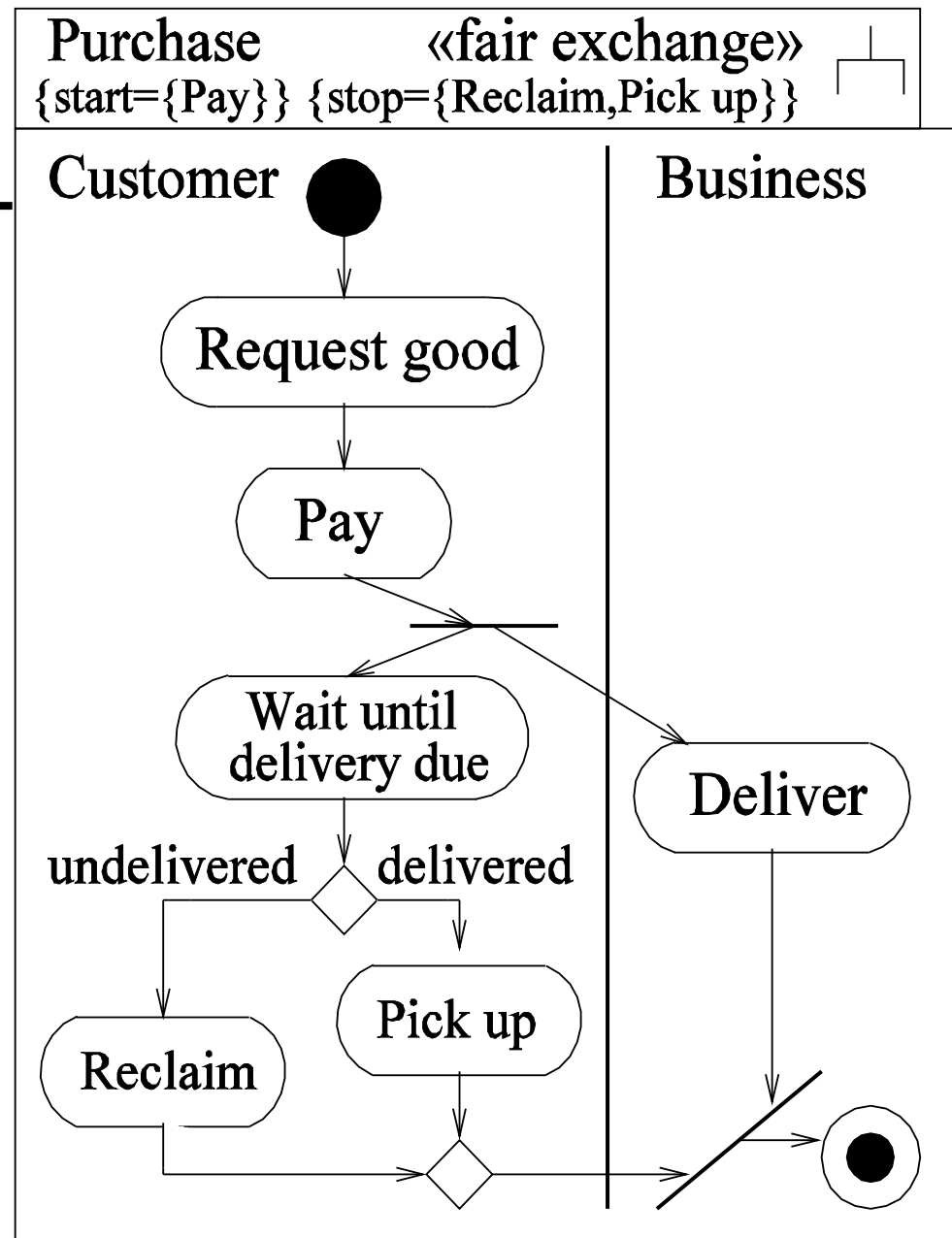
Constraint: after a **{start}** state in activity diagram is reached, eventually reach **{stop}** state.

(Cannot be ensured for systems that an attacker can stop completely.)

Example

<<fair exchange>>

fulfilled if adversary cannot stop system:
After payment, customer is eventually either **delivered** good or able to **reclaim** payment.



<<Internet>>, <<encrypted>>, ...

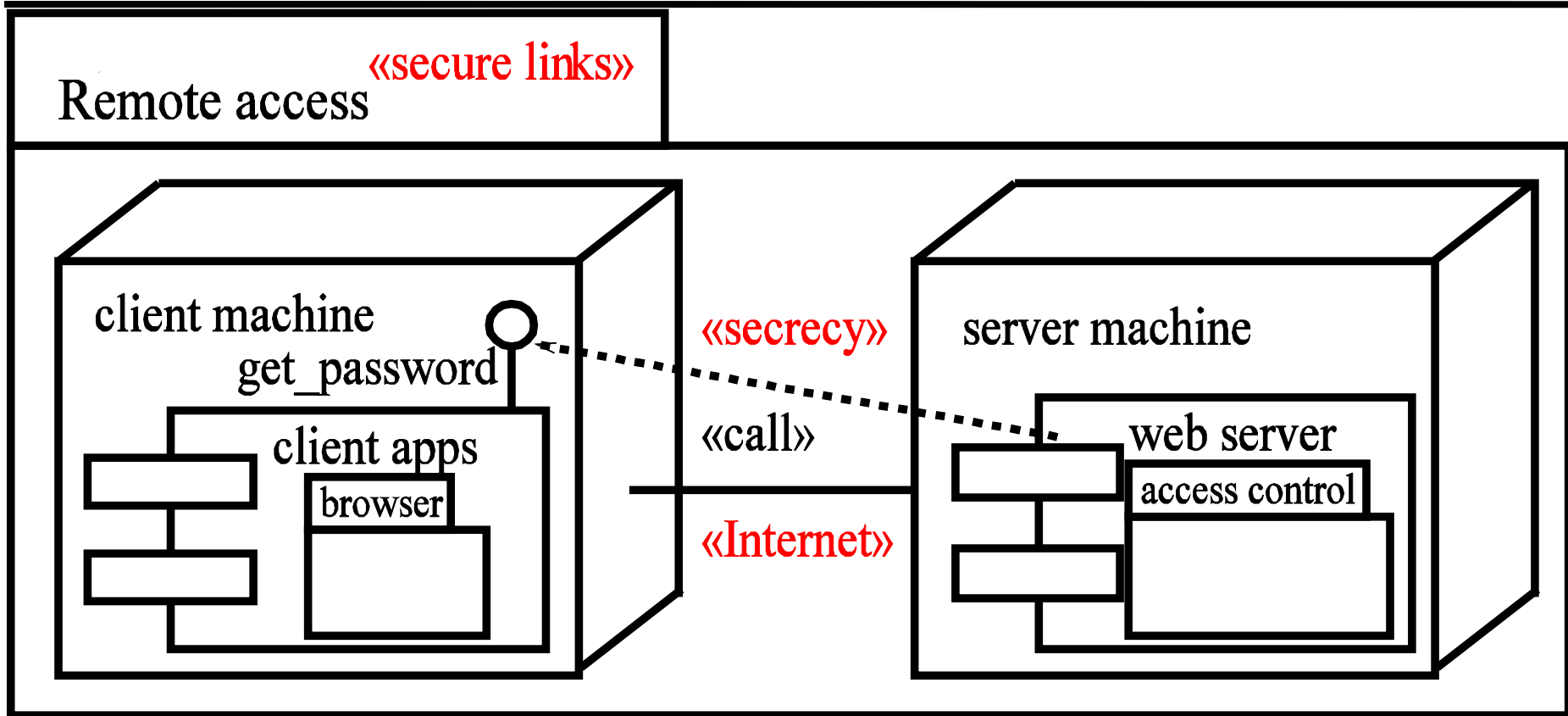
Kinds of communication **links** resp. system **nodes**.

For adversary type A , stereotype s , have set $\text{Threats}_A(s) \in \{\text{delete, read, insert, access}\}$ of actions that adversaries are capable of.

Default attacker:

| Stereotype | $\text{Threats}_{\text{default}}()$ |
|------------|-------------------------------------|
| Internet | {delete, read, insert} |
| encrypted | {delete} |
| LAN | \emptyset |
| smart card | \emptyset |

Secure Architecture



Architecture secure against **default** adversary ?

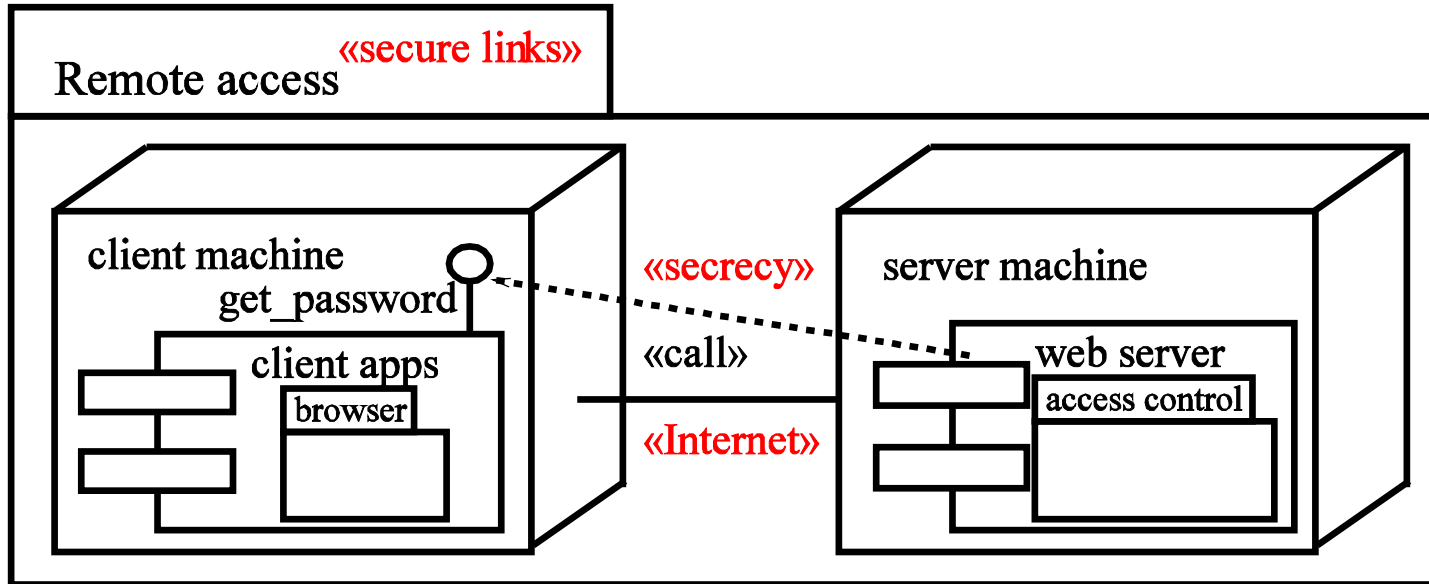
<<secure links>>

Ensures that physical layer meets security requirements on communication against a given attacker of type A .

Constraint: for each dependency d with stereotype $s \in \{\llbracket \text{secrecy} \rrbracket, \llbracket \text{integrity} \rrbracket\}$ between components on nodes $n \neq m$, the communication is over a communication link l between n and m with stereotype t such that

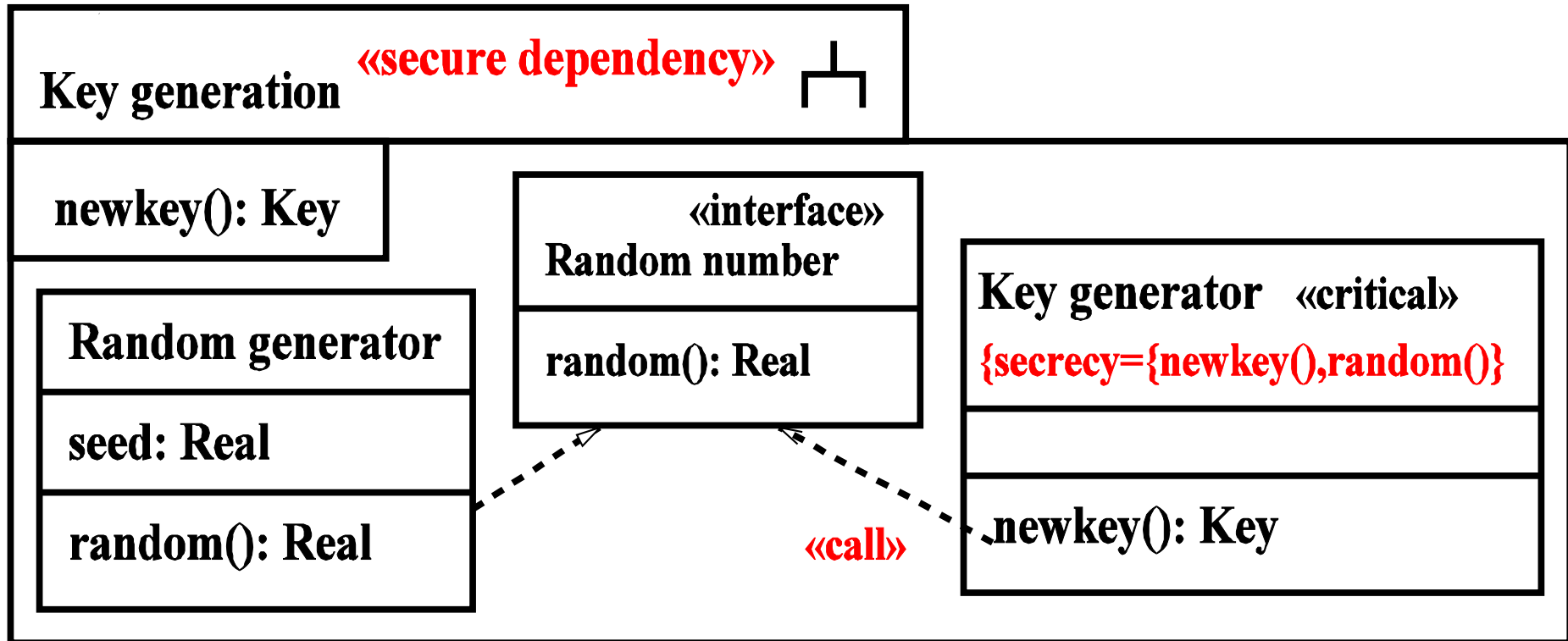
- if $s = \llbracket \text{secrecy} \rrbracket$: have $\text{read} \notin \text{Threats}_A(t)$.
- if $s = \llbracket \text{integrity} \rrbracket$: have $\text{insert} \notin \text{Threats}_A(t)$.

Example <<secure links>>



Given **default** adversary type, constraint for stereotype <<secure links>> **violated**:
According to the **Threats_{default}(Internet)** scenario, <<Internet>> link does not provide secrecy against **default** adversary.

Secure Data Structure



Data structure secure ?

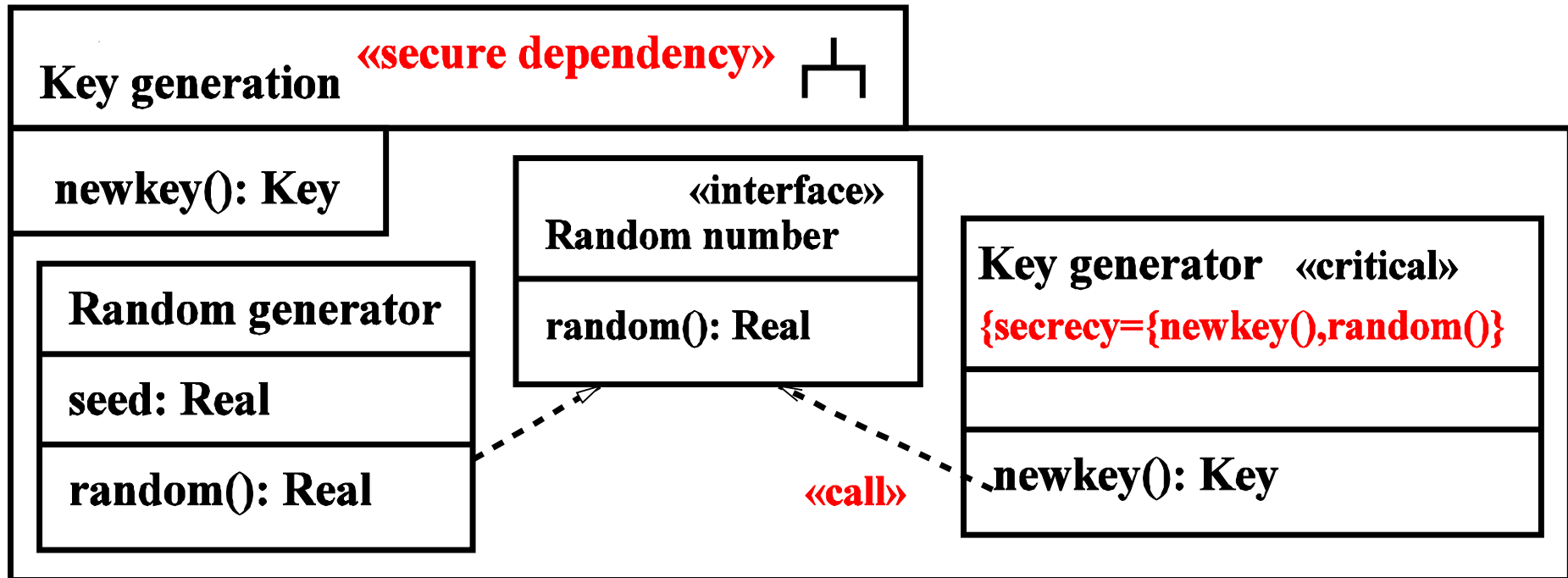
<<secure dependency>>

Ensure that <<call>> and <<send>> dependencies between components **respect** security requirements on communicated data given by tags {**secrecy**}, {**integrity**}.

Constraint: for <<call>> or <<send>> dependency from *C* to *D* (and similarly for {**integrity**}):

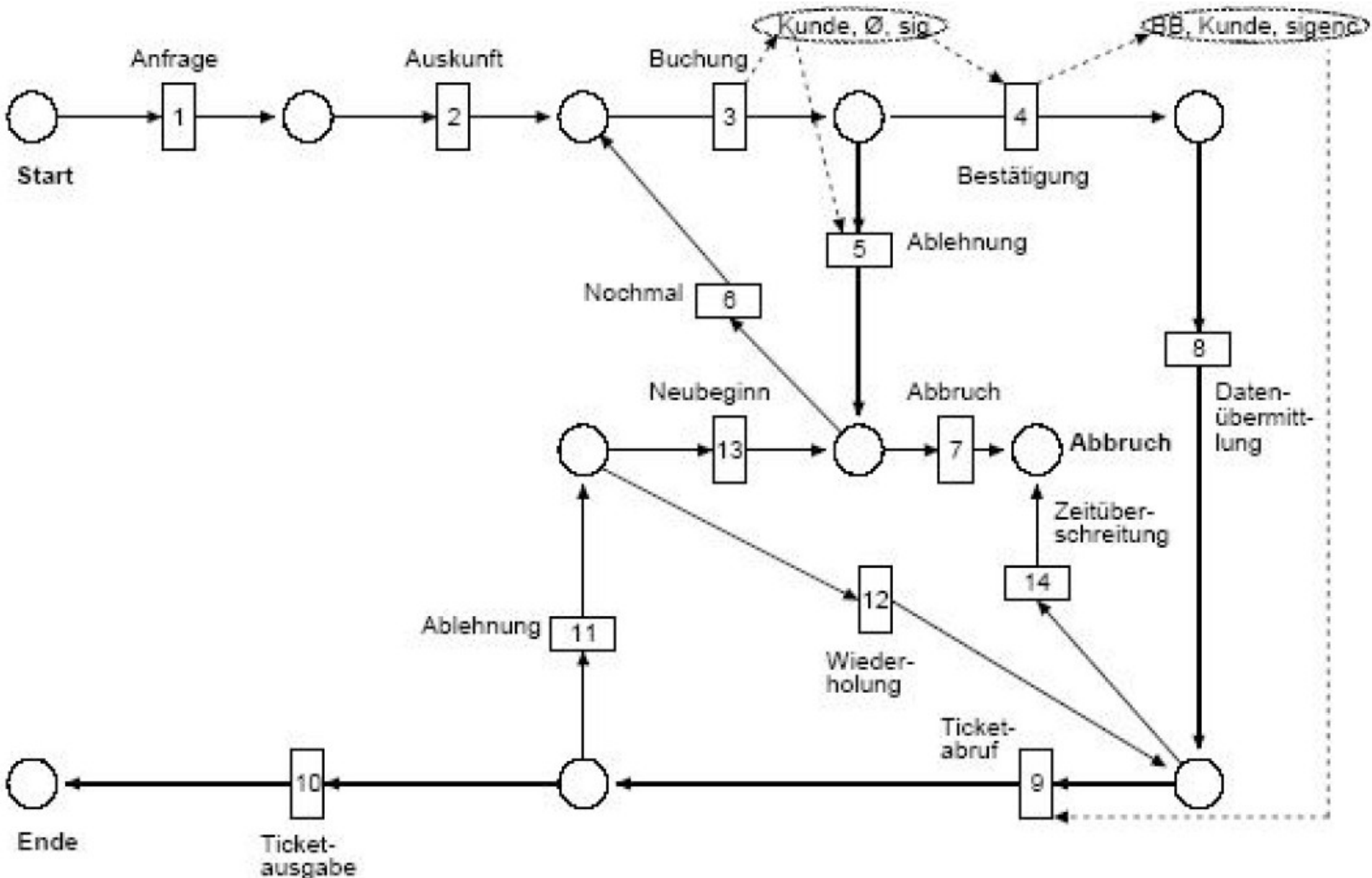
- Msg in *D* is {**secrecy**} in *C* if and only if also in *D*.
- If msg in *D* is {**secrecy**} in *C*, dependency stereotyped <<secrecy>>.

Example <<secure dependency>>



Violates <<secure dependency>>: Random generator and <<call>> dependency do not give security level for random() to key generator.

Aufgabe 7 Elektronischer Ticketverkauf



Aufgabe 7

- a) Stelle den Anwendungsfall
``Kartenreservierung`` in einem Use-case-
Diagramm mit Akteuren Kunde, Kinosever
und Abendkasse dar. [2 P.]
- b) Füge die relevanten
Sicherheitsanforderungen als Stereotypen
(ggf. selbstdefiniert) ein. [3 P.]

Aufgabe 7

- c) Spezifiziere den zugehörigen vereinfachten Geschäftsprozess als Aktivitätsdiagramm. [5 P.] (Tipp: eine mögliche Lösung enthält z.B. die Aktivitäten „Anfrage starten“, „Plätze anzeigen“, „Plätze auswählen“, „Bestätigung abrufen“, „Auftrag weiterleiten“, „Auftrag speichern“, „Ticket abrufen“, „Ticket ausgeben“.)
- d) Füge die start und stop Tags für den Stereotype Fair exchange ein. [2 P.]

Aufgabe 7

- e) Entwerfe eine vereinfachte Architektur des Systems mit den Komponenten Kinosever und Abendkasse auf dem Knoten Kino und der Komponente Browser auf dem Knoten Kunden-PC in einem UML Deploymentdiagramm. [2 P.]
- f) Füge die relevanten Sicherheitsanforderungen als Stereotypen (ggf. selbstdefiniert) ein. [3 P.]

Aufgabe 7

- g) Spezifiziere die vereinfachte Daten-struktur als Klassendiagramm mit 3 – 7 Klassen.
[4 P.]
- h) Füge die relevanten Sicherheitsanforderungen als Stereotypen (ggf. selbstdefiniert) ein. [3 P.]