Modellbasierte Entwicklung von Technologien für selbstorganisierende dezentrale Informations-systeme im Katastrophen-management

METRIK

Projekt Erdbebenfrühwarnung
im SoSe 2011

HUMBOLDT-UNIVERSITÄT ZU BERLIN

Entwicklung verteilter echtzeitfähiger Sensorsysteme

Joachim Fischer
Klaus Ahrens
Ingmar Eveslage

fischer|ahrens|eveslage@informatik.hu-berlin.de

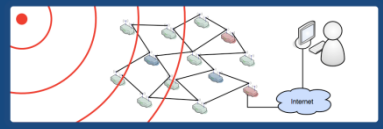
SAFER
Seismic eArly warning For EuRope

EDIM

SOSEWIN-extented

GFZ
Helmholtz-Zentrum
POTSDAM

Systemanalyse



SDL als UML-Ersatz

1. UML und SDL-Zustandsmaschinen im Vergleich
2. ITU-Standard Z.100
3. Werkzeuge
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL

Überblick

SDL-96
SDL-2000
UML-2.0

- TAU-Werkzeug, IBM Rational
  (ursprünglich schwedische Firma TeleLogic)
- Cinderella (Standard-SDL, **SDL 96**, dänische Firma)
  mit Codegenerator der HU Berlin
- SDL-2000 ASM-Interpreter
  (MicroSoft, HU Berlin, Uni Kaiserslautern)
- PragmaDev Developer Studio (pragmatische SDL-Variante)
  - Eingeschränktes SDL
  - C/C++ Actionsprache
  - Pragmatische Kombination mit UML: KlassenDG, UseCaseDG, SequenceDG, DeploymentDG

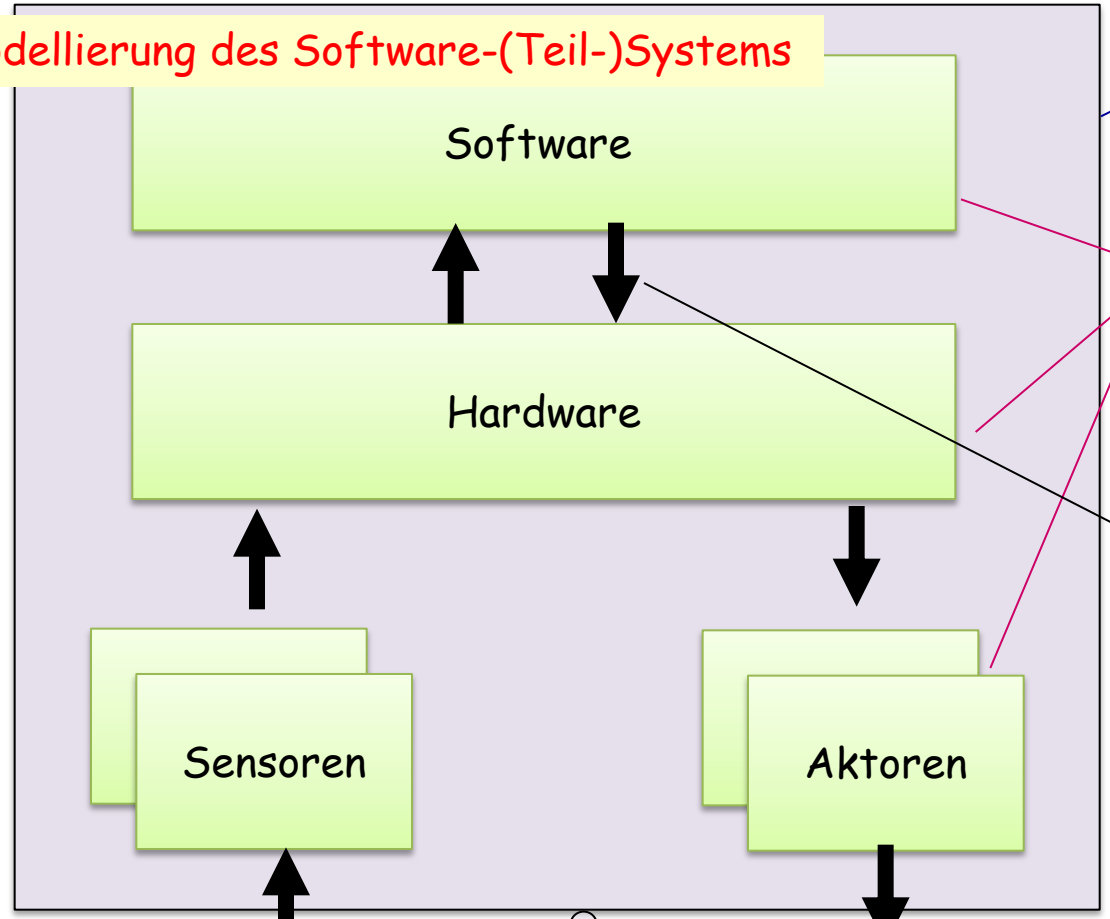
  mit modifiziertem Code-Generator der HU Berlin
- SDL-Werkzeug auf Metamodell-Basis nur rudimentär
  (HU Berlin)

SDL als UML-Ersatz

1. UML und SDL-Zustandsmaschinen im Vergleich
2. ITU-Standard Z.100
3. Werkzeuge
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL

Eingebettete u. Reaktive Systeme und SDL

SDL/UML zur Modellierung des Software-(Teil-)Systems



Systemgrenze

Struktur

Verhalten (in Abh. der Zeit)

weiter zerlegbare System-komponenten

Komponenten-relationen

nichtautonom (autonom)

SysML zur Modellierung des Gesamtsystems

Referenzierung und Lebenszeit von Agenten

- Ausgezeichneter Datentyp PId (*process identification*)
  - Werte des vordefinierten Datentyps PId stellen systemweit **eindeutige Referenzen** für Prozessinstanzen dar
  - Referenzen können zur **Nachrichtenadressierung** genutzt werden (alternative oder zusätzliche Angaben zu Kommunikationspfaden)
  - Wissen über Existenz von Prozessen muss per Kommunikation oder Prozessgenerierung erworben werden

- Lebenslauf eines Prozesses
  - **Start** –(Pseudo-)Zustand, Einnahme (ohne zu verharren) per
    - Systemexistenz als initialer Prozess einer Prozessmenge
    - Erzeugung durch eine *process*-Instanz einer **anderen** Instanzmenge desselben Blockes oder der gleichen Instanzmenge während der Systemexistenzdauer
  - **Stop** , Einnahme (ohne zu verharren) führt zum Existenzende

Achtung: Laufzeitfehler, falls eine Nachricht an eine nicht mehr existente *process*-Instanz gesendet wird

- Systemexistenz
  - endet, sobald die letzte *process*-Instanz des Systems stoppt

Systemsichtweisen von SDL-96

stabile SDL-Version

- **Strukturelle Sichtweise**
  - Instanzsicht

    Beschreibung der Konfiguration eines Systems (system) bestehend aus funktionalen Einheiten (block) und ihren Relationen (channel) bei Identifikation der Systemgrenzen

    Unterschiede: Block und Process
    (Verallgemeinerung als Agent in SDL-2000)

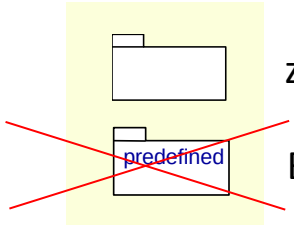
    ab SDL-2000: Block ist statisch oder dynamisch (zuvor immer nur statisch)

  - Typsicht (Klasse)
  - Pakete (package) zur bequemen Wiederverwendung von Typen (Klassen)
    vordefiniertes Paket für Datentypen

- **Verhaltensorientierte Sichtweise**
  - Verhalten einer funktionalen Einheit wird durch kommunizierende nebenläufig agierende Zustandsautomaten erbracht (process)
  - Prozessverhalten: zeitdiskret/ ereignisorientiert
  - Strukturierungsmöglichkeiten von Verhaltensbeschreibungskonstrukten (procedure, service)

Weitere Referenzsymbole

- Paket (Package)

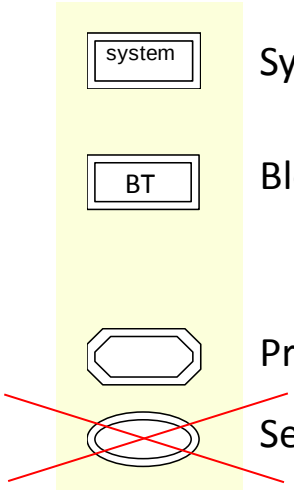


zur Zusammenfassung und Nutzung wiederverwendbarer Komponenten

Beispiel einer Paketreferenz: vordefinierte Datentypen

nicht in SDL-RT

- Strukturtypen (Vorlagen für Instanz- bzw. Instanzmengendefinition)



Systemtyp

nicht in SDL-RT

Blocktyp

b1: BT	b1 als typbasierte Blockinstanz
b2(2): BT	b2 als typbasierte Blockinstanzmenge
b3	b3: als Blockinstanz mit implizitem Typ
b4(12)	b4: als Blockinstanzmenge mit implizitem Typ

Prozesstyp

Servicetyp

nicht in SDL-RT

Inhalt folgt in SDL/RT der C-Syntax

Achtung: Elementare Datentypen haben in SDL **keine** Referenzsymbole werden als Text in universellen Textboxen erfasst

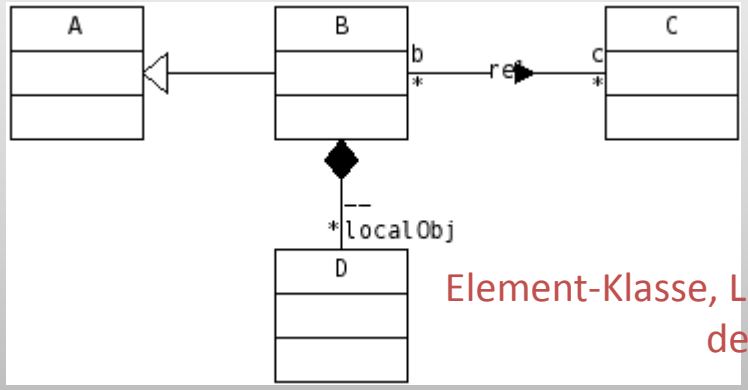
In SDL-RT haben passive (UML-)Klassen ein Referenzsymbol

```
dcl x int=0;
```

Weitere SDL-Referenzsymbole

Container-Klasse

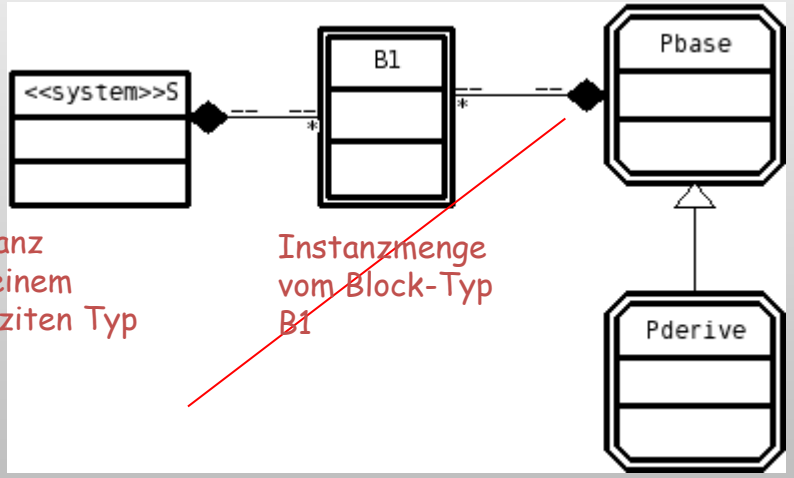
nur in SDL-RT



Klassendiagramm
(passive Klassen)

Element-Klasse, Lebenslauf der Elementobjekte ist an Existenz des Objektes der Container-Klasse gebunden

System Blocktyp Processtyp



Instanzmenge vom Proces-Typ Pbase

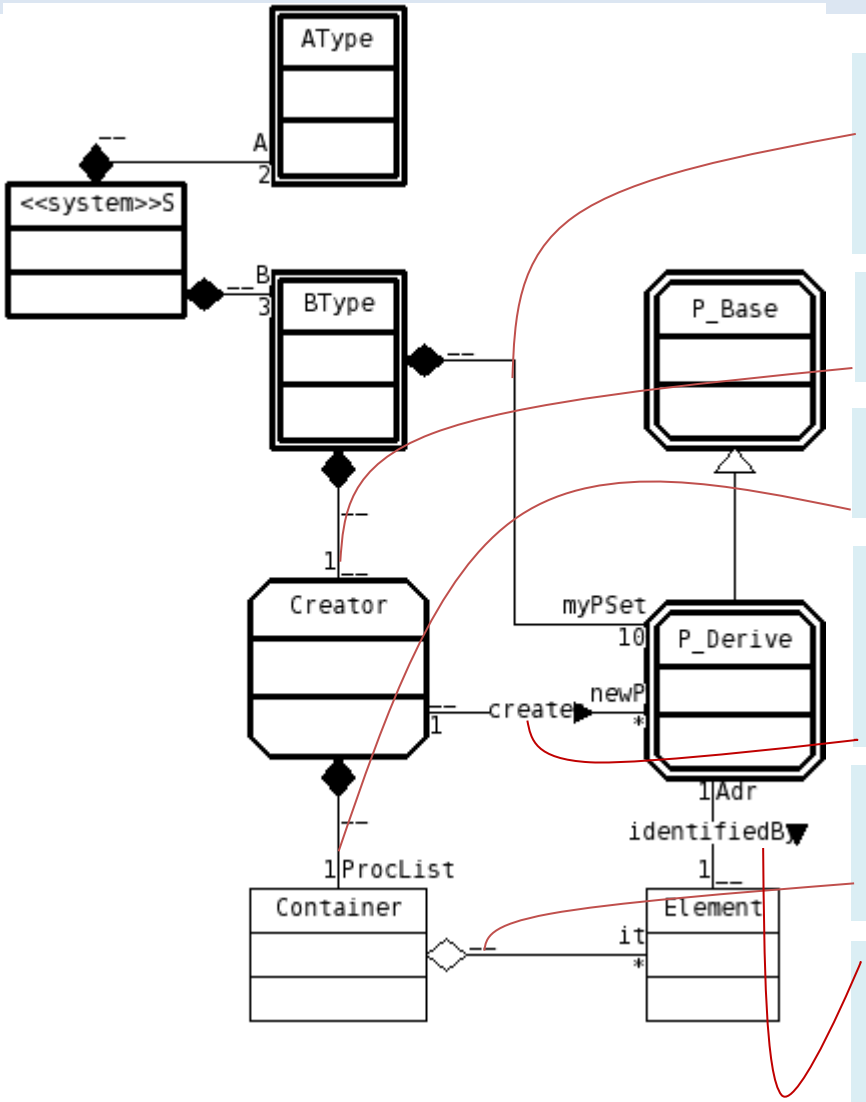
Klassendiagramm
(aktive Klassen)

Instanz von einem impliziten Typ

Instanzmenge vom Block-Typ B1

ACHTUNG:
unzulässige Kompositionsbeziehung aus statischer SDL-Semantiksicht

Anwendungsbeispiel



Komposition:
Instanz von P_Base (oder Ableitung)
gehört zu einer Instanz von BType
(zu deren lokalen Prozessmenge myPSet)

Komposition:
BType-Instanz besitzt Creator (als Process-Unikat)

Komposition:
Creator besitzt Container-Objekt ProcList (als Unikat)

Assoziation: creates
Creator erzeugt Instanzen von P_Derive
(beliebig viele, die in myPSet erfasst werden, deren Kardinalität auf 10 beschränkt ist)

Aggregation:
Container-Objekt verwaltet Element-Objekte (0,*)
erreicht die Elemente mit it (Iterator)

Assoziation: identifiedBy
über ProcList→it→Adr gelangt Creator an die Referenz (Adr) aller P_Derive-Instanzen von ProcList
über newP gelangt Creator an die Referenz des zuletzt generierten Prozesses vom Typ P_Derive

Prozessidentifikation

- jede Prozessmengen-Instanz hat eine systemweit-eindeutige **Identifikation** vom Typ PId (PId = process identification)

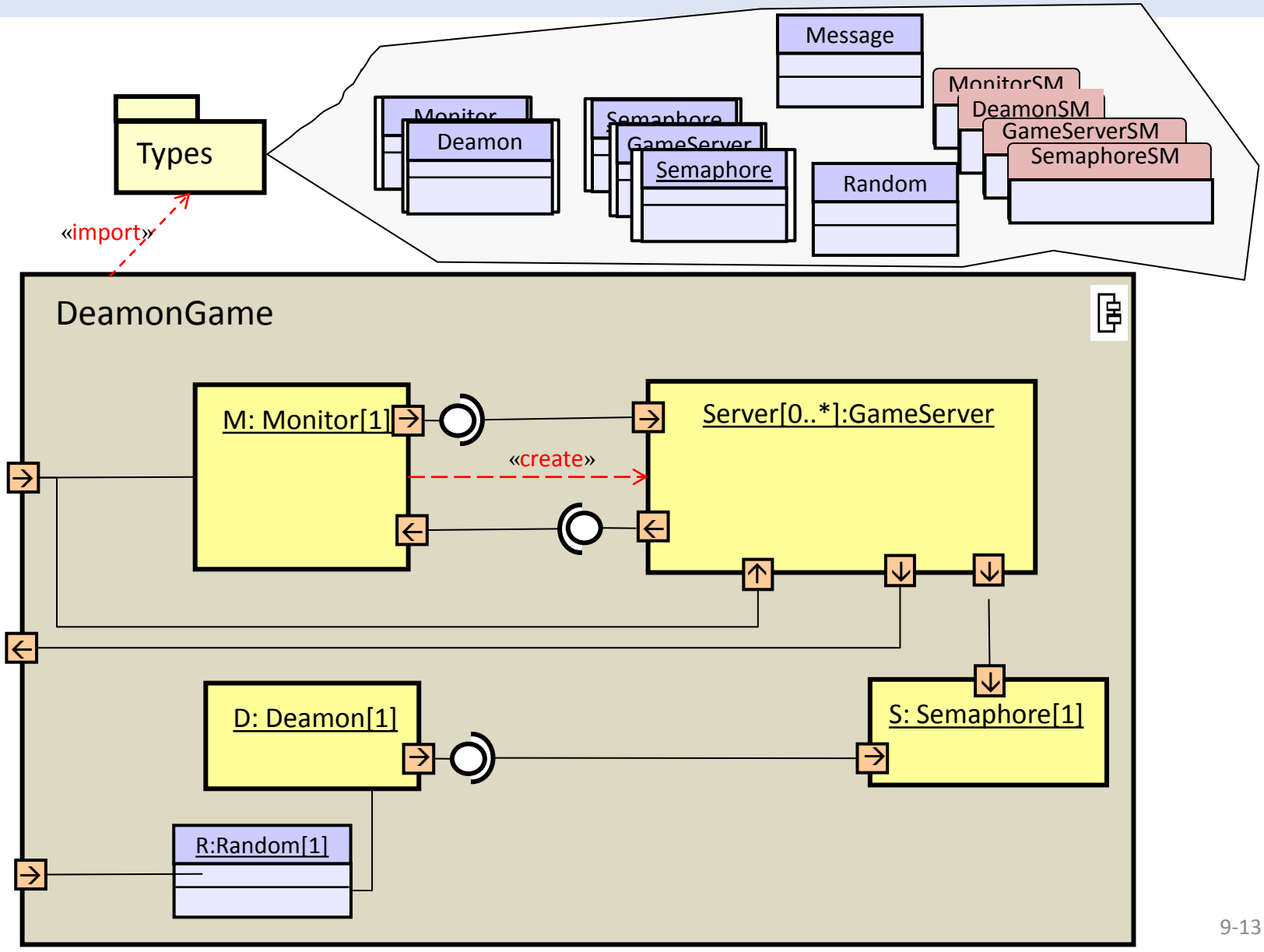
in SDL-RT: RTDS_QueueId

- Vergabe eines PId-Wertes erfolgt implizit per Instanzgenerierung
- keine Kompatibilität von PId zu anderen Typen
- PId-Werte können nur
  - verglichen werden (==, =/) und
  - in Variablen vom Typ PId zur Adressierung von Nachrichten gespeichert werden
- einziges Literal (explizite Notation für einen Wert): null
- eine Prozessmengen-Instanz existiert mit Systemstart oder wird zur Laufzeit (durch einen anderen Prozess) explizit generiert
- die Lebensdauer einer Prozessinstanz bestimmt nur die Instanz selbst

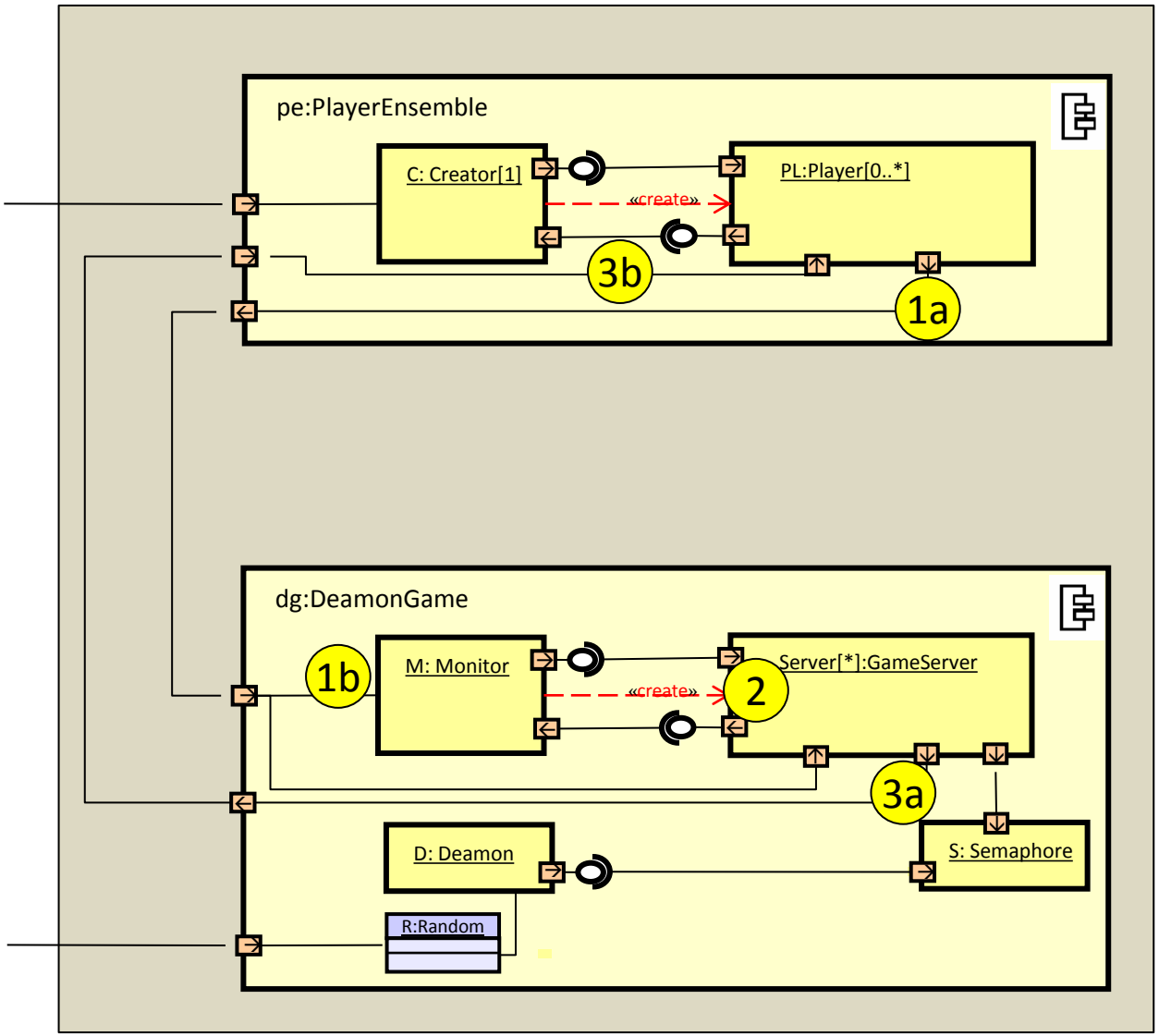
SDL als UML-Ersatz

1. UML und SDL-Zustandsmaschinen im Vergleich
2. ITU-Standard Z.100
3. Werkzeuge
4. SDL-Grundkonzepte
5. Musterbeispiel (in UML-Strukturen)
6. Struktur- und Verhaltensbeschreibung in SDL

UML-Systemtypdefinition: DeamonGame

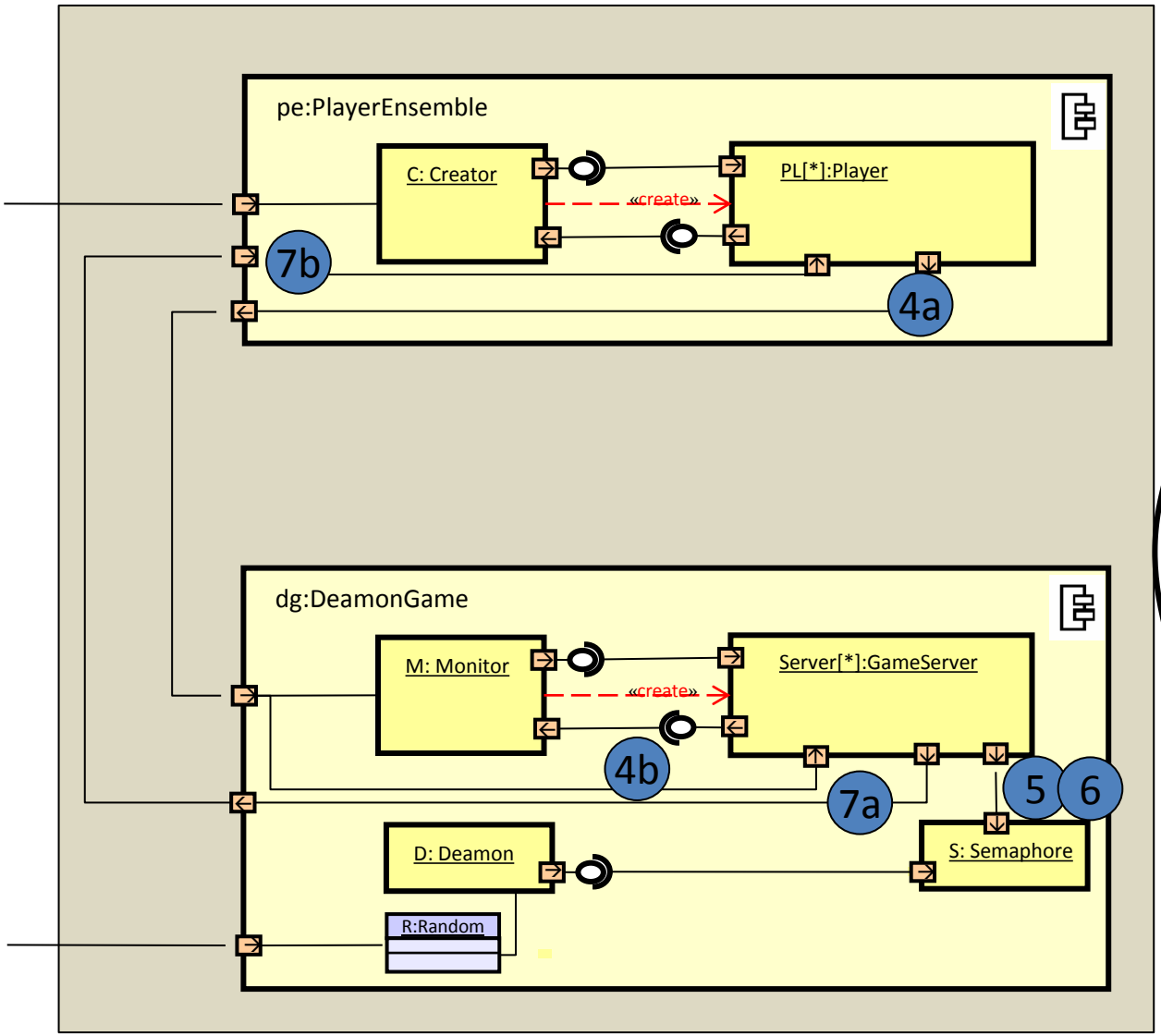


Erweitertes System (Nutzeranme



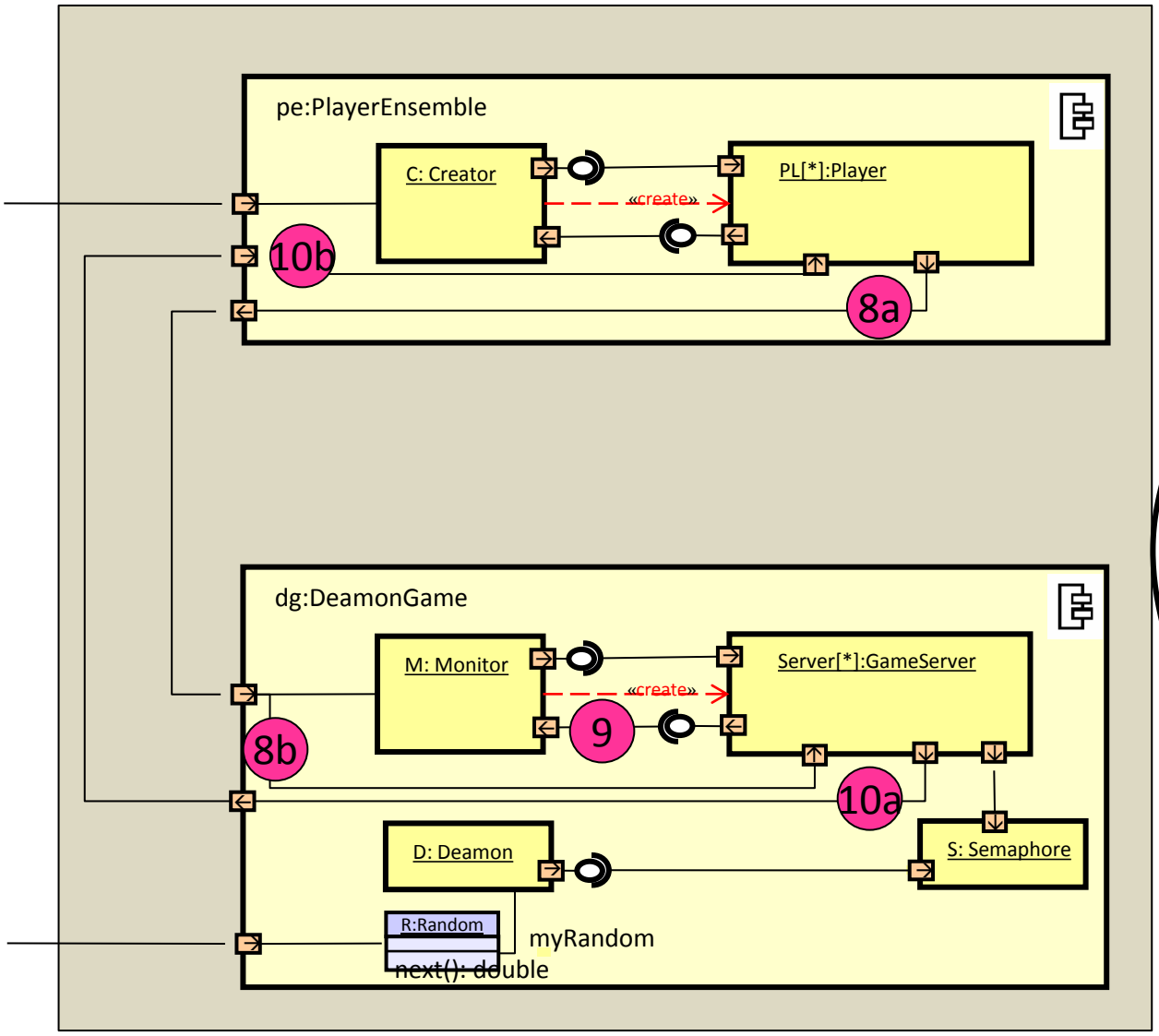
1. Anmeldung des Nutzers
2. Bereitstellung von Ressourcen zum Dienstzugang
3. Zugangsdaten an Nutzer

Erweitertes System (Dienstnutzu



1. Anmeldung des Nutzers
2. Bereitstellung von Ressourcen zum Dienstzugang
3. Zugangsdaten an Nutzer
4. Dienstaktivierung
5. Dienstrealisierung (Start)
6. Dienstrealisierung (Ende)
7. Dienstfinalisierung

Erweitertes System (Nutzerabmeld



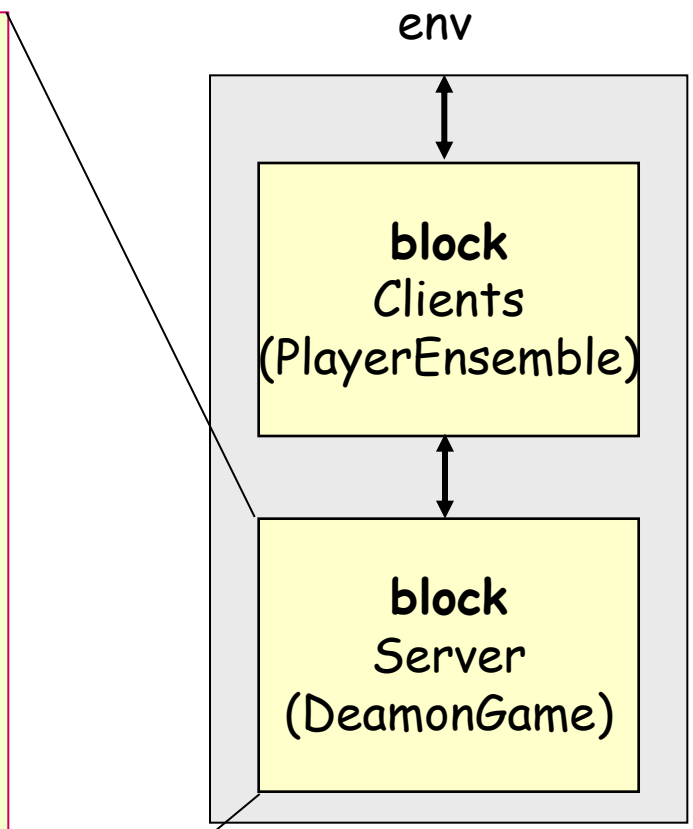
1. Anmeldung des Nutzers
2. Bereitstellung von Ressourcen zum Dienstzugang
3. Zugangsdaten an Nutzer
4. Dienstaktivierung
5. Dienstrealisierung (Start)
6. Dienstrealisierung (Ende)
7. Dienstfinalisierung
8. Abmeldung des Nutzers
9. Ressourcenfreigabe
10. Bestätigung der Abmeldung

Informale Festlegung des Dienstes (1)

Dienst: Computerspiel als reaktive Komponente

- unterstützt unbegrenzte (unbekannte) Anzahl von Spielern
- Spieler treten gegen den Computer an
  - nach Registrierung
  - bis zur Abmeldung
- eigentliche Spiel ist trivial
- Unterstützung
  - mehrerer,
  - von einander unabhängiger Spiele,

wobei ein Spieler zu einem Zeitpunkt nur bei höchstens einem Spiel angemeldet sein kann und mitwirken darf

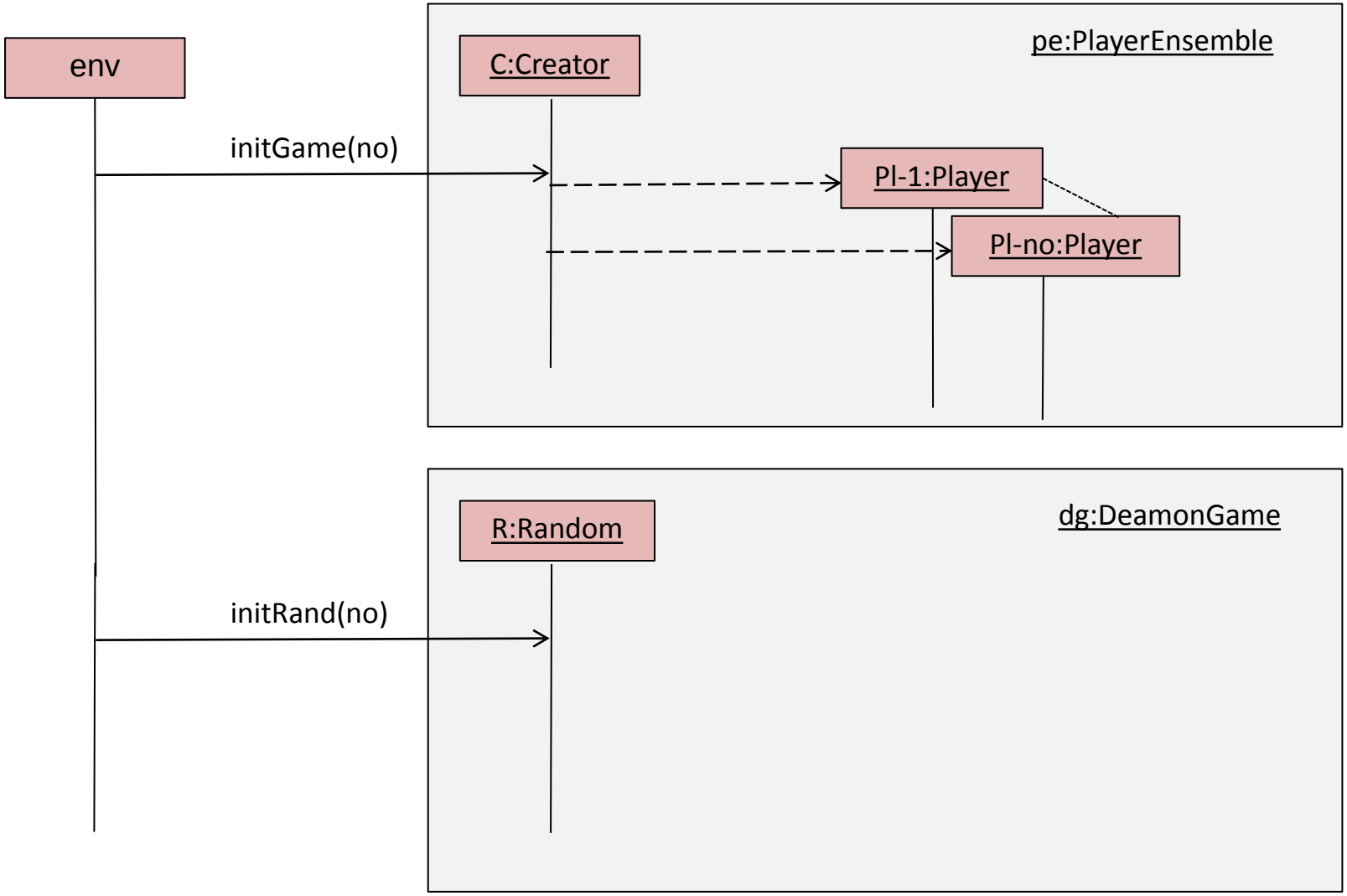


Informale Festlegung des Dienstes (2)

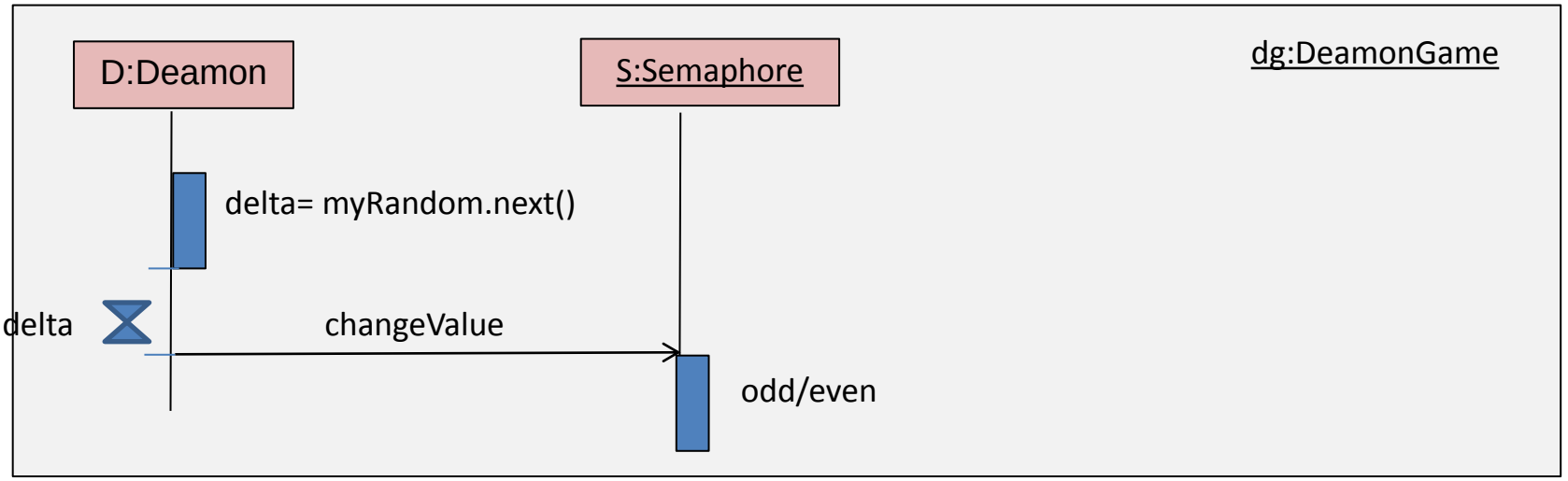
Spielregeln von DeamonGame

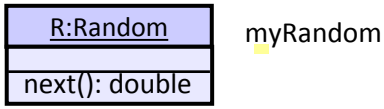
- der Wert **einer** nicht sichtbaren Variable ändert sich nichtdeterministisch: even ←→odd
- zu diskreten Zeitpunkten rät ein Spieler (als Client), ob der Wert ungerade (odd) ist
  - ist das der Fall, **gewinnt** er einen Punkt
  - wenn nicht, **verliert** er einen Punkt
- zu jedem Zeitpunkt kann von den Spielern ihr jeweiliger Punktestand abgefragt werden

Schnittstellenprotokolle: Spielinitialisierung

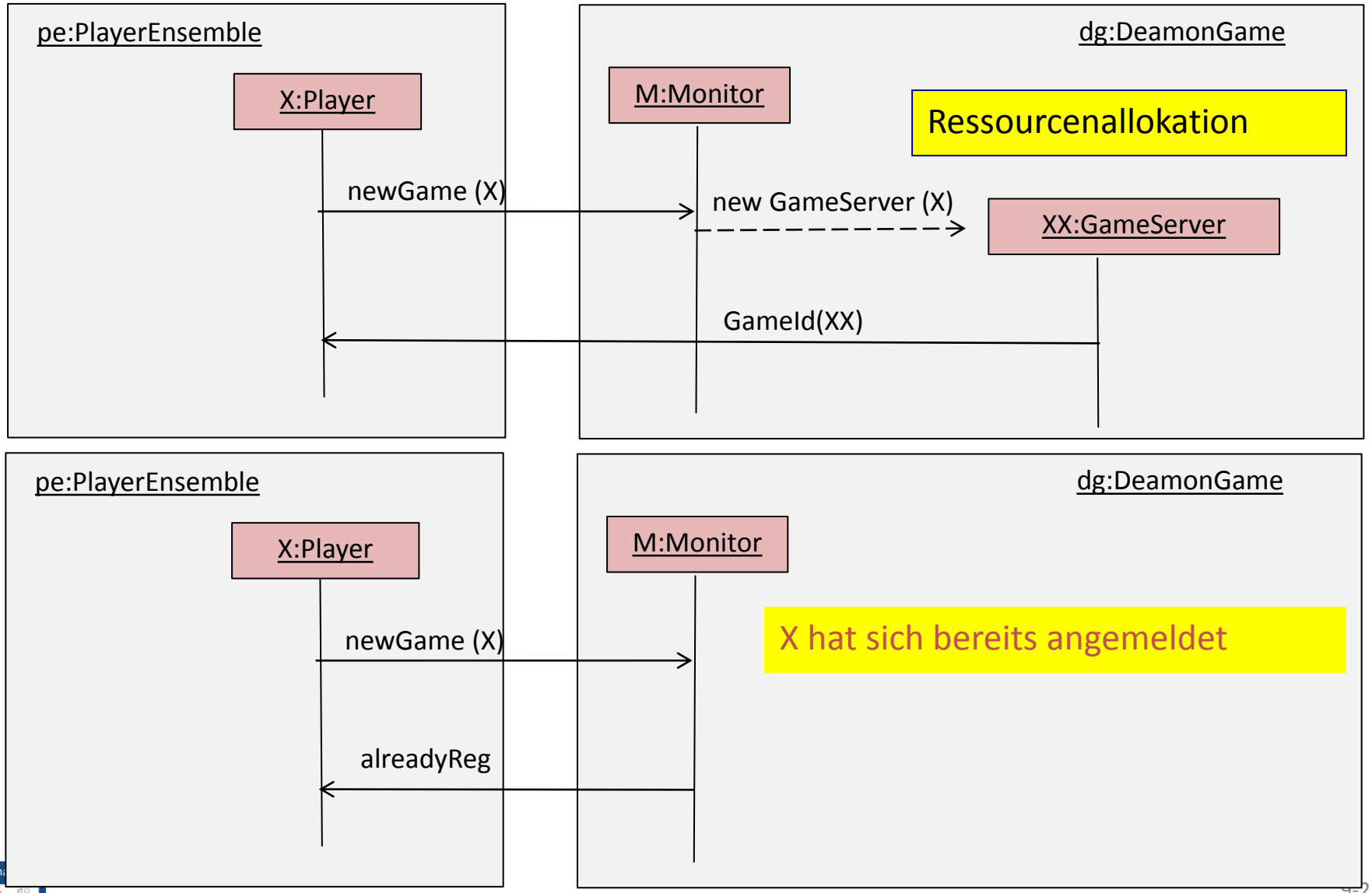


Schnittstellenprotokolle: Zufällige Variablenänderung

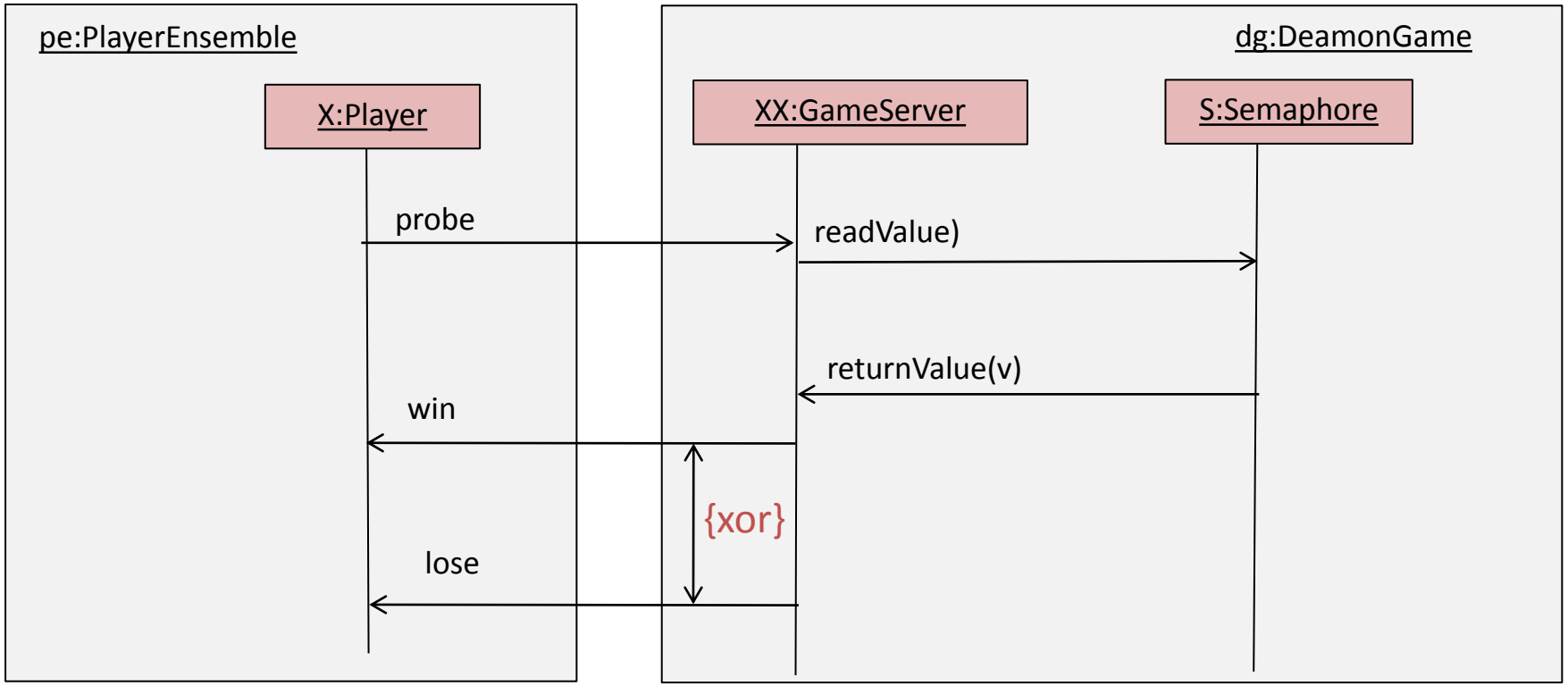




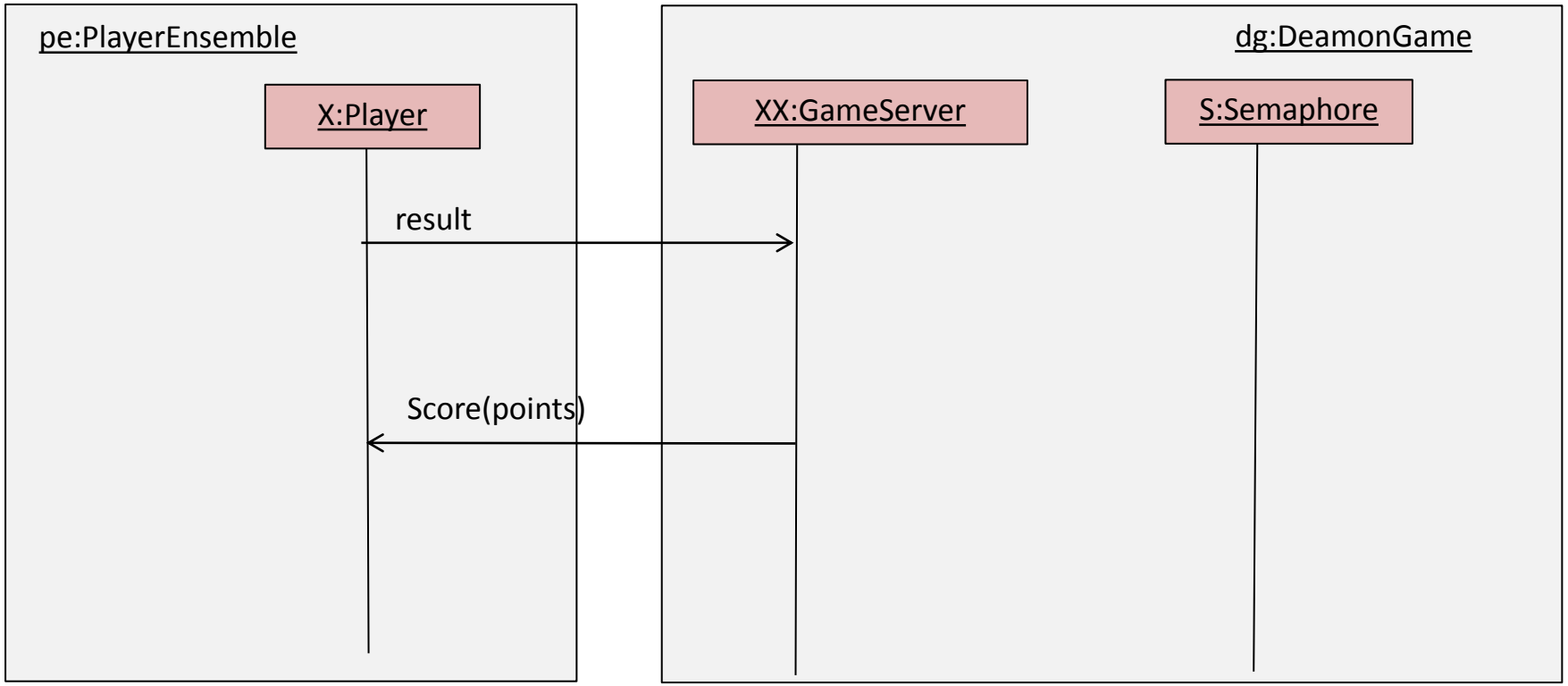
Schnittstellenprotokolle: Anmeldung



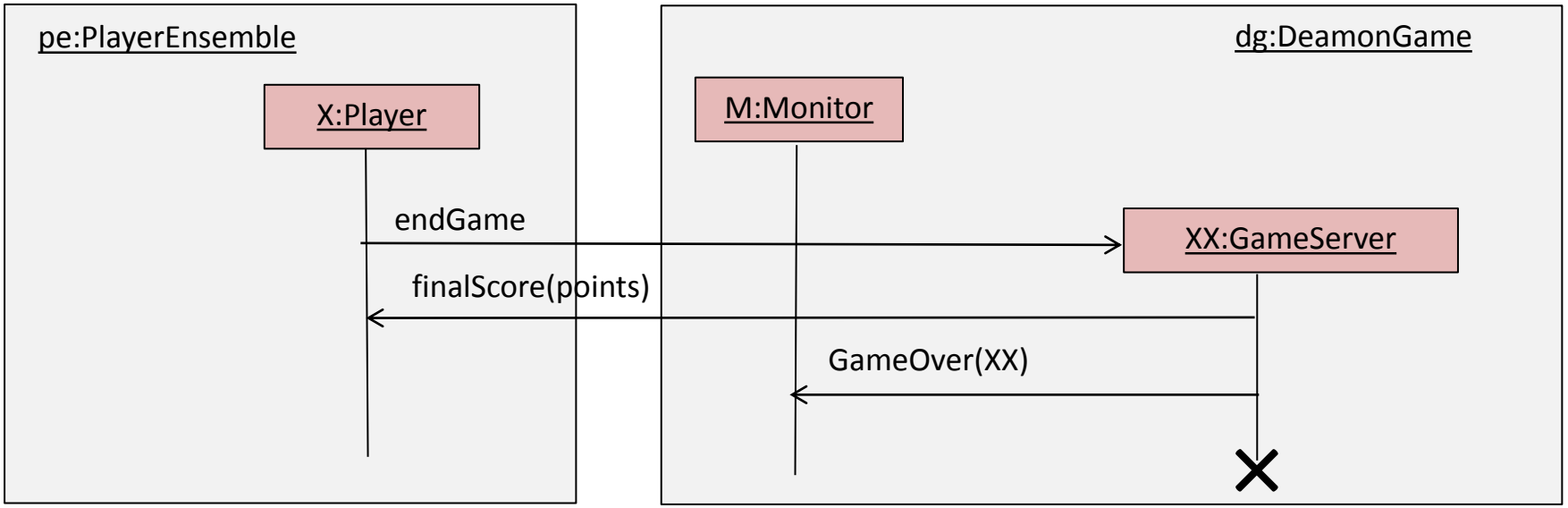
Schnittstellenprotokolle: Spielzug



Schnittstellenprotokolle: Spielstandabfrage



Schnittstellenprotokolle: Abmeldung



Ressourcenfreigabe

A-4: Nutzerverhalten

Normales Verhalten