Modul OMSI-2
im SoSe 2010

Objektorientierte Simulation mit ODEMx

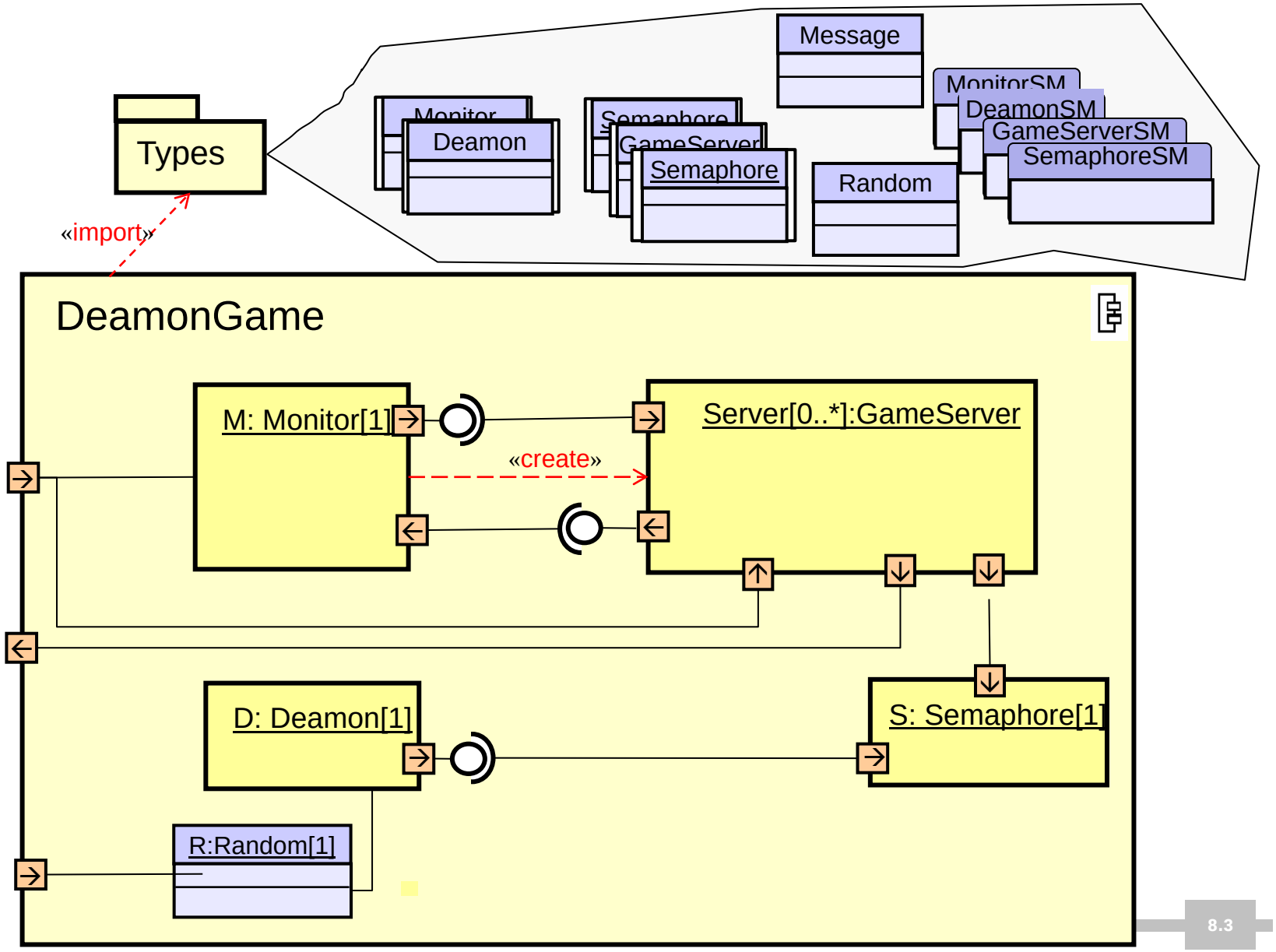
Prof. Dr. Joachim Fischer
Dr. Klaus Ahrens
Dipl.-Inf. Ingmar Eveslage
Dipl.-Inf. Andreas Blunk

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

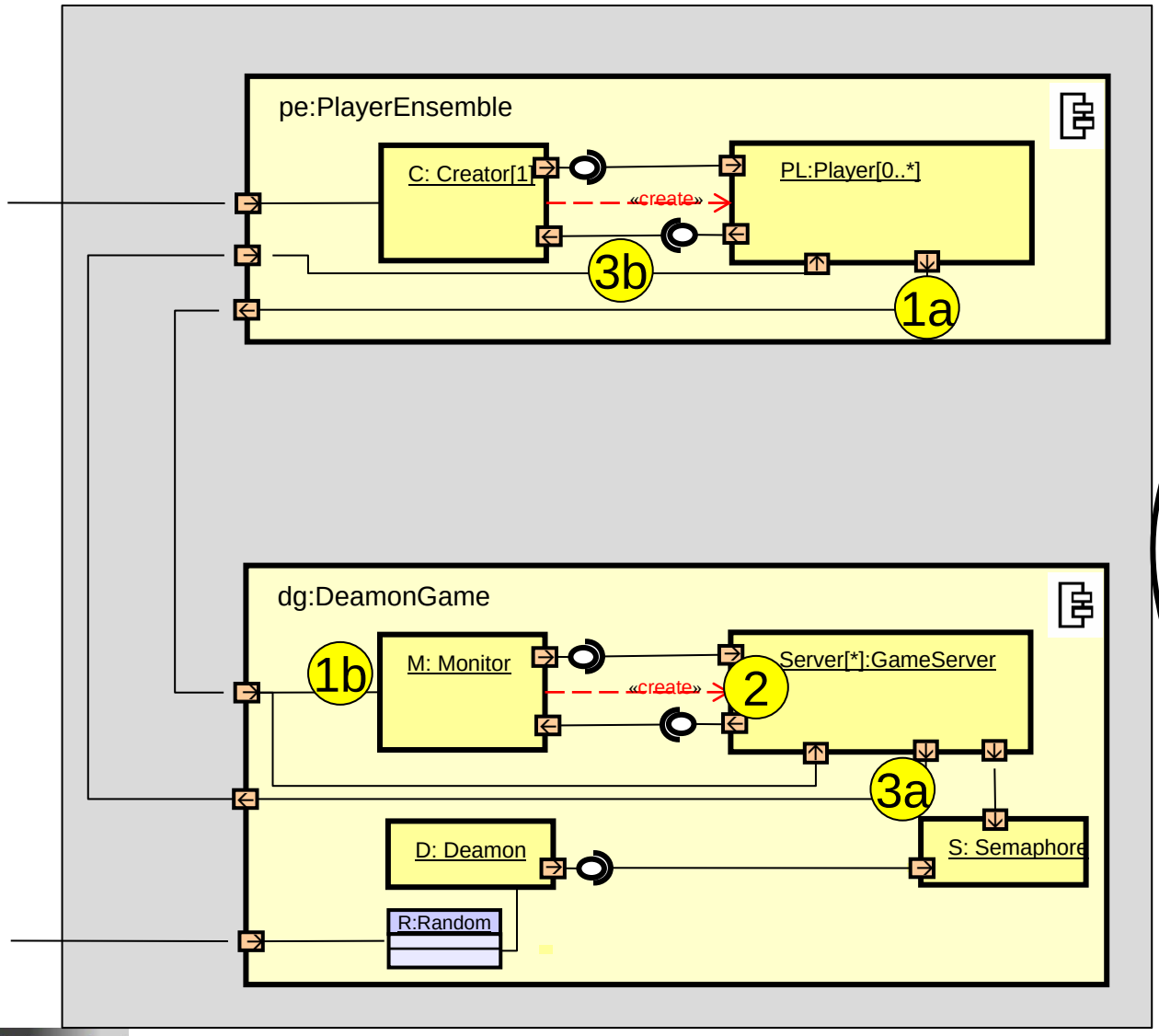
6. SDL

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

UML-Systemtypdefinition: DeamonGame

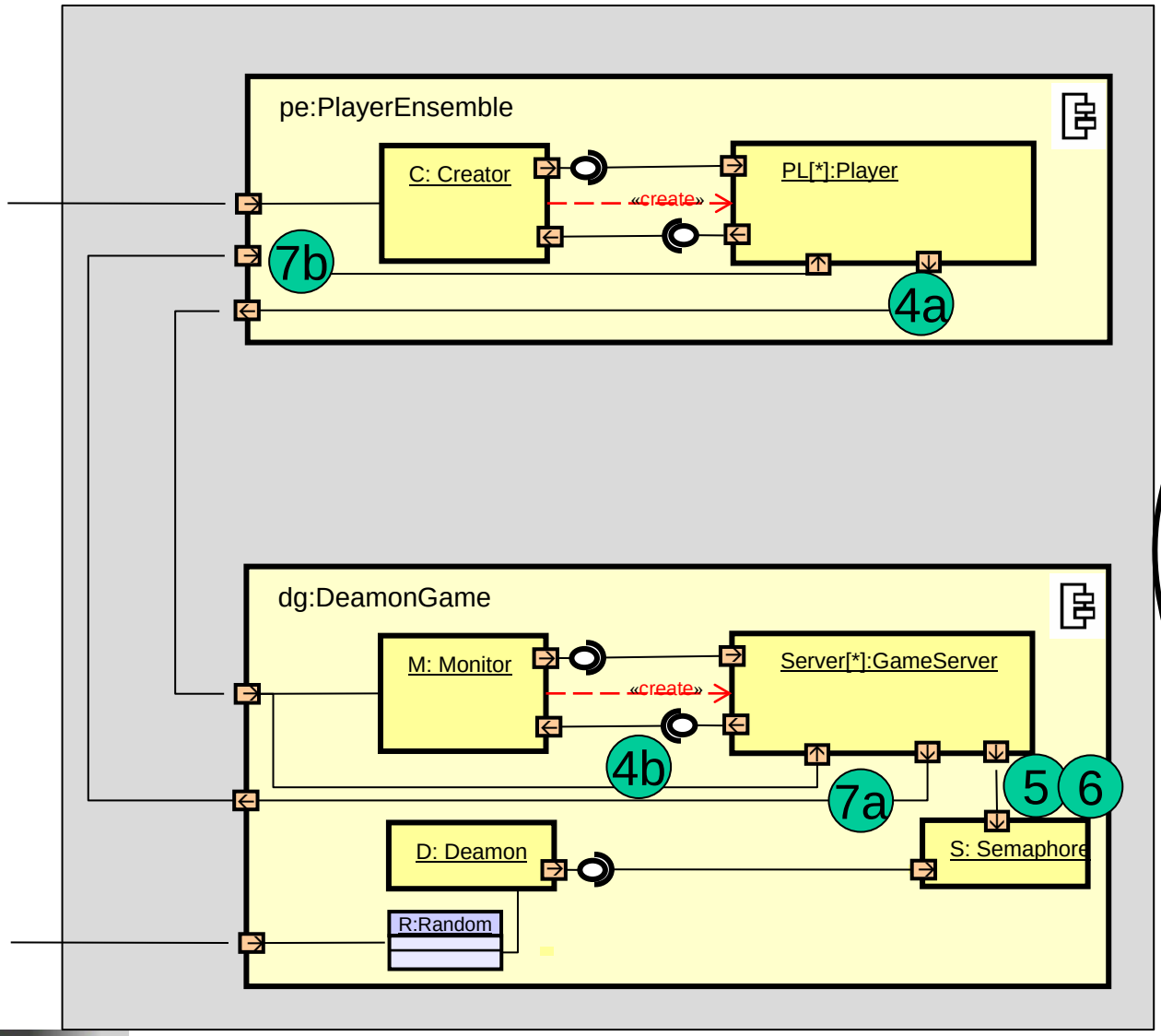


Erweitertes System *(Nutzeranmeldung)*



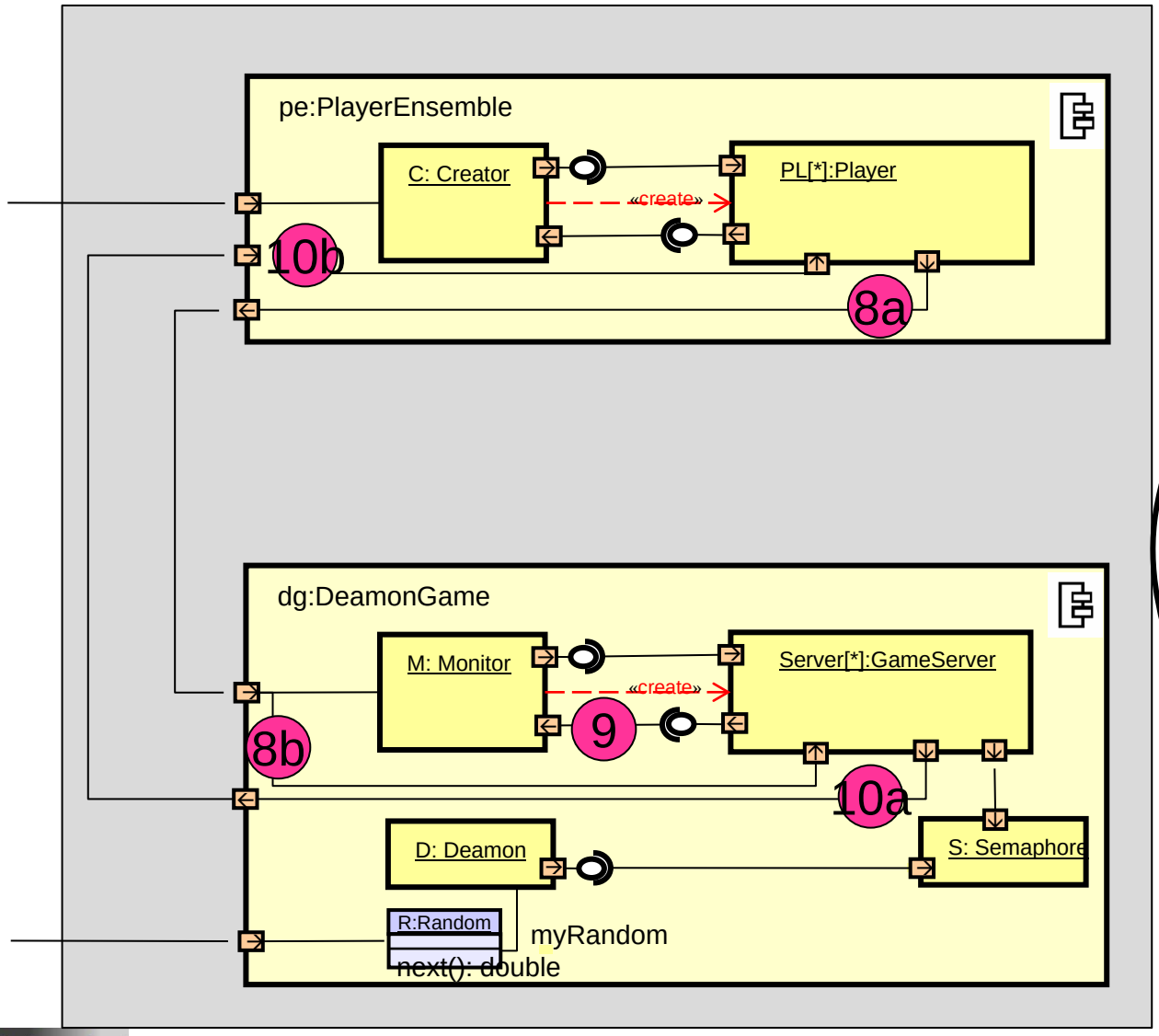
1. Anmeldung des Nutzers
2. Bereitstellung von Ressourcen zum Dienstzugang
3. Zugangsdaten an Nutzer
4. Dienstaktivierung
5. Dienstrealisierung (Start)
6. Dienstrealisierung (Ende)
7. Dienstfinalisierung
8. Abmeldung des Nutzers
9. Ressourcenfreigabe
10. Bestätigung der Abmeldung

Erweitertes System *(Dienstnutzung)*



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

Erweitertes System *(Nutzerabmeldung)*



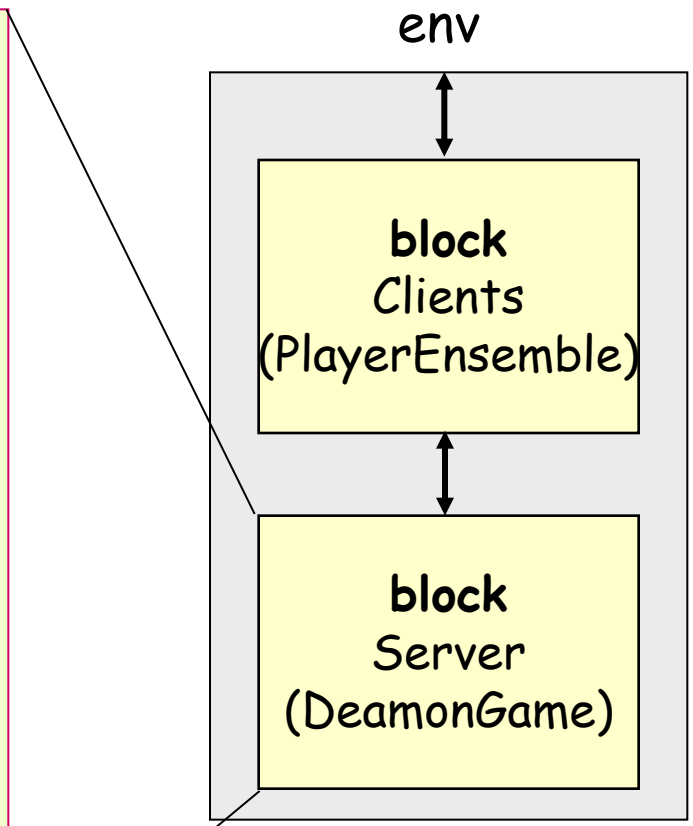
1. Anmeldung des Nutzers
2. Bereitstellung von Ressourcen zum Dienstzugang
3. Zugangsdaten an Nutzer
4. Dienstaktivierung
5. Dienstrealisierung (Start)
6. Dienstrealisierung (Ende)
7. Dienstfinalisierung
8. Abmeldung des Nutzers
9. Ressourcenfreigabe
10. Bestätigung der Abmeldung

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

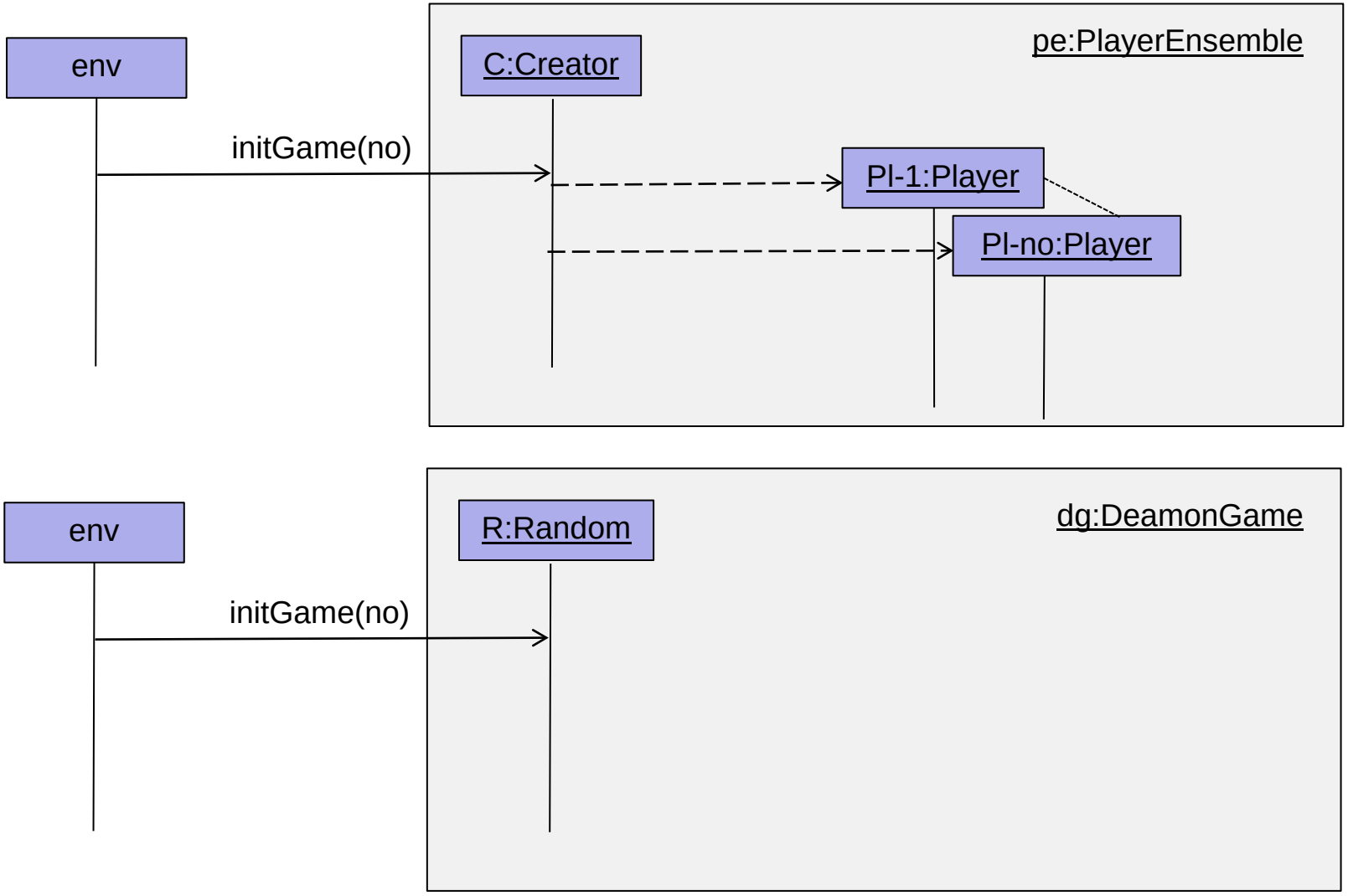


Informale Festlegung des Dienstes (2)

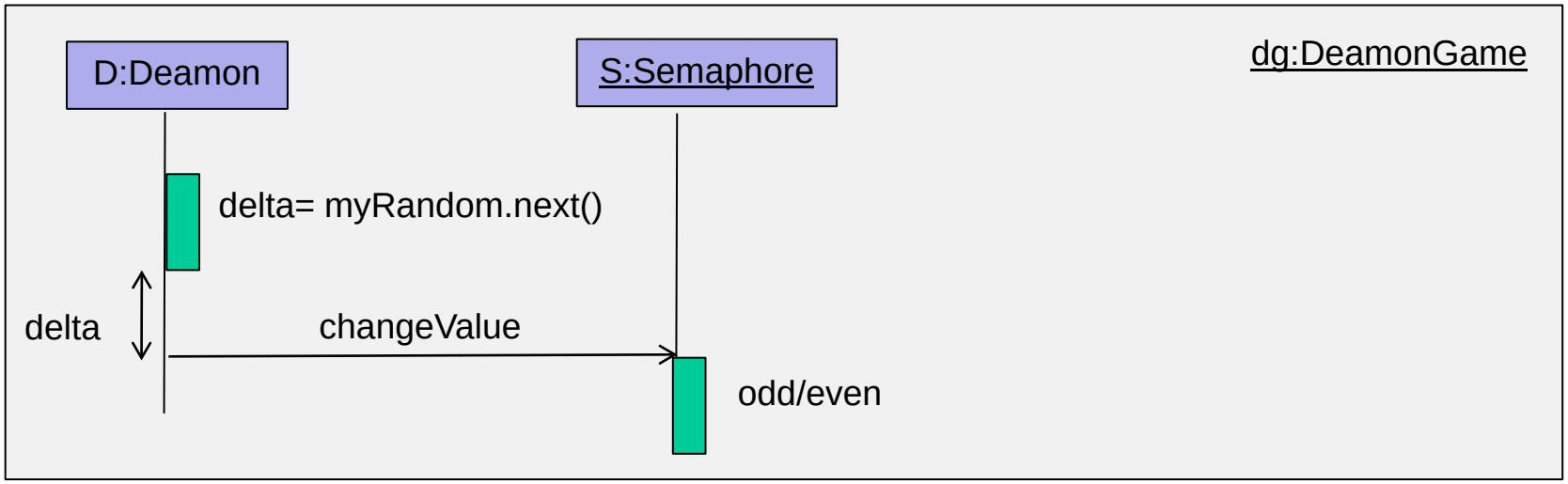
Spielregeln von DeamonGame

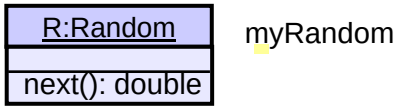
- der Wert **einer** nicht sichtbaren Variable ändert sich nichtdeterministisch: even ←→odd
- zu diskreten Zeitpunkten rät ein Spieler (als Client), ob der Wert ungerade (odd) ist
  - ist das der Fall, **gewinnt** er einen Punkt
  - wenn nicht, **verliert** er einen Punkt
- zu jedem Zeitpunkt kann von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

Schnittstellenprotokolle: Spielinitialisierung

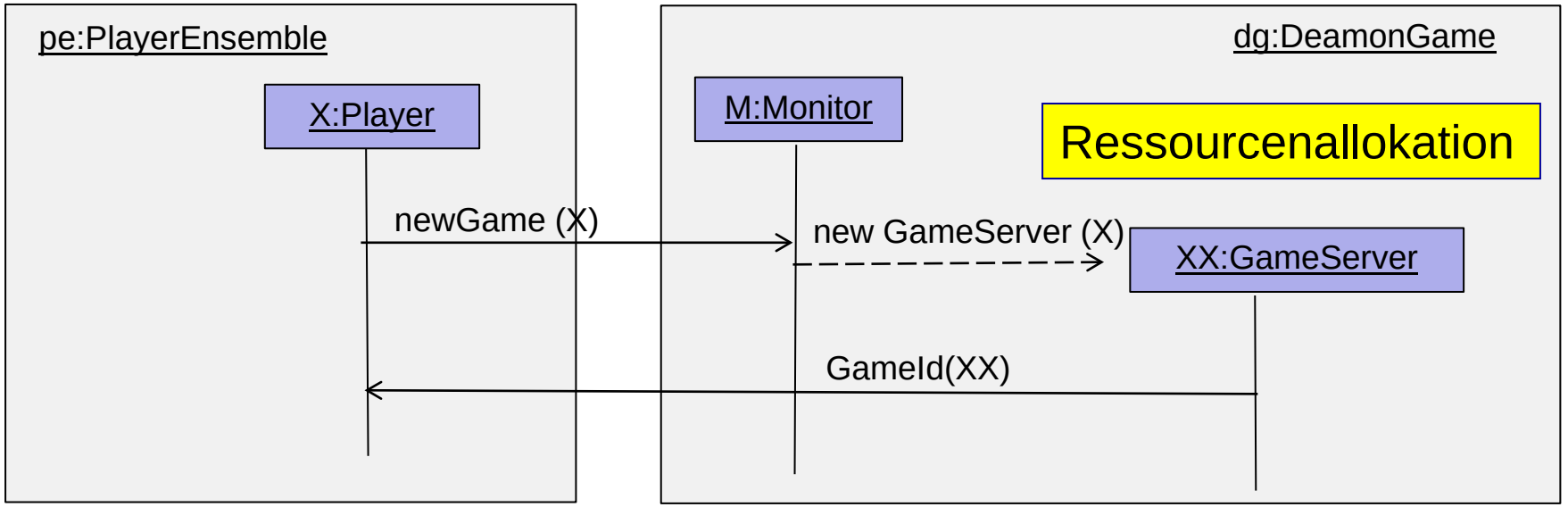


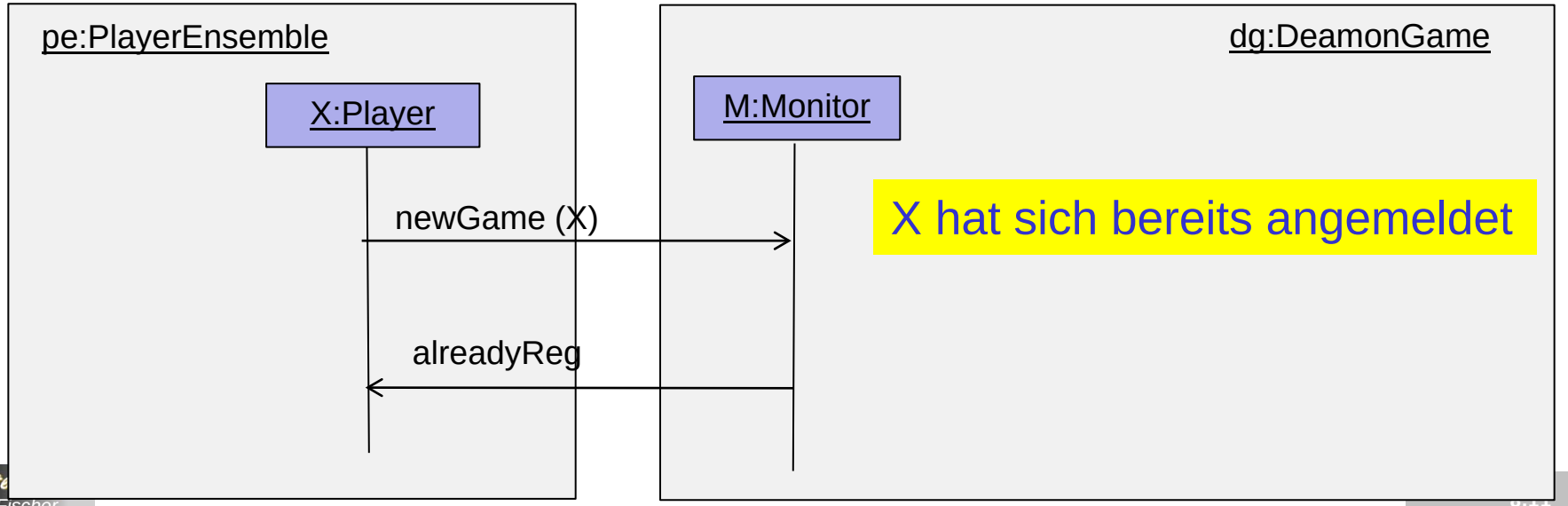
Schnittstellenprotokolle: Zufällige Variablenänderung



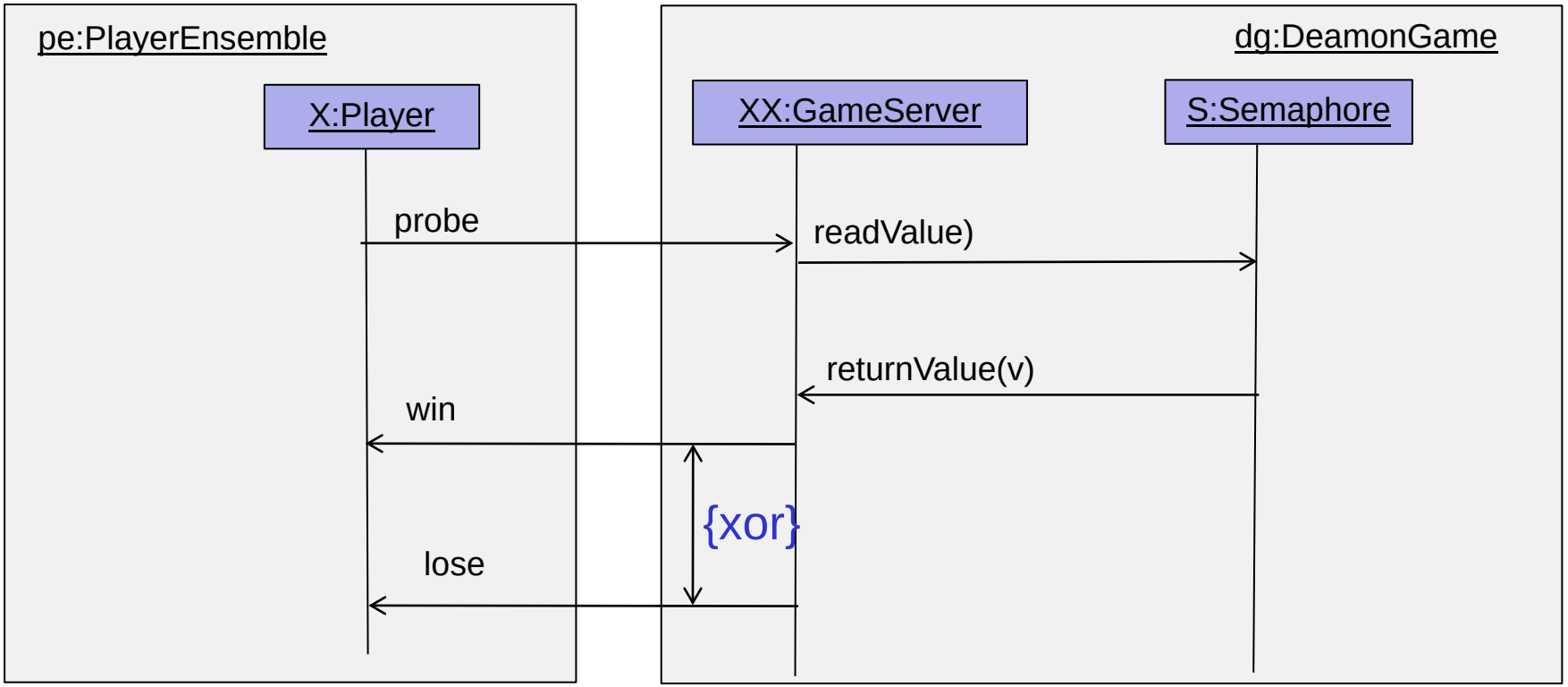


Schnittstellenprotokolle: *Anmeldung*

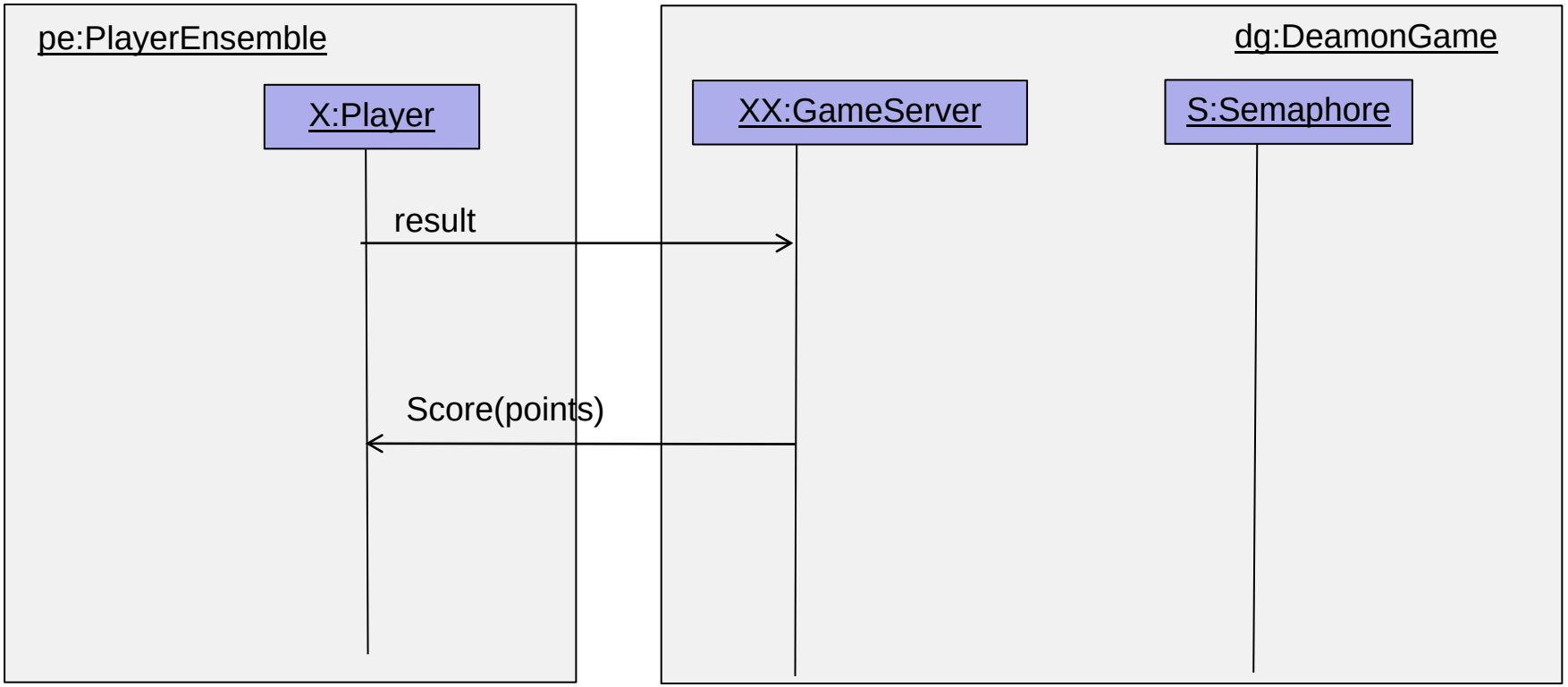




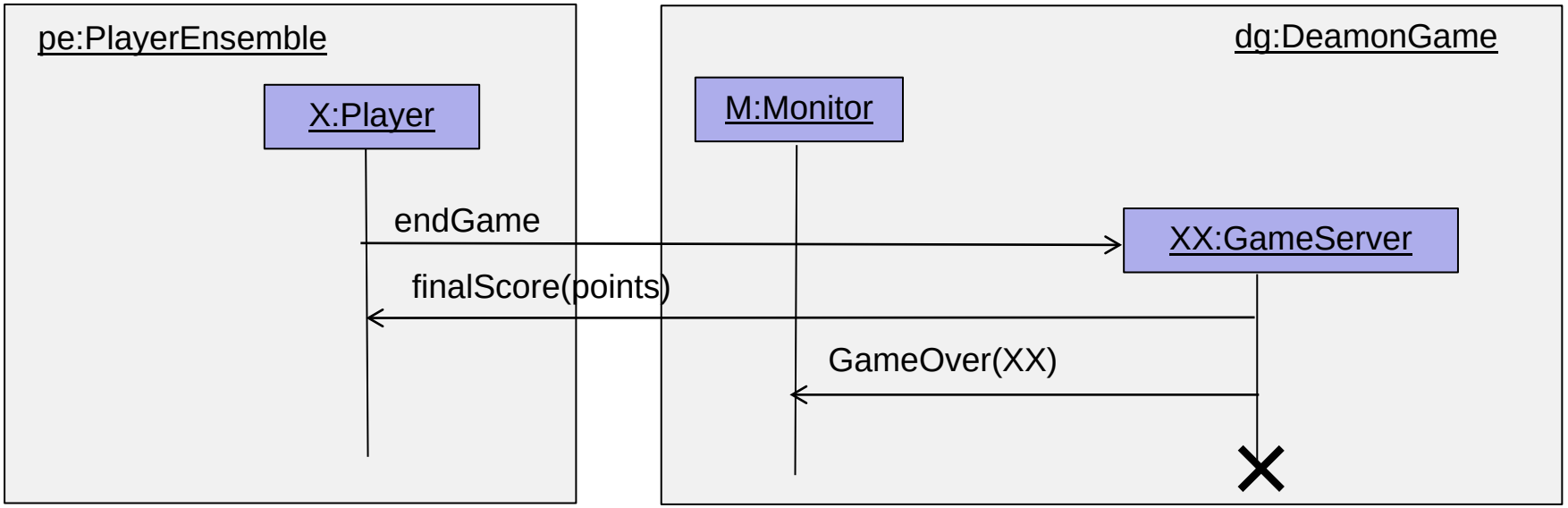
Schnittstellenprotokolle: Spielzug



Schnittstellenprotokolle: Spielstandabfrage



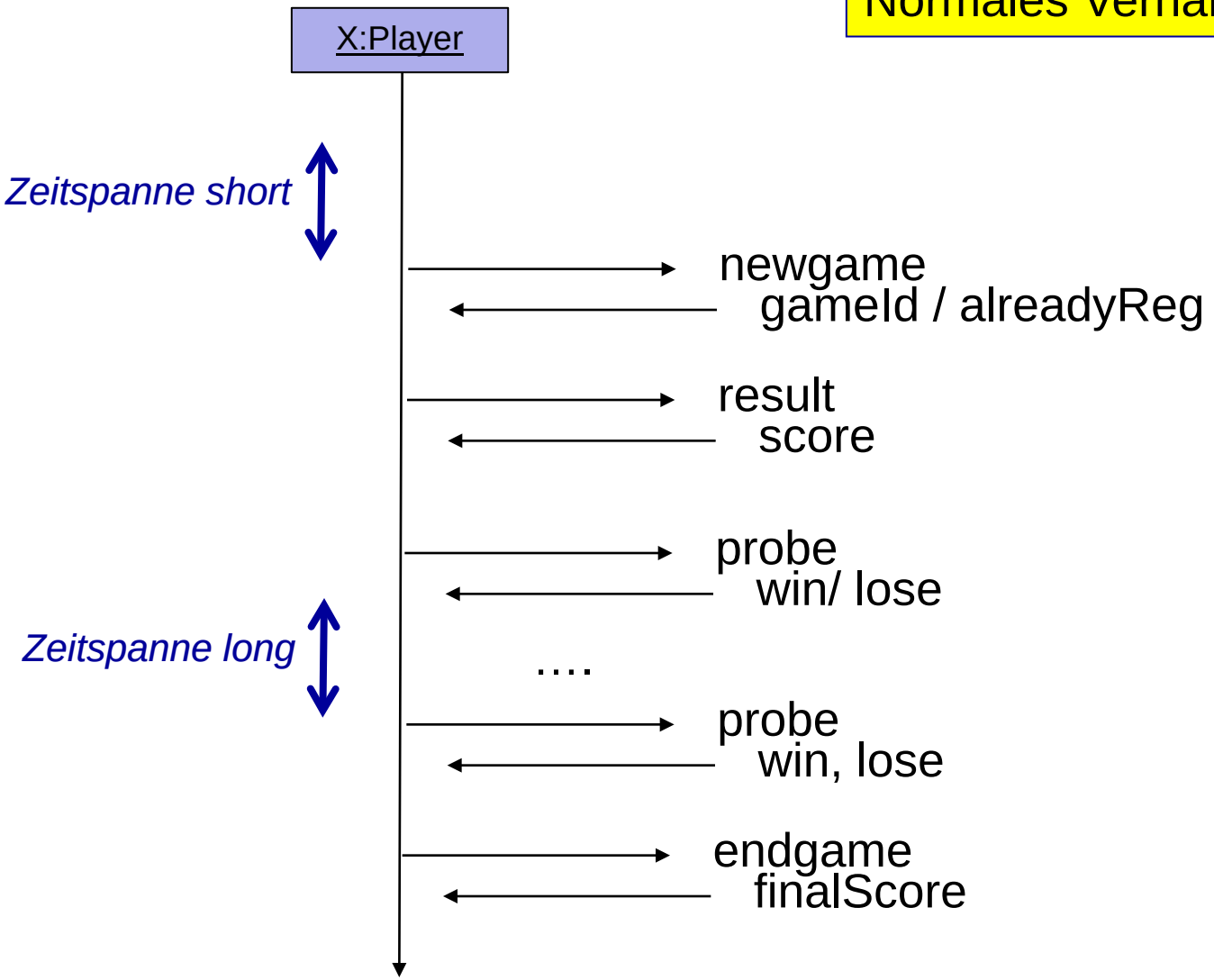
Schnittstellenprotokolle: *Abmeldung*



Ressourcenfreigabe

A-4: Nutzerverhalten

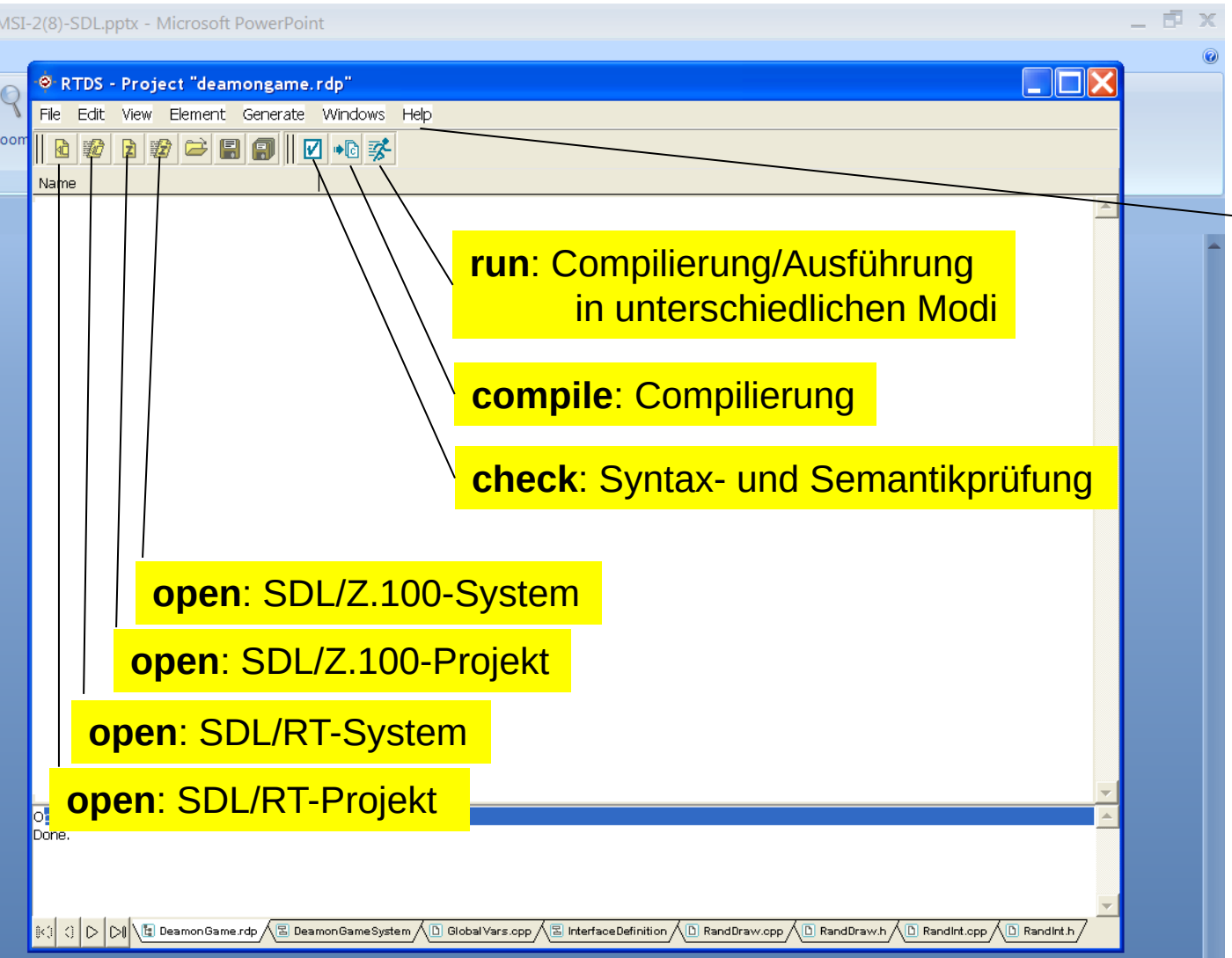
Normales Verhalten



6. SDL

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

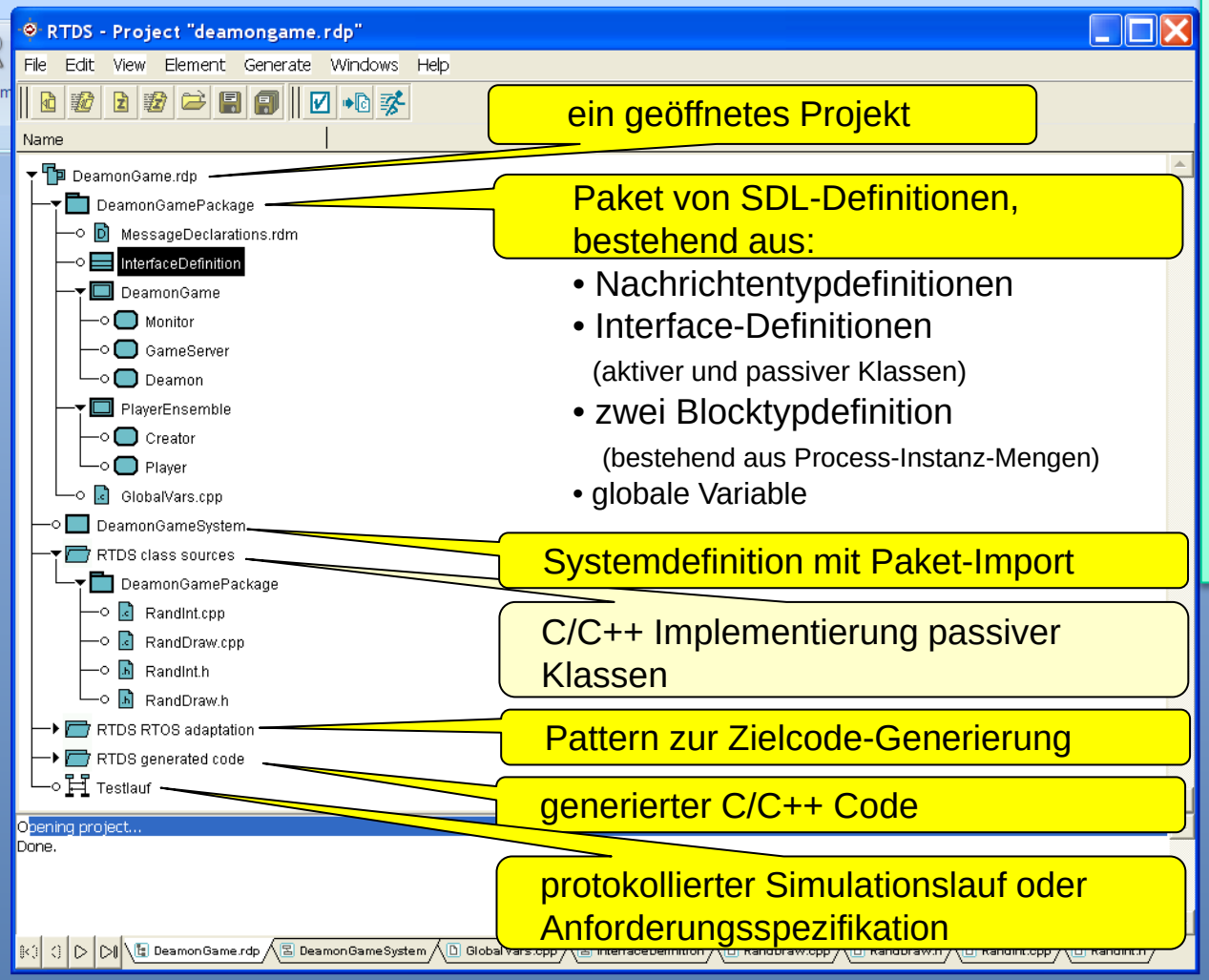
PragmaDev- Developer Studio



Operationen:

- User Manual
- Reference Manual für SDL-Dialekte
- Tutorial

Projektmanager



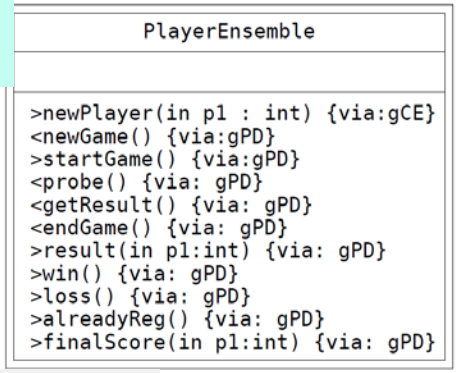
Operationen:

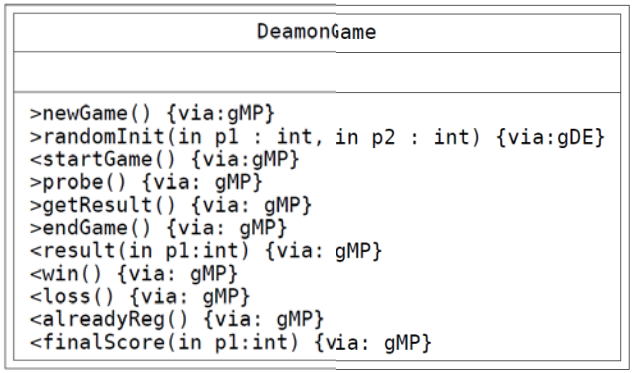
- Hinzufügen und Löschen von Projektkomponenten
- Namensänderungen
- Anklicken der angelegten Komponente führt zur Eröffnung spezifischer Editoren:
  - Text
  - SDL/UML (entspr. Diagrammtyp)
  - C/C++
  - …

Paket- Komponente: Interface-Definition

aktive Klassen
(explizite Blocktypen)

Richtungsangabe: </>
Nachrichtentyp: Id
(Signatur) : Parameterliste
{via: Gate-Angabe}: Id

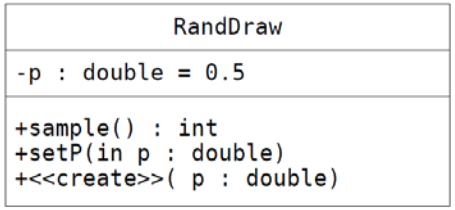


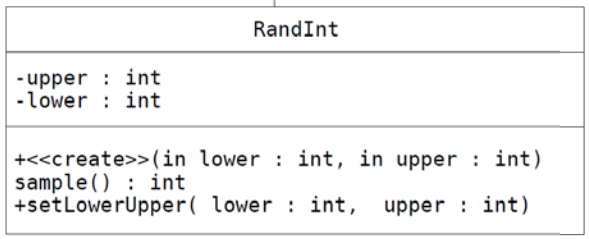


Jede Blockinstanz hat individuelle
Abschlusspunkte für Kanäle
(Anschlusskompatibilität gefordert)

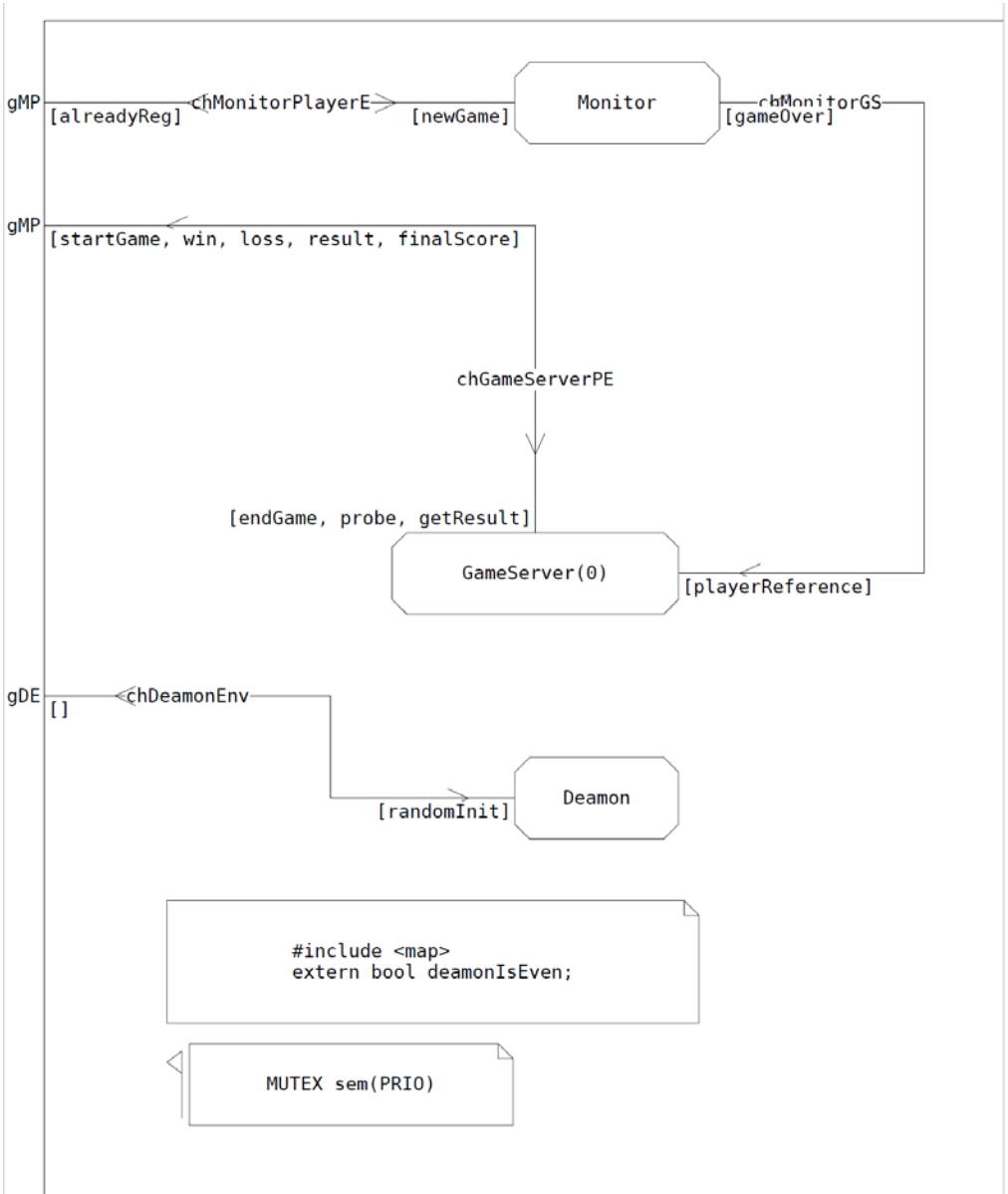
aktive Klasse
(impliziter Process-Typ)

passive Klassen
(explizite Klassen)





Paket- Komponente: Bocktyp-Definition



Deamon-Game

Gate
entspr.
der Interface-
Definition

Paket-Komponente: Deklarationsdatei

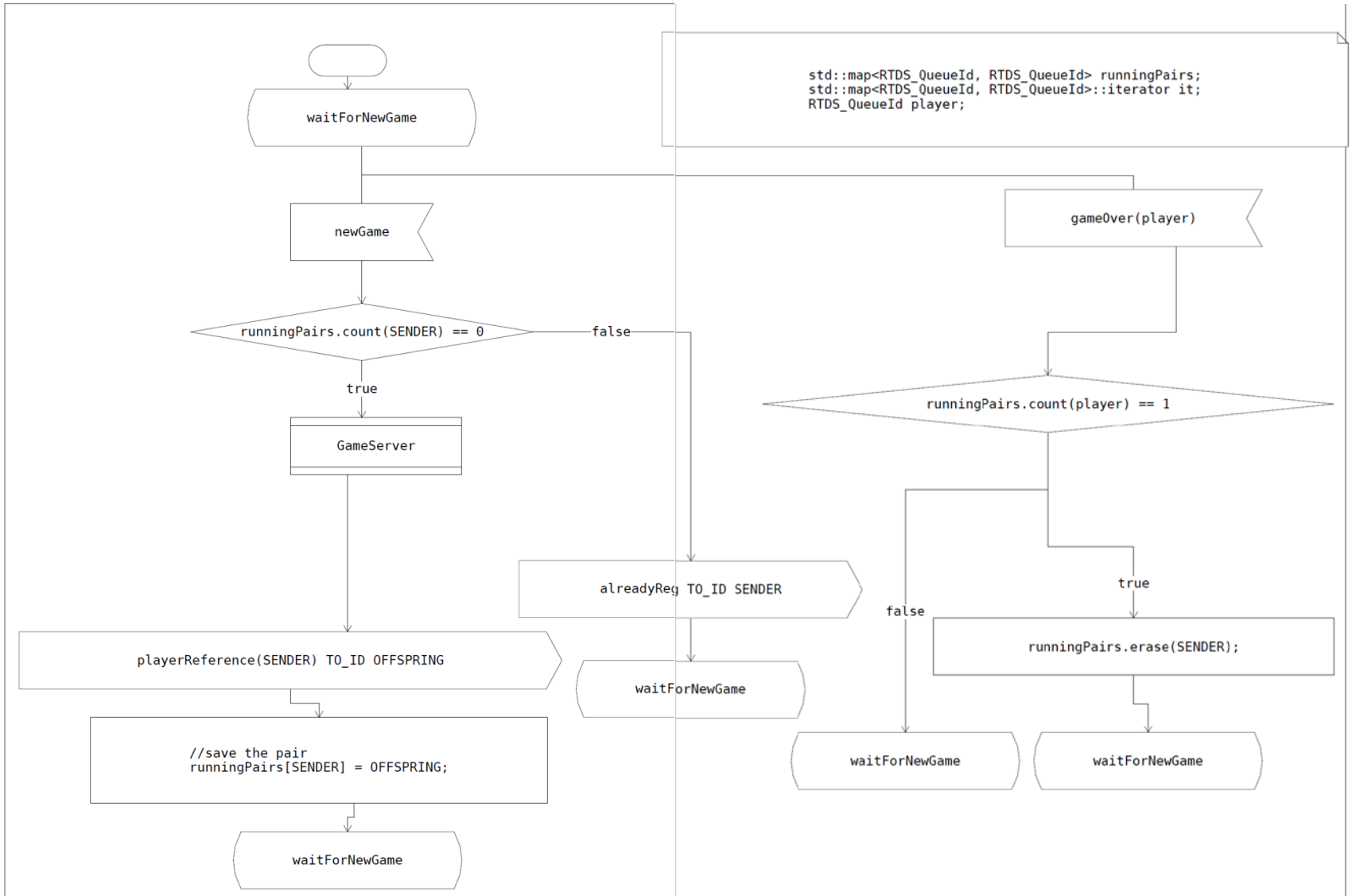
```
MESSAGE newPlayer(int);
MESSAGE newGame();
MESSAGE randomInit(int, int);
MESSAGE startGame();
MESSAGE playerReference(RTDS_QueueId);
MESSAGE endGame();
MESSAGE getResult();
MESSAGE result(int);
MESSAGE probe();
MESSAGE win();
MESSAGE loss();
MESSAGE gameOver(RTDS_QueueId);
MESSAGE finalScore(int);
MESSAGE alreadyReg();

MESSAGE_LIST pe2dg = newGame, probe, getResult, endGame;
MESSAGE_LIST dg2pe = startGame, result, win, loss, finalScore, alreadyReg;
```

Nachrichtentypen

Listen von Nachrichtentypen

Block(typ)-Komponente: Process



Weitere mögliche Paket-Komponenten

- Dateiverzeichnis
- Paket

  } Unterstrukturierung der Bestandteile
- System — ???
- aktive Klassen — Block-Typ, Process-Typ
- passive Klassen — UML-Klassendiagramme (Klassen, Assoziationen)
- Anwendungsfälle — UML-Use-Case-Diagramme
- SW-Komponenten — UML-Deployment-Diagramme
- ??? — IF-Observer-Diagramme
- C,C++ — Header- und Impl.Dateien
- Testfälle — TTCN-3-Dateien
- Standard für Datenübertragung — ASN.1
- Makros — ???
- MSC, HMSC — MSC-, HMSC-Diagramme (UML-SD-Diagramme)
- Dokumente — Textdatei
- Extern — ???
- GUI — GUI-Datei