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

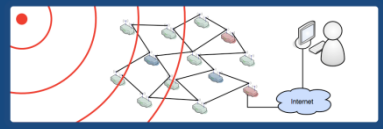
SAFER
Seismic eArly warning For EuRope

EDIM

SOSEWIN-extented

GFZ
Helmholtz-Zentrum
POTSDAM

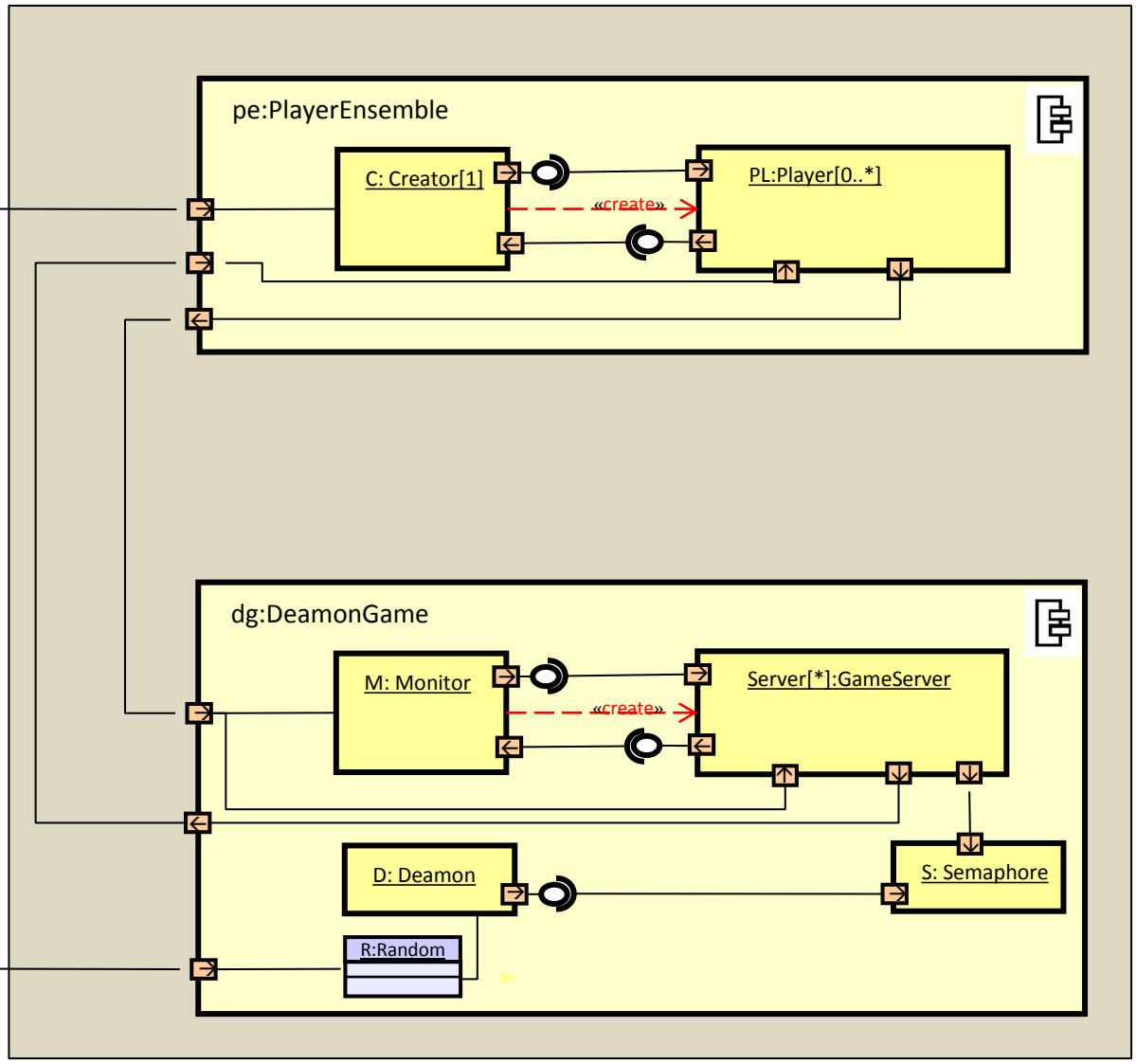
Systemanalyse



6. SDL

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

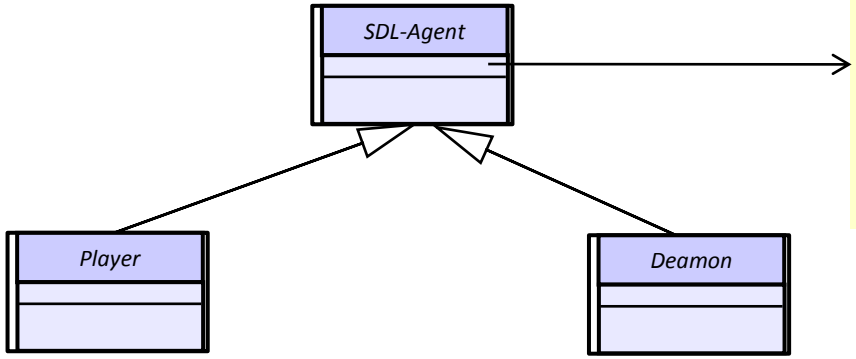
Erweitertes System



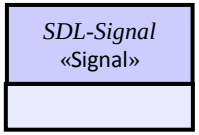
UML-like Definition

- Strukturierte Klassen
  (SDL: Blocktypen)
- Ports
  (SDL: Gates)
- Events
  (SDL: Ankunft von Nachrichten)
- Interface
  (SDL-RT: Schnittstelle aktiver Klassen)
- Parts
  (SDL: Blockinstanzmenge/ Prozessinstanzmenge)
  Unterschied:
  Kardinalitätsangabe
- Verbindung
  (SDL: gerichteter Übertragungskanal)

Weitere Präzisierungen (SDL als UML-Profil)



Existenz verbunden mit Vergabe systemeindeutiger PId-Werte

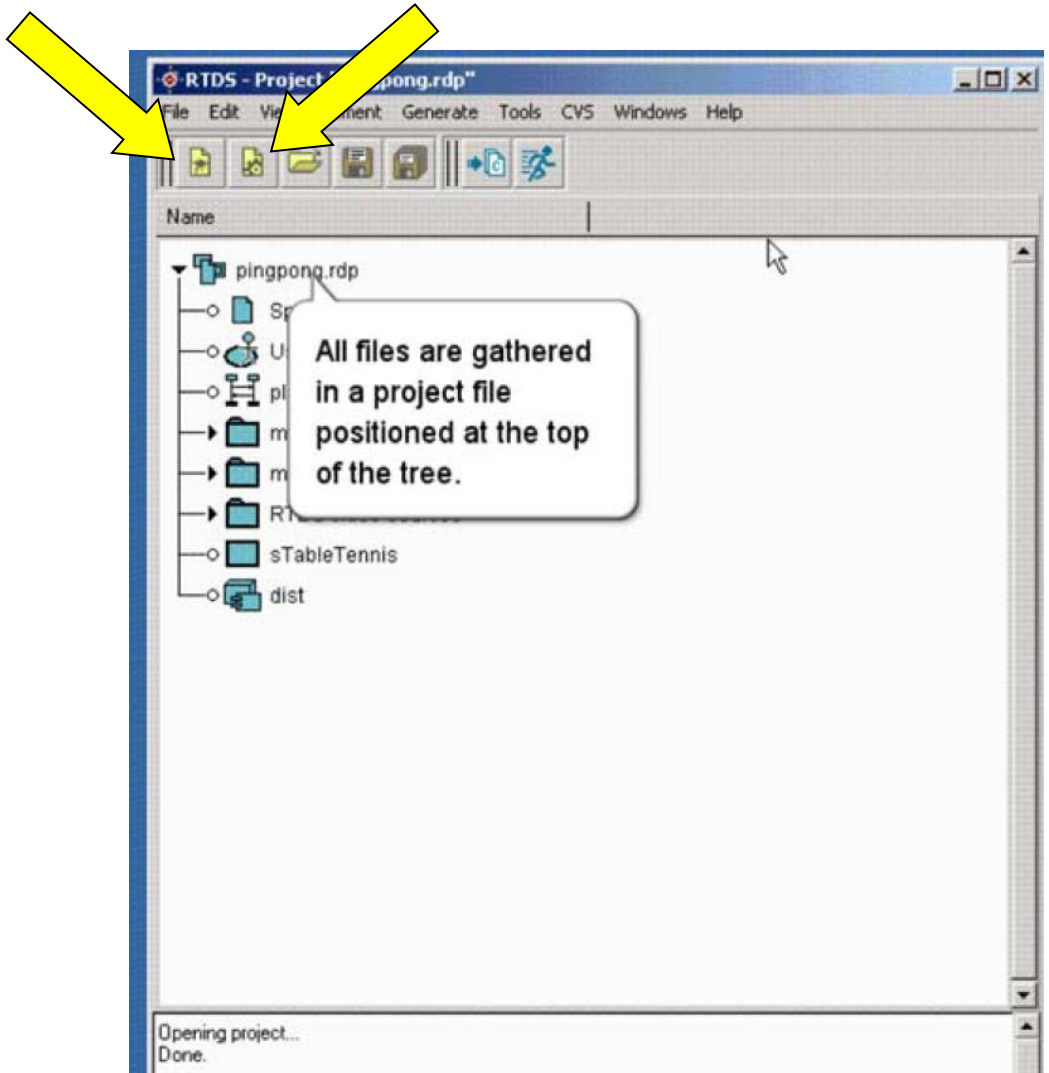


jedes Signal überträgt die PId des sendenden Prozesses

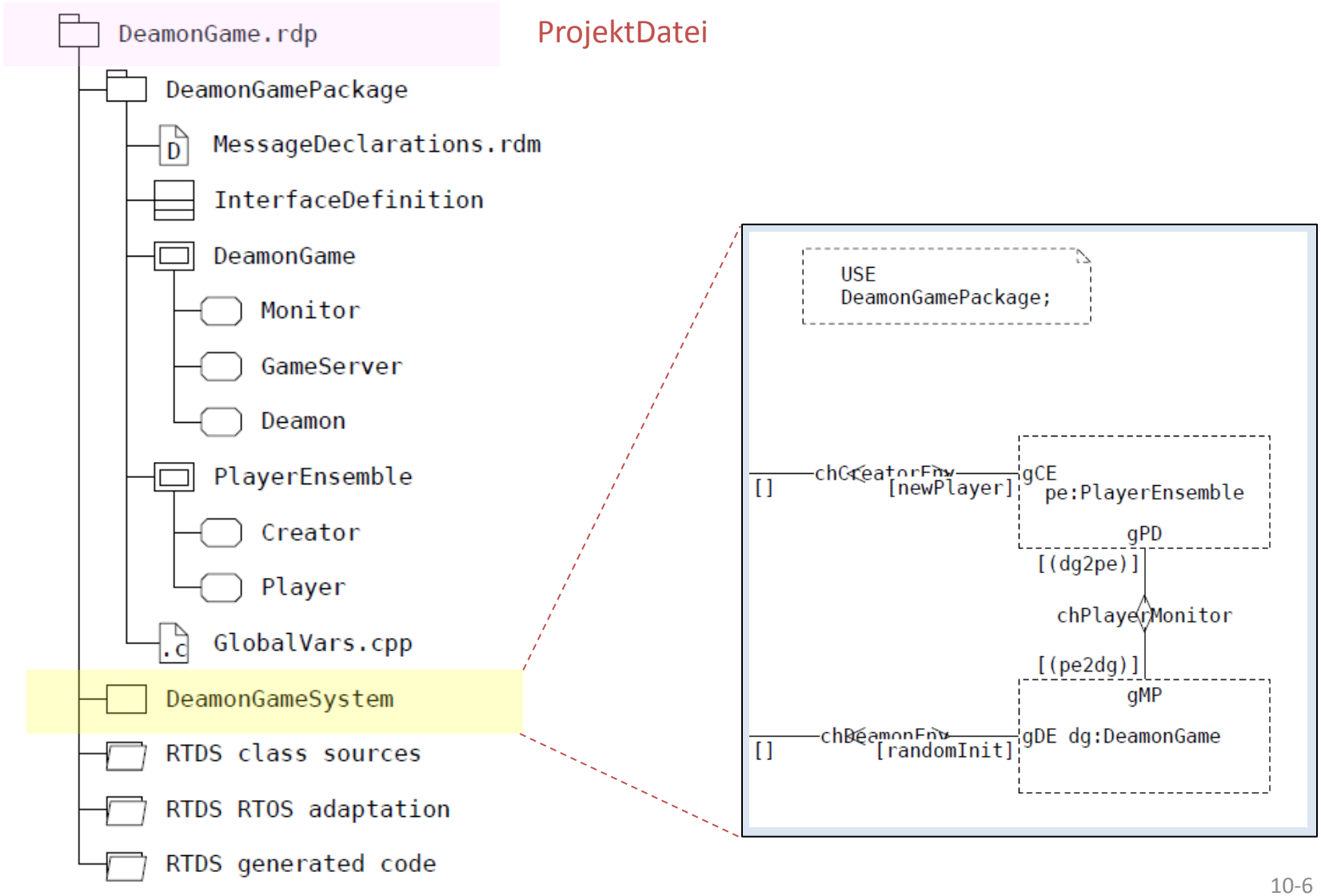
PragmaDev DS- Project Manager

SDL-RT

SDL-Z100

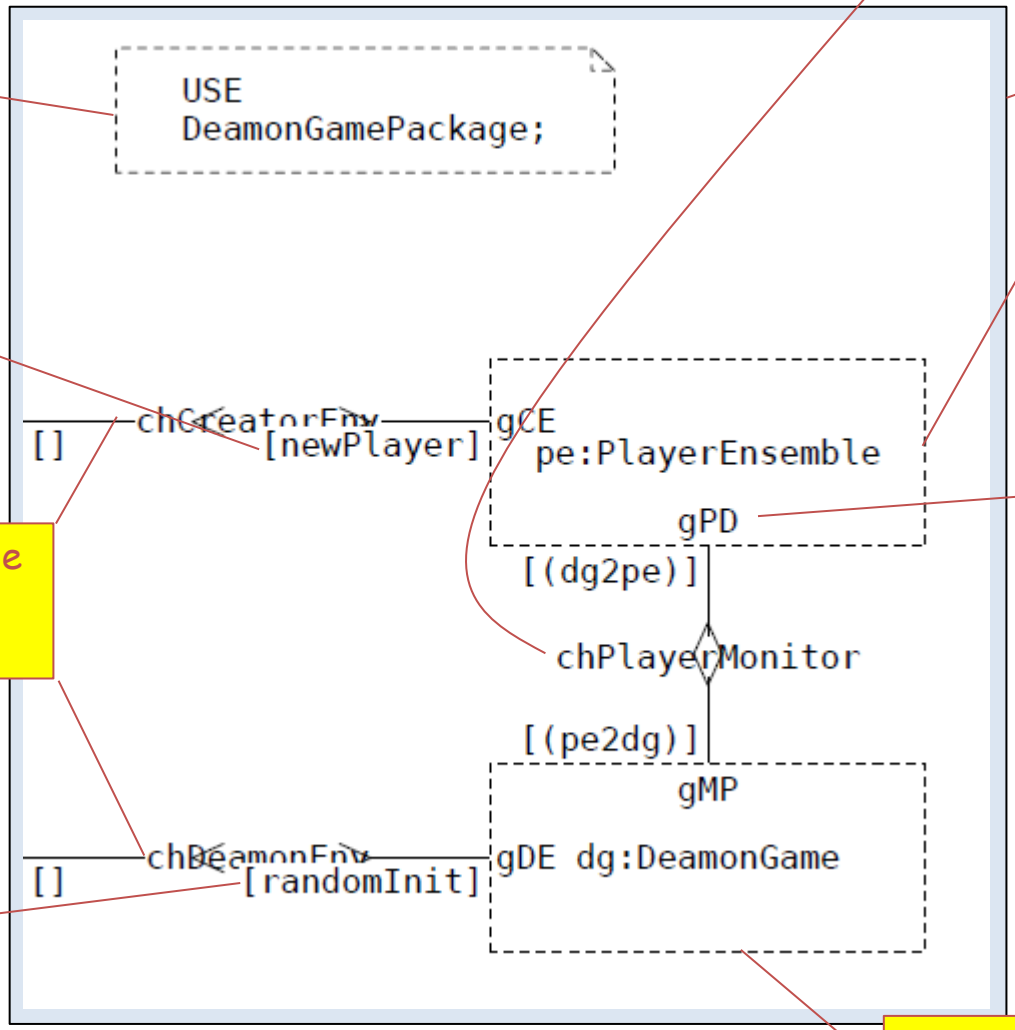


DeamonGame-Projekt



Systemname entspricht Dateinamen/Projektnamen

Systemspezifikation



bidirektionalerKanal chPlayerMonitor mit Angabe zu transportierender Nachrichten

Package-Import

Systemgrenze

Signal newPlayer von der Umgebung

Block pe vom Typ PlayerEnsemble

Gate gPD (= UML-Port) mit Eingang dg2pe und Ausgang p22dg

unidirektionale Kanäle chCreatorEnv chDeamonEnv

Nachrichten-Listen dg2pe pe2dg

Signal randomInit von der Umgebung

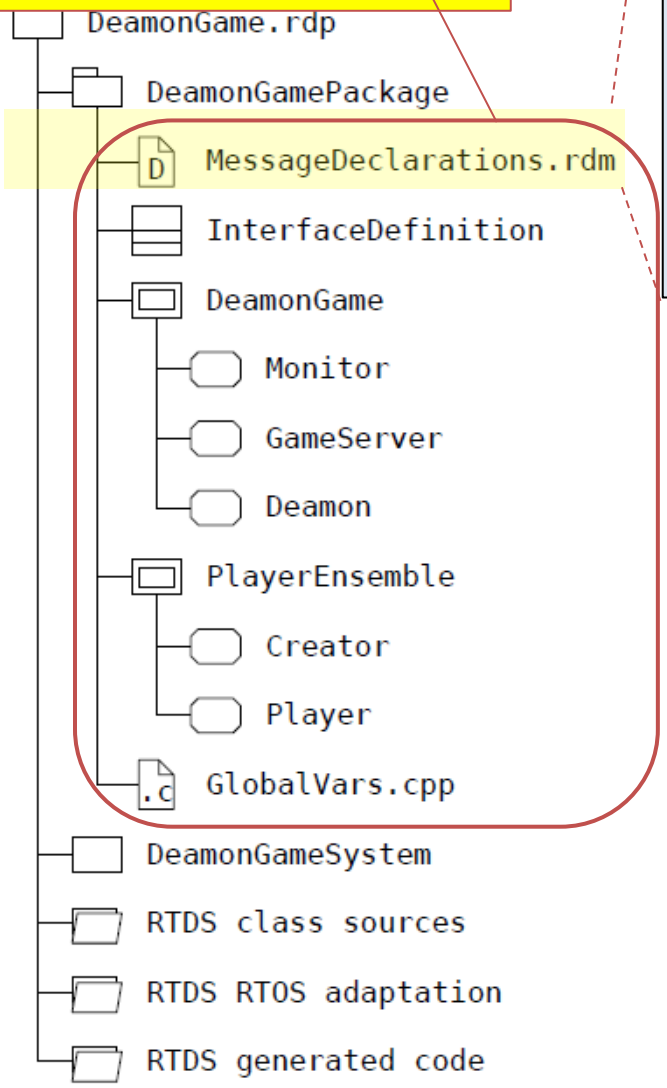
entweder lokal oder im Paket defniert

entweder lokal oder im Paket defniert

Block dg vom Typ DeamonGame

Pac

besteht aus verschiedenen Typdefinitionen und systemglobalen Variablen




```
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 stopGameServer();
MESSAGE_LIST pe2dg = newGame, probe, getResult, endGame;
MESSAGE_LIST dg2pe = startGame, result, win, loss;
```

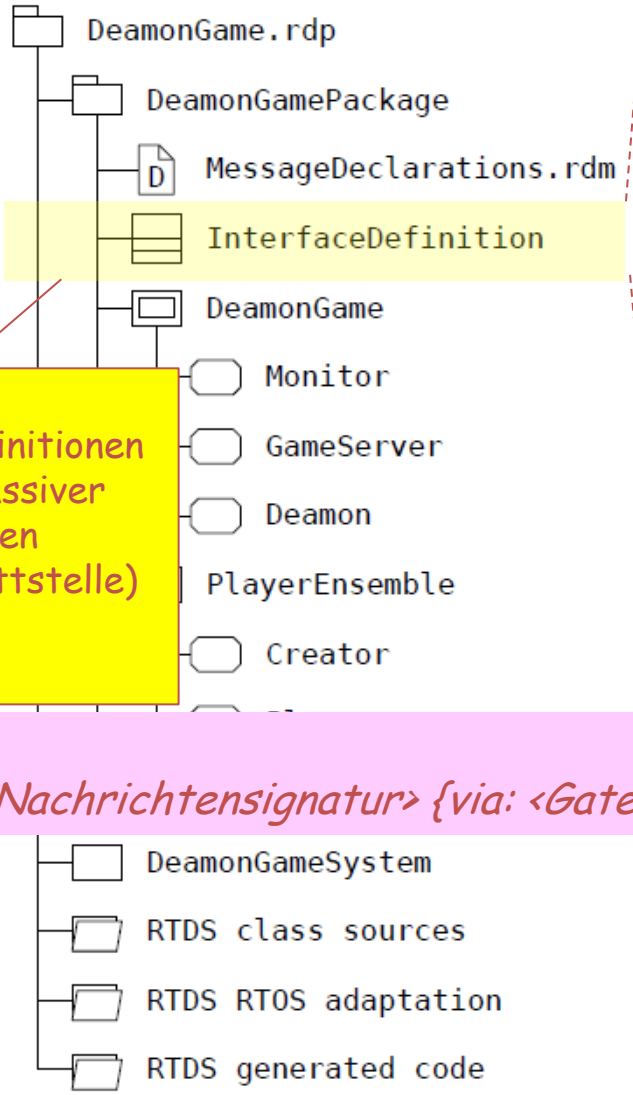
Definition von Nachrichtentypen
und
Definition von Nachrichtenlisten

SDL-RT-Besonderheiten

Nachrichtenparametertypen:
- Standard C-Typen
- SDL-Typ PId
- nutzereigene C++ Typen

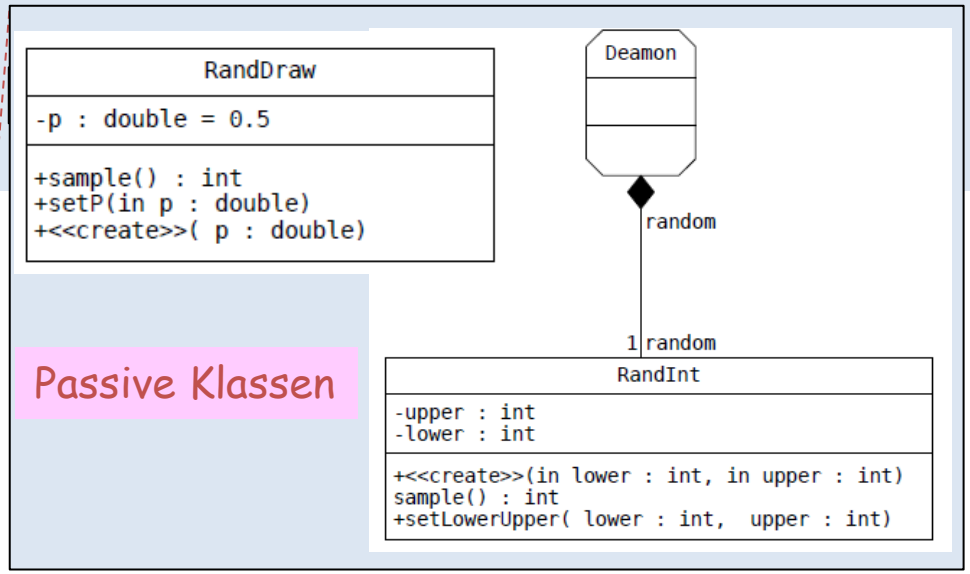
PId = RTDS_QueueId
SIGNAL = MESSAGE

Pac

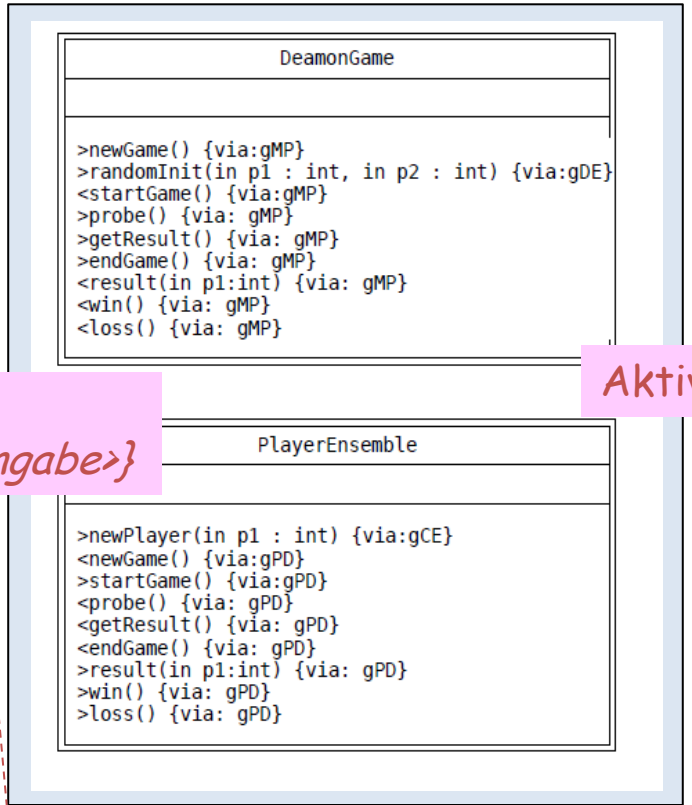


besteht aus UML-like-Definitionen aktiver und passiver SDL-RT-Klassen (äußere Schnittstelle) und deren Assoziationen

Interface:
<Richtung> <Nachrichtensignatur> {via: <Gate-Angabe>}

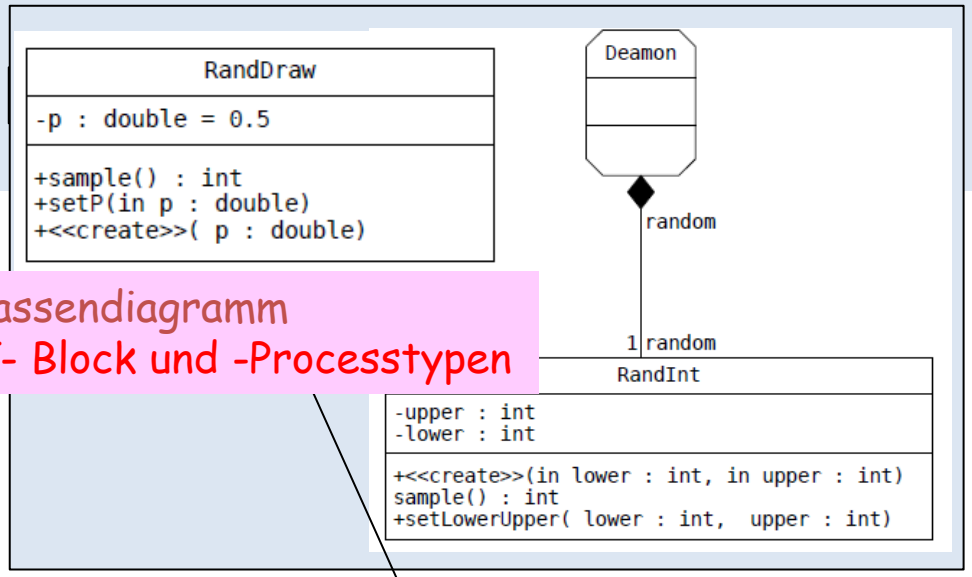


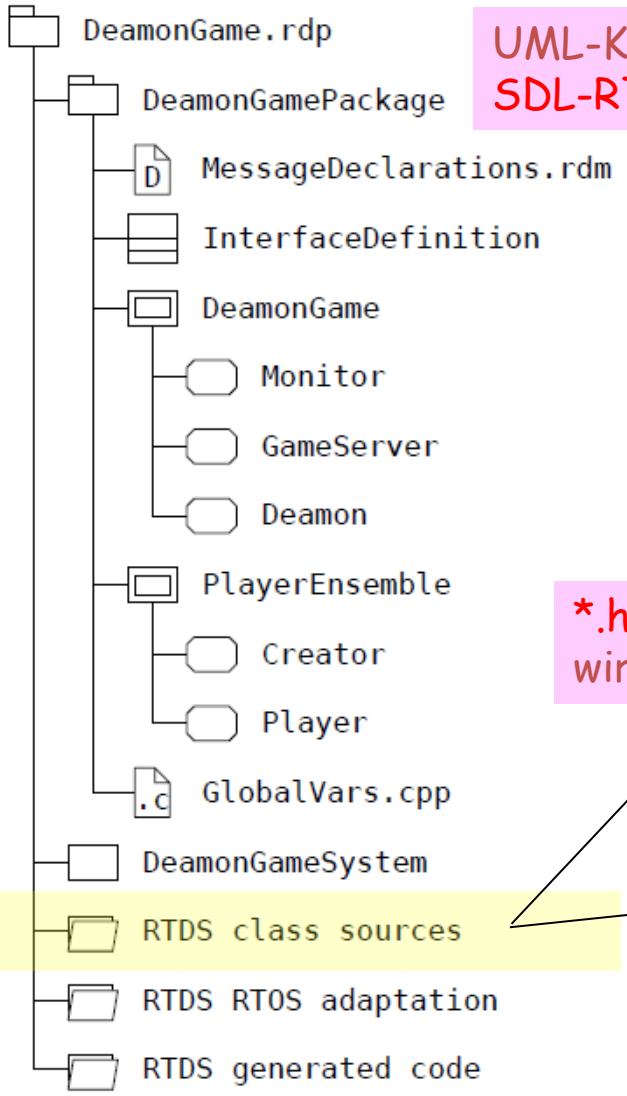
Passive Klassen



Aktive Klassen

Pac





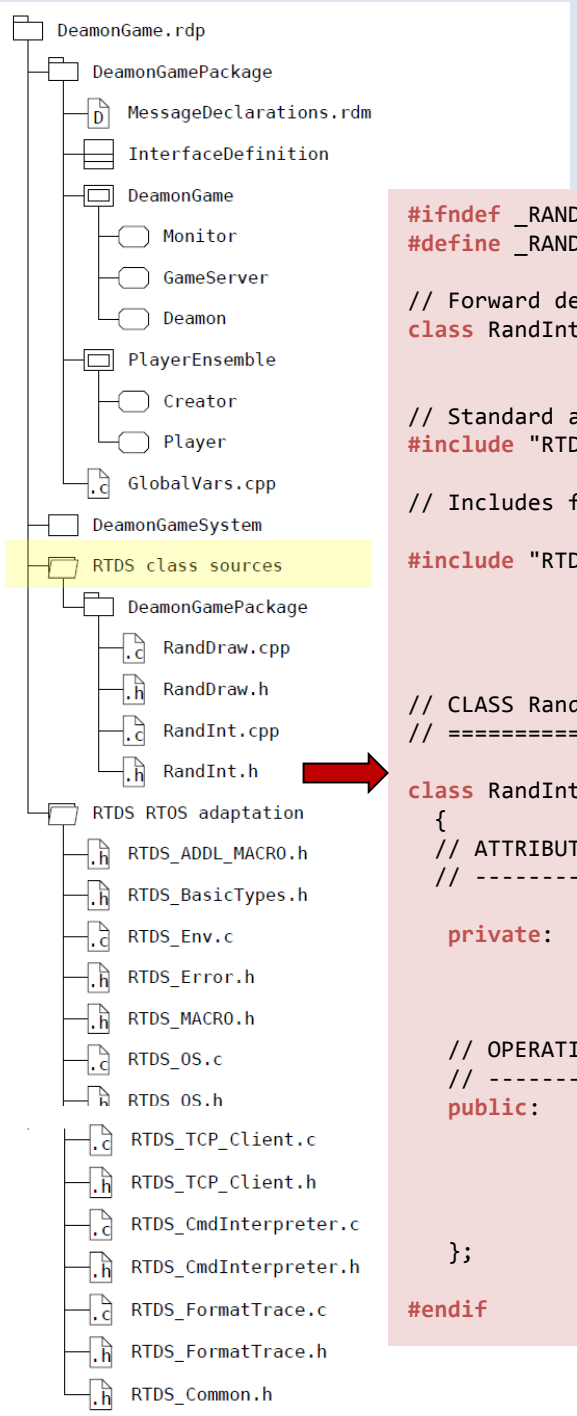
UML-Klassendiagramm
SDL-RT- Block und -Processtypen

Random wird lokale Variable von Deamon
(Typ: RandInt)

*.h
wird generiert

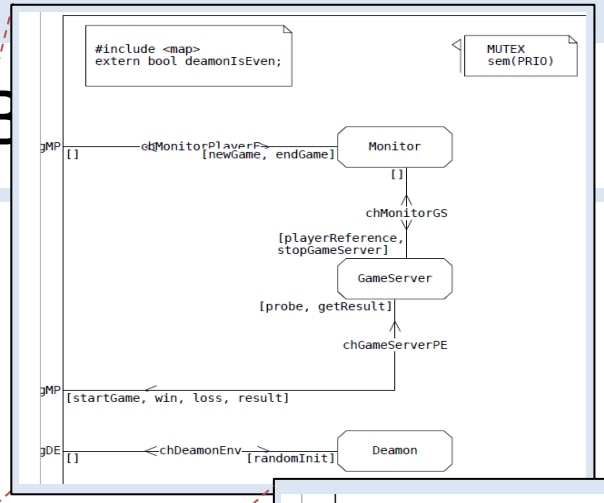
*.ccp
ist nutzerseitig bereitzustllen

C++ -Quellen

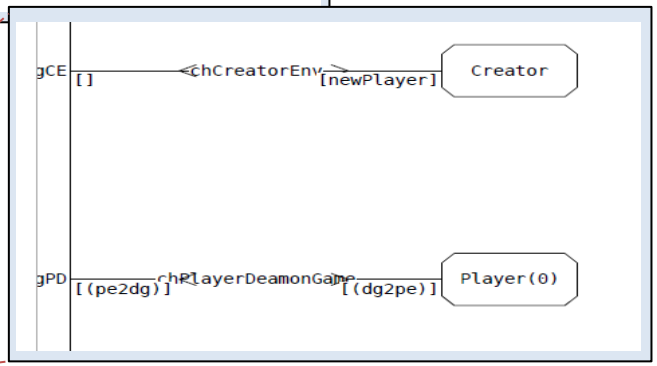



```
#ifndef _RANDINT_H_
#define _RANDINT_H_

// Forward declaration
class RandInt;


// Standard and common includes
#include "RTDS_gen.h"

// Includes for related classes

#include "RTDS_messages.h"




// CLASS RandInt:
// ==============

class RandInt
  {
  // ATTRIBUTES:
  // -----------

   private:
               int    lower;
               int    upper;

   // OPERATIONS:
   // -----------
   public:
               RandInt(int lower, int upper);
               virtual int   sample();
               virtual void  setLowerUpper( int
                             lower, int upper);
   };

#endif
```

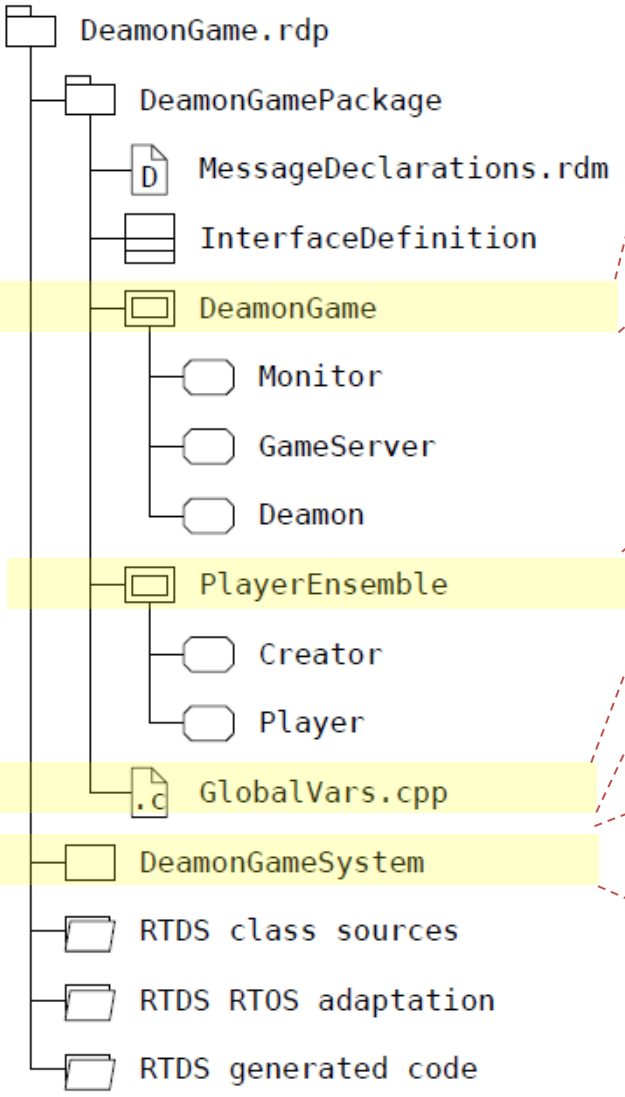
```
#include "DeamonGamePackage\RandInt.h"
#include <cmath>
/*
 * ATTRIBUTES FOR CLASS:
 * ---------------------
 *
 * [From RandInt]
 * - int   lower;
 * - int   upper;
 */


// PUBLIC OPERATIONS:
// ==================


// Operation RandInt:
// ------------------
RandInt::RandInt(int lower, int upper)
               {
               this->lower = lower;
               this->upper = upper;
               }

// Operation sample:
// -----------------
int RandInt::sample()
               {
               int span = upper -lower;
  return (rand() % span + lower);
               }

// Operation setLowerUpper:
// ------------------------
void RandInt::setLowerUpper(int lower, int upper)
               {
               this->lower = lower;
               this->upper = upper;
               }
```



Blocktyp DeamonGame
Name = Dateiname

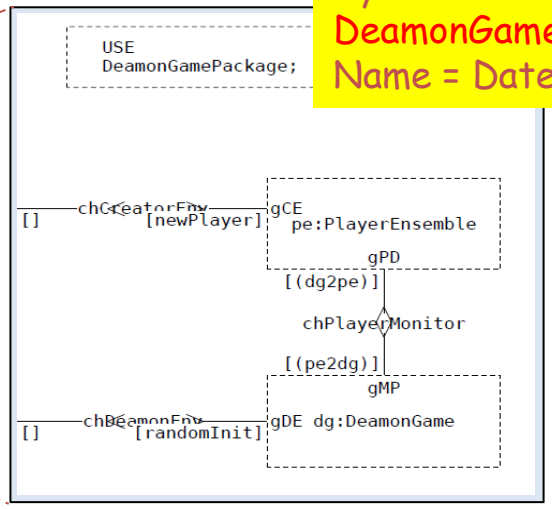


Blocktyp PlayerEnsemble
Name = Dateiname

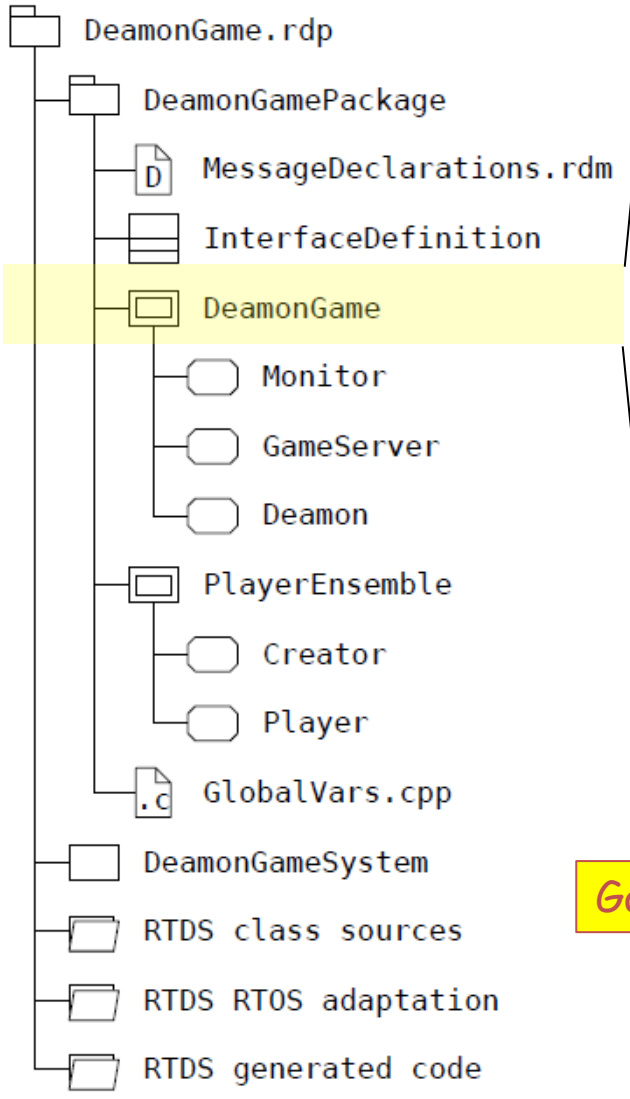


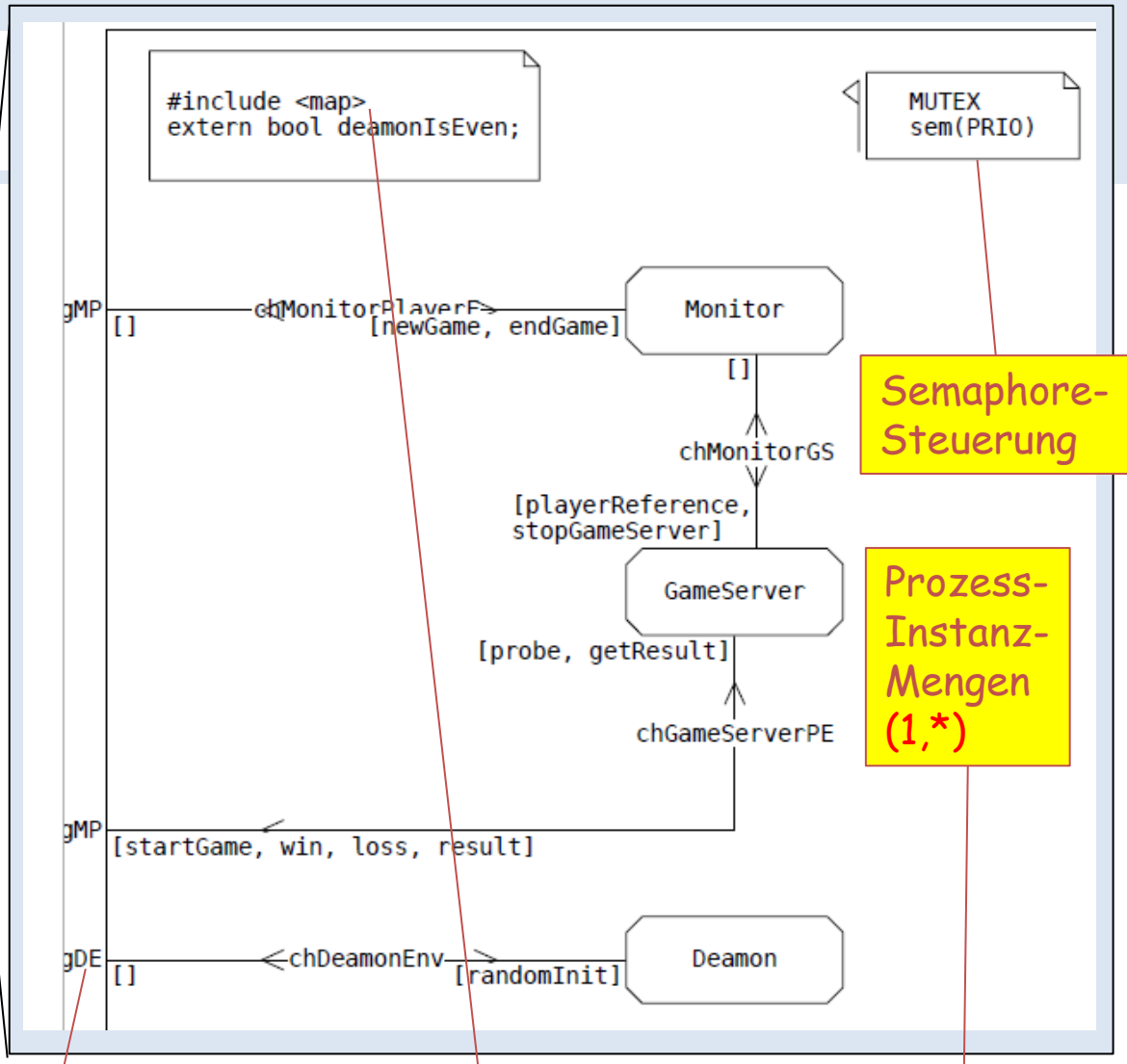

```
bool deamonIsEven = true;
```

C/C++ Erweiterung
globale Variable



System
DeamonGameSystem
Name = Dateiname





Semaphore-Steuerung

Prozess-Instanz-Mengen (1,*)

Gate-Angabe

C++ StandardTemplateLibrary

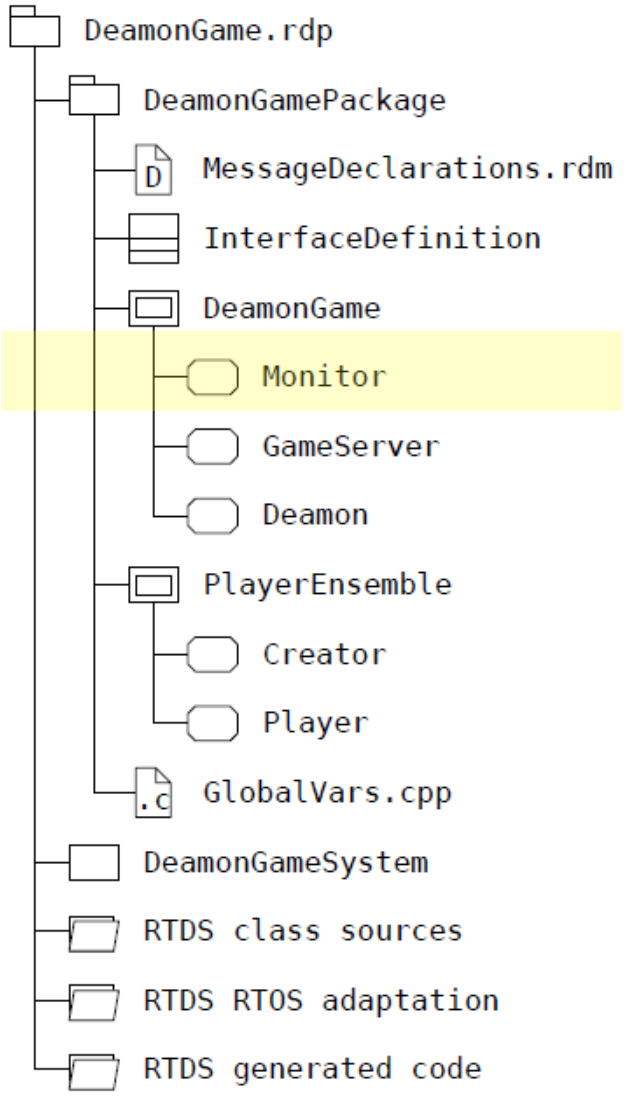
besser:
Monitor(1,1)
GameServer(0,*)
Deamon(1,1)

C++ Standard Template Library

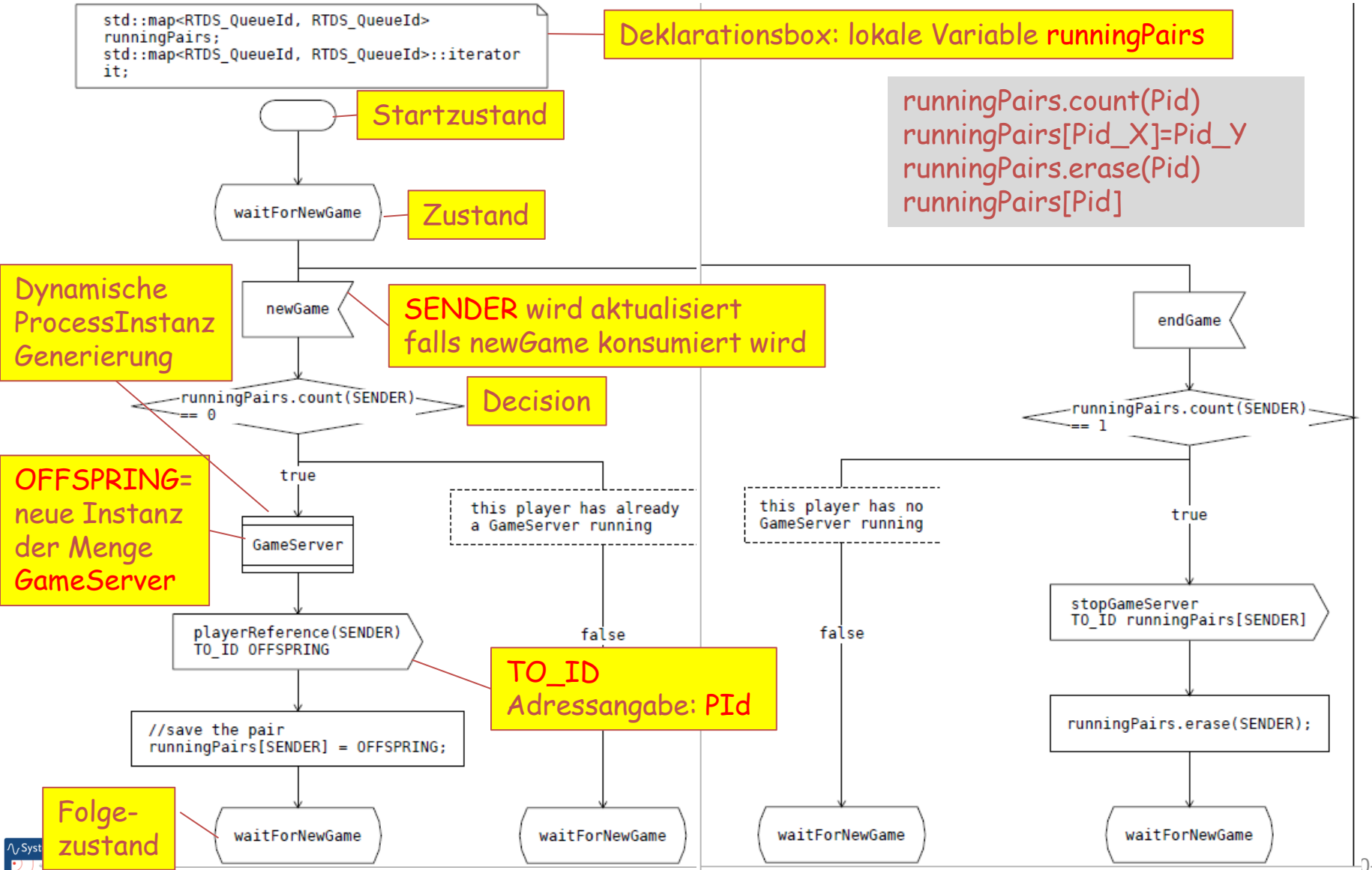
C++ Maps are sorted associative containers
that contain unique key/value pairs.
Maps are sorted by their keys.

Map Constructors & Destructors	default methods to allocate, copy, and deallocate maps
Map operators	assign, compare, and access elements of a map
Map typedefs	typedefs of a map
begin	returns an iterator to the beginning of the map
clear	removes all elements from the map
count	returns the number of elements matching a certain key
empty	true if the map has no elements
end	returns an iterator just past the last element of a map
equal_range	returns iterators to the first and just past the last elements matching a specific key
erase	removes elements from a map
find	returns an iterator to specific elements
…	

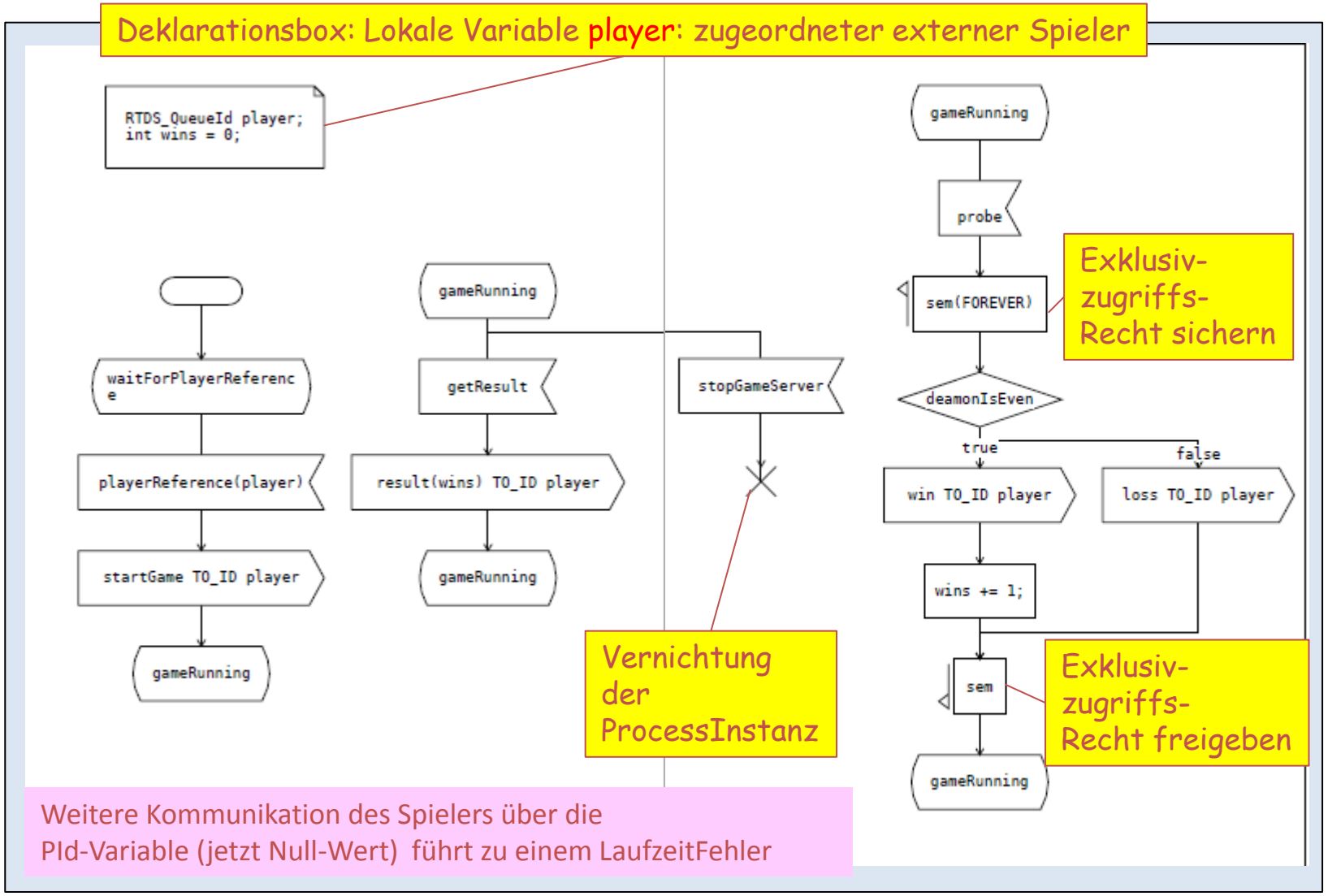
Package


DeamonGame.rdp
DeamonGamePackage
MessageDeclarations.rdm
InterfaceDefinition
DeamonGame
Monitor
GameServer
Deamon
PlayerEnsemble
Creator
Player
GlobalVars.cpp
DeamonGameSystem
RTDS class sources
RTDS RTOS adaptation
RTDS generated code

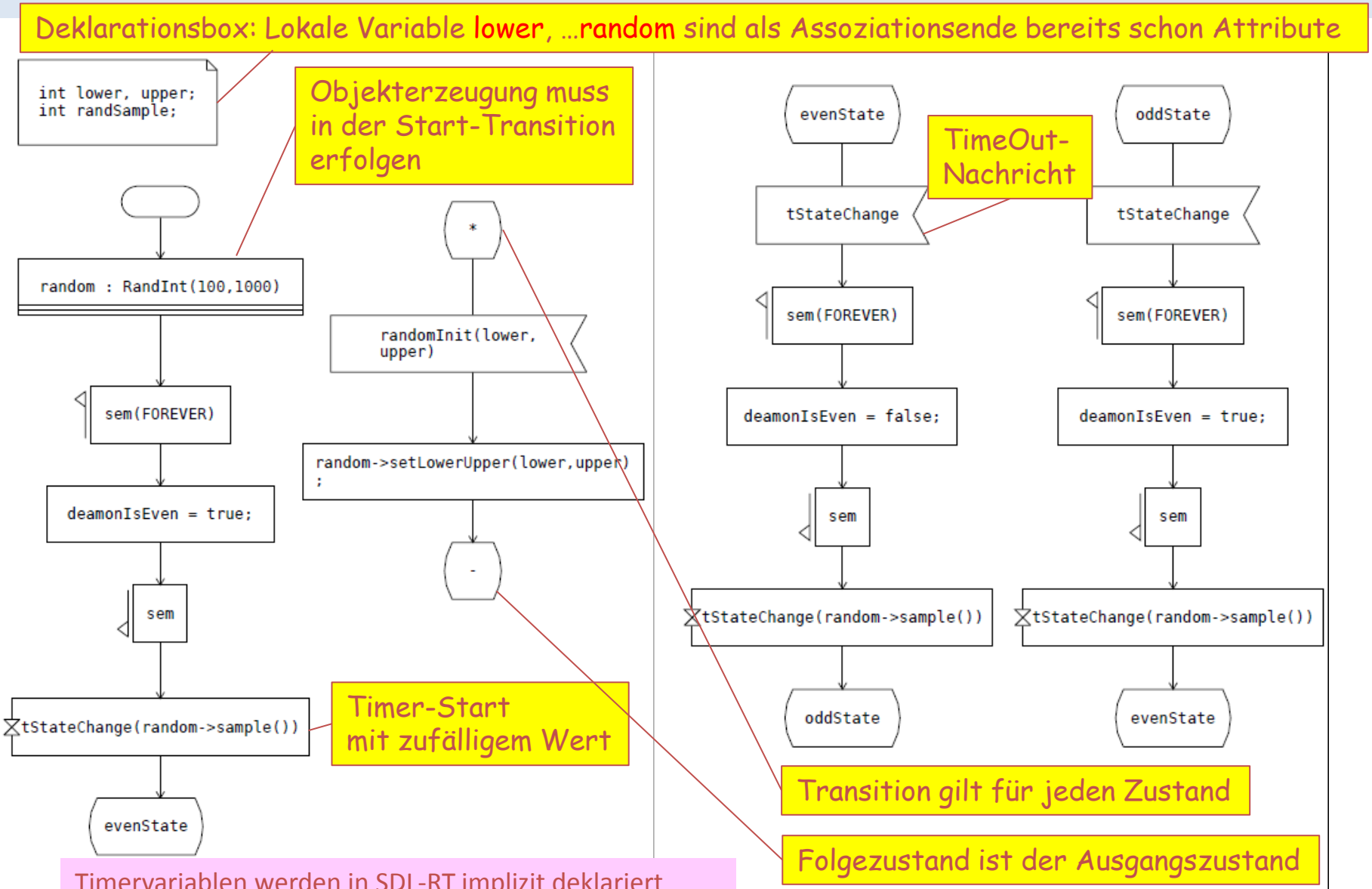
Prototypinstanz der Processmenge Monitor



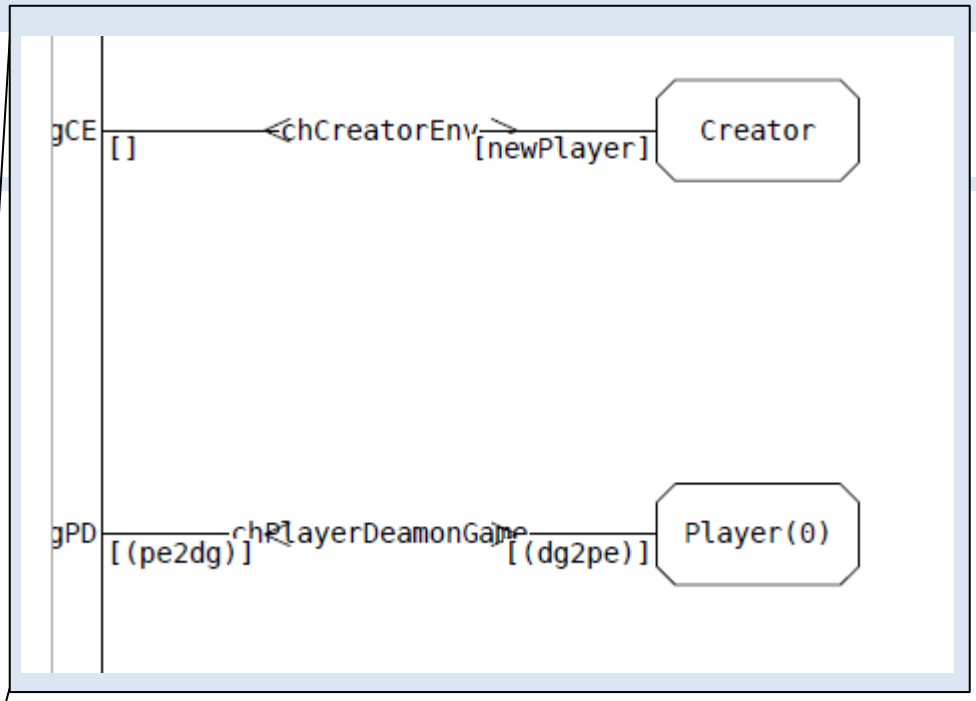
Prototypinstanz der Processmenge GameServer



Prototypinstanz der Processmenge Deamon



Timervariablen werden in SDL-RT implizit deklariert

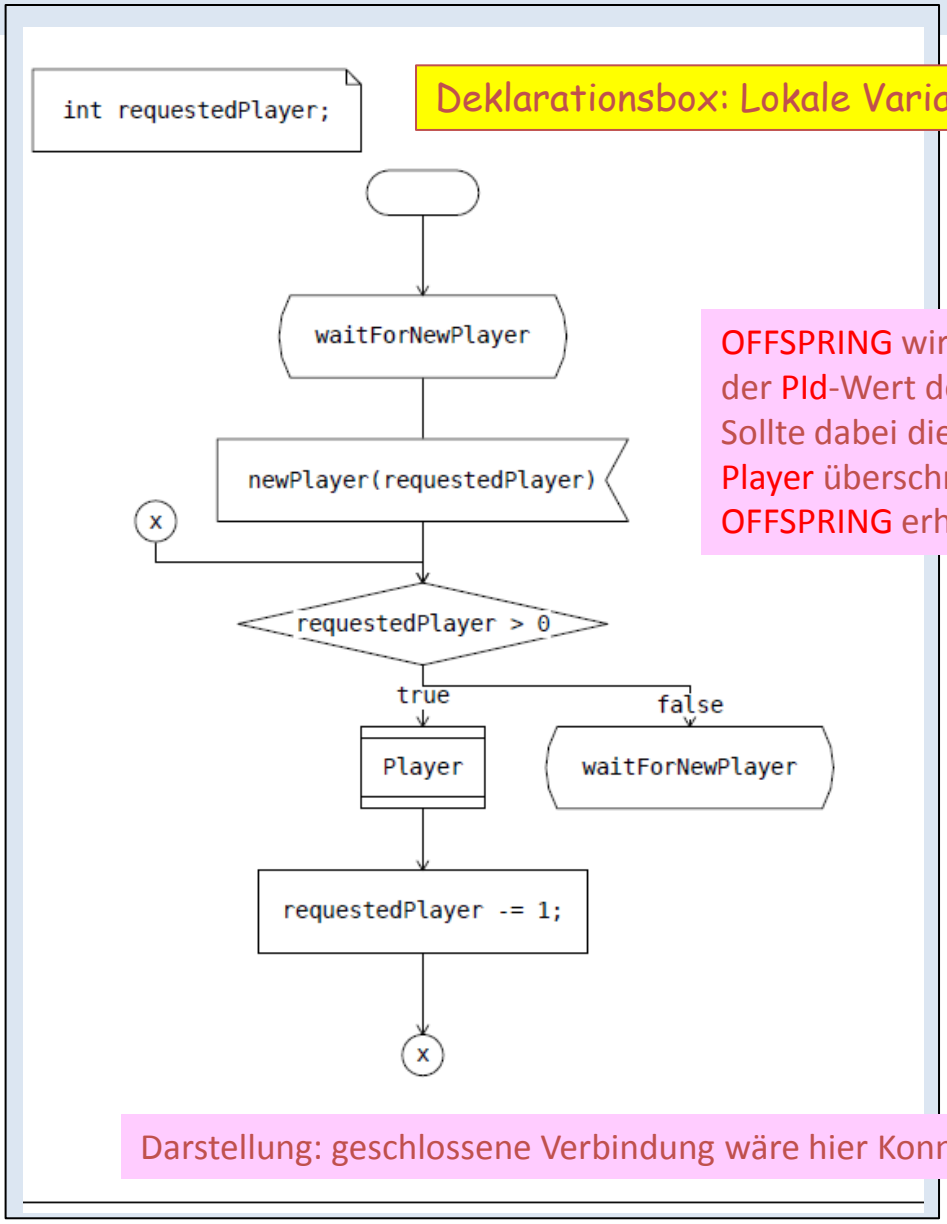


- DeamonGame.rdp
  - DeamonGamePackage
    - MessageDeclarations.rdm
    - InterfaceDefinition
    - DeamonGame
      - Monitor
      - GameServer
      - Deamon
    - PlayerEnsemble
      - Creator
      - Player
    - GlobalVars.cpp
  - DeamonGameSystem
  - RTDS class sources
  - RTDS RTOS adaptation
  - RTDS generated code

```
bool deamonIsEven = true;
```

Verbindung zur Semaphore-Variablen ist nutzerseitig zu organisieren

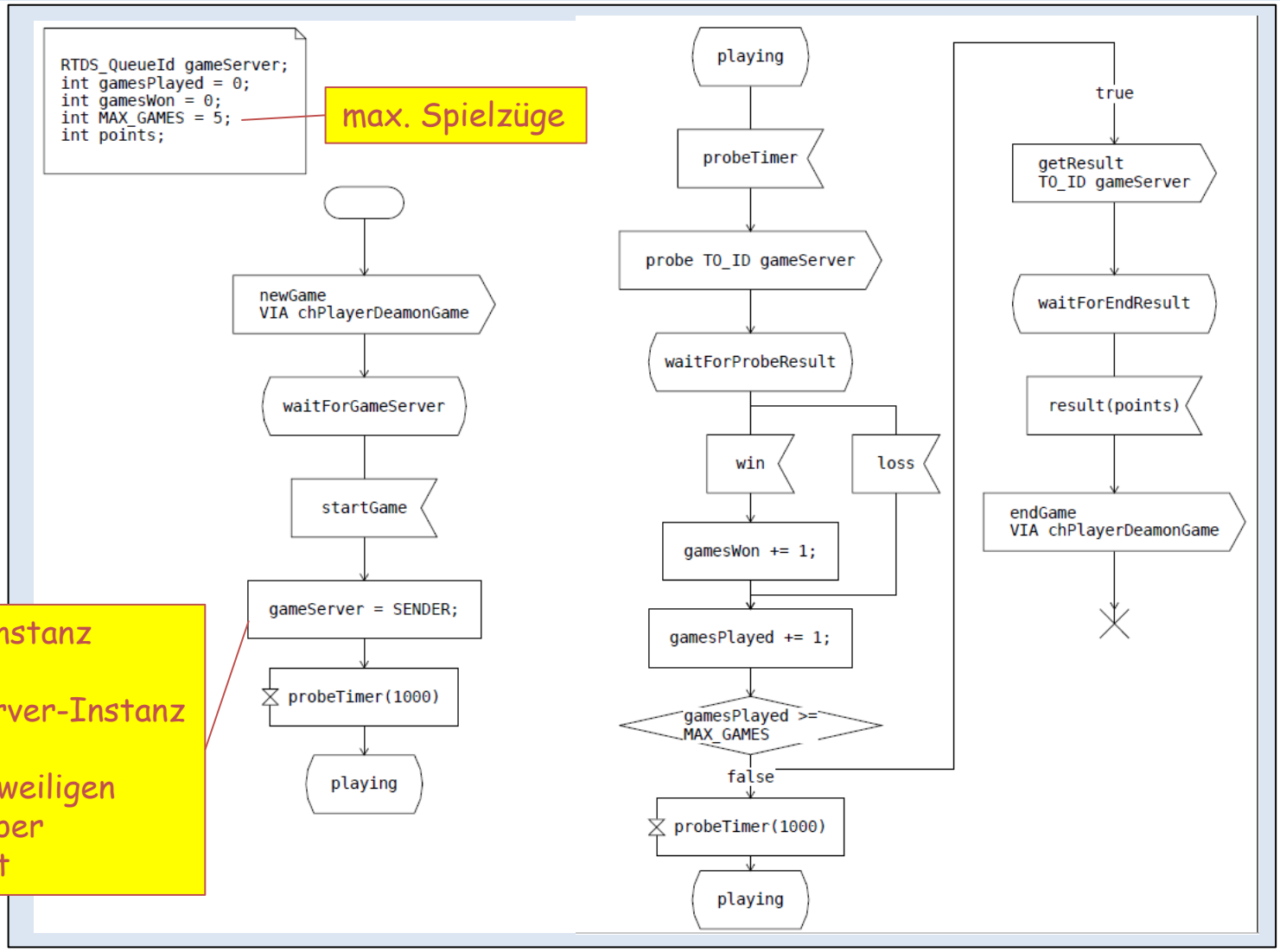
Prototypinstanz der Prozessmenge Creator



OFFSPRING wird bei jeder erfolgreichen Instanziierung der PId-Wert der Instanz zugewiesen
Sollte dabei die maximale Kardinalität der Instnzmenge Player überschritten werden, wird keine Instanz erzeugt, OFFSPRING erhält den Wert Null

Darstellung: geschlossene Verbindung wäre hier Konnektoren vorzuziehen

Prototypinstanz der Prozessmenge Player



Player-Instanz
Und
GameServer-Instanz
Kennen
Ihren jeweiligen
Partner per
Pid-Wert