Conceperea aplicaţiilor în UML
RT ➔ Capsule

1. **Capsule şi clase**
2. **Rolul capsulelor**
3. **Porturi**
4. **Protocoale**
5. **Cardinalitatea capsulelor**
6. **Acţiuni, mesaje şi evenimente**

1. Capsule şi clase

Capsulele reprezintă fluxurile independente de control dintr-un sistem.

Capsulele pot fi parte în relaţiile de:

- dependenţă
- generalizare şi
- asociere

O capsulă poate avea atribute definite în structura sa. Atributele sunt proprietăţi ale clasei capsulei care se regăsesc în toate instanţele ei.

Capsule Structure Diagram este o diagramă pentru specificare şi nu una de interacţiune. Descrie colaborări.

Capsulele sunt şabloane care conţine obiecte de tip fire de execuţie (light weight concurrent objects)

Capsulele
- reprezintă concurenţa direct în model.
- sunt o formă specială de clase
- pot trata simultan activităţi fără implicarea supraîncărcării sistemului de operare

a. Operaţii

Clasele au operaţii publice, protejate sau private.
Capsulele au operaţii numai private. Operaţiile pot fi invocate numai din interiorul capsulei.
Capsulele au porturi publice.
Clasele sau instanţele lor pot invoca operaţii (publice) din alte clase (sau din instanţele lor).
Capsulele nu pot invoca direct operaţii din alte capsule.
Capsulele pot să trimită (una alteia) mesaje sau să semnalizeze evenimente.
Capsulele pot comunica numai prin porturi.

b. Atribute publice

Clasele au atribute publice, protejate sau private. Atributele publice pot fi accesate din exteriorul clasei (sau instanţei).

Capsulele au numai atribute private. Atributele capsulelor nu pot fi accesate din exteriorul lor.

Dacă în protocoale sunt prevăzute metode de transmitere ale unor valori sau de citire de valori, ele pot fi apelate prin intermediul porturilor.

c. Comportament

Comportamentul claselor este realizat de ***metode***. Când este invocată o metodă dintr-o clasă (sau instanţă), ea se execută aşa cum este implementată ca o continuare a metodei care o invocă.

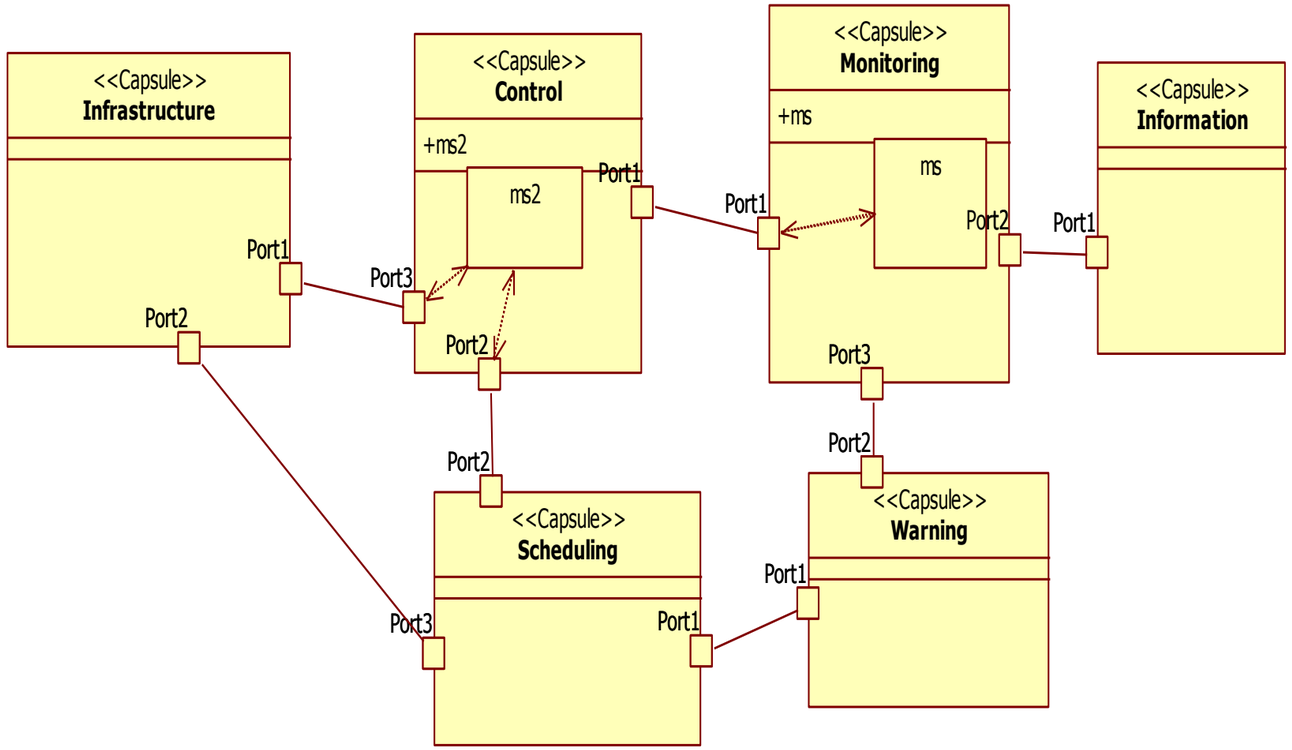
Comportamentul capsulelor este realizate de către ***maşinile de stare***. O capsulă semnalizează un eveniment altei capsule sau transmite un mesaj prin intermediul porturilor.

Maşina de stare din capsulă preia mesajul când este în starea corespunzătoare şi reacţionează conform cu comportamentul implementat în maşina de stare.

O maşină de stare poate invoca o metodă a unui port pentru transmite ceva altei capsule.

2. Rolul capsulelor

Capsulele trebuie să colaboreze pentru realizarea specificaţiilor unei aplicaţii.
Specificaţiile aplicaţiei sunt realizate de structura de capsule.



Rolul unei capsule reprezintă (corespunde la) o specificaţie a unui tip de capsule care ocupă o poziţie particulară în colaborările capsulelor sau în structura de capsule.

O descriere a unei aplicaţii constă dintr-o reţea de roluri de capsule aflate în colaborare şi care sunt reunite prin conectori.

Clasificarea rolurilor capsulelor:

- ***fixat*** – Rolurile capsulelor sunt fixate implicit prin a fi create automat când sunt create capsulele care le conţin
- ***opţional*** - Rolurile unor capsule dintr-o structură nu sunt create în acelaşi timp cu structura care le conţine, ci sunt create când este necesară de către maşina de stare a capsulei care o include
- ***plug-in*** – Structura unei capsule poate conţine capsule cu rol *plug-in*. Acestea sunt deţinătoare de roluri de capsule care sunt create dinamic. Se utilizează în situaţia în care nu se cunoaşte anticipat care este obiectul care va juca un anumit rol la un moment dat în timpul execuţiei. Când relaţia dinamică nu mai este necesară, capsula este eliminată din slotul *plug-in*, iar conectorul la ea este eliminat de asemenea.

Cardinalitatea

Cardinalitatea rolurilor capsulelor specifică numărul de copii de acelaşi tip grupate într-un şablon reutilizabil.

Substituibilitatea

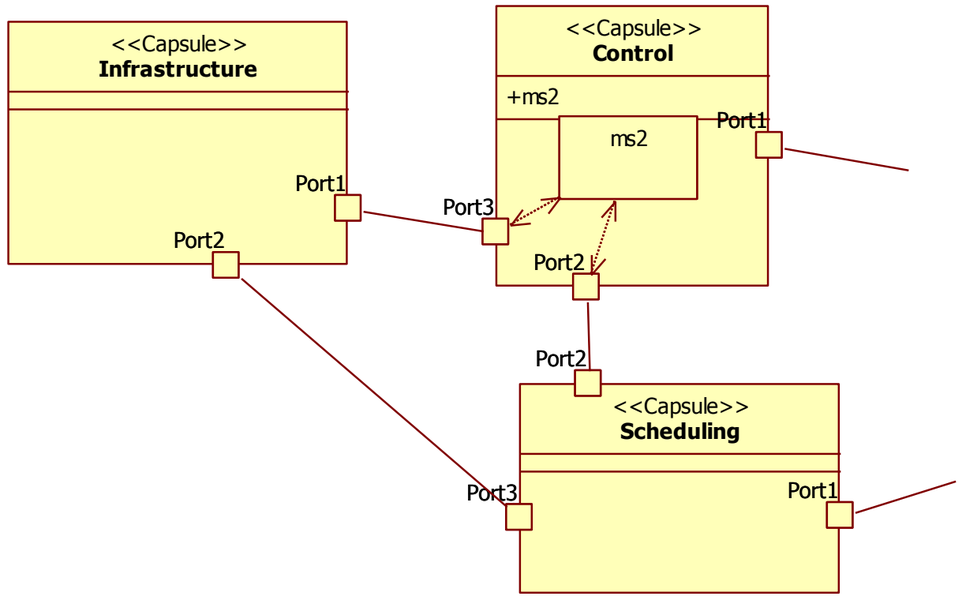
Rolurile capsulelor opţionale şi pug-in pot fi proiectate să fie substituibile.

3. Porturi

Porturile sunt obiecte construite pentru transmiterea mesajelor între instanţele capsulelor.

Porturile:

- sunt deţinute de instanţele capsulelor
- sunt create de capsule la crearea instanţelor lor
- sunt distruse de capsule când instanţele lor sunt distruse

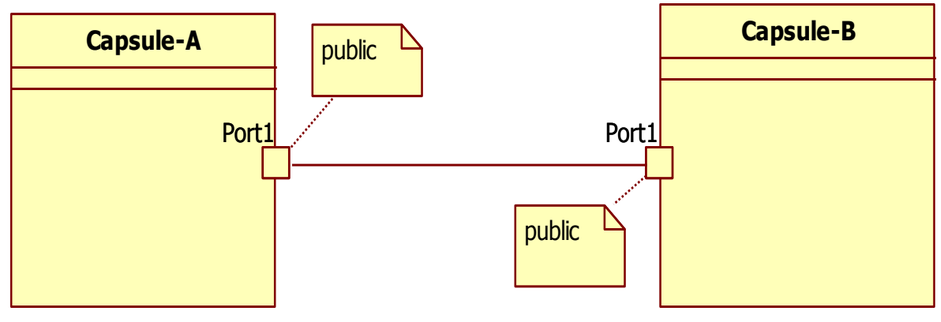


Fiecare port are identitatea sa, diferită de identitatea capsulei.

Portul are starea capsulei care-l conţine.

Prin *protocolul portului* se realizează specificarea setului de mesaje care se pot trimite (***out***) sau recepţiona (***in***) prin port. Acestea sunt asociate cu rolul protocolului. Rolul protocolului defineşte tipul portului.

Reguli de comunicare



Două porturi legate printr-un conector trebuie să fie compatibile.

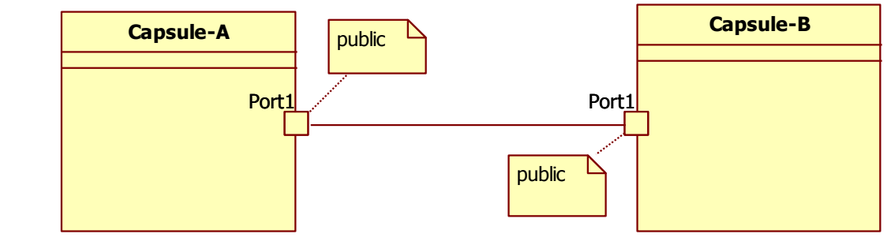
Fiecare semnal din setul *out* al unui port trebuie să fie inclus în setul *in* al celuilalt port.

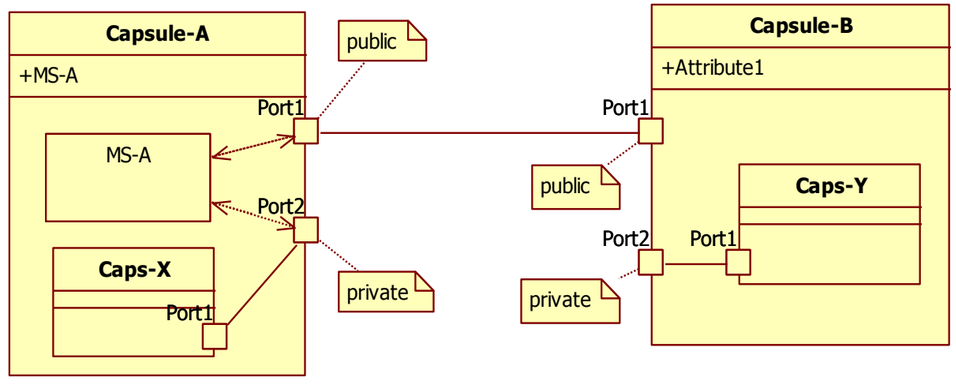
Setul *in* al unui port trebuie să fie un superset al setului *out* al celuilalt port.

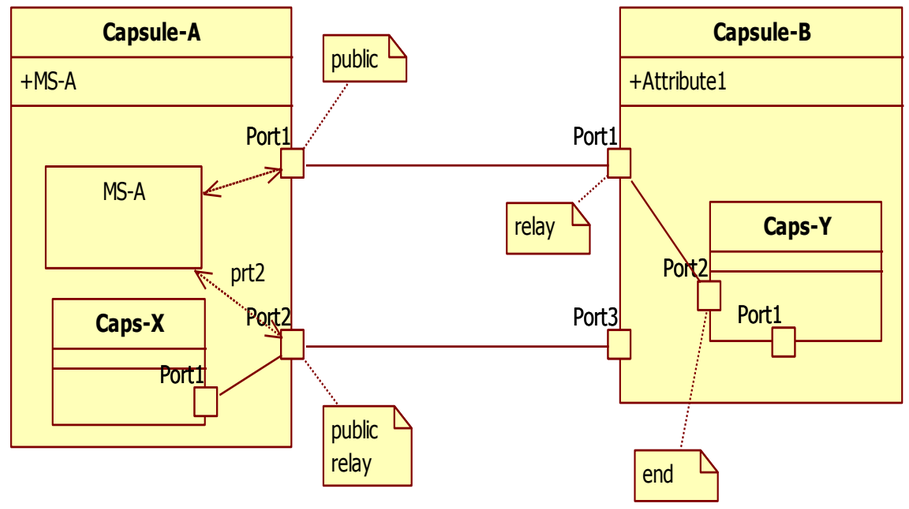
Clasa datelor unui semnal *out* trebuie să fie aceeaşi cu o subclasă a datelor unui semnal *in* din celălalt port.

Clasificarea porturilor:

- d.p.d.v. al vizibilităţii
  - *public*
  - *protected*
- d.p.d.v. al conectorului
  - *wired* – conectate (legate) la alt port pentru a trimite mesaje
  - *non-wired* – care modelează comunicaţii dinamice; aceste conexiuni sunt controlate dinamic. Ex. Modelul *client-server* (clienţii nu sunt cunoscuţi în momentul proiectării)
- d.p.d.v. al terminării
  - *relay* – tip releu – Sunt prin natura lor publice şi legate. Ele transmit direcţionat semnalizări de evenimente la capsule protejate. Dacă nu este conectat la un port destinatar, semnalul se pierde.
  - *end* – poate fi public sau protejat, legat sau nelegat. Mesajele le trimite la destinatarul final care realizează comportamentul capsulei (MS).





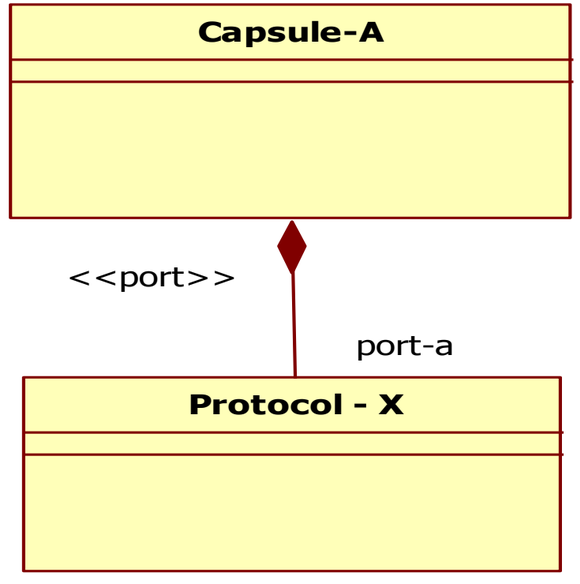


Reprezentări grafice

Reprezentarea claselor

În diagrama claselor se adaugă un compartiment pentru porturi. Stereotipul <<*port*>> arată relaţia dintre capsulă şi protocol. Portul apare ca un obiect ce implementează protocolul.

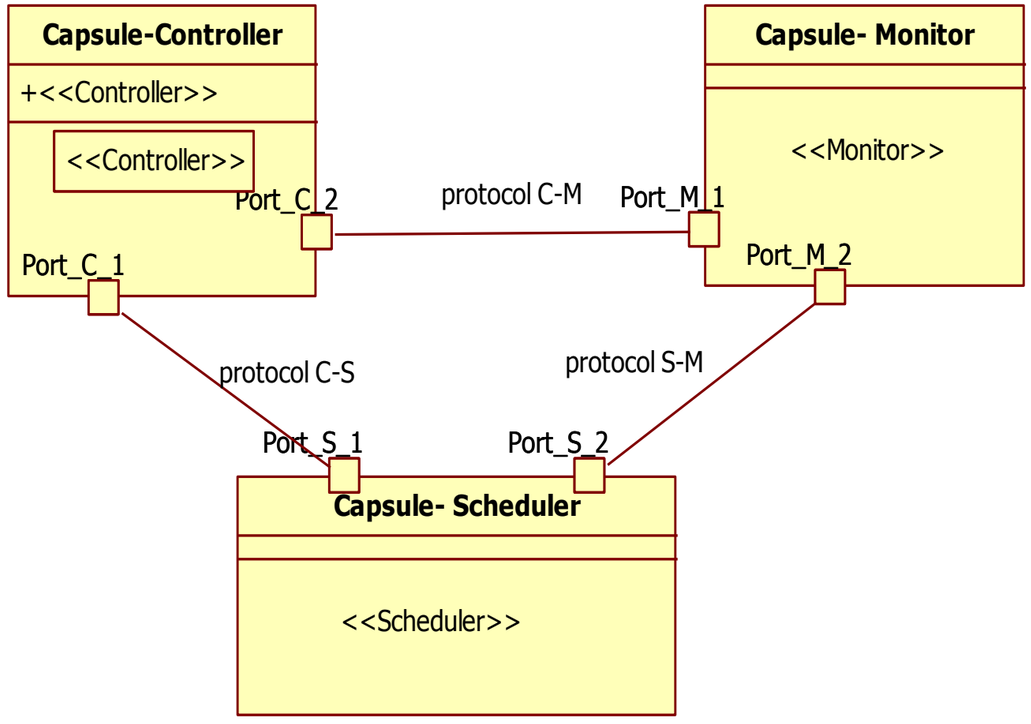
Protocol-X este o clasă care implementează metodele protocolului.



Reprezentarea rolurilor capsulelor

Se reprezintă numai porturile publice. Nu se deosebesc porturile *end* de cele *relay* din punctul de vedere al rolurilor.

Se reprezintă numele porturilor şi rolurile protocoalelor.



Reprezentarea colaborăriilor capsulelor

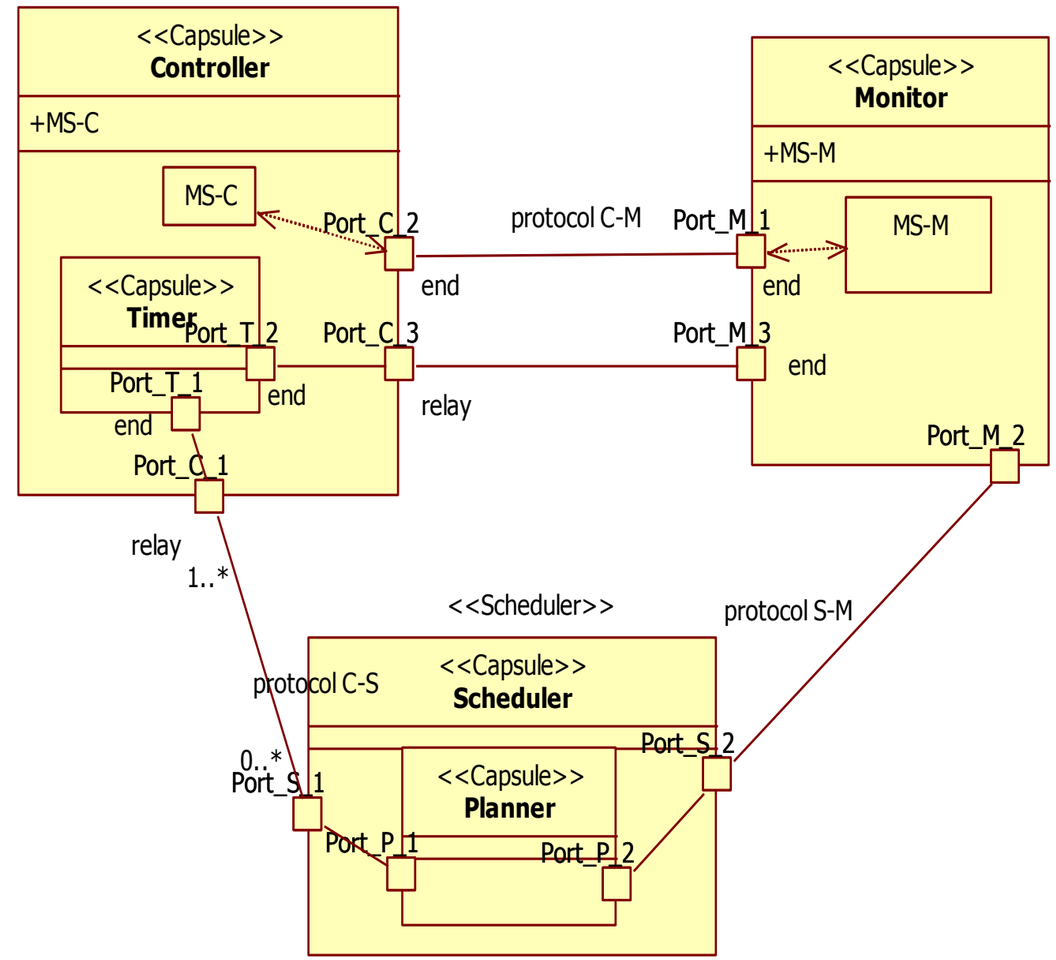
Sunt vizibile toate porturile într-o viziune a structurilor.

Este reprezentată plasarea fizică în containerul capsulelor.

Se diferenţiază porturile publice şi private.

Porturile *end* sunt conectate la maşini de stare.

Se marchează porturile legate şi cele ne-legate.



4. Protocoale

Un ***protocol*** constă dintr-un ***şablon*** de comunicare care precizează setul de mesaje schimbat.

Se definesc ***tipurile*** mesajelor schimbate, setul participanţilor şi ***rolul*** lor specific pe care îl au în protocol.

Rolul protocolului are un nume unic şi specifică:

- setul mesajelor recepţionate de rol (eventual vid)
- setul mesajelor transmise de rol (eventual vid)

Opţiune: protocolul poate specifica secvenţe de mesaje valide prin:

- maşini de stare
- diagrame de secvenţe

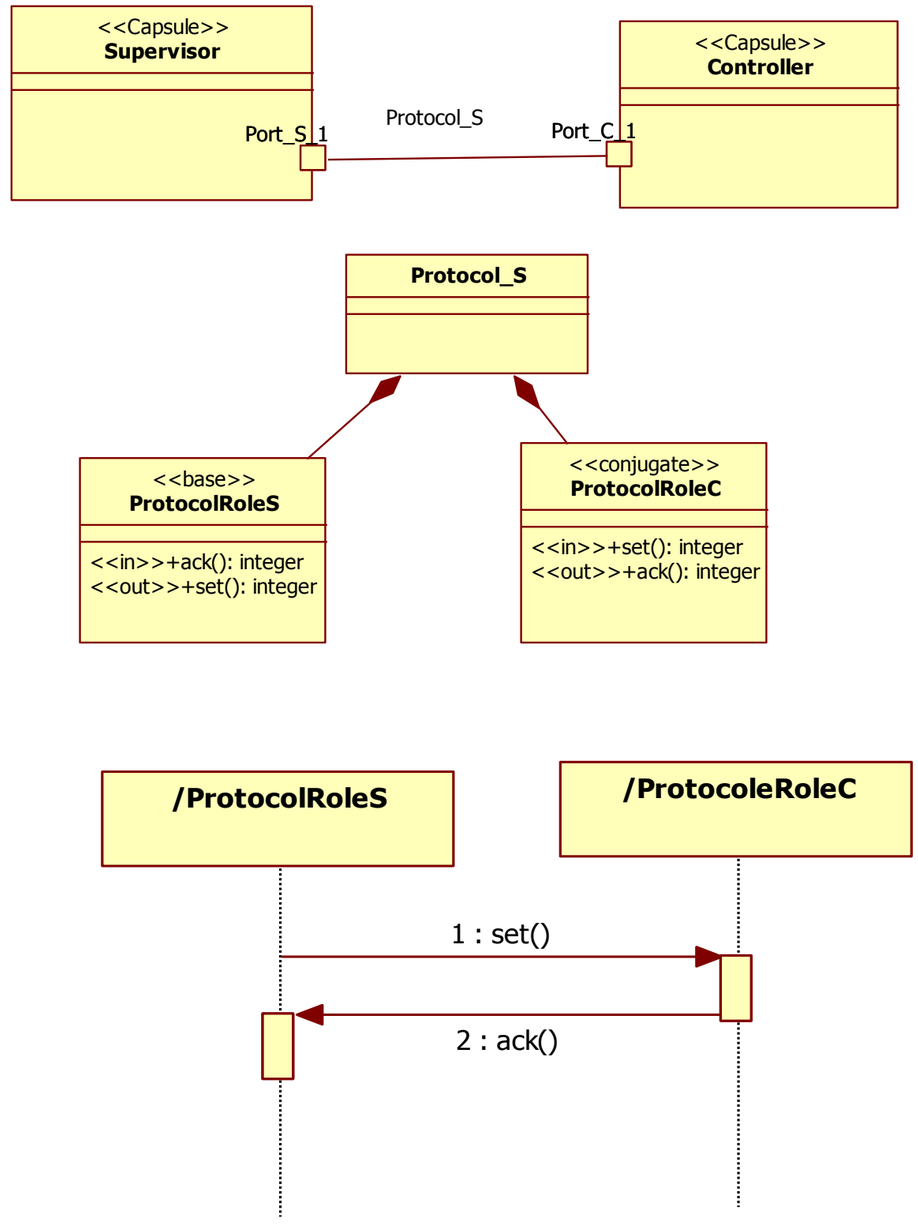
Un protocol este compus din roluri de protocol diferite.

Protocoale binare

Acestea implică doi şi numai doi participanţi.

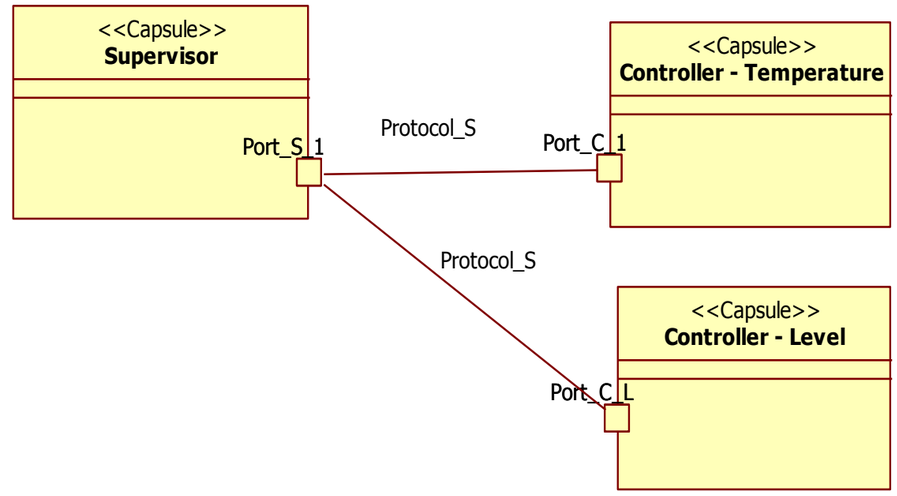
În protocoalele binare se definesc explicit rolurile conjugate ale celor doi participanţi.

Porturile joacă rolul unuia dintre participanţi. Tipul unui port este specificat de rolul protocolului şi nu de protocol.
Ex.:
Tipul portului Port_S_1 este: ProtocolRoleS
Tipul portului Port_C_1 este: ProtocolRoleC



5. Cardinalitatea capsulelor

Cardinalitatea este un atribut al rolului capsulei şi nu a clasei capsulei.

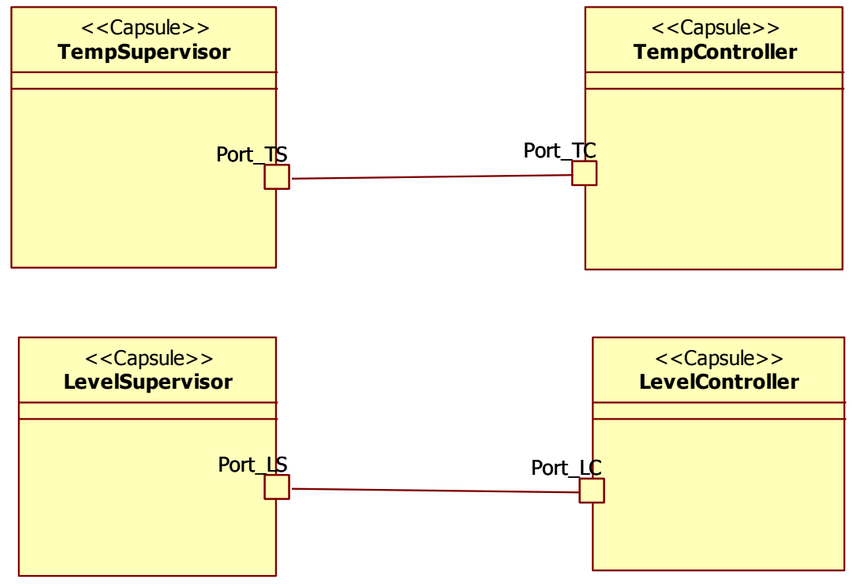


Structură stea (star)

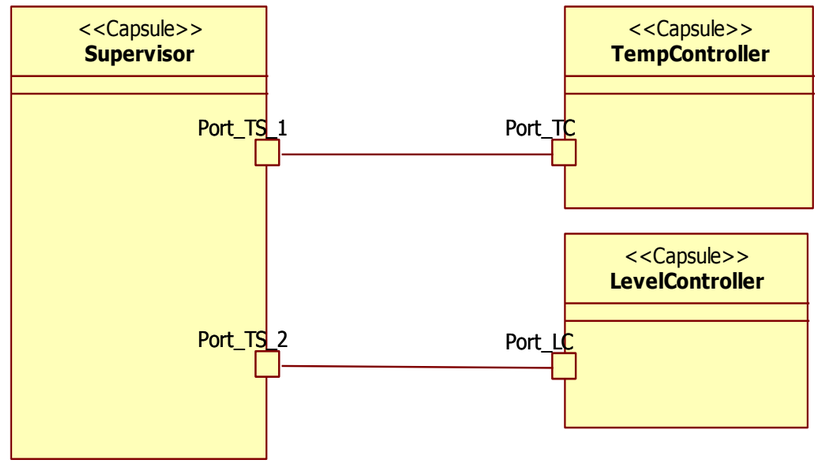
Ex.: supervizorul este legat la două sau mai multe controllere la care le setează referinţele.

Structură tablou (array)

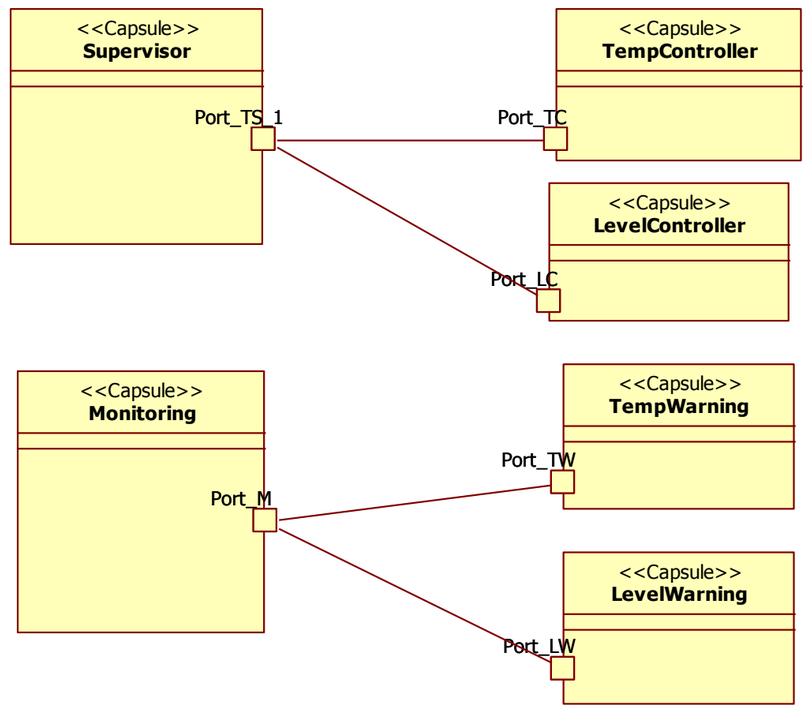
Perechi de baze si conjugate sunt legate între ele.



Structură Bus



Structură combinată



Substituirea (substituibilitatea)

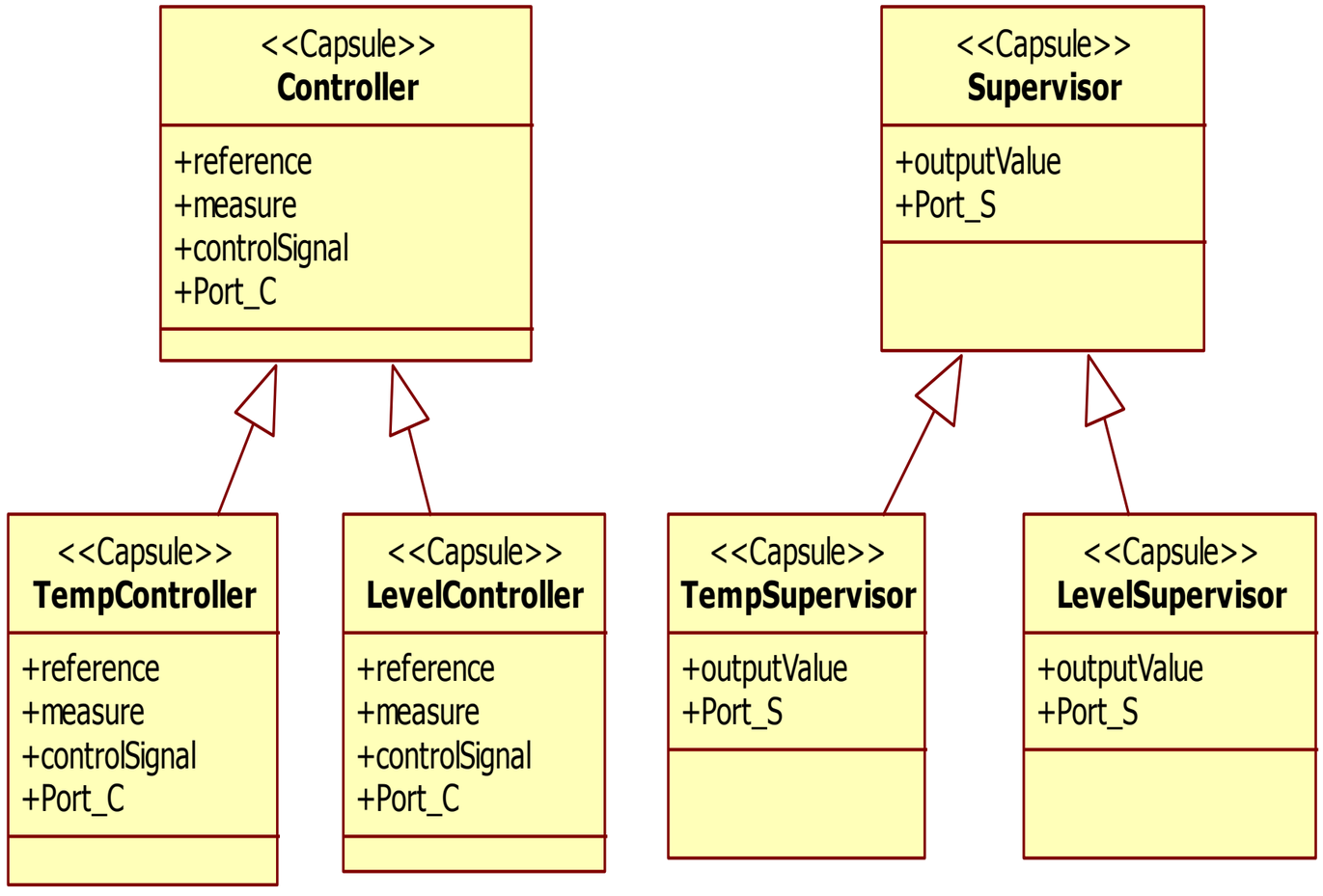
Substituirea este proprietatea rolului unei capsule opţionale sau plug-in de a permite rolului capsulei să fie instanţiat de către o instanţă clasei capsulei reprezentată prin rolul capsulei sau de către o altă clasă a unei capsule compatibile.

Două capsule sunt compatibile dacă au aceeaşi interfaţă sau sunt subclase ale aceleiaşi superclasă.

Interfaţa capsulei se defineşte prin proturile sale publice. Deci, capsulele compatibile au porturi publice.

O subclasă este compatibililă cu superclasa sa dacă are porturi compatibile care se potrivesc cu fiecare interfaţă conectată a referinţelor superclasei.

Port_S sau Port_C din subclasă trebuie să implementeze ***acelaşi protocol*** ca şi Port_S sau Port_C din superclasă pentru ca să fie capsulele compatibile.



6. Acţiuni, mesaje şi evenimente

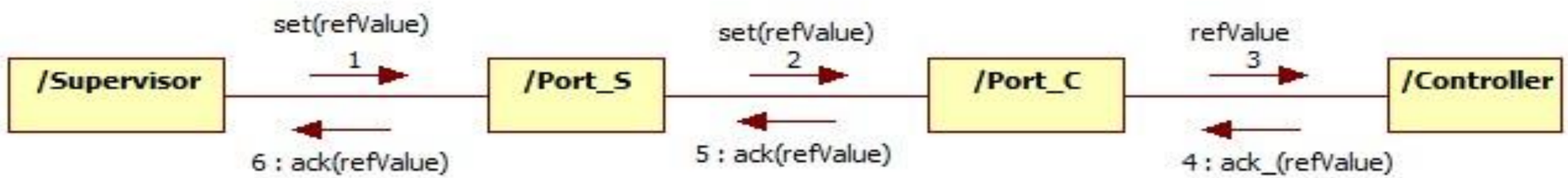


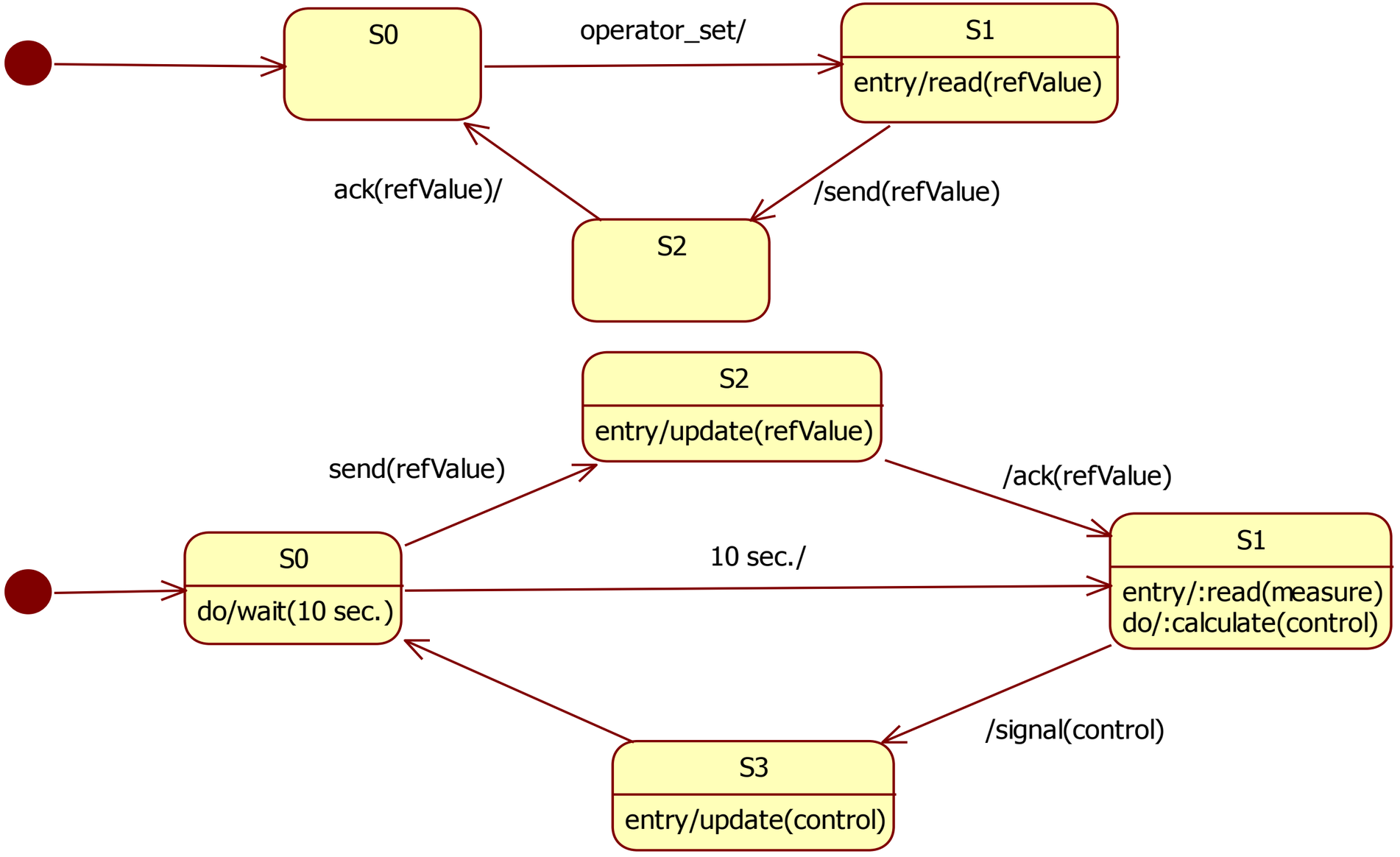
Diagrama de colaborare dintre obiecte (instanţele claselor marcate pe obiecte). Este diagrama de colaborare dintre roluri (nu dintre instanţe = obiecte).

UML Object notation:
ObjectName'/'ClassifierRoleName ':' ClassifierName['','ClassifierName]
Combinations:
:C – unnamed instance from a class C
/R – unnamed instance playing the role R
/R:C - unnamed instance from a class C playing the role R
O/R – instance named O playing the role R
O/R:C – instance named O from a class C playing the role R
O – instance named O
/ - unnamed instance, unnamed class, unnamed role

Comunicaţie:
- sincronă
- asincronă

Obiectele comunică prin:

- ***mesaje*** – Un mesaj este o specificaţie a unui obiect care va fi comunicat între două obiecte. Un semnal este un tip special de mesaj care este transmis asincron. Mesajul şi instanţa sa sunt două lucruri diferite deşi se folosesc interschimbabil. Mesajul este echivalentul referinţei, iar instanţa este conţinutul mesajului.
- ***acţiuni*** – O acţiune reprezintă transmiterea sau distribuirea unui mesaj de către transmiţător. Tipul acţiunii specific tipul cererii:
  - *invocarea unei operaţii*
  - *semnalizarea unui eveniment*
- ***evenimente*** – Un eveniment reprezintă recepţionarea unui mesaj de către receptor (receiver). La recepţionarea lui, receptorul construieşte un eveniment (instanţiat ca un obiect). Evenimentul poate fi:
  - *call event* (eveniment de apel) ➔se aplează o metodă
  - *signal event* (eveniment de semnal) ➔se semnalizează producerea unui eveniment



Supervisor + Controller State Machines

Guards

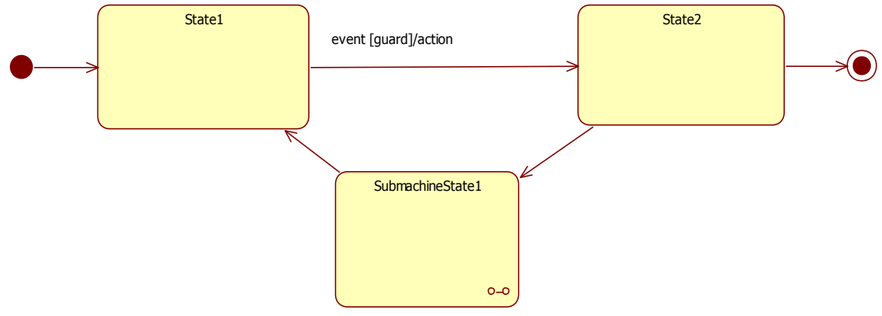


Fig. 1.x. Example of State Machine Diagram.

A guard is a condition that must be fulfilled (true) for the transition to be triggered and are optionally represented.

Representation: ***trigger [guard]effect***

or ***trigger [guard]/list of methods***

trigger – the cause of the transition

guard - logic expression

- free language expression

effect – operation(s) or action executed as effect of the transition – it is automatically invoked by object when the transition occurs.

a. Acţiuni

Acţiunile sunt secvenţe de calcule (descrise prin secvenţe de instrucţiuni) care sunt executate atomic şi care sunt încorporate într-o maşină de stare. Sosirea unui eveniment prioritar celui la care reacţionează componenta curentă nu poate întrerupe acţiunea în curs de desfăşurare. Se va reacţiona la noul eveniment sosit numai dupăterminarea completă a acţiunii.

Acţiunile pot fi:

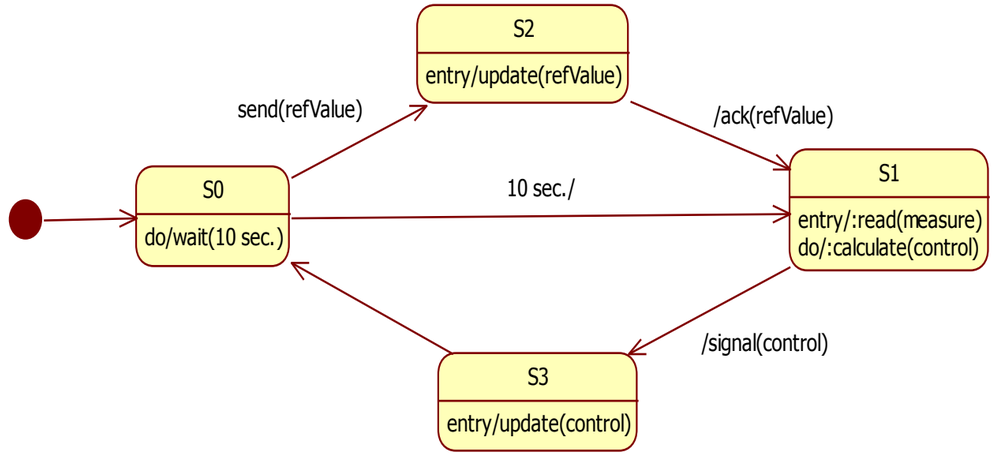
- apel de operaţii
- crearea unor obiecte
- distrugerea unor obiecte
- transmiterea de mesaje

O ***acţiune*** este ataşată într-o maşină de stare la o:

- tranziţie
- stare ca
  - acţiune *entry*
  - acţiune *exit*

Acţiuni blocante

Invocarea unei operaţii de apel este o comunicare de tip sincron. Ea blochează întrega maşină de stare a apelantului. Aceasta nu poate prelua un nou mesaj până când nu s-a terminat acţiunea anterioară.



Ex.: *send(refValue)* dacă s-a recepţionat acest apel, se va termina înainte de a executa altă tranziţie.

b. Evenimente de apel (Call events)

Un eveniment de apel constă dintr-o cerere de invocare sincronă a unei operaţii precizate.
Maşina de stare poate accepta cererea dacă se află într-o stare corespunzătoare.

Comunicaţia este de tip sincron. Când apelantul invocă o operaţie, se transmite controlul apelatului până la terminarea ei. După terminarea operaţiei, controlul revine apelantului (transmiţătorului).

c. Evenimente de semnalizare (Signal events)

Un eveniment de semnalizare constă din recepţionarea asincronă a unui mesaj. Acesta poate fi trimis mai departe la o tranziţie a unei maşini de stare.

Execuţia unei operaţii de asemenea poate transmite un eveniment de semnalizare.

Recepţionarea unui eveniment de semnalizare va determina (uzual) comutarea stării maşinii de stare.

La transmiterea unui eveniment de semnalizare, transmiţătorul nu pierde controlul (ca la transmiterea sincronă) în favoarea destinatarului. Acesta este transmis asincron. Transmiţătorul poate continua cu executarea altor acţiuni, fără a fi blocat ca la comunicaţia sincronă.

*

*****

END

*****

*