Diagramy stanów

Na podstawie

UML 2.0 Tutorial

http://sparxsystems.com.au/resources/uml2_tutorial/

Zofia Kruczkiewicz

Diagramy stanów

1. **Diagramy stanów UML**

   http://sparxsystems.com.au/resources/uml2_tutorial/

2. **Przykład diagramów stanów UML – modelowanie wpływu przypadków użycia na stany obiektu**
3. **Przykład diagramu stanów UML – inżynieria wprost i inżynieria odwrotna**

Dwa rodzaje diagramów UML 2

Diagramy UML modelowania strukturalnego

- ***Diagramy pakietów***
- ***Diagramy klas***
- **Diagramy obiektów**
- **Diagramy mieszane**
- **Diagramy komponentów**
- **Diagramy wdrożenia**

Diagramy UML modelowania zachowania

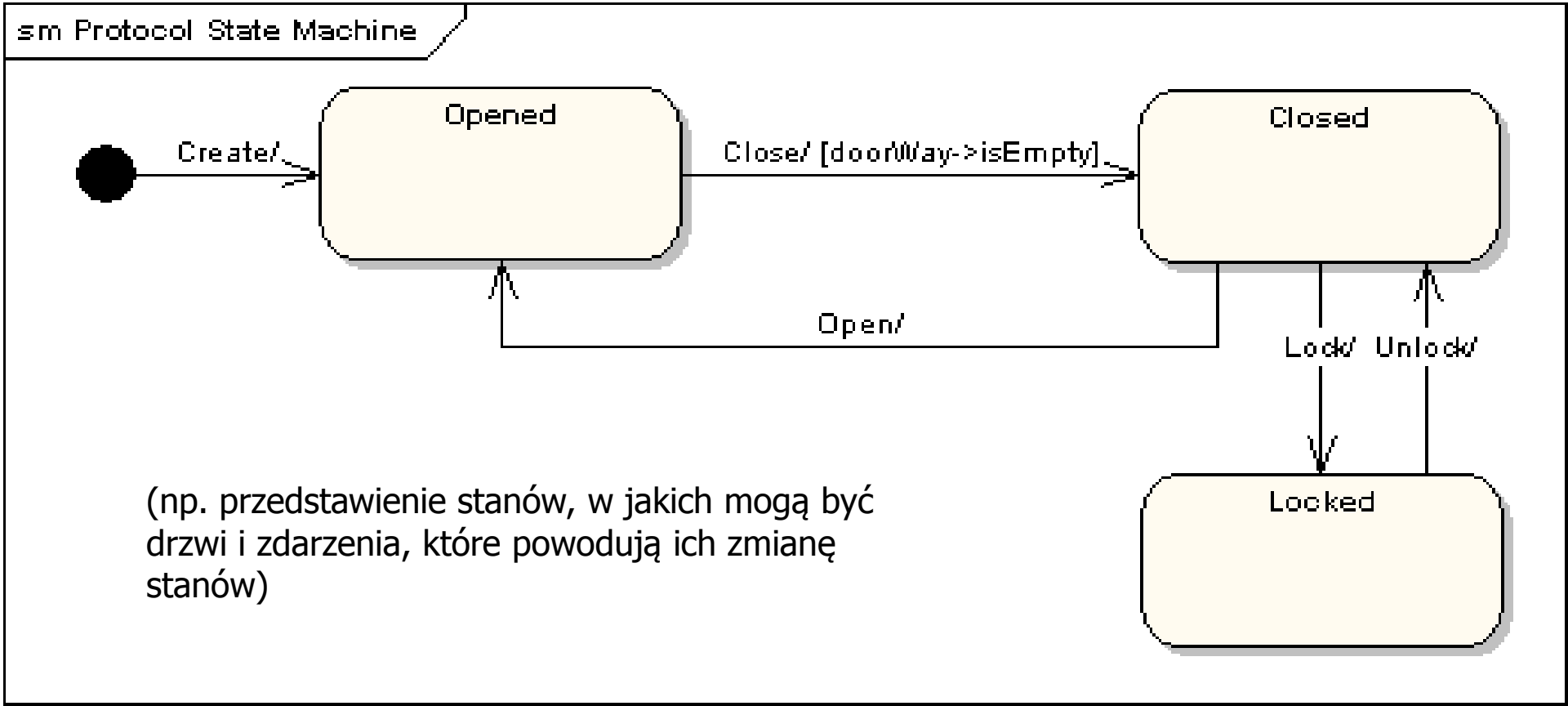
- ***Diagramy przypadków użycia***
- ***Diagramy aktywności***
- ***Diagramy stanów***
- **Diagramy komunikacji**
- ***Diagramy sekwencji***
- **Diagramy czasu**
- **Diagramy interakcji**

Diagramy stanów

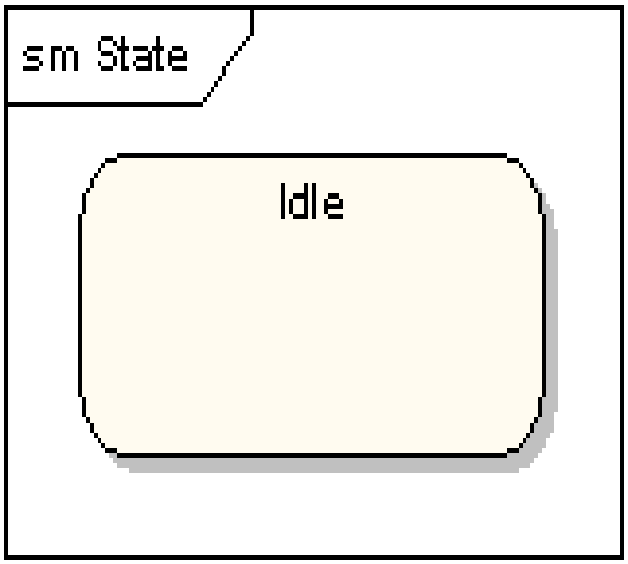
1. **Diagramy stanów UML**
   http://sparxsystems.com.au/resources/uml2_tutorial/

Diagram stanu opisuje zmiany stanu obiektu, podsystemu lub systemu pod wpływem działania operacji - iest szczególnie przydatny, gdy zachowanie obiektu jest złożone. Przedstawia on ***maszynę stanów*** jako przepływ sterowania między stanami.

Diagram stanów jest grafem złożonym z wierzchołków i krawędzi: wierzchołkami są stany (prostokąty o zaokrąglonych rogach), krawędziami są przejścia (strzałki).

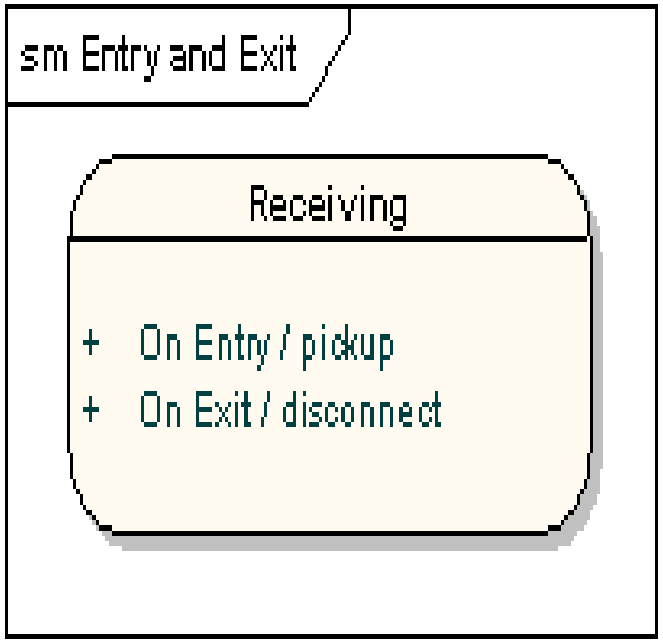


(np. przedstawienie stanów, w jakich mogą być drzwi i zdarzenia, które powodują ich zmianę stanów)



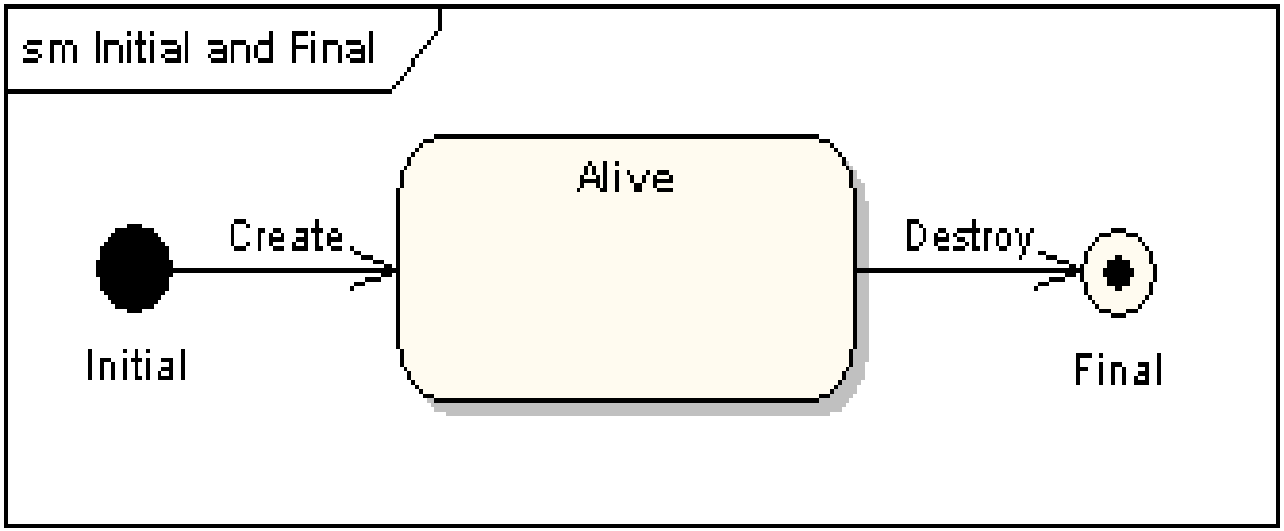
Stan jest okolicznością lub sytuacją, w jakiej znajduje się obiekt

- jest rezultatem poprzedniej aktywności
- spełnia jakiś warunek
- jest określony przez wartości własnych atrybutów i powiązań do innych zadań
- wykonuje pewne czynności
- czeka na jakieś zdarzenie



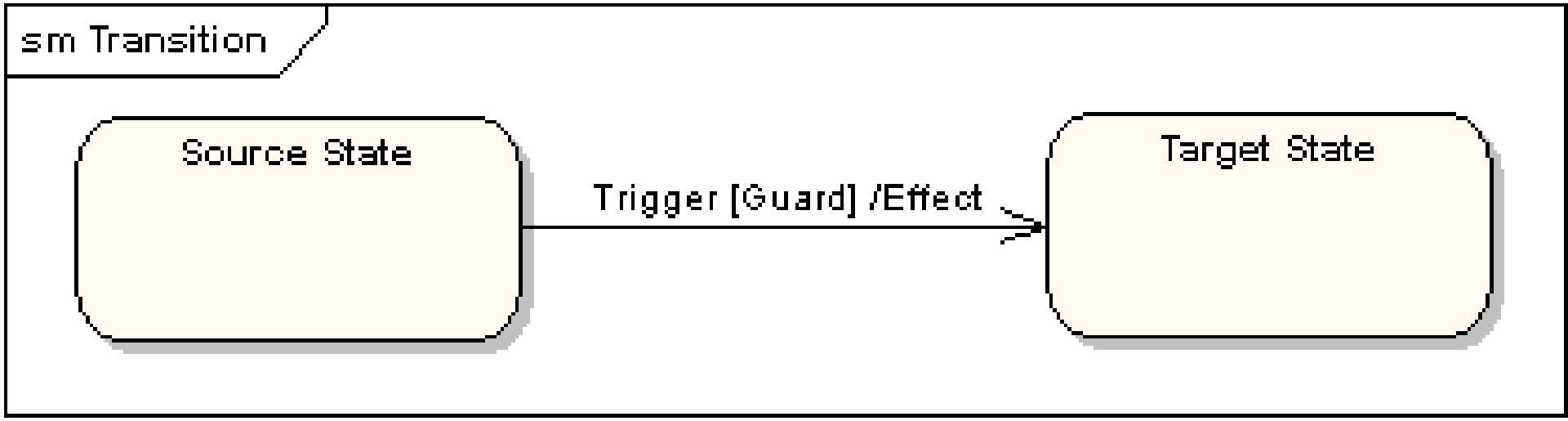
- **Nazwa** -unikatowy ciąg znaków, brak nazwy dla stanu anonimowego

- **Akcje wejściowe** (entry) i **wyjściowe** (exit) - akcje wykonywane odpowiednio przy wejściu do stanu i przy wyjściu)



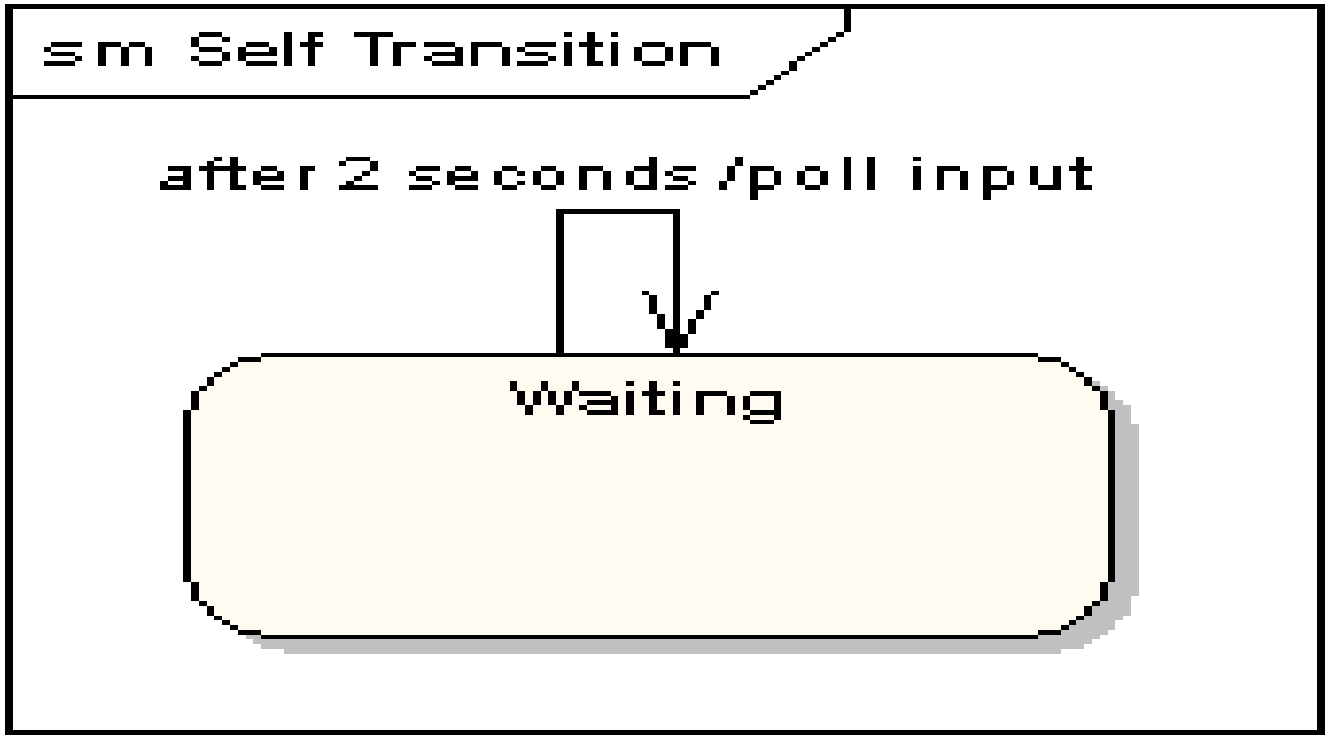
Stan początkowy (Initial) – może być tylko jeden stan początkowy

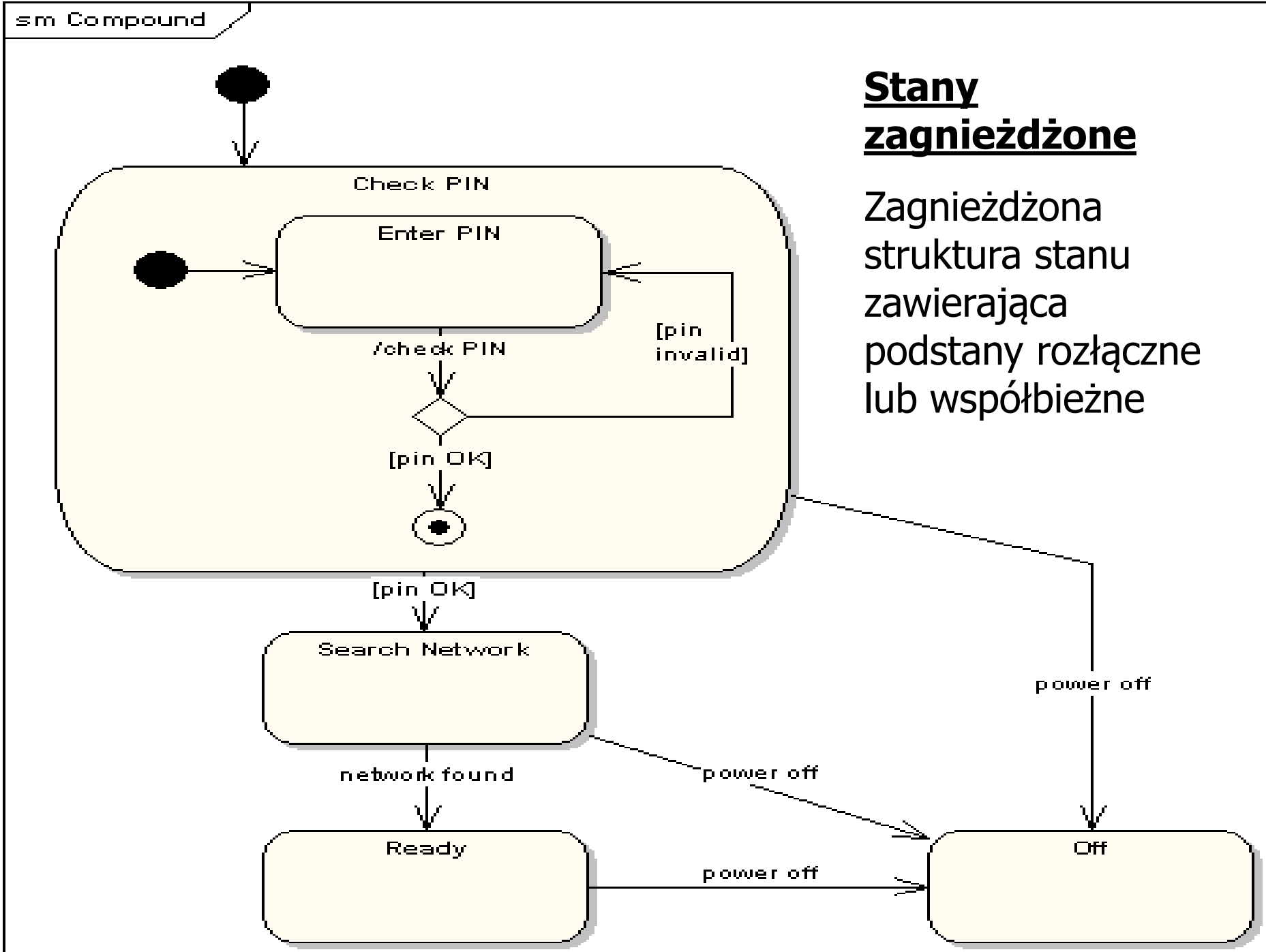
Stan końcowy (Final) – może być kilka stanów końcowych



Przejście (transition) jest związkiem między dwoma stanami, który wskazuje, że np. obiekt znajdujący się w pierwszym stanie wykona pewne **akcje (Effect)** i przejdzie do drugiego stanu, ilekroć zaistnieje określone **zdarzenie (Trigger**) i będą spełnione określone **warunki (Guard).**

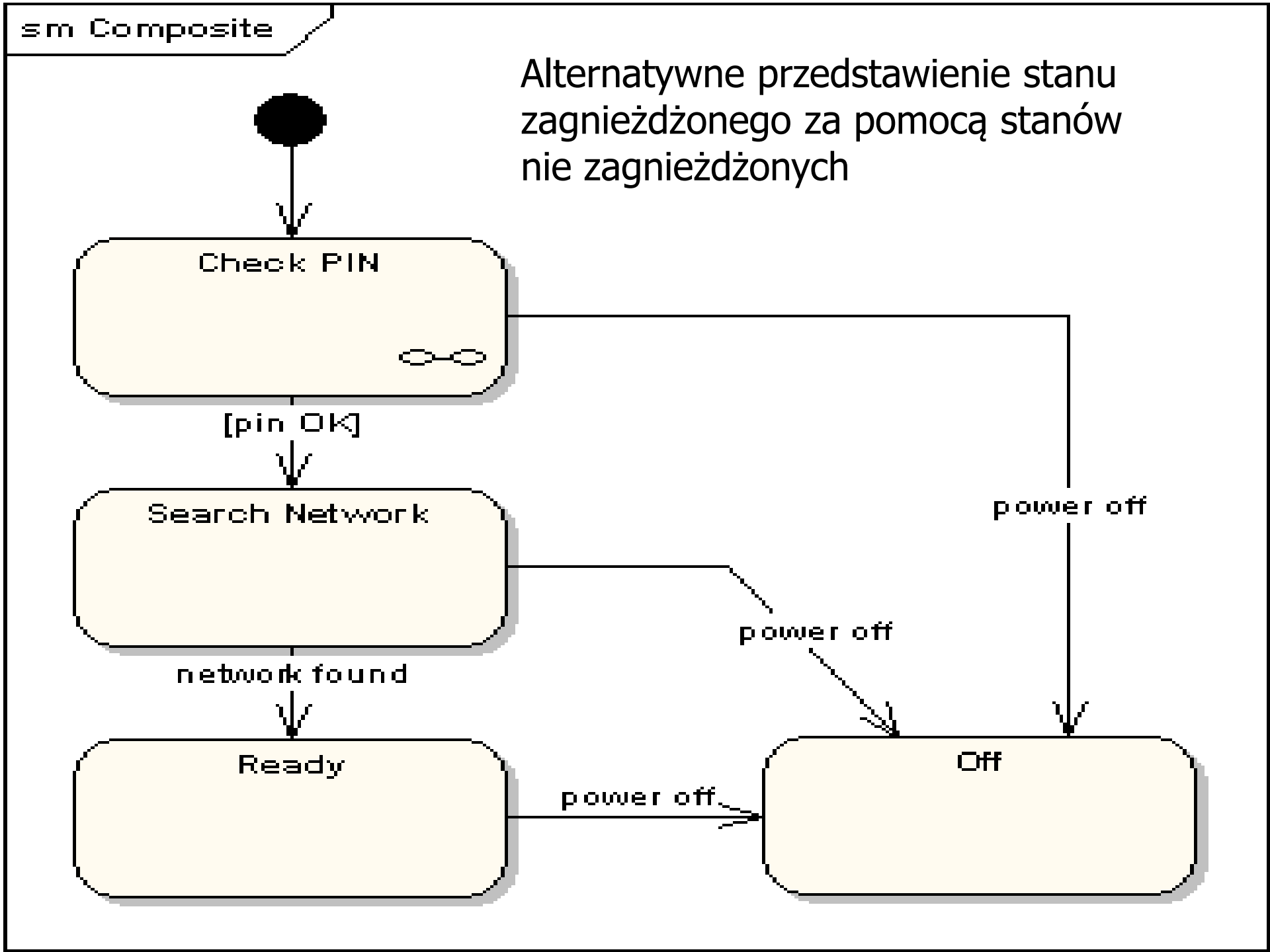
Przejście własne jest związkiem między tym samym stanem, który wskazuje, że np. obiekt znajdujący się w pewnym stanie wykona pewne **akcje** i powróci do tego samego stanu, ilekroć zaistnieje określone zdarzenie i będą spełnione określone warunki.

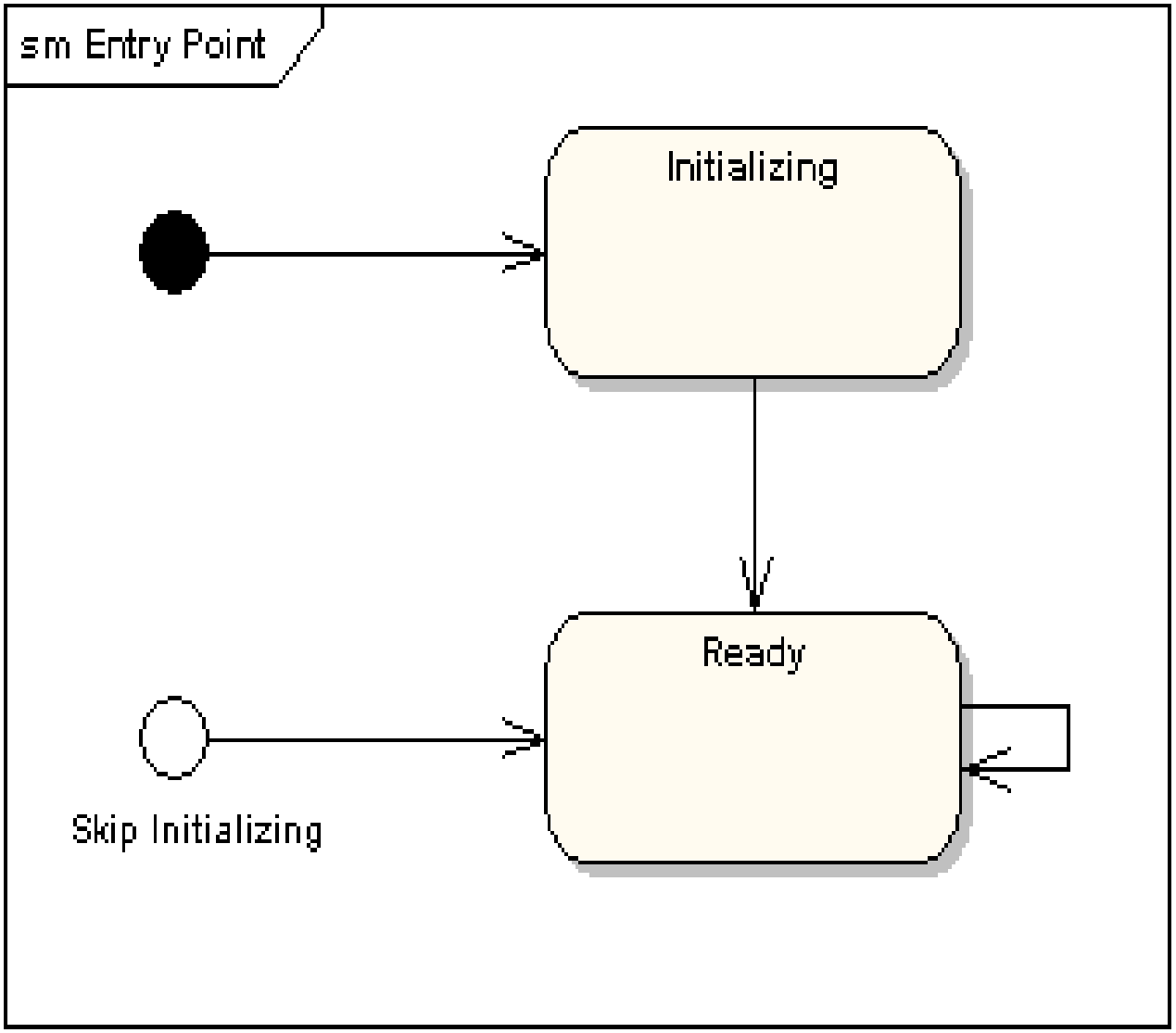




Stany zagnieżdżone

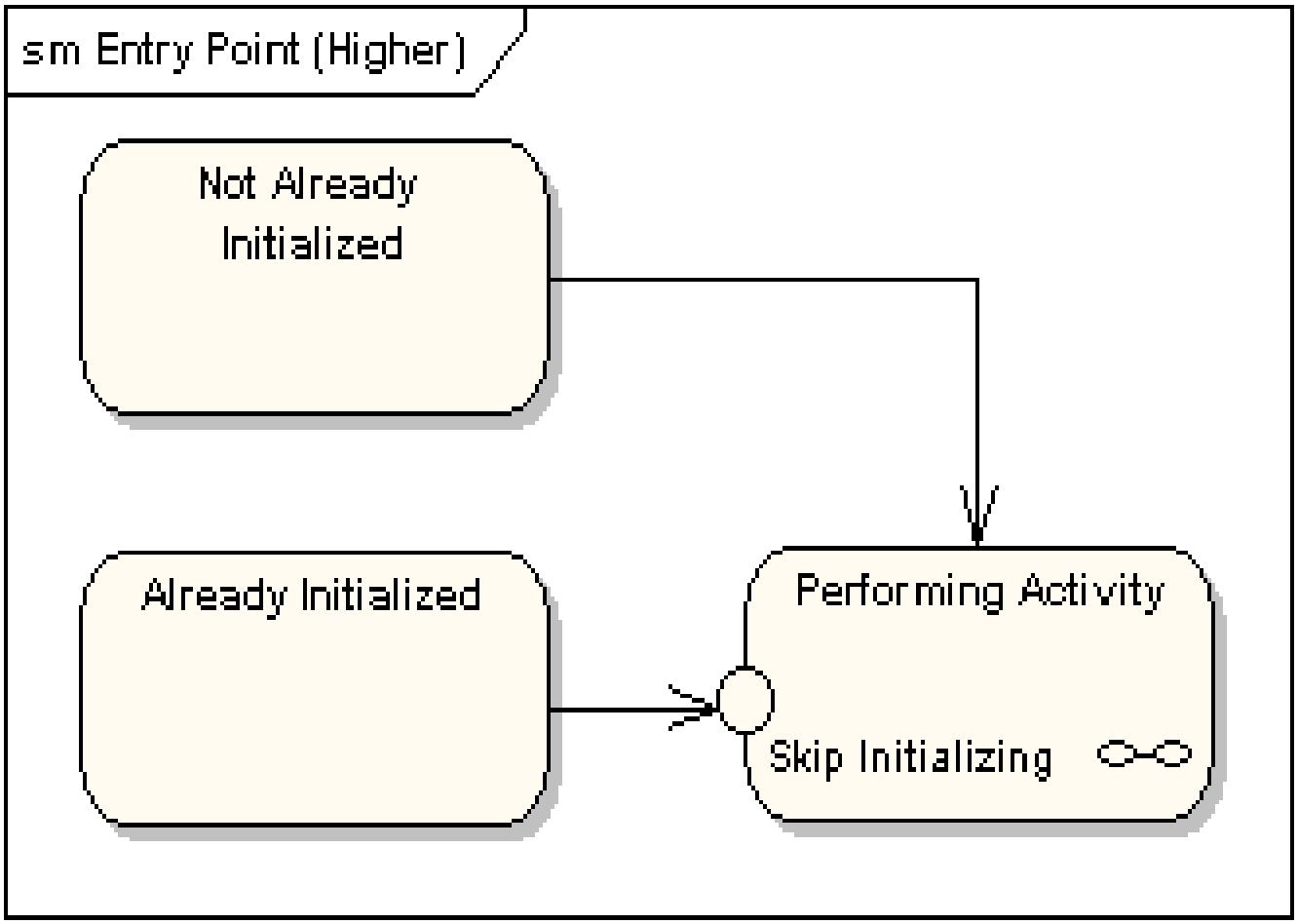
Zagnieżdżona struktura stanu zawierająca podstany rozłączne lub współbieżne


sm Composite
Alternatywne przedstawienie stanu zagnieżdżonego za pomocą stanów nie zagnieżdżonych
Check PIN
[pin OK]
Search Network
network found
Ready
power off
power off
power off
Off



<u>Entry point</u> – wskazanie różnych stanów początkowych (Initial) obiektu:

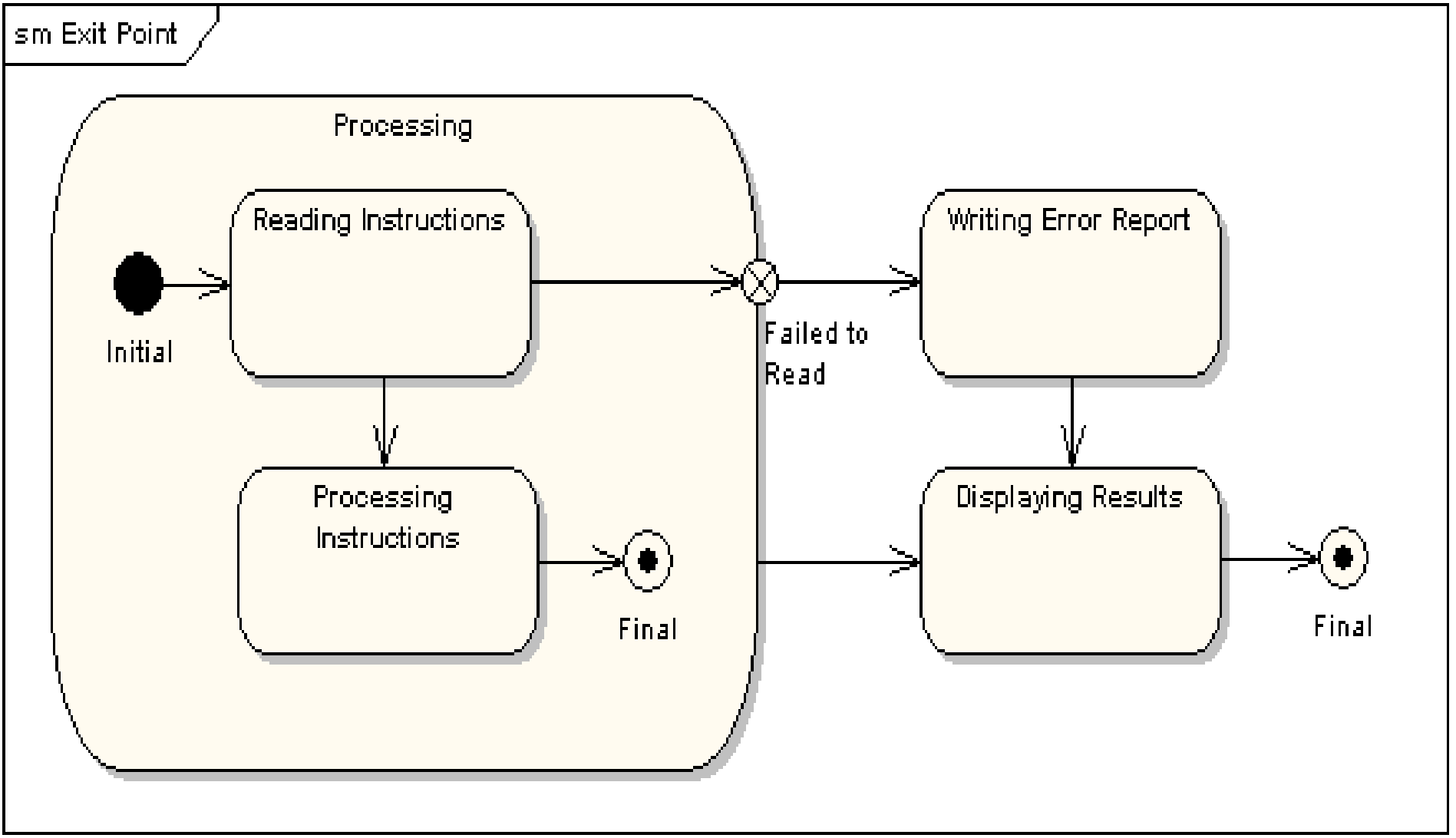
- rozpoczęcie stanu z inicjalizacją (normalne)
- bez inicjalizacji (wyjątkowe)



Entry point – szczegółowy model startu (z poziomu wyższego na poziom niższy), czyli wskazanie różnych stanów początkowych obiektu:

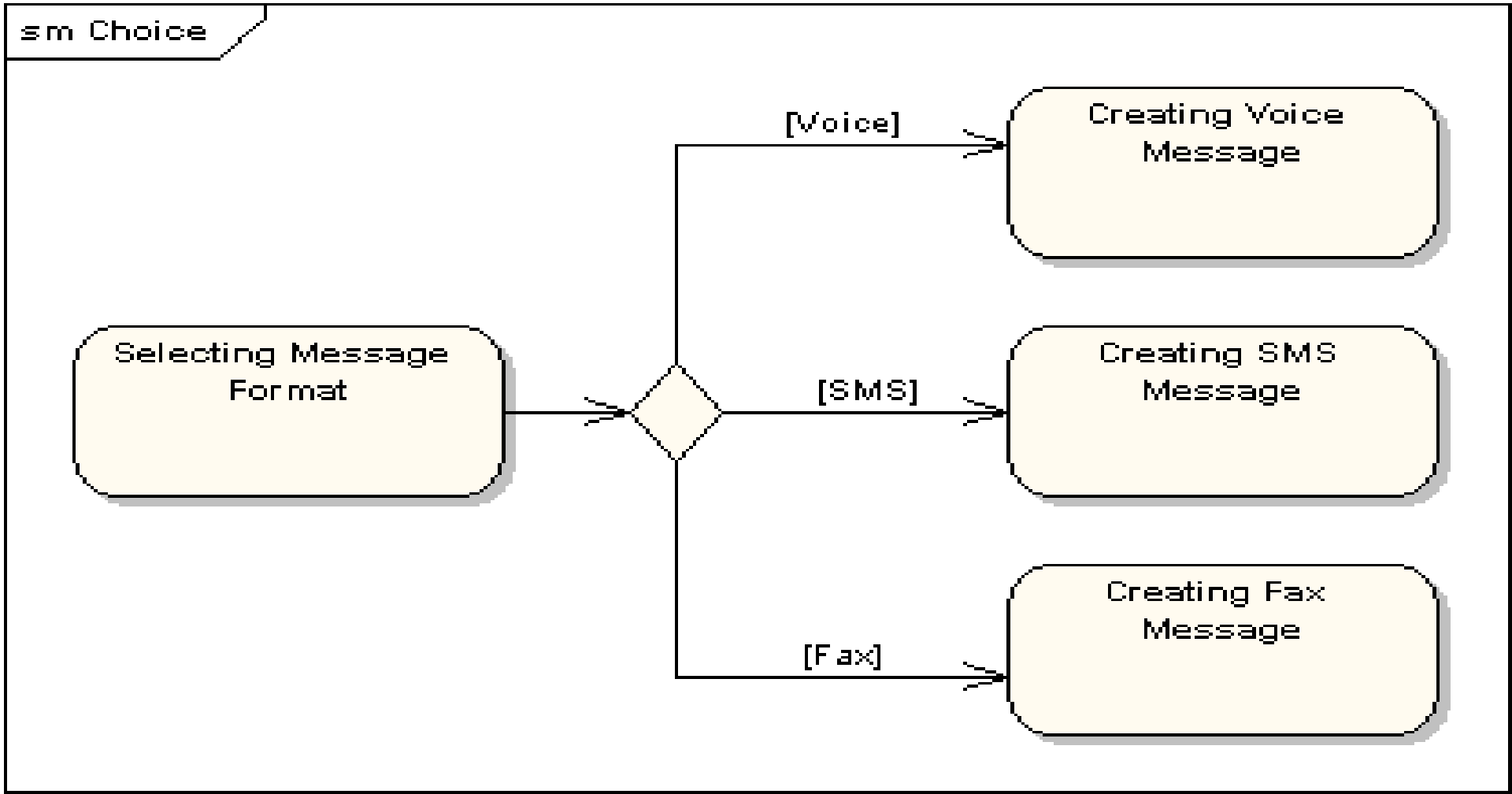
- rozpoczęcie stanu z inicjalizacją (normalne)
- bez inicjalizacji (wyjątkowe)

Punkt wyjścia – modelowanie osiągnięcia alternatywnych stanów końcowych (Final) przez obiekt

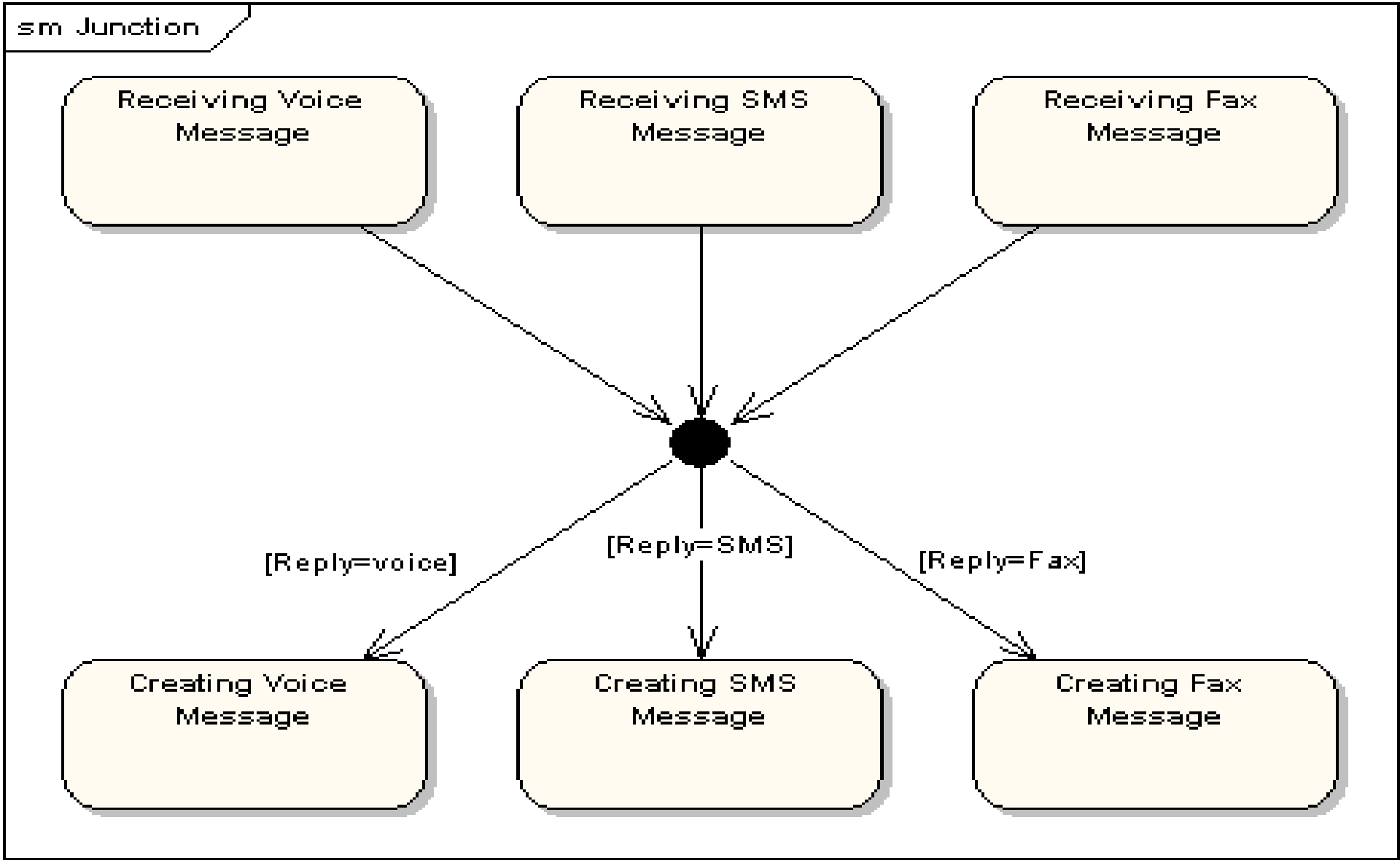


Pseudo stan wyboru:

- Jeden stan początkowy
- W wyniku zdarzenia następuje przejście ze stanu początkowego i na podstawie formatu wygenerowanej wiadomości możliwość wyboru jednego ze stanów wyjściowych

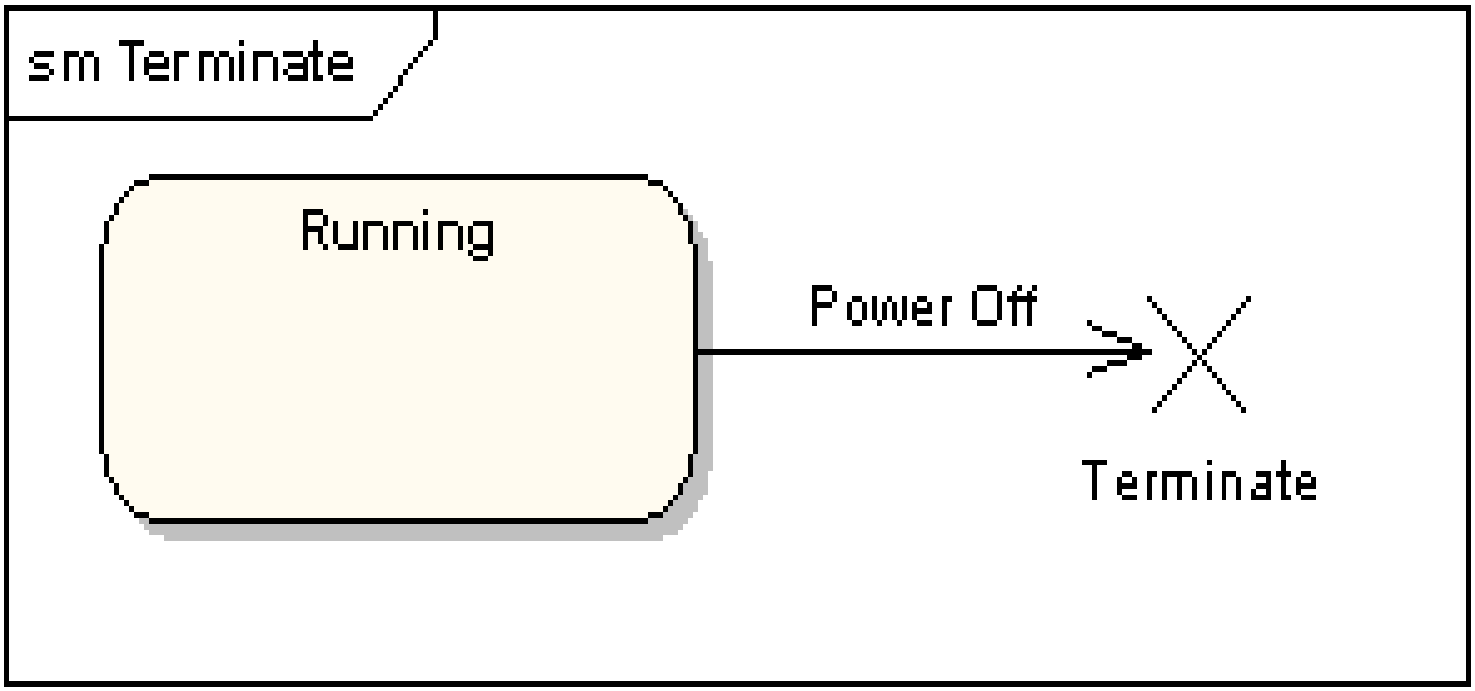


Pseudo stan typu połączenie – możliwości wyboru stanów końcowych po zdarzeniach zachodzących po stanach początkowych

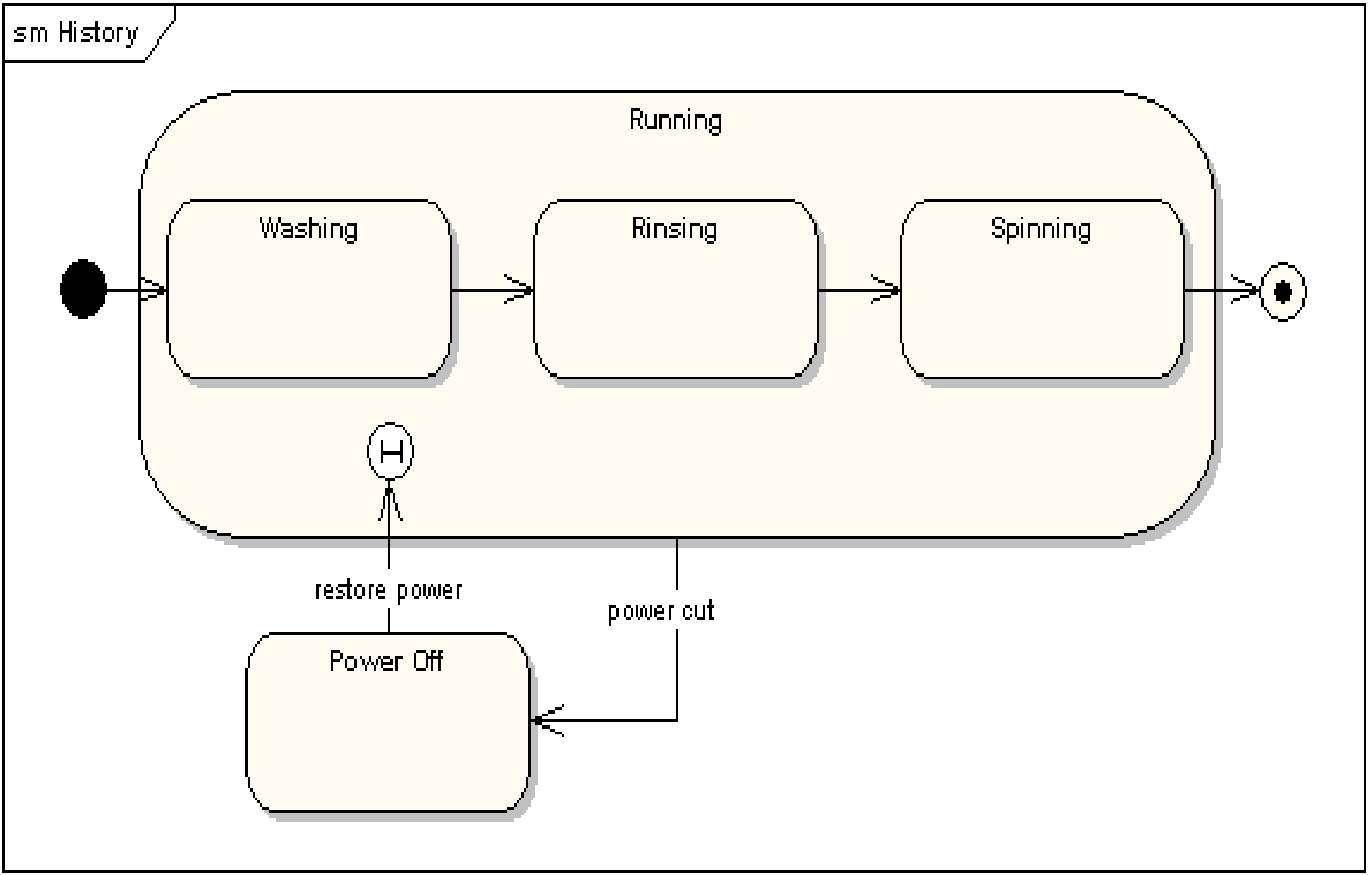


Pseudo stan typu zakończenie

oznacza zakończenie linii życia maszyny stanowej



<u>Stany historyczne</u> – przedstawiają stany wcześniejsze (historyczne) przed przerwaniem działania maszyny stanowej (np. w chwili załączenia zasilania maszyna stanowa zmywarki pamięta stan, w którym ma wznowić działanie)



Równolegle podstany

Stan może być podzielony między równoległe podstany wykonywane jednocześnie. (np. sterowanie przednimi (front) i tylnymi (rear) hamulcami odbywa się równolegle i musi być zsynchronizowane – wyrażone za pomocą symbolu rozdzielenia na pseudo stany oraz symbolu połączenia pseudo stanów. Równoległe podstany są używane do modelowania synchronizacji wątków

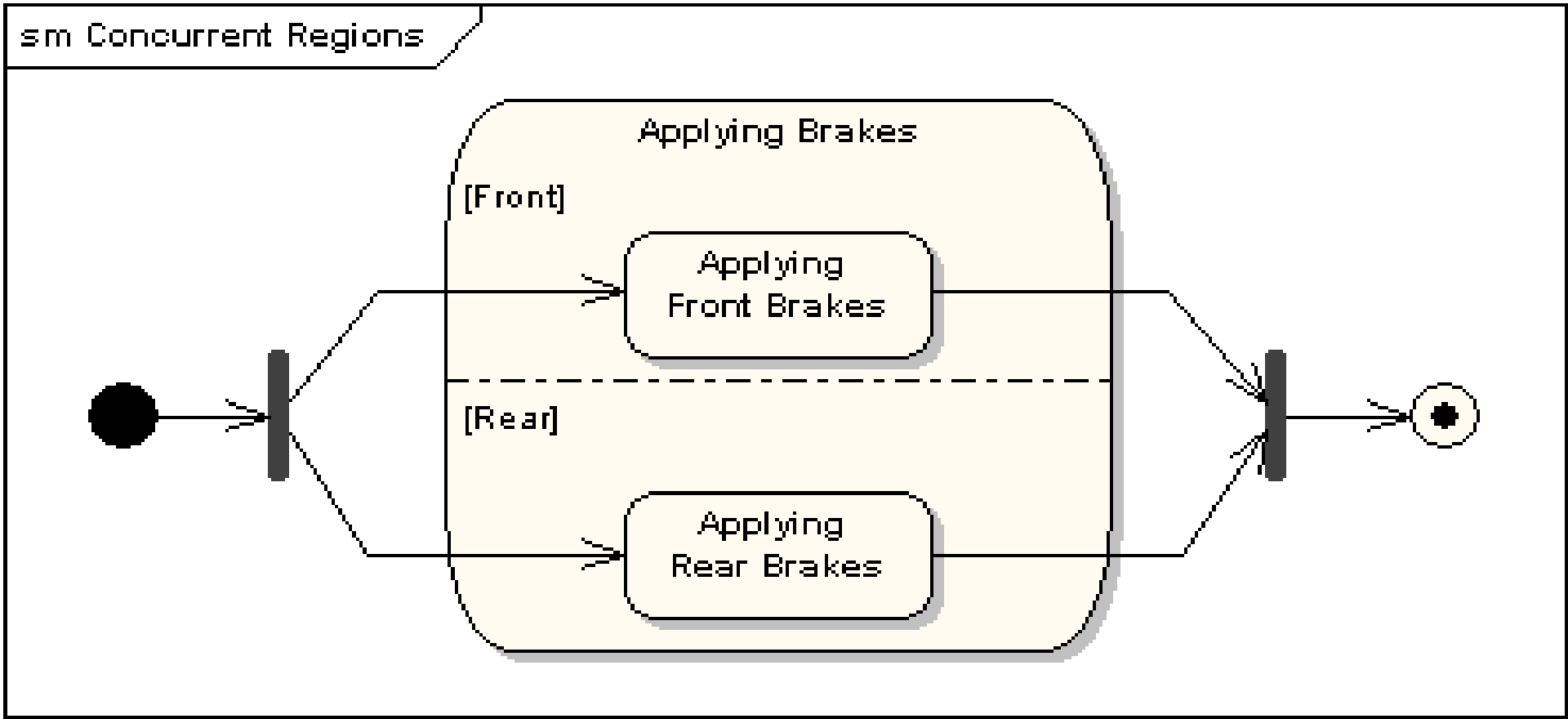
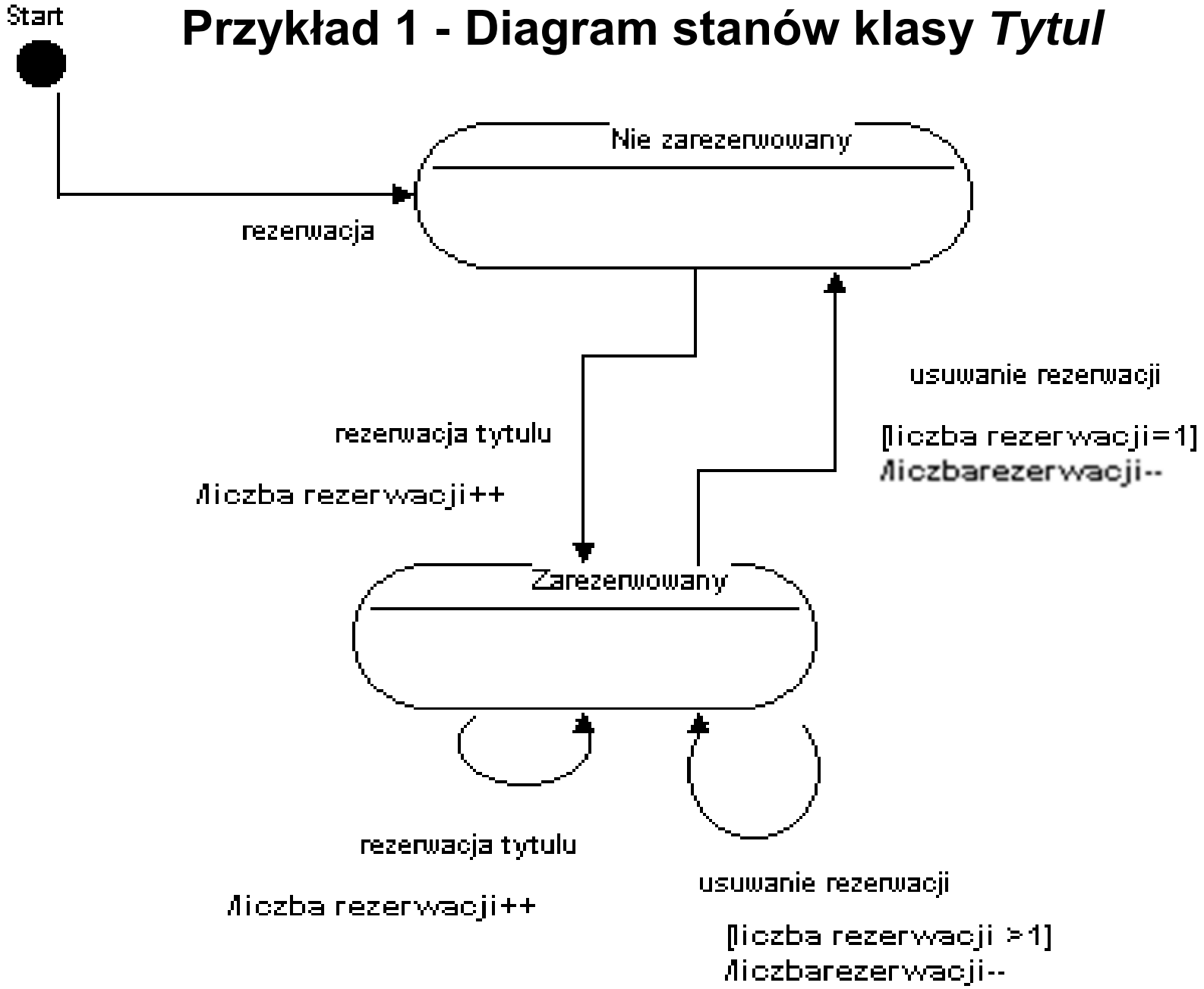


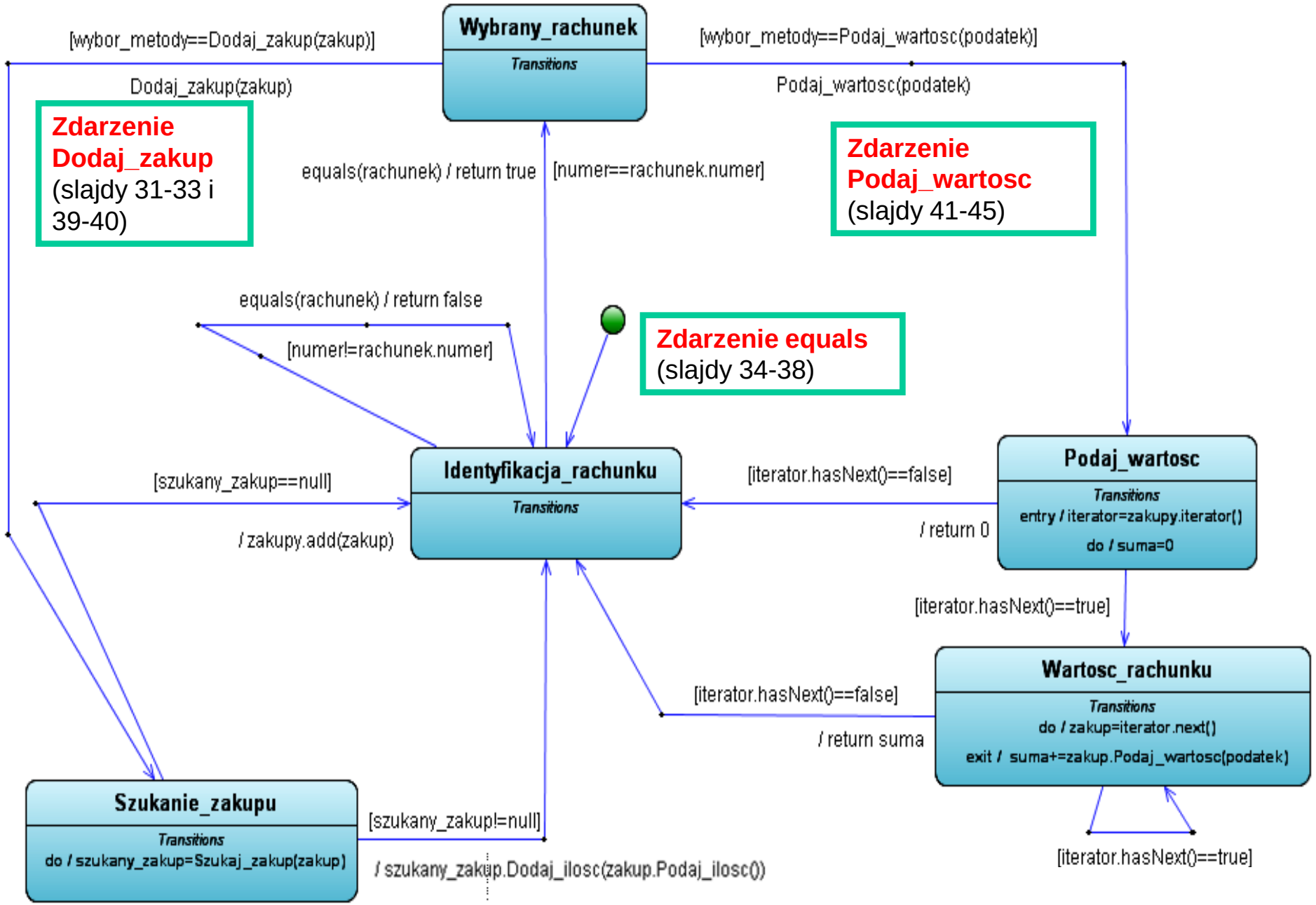
Diagram stanów

1. **Diagramy stanów UML**
   http://sparxsystems.com.au/resources/uml2_tutorial/
2. **Przykład diagramów stanów UML – modelowanie wpływu przypadków użycia na stany obiektu**

Przykład 1 - Diagram stanów klasy *Tytul*

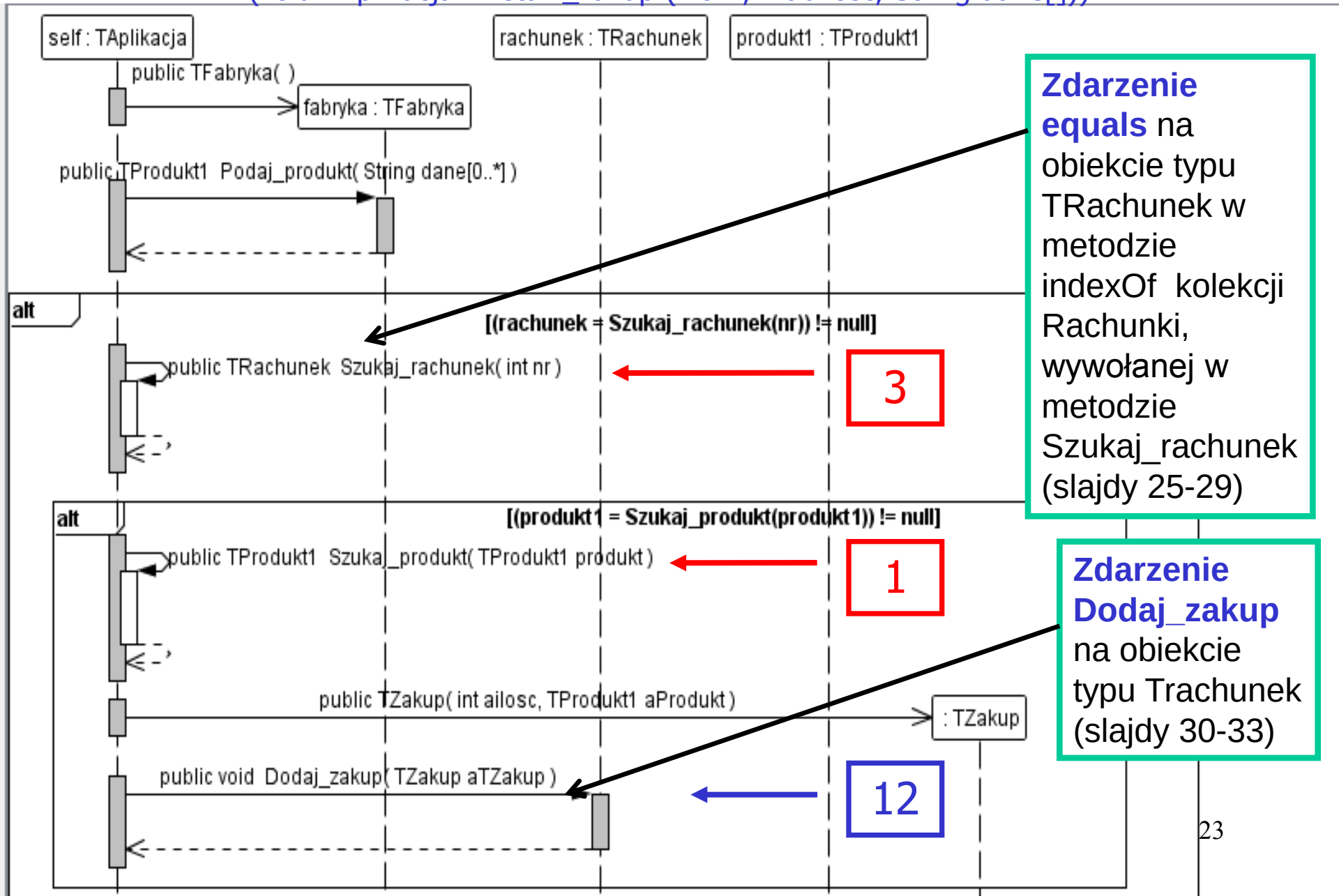


Przykład 2 - Klasa *TRachunek*



Projekt przypadku użycia –
zdarzenie Dodaj_zakup
„Wstawianie nowego zakupu”
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.
Definiowanie kodu metod realizujących przypadek użycia
na podstawie diagramów sekwencji

(5) Wstawianie nowego zakupu – generowanie zdarzeń
equals i Dodaj_zakup na obiekcie typu TRachunek
(void TAplikacja::Wstaw_zakup (int nr, int ailosc, String dane[]))



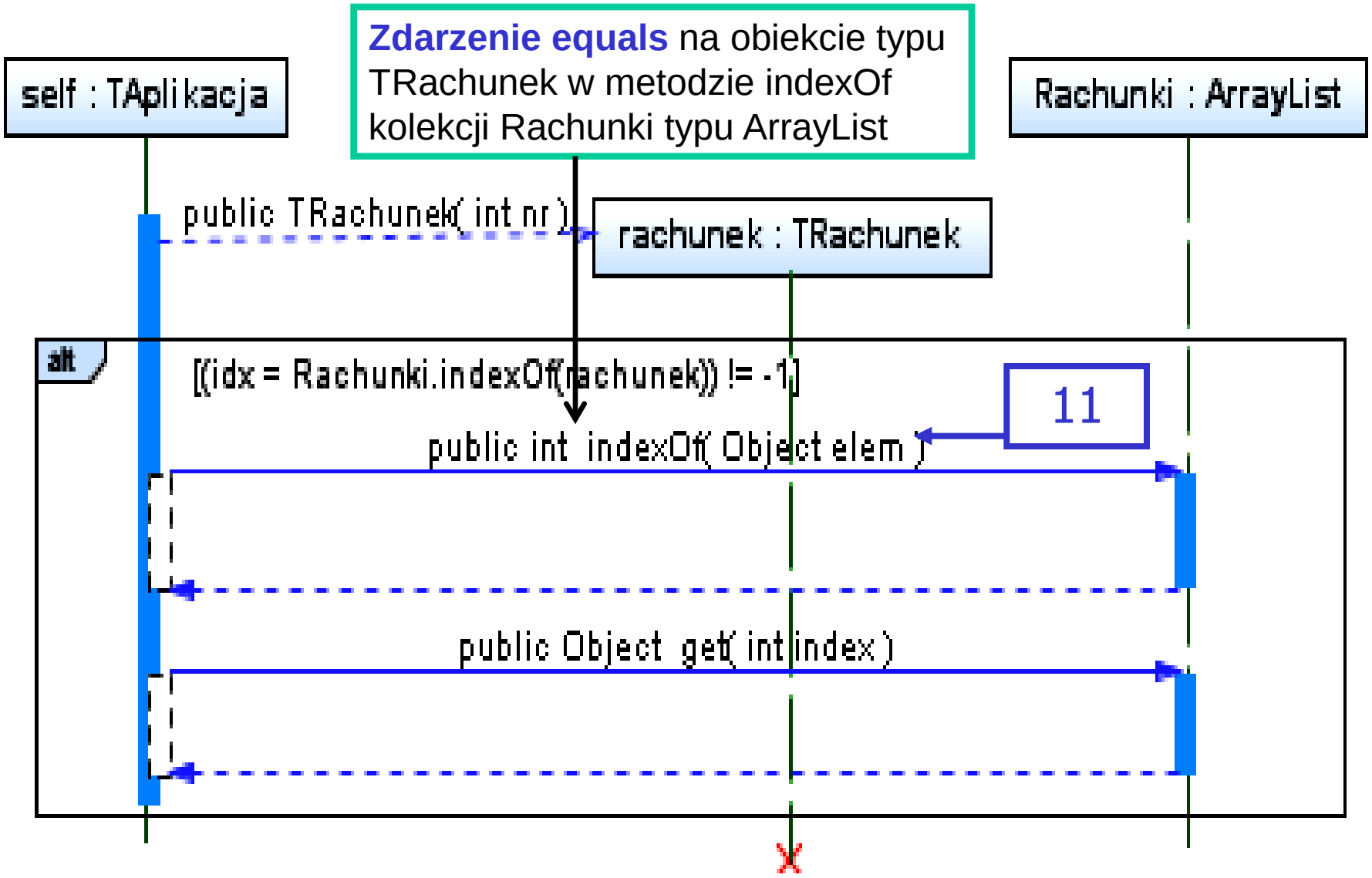
Zdarzenie equals na obiekcie typu TRachunek w metodzie indexOf kolekcji Rachunki, wywołanej w metodzie Szukaj_rachunek (slajdy 25-29)

Zdarzenie Dodaj_zakup na obiekcie typu Trachunek (slajdy 30-33)

```
//TAplikacja

  public   void Wstaw_zakup (int nr, int ile, String dane[])
  {
    TRachunek rachunek;
    TFabryka fabryka = new TFabryka();
    TProdukt1 produkt1 = fabryka.Podaj_produkt(dane);
     if ((rachunek=Szukaj_rachunek(nr)) != null)
      if ((produkt1=Szukaj_produkt(produkt1)) != null)
       rachunek.Dodaj_zakup(new TZakup(ile, produkt1));
   }
```

(3) Szukanie rachunku

(TRachunek TAplikacja::Szukaj_rachunek(int nr))

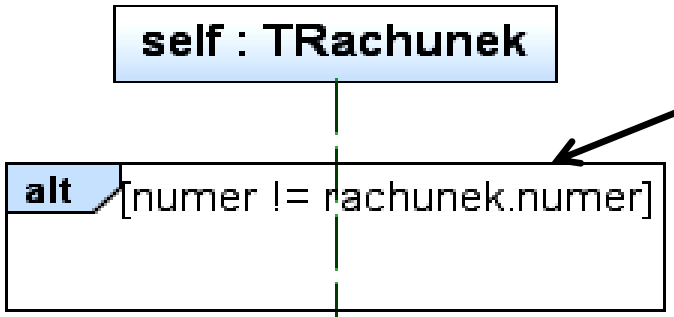


```
//TAplikacja
static private ArrayList <TRachunek> Rachunki  =
                                  new ArrayList <TRachunek>();

  public  TRachunek Szukaj_rachunek (int nr)
   {
      TRachunek rachunek = new TRachunek(nr);
      int idx;
       if ((idx=Rachunki.indexOf(rachunek)) != -1)
        {
           rachunek=Rachunki.get(idx);
           return rachunek;
        }
       return null;
   }
```

```
//metoda indexOf obiektu Rachunki klasy typu
//ArrayList

public int indexOf(Object o) {
        if (o == null) {
            for (int i = 0; i < size; i++)
                if (elementData[i]==null)
                    return i;
        } else {
            for (int i = 0; i < size; i++)
                if (o.equals(elementData[i]))
                    return i;
        }
        return -1;
    }
```

Zdarzenie equals na obiekcie typu TRachunek w metodzie indexOf kolekcji Rachunki typu ArrayList

(11) boolean TRachunek::equals(Object rachunek)



Dozór zdarzenia **equals** na obiekcie typu TRachunek

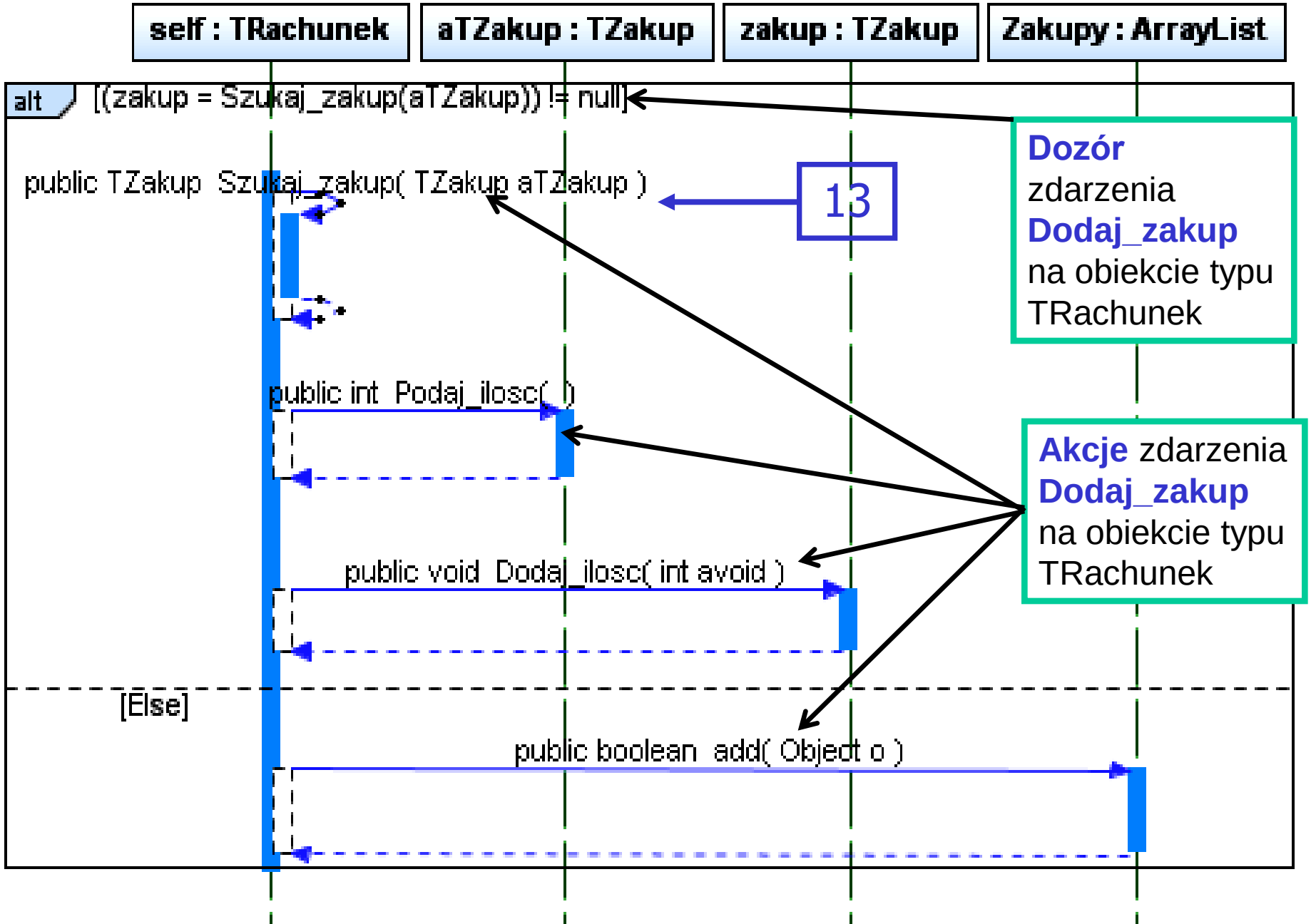
Akcja zdarzenia **equals** na obiekcie typu Trachunek – zwrócenie wyniku dozoru true lub false

```
//TRachunek
 //metoda zdarzeniowa equals
// metody użyte w kodzie metody są akcjami zdarzenia
//instrukcje warunkowe mogą być użyte jako dozory

public boolean equals (Object aTRachunek)
   {
      TRachunek rachunek= (TRachunek)aTRachunek;
      boolean bStatus = true;
       if ( numer!= rachunek.numer )
           bStatus = false;
      return bStatus;
    }
```

Dozór zdarzenia **equals** na obiekcie typu TRachunek

Akcja zdarzenia **equals** na obiekcie typu TRachunek – zwrócenie wyniku dozoru true lub false

(12) void TRachunek::Dodaj zakup(TZakup aTZakup) – **metoda zdarzeniowa**



```
//TRachunek
//metoda zdarzeniowa Dodaj_zakup
// metody użyte w kodzie metody są akcjami zdarzenia
//instrukcje warunkowe mogą być użyte jako dozory

 private ArrayList<TZakup> Zakupy =
                                        new ArrayList<TZakup>();

public void Dodaj_zakup (TZakup aTZakup)
    {
      TZakup zakup;
      if ((zakup = Szukaj_zakup(aTZakup)) != null)
            zakup.Dodaj_ilosc(aTZakup.Podaj_ilosc());
      else
            Zakupy.add(aTZakup);
  }
```

Projekt przypadku użycia

zdarzenie Podaj_wartosc

„Obliczanie wartości rachunku”

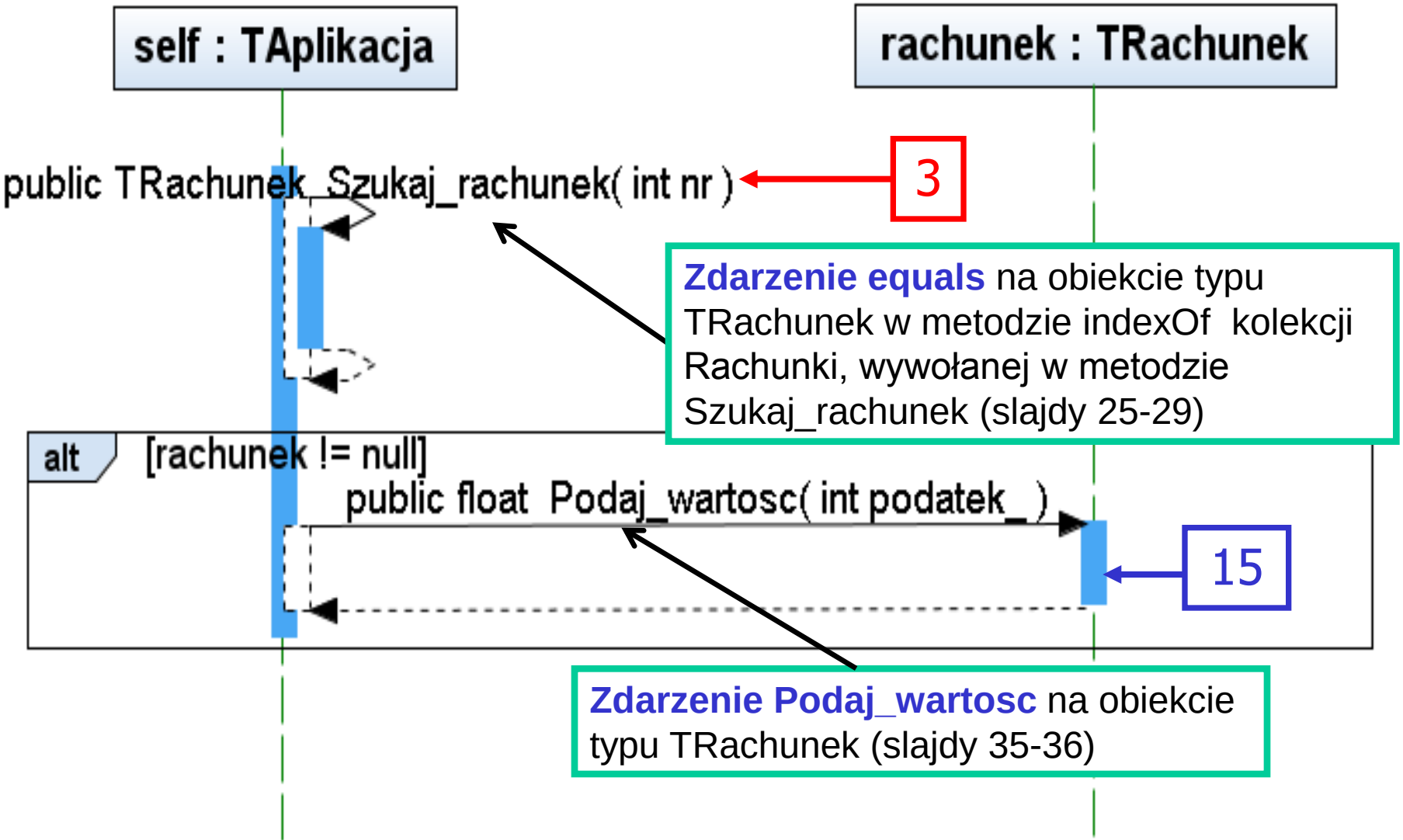
za pomocą diagramu sekwencji i diagramu klas. Diagram klas jest uzupełniany metodami zidentyfikowanymi podczas projektowania scenariusza przypadku użycia za pomocą diagramu sekwencji.

Definiowanie kodu metod realizujących przypadek użycia

na podstawie diagramów sekwencji

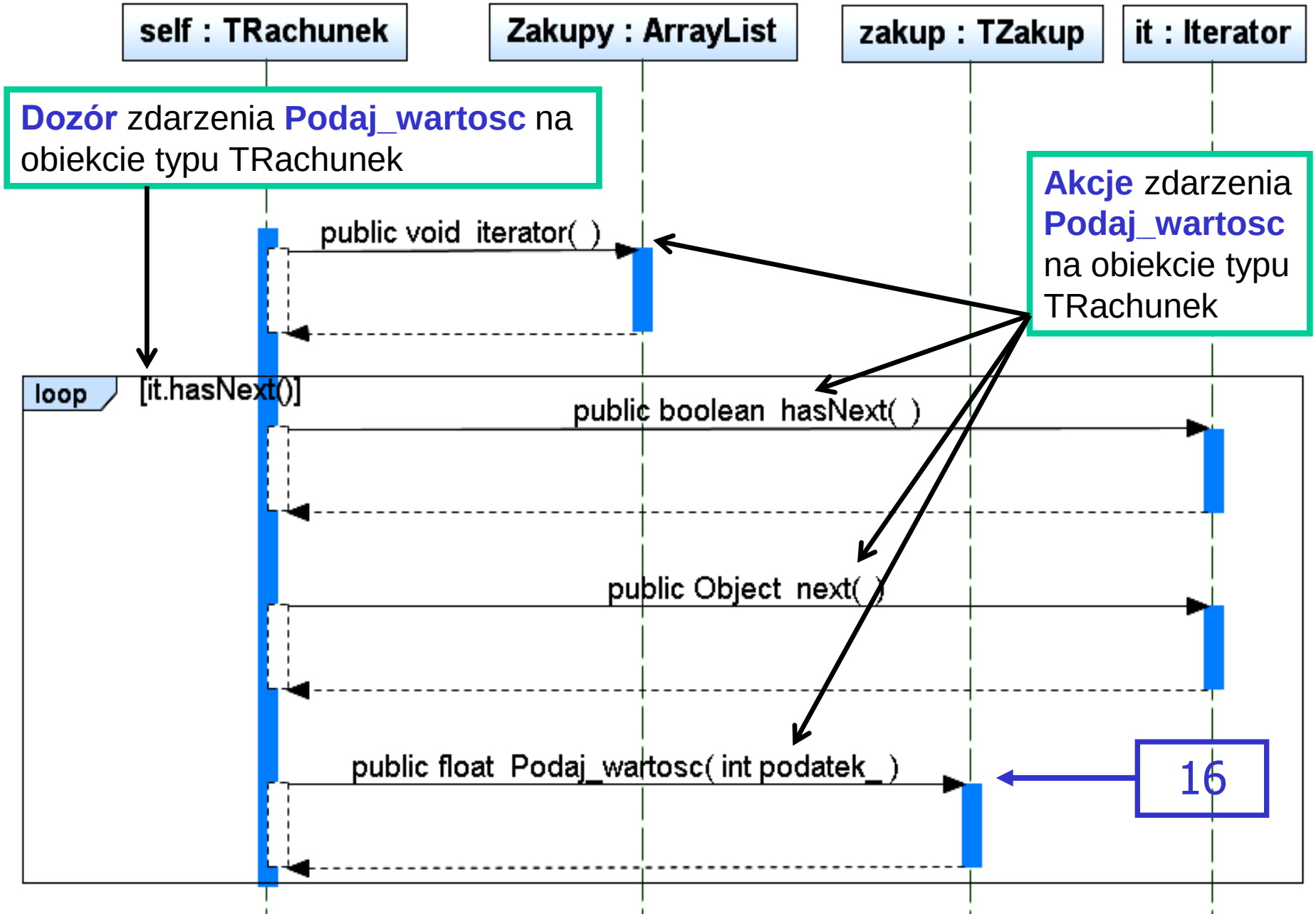
(6) Obliczanie wartosci rachunku – generowanie zdarzeń **equals** i **Podaj_wartosc na obiekcie typu TRachunek**

(float TAplikacja::Podaj_wartosc(int nr, int podatek_))



```
//TAplikacja

 public  float Podaj_wartosc (int nr, int podatek_)
  {
   TRachunek rachunek;
    rachunek = Szukaj_rachunek(nr);
    if (rachunek != null)
        return rachunek.Podaj_wartosc(podatek_);
    return 0F;
  }
```

(15) float TRachunek::Podaj_wartosc(int podatek_)

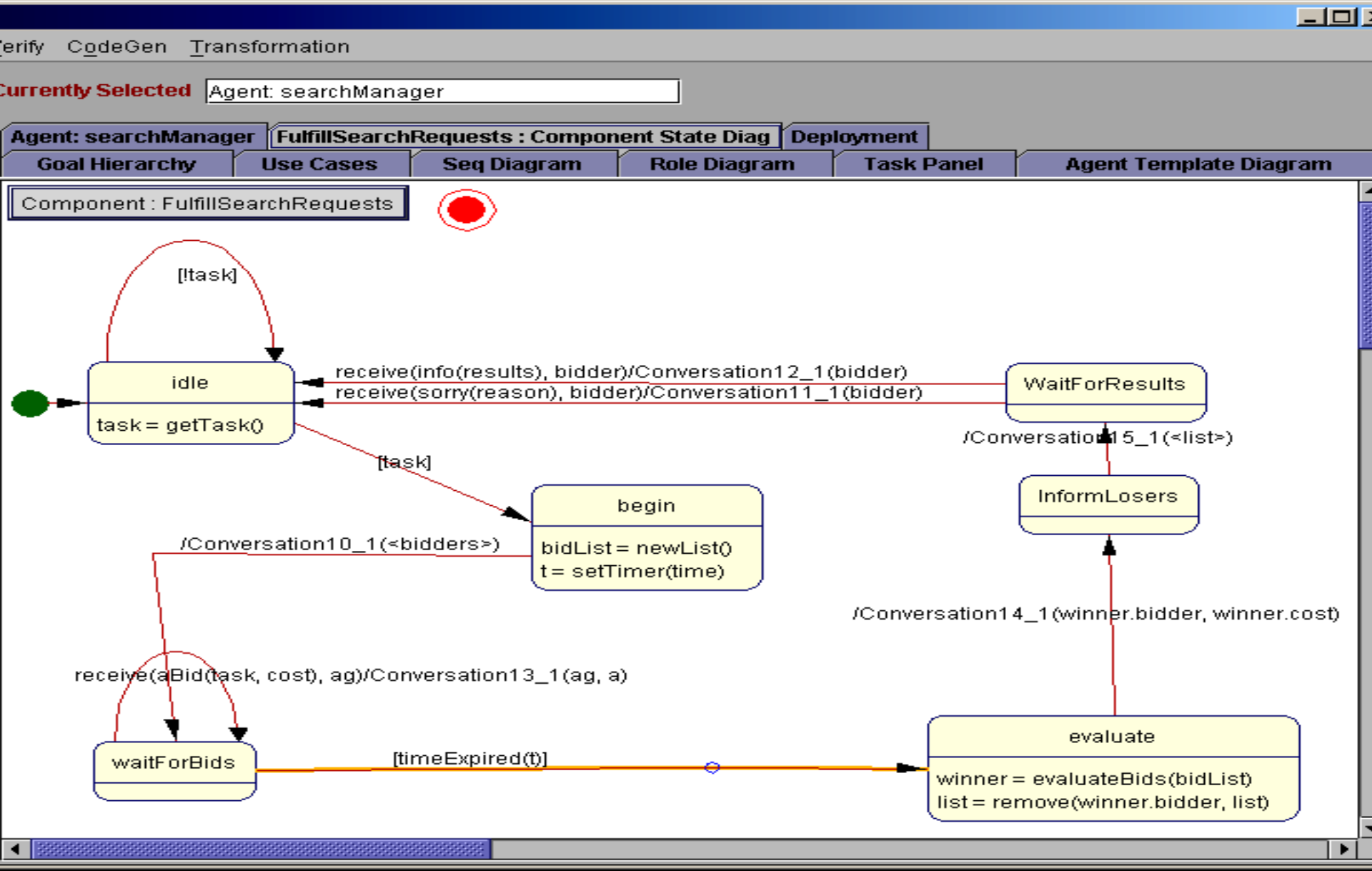


```
//TRachunek
 //metoda zdarzeniowa Dodaj_zakup
// metody użyte w kodzie metody są akcjami zdarzenia
//instrukcje warunkowe mogą być użyte jako dozory

private ArrayList<TZakup> Zakupy =
                                        new ArrayList<TZakup>();

public float Podaj_wartosc (int podatek_)
  {
     float suma=0;
     TZakup zakup;
     Iterator <TZakup> it=Zakupy.iterator();
     while (it.hasNext())
      {  zakup = it.next();
          suma += zakup.Podaj_wartosc(podatek_);
      }
     return suma;
   }
```

Diagramy stanów

1. **Diagramy stanów UML**
   http://sparxsystems.com.au/resources/uml2_tutorial/
2. **Przykład diagramów stanów UML – modelowanie wpływu przypadków użycia na stany obiektu**
3. **Przykład diagramu stanów UML – inżynieria wprost i inżynieria odwrotna**

Przykład 3.1 – diagram stanu protokołu komunikacji klasy FulfillSearchReaquest (na podstawie projektu dostarczonego z programem AgentTool1.8.3)



Przykład 3.2 – kod wygenerowany na podstawie diagramu stanu protokołu komunikacji klasy FulfillSearchReaquest (na podstawie projektu dostarczonego z programem AgentTool1.8.3)

```
/* Class for Component FulfillSearchRequests
* automatically generated by agentTool - agentMom transform ver 0.5*/
package wyklad2;
import java.net.*;
import java.io.*;
import java.awt.*;
import afit.mom.*;
import java.util.*;
public class FulfillSearchRequests extends afit.mom.Component
{
    public Object  task;
    public Object  bidList;
    public Object  t;
    public Object  time;
    public Object  winner;
    public Object  bidder;
    public Object  cost;
    public Vector  bidders;
    public Vector  list;
    public task_cost_Type  task_cost;
    public Object  ag;
    public Object  a;
    public Object  results;
    public Object  reason;
```

```
/* Constructor for component.
* automatically generated by agentTool - agentMom transform ver 0.5
*/
public FulfillSearchRequests (Agent a)
{
    parent = a;
}
/* getTask Method
* automatically generated by agentTool - agentMom transform ver 0.5
*
*/
public Object getTask()
{
    /* Enter method code here */
    return null;
}
/* newList Method
* automatically generated by agentTool - agentMom transform ver 0.5
*
*/
public Object newList()
{
    /* Enter method code here */
    return null;
}
```

```
/* setTimer Method
* automatically generated by agentTool - agentMom transform ver 0.5
*
*/
public Object setTimer(Object time)
{
    /* Enter method code here */
    return null;
}
/* evaluateBids Method
* automatically generated by agentTool - agentMom transform ver 0.5
*
*/
public Object evaluateBids(Object bidList)
{
    /* Enter method code here */
    return null;
}
/*remove Method
* automatically generated by agentTool - agentMom transform ver 0.5
*
*/
public Object remove(Object bidder, Object list)
{
    /* Enter method code here */
    return null;
}
```

```
/*Conversation14_1 Method
* automatically generated by agentTool - agentMom transform ver 0.5
*/
public void Conversation14_1(Object bidder, Object cost)
{
   ConnectionData connData = (ConnectionData)bidder;
   String serverHost = connData.serverHost;
   int serverPort = connData.serverPort;
   Conversation14_1_searchManager_I conv =
         new Conversation14_1_searchManager_I(this,serverHost,serverPort,cost);
   conv.run();
}
/*Conversation10_1 Method
* automatically generated by agentTool - agentMom transform ver 0.5
*/
public void Conversation10_1(Vector bidders)
{
  if (bidders != null)
     for(int i = 0; i < bidders.size(); i++) {
       ConnectionData connData = (ConnectionData)bidders.elementAt(i);
       String serverHost = connData.serverHost;
       int serverPort = connData.serverPort;
       Conversation10_1_searchManager_I conv =
            new Conversation10_1_searchManager_I(this, serverHost, serverPort);
      conv.run();
 }
}
```

```
/*Conversation15_1 Method
* automatically generated by agentTool - agentMom transform ver 0.
*/
public void Conversation15_1(Vector list)
{
  if (list != null)
    for(int i = 0; i < list.size(); i++) {
      ConnectionData connData = (ConnectionData)list.elementAt(i);
      String serverHost = connData.serverHost;
      int serverPort = connData.serverPort;
      Conversation15_1_searchManager_I conv =
                new Conversation15_1_searchManager_I(this, serverHost, serverPort);
      conv.run();
 }
}
/*Conversation13_1 Method
* automatically generated by agentTool - agentMom transform ver 0.5
*/
public void Conversation13_1(Object ag, Object a)
{
   ConnectionData connData = (ConnectionData)ag;
   Socket server = connData.server;
   ObjectInputStream input = connData.input;
   ObjectOutputStream output = connData.output;
   Thread t = new Thread(new Conversation13_1_searchManager_R(server,input,output,
                                    this, m, a));
    t.start();
}
```

```
/*Conversation12_1 Method
* automatically generated by agentTool - agentMom transform ver 0.5
*/
public void Conversation12_1(Object bidder)
{
    ConnectionData connData = (ConnectionData)bidder;
    Socket server = connData.server;
    ObjectInputStream input = connData.input;
    ObjectOutputStream output = connData.output;
    Thread t = new Thread(new Conversation12_1_searchManager_R(server,
                                    input, output, this, m));
    t.start();
}
/*Conversation11_1 Method
* automatically generated by agentTool - agentMom transform ver 0.5
*/
public void Conversation11_1(Object bidder)
{
    ConnectionData connData = (ConnectionData)bidder;
    Socket server = connData.server;
    ObjectInputStream input = connData.input;
    ObjectOutputStream output = connData.output;
    Thread t = new Thread(new Conversation11_1_searchManager_R(server,
                                    input, output, this, m));
    t.start();
}
```

```
/*add Method
* automatically generated by agentTool - agentMom transform ver 0.5
*
*/
public Object add(Object bid, Object bidList)
{
    /* Enter method code here */
    return null;
}
/*display Method
* automatically generated by agentTool - agentMom transform ver 0.5
*
*/
public void display(Object results)
{
    /* Enter method code here */
}
```

```
/*run method - controls state table behavior.
* automatically generated by agentTool - agentMom transform ver 0.5*/
public void run()
{
    int state = 0;
    boolean notDone = true;
        /* state constant definitions */
    final int StartState = 0;
    final int idle = 1;
    final int begin = 2;
    final int waitForBids = 3;
    final int evaluate = 4;
    final int WaitForResults = 5;
    final int InformLosers = 6;
    final int StartState_out = 7;
    final int idle_out = 8;
    final int waitForBids_out = 9;
    final int evaluate_out = 10;
    final int begin_out = 11;
    final int InformLosers_out = 12;
    final int WaitForResults_out = 13;
        //set up
    while (notDone) {
        switch (state) {
            case StartState :       state = StartState_out;     break;
            case StartState_out :   state = idle;               break;
```

```
case idle :          task = getTask();
                     state = idle_out;                                  break;
case idle_out :      if  (task)              // [task]
                          state = begin;
                     else  if ( !task)       // [!task]
                          state = idle;                                 break;
case begin :         bidList = newList();
                     t = setTimer(time);
                     state = begin_out;                                 break;
case begin_out :                             //  /Conversation10_1(<bidders>)
                     Conversation10_1(bidders);
                     state = waitForBids;                               break;
case waitForBids :   state = waitForBids_out;                           break;
case waitForBids_out : m = checkExternal();
                     if (m != null) {
                          // receive(aBid(task, cost), ag) /Conversation13_1(ag, a)
                     if (m.performative.equals("aBid")) {
                          task_cost = (task_cost_Type)m.content;
                          task = task_cost.task;
                          cost = task_cost.cost;
                          Conversation13_1(ag, a);
                          state = waitForBids;   }
                      }
                     else        // [timeExpired(t)]
                          if ( timeExpired(t) )
                               state = evaluate;                        break;
```

```
case evaluate :          winner = evaluateBids(bidList);
                         list = remove(winner.bidder, list);
                         state = evaluate_out;                                  break;
case evaluate_out :                 //  /Conversation14_1(winner.bidder, winner.cost)
                         Conversation14_1(winner.bidder, winner.cost);
                         state = InformLosers;                                  break;
case InformLosers :      state = InformLosers_out;                              break;
case InformLosers_out :             //  /Conversation15_1(<list>)
                         Conversation15_1(list);
                         state = WaitForResults;                                break;
case WaitForResults :    state = WaitForResults_out;                            break;
case WaitForResults_out : m = checkExternal();
                          if (m != null) {
                           // receive(info(results), bidder) /Conversation12_1(bidder)
                          if (m.performative.equals("info")) {
                             results = m.content;
                             Conversation12_1(bidder);
                             state = idle;  }
                          // receive(sorry(reason), bidder) /Conversation11_1(bidder)
                          else if (m.performative.equals("sorry")) {
                               reason = m.content;
                               Conversation11_1(bidder);
                               state = idle; }
                          }
                                                                                 break;
    } }
 } // end run
}  // end class
```