25 Февраля 2012
06 Марта 2012

Инсерционное моделирование 2

Лекция 4

Графические модели

UML languages

Object Management Group, object-oriented engineering

2007, весна 4 инсерционное моделирование
Лекции 09-10

UML: United Marxist-Leninist (Nepal) ☺

ITU (International Communication Union) languages

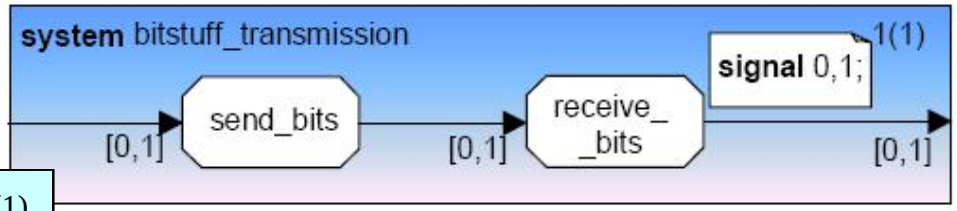
SDL, MSC, UCM

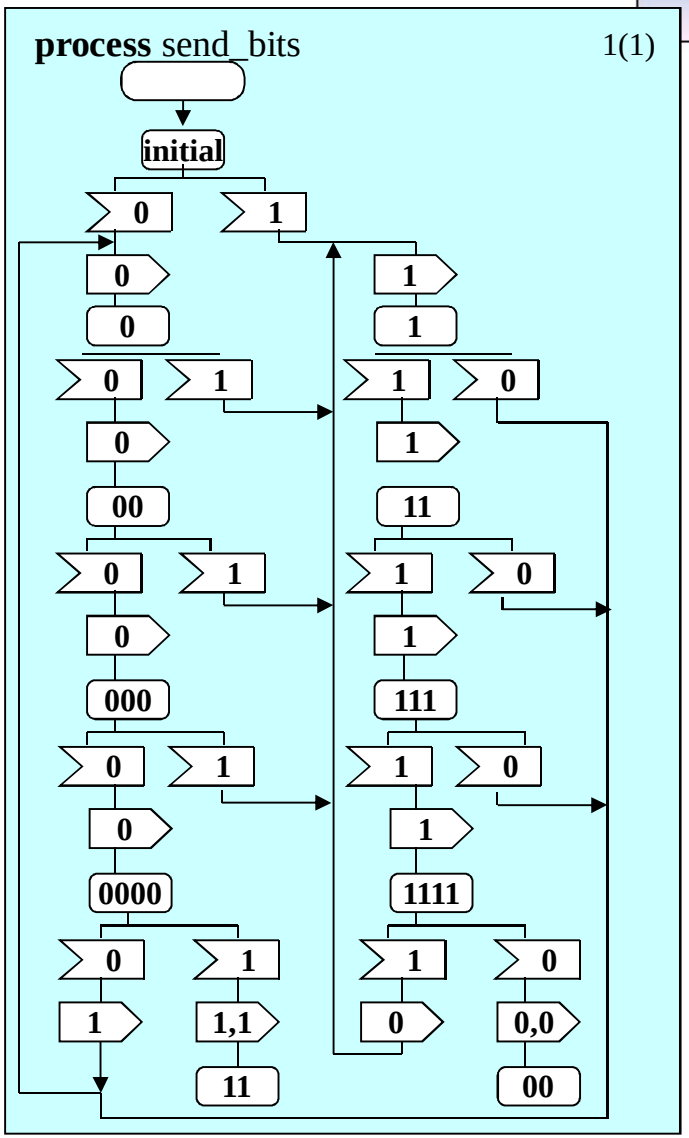
Behavioral languages

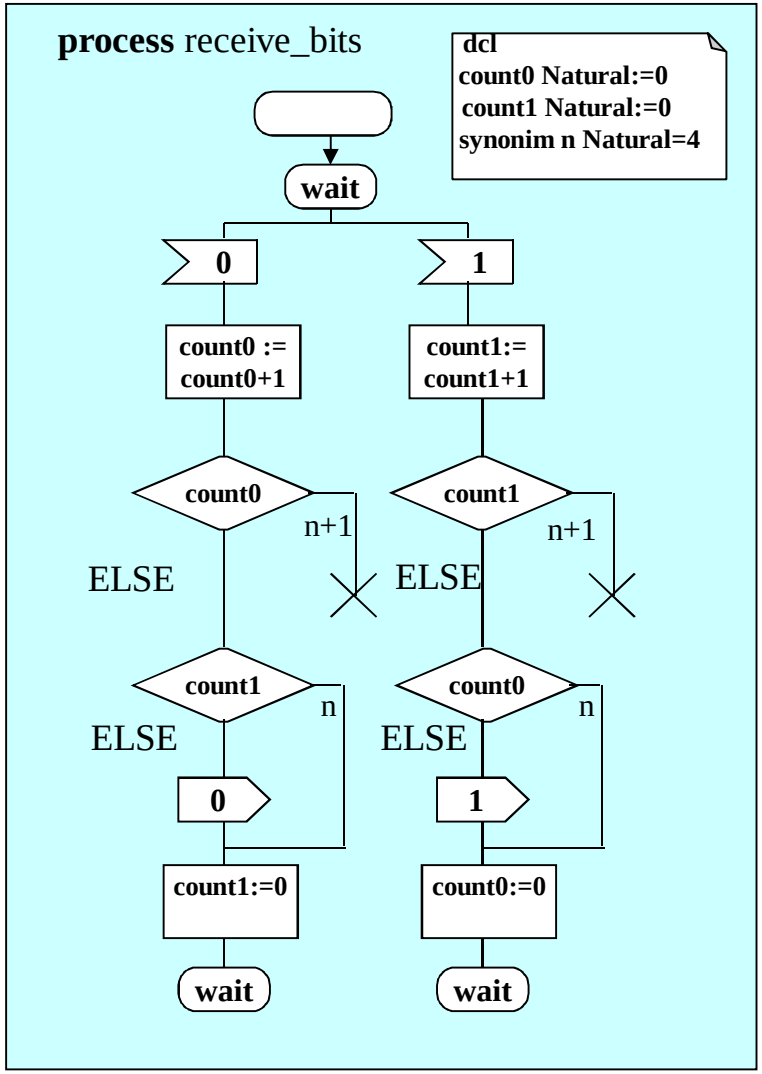
Convergence with UML

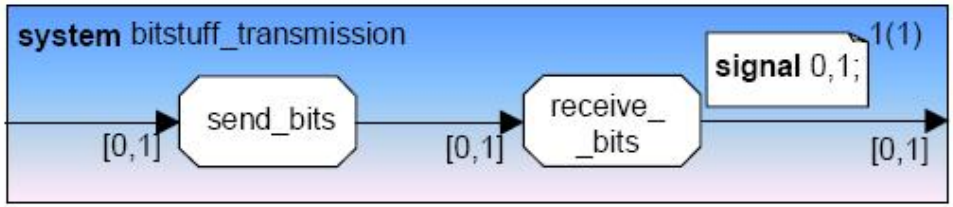
Use cases, sequencing diagrams, State machines

SDL


system bitstuff_transmission
1(1)
send_bits
receive_
_bits
signal 0,1;
[0,1]
[0,1]
[0,1]


process send_bits
1(1)
initial
0
1
00
11
000
111
0000
1111
1,1
0,0


process receive_bits
dcl
count0 Natural:=0
count1 Natural:=0
synonim n Natural=4
wait
0
1
count0 :=
count0+1
count1:=
count1+1
count0
count1
n+1
ELSE
n
count1:=0
count0:=0



Example simple system model **Bit-stuffing one-way transmission**. This system consists of a send-bits transmitter and a receive-bits receiver. The transmitter inserts (stuffs in) bits so that there are never n bits the same. The receiver removes the inserted bits.

This technique is used in real systems to protect against stuck at zero or one or (for example in Signalling System 7) to allow flags that consist of n ones or zeros to be inserted without the risk that they are imitated by signals.

Задание: Доказать, что на выходе системы будет то же самое, что и на входе

Инсерционная модель

Уровень блоков
Среда: сеть, связывающая блоки
Буферизованными каналами
Агенты: процессы или блоки
Прием сообщений
Передача сообщений
Параллельное погружение
Много агентов

Уровень процессов
Среда: память
Агенты:
Условия, присваивания
Передача сообщений
Прием сообщений
Последовательное погружение
Один агент

Функция погружения

$$\frac{E \xrightarrow{a} E', u \xrightarrow{a} u'}{E[u] \longrightarrow E'[u']}$$

a – внутреннее действие

$$\frac{E \xrightarrow{a} E', u \xrightarrow{a} u'}{E[u] \xrightarrow{a} E'[u']}$$

a – внешнее действие

Внутренние действия процессов: условия и присваивания

Внешние действия процессов: обмен сообщениями

Внутренние действия блоков: обмен по внутренним каналам

Внешние действия блоков: обмен по внешним каналам (входным-выходным)

Параллельная композиция: чистый интерливинг

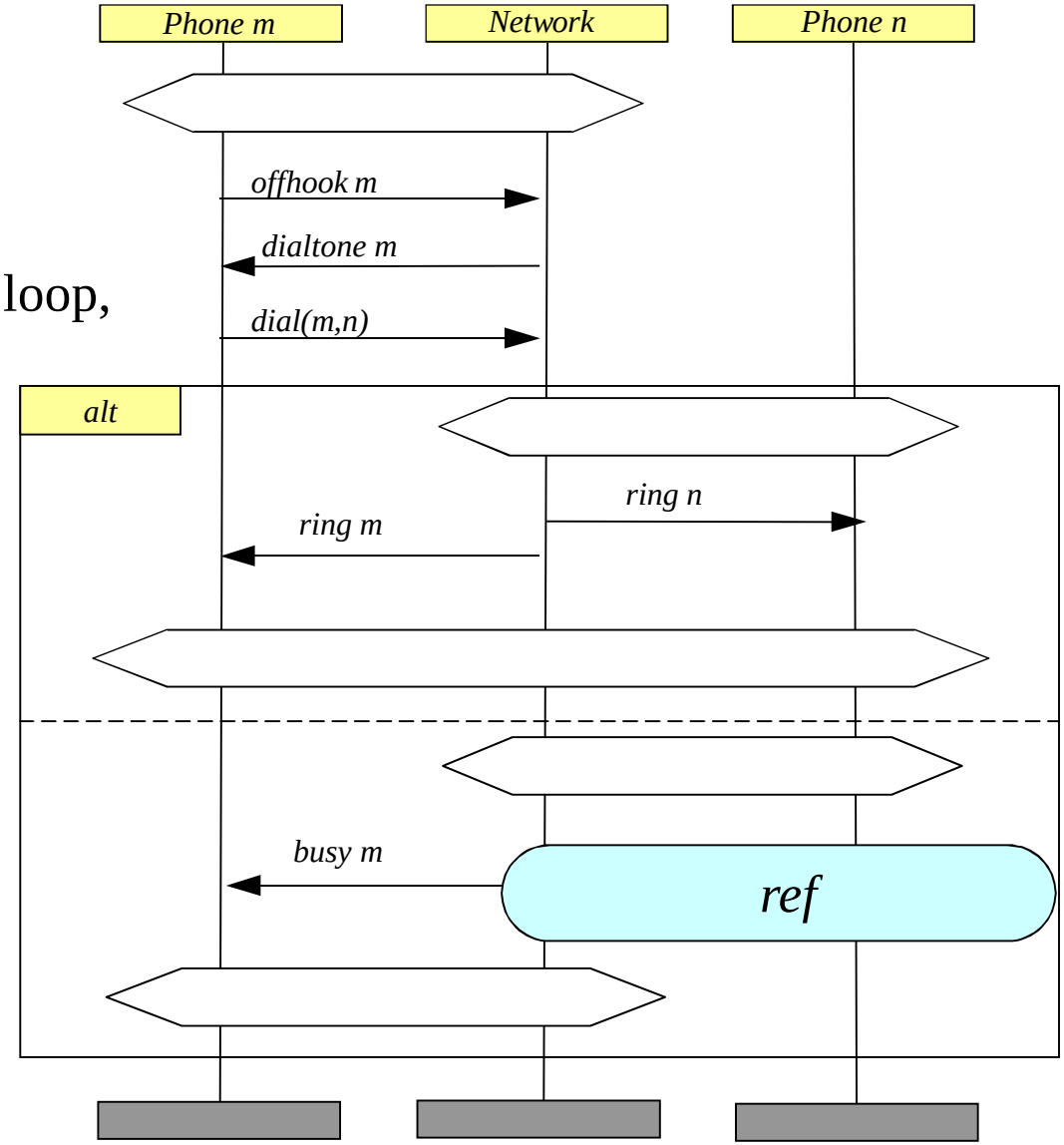
Имена агентов: передаются через действия

Состояния среды: структуры данных, содержащие информацию о структуре сети и внутренней памяти

$$u \xrightarrow{a} u' \Rightarrow m:u \xrightarrow{m:a} m:u'$$

MSC

opt, exc
par, seq, loop,



MSC агенты-инстанции

Действия

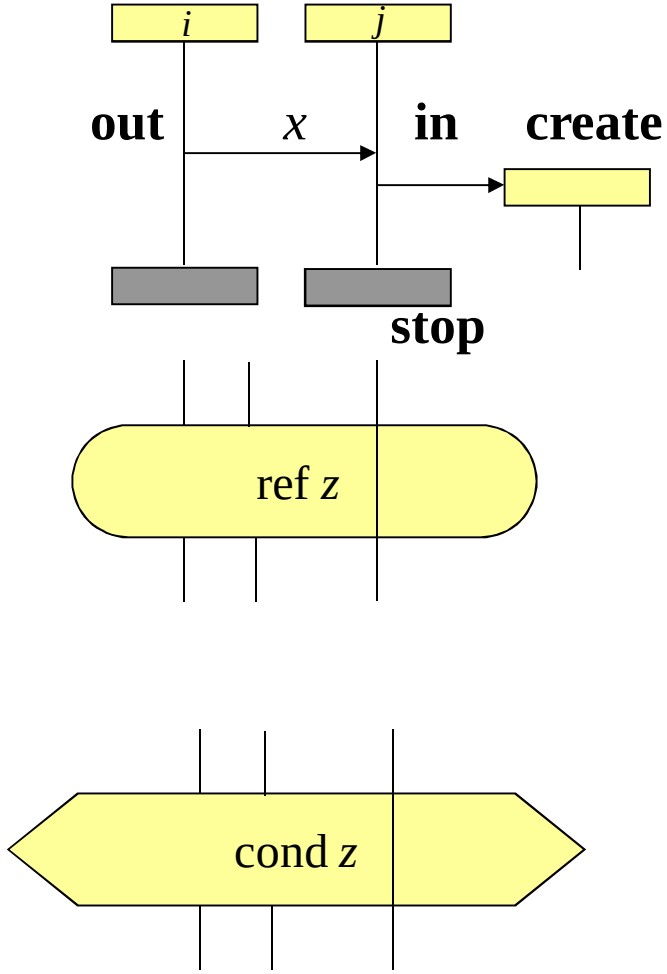
Сообщения: **out** x(i,j), **in** x(i,j)

Локальные действия: **action** x(i)

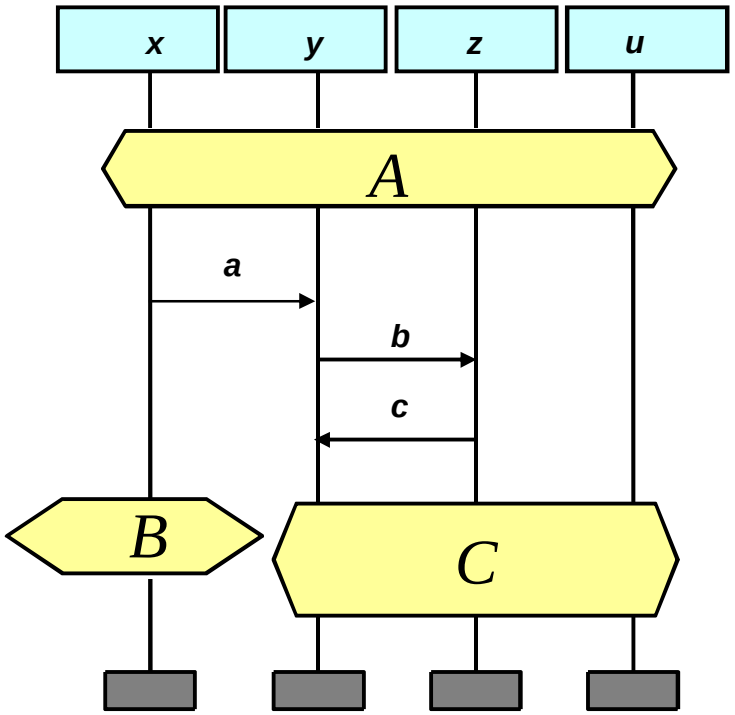
Инстанции: **inst**(i), **create**(i,j), **stop**(i)

Управление: **cond** z(i,J), **ref** z(i,J)

Functions: (*P*;*Q*), (*P*| |*Q*), …



MSC диаграмма => MSC агент



inst(x).cond A(x,J).out(a,x,y).cond B(x,{x}).stop x | |
inst(y).cond A(y,J). in(a,x,y).out(b,x,y). in(c,x,y).
cond C(y,K).stop y | |
inst(z).cond A(z,J). in(b,y,z).out(c,z,y).
cond C(z,K).stop z | |
inst(u).cond A(u,J).cond C(u,K).stop u

J={x,y,z,u}, K={y,z,u}

on-line выражения определяются с помощью функциональных выражений и недетерминированного выбора (статья в Кибернетике).

Правильный порядок:

out =>in

Синхронизация по условиям

За порядком следит среда

Слабая последовательная композиция MSC-агентов

$$s[P,Q] = (s[P])[Q] = s[P * Q]$$

$$P = p_1 \| \ldots \| p_m \| q_1 \| \ldots \| q_k$$
$$Q = r_1 \| \ldots \| r_l \| q_1' \| \ldots \| q_k'$$
$$P * Q = p_1 \| \ldots \| p_m \| (q_1; q_1') \| \ldots \| (q_k; q_k') \| r_1 \| \ldots \| r_l$$

$p_1, \ldots, p_m, r_1, \ldots r_l$ — *различные инстанции*

q_i, q_i' — *одинаковые инстанции*

$$(\sum P_i) * Q = \sum (P_i * Q)$$

Вложенные выражения
(on-line expressions)

F – функция перевода в язык процессов

F(**loop**(m,n) E’, P1,P2 ,...) = **loop**(m,n,F(E’ P1,P2 ,...));
F(**alt** E1 **alt** E2 **alt**..., P1,P2 ,...)= F(E1,P1,P2 ,...)+ F(E2,P1,P2 ,...)+...;
F(**opt** E’, P1,P2 ,...) = F(E’ P1,P2 ,...) +Δ;
F(**par** E1 **par** E2 **par**... , P1,P2 ,...) = F(E1,P1,P2 ,...) | | F(E2,P1,P2 ,...) | | ... ;
F(**seq** E1 **seq** E2 **seq**... , P1,P2 ,...) = (F(E1,P1,P2 ,...); F(E2,P1,P2 ,...) ;...);
F(**exc** E, P1,P2 ,...) = (F(E, P1,P2 ,...);0) +Δ;
F(xi, P1,P2 ,...) = Pi ;

loop(0,0,G) = Δ,
loop(0,inf,G) = (G*__loop__(0,inf,G)) + Δ,
loop(m,inf,G) = (G*__loop__(m-1,inf,G)),
loop(0,n,G) = (G*__loop__(0,n-1,G)) + Δ,
loop(m,n,G) = (G*__loop__(m-1,n-1,G)).

MSC машина

Состояние среды разрешает или запрещает выполнение действий в соответствии с их частичным порядком и синхронизацией определяемой условиями и ссылками.

Разрешающая аддитивная
Параллельная функция погружения

$$\frac{e \xrightarrow{a} e'[v], u \xrightarrow{a} u'}{e[u] \xrightarrow{a} e'[v \parallel u']}, P(e,a)$$

Среда

Состояние : история, последовательность выполненных действий

a	$P(e,a)$
out x(i,j),	инстанции i и j открыты и i не заблокирована
in x(i,j)	инстанции i и j открыты, **out** находится в состоянии ожидания, j не заблокирована
action x(i)	инстанция i открыта и не заблокирована
inst(i),	инстанция i не открыта
create(i,j),	i открыта и не заблокирована, j не открыта
stop(i)	i открыта и не заблокирована
cond z(i,J),	инстанция i открыта и не заблокирована
ref z(i,J)	инстанция i открыта и не заблокирована

Погружение нового агента

$$\frac{e \xrightarrow{a} e'[v], u \xrightarrow{a} u'}{e[u] \xrightarrow{a} e'[v \,||\, u']}, P(e,a)$$

a=**ref** z(i, J), все инстанции из J не заблокированны
v=val(z)

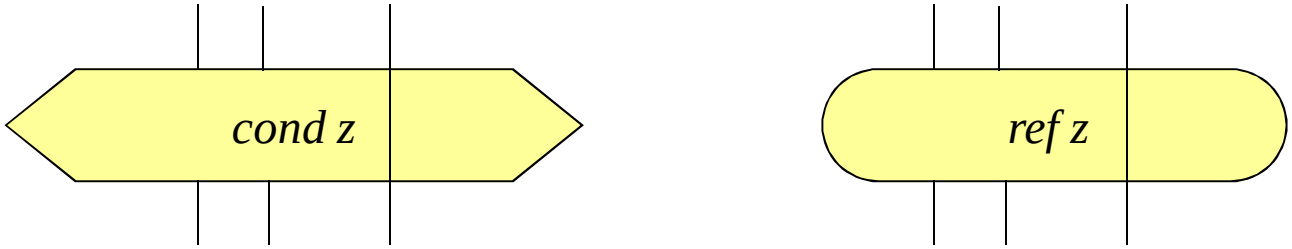
Другие варианты состояния среды

- ***Состояние среды:***
  - *история*
  - *формула линейной темпоральной логики*
  - *состояние автомата*
  - *простой объект*

MSC среда простой объект

Состояние

<process: *U*, ***out:*** *O*, ***synchr:*** *S*, ***perform:*** *P*, ***allinst:*** *F*>



$U : names \rightarrow MSCs$

$O(x, i, j) = m \Leftrightarrow$ *m событий* **out** *x(i,j) не имеют соответствующих* **in-ов**

$S(x, y, J) = I \subseteq J$, $y =$ ***wait cond*** *z*, ***wait ref*** *z*, *x* – *имя ссылки*

$P(z) = (J, y) \Leftrightarrow$ *ссылка z присоединенная к J выполняется внутри ссылки y*

F(x): все активные инстанции внутри ссылки x

laoxiong@iis.nsk.su