Лекція 32.

Загальні принципи побудови моделюючих алгоритмів

При реалізації моделюючих алгоритмів на цифрових машинах є певні загальні закономірності, які ми обговоримо в даній лекції.

Основна проблема при складанні алгоритмів на машині з послідовною обробкою процесів полягає в тому, що при моделюванні необхідно відслідковувати множину процесів, які в реальному часі відбуваються паралельно. У зв'язку із цим алгоритми моделювання мають свої особливості:

- просування системи в часі, відстеження тимчасової координати;
- забезпечення синхронної роботи об'єктів, з яких складається модельована система.

У даний момент відомі чотири основних принципи регламентації подій.

1. Принцип Δt («дельта-те»).
2. Принцип особливих станів.
3. Принцип послідовної проводки заявок.
4. Принцип паралельної роботи об'єктів (об'єктний принцип моделювання).

Розглянемо на прикладах, як реалізується в моделюючих алгоритмах кожен принцип окремо.

Принцип Δt

Принцип полягає в тому, що алгоритмом моделювання імітується рух, тобто зміна стану системи, у фіксовані моменти часу: t, $t + \Delta t$, $t + 2\Delta t$, $t + 3\Delta t$, …

Для цього заводиться лічильник часу (годинники), що на кожному циклі збільшує своє значення t на величину кроку в часі Δt, починаючи з нуля (початок моделювання). Таким чином, зміни системи відслідковуються такт за тактом у задані моменти: t, $t + \Delta t$, $t + 2\Delta t$, $t + 3\Delta t$, …

Особливості реалізації принципу Δt

Це **найбільш універсальний** з розглянутих принципів, тому що застосовується для дуже широкого класу систем. Він же є **найбільш простим** у реалізації, оскільки принцип Δt збігається з розумінням людини про час, як про послідовне явище, що текет з постійним темпом.

Однак це **самий неекономічний** принцип, оскільки вся система аналізується моделюючим алгоритмом на кожному такті, навіть якщо в ній не відбувається ніяких змін.

Інший недолік полягає в тому, що часи подій округляються до величини Δt, що веде до **погрішностей** у визначенні змінних, що характеризують систему.

Розглянемо приклад. Моделюється склад виробів з максимальною ємністю G (див. мал. 32.1).

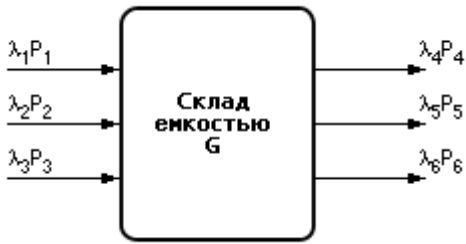

Рис. 32.1. Схема модельованої виробничої системи (приклад)

Склад приймає вироби від трьох постачальників (позначимо номера вхідних потоків виробів 1, 2, 3) і видає їхнім трьом споживачам (позначимо номера вихідних потоків виробів 4, 5, 6). Приймемо як характеристика кожного входу інтенсивність *i-го* потоку виробів (λ_i). Інтенсивність характеризує в середньому відстань між двома моментами часу приходів (відходів) виробів на склад. Позначимо через P_i розмір партії виробів, що йдуть або приходять на склад. Тобто цими змінними ми визначили, коли й скільки приходить або йде виробів на той або інший вхід або вихід складу (**блок 1** — тут і далі див. блоки на мал. 32.2).

Позначимо змінної Z поточна кількість виробів на складі. Якщо виробу приходять на склад (потоки 1, 2, 3), то Z збільшується на P_1, P_2 або P_3. Якщо вироби вилучаються зі складу (потоки 4, 5, 6), то Z зменшується на P_4, P_5 або P_6. Тобто Z відіграє роль лічильника виробів, що перебувають на складі в окремий момент часу t. Початковий стан складу на момент $t := 0$ приймемо $Z := Z_0$.

Алгоритм побудуємо таким чином, щоб обчислити ймовірності подій виникнення дефіциту $P_д$ і переповнення $P_п$ на складі.

Для накопичення надійної статистики експеримент повторюється KK раз. За кількістю експериментів стежить лічильник експериментів k (**блоки 2, 3, 8**). Кожен експеримент триває від 0 до T_k моменту часу (**блоки 5, 7**). Лічильник часу t відраховує час від 0 до T_k з дискретністю Δt (**блок 11**).

У кожному експерименті підраховується, скільки разів на складі в результаті дії вхідних сигналів (кількість поставлених і вилучених виробів) виникає ситуація дефіциту (**блок 13**). За цим стежить лічильник D, якщо на складі виникає ситуація $Z < 0$ (**блок 13**), виходить, виникає дефіцит виробів для обробки й лічильник збільшує своє значення на 1 (**блок 16**). Якщо дефіцит не виникає $Z \geq 0$, то лічильник свого значення не міняє.

Оскільки всього може бути розглянуте $N := T_k/\Delta t$ тактів, то ймовірність виникнення дефіциту може бути приблизно оцінена частостю виникнення цих подій як $P_д := D/N$ або з обліком того, що лічильник D накопичувався в плині KK експериментів, то $P_д := D/N/KK$. Остаточно, підставляючи N: $P_д := D \cdot \Delta t/T_k/KK$ (**блоки 4, 6**). Розрахунок імовірності переповнення складу моделюється аналогічно (ураховується, що переповнення виникає, коли Z стає більше ємності складу G) (**блоки 15, 19**).

Помітимо, що оскільки на складі в реальності не може бути виробів менше нуля, то значення Z у момент виявлення цього факту повинне бути повернуте на найближчу границю, тобто: $Z := 0$ (**блок 16**). Теж саме стосується ситуації переповнення ($Z := G$) (**блок 19**).

Алгоритм представляє збій цикл за часом від 0 до T_k із кроком Δt — моделювання виробляється в часі. У кожному циклі (на кожному такті часу) перевіряється, чи лежить згенероване T_i подія в інтервалі часу $[T, T + \Delta t]$: $T \le T_i < T + \Delta t$ (**блок 12**). Якщо подія відбувається в цей момент, то визначається, який *і-ий* вхідний сигнал визначив цю подію (**блоки 22, 10**), обробляється ситуація (**блок 17 або 18**) і генерується час наступної події (**блоки 20, 21**). Перевірка на кожному такті відбувається по всім можливих вхідних і вихідних сигналах.

Блок-схема алгоритму показана на мал. 32.2.

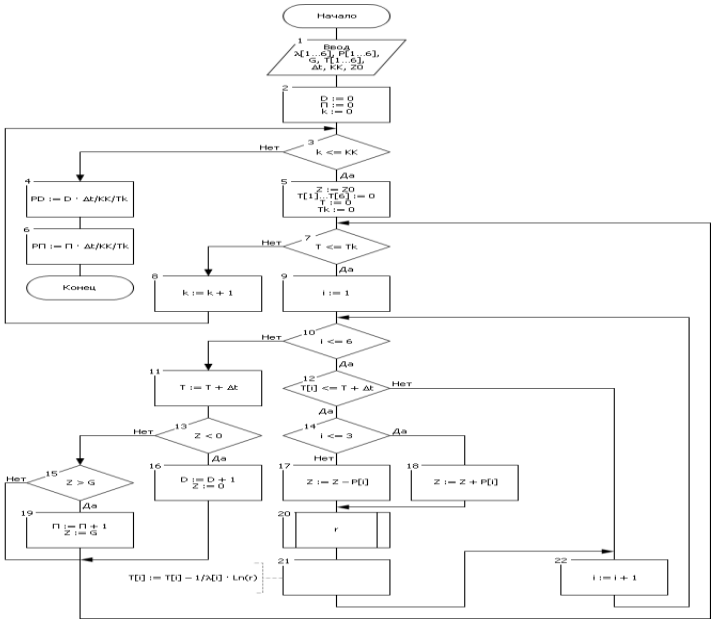

Рис. 32.2. Блок-схема алгоритму, реалізованого за принципом Δt. Приклад — моделювання виробничого складу

Принцип особливих станів

Назвемо стан, у якому звичайно перебуває система, звичайним станом. Такі стани інтересу не представляють, хоча займають більшу частину часу.

Особливі стани - це такі стани в ізольовані моменти часу, у яких характеристики системи змінюються стрибкоподібно. Для зміни стану системи потрібна певна причина, наприклад, прихід чергового вхідного сигналу. Ясно, що з погляду моделювання інтерес представляє саме зміну характеристик системи, тобто принцип жадає від нас відслідковувати моменти переходу системи з одного особливого стану в інше.

У порівнянні з попереднім випадком, ми не будемо перевіряти зміну стану системи в кожен момент часу. Виберемо серед майбутніх моментів тільки ті, у які відбувається надходження або відхід виробів зі складу, найближчий до сучасний момент часу (**блок 7** — тут і далі див. блоки на мал. 32.3). Визначивши такий момент, відразу перейдемо в нього стрибком, змінивши значення лічильника часу на величину $(-\mathrm{Ln}(r)/\lambda_i)$ (**блок 18**). В іншому, алгоритм схожий на попередній. Помітимо тільки, що цикл опитування вхідних сигналів ліквідований, оскільки **блок 7** чітко відповідає на запитання, який з *i-их* сигналів мав місце.

Все це істотно заощаджує час моделювання.

Блок-схема алгоритму показана на мал. 32.3.

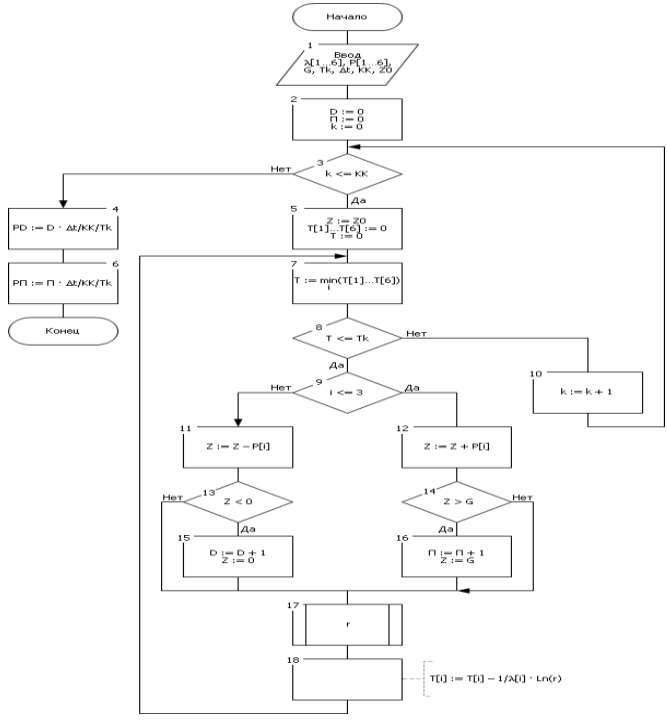
Начало
1
Ввод
λ[1..6], P[1..6],
G, Tk, Δt, KK, Z0
2
D := 0
П := 0
k := 0
3
k <= KK
Нет
Да
4
PD := D · Δt/KK/Tk
5
Z := Z0
T[1]...T[6] := 0
T := 0
6
PП := П · Δt/KK/Tk
7
T := min(T[1]...T[6])
i
Конец
8
T <= Tk
Нет
Да
9
i <= 3
Нет
Да
10
k := k + 1
11
Z := Z − P[i]
12
Z := Z + P[i]
13
Z < 0
Нет
Да
14
Z > G
Нет
Да
15
D := D + 1
Z := 0
16
П := П + 1
Z := G
17
r
18
T[i] := T[i] − 1/λ[i] · Ln(r)

Рис. 32.3. Блок-схема алгоритму, реалізованого за принципом особливих станів. Приклад — моделювання виробничого складу

Принцип послідовної проводки заявок

Принцип полягає в тому, що кожна заявка відслідковується від моменту надходження її в систему до моменту її виходу із системи. І тільки потім розглядається наступна заявка.

Розглянемо роботу алгоритму на прикладі двоканальної системи масового обслуговування заявок із двома місцями в загальній черзі з дисципліною FIFO і відмовами при переповненні черги (див. мал. 32.4), див. також лекцію 30.

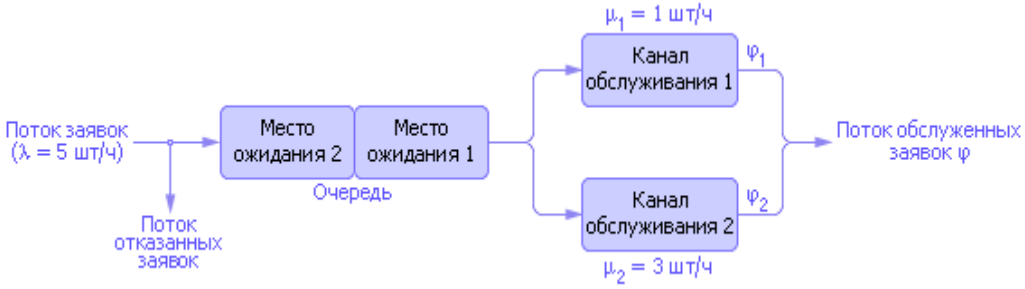

Рис. 32.4. Схема системи масового обслуговування

із двома каналами й обмеженою чергою

Позначимо: λ — інтенсивність приходу заявки; μ_i — інтенсивність обслуговування заявки.

Побудуємо алгоритм, що визначає ймовірності обслуговування заявок і відмови заявок, а також пропускну здатність системи.

Щоб зрозуміти роботу алгоритму, уявіть собі для наочності паралельні лінійки - осі часу для кожного з місць, у яких може виявитися заявка в процесі обслуговування - так, як ми це робили раніше (див. лекцію 30).

Генеруємо заявку (**блоки 3, 4** — тут і далі див. блоки на мал. 32.6). Випадковий момент часу, коли заявка прийшла в систему, дорівнює T_c. Час між двома випадковими заявками імітується формулою $\tau := -1/\lambda \cdot \mathrm{Ln}(r)$, що додається до T_c попередньої заявки $T_c := T_c - 1/\lambda \cdot \mathrm{Ln}(r)$. Урахуємо цей факт у лічильнику заявок, що *прийшли*, N **(блок** 4**).**

Послідовно порівнюємо T_c у порядку пріоритетів (**блоки 5, 6, 7, 8)** із часами звільнення T_1 каналу 1, каналу 2 — T_2, місця в черзі 1 — T_3, місця в черзі 2 — T_4. Як тільки виявляється факт того, що місце в системі вільно (див. мал. 30.5): $T_c > T_1$, або $T_c > T_2$, або $T_c > T_3$, або $T_c > T_4$, так негайно переводимо заявку на вільне місце й імітуємо обробку її на цьому місці.

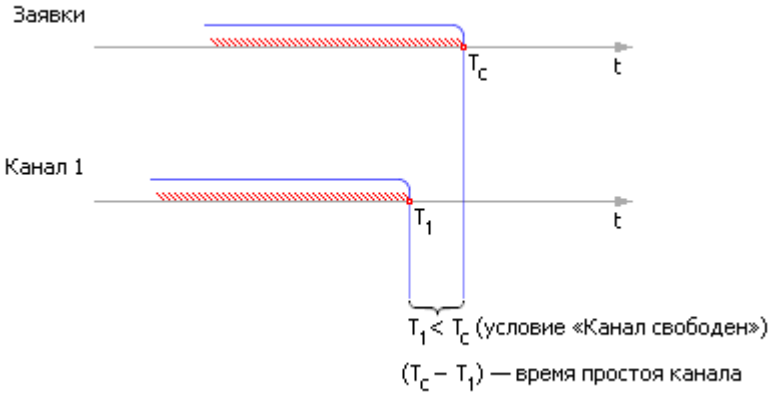

Рис. 32.5. Механізм визначення звільнення каналу (ілюстрація)

Допустимо, що звільнилося місце в першому каналі. Обробка полягає в тому, що обчислюється час простою цього каналу до приходу заявки (наприклад, $T_c - T_1$), і цей час додається в лічильник часів простою (**блок 15**). Обчислюється наступний час зміни стану каналу — модифікується змінна T_1, що вкаже нам надалі, коли знову звільниться канал. Величина T_1 змінюється на величину $\tau := -1/\mu_1 \cdot \mathrm{Ln}(r)$ — час обслуговування, відлічувану від початку обслуговування T_c. Лічильник обслужених заявок $N_{об}$ збільшується на одиницю.

Аналогічно обробка відбувається й у другому каналі, якщо заявка потрапить саме туди (**блок 14**).

Особливість обробки заявки в черзі полягає в тому, що перше місце в черзі звільняється, коли звільняється місце в одному з каналів, звичайно, заявка йде туди, де місце звільняється раніше (**блоки 5, 6**). Друге місце в черзі звільняється одночасно з першим, тому що заявка в черзі пересувається на перше місце, що звільнилося **(блок** 12**).**

Надалі алгоритм генерує в циклі наступну заявку (**блоки 3, 16**). Зупинка моделювання відбувається тоді, коли кожна лінійка буде заповнена до моменту T_k (**блок 16**). Після цього відбувається обробка статистичних результатів, накопичених у лічильниках (**блок 17**). Імовірність оцінюється частістю появи події, що обчислюється як відношення кількості подій, що з'явилися, до кількості можливих таких появ.

Блок-схема алгоритму показана на мал. 30.6.

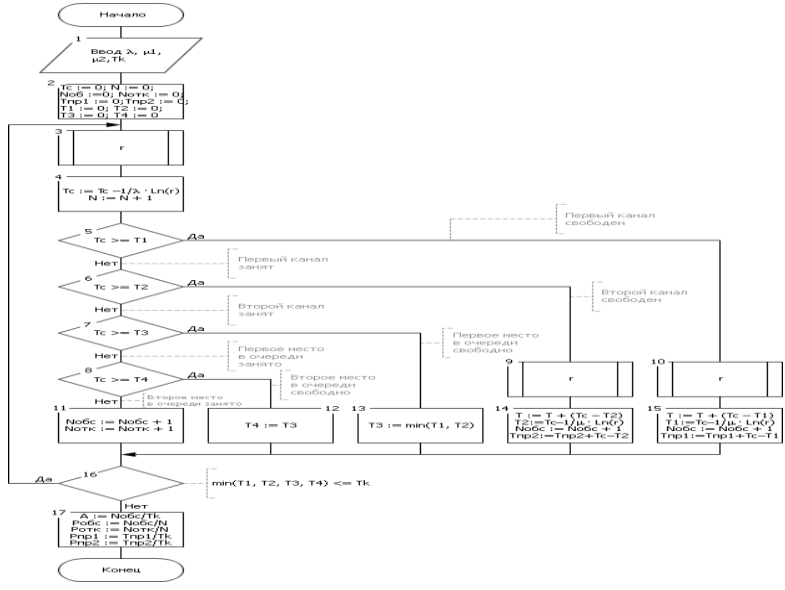

Рис. 32.6. Блок-схема алгоритму, реалізованого за принципом послідовної проводки заявок. Приклад — моделювання системи масового обслуговування

І, звичайно, треба пам'ятати, що чим більше час моделювання, тим точніше буде обчислений результат.

Має сенс нагадати ще раз, що необхідно спостерігати за поводженням статистичної характеристики, який, наприклад, є $P_{об}$. Раніше (див. лекцію 21, лекцію 34) ми відзначали, що статистична величина міняється залежно від часу спостереження. Як тільки статистична величина перестає мінятися в межах оголошеної точності, тобто крива входить у коридор, відведений їй точністю, те це сигналізує про достатність кількості експериментів.

Необхідно уважно стежити, щоб всі шукані змінні ввійшли в інтервал оголошеної точності, тільки тоді можна припинити моделювання й бути впевненим у результаті.

Для підвищення ефективності алгоритму (зменшення часу його роботи) можна відкинути нехарактерну частину реалізації - звичайно це початкова ділянка роботи системи, «вихід її на режим».

Помітимо також, що не важливо, чи маємо ми справа з однією довгою реалізацією або з більшою кількістю коротких реалізацій (у яких, звичайно, вирізана ділянка «вихід на режим»), у сумі даючих реалізацію такої ж довжини - статистичний результат буде тим же. Це міркування встановлює рівність усереднень по ансамблі реалізацій усередненням за часом.

Примітка. На практиці звичайно застосовують комбінації всіх трьох методів.

Об'єктний принцип моделювання

Як правило, алгоритми, спроектовані по першим трьох принципах, погано модернізуються. А виробнича ситуація часто міняється й вимагає відповідних моделей для знаходження раціональних рішень у процесі керування.

Таким чином, виникла необхідність у прийомах моделювання, що забезпечують незалежність складання моделей елементів складної системи від зміни задачі або структури виробництва. Такий підхід моделювання окремих об'єктів незалежно друг від друга **дозволяє збирати як завгодно складні системи без зміни їхніх складових**. Принцип об'єктного моделювання забезпечує модернізацію складних систем, подовжуючи життєвий цикл АСУ.

Приклад. СМО складається з наступних елементів, що існують самих по собі: джерело заявок, черга, канал (див. мал. 36.7). Змоделюємо їх окремо.

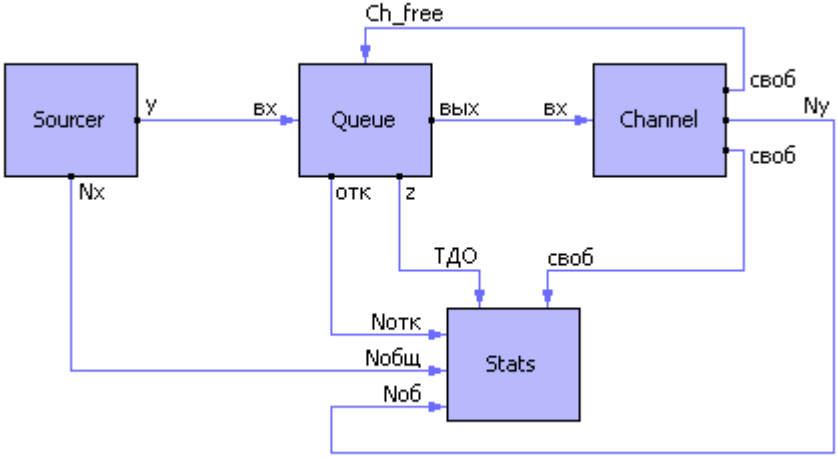

Рис. 32.7. Схема реалізації об'єкто-орієнтованого моделювання (на прикладі СМО)

1. Джерело заявок (Sourcer) генерує послідовність випадкових подій.

```
// якщо лічильник x дорівнює 0, те це момент появи заявки
 // tau — забезпечує генерацію часу між заявками в момент появи заявки
 tau := –1/λ · ln(rnd) · delta(x) + tau · not(delta(x))

// y — фиксирует факт появления заявки
y:= delta(x)

// лічильник моделирует заявочный процесс во времени, отсчитывая время между
заявками
// если лічильник x равен 0, то это момент появления заявки
// если x больше нуля, то новая заявка пока не появилась
x := x + ed(~tau – x) · dt – x · ed(x – ~tau )

// лічильник заявок
Nx := Nx + delta(~x)
```

2. Канал обслуговування (Channel).

```
// канал вільний, якщо не обробляє заявку
 своб := not(оброб)

 // своб — прапор-ознака, якщо канал вільний, те «своб» дорівнює 1.
// оброб — прапор-ознака, якщо канал зайнятий, те «оброб» дорівнює 1.
// Якщо канал був вільний, і прийшла заявка, то канал починає неї обробляти.
оброб := ed(~своб · вх + not(delta(у))

// вх — сигнал о подаче заявки на обслуживание в канале
// tau — обеспечивает генерацию необходимого времени обслуживания заявки в момент ее появления
// Новая генерация происходит только в том случае, если канал свободен и пришла заявка
// Если в канале обслуживается заявка, то новое tau не генерируется
// mu — интенсивность потока обслуживания (задается константой)
tau := –1/mu · ln(rnd) · ~своб · ~вх + tau · not(delta(~у))

своб: = not(~оброб)

// лічильник «у» моделирует заявочный процесс во времени, отсчитывая время обслуживани
// если «у» выдал 1, то значит канал выдал обслуженную заявку
```

y := y + ed(~tau – ~y) · dt – ~y · ed(~y – ~tau)

// лічильник обслуженных заявок
Ny := Ny + delta(~y)

// флаг «обработка» будет погашен, если закончится время обслуживания
оброб := not(delta(~y))
// канал будет свободен, если канал не обробатывает заявки
своб := not(~оброб)

// лічильник накапливает статистику — общее время простоя канала
ОВП := ОВП + ~своб · dt

// лічильник накапливает статистику — общее время работы канала
ОВР := ОВР + ~оброб · dt

3. Черга (Queue).

```
// «відмова» — лічильник фіксує відмова, якщо всі місця в черзі зайняті (Z >
Zm)
 // і приходить чергова заявка (вх = 1)
 // Zm — максимальна кількість місць у черзі
 // Z — поточна кількість зайнятих місць
 відмова := ~отказ + delta(~вх – 1) · ed(~Z – Zm + 1)

 // лічильник кількості зайнятих місць у черзі збільшується, якщо приходить
// заявка і є незайняті місця в черзі
 Z := Z + delta(~вх – 1) · ed(Zm – Z)

 // передаємо заявку із черги в канал
 // прапор, що фіксує звільнення місця в черзі в момент
 // звільнення каналу, якщо черга є
 // ch_free — прапор (вільний (1) або зайнятий канал обслуговування (0))
 вых := delta(~ch_free – 1) · ed(~Z)

// лічильник зменшує кількість зайнятих місць у черзі,
// після взяття заявки до каналу обслуживания
Z := Z – ~вих
```

4. Статистика (Stats).

```
Тэкс := Тэкс + 1 · dt
проп_сп := Nоб/Тэкс
Роб := Nоб/Nобщ
Ротк := Nотк/Nобщ
Тпрост1 := Тпрост1 + ~своб1 · dt
Тпрост12 := Тпрост12 + ~своб1 · ~своб2 · dt
Тпрост123 := Тпрост123 + ~своб1 · ~своб2 · ~своб3 ·
dt
ср_кол_пр_КО := Тпрост1 + Тпрост12 · 2 + Тпрост123
· 3
ср_кол_зан_КО := всего_КО – ср_кол_пр_КО
S := S + ~тек_дл_очер · dt
ср_дл_очер := ~S/Тэкс
```

Дані моделі можуть збиратися в будь-які конфігурації без зміни їхнього змісту.