УДК 004.415.3
Пех П.А., Войтович А.О.
Луцький національний технічний університет

ПРОГРАМНИЙ С++КОМПЛЕКС ДЛЯ ДОСЛІДЖЕННЯ ПРОЦЕСУ ФУНКЦІОНУВАННЯ АВТОМАТИЗОВАНИХ ЛІНІЙ ТА ЙОГО ВЕРИФІКАЦІЯ

Пех П.А., Войтович А.О. Програмний С++комплекс для дослідження процесу функціонування автоматизованих ліній та його верифікація. В статті запропоновано програмний комплекс засобами середовища C++Builder для дослідження процесу функціонування автоматизованої лінії зі стохастичним характером роботи. Запропоновані також результати верифікації цього алгоритму, які підтверджують його адекватність.
Ключові слова: C++Builder проект, автоматизована лінія, розподіл Ерланга, імітаційна модель, кібертнетичний експеримент

Пех П.А., Войтович А.О. Программный С++комплекс для исследования процесса функционирования автоматизированных линий и его верификация. В статье предложено программный комплекс средствами C++Builder для исследования процесса функционирования автоматизированной линии со стохастическим характером работы. Предложены также результаты верификации єтого алгоритма, которые подтверждают его адекватность.
Ключевые слова: C++Builder проект, входящий поток, распределение Пуассона, распределение Эрланга.

Pekh Petro, Vojtovich Andrij. Software C++ complex of an automated lines functioning process research and it's verification. The paper proposes a software complex by means C++Builder to research the process of an automated line with a stochastic work nature. Some results obtained through the realisation of cybernetic experiments series on this complex are also offered.
Keywords: C++Builder project, automated line, Erlang distribution, simulation model, cybernetic experiment.

Постановка задачі. Розглянемо процес функціонування автоматизованої лінії зі стохастичним характером функціонування, яка складається із s послідовно з'єднаних верстатів, між якими встановлені буферні пристрої певної місткості (рис. 1). Процеси надходження предметів праці (заготовок) та їх оброблення на верстатах автоматизованої лінії вважаються випадковими – а саме такими, що описуються розподілом Ерланга. Вважаються відомими (заданими) наступні параметри досліджуваної лінії:

кількість s верстатів, що входять до складу лінії;
номінальні продуктивності кожного верстату;
параметри стабільності робочих циклів (параметри Ерланга) кожного верстату;
місткості буферних пристроїв між верстатами.

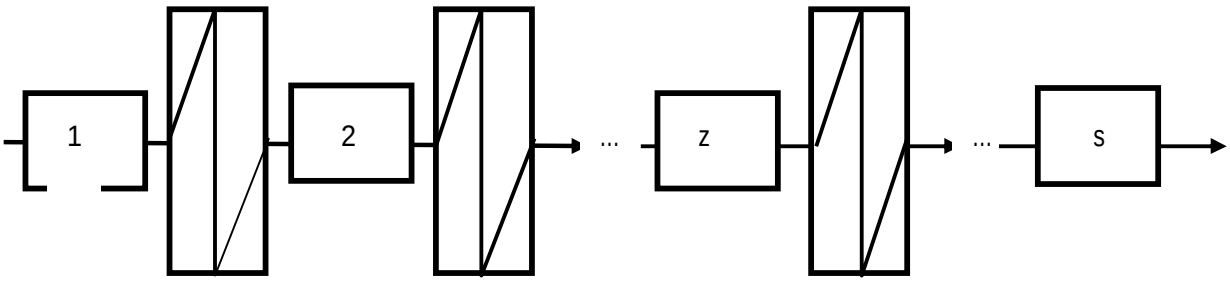

Рис.1 – Структурна схема досліджуваної автоматизованої лінії

Процес оброблення заготовок на верстатах лінії здійснюється наступним чином. Вважається, що кожна заготовка у відповідності з вимогами технологічного процесу повинна послідовно оброблятися на всіх верстатах лінії. Заготовка, оброблення якої закінчено на першому верстаті, негайно надходить на оброблення до другого верстату, якщо цей верстат вільний. Якщо ж другий верстат зайнятий обробленням попередньої заготовки, то поточна заготовка надходить на тимчасове перебування до буферного пристрою, встановленого між верстатами, і цей пристрій ще повністю не заповнений. Якщо ж буферний пристрій заповнений повністю, або він відсутній як такий, то перший верстат буде простоювати до моменту звільнення другого верстата. Навпаки, якщо на момент закінчення оброблення поточної заготовки на першому верстаті буферний пристрій буде порожнім, а другий верстат вже закінчить оброблення попередньої заготовки, то другий верстат також певний час простоюватиме. Виникають ситуації, коли простоює то один, то другий верстати. Аналогічно взаємодіють всі верстати та буферні пристрої лінії. Отже, залежно від параметрів лінії і

технологічного процесу, зазначених вище, час чистої роботи верстатів, а, значить, і ефективність їх використання, буде змінюватися.

Метою дослідження є розроблення засобами C++Builder середовища програмного комплексу для імітаційного моделювання процесу функціонування автоматизованої лінії і встановлення з допомогою цього комплексу залежності показників якості функціонування лінії, зокрема, коефіцієнтів використання робочого часу кожного верстату та лінії в цілому від величин вхідних параметрів, перелічених вище. Варто зазначити, що задача імітаційного моделювання процесу функціонування автоматизованих ліній розглядалася раніше багатьма дослідниками [1,2], в тому числі і одним із авторів цієї статті [3], однак розроблення імітаційної моделі засобами C++Builder середовища здійснюється вперше, і в цьому ми вбачаємо новизну дослідження. На нашу думку, це дозволяє покращити імітаційну модель і значно розширити діапазон досліджуваних проблем.

Основна частина. Планом передбачалося проведення досліджень у три етапи:

— розроблення програмного комплексу засобами середовища C++Builder для імітаційного моделювання автоматизованих ліній;
— перевірка адекватності отриманої імітаційної моделі;
— проведення досліджень роботи ліній на створеній імітаційній моделі.

У даній статті викладені результати перших двох етапів.

Основою програмного комплексу є реалізована на консолі середовища C++Builder програма імітаційного моделювання процесу функціонування автоматизованих ліній/ Нижче наведено код цієї програми, а далі пояснюються її окремі аспекти.

```
// Імітаційне моделювання роботи автоматизованої лінії
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
#include <math.h>
#include <windows.h>
#define s 3    //s-кількість послідовно з'єднаних верстатів
using namespace std;
int i,j;
int z;     //номер поточного верстата та буферного пристрою за ними
int v;     //номер поточного досліду
int w=5;   //кількість дослідів в експерименті
int r;     //номер поточної заготовки
int q=500; //кількість заготовок, оброблення яких імітується
           //в одному досліді
double c;  //згенерована тривалість робочого циклу  верстата
double c1;
double tmin;
double sum;
double suml;
int flag;
int ozn;
double h[s+1];   //масив номінальних продуктивностей верстатів
double cc[s+1];  //масив тривалостей робочих циклів верстатів
int k[s+1];      //масив параметрів стабільностей циклів верстатів
double t[3*s+2]; //робочий масив моментів часу, з допомогою яких
                 //відслідковуються проходження заготовкою усіх
                 //верстатів та буферних пристроїв лінії
int n[s+1];      //масив, що відслідковує поточну кількість
                    //заготовок у кожному буферному пристрої
int m[s+1];         //масив, що містить номінальні місткості заготовок
                    //кожного буферного пристрою
double tb[s+1][50]; //робочий масив значень моментів часу вивільнення
                    //z-го буферного пристрою від заготовок,
                    //що очікували в ньому на оброблення
double av[s+2][10]; //масив середніх значень коефіцієнтів використання
                    //робочого часу верстатів у кожному досліді
```

```
                    //(не більше 10 дослідів в експерименті)

int main() {
  SetConsoleCP(1251);        //Підключення кодів букв
  SetConsoleOutputCP(1251);  //українського алфавіту

  //srand(time(0));          //Запуск генератора випадкових чисел

  //Задання значень номінальних продуктивностей верстатів
  for (z=1; z<=s; z++) {
    cout<<"\n Введіть значення номінальної продуктивності "<<z
        <<" -го верстата: h["<<z<<"]= ";
    cin>>h[z];
  }

  // Задання значень параметрів стабільності робочих циклів верстатів
  for (z=1; z<=s; z++) {
    cout<<"\n Введіть значення параметра стабільності робочого циклу "
        <<z<<" -го верстата: k["<<z<<"]= ";
    cin>>k[z];
  }

  // Задання значень місткості буферних пристроїв
  for (z=1; z<s; z++) {
    cout<<"\n Введіть значення місткості буферного пристрою після "
        <<z<<" -го верстата: m["<<z<<"]= ";
    cin>>m[z];
  }
  m[s]=0;
  suml=0;

  //Імітаційне моделювання роботи лінії.
  for (v=1; v<=w; v++) { // Дослід номер v. Всього дослідів - w.

   // Очищення масиву t[i] від попередньої інформації
   for (i=1; i<=3*s+1; i++) t[i]=0;
   t[3*s+2]=0;

   // Очищення масиву tb[z][i] від попередньої інформації
   for (z=1; z<=s; z++)
    for (i=1; i<=m[z]; i++) tb[z][i]=0;

    //Очищення масиву n[i] від попередньої інформації
    for(i=1; i<=s-1; i++) n[i]=0;

    //Імітаційне моделювання обробки заготовки номер r
    for (r = 1; r<=q; r++) { //Всього заготовок у досліді - q.
     //Імітаційне моделювання обробки заготовки на верстаті номер z.
     for (z=1; z<=s; z++) {// Всього верстатів  у лінії - s.
      c = rand()% 32767;   // Генерування випадкового числа
                            // в діапазоні [0 .. 32767]
      if (c<10) continue;
      c=-h[z]*log(c/32767);
      cc[z]=c;


      // Розрахунки по z-ому верстату
      t[3*z]=t[3*z-2]+c;   //Момент виходу заготовки із z-го верстата
```

```
t[3*z-1]=t[3*z-1]+c; //Сумарний час роботи z-го верстата

//Якщо m[z]=0, то маємо жорстке з'єднання верстатів
//Якщо m[z]>0, то маємо гнучке з'єднання верстатів
//Аналіз типу з'єднання верстатів та з'ясування,
//яка з шести можливих ситуацій має місце
if ((m[z]==0) && (t[3*z]<=t[3*z+1]))        flag=1;
 else if ((m[z]==0) && (t[3*z]> t[3*z+1]))  flag=2;

if ((m[z]>0) && (n[z]>0)) {
 i=1;
 while (i<=m[z]) {
  if ((tb[z][i]>0) && (tb[z][i]<=t[3*z])) {
   tb[z][i]=0;
   n[z]=n[z]-1;
  }
  i++;
 }
}

if ((m[z]>0) && (n[z]<m[z]) && (t[3*z]<t[3*z+1]))        flag=3;
 else if ((m[z]>0) && (n[z]<m[z]) && (t[3*z]==t[3*z+1])) flag=4;
  else if ((m[z]>0) && (n[z]<m[z]) && (t[3*z]>t[3*z+1])) flag=5;
   else if ((m[z]>0) && (n[z]==m[z]))                    flag=6;

switch (flag) {
 case 1: t[3*z]=t[3*z+1];
         t[3*z-2]=t[3*z+1];
         break;

 case 2: t[3*z-2]=t[3*z];
         t[3*z+1]=t[3*z];
         break;

 case 3: i=1;
         ozn=0;
         while ((ozn==0) && (i<=m[z])) {
          if (tb[z][i]==0) {
           tb[z][i]=t[3*z+1];
           n[z]=n[z]+1;
           ozn=1;
          }
          i++;
         }
         t[3*z-2]=t[3*z];
         break;
 case 4: t[3*z-2]=t[3*z];
         break;
 case 5: t[3*z-2]=t[3*z];
         t[3*z+1]=t[3*z];
         break;

 //Визначаємо момент часу tmin,
 //коли буфер звільниться від z-ої заготовки
 case 6: tmin=tb[z][1];//Присвоюємо  tmin початкове значення
 j=1;                  //tb[z][1] та запам'ятовуємо його номер 1
 for (i = 2; i<=m[z]; i++)
   if (tb[z][i]<=tmin) { //Пошук дійсного значення моменту часу
```

```
                                  //tmin і його номера j
            tmin=tb[z][i];
            j=i;
          }
        tb[z][j]=0;     //Вилучення з масиву tb[z][j] значення,
                        //що дорівнює tmin=tb[z][i]
        n[z]=n[z]-1;    //Зменшення на одиницю кількості заготовок
                        //у буферному пристрої
        t[3*z]=tmin;    //Коригування моменту виходу заготовки t[3*z]
                        //з z-го верстата
        t[3*z-2]=tmin;   //Коригування моменту входу наступної заготовки
                        //t[3*z-2] в z-ий верстат
        break;

       } // Кінець оператора switch

      } //Кінець циклу по імітації роботи z-го верстата
        //і z-го буферного пристрою

     } // Кінець циклу по імітації обробки заготовки номер r
     sum=0;
     for (z=1; z<=s; z++) {
      av[z][v]=t[3*z-1]/t[3*z];
      sum=sum+av[z][v];
     }
     av[s+1][v]=sum/s;
     suml=suml+av[s+1][v];
     cout<<"\n Середнє значення КВРЧ лінії в "<<v<<
           "-ому досліді дорівнює "<<av[s+1][v]<<"\n";
     //getch();
     //cout<<"\n Номер досліду  v=" << v << endl;

   } // Кінець циклу по реалізації досліду номер v

   suml=suml/w;
   cout<<"\n Середнє значення КВРЧ лінії в експерименті з "<<w<<
         " дослідів дорівнює "<<suml<<"\n";
   getch();
   return 0;
}
```

Дамо детальні коментарі до наведеного вище коду програми імітаційного моделювання процесу функціонування автоматизованої лінії. Оскільки нами розроблено консольний варіант програмного комплексу, то оголошення змінних та масивів виконано у розділі глобальних змінних. Зрозуміло, перш за все мають бути задані параметри лінії, робота якої досліджується. Саме тому тіло головної функції починається із введення значень наступних параметрів:

— номінальних продуктивностей верстатів лінії;`h[z],z=1..s` усіх верстатів лінії;
— стабільності робочих циклів верстатів лінії `k[z],z=1..s`;
— місткостей буферних пристроїв лінії `m[z],z=1..s-1.`

Зазначимо, що один з основних параметрів лінії – кількість верстатів у програмі задано з допомогою макроконстанти `s.` У програмі це забезпечується директивою

```
#define s 3
```

тобто в даний момент досліджується лінія з трьох верстатів.

Для того, щоб визначити величини коефіцієнтів використання робочого часу кожного верстата та лінії в цілому, передбачається проведення кібернетичного експерименту. Цей експеримент передбачає проведення серії дослідів, у кожному з яких значення параметрів лінії не змінюються. Кількість дослідів в експерименті задається змінною `w.` У нашому випадку `w=5,` тобто експеримент передбачає проведення п'яти дослідів. Під дослідом ми розуміємо імітацію оброблення на верстатах

лінії певної кількості заготовок, яка задається значенням змінної `q`. У нашому випадку `q=500`, тобто у кожному досліді імітується оброблення 500 заготовок. Отже, за результатами моделювання ми маємо можливість визначити значення коефіцієнтів використання робочого часу кожного верстата та лінії в цілому в кожному досліді, для зберігання яких передбачено двомірний масив `av[z][v]` (z=1..s; v=1..w), а тоді і середні значення цих параметрів за результатами усіх дослідів. Наприклад, `av[1][1]` – це значення коефіцієнта використання робочого часу першого верстата за результатами першого досліду.

Розглянемо більш детально ту частину програми, де здійснюється проведення заготовок, тобто реалізується окремий дослід. В ході моделювання порядковому номеру z послідовно присвоюються значення від 1 до s. Відповідно, проведення однієї заготовки може бути розділена на s кроків. Один крок моделювання полягає в наступному.

Для фіксованого значення z формується тривалість робочого циклу c, розподілена за законом Ерланга з параметрами h[z] і k[z], за допомогою датчиків ерлангівських чисел, який базується на стандартному датчику квазірівномірно розподілених псевдо випадкових чисел. Використовуючи згенероване значення тривалості робочого циклу c, визначаємо сумарний час роботи z-го верстату `t[3*z]` і час закінчення оброблення на ньому чергової заготовки `t[3*z]`:

```
t[3*z-1]=t[3*z-1]+c;
t[3*z]=t[3*z-2]+c.
```

Далі перевіряється, яка з наступних шести умов має місце:

`m[z]=0 ∪ t[3z]<= t[3z+1]`, якій відповідає значення прапорця `flag=1;`
`m[z]=0 ∪ t[3z]>t[3z+1]`, якій відповідає значення прапорця `flag=2;`
`m[z]>0 ∪ n[z]<m[z] ∪ t[3z]<t[3z+1]`, якій відповідає значення прапорця `flag=3;`
`m[z]>0 ∪ n[z]<m[z] ∪ t[3z]= t[3z+1]`, якій відповідає значення прапорця `flag=4;`
`m[z]>0 ∪ n[z]<m[z] ∪ t[3z]> t[3z+1]`, якій відповідає значення прапорця `flag=5;`
`m[z]>0 ∪ n[z]=m[z]`, якій відповідає значення прапорця `flag=6;`

У випадку виконання першої умови `(flag=1)` моделюється жорстке з'єднання `z`-го і `(z+1)`-го верстатів, причому `t[3z]<= t[3z+1]`. Простоює протягом певного часу `z`-ий верстат. Визначаються нові значення таких величин:

```
t[3*z]=t[3*z+1];
t[3*z-2]=t[3*z+1].
```

У випадку виконання другої умови `(flag=2)` моделюється жорстке з'єднання `z`-го і `(z+1)`-го верстатів, причому `t[3z]>t[3z+1]`. Простоює протягом певного часу `(z+1)`-ий верстат Визначаються нові значення таких величин:

```
t[3*z-2]=t[3*z];
t[3*z+1]=t[3*z].
```

Перед перевіркою наступних чотирьох умов, які моделюють жорстке з'єднання z-го і (z+1)-го верстатів, масив `tb[z][i]` очищується від значень, що менші або дорівнюють значенню моменту часу `t[3z]` з одночасним зменшенням кожного разу на одиницю значення масиву `n[z]`.

У випадку виконання третьої умови `(flag=3)` моделюється гнучке з'єднання z-го і (z+1)-го верстатів. Буферний пристрій повністю не заповнений, причому `t[3z]<t[3z+1]`. Жоден з верстатів не простоює, оскільки є можливість використання буферного пристрою. У масив `tb[z][i]` записується значення `t[3*z+1]`, а до значення масиву `n[z]` додається одиниця. Визначається нове значення величини:

```
t[3*z-2]=t[3*z+1].
```

У випадку виконання четвертої умови `(flag=4)` моделюється гнучке з'єднання z-го і (z+1)-го верстатів. Буферний пристрій повністю не заповнений, причому `t[3z]= t[3z+1]`. Жоден з верстатів не простоює. Визначається нове значення величини:

```
t[3*z-2]=t[3*z+1].
```

У випадку виконання п'ятої умови `(flag=5)` моделюється гнучке з'єднання z-го і (z+1)-го верстатів. Буферний пристрій повністю не заповнений, причому `t[3z]> t[3z+1].` Жоден з верстатів не простоює. Визначаються нові значення таких величин:

```
t[3*z-2]=t[3*z];
t[3*z-2]=t[3*z].
```

У випадку виконання шостої умови `(flag=6)` моделюється гнучке з'єднання z-го і (z+1)-го верстатів. Буферний пристрій повністю заповнений. Простоює протягом певного часу `z`-ий верстат. У масиві `tb[z][i]` знаходимо момент часу, коли буферний пристрій зможе прийняти на зберігання z–у заготовку. Цим моментом буде мінімальне значення масиву `tb[z][i]`, яке позначимо `tmin.` Це значення вилучаємо з масиву `tb[z][i],` і від значення масиву n[z] віднімаємо одиницю. Визначаються нові значення таких величин:

```
t[3*z]=tmin;
t[3*z-2]=tmin.
```

Отже, проведення однієї заготовки полягає в реалізації s кроків моделювання при послідовній зміні параметра z від 1 до s.

Верифікацію алгоритму виконуємо шляхом порівняння значень коефіцієнтів використання робочого часу лінії, отриманих теоретичним шляхом зі значеннями, отриманими шляхом імітаційного моделювання. Зокрема, відомо, що коефіцієнт використання робочого часу лінії з двох верстатів у разі, якщо час оброблення заготовок має експоненціальний розподіл, визначається за формулою:

$$\rho=(M=2)/(M+3),$$

де M – місткість буферних пристроїв.

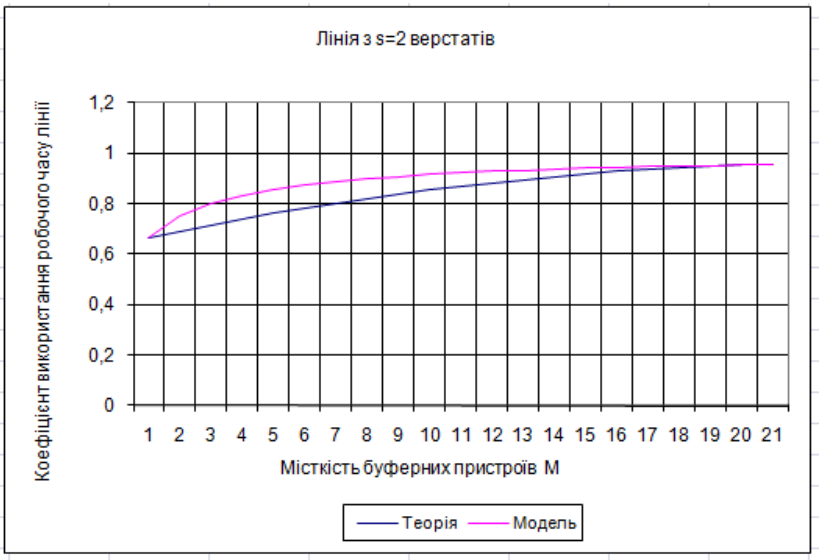

Рис.2 – Теоретична та емпірична залежності коефіцієнта використання робочого часу автоматизованої лінії з двох верстатів від місткості буферних пристроїв M

В роботі розроблено C++Builder програмний комплекс, у якому сучасними програмними засобами розв'язується актуальна задача імітаційного моделювання процесу функціонування автоматизованих ліній.

1. Д.Л. Дудюк та інші. Моделювання та оптимізація об'єктів і систем керування: Навчальний посібник для студентів вищих навчальних закладів. –К.:ІЗМН, 1998.-248 с.
2. Дудюк Дмитро та інші. Елементи теорії автоматичних ліній Навчальний посібник для студентів вищих навчальних закладів. –Київ-Львів:ІЗМН, 1998.-192 с.
3. Дудюк Д.Л., Пех П.А. Моделирование автоматизированных линий последовательного и параллельного агрегатирования методом Монте-Карло //Лесн. хоз-во, лесн., бум. и деревообраб. пром-сть: Респ. межведомственный науч.-техн. сб. - Киев: Будівельник, 1977. - Вып. 8. -С. 17-20.