МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

МАРИЙСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ИСЭ

**Курсовая работа**

**«Контрольные системы управления»**

Выполнила: студентка ЭФ гр. ПИЭ-32

Короткова А.М.

Научный руководитель:

доцент, к.э.н. Еклашева О.В.

Йошкар-Ола

2008

1. Планирование этапов производства (sf-2 algo)

1.1 Цель

Определить увеличится ли производительность завода при новом распределении этапов производства.

1.2 Описание

В машинном цехе расположены 3 вида станков: А, В и С. В этот цех поступают заказы, причем сначала в рабочую область, а оттуда в плановый отдел, где учитывается критерий наименьшей очереди заказов, распределенных по станкам. 50% заказов могут обрабатываться всеми станками, 30% только станками типа В или С, остальные 20% только станками типа С.

Но иногда обработка заказа машиной типа С занимает много времени. Тогда предполагается поступление минимального количества заказов из рабочей области в производственный цех. Это позволяет более рационально распределить заказы и сократить время на их обработку. Для этого в имитационную модель добавляются еще 2 элемента, один из которых отвечает за входящий поток заказов, а второй – за их сортировку по типам и за их направление в соответствующий буфер.

1.3 Операции

Рассмотрим структуру данной имитационной модели. Начнем по порядку.

Рабочий день состоит из 8 часов.

 **Settings | Time representation**

**60** units make 1 **minute**

**60** minute make 1 **hour**

**8** hour make 1 **day**

Время имитации модели равно одному дню.

**Settings | Simulate| Stop time =** 1day

Далее рассмотрим элементы, из которых построена модель и связи между ними.

Элементы – Inou\_1, Inou\_19 служат для генерации входящего потока заказов.

Элементы – Buff\_2, Buff\_20 сортируют заказы по типам и направляют их в соответствующий буфер.

Элементы – Buff \_3, Buff\_4, Buff\_5, Buff\_6, Buff\_7, Buff\_8 содержат очередь продуктов, принятых к исполнению.

Элементы с Mach\_9 по Mach\_14 отражают машины, обрабатывающие заказы.

Элементы – Buff\_15, Buff\_16 содержат уже готовые заказы, которые затем передают на выход.

Элементы – Inou\_17, Inou\_18 являются выходом для выполненных заказов.

По условию задачи, в заводской цех поступают заказы разных типов, из которых 50% обрабатывается всеми машинами, 30% - машинами типа В или С и 20% - машинами типа С. Поступление заказов задаётся эмпирическим распределением. Отсюда имеем:

 **Model | Elements | Job parameters**

*Element 1: Trigger on exit* = product[C]:=empirical [1]

*Element 19: Trigger on exit* = product[C]:=empirical [2]

*Заказ поступает каждые 5 единиц времени.* Это время, через которое Элементы – Inou\_1 и Inou\_19 генерируют входящий поток заказов. Задаётся пуассоновским законом распределения:

 **Model | Elements | Job parameters**

*Element 1: Time* = 5.0 Neg.Exp

*Element 19: Time* = 5.0 Neg.Exp

Так как заказы поступают случайным образом, то пропускная способность элементов Inou\_1 и Inou\_19 не тождественна. На выходе каждого генератора входящего потока заказам присваиваются свои номера или коды, которые определяют их путь. Например, 1, 2, 3 при выходе заказов из Inou\_1 и 4, 5, 6 – из Inou\_19. Это говорит о том, что заказы, вышедшие из первых трех буферов обрабатываются первыми тремя машинами, из последних трех буферов – последними тремя машинами. Все преобразования в каждом элементе InOut происходят в соотношении 50:30:20.

Рассмотрим элементы Buff\_2 и Buff\_20. Их вместимость (Capacity) равна 20 и 40 ед. соответственно. Сходство данных элементов в том, что при отправке первого заказа они создают короткую очередь, состоящую из этого продукта. Но условием выхода заказа из добавочного Buff\_20 является наличие в нем как минимум 3 заказов одновременно:

 **Model | Elements | Element parameters**

*Element 20: Exit condition* = elqueue[E]>3

На этом этапе заказы, поступившие в цех через Inou\_1 и Inou\_19 сортируются по типам и направляются в буфер в соответствии с таблицей «plan», где ячейка берется с номером из списка 3..5, 6..8, а номер столбца равен коду продукта, являющегося в текущем элементе первым в порядке наименьшей очереди:

 **Model | Elements | Stage parameters**

*Element 2: Send to* = select 1 with plan[L,product[E,1]]=1 from 3..5 order -(elqueue[L]+elqueue[L+6])

*Element 20: Send to* = select 1 with plan[L,product[E,1]]=1 from 6..8 order -(elqueue[L]+elqueue[L+6])

L – значение в списке

product[E,1] – продукт (заказ), являющийся в текущем элементе первым в очереди

elqueue[L] – текущее число продуктов (заказов) в элементе

Следует отметить, что строки в таблице соответствуют машинам (станкам), а столбцы - типу заказа (вид продукта). Если значение ячейки ij=1, то это означает, что заказ j может быть обслужен машиной i, если же ij=0 - тогда не может.

Например, элемент, находящийся на пересечении 5 строки и 3 столбца, говорит о том, что заказ может обрабатываться только машиной типа С.

Элементы Buff\_3, Buff\_4, Buff\_5, Buff\_6, Buff\_7, Buff\_8 вмещают по 16 заказов каждый (Capacity=16) и направляют их в соответствующие машины (Mach\_9 - Mach\_14):

 **Model | Elements | Stage parameters**

*Element 3: Send to* = E+6

*Element 4: Send to* = E+6

*Element 5: Send to* = E+6

*Element 6: Send to* = E+6

*Element 7: Send to* = E+6

*Element 8: Send to* = E+6

Данные выражения можно пояснить следующим образом: если к номеру элемента, т.е. буфера прибавить 6, то получим номер машины, которая будет обрабатывать заказ, вышедший из этого буфера.

Таким образом, распределяются все заказы, которые нужно обработать той или иной машиной.

*Т.к. среднее время необходимое для выполнения заказа равно 10 единицам времени(включая поломку, и т.п.) и задаётся пуассоновским законом распределения, то оно может сильно меняться.*

Это время стоит во времени обработки в элементах Mach\_9 - Mach\_14.

 **Model | Elements | Element parameters**

*Element 9: Time =* 10.0 Neg.Exp

*Element 10: Time =*10.0 Neg.Exp

*Element 11: Time =* 10.0 Neg.Exp

*Element 12: Time =* 10.0 Neg.Exp

*Element 13: Time =* 10.0 Neg.Exp

*Element 14: Time =* 10.0 Neg.Exp

*В настоящий момент времени доступны все машины.* У всех 6 машин стоит галочка в поле Element active.

 **Model | Elements | Element parameters | More**

*Element 9: Element active* ***=*** On

*Element 10: Element active* ***=*** On

*Element 11: Element active* ***=*** On

*Element 12: Element active* ***=*** On

*Element 13: Element active* ***=*** On

*Element 14: Element active* ***=*** On

После обработки заказы, прошедшие через первые 3 машины (Mach\_9, Mach\_10, Mach\_11) направляются в Buff\_15, через оставшиеся 3 (Mach\_12, Mach\_13, Mach\_14) - в Buff\_16:

 **Model | Elements | Stage parameters**

*Element 9: Send to* = 15

*Element 10: Send to* = 15

*Element 11: Send to* = 15

*Element 12: Send to* = 16

*Element 13: Send to* = 16

*Element 14: Send to* = 16

Элементы Buff\_15, Buff\_16 содержат по 14 обработанных заказов и направляют их на выход: Inou\_17, Inou\_18 соответственно.

Рассмотрим задание финансовых данных. Все финансовые расчёты в модели выполняются в долларах:

 **Settings| Finances | Currency =** $

И, наконец, у нас имеются две динамические иконки WIP, которые отображают производительность системы. Рассмотрим их. Начнем сверху.

Верхняя иконка показывает суммирование длины очереди во всех элементах с 3 по 5 и с 9 по 11. Это отражает сумма переменных elqueue@sum[3..5] и elqueue@sum[9..11].

Нижняя иконка показывает суммирование длины очереди во всех элементах с 6 по 8 и с 12 по 14, т.е. здесь представлена сумма переменных elqueue@sum[6..8] и elqueue@sum[12..14].

Максимальное значение, которое могут принимать данные выражения, равно 10.

В ShowFlow знак @ является знаком присвоения в TLI выражениях. Другими словами, данное выражение суммирует объём работы для всех активных элементов.

Теперь запустим модель.

 **Simulate | Single run | Start**

(или используйте скоростную кнопку запуска)

При запуске модель работает как одна комплексная система, результат которой отображен в динамических иконках.

В течение моделирования можно переключаться между анимацией и статистикой, переключая Animation в Simulation Control Window.

Лучше всего прогнать модель при маленькой скорости, чтобы просмотреть и сравнить результаты, представленные в динамических иконках.

Поскольку заказы поступают случайным образом, то для получения наиболее общего результата необходимо большее количество прогонов.

2. Моделирование бизнес-процесса реинженеринга в офисе

2.1 Цель

Определить, насколько эффективно будет новое распределение этапов обработки заявок.

2.2 Описание

В некоторый офис поступают заявки, которые сначала регистрируются, затем поступают на рассмотрение, принимаются или отвергаются, и наконец выполняются, если принято такое решение. Также представлены 2 последовательности этапов обработки заявок. Результаты обработок представлены в динамических иконках.

2.3 Операции

Рассмотрим структуру данной имитационной модели.

В неделе 5 рабочих дней.

 **Settings | Time representation**

**60** units make 1 **hour**

**8** hour make 1 **day**

**5** day make 1 **week**

Время, в течение которого имитируется модель с начала ее запуска равно 12 неделям.

**Settings | Simulate| Stop time =** 12 week

Рассмотрим элементы, из которых построена модель и связи между ними.

Элементы – Inou\_1, Inou\_24 служат для генерации входящего потока клиентов.

Элементы – Oueu\_2, Oueu\_25 содержат очередь поступивших клиентов с заявками.

Элементы – Task\_3, Task\_26 предназначены для отображения приема клиента с заявкой.

Элементы – Oueu\_4, Oueu\_27 содержат заявки для регистрации.

Элементы – Task\_5, Task\_6, Task\_28, Task\_29 отражают регистрацию заявки.

Элементы – Oueu\_7, Oueu\_30 содержат уже зарегистрированные заявки.

Элементы – Task\_8, Task\_31 отражают сортировку заявок для дальнейшего рассмотрения.

Элементы – Oueu\_9, Oueu\_11, Oueu\_13, Oueu\_37 содержат заявки, ожидающие своего рассмотрения.

Элементы – Task\_10, Task\_12, Task\_14, Task\_38, Task\_39, Task\_40 отражают рассмотрение принятой заявки.

Элементы – Oueu\_15, Oueu\_32 содержат заявки, которые будут либо приняты, либо отвергнуты.

Элементы –Task\_16, Task\_33 принимают либо отвергают заявки.

Элементы – Oueu\_17, Oueu\_41 содержат заявки для регистрации об их выполнении.

Элементы – Task\_18, Task\_42 отражают регистрацию о выполнении заявок.

Элементы – Oueu\_19, Oueu\_34 содержат рассмотренные заявки.

Элементы – Task\_20, Task\_21, Task\_35, Task\_36 отражают хранение рассмотренных заявок.

Элементы – Oueu\_22, Oueu\_43 содержат заявки с регистрацией об ее выполнении.

Элементы – Inou\_23, Inou\_44 являются выходом для заявок, прошедших обработку.

По условию задачи нам даны 2 различных последовательности этапов обработки заявок в одном и том же офисе. Заявки поступают в тот или иной процесс обработки:

 **Model | Elements | Stage parameters**

*Element 1: Send to* = Select 2 from 2, 25

Причем для каждой из них в какой-то момент фиксируется ее время:

 **Model | Elements | Job parameters**

*Element 2: Trigger on entry* leadtime[C]:=time

*Element 25: Trigger on entry=* leadtime[C]:=time

Очередь клиентов с заявками может состоять лишь из 16 человек:

 **Model | Elements | Element parameters**

*Element 2: Capacity* = 16

*Element 25: Capacity* = 16

Время приема заявки от клиента задается пуассоновским законом распределения. Оно равно 6 и 4 единицам времени:

 **Model | Elements | Job parameters**

*Element 3: Time* = 6.0 Neg.Exp

*Element 26: Time* = 4.0 Neg.Exp

Выражение в элементах Task\_3 и Task\_26, а именно:

 **Model | Elements | Job parameters**

*Element 3: Trigger on exit* = product[C]:=2

*Element 26: Trigger on exit* = product[C]:=2

обозначает, что заявка принята.

Далее заявка направляется на регистрацию в какой-либо регистрирующий отдел:

 **Model | Elements | Stage parameters**

*Element 4: Send to* = Select 1 from 5, 6

*Element 27: Send to* = Select 1 from 28, 29

При этом количество заявок, ожидающих регистрацию, может достигать 24:

 **Model | Elements | Element parameters**

*Element 4: Capacity* = 24

*Element 27: Capacity* = 24

Заявка регистрируется в течение 13 единиц времени:

 **Model | Elements | Job parameters**

*Element 5: Time* = 13.0 Neg.Exp

*Element 6: Time* = 13.0 Neg.Exp

*Element 28: Time* = 13.0 Neg.Exp

*Element 29: Time* = 13.0 Neg.Exp

Об ее регистрации говорит выражение:

 **Model | Elements | Job parameters**

*Element 5: Trigger on exit* = product[C]:=3

*Element 6: Trigger on exit* = product[C]:=3

*Element 28: Trigger on exit* = product[C]:=3

*Element 29: Trigger on exit* = product[C]:=3

Т.е. заявка уже зарегистрирована.

Далее зарегистрированные заявки сортируются в течение 10 единиц времени:

 **Model | Elements | Job parameters**

*Element 5: Time* = 10.0 Neg.Exp

*Element 6: Time* = 10.0 Neg.Exp

а отсортированные:

 **Model | Elements | Job parameters**

*Element 8: Trigger on exit* = product[C]:=4

*Element 31: Trigger on exit* = product[C]:=4

рассматриваются за 15 в офисе с текущей последовательностью этапов обработки заявок и за 35 в офисе с запланированной последовательностью этапов обработки.

Выражение:

 **Model | Elements | Job parameters**

*Element 14: Trigger on exit* = product[C]:=5

*Element 40: Trigger on exit* = product[C]:=5

обозначает, что заявка уже рассмотрена.

Следует заметить, что количество уже зарегистрированных заявок в очереди не должно превышать 8:

 **Model | Elements | Element parameters**

*Element 7: Capacity* = 8

*Element 30: Capacity* = 8

Количество заявок, ожидающих своего рассмотрения в офисе с текущей последовательностью этапов обработки – 24:

 **Model | Elements | Element parameters**

*Element 9: Capacity* = 24

*Element 11: Capacity* = 24

*Element 13: Capacity* = 24

а в офисе с запланированной последовательностью этапов обработки – 104 заявки:

 **Model | Elements | Element parameters**

*Element 37: Capacity* = 104

которые далее отправляются на рассмотрение:

 **Model | Elements | Stage parameters**

*Element 9: Send to* = 10

*Element 11: Send to* = 12

*Element 13: Send to* = 14

*Element 37: Send to* = Select 1 from 38, 39, 40

Очередь, состоящая из заявок, которые примут или отвергнут, одновременно может содержать 5 таких заявок:

 **Model | Elements | Element parameters**

*Element 15: Capacity* = 5

*Element 32: Capacity* = 5

Процесс принятия решения занимает лишь 10 единиц времени:

 **Model | Elements | Job parameters**

*Element 16: Time* = 10.0 Neg.Exp

Заявки, подлежащие регистрации о выполнении, поступают с кодом продукта, равным 7:

 **Model | Elements | Job parameters**

*Element 17: Trigger on entry* = product[C]:=7

*Element 41: Trigger on entry* = product[C]:=7

Одновременно в очереди могут находиться 16 таких заявок:

 **Model | Elements | Element parameters**

*Element 17 Capacity* = 16

*Element 41: Capacity* = 16

После регистрации о выполнении в течение 18 единиц времени:

 **Model | Elements | Job parameters**

*Element 18: Time* = 18.0 Neg.Exp

*Element 42: Time* = 18.0 Neg.Exp

80 % заявок поступают к заявкам, выполнение которых уже зарегистрировано, а 20 % пока остаются в очереди (офис с текущей последовательностью этапов обработки):

 **Model | Elements | Stage parameters**

*Element 18: Send to* = Select 1 from Bernoulli [80, 20, 19]

Закон распределения Бернулли.

В офисе с запланированной последовательностью этапов обработки заявки с регистрацией о выполнении поступают в хранилище данных заявок:

 **Model | Elements | Stage parameters**

*Element 42: Send to* = 43

Рассмотренные заявки поступают в очередь с кодом продукта 6:

 **Model | Elements | Job parameters**

*Element 19: Trigger on entry* = product[C]:=6

*Element 34: Trigger on entry* = product[C]:=6

Одновременно в очереди могут находиться 12 таких заявок:

 **Model | Elements | Element parameters**

*Element 19 Capacity* = 12

*Element 34: Capacity* = 12

Затем эти заявки поступают в хранилище:

 **Model | Elements | Stage parameters**

*Element 19: Send to* = Select 1 from 20, 21

*Element 34: Send to* = Select 1 from 35, 36

В хранилище заявки находяться в течение 100 единиц времени:

 **Model | Elements | Job parameters**

*Element 20: Time* = 100.0 Neg.Exp

*Element 21: Time* = 100.0 Neg.Exp

*Element 35: Time* = 100.0 Neg.Exp

*Element 36: Time* = 100.0 Neg.Exp

При выходе всех заявок с регистрацией о выполнении их суммарное время нахождения в системе обнуляется:

 **Model | Elements | Job parameters**

*Element 22: Trigger on exit* = leadsum[1,1]:= leadsum[1,1]+time-leadtime[C]

*Element 43: Trigger on exit* = leadsum[2,1]:= leadsum[2,1]+time-leadtime[C]

Данные выражения ссылаются на таблицу leadsum

где leadsum[1,1] и leadsum[2,1] – суммарное время нахождения всех заявок в системе,

time – текущее время нажождения заявки в системе,

leadtime[C] – время входа заявки в систему.

Теперь рассмотрим динамические иконки.

Иконки под названием Applications Completed отображают объем выпуска заявок с регистрацией об их выполнении (produced[22] и produced[42]). Максимальное число заявок в обоих случаях равно 100000.

Иконки 1и 4 с текстомWIP T1..T2 (current) и WIP T1..T2 (prop) показывают соответственно суммирование длины очереди во всех элементах с 9 по 14 (elqueue@sum[9..14]) и с 37 по 40 (elqueue@sum[37..40]). Максимальное значение суммы в обоих случаях равно 75.

В ShowFlow знак @ является знаком присвоения в TLI выражениях. Другими словами, данное выражение суммирует объём работы для всех активных элементов.

Оставшиеся 2 иконки 2 и 3 с соответсвующим текстом Leadtime (current) и Leadtime (prop) отображают среднее время обработки заявки.

Отметим, что current – это текущая последовательность этапов обработки заявок, а prop – запланированная.

Теперь запустим модель.

 **Simulate | Single run | Start**

(или используйте скоростную кнопку запуска)

В результате получим максимальный объем выпуска обработанных заявок при минимальной затрате времени, учитывая новое распределение этапов обработки. Следовательно, наиболее эффективно использовать запланированный бизнес-процесс.

В течение моделирования можно переключаться между анимацией и статистикой, переключая Animation в Simulation Control Window.