Кулик Павел, ИС-241. Объектный подход к моделированию. Нечётный вариант.

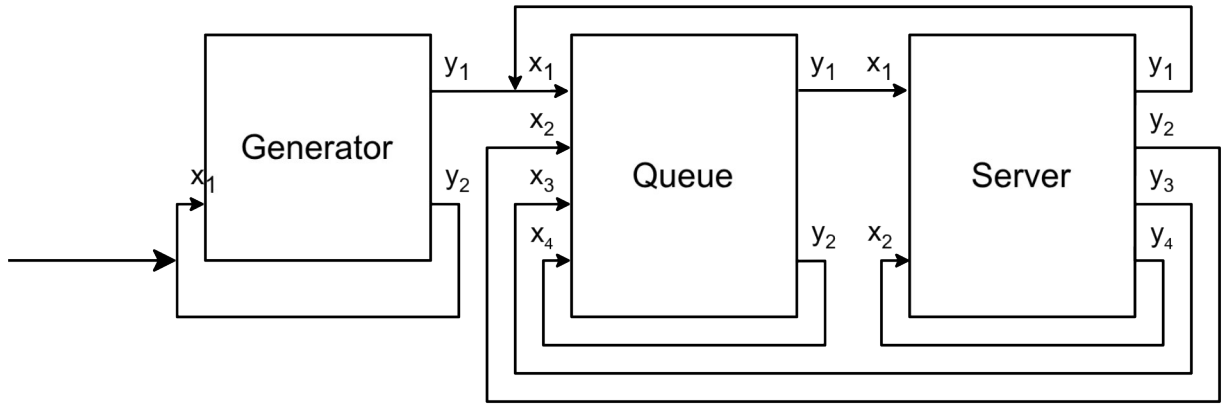
Постановка задачи

Требуется построить диаграмму объектов, описать состояния и специальные методы всех объектов. Должна быть создана имитация системы массового обслуживания, содержащей очередь требований, в которую через равные промежутки времени t_1 приходят новые требования и прибор, который обслуживает по одному требованию за раз за фиксированное время t_2. В ходе работы прибор ломается, отработав время t_{work} и приостанавливает работу до восстановления, которое происходит через время t_{repair}. Если прибор сломался в момент обслуживания, то требование, которое обслуживалось в этот момент, возвращается обратно в очередь.

Релизация

Релизация была перенесена на следующую страницу для более удобного просмотра.

В начале моделирования 1 раз подаётся сигнал извне, после чего система работает сама. Также, я решил, что выходы просто выдают сигналы без особой семантики, а входы эти сигналы просто считывают. Ну и соответственно, при считывании сигналов объект реагирует в соответствие со внутренними правилами, то есть сначала F в ответ на сигнал изменяет состояние, а G и H реагируют при изменении состояния в соответствие со своей задачей: G планирует новое изменение состояния через временной промежуток, а H посылает сигнал по условию.




```
F: X+S->S
G: S+T->S
H: S->Y

Generator:
     S: int counter = 0;
     F: if(x_1) {counter++;}
     G: if(counter == 1) {with_delay(t1, {counter--;})}
     H: if(counter == 0) {send(y1); send(y2);}
Queue:
     S: int queue_len = 0; need_send = false;
     F:
     if(x_1) {queue_len++; need_send = true;}
     if(x_2) {queue_len--;}
     if(x_3 && queue_len > 0) {need_send = true;}
     if(x_4) {need_send = false;}
     G: empty
     H: if(need_send) {send(y_1); send(y_2);}
Server:
     S: busy = false; broken = false; need_return = false;
     F:
     if(x_1 && !busy && !broken) {busy = true;}
     if(x_2) {need_return = false;}
     G:
     if(busy) {with_delay(t2, {busy = false;})}
     if(!broken) {
         with_delay(t_work, {
             broken = true; busy = false; need_return = true;
         })}
     if(broken) {with_delay(t_repair, {broken = false;})}
     H:
     if(!busy) {send(y_3);}
     if(busy) {send(y_2);}
     if(need_return) {send(y_4); send(y_1);}
```