В такой ситуации может быть полезна система, способная моделировать процесс защиты конфиденциальных работ в зависимости от наличия или отсутствия всех перечисленных факторов. В связи с этим разработаем такую систему на основе использования сетей Петри с помощью языка программирования Java Script.

Для начала в программе требуется разработать холст для моделирования и отрисовки объектов, затем добавить возможность масштабирования холста, чтобы удобно было строить сеть на основе Сетей Петри:

svgCanvas.addEventListener('pointerdown', function (event) {

    if (!event.target.classList.contains('draggable') || !event.target.classList.contains('draggable-group'))

    {

      drag2 = true;

      offset = { x: event.offsetX, y: event.offsetY };

    }

});

svgCanvas.addEventListener('pointermove', function (event) {

    if (drag2) {

        var tx = event.offsetX - offset.x;

        var ty = event.offsetY - offset.y;

        offset = {

            x: event.offsetX,

            y: event.offsetY

        };

        matrix.preMultiplySelf(new DOMMatrix()

            .translateSelf(tx, ty));

        viewPort.style.transform = matrix.toString();

    }

});

svgCanvas.addEventListener('pointerup', function (event) {

    drag2 = false;

});

svgCanvas.addEventListener('wheel', function (event) {

    var zoom = event.deltaY > 0 ? -1 : 1;

    var scale = 1 + factor \* zoom;

    offset = {

        x: event.offsetX,

        y: event.offsetY

    };

    matrix.preMultiplySelf(new DOMMatrix()

        .translateSelf(offset.x, offset.y)

        .scaleSelf(scale, scale)

        .translateSelf(-offset.x, -offset.y));

    viewPort.style.transform = matrix.toString();

});

Так как сеть Петри представляет из себя множество позиций, переходов и связей между ними, добавим возможность создания этих элементов:

P

Byid('GroupBTN2\_P').onclick = function()

    {

      var id = ++OptionsCircle.count;

      Byid('general\_group2').innerHTML += ' <g class=" draggable-group">' + Circle(200,200,id) + Text1('P' + id,200,140,'textP','TextP'+id) + '</g>';

      AllElement['P'+id] = {};

    }

T

Byid('GroupBTN2\_T').onclick = function()

    {

      var id = ++OptionsRect.count;

      Byid('general\_group2').innerHTML += '<g class=" draggable-group">' + Rect(200,200,id) + Text1('T' + id,216,180,'textT','TextT'+id) + '</g>';

      AllElement['T'+id] = {};

    }

Line

function CreateLine(FinalEL,Type, idStartEL)

    {

      if(Type == FinalEL.tagName || FinalEL.tagName == 'svg'){return}

      if(FinalEL.parentNode.attributes.transform)

      {

        var Elem = FinalEL.parentNode.attributes.transform.value;

        FinX = +Elem.substring(10,Elem.length).split(' ')[0] + 200;

        FinY = Elem.substring(10,Elem.length).split(' ')[1];

        FinY = +FinY.substring(0,FinY.length-1) + 200;

      }else

      {

        FinX = 200;

        FinY = 200

      }

      var biasY = OptionsRect.height.split('p')[0] / 2;

      var biasX = OptionsRect.width.split('p')[0] / 2;

      var CountLink = 0;

      for(key in Links)

      {

        if(Links[key].Start == idStartEL && Links[key].Final == FinalEL.id)

        {

          CountLink +=1;

        }

      }

      CountLink = CountLink \*10;

      if(Type == 'circle')

      {

        Byid('general\_group\_Links').innerHTML += Line(startX,startY + CountLink,(FinX+biasX),(FinY+biasY) + CountLink,'green',LinkCount);

      }

      if(Type == 'rect')

      {

        Byid('general\_group\_Links').innerHTML += Line((startX),startY+ CountLink,FinX,FinY + CountLink,'black',LinkCount);

      }

      Links['Link'+LinkCount] = {};

      Links['Link'+LinkCount]['Start'] = idStartEL;

      Links['Link'+LinkCount]['Final'] = FinalEL.id;

      Links['Link'+LinkCount]['Speed'] = {};

      Links['Link'+LinkCount]['Speed'][0] = SpeedM[0];

      Links['Link'+LinkCount]['Speed'][1] = SpeedM[1];

      Links['Link'+LinkCount]['Speed'][2] = SpeedM[2];

      AllElement[idStartEL]['Link'+LinkCount] = {}

      AllElement[FinalEL.id]['Link'+LinkCount] = {}

      AllElement[idStartEL]['Link'+LinkCount]['Type'] = 'set';

      AllElement[FinalEL.id]['Link'+LinkCount]['Type'] = 'get';

      console.log(Links)

      LinkCount +=1;

    }

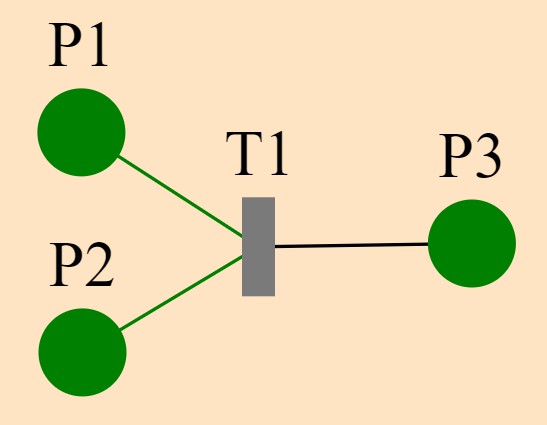


Рисунок - Связь между позициями и переходами.

Для удобства, добавим возможность менять имена позиций и переходов, при помощи одного нажатия на название элемента:

function SetNewText(elem)

    {

      var text =  prompt('Введите название элемента',elem.innerHTML);

      if(text)

      {

        elem.innerHTML = text;

      }

    }

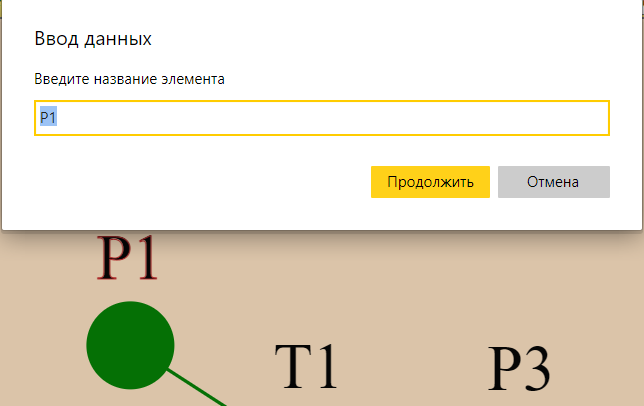


Рисунок - Наименование элемента.

Далее, добавим возможность перемещения элементов и перетаскивание связей:

function startDrag(evt) {

      Type = evt.path[0].tagName;

      if(Type != 'svg' && Type != 'Metka' && Type != 'line')

      {

        StartEL =  evt.path[0].id;

        if(FocusElement)

        {

            Byid(FocusElement).style = 'stroke: none;';

        }

              Byid(StartEL).style = 'stroke: rgb(206, 46, 46);';

              FocusElement = StartEL;

      }else

      {

        try

        {

          Byid(FocusElement).style = 'stroke: none;';

          FocusElement = '';

        }catch

        {

          FocusElement = '';

        }

      }

      if(evt.path[0].id[0] == 'M'){return}

      if (evt.target.classList.contains('draggable'))

        {

            selectedElement = evt.target;

            initialiseDragging(evt);

        } else if (evt.target.parentNode.classList.contains('draggable-group'))

        {

                  selectedElement = evt.target.parentNode;

                  initialiseDragging(evt);

                  if(evt.ctrlKey)

                  {

                      var Elem = evt.toElement.parentNode.attributes.transform.value;

                      startX = +Elem.substring(10,Elem.length).split(' ')[0] + 200;

                      str = Elem.substring(10,Elem.length).split(' ')[1];

                      startY = +str.substring(0,str.length-1) + 200

                      if(Type == 'rect')

                      {

                        startY += OptionsRect.height.split('p')[0] / 2;

                        startX += OptionsRect.width.split('p')[0] / 2;

                      }

                      Byid('general\_group2').innerHTML += TempLine(startX,startY,startX,startY,'black');

                      TempLineF = true;

                  }

        }

    }

function drag(evt) {

      if (selectedElement) {

        drag2 = false;

        if(!evt.ctrlKey)

        {

        evt.preventDefault();

        var coord = getMousePosition(evt);

        transform.setTranslate(coord.x - offset.x, coord.y - offset.y);

        GoodLink(StartEL,coord.x - offset.x+200,coord.y - offset.y+200);

        }else

        {

          evt.preventDefault();

          var coord = getMousePosition(evt);

          Byid('Lin1').setAttributeNS(null, "x2", coord.x - offset.x +200);

          if(Type == 'rect')

          {

            Byid('Lin1').setAttributeNS(null, "y2", coord.y - offset.y +200 + OptionsRect.height.split('p')[0] / 2);

          }else

          {

            Byid('Lin1').setAttributeNS(null, "y2", coord.y - offset.y +200);

          }

          FinalEL = document.elementFromPoint(evt.clientX, evt.clientY);

        }

      }

    }

function endDrag(evt) {

      selectedElement = false;

      if(evt.ctrlKey)

      {

        removeElement('Lin1');

        CreateLine(FinalEL,Type, StartEL);

        StartEL = '';

      }

      if(TempLineF)

      removeElement('Lin1');

      TempLineF = false;

    }

Добавим возможность создание метки в позициях. Как только метка помещается в позицию, кружок становится больше в диаметре, для наглядности начальной точки работы:

function CreateMet(Elem,classMetki,client)

{

  var g = Elem.parentNode;

  var color;

  var count = 0;

  switch(classMetki)

  {

    case 0: color = 'blue'; break

    case 1: color = 'red'; break

    case 2: color = 'rgb(19,19,19)'; break

  }

  for(key in AllElement[Elem.id])

  {

    if(AllElement[Elem.id][key]['Type'] == 'set' || AllElement[Elem.id][key]['Type'] == 'bosy')

    {

      if(AllElement[Elem.id][key]['Metka'])

      {

        AllElement[Elem.id][key]['Metka'].push('M'+'C'+classMetki + '$' +CountMet);

      }else

      {

        AllElement[Elem.id][key]['Metka'] = [];

        AllElement[Elem.id][key]['Metka'].push('M'+'C'+classMetki + '$' +CountMet);

      }

      if(count == 0)

      count = AllElement[Elem.id][key]['Metka'].length;

      if(count == 2 && Byid(Elem.id).style.fill != 'rgb(119, 0, 255)'){

        Byid(Elem.id).style.fill = 'rgb(119, 0, 255)';

        alert('Внимание образовался затор!!');

      }

      if(count == 1)

      {

        if(Elem.type == 'cirle')

        Byid(Elem.id).style.fill = 'green';

      }

      CreateObM(Elem,'C'+classMetki + '$' +CountMet,color,g,count);

      CountMet +=1;

    }

  }

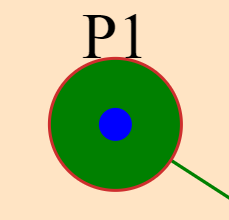


Рисунок - Метка внутри позиции.

Перейдем к созданию логики программы на основе Сетей Петри. Учитываем все правила активации переходов и перемещения меток.

После написания логики программы, добавляем возможность задания и редактирования скорости прохождения метки по связям в миллисекундах, для удобства моделирования и экономии времени:

function SetNewSpeedLink(id)

{

  var n;

  var f = false;

  for(key in SpeedM)

  {

    n = prompt('Введите время прохождения дуги(ms)'+'\n' + 'Для ' + (+key + 1) +' проекта' ,Links['Link'+id].Speed[key]);

    if(!n){return}

    if(n == 'delete')

    {

      f= true;

      break;

    }

    Links['Link'+id].Speed[key] = +n;

  }

}

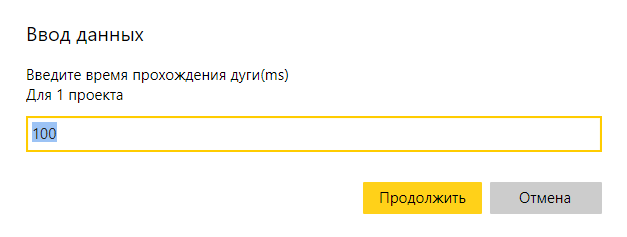


Рисунок - Ввод данных времени прохождения метки по связи.

Добавляем в модель создание нескольких меток, чтобы иметь возможность смоделировать работу объекта, когда идет нагрузка в несколько проектов:

Тот же самый код что выше

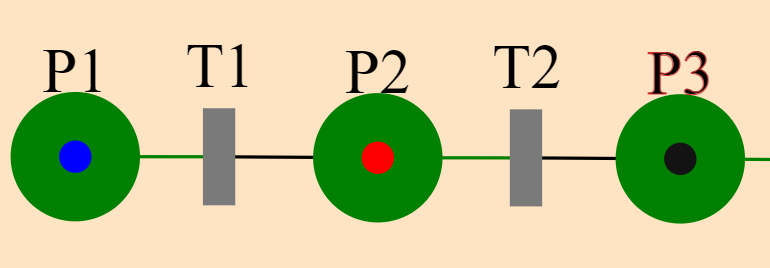


Рисунок - Добавление нескольких меток в модели.

В интерфейс программы добавляем глобальную настройку скорости меток по цвету. Связи, в которые не были внесены временные задержки, подчиняются глобальным настройкам скорости:

Byid('btnSpeed').onclick = function()

{

  for(key in SpeedM)

  SpeedM[key] = +(Byid('in'+key).value)

}

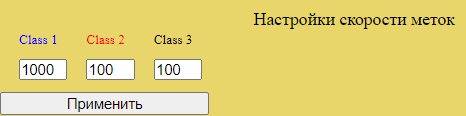


Рисунок - Возможность задания глобальной скорости меток.

При несрабатывании позиции, возникает уведомление о необходимости ее диагностики. В случае неисправности, сеть прекращает работу. Приведем в пример модель светофора, на которой не загорелся «Красный» свет:

 if(f)

  {

    AllElement[Links['Link'+id].Final]['Link'+id]['Type'] = 'delete';

    return;

  }

  if(AllElement[Links['Link'+id].Final]['Link'+id]['Type'] == 'delete')

  {

    AllElement[Links['Link'+id].Final]['Link'+id]['Type'] = 'get';

    AllElement[Links['Link'+id].Start]['Link'+id]['Type'] = 'set';

    AllElement[Links['Link'+id].Start].style.fill = 'green';

  }

 if(AllElement[FinElement.Final][key]['Type'] == 'delete')

          {

            AllElement[FinElement.Final][key]['Type']= 'get';

            var mes = Byid(FinElement.Start).parentNode.childNodes[1].innerHTML;

            Byid(FinElement.Start).style.fill = 'red';

            AllElement[FinElement.Start][key]['Type'] = 'delete';

            alert('Внимание не сработала позиция '+mes );

            return;

          }

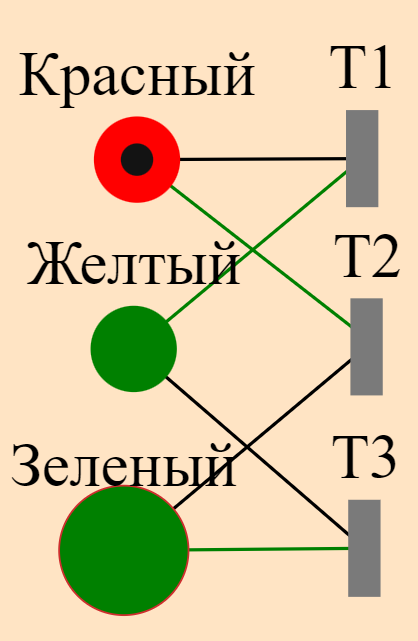
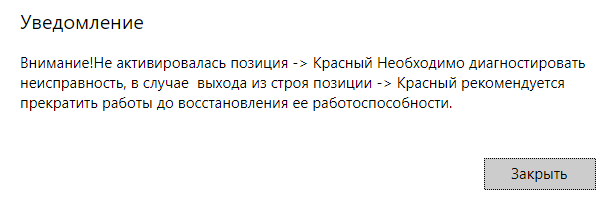


Рисунок - Пример несрабатывания позиции.

При моделировании сети с нагрузкой в несколько проектов, в ней возникают, так называемые, «узкие» места – ситуация, когда в модели метки догоняют друг друга и создают очередь в прохождении через позиции, связи и переходы. Для наглядности, места, где создается затор, выделяем фиолетовым цветом.

 if(count == 2 && Byid(Elem.id).style.fill != 'rgb(119, 0, 255)'){

        Byid(Elem.id).style.fill = 'rgb(119, 0, 255)';

        alert('Внимание образовался затор!!');

      }

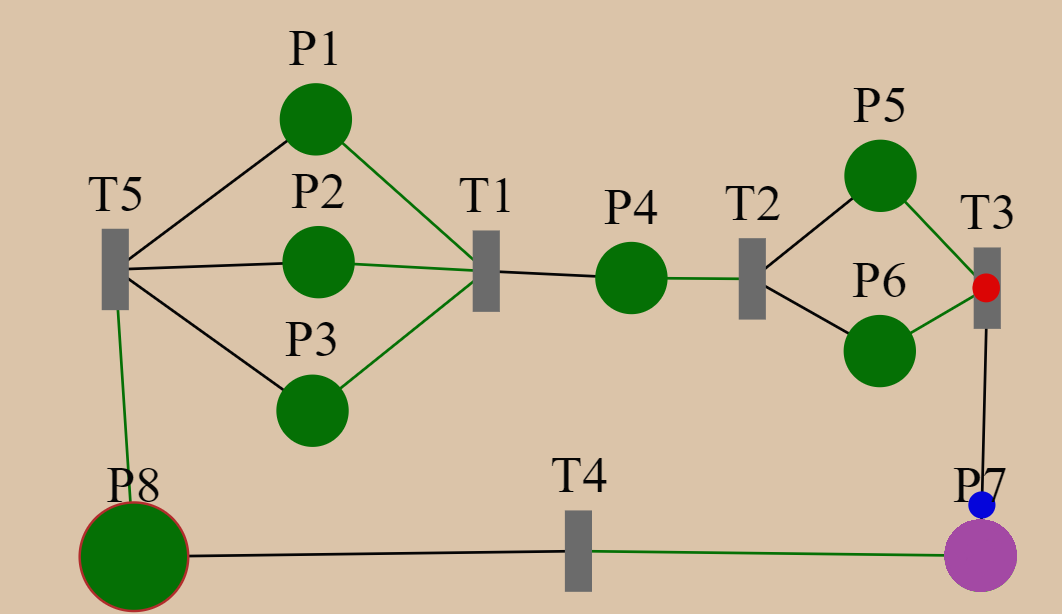


Рисунок - Пример образования узкого места.

Необходимо добавить рекомендации для решения проблемы образования узкого места. Добавляем в программу уведомление с рекомендациями решения:

Тоже самое

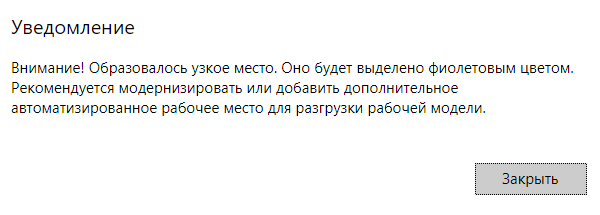


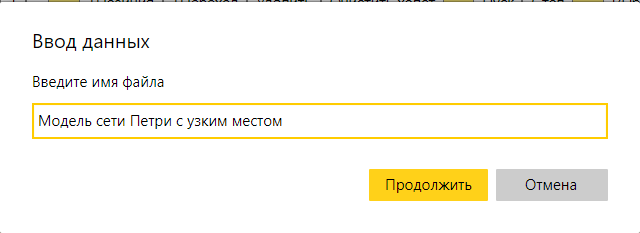
Рисунок - Пример рекомендации при образовании узкого места

В заключение к программированию ПО необходимо добавить возможность сохранения со всеми настройками модели в файл, для последующей его загрузки в программу. Так как модели конфиденциальных работ на основе сетей Петри получаются очень объёмными, то пользователю будет неудобно каждый раз выстраивать ее вновь:

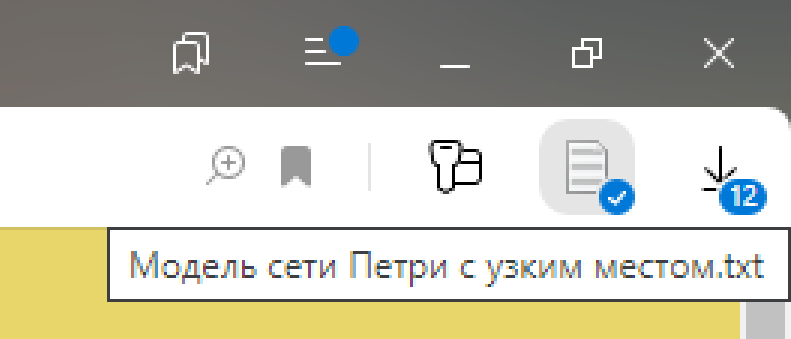
1. Нажимаем кнопку «Сохранить»:



2. Нам предлагается ввести имя файла, в который сохранится данная модель:



3. Модель сохраняется в папку «Загрузки» в формате txt:



Save

Byid('SaveFile').onclick = function()

  {

    var GeneralSTR = JSON.stringify(AllElement) + '\n' + '&';

    GeneralSTR += JSON.stringify(Links) + '\n' + '&';

    GeneralSTR += OptionsCircle.count + '\n' + '&';

    GeneralSTR += OptionsRect.count + '\n' + '&';

    GeneralSTR += LinkCount+ '\n' + '&';

    GeneralSTR += JSON.stringify(matrix) + '\n' + '&';

    GeneralSTR += Byid('general\_group2').outerHTML;

    Save(GeneralSTR)

    function Save(str)

    {

      var a = document.createElement("a");

      Byid('alink').innerHTML = a;

      a.style = "display: none";

      var blob = new Blob([str],{type:"text/plain"});

      var url = window.URL.createObjectURL(blob);

      a.href= url;

      var name = prompt('Введите имя файла','Модель сети Петри');

      a.download = name+'.txt';

      a.id = "adw";

      a.click();

      window.URL.revokeObjectURL(url);

    }

  }

Load

function showFile(input)

  {

      let file = input.files[0];

      let reader = new FileReader();

      reader.readAsText(file);

      reader.onload = function()

          {

              var result = reader.result.split('&');

              AllElement = JSON.parse(result[0]);

              Links = JSON.parse(result[1]);

              OptionsCircle.count = +result[2];

              OptionsRect.count = +result[3];

              LinkCount = +result[4];

              Tempmatrix = JSON.parse(result[5])

            matrix = Tempmatrix;

              Byid('general\_group').innerHTML = result[6];

          }

      reader.onerror = function()

          {

              console.log(reader.error);

          };

  }