บทที่ 2
วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ในการวิจัยเรื่อง การพัฒนาภาษาไพธอนสำหรับเขียนแบบก่อสร้างชิ้นส่วนโครงสร้างคอนกรีตเสริมเหล็ก ผู้วิจัยได้ศึกษาทฤษฎี และงานวิจัยที่เกี่ยวข้องเพื่อเป็นแนวทางและสร้างกรอบแนวคิดในการวิจัยตามลำดับดังนี้

1. ทฤษฎีที่เกี่ยวข้อง

1.1. ภาษา SVG

SVG หรือ Scalale Vector Graphics เป็นการประยุกต์ใช้งานภาษา XML เพื่อให้สามารถแสดงข้อมูลของภาพกราฟฟิกสองมิติ จะประกอบด้วยส่วนวัตถุกราฟฟิก(Graphics objects) และส่วนโครงสร้างเอกสาร(Document Structure)

ในปัจจุบัน SVG ได้เป็นมาตรฐานของ HTML5 ดังนั้นโปรแกรมเว็บบราวเซอร์จะต้องสามารถมีความสามารถในการดูไฟล์ SVG ได้(ในขณะที่ดำเนินการวิจัยจะใช้ด้วยโปรแกรมเว็บบราวเซอร์ Mozilla FireFox) และโปรแกรมเขียนภาพหรือโปรแกรม CAD ในปัจจุบันก็จะสามารถส่งออก(Export) เป็นไฟล์รูป SVG ได้

ประโยชน์ของ SVG คือความละเอียดของการแสดงผลเมื่อทำการขยายดูรายละเอียดจะมีความคมชัดกว่าภาพแบบราสเตอร์ซึ่งเมื่อดูขยายจะเห็นเป็นลายแตกขอบ ดังแสดงตัวอย่างในภาพและขนาดของไฟล์ก็จะมีขนาดเล็กลง

ก) ภาพแบบราสเตอร์ ข) ภาพแบบเวคเตอร์ด้วย SVG

ภาพที่ 1 เปรียบการแสดงผลแบบราสเตอร์ และแบบเวคเตอร์

1.2. รูปแบบคำสั่งในภาษา SVG

SVG จัดเป็นเอกสารประเภท XML อย่างหนึ่ง การประกาศใช้งานในกรณีที่ต้องการใช้งานกับเว็บเพจจะต้องมีการประเภทชนิดของเอกสารก่อน และอีกส่วนคือส่วนของตัวเอกสาร SVG จะประกาศด้วย <svg> และปิดด้วย </svg> ระหว่างส่วนประกาศจะบรรจุดชิ้นส่วนกราฟฟิกต่าง ๆ เช่น เส้นตรง วงกลมเป็นต้น ในการประกาศส่วน <svg> จะต้องมีการประกาศตัวแปรเป็นความกว้าง และความสูงของภาพเอสวีจี และประกาศส่วนแสดงภาพด้วยคำสั่ง viewBox หน่วยความจะระบุไว้ ณ ตอนนี้ ในงานวิจัยนี้เลือกใช้หน่วยเป็น มม ในการทำงาน แล้วครั้งต่อไปที่ใช้งานจะถือว่าค่าตัวเลขทั้งหมดจะมีหน่วยเป็น มม ยกเว้นมีประกาศหน่วยเพิ่มเติมต่อท้ายเป็นอย่างอื่น

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg  xmlns="http://www.w3.org/2000/svg
xmlns:xlink="http://www.w3.org/1999/xlink"
   width="210mm"  height="297mm" version="1.1"   viewBox="0 0 210 297">

</svg>
```

ประกาศชนิดของเอกสาร

โครงสร้างเอกสาร SVG

คำสั่งที่ใช้ในการเขียนภาพกราฟิกต่าง ๆ จะแสดงไว้ใน

ตารางที่ 1 เช่น เส้นตรง วงกลม สี่เหลี่ยม และข้อความรายละเอียดเพิ่มเติมของคำสั่ง SVG หาศึกษาเพิ่มเติมได้ที่เว็บไซท์ของ W3C

ตารางที่ 1 รูปร่างเรขาคณิตกับคำสั่ง SVG

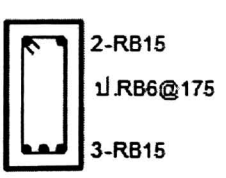
รูปร่าง	คำสั่ง
	<line x1="10" y1="0" y2="21" x2="20" />
	<circle cx="0" cy="0" r="0.375" stroke="black" />
	<rect " x="0" y="0" width="190" height="25" fill="white" />
2-RB12	<text " x="8.65" y="1.25 font-size="4.5 font-family="Browallia New" " text-anchor="start" stroke="none" fill="black">2-RB12 </text>

ตัวอย่างการสร้างหน้าตัดคานด้วยคำสั่งของภาษา SVG จะประกอบด้วยคำสั่ง SVG ต่าง ๆ เช่น line circle text จะมีการจัดกลุ่มเพื่อให้ง่ายต่อการแก้ไขด้วยโปรแกรม Inkscape ดังแสดง

รหัสคำสั่งภาษา SVG

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg style="stroke-linejoin:round; stroke:black; stroke-width:0.5pt; text-anchor:middle; fill:none"
xmlns="http://www.w3.org/2000/svg" font-family="Helvetica, Arial, FreeSans, Sans, sans, sans-serif"
height="297mm" width="210mm" version="1.1" xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0
210 297">

<rect y="10" width="190" height="277" x="10" />

<g id="BeamSection_B1" transform="translate(30,60)">
   <rect y="0" width="10" fill="white" x="0" height="20" />
   <g id="BeamRebarSVG_bottom" transform="translate(2.675,17.325)">
      <g id="rebar3">
         <circle cy="0" cx="0" r="0.375" fill="black" />
         <circle cy="0" cx="4.65" r="0.375" fill="black" />
         <circle cy="0" cx="2.325" r="0.375" fill="black" />
      </g>
      <text font-size="4.5" text-anchor="start" stroke="none" y="1.25" x="8.65"
          font-family="Browallia New" fill="black">3-RB15</text>
   </g>

   <g id="BeamRebarSVG_top" transform="translate(2.675,2.675)">
      <g id="rebar2">
         <circle cy="0" cx="0" r="0.375" fill="black" />
         <circle cy="0" cx="4.65" r="0.375" fill="black" />
      </g>
      <text font-size="4.5" text-anchor="start" stroke="none" y="1.25" x="8.65"
         font-family="Browallia New" fill="black">2-RB15</text>
   </g>

   <g id="StirrupSVG" transform="translate(2.15,2.15)">
      <text font-size="4.5" text-anchor="start" stroke="none" y="7.85" x="9.7"
          font-family="Browallia New" fill="black">ป.RB6@175</text>
      <g id="Stirrup_Type1">
         <rect style="fill:none;stroke:black;" stroke-width="0.3" rx="0.375" ry="0.375" height="15.7"
width="5.7" y1="0" x1="0" />
         <line stroke-width="0.3" y2="2.25" x2="1.5" stroke="black" y1="0.75" x1="0" />
         <line stroke-width="0.3" y2="1.5" x2="2.25" stroke="black" y1="0" x1="0.75" />
      </g>
   </g>

  <text font-size="4.5" text-anchor="middle" stroke="none" y="27" x="5" fill="black">B1</text>
</g>
</svg>
```

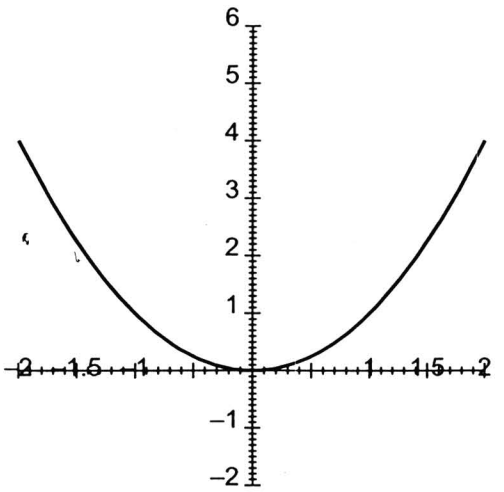
ผลลัพธ์ของ คำสั่งภาษา SVG ข้างบนจะได้แสดงดังภาพ



B1

1.3. แพคเกจ SVGFig

SVGFig เป็นแพคเกจเสริมของภาษา Python สำหรับเขียนแผนภาพคณิตศาสตร์ในรูปแบบ SVG จะออกแบบส่วนแสดงผล SVG เป็นลักษณะของ Document Object Model ทำให้สร้างภาษา SVG ในรูปแบบของวัตถุได้ ดังแสดงเป็นตัวอย่างในภาพที่ 2 โดยงานของ svgfig จะเน้นไปที่งานสร้างแผนภูมิทางคณิตศาสตร์ แต่ส่วน base ที่ใช้ในการแสดงภาพจะเป็น SVG Document Model




```
from svgfig import *
c = Curve( "t, t**2", -2, 2 )
p = Plot( -2, 2, -2, 6, c )
s = p.SVG()
s.save()
```

ภาพที่ 2 ตัวอย่างการเขียนกราฟด้วย SVGFig

คำสั่งการสร้างวัตถุกราฟฟิก line และแสดงผลลัพธ์ หลังจากทำการติดตั้ง svgfig แล้ว Python Shell จะสามารถดึงโมดูลในส่วนคลาส SVG มาใช้งานได้ คลาส SVG จะมีลักษณะเป็น List ของวัตถุที่สามารถเพิ่มวัตถุเข้าไปได้ ในการบันทึกไฟล์ SVG จะใช้งานผ่าน canvas ซึ่งจะเป็นฟังค์ชันที่เขียนครอบคลาส SVG อีกชั้นหนึ่ง

ภาษา SVG ในรูปแบบของภาษา XML

```
<g id="mygroup" fill="blue">
   <rect x="1" y="1" width="2" height="2" />
   <rect x="3" y="3" width="2" height="2" />
</g>
```

ภาษา SVG ในรูปแบบของภาษา Python (เรียกใช้งานผ่าน svgfig)

```
>>> svg = SVG("g", SVG("rect", x=1, y=1, width=2, height=2), \
...            SVG("rect", x=3, y=3, width=2, height=2), \
...          id="mygroup", fill="blue")
```

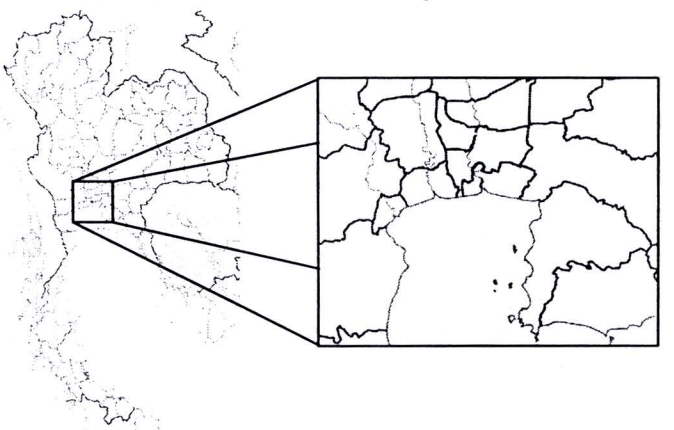
ตัวอย่างการสร้างวัตถุประเภทเส้นตรงด้วย svgfig

```
>>> from svgfig import SVG
>>> line1  = SVG('line'  , x1='0' , y1='0' , x2 = '5' , y2='10' )
>>> line1
<line x2='5' y1='0' x1='0' y2='10' />
>>> line1.xml()
"<line x2='5' y1='0' x1='0' y2='10' />"
```

2. งานวิจัยที่เกี่ยวข้อง

2.1. การใช้งาน SVG ทางด้านวิศวกรรม

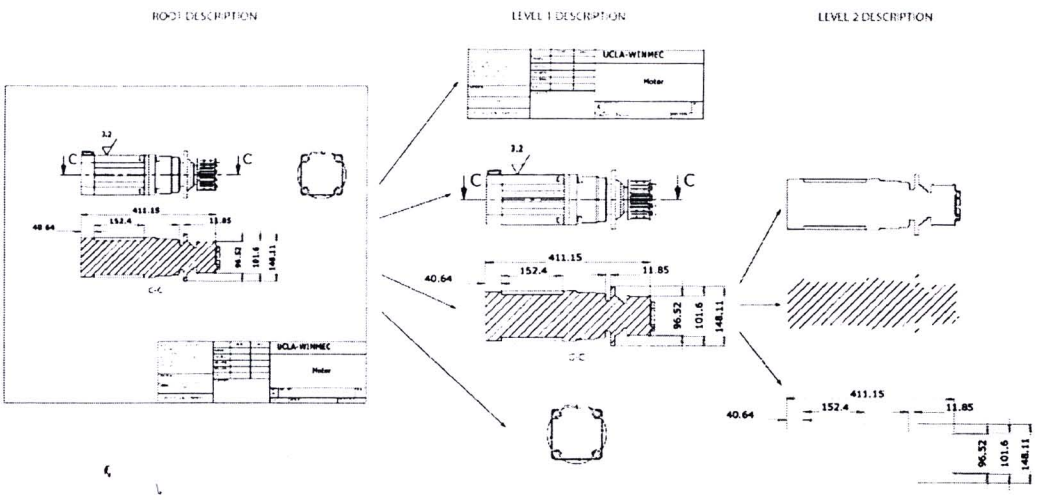
โดยส่วนใหญ่ในปัจจุบันจะใช้งานภาษา SVG ในการแสดงผลของรูปแผนที่ทางเว็บเพจ เนื่องจากจะได้ข้อมูลที่มีขนาดเล็กและมีความชัดเจนของลายเส้น จึงเห็นการใช้งาน SVG ส่วนใหญ่อยู่รูปแบบของการนำเสนอแผนที่ หรืองานทางด้านระบบสารสนเทศภูมิศาสตร์ (GIS) ซึ่งจะใช้โปรแกรมระบบสารสนเทศภูมิศาสตร์แปลงไฟล์แผนที่จาก Shape File ไปเป็นไฟล์ระบบ SVG แล้วจึงนำไปแสดงแผนที่บนเว็บเพจอีกที จะทำให้สามารถเปิดแผนที่ได้เร็วมากขึ้น เช่นไฟล์แสดงรูปแผนที่ประเทศไทยของวิกิพีเดีย เมื่อทำการขยายดูรายละเอียดขอบเขตของจังหวัดด้วยโปรแกรมภาพ SVG ก็จะไม่มีการแตกลายหยักขั้นบันได(Gain) และไฟล์มีขนาดไม่ใหญ่มาก



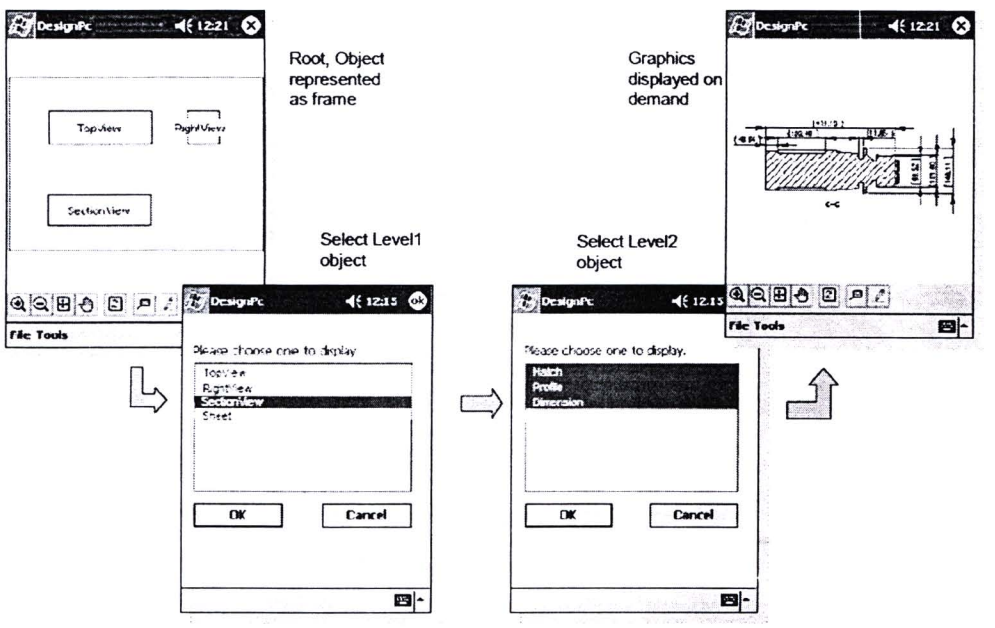
ภาพที่ 3 ภาพแผนที่ประเทศไทยของเว็บเพจวิกิพีเดียที่ใช้ไฟล์ SVG

Su และคณะได้ใช้ SVG ในการแสดงแบบผ่านทางมือถือ โดยทำการแปลงไฟล์ DXF เป็นไฟล์แบบ SVG เนื่องจากไฟล์แบบ DXF จะมีขนาดใหญ่ทำให้ใช้เวลาในการส่งข้อมูล แต่ไฟล์ SVG จะมีขนาดเล็ก แล้วยังสามารถทำการการแยกส่วนเป็นแต่ละส่วนเพื่อให้ขนาดไฟล์เล็กลง แล้วทำการส่งไฟล์ไปทีละส่วนพร้อมกันอาจจะบีบอัดด้วยจะได้เป็นไฟล์ svgz ทำให้การแสดงผลของแบบทางโทรศัพท์มือถือมีความเร็วมากขึ้น ดังแสดงการแยกส่วนในภาพที่ 4 และนำแสดงผลทางโทรศัพท์มือถือดังแสดงในภาพที่ 5 และไฟล์รหัสคำสั่งหน้าหลักจะดึงไฟล์ svg แต่ละระดับแสดงดังคำสั่งข้างล่าง

```
<?xml version="1.0" standalone="yes"?>
<svg width="600.0" height="391.429" viewBox="0 0 1050.000 685.000" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<title>SectionView</title>
<parent>MotorDrawing</parent>
<g>    <image xlink:href=hatch.svg/>
       <image xlink:href=profile.svg/>
       <image xlink:href=dimension.svg/>
</g>
</svg>
```

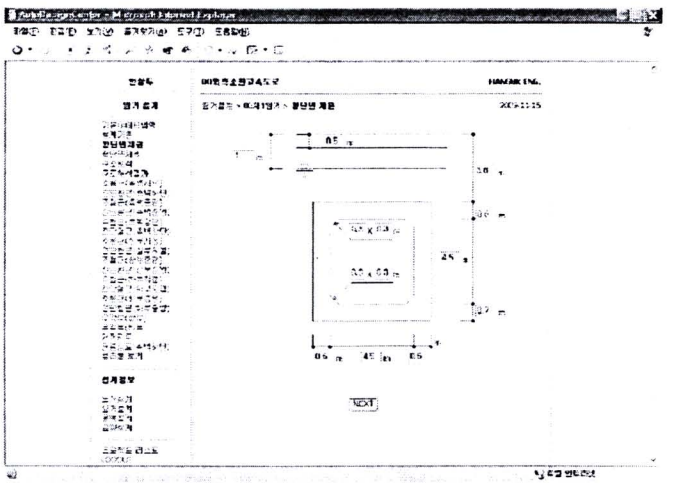


ภาพที่ 4 การแยกส่วนไฟล์แสดงผลงานเขียนของไฟล์ SVG เพื่อแสดงผลทางโทรศัพท์มือถือ

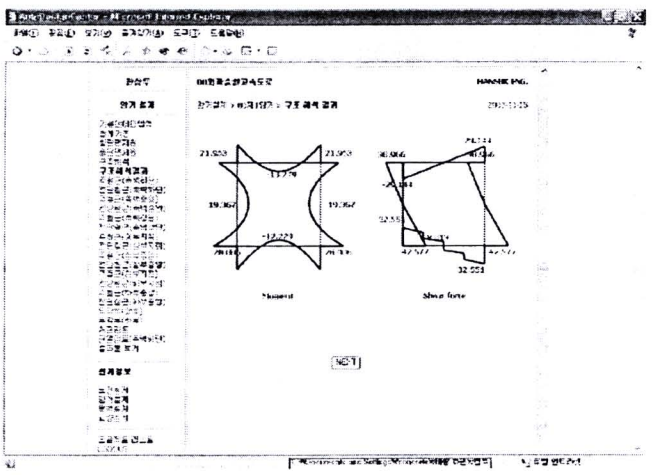


ภาพที่ 5 การแสดงผลงานเขียนแบบทางโทรศัพท์มือถือ

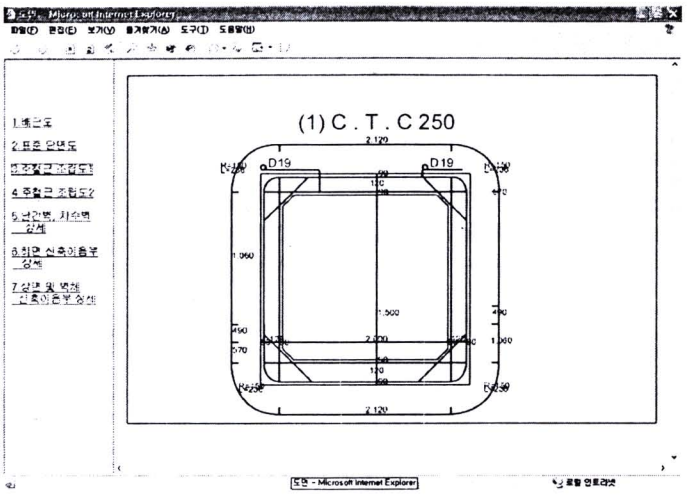
. Kang และคณะได้พัฒนาเว็บเพจสำหรับการวิเคราะห์และออกแบบท่อลอด ซึ่งจะแสดงผลการวิเคราะห์และการออกแบบพร้อมกับแสดงการเสริมเหล็กของท่อลอดเป็นรูปกราฟฟิกด้วยไฟล์แบบ SVG และสามารถจัดเก็บข้อมูลในฐานข้อมูลได้ โดยระบบจะแสดงข้อมูลเป็นกราฟฟิกที่ได้มาตราส่วนผ่านทางเว็บเพจ ไม่ต้องผ่านโปรแกรม CAD ในภาพที่ 6 เป็นการแสดงการป้อนข้อมูลของขนาดท่อลอดและความลึกของท่อลอด ในภาพที่ 7 แสดงผลการวิเคราะห์หาค่าแรงเฉือนและโมเมนต์ที่เกิดขึ้นในท่อลอด และภาพที่ 8 จะแสดงการเสริมเหล็กที่ได้จากการออกแบบ



ภาพที่ 6 การป้อนข้อมูลออกแบบทางเว็บเพจของระบบท่อลอด



ภาพที่ 7 ผลการวิเคราะห์แรงเฉือนและโมเมนต์ของระบบท่อลอด

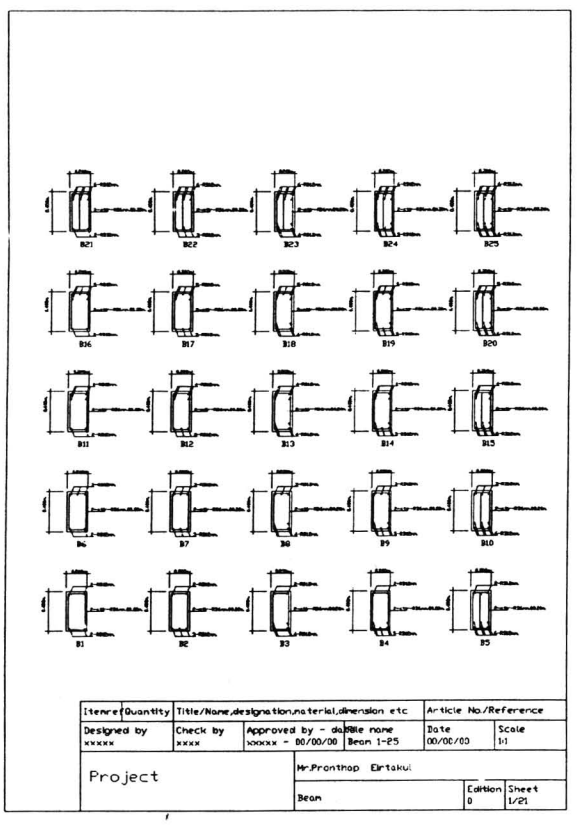


ภาพที่ 8 ผลลัพธ์การคำนวณออกแบบท่อลอดทางเว็บเพจ

พรเทพ และคณะ ได้พัฒนาโปรแกรมโดยภาษา Python เพื่อให้สร้างรายละเอียดหน้าตัดคาน โดยส่งผลลัพท์เป็นไฟล์ DXF(Data Exchange Format) ในรูปแบบข้อมูลแลกเปลี่ยนระหว่างโปรแกรม CAD จะมีลักษณะเป็นไฟล์ข้อความและมีรหัสตัวเลขในการควบคุมข้อมูล เช่น 10 คือพิกัด x 20 คือพิกัด และ 30 คือพิกัด z โดยจะทำการ เขียนหน้าตัดคาน เสา และตัวอย่างพื้นและฐานราก เพื่อส่งผลลัพท์ให้กับโปรแกรม CAD ตัวโปรแกรม Python จะมีรหัสคำสั่งสำหรับสร้างวัตถุกราฟฟิกแบบ DXF แล้วนำมารวมกันเพื่อบันทึกเป็นไฟล์นามสกุล .dxf แล้วนำไปเปิดด้วยโปรแกรม CAD เพื่อดูผลลัพธ์การเขียน ไฟล์ DXF ที่สร้างจะใช้เป็น dxf รุ่น 14 จะมีรายละเอียดไม่มากนัก แต่พอใช้งานได้ในระดับหนึ่ง ตัวอย่างรหัสโปรแกรมดังแสดง

```
from Beam import *
from Border_1 import*
def test():
    i = DrawingObject()
    Border(i,0.0,0.0,420.0,297.0,20)
    b=200
    h=400
    x1=800
    y1=2000
    DrawBeam(i,x1,y1,"B1",b,h,"2-RB12mm.","2-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x1+1000,y1,"B2",b,h,"2-RB12mm.","3-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x1+2000,y1,"B3",b,h,"2-RB12mm.","4-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x1+3000,y1,"B4",b,h,"2-RB12mm.","5-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x1+4000,y1,"B5",b,h,"2-RB12mm.","6-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    x2=x1
    y2=y1+1000
    DrawBeam(i,x2,y2,"B6",b,h,"3-RB12mm.","2-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x2+1000,y2,"B7",b,h,"3-RB12mm.","3-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x2+2000,y2,"B8",b,h,"3-RB12mm.","4-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x2+3000,y2,"B9",b,h,"3-RB12mm.","5-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x2+4000,y2,"B10",b,h,"3-RB12mm.","6-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    x3=x1
    y3=y2+1000
    DrawBeam(i,x3,y3,"B11",b,h,"4-RB12mm.","2-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x3+1000,y3,"B12",b,h,"4-RB12mm.","3-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
```

```
    DrawBeam(i,x3+2000,y3,"B13",b,h,"4-RB12mm.","4-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x3+3000,y3,"B14",b,h,"4-RB12mm.","5-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x3+4000,y3,"B15",b,h,"4-RB12mm.","6-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    x4=x1
    y4=y3+1000
    DrawBeam(i,x4,y4,"B16",b,h,"5-RB12mm.","2-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x4+1000,y4,"B17",b,h,"5-RB12mm.","3-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x4+2000,y4,"B18",b,h,"5-RB12mm.","4-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x4+3000,y4,"B19",b,h,"5-RB12mm.","5-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x4+4000,y4,"B20",b,h,"5-RB12mm.","6-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    x5=x1
    y5=y4+1000
    DrawBeam(i,x5,y5,"B21",b,h,"6-RB12mm.","2-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x5+1000,y5,"B22",b,h,"6-RB12mm.","3-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x5+2000,y5,"B23",b,h,"6-RB12mm.","4-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x5+3000,y5,"B24",b,h,"6-RB12mm.","5-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")
    DrawBeam(i,x5+4000,y5,"B25",b,h,"6-RB12mm.","6-RB12mm."," "," ","2-stir-RB6mm.@0.20m.")


    i.save('NewBeam002.dxf')
if __name__ == "__main__":
    test()
```



ภาพที่ 9 ผลการรัน pyDrawingDXF เพื่อเขียนหน้าตัดคานเป็นไฟล์ DXF

โกมล นกสว่าง และคณะ ได้ทำการพัฒนาโปรแกรมการถ่ายน้ำหนักบรรทุกลงคานต่อเนื่อง โดยใช้ภาษา Python ให้ทำการป้อนข้อมูลในลักษณะของแปลนโครงสร้าง และได้แสดงผังแปลนคานโครงสร้างด้วย SVG พร้อมแสดงผลลัพธ์การถ่ายน้ำหนักบรรทุกจากแผ่นพื้น น้ำหนักตัวคานเอง และกำแพงลงสู่คานดังแสดงในภาพที่ 11 โดยการป้อนข้อมูลจะเป็นลักษณะดังนี้

- ข้อมูลกริดซึ่งจะระบุชื่อของกริดและตำแหน่งของกริดสามารถระบบเป็นตัวเลขหรือในลักษณะของอ้างอิงสัมพัทธ์กับกริดก่อนหน้าได้
- ข้อมูลจุดต่อ จะระบุเฉพาะตำแหน่งที่เป็นจุดรองเท่านั้น
- ข้อมูลคาน จะระบุจุดต่อได้ทุกจุดที่อยู่บนกริดอ้างอิงแต่ต้องเป็นแนวดิ่งหรือแนวนอนเท่านั้น โดยระบุเป็นจุดเริ่มและจุดสุดท้าย และค่าระดับชั้นของการคำนวณ ชื่อหน้าตัด และค่าระดับหลังคาน
- ข้อมูลของแผ่นพื้น จะระบุตำแหน่งมุมซ้ายบน และมุมขวาล่างเท่านั้น ระบุความหนา น้ำหนักของ Super Impose Dead Load และน้ำหนักบรรทุกจร
- ข้อมูลของกำแพง จะระบุจุดเริ่ม และจุดสุดท้าย และน้ำหนักกำแพง และความสูง

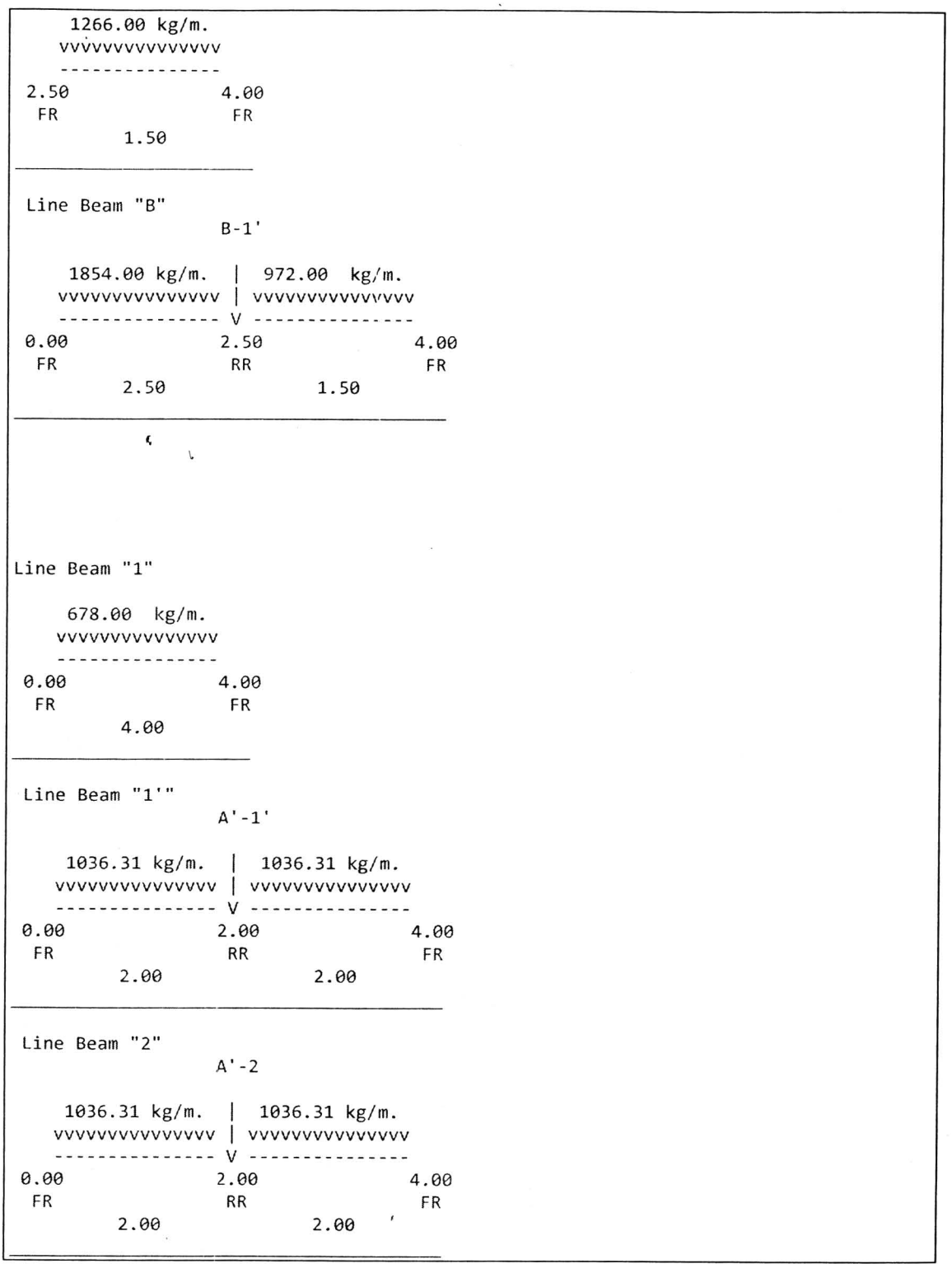
รหัสโปรแกรมการใช้งาน pyGridPlan

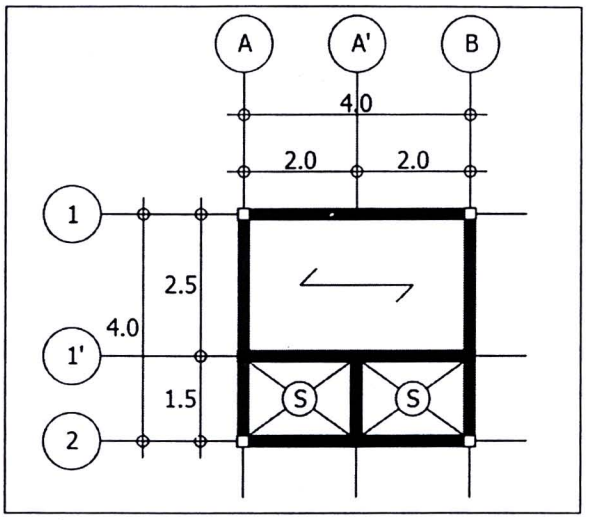
```
'''

  2010 jan 21
 ทดสอบ Package PyGridPlan
'''

from PyGridPlan.Plan import *

# กรอกชื่อ Project ของคุณ
NameProject = "SimpleRoom"

# ส่วนของการกรอกข้อมูลต่าง ๆ
gridX = [["A", 0.0],["A'", "A:2.0"], ["B", "A:4.0"]]
gridY = [["1", 0.0], ["1'", "1:2.5"], ["2", "1:4.0"]]
nodes = ["A-1", "B-1", "A-2", "B-2"]
members = [["B-1", "A-1", 1, "B1", 0.0],\
           ["A-1", "A-2", 1, "B1", 0.0],\
           ["A-2", "B-2", 1, "B1", 0.0],\
           ["B-1", "B-2", 1, "B1", 0.0],\
           ["A-1'", "B-1'", 2, "B1", 0.0],\
           ["A'-1'", "A'-2", 3, "B1", 0.0]]
slabs = [["A-1'", "A'-2', 0.12, 0, 300],\
         ["A'-1'", "B-2', 0.12, 0, 300],\
         ["A-1", "B-1'", "X", 0.12, 0, 300]]
walls = [["A-1", "B-1", 180, 2.7],\
         ["A-1", "A-2", 180, 2.7],\
         ["A-2", "B-2", 180, 2.7],\
         ["B-1", "B-2", 180, 2.7],\
         ["A-1'", "B-1'", 180, 2.7],\
         ["A'-1'", "A'-2", 180, 2.7]]
sectionBeam = [["B1", 0.20, 0.40]]
```

```
# แสดงข้อมูลทั้งหมดที่รับเข้ามา
print '        Project " ' +NameProject + ' "\n'

# ส่งข้อมูลทั้งหมดเข้าสู่ระบบ
HomePlan = Plan()
HomePlan.AddList(gridX, gridY, nodes, members, slabs, walls, sectionBeam)

HomePlan.GenFullNodes()

# ตรวจสอบข้อมูลทั้งหมด
CheckData, Report = HomePlan.CheckAll
print Report + "\n\n"

# สร้าง File SVG เพื่อใช้เปิดในโปรแกรม Mozila Firefox
String = HomePlan.Draw
fName = NameProject + ".svg"
fileHandle = open( fName, 'w' )
fileHandle.write(String)
fileHandle.close()


# ขอค่าน้ำหนักบรรทุกที่ถ่ายลงคานแต่ละแนว
HomePlan.TransferLoad()
print "              TransferLoad\n"
for i in HomePlan.GridX:
   print ' Line Beam "%s" \n\n'%(i.Name)
   Line_A = HomePlan.GetLineBeam("%s"%(i.Name))
   print Line_A.StringLoad()

for i in HomePlan.GridY:
   print ' Line Beam "%s" \n\n'%(i.Name)
   Line_1 = HomePlan.GetLineBeam("%s"%(i.Name))
   print Line_1.StringLoad()
```

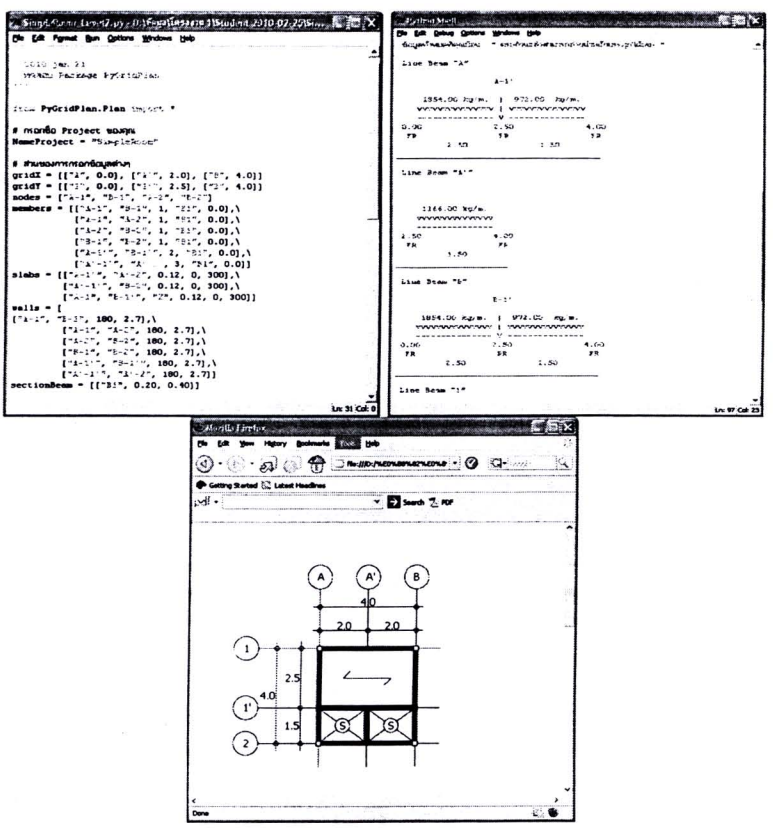
ผลลัพธ์การถ่ายน้ำหนักบรรทุกในลักษณะของ ASCII mode

```
Project " SimpleRoom "

          CheckDataAll

ตรวจสอบข้อมูล Nodes แล้วไม่พบข้อผิดพลาด
ตรวจสอบข้อมูล Members แล้วไม่พบข้อผิดพลาด
ตรวจสอบข้อมูล Slabs แล้วไม่พบข้อผิดพลาด
ตรวจสอบข้อมูล Walls แล้วไม่พบข้อผิดพลาด
ตรวจสอบข้อมูล SectionBeams แล้วไม่พบข้อผิดพลาด

                TransferLoad

 Line Beam "A"
                    A-1'

     1854.00 kg/m.   |  972.00  kg/m.
    vvvvvvvvvvvvvvv | vvvvvvvvvvvvvvv
    --------------- V ---------------
 0.00               2.50               4.00
  FR                 RR                 FR
          2.50                1.50
_____________________________________________

 Line Beam "A'"
```


1266.00 kg/m.
2.50
4.00
FR
FR
1.50
Line Beam "B"
B-1'
1854.00 kg/m.
972.00 kg/m.
0.00
2.50
4.00
FR
RR
FR
2.50
1.50
Line Beam "1"
678.00 kg/m.
0.00
4.00
FR
FR
4.00
Line Beam "1'"
A'-1'
1036.31 kg/m.
1036.31 kg/m.
0.00
2.00
4.00
FR
RR
FR
2.00
2.00
Line Beam "2"
A'-2
1036.31 kg/m.
1036.31 kg/m.
0.00
2.00
4.00
FR
RR
FR
2.00
2.00



ภาพที่ 10 ไฟล์ SVG สำหรับแสดงแปลนคาน



ภาพที่ 11 การป้อนข้อมูล และผลลัพธ์ของน้ำหนักบรรทุกที่ถ่ายลงคานต่อเนื่องและแปลนคาน