บทที่ 4
ผลการศึกษา

เพื่อให้การวิจัยเป็นไปตามวัตถุประสงค์ ผู้วิจัยขอเสนอผลการศึกษาการนำภาษา SVG มาใช้งานเพื่อแสดงแบบรายละเอียดก่อสร้าง ดังนี้

1. ผลการศึกษาการควบคุมคำสั่งภาษา SVG เพื่อใช้ในการแสดงรายละเอียดแบบก่อสร้าง

ในการศึกษาการนำภาษา SVG มาใช้งานเพื่อการแสดงแบบรายละเอียดก่อสร้างนั้น ได้พบกับปัญหาการใช้งานต่าง ๆ และได้ค้นหาวิธีที่จะใช้งาน ดังนี้

ในการใช้งานภาษา SVG ต้องการสร้างข้อมูลโดยไม่ต้องระบุหน่วย ค่าปกติที่ภาษา SVG ระบุไว้จะเป็นค่าที่อ้างอิงกับการแสดงผลทางจอภาพคอมพิวเตอร์ จึงใช้งานเป็น px หรือพิกเซล แต่งานเขียนแบบต้องการแค่ระบุตัวเลขโดยให้ตัวเลขทุกตัวอยู่ในหน่วยเดียวกันและในประเทศไทยจะนิยมในระบบเมตริก ซึ่งในภาษา SVG มีให้ระบุได้อยู่สองแบบ คือ cm หรือ mm ดังนั้นในงานวิจัยนี้จึงตัดสินใจใช้งานในหน่วย มม เนื่องในงานก่อสร้างถือว่าเป็นหน่วยที่เล็กที่สุด จึงต้องกำหนดในส่วนคำสั่ง <svg > ดังแสดง

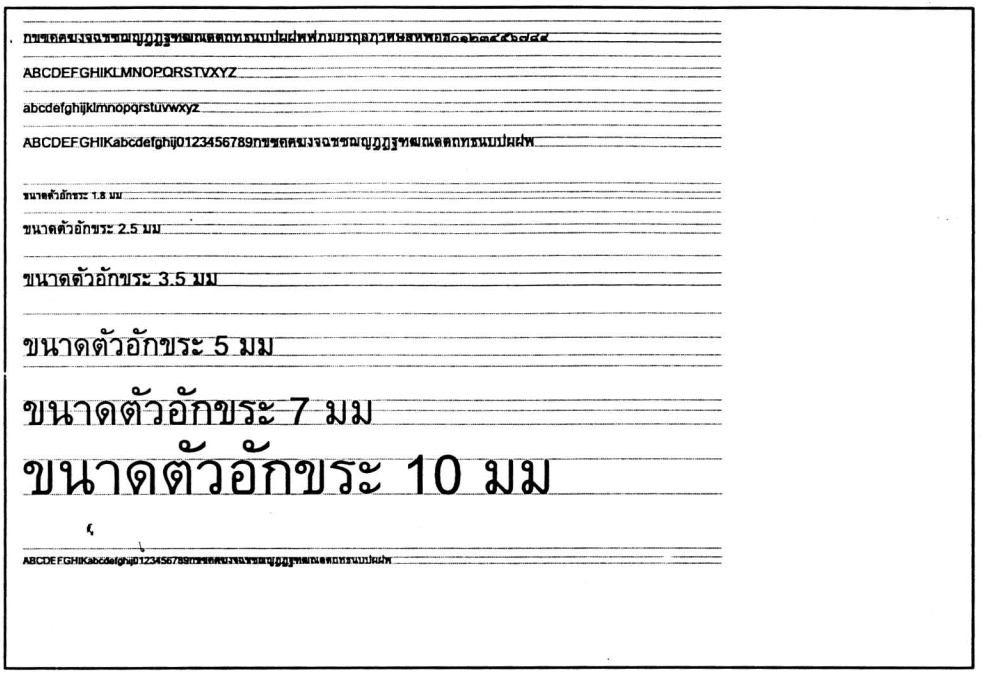
```
<svg  xmlns="http://www.w3.org/2000/svg"  " xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1"
   width="210mm"  height="297mm"    viewBox="0 0 210 297">
```

ขนาดกระดาษ A4 และต้องใส่หน่วยเป็น mm และใช้ร่วมกับการกำหนดคำสั่ง viewBox ให้มีขนาดสอดคล้องกัน จะส่งผลให้หน่วยที่ใช้งานในไฟล์เป็น mm ทั้งหมด

ภาพที่ 18 การกำหนดขนาดกระดาษ และกำหนดหน่วยในการใช้งานภาษา SVG ในหน่วย มม

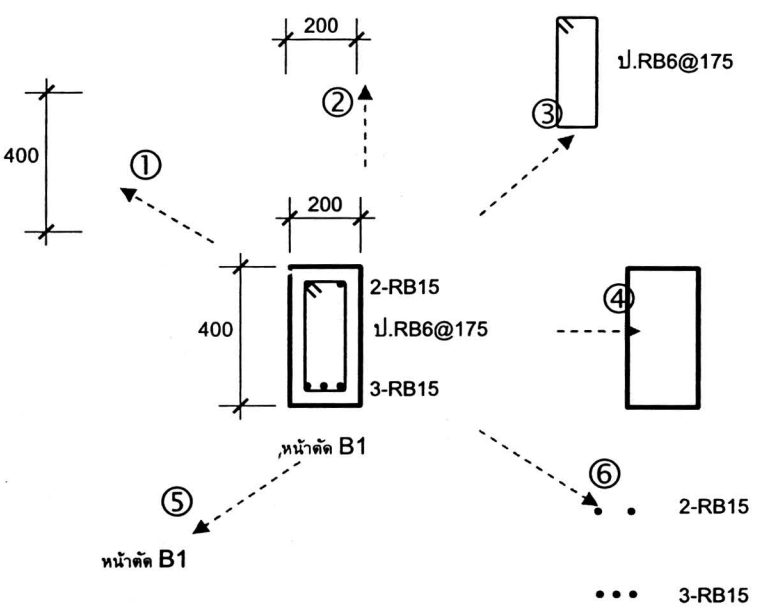
แต่อย่างไรก็ตามเมื่อเขียนเป็นโปรแกรมภาษาไพธอน จะกำหนดให้สคริปต์จัดให้ทั้งหมดโดยที่ผู้ใช้บอกแต่ว่าเป็นกระดาษ A4 หรือ A3 ก็พอ

ในการใช้งานอิลีเมนต์ <text> พบว่าข้อความที่เป็นภาษาไทยในสคริปต์ภาษาไพธอน จะต้องใช้เป็น u'หน้าตัดคาน' นั่นคือขึ้นต้นด้วยอักษร u เพื่อบ่งให้ภาษาไพธอนทราบว่าเป็น Unicode และไฟล์ภาษา SVG ควรจะบันทึกในรูปแบบของ Unicode ด้วยจะได้แสดงผลเป็นภาษาไทยได้ แต่จากการศึกษาพบว่าความสูงของข้อความที่กำหนด จะต้องมีการเทียบความสูงก่อน เช่น ถ้าต้องการให้ความสูงจริงของอักษรไทยเท่ากับ 2.5 มม จะต้องกำหนดตัวแปร font-size เท่ากับ 4.5 ดังนั้นควรทำการทดสอบความสูงของฟอนต์ให้ได้ขนาดตามที่ต้องการเสียก่อน ดังแสดงตัวอย่างในภาพ



ภาพที่ 19 ผลการทดสอบขนาดของตัวอักษร

ในการศึกษาการนำเอาอิลีเมนต์ของ SVG ต่าง ๆ มาประกอบกันเป็นชิ้นส่วนของรายละเอียดแบบก่อสร้าง พบว่าใช้งานได้ดี สามารถกำหนดขนาดของเส้นได้ เมื่อทำการศึกษาผู้วิจัยทำการพิจารณารายละเอียดของแบบก่อสร้าง ยกตัวอย่างเช่น หน้าตัดคานจะประกอบด้วยอิลีเมนต์ของภาษา SVG ดังแสดงในภาพ



ภาพที่ 20 องค์ประกอบของอิลีเมนต์ภาษา SVG ที่นำมาใช้สร้างเป็นรายละเอียดหน้าตัดคาน

ส่วนที่ ① และ ② คือการเขียนเส้นบอกมิติ จะเป็นกรุ๊ปของอิลีเมนต์ภาษา SVG ซึ่งประกอบด้วยอิลีเมนต์ <line> และอิลีเมนต์ <text> ประกอบกันดังแสดงด้วยตัวอย่างรหัสคำสั่งภาษา SVG ดังนี้

```
<g  id="xDimension"  transform="translate(0,-7)">
    <line x1="-2" y1="0"  x2="12" y2="0"  style="stroke-width:0.13"  />
    <line x1="0"  y1="-3" x2="0" y2="5"    style="stroke-width:0.13"  />
    <line x1="10" y1="-3" x2="10" y2="5"   style="stroke-width:0.13" />
    <line x1="-1"  y1="1"  x2="1" y2="-1"  style="stroke-width:0.35"  />
    <line x1="9" y1="1"  x2="11" y2="-1"   style="stroke-width:0.35" />
    <text x="5"  y="-1"   font-size="4.5" font-family="Browallia New"
      text-anchor="middle"  stroke="none"  fill="black" >200</text>
</g>
```

ส่วนที่ ③ คือการเขียนส่วนของเหล็กปลอก จะเป็นกรุ๊ปของอิลีเมนต์ภาษา SVG ซึ่งประกอบด้วยอิลีเมนต์ <line> , <rect> และอิลีเมนต์ <text> ประกอบกันดังแสดงด้วยตัวอย่างรหัสคำสั่งภาษา SVG ดังนี้

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

ส่วนที่ ④ คือการเขียนส่วนกรอบคานส่วนขอบรอบของเนื้อคอนกรีต จะเป็นอิลีเมนต์ภาษา SVG <rect> ดังแสดงด้วยตัวอย่างรหัสคำสั่งภาษา SVG ดังนี้

```
<rect x="0"  y="0"  width="10" height="20"    fill="white"  />
```

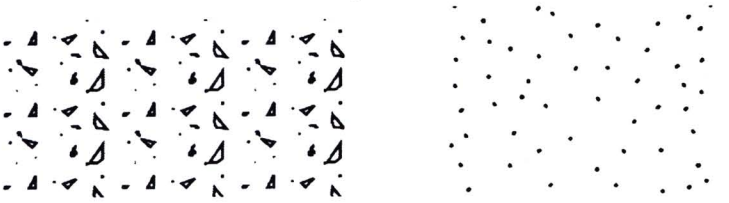
ส่วนที่ ⑤ คือการเขียนส่วนชื่อของคาน จะเป็นอิลีเมนต์ภาษา SVG <text> ดังแสดงด้วยตัวอย่างรหัสคำสั่งภาษา SVG ดังนี้

```
<text x="5"  y="27"   font-size="3.5"  " font-family="Browallia New"
     text-anchor="middle"   stroke="none"    fill="black">หน้าตัด B1</text>
```

ส่วนที่ ⑥ คือการเขียนส่วนเหล็กเสริมในหน้าตัดคาน จะเป็นกรุ๊ปของอิลีเมนต์ภาษา SVG ซึ่งประกอบด้วยอิลีเมนต์ <circle> และอิลีเมนต์ <text> ประกอบกันดังแสดงด้วยตัวอย่างรหัสคำสั่งภาษา SVG ดังนี้

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

Patterns หรือลายที่จะเติมว่าเป็นวัสดุชนิดใด เช่น คอนกรีต หรือทราย ในภาษา SVG ยังไม่มีแบบที่เป็นมาตรฐานผู้ใช้จะต้องกำหนดเอง และในเว็บไซท์ยังไม่มีให้ดาวน์โหลด ดังนั้นส่วนนี้จึงจำเป็นต้องสร้างเอง โดยใช้โปรแกรม Inkscape ถอดลายเส้นจากภาพแบบ pixel เป็นวัตถุเวคเตอร์ แล้วจึงสั่งให้กลายเป็น Pattern หลังจากสั่งบันทึกไฟล์แล้วจึงเปิดดูรหัสคำสั่งและสำเนามาใช้งานอีกครั้ง ในอนาคตเมื่อมีผู้ใช้งานภาษา SVG มากขึ้นก็จะมีรูปแบบ Patterns ให้ใช้งานได้มากขึ้น

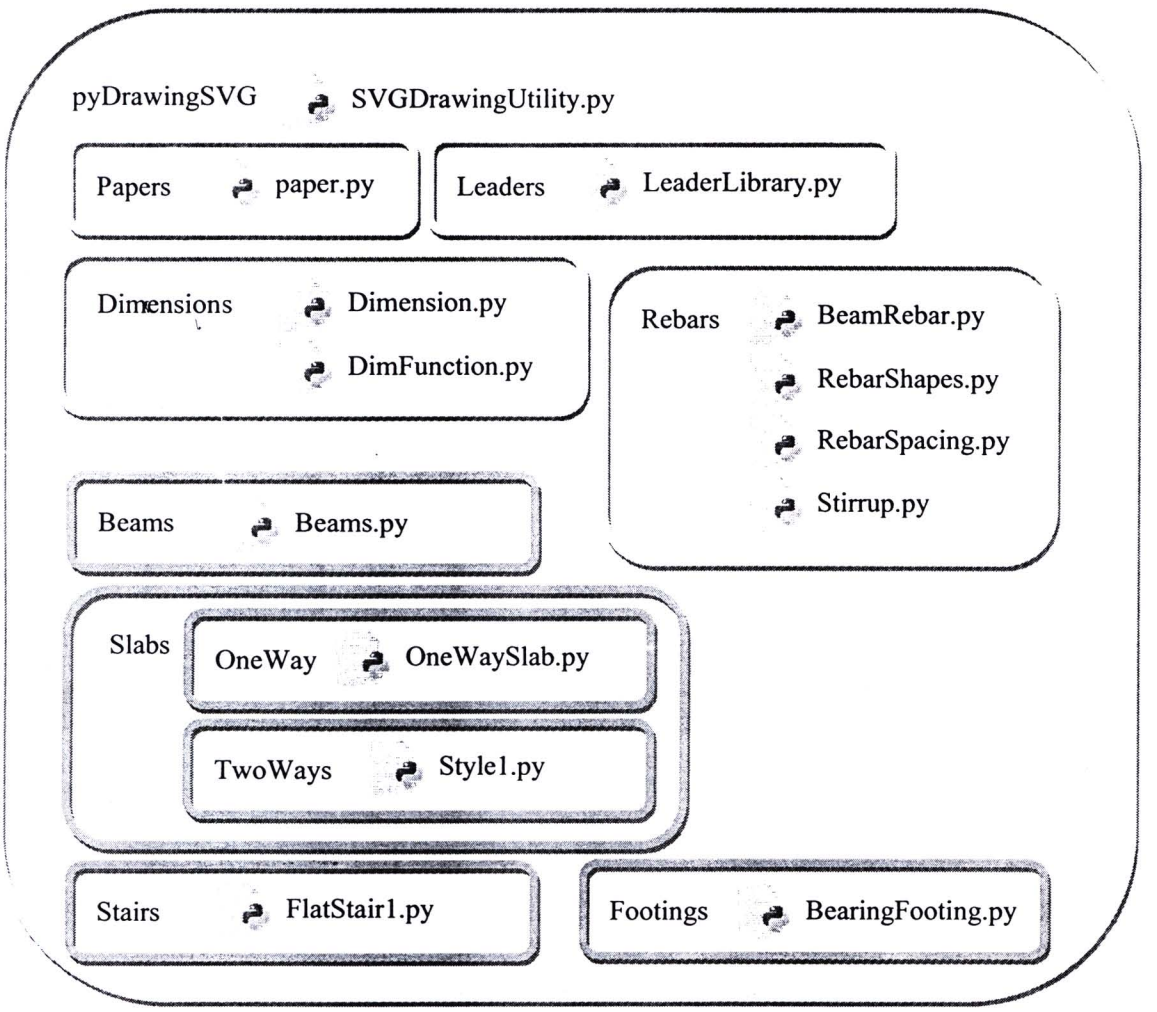


ภาพที่ 21 ตัวอย่างลาย Pattern ของคอนกรีต และทราย

ศึกษาเข้าใจรูปแบบของภาษา SVG แล้วเราจะนำไปพัฒนาต่อเป็นส่วนที่จะทำให้สามารถเขียนรายละเอียดรูปตัดได้โดยให้ผู้ใช้กำหนดความกว้างความยาวของชิ้นส่วน โดยไม่ต้องสนใจในรายละเอียดของคำสั่งภาษา SVG มากนัก โดยการใช้งานแพคเกจภาษาไพธอนสำหรับใช้งานในการสร้างแบบรายละเอียดก่อสร้างโดยตรง และไม่ต้องทำความเข้าใจภาษา SVG (กรณีเป็นระดับผู้ใช้งาน)

2. ผลการศึกษาการพัฒนาภาษาไพธอนเพื่อแสดงรายละเอียดแบบก่อสร้าง

หลังจากที่ได้ออกแบบและพัฒนาแพคเกจแล้ว ในการดำเนินการวิจัยครั้งนี้จะได้แพคเกจของภาษาไพธอน ดังแสดงในภาพ



ภาพที่ 22 องค์ประกอบต่างของแพกเกจ pyDrawingSVG

จะแยกเป็นส่วนต่าง ๆ ตามแต่และโฟลเดอร์ เพื่อให้ง่ายต่อไปควบคุมการเขียนโปรแกรม จึงจัดเป็นกรอบใหญ่ ๆ และวางตัววัตถุเป็นแบบพื้นฐานที่ต้องใช้งานบ่อย ๆ เช่น กระดาษ เส้นบอกมิติ เส้นชี้บอก เหล็กเสริมในลักษณะต่าง ๆ กลุ่มวัตถุเหล่านี้จะเป็นคลาสพื้นฐาน (Base Class) แล้วนำไปประกอบเป็นชิ้นส่วนขึ้นมาตามหลักการ Composite of Object นั่นคือจะมองให้หน้าตัดประกอบด้วยส่วนประกอบหลาย ๆ ชิ้น แบบการการผลิตรถยนต์ แบ่งเป็นส่วน ๆ เช่น ล้อ ประตู พวงมาลัย เป็นต้น

3. คลาสไดอะแกรมของวัตถุกระดาษเขียนแบบ

จะเป็นคลาสใช้ในการสร้างกระดาษงานเขียนแบบ และเป็น container สำหรับบรรจุวัตถุรายละเอียดแบบก่อสร้างต่าง ๆ

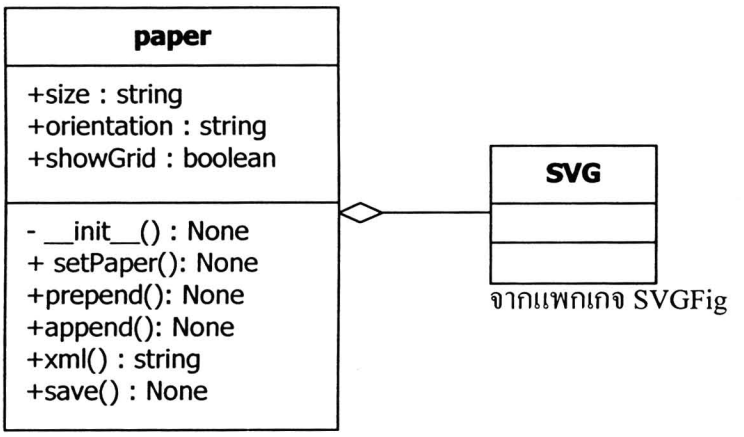
1. ความต้องการ (Requirements)

ในคลาสนี้จะออกแบบให้ผู้ใช้สามารถกำหนดขนาดของกระดาษได้ เป็นขนาด A2,A3,A4 และ กำหนดแนวการวางว่าเป็นแนวตั้งหรือแนวนอน พร้อมกับใช้มีความสามารถในการแสดงเส้นกริดอ้างอิงสีฟ้า ระยะห่าง 1 ซม เพื่อใช้ในขั้นตอนการวางตำแหน่งรูปรายละเอียดก่อสร้างได้ (เป็นคลาส SVG) เมื่อทำการบรรจุวัตถุเข้าไปแล้วก็จะทำการบันทึกลงเป็นไฟล์ SVG ดังนั้นจะอาศัยการใช้งานฟังค์ชัน canvas ของแพคเกจ SVGFig

2. การออกแบบและพัฒนา

2.1. การออกแบบเชิงโครงสร้าง

เมื่อเริ่มทำการออกแบบแล้วจะได้โครงสร้างของคลาส paper ดังแสดงในภาพ

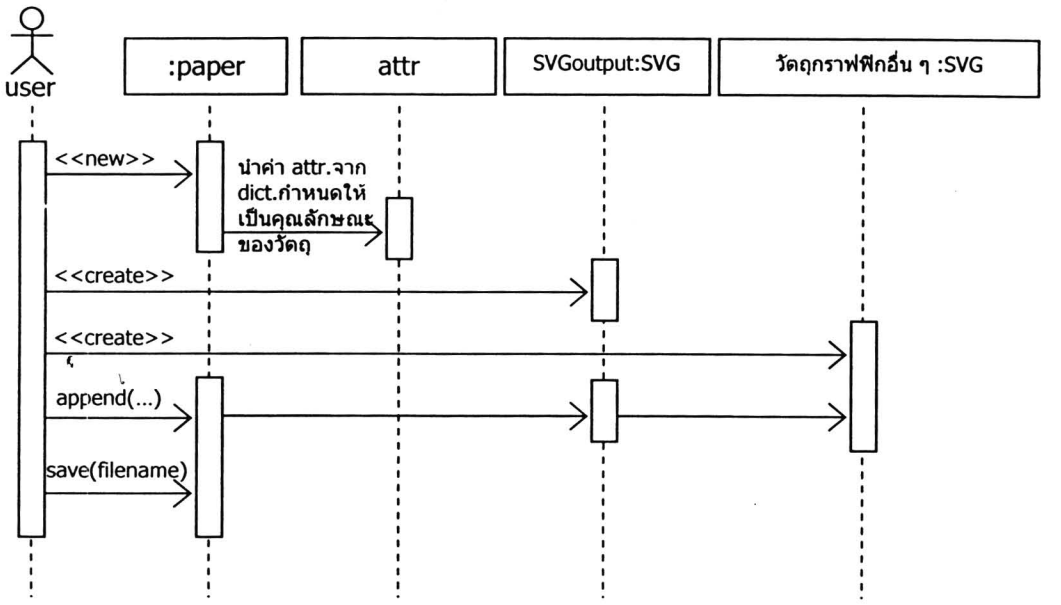


คลาส paper จะทำหน้าที่ดังต่อไปนี้

- ประกาศโครงสร้างและสร้างวัตถุที่จะเป็นองค์ประกอบ คือ คลาส SVG
- ประกาศ Method สำหรับการกำหนดขนาดกระดาษได้
- ประกาศ Method สำหรับการเพิ่มองค์ประกอบ
- ประกาศ Method สำหรับการบันทึกและแสดงผลลัพธ์

2.2. การออกแบบเชิงกิจกรรม

จากความต้องการของระบบ เราสามารถอธิบายกิจกรรมที่จะเกิดขึ้นในระบบคลาส paper ได้ดังนี้



4. คลาสไดอะแกรมของวัตถุเส้นชี้

จะเป็นคลาสใช้ในการสร้างเส้นชี้บอก

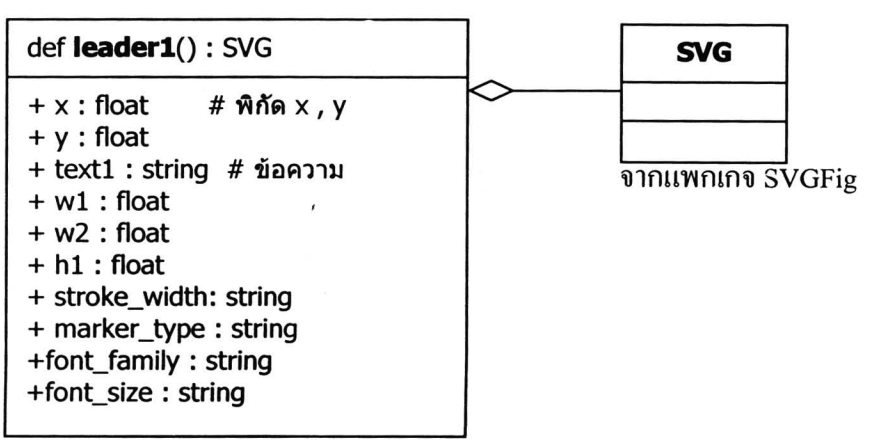
1. ความต้องการ (Requirements)

ในคลาสนี้จะออกแบบให้ผู้ใช้สร้างเส้นชี้ได้ เนื่องจากทำการศึกษาเบื้องต้น กำหนดไว้ในรูปแบบฟังค์ชันก่อน เนื่องจากภาษาไพธอนจะมองเห็นฟังค์ชันเป็นวัตถุได้

2. การออกแบบและพัฒนา

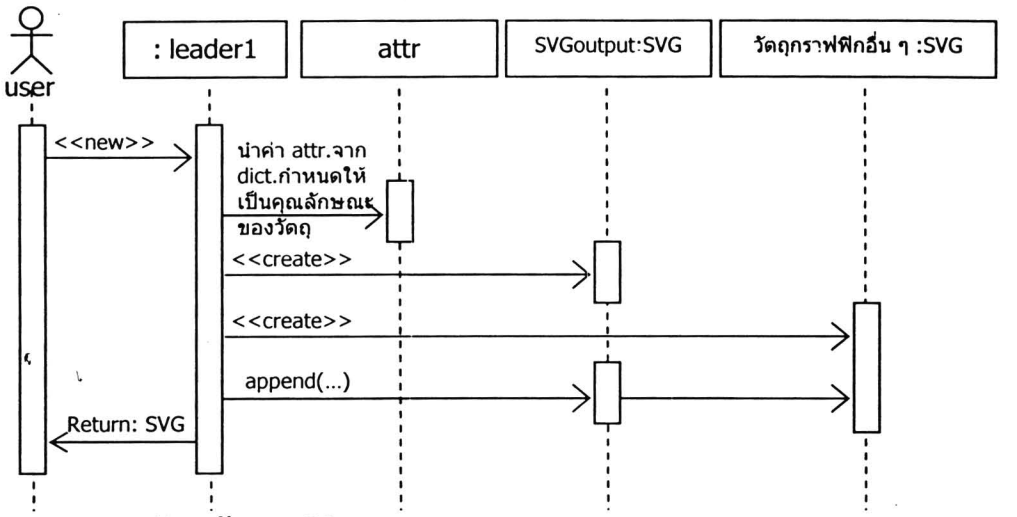
2.1. การออกแบบเชิงโครงสร้าง

เมื่อเริ่มทำการออกแบบแล้วจะได้โครงสร้างของคลาส leader1 ดังแสดงในภาพ



2.2. การออกแบบเชิงกิจกรรม

จากความต้องการของระบบ เราสามารถอธิบายกิจกรรมที่จะเกิดขึ้นในระบบคลาส leader1 ได้ดังนี้



5. คลาสไดอะแกรมของวัตถุเส้นบอกมิติ

จะเป็นคลาสใช้ในการสร้างเส้นบอกมิติ เพื่อใช้ในการบอกขนาดของชิ้นส่วน

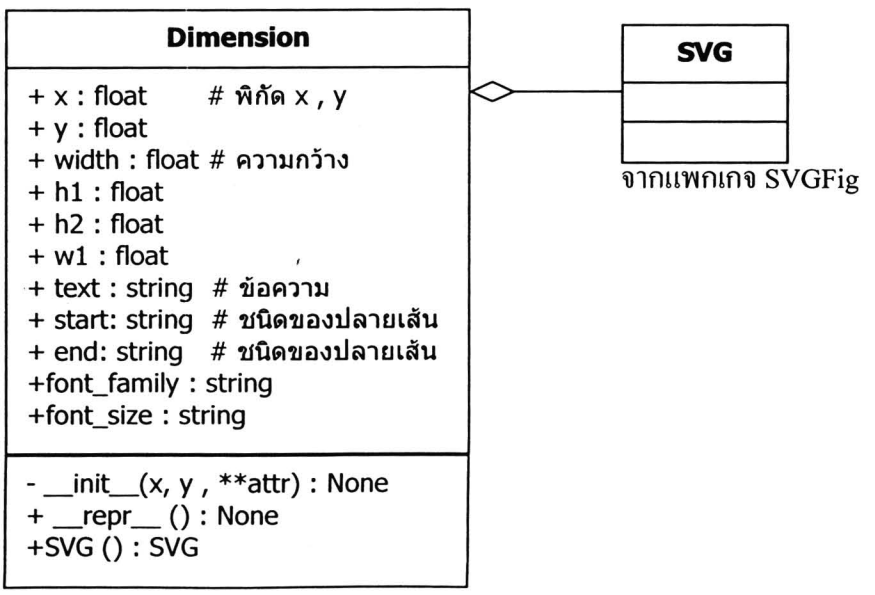
1. ความต้องการ (Requirements)

ในคลาสนี้จะออกแบบให้ผู้ใช้สร้างเส้นชี้ได้ และสามารถเปลี่ยนแปลงความยาวของเส้นมิติต่าง ๆ ได้ เนื่องจากทำการศึกษาเบื้องต้น กำหนดไว้ในรูปแบบฟังค์ชันก่อน เนื่องจากจะมองเห็นฟังค์ชันเป็นวัตถุ

2. การออกแบบและพัฒนา

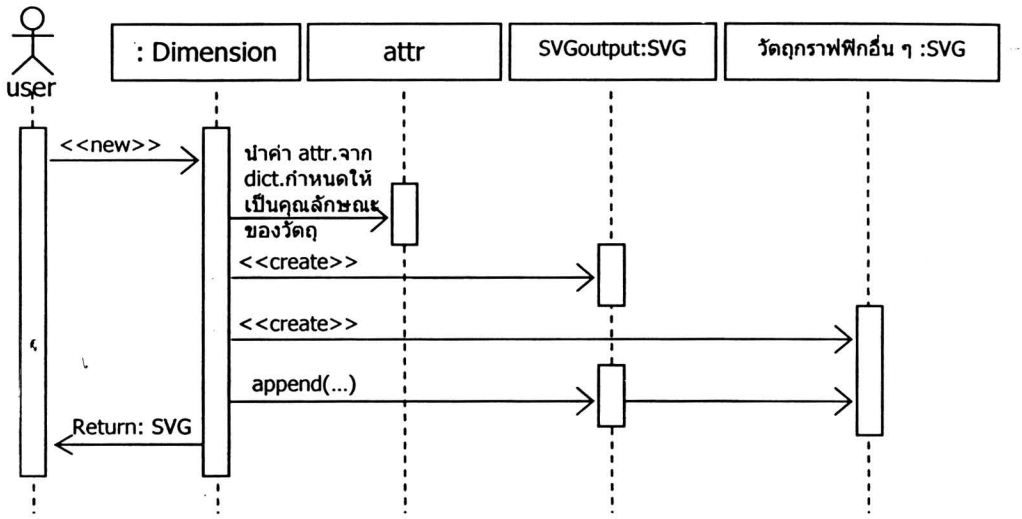
2.1. การออกแบบเชิงโครงสร้าง

เมื่อเริ่มทำการออกแบบแล้วจะได้โครงสร้างของคลาส Dimension ดังแสดงในภาพ

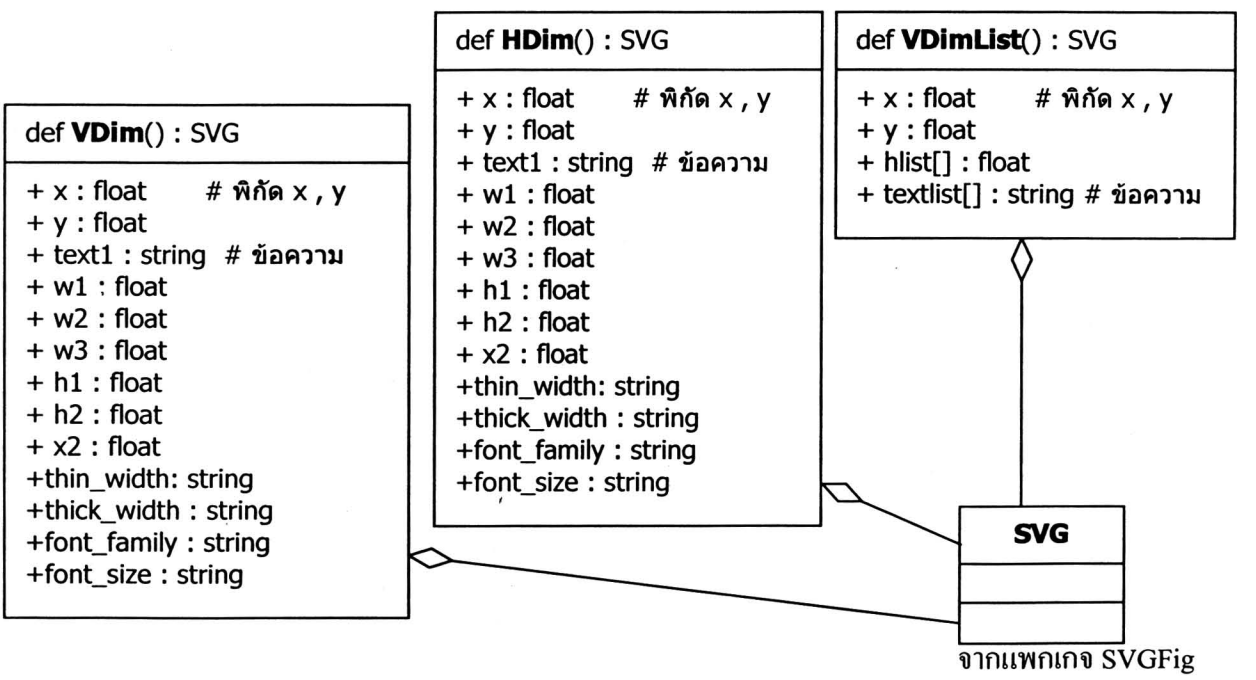


2.2. การออกแบบเชิงกิจกรรม

จากความต้องการของระบบ เราสามารถอธิบายกิจกรรมที่จะเกิดขึ้นในระบบคลาส Dimension ได้ดังนี้



คลาสไดอะแกรมของวัตถุเส้นบอกมิติ เป็นฟังค์ชันเสริมอีกชุด ซึ่งการออกแบบและการใช้งานจะออกแบบเหมือนกันใช้ผังกิจกรรมเดียวกันได้ จะแสดงผลการออกแบบไว้เพียงคลาสไดอะแกรม



จากแพกเกจ SVGFig

6. คลาสไดอะแกรมของเหล็กเสริมในหน้าตัดคาน

จะเป็นคลาสใช้ในการสร้างเหล็กเสริมในหน้าตัดคาน

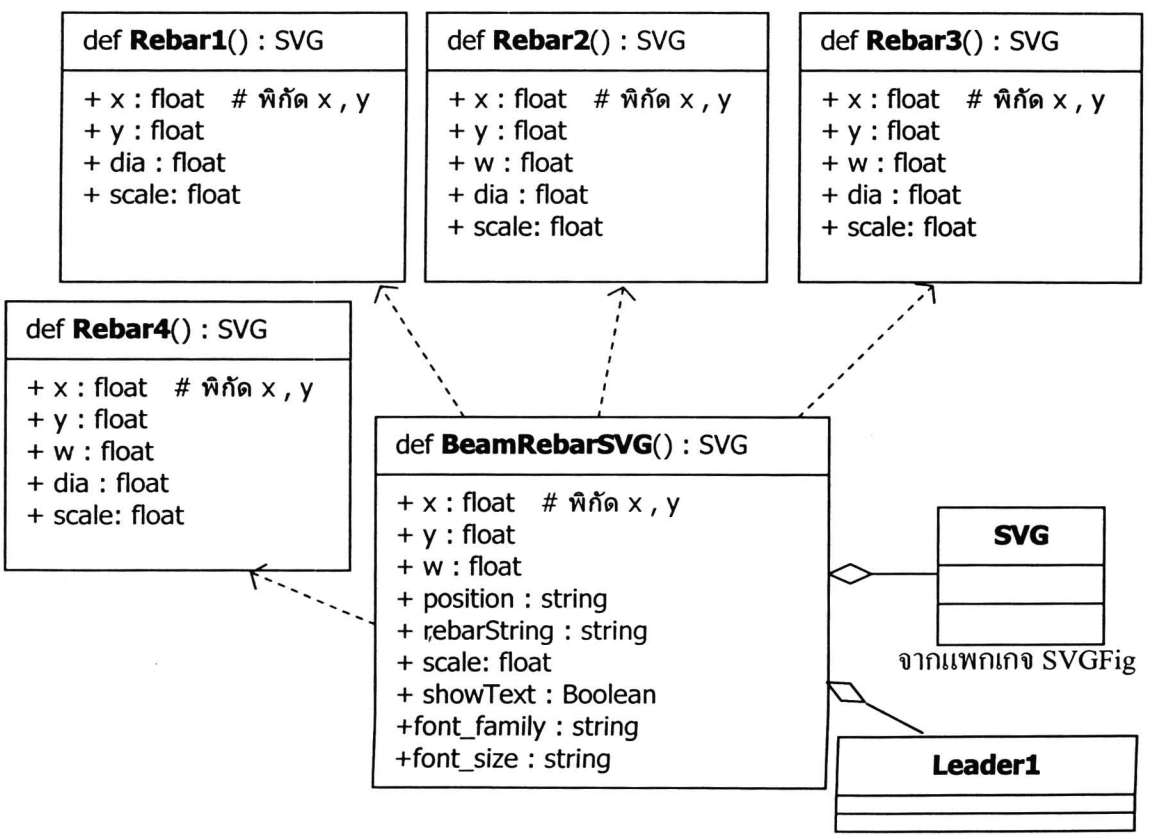
1. ความต้องการ (Requirements)

ต้องการให้รับข้อมูลเหล็กเสริมเป็นข้อความได้ โดยจะรับเป็นสองเทอม กั้นด้วยอักขระ '+' เช่น '2-RB15 + 1-RB15' พจน์แรกจะหมายถึงเหล็กเสริมหลัก และพจน์หลังจะหมายถึงเหล็กเสริมพิเศษ แล้วทำการแยกอักขระเพื่อดึงจำนวนเหล็กเสริมแล้วนำไปตีการวาดรูปวงกลมแทนหน้าตัดเหล็กเสริม โดยให้มีการตัดสินใจเลือกการเขียนออกมา จะออกแบบให้จำนวนเหล็กเสริมหลักเท่ากับ {2,3,4} และจำนวนเหล็กเสริมพิเศษเท่ากับ {1,2,3,4}

2. การออกแบบและพัฒนา

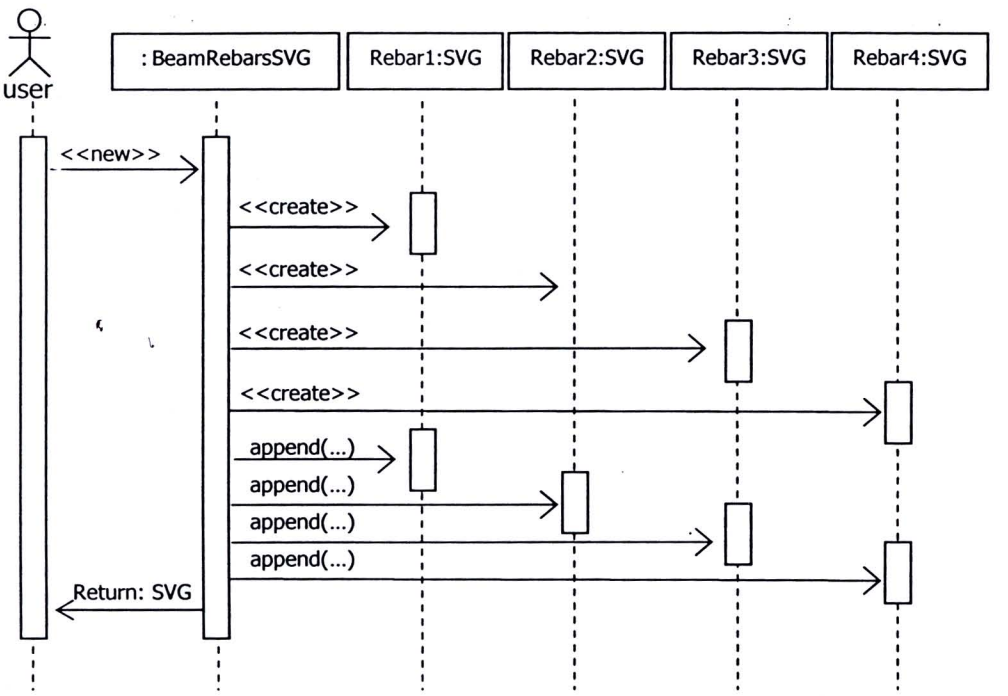
2.1. การออกแบบเชิงโครงสร้าง

เมื่อเริ่มทำการออกแบบโดยอาศัยแนวคิด Factory Method และ Composite ในการวาดเหล็กเสริมจะวาดเป็นชั้น ชั้นละ 1-4 เส้น ให้ฟังค์ชัน Rebar1, Rebar2, Rebar3 และ Rebar4 ทำหน้าที่ แล้วให้ BeamRebarSVG เป็นคลาสที่นำข้อความมาตีความแล้วตัดสินใจรูปแบบของการจัดเหล็กเสริม แล้วจะได้โครงสร้างของคลาส BeamRebars ดังแสดงในภาพ



2.2. การออกแบบเชิงกิจกรรม

จากความต้องการของระบบ เราสามารถอธิบายกิจกรรมที่จะเกิดขึ้นในระบบคลาส BeamRebars ได้ดังนี้



7. คลาสไดอะแกรมของเหล็กเสริม

จะเป็นคลาสใช้ในการสร้างเหล็กเสริมใช้งานในแผ่นพื้น บันได หรือฐานราก ดัดเป็นรูปทรงแตกต่างกันไป

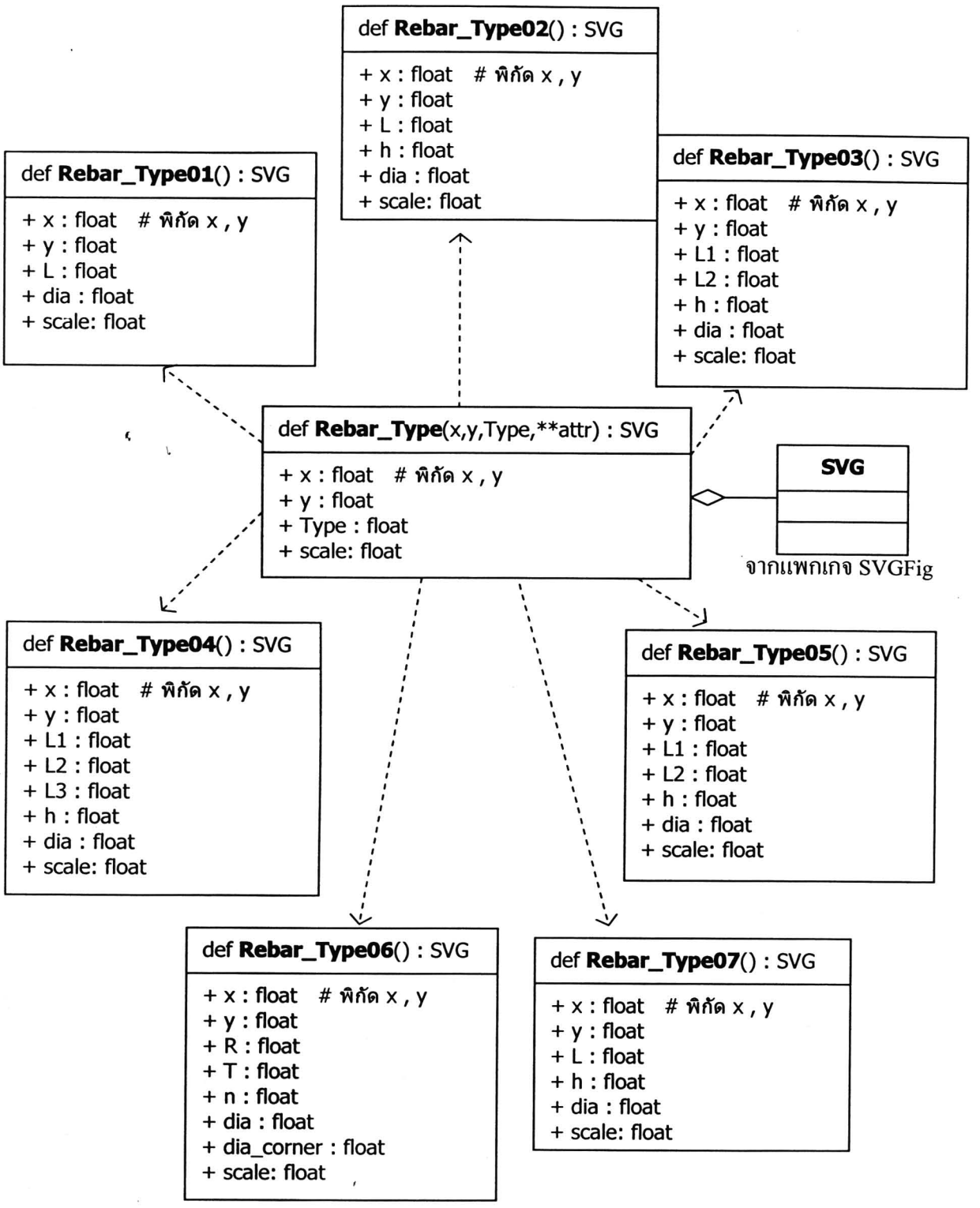
1. ความต้องการ (Requirements)

ต้องการให้ได้เหล็กเสริมรูปทรงต่าง ๆ กันในลักษณะของ Bar-Cut List ดังนั้นจะต้องออกแบบให้มีตัวแปรต่าง ๆ ให้สอดคล้องกับระบบของเหล็กเสริม แล้วจะควบคุมด้วย Factory Method อีกครั้ง

2. การออกแบบและพัฒนา

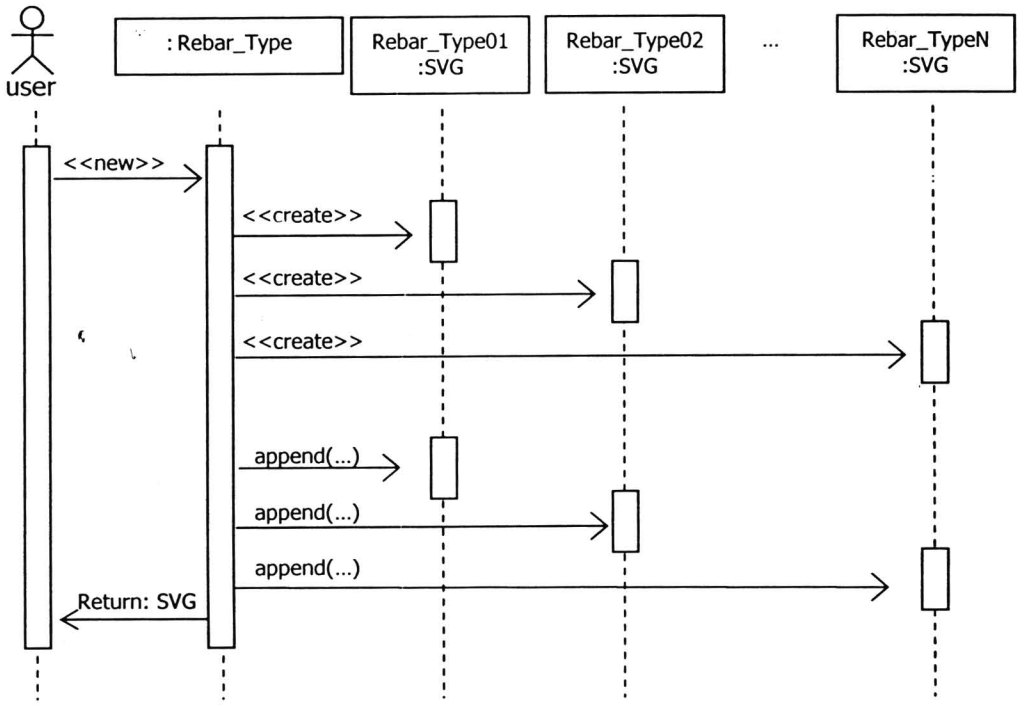
2.1. การออกแบบเชิงโครงสร้าง

เนื่องจากส่วนใหญ่เป็นเป็นเส้นเดียว จึงใช้อีลีเมนต์ <path> ของภาษา SVG แล้วนำตัวแปรต่าง ๆ ไปคำนวณหาพิกัดการวาด แล้วส่วนที่เป็นปลายแบบ Hook ก็จะคำนวณความยาวและรัศมีการดัดตามมาตรฐานของการเสริมเหล็ก จะได้คลาสไดอะแกรมดังแสดง


def Rebar_Type02() : SVG
+ x : float # พิกัด x , y
+ y : float
+ L : float
+ h : float
+ dia : float
+ scale: float
def Rebar_Type01() : SVG
+ x : float # พิกัด x , y
+ y : float
+ L : float
+ dia : float
+ scale: float
def Rebar_Type03() : SVG
+ x : float # พิกัด x , y
+ y : float
+ L1 : float
+ L2 : float
+ h : float
+ dia : float
+ scale: float
def Rebar_Type(x,y,Type,**attr) : SVG
+ x : float # พิกัด x , y
+ y : float
+ Type : float
+ scale: float
SVG
จากแพกเกจ SVGFig
def Rebar_Type04() : SVG
+ x : float # พิกัด x , y
+ y : float
+ L1 : float
+ L2 : float
+ L3 : float
+ h : float
+ dia : float
+ scale: float
def Rebar_Type05() : SVG
+ x : float # พิกัด x , y
+ y : float
+ L1 : float
+ L2 : float
+ h : float
+ dia : float
+ scale: float
def Rebar_Type06() : SVG
+ x : float # พิกัด x , y
+ y : float
+ R : float
+ T : float
+ n : float
+ dia : float
+ dia_corner : float
+ scale: float
def Rebar_Type07() : SVG
+ x : float # พิกัด x , y
+ y : float
+ L : float
+ h : float
+ dia : float
+ scale: float

2.2. การออกแบบเชิงกิจกรรม

จากความต้องการของระบบ เราสามารถอธิบายกิจกรรมที่จะเกิดขึ้นในระบบคลาส Rebar_Type ได้ดังนี้



8. คลาสไดอะแกรมของเหล็กเสริมแบบระยะห่าง

จะเป็นคลาสใช้ในการสร้างเหล็กเสริมแบบระยะห่าง ใช้งานในแผ่นพื้น บันได หรือฐานราก แสดงเป็นหน้าตัดวงกลม ระยะห่างตามที่ต้องการ

1. ความต้องการ (Requirements)

ต้องการให้ได้เหล็กเสริมวางที่ระยะห่างกัน โดยจะออกแบบเป็นสองชุดฟังค์ชัน ในฟังค์ชันแบบแรกรับค่าระยะห่าง และจำนวนเหล็กเสริม ฟังค์ชันชุดที่สองจะรับค่าความกว้างและระยะห่างแล้วจะคำนวณจำนวนเหล็กเอง

2. การออกแบบและพัฒนา

2.1. การออกแบบเชิงโครงสร้าง

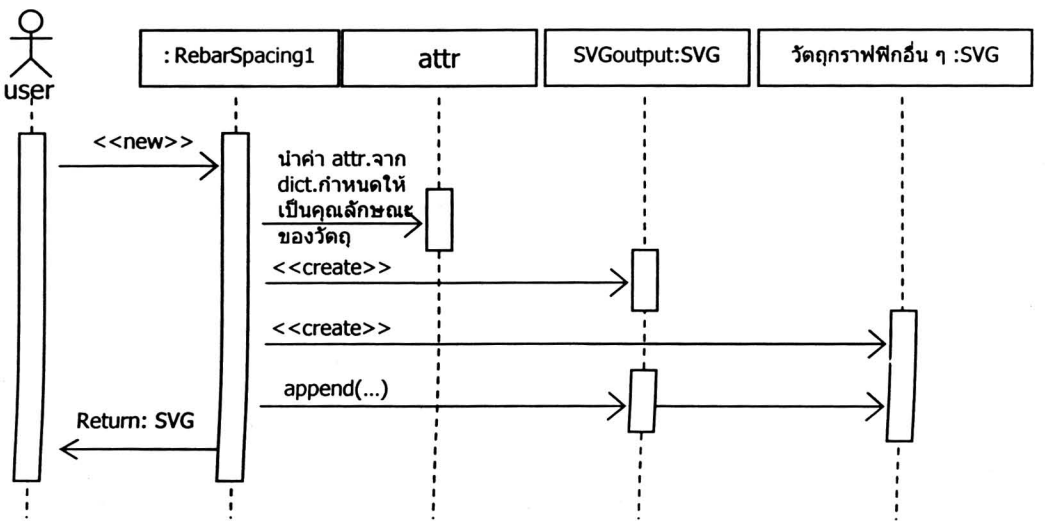
เป็นการออกแบบโดยเป็นชุดของ SVG วงกลมมาวางเรียงกันเป็นระยะ จะได้คลาสไดอะแกรม RebarSpacing ดังแสดง

def **RebarSpacing1** (x,y,**attr) : SVG
+ x : float # พิกัด x , y + y : float + dia : float + num : integer + Spacing ;float + scale-x: float + scale-y: float + stroke : string

def **RebarSpacing2** (x,y,**attr) : SVG
+ x : float # พิกัด x , y + y : float + Rebar: string + width : float + scale-x: float + scale-y: float + n-leader: integer + SHowLeader : boolean + stroke : string

2.2. การออกแบบเชิงกิจกรรม

จากความต้องการของระบบ เราสามารถอธิบายกิจกรรมที่จะเกิดขึ้นในระบบคลาส RebarSpacing1 และ RebarSpacing2 ได้ดังนี้



9. คลาสไดอะแกรมของวัตถุเหล็กเสริมในหน้าตัดคาน

จะเป็นคลาสใช้ในการสร้างเส้นบอกมิติ เพื่อใช้ในการบอกขนาดของชิ้นส่วน

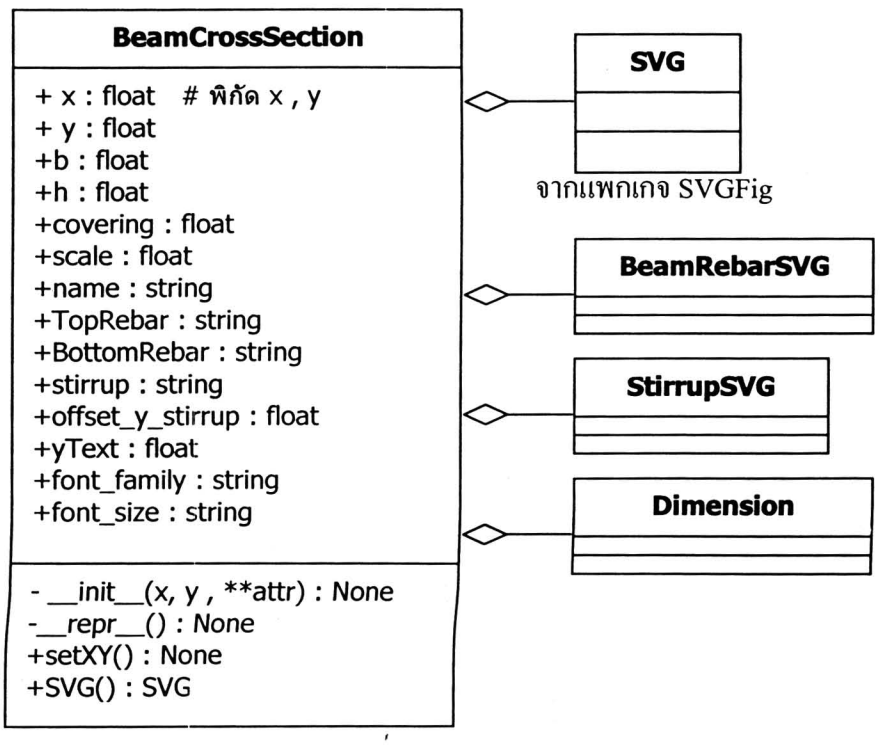
1. ความต้องการ (Requirements)

ในคลาสนี้จะออกแบบให้ผู้ใช้สร้างหน้าตัดคานได้โดยกำหนดขนาดความกว้างความลึกของหน้าตัดคาน และระบุเหล็กเสริมในลักษณะของข้อความที่สามารถนำไปตีความได้ เช่น '2-RB15 + 1-RB15' จะมีเครื่องหมาย + เป็นตัวกั้นแยกระหว่างเหล็กเสริมหลัก กับเหล็กเสริมพิเศษ เป็นต้น ดังนั้นจะส่งผ่านให้ คลาส BeamRebarSVG จัดการให้ และเหล็กปลอกก็จะส่งเป็นพารามิเตอร์เป็นข้อความให้กับคลาส StirrupSVG และเรียกใช้งานคลาส Dimension

2. การออกแบบและพัฒนา

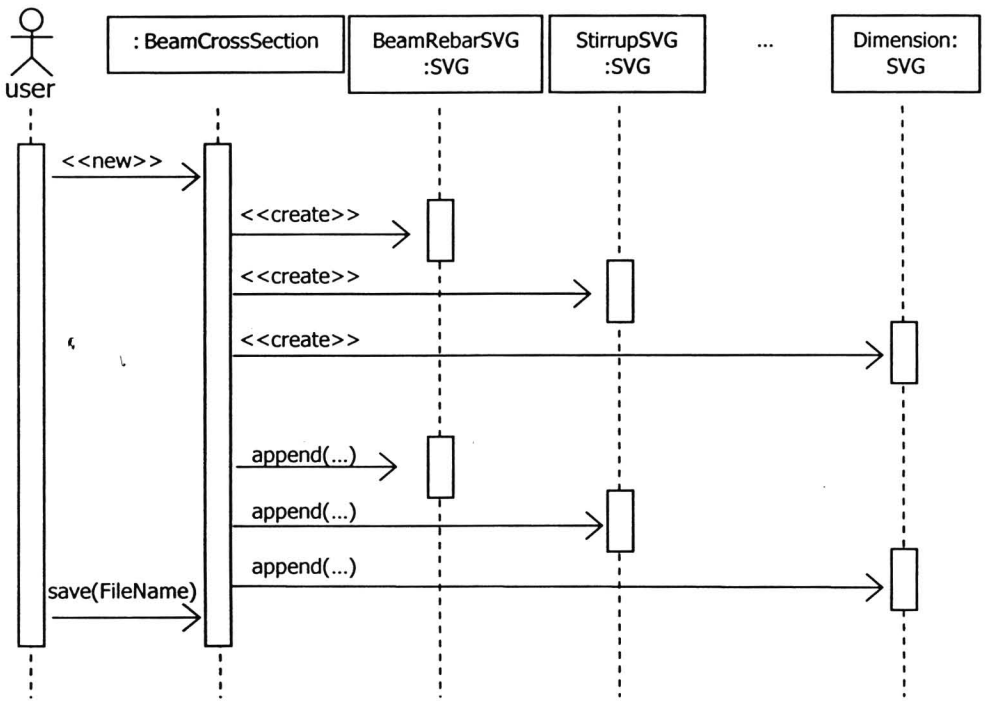
2.1. การออกแบบเชิงโครงสร้าง

เมื่อเริ่มทำการออกแบบแล้วจะได้โครงสร้างของคลาส BeamCrossSection ดังแสดงในภาพ



2.2. การออกแบบเชิงกิจกรรม

จากความต้องการของระบบ เราสามารถอธิบายกิจกรรมที่จะเกิดขึ้นในระบบคลาส BeamCrossSection ได้ดังนี้

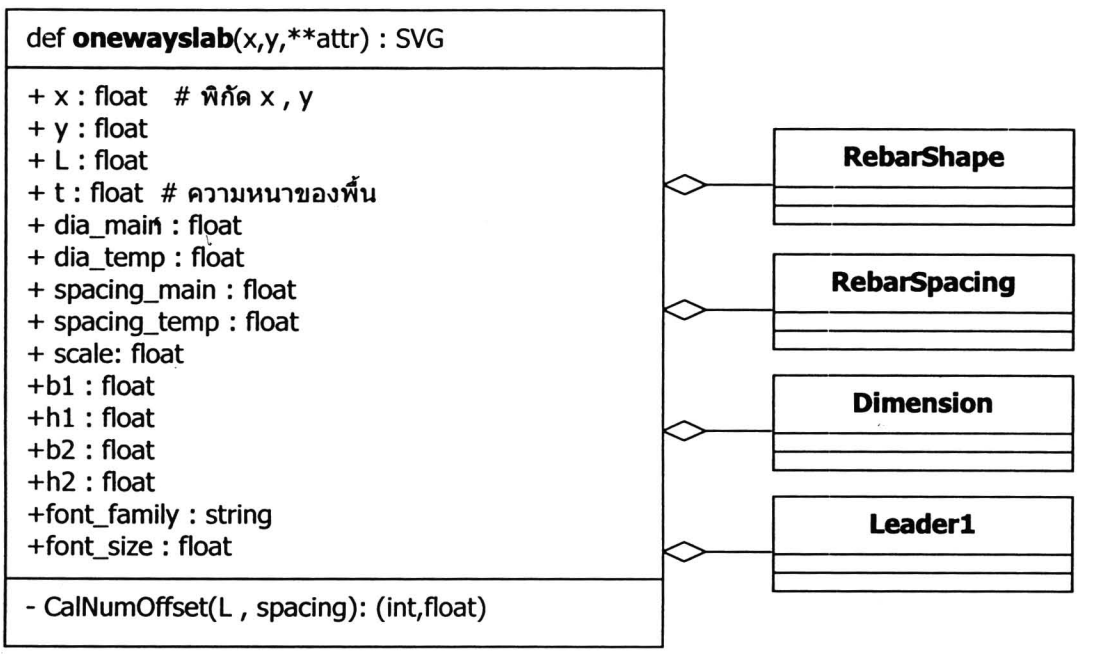


คลาสไดอะแกรม Stirrup จะใช้ในการเขียนเหล็กปลอกสำหรับหน้าตัดคาน โดยจะให้ค่าเป็นข้อความแล้วนำไปเขียน เช่น ป.RB6@200mm หรือ 2-ป.RB6@200mm ออกแบบให้สามารถเขียนเหล็กปลอกได้แบบวงเดียว และแบบสองวง โดยจะคำนวณหาค่าระยะห่างภายในเมื่อเสริมเหล็กเสริมหลักจำนวนเส้นต่างกัน {2,3,4}

def **StirrupSVG** (x,y, stirrupString,**attr) : SVG
+ x : float # พิกัด x , y + y : float + stirrupString: string + diaMain : float + numMain : integer + offset_y_text :float + scale: float +font_family : string +font_size : string

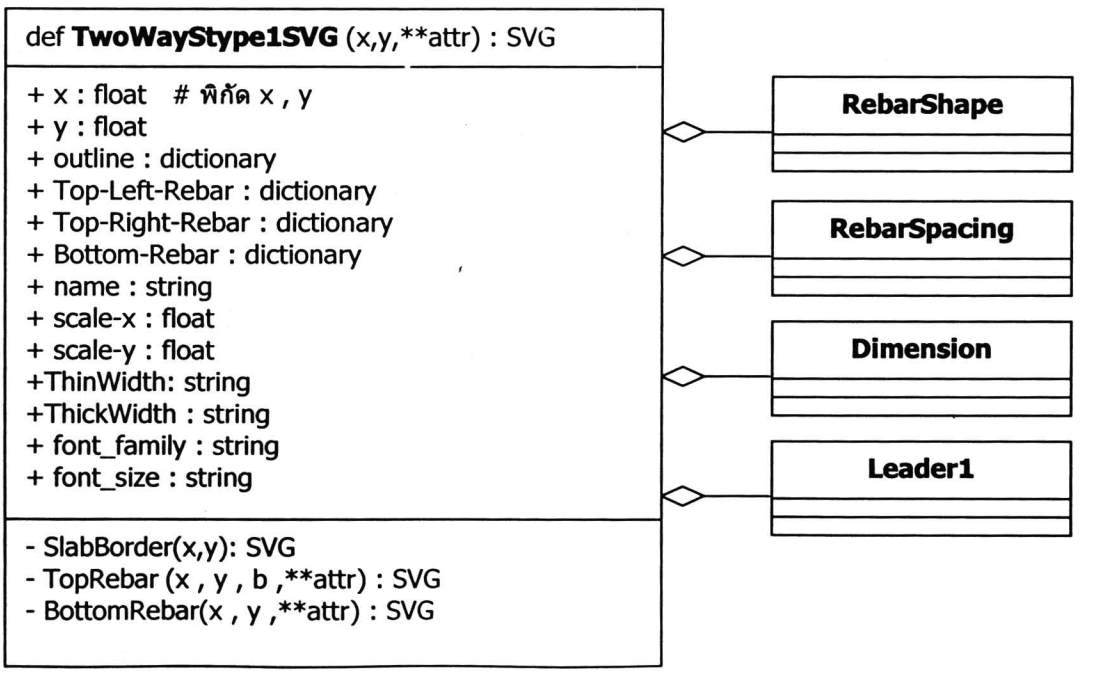
10. คลาสไดอะแกรมของแผ่นพื้นทางเดียว

คลาสไดอะแกรม OneWaySlab ใช้ในการเขียนรูปแผ่นพื้นคอนกรีตเสริมเหล็กแบบทางเดียว สามารถกำหนดความยาว ความหนา และป้อนระยะห่าง และขนาดเส้นผ่านศูนย์กลางของเหล็กเสริม และจะมีฟังค์ชันภายใน CalNumOffset เพื่อใช้ในการคำนวณหาค่าการเยื้องของเหล็ก ทำให้ได้เหล็กเสริมอยู่ตรงกลางโดยอัตโนมัติ



11. คลาสไดอะแกรมของแผ่นพื้นสองทาง

คลาสไดอะแกรม TwoWayStype1SVG จะวาดหน้าตัดของแผ่นพื้นสองทาง โดยออกแบบให้สามารถตัวแปรเป็นชุดในลักษณะของ Dictionary ทำให้กำหนดค่าผ่านตัวแปรได้และเข้าใจได้ง่าย ในกรณีที่ ระบุว่า direction ของเหล็กเสริมต่างกันจะทำให้วาดตำแหน่งของเหล็กเสริมสลับกันได้



outline : dictionary
+ L : float + t : float # ความหนาของพื้น + L1 : float + L2 : float +b1 : float +h1 : float +b2 : float +h2 : float

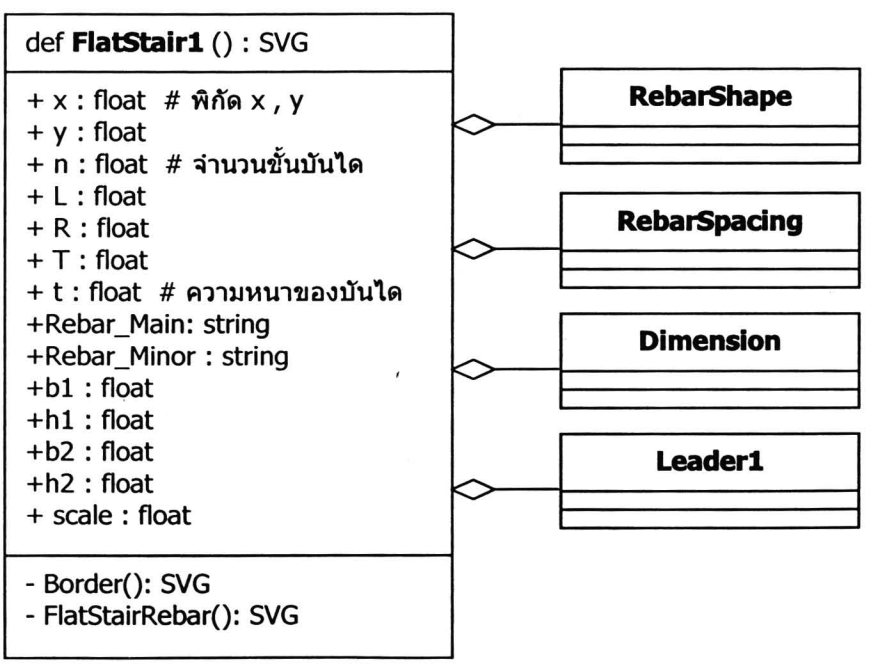
Bottom-Rebar : dictionary
+ main : string + second: string + direction : string

Top-Left-Rebar : dictionary
+ main : string + Left-Rebar : string + Right-Rebar : string + L1 : float + L2 : float

Top-Right-Rebar :
+ main : string + Left-Rebar : string + Right-Rebar : string + L1 : float + L2 : float

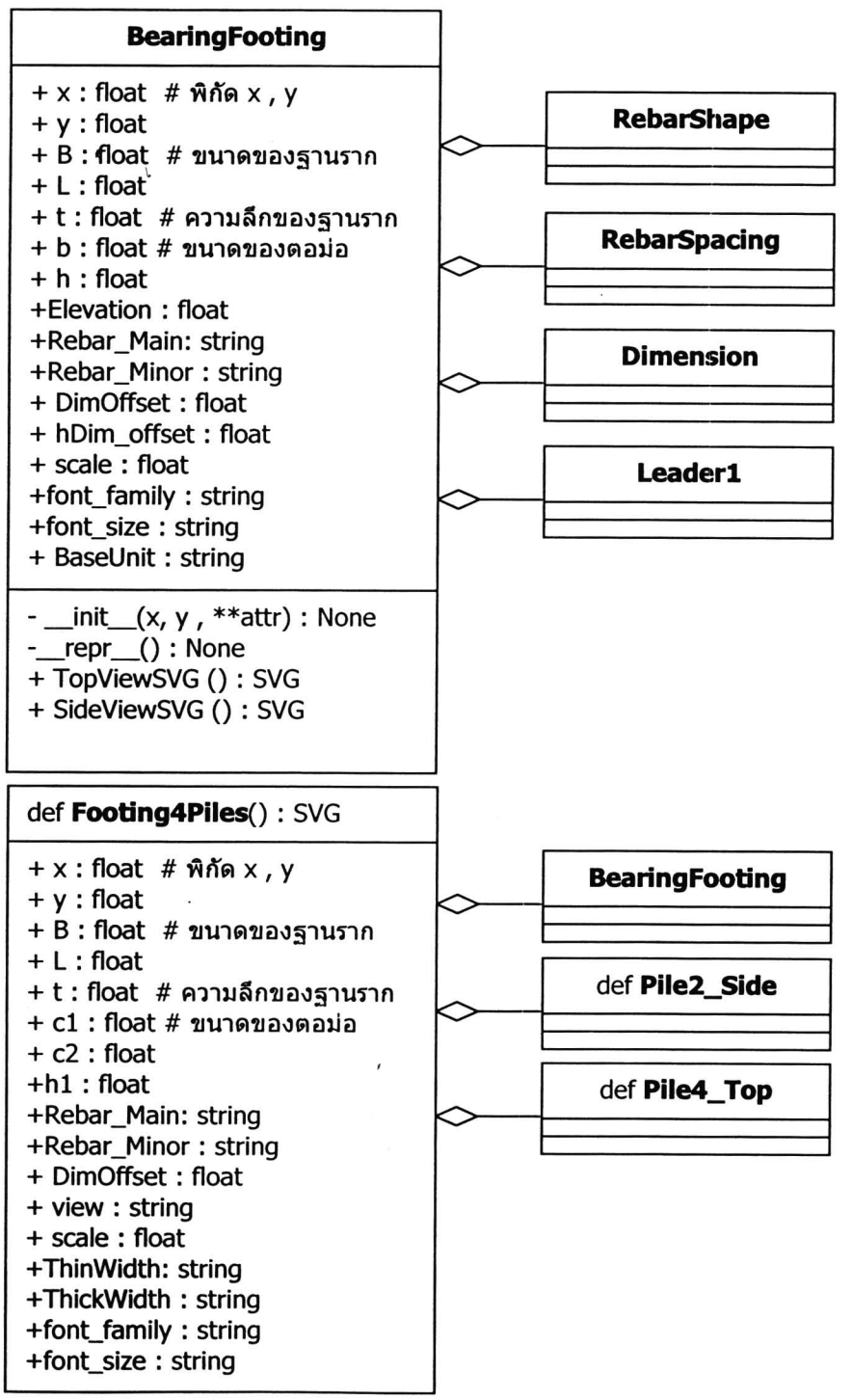
12. คลาสไดอะแกรมของบันไดท้องเรียบ

คลาสไดอะแกรม FlatStair1 จะวาดบันไดแบบท้องเรียบชนิดมีชานพักบนล่าง ในการเขียนกรอบของบันได จะต้องใช้การคำนวณจุดพิกัดค่อนข้างมากและ ส่งข้อมูลให้กับอิลีเมนต์ <path> ของภาษา SVG และใช้คลาส RebarShapes ในการเขียนเหล็กเสริม ดังนั้นต้องคำนวณตำแหน่งและความยาวของเหล็กเสริมที่มีการงอเอียงทำมุม และเหล็กเสริมกันร้าวก็จะใช้งานคลาส RebarSpacing ที่คืนค่าเป็น SVG มาก่อนแล้วค่อยอาศัยความสามารถให้การ Transform ของภาษา SVG ในการหมุนวัตถุอีกครั้ง



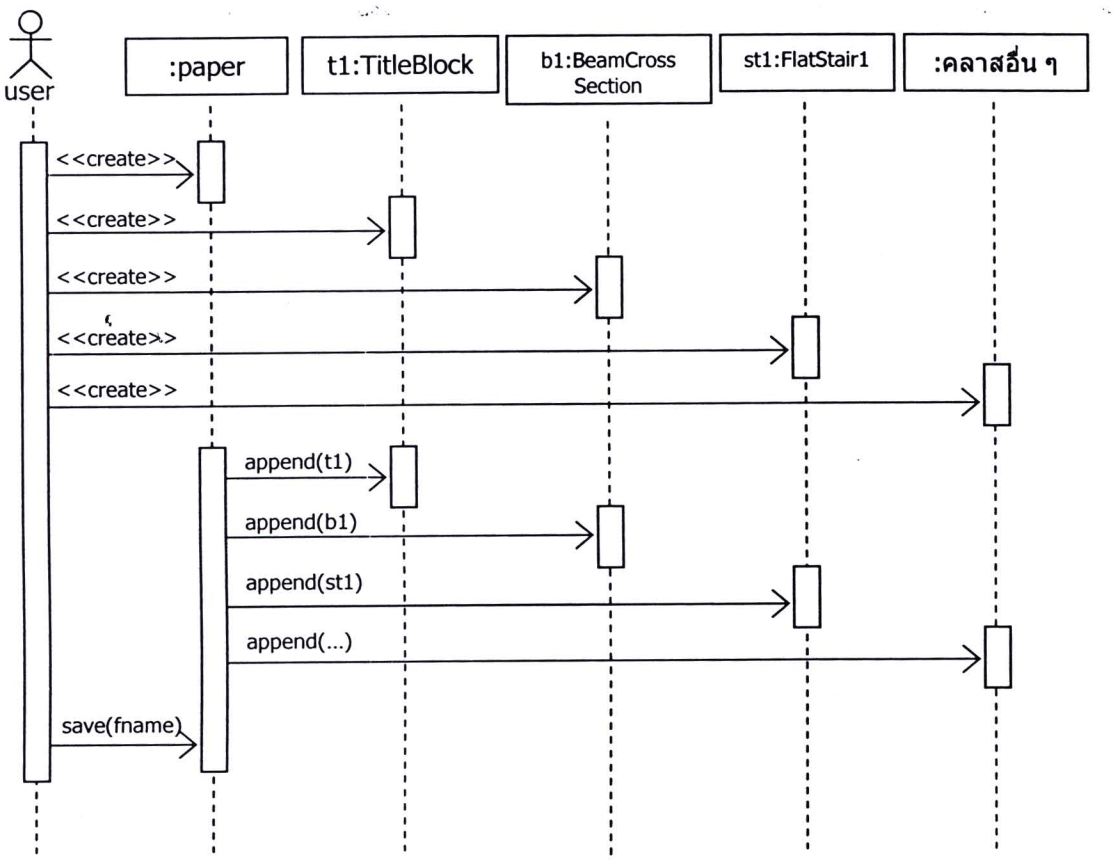
13. คลาสไดอะแกรมของฐานรากแบบแผ่ และวางบนเสาเข็ม

คลาสไดอะแกรม BearingFooting จะเขียนรายละเอียดของฐานรากแบบแผ่ โดยแสดงผลได้ทั้งรูปมองจากด้านบน(เมทรอดTopViewSVG) และมองจากด้านข้าง(เมทรอด SideViewSVG) จะใช้งานจากคลาสพื้นฐานต่างๆ นำมาประกอบเป็นรูปภาพ สำหรับคลาสไดอะแกรม Footing4Piles จะดึงคลาส BearingFooting ไปเป็นคลาสพื้นฐานที่นำมาใช้งานซ้ำ(Reusable) แล้วจึงค่อยเขียนเสาเข็มทับลงไป จึงเพียงออกแบบและเขียนคำสั่งของวัตถุเสาเข็มเพิ่มเติมเท่านั้น

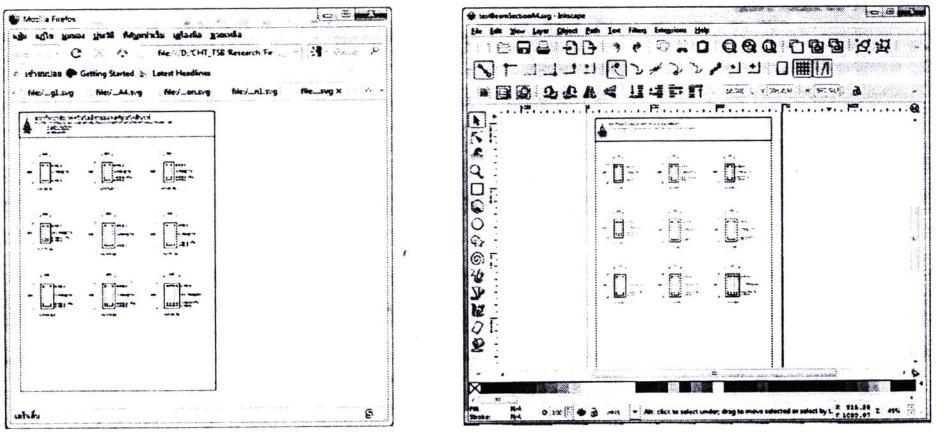


14. คลาส pyDrawingSVG

ในแพคเกจ pyDrawingSVG นี้จะใช้ลักษณะของ Design Patterns แบบ Factory Method ร่วมกับแบบ Composite แผนภาพกิจกรรมที่เกิดขึ้นเมื่อผู้ใช้นำแพคเกจไปใช้งาน



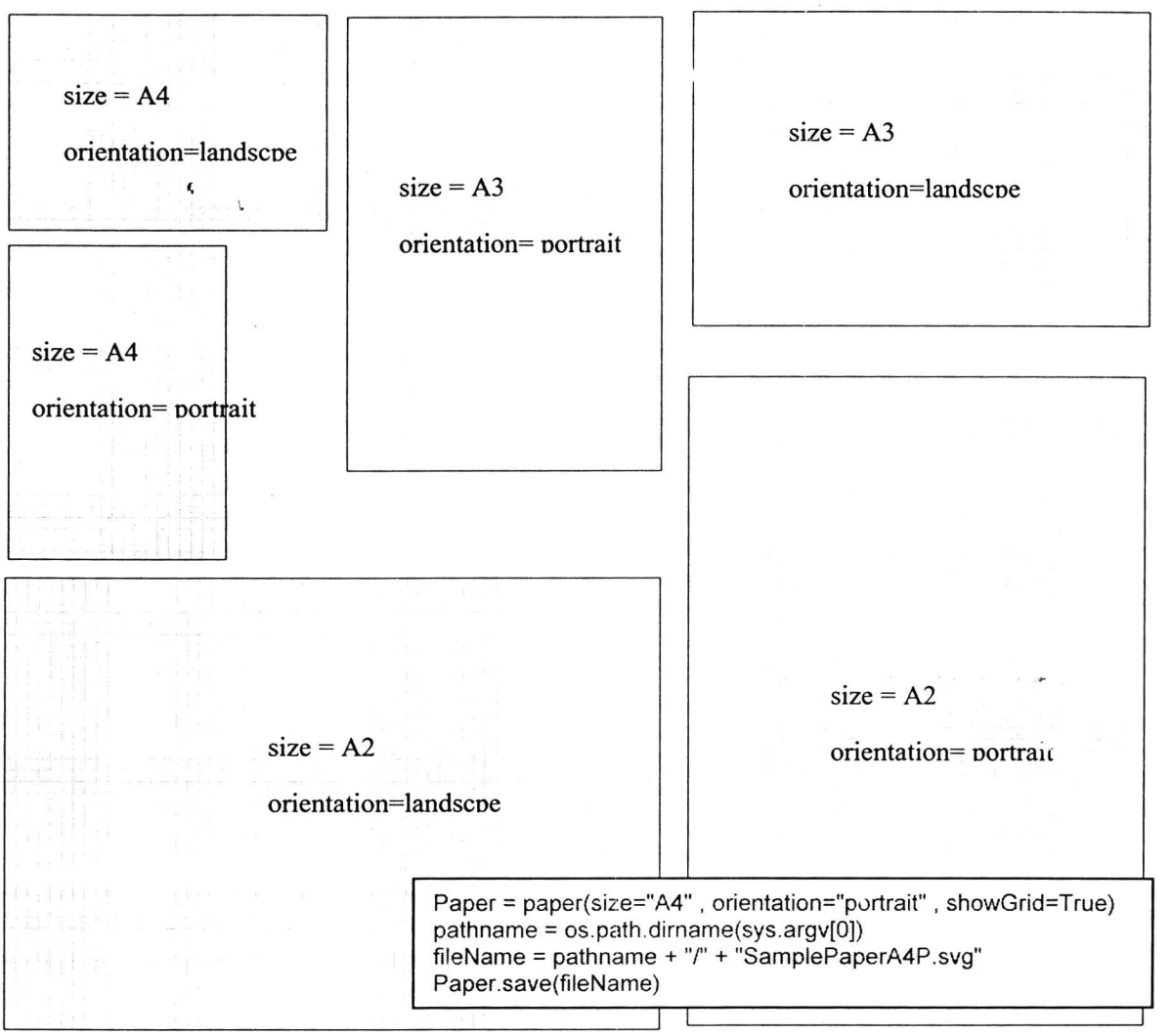
เมื่อรันแล้วจะได้ไฟล์ SVG เป็นผลลัพธ์ ซึ่งต้องนำไปเปิดดูผลด้วยโปรแกรม Mozilla FireFox หรือโปรแกรม Inkscape



ภาพที่ 23 การแสดงผลไฟล์ภาษา SVG ด้วยโปรแกรม Mozilla FireFox และโปรแกรม Inkscape

15. โมดูลต่าง ๆ ในแพคเกจ pyDrawingSVG

โมดูล SVGDrawingUtility การเรียกใช้งานคลาส paper จะเป็นการเรียกใช้งานกระดาษต่าง ๆ A4,A3,A2 และแสดงเส้นกริด โดยสามารถสั่งให้กระดาษเป็นแบบแนวนอนหรือแนวตั้งก็ได้ และวัตถุ paper นี้จะเป็นคลาสหลักที่ใช้งานแล้วเมื่อต้องการเพิ่มวัตถุใดเพิ่มเติม เราจะทำการสร้างแล้วเพิ่มเข้าไปในวัตถุกระดาษ วัตถุ paper จะจัดการสร้างไฟล์ SVG ของชิ้นส่วนลูกทั้งหมดแล้วบันทึกลงไฟล์



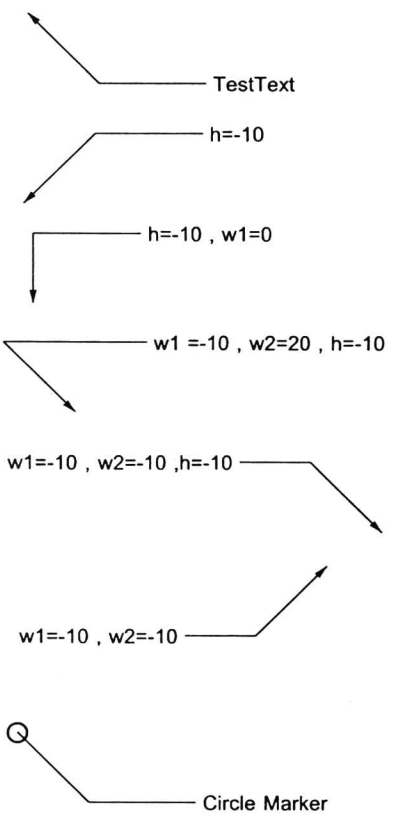
ภาพที่ 24 ผลการรันคลาส paper ที่ได้กระดาษขนาด A2, A3 และ A4 ทั้งแนวตั้งและแนวนอน

โมดูล Dimension


```
dim1 = Dimension(50 , 50 , width=50 , text = u'ทดสอบ slash')
dim2 = Dimension(50 , 60 , width=50 , text = u'ทดสอบ circle' ,
                                start='circle' ,end='circle' )
dim3 = Dimension(50 , 70 , text =  u'ทดสอบ arrowhead' ,
                                 start='arrowhead' ,end='arrowhead')
dim4 = Dimension(50 , 80 , text =  u'ทดสอบ triangle',
                                 start='triangle' ,end='triangle')
```

ภาพที่ 25 ผลลัพธ์ที่ได้จากคลาส Dimension กับลักษณะของปลายเส้นชี้แบบต่าง ๆ

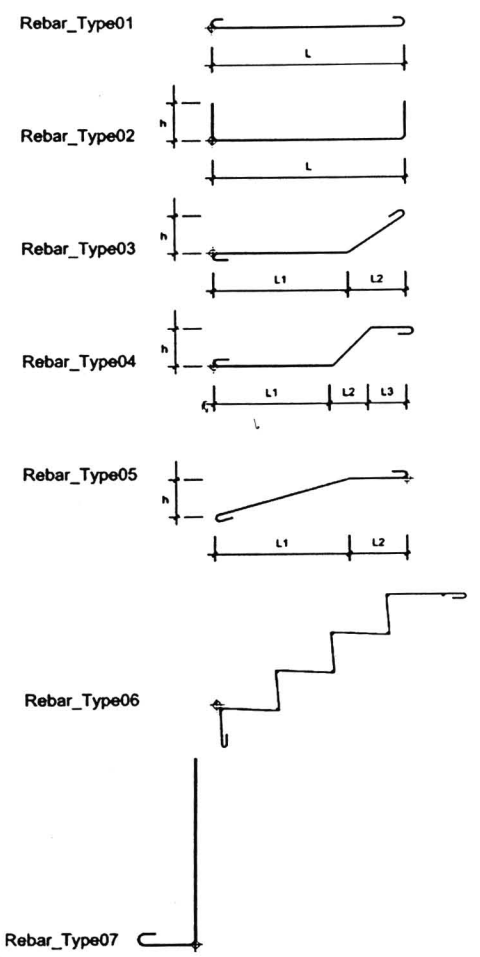
โมดูล Leaders




```
lead1=leader1(80,50 , 'TestText')

lead2=leader1(30,80 , u'h=-10' , h=-10)

lead2_2=leader1(80,80 , u'h=-10 , w1=0' ,w1 =0 , h=-10)

lead2_3=leader1(140,80 , u'w1 =-10 , w2=20 , h=-10' ,
                                w1 =-10 , w2=20 , h=-10)

lead3=leader1(80,100 , u' w1=-10 , w2=-10 ,h=-10' ,
                                w1=-10 , w2=-10 ,h=-10)

lead4=leader1(80,110 , u' w1=-10 , w2=-10' , w1=-10 , w2=-10 )

lead1c=leader1(80,150 , 'Circle Marker' , marker_type='Circle1')
```

ภาพที่ 26 ผลลัพธ์ และคำสั่งของการเรียกใช้ฟังค์ชัน leader1

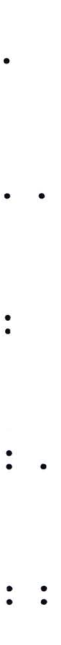
โมดูล RebarShapes




```
rebar_sample1 = Rebar_Type01(x1 + 50 ,y1 ,L= 1000. ,
                             dia=9. , scale=20 )

rebar_sample2 = Rebar_Type02(x1+50 , y1 +dy ,
                L=1000. , h=200. , dia=9. , scale=20)

rebar_sample3 = Rebar_Type03(x1+50 , y1 +2*dy ,
                L1=700. , L2=300. , h=200. )

rebar_sample4 = Rebar_Type04(x1+50 , y1 +3*dy ,
                L1=600. , L2=200., L3=200. , h=200 ,
                scale=20)

rebar_sample5 = Rebar_Type05(
        x1+50 + 1000/20. , y1 +4*dy ,
        L1=700. , L2=300. , h=200. . ,scale=20)

rebar_sample6 = Rebar_Type06(x1+50  , y1 +6*dy ,
   n=4 , R=200. , T=290. ,
   dia=6., dia_corner=9. ,scale=20)

rebar_sample7 = Rebar_Type07(x1+50  , y1 +1*dy  ,
                L=1000. , h=200. , dia=9. , scale=20)
```

ภาพที่ 27 ผลลัพธ์ของการรันฟังค์ชันในโมดูล RebarShape

โมดูล BeamRebars

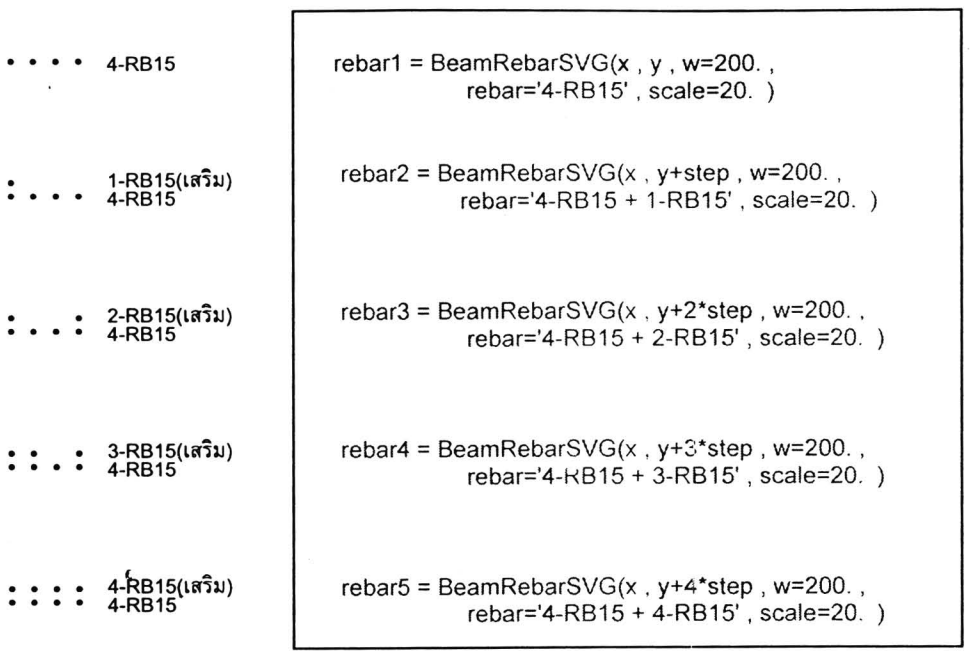



```
rebar1 = BeamRebarSVG(x , y , w=200. ,
                      rebar='2-RB15' , scale=20. )

rebar2 = BeamRebarSVG(x , y+step , w=200. ,
                      rebar='2-RB15 + 1-RB15' , scale=20. )

rebar3 = BeamRebarSVG(x , y+2*step , w=200. ,
                      rebar='2-RB15 + 2-RB15' , scale=20. )

rebar4 = BeamRebarSVG(x , y+3*step , w=200. ,
                      rebar='2-RB15 + 3-RB15' , scale=20. )

rebar5 = BeamRebarSVG(x , y+4*step , w=200. ,
                      rebar='2-RB15 + 4-RB15' , scale=20. )
```

ภาพที่ 28 ผลลัพธ์และคำสั่งสำหรับเขียนเหล็กเสริมในหน้าตัดคานแบบเหล็กเสริมหลักจำนวนสองเส้น

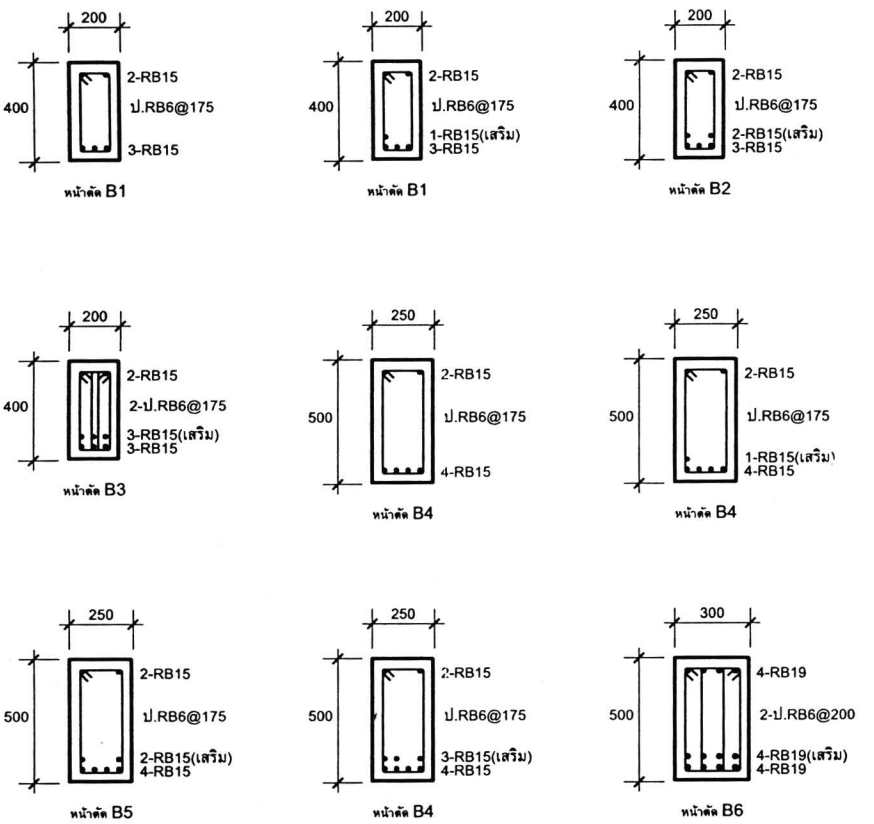

```
rebar1 = BeamRebarSVG(x , y , w=200. ,
                      rebar='3-RB15' , scale=20. )

rebar2 = BeamRebarSVG(x , y+step , w=200. ,
                      rebar='3-RB15 + 1-RB15' , scale=20. )

rebar3 = BeamRebarSVG(x , y+2*step , w=200. ,
                      rebar='3-RB15 + 2-RB15' , scale=20. )

rebar4 = BeamRebarSVG(x , y+3*step , w=200. ,
                      rebar='3-RB15 + 3-RB15' , scale=20. )
```

ภาพที่ 29 ผลลัพธ์และคำสั่งสำหรับเขียนเหล็กเสริมในหน้าตัดคานแบบเหล็กเสริมหลักจำนวนสามเส้น



ภาพที่ 30 ผลลัพธ์และคำสั่งสำหรับเขียนเหล็กเสริมในหน้าตัดคานแบบเหล็กเสริมหลักจำนวนสี่เส้น

โมดูล BeamCrossSection

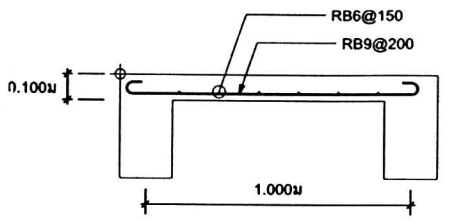


ภาพที่ 31 ผลลัพธ์ที่ได้จากคำสั่งของคลาส BeamSection

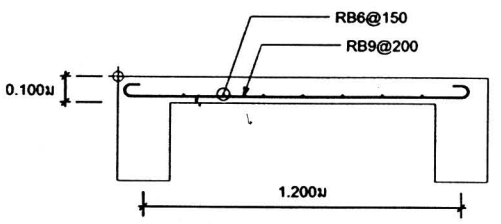
ตัวอย่างรหัสคำสั่งที่ใช้ในการสร้างหน้าตัดคาน

```
b2_0 = BeamCrossSection(x1  + 2*dx , y1 + 3*dy , b=200 , h=400,
                TopRebar='2-RB15' , BottomRebar='2-RB15',name = 'B1')
b3_0 = BeamCrossSection(x1  + 2*dx , y1 + 3*dy , b=200 , h=400,
                TopRebar='2-RB15' , BottomRebar='3-RB15',
                stirrup=u'ปRB6@175' , name = 'B1')
b3_1 = BeamCrossSection(x1  + 2*dx , y1 + 3*dy , b=200 , h=400,
                TopRebar='2-RB15' , BottomRebar='3-RB15 + 1-RB15',
                stirrup=u'ปRB6@175' , name = 'B1')
b3_2 = BeamCrossSection(x1  + 2*dx , y1 + 3*dy , b=200 , h=400,
                TopRebar=u'2-RB15' , BottomRebar='3-RB15 + 2-RB15',
                stirrup=u'ปRB6@175' , name = 'B2')
b3_3 = BeamCrossSection(x1  + 2*dx , y1 + 3*dy , b=200 , h=400,
                TopRebar='2-RB15' , BottomRebar='3-RB15 + 3-RB15',
                stirrup=u'2-ปRB6@175' , name = 'B3')
b4_0 = BeamCrossSection(0 , 0 , b=250 , h=500,
                TopRebar='2-RB15' , BottomRebar='4-RB15',
                stirrup=u'ปRB6@175' , name = 'B4')
b4_1 = BeamCrossSection(0 , 0 , b=250 , h=500,
                TopRebar='2-RB15' , BottomRebar='4-RB15 + 1-RB15',
                stirrup=u'ปRB6@175' , name = 'B4')
b4_2 = BeamCrossSection(x1  + 2*dx , y1 + 3*dy , b=250 , h=500,
                TopRebar='2-RB15' , BottomRebar='4-RB15 + 2-RB15',
                stirrup=u'ปRB6@175' ,name = 'B5')
b4_3 = BeamCrossSection(0 , 0 , b=250 , h=500,
                TopRebar='2-RB15' , BottomRebar='4-RB15 + 3-RB15',
                stirrup=u'ปRB6@175' ,name = 'B4')
b4_4 = BeamCrossSection(x1  + 2*dx , y1 + 3*dy , b=300 , h=500,
                TopRebar='4-RB19' , BottomRebar='4-RB19 + 4-RB19',
                stirrup=u'2-ปRB6@200' ,name = 'B6')
```

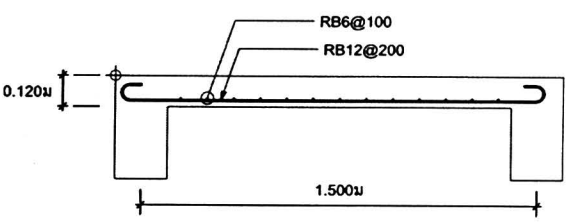
โมดูล OneWaySlab



ขยาย S1
มาตราส่วน 1:20



ขยาย S2
มาตราส่วน 1:20

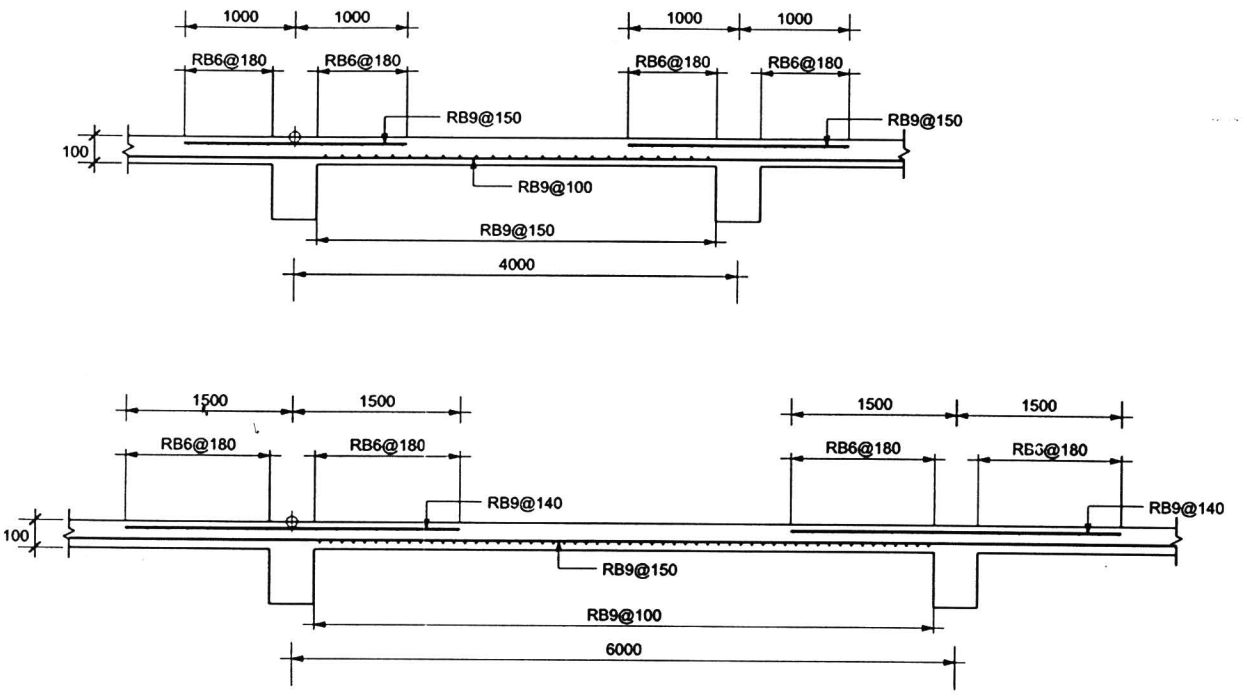


ขยาย S3
มาตราส่วน 1:20

```
s1 = onewayslab(50 , 50 , L=1000. ,
          name='S1' ,spacing_temp=150)

s2 = onewayslab(50 , 110 , L=1200. ,
          name='S2' , spacing_temp=150)

s3 = onewayslab(50 , 180 ,
      L=1500. , t=120. ,
      dia_main=12. , dia_temp=6. ,
      spacing_temp=100  , name='S3')
```

ภาพที่ 32 ผลลัพธ์ที่ได้จากคำสั่งของคลาส OneWaySlab

โมดูล TwoWaysSlabStyle1

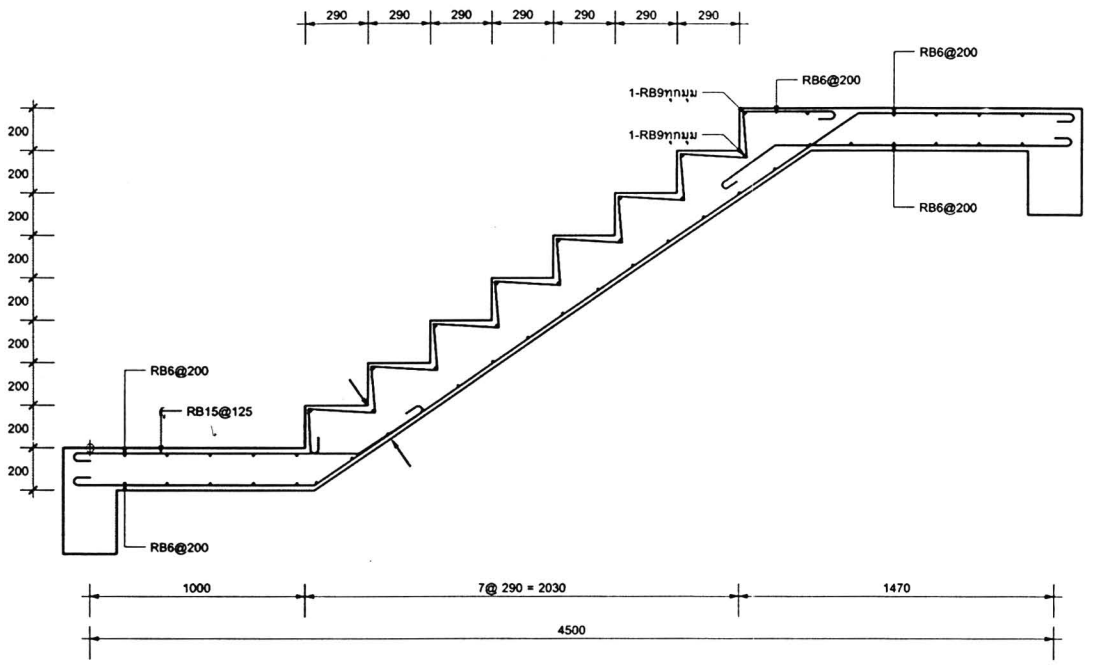


ภาพที่ 33 ภาพแสดงผลจากการใช้งานคลาส TwoWaysSlabStyle1

รหัสคำสั่งในการใช้งานคลาส TwoWaysSlabStyle1

```
s1 = TwoWayStype1SVG(90, 60 ,
                Bottom_Rebar=dict(main='RB9@100' , second='RB9@150' )
        )
s2 = TwoWayStype1SVG(90, 130 ,
                outline=dict(L=6000. , L1=2000 , L2=2000) ,
                Top_Left_Rebar =dict(main='RB9@140',L1=1500 , L2 = 1500) ,
                Top_Right_Rebar =dict(main='RB9@140',L1=1500 , L2 = 1500) ,
                Bottom_Rebar=dict(main='RB9@150' , second='RB9@100' , direction='long' )
```

โมดูล FlatStair1

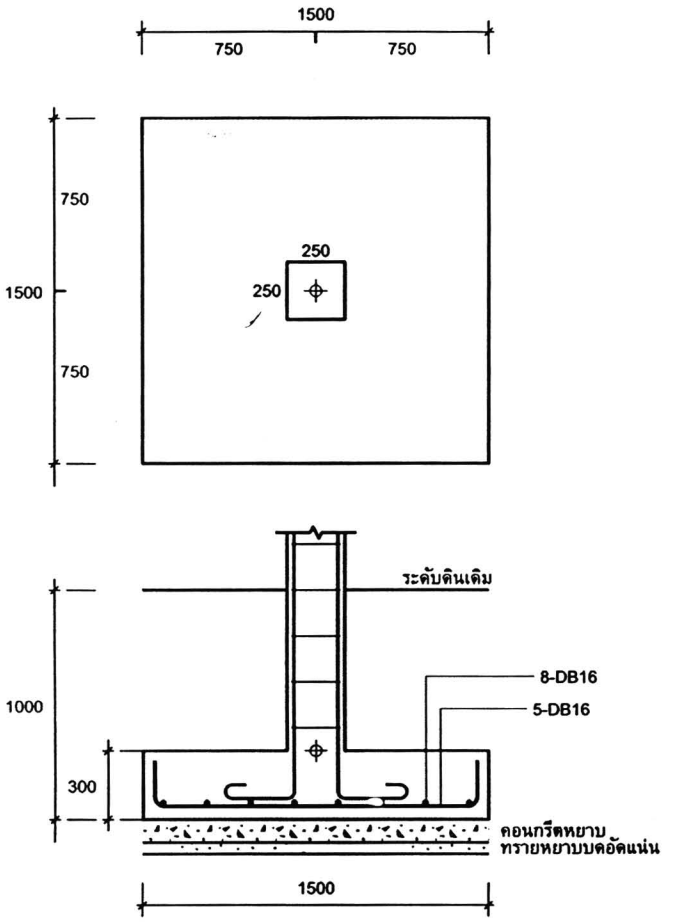


ภาพที่ 34 ภาพแสดงผลจากการใช้งานคลาส FlatStair1

รหัสคำสั่งในการใช้งานคลาส FlatStair1

```
stair1 = FlatStair1(40,130 , n=8 , L=4500. , t = 200. , L1=1000.)
```

โมดูล BearingFooting

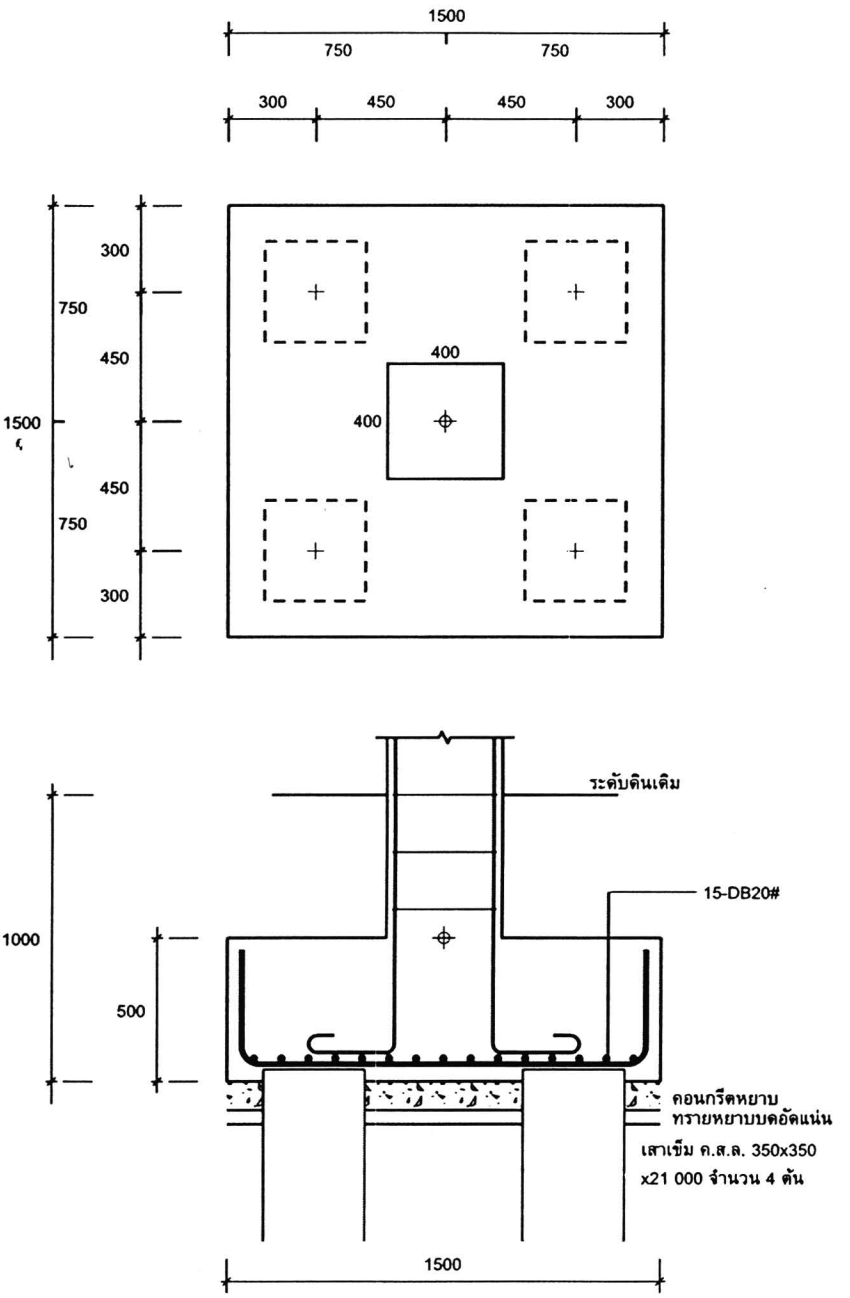


ภาพที่ 35 ภาพแสดงผลจากการใช้งานคลาส BearingFooting

รหัสคำสั่งในการใช้งานคลาส BearingFooting

```
F1 = BearingFooting(100,80 , B=1500. , L=1500. , t = 300.,
                    b = 250. , h=250. , Elevation = 1000. ,
                    Rebar_Main ='5-DB16' ,  scale=25.)
```

โมดูล Footing4Piles



ภาพที่ 36 ภาพแสดงผลจากการใช้งานคลาส Footing4Piles

รหัสคำสั่งในการใช้งานคลาส Footing4Piles

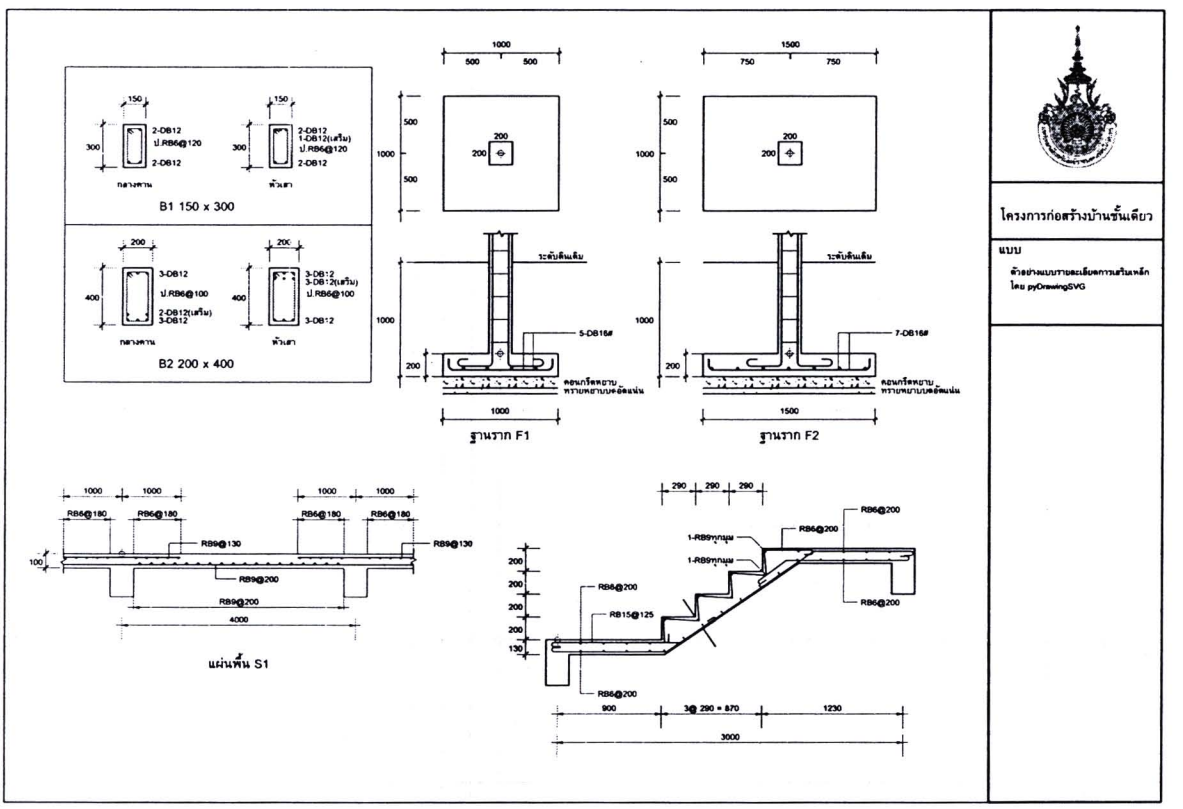
```
F1top = Footing4Piles(x,y , L=L , B=B , t=t , c1=c1 , c2=c2 , h1=h1 )
F1side = Footing4Piles(x,y + 90 , L=L , B=B , t=t , c1=c1 , c2=c2 , h1=h1 , view='side')
```

16. การใช้งานแพคเกจ pyDrawingSVG

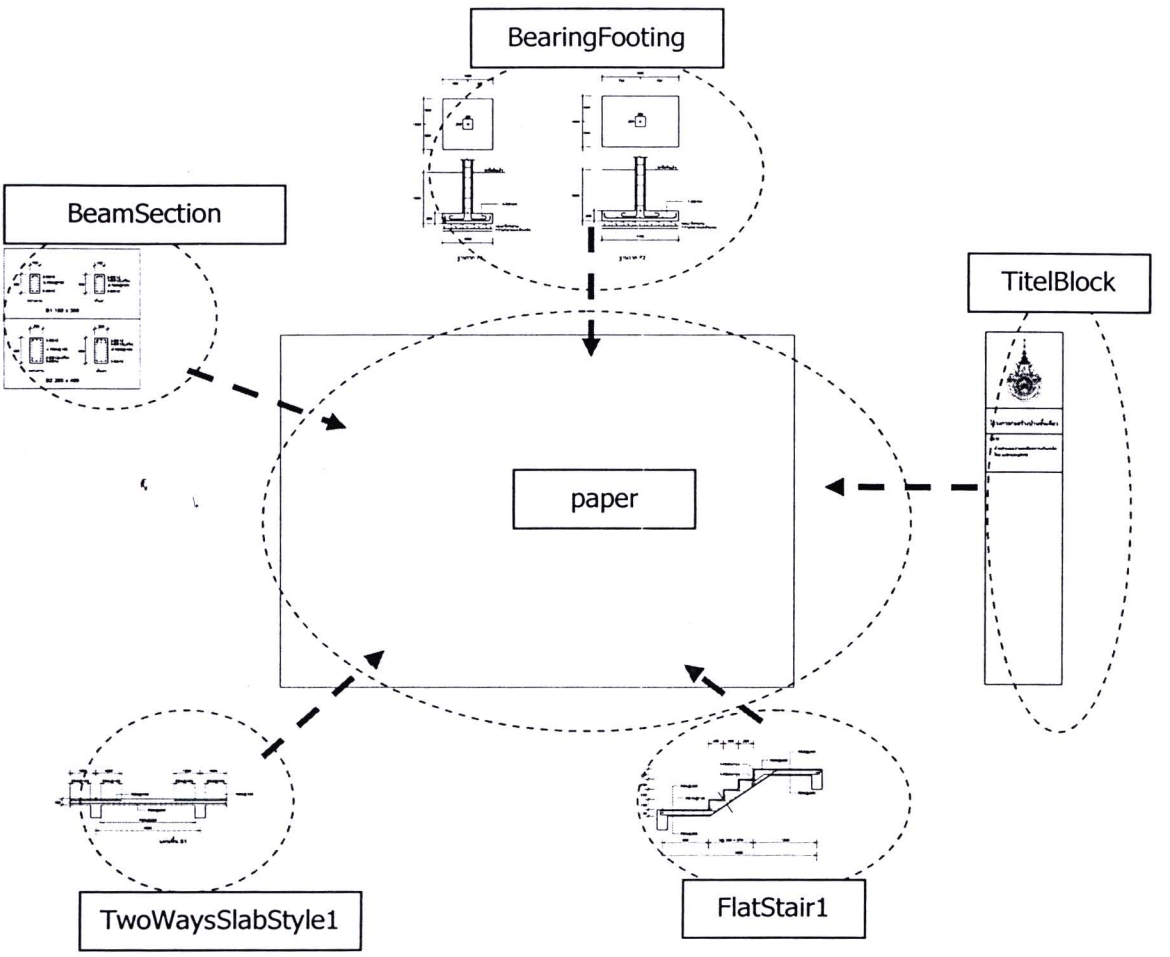
ก่อนจะใช้งานแพคเกจ pyDrawingSVG ผู้ใช้จะต้องติดตั้งโปรแกรมดังต่อไปนี้

1. โปรแกรมภาษา Python ซึ่งดาวน์โหลดได้โดยไม่คิดมูลค่าได้ที่ www.python.org
2. แพคเกจเสริมภาษาไพธอน ชื่อว่า SVGFig ซึ่งดาวน์โหลดได้โดยไม่คิดมูลค่าได้ที่เว็บไซต์ http://code.google.com/p/svgfig/
3. แพคเกจ pyDrawingSVG ซึ่งเป็นผลของงานวิจัยนี้ ส่วนนี้ไม่ต้องติดตั้งให้สำเนาไปไว้ที่โฟลเดอร์ที่ต้องการใช้งานได้เลย เพื่อความสะดวกให้แก้ไขและปรับปรุง รหัสคำสั่งเพิ่มเติม

เมื่อทดสอบการสร้างแบบก่อสร้างบนกระดาษขนาด A3 แนวนอน โดยการใช้งานแพคเกจ pyDrawingSVG จะได้ไฟล์ SVG ดังแสดง



ภาพที่ 37 แสดงภาพผลลัพธ์ที่ได้จากการทดลองใช้งานแพคเกจ pyDrawingSVG



ภาพที่ 38 แสดงแผนภาพขององค์ประกอบของผลลัพธ์เกิดจากคลาสย่อย ๆ นำมาประกอบกัน

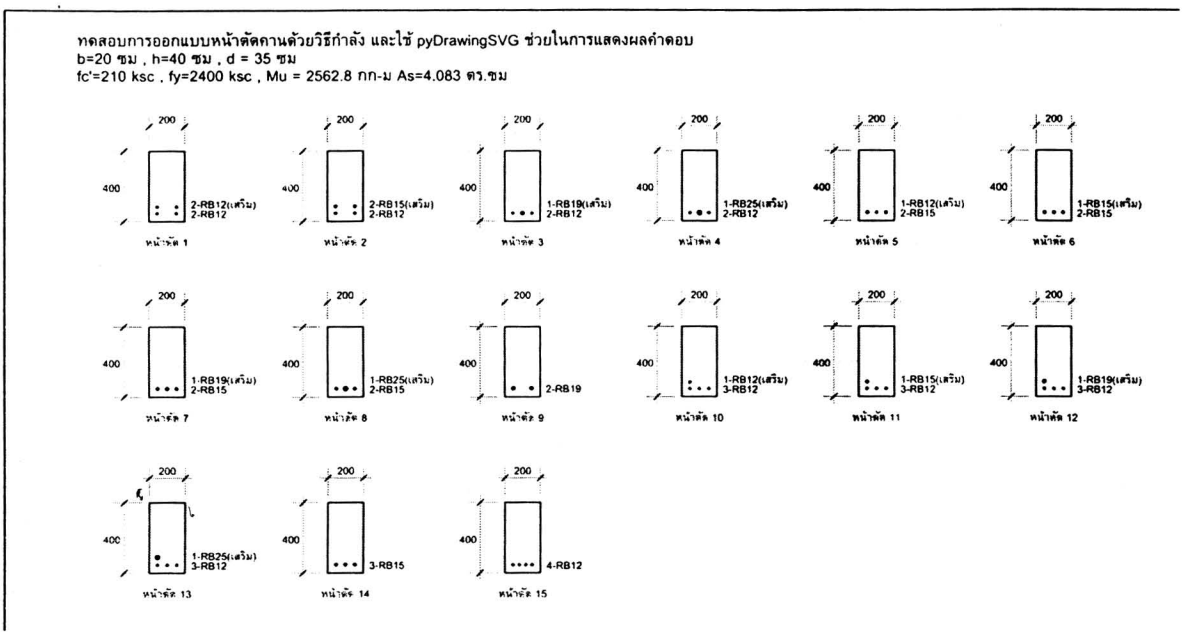
โดยรหัสคำสั่งเรียกใช้งาน ปรากฏตามภาพประกอบข้างบนให้เรียกใช้ตามคำสั่งดังแสดงต่อไปนี้

```
# user script

import sys , os
sys.path.append( os.path.join( os.getcwd(),'D:\CHT_TSB\Research\FirstStep\State\State2' ) )
sys.path.append( os.path.join( os.getcwd(),'D:\CHT_TSB\Research\FirstStep\State\State2\pyDrawingSVG'
) )

from pyDrawingSVG.Paper.paper import paper
from pyDrawingSVG.Beams.Beams import *
from pyDrawingSVG.Footings.BearingFooting import BearingFooting
from pyDrawingSVG.Slabs.TwoWays.Style1 import TwoWayStype1SVG
from pyDrawingSVG.Stairs.FlatStair1 import FlatStair1

print 'test'

font_family= 'Browallia New' ; font_size=6.5
#
Draw01 = paper(size='A3' , orientation='landscape')
B1 =BeamCrossSection(50 , 50 ,  b=150 , h = 300 ,
        TopRebar='2-DB12', BottomRebar='2-DB12' ,
        stirrup=u'RB6@120' , covering=25.,
```

```
        name=u'กลางคาน')
b1 = BeamCrossSection(100, 50 , b=150 , h = 300 ,
        TopRebar='2-DB12+1-DB12', BottomRebar='2-DB12' ,
        stirrup=u'ป.RB6@120' , offset_y_stirrup=2 , covering=25.,
        name=u'หัวเสา')
text1 = SVG('text' , u'B1 150 x 300' , x=75,y=80 ,
        font_family='%s'%font_family ,font_size='%g'%font_size ,
        stroke="none", fill='black' ,text_anchor='middle')
border1=SVG('rect' , x='30' , y='30' , width='105' , height='55' ,stroke_width='0.25' )

B3 =BeamCrossSection(50 , 100 ,  b=200 , h = 400 ,
        TopRebar='3-DB12', BottomRebar='3-DB12+2-DB12' ,
        stirrup=u'ป.RB6@100' , covering=25.,
        name=u'กลางคาน')
b3 =BeamCrossSection(100 , 100 ,  b=200 , h = 400 ,
        TopRebar='3-DB12+3-DB12', BottomRebar='3-DB12' ,
        stirrup=u'ป.RB6@100' , covering=25.,
        name=u'หัวเสา')
text2 = SVG('text' , u'B2 200 x 400' , x=75,y=135 ,
        font_family='%s'%font_family ,font_size='%g'%font_size ,
        stroke="none", fill='black' ,text_anchor='middle')
border2=SVG('rect' , x='30' , y='85' , width='105' , height='55' ,stroke_width='0.25' )

y2 = 60
F1 =BearingFooting(180 , y2 , B=1000,L=1000,t=200,
                   b=200. , h=200. ,Elevation=1000. ,
                   Rebar_Main='5-DB16' , Rebar_Minor='5-DB16' ,
                   scale=25)

text3=SVG('text' , u'ฐานราก F1' , x=180 , y=F1.y + 100. ,
        font_family='%s'%font_family ,font_size='%g'%font_size ,
        stroke="none", fill='black' ,text_anchor='middle')

F2 =BearingFooting(280 , y2 , B=1500,L=1000,t=200,
                   b=200. , h=200. ,Elevation=1000. ,
                   Rebar_Main='7-DB16' , Rebar_Minor='7-DB16' ,
                   scale=25)
text4=SVG('text' , u'ฐานราก F2' , x=280 , y=F1.y + 100. ,
        font_family='%s'%font_family ,font_size='%g'%font_size ,
        stroke="none", fill='black' ,text_anchor='middle')

s1 = TwoWayStype1SVG(50, 200 ,
                  outline=dict(L=4000. , L1=1000 ,  L2=1000) ,
                  Top_Left_Rebar =dict(main='RB9@130',L1=1000 , L2 = 1000) ,
                  Top_Right_Rebar =dict(main='RB9@130',L1=1000 , L2 = 1000) ,
                  Bottom_Rebar=dict(main='RB9@200'  , second='RB9@200' , direction='short' )


text5=SVG('text' , u'แผ่นพื้น S1' , x=50 +4000/50/2 , y=240 ,
        font_family='%s'%font_family ,font_size='%g'%font_size ,
        stroke="none", fill='black' ,text_anchor='middle')

stair1 = FlatStair1(200,230 , n=4 , L=3000. , t = 130. , L1=900. , scale=25.,
                 b1=200.,h1=400 , b2=200., h2=400 )

Draw01.append(B1.SVG())
Draw01.append(b1.SVG())
Draw01.append( text1 )
Draw01.append( border1 )

Draw01.append(B3.SVG())
Draw01.append(b3.SVG())
Draw01.append( text2 )
```

```
Draw01.append( border2 )

Draw01.append(F1.TopViewSVG())
F1.y=F1.y+70 ; Draw01.append(F1.SideViewSVG())
Draw01.append( text3 )

Draw01.append(F2.TopViewSVG())
F2.y=F2.y+70 ; Draw01.append(F2.SideViewSVG())
Draw01.append( text4 )

Draw01.append( s1 )
Draw01.append( text5 )

Draw01.append( stair1 )

#Draw01.append(  BeamCrossSection(100 , 50 , b=250).SVG()  )


Draw01.append( TitleBlockA3Landscape(ProjectName=u'โครงการก่อสร้างบ้านชั้นเดียว' ,
                    DrawingName = [u'ตัวอย่างแบบรายละเอียดการเสริมเหล็ก' , u'โดย pyDrawingSVG']) )
#
pathname = os.path.dirname(sys.argv[0])
fileName = pathname + "/" + "Drawing-S01.svg"
Draw01.save(fileName)
```

ทดสอบการสร้างระบบช่วยในการตัดสินใจในการออกแบบ (Decision Support System) ดังนั้นเมื่อมีการแสดงผลเป็นรูปภาพก็จะทำให้เราสามารถตัดสินใจในการออกแบบได้ดีขึ้น เมื่อพัฒนาต่อเนื่องไปจะได้เป็นระบบผู้เชี่ยวชาญ(Expert Systems)ในการออกแบบโครงสร้างคอนกรีตเสริมเหล็กได้ ดังนั้นต่อไป เมื่อพัฒนาเป็นระบบฐานข้อมูลที่สามารถแสดงหน้าตัดคานและการเสริมเหล็กโดยที่ควบคุมปริมาณเหล็กเสริมให้มีค่าอัตราส่วนการเสริมเหล็กในหน้าตัดระหว่างต่ำสุด และสูงสุด ($\rho_{min} < \rho < \rho_{max}$) แล้วคำนวณหาค่ากำลังต้านทานประลัย เราก็จะได้ระบบฐานความรู้(Knowledge Base) สำหรับงานออกแบบ หรือสำหรับผู้ที่ไม่ชำนาญการเขียนโปรแกรม ก็อาจจะนำแพคเกจ pyDrawingSVG ไปสร้างเป็นไฟล์ SVG แล้วให้ดาวน์โหลดทางอินเทอร์เน็ตได้



ภาพที่ 39 ผลลัพธ์จากการนำแพคเกจ pyDrawingSVG ไปร่วมใช้งานกับการออกแบบหน้าตัดคาน

รหัสคำสั่งที่ทดลองออกแบบคาน แล้วนำความสามารถการวาดผลลัพธ์เป็น SVG มาแสดงเพื่อช่วยตัดสินใจในการออกแบบ จะเห็นได้ว่าแพคเกจ pyDrawingSVG มีประโยชน์มากในการช่วยงาน

```
'''
 SDM : Singly Section
       MKS unit
 Non-OOPs
 python
 by Chakkree Tiyawongsuwan
 2010 Jan 31
 2010 Aug 22
 ลองออกแบบให้ได้เหล็กเสริม แล้วนำไปวาดหน้าตัด
'''
from math import sqrt , pi

# สำหรับทดลองใช้งาน ไลบรารีต้องติดตั้งใน sitepackage
import sys , os
sys.path.append( os.path.join( os.getcwd(), 'D:\CHT_TSB\Research\FirstStep\State\State2' ) )
sys.path.append( os.path.join( os.getcwd(),
'D:\CHT_TSB\Research\FirstStep\State\State2\pyDrawingSVG' ) )

from pyDrawingSVG.Beams.Beams import *
from pyDrawingSVG.Paper.paper import *

def beta1(fcu):
    if fcu<=280:
        val = 0.85
    elif fcu>=560:
        val = 0.65
    else:
        val = 0.85 - (fcu-280)*0.05/70.
    return val
```

```
def p_design(Ru , fcu ,fy):
    term1 = 1- (2 * Ru /(0.85*fcu) )
    if term1 < 0 :
        print "หน้าตัดเล็กเกินไป \n ขยายหน้าตัด หรือออกแบบเป็น Doubly"
    return  0.85*(fcu/fy) * (1. - sqrt(term1) )

def DesignSingly(Mu , b ,d , fcu ,fy):
    Ru = Mu * 100. / (0.90 * b*d**2)
    p = p_design(Ru , fcu ,fy)
    As = p * b* d
    print "Mu = %.1f กก-ม"%Mu
    print "b = %g ซม , fy = %g ซม " %(b,d)
    print "fc' = %g ksc , fy = %g ksc " %(fcu,fy)
    print "Ru = %.2f ksc "%Ru
    print "p = %.4f  "%p
    if p<(14./fy):
        As = (14./fy) * b * d
    print "p = %.4f "%p
    print "As = %.3f"%As

    return As

RebarList_default = [12,15,19,25]
def As2Rebar(As , RebarList=RebarList_default , typeRebar='RB'):
    Rebars = []

    numMain = {2,3,4}
    for numM in numMain:
        for diaM in RebarList:
            AsMain = pi*(diaM/20.)**2 *numM
            rebarForm = '%d-'+typeRebar +'%d'
            strRebar = rebarForm%(numM , diaM)
            if AsMain<As:
                for diaAdd in RebarList:
                    strAdd = ''
                    strRebar2 = ''
                    numAdd = GetNumRebar(As-AsMain , diaAdd)
                    if type(numAdd)==int:
                        strAdd = ('+'+rebarForm)%(numAdd , diaAdd)
                        strRebar2 = strRebar + strAdd
                    if len(strAdd)>0:
                        Rebars.append(strRebar2)
            else:
                Rebars.append(strRebar)
                break
            #print '%d-RB%d  As = %g'%(numM , diaM , AsMain)
    return Rebars

def GetNumRebar(As , dia):
    Area = pi*(dia/20.)**2
    for num in range(1,5):
        if As< num*Area:
            return num
    return "Error : จำนวนเหล็กเสริม มากกว่า 5เส้น"


if __name__=="__main__":
    print "Test for Beta "
    print "beta = %.3f " % beta1(150)
    print "beta = %.3f " % beta1(300)
    print "beta = %.3f " % beta1(600)

    print "\n\nTest for Design Singly style 2"
```

```
# ย.วินิล,หน้า98
print " As = %.3f sq.cm.  " % \
    DesignSingly( 23800.0 , 25. , 41.25 , 200. ,3000.)

print "\n\nทดสอบ SVG"

b = 20.
h = 40.
d= h - 5.
fcu=210. ;    fy = 2400. ; typeRebar = 'RB' ; RebarList=[12,15,19,25]
#fcu=210. ; fy = 3000. ; typeRebar = 'DB' ; RebarList=[12,16,20,25,32]

Mu = 2562.8
As =DesignSingly( Mu , b , d , fcu , fy)
myRebarList = As2Rebar(As, RebarList , typeRebar)

myDrawing =  paper(size="A3" , orientation="landscape" , showGrid=True)
x1 = 50. ; y1 = 50.
dx = 50. ; dy = 50.
maxCol = 5
curCol = 0 ; curRow = 0
curNumber =1
for i in myRebarList:
   print i
   b1 = BeamCrossSection(x1  + curCol*dx , y1 + curRow*dy , b=b*10. , h=h*10 ,
              TopRebar='' , BottomRebar=i ,name = str(curNumber) )
   curCol+=1 ; curNumber+=1
   if curCol>maxCol:
      curCol = 0 ; curRow += 1
   myDrawing.append( b1.SVG() )

strDescrip1 = u" ทดสอบการออกแบบหน้าตัดคานด้วยวิธีกำลัง "
strDescrip1 += u" และใช้ pyDrawingSVG ช่วยในการแสดงผลคำตอบ "
strDescrip2 = u" b=%g ซม , h=%g ซม , d = %g ซม \n"%(b,h,d)
strDescrip3 = u" fc'=%g ksc , fy=%g ksc , Mu = %g กก-ม "%(fcu,fy,Mu)
strDescrip3 += u"   As=%.3f ตร.ซม \n"%(As)

myDrawing.append( SVG('text' , strDescrip1 ,
                      x='30' , y='20' , \
                      font_size='6.5', stroke="none", fill="black", text_anchor="start")    )
myDrawing.append( SVG('text' , strDescrip2 ,
                      x='30' , y='25' , \
                      font_size='6.5', stroke="none", fill="black", text_anchor="start")    )
myDrawing.append( SVG('text' , strDescrip3 ,
                      x='30' , y='30' , \
                      font_size='6.5', stroke="none", fill="black", text_anchor="start")    )

pathname = os.path.dirname(sys.argv[0])
fileName = pathname + "\\" + "testBeamDesign2SVG-%gx%gfc'%gfy%gMu%.0f.svg" % (b,h,fcu,fy,Mu)
print fileName
myDrawing.save(fileName)
```