根据桓博士上次的讲解，我先简单的阐述下我理解到的东西，理解太浅显、简单，请领导指正。

提出这个主要是为了形成一套通用的解决方案，依据用户给定的输入、计算规则等，设计一个满足工艺要求的工艺方案，供用户依据此方案来作业，提供资源利用率。

我以前做过一个数据处理方面的功能，和这个看起来有点相似，不同的是：它有一套标准的输入、输出，只是中间的数据处理算法上有差异；而这个看起来，只有输入是一样的，处理和结果都是因规则而异。

输入：给定的模型，长、宽、厚度等参数；

处理：因模型的形状、额外的参数、计算规则等会有截然不同的处理方案；

结果：把处理后的结果、中间结果等都返回给用户

我的想法是大部分来源于桓博士的讲解，把所有的变量、规则等都写入一个脚本，最后调用这个脚本，获取脚本的输出结果。

里面涉及到的技术其实无外乎：

* + 1. XML文件的读写：xml的定义可如下图所示，解析相对比较简单



* + 1. 规则的解析：就是把用户定义的规则，转换成脚本可执行的语句
    2. C++和脚本的交互：检查写入的脚本是否可执行、能执行是否可出结果、能出结果是否正确、结果正确之后是否能获取到该结果等

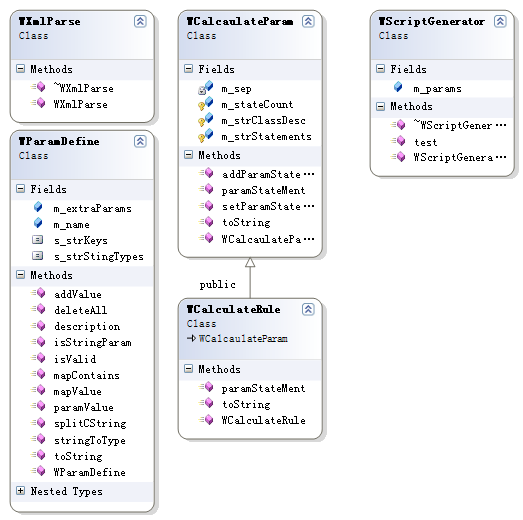
难点：

1. 规则的定义和解析：用户定义的规则，应该如何正确的转换成脚本可解释的语句
2. 与脚本的交互，有两项：

* 检查脚本正确性：运行前检查写入的内容是否符合脚本语法规范；运行时检查，脚本执行是否出现异常；
* 结果的获取：返回来可能是一堆东西，怎样解释及展现给用户；

基于此，我采用python脚本和C++写了一个简单的测试例子，写入、执行脚本及结果的获取都是没问题的，但是很多细节的东西变化太多，尚不能下定论；因为对桓博士说的VB不了解，暂时先用我熟悉的python做测试，相信VB应该和python的执行是相似的。

再来就是介绍我写的这个测试例子，类图如下：



class WxmlParse; //解析xml的类 例子中未实现

class WparamDefine; //单个参数的定义类，预定义的那些输入参数

class WcalcaulateParam;//参与计算的额外参数，主要是赋值语句，形如：aaa=100

class WCalculateRule : public WcalcaulateParam;//计算规则，继承WcalcaulateParam，主要是继承它的成员和添加规则语句的方法，这些语句都保存在CstringArray里面，可以调用paramStateMent获取到转换之后的语句，写入脚本；

class WscriptGenerator; //这个是对外部的类，当用户去load一个xml文件时，会调用WxmlParse把参数、规则等读取出来，放在对应的类里；用户点击执行之后会写入脚本，执行该脚本；未全部实现，只是简单的写了个test函数。

具体实现内容见，附件中的工程。

所有客户定义的东西相信都不是无的放矢的，大部分公式、计算方法都是多年的积累；我觉得让客户通过UI去定义这些东西，不太现实；一个完整的工序太复杂了，桓博士给你这个例子，首先参数、规则，还有各种计算，让客户去定义都是不好的；我个人认为一套完整的东西，都需要反复测试才能拿给客户去使用，而客户的工艺方案流程这么复杂，对他们来讲去测试是不好的。

我有一个想法：

我们可以给他们现有的工艺方案都整理出来，然后在定义一套通用的模板，然后他们每个方案都相当于一个模块，各个模块的处理方法不一样，我们可以把现有的工艺方案都实现了，以后用户需要添加或修改已有的工艺方案时，可以自己添加进来；

这种方法就不需要用到脚本，直接用C/C++语言就可以处理的很好了；

我通过看客户给的那个工艺的例子(EXCEL记录的产品归类1)：

#参数定义 参数以及各自的成品上下偏差

#L长 W宽 H厚度 R圆角

#长

L=5.54

L\_UPPER=0.01

L\_LOWER=-0.03

#宽

W=4.17

W\_UPPER=0.01

W\_LOWER=-0.03

#厚度

H=0.75

H\_UPPER=0.02

H\_LOWER=-0.03

#圆角半径

RADIUS=1.2

RADIUS\_UPPER=0.3

RADIUS\_LOWER=0

#表面积

Surface=56.75125362

#体积

Volume=16.39927007

#重量

Weight=0.122994525

#数量 pcs

Count=2000

#总面积 dm^2

GrossArea=11.35025072

#镀层种类

CoatingType='环保彩锌'

#镀层厚度 5-10 um

CoatingThick=6

#烟雾要求

SmogRequire='48h, 7级'

#结合力

BindingForce='有要求'

#计算的上下偏差

L\_CAL\_UPPER=round((L\_UPPER+L\_LOWER)/2 + 0.04, 2)

L\_CAL\_LOWER=L\_CAL\_UPPER-0.03

W\_CAL\_UPPER=round((W\_UPPER+W\_LOWER)/2 + 0.04, 2)

W\_CAL\_LOWER=W\_LOWER-0.03

#厚度的计算

if (H\_UPPER-H\_LOWER < 0.08):

H\_CAL\_UPPER=round(H\_UPPER+0.02, 2)

else:

H\_CAL\_UPPER=round((H\_UPPER+H\_LOWER)/2+0.06, 2)

if (H\_CAL\_UPPER-0.04 > H\_LOWER + 0.06):

H\_CAL\_LOWER=round(H\_CAL\_UPPER-0.04, 2)

elif (H\_CAL\_UPPER - H\_LOWER >= 0.07):

H\_CAL\_LOWER=round(H\_CAL\_UPPER-0.03, 2)

else:

H\_CAL\_LOWER=H\_LOWER+0.03

#圆角

RADIUS\_CAL\_UPPER=RADIUS\_UPPER-0.05

RADIUS\_CAL\_LOWER=RADIUS\_LOWER+0.05

#对角线

DIAGONAL\_UPPER = round(math.sqrt(math.pow((L+(L\_CAL\_UPPER+L\_CAL\_LOWER)/2-2 \* (RADIUS+RADIUS\_CAL\_UPPER)), 2) + \

math.pow((W+(W\_CAL\_UPPER+W\_CAL\_LOWER)/2-2\*(RADIUS+RADIUS\_CAL\_UPPER)),2)) + \

2\*(RADIUS+RADIUS\_CAL\_UPPER), 2)

DIAGONAL\_LOWER = round(math.sqrt(math.pow((L+(L\_CAL\_UPPER+L\_CAL\_LOWER)/2-2 \* (RADIUS+RADIUS\_CAL\_LOWER)), 2) + \

math.pow((W+(W\_CAL\_UPPER+W\_CAL\_LOWER)/2-2\*(RADIUS+RADIUS\_CAL\_LOWER)),2)) + \

2\*(RADIUS+RADIUS\_CAL\_LOWER), 2)