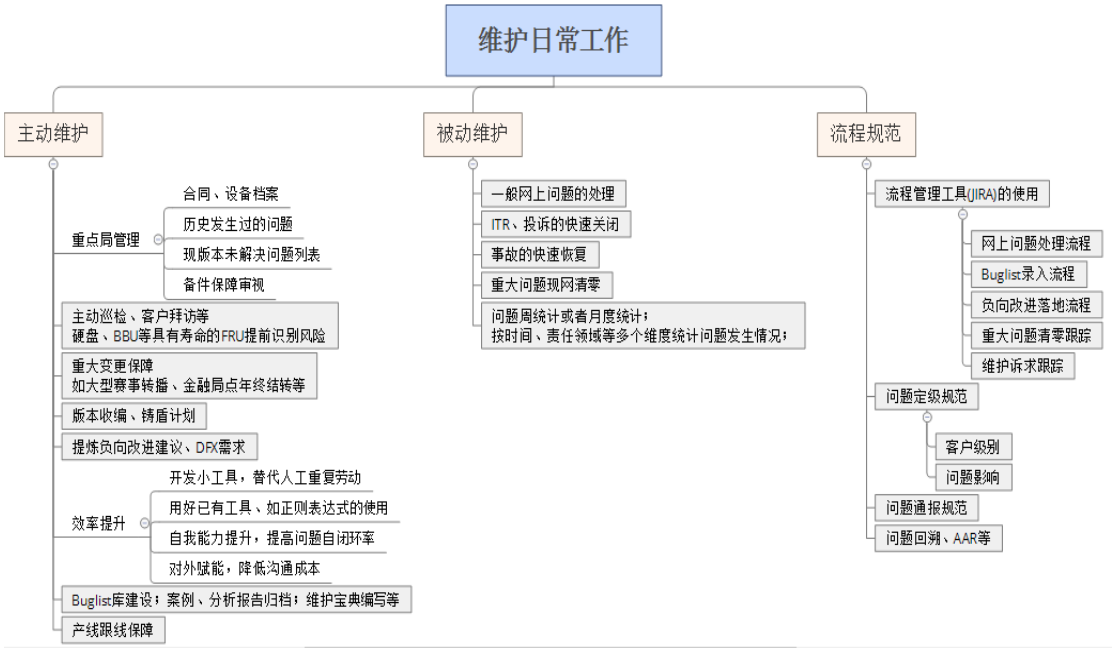
一、上市保障日常工作：

- 被动维护：如上市保障问题处理，事故的快速恢复，ITR、投诉的快速关闭等
- 主动维护：如针对金融、国家电网等客户设立重点局设备档案；阿里等重点客户在双十一等关键时间点的保障；避免处理问题过程中发现的低级错误重犯；高效地跟踪改进点落地；快速将典型问题的处理技能传递给轮转到上市保障组的同事等。

上市保障工作事项繁杂，每件事项具有多个环节，每个环节需要不同的角色来处理，怎样将事项、环节、员工角色系统地组织起来？



二、上市保障痛点：

- 被问题攻关搞得焦头烂额，彻夜不眠，最后发现是一个低级问题
- 被客户骂的抬不起头来，因为产品质量羞愧难当。
- 解决这些问题的过程中有很多闪亮的点子，一些可供后续设计、开发时参考进而避免重蹈覆辙的规范，可惜没有积累下来，新一代产品开发时又在问题的泥潭中苦苦挣扎。
  怎样避免初级问题重犯？上市保障的典型问题如何积累下来，怎样对上市保障组的新组员进行快速赋能？

三、解决方案：

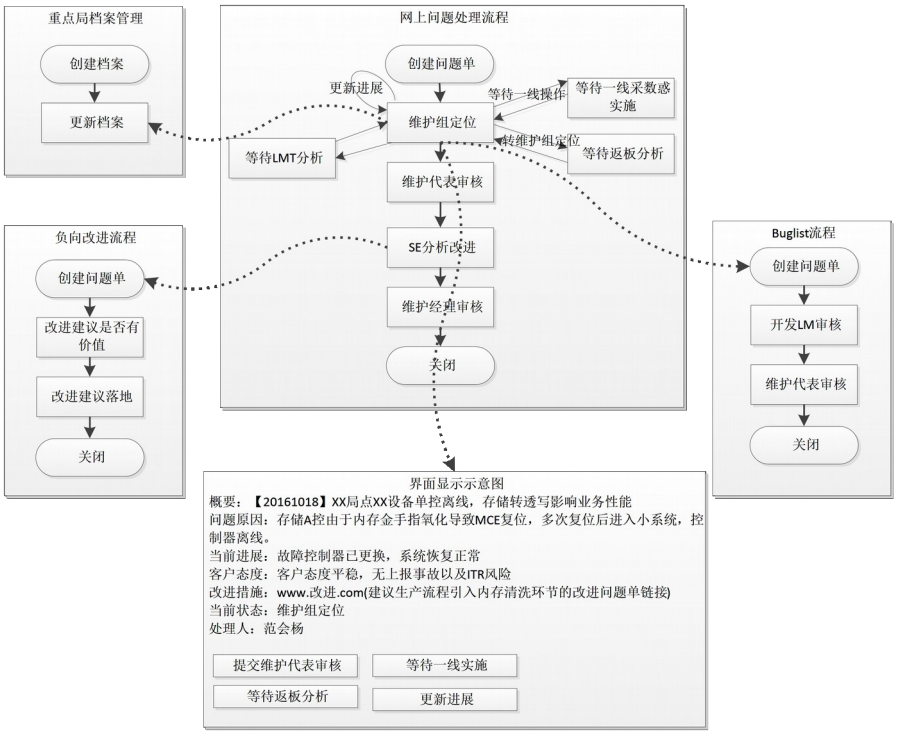
- 梳理日常工作，将例行工作划分为多个业务流程，如问题处理流程，改进流程等；
- 每个业务划分多个环节，并为每个环节指定处理人；
- 规划好以后由专人利用业务流程管理(BPM)工具进行实施；
- 工具实施后，各角色作为用户登录系统，处理分配给自己的问题单即可；

TaskSteper 是基于 django 框架实现的建议的 BPM 工具，可以协助各任务角色高效地分工协作。

四、工具使用示例：

1.流程梳理

将上市保障组的日常工作分为网上问题处理、重点局档案管理、负向改进、典型 Buglist 管理等几个流程，如下所示：



2.事务跟踪在 TaskSteper 系统中的实现

以网上处理流程为例，步骤如下：

2.1)定义字段 Model

此处定义为 IssueTraceModel，一个 Model 对应数据库的一张表，Model 的一个 field，如 summary、detail 等对应表中的一列。

```
class IssueTrackModel(models.Model):
    summary = models.CharField(max_length=200, verbose_name="概要", default="【20170101】XX局点XX设备发生XX问题")
    detail = models.CharField(max_length=5000, verbose_name="详细描述")
    current_process = models.CharField(max_length=5000, verbose_name="处理进展",default="##由维护处理人填写##")
    issue_level = models.CharField(max_length=200, verbose_name="问题级别", choices= CHOICE_ISSUE_LEVEL)
    customer_type = models.CharField(max_length=200,  verbose_name="客户类别", choices=CHOICE_CUSTOMER_TYPE)
    class_1 = models.CharField(max_length=200, verbose_name="问题大类", choices= CHOICE_ISSUE_CLASS_1)
    class_2 = models.CharField(max_length=200, verbose_name="问题小类", choices= CHOICE_ISSUE_CLASS_2)
    class_3 = models.CharField(max_length=200, verbose_name="责任领域", choices= RESPONSE_FIELD_CHOICES)
    buglist_url = models.CharField(max_length=200, default="#Buglist单链接(若有)#", verbose_name="Buglist链接(URL)")
    improvement_url = models.CharField(max_length=200, default="#改进单链接(若有)#", verbose_name="改进项链接(URL)")
    issue_processor = models.CharField(max_length=200, default="#提交人填写#", verbose_name="维护处理人")
    issue_checkor = models.CharField(max_length=200, default="#维护处理人指定#", verbose_name="维护代表")
    issue_se = models.CharField(max_length=200, default="#维护代表指定#", verbose_name="改进提取SE")
    assigned_to    = models.CharField(max_length=200, default="", verbose_name="当前处理人(只读)")
    created_by = models.CharField(max_length=200, default="", verbose_name="创建人(只读)")
    curent_state = models.CharField(max_length=200, verbose_name="当前状态(只读)")
```

Field 的属性说明如下：

max_length	字段的最大长度
verbose_name	字段在界面的显示名称，见下图
default	字段默认值
choices	下拉型字段的选择列表，class_1 字

```
CHOICE_ISSUE_CLASS_1 = (
    ('软件','软件'),
    ('硬件','硬件'),
    ('结构','结构'),
    ('误操作','误操作'),
    ('配置','配置'),
)
```

	段的 choices 如右图所示，需在 IssueTraceModel 之前定义。

2.2)定义字段 Form

Form 对应于界面显示一个表单，其中主要定义了字段在显示时的一些属性，如显示的大小，是否只读等。此处需要将 form 关联的 model 设置成步骤 1 中的 model。

```
class IssueTrackForm(ModelForm):
    class Meta:
        model = IssueTrackModel
        fields = '__all__'
        widgets = {
            'summary' : forms.TextInput(attrs={'size':79}),
            'current_process' : forms.Textarea(attrs={'cols': 80, 'rows': 5}),
            #'viewer_advice' : forms.Textarea(attrs={'cols': 80, 'rows': 5}),
            'detail' : forms.Textarea(attrs={'cols': 80, 'rows': 5}),
            'buglist_url' : forms.TextInput(attrs={'size':79}),
            'improvement_url' : forms.TextInput(attrs={'size':79}),
            'assigned_to': forms.TextInput(attrs={'readonly': True,'size':20}),
            'created_by': forms.TextInput(attrs={'readonly': True}),
            'curent_state': forms.TextInput(attrs={'readonly': True}),
        }
```

2.3)将上述两个步骤定义的 Model 和 Form 添加到 FormAndModelDict 中。

TaskSteper 支持创建多个项目，每个项目有独立的流程和字段，FormAndModelDict 中’ issue_track’为项目名,’网上问题处理’为网页上显示的项目中文名。PrjModelClass 和 PrjFormClass 分别指定每个项目使用的 Model 和 Form 类。

```
FormAndModelDict = {
    'improvement':{'PrjNameZh':'改进建议',    'PrjModelClass':NameModel,      'PrjFormClass':NameForm},
    'device_card':{'PrjNameZh':'设备管理',    'PrjModelClass':DeviceCardModel,'PrjFormClass':DeviceCardForm},
    'issue_track':{'PrjNameZh':'网上问题处理','PrjModelClass':IssueTrackModel,'PrjFormClass':IssueTrackForm}
}
```

定义完成后界面显示效果如下所示：

2.4）定义状态转换表 FSM_TRANS_ISSUE_TRACK

```
#上市保障问题跟踪
issue_track_action_1 = {'assign_to':'维护代表'}
issue_track_action_2 = {'assign_to':'维护处理人'}
issue_track_action_3 = {'assign_to':'改进提取SE'}
FSM_TRANS_ISSUE_TRACK = [
    {'source': '维护人员处理',      'trigger': '更新进展',      'dest': '维护人员处理',     'trans_condition':{},   'trans_action':{'assign_to':'创建人(只读)'}},
    {'source': '维护人员处理',      'trigger': '提交审核',      'dest': '维护代表审核',     'trans_condition':{},   'trans_action':issue_track_action_1},
    {'source': '维护代表审核',      'trigger': '打回补充信息',  'dest': '维护人员处理',     'trans_condition':{},   'trans_action':issue_track_action_2},
    {'source': '维护代表审核',      'trigger': '转SE分析改进',  'dest': 'SE分析改进',       'trans_condition':{},   'trans_action':issue_track_action_3},
    {'source': 'SE分析改进',        'trigger': '问题关闭',      'dest': '关闭',             'trans_condition':{},   'trans_action':{}}
]
```

表中每一行表示一个状态转换，包括以下元素：

source	源状态
trigger	对应界面上的触发按钮，点击按钮时会将问题单转到目标状态
dest	目标状态
trans_condition	转换条件，只有满足转换条件才会生成相应的按钮
trans_action	转换时的操作，目前支持 assign_to（将问题分配给某个角色）等

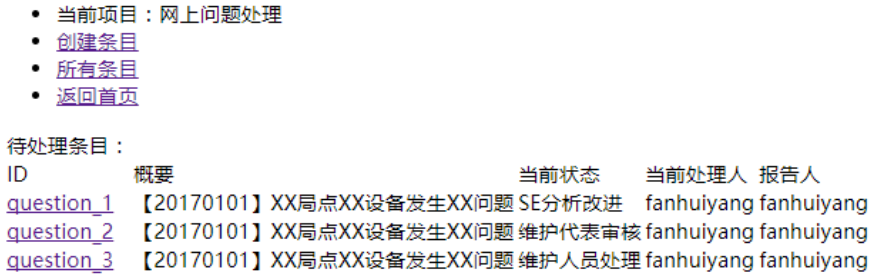
定义完成后将项目名称和对应的状态转换表添加到 TRANS_TABLE_DICT 中：

```
#定义各个项目对应的状态转换表
TRANS_TABLE_DICT = {
    'improvement':FSM_TRANS_TABLE,
    'device_card':FSM_TRANS_TABLE_DEVICECARD,
    'issue_track':FSM_TRANS_ISSUE_TRACK
}
```

3.用户使用 TaskSteper 跟踪问题处理

具体到一个问题的处理，流程如下所示

- 维护人员接收到问题后创建条目，录入问题单库，记录时间、局点、设备型号、问题描述等信息；



当前项目: 网上问题处理

【概要:】 【20170101】XX局点XX设备发生XX问题

【详细描述:】
【问题现象】：xx
【问题影响】：xx
【客户诉求】：xx
【客户态度】：xx

【处理进展:】 ##由维护处理人填写##

【问题级别:】 一般问题

【客户类别:】 金融

【问题大类:】 硬件

【问题小类:】 线缆

【责任领域:】 线缆

【Buglist链接(URL):】 #Buglist单链接(若有)#

【改进项链接(URL):】 #改进单链接(若有)#

【维护处理人:】 #提交人填写#

【维护代表:】 #维护处理人指定#

【改进提取SE:】 #维护代表指定#

【当前处理人(只读):】 fanhuiyang

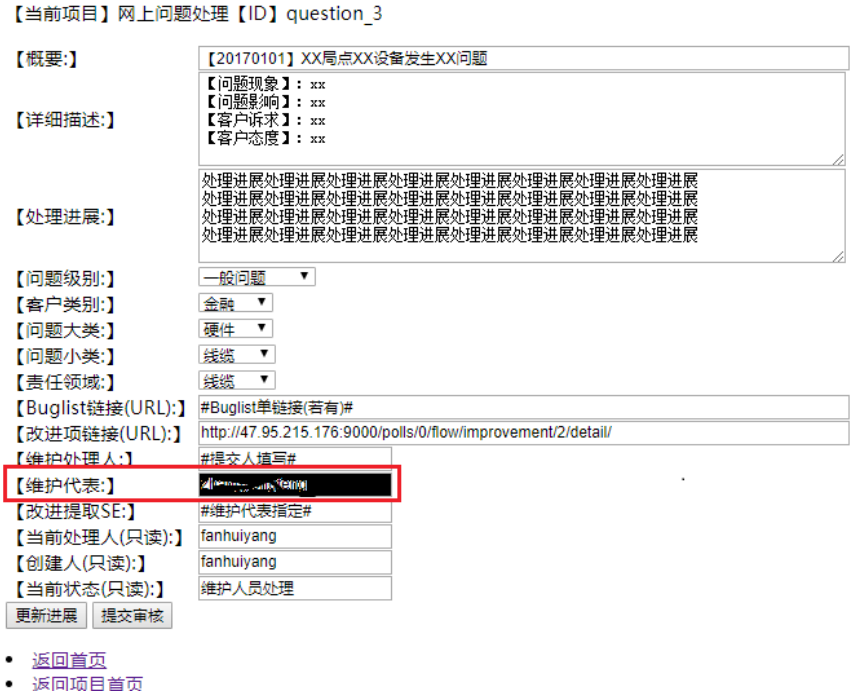
【创建人(只读):】 fanhuiyang

【当前状态(只读):】 维护人员处理

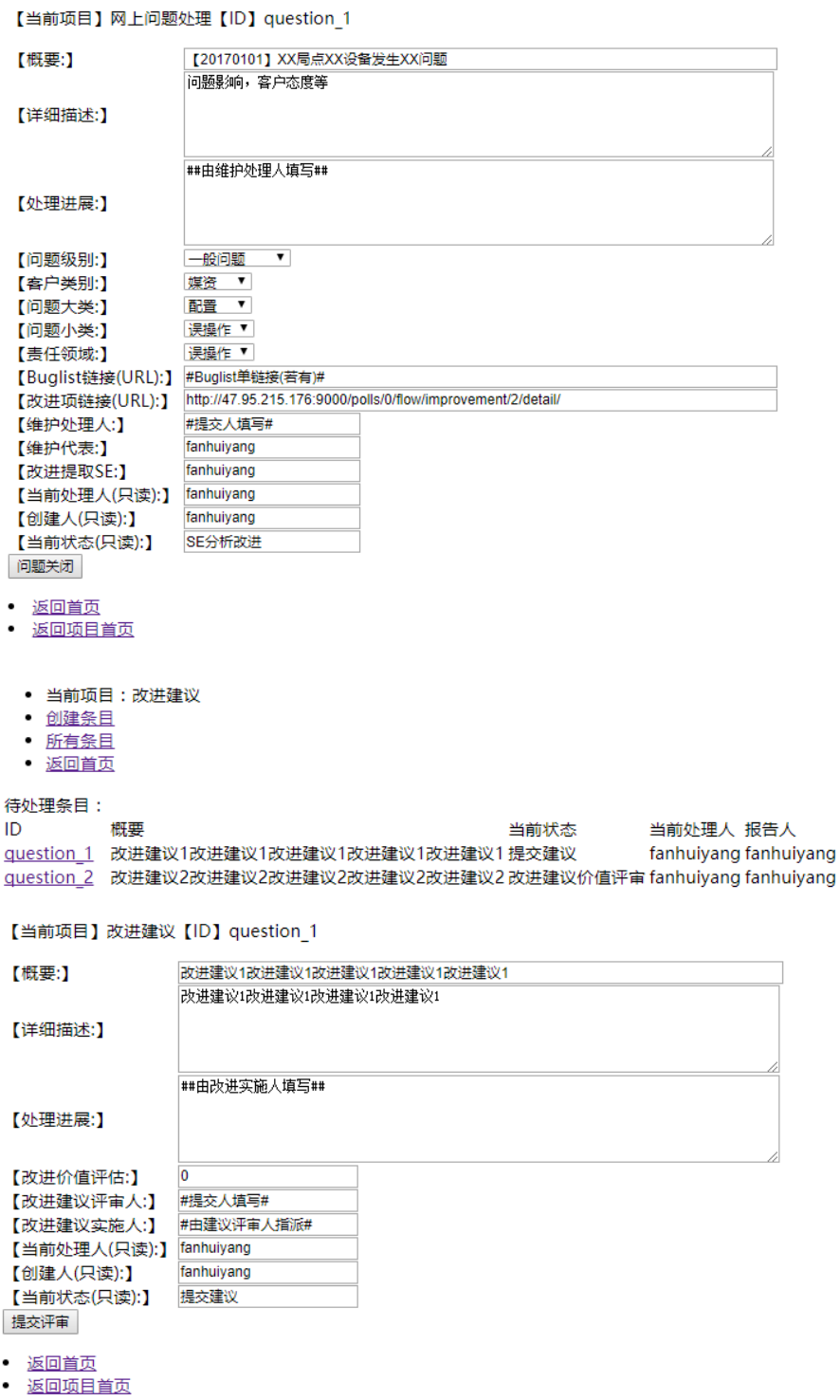
创建

- 首页

- 在“维护人员处理”阶段，问题处理人补充问题信息，更新问题进展；
  如果该问题是多次出现的典型问题，需要在“buglist”管理项目中增加一个问题，并将问题地址填入此处的“Buglist 连接(URL)”字段中。
  处理完成后填入维护代表，点击“提交审核”将问题单提交审核，问题转给下个角色处理。



- SE 分析改进阶段，若从该问题中发现需改进点，则创建一个改进单用于跟踪改进落地，并将改进单的地址填入“改进项链接(URL)”字段。
  从问题处理过程完成了 Buglist 的积累，改进点的提取等，并且在问题处理项目和 Buglist、改进流程之间通过 URL 链接建立联系，可以方便地查看问题的发生情况、问题的分析报告、问题提取的改进点等信息。



五、进一步优化

增加按字段搜索、统计、导出等功能。

例如现在每次发版本需要手动写 release notes，实际上可以增加“release notes”字段，每次发版本时，根据问题单走到解决状态的时间、技术小组等过滤出这一版本解决问题，然后将这些问题的“release notes”字段导出即可。