敏捷实战：你要了解的燃尽图都在这里

在运用敏捷的公司中，我们通常离在几米开外的地方就可以看到团队工作空间中有各种颜色的便笺，白板，各种图表等。在敏捷中我们称这些为信息发射源 Information Radiator。
它可以很好的用于项目的沟通、展示项目的状态。其中有一种弯曲的曲线，叫做：燃尽图，它在我们的项目中扮演着非常重要的角色，可以反映当前冲刺执行情况、进度及风险等。这也是每一位敏捷教练及团队成员都需要能“*读懂*”的图。那么问题来了：

- 何谓燃尽图？
- 如何画燃尽图？
- 常见的燃尽图有典型类型及反映出的问题、应对措施？
- 燃尽图常见问题有哪些及如何处理？
- 有没有实际项目中的实例分析？

以上，正是今天要跟大家分享的内容。

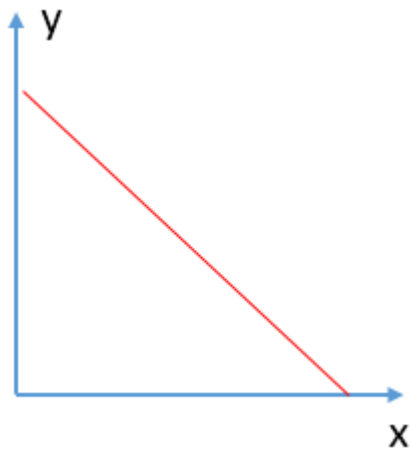
闲话不多说，艾瑞巴蒂请系好安全带，老司机要发车了！

何谓燃尽图

每天都有人搬砖，然后砖头就会越来越少。也就是总工作量每天都会随着时间推移而减少，所以就会形成一个斜向右下延伸的直线（理想状态）。高逼格一点的表示就是会有一个类似这个几何特征的图形：y=kx+b（k<0,b>0）。

我们通常会在在一个XY坐标轴的坐标系表示他们。其中，X轴通常代表时间（天），Y轴代表剩余工作量。我们把这个起名叫：燃尽图。

理想状态下，大家每天都按时搬砖，不打酱油。那么它将会是一条直线，形如下图：



但现实中，我们怎么可能不打酱油！所以就会变成了弯弯曲曲的折现，也就有了实际曲线这么一说。我们通常用实际曲线来标识实际每天完成的工作量。

这是个很好的进度展示的工具，想想看我们之前的项目进度计划，要么是存在PM的电脑里，要么是存在PM的脑海里…PM不在周会上通报现在的进度，鬼知道现在进度如何了。现在我们把这个燃尽图放到团队工作空间中，这样每个人都能看到现在项目的进度状态了。而且在Daily Meeting上也会同步更新燃尽图，这样可以给团队一种紧迫感。信息透明化给团队带来的内部的自我刺激，远比PM的指令式管理要好很多。所以这也是燃尽图的非常重要的作用之一。

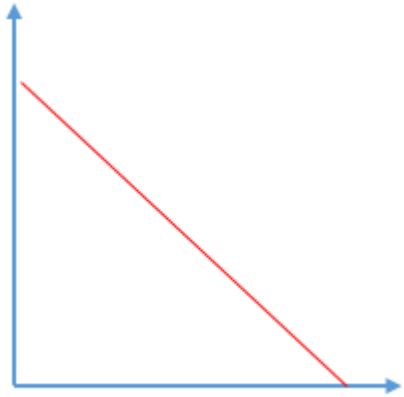
如何画燃尽图

我们在每日的Daily Meeting会议上，每个团队成员都会介绍昨天完成了哪些事情，那这些统计数据就是实际完成的活，用总量-实际完成，就得到了当前时间的剩余工作量了。需要注意的是，Y轴代表的是当前剩余的工作量，做了90%，和做了1%，都叫没做。

常见燃尽图类型有哪些

Kane Mar将燃尽图分为七种情况，个人根据理解及实际情况修改如下几种：

Case1：完美型。特征：一条直线。图形如下：

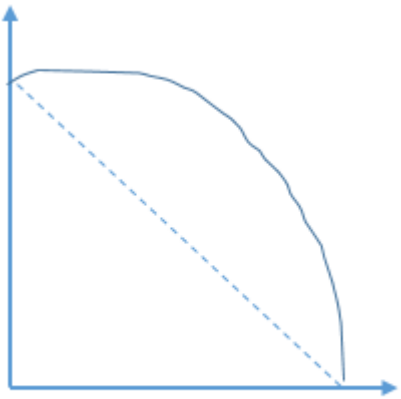


完美

这种情况除非你团队都是神，否则基本上不可能出现这种情况。如果你的团队是这样的图，很可能遇到了个假的团队：本燃尽图纯属虚构，如有完美纯属巧合！

有的Team领导过分关注绩效，看到燃尽图不完美团队就挨批。所以团队就画个完美的图，领导一看，大呼：666，这个sprint大家都很努力，下班我请客去嗨皮！

Case2：啤酒肚。特征：圆弧状，且实际曲线一直在理想直线上方。图形如下：



啤酒肚

出现这种情况的原因：

1. 前期需求估算不准确。

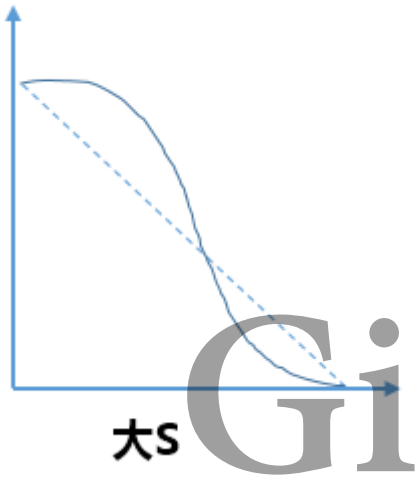
2. 团队遇到障碍，团队成员能力有限。

3. 团队有很强的自我调整能力，可以完成很多额外的工作，不完成目标不回家的愿景。

这种情况要注意，前期发现一直有这个问题的时候，我们处于高风险状态，很可能这个sprint的目标无法实现。在中期SM就应当进行调整，比如：

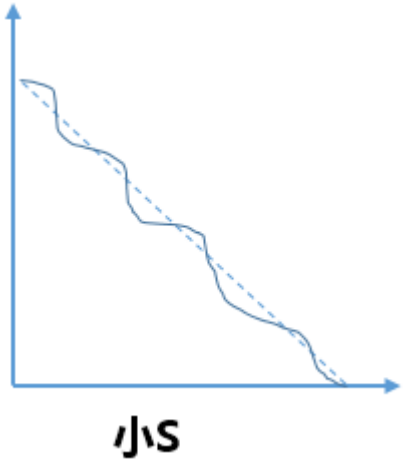
1. 当前需求，有没有其他优化方法可以实现。
2. 某一个人或某几个人在一个问题上卡住了，移除障碍。
3. 能不能跟PO协商，减少当前sprint承诺的内容，如果一定做不完那就跟PO协商，做高优先级的，其他的放到下个sprint。

Case3：大S型。特征：前期啤酒肚，后期快速下降且在直线下方。图形如下：



这种情况说明团队的应变能力强，不断调整速度来完成目标。当然，也有一种可能是大家看到现在风险比较高，然后加班（多么痛的领悟，谁说敏捷没加班的，那得看是谁在做…），不把曲线降下来谁都别回家！！

Case4：小S型。特征：摇摆摇摆…我摇摆摇摆…… 图形如下：



这种情况属于正常情况，团队不断的在修正自己的速率，而且最终完成了目标。如果你的团队实际燃尽图与此类似，恭喜你。

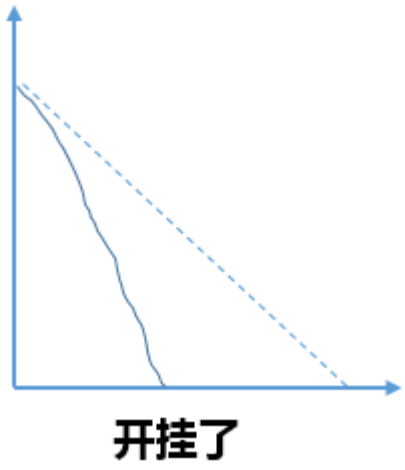
Case5：TBD。特征：没终点，剩一些活做不完了。 图形如下：

可能情况：

1. 领导压迫给活多，每天都有新需求。
2. 团队开黑打酱油去了，很少人做事。
3. SM打酱油，无风险预估及处理能力。
4. 团队不能合理安排工作。

遇到这种问题，还是要做风险预判。实在做不完的推到下个sprint，但千万别给这个sprint延长周期（比如2周调成3周）。规则一旦被打破，下次团队会不再遵守Timebox（时间盒），也就没有固有的节奏感了。

Case6：开挂。特征：直线下降，一直处于理想曲线下方。 图形如下：

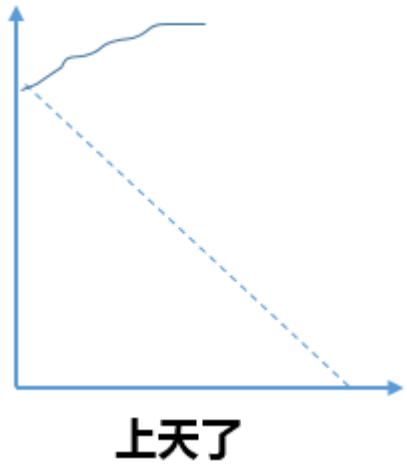


可能情况：

1. 需求被高估了，实际并没有这么多的活做。
2. 需求被删减了。
3. 公司来了新的鼓励师，程序员玩命工作，效率很高。

这种情况最看不下去的就是PO，通常会鼓（ya）励（po）大家多领一些任务，同样，作为一名专（da）业（za）的SM，也应当鼓励团队多搬砖多领钱，这样才能给妹子买包包啊！
SM要优化团队效率，保持稳定的节奏感。

Case7：上天了。特征：一直上翘。图形如下：



这种情况曲线没有下降，反而一直上升，最可能的情况是每天都有新需求加入到当前sprint中。当然，也有可能是开发人员评估工作的时候太过于乐观，导致工作量不断膨胀，最后的结果是当前sprint必须终止。

以上情况描述了绝大多数情况下的燃尽图类型，SM要做的就是提前预判，风险预防，而不是等到了问题发生才去解决。

燃尽图常见问题

Q1：不知道怎么画

这种情况可能有：

- 团队成员都是新成员，没接触过敏捷，所以前期都是交由SM来绘制燃尽图，但是后期，最好是让团队成员来进行更新。
- 没有工时估算，也自然就没办法知道做了多少活，所以最好是估算每个活动的工时。

Q2：颗粒度大小问题

这种情况可能有：

- 一般来说Y轴常用小时数或者故事点数。
- 颗粒度最好是1天可以完成的事情。这样不至于每天Daily Meeting都说同一句话：“我昨天做这部分的代码编写，今天继续……”。二来，可以更好的跟踪任务项，减少进度风险。

这里面还有个问题是story数量的问题。如果story太少，我们画燃尽图的时候会发现折现非常明显，可能会让我们错误的认为当前速度及风险比较大。

Q3：测试背锅

燃尽图变成了Case5:TBD的情况，领导一问，开发同学异口同声：测试不给力，我们的活都干完了。

很多时候开发和测试是敌对的双方。研发小伙伴在sprint最后一刻才提交给测试，测试自然就没有时间来做测试的工作。然后就很无辜的做了背锅侠…

又或者，开发质量不过关，测试测出来很多bug，然后开发返工，导致sprint延期，开发小伙伴同声：你丫测试不给力，这么晚才告诉我有问题。测试同学继续背锅…
这种情况，我们最好不要把所有的故事都在同一个时间完成，同样，也要提倡团队跨职能发展，打造T型团队人才。

Q4：燃尽图日期加了1天

当还差1天的工作就可以完成这个sprint的时候（此时团队24h工作无休），你会怎么做？很多Team会在燃尽图加1天，这样就可以完美的抵达Y=0的点。

切记不可这么做！一旦规则被打破，团队成员也会认为这个时间都是可协商的，也会破坏固有的节奏感。这次加了，那下次还剩1.5天的工作量的时候，要不要加？

Q5：只统计某个工作量

燃尽图显示的是整个team的工作实际统计情况，而非单独开发或者测试的工作量（排除单一组件团队的特例）

Q6：燃尽图比较

如果公司有几个敏捷项目同时在运作，领导看到A组同学的燃尽图说到：“你看B组，他们速度多快啊，你们这速度太慢了，你们这周过来加加班吧，把速度搞上去…”
这种情况很明显是被 “懂”敏捷的领导给坑了。遇到个完全不懂的倒还好，最怕的就是一知半解的。

不同team甚至是同一个team，燃尽图都可能无法比较。比如有如下原因都可能导致燃尽图不同：

1. Story拆分的粒度是否统一。
2. 团队成员是否有变化。
3. 时间周期是否一致。
4. story point的基准是否一致。

燃尽图实例剖析

实例1：是否会画燃尽图？下图是某项目2天的执行情况，请问A B C三个点分别应该是多少？

Day 0

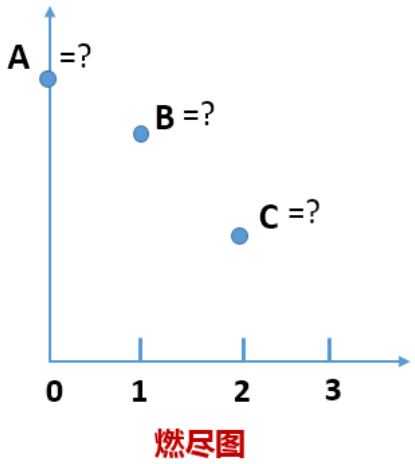
To Do	Doing	Done
No.1 xxxxx 5 No.2 xxxxx 3 No.6 xxxxx 8	No.5 xxxxx 5 No.4 xxxxx 7	

Day 1

To Do	Doing	Done
No.1 xxxxx 5 No.2 xxxxx 3 No.6 xxxxx 8	No.5 xxxxx 5	No.4 xxxxx 7

Day 2

To Do	Doing	Done
No.1 xxxxx 5 No.2 xxxxx 3	No.6 xxxxx ~~8~~ 3	No.4 xxxxx 7 No.5 xxxxx 5



分析：燃尽图中Y轴代表的是剩余工作量，掌握这个即可知道是多少数值。只有放到Done列表的，叫完成。

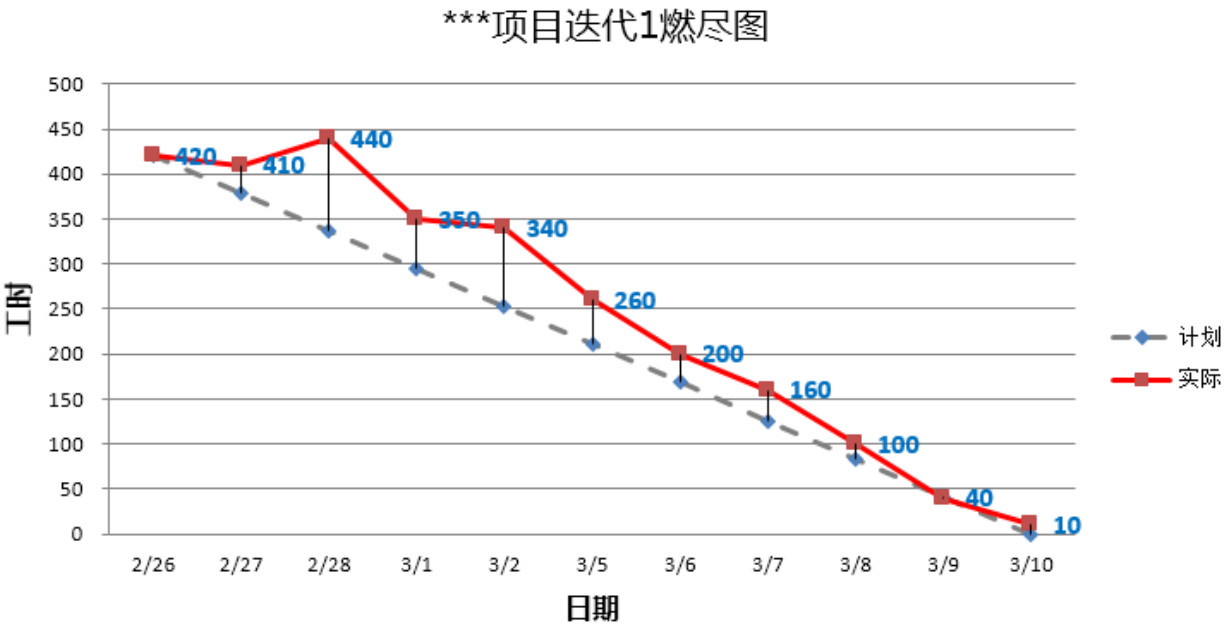
A=5+3+8+5+7=28

B=28-7=21

C=21-5-(8-3)=11，因为No.6这个编号的工作，8个点的活变成了3个点，所以剩余少了5个。再加上No.5完成了。故总计完成10个点的活。

实例2：截取了IT事业部某软件外包项目燃尽图实例，跟大家一起分享我们出了什么情况以及如何解决的。（2周为一个sprint，团队已经运用敏捷一段时间）

我们在每日站会上进行数据的收集和更新，然后根据SBI（sprint backlog items）来更新数据。



迭代1分析：

1. 在2.27-28这一天实际曲线不但没有下降，反而上升。主要原因：

（a）开始第一个sprint，需求梳理的不是很清楚，有新需求新增到SPI中（团队一开始是拒绝的，但PO比较强势，必须得加进来…）。

（b）工作项的分解不熟悉。task重新估算之后，用时有所增加。

此时项目已经出现了风险，而且有将近30个点新增。所以为了解决问题，符合预期。在2.28-3.1，我们采用了加班的方式。

1. 3.1-3.2，仅下降了10个点，主要原因：

（a）前一晚加班太晚，今天效率不高，还有几个同事休假。也可以看出，加班过猛也不是好事…效率最重要。

（b）有个技术坑阻碍了当前的项目进度，team研究好久都没搞定。

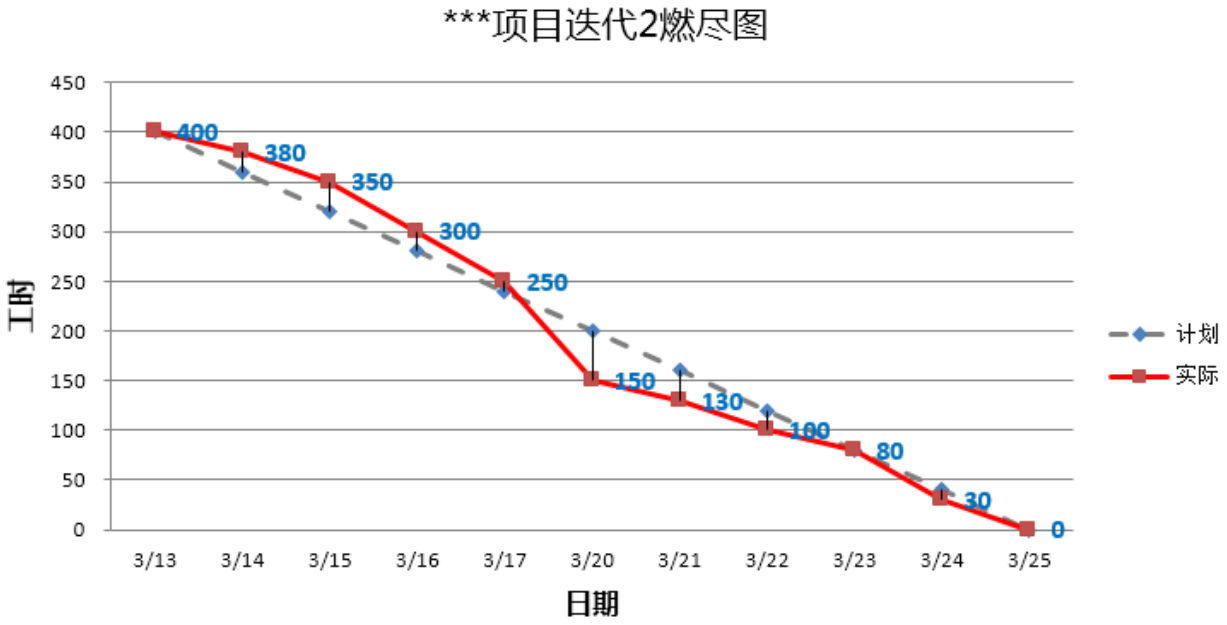
1. 3.3-3.4 休息日。开发小伙伴顺便在周末研究了一下技术难题。

2. 3.5-3.9，从曲线的斜率可以看出，团队的实际速度高于理想速度，逐步贴近我们的理想曲线。

3. 到3.10号，我们还剩10个点的活没干完，只要原因：开发组将迭代版本提交给测试进行迭代系统测试导致。

（a）开发将迭代版本提交给测试，遇到一些重要bug需要处理，导致待开发项也没有时间安排处理。

（b）估算存在一些问题，项目设置的冗余时间也已经用完。



迭代2分析：

1. 鉴于在迭代1中最终有故事没有完成，而且团队在整个sprint中工作压力较大，所以我们在估算工作量总工时的时候，只挑选了400个点的工作，相比迭代1少了20个。这也说明在敏捷中，即使是同一个团队，每次能完成的工作量，也会有不同。
2. 3.17-3.20期间，从250点直落到150点，主要是由于抽调了部分的需求，放到下个迭代中。所以曲线直接掉到理想曲线下方。
3. 3.20-3.25，速率正常，3.25刚好完成所有的工作。

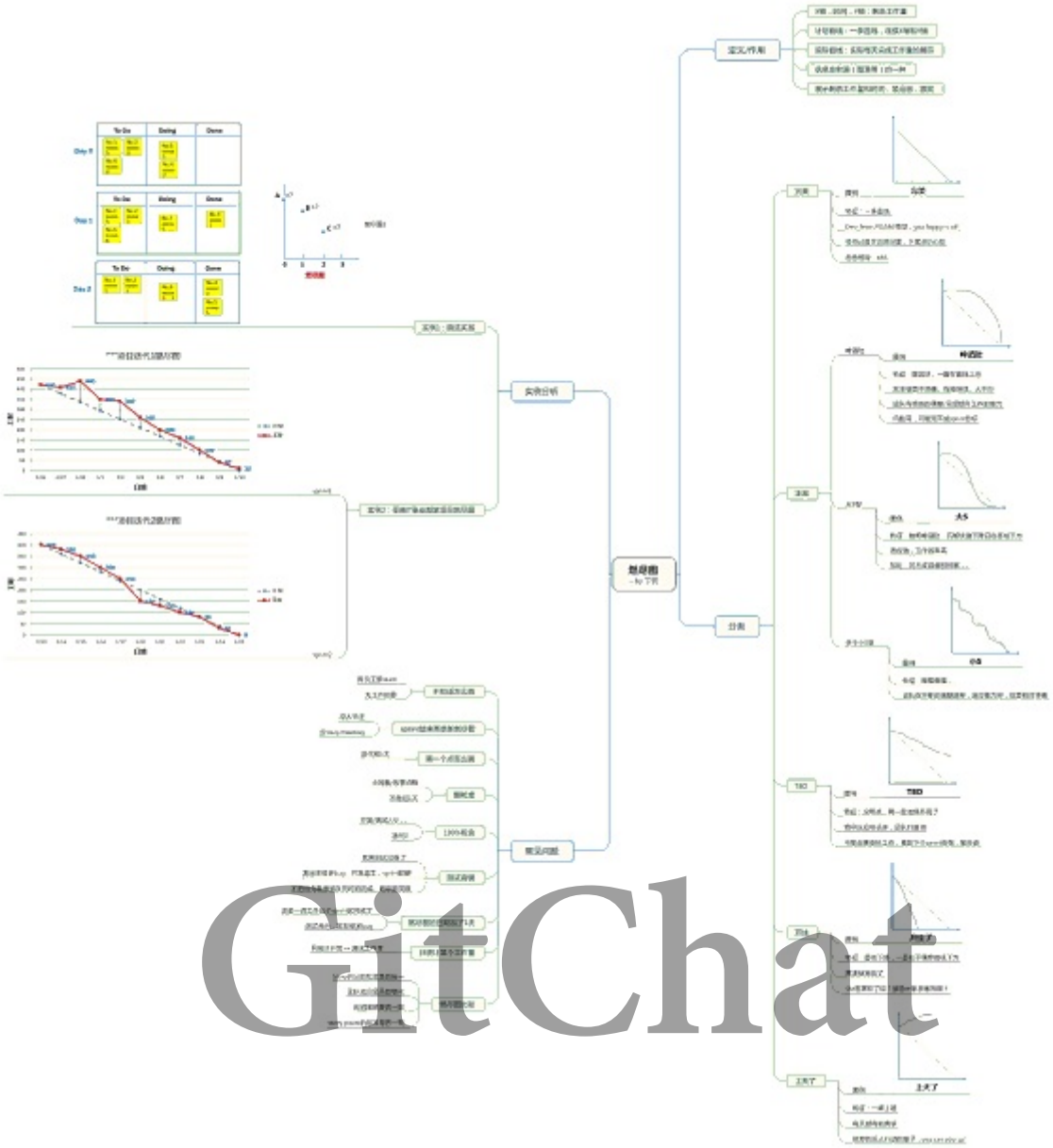
总结

燃尽图作为敏捷可视化管理的工具，发挥着重要的作用。一叶知秋，作为敏捷教练应当能够从每一张图表、每一条折线，分析出项目背后的问题，防范于未然。燃尽图的数据来源于日常工作，出自于团队本身。所以数据的准确性，直接决定了燃尽图的价值。

怎么样的燃尽图才是有价值的呢？

1. 数据要真实，统计全面。这样才能反映出task的执行情况。
2. 对于工作量的预估要尽量准确，这就对团队的技术水平有一定的要求。
3. 粒度的分解要尽量落实到每一天。太大的粒度无法跟踪，而且不利于工作的估计。

文末彩蛋，本文思维导图



下载地址：
链接：http://pan.baidu.com/s/1jIn2K66 密码：tkmt

以上浅见，还望与各位共同交流。

下一期，我们来聊聊那些年被坑的每日站会……