1

00:00:01,000 --> 00:00:05,000

本中文字幕由自发性 Scrum 团队提供：

PO 周龙鸿／翻译：王可帆, 任兰怡, 江岳龙, 余俊杰,

周玉萍, 林士智, 林清雅, 邱畯丞, 张峰睿, 张越程,

张巨鑫, 陈美凤, 黄久娟, 廖淑萍／时间轴：ZephyR(西风) / 审校：申健

2

00:00:06,000 --> 00:00:10,760

Sportify成功的重要原因之一

就是我们的敏捷(Agile)工程文化

3

00:00:10,760 --> 00:00:12,040

文化，是无形的

4

00:00:12,040 --> 00:00:14,110

我们察觉不到

因为文化随时随地都在那里

5

00:00:14,110 --> 00:00:15,730

就像是我们呼吸的空气一样

6

00:00:15,730 --> 00:00:19,140

但若是大家都理解这个文化

我们就有机会去保持它

7

00:00:19,140 --> 00:00:20,770

甚至在我们成长的同时能强化它

8

00:00:20,770 --> 00:00:22,930

这就是这部影片的目的

9

00:00:22,930 --> 00:00:25,460

当2008年我们推出第一版音乐播放器时

10

00:00:25,460 --> 00:00:27,370

我们公司就已经熟悉运用Scrum了

11

00:00:27,370 --> 00:00:30,070

Scrum是一种广为接受的敏捷开发方法

12

00:00:30,070 --> 00:00:32,149

Scrum提供了很棒的基于团队的文化

13

00:00:32,149 --> 00:00:35,550

然而，几年之后

我们已发展成数量庞大团队

14

00:00:35,550 --> 00:00:37,730

并且发现，有些Scrum常见的标准实践

15

00:00:37,730 --> 00:00:39,410

其实造成了阻碍

16

00:00:39,410 --> 00:00:41,920

所以我们决定有选择地运用这些实践

17

00:00:41,920 --> 00:00:44,450

遵循规则是好的开始

但有需要时，是可以突破的

18

00:00:44,450 --> 00:00:46,750

那时我们做出决定

敏捷比Scrum重要

19

00:00:46,750 --> 00:00:50,140

而敏捷原则比任何特定实践重要

20

00:00:50,150 --> 00:00:52,840

所以我们把「ScrumMaster」

改称为「敏捷教练」

21

00:00:52,840 --> 00:00:55,640

因为我们更需要的是仆人式领导者

而不是流程主管

22

00:00:55,640 --> 00:00:58,630

我们也开始使用「小队」

来代替「Scrum团队」

23

00:00:58,630 --> 00:01:01,090

而关键驱动力是自主(Autonomy)

24

00:01:01,090 --> 00:01:03,500

什么是「自主小队」？

25

00:01:03,500 --> 00:01:07,090

「小队」指的是小型、

跨职能、自组织的团队

26

00:01:07,090 --> 00:01:08,950

人数通常少于八人

27

00:01:08,950 --> 00:01:11,770

小队成员聚集一起

28

00:01:11,770 --> 00:01:13,090

对于研发的所有端到端事项

29

00:01:13,090 --> 00:01:16,100

像是设计、提交、部署、维护、运营等

30

00:01:16,100 --> 00:01:17,180

共同负起全部的责任

31

00:01:17,180 --> 00:01:21,890

每个小队都有其长期使命，例如：

让Spotify成为探索音乐的最佳应用程序

32

00:01:21,890 --> 00:01:25,670

或是内部事务，例如：

执行A/B 测试的基础架构

33

00:01:25,670 --> 00:01:29,650

基本上，「自主」指的是小队

自行决定开发什么产品、如何开发

34

00:01:29,650 --> 00:01:31,440

以及如何共同合作来开发

35

00:01:31,440 --> 00:01:35,659

当然，这种做法也会有限制

例如，小队的使命

36

00:01:35,659 --> 00:01:39,000

合作开发产品的整体策略

37

00:01:39,000 --> 00:01:42,939

以及每季度重新探讨的短期目标

38

00:01:42,939 --> 00:01:44,570

办公室空间也为了加强合作而优化

39

00:01:44,570 --> 00:01:47,010

这是一个典型的小队办公区

40

00:01:47,010 --> 00:01:52,000

小队成员透过可调整的办公桌和

方便互相访问的计算机屏幕来密切合作

41

00:01:52,049 --> 00:01:56,592

小队通常在休息室开会

讨论规划事宜及回顾检讨会议

42

00:01:56,610 --> 00:02:00,800

另一边则是视频(HALO)室，用来开小型会议

或单纯只想独处一下

43

00:02:00,810 --> 00:02:02,930

几乎所有的墙面都是白板

44

00:02:02,930 --> 00:02:05,000

为什么独立自主如此重要？

45

00:02:05,000 --> 00:02:06,500

因为它的驱动力！

46

00:02:06,560 --> 00:02:08,560

受到激励的人能开发出更好的产品

47

00:02:08,820 --> 00:02:12,940

独立自主当然也让我们行动迅速

小队在团队里即能做出决定

48

00:02:12,950 --> 00:02:15,800

而不是透过层层主管

众多的委员会及规则

49

00:02:15,890 --> 00:02:18,590

独立自主帮助我们尽量避免工作交接及等待，

所以我们可以扩展

50

00:02:18,590 --> 00:02:22,040

而不致因为依赖及等待而阻碍

51

00:02:22,040 --> 00:02:26,489

虽然每个小队有自己的使命

他们还是得向产品策略看齐

52

00:02:26,489 --> 00:02:28,380

与公司整体优先级和其他小队保持一致

53

00:02:28,380 --> 00:02:31,319

基本上，就是在Spotify生态里当个好公民

54

00:02:31,319 --> 00:02:34,860

因为Spotify整体使命的重要性

高于个体小队的任务

55

00:02:34,860 --> 00:02:39,170

所以最主要的原则是独立自主

但不能局部优化

56

00:02:39,170 --> 00:02:40,810

这就像一个爵土乐团

57

00:02:40,810 --> 00:02:43,920

虽然每个乐手都独立自主地

演奏着自己的乐器

58

00:02:43,920 --> 00:02:47,220

却又同时聆听彼此的演奏

一起专注整首曲子的演出

59

00:02:47,220 --> 00:02:49,459

好的音乐就是这样产生的

60

00:02:49,459 --> 00:02:53,050

所以我们的目标是形成一种彼此松耦合

但整体上高度一致的小队

61

00:02:53,050 --> 00:02:57,600

尽管我们尚未完全达到以上所说的

但我们尝试了许多方法

62

00:02:57,629 --> 00:03:00,349

事实上，影片里所谈到的多是这个原则

63

00:03:00,349 --> 00:03:03,660

我们目前已达到这种工程文化的部分状态

64

00:03:03,660 --> 00:03:06,580

未来还有要努力的地方

65

00:03:06,580 --> 00:03:09,510

「一致性」和「自主性」

就像天平的两端

66

00:03:09,510 --> 00:03:12,610

一般认为，自主性高相对的一致性就少

67

00:03:12,610 --> 00:03:16,019

但我们比较认为

这是两个不同维度的事情

68

00:03:16,019 --> 00:03:19,000

左下这里代表低一致性与低自主性

69

00:03:19,000 --> 00:03:20,819

也就是一种微管理文化

70

00:03:20,819 --> 00:03:22,110

这种管理没甚么高阶目的

71

00:03:22,110 --> 00:03:24,010

有的只有闭嘴和听命行事

72

00:03:24,010 --> 00:03:27,019

左上这里是一致性高，但是自主性低

73

00:03:27,019 --> 00:03:30,330

领导者除了擅于沟通哪些问题需要解决

74

00:03:30,330 --> 00:03:32,790

还会去告诉他人问题该如何解决

75

00:03:32,790 --> 00:03:38,340

但是高一致性和高自主性意味着

领导者只专注在要解决什么问题

76

00:03:38,340 --> 00:03:41,129

但让团队成员去找出解决问题的方式

77

00:03:41,129 --> 00:03:42,750

右下又是如何呢？

78

00:03:42,750 --> 00:03:46,680

低一致性与高自主性代表团队各行其事

79

00:03:46,680 --> 00:03:49,290

各做各的，没有一致的方向

80

00:03:49,290 --> 00:03:51,560

领导者很无助

而产品则成了怪胎

81

00:03:51,560 --> 00:03:55,849

这一块是我们较想达到的

具备一致性的自主

82

00:03:55,849 --> 00:03:58,980

我们用尽了各种办法来达到这种状况

83

00:03:58,980 --> 00:04:01,180

一致性使得自主成为可能

84

00:04:01,180 --> 00:04:05,330

我们的一致性越强

我们就越能下放自主

85

00:04:05,330 --> 00:04:10,129

这表示领导者的角色是去沟通

哪些问题需要解决及其原因

86

00:04:10,129 --> 00:04:13,860

而整个敏捷小队就要协力找出最好的解决方案

87

00:04:13,860 --> 00:04:18,190

自主的一个结果是我们少有需要标准化的情形

88

00:04:18,190 --> 00:04:21,030

当有人问道：

你们使用哪一种程序编辑器

89

00:04:21,030 --> 00:04:22,369

或是你们如何计划？

90

00:04:22,369 --> 00:04:25,199

答案通常是：

这要看是哪一个小队

91

00:04:25,199 --> 00:04:27,759

有些用的是Scrum Sprints

有些采用了Kanban管理

92

00:04:27,759 --> 00:04:30,819

有些会估算故事点，并测量开发速率(Velocity)

有一些则不会

93

00:04:30,819 --> 00:04:32,749

这真的是因各小队而异

94

00:04:32,749 --> 00:04:37,089

所以与其用标准规范

我们要的是这种异花传粉的文化

95

00:04:37,089 --> 00:04:40,999

当越来越多的小队都采用某种特定的方法

如Git这种版本控制软件

96

00:04:40,999 --> 00:04:45,819

阻力就会越来越少

因为其他小队就会开始跟进

97

00:04:45,819 --> 00:04:48,129

当小队间都使用这种工具并彼此协助时

98

00:04:48,129 --> 00:04:50,419

它就成了一个实质上的标准

99

00:04:50,419 --> 00:04:55,600

这种非正式的作法，让我们得以

在一致性和灵活性之间保持良好的平衡

100

00:04:55,689 --> 00:05:00,500

我们的架构含有一百多个

独立开发和部署的系统

101

00:05:00,529 --> 00:05:04,839

系统间会有许多的互动

但每一个系统各自专注于一个特定的需求

102

00:05:04,839 --> 00:05:08,119

像是播放清单的管理、搜索或监控

103

00:05:08,119 --> 00:05:11,969

我们以清楚的接口及规则

试着让这些需求单纯化而且解耦合

104

00:05:11,969 --> 00:05:14,430

技术上来说

每个系统会都会由一个小队负责

105

00:05:14,430 --> 00:05:16,089

事实上

大多数的小队会负责好几个系统

106

00:05:16,089 --> 00:05:18,659

但我们有一个「内部开源」模式

107

00:05:18,659 --> 00:05:21,529

而我们的文化重视分享更甚于拥有

108

00:05:21,529 --> 00:05:24,589

试想我们的第一小队需要在系统B完成某件事

109

00:05:24,589 --> 00:05:26,849

但第二小队对系统B的程序撰写比较在行

110

00:05:26,849 --> 00:05:29,020

通常第一小队会请第二小队协助处理

111

00:05:29,020 --> 00:05:32,629

然而，如果第二小队没有时间

或他们有其他更重要的任务

112

00:05:32,629 --> 00:05:35,240

第一小队不必等待…

113

00:05:35,240 --> 00:05:36,289

我们讨厌等待

114

00:05:36,289 --> 00:05:39,249

反而，我们鼓励继续进行

由第一小队自行去写代码

115

00:05:39,249 --> 00:05:41,619

之后再请第二小队评审改动

116

00:05:41,619 --> 00:05:45,649

所以任何人都可以修改程序

但我们有一个同侪代码评审的文化

117

00:05:45,649 --> 00:05:48,009

这有助于质量的改善，更重要的是

118

00:05:48,009 --> 00:05:49,240

知识的传播

119

00:05:49,240 --> 00:05:52,419

随着时间累积

我们已经发展出一些设计指引

120

00:05:52,419 --> 00:05:54,969

代码规范以及其他

来降低工程上的摩擦

121

00:05:54,969 --> 00:05:56,229

但唯有在非常时期才会使用

122

00:05:56,229 --> 00:06:01,080

所以，在权威与自由的天平上

我们无疑是倾向自由的

123

00:06:01,080 --> 00:06:04,819

现在，唯有以人为本，工作才得以顺利推展

124

00:06:04,819 --> 00:06:07,529

我们有坚实的互相尊重的文化

125

00:06:07,529 --> 00:06:10,960

我随时听到像是

「我的队友真棒呀」的意见

126

00:06:10,960 --> 00:06:13,449

人们常会归功于其他人的杰出表现

127

00:06:13,449 --> 00:06:15,879

而非自私地为自己争功

128

00:06:15,879 --> 00:06:19,869

如果要说我们这里确实有许多天才

却很少一人独大

129

00:06:19,869 --> 00:06:24,039

对菜鸟而言，第一眼对我们感到惊叹的地方

就是我们的独立自主

130

00:06:24,039 --> 00:06:26,969

你和你的队友会被期望自己去找答案

131

00:06:26,969 --> 00:06:28,860

没人会告诉你该做什么

132

00:06:28,860 --> 00:06:32,499

但当你需要协助时

你会很快地获得许多援助

133

00:06:32,499 --> 00:06:36,259

大家一致认同的真理是

我们在同一条船上

134

00:06:36,259 --> 00:06:38,639

而且必须帮助他人成功

135

00:06:38,639 --> 00:06:40,719

我们非常聚焦在激励这件事

136

00:06:40,719 --> 00:06:43,740

举个实例，一封由人力资源部主管

寄出的邮件中写道：

137

00:06:43,740 --> 00:06:47,689

「大家好，

根据我们的员工满意度调查显示

138

00:06:47,689 --> 00:06:51,569

91%表示在这里工作感到满意，4%不满意」

139

00:06:51,569 --> 00:06:54,169

那满意度看起来可能相当不错

140

00:06:54,169 --> 00:06:59,839

尤其是考虑到从2006到2013年

我们每年翻倍成长的痛苦期

141

00:06:59,839 --> 00:07:03,249

而现在我们员工数已超过1,200名

然而人力资源部主管仍表示：

142

00:07:03,249 --> 00:07:07,219

「这份员工满意度当然不算高

我们想要修正

143

00:07:07,219 --> 00:07:10,379

若你是那4%不满意的其中一位

请与我们联络

144

00:07:10,379 --> 00:07:12,839

我们会在此为您服务，绝无他意」

145

00:07:12,839 --> 00:07:15,179

所以，高度满意不代表真正满意

146

00:07:15,179 --> 00:07:19,209

半年后情况已改善到满意度升至94%

147

00:07:19,209 --> 00:07:22,059

毫无疑问的，原因就是坚持

以激励为公司文化重点

148

00:07:22,059 --> 00:07:25,179

我们因而获得最佳职场的封号

149

00:07:25,179 --> 00:07:27,959

我们当然还有许多问题需要解决

150

00:07:27,959 --> 00:07:29,899

所以我们需要持续精进

151

00:07:29,899 --> 00:07:34,349

由于我们有50个以上的小队

分布于四个城市

152

00:07:34,349 --> 00:07:36,469

某种的组织架构还是需要的

153

00:07:36,469 --> 00:07:38,599

小队现在分组成「部落」

154

00:07:38,599 --> 00:07:40,889

部落是一种轻量级的矩阵型组织

155

00:07:40,889 --> 00:07:44,139

个人是小队成员，同时也是

「分会」一员

156

00:07:44,139 --> 00:07:48,389

小队是主要维度

聚焦在产品交付与质量

157

00:07:48,389 --> 00:07:52,110

而分会是一个技能领域

例如对质量的支持

158

00:07:52,110 --> 00:07:54,409

敏捷教导(Coaching)或Web开发

159

00:07:54,409 --> 00:07:57,569

作为一个小队成员

我的分会领导就是直属主管

160

00:07:57,569 --> 00:08:01,559

作为「仆人式领导者」聚焦在

教导和指导我这个工程师

161

00:08:01,559 --> 00:08:04,279

所以我可以在不换新主管的

情况下转换小队

162

00:08:04,279 --> 00:08:05,999

这是一个很棒的图画，对吧？

163

00:08:05,999 --> 00:08:07,819

只可惜，并不全然是事实

164

00:08:07,819 --> 00:08:11,610

实际上，任务编组并非一帆风顺

而是瞬息万变

165

00:08:11,610 --> 00:08:15,469

举个实例，部落在某一瞬间

可能就全变了

166

00:08:15,479 --> 00:08:18,330

而那算是还好的

167

00:08:18,330 --> 00:08:22,749

最有价值的沟通会出现在

非正式且无法预测的方式上

168

00:08:22,749 --> 00:08:25,139

为了支持这种方式

我们也有「公会」

169

00:08:25,139 --> 00:08:28,239

公会是一个轻量级的兴趣社群

170

00:08:28,239 --> 00:08:32,020

跨整个公司的成员齐聚

并分享特定领域知识

171

00:08:32,020 --> 00:08:35,810

例如：领导力、Web开发

或持续交付

172

00:08:35,810 --> 00:08:38,229

任何人皆可随时参加或离开公会

173

00:08:38,229 --> 00:08:41,589

基本上公会有电邮名单

可召开半年度会议

174

00:08:41,589 --> 00:08:44,159

和其他非正式的沟通渠道

175

00:08:44,159 --> 00:08:48,399

多数的组织结构都是假象

所以我们聚焦在社区

176

00:08:48,399 --> 00:08:50,209

而不是阶层式架构

177

00:08:50,209 --> 00:08:55,459

我们认为一个够强大的社区可以

避免变成不稳定的组织架构

178

00:08:55,459 --> 00:08:58,519

如果你永远需要确切地知道谁做决定

179

00:08:58,519 --> 00:09:00,560

你其实不适合这里

180

00:09:00,560 --> 00:09:06,600

对于自主性最重要的一点是

我们去部署上线的难度

181

00:09:06,600 --> 00:09:09,709

如果程序发布很困难而我们选择减少发布

182

00:09:09,709 --> 00:09:11,519

来避免成员痛楚

183

00:09:11,519 --> 00:09:14,430

那意味着每次发布的规模更大，而发布就更困难

184

00:09:14,430 --> 00:09:16,350

那是一个恶性循环

185

00:09:16,350 --> 00:09:18,910

若程序发布容易我们即可频繁发布

186

00:09:18,910 --> 00:09:22,279

那意味着每次发布的规模更小，而程序发布更加容易

187

00:09:22,279 --> 00:09:27,540

维持这样的循环而避免另一种

我们鼓励小而频繁的程序发布

188

00:09:27,540 --> 00:09:31,810

且投资更多在测试自动化和

持续交付的基础设施

189

00:09:31,810 --> 00:09:35,139

程序发布应该是例行性的而非戏剧化出场

190

00:09:35,139 --> 00:09:37,819

有时候我们投入很多资源

让发布更容易

191

00:09:37,819 --> 00:09:41,040

举例来说

原本的Spotify桌面程序

192

00:09:41,040 --> 00:09:42,930

是一只单独、庞大的应用程序

193

00:09:42,930 --> 00:09:45,819

在早期，只由少数的开发者负责

194

00:09:45,819 --> 00:09:47,250

这还好

195

00:09:47,250 --> 00:09:50,079

但是随着我们成长

这变成很大的问题

196

00:09:50,079 --> 00:09:52,750

许多小队针对每个发布必须彼此同步

197

00:09:52,750 --> 00:09:55,970

这可能耗时数月

才能发布一个稳定的版本

198

00:09:55,970 --> 00:09:59,350

与其建立大量的流程

规则之类的东西来管理这情况

199

00:09:59,350 --> 00:10:02,930

我们改变了架构

让发布解耦合

200

00:10:02,930 --> 00:10:07,699

使用Chromium嵌入式架构

基本上客户端就是个伪装的网页浏览器

201

00:10:07,699 --> 00:10:12,550

每个区块就像网站的一个框架

让小队可以直接发布他们的产品

202

00:10:12,550 --> 00:10:14,810

当这部分的架构改变

203

00:10:14,810 --> 00:10:18,420

我们开始把每个客户平台

看做一个客户App

204

00:10:18,420 --> 00:10:21,100

并且展成三种不同重点的小队

205

00:10:21,110 --> 00:10:25,430

客户App小队、功能特性小队

与基础设施小队

206

00:10:25,430 --> 00:10:29,410

功能特性小队专注在某块

功能特性，比如搜索功能

207

00:10:29,410 --> 00:10:33,790

这个小队会在全部平台上

构建、交付及维护搜索相关的特性

208

00:10:33,790 --> 00:10:39,100

客户App小队专注于使

特定的客户端平台容易发布

209

00:10:39,100 --> 00:10:42,149

像是桌面程序、iOS或Android

210

00:10:42,149 --> 00:10:45,240

基础架构小队专注在

让其他小队更有效率

211

00:10:45,240 --> 00:10:47,680

他们提供工具和例行事务，像是：

212

00:10:47,680 --> 00:10:52,209

持续交付、A/B测试

监控和运维

213

00:10:52,209 --> 00:10:56,329

无论目前结构为何

我们始终采用追求一个「自助服务模式」

214

00:10:56,329 --> 00:11:00,410

有点像是自助餐

餐厅服务员不会直接服务顾客

215

00:11:00,410 --> 00:11:02,240

他们让你自己来服务自己

216

00:11:02,240 --> 00:11:04,850

所以我们避免交接

217

00:11:04,850 --> 00:11:08,189

举例来说，一个运维小队

或客户端App小队

218

00:11:08,189 --> 00:11:10,980

他们并不会把帮人们把代码放入产品中

219

00:11:10,980 --> 00:11:16,360

他们的工作是让功能特性小队

更容易将自己的代码上线

220

00:11:16,360 --> 00:11:18,290

尽管是自助模式

221

00:11:18,290 --> 00:11:21,699

我们有时需要

小队之间有一些同步来进行发布

222

00:11:21,699 --> 00:11:25,800

我们采用发布火车和特性开关

来处理这个议题

223

00:11:25,800 --> 00:11:30,240

每一个客户端App有一个发布火车

依照固有的日程离站

224

00:11:30,240 --> 00:11:33,410

通常是每周或每三周

这要看哪一个客户端

225

00:11:33,410 --> 00:11:36,889

就像在实体的世界

如果列车频繁且可靠的离站

226

00:11:36,889 --> 00:11:39,439

你就不需要太频繁的事前规划

227

00:11:39,439 --> 00:11:41,600

只要现身并搭乘下一班车

228

00:11:41,600 --> 00:11:45,620

时

即使这三个小队还在构建产品中

229

00:11:45,620 --> 00:11:50,779

功能特性A、B和C已经完成

而D仍然在进行中

230

00:11:50,779 --> 00:11:53,559

发布火车将会包含所有功能特性

231

00:11:53,570 --> 00:11:56,639

但是未完成的那个会通过

特性开关隐藏起来

232

00:11:56,639 --> 00:12:00,019

这也许听起来很怪

发布未完成的功能特性却隐藏起来

233

00:12:00,019 --> 00:12:03,120

这个的好处在于尽早暴露集成问题

234

00:12:03,120 --> 00:12:05,319

并且尽量减少代码版本分支的需要

235

00:12:05,319 --> 00:12:09,610

未合并的代码隐藏了问题

这是种典型的技术债

236

00:12:09,610 --> 00:12:13,670

功能特性开关让我们可以

动态的呈现与隐藏产品中正在测试的部分

237

00:12:13,670 --> 00:12:16,230

除了隐藏未完成工作外

238

00:12:16,230 --> 00:12:20,340

我们也在此使用A/B测试

来逐渐推出新完成的功能特性

239

00:12:20,350 --> 00:12:23,480

总的来说，我们发布代码比过去更好

240

00:12:23,480 --> 00:12:25,589

但是我们仍然看到很多待改善的部分

241

00:12:25,589 --> 00:12:27,660

所以我们继续实验

242

00:12:27,660 --> 00:12:30,069

这模式看起来可能很恐怖的

243

00:12:30,069 --> 00:12:34,820

让每个小队添加功能到产品中

却没有任何中央管控

244

00:12:34,820 --> 00:12:36,360

我们有时候也会搞砸

245

00:12:36,360 --> 00:12:40,100

但是我们学到信任比控制更重要

246

00:12:40,110 --> 00:12:42,689

我们怎么会雇用我们不信任的人？

247

00:12:42,689 --> 00:12:48,189

敏捷的扩展得有多广，信任的范围就多大

这意味不能有勾心斗角

248

00:12:48,189 --> 00:12:50,519

也无需恐惧

249

00:12:50,519 --> 00:12:53,910

恐惧不仅会磨灭信任

恐惧也会扼杀创新

250

00:12:53,910 --> 00:12:57,079

因为若是失败犯错会被惩罚

人们会不敢尝试新的事物

251

00:12:57,079 --> 00:12:59,100

接下来，让我们来谈谈「失败」

252

00:12:59,100 --> 00:13:00,839

好吧，算了！

253

00:13:00,839 --> 00:13:02,160

休息一下

254

00:13:02,160 --> 00:13:04,600

放慢脚步

泡杯咖啡

255

00:13:04,600 --> 00:13:08,200

让上面的内容稍微放一下

当您准备好，我们再继续下一集

256

00:13:11,000 --> 00:13:12,020

字幕版本：20150915.3