

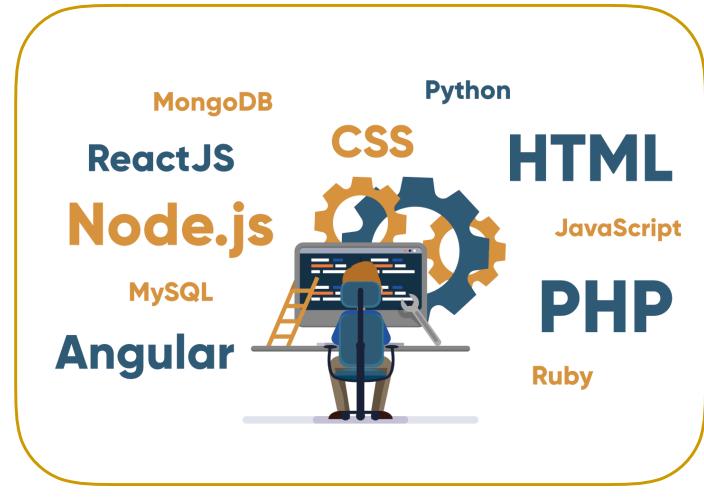
# 团队在高速扩张中的能力构建与质量保证

张思楚

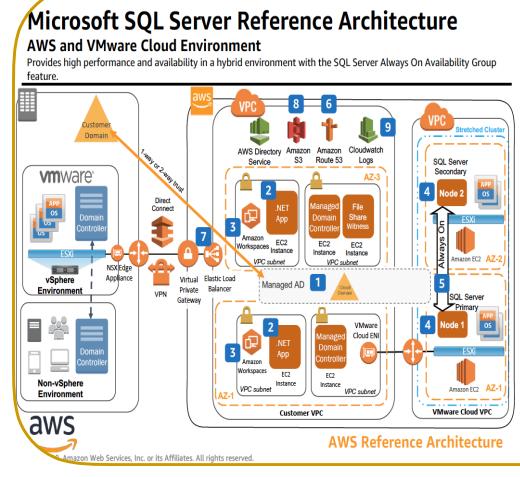
ThoughtWorks Technical Principal

TOP1

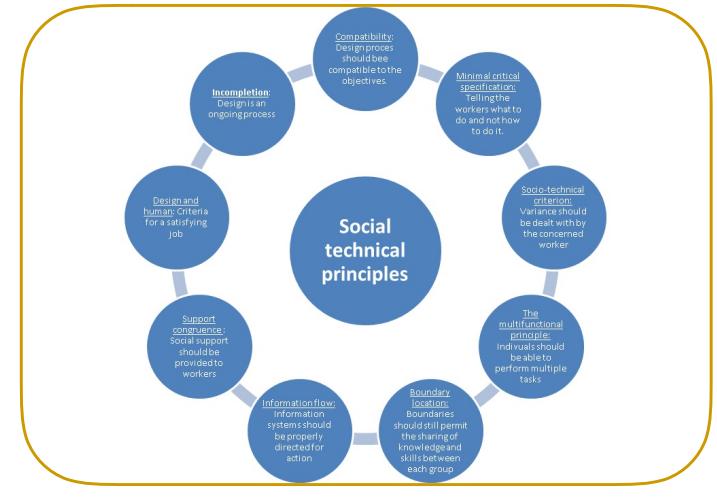
我是谁？



Full Stack Developer



Architect



Technical Principal

## 案例背景

2019新财年，客户希望在中国的ODC(Offshore Delivery Center, 离岸交付中心)，从原来的不到20人的规模，在未来3个月内扩张到60多人，希望中国的ODC能按照规定的时间完成人才招聘和供给，希望更快速的交付更多的新功能、希望构建完善的人才梯队，同时避免因为新人快速扩张而产生质量问题、线上事故。这尤其对于客户的数字化部门的高级经理来说是业绩的重要衡量指标。

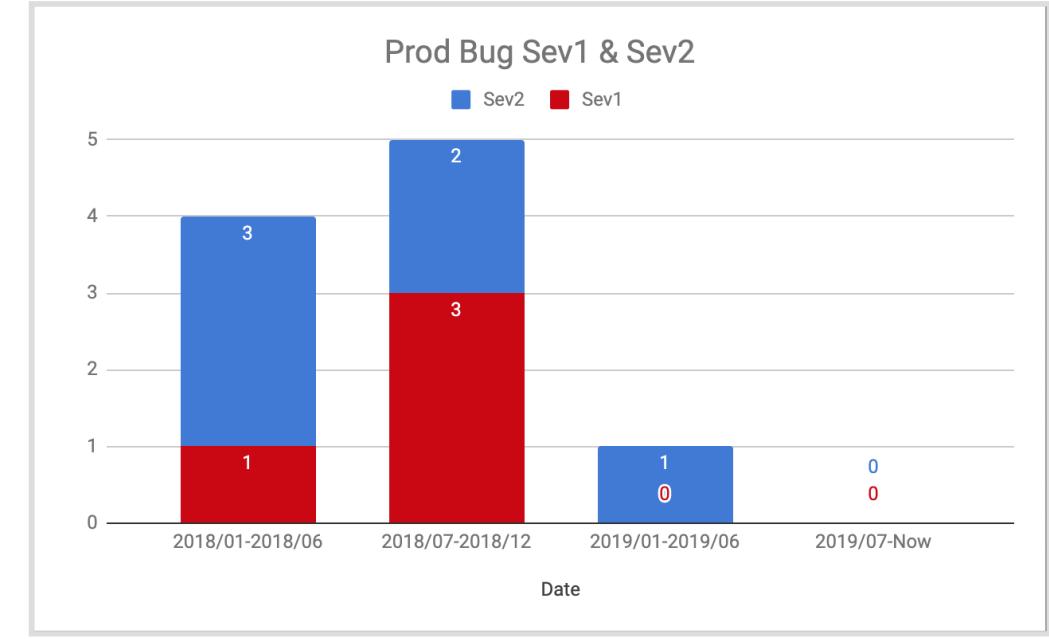
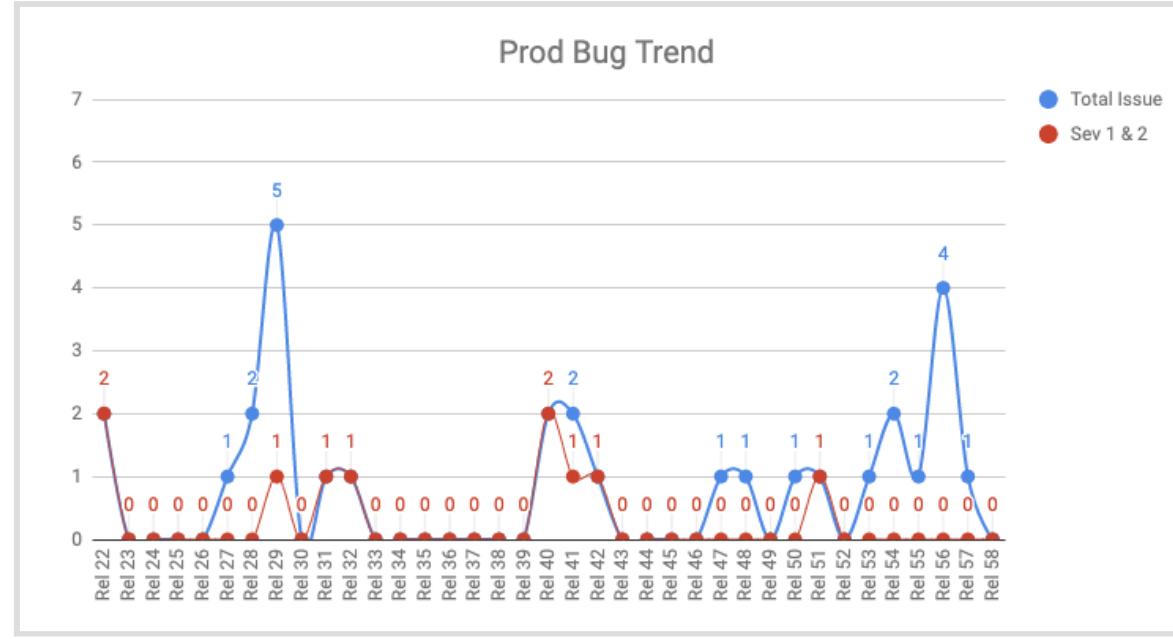


## 摘要

- ★ 研发团队需要从原来的不到20人规模，在未来3个月内扩张到60多人。
- ★ 希望交付更多新功能、同时避免因为新人快速扩张而产生质量问题、线上事故。
- ★ 如何缩短新人的成熟时间、加快交付速率的同时保证质量、避免线上事故？
- ★ 本案例将系统化的介绍整个过程中的经验和教训，帮助您安全的完成团队快速扩张。

## 问题与挑战

- ★ 如何缩短新人的成熟时间、加快交付速率的同时保证质量、避免线上事故？
- ★ 如何构建良性团队氛围，减少知识的稀释，形成合适的人才梯队？
- ★ 如何从手把手的知识传递，变为自组织自学习团队？



**TOP1**🌟

## 怎么做到的



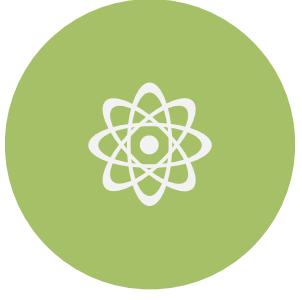
★ 快速人员成长



★ 线上事故回顾—报忧文化



★ 人才梯队构建



★ 社区&自学习团队

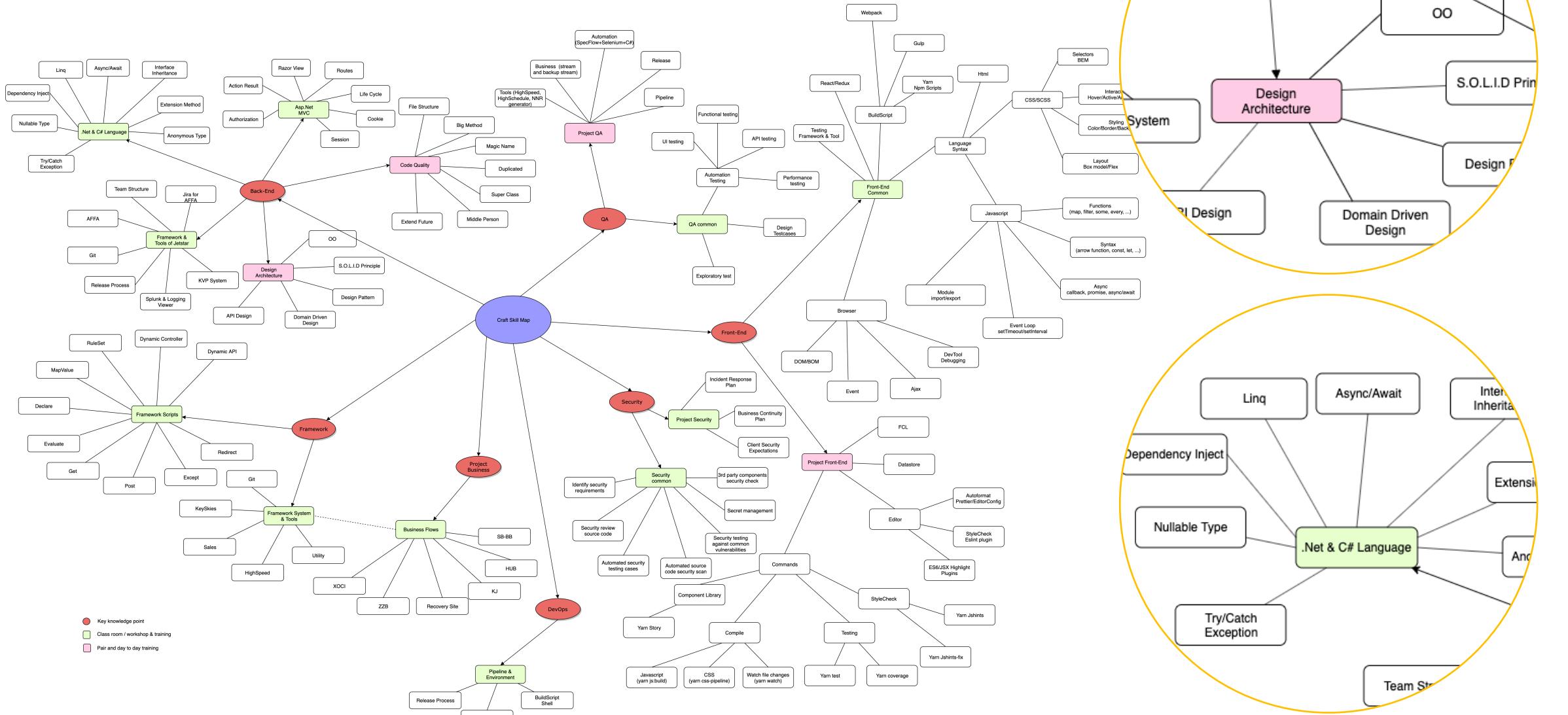


★ 快速人员成长

## 快速人员成长

- ★ CraftSkill Map，梳理完整的技术能力图谱，可视化人员需要掌握的能力。
- ★ 制定Onboarding流程，各个阶段的Homework和检查点。
- ★ 一致期望，新成员状态看板，红黄绿三状态跟踪人员状态，尽早发现风险并采取措施。
- ★ Case by Case 针对性培训，量身定制、认知转变、技能转换。

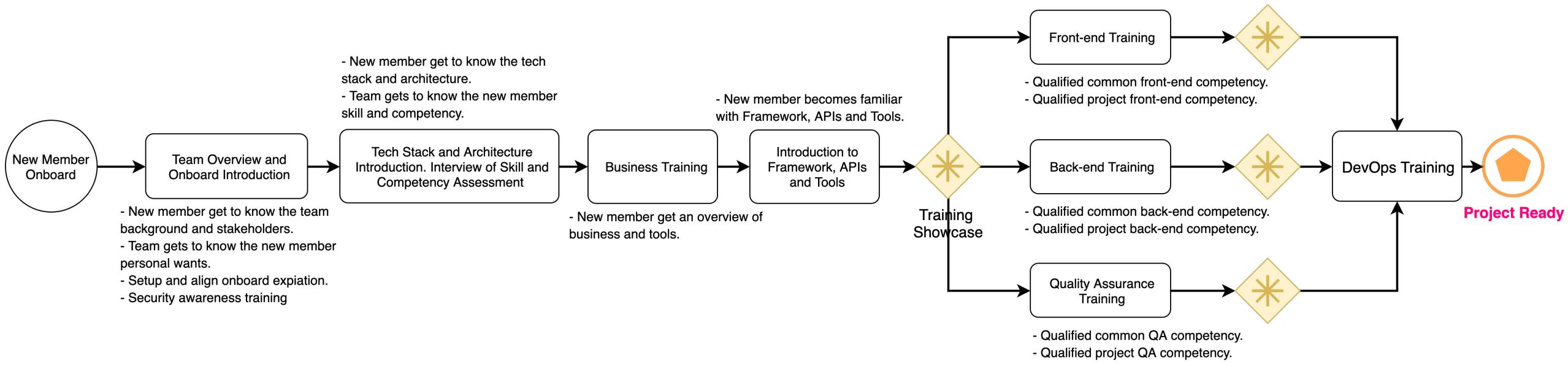
# 能力图谱



# 能力图谱

Security	FrontEnd	Back-End	QA	Enterprise	Project Business	DevOps	Programming General
General	General	Asp.Net MVC	General	System Scripts	AFFA Flow	Pipeline & Environment	Git
Identify security requirements	HTML basic	Authorization	Exploratory test	Dynamic API	Standard Process	Release Process	File Structure
Security review source code	CSS Basic	Action Result	Design testcases	Dynamic Controller	Hub Users	Pipeline	Big Method
Automated security testing cases	Javascript	Razor View	Automation Testing	RuleSet	Club Users	BuildScript	Magic Name
Automated source code security scan	1. Syntax	Routes	1. UI testing	MapView	Recovery Site	Shell	Duplicated
Security testing against common vulnerabilities	2. Unit Testing	Life Cycle	2. Functional testing	Declare	Manage Order		Super Class
Secret management	3. Design Pattern	Cookie	3. API testing	Evaluate	XOCI		Middle Person
3rd party components security check	Browser Knowledge(How Browser Works and browser devtool)	Session	4. Performance testing	Get			Extend Future
<b>Security In Project</b>		<b>.Net &amp; C# Language</b>		<b>QA In Project</b>		<b>Design Architecture</b>	
Security Policy		Async/Await	Automation(SpecFlow + Selenium + C#)	KeySkies	Object Oriented Programming		
PCI and Secure Development		Interface Inheritance	Business(stream and backup stream)	NewSales	S.O.L.I.D Principle		
Incident Response Plan		Nullable Type	Tools(Order generator)	NewSpeed	Design Pattern		
Business Continuity Plan		Extension Method		Utility	Domain Driven Design		
Client Security Expectations		Anonymous Type			API Design		
OWASP Top 10		Try/Catch Exception					
<b>FrontEnd In Project</b>		<b>Framework &amp; Tools</b>		<b>Frameworks &amp; Tools</b>		<b>Code Quality</b>	
Component Library		Team Structure					
Datastore		KVP System					
Editor Config		Splunk & Logging Viewer					
Eslint							
SPA with React Router							

# ONBOARDING 流程



# CHECKPOINTS

Step1. (1st week)	Step 2. (2nd week)	Step 3. (3rd week)	Step 4. (4th week)
<b>Baseline</b>			
Learn business by watching videos and perform running test cases  Learn security guidelines  Know Front-End / Back-End Basic knowledge.  Focus on <a href="#">Checklist For FED</a> if you will take Front-End work	Build FED/BED development environment.  <a href="#">How to setup DevBox</a>  Learn Project business by watching videos and perform running test cases  Know Project FED/BED development.	Know Framework system.  Code in System framework.  Pair work, pick up real stories.	Start to work individually.
<b>Experienced newcomer</b>			
Step1. (1st week)	Step2. (2nd week)	Step3. (3rd week)	Step4. (4th week)
Know Front-End / Back-End Basic knowledge.  Build FED/BED development environment.  <a href="#">How to setup DevBox</a>  Learn Project business by watching videos and perform running test cases  Learn security guidelines  Know Project FED/BED development.	Know Framework system.  Code in System framework.  Pair work, pick up real stories.  Start to work individually.		

TOP1

# 学习资料

Project Tech Stack & Architecture Introduction	
<b>N Framework</b>	
High Level Architecture	<a href="#">Architecture-Highlevel-Current.png</a> <a href="#">Architecture-Highlevel-ServiceLayer.png</a> <a href="#">Overview and N Framework Basic</a>
<b>NSystem &amp; Tools</b>	
Ne... installers	Goo...
Sh... environment	00... Sole Links
Si... tool	03.I...
Si... ities	05.I...
Sh... eduler	04.I...
<b>N... oots</b>	
Dy...	otRe...
Dy...	otRe...
Rul...	otRe...
Ma...	otRe...
De...	otRe...
Ge...	otRe...
Pos...	otRe...
Exc...	otRe...
Rec...	otRe...
<b>Security - Security Practice Overview</b>	
<b>Common Security</b>	

31_Knowledge Base > QA讲业务系列		
Name	Last modified	File size
Test Plan	May 15, 2019	-
Business Series - Payment	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
Business Series - ...	Mar 31, 2019	-
QA Sharing - Pipeline and FT	Mar 31, 2019	-
QA sharing - ...	Mar 31, 2019	-
QA Sharing - FT	Mar 31, 2019	-

QA讲业务系列 > Business Series		
Name	Last modified	File size
H2.mp4	Jun 12, 2019	173 MB
Se... .mp4	Feb 20, 2019	55 MB
M... .mp4	Jan 9, 2019	104 MB
Bac... .mp4	Jan 7, 2019	104 MB

31_Knowledge Ba... > 轻松学业务系...		
Name	Last modified	File size
[...]	Sep 18, 2018	-
[...]-08-29.mp3	Aug 29, 2018	8 MB

# HOMEWORK

**Frontend Workshop**

**Frontend 101**

Refer: FE-101

- Setup IDE
- Webpack
- Commands

**CSS 101**

Refer: CSS-101

- CSS Box Model
- Flex

**Javascript 101**

Refer: JS-101

- Js basic
- Js browser API & debugging

**React**

Refer: react-workshop

**Redux**

Refer: redux-workshop

- Redux basic
- Workshop: counter && todomvc

**Test**

Refer: test-workshop

- Jest
- Enzyme
- Workshop: counter

**Basic Asp.Net MVC**

Hi! This project is to quickly get familiar with Asp.Net MVC. The basic point is to get familiar Asp.Net MVC some basic use, the homework is to check your result.

**Concepts you need to know**

- Action Result
- Razor View
- Authorization
- Session
- Cookie
- Route
- Life Cycle

doc: <https://www.jianshu.com/p/5f6156cacc76>

**Basic Point**

You can search "basic points" in this project to finish it

**1. Result**

code is in the BasicASP.NETMvc.Controllers.ActionResultController  
TestCase is in the BasicASP.NETMvc.Controllers.TestsActionResultControllerTests  
Please modify code and pass TestCase  
doc: <https://www.cnblogs.com/supersnowyao/archive/2018/01/15/8287775.html>

**2.Authorization,Cookie,Session**

BasicASP.NETMvc.Controllers.AuthController.Page() will be authorized in Basic point part  
cshtml is in the BasicASP.NETMvc.Views.Auth  
code is in the BasicASP.NETMvc.Controllers.RazorViewController  
Please modify .cshtml and .cs files to achieve the desired effect  
Reference doc:  
<https://www.cnblogs.com/JoeSnail/p/0250231.html>  
<https://blog.csdn.net/slowlifes/article/details/79521680>

**3.Routes**

code is in the BasicASP.NETMvc.Controllers.RouteController  
Please modify cshtml files to achieve the desired effect

**4.Life Cycle**

doc:  
<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/lifecycle-of-an-aspnet-mvc-5-application>  
<https://www.cnblogs.com/xiao-bei/p/5165884.html>

Executable File | 101 lines (78 sloc) | 3.39 KB

```

1 using System;
2 using System.Text;
3 using Xunit;
4
5 namespace BasicCSharp
6 {
7     public class StringAndCharOperations
8     {
9         [Fact]
10        public void should_concat_string()
11        {
12            const string title = "Mr. ";
13            const string name = "Hall";
14
15            // change "default(string)" to correct value.
16            const string expectedResult = default(string);
17
18            Assert.Equal(expectedResult, (title + name));
19        }
20
21        [Fact]
22        public void should_using_stringbuilder_to_concat_string_efficiently()
23        {
24            #pragma warning disable 0219 // suppress compiler warning about unused variables
25
26            const string title = "Mr. ";
27            const string name = "Hall";
28
29            #pragma warning restore 0219
30
31            var builder = new StringBuilder();
32            // add at most 2 lines of code here concating variable "title" and "name".
33
34            Assert.Equal("Mr. Hall", builder.ToString());
35        }
36
37        [Fact]
38        public void should_create_a_new_string_for_replace_operation()
39        {
40            string originalString = "Original String";
41            string replacement = originalString.Replace("Str", "W");
42
43            // change "" in the following 2 lines to correct values.
44            const string expectedOriginalString = "";
45            const string expectedReplacement = "";
46
47            Assert.Equal(expectedOriginalString, originalString);
48            Assert.Equal(expectedReplacement, replacement);
49        }
50
51        [Fact]
52        public void should_use_string_builder_for_inplace_string_replacement()
53        {

```



# HOMEWORK

**ACTIONS**

- Clone
- Create branch
- Create pull request
- Create fork

**NAVIGATION**

- Source
- Commits
- Branches
- Pull requests
- Forks

## Homework Introduction

This homework is for practising the framework script syntax, basic asp.net mvc concept and razor view syntax. you will be required to implement a feature named "XWay".

### XWay

XWay is a third party service offered by Taxi, it will offer different types of vehicles (shuttle bus/car/minivan) delivering passengers from airport to resort or from resort to airport. You are required to implement this feature, fetch all available transfers cars using predefined search condition, display the transfers car on the page. when user make the selection, commit the user selected product order.

### Prerequisite

1. Asp.net MVC basic concept (Route, Controller, View, PartialView, HttpMethod, Layout) see the microsoft tutorial <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/>  
 2. Basic Framework script concept, read the script document first <http://developer.xway.com/documentation/> (Search evernote for framework developer account)

## Framework Introduction

```
TransferCar.cshtml" > x @using Transfers.ViewModels
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
#homework-5 bind ViewModel
# have a look at
# TODO still find a decent English resource
@model xxxModel
{
    <!--don't remove this line--&gt;
    var cars = Model.GetTransfersList().Where(c =&gt; c.Type == "Port to Dropoff");
}

#homework-6 display KVP Key:Summary.Products.CarHire
# best if you browse through a few other .cshtml files

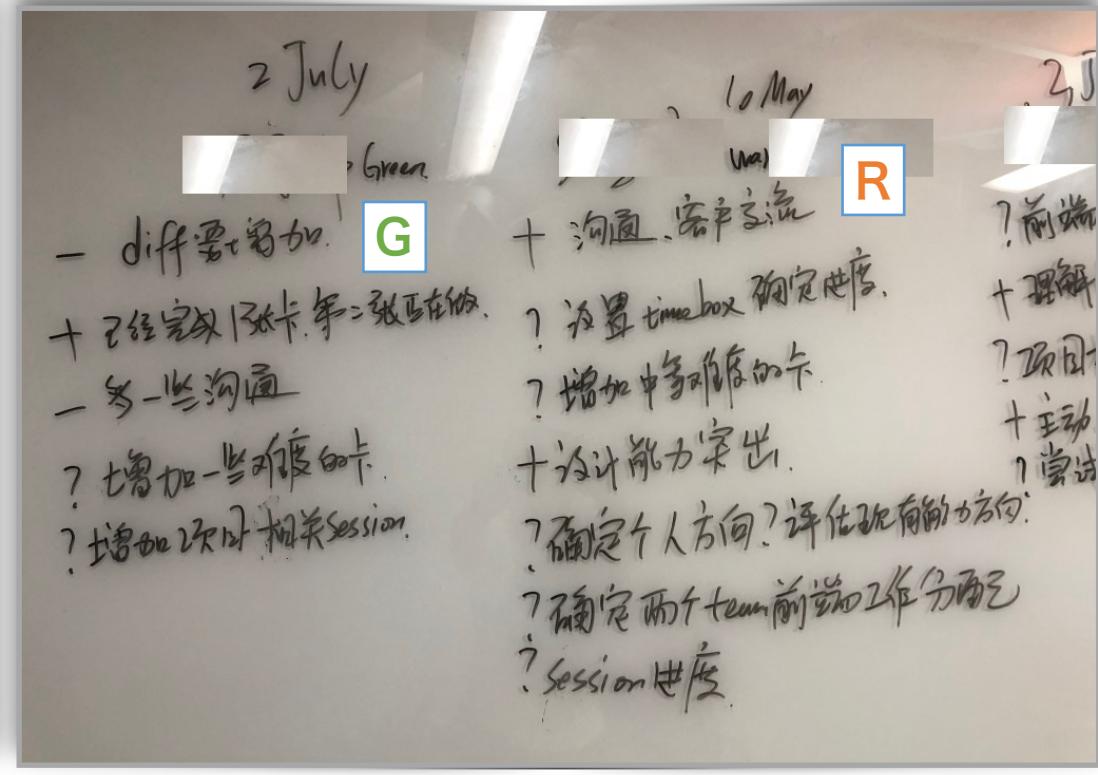
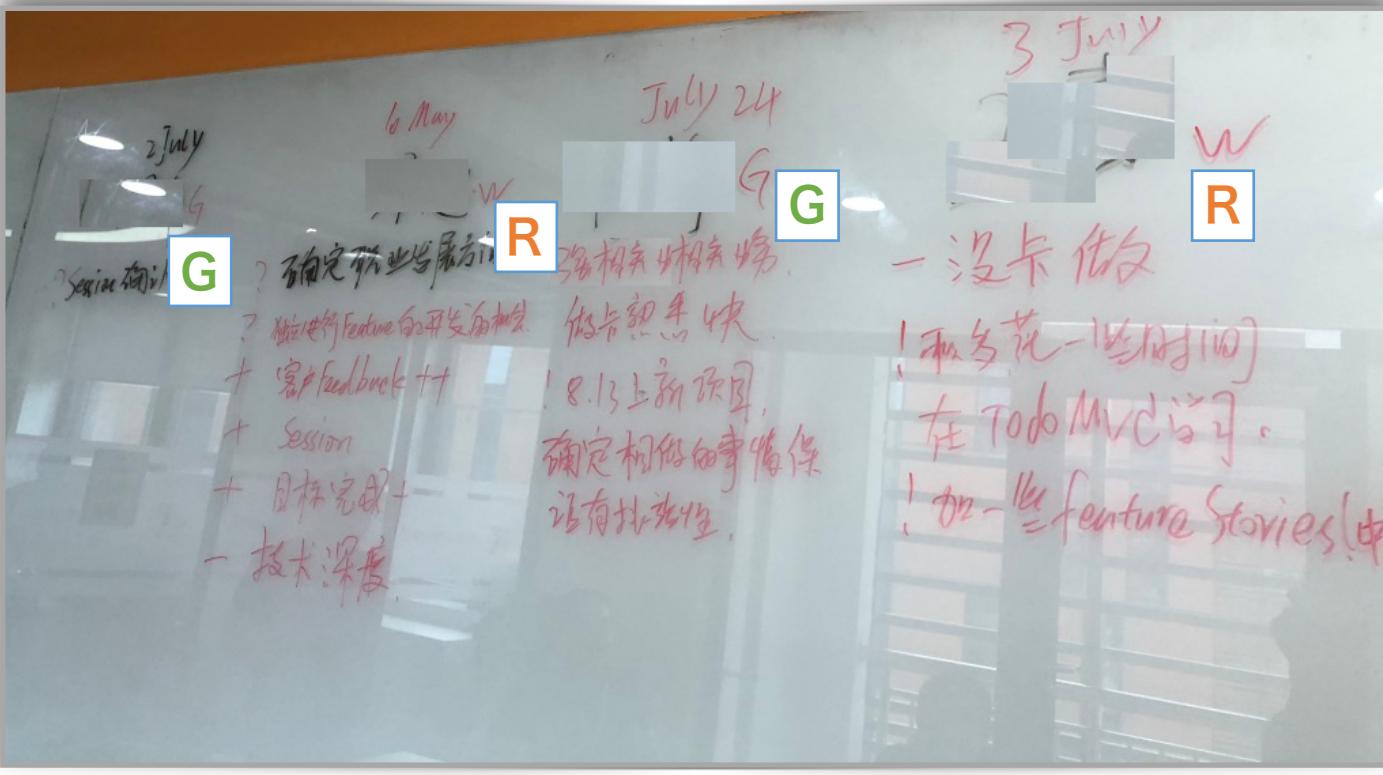
#homeworks-7 display the cars by radiobutton
# feel free to use either plain html elements or try some fancy razor stuff

181 % &lt; tobi.vogel, 43 days ago | 2 authors, 4 changes &lt; Find Results 1
SourceWebConfigurationRuleSets.cs(218): #homework-10 this is the Class
SourceWebConfigurationRuleSets.cs(921): #homework-2 The Transfer Car feature should be available after selecting a bundle.
SourceWebConfigurationRuntime.cs(592): #homework-1 add route for the transfer car feature.
SourceWebConfigurationTask.cs(6): #homework-10 implement a ruleset, when toggle is on, jump to this view, otherwise, redirect to baggage page
SourceWebConfigurationTask.cs(15): #homework-3 implement controller action
SourceWebConfigurationTask.cs(24): #homework-9 implement the controller action
SourceWebViewsTask PartialView&lt;TransferCar.cshtml&gt;():#homework-5 bind ViewModel
SourceWebViewsTask PartialView&lt;TransferCar.cshtml&gt;():#homework-6 display Products.CarHire
SourceWebViewsTask PartialView&lt;TransferCar.cshtml&gt;():#homework-7 display the cars by radiobutton
SourceWebViewsTask Test.cshtml():#homework-4 render the transfer car partial view
SourceWebViewsTask Test.cshtml():#homework-8 post the selection to controller with an html form and input submit
README.md(1):# Homework Introduction
README.md(2):# This homework is for practising the navitaire script syntax, basic asp.net mvc concept and razor view syntax, you will be required to implement a feature named "Transfer".
README.md(30):# You need to use "TransfersCarSearchViewModel" and "TransfersCarQuoteViewModel" and "TransfersCarSellViewModel" to help you finish this homework. "TransfersCarSearchViewModel" is used for
README.md(31):# You need to hack the existing bundle page redirection, to jump to the homework page
README.md(32):# You need to implement a ruleset, control the redirection, the ruleset will read a feature toggle, if the toggle is on, then jump to the homework view, otherwise, jump to baggage page
README.md(41):# git checkout -b [yourBranchName] origin/homework replace the "[yourBranchName]" with the branch name you wanted i.e. [yourname-homework]
README.md(43):# Press Ctrl+Shift+F, search "#homework" keyword in the entire solution, you will see there are quite number of "#homework-x", implement them one by one
README.md(65):# Go the baggage page, and check the shopping cart, whether there is the product you previous selected, if yes, then you are all set, and pass the homework, if not, check your implementation
files searched: 7689</pre>

```

## TOP1

## 状态看板

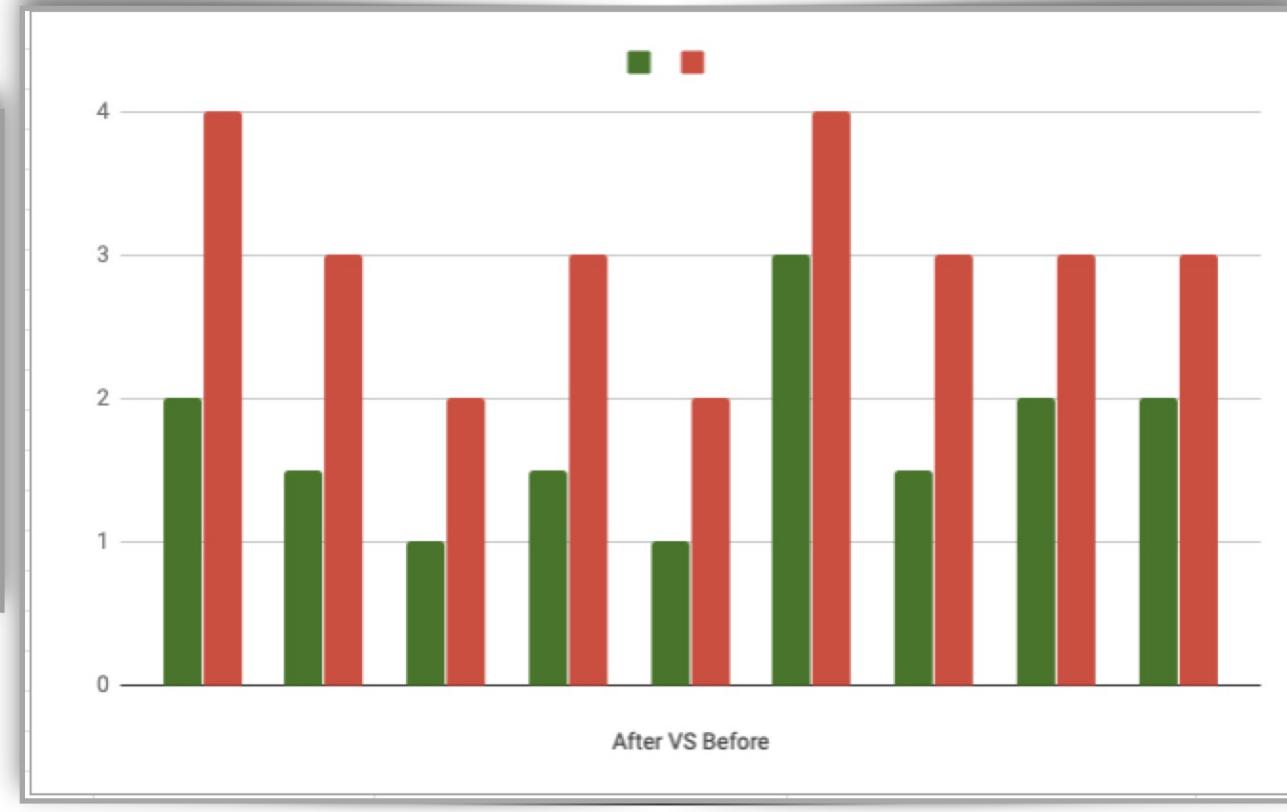


## 经验教训

- ★ 总共加入55位新成员，4位未通过Onboarding流程，被淘汰。
- ★ 新成员明确知道“Project Ready”到底需要什么，完成赋能，开始独立交付工作。
- ★ 自组织，自驱动，自迭代的Onboarding赋能过程。
- ★ 形成人员快速成长标准流程，加速新成员成长。

## 经验教训

Weekly Status	After applying onboarding process The time cost of new member to be qualified	Before applying onboarding process The time cost of new member to be qualified
ALERT	2	4
ALERT	1.5	3
GREEN	1	2
GREEN	1.5	3
GREEN	1	2
GREEN	3	4
ALERT	1.5	3
GREEN	2	3
GREEN	2	3



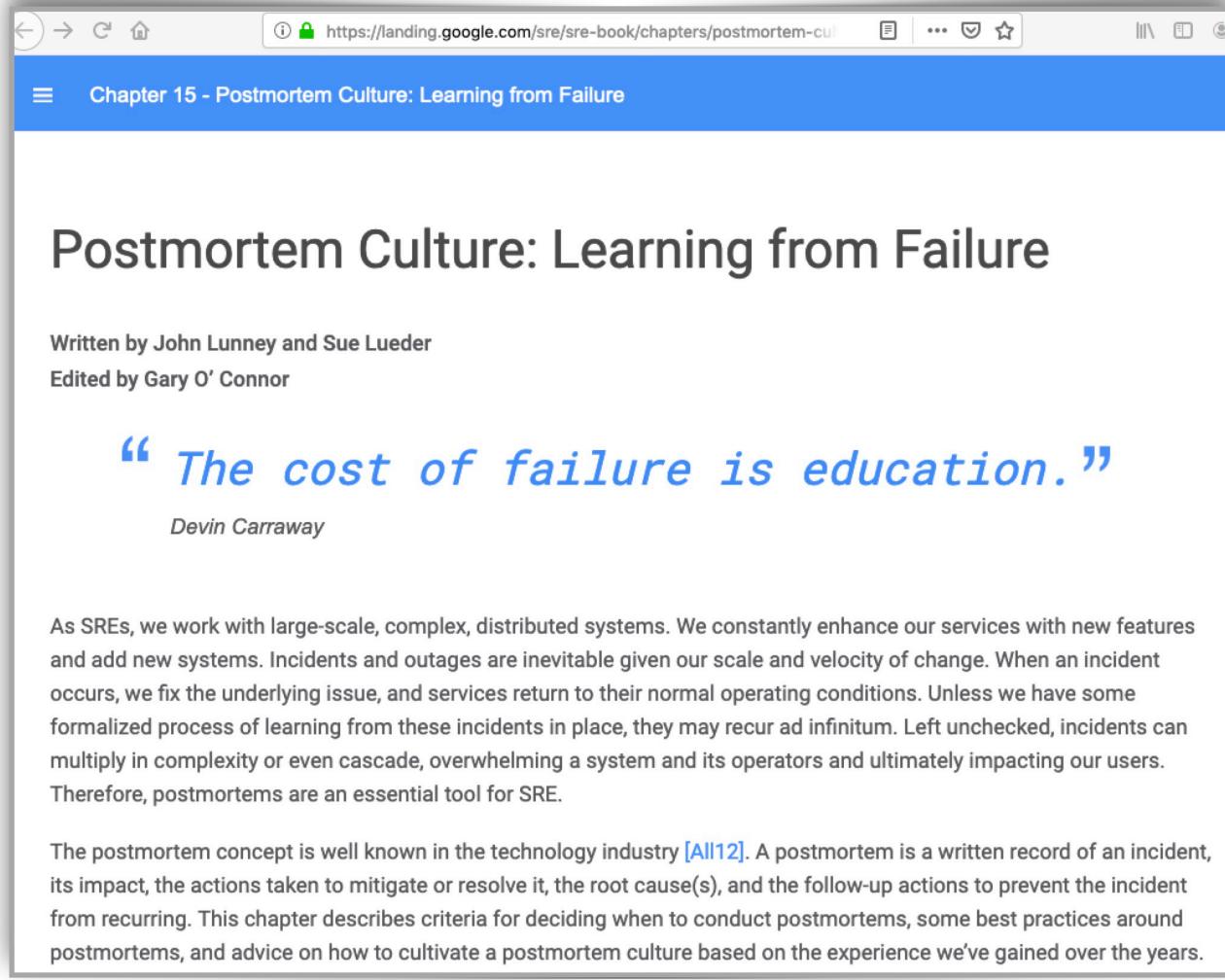
# ONBOARDING





## ★ 线上事故回顾—报忧文化

# GOODNEWS&BADNEWS



The screenshot shows a web browser window with the URL <https://landing.google.com/sre/sre-book/chapters/postmortem-culture>. The page title is "Chapter 15 - Postmortem Culture: Learning from Failure". The main heading is "Postmortem Culture: Learning from Failure". Below it, it says "Written by John Lunney and Sue Lueder" and "Edited by Gary O' Connor". A quote is displayed in blue: "*The cost of failure is education.*" attributed to Devin Caraway. The text below discusses the challenges of working with large-scale systems and the importance of postmortems. It also explains what a postmortem is and its purpose.

As SREs, we work with large-scale, complex, distributed systems. We constantly enhance our services with new features and add new systems. Incidents and outages are inevitable given our scale and velocity of change. When an incident occurs, we fix the underlying issue, and services return to their normal operating conditions. Unless we have some formalized process of learning from these incidents in place, they may recur ad infinitum. Left unchecked, incidents can multiply in complexity or even cascade, overwhelming a system and its operators and ultimately impacting our users. Therefore, postmortems are an essential tool for SRE.

The postmortem concept is well known in the technology industry [All12]. A postmortem is a written record of an incident, its impact, the actions taken to mitigate or resolve it, the root cause(s), and the follow-up actions to prevent the incident from recurring. This chapter describes criteria for deciding when to conduct postmortems, some best practices around postmortems, and advice on how to cultivate a postmortem culture based on the experience we've gained over the years.

## Google's Postmortem Philosophy

The primary goals of writing a postmortem are to ensure that the incident is documented, that all contributing root cause(s) are well understood, and, especially, that effective preventive actions are put in place to reduce the likelihood and/or impact of recurrence. A detailed survey of root-cause analysis techniques is beyond the scope of this chapter (instead, see [Roo04]); however, articles, best practices, and tools abound in the system quality domain. Our teams use a variety of techniques for root-cause analysis and choose the technique best suited to their services. Postmortems are expected after any significant undesirable event. Writing a postmortem is not punishment—it is a learning opportunity for the entire company. The postmortem process does present an inherent cost in terms of time or effort, so we are deliberate in choosing when to write one. Teams have some internal flexibility, but common postmortem triggers include:

- User-visible downtime or degradation beyond a certain threshold
- Data loss of any kind
- On-call engineer intervention (release rollback, rerouting of traffic, etc.)
- A resolution time above some threshold
- A monitoring failure (which usually implies manual incident discovery)

It is important to define postmortem criteria before an incident occurs so that everyone knows when a postmortem is necessary. In addition to these objective triggers, any stakeholder may request a postmortem for an event.

Blameless postmortems are a tenet of SRE culture. For a postmortem to be truly blameless, it must focus on identifying the contributing causes of the incident without indicting any individual or team for bad or inappropriate behavior. A blamelessly written postmortem assumes that everyone involved in an incident had good intentions and did the right thing with the information they had. If a culture of finger pointing and shaming individuals or teams for doing the "wrong" thing prevails, people will not bring issues to light for fear of punishment.

Blameless culture originated in the healthcare and avionics industries where mistakes can be fatal. These industries nurture an environment where every "mistake" is seen as an opportunity to strengthen the system. When postmortems shift from allocating blame to investigating the systematic reasons why an individual or team had incomplete or incorrect information, effective prevention plans can be put in place. You can't "fix" people, but you can fix systems and processes to better support people making the right choices when designing and maintaining complex systems.

When an outage does occur, a postmortem is not written as a formality to be forgotten. Instead the postmortem is seen by engineers as an opportunity not only to fix a weakness, but to make Google more resilient as a whole. While a blameless postmortem doesn't simply vent frustration by pointing fingers, it **should** call out where and how services can be improved. Here are two examples:

# 线上事故回顾

**Summary:** The monitoring system reported a **S1(the highest priority, need to fix it immediately)** production issue that the web server which is hosting client's web applications was constantly restarting worker processes in very high frequency in production.

**Impact:** End users weren't aware of this issue, no revenue impact.

**Root Causes:** Code refactoring changed and removed some code in method *ApplicationError* that would cause some necessary data missing when dealing with invalid url (e.g. /not-found) triggered by end users. This data missing error would cause invoking *ApplicationError* method again to deal with this latest error.

Then it turned into a dead loop situation. The application server would restart processes if the number of errors exceeds a threshold like 20.

**Trigger:** Any end users accessing an invalid URLs on the website will trigger this issue.

**Resolution:** Rolled back the removed code in *ApplicationError* method.

**Detection:** The monitoring system alerted that web processes have been restarted lots of times.

## Action Items:

Action Items	Type	Status
1. Team immediately responded to client's email to ensure communication is engaged regarding this production issue	Mitigate	Done
2. The issue was defined as S2 when we received it, according to production support process, we informed ThoughtWorks stakeholders about this issue by email		
3. A temporary hipchat group was set up involving related Devs, QAs, and client side stakeholders to keep status up to date.		

4. Team started to analyze Splunk Log, to compare the code changes between 2 releases, so to detect root cause.		
5. Suspected root cause was confirmed and issue was replicated in local environment.		
6. Fixed the issue and tested it in local successfully.		
7. Aligned go-live time with client for the fix.		
8. Ran release process again and pushed the fix on live.		
Must raise the risk of code changes to the team and organize a review meeting if feeling less confident. e.g. especially when code deleting or big change.	Prevent	Ongoing
MUST get QA involved to design ACs and test cases for any tech tickets	Prevent	Ongoing
Develop a tool to send all kinds of invalid request to system and run it before each release (open to discuss) Put this scenario into regression test suite	Prevent	Ongoing

## Lessons Learned

### What went well

- The team was able to respond to the issue in time and got it fixed with stakeholders together proactively and collaboratively.
- A process of how to handle urgent issues has been established already and all team members understand what to do to mitigate the impact.

- All information has been recorded for post-incident analysis and improvement
- The newly well-configured logging system, **Splunk**, helped to debug and find the root cause for us while fixing this issue in 24 hours.

### What went wrong

- No tests to cover this edge case.
- Pull Request is an optional practice for teams back then, therefore we probably missed the opportunity to figure it out earlier.
- Tech cards/tickets don't get QA involved to verify the correctness.

### What we got lucky

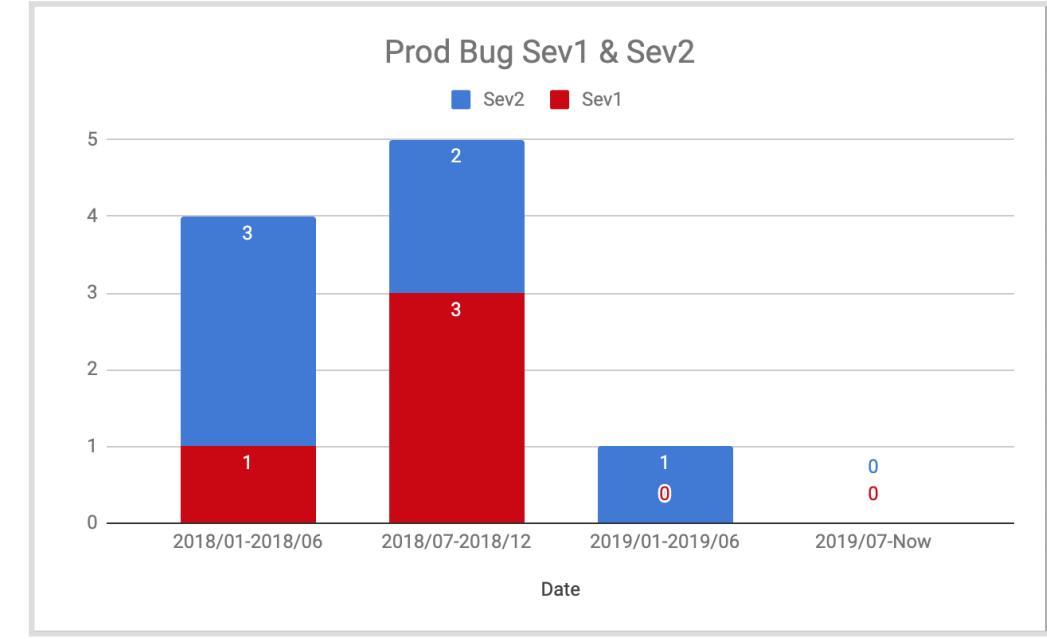
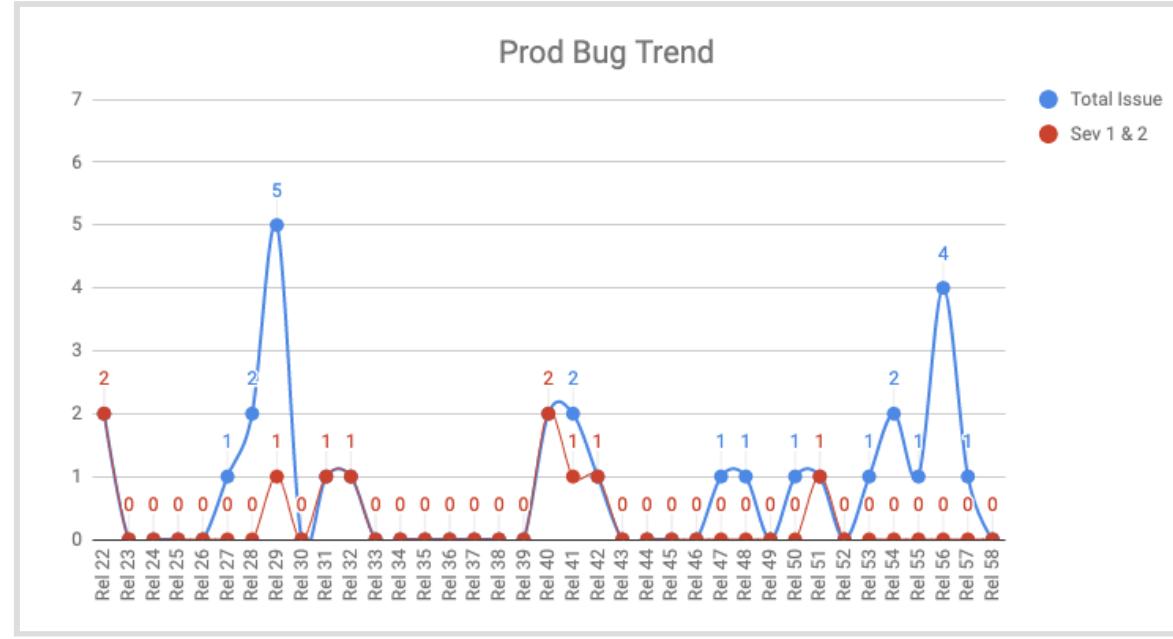
- The reason why end users weren't aware of this issue and there is no revenue loss is because the dedicated application platform hosted and operated by our customer and their third-party vendor provides session storage and application cluster.

### Timeline

- 2018-06-04, Code change, because of the task XXX-1: Error handling.
- 2018-06-21, Release 101 with the task XXX-1, Go live.
- 2018-06-26 10:00 AM, Delivery team received an issue report.
- 2018-06-26 08:00 PM, Development team found the root cause.
- 2018-06-26 10:00 PM, Development team reproduced in local environment.
- 2018-06-26 11:30 PM, Development team fixed and tested.
- 2018-06-27 10:30 AM, New released 101.1 with fix, Go live.

## 经验总结

- ★ Lessons and Learned、Timeline、增强Log、后续Actions、实施效果。
- ★ 提升功能测试覆盖率，增强质量保障。
- ★ One Team 线上事故实战经验分享，增进团队融合。



## 业界实践

Aspect of Software Delivery Performance*	Elite	High	Medium	Low
<b>Deployment frequency</b> For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per day and once per week	Between once per week and once per month	Between once per month and once every six months
<b>Lead time for changes</b> For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one day	Between one day and one week	Between one week and one month	Between one month and six months
<b>Time to restore service</b> For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day <sup>a</sup>	Less than one day <sup>a</sup>	Between one week and one month
<b>Change failure rate</b> For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0-15% <sup>b,c</sup>	0-15% <sup>b,d</sup>	0-15% <sup>c,d</sup>	46-60%



**ACCELERATE**  
**State of DevOps**  
**2019**

The industry continues to improve, particularly among the elite performers. The proportion of our highest performers has almost tripled, now comprising 20% of all teams. This shows that excellence is possible—those that execute on key capabilities see twice as much growth.

**KEY FINDINGS**

Delivering software quickly, reliably, and safely is at the heart of technology transformation and organizational performance. We see continued evidence that software speed, stability, and availability contribute to organizational performance (including profitability, productivity, and customer satisfaction). Our highest performers are twice as likely to meet or exceed their organizational performance goals.

- 1** The best strategies for scaling DevOps in organizations focus on structural solutions that build community. High performers favor strategies that create community structures at both low and high levels in the organization, including Communities of Practice and supported Proofs of Concept, likely making them more sustainable and resilient to reorgs and product changes.
- 2** Cloud continues to be a differentiator for elite performers and drives high performance. The use of cloud—as defined by NIST Special Publication 800-145—is predictive of software delivery performance and availability. The highest performing teams were 24 times more likely than low performers to execute on all five capabilities of cloud computing.
- 3** Productivity can drive improvements in work/life balance and reductions in burnout, and organizations can make smart investments to support it. To support productivity, organizations can foster a culture of psychological safety and make smart investments in tooling, information search, and reducing technical debt through flexible, extensible, and viewable systems.
- 4** There's a right way to handle the change approval process, and it leads to improvements in speed and stability and reductions in burnout. Heavyweight change approval processes, such as change approval boards, negatively impact speed and stability. In contrast, having a clearly understood process for changes drives speed and stability, as well as reductions in burnout.



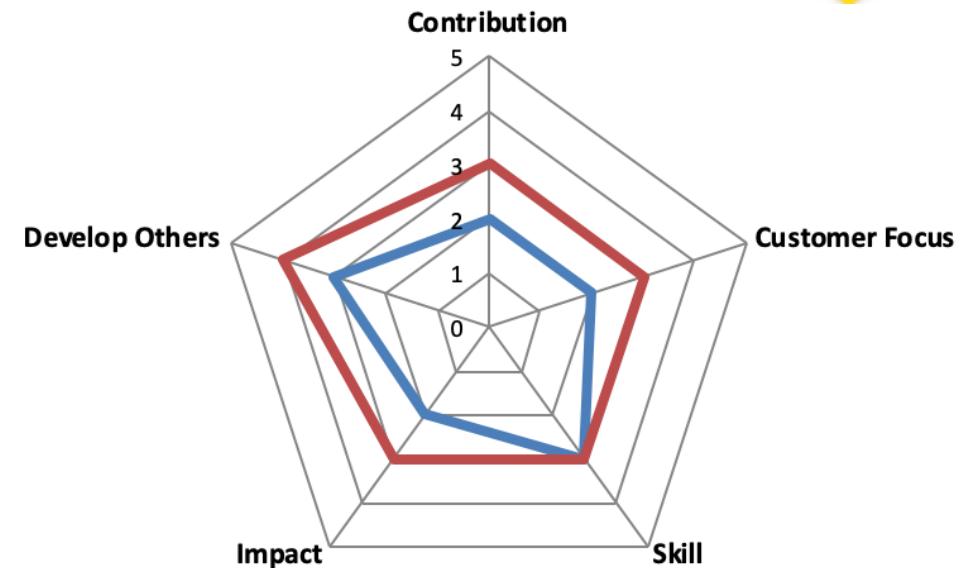
## ★ 人才梯队构建

## 人才梯队构建

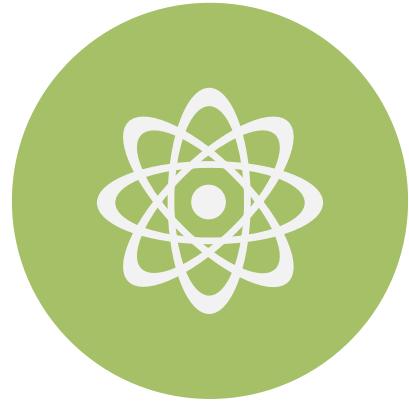
- ★ 可视化人才梯队看板。PM/TL、SecondTire、KeyContributor、Others、Risk
- ★ 每季度基于Facts的Review，进行梯队调整。
- ★ 梳理人员提升Actions、帮助团队成员提升。

## 人才看板

Measures	Current State	Quarterly Target
<b>Contribution</b>	2	3
<b>Customer Focus</b>	2	3
<b>Skill</b>	3	3
<b>Impact</b>	2	3
<b>Develop Others</b>	3	4
	12	16



Team	Tech Lead/Team Lead/PM						Second Tier						Potential/Key Contributor					
	Well Done		Medium		Medium Rare		Well Done		Medium		Medium Rare		Well Done		Medium		Medium Rare	
	Senior	Consultant	Senior	Consultant	Senior	Consultant	Senior	Consultant	Senior	Consultant	Senior	Consultant	Senior	Consultant	Senior	Consultant	Senior	Consultant
Team A			刘xx	李xx												冯x		
Subtotal	2						0						1					
Team B					张xx						邓x'x				彭xx		张xx	
Subtotal	1						1						2					
Team C									苏x						王x		姚xx	
Subtotal	0						1						3					



★ 社区&自学习团队

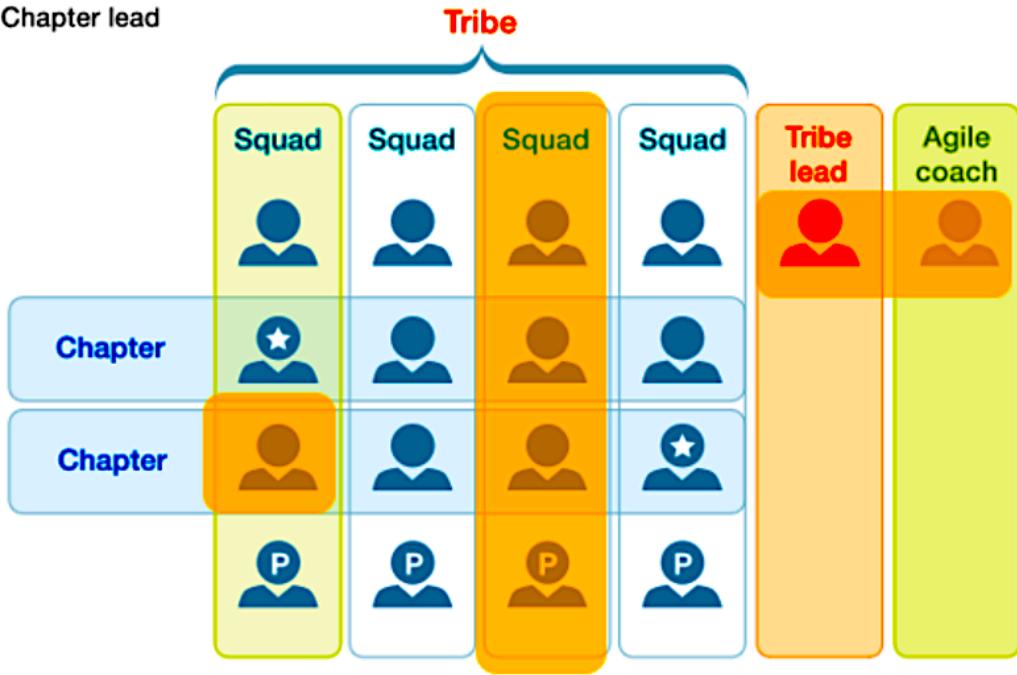
**TOP1** 

## 社区&自学习团队

- ★ 内部形成技术Chapter, 构建规律的技术分享活动。
- ★ 外部打开眼界, 关注行业, 融入社区, 从参与者到讲师、激活团队氛围、形成良性循环。

# 技术CHAPTER

P Product owner  
★ Chapter lead



**Tribe** – a collection of squads all working in a related area

**Delta Capita** can provide the **Tribe leads** and **Agile coaches** that will enable and provide the conditions for success, driving the creation of a culture of continuous improvement, continuous delivery, and fostering strong and effective collaboration across the tribe

**Squad** – the basic unit of development

**Delta Capita** can provide complete Squads that fit seamlessly into your existing Tribes, employing lean start-up principles, minimising waste, delivering minimum viable product, and maximising delivery of value.

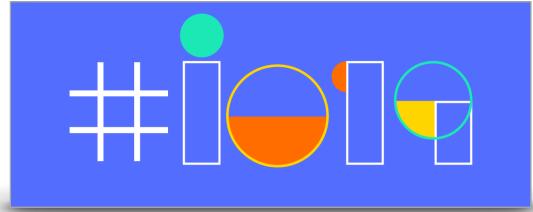
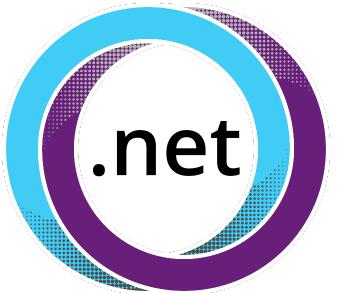
**Delta Capita** can provide the **Product Owner**, the **proxy Product Owner** or the **Business Analyst** as appropriate.

**Chapter** – a specific discipline or area of expertise

**Delta Capita** can provide expertise across any specific discipline. For example, test and test automation expertise can be provided as a testing Chapter that slides into your existing tribes and squads as needed.

**Delta Capita** can provide the **Chapter Lead** and the **thought leadership** within any particular chapter.

技术社区





**TOP1** ☀️

## 回顾—投入产出

- ★ 不断完善的Onboarding流程，顺利完成了团队的高速、高质量扩张，避免了风险，提升了效率。
- ★ 总计加入55位新成员，4位未通过Onboarding流程，被淘汰。
- ★ 从1对1的老带新的方式，演变为自组织自驱动体系，大大节约了时间成本。
- ★ 构建人才梯队，防止知识稀释，并没有因为团队快速扩张，而产生额外的线上事故。

## 回顾一启示

- ★ 如何快速的完成新成员的能力构建？ — CraftSkill Map, Onboarding流程，状态看板。
- ★ 如何系统化的进行人才梯队构建，防止知识稀释？ — 人才看板，报忧文化。
- ★ 如何激活团队氛围，形成良性的知识分享土壤。 — 内部Chapter赋能，外部打开眼界，加入社区。



[sczhang@thoughtworks.com](mailto:sczhang@thoughtworks.com)

THANKS Q&A



微信官方公众号：壹佰案例  
关注查看更多精彩实践案例

# 100

TOP 100 CASE STUDIES OF THE YEAR

全球软件案例研究峰会