**关于Peter及其项目组的若干项问题概述**

Calvin, Si

**一、从整个research项目的角度看:**

自我参与research相关开发一年的时段内，对于项目开发，部署，测试等方面均有参与或侧面观察：

**1. 关于开发与部署所存在的问题：**

**(1) 前端开发**

项目进行至今，前端工程的开发（insight 3.0）一直处于和Matt并行合作的模式。虽然这个insight平台声称为3.0的版本，但其本质上保留了2.0版本绝大多数的特性，Peter对于此事的态度极为保守与妥协，拒绝交涉，对于有积极作用的大规模重构表现出明显的抗拒，这导致3.0版本的工程体量自一开始便极为冗杂繁重，这给前端开发人员带来了极大的开发难度，也限制了新技术新框架的使用，其中具体的影响因素包括但不限于：

* 工程的build的速度极慢，甚至导致在较高环境下部署前端工程时，服务器会承受巨大压力。一段时期内仅通过调节服务器配置增大内存等方式暂时缓解，后续在彭晓的牵头下大规模调整了前端部署的流程，才实现目前的稳定部署。
* Insight 2.0 旧有代码有诸多bug和不稳定性，其中还有大量过度抽象的情况，这导致了即使在视图中极为简单的功能，它的背后依然有着一套非常复杂的逻辑关系。然而这种复杂逻辑很多是不必要的。但在旧有框架下，这种复杂逻辑是无法摆脱的。
* Insight 2.0由于开发时间较早，其中包含了大量应被取代的旧语法或旧的组件与框架，但其拥有着错综复杂的依赖关系，将旧的依赖包进行替换是一件成本极高的事。
* Insight 2.0由于是Matt主力开发，几乎是独立完成，因而工程中带有极其浓重的个人色彩，能感受到自己做主开发时的任性而为。有时为了实现相同功能或效果，会引用多种类库，这导致我们开发时为了保证依赖的最简化额外付出了巨额代价。

而针对以上种种情况，在我们多次向Peter交代以上背景的情况下，他依旧无法理解其中的痛点，在安排前端的工作进度时仍然不能理清主次，多次的瞎指挥、乱测试导致真正急需解决的（不需要demo的 但决定了稳定性和开发速度的方面）被置之不理，而只急于把能够demo的部分赶制出来，客串测试人员也无非鼠标点一点，完全看不出关键的问题所在（比如很细节的显示的值与格式，交互逻辑），当然这也不出意料地导致bug不断，经常出现部署失败或者临时发现样式错误、功能失灵等问题。

**(2) 后端开发:**

至于后端开发，由于前期业务方面梳理不清醒，数据整合不符合逻辑，导致一直主张的微服务架构常常走向错误的方向。我们经常发现，关于业务或数据的文档虽然看起来书写得段落清晰，内容很多，但很多是基于了错误的前提得出了错误的结论，这也导致所开发的后端服务经历了数次大型重构，常有前脚刚做完，后脚马上推翻重来的情况。

而Peter在整个过程中除了push进度以外，几乎没有作出任何有正面影响的行为，完全不关注工程稳定性和代码质量。在其客串测试人员的时候，对于相应的接口测试也仍然是随手点一点，没有对细节（包括字段是否完整，取值是否正确）的把控能力。

而且在后端服务的“拨乱反正”的过程中，Peter所表现的行为让人怀疑他可能真的完全不理解整个架构设计与实现模式，常常挂在嘴边的“微服务”也无非是个名词而已，它为了解决什么问题，它的特性与思路几乎一概不知或者只是浮于表面。很多真正有意义的建议和意见都是由彭晓和Eagle提出，而这种语境中，Peter常常以“对对对，我也是这么想的”（然而他自己几乎从来没有自主提出过任何正确的言论）或者“这个我们再讨论一下”（少数在彭晓等人的坚持下得以实施，其他的建议在之后基本石沉大海）进行搪塞。

**(3) 项目部署：**

我个人没有直接参与部署的流程，但据我了解，Peter忽视不同环境（dit sit uat等）各自需要做的工作（测试的侧重点不同），为了部署而部署，没有进行完整的调试与测试便急于部署到更高环境，导致无论sit还是uat部署都从来没有真正地按照正常流程和节奏进行过。而Peter对此事表示满意，甚至美其名曰“我们甚至达到了网络公司的部署频率”。

**三 组织管理方面**

**1. 组织运行模式与Peter所扮演的角色：**

Peter项目组一直以来以一种类似“君主集中制”的模式存在，大搞一言堂（btw：我大胆估计 他似乎非常喜欢开会时有一大群人所带给他的虚荣心的满足感，纯主观臆测），喜欢直接管理到个人，经常单独pin每个人去讨论他根本就是外行的技术细节，导致开发人员正常的工作节奏被打断打乱。由于在之前的某次retro会议里面，我提出了“应该避免高频的低效沟通，Peter不应直接干涉到开发”，这种情况才得以缓解，然而据了解有若干人仍在遭受此种摧残。

**2. 一个完整的需求的提出与实现：**

这个项目组拥有BA属实没有几天，在这之前其实开发人员在自己扮演BA，去找“提出需求的人”即Alex拿业务逻辑（所谓业务逻辑经常包含一些模棱两可的东西，可能过几天就换了一套说法，可信度很不稳定），这个业务逻辑的可信度 稳定性一直存疑。

前端或者后端开发基本可以用四个字概括：“开始干吧！”，在风险，effort无法准确估计，业务逻辑没有完全理清，数据库高频重构，以上种种情况下，开发人员就基于不那么正确的前提，痛苦地朝着未知的方向前进… …

**3.分工与协作：**

引用言论如下：

*Peter：“我觉得我们这支队伍好像在一家创业公司工作一样”—— 2020年中下旬*

*David（大概是）：“We're going to build this team into a special force” —— 最近*

以上言论算是管中窥豹，这种言论所概括出来的“全员全栈”模式充满蛊惑性。表面上看，大多数开发人员都在一个业务需求下接手了从前端到后端的完整开发链条，这似乎是一个很先进的运作模式，但其实这是因为Peter不懂如何拆分业务，不懂如何实现每个人的分工合作造成的后果。但正因表面上运行良好，也恰恰掩盖了Peter在管理人员与安排分工上的无能。我的理由有如下几点：

(1) 虽然每个开发都至少负责一个业务点的整理及其对应的整个前后端开发的链路，但是业务需求却完全出自Alex一个人，这导致全员依赖在同一个人身上，这有两点隐患：首先一对多的沟通无法保证效率和同个问题的答案的一致性（因为开发人员基本不会互相核对关于同一个疑问 在Alex那得到的结论是否一致 实际上确实经常会不一致），其次，我们无法保证Alex一人靠回忆和经验，提供给我们的信息是否正确，而我们也没有渠道进行核查，只有真正开发完，出现了问题才追悔莫及的份（而且这个情况层出不穷，经常Alex提供的信息前后矛盾 而且多是基于2.0旧版本的思路 对于微服务实现的阻力也很大）

(2) 虽然每个人以全栈开发的形式在工作，但是各自负责的部分基本在业务逻辑上是隔离开的，互相之间的依赖和沟通极少，基本名义上是团队合作，但实际上是各自为战。这种模式平时看不出问题，但开发过程中，每个开发人员只管自己一块，无暇顾及别人正在进行的部分，既无管控又无测试，一旦进行功能联调就会发现各种问题存在。Peter对于此事完全看不出要害，至今仍是兵来将挡水来土掩的土办法，这只会把坑越挖越深。

(3) 对于这种野蛮生长下成长的团队，只是因为成员个人能力比较强（btw:毫不谦虚 Jerry全组敢跟富达任何开发团队battle）才弥补了这种无能的管理，然而这个表面上看来还不错的效果却逐渐滋生了Peter日益膨胀的虚荣和野心（btw：个人推断 Peter最近的野心已经膨胀到与他能力严重不匹配的地步了），会让他自以为领导有方，未来或许会保持现状或许情况变得更糟糕。如此恶性循环下去，我个人对Peter组内成员的未来持悲观态度，所谓一将功成万骨枯，Peter正在这条路上越走越远。

**三、总结 我个人对于Peter的判断与评价：**

1. **没有做出正确判断和正确决定的能力**，大多数情况都是被上层牵着鼻子走以及随声附和，专业知识欠缺，知识碎片化，没有体系，在与其沟通中经常能够发现，其只懂得一些名词和概念，而概念与概念之间的关系网络完全不具备

（包括技术架构的理解 技术框架的选择 业务架构和业务逻辑的梳理）

2. **只走上层策略**。谄媚上司 无条件服从上层领导的任务 不考虑风险和实施复杂度，不懂得如何分配优先级 ，会在其他成员完全不知情的情况下以肯定的态度接受任务。

(1) 经常在部署或demo前临时加入非计划内的内容，影响正常开发进度，而所加入的内容经常来自于与David的某次聊天（甚至是闲聊），而不是经过正常的沟通-评估-计划与设计-实施的流程，这也导致临时加入的功能经常有大量bug，而当前sprint内正常的进度也会受到影响

(2) 会在不完全理解（主要是David或Scottie）提到的新的项目分支或者新功能，没有评估风险和effort的情况下，盲目接受。导致人员不足的处境下，无人能够接手，只能通过让一部分人分出额外精力，强行产出，质量无法保证，而且经常由于仓促之下和业务不清导致高频率返工。

3. 整个团队运行的名义上的scrum实际上只是类似君主集中制的一言堂的模式，很多有建设性的建议或被否决或被忽视或流于表面地实施。（其中包括近期提出的要把高速产出的模式切换到降低速度提高质量的模式，实际上“只push进度而不管质量”的情况没有任何改善，甚至某些方面变得更加严重）。

4. **缺乏对自己技术知识储备何其匮乏的自知之明**，经常插手开发人员的工作细节，然而他虽自称“技术出身”却严重缺乏技术意识和能力，导致开发人员经常花费无意义的精力与其沟通技术细节，最终结果是其几乎完全没有接受吸收。

对于不同环境的部署尤其混乱，完全忽视不同环境各自的含义和对应不同环境应该要完成的工作。（实际上经常dev和sit一起上，在完全没测试或者鼠标随意点几下就算测完的情况下便上uat环境，而且上线频率极高，问题也极多。peter本人对此表示自豪 因为说这几乎达到了互联网公司的部署频率 可见其对于部署相关的知识匮乏的程度）

6. **管理上无能**，其管理能力的缺失在项目逐渐扩大，人员逐渐增多的过程中被逐渐扩大，导致整个团队的分工合作处于混乱的状态（各自为战 互相不了解状态 负责业务的人每天被会议淹没 但对于开发来说没有获得什么有意义的信息）虽然近期将scrum拆分成两份，后续会不会出现其他问题仍有待观察，个人持悲观态度（因为这种拆分只是治标不治本，本质上的信息流不清楚 无效沟通太多而有效沟通太少 业务梳理没有细节和管理只走形式 纯靠个人能力支撑项目前进的模式 崩溃只是时间问题）

7 **性格品质不敢恭维**，员工各有自己的抱怨点，我大概列出几点（仅为我个人主观感受）：

*心胸狭隘*，对于提出质疑和意见经常抱有警惕情绪，对团队和项目改善的建议常被怀疑是针对其个人。热衷调查组员心理动向，常会在被提出challenge之后私下找对方“交流”以探口风。

*格局狭小* 经常只顾眼前利益，对于长远的发展没有太高兴趣度。对于技术上的改善（代码标准 框架调整）持有过度的保守态度，经常在项目因为这些问题实在难以进行下去的情况下才意识到问题，后果就是要暂停开发，集中重构，导致项目进度停滞。

*以模棱两可的话作为邀功和推卸责任的工具*。 push开发人员没日没夜的堆代码并不能很好的提高效率，只会导致bug变多且后续维护成本极高。将短暂的所谓的高效率的成果几乎都归功于个人，以为是自己安排有方。出现问题便找替罪羊，让替罪羊连夜修改，而其自己则早早撇清关系。

**四、书写本文的目的和期望：**

首先，据我了解，关于Jerry全组的组织关系会有一定的变动（据悉 将要report to Peter），对此我个人持不接受的态度，理由便是以上所述。我们当初作为Innovation Team开发了很多优秀的项目。然而现在由此培养出来的高效与默契将要面对被Peter消耗殆尽的危险，个人无法接受这种落差。放眼全公司，我相信我们组有能力接手并出色完成任何项目。而且我相信全组持有同种态度的人绝不止我一人，我想后续逐个问询便可以得出结论。

我必须承认，以上全文所述不可避免带有个人情绪和主观臆断，但我依旧尽量标示出哪些是事实，哪些是“我认为”的。即使文章中可能依旧存在一些经不得推敲的描述，但我仍然希望能通过这些文字让看到这些内容的人知道我的期望是有依据和以此得出的推断的，而并非因为某种个人恩怨或者不理智的抵抗情绪。不管未来结果怎样，我希望能发声便发声，把真实的信息沟通给你们，希望以上所述能够给未来的任何结果以参考的价值。本文所述内容我全部负责，愿意接受任何核查和对质。

表达本身即有意义，表达及抵达。感谢！

附件：

借此机会，附上我对于Peter与Jerry在多种维度的对比表格，也算是顺带给Jerry的一点我自己的feedback。

|  |  |  |
| --- | --- | --- |
| **对比维度** | **Peter** | **Jerry** |
| **管理团队** | 粗暴管理 野蛮生长 忽视个人之间的区别但极关注心理动向 经常私自进行职场PUA | 因项目因人制宜 合理分配任务  也会为了拓宽每个人的技能而有意进行技术方向的轮换 |
| **项目管理** | 上层路线，只谄媚上层，无条件服从上层一切指示，不考虑开发团队的负担  不具备预估任务量，安排进度，规划优先级的能力  忽视客观存在的种种问题，甚至是决定某个技术方向能否继续走下去的严重问题，只关注time line  没有清楚的业务逻辑，没有准确的交互细节，便着急实施，导致大量返工  与Global的所谓合作更多是一种妥协 | 重视业务需求的梳理，将细节了解透彻，输出成清晰的文档以供随时查阅  对于开发周期有准确的预估，会事先与开发人员充分沟通，了解大致的技术框架和可能出现的难点痛点并早做准备  能够合理分配优先级，会与需求方和开发双向沟通，保证能够高效开发，快速交付给用户相对满意的一系列版本 |
| **技术能力** | 自称技术出身实际上丝毫没有技术sense，  无论技术会议还是平时交谈，涉及到技术的点基本说不出什么所以然或者人云亦云。 | 真正的技术出身永远谦虚低调，而且对于技术问题涉猎足够广泛，看问题常能看到关键的痛点 |
| **格局** | 只追求眼前利益，也许除了实现自己的野心，其他的长远规划并没有建立，上层说什么是什么。 | 清楚作为一家帮助客户长期管理资产，平衡收益和风险的公司的一员，目光会放得更加长远，具有一定risk control的能力，会为了长远着想，即使是牺牲自己的短期利益，也要顾全大局 |
| **沟通模式** | 交流中能明显的感受到不坦诚，带着目的交谈，给人一种随时要套路我们的不适感觉  不能掌握聊天的度 经常有意无意妄谈他人私事或者轻度的人身攻击  经常安排一系列冗长的会议，形式大于实际作用（近期有所改善） | 真正的听取别人的意见，即使是challenge也会认真听取，并落实于实际行动  坦诚 轻松 不搞官僚主义，平等交流  按需安排会议 追求效率不求频率，很多情况几分钟就能聊完，动态安排即可 |
| **性格品质** | 智谋不足而心胸狭窄  没有做出正确判断和正确决定的魄力和能力  经常导致全队弥漫着微妙的压抑气氛 | 大体上相当佛系，与世无争。但在工作中总能全力以赴，能够可靠地支持所有组员  关爱组员，既能保证工作时间高效运转，其余时间也能保持气氛轻松活跃，大家都可畅所欲言。 |