第一次到俄罗斯

当说到要做个演讲

他让我想到

一件在我们两个社会几十年前人尽皆知的事情

比如在1957年

当苏联成功发射了斯普特尼克

世界上第一个绕地球公转的人造卫星

在美国大家都吓傻了

他们不喜欢在太空这方面中落后

然后他们说

天啊我没得赶上

我们得在90天里发颗卫星上轨道

当然这听起来很疯狂

然后我们的第一次试射就炸在了发射台上

但是第二次试射成功了

那是1958年1月31号

差不多是苏联成功的120天以后

在现代的标准下还挺快的

然后在1961年4月6号

尤里·加加林成为了第一个

环绕地球的人类

又一次，这又成了大新闻

所有人都对此留下了深刻的印象

然后美国那边人们都很沮丧

因为又一次，我们落后了

太空竞赛也初见雏形

所以我们该怎么办

在1961年五月

尤里加加林上天一个月以后

我们那是的总统肯尼迪

对议会发表了一个演讲说

嘿，如果我们要赶上

不永远落后我们得做点大的

投入好多资金、好多资源

我们要去月球

这是个挺疯狂的想法

然后在1962年他在这个精彩绝伦的

公共演讲里又重申了一遍

我们要在1970年之前登月

这就完全是痴人梦呓

因为我从来没有做过任何类似的事情

但是看吧最终我们做成了

阿波罗11号在1969年7月16号发射了

刚刚好在1970年之前

就在今年（2019）有一个极好的纪录片

关于整个阿波罗11号任务的

他全都是采用的被藏在箱子里柜子里的

原来NASA在任务期间录下的素材

把素材重制了

然后他们做了个某种意义上的再创作

来再现整个事件的全程

我现在要放一小段电影的节选

用来让你们对整个任务有一个尺度上的概念

这很大，很疯狂

我们从一无所有到这些只用了差不多12年

在斯普特尼克升空之前美国甚至还没有一个太空计划

到最后，我们有了所有这些东西

然后当然在这之后我们继续去研究太空

我们做了航天飞机

这看起来酷疯了

像个从科幻片里跑出来的飞船

chua就飞了然后又着陆了。绝了。

问题是绝大部分图里这些都不能着陆

所以因此发射这个依然很贵

然后它也很不可靠

有人在几个不同的任务上

在这上面失去了生命

所以我们决定不用它了

在这之后我们要想送人上轨道

我们得靠联盟号搭把手

然后自此我们的太空任务一落千丈

所以如果你跟像我这样的人聊天

在大概2002年到2005年

我们都有这么个态度说

“这不耻辱吗？”

美利坚曾经在太空做这些牛逼疯了的事

但是现在我们就不太真的做什么事

哪个我们畅想的“科幻未来”不太会发生了

我们也不见任何的改观

但是我又能怎么办呢？

摊手.jpg

这就是绝大部分人的态度

但毕竟不是所有人

在某些时候某些人

通过网站赚了点钱

然后说：

“嘿，我想做点什么

尽管没什么火箭上的经验

我会开个公司去发火箭

去做那些我们没做过的大事”

这是个视频的节选

关于他做那些的原因

然后我想到了跨星球的物种和在宇宙中繁衍的文明

这不是必然发生的

这非常重要

这不是必然发生的

可再生能源的未来很大程度上是必然发生的

但是成为能够在宇宙间飞行的文明

显然不是必然要发生的未来

如果你看一下太空技术的发展

在1969年我们能把人送上月球

1969年

然后我们有了航天飞机

航天飞机只能把人送上近地轨道

然后航天飞机退役了

然后美国没办法把人送上轨道

这就是趋势

竖直下到一无所有

这不...有时候人们会有错误的理解

觉得技术肯定是自动提升的

他并不会自动的提升

只有很多人投入大量精力努力去改善的时候才有可能

我想实际上，技术很可能自动倒退

去看看古埃及的伟大文明

他们做了金字塔

他们现在忘了怎么做了

然后罗马人

他们建造了难以置信的沟渠系统

他们也忘了怎么造了

av17486506

他的想法还挺成功的

今天我们能让火箭着陆

我们很严肃的要在2024年再登月

我们拭目以待

但是真正去严肃关注它就是个很好的事了

基于我们之前是个什么状况

所以Elon说了几个古代的伟大成就

然后到今天被遗忘了

我还想再补充几个

来重申他的观点

“科技会自动倒退”

你看见的这个东西

是莱氏杯（公元300年）

这是一个可以追溯到在公元380年古罗马的文物

这是玻璃做的

做成他的玻璃是人类已知最早的纳米材料

玻璃杯的颜色会因为你看他的方式变化

比如光源在不同的地方

如果你从正面看他

光源和你同一个方向你看杯子的反光

这高脚杯就是绿色的

但如果光线穿杯而过

高脚杯就是红色的

他们在公元300年就有这个了

然后罗马帝国覆灭

这个知识就失传了

基本上永远失传了（笑）

这东西的原理基本上是

我们在1990年左右研究出来了

这里面的玻璃被很小的金银微粒充满了

这里的“小”指的是50-70纳米

它是如此之小以至于光学显微镜根本看不见

你得用电子显微镜才能看见这些微粒

但是罗马帝国覆灭了

然后他们忘了怎么做这个

在这个杯子里你能看见很多的匠心

你能看见它怎么在内部被掏空

就在那个小人身体的地方

去给他一个紫色的光泽

和背景的红色

然后如果你听见人们今天谈论这个

或者如果你读过有关的文章

他们会有一种轻蔑的态度

“哦那帮傻x罗马人

根本不懂科技

他们估计都不知道那是金和银让这个得以实现

这估计就是个意外

他们就这么做了四五个”

这完全就是胡扯

任何真正做东西的人

都应该反对他们

你不会在没有持续的迭代和改善的情况下

做到这么好

你可以想象

这可能是最开始的时候的一个意外

比如有人想让玻璃亮晶晶

试着把金和银放进去

然后他们发现出现了一点变色

他们就开始想“为什么会这样？”

然后他们继续研究

如果我们改变配比会发生什么

玻璃该多厚

这么好的结果需要很长时间

这意味着

在罗马，人们在做

我们今天认为的材料科学

然后这失传了

也有别的事情发生

比如在拜占庭帝国

他们有火焰喷射器

不是那种小不点

他们有巨大的，加压的容器

就在船身里

从金属管里发射燃烧弹似的实体

用来点燃边上的船

它很像燃烧弹

因为水没有把这火扑灭

这是很厉害的武器

是拜占庭帝国的机密

用以抵御君士坦丁堡（东罗马）的攻击

一次又一次几百年

直到有一天因为各种原因他们顶不住了

然后这个军事机密就消失了

现在没人知道怎么做了

很显然我们又发明了火焰喷射器一次

但是他们还是不大一样

这是安提基特拉机械

以一个希腊的小岛命名

在那个小岛附近中的沉船上找到了它

他就是一大块被腐蚀了的金属

或者说一大堆被腐蚀了的金属

但是很明显在他们刚被发现的时候有齿轮存在

人们一天天研究它

发现他是一个机械的日历

用来表示年份、月相、各个星球的位置

下次奥运会在什么时候

人们对它残存的部分做了检查

尝试去推导出各个齿轮的位置

结果和我我想的大相径庭

一开始我听说它的新闻，我想

“噢，在古希腊的是哦胡一定有小巧的齿轮

真不可思议”

但是请让我给你展示一下这个东西

在重修中研究得出的规模

和这个东西到底是什么

这看起来像很多齿轮是吧

但是，看这还有更多

古希腊就有了这个

但是这不是我们今天对古希腊的概念

需要明白的是

这东西前一天什么都没有

第二天就有个哥们把这建好了

你需要一整个科学的流程才能建成

一个如此复杂精密的设备

我们今天对此一无所知

所有这些科学都被遗失了

我还可以源源不断地举例子

比如这些都是历史上被遗失的东西

但是我们没有时间

我只是想重申

现在我们生活在一个非常幸运的时代

科技成形已久

我们看着科技变好

所以我们很自然地相信科技会随历史不断变好

这些在历史中的瞬间就只是我们注意到的小闪点

但是他们不是小闪点

它实际上某种意义上是世界历史的正常轨迹

伟大的科技就完全丢失了

因为创造他的文明覆灭了

或没能把这些知识传递下去

科技无时无刻不在倒退

不只在古代历史中还在现代中

我们无时无刻不在丢失知识

我要读一段采访Bob Colwell的选段

他是有一段时间曾是英特尔的首席微处理器工程师

但是这段采访发生在他任职英特尔之前

是在硅谷蓬勃发展的时候

那时候他在一个叫Multi-Flow的创业公司工作

他们那时在尝试去做一个超长指令字处理器（\*long）

那是这还是个很实验性的想法

他们在尝试设计它时也遇到了很多问题

比如当你想要设计一个用别的制造商的组件的芯片时

他就是不能让组件可靠地工作

然后他就陷入了“可咋整嘛”的境地

所以他说：

Rich Lethin和我去德克萨斯理查森德州仪器朝圣

我们说：“据我们所知，很多您的芯片不能正常工作

这对您来说很意外吗？”

我有一半担心他们说

“啥？您疯了么？您肯定做错了什么！

算了吧，您根本不知道您在做什么。

用别人的芯片去吧”

但是没有他们说：

“是的我们知道，让我看看你的单子”

然后他们看着单子说

“这是一些你不知道的”

顺便说不只是德州仪器

他们的部件不比任何人的差

摩托罗拉也不咋地

仙童也不咋地

他们都有这个问题

所以因此我问了德州仪器

“为什么整个产业全都同时在这跌跟头了

我们想弄明白您芯片的短处都快想疯了”

然后德州仪器的老哥说：

“第一代的晶体管－晶体管逻辑

是那些真正知道自己在干什么的

白头发老哥做的”

“新一代的是刚毕业的小孩们做的。

他们不知道去问改变封装会对感应峰有什么影响”

当你在芯片上改变电压时

它会生成一个磁场，事实如此

当那些磁场穿过芯片bat的时候，就很不好

你知道那些新人，设计这些芯片的

不知道去重视这件事

这就是为什么科技会退化

或者说至少这是原因之一

在一代与一代人之间传递

成就科技进步

所需要知道的重要知识

是需要很多工作和努力的

在这个传递过程中是有不可避免的损耗的

如果没有这种代与代之间的知识传递

文明就可能灭亡

因为那些文明赖以生存的知识会退化和失传

让我们来聊些覆灭的文明

实际上是一群覆灭的文明

这展示的图表是来自一个

你可以从油管上找到的演讲

叫《1177BC the year civilization collapsed》

（公元前1177年文明覆灭）Eric Cline

我们现在来说青铜时代末期

那个时代有些你听过的文明

埃及、迈锡尼、赫梯帝国、巴比伦人

这些文明或者说文明网

贯穿整个美索不达米亚平原和地中海

他们发展出了挺复杂的贸易网络

在这个图里每一个点都是一个文明

线是文明与文明之间已经建成的交流贸易线路

但不是所有文明都互相连通

他们互相连接的程度足以

让他们在需要时，间接地有效率地传递东西

从一个地方到另一个地方

这很重要

因为他们用以防御的青铜

在那时候很难制成

你得把铜和锡混合在一起

铜相对来说难找一些

只在像塞浦路斯这些地方找到

锡也很难找到

并且离那些铜矿很远

比如阿富汗

然后你得持续的来回运输这些东西

才能成功的做出青铜和其他文明赖以生存的物品

我们不知道在这次文明灭亡中发生了什么

但是我们认为是环境应激源所引起的

比如有一次大干旱

也有理论认为上一次大洪水

导致了人们互相攻击

所以你得把你的商船用来防御

这些眼花缭乱的文明在几百年后

几乎所剩无几

我这里的所剩无几甚至不是国家灭亡

城市付之一炬

而是语言和文化都没有幸存

尽管他们往石头上刻字

来尝试记录

但是知道现今还不能翻译绝大部分那些文字

有太多的知识在这次大崩塌中流失了

我们待会还会再回来说这个

我想把这个桥接到现代社会

剩下的演讲的主旨就是

软件现在实际上在倒退

这可能是一个软性的倒退

只是让事情变得对我们来说不方便

但它以后可能导向一个硬性的倒退

因为我们的文明依靠软件

软件无处不在

我们所有的通讯系统都是软件

我们的交通工具是软件的

我们现在有单纯因为软件问题

导致数百人丧生的飞机

飞机的其他部分完全没有问题

我不认为大多数人相信我所说的

“软件在倒退”

软件现在看起来依然在蓬勃发展

我想至少说服你

软件倒退是有可能发生的事情

这就是我演讲剩下部分的目标

我要说的是：  
我们所说的这些崩塌

规模都很大

所有那些文明都覆灭了

但是这需要上百年的时间

如果你在崩塌的前期比如前20年

你可能只会想到：

“事情没有20年前那么好了

但是问题不大

我们还没什么变化对吧”

然后你一直这么想

每20年就这么想一次

几个城市崩溃了

到最后就一无所有了

罗马帝国的覆灭用了大约300年

如果你正处于一个如此之慢的崩塌中

你会发现它吗？

你会知道从内部看他是什么样的吗？

当然我想到了会有这样的回应

“你疯了，软件很好

看看这些赚了钱的互联网公司

他们也在改变我们生活的方式”

我说“确实，这些都在发生

但实际上在发生的是

软件过去几年只是借着硬件发展的东风

过去的几十年我们在硬件科技上有惊人的进步

计算机变得越来越快

这绝对是人类历史上最伟大的成就之一

我们不知道怎么就达成了这个

软件变得‘更好’了（引号）

因为运行他的硬件更好了

这是主要原因

软件科技本身有很长时间没有进步了”

那你还可以说

“看看这些我们近几年能做到的牛x东西

比如AlphaGo能在围棋上打败人类的AI

你还可以说像Instagram这样的软件

可以把你的脸变成别人的脸，绝了

我们之前可不能这样”

事实如此

但是这些大多是因为硬件在变快

绝大部分我们做的牛x东西

都是因为机器学习算法

这种算法主要就是依赖于算力

来产出惊人的结果

很难想象在20年前的电脑上

去训练AlphaGo

这些例子之中当然有软件科技的进步

软件学习算法绝对在进步

但是这有两个问题

主要是这只是软件科技总量的很小的一部分

在用机器学习算法产出惊人结果的东西

只是程序中很小的一部分

机器学习算法实际上很简单

只要你理解了其中的数学原理

特别是你不需要去训练他你只需要用它

比如说你手机上的那些有意思的app

比如给你p图的软件

它其中做那些我们认为有趣有价值的事的部分

那部分程序和所有那些其他的程序

比如加载你脸的位图的程序

对输入事件产生响应的程序相比

是极简单的

加载位图、响应部分的程序是很复杂的

也正是他们有些许的分崩离析

我会把想机器学习那类小的实际的

技术进步和整个剩下的保守或说退化的

部分区分出来

当然这些进步依然是让我们感到激动的

让我来尽我所能解释退化的部分

假设我们不再指望软件正常运行了

我不太确定这什么时候会发生

你知道电脑一直有一个“有点奇怪”的名声

如果你倒退几十年前

这名声主要是因为不太易用

很难明白怎么用它

但是今天如果你在用一个程序

然后程序做错了事

你基本上会觉着“毕竟是软件”

重启就完事了

以前可不是这样

如果我们的标准在逐渐缩水

最低标准能缩水到什么样

在它不能持续下去之前

所以我决定去说

“我想要量化或者说做个图表

我每天忍受烂软件的量

从现在起我要在我用的软件出bug

或者不合直觉不正确的时候截张图”

我在命令行里给我的编码器写代码的时候

控制台开始提示说

“尝试索引的值是零”

因为这是在Lua中写的

然后我去Emacs去整我的代码

Emacs被设定成重新加载改过的文件

它之前运行的很好

但是突然他出毛病了

它重新加载文件太早了

导致它只加载了一半的文件

剩下一半就被割裂开了

所以每一次它这样的时候我都得手动重置它

然后我去Gmail我得给团队别的人发个邮件

关于画面上的东西和决定去做什么

然后我复制之前邮件里的一行

粘贴到回复区里

然后我开始打自己的回复

然后它就成了个3个字符宽的柱子

因为他们不知道怎么

复现了微软各种傻x格式bug

就是那个在90年代到2000年

让所有人都失望了的那个

他们现在到Gmail里了！

我也不知道怎么修好这个

你尝试去修它

一会之后你就得删个什么隐形的文件

我也不知道了

烦人死了

然后我就说：“得了

我去做点正经的编程吧”

我打开Visual Studio

我想把我的命令行引数敲进去

我一尝试做这个

就弹出这个窗口说：

“集合被修改枚举选项可能不能执行”

为什么？我不知道

我甚至不知道这还是个问题

我就是告诉他一个字符串

我们甚至还没尝试对字符串做什么事

这就是存着它到我们跑程序的时候再用

很显然这太难了

这只是Visual Studio问题的冰山一角

Visual Studio有无数的bug

但是这是最有意思的那个

因为我在做的事实在是太简单了

然后他还不能稳定地做好

我不知道这发生的概率是多少

估计可能是5%或者4%

我又决定打开steam玩点游戏

让我在Epic Store上下个游戏

然后不知道为什么我们不太能开始下载

然后我去steam因为它更可靠

历史更悠久

我实际上能下个游戏

当我去安装窗口时

他就是个黑窗口

所以我得重启steam才能玩游戏

游戏打开了我Alt+Tab切出去去看点东西

现在全屏全都乱了套了

游戏就缩在一个角上

我又得重启游戏才能让它全屏

我又去看点CS比赛

C9跟LG打了一场不错的比赛

大概一个月之前

但是整局比赛都有一个神秘的第六人

在C9这边叫“未定义”在左上角

我给你们CS粉丝们把地图放大一下

“未定义在左边”

十万人在看这场比赛

这玩意从头到尾都在那

我在想我喜欢的一款游戏

叫《创世纪4》

所以我上了这个网站

网站上面有地图

然后这地图基本上就全乱了

因为他看起来好像行都混了

我换了个浏览器才能正常看

我需要签证才能来俄罗斯

所以我去办签证的网站上

输我的信息

我把我的电话号码敲进去

我把+1敲了进去（区号）

它好像不太喜欢

然后说电话号码无效

但是我不能把电话号码改好

无论我输什么进去他都不接受

因为无论电话号码的变量是什么

它都是无效的

也不会被重置

所以我得把程序关了

把网页关了

把我的cookies更新

回去从头来过

还得在敲电话号码时十分小心

这种类似的问题实在是太多了

这些都是在几天之内

我都不需要费劲去找这些毛病

我就需要不再收集这些了

然后我就来这了

就好像它上赶着来给我演讲添点例子似的

就在这个酒店里头

我在这写这个演讲写了几天

他们有这种软件控制的加热和灯光系统

你得按这个不是按键的按键

才能控制东西

有一定概率不是每一次

我开关空调的时候

电话会响

甚至不是整个铃声

他就卟卟卟响两声然后就停了

我知道这不是故意的

因为这不是每次都发生

这不是我编的

这现在也在我房间里发生

为了这个演讲两小时以前

我正在最后做一个图表

我下了完全正版授权的

Creative Cloud Photoshop

我第一件事就是去加载New Document

BAM

New Document插件因为程序错误加载失败

我想说的是这对我们来说已经司空见惯了

而且在越发地严重起来

你也试着每天数数

因为我们习惯了

在我一开始想数一下的时候

我自己甚至都没想到会有如此之多

试着自己每天数一下

做个小单子记下每天遇上的这种事

我确信你肯定会对有这么多问题惊讶的

我不确定你们有人知道这个短语是什么意思吗

五个9

我很确定很多人不知道

这在90年代到00年代是一个很常见的短语

当人们想向你推销软件或者硬件系统的时候

这意味着这个系统99.999%的时间

都能正常工作

四个9就是99.99%

我们不再这么说了

我认为部分原因是因为9的数量会减少

我们也不太能再让他多个9了

没人…绝大部分人都不太担心

我在过去的一周里都在做这个演讲

两次，一次是我在飞机上睡觉的时候

另一次是我晚上在房间

我的笔记本就在睡眠模式直接重启了

把我所有的程序啥的都关了

因为他在更新也许不是在更新

我觉着它是在更新

没准就是系统崩溃了

但是我认为他是个更新

这自动把我的电脑减到了三个9

甚至比三个9更少

笔记本本身都少于三个9了

在它上面的程序不可能是四个或者五个9了

我们甚至失去了我们曾有的

对于质量的花言巧语

如果你说软件bug很多

那些比如网页程序员、黑客或者新闻人就会说

“啊，我们知道

但是市场不会为之买单的

我们可以把他做得更好

但是这需要时间和钱来修bug等等

客户不会为之买单

或者说市场会惩罚这种行为

因为上市的时间更长了

从某种意义上是对的

这句话当然不完全对

今天我在想的是

如果你没见过一整个工业

几十年能生产牢靠的软件

凭什么你认为他们能

他们在说只要想我们就可以做

我们只是不在做

但是为什么我相信他们真的可以呢？

因为我们说过

在代与代之间传递

有些知识的要素我认为没有被传承下去

有关如何使软件bug少的知识丢失了

甚至科技公司的知识都变化了

再一次这强调了软硬件之间的区别

一个硬件科技公司曾经是做先进材料

或者设计新型雷达

让你做到之前所不能的事情的地方

现在在硅谷据我所知

一个软件“科技”公司

只是一个做与电脑有关的东西

希望能混出一个市场定位然后分一杯羹的公司

重点在于市场定位

重点不在于软件本身

重点更不是设计更先进的科技软件

把科技边界再拓展一点

这是之前硬件公司一直在做的

我们甚至腐化了“科技公司”这个词

我现在想把话题转得贴近生活一点

作为程序员几十年来一直有有一系列的干扰

最开始你得用机器语言给你的电脑编程

然后有了汇编语言

然后我们有了高级语言

像Fortran、C、C++

现在我们有C#或者Haskell

或离机器更远的JavaScript

为这个开脱的说法是

我们在一个有更多干扰的环境里

你的干扰越多

已经做好的工作就越多

因为你不用去担心

调度机器指令

我们真的在变得更聪明

我们在节约力量

我认为这是对的

我不认为有人还想再在汇编里写东西

这很浪费时间

但是在这条链上的某一点出错了

很多时候错误就是这么来的

你一开始是对的

然后你把他推导的太远以至于到了错误的地方

但是所有这些里面重要的是

我们只看见了一面

我们看见了我们在变智能在省力气

我们看不见所有这些的另一面

另一面是会存在一个相对的能力的丧失

因为我不再用汇编写程序

我失去了在汇编里写程序的能力

如果我用一个过于高级的语言

然后还像所有人一样懒

我不知道我的变量在内存里的地方

或者它们长什么样

甚至他们有多大

我当然不知道CPU在干什么来响应我的代码

我可能很害怕用非托管语言

因为“内存分配”这个概念听起来就太难太吓人

即使我是一个用非托管语言写程序的人

可能我也害怕指针

然后开始形成“怕指针教”

像那帮现代C++的人

我们开脱的修辞是

“我更智能”

“我不应该需要去做那些低级的事”

但是现实的一部分是因为这些选择

我们丧失了一部分能力

这两方面可能同时都是对的

我不是说我们在升级的过程中

没有变得更智能，可能也有点

我是说这有个问题

升这些级的意义应该是让人们更有生产力

但是现在程序员不像过去那么有生产力

事实上对于我来说平均每个程序员的生产力

正在接近于0

如果这是对的，一级一级地走向更高级的干扰这件事

有用的证据是什么

大概能感觉一下生产力下降的方法是

你去观察一个公司，比如

Twitter或者Facebook他们都雇很多人

然后你看看他们的产品

你感觉一下这个产品每年变化多少

Twitter上每年新加了多少功能

Facebook上加了多少功能

不太多对吗

然后除以公司的工程师数量

一般是几千或者有时几万名

除以之后是一个很小的数

它会很接近于0

这是怎么回事

为了重申这种生产力的区别

不只有我认为存在这种区别

我给你们放一个采访的节选

主角是Ken Thompson

他是原来UNIX系统的作者

他正在谈论他在贝尔实验室的时候

当时他刚开始做UNIX

在一个在现在看来完全没有软件的计算机上

在某一个时间，我意识到

之前没有意识到

我距离一个操作系统有3周

每周写一个程序

一个编辑器，我需要一个编辑器

来写代码

我需要一个汇编程序

来把代码变成我可以运行的程序

然后还需要一个内核类的覆盖称之为操作系统

幸运的是那时候我妻子正好去度假3周

带着我大概一岁的孩子

去加州看我的岳父岳母

妻子孩子都不见了，我自己一个人

然后一周、一周、一周

然后我们有了UNIX

我觉着现在的程序员不这么有生产力了

他说了“现在的程序员不这么有生产力了”

然后所有人都笑了

这并不好笑

这真的不好笑

当你考虑到在人们有的生产力和应该有的之间

有多少浪费的时候

如果事情不是一团糟的话

我认为软件的耐用度在下降

程序员的生产力在下降

如果你要是说实际上软件科技在进步

感觉上和这些事实有点矛盾

我认为软件在进步这个论证很明显是错误的

除了一些小的部分

为什么这这么不好

为什么写程序这么难

为什么今天我们在写程序的时候这么悲惨

因为我们给一切都增加了过多的复杂性

我有一个思考这些的方法叫“你不能简单地”

是各种你以前能在计算机上做的事情

现在不行了

今天你不能简单地从一台计算复制程序到另一台

然后程序就能运行

你得有安装程序

或者Linux上的Flatpak

或者一个“容器”如果你是搞服务器的人、黑客、新闻业的人

人们认为这很酷

啊我们现在有“容器”了

这是个优势或者软件科技的进步

所有“容器”的作用就是让我们回到六十年代

那时候这些里的任何一个我们都不需要去做

因为那时候根本就没有这种事

我们要做的就是一点点添加步骤，添加你需要维持的东西

现在再想一下

为什么你需要一个安装程序来安装软件

是因为CPU吗

并不是

想象你有一些x64的机器语言

别管你怎么把它弄进你计算机的内存里的

它就在那

你就直奔它而去

把程序计数器指向那段代码

那段代码会在Windows PC、Mac、Linux，在Xbox、PS4上

做同样的事情

因为所有这些系统用的都是相互兼容的CPU

那安装程序是为了什么

安装程序是为了克服我们自己在系统层面增加的不兼容性

这是一个极其复杂的东西

实际上绝大部分时间我们都不想要它

我们倾向于认为操作系统是向硬件添加功能

但是他们也移除能力

比如兼容性

移除的时候一般是很武断的

现在我觉着最恶心的就是着色器语言

做3D引擎的人应该知道我在说什么

以前你要是想给多平台编译一个程序

你可以用可移植语言来写

比如C或者C++

你可能还需要做点ifdef什么的来适配不同的平台

这都不难而且大体上还是同一个程序

今天你不能这样

因为如果你决定要用着色器

它在每一个不同的平台上都要用不同的语言

即使硬件是一样的

如果你有一个x86的CPU和一个英伟达的显卡

在一个系统里你得用Metal来写着色器

另外一个系统里就得是HLSL

它们完全不同纵使他们本质上是一样的

要不然你就得n次重写着色器，n很大

要不然你就得用自动翻译系统

来重写着色器

自动翻译系统都很复杂而且烦人

还有bug

着色器比原来那些老的程序简单多了

为什么建一个简单的程序反而更难了

这没道理啊

我们不在乎

这是你不能简单做的事情的列表

你不能简单地复制一个程序

你不能简单地静态链接

你不能简单地往屏幕上画像素

天哪今天要画个像素需要的步骤多疯了

你不能简单地写着色器

你不能简单地在Windows上不用清单编译一个程序

在那些新的封闭系统里你甚至不能运行一个可执行文件

除非它经过一整个过程签了名

所有这些和无数没有在这个列表上的

增加了阻力、bug、工程时间

占用了脑力

让我们不能去想那些有趣的事情

我遇到了几个例子说明了这种乱象不会很快解决

我的一个副项目是关于编译器的

要编译程序你得把它们和人们设备上的库链接起来

比如Windows SDK和C Runtime库

但是现在不同版本的库装在不同的地方

你得能找到他们来做链接

没有让这些东西更简单

微软给你一个叫vswhere的程序

你能在Github上找到它

Vswhere的工作就是告诉你这些库装在哪

这样你就能找到这些库了

这东西有7万多行源代码

分布在70个文件里

他们甚至都没试着把程序打包成一个库

它是个独立的程序

他们现在在想的是

你不能做一个独立程序式的编译器

它很显然得是一套程序

但是一旦你有了一整套程序

多一个怎么办

小vswhere怎么办

他们都不想想这会变得很坏

疯了

我基于别人的做了我自己的版本

把它缩减到大概500行代码

对于一个本质上只有两个问题的事情

应该用两行代码解决的问题来说还是太多了

翻了250倍

同时也在编程世界里

有一个东西叫做LSP

基本上是我听过的最邪恶的东西

LSP的支持者遍地都是，他们正在为它做系统

这系统不是今天就是明天就会睡在你电脑里

可能已经在了

以我的理解，它基本上就是更慢更复杂的库

比如说你有某个语言的编辑器

你想能做到我们几十年来一直能做到的事情

比如你想点一下看见一个标识符的声明

或者做一个工具提示告诉你这个值是什么类型

他们说你应该这么做这个东西

你有了编辑器

再做插件不是自讨苦吃

当然这是个编出来的麻烦

“给不同的东西做插件很麻烦”

为了标准化，你得在你的设备上运行一个服务器

然后你的编辑器和服务器的接口对话

服务器回话告诉你结果

把一个程序变成了分布式的系统

这一整个思绪的瑕疵在于

做LSP的这些人好像都没想到

找标识符的位置没什么特别的

这就是个我们一直有什么都有的API

如果要这么构建我们的API很显然下一步就是

给别的任务也做个LSP

那么现在你的编辑器或者随便什么程序

都要和好几个这样的东西对话

如果你想要出版署名你的程序

你得出版署名debug一整个分布式系统

它的状态记录（State）不存储在中央部分

我们都知道那有多“好玩”

但是库没有那么简单

库还会用其他的库

那时候会怎么样

你的系统上有这么一对服务器

有的可能会宕机重启

相互之间又在同步

就是个灾难

人们正在积极地建这个

在我们花这么大时间重现我们在六十年代能轻易做到的事

在游戏行业我们甚至不能做到我们自始至终都应该能做到的事

今天游戏不能稳定地在全屏下运行

截图里你们也看见了

我不是只说这一个游戏

因为我们都费很大力气让我们的游戏能全屏

挺尴尬的，为什么这样啊

同时现在在PC上渲染出帧数顺滑的画面几乎不可能

无论你做什么它就是不可能

Allen Ladavac和他的团队在GDC有一个演讲

和关于我们达到稳定顺滑的帧数需要什么的论文

我们就是没有这种能力

就很疯狂

然后我们还在别的地方花费精力

所以这种引入到我们各种系统中的复杂

不仅让我们现在的生活变得困难

当我们想做点东西的时候

同时也加快知识随时间的丢失

首先事情变得更复杂的时候

需要知道的就更多

如果你谈论一个广泛分布在人群中的工作

每个个体都只知晓他们要做的一小部分

他们有一个很片面的视角

这样就很难做出好的工作

更难以向未来的人传递知识

另一个事情是

深刻的知识被琐事所代替

深刻的知识可能是一个总体的概念

比如缓存一致性怎么运作

这让在不同的处理器上软件运行得更快

琐事是Unity里的子图形不知道为什么不能正常显示

但是我知道如果打开这个面板切换布尔型

可以暂时修复它

几周以后因为神秘原因它就又回去了

记着发出去之前检查一下就万事大吉了

这是琐事的原因

不仅是因为别的事情不这样

而且因为6个月以后新Unity出来后就过时了

而且把我们的脑力花在这方面就很冒犯

第三个事是好信息被噪音淹没

如果有件事很难理解

努力去理解它的人的比例会很低

事情越难比例越低

所以如果你问别人或者你在学校学或者在网上搜

你得到不好的解答的可能性更高

问题越复杂可能性越高

导致问题的复杂程度增加扩大

再回到文明崩塌来

向系统引入越多的复杂性，越难在灾难中存活

因为我们得保持所有的复杂性

我们现在表现得像是我们能处理无穷的复杂

我不认为这有道理

所以上限在哪

我们怎么决定我们能处理多少复杂性

这和人们今天能处理的量是不同的

如果你有一个工程师能把一整个复杂系统

装在他脑子里还能用之工作

当他退出的时候他得把他的工作传给一个新人

他不一定能把所有知识都传达了

随着时间能传递的复杂性比现在个人能接受的复杂性要低

我为什么要在个游戏论坛聊这个

所有人都知道游戏不严肃或者什么的

至少曾经电子游戏和压榨设备的极限有关

让玩游戏的人惊喜

压榨设备的极限意味着你得十分了解设备

这也和稳健的软件有关

如果你很了解设备

你更不容易因为理解错误而出bug

和软件的稳健性也有负相关性

但是现在我们和这些都不太有关系了

尤其对于独立开发者来说

人们在向Unity和虚幻转移

也有向Mass？的

没有很多人自己写引擎了

我们还没到雇佣做C#小片段

然后插进Unity的某一部分

从没有写过系统性的东西

也没有写过底层的东西

在一方面没什么问题

我没说我们不应该如此

一定程度上这很智能，减少开发时间

能让你更快完成你的游戏

但是我之前说过会有另一面

另一面是放弃做其他事情的能力

放弃怎么做其他事情的知识

与世隔绝没什么不好的

很多人做游戏他们就向Unity里放小片段

但是如果所有人都这么做

就没人知道除了这还能做什么了

一段时间以后会发生什么

我们一直在假设我们永远都能用这些引擎

但是Unity和虚幻开发的时候

还有很多游戏公司的人一直在做引擎

Unity和虚幻就从这雇人

但是当没有自然的途径去学习怎么做引擎之后

因为没人再做了

Unity和虚幻从哪雇人

来维护这些所有人都在用的引擎

到了一定程度他们雇不了人了

人们的质量在下降

因为他们没有那么多经验

再提升回来就很费精力

也许，必然在某一刻没有足够多的人去做与之竞争的引擎

也许在某一刻你都不能维持老的引擎

引擎随着时间衰败

这确实可能会发生

我们习惯于把游戏开发者当作

阿西莫夫书里的“基地”

我们好像知道怎么给电脑写程序

也算上别的程序员

比如说弄嵌入式系统的人和搞高性能计算的人

他们都知道计算机里再发生什么

在软件衰退之后我们还有知识能恢复它归还于人们

我不是很确定这现在会发生

因为我不知道

我不知道是否有足够的人做底层的事情

或者有做高端事情的人知道底层在发生什么

也许可能需要一个“第二基地”

没读过书的人被剧透了

回到青铜时代

那些文明消失的原因之一是

只有少数精英会读写

他们上好几年的学

这是被封锁的

大众不能会读写

估计他们也不想会

因为那些技能没有广泛传播

这些技能就很脆弱

如果社会受到干扰它们就很难持续下去

因为没有足够的人能传递下去

现在几乎没有人知道CPU里发生了什么

这个技能不是广泛传播的

所以他很脆弱

我们真的认为这个我们建造的极其复杂的东西

莫名其妙地比只做青铜器的青铜时代更稳健

青铜时代那些都没有幸存

那些都没幸存，我们怎么知道我们在做的就能

我们可能也有相似的压力源

我们有气候变化问题

我们可能还有新的问题

有那么多的网络攻击

如果国家之间互相断网会怎么样

现在很多国家的很多人都上不去Stack Overflow

去弄明白怎么复制粘贴他们的代码

所以代码产出会受影响

如果中国说我们把CPU都留下了

不卖给你们了

会发生什么

独立的这些可能都不会整垮文明

但是肯定会给系统以重击

如果系统太复杂

它可能就不会顺利地度过这个重击

我就是在说

像Elon Musk说的

科技自己会退化

我们现在就需要开始防止它

在我们能触及到的所有领域

我们得简化运行的硬件

我们得简化用的操作系统、库

简化写的应用代码

简化我们用的通信系统，比如互联网

简化我们编译、debug、分布的方法

简化人们和软件连接的方法

这听起来有很多要做

但是好消息是

所有这些都很可笑地复杂

想找到能改善的地方很容易

简化任何一个系统只需要做它的意志

还有品味

你得有能品出这些东西有多复杂的品味

知道他们简单了会有多好

很多人会说

随便，软件是很复杂，我不信文明真会崩塌

也许是这样

我会说，如果你是个程序员你应该关系这个

即使只为了你个人的未来

现在程序员都不是很快乐

我们通常都很暴躁

暴躁的原因是因为我们一直在做蠢事

而不是做有意思的事情

如果我们还照旧的话就不会有任何改善

如果我们改变做事方法

你自己会更开心

如果我们照旧，未来可能就会很平庸

就像美国航天的未来一样平庸

即使你就是个像活下去的游戏开发者

你就想完成我的游戏

发出去，经济上成功

即使你的视野仅此而已

去除复杂短期来看依然是正确的

即使看起来不是这样

我们都很熟悉这样的故事

5个月之内发售，现在系统还问题成山，充满了bug

一直在丢失进度

我们就得坚持5个月

就过去了就是历史了

重写它要花费很大的精力要跳票

所以我们得坚持5个月

但是这绝大部分时候都是错的

实际上发生的都是用了2年才发售而不是5个月

你受这个系统损害远超过别的事情能损害的

实际上系统可能是花了这个长时间发售的罪魁祸首

所以要简化

简化你自己的代码解决局部问题的过程中

也在形成体系化如何简化的知识

听起来很基础但是实际上我们也已经不会了

如果你很感兴趣，这是一些参考和你能看的视频

Casey Muratori的视频“The Thirty Million Line Problem”

Samo Burja的视频 “Civilizations: Institutions, Knowledge and the Future”

Eric Cline的视频，之前我给你们看过截图

“1177 BC: The Year Civilization Collapsed”

这是全部我要说的

谢谢各位的耐心

Q：很惊人

崩溃会在我们到达技术奇点的时候发生，你不这么认为吗？

我认为简化是一种能防止发生的方法

A：你一开始说“奇点”这就变得太困难了

奇点很重要的就是你不能预测什么会发生

Q：当然，可能会变坏可能会变好。

A：我不像很多人一样相信奇点

对我来说它不是很实际

但是，随着你接近那种状态，事情会变化得更快

如果事情变化得更快，他们就很容易崩溃

Q：对《基地》说的就是这个

A：反对的声音是什么，我一直在批判我自己

反驳的点之一是，如果我们就放任软件变得十分复杂

然后就做个能理解它的AI就好了

也许可以但是，但是你真的希望人类理解不了软件吗？

听起来不怎么好

Q：好我们大概有5到10分钟的提问时间

如果你有问题你可以问

上来问就好了

Q2：谢谢您带来这么棒的演讲，非常好的思想，

也欢迎来俄罗斯。

A：谢谢您

Q2：我做游戏和game maker？已经有17年了

我们花了一年做game maker？的3D

我问自己：为什么我们要做这个

实际上现在我知道为什么了

我之前没想过

一个自由技术人员，上岁数的人

视力开始渐渐衰退

它不再能写，也不再能思考了

因为他习惯一边想一边写

所以他开始用Hansen’s Writing Ball

这是最早的打字机

但是他的风格改变了

当我从Game Maker换到虚幻的时候

作为一个游戏设计师

感觉到我的风格，思考方式改变了

您认为工具会强迫我们以一种特定的方式思考吗？

对于您来说做一种全新的语言是打破这一桎梏

来思考得更宽阔的方法吗？

A：我认为我会同意这个观点，我同时也会说

因为我们经常用工具思考

不必要的并发症或者工具的bug会干涉你思考的过程

比如说你在状态里，你正在工作

然后有什么不好的发生了

你就得去修复它

你不能做你正在做的事了

我认为这很重要

我在做我的语言的时候

我尽可能地去去除这些不合理的并发症

但是它们实在是太多了

有一部分已经印在了我们的假设当中

我通过学习这些复杂的系统来学习编程

我看起来很简单的

实际上和我们能达到的简单程度相去甚远

因为我们的思考就是在那些工具上训练的

我们就拭目以待吧

Q2：谢谢

A：谢谢

Q3：我是个独立开发者

我很赞同您的观点

我也不喜欢Unity

我也想轻松地上画像素

但是我还没准备写自己的引擎

A：你要像自己在屏幕上画像素得自己写操作系统

Q3：有没有现存的我能用的工具？

A：我不知道，因为问题的一部分就是什么东西都是这样的

真正需要发生的是不是关于具体你用了哪个工具

而是发展出对于没有乱作一团麻的东西的美学

就用你自己觉得不乱有美学的工具就好了

也许我们可以让大家都慢慢转变过来

Q3：问题在于在我找去底层的方法的时候

Visual Studio和C++什么的都没这么用

它们都超级复杂还总是故障

A：是啊，我也不知道怎么办

Q4：谢谢，在1968年，

电子计算机这个概念出现以后30年

说大型机有2MB的内存

艾兹赫尔·戴克斯特拉说：

编程从概念上来说让人来做是十分困难的

您确定“简化”一定程度上会有帮助吗？

A：它会有帮助，会不会变得完全有帮助我就不知道了

我没读过那篇戴克斯特拉的文章

Q4：它叫《The Humble Programmer》

A：我认为你可以通过你要解决什么样的问题来评判

首先任何事物本质上都有一定程度的复杂性

所以可能会有问题太复杂以至于难以理解

很难用软件看起来是什么样的来解决

同时也有后加的复杂性

因为我们在用现有的系统来解决这个问题

系统本身已经阻止我们做一部分事情

理想复杂性和实际复杂性是有差别的

我们做的就是尽可能接近理想复杂性

具体这够不够好我就不知道了

Q1：我们省的时间只够一个小问题了

结束之后你们也可以再问他

A：我会逛逛找人聊天的

Q5：谢谢您的演讲

有一段时间以前，当时我正在简化

我的软件界面#￥%\*&@！一堆

为什么我这个方向错了

怎么才能正确的简化

A：你说的是哪部分的简化

Q5：界面、抽象接口

A：我不认为所有人都理解这个

简化和高度概括是有区别的

我们说那是简化

但是实际上它会让事情更复杂

因为有更多不明显的东西

这一点作为一个编程社区我们还没有意识到

以前有过尝试简化编程的运动

比如Smalltalk 或者Logo和类似的语言

他们都是很好的尝试

但是某种程度上也都不切实际

比如Smalltalk在我看来就不是一个能写稳健的软件的语言

它有些好的想法

但是也有不好的想法

我们没有那么明显的不好

这也就是为什么没人用它

很多时候正确的简化很难

但是我们得着手解决这个问题

Q5：谢谢