Flash 协作开发之路

各位同行们，大家好！今天非常荣幸与大家分享一下土豆网在协作开发 Flash 应用程序时的一些经验和技巧。

和大部分的 Flash 开发人员 一样，我在加入土豆之前甚至是在加入土豆之后的一段时间里，都是一位“孤独”的开发者，从拿到笼统的需求开始，便是一个人在战斗。随着项目的不断进行，需求的不断变更，程序本身变得愈发庞大繁杂，整个项目的开发和维护需要花费越来越多的时间。当遇到并行的任务时，我们的精力更是没有办法集中，这些都会导致项目不断延期并且 bug 丛生。

在随后的几个月里，土豆的项目经理们为了解决这个问题陆陆续续招进了一些和我一样的 Flash 开发工程师，希望藉由不同开发人员负责不同项目的做法来提高 Flash 项目的开发效率，但是在很多项目的交接和文档整理过程中又出现了更多的问题，新的开发人员无法迅速加入到开发中来，开发环境无法统一，代码缺少维护和版本控制，同时也缺少可以节约大量时间的重用性。action script 是一门年轻的计算机语言，因此并没有太多的直接帮助可以让我们解决这些项目中的问题。为此我和其他几位负责 Flash 开发的同事，本着能“早点下班”的朴素目的，在经过一些列繁杂的讨论和实践中，总结出了一些在 Flash 协作开发中比较实用的工具和编写代码的规则，用来将 action script 项目掌握在自己手中，并且希望能够用协作开发的方式利用集体的力量更有效率的进行 actionscript 项目的开发。

一） 使用一些必要的工具

在 flash 开发团队的构建中，首先需要搭建一个可以方便协作的开发环境，我们以尽量简约并且部署快速的原则，尝试了一系列的工具，其中有的是开发环境，有的则是程序框架，这些工具无疑对我们的编码和协作开发起到了非常大的帮助作用。由于 as3 在代码风格方面越来越像 java，所以我们很偷懒的参考了坐在旁边的 java 项目组的开发环境，并依葫芦画瓢为我们自己的 actionscript 开发搭建了一个相对完整的环境。其实我相信很多台下的 flash 开发者们有的早已开始使用了这些工具，但使用的方法和我们有所不同，而也有刚入行不久的 actionscript 开发者并不知道他们，那么我们来一一介绍这 5 样工具吧。

1. Eclipse

这是很自然的选择，我们没有办法在 Flash Cs3 或者 Cs4 中编写较为复杂的代码，当然他们也并不是完整的变成软件。现在较为成熟的 action script 开发环境无非有两个，一个是 Adobe 公司推出的 Flex Builder 另一个是 PowerFlash 公司的 FDT，它们都是以插件形式安装到 Eclipse 上的。

他们都是我非常喜欢的开发平台，FlexBuilder 有着优秀的官方支持文档和并将 UI 设计，编译环境测，试环境全部集成了起来； FDT 则提供更强大的代码编辑功能（比如代码模板）和更开放的开发环境（对 ant 有更多样的使用方法）以及对 as2 的支持。土豆网的主要 flash 产品基本都是嵌入在网页中的体积较小的 swf 控件，因此当放弃了使用非常庞大的 Flex 框架的时候，纯粹的 as3 代码在这两个开发平台上的开发并没有任何本质上的区别，可是这两个平台上的建立的项目必然不能够我们想到了一个折中的方法就是无论使用哪种环境开发，或是在哪种环境下测试，在发布的时候，我们都用统一的 ant 脚本和 FlexSDK 进行编译，那么编译的结果也就完全不受到开发平台之间差异的影响，保持一致。

2. Ant

在安装了 JDT 的插件后，我们就自动获得了 ant 这个强大的发布工具。在我们的协作开发中它起到了一个类似于“自动化”的作用，测试的时候可以用它自动检查 svn 中有没有别人的代码和自己的相冲突，发布的时候又可以使用统一的 ant 脚本将设计师最新提交的图片库或者组件拷贝到当前项目中并用统一的编译参数进行编译，最终生成发布后，又可以自动提交当前的代码。这当然对于一个经常需要 check in， check out 的团队来说，当然有着无比的诱惑力。

3. Svn

在安装了 Subclipse 插件后，Eclipse 就有连接到 SVN 的功能，我们一般使用 svn 管理 as3 project 中的源代码，ant 编译脚本以及编译用的 Flex SDK，由于大家的开发环境不同，因此并不需要提交整个项目文件，保证了复制代码时的灵活性。在观察每个 release 版本中代码的变化时，我们开发者可以很好的发现代码中存在的腐化，并为下一次重构做好准本。提交代码到 svn 时良好的注释也是对每个开发者 review 代码最好的帮助之一。

4. PureMVC 框架

首先我非常感谢 PureMvc 的开发团队，为我们提供了一个有着良好文档、完善的单元测试并且试用于各种语言的 MVC 框架（它甚至在最近支持了 JavaScript）。在一个较大的开发项目中，每个参加的开发成员有着明确的模块开发任务，我们发现使用 MVC 框架可以很好的将整个项目切割成各个相应的功能模块，并且由于使用了 Observor 的模式，MVC 框架中的各个类成员使用发送接收消息的方式进行通讯，因此减少了大量的依赖关系。开发者们可以相对独立的对自己负责的部分进行开发。同时 MVC 框架又拥有着可以将 UI 和业务逻辑分离的优点，在同时为某个程序开发多个操作界面的时候发挥了非常理想的效果，这个我们会在后面再次提到。而为什么我们选择了 PureMVC，第一是因为它有着比别的 as3 更好的文档支持以及更高的维护频率，其次是它可以运行在所有的 as3 和 as2 的项目中，而其他的框架如（Cairngorm）只能够在 flex 的 project 中使用，缺乏通用性，并且文档较少。

5. flexunit

感谢 Adobe 公司为我们这些开发人员提供了这样的一款 as3 单元测试工具。flexunit 提供了 junit 中大部分的功能，可以让我们在开发的时为自己或者别人编写单元测试，我们常常会在测试对象的同级目录新建一个文件夹，用来存放单元测试代码。良好的单元测试是避免遭遇发布噩梦最有效的方法。当测试人员发现 bug 时，你可以迅速利用已有的和新的单元测试代码定位 bug，修改后重新让其通过测试。也可以尝试使用测试驱动的开发方法进行 as3 的开发，体验极限编程所带来的快感。

除了以上五个共用的开发工具外，在开发过程中，我们也会经常交流在网络上发现的开发工具，我们鼓励尝试这些工具，但并不硬性要求安装它们，而当它们和团队协作产生冲突的时候我们会毫不犹豫的删除它。比如曾经我尝试使用一种代码自动化工具“Code Smith”编写代码，一开始它在生成 as 代码时表现出了极高的效率，各种各样的功能模块在其强大的模板功能中被构建出来，然而在后续的开发中，我发现这个环境无法迅速的与别的开发者产生互动（比如针对与代码模板有冲突的代码，通常它们是由别的开发人员所编写），效率就会极大的降低，直到最后我无奈地放弃了 CodeSmith 的使用。不过，如果作为一个独立的开发者我仍然会使用这个工具编写自己的代码。

学习使用工具的过程，其实也是我们开发者构建成为一个团队的过程。在安装搭建开发环境的时候我们会互相了解到彼此的编程习惯，以及各种工具的优劣点。合适的工具对高效率的编程至关重要，但是我们认为团队成员间的交流和互动比这些工具更加重要，繁杂的工具和复杂的开发环境往往会带来不必要的麻烦和隐藏的危害。

二） 开发中的技巧

在 as 程序的开发过程中，会遇到许多困难，同时也会遇到许多种选择。比如如何对设计师提供的设计素材进行管理，如何在界面逻辑设计经常改变的情况下稳定住整个程序。<敏捷软件开发>这本书为我们前端的程序员们提供了很好的参考。前端项目往往发布周期非常短，如何能够保质保量的完成任务是我们每个人都会遇到的问题。在不断摸索的过程中我们发现以下的几个构建代码的原则能够帮助我们 flash 团队做出相对正确的选择。

1. “可以运行的程序比什么都重要！”

这句话或许有些极端，但是在互联网公司，产品的需求往往朝着需求疯狂变化的极端方向发展，一段可以立刻用于发布并且测试良好的代码往往比经过精心设计并具有详细文档的代码更能够让你的老板满意，也同样能够让你早些下班…

我们为了这个目的不惜一切代价，在互联网产品变化的速度是取得优势的关键。而对于 as 编程来说，它有一个相对封闭的开发环境，不需要对数据库和底层逻辑进行设计，当拥有一个结构良好的 XML 数据样例时一切便可以开始了。因此，停止对完美的稳定的需求文档的等待，立刻用最简单的方式着手进行编码，往往第一

次编写的程序仅仅能够完成最为基本的功能，但我们并不会因为它功能简单缺乏扩展而采用别的方式。

2. "不预期需求的变化"

在土豆网，疯狂的念头和想法往往会随时随地的从你身边的人身上迸发出来，因此这些变化往往无法以一个程序员的眼光为它们在代码中留出位置。因此尽快完成自己的那个功能模块并提交测试，并相信在新的需求到来时我们仍然可以使用最简单的方式解决它。这种短平快的理念往往比左思右想，预留各种接口和抽象的设计思路要快上很多。我们为离自己最近的任务作出详细的计划，而之后可以粗糙一些，而最后的事情最后再考虑吧。

3. "经常提交可以运行的程序"

为了防止产品经理或者老板到最后"抵赖"他们设计出的产品，我们会经常性的将现阶段可以运行的代码运行起来，套上某个版本的界面交给产品经理们评议，以确定这个东西不会最终与需求南辕北辙，也给了产品经理们重新审视产品设计的机会，避免不必要的资源浪费。这一点在我们的开发过程中至关重要，例如大家对于某种 Flash 广告的展现形式争论不休的时候，我们用最粗糙的方法搭建一个可以看到大致效果的 Demo，也能给做决定的人无比帮助。

4. "选择最重要的功能优先开发"

一开始设计师们不可能为你提供全部的素材，因此首先要求产品经理确认最为重要或者最能反映这个产品特点的功能进行优先开发，这点当人手不够的时候非常重要。因为这些功能除非项目死亡，否则不会发生大的变更，也因此相对稳定，当发布的时候也最为重要。因此让设计师们优先将这些图片从 photoshop 中切出来给你，能让你迅速展开工作。

5. "灵丹妙药-重构"

上面的几条都会加速代码的腐化速度，在 as 代码的开发过程中，由于界面元素和用户交互逻辑经常发生更改，反复打补丁的方法最终会导致程序 bug 难以发现，运行效率下降并且难以理解。当你和同伴们开发的程序安全上线了之后，老板们满意的笑脸往往会为你赢来差不多一个星期的奢侈时间，期间无非是一些简单的修改工作和对新需求的准备，我们可以在轻松的环境和相对宽松的时间里，翘着腿喝着咖啡对代码进行 review，这种放松的心态往往更容易发现代码中的坏味道。额，其实我想说的是更容易发现别人代码中的坏味道…那么让我们来进行重构吧，目标很清楚:简单，将一切在 as3 代码中过于复杂的部分拆分出来简单化，有时间的话还可以把库中的图形重新命名并分类提交给设计师。偷偷发布一个重构过的版本到测试环境上，并提醒测试人员"嘿，我改了'一点点'代码，帮忙测试一下吧…"

三)　土豆的播放器

到最后让我们来谈谈土豆网的播放器-它无疑是每个视频分享网站的核心产品。如何让它在每天数千万次的视频播放中保证最高的稳定性和效率同时又将广告和统计信息等繁杂功能集成在其中，这是一个很大的问题。

1. 如何应用 MVC 框架

说到土豆播放器的构建，我们必须要提到 PureMVC 框架在代码编写过程中为团队协作起到的关键作用。它的最大优点就是把我们播放器的代码编写分成了三个部分，Model，Controller 和 View 三个低耦合的层，这样保证了每个模块之间都可以相对独立的编码。我们的播放器由广告播放模块，视频播放模块和统计模块三个部分组成，这三个模块各自有自己 Model 用来接收和发送数据，并将这些数据处理成容易控制的形式，它们完全独立互不影响，在这一部分我们的程序员只需要专注于对这些数据的处理便可以了。每当到周五的时候，我们将自己开发的部分告一段落把代码放在一起并用 Controller 连接起来，进行整体的调试工作。

在 model 层中，我们使用 3 个 Proxy 处理底层的数据:AdProxy 接收广告投放信息，并加载所有的广告素材(比如图片或者 swf)，UIProxy 用来加载和管理播放器本身的皮肤和图形库，PlayerProxy 则起到加载视频信息并控制播放的功能。

在 View 层中，我们同样有 3 个对应的 Mediator 对应三个 Proxy 分别用来显示和播放广告，视频，以及用户界面。

当这些都搭建好了之后，我们制作了一系列的 Command 来让各个部分联系并运转起来。