CSI Ruminations

Chunting Gu

2012-04

关于 CSI 的过程、设计和实现的反思。

CSI 项目的选择和建立

CSI 开始时的一些情况如下：

- 项目类型
  - 产品型项目，改写一个旧有的产品
  - PCP；增量开发
- 人员
  - 原有系统专家
  - 技术专家（语言和框架专家，OO 专家）
  - 普通程序员
  - 架构师（系统结构设计师）：Neil 在项目开始时还没有到职
- 技术选型
  - C++ (Not Java or .NET)
  - wxWidgets (Not QT or MFC)

CSI项目的选择和建立（cont.）

- 技术的培训和规范化应用
  - 没有培训，默认开发人员已具备足够能力
  - 没有在一开始建立语言的子集和规范
  - Coding standards 虽有，但是没有强制执行，也不能使人信服
- 工具
  - C++Test，没有强制使用
  - DialogBlocks，没有意识到全新的界面需求并不适合使用 DialogBlocks
  - 建模工具 EA，早期有一些 trivial 的代码由 EA 自动生成，应该制止
- 其他
  - Design review DB sucks，design/code review 没有规范
  - Unit test 未做要求

C++能力调查

自评

	A	B	C	
C++ & C	7	8/8	6/7	
Compilation Unit	7	7/9	6/5	
Object	6	7/8	7/8	
Type of object	7	7/8	7/8	
STL (Boost)	7	7/8	5/7	

满分为 **10**（进项目前 / 现在）

Coding Standards 调查

> http://cs-rnd.carestreamhealth.com/confluence/display/KDISVII/Coding+Standards

Naming Conventions

>> File, class, attribute, and method naming conventions shall be consistent for a cohesive set of

E.g. function names GetUserID() and get_user_id() are both OK, but choose one and use it consis

- **Hungarian Naming** (匈牙利命名法, szName, strName, pName, etc.) is **NOT** recommended, /wiki/Comparison_of_programming_languages)
- **Naming samples**
  - Class Name: XxxYyyZzz
  - Class Member Function: XxxYyyZzz
  - Global Function: xxx_yyy_zzz

Coding Standards调查 (cont.)

学习Coding Standards所费时间及应用情况

时间	为什么没有遵守？
5m，过了一遍	N/A
10m，大概过了一遍	没有强制要求，便仍沿用以前的习惯。
5m，没怎么看	不够权威（比如Google C++ Style），没有开会review、强制执行，曾看过公司的C++规范文档，觉得差不多。
20m	基本上遵守了规范里的要求。

最差模块调查

1. 你觉得 CSI 里哪一部分的设计或实现最差（难以维护或扩展）、最需要改进，或者让你很不爽？（列出一两点就可以了）
2. 如果让你现在来重构，有多大把握？（满分为 10，5 表示只有一半把握）
3. 你觉得从什么时候起问题变得这么严重？（从你在 CSI 的最后一天算起：3 个月前，半年前，1 年前）
4. 你觉得根本原因是什么？（开发人员技术不行，流程的问题，或其他问题？）

最差模块调查（cont.）

Q1	Q2	Q3	Q4
没有 domain 设计，如 patient/image/series 这些类	6	1 年前	设计与 coding 不一致
Context, ImageModel	5	2 年前	ImageModel：没有足够的知识功底；Context：严重违背 OO 思想，破坏设计
ImageModel, Toolbar (Configure: 添加删除按钮时难以维护; Create: AcquireToolbar 动态创建时无法适应)	7, 9	半年前	由于进度紧张，一开始就没有设计好，缺乏 review，对后面的变化估计不足

C++

测验：用 C++实现 split 算法

Input: Array A[low ... high]
Output: New array splited by A[low]
	i = low; x = A[low]
	for j = low + 1 to high
	  if A[j] <= x then
	  ++i
	  if i != j then
	  exchange A[i] & A[j]
	  end if
	  end if
	end for
	exchange A[low] & A[i]
	w = i
	return A & w

C++ (cont.)

Split 算法测验结果（仅函数签名）

```cpp
// 大多数
void split(int arr[], int low, int high);

// 很少
template <class T>
void split(T arr[], int low, int high);

// 极少
template <class Iter>
void split(Iter begin, Iter end);
```

C++ (cont.)

不理解值和引用的例子

```
void SetName(std::string name);
void SetName(const std::string name);

std::string GetName() const;
const std::string GetName() const;

BOOST_FOREACH(OpenImageInfo info, infoList) { ...
```

C++ (cont.)

不理解对象作用域和编译单元的例子

```
// Large static function defined in header file.
static void SomeFuntion() {
  // Lots of lines here...
}

// Large global object defined as static in header file.
static const int TOOTH_BITEWING_LM[] =
  {16,15,14,17,18,19,0,0};
```

C++ (cont.)

不理解数据结构和算法的例子

```
if (!imageInfoMap["kvp"].empty()) {
    imgData->SetKV(QualifyDoubleValue(imageInfoMap["kvp"]));
}

void RemoveAll(vector<int>& v, int value) {
  vector<int>::iterator it = find(v.begin(), v.end(), value);
  if (it != v.end()) {
    v.erase(it);
    RemoveAll(v, value);
  }
```

C++ (cont.)

结论：

- 开发人员高估了自身的C++能力
- 开发人员低估了C++的难度

问：

- 开发人员系统而完整的学习过C++吗？
- 或者接受过系统而完整的C++培训吗？

Design Patterns

开发人员没有真正理解设计模式！

Singleton

> Ensure a class only has one instance, and provide a global point of access to it.

- **Only one instance**
- **Easily access**

Design Patterns (cont.)

为了 easily access 而使用 singleton 模式！

建议： 不用 singleton，也不提供 global point of access，改用依赖注入（Dependency Injection），便于解耦与单元测试。

```
class MainFrame ...
  MainFrame(Preference* pref, wxWindow* parent)
```

```
bool App::OnInit() {
  pref_->Load();
  MainFrame* main_frame = new MainFrame(pref_, NULL);
```

保证项目成功的四个条件

- 采用增量式进度安排和阶段划分
  - Drop
- 拥有发现和改正错误的机制
  - 大多数项目在开发过程中都会出现错误，成功的关键在于是否拥有错误恢复机制。
  - Process? Design? Implementation?
- 建立一个良好的产品发布习惯
  - A drop a release
  - BVT
- 拥有优秀的项目负责人、项目经理或技术主管

CSI 的开发模型

CSI 使用的是下列哪一种软件开发模型？

A) 瀑布

B) 增量

C) 螺旋

D) 迭代

E) 敏捷

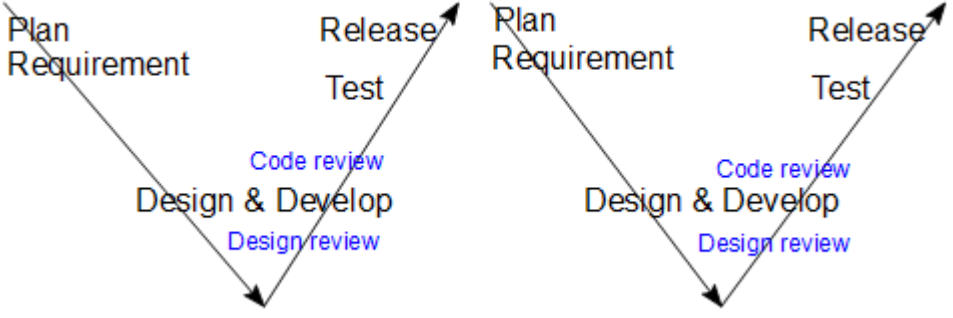
问：迭代是一种软件开发模型吗？

增量和迭代（Incremental and Iterative）

	增量	迭代
同（内容）	分多次进行分析、设计，甚至分多次搜集需求	
异（内容）	以不同的时间、速度完成系统的不同部分	修订已经完成的部分
目的	纠正开发过程中的错误概念来提高整个过程的质量，并最终提高产品质量	通过改写系统某部分的方式来提高产品的质量
	让你能够反复修改，从而改进产品质量	
难易度	较易（进度安排是提前的、线性的）	较难（需要最初阶段的推测、持续的观察、灵活的执行方式）
重要度	更重要	次重要

增量和迭代（cont.）

CSI 里的增量——Drop（一个 V 表示一个增量）



增量和迭代（cont.）

CSI用的是增量模型，但没有结合使用迭代。

从字面上理解，

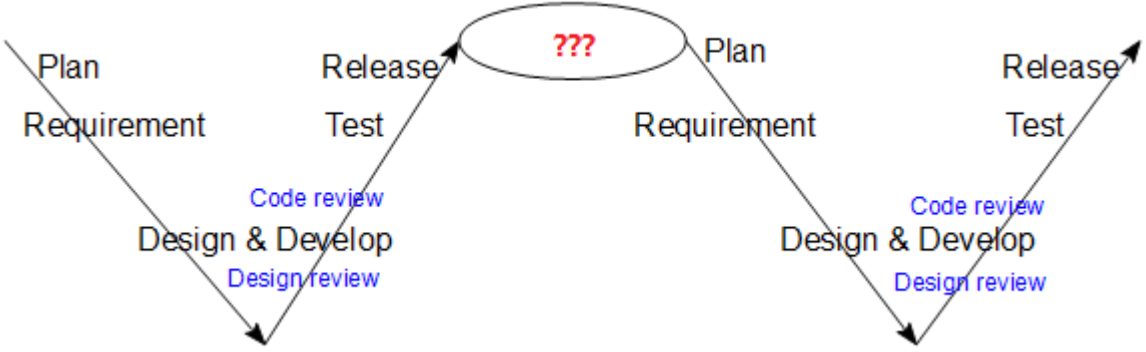
> 每一次增量所“增加的量”就是软件的某些待实现部分，通过不断的增量，即不断的完成软件的部分，最终完成整个软件。而迭代（iterative）也有遍历的意思，所以迭代可以理解为，“遍历”现有的设计和实现，如有需要则以新的更好的方式代替之。

敏捷里的迭代，

> 迭代的不是设计和实现，而是过程，所以一个迭代就是一个增量。而敏捷之外，一个增量可能会有几个迭代。敏捷不需要通过上述定义的迭代去重新设计或改写实现，它用随时随地无时无刻不在发生的重构达到通常迭代的目的。（？？？）

增量和迭代（cont.）

两个增量之间，做些什么？



增量和迭代（cont.）

两个增量之间，暂停和学习：

- 回顾上一个增量的过程、设计和实现，总结成功和失败之处；
- 决定哪些需要坚持、加强，哪些需要调整、改进。

方式：

- 2~4 小时的小组讨论
- 持续几天去尝试发现一种新方法来解决现有的一些问题

错误恢复机制

XP 里的各种实践就是错误防范和恢复机制

Requirement	• On-site customer
Design	• Simple design • Refactoring
Implementation	• Pair programming • Test driven • Collective ownership • Coding standards • etc.

重构

重构的正确时机：

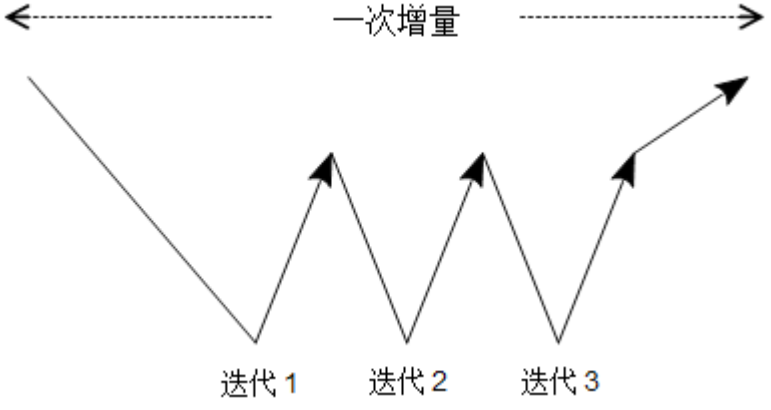
> 重构是持续进行的，而不是在项目结束时、发布版本时、迭代结束时、甚至每天快下班时才进行的。重构是我们每隔一个小时或者半个小时就要去做的事情。
>
> ——《敏捷软件开发原则、模式与实践》，Robert Martin

重构就好比清理厨房：

- 每次做饭后，花一点时间清理，厨房就能永远保持干净。
- 每次做饭后，不做清理，一个月后，想清理就比较难了。

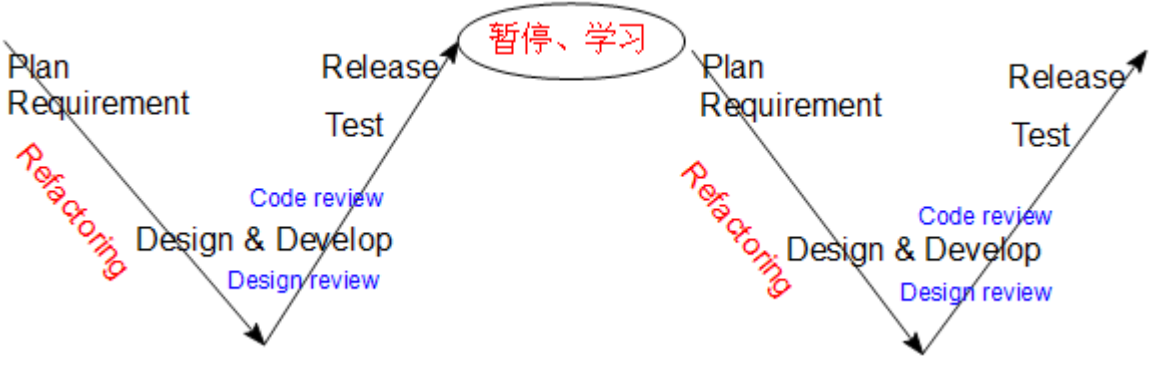
重构（cont.）

增量之中的迭代



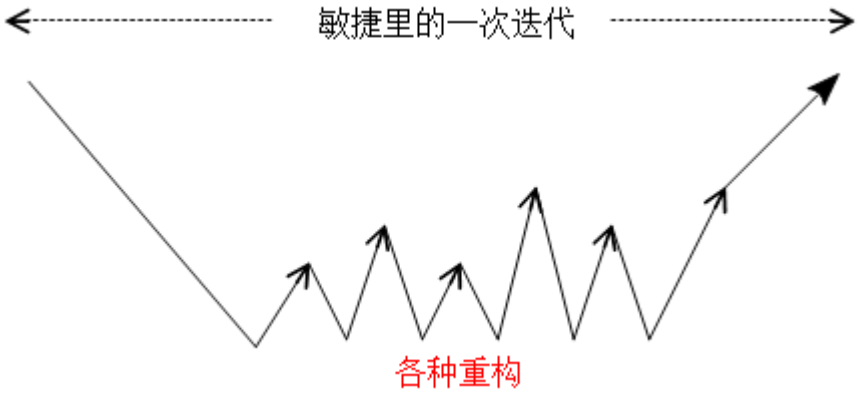
重构（cont.）

我自己喜欢在一个 drop 刚开始时做一些较大的“重构”



重构（cont.）

重构是持续进行的，随时随地。



任务分配和所有权

通常的做法：

- 如果按照组件分配任务，则会影响功能
- 如果按照功能分配任务，则会影响组件 **(CSI)**

应该怎么做：

每个功能都有一个惟一的所有者，每个组件/类也都有一个惟一的所有者。

XP 怎么做：

Collective Ownership (集体所有权)

前提是有单元测试保证，否则死路一条。

任务分配和所有权（cont.）

CSI 的界面类按照所属功能模块分配

GUI Classes	Creator
Button, ToggleButton, ListMenu, etc.	Adam
CheckBox, ComboBox	Andy
FloatingPanel, FloatingWindow	Martin
ScrollBar, ScrollWindow	Steven

结果： 风格、用法不一，重复劳动；返工，浪费时间。
建议： 所有可复用界面类由 1 或 2 人专门负责。

领域模型（Domain Model）（TODO）

Domain Model: An object model of the domain that incorporates both behavior and data.

我们在软件开发过程中，会持续碰到客户需求变更的情况。如果没有领域建模，我们单纯使用直觉将问题解决，那么等到客户需求变更或者有新的需求时，就会面临一个僵硬的前设计！无法在以前的设计上持续深入的优化模型，导致需求变更无法及时深化。设计实现均滞后与变更！ ——@假装刺猬的猪

各种不同的“类”

- 业务类（Patient, Image, PresentationState, FMS, Analysis, etc.）
- 界面（Button, CheckBox, FloatingWindow, etc.）
- 框架
- 技术类（基础结构类、实用工具等）(XML, String, Graphics, etc.)