派生开发让你左右为难吗？

～迈出 XDDP 流程的第一步～

前言：什么是派生开发

在本团体名中出现的派生开发一词并未在软件工程或有关标准中明确定义。但为了区别于”新开发”一词，人们也经常使用”差分开发”,”维护开发”,”改造开发”等各种各样的词语，派生开发一词也是其中之一。本文中的派生开发，定义如下：

- 以现有系统（无论是已经完成的系统还是在开发中的系统）为基础，通过对其中一部分功能更改或进行增加、删除，来开发新系统。
- 如果只是利用现有系统的一部分，那就不能称作为派生开发。

在派生开发类型的项目中，要求开发的时间短，且对原有系统进行理解的工作量大，因而给开发造成了各种各样的制约。

XDDP 就是为派生开发而专门设计的一套开发流程。

注解：维护开发和派生开发

ISO/IEC 14764:2006 中对软件维护的定义如下:”维持软件完整性的同时，对现有软件产品进行的修改”。”维护开发”一词正是用来表示这种开发过程。”维护开发”表示对现有系统进行某种”维护”，而相比之下，派生开发则是指以现有系统为基础进行的新系统开发。

第一章：XDDP 的宗旨（XDDP 的基本概念）

XDDP 并非针对（面向）面向全新的开发，而是为了对现有系统进行设计变更以及增加新功能而设计的一套开发流程。虽然对 XDDP 进行理解和应用需要首先学习这一方法并具备一定经验的，但一般而言，我们需要首先理解 XDDP 的目标是什么，换而言之也就是要明白 XDDP 的宗旨（基本概念）。

我们研究会认为 XDDP 的宗旨（基本概念）如下

- 从多个角度看问题
- 合理分配时间
- 成果物作成时要有会被评审的意识
- 事半功倍

“多角度看问题”是指从各种各样的角度出发，准确把握设计变更所产生的影响范围和避免产生前后矛盾。如果确认了一个设计变更点之后就马上修改代码（编程），这样就很可能会因为没有注意到功能块之间的相互影响，而造成很多返工。派生开发和新开发的不同之处在于，因为在派生开发的时候需要考虑到设计变更可能对现有功能产生的影响，从而造成多个设计变更的前后矛盾，所以有必要对各种各样的影响因素进行相互确认。在 XDDP 的开发流程中，可以从 What，Why，Where，How 几个角度进行互相补充。What（变更需求说明书（需求变更设计书））是指要变更什么，Why 是指为什么必须进行变更，Where（TM）在什么地方进行变更，How（变更设计书）是指如何进行变更，通过多角度进行明确，从而减少设计变更中产生错误和遗漏的可能性。

“合理分配时间”。XDDP 是先完成变更需求说明书（需求变更设计书），TM，变更设计书，最后才进行编程。一般在进行派生开发的时候，我们经常会根据需求发现一处要修改的地方后就马上对其代码进行修改，但是后来随着对整个系统和代码理解的深入，会发现这样的方法会造成代码间的相互影响，此时有可能会增加修改难度。如果在代码修改之前明确代码之间的相互影响，那么原来一些容易产生的问题就会很大程度上得到抑制。而且由于在明确代码间的相互影响上花了很多时间（足够多的时间），这样也会对设计变更的理解更加深刻，并且容易发现那些原来容易忽略的问题。而且由于这时还没有编码，所以就不会有人因为修改程序而造成（造成而）程序间的前后矛盾，从而影响到团队的其他成员，这样做也有利于促进团队成员间的相互信任，打造良好的团队氛围。

另外，虽然大家都知道评审是一种很有效的手段，但是在后期还是会发现有些在前期评审时没有被发现的问题。（但是还是会有些在评审时中没有被注意到的问题被留下来。）XDDP 中通过强调设计者要有成果物会被评审的意识，这样设计者就会用简洁易懂的语句来描述，（这

样设计者就会用言简意赅的文章来进行表述，）从而使得参加评审的人很容易发现问题。

最后关于“事半功倍”，XDDP 在实现高品质的同时避免做无用功。由于 XDDP 的导入大幅减少了缺陷的发生，而把剩下的问题作为产品本身（团队自身）固有的问题来逐个解决。由于通过 XDDP 开发流程在事前减少了缺陷的发生，也就会留出足够的时间去解决这些剩下的问题了。

对 XDDP 的基本概念理解了吗？在 XDDP 的实际运用中，经常会有只是形式上采用了 XDDP，而对其基本概念和实质不甚了解的情况。一定要真正理解和掌握 XDDP 的宗旨（基本概念）。在实际开发中如果出现困惑的时候，不妨再回头去想一想到底什么是 XDPP 的目的（基本概念）。

第二章：我的 XDDP 日记

〈前言〉

下面是新员工山田君第一次进行派生开发时的一些故事。

不久前山田君结束了两个月的研修学习，被分配到了 XX 开发部门。佐藤前辈让他去做计算器程序的变更工作。

6 月 18 日（星期一）

今天，佐藤要我做计算器程序的变更工作。

想要实现的功能是，计算结果的保存，10 进制和 16 进制的变换，还有负数的处理。

佐藤前辈说有两周的时间应该可以完成的，并把代码给了我，按照大学时的做法，我一边看代码一边修改时，被清水课长看到了，他说不能用这样的方法。(>_<)

清水课长说**首先需要清楚想要变更什么，为此需要做什么**。随后他告诉了我**变更需求说明(设计)书**的写法。

这种写法虽然有点像 USDM，但是还不知道有这种把想要做的内容和需要做的内容关联起来的写法。

因为在修改代码的过程中，确实常常不清楚要做什么，所以事前把要做什么事情弄清楚还是很好的。(^ ^)

6 月 19 日（星期二）

今天，把拿到的代码读了。

确认了代码中的已有功能，并试着粗略地标记了什么地方需要变更什么内容。

因为大体上已经知道了什么地方需要变更什么内容，所以随时可以对代码进行修改，但是我一直忍着没有改，只是把要变更的内容先记录了下来。

因为昨天清水课长说**如果马上就改代码，反而花的时间会比较多**，所以一直没有修改代码，但是总觉得有点不放心。

6 月 20 日（星期三）

今天，基于已经理解的代码，把需要变更的内容写到了变更需求说明(设计)书里。

虽然觉得写代码很简单，但是却**意外发现很难写成文章的形式**。我对用文章的方式写设计说明书以及确认代码哪里需要修改有了更深的理解。

另外**如果用文章的方式去考虑如何写设计说明书的话，会发现其实有更好的做法**。这就是立刻修改代码的做法无法察觉到的内容吧。不去立刻修改代码，为的也就是有这样的好处吧。

但是这样写出来的设计说明书最终是如何与代码的修改联系起来的还不是很清楚，还是去问问清水课长吧。

6 月 21 日（星期四）

今天，因为不知道代码的修改与设计的变更是如何联系起来的，所以去问了清水课长这个问题，清水课长教会了我如何使用 TM 表。

所谓 TM 就是追踪矩阵表，就是在行方向写需求规格（设计需求），在列方向写代码文件等内容的一种表格形式的写法。

我试着按照学到的方法去做，**发现设计和代码之间的对应关系变的一目了然**。

这种方法的好处在只修改一处代码的时候还不能很好体现出来，当代码有多处必须修改的时候，需要修改的地方很容易体现出来。

明确了设计和代码的对应关系是怎样的，让人觉得一目了然，确实很方便。看来我真是学到了一个好方法啊。(^ ^)

在 TM 中不需要写变更的方法，而只是标记圆圈，最多再标记上函数名。实际的变更方法写在别的文档中。

把变更方法写在那么小的方格中确实有点难啊。

6 月 22 日（星期五）

描述和 TM 表相对应的每个地方如何进行变更的文档就是**变更设计书**。

在变更设计书中好像不能直接就写代码，要用文章的方式进行描述。这点和写变更需求说明（设计）书是一样的，也是需要注意的。

把变更方法**用文章的方式写出来，这样容易发现变更方法中不太好的地方**。与此相比，如果直接写代码的话，就可能因为没有注意到这点而产生缺陷。

这样的修改方法，也许和对代码多次修改是类似的。虽然没有做过,应该和调试差不多吧。

现在还没有开始改代码，是为养成最后才能去修改代码的习惯。

很期待最后进行代码修改的时候到底是什么样的感觉。(^ ^)

6 月 25 日（星期一）

今天对变更部分进行了评审。

因为佐藤问我做到哪一步了，我就汇报了现在的状况，于是接着佐藤就对我的成果物进行评审和检查。

对变更需求说明（设计）书和变更设计书进行说明后，我在 TM 表中几个遗漏的地方被指出来了。**因为被指出的地方确实是我没注意的地方**，所以我觉得在写代码前被指出来还是很好的。

虽然可能自己觉得已经很完美了，但进行评审还是很重要的。佐藤把课题交给我做，但是到此次评审为止还没有指导过我，所以我有点不满，不过被指出的地方我确实没注意，不愧是佐藤啊。

通过这次评审，我觉得与对代码进行的评审相比，对变更需求说明（设计）书和变更设计书这样文档的评审方式应该说是更有效。

因为是文章的形式，所以比较容易懂，但其实可能是因为整体上有这样一个文档才让大家觉得比较容易理解。

6 月 26 日（星期二）

今天进行了代码变更。

对评审中被指摘的部分修改后，经过佐藤的再次确认，告诉我可以了才开始进行代码的修改。

写了变更需求说明书（需求设计书），也写了变更设计书，评审也做过了，在这样一种状态下再写代码，没有想到能如此顺畅。

因为变更设计中变更方法写的很详细了，所以像以前那样总是弄错应该修改的地方的烦恼也就没有了。（所以像以前那样绞尽脑汁都改不好的烦恼也没有了。）

写代码的时候，一气哈成的感觉真爽啊。（＾＾）

而且没想到一天就把代码改完了，怎么说呢，我终于明白了**比起先修改代码的方法，这样的方式反而会更快**这句话的含义了

6 月 27 日（星期三）

今天，通过单元测试对函数进行了确认，对整体（包括变更的部分）进行了功能确认。

基本上没有做什么修改，一下子就运行起来了，这样太棒了。

以前，我觉得**写完代码后，反复地运行调试然后进行修改是很正常的事情**，但是采用这样一种做法之后几乎不需要再修改代码。

因为要修改什么大脑中已经成竹在胸，所以修改也变得很简单。进行单元测试的时候当然也有发现一些简单的错误然后进行修改的情况，这种不怎么需要修改代码就可以一下子将程序运行起来的效果就会很真实的体会到。

因为佐藤说把计算器设计书也进行修改，所以后期只需要在这次代码调查的内容和变更需求说明书（需求设计书），变更设计书等内容的基础上再把设计书做完的话就结束了。虽然佐藤说要两个星期的时间，但是比预计提早完成了。

总之有种水平得到了提高的满足感。这次通过修改计算器程序也觉得有很强的成就感。（＾＾）

第三章：How to XDDP（概要）

（1） XDDP 中的两种流程

派生开发中，一般有【功能变更需求】和【功能新增需求】这两种需求。

【功能变更】和【功能新增】这两种需求形式在本质上是有很大不同的，因此两者在需求说明书（需求设计书）的写法上也是有所不同的。另外相应地在流程上也分为**【变更流程】**和**【新增流程】**两种。

在下表中整理了不同情况下的使用方法：

<table>
<tr><th colspan="2"></th><th>对应的情况</th><th>相应需求说明（设计）书</th><th>相应的流程</th></tr>
<tr><td rowspan="4">功能级别</td><td>新增</td><td rowspan="2">作为功能
新增来处理</td><td>新增部分：
新增功能需求说明（设计）书</td><td>新增流程</td></tr>
<tr><td>变更部分：
变更需求说明（设计）书</td><td rowspan="4">变更流程</td></tr>
<tr><td>移植</td><td>通常作为功能
变更来处理</td><td rowspan="3">变更需求说明（设计）书</td></tr>
<tr><td>删除</td><td rowspan="3">作为功能
变更来处理</td></tr>
<tr><td rowspan="3">设计级别</td><td>新增</td></tr>
<tr><td>变更</td></tr>
<tr><td>删除</td></tr>
</table>

对于功能变更的情况，根据**【变更流程】**，完成变更部分的需求说明（设计）书（**【变更需求说明（设计）书】**）。

对于功能新增的情况，根据**【新增流程】**，不仅要完成新增部分的需求说明（设计）书（**【新增功能需求说明（设计）书】**），还要完成变更部分的需求说明（设计）书（**【变更需求说明（设计）书】**）。

这是因为功能新增是在现有的系统中添加功能，所以也必然会修改现有系统的某些部分。

【变更需求说明（设计）书】中要特别注意以下几点：

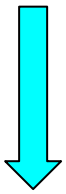
① 对新增功能这部分的变更。（例：为了能够在现有菜单中调用新的功能而新增一个按钮）

② 对现有功能的影响（例：共享的资源和数据）

完成【**变更需求说明（设计）书**】后要让顾客确认，直到顾客满意为止。这样也就没有【设计误解】的问题了。

（2） 变更流程

① 在设计书中记录变更项

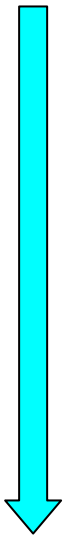


- 基于顾客的变更需求，根据每个需求项的【USDM】，来完成【**变更需求说明（设计）书**】。

目的：通过明确每个变更项并使之可视化来防止变更内容的遗漏。

⇒【变更项目不明确的话，就不知道下一步该如何去做!】

② 对代码等资料进行调查的同时，提取需求项

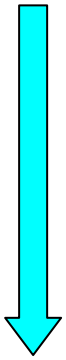


- 调查整理现有的文档和代码。
  - 收集关联的功能设计书、各种设计书、函数设计书，汇总成一览表。
  - 将需要留意的内容记下来。
- 找到并确认变更设计所对应的代码。
  - 通过阅读代码，理解设计、结构和动作，写出调查结果（规格）。
  - 查找与变更需求相对应的变更内容（变更设计）的时候，也要调查【数据结构】【处理结构】【控制结构】。
- 确认系统的架构。

目的：理解派生之前原来的程序，确保所做的变更准确无误。

⇒【如果不理解派生之前原有的程序，就不能实现正确的变更!】

③ 完成追加功能的变更需求说明（设计）书

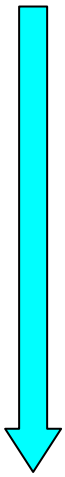


- 为了向在派生之前原有的程序中添加功能，需要抽出变更需求说明（设计）（新增功能时对原有系统的修改部分），然后用【USDM】来写【**变更需求说明（设计）书**】。
- 新增的功能写在【**新增功能需求说明（设计）书**】。
- 与新增功能有关的地方，【**变更需求说明（设计）书**】中也要有记载。
- 确认这两个文档的之间的关联关系。

目的：弄清楚新增功能的切入点。功能新增其实也是一种变更，要防止遗漏变更点。

⇒【新增功能也是变更的一种!】

④ 完成变更需求 TM

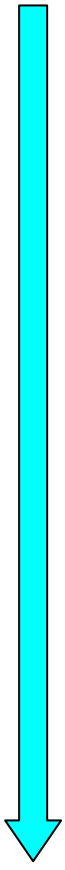


- 【行】方向写变更需求说明（设计）的项目名，【列】方向写代码文件的名称。
- 在有变更的情况下，在交点处标记。
  - 在功能设计书和设计书中发现需要变更的设计的时候，注上标记。
  - 在源代码中发现了相应的地方时，注上标记。
  - 审查等场合发现有错误的时候，注上标记。

目的：在什么地方做了修改就会很明确，这就很便于审查，减少了缺陷。另外，作业中有冲突的地方很明确，避免了返工作业。

⇒【TM 可用于审查!】

⑤ 完成变更设计书



- TM 的每一个交点都要创建一个**【变更设计书】**。
- 变更设计书的构成：
  - 变更设计等的位置信息（标题信息）
    变更需求说明（设计）、变更需求说明（设计）的编号、模块名等在 TM 中的交叉点所表示的信息。
  - 结构的变更
    数据结构和处理结构有变化的时候，其变化用图来表示。
  - 声明语句的变更
    记录函数外声明语句的变更。
  - 函数内的变更
    类：记录函数的变更。
  - 变更后的确认项目
    记录变更后的确认内容（相当于白盒测试）。

目的：通过完成变更设计书，可以把握需要修改的代码的整体情况。另外头脑中也会对设计的意图有一个整体印象。

⇒【如果明确了待变更代码的情况，就谁都可以修改源代码了！知道了为什么要这样设计，下面修改代码就会很轻松!】

⑥ 源代码的变更

- 以代码文件为单位汇总变更设计书。
- 根据变更设计书，一齐对代码的所有变更点进行修改。
- 如果中途发生设计变更，因为此时还没有开始修改下面的代码，所以不会产生返工的时间。只要按照变更代码文件的顺序进行修改就好了。

目的：编码只是对变更设计书的一种单纯的转换过程，所以编码效率应该能得到保证。因为只是单纯的转换过程，所以可以通过增加作业人员来缩短整体的作业时间。

⇒【在此之前因为都仔细考虑过了！那么剩下也就只是简单的作业！一口气做完吧！】

（３） 新增流程

新功能新增的开发流程，基本上和新开发的过程是一样的。

① 编写新追加功能的需求说明（设计）

② 设计新追加的功能

③ 完成追加功能的函数设计书

④ 编写追加功能的代码

Appendix A:术语字典

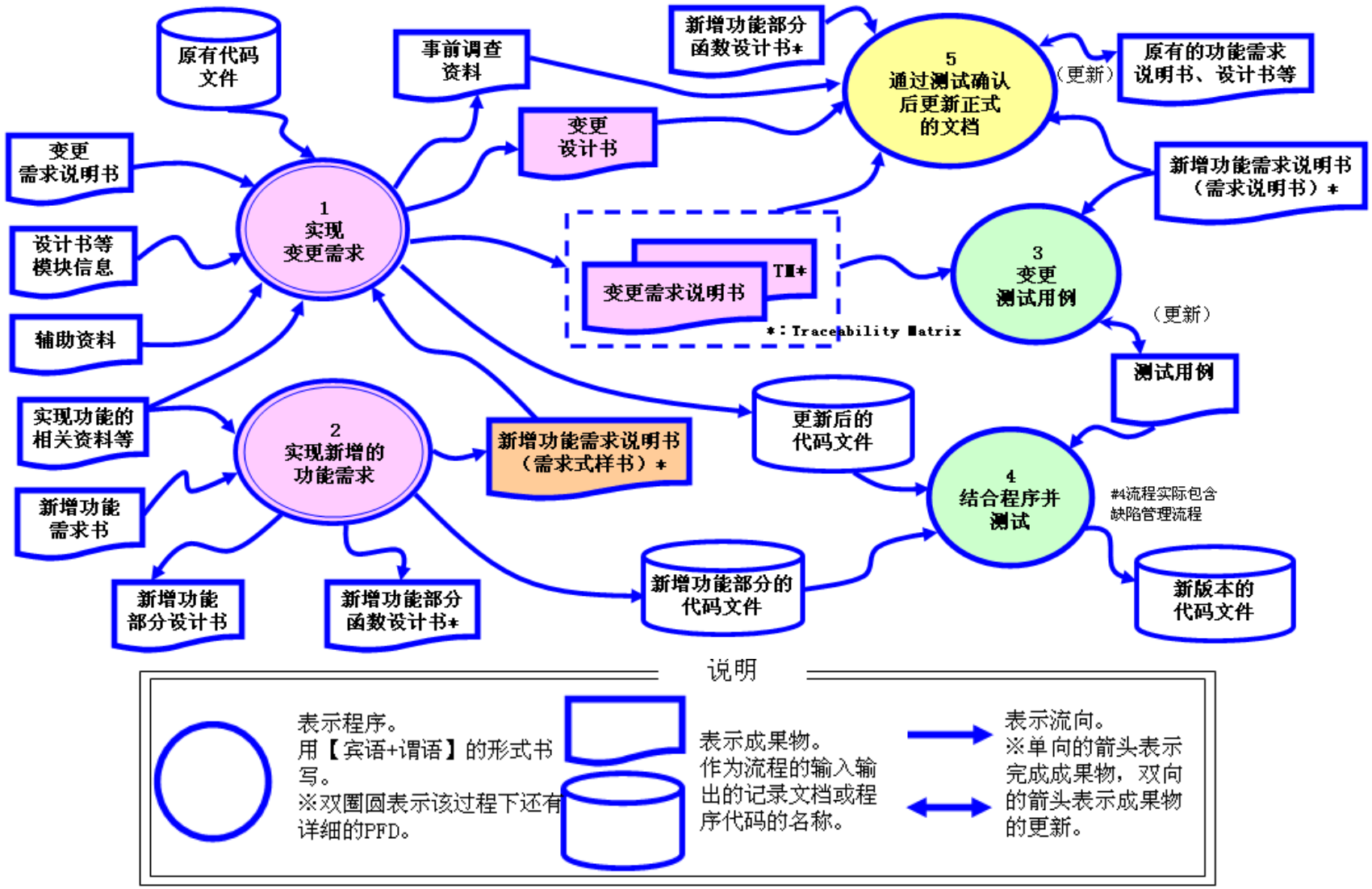
在本文中，术语的定义归纳如下：

术语	定义
【按字母顺序】	
XDDP	[eXtreme Derivative Development Process] 专门为派生开发而设计的开发方法论。作为合理应对【交付期短】和【理解片面】等派生开发特有的问题的对策，XDDP 通过编写三份文档： • 需求变更设计书 • TM（Traceability Matrix） • 变更设计书 用以确保能够进行有效的评审，并且提倡按照流程进行开发的方法。
PFD	[Process Flow Diagram]
※附加示例	记述过程和成果物之间联系的一种表示方法，是基于结构分析手法 DFD(Data Flow Diagram)用于开发过程的设计的一种方法。 PFD 只表示了【成果物】和【过程】的“关系”，没有记录“顺序”。仅仅只记录了“执行该过程时必要的输入”以及“过程的输出”而已。PFD 通过使开发流程可视化，明确了项目的成果物是通过怎样的流程编写的，又是被什么样的流程所使用的，最终设计出最合适的项目计划。
USDM	[Universal Specification Describing Manner]
※附加示例	需求说明（设计）的表示方法。将需求和设计分阶层表示。 USDM 中，功能性【需求】用“动作”来表示，而【设计】承担由动作而产生的职责。这样一种考虑方法把需求和设计在表示方法上分开了。另外，为了防止设计的遗漏，还需要描述需求产生的背景也就是“理由”。
TM	[Traceability Matrix]
※附加示例	为了确保【需求变更说明（设计）书】和【变更设计书】之间的可追溯性的一种利用矩阵形式的文档。 TM 针对“变更需求”，为了使影响范围可视化，通过以下几方面进行描述： • 在矩阵的行上，记录“变更需求说明（设计）”。 • 在列上记录统一的源文件和任务等实际的代码。 • 如果行（设计）和列（实际模块的代码）相交的话，在矩阵的交点处标记。

术语	定义
变更设计书 ※附加示例	在派生开发中，具体记述变更部分的文档。原则是要与在 TM 中矩阵的交点处的标记相对应。 变更设计书有以下几个特征： • 具体的变更方法尽量用文章的形式来描述。 • 需要彻底描述有差异的地方。 • 用 How 的视角来写。
变更需求	需求（Requirement）的一种。对于原有作为派生基础的软件和系统，修改全部或部分（包含删除部分功能、新增功能等）的需求。
变更需求说明（设计）书	在派生开发中，干系人对于在本次开发中想要变更的内容（Change Requirement）进行细化识别（Specify）后总结的文档。 变更需求说明（设计）书有以下几个特征： • 用阶层化的方法把握“变更需求”和“变更设计”。 • 变更需求和变更设计必须以“before”、“after”的形式表现。 • 用 What（以及 why）的视角来写文档。
成果物	在开发过程中完成的文档或程序代码。
功能说明（设计）书	这是一份对于开发对象的系统来说，以非专业人士的角度描述需要实现什么样的功能，以及如何实现的功能说明文档。 不仅仅是功能，文档中同时也要穿插一些对品质、性能等的描述。
功能新增需求说明（设计）书	派生开发中，功能新增相关的需求说明（设计）书。 在 XDDP 中，派生开发的说明（设计）写在【需求变更说明（设计）书】中。功能新增需求说明（设计）书在以下文档中可以找到。 新增功能时进行的必要的开发： • 在现有系统的何处新增功能（为了新增该功能而做的变更）记录在【需求变更说明（设计）书】。 • 需要新增的功能记录在【功能新增需求说明（设计）书】。 合并以上两种设计书的定义，就是【功能新增需求说明（设计）书】的定义。
过程	作业工程，以及工程内执行的作业项目。
片面理解	在派生开发中，并不是要作为派生基础的系统全部都要去理解，受限于时间以及成本的原因，只对本次变更的部分进行理解并进行开发。

术语	定义
前期调查	通过解析派生基础系统的程序代码，做成必要的设计信息的行为。在对派生基础系统理解很不充分的情况下，作为派生开发程序前期的准备工作。 前期调查的目的是为了确定变更对象的位置、重构、构架修改等内容。为了这样一个目的进行范围调查并理解资料，最终完成程序的结构、设计目的等前期调查资料。
设计	记述为满足需求而做的具体的行为。 设计需满足以下条件： • 相关人员可以达成共识 • 具体实现方法明确（可以转化成代码） • 能够进行测试 满足以上三点很重要。
需求	利用软件或系统想要实现的内容（Requirement）。
需求说明（设计）书	当开发系统的时候，记述了通过该系统想要实现的内容的文档（有时与功能设计书表示的含义相同）。

PFD 示意图：通过部分修改，设计派生开发流程



出处：AFFORDD学习会资料　派生开发的手法：XDDP的详细

USDM 例子:”派送货物系统”需求说明（设计）书

			需求/需求说明		
	使用案例①		发送取件申请给配送人员		
取货申请	需求	Cont10	接受取货的申请后，把住所和名字等信息发送到负责该区域的派送员的终端上。		
		理由	通过给配送中的配送人员发送信息，希望配送人员可以在配送中途也顺便取货。		
		说明	受理申请人打来的电话，询问并记下待取货处联系人的姓氏、地址和邮件地址等，并将输入的信息在画面上表示出来。		
		需求	Cont10.1	切换到待取货处信息输入画面，补充输入其他信息	
			理由	受理来电时记录了住址等信息，希望重复利用该信息	
			说明		
			<取货信息输入画面的显示>		
		□□□	Cont10.1.1	按下【指定取货按钮】，跳转到【取货信息输入画面】	
		□□□	Cont10.1.2	把通过顾客的来电得到的以下信息放入【取货信息输入画面】： ③ 指定的取货日期 <取货处的信息> ③ 姓名 ③ 地址 ③ 邮政编码 ③ 电话号码 ③ 取货时的注意事项	
		□□□	Cont10.1.3	【指定编号】显示由系统自动设定的数据。	
			<输入补充信息>		
		□□□	Cont10.1.6	指定取货日期栏为空的情况，根据公司规定在受理申请电话的时间后安排出一定的时间输入【指定取货日期】栏。	
		需求	Cont10.2	查找负责申请者所在区域的派送员，并将取货指示信息发送到派送员的终端。	
			理由	根据区域决定派送人员，通常会选择正在派送途中的人员。	
			说明		
			<检索区域派送人员>		
		□□□	Cont10.2.1	输入地址，按下【搜索派送人员】按钮。	
		□□□	Cont10.2.2	显示找到的派送人员的名字。	
		□□□	Cont10.2.3	当搜索出多名派送人员时，从中选择一名。	
			<派送人员信息的表示>		
		□□□	Cont10.2.6	根据派送员信息中的【负责区域编号】取得【区域名】 【说明】从负责区域编号中获取区域名的处理另行准备。	

出处：AFFORDD学习会资料　USDM入门

TM 例子：变更需求说明（设计）书相关的变更点一览

			XDDP事例（变更需求说明） 变更需求说明书和与功能新增需求说明书同时使用时下仅供参考。		TM														
					功能分类	启动部分	StartUp.src	StarDvcChk.src	主操作界面	MainDispOpe.src	设定操作界面	MainSetOpe.src	GrafSet.src	MeasFnSet.src	打印装置设定画面	测量画面	MainMeasOpe.src	CalGrafTyp1.src	Grafdsn.src
类别名（记号）	需求	CAL05 理由 说明	在实时显示收到的数据的地方，画面撑满时增加打印显示内容的功能。 通过将数据打印到纸上，希望能够和过去的图表进行比较。 显示功能已经实现了，只增加打印功能。																
		需求	CAL05.1 理由 说明	在测量信息的设置画面中，增加用于显示打印装置设定的操作画面的按钮。 希望作为测量信息设定的一环进行操作	追加对应（PRT0 1.1）														
			<新增选择按钮>																
		□□□	CAL05.1.1	测量信息的设定画面中的【测量点】下方，增加【打印信息的设定】按钮。					■		○	Fa()			■	■			
			<显示设定打印信息画面>																
		□□□	CAL05.1.5	按下【打印信息的设定】按钮时，增加通过【PRT01.1】调用设定处理。					■		○		Fb()		■	■			
		需求	（需求号码）	进入测量模式后，增加确保缓存处理的功能。	追加对应（PRT0 1.2）														
			理由																
			说明																
			<获得缓存的时机>																
		□□□		捕获到主要操作界面按下【测量模式】按钮事件后，在显示测量模式画面前增加获取缓存处理。 【说明】实际获取处理在新增功能说明书中实现。					■		■				■	○	Fc()		
		需求	CAL05.2	当图表表示区域撑满时，增加打印处理。	追加对应（PRT0 1.3）														
			理由	只有此时才需要进行打印															
			说明	对应新增功能说明书编号[PRT01.2]。															
			<增加日常打印处理>																
		□□□	CAL05.2.1	表示区域撑满时增加由【PRT01.2】调用已设定的印刷处理。					■		■				■	○			Fc()
			<增加测量完成时的打印处理>																

出处：AFFORDD学习会资料　XDDP入门

变更设计书写法示例：

变更设计书的构成（头部信息）

变更设计书

过程名	XDDP 项目	完成日	2007 年　7 月　10 日		
代码名 / 任务名	RectCalPart1.c	完成者			
		修改者			□修正
变更需求说明	因为计算基准只能是设定值，所以需要使之可以临时改变基准值。	确认者			□确认
#CAL.01.2		预估行数	35	预计时间	30
		变更行数		作业时间	

■改善方针

无。

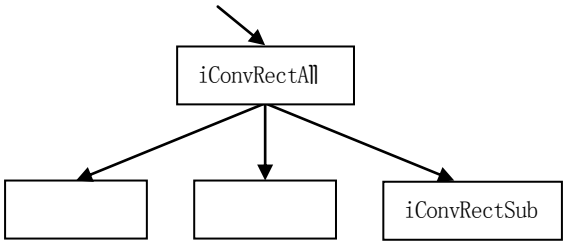
■数据结构的变更

无。

变更方法：
详细地记述过程结构・数据结构・函数等的变更方法。

■函数调用结构的变更

为了避免 iConvRectAll 模块的内聚性变差，分离了其中一部分处理功能。将分离出的功能转移到新增加的（iConvRectSub）函数中，使之实现移动处理和基准值的改变的功能。其他低层的函数没有变更。



■函数外的变更

项目#	变　更　内　容	预计行数	
1	因为函数名变更了（iConvRect → iConvRectAll），所以定义需进行变更。	1	□
2	增加分离出来的独立的函数（iConvRectSub）的定义	1	□

■函数的变更

函数名	iConvRectAll (Vert, Rect, Base)		■变更、□新增、□删除

变更内容：

项目#	变　更　内　容	预估行数	
1	在函数的参数中，增加表示之前基准值的信息（Baseint）。	1	□
2	在计算之前，从测量关系设定信息（TblSetInf）的【Kijun】，中分离出取基准值处理，把分离出的内容放入新函数（iConvRectSub）中，并把 Base 值继承到新函数中。把函数的返回值作为新的基准值进行计算。 在此之后的函数保持不变。	7	□

确认项目：

项目#	确　认　内　容	检查
	结合 iConvRectSub 函数进行测试	
1	将值设置为 Kijun=35, Vert=120, Rect=25, Base=0 时，采用 Kijun 的值作为基准值，函数返回值为：【1230】。	
2	将值设置为 Kijun=35, Vert=120, Rect=25, Base=45 时，采用 Base 的值作为基准值，函数返回值为：【1375】。	

变更后的确认项目

出处：AFFORDD学习会资料　XDDP入门

变更履历

第一版	2012 年 5 月 25 日	初版