Mozart

一个基于特征模型和模型映射的SOA变化性管理方法及在凤凰公司整合项目中的应用

北大 Mozart

一种基于特征模型与模型映射的 SOA 变化性管理方法及在凤凰公司整合项目中的应用

【文档介绍】本文档详细介绍我们在本次比赛中最重要的创新之一：FARMS方法，并展示了将FARMS方法应用到凤凰公司整合项目上的成果。本文档内容是我们团队独立研究所取得成果，除文中已经注明引用的内容和一些具备广泛共识的理论外，本文档不含任何其他个人或集体已发表或撰写过的作品与成果。

1. 绪论

随着软件应用领域的不断扩展和深入，软件的规模和复杂性在逐渐扩大和提高，当需求发生变化时，修改或重新开发软件的代价越来越大，而另一方面，激烈的竞争又对企业响应变化的能力提出了更高的要求，人们迫切需要一种高效、灵活的软件技术来构建企业的 IT 设施。

面向服务的架构（Service Oriented Architecture, SOA）为软件复用提供了非常好的环境，服务的平台无关规约和松耦合的交互方式使之成为良好的复用单元，而服务的编排组装机制更是为软件的构造和应对需求的变化提供了强大的支持。

然而，目前还没有一种比较有效的方法对 SOA 的变化性进行管理：一方面，缺少一套机制支持在服务需求分析和设计时就将来可能的变化进行建模表示；另一方面，当需求发生变化，组装流程需要重新编排服务或增加、替换一些服务时，也缺少一套高效的机制对流程的修改进行管理，这两方面的不足，都严重掣肘了 SOA 方案应对变化性的能力。SOA 作为一种松耦合、灵活性高的技术方案，却缺少对变化性进行显式管理的方法学和支持工具，这是很多场景下阻碍 SOA 成功实施的一个重要因素。

在我们给出的凤凰公司整合项目的解决方案中，采用特征模型来辅助进行需求的组织和和变化性管理，并在引入特征建模的基础上，提出了一种基于特征模型与模型映射的 SOA 变化性管理方法（FARMS， Feature-based vARiants Management for Soa，读作[fa:ms]），通过建立特征模型与服务模型和 BPEL 规约之间的关联，对 SOA 系统的变化性进行管理，并对 SOA 系统的自动重构提供支持。

FARMS 方法的主要能力与贡献有：

- 引入模型模板（model template）技术，解决了模型中变化性表示的问题。我们同时还引入着色（coloring）等标注技术，方便分析人员对模型模板的理解。对模型模版的详细介绍请参考第 2.2 节与第 3 节的内容。
- 定义了特征模型与服务模型模板、BPEL 模板的关联机制，从而可以通过特征模型来管理服务模型和 BPEL 的变化性；
- 使用一种基于 XML 的配置语言 XVCL，实现了特征模型与服务模型和 BPEL 在 XML 文件层次上的关联，在此基础上，可以开发相应的自动化工具，将特征模型的剪裁结果直接转换为 XVCL 配置脚本，实现服务模型与 BPEL 流程的自动实例化，从而能够有效管理变化性从需求到设计的传递，减少系统重配置的工作量，提高 SOA 应对变化的能力。

基于 FARMS 方法，我们对凤凰公司的整合项目进行了变化性建模，从而凤凰公司可以基于特征模型的剪裁对他们的整合方案进行配置。配合一定的工具支持，对复杂且繁琐的 XML 配置文件或 BPEL 文件进行更改的工作量会有显著降低，甚至在特定场景下能完全自动完成。因此，FARMS 方法将会使凤凰公司的 IT 系统应对业务变化的能力得到很大的提高。

文档的其它部分按如下组织：第 2 节介绍 FARMS 方法的理论基础；第 3 节结合实例对 FARMS 方法给出详细介绍；第 4 节介绍了 FARMS 方法在凤凰公司整合项目中应用；第 5 节是给出了方法支持工具的实现分析；第 6 节总结全文并给出与相关工作的比较。

2. 理论基础

在这一部分，我们首先介绍一下 FARMS 的理论背景，然后对方法进行详细的介绍，考虑到这是一篇技术文档而不是一篇学术论文，我们略去了对一些理论问题的深入阐述与论证，更详细的研究背景请参考文后给出的参考文献。

FARMS 方法基于特征建模理论，并受到[CZA05]和[JAR01]等研究工作的启发。

2.1 特征模型

FARMS 所基于的特征建模主要是一种领域工程方法，特征模型在组织需求、发现与管理变化性方面有很强的能力，是当今领域工程研究中最重要的方法论之一。

FODA 方法[KANG 90]首先将特征的概念引入这一领域，FORM[KANG98]、FeatureRSEB[GRI98]、[KAL97]、[CZA05]和[MEI06]等研究又对特征模型及其建模进行了不同的扩展。

FARMS 所采用的特征建模理论主要基于文献[CZA05C]、[ZHANG04]和[MEI06]，前者为特征提供了基数的表达能力，同时定义了特征子树“克隆”的概念；而后两者在特征关系及其形式化定义以及特征模型的验证方面作了很大的贡献。

特征的定义有很多，总的来说，特征描述了软件系统的某一个方面的特性，可能是功能性也可能是非功能性的。特征的有助于捕捉一个软件家族中不同软件系统的共性与变化性。特征通过特征图来组织，特征图是一棵树。一个完整的特征模型应该包括一组特征图、和一些附加信息，如特征的描述、全局约束、绑定时间等。

图 1是一个示例特征模型图，图中已经表示出了特征间的一些基本关系，如精化（refine）关系，并且这是一棵可以克隆的特征树，“数据同步”特征旁的基数范围[1, 18]表示这棵特征树可克隆成 1 到 18 棵树。图中带小圈的特征是可选特征，其它为必选特征。每个特征有一个绑定状态，当一个特征被绑定时，我们认为目标系统就应该具有这一特征。

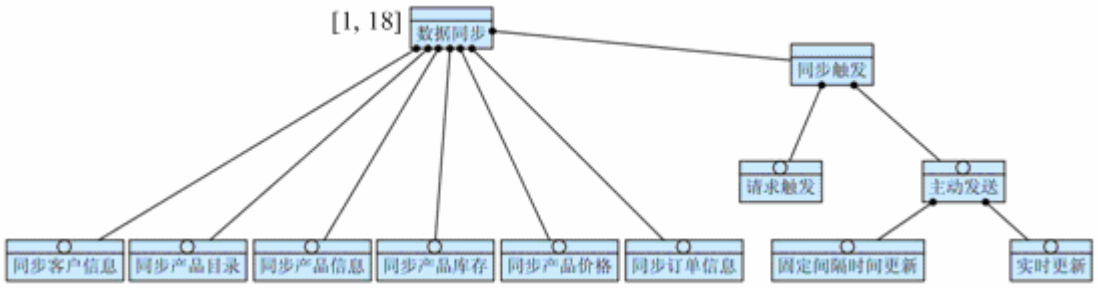

图 1 示例特征模型

同时特征间还有一些复杂的关系，最基本的如全局的依赖与互斥，甚至更为复杂的组间关系。图 2是对特征间全局关系的示例，虚线箭头代表一个依赖关系，而圆圈内一个“S”是一种组约束，代表一个组内有且只能有一个特征被绑定。[MEI06] 中详细定义特征间的全局关系。[ZHANG04]中还提出了对特征模型进行形式化及验证的行之有效方法。详细介绍这些理论不是本文档的重点，具体理论与技术细节在这里就不再详述，读者如果感兴趣可以查阅本文所给参考文献。

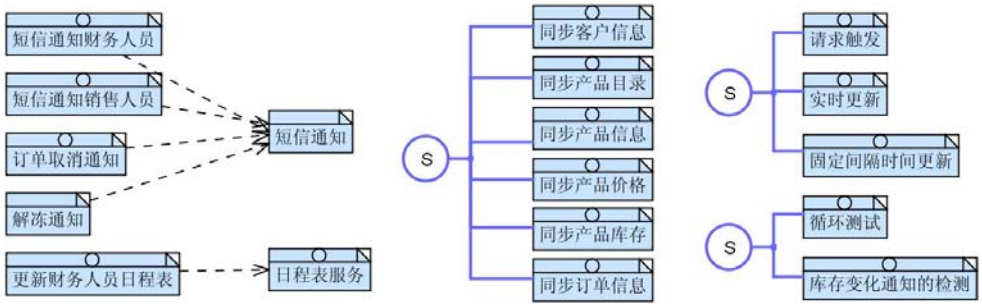

图 2：特征间全局关系示例

我们在凤凰公司整合项目中所采用的特征建模方法主要参考了[MEI06]与[ZHANG04]，所采用的主要特征有：

- 特征的三种精化关系：一般/特殊化、分解和属性化，并通过这三种精化关系来组织特征树；
- 特征的引用以及多视图表示；
- 特征全局关系特别是复杂关系的定义与形式化；
- 特征模型检验准则与方法。

部分参考了[CZA05C]，并具备了其如下特征：

- 特征子树的可克隆能力；
- 对可克隆次数的基数表示；
- 特征内部属性表示。

2.2 模型模板及与特征模型的映射方法

Czarnecki 在[CZA05A]中提出了一种基于“模型模板”将特征模型映射到其它模型的方法，在 Czarnecki 的文章中主要针对 UML2.0 中的活动图和类图进行了分析。这里对 Czarnecki 的方法进行简要的介绍。

在 Czarnecki 方法中提出了类似于软件家族的模型家族（Model family）概念，并认为一个模型家族就代表了一个软件家族。模型家族通过一个模型模板和一个特征模型来表示，其中，模型模版借鉴了 C++等编程语言中“template”的概念，可通过给定不同的参数生成不同的模型实例。模型模板和目标模型采用相同的表示方法，比如活动图模板也就是用活动图来表示，只是在原来的表示基础上增加了一些注释。这些注释分为两种：

- PC（presence condition）：标识一个模型元素是否应该存在；
- ME（meta-expressing）：用来计算模型元素的属性，包括名和值。

这些注释是用特征来定义的，因此特征模型的剪裁会对PC和ME的取值带来影响，从而特征模型剪裁会实例化模型模板，从而得到所需的模型。图 3展现了这样一个过程。

除了这些基本思想，Czarnecki 方法还提出了模型模板的“分色”机制，这一机制有助于分析人员对模型模板的理解；同时还提出了 Implict PC 的概念，即在不同的目标模型中都存在一些特定的约束，这些约束不需要显示的用 PC 表示出来，这样有助于提高建模过程的效率；另外 Czarnecki 方法还给出了简化模型构建和实例化过程的准则与手段，并给出了一个原型实现。最后，Czarnecki 方法定义了在其它目标模型中使用这一方法的建议。

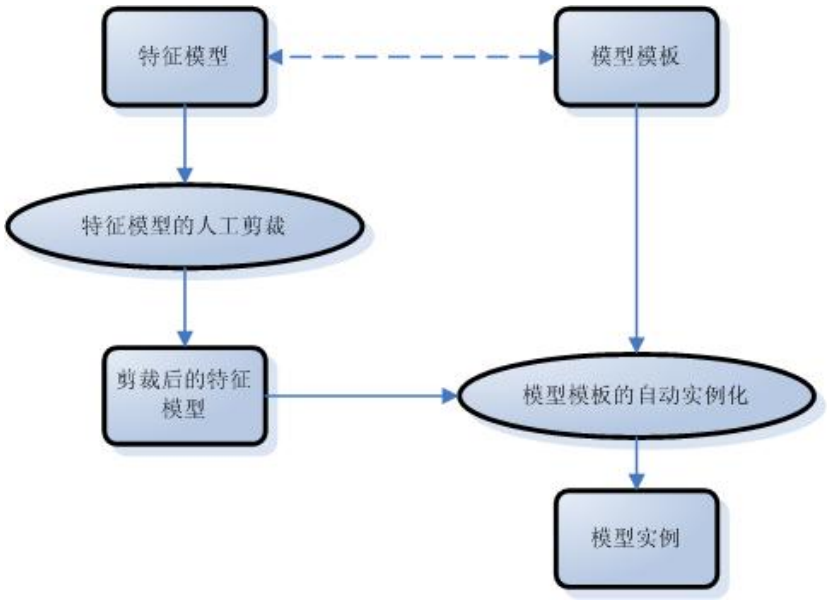

图 3：Czarnecki 方法概览

2.3 基于 XML 的变化性管理方法

Wong 在[WONG 01]中提出了一种基于 XML 技术对领域模型的变化性进行管理的方法，其基本思想是提供一种变化性配置语言（XVCL, XML-based Variant Configuration Language），利用 XVCL 对使用 XML 表示的领域模型中的变化点及变化点间的依赖进行标记，并提供一种操作能力进行领域模型的实例化。

XVCL 实际上是一种文本操作语言，这种语言用一组 XML tag 组成，通过调整这些 tag，就可以操作 XVCL 组织的文本。

Wong 方法和 Czarnecki 方法本质上是相同的，区别在于 Czarnecki 方法通过模型提供了一种将变化性进行图形表示的方法，特别是分色的方法，对于建模与实例化过程的人工参与有帮助，并且用 PC 和 ME 定义特征与目标模型关系也非常易用。而 Wong 方法等于是在技术上提供了建立特征模型与目标模型关联的能力，只要将目标模型表示成结构化文本（XML），就能通过 XVCL 对其进行包装和操作。

3 . FARMS 方法

3.1 FRAMS 基本思想

在上述方法与理论的基础上，我们提出了一种基于特征模型和模型映射的 SOA 变化性管理方法（FARMS）。

FARMS 利用特征模型进行变化性管理，将服务模型和 BPEL 作为目标模型，建立服务模

型模板与 BPEL 模板，通过 PC 和 ME 建立特征模型与服务模型及 BPEL 的关联，这种关联是通过利用 XVCL 对服务模型定义和 BPEL 定义的 XML 进行重组织来实现的。

这里 PC 和 ME 是用特征来表达的，PC 的取值决定了相应模板元素将在实例的模型中存在与否，而 ME 用来计算一些属性的名或值；复杂的 PC 和 ME 可以采用 X-Path 来表示。基于 XVCL 的文本操作能力，在 FARMS 中通过该语言对模型 XML 文件进行组织来管理其变化性。

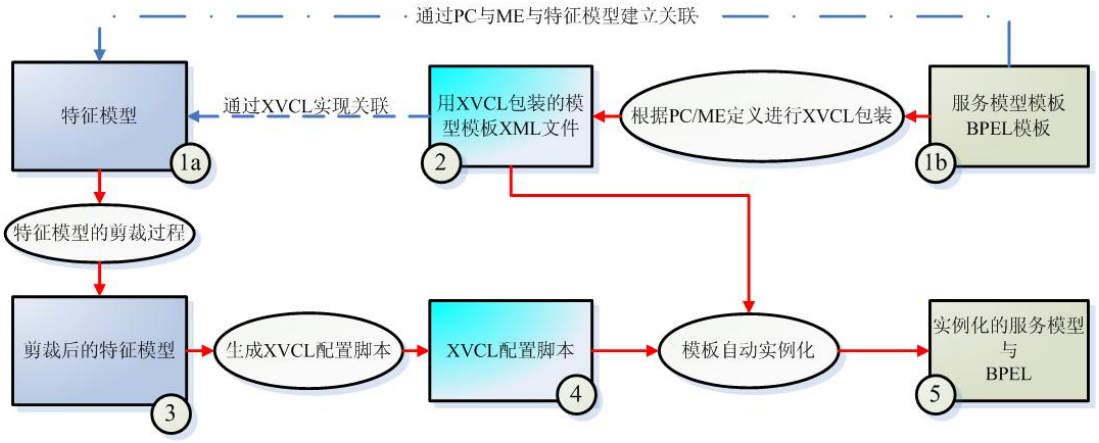

图 4：FARMS 方法概览

图 4显示了FARMS方法的基本元素与过程，其中椭圆代表活动，方框代表文档或模型制品，实箭头代表活动路径，序号代表制品生成的序列。在FARMS中首先应该建立特征模型、服务模型模板和BPEL模板，然后通过在模板中定义PC/ME建立与特征模型的关联，这种关联是通过对于服务模型模板和BPEL模板XVCL包装来实现的。特征模型在剪裁后，生成一个基于XVCL的配置脚本，利用这个配置脚本就可以自动生成服务模型与BPEL实例了。

在 SOA 环境下，利用 FARMS 方法，当需求发生改变时，在一定范围内，只需要重新剪裁特征模型，就可以自动得到相应的运行配置。而特征模型是一种非常有效的需求组织工具，利用特征模型可以直接和业务人员打交道，因此 FARMS 方法具有很强的可应用性。

3.2 服务模型模板

图 5 是一个示例特征模型，后面的例子会基于这个特征模型。图中表达的是一个数据同步的功能，它包括“同步产品信息”和“同步订单信息”三个可选的子特征，同时包括一个“同步触发”的子特征，这个子特征又包括两个可选的子特征，分别是“固定时间间隔更新”和“请求触发”。

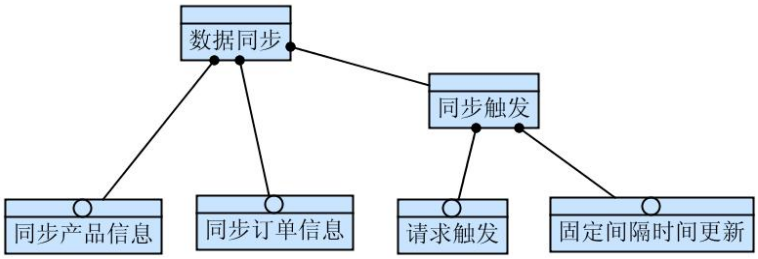

图 5：特征模型示例

服务模型模板实际上可以理解成是一个扩大化的服务模型，即不仅定义出了当前需要有服务，还定义了将来可能用到的服务，并定义这些服务间的关系。图 6 左半部分是一个服务

模型模板示例，其中虚线边的圆圈代表概念上的服务，图中表示出了服务和继承关系，服务间的依赖关系也在模型中体现出来了。通过模型元素的 PC 和 ME，这一模板与特征模型建立了关联，PC 在图上是通过着色来体现的。

图 6的右半部分是当特征模型中“同步产品信息”和“手动触发”两个特征被绑定时的实例化结果，可以发现“数据同步”这个虚服务被去掉了。FARMS规定，虚服务是在模板中用于帮助理解的元素，在实例化时可以去掉。

另外，产品数据同步这个服务的名字成了“手动触发的产品数据同步”，这是因为“手动触发”特征被绑定，这一服务的ＭＥ计算得到了新的名字。表 1是ＭＥ列表，这里只是示意性表示，在后文会对ＭＥ的实现进行详述。

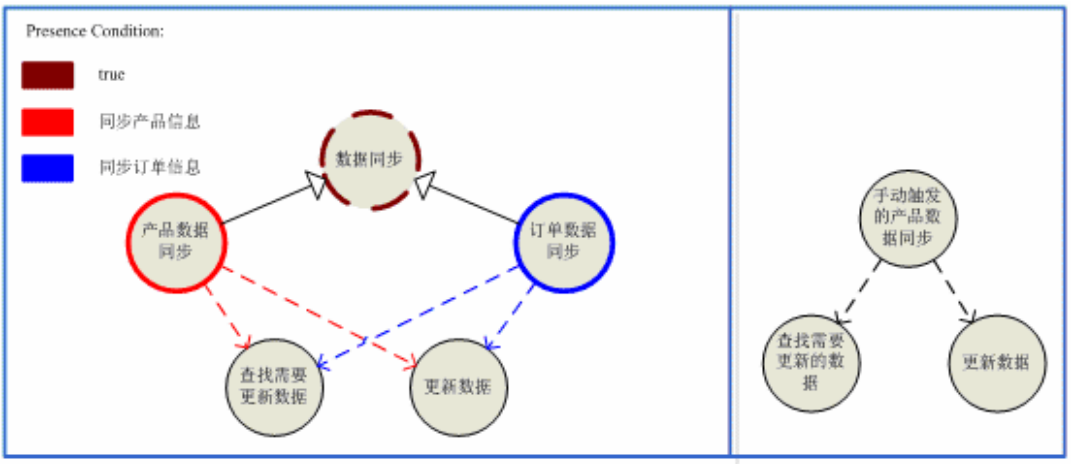

图 6：服务模型模板及实例化示例

编　号	服　务	Meta-expressing
1	产品数据同步	If 手动触发　改名为:“手动触发的产品数据同步” Else if 固定时间间隔更新　改名为:“固定时间间隔更新的产品数据同步”
2	订单数据同步	If 手动触发　改名为:“手动触发的订单数据同步” Else if 固定时间间隔更新　改名为:“固定时间间隔更新的订单数据同步”

表 1：服务模型模板的ＭＥ

另外可以看出，服务和某些特征很多时候是一种简单的对应关系。因此在有些场景下，使用列表来表示服务模板也是一个选择，表 2是一个列表表示的例子。使用这种简单的列表会丢失一些信息，但在简单的场景下已以够用了。实际应用中可以考虑图形模板与列表模板结合使用。

编　号	服　务	Presence Condition
1	产品数据同步	同步产品信息
2	订单数据同步	同步订单信息

表 2：服务模型模板的列表示例

注意在图 6 中一些图形元素没有具体指定PC，解释这个问题需要引进IPC的概念，IPC是隐式存在条件的缩写（Implicit Presence Condition），IPC是特定于目标模型，由目标模型本身特性所决定的PC，引入IPC有助于减少建模工作量，简化模板表示。因此对于在图 6中未彩色标注的图元，其PC表达式可以通过其它已标识图元的PC表达式推出。表 3给出了服务模型模板的IPC。

模型元素类型	IPC
Generalization	子结点或父结点的 PC 都为 true，则 IPC 为 true
Dependency	依赖方任何一个结点的 PC 为 true，且被依赖方任何一个结点的 PC 为 true，则 IPC 为 true
Service	如果是一个父结点，任何一个子结点的 PC 为 ture，则 IPC 为 true； 如果是一个被依赖方，那个依赖的 PC 为 true，且对应这一依赖至少一个依赖方的 PC 为 true，则 IPC 为 true

表 3：服务模型模板的 IPC

3.3 BPEL 模板

上小节图 6中的“产品数据同步”与“订单数据同步”两个服务实际上是组合服务，通过对下面两个基础服务的不同组合分别可以得到“手动触发“和”固定时间间隔更新“两种更新模式，这种组合及其变化性是通过 B P E L 模板来表示的。

BPEL 模板本身就是一个 BPEL，只是一些可能的变化也构造在一起了，其取舍及实例化后的一致性通过特征模型来保证。

图 7是左半部分一个BPEL模板的例子，这一BPEL模板也是和上一小节中所给特征模型关联的。通过剪裁特征模型来计算相应的PC值，就可以得到实例化的BPEL。图 7中右半部分就是当绑定了“主动触发”特征后的一个BPEL图。

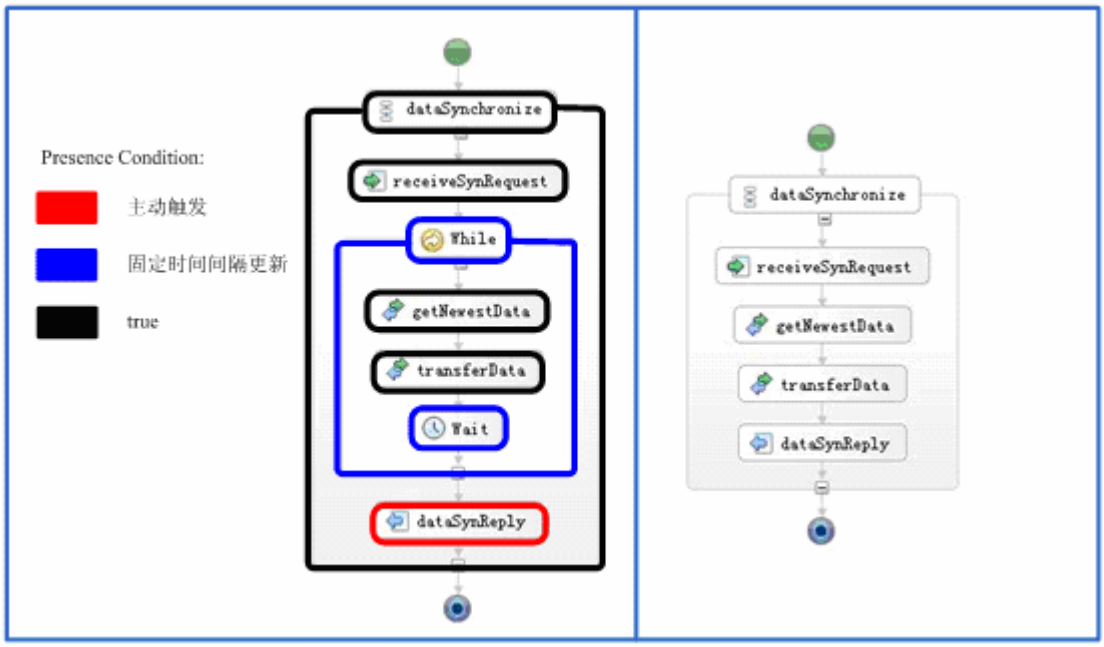

图 7：BPEL 模板及实例化示例

BPEL 模板有时会遇到这样的问题，就是一个模板可能需要在一次配置中就实例为多个不同的模型实例，比如，客户需要针对不同的数据产生不同的数据同步 BPEL，针对一种数据同步，客户也可能还希望既存在一个服务能每请求一次便同步一次，还存在一个服务能够一直运行，定时同步。在服务模型模板中，也可能会出现类似的问题。

对于模板的多实例问题，需要在特征模型中就提供支持，[CZA05C]中提出的特征子树克隆解决了这一问题，图 8是上文中特征模型（可理解为完整特征模型的一个子树）的克隆后的剪裁实例，这三个克隆并剪裁后的子树便可以分别实例化出三个BPEL模型。

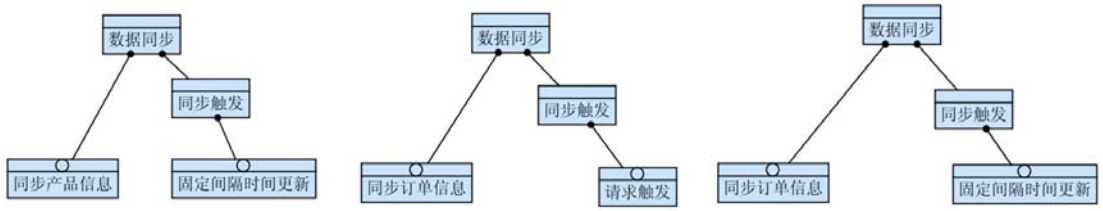

图 8：特征子树克隆后的剪裁示例

现在还存在的问题是[CZA05C]并没有解决全局约束，特别是一些复杂约束在克隆时如何处理的问题。FARMS 中约定，特征子树克隆后，每个特征都要重新命名，内部约束要完全克隆过去。而对于简单的全局约束，FARMS 采取以下处理准则：

- 可克隆子树内特征对外部特征的依赖或互斥关系在克隆时保留
- 尽量避免外部特征对可克隆子树内特征的依赖，尽量将这种依赖向祖先特征转移
- 如果需要保留上述依赖，在克隆时需要依据语义环境决定是否保留

对于复杂的全局约束，如组约束，FARMS 采取以下处理准则：

- 如果不是必须，避免在可克隆子树内出现与外部特征的组约束
- 如果有采用组约束的必要，考虑增加虚结点，将组约束转化为一般的全局约束

最后一个问题，BPEL 模板实例化为多个模型实例，这些实例的区别可能不仅仅或根本就不是模型图的区别，而是其它一些属性的区别，比如所调用服务的不同，wait 操作的时间参数不同。这种问题是 ME（Meta-expressing）所要解决的，即属性名或值的计算，但 BPEL 和服务模型中没有很好的标注机制来标识这些变化点，FARMS 采取直接操作服务模型和 BPEL 的 XML 文件的办法来解决，这部分内容的细节将在下一节中阐述。

最后，表 4给出了BPEL模板的IPC列表。

模型元素类型	IPC
Sequence	该容器内至少一个结点的 PC 为 ture，则 IPC 为 true
Switch	该容器内至少一个 caser PC 为 true，则 IPC 为 true
Pick	至少一个 onMessage 的 PC 为 true，则 IPC 为 true
Scope	该容器内至少一个结点的 PC 为 ture，则 IPC 为 true
Flow	该容器至少有两个或以上的工作流的 PC 为 true，则 IPC 为 true

表 4：BPEL 模板的 IPC 列表

3.4 基于 XML 文件操作的模型映射实现

服务模型和 BPEL 实质上都是用 XML 文件来表示的，FARMS 方法实际上也是直接对 XML 文件进行操作。

直接对 XML 文件进行操作有两个方面原因：

- 出于实现上的原因，特征模型与两种目标模型的映射以及自动实例化，通过对 XML 文件的直接操作来实现是很显而易见的；
- 特征模型与两种目标模型间，除了结构上的映射，还有一些关于名称与值的映射，而两种目标模型的图形表示中都没有合适的标注机制，所以需要对 XML 文件操作来实现。

两种目标模型的变化性实际就体现为XML文件的变化性，我们引入一种XVCL（XML-based Variant Configuration Language）语言[WONG01]来管理这种变化性，一个用XVCL命令组织的文档称为一个X-Frame。表 5列出了一些重要的XVCL命令，FARMS方法中也只使用到了这些命令。

XVCL 命令	描　述
<X-FRAME name=”name”> </X-FRAME>	标记一个 X-Frame
<DECLARATION> </DECLARATION>	声明全局变量及其赋值
<BODY> </BODY>	标记 XVCL 配置命令发挥作用的区域
<COPY x-frame=”x-frame”> Customization commands </COPY>	将一个 X-Frame 用配置命令配置后拷贝到这里
<INSERT-BEFORE name=“breakpoint’> </INSERT-BEFORE> <INSERT name=“breakpoint’> </INSERT> <INSERT-AFTER name=“breakpoint’> </ INSERT-AFTER >	允许在断点（breakpoint）处加入内容，可以加在断点前或断点后，也可以取代断点处的内容
<BREAK name=“breakpoint”> </BREAK>	定义 X-Fram 中的一个断点
<SET name=“varname” value=“varvalue”> </SET>	声明一个 XVCL 变量及其值
<VAR name=“varname”/>	标记一个 XVCL 变量
<SELECT name=“variable”> <OPTION value=“value”> </OPTION>	基于变量的值选择配置项

表 5：主要的 XVCL 命令

图 9 是一个将 BPEL 模板表示为用 XVCL 包装的 XML 文件的例子（考虑篇幅原因，这里给出的 BPEL 文件是经过简化处理的），它包括 3 个 X-Frame 文件，这个 BPEL 模板其实就是上一小节图 7 中左关部分 BPEL 模板的 XML 规约，图中蓝色加粗字体为 XVCL 标记。只

要定义好了 PC 和 ME，这个包装过程可以自动完成。在 FARMS 中，我们规定，容器和流程本身的 PC 使用 XVCL 中的 SELECT 与 OPTION tag 来标记，而其它元素的 PC 用 BREAK 和 COPY&INSERT tag 来标记；而ＭＥ则通过定义变量和规定变量的赋值来实现。

```
<X-FRAME name=”dataSynProcessTemplate.xml”>
<DECLARATION>
      <VAR name=“processName”/>
<VAR name=”watiTime”/>
</DECLARATION>
<BODY>
<SELECT name=”DATASYN”/>
<OPTION value=”NEED_DATASYN”>
<process name=”<VAR name=“processName”/>”
<partnerLinks>
<partnerLink name="dataSource"/>
</partnerLinks>
<variables>
<variable name="dataTable"
messageType="lns:dataTableMessage"/>
</variables>

<sequence>
    <receive>
      </receive>
<SELECT name=”LONGRUN”/>
<OPTION value=”REALTIME_REPETCHECK”>
      <while condition=”true”>
</OPTION>
<invoke>   </invoke>
<invoke>   </invoke>
<BREAK name=”WAIT_POINT”>
<SELECT name=”LONGRUN”/>
<OPTION value=”REALTIME_REPETCHECK”>
      </while>
</OPTION>
<BREAK name=”REQUEST_REPLY”>
</sequence>
</process>
</OPTION>
</BODY>
</X-FRAME>
<X-FRAME name=”waitPoint.xml”>
<BODY>
<wait>for=”<VAR name=”watiTime”/>”
```

```
</wait>
</BODY>
</X-FRAME>
<X-FRAME name=”synReply.xml”>
<BODY>
    <reply >
        </reply>
</BODY>
</X-FRAME>
```

图 9：用ＸＶＣＬ包装的ＢＰＥＬ模板示例

表 6和表 7是BPEL结构元素的PC定义，按照前文所述分成两类。

SELECT 名	PC 为 true 时的赋值	Presence Condition (PC)
DATASYN	NEED_DATASYN	数据同步
LONGRUN	REALTIME_REPETCHECK	固定时间间隔更新

表 6：流程本身和容器类元素的 PC 定义

断　点	Presence Condition (PC)	应导入的 X-Frame 文件
WAIT_POINT	固定时间间隔更新	waitPoint.xml
REQUEST_REPLY	请求触发	synReply.xml

表 7：其它结构元素的 PC 定义

表 8是这个案例的ME表示。

序号	变量名	Meta-Expression (ME)
1	processName	dataType+”_”+”SynProcess”
2	watiTime	“H”+固定时间间隔更新. interval_time
备注	为了方便这里的表达，定义一个临时字符串变量：dataType 当特征“同步订单信息”被绑定时，dataType=”order”； 当特征“同步产品信息”被绑定时，dataType=”product”； ME 栏表达式按照 Java 中的字符串操作规则	

表 8：ME 表示

通过PC和ME，就建立了特征模型和BPEL模板的关联，当特征模型剪裁后，便可以根据相应的PC和ME，生成一个配置脚本，图 10是仅选择了“同步产品信息”和“请求触发”两个可选特征后的配置脚本。

```
<customization>
    <SET name=“processName” value=”product_SynProcess”/>
<SET name=” DATASYN” value=” NEED_DATASYN”/>
```

```
<SET name=” LONGRUN” value=”  ”/>
<COPY x-frame=” synReply.xml”>
    <INSERT name=” REQUEST_REPLY “>
</COPY>
</customization>
```

图 10：ＸＶＣＬ配置脚本示例

解释器根据配置脚本相应X-Frame文件进行操作，便得到了一个实例化的BPEL。图 11是根据前面所给配置脚本得到的实例BPEL。

```
<process name=” product_SynProcess”
<partnerLinks>
<partnerLink name="dataSource "/>
</partnerLinks>
<variables>
<variable name="dataTable "
messageType="lns:dataTableMessage"/>
</variables>

<sequence>
    <receive>
    </receive>
    <invoke>
    </invoke>
<invoke>
</invoke>
<reply >
</reply>
</sequence>
</process>
```

图 11：ＢＰＥＬ实例化示例

3.5 模板实例化

在[CZA05A]方法中，提出了一些方法用于对实例化后的模型进行简化，由于 BPEL 与普通活动图的定义有一定的差别，而且 FARMS 方法直接对 XML 文件进行操作，[CZA05A]中提出的简化措施基本上都不适用也不需要了。

FARMS 方法的模板实例化发生在特征模型剪裁之后，其具体过程包括：

⑴．计算所有 ME 和显式 PC 的值

⑵．计算所有 IPC

⑶．生成 XVCL 配置脚本

⑷．利用配置脚本实例化模板的 XML 文件

4. FARMS 方法在凤凰公司整合项目中的应用

4.1 特征模型

图 12是我们为凤凰公司ERP与CRM整合项目建立的特征模型，图中带小圈的特征是可选特征；带小角的特征是特征的引用，用以将规模太大的特征树分解和处理一个特征在不同视图中的出现；特征属性另外定义，未出现在图中。

特征模型按照视图来组织，以下为各个视图的简单解释：

- 视图(1)：顶层视图
- 视图(2)：订单生成流程部分
- 视图(3)：订单审批部分
- 视图(4)：订单冻结部分
- 视图(5)：数据同步部分，这里“数据同步”旁边的“[1，18]”表示“数据同步“特征可以克隆成为 1 到 15 个特征子树。
- 视图(6)：全局约束部分，定义了一些依赖关系（虚线箭头）和组约束，小圈中有 S 的组约束代表组中有且只能有一个特征可以被绑定。

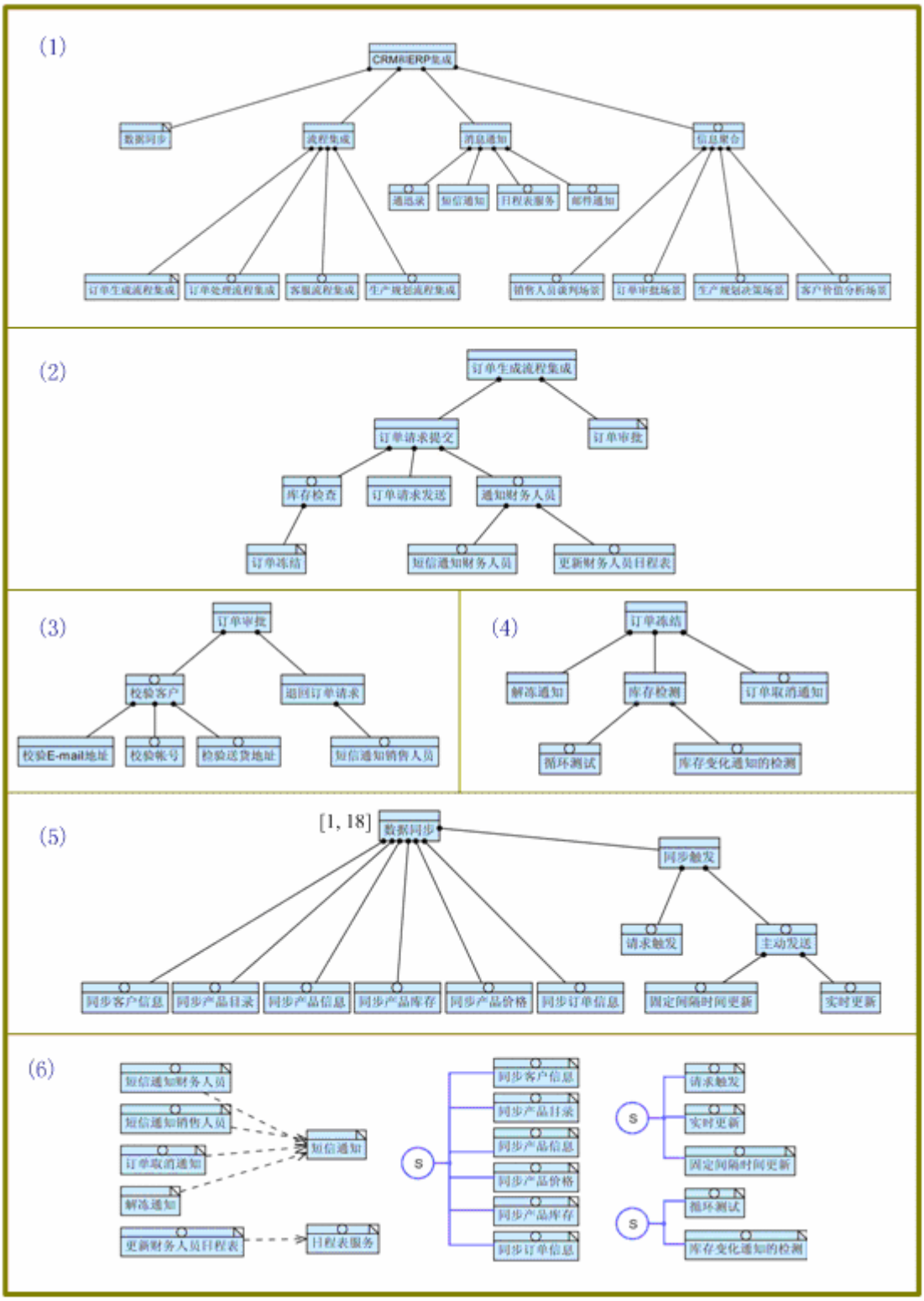

图 12：凤凰公司整合项目的特征模型

4.2 服务模型模板

由于篇幅关系，我们只列出针对数据同步子特征树所作的工作，前文中出现的例子实际上就是这部分工作的片断。同样出于篇幅的原因，我们在本文中略去了服务模型的 XML 文

件部分。

图 13是服务模型模板，表 9是其PC的列表表示，表 10是ME的列表。

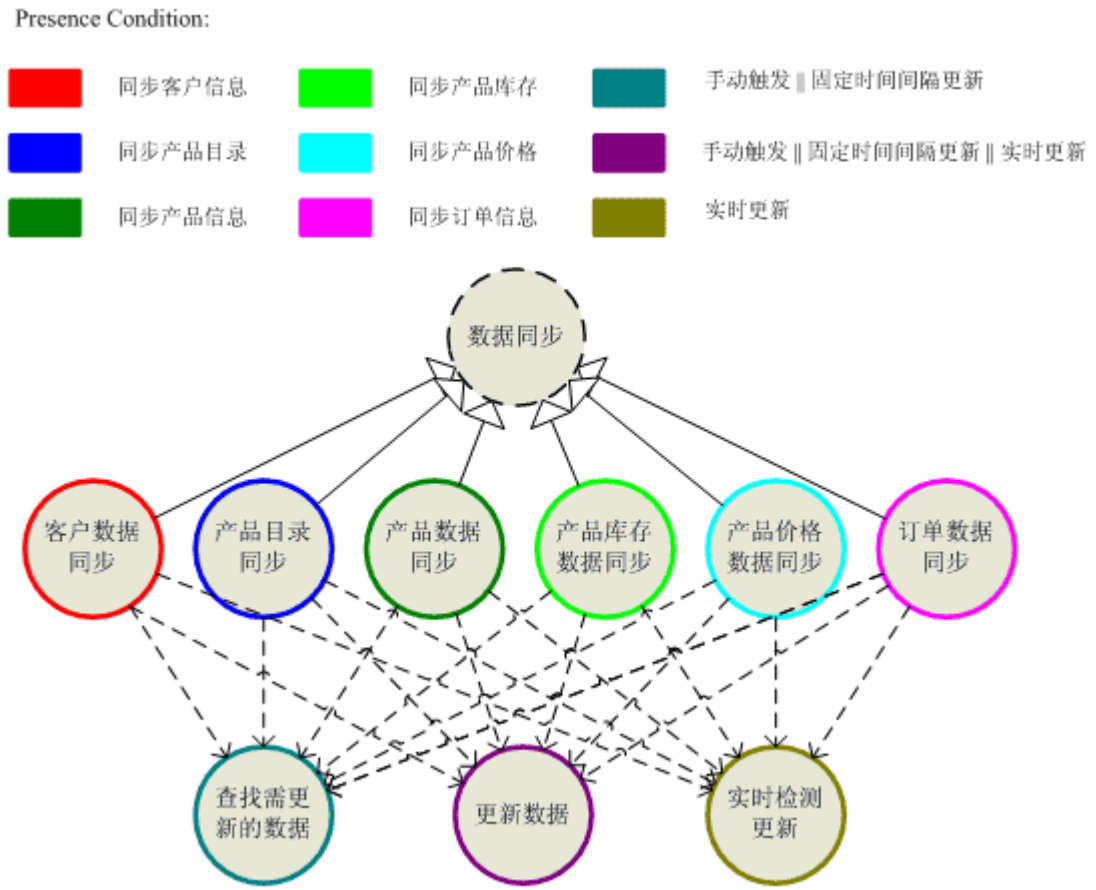

图 13：数据同步的服务模型模板

编　号	服　务	Presence Condition
1	客户数据同步	同步客户信息
2	产品目录同步	同步产品目录
3	产品数据同步	同步产品信息
4	产品库存数据同步	同步产品库存
5	产品价格数据同步	同步产品价格
6	订单数据同步	同步订单信息
7	查找需更新的数据	手动触发 || 固定时间间隔更新
8	更新数据	手动触发 || 固定时间间隔更新 || 实时更新
9	实时检测更新	实时更新

表 9：数据同步服务模型模板的列表表示

序号	服务名	Meta-Expression (ME)
1	客户数据同步	synType + “客户数据同步”
2	产品目录同步	synType + “产品目录同步”
3	产品数据同步	synType + “产品数据同步”
4	产品库存数据同步	synType + “产品库存数据同步”
5	产品价格数据同步	synType + “产品价格数据同步”
6	订单数据同步	synType + “订单数据同步”
备注	为了方便这里的表达，定义一个临时字符串变量：synType 当特征“手动触发”被绑定时，synType=”手动触发的”； 当特征“固定时间间隔的”被绑定时，synType=”固定时间间隔的”； 当特征“实时更新”被绑定时，synType=”实时更新的”； ME 栏表达式按照 Java 中的字符串操作规则	

表 10：服务模型的ME定义

4.3 BPEL 模板

图 14是数据同步的BPEL模板，图 15是其XML文件的XVCL包装，表 11到表 13是相关PC和ME的列表。

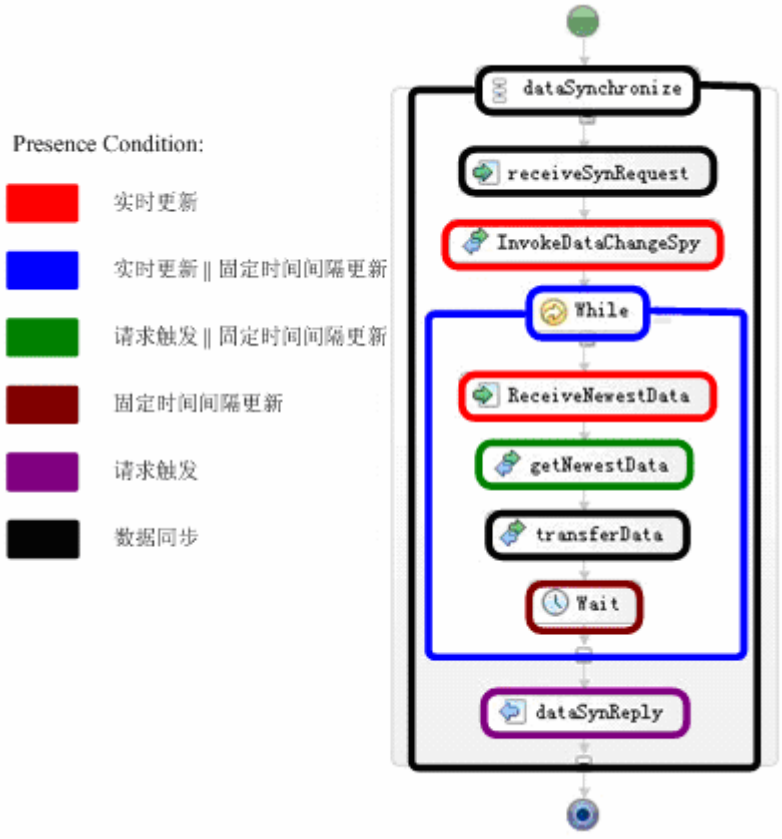

图 14：数据同步的 BPEL 模板图

```
<X-FRAME name=”dataSynProcessTemplate.xml”>
<DECLARATION>
      <VAR name=“processName”/>
<VAR name=”watiTime”/>
</DECLARATION>
<BODY>
<SELECT name=”DATASYN”/>
<OPTION value=”NEED_DATASYN”>
<process name=”<VAR name=“processName”/>”
targetNamespace="http://mozart.pku.edu.cn/dataSyn"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:lns="http://mozart.org/wsdl/data-syn
suppressJoinFailure="yes">
<partnerLinks>
<partnerLink name="dataSource"
partnerLinkType="dataSourceLinkType"
partnerRole=dataSourceRole”
myRole="dataSourceMyRole"/>
            <partnerLink name="dataTarget"
partnerLinkType="dataTargetLinkType"
partnerRole=” dataTargetRole”
myRole="dataTargetMyRole"/>
<partnerLink name= “synRequester”
partnerLinkType="synRequesterLinkType"
myRole="synRequesterRole"/>
</partnerLinks>
<variables>
<variable name="synRequest"
messageType="lns:synRequestMessage"/>
<variable name="replyOK"
messageType="lns:replyOKMessage"/>
<variable name="dataTable"
messageType="lns:dataTableMessage"/>
</variables>
<sequence>
      <receive partnerLink=”synRequester”
portType="lns:synRequestPT"
operation="synStartRequest"
variable=" synRequest " createInstance="yes">
      </receive>
      <BREAK name=”REALTIME_INVOKE”>
<SELECT name=”LONGRUN”/>
<OPTION value=”REALTIME_REPETCHECK”>
```

```
      <while condition=”true”>
</OPTION>
            <BREAK name=”REALTIME_RECEIVE”>
            <BREAK name=”GET_NEWDATA”>
<invoke partnerLink="dataTarget"
portType="dataTargetTransferPT"
operation="dataTableTransfer"
inputVariable="dataTable">
</invoke>
<BREAK name=”WAIT_POINT”>
<SELECT name=”LONGRUN”/>
<OPTION value=”REALTIME_REPETCHECK”>
      </while>
</OPTION>
<BREAK name=”REQUEST_REPLY”>
</sequence>
</process>
</OPTION>
</BODY>
</X-FRAME>
<X-FRAME name=”realtimeInvoke.xml”>
<BODY>
<invoke partnerLink=dataSource"
portType=”dataSourceSpyPT"
operation="dataChangeSpy"
inputVariable="synRequest">
</invoke>
</BODY>
</X-FRAME>
<X-FRAME name=”realtimeReceive.xml”>
<BODY>
<receive partnerLink=” dataTarget”
portType=”dataSourceSpyCBPT"
operation="sendNewestData"
variable="dataTable" createInstance="no">
</receive>
</BODY>
</X-FRAME>
<X-FRAME name=”getNewData.xml”>
<BODY>
<invoke partnerLink=”dataSource"
portType="dataSourceGetNewDataPT"
operation="getNewestData"
```

```
inputVariable=”synRequest”
outputVariable="dataTable">
</invoke>
</BODY>
</X-FRAME>
<X-FRAME name=”waitPoint.xml”>
<BODY>
<wait>for=”<VAR name=”watiTime”/>”
</wait>
</BODY>
</X-FRAME>
<X-FRAME name=”synReply.xml”>
<BODY>
      <reply partnerLink=”synRequester”
portType="lns:synRequestPT"
operation="synStartRequest"
variable="replyOK ">
            </reply>
</BODY>
</X-FRAME>
```

图 15：数据同步的 BPEL 模板，用 XVCL 包装的 XML 文件

SELECT 名	PC 为 true 时的赋值	Presence Condition (PC)
DATASYN	NEED_DATASYN	数据同步
LONGRUN	REALTIME_REPETCHECK	实时更新 ‖ 固定时间间隔更新

表 11：流程本身和容器类元素的 PC 定义

断 点	Presence Condition (PC)	应导入的 X-Frame 文件
REALTIME_INVOKE	实时更新	realtimeInvoke.xml
REALTIME_RECEIVE	实时更新	realtimeReceive
GET_NEWDATA	请求触发 ‖ 固定时间间隔更新	getNewData.xml
WAIT_POINT	固定时间间隔更新	waitPoint.xml
REQUEST_REPLY	请求触发	synReply.xml

表 12：其它结构元素的 PC 定义

序号	变量名	Meta-Expression (ME)
1	processName	dataType+"_"+"SynProcess"
2	watiTime	"H"+固定时间间隔更新. interval_time
备注	为了方便这里的表达，定义一个临时字符串变量：dataType 当特征"同步客户信息"被绑定时，dataType="customer"； 当特征"同步产品目录"被绑定时，dataType="product_dir"； 当特征"同步产品信息"被绑定时，dataType="product"； 当特征"同步产品库存"被绑定时，dataType="product_stock"； 当特征"同步产品价格"被绑定时，dataType="product_price"； 当特征"同步订单信息"被绑定时，dataType="order"； ME 栏表达式按照 Java 中的字符串操作规则	

表 13：ME 定义

4.4 实例化分析

我们的方案就是按照 FARMS 方法进行建模，并根据现阶段的分析进行实例化所得到的结果。限于篇幅，本文档只给出了示例性的实例化结果。

图 17是服务模型的一个实例化结果，它是根据图 16中特征模型的剪裁结果得到的。

图 19是BPEL的一个配置脚本，它是根据图 18中特征模型的剪裁结果得到的，图 20是执行配置脚本得到的实际BPEL，图 21是实例化后的BPEL图。

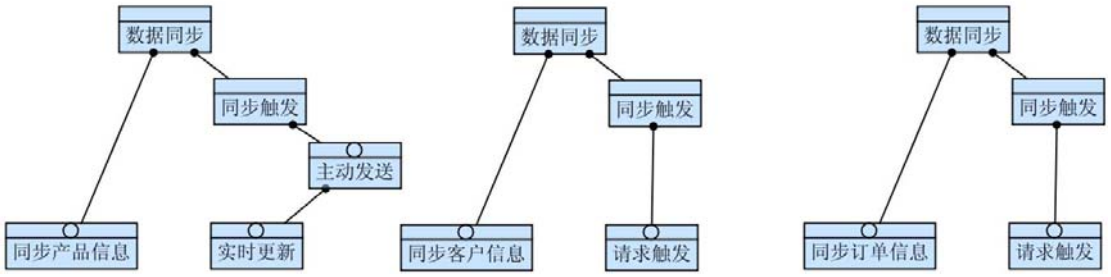

图 16：数据同步子特征树的一次剪裁结果，克隆了三次

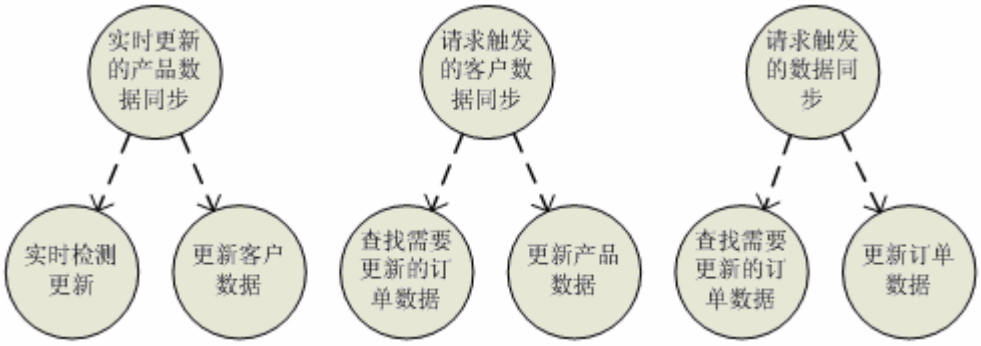

图 17：根据上面剪裁结果得到的实例化结果

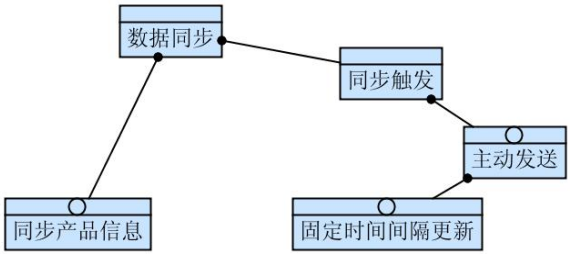

图 18：数据同步子特征树的又一种剪裁结果

```
<customization>
      <SET name=“processName” value=”product_SynProcess”/>
<SET name=”watiTime” value=”H2”/>
<SET name=” DATASYN” value=” NEED_DATASYN”/>
<SET name=” LONGRUN” value=” REALTIME_REPETCHECK”/>

<COPY x-frame=”getNewData.xml”>
      <INSERT name=”GET_NEWDATA “>
</COPY>
<COPY x-frame=” waitPoint.xml”>
      <INSERT name=” WAIT_POINT “>
</COPY>
</customization>
```

图 19：根据图 18 的剪裁结果生成的配置脚本

```
<process name=” product_SynProcess”
targetNamespace="http://mozart.pku.edu.cn/dataSyn"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:lns="http://mozart.org/wsdl/data-syn
suppressJoinFailure="yes">

<partnerLinks>
<partnerLink name=" dataSource"
partnerLinkType="dataSourceLinkType "
partnerRole=” dataSourceRole”
myRole="dataSourceMyRole "/>
          <partnerLink name="dataTarget "
partnerLinkType=" dataTargetLinkType "
partnerRole=” dataTargetRole”
myRole=" dataTargetMyRole"/>
<partnerLink name= “synRequester”
partnerLinkType="synRequesterLinkType"
myRole="synRequesterRole"/>
</partnerLinks>
```

```
<variables>
<variable name="synRequest"
messageType="lns:synRequestMessage"/>
<variable name="replyOK"
messageType="lns:replyOKMessage"/>

<variable name=" dataTable"
messageType="lns:dataTableMessage"/>
</variables>

<sequence>
      <receive partnerLink=”synRequester”
portType="lns:synRequestPT"
operation="synStartRequest"
variable="synRequest" createInstance="yes">
      </receive>
      <while condition=”true”>
            <invoke partnerLink=" dataSource "
portType=" dataSourceGetNewDataPT "
operation="getNewestData"
inputVariable=”synRequest”
outputVariable=" dataTable">
</invoke>

<invoke partnerLink=" dataTarget"
portType=" dataTargetTransferPT "
operation="dataTableTransfer"
inputVariable=" dataTable">
</invoke>

<wait>for=”<VAR name=”watiTime”/>”
</wait>
      </while>
</sequence>
</process>
```

图 20：根据图 20 中的配置脚本得到的 BPEL 实例

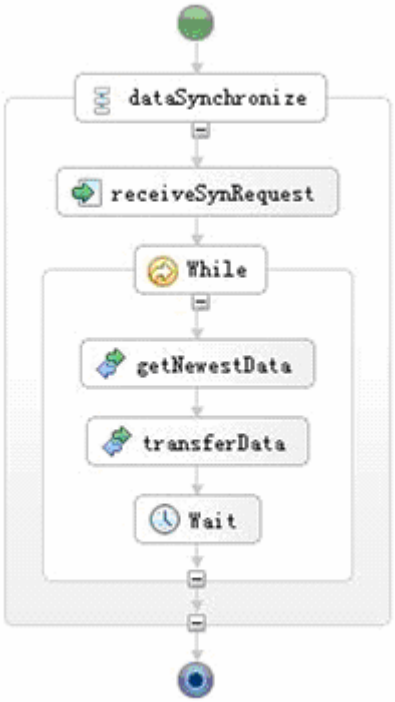

图 21：与图 20 中 XML 文件对应的 BPEL 图

5. 支持工具的实现分析

图 22是FARMS支持工具整体架构。

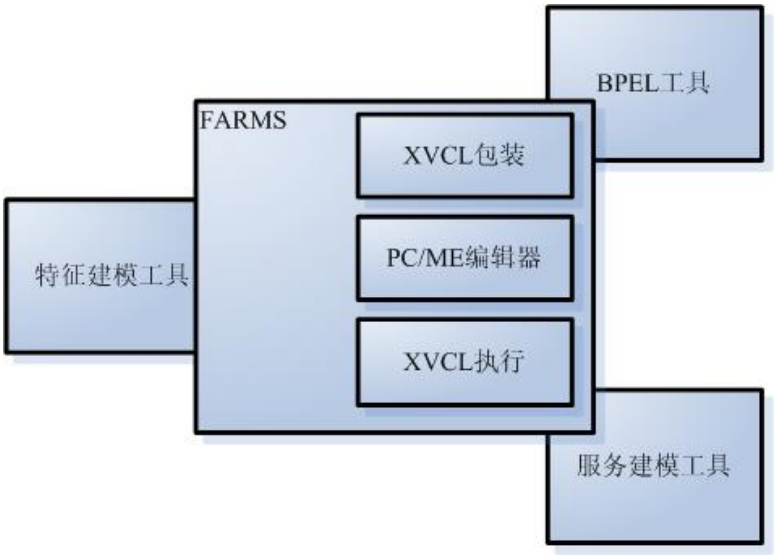

图 22：FARMS 的整体结构

FARMS 支持工具需要集成现在的特征建工具和服务建模工具和 BPEL 工具，另外需要增加以下几个模块：

- PC/ME 编辑器：用于编辑ＰＣ和ＭＥ表达式；

- XVCL 包装：用于将定义好了ＰＣ和ＭＥ的模型文件进行包装，并根据剪裁后的特征模型生成配置脚本；
- XVCL 执行：根据配置脚本自动实例化。

6．FARMS 方法的总结

FARMS 方法将几种变化性管理的方法综合应用到 SOA 领域，从凤凰公司的整合案例来看，取得了很好效果，可以作如下总结：

- 通过特征模型来制导服务建模，可以充分考虑到可能的变化，还能利用特征模型本身的优良特性对变化性进行组织与管理；
- FARMS 方法提供的模型映射与自动实例化能力，能够有效管理变化性从需求到设计的传递，减少系统重配置的工作量，提高 SOA 应对变化的能力；
- FARMS 定义了详细的模型映射机制，易于开发相应的工具来辅助进行 SOA 的变化性管理

在特征模型上，FARMS 方法充分借鉴了[CZA05C] 、[ZHANG04]和[MEI06] 的优点，针对ＳＯＡ的特点定义了我们所需要的特征模型与建模方法。

在模型映射上也不仅仅是对 [CZA05A]和[JAR01]工作的简单组合，而是对两种方法进行了充分的整合，并对一些在两个方法中都没有解决的问题提出了我们自己的方案，贡献主要包括：

- 全局约束关系在特征子树克隆时的处理
- 服务模型模板的组织
- BPEL 与服务模型ＩＰＣ的定义
- XVCL 对模板文件进行包装的准则

参考文献:

[CZA05A] K. Czarnecki and M. Antkiewicz. Mapping Features to Models: A Template Approach Based on Superimposed Variants. Proceedings of GPCE'05, 2005

[CZA05B] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged Configuration Through Specialization and Multi-Level Configuration of Feature Models. Software Process Improvement and Practice, special issue on "Software Variability: Process and Management, 10(2), 2005, pp. 143 - 169

[CZA05C] K. Czarnecki, S. Helsen, and U. Eisenecker. Formalizing Cardinality-based Feature Models and their Specialization. Software Process Improvement and Practice, special issue of best papers from SPLC04, 10(1), 2005, pp. 7 – 29

［GRI98] Griss, M. L., Favaro, J. and d'Alessandro, M. (1998), Integrating Feature Modeling with the RSEB, in Proceedings of 5th International Conference on Software Reuse, Victoria, Canada, June, IEEE, pp. 76-85.

[JAR01] Jarzabek, S. and Zhang, H., "XML-based Method and Tool for Handling Variant Requirements in Domain Models", Proc. Of 5th IEEE International Symposium on Requirements Engineering, RE'01, IEEE Press, August 2001, Toronto, Canada, pp. 166-173

[KAL97] Jarmo Kalaoja, Eila Niemela, and Harri Perunka: Feature MOdelling of Component-Based Embedded Software, 8th IEEE International Workshop on Software Technology and Engineering Practive, pp. 444-451, July 1997

[KANG90] Kang, K.C., S.G. Cohen, J.A. Hess, W.E. Novak and A.S. Peterson (1990), "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[KANG98] K. Kang, S. Kim, J. Lee, K. Kim, E. Shin and M. Huh, FORM: A Feature-Oriented Reuse Method with Domain -Specific Reference Architectures , Annals of Software Engineering , 5, 143-168, 1998

[MEI06] Hong Mei, Wei Zhang, and HaiyanZhao: A Metamodel for Modeling System Features and Their Refinement, Constraint and Interaction Relationships. To appear in Journal on Softeare & System Modeling (SoSyM), 2006

[WONG01] Wong, T.W., Jarzabek, S., Soe, M.S., Shen, R. and Zhang, H.Y. "XML Implementation of Frame Processor," Symposium on Software Reusability, SSR '01. Toronto, Canada, May 200 1, pp. 164-172

[ZHANG04] Wei Zhang, Haiyan Zhao, and Hong Mei: A Propositional Logic-Based Method for Verification of Feature Models, 6th International Conference on Formal Engineering Methods (ICFEM'04), Seattle, WA, USA, pp. 115-130,2004.