**问题1.1**

**摘要**

企业在开发非关键任务的Web应用程序时采用的一种新开发风格——Mashup。Mashup应用程序是为了满足某种业务需求而创建的，通常只在该需求存在的短时间内使用。开发Mashup需要检索、理解和组合异构的软件组件，这些组件通常以Web API的形式提供。

论文中提到，Mashup互操作性（Mashup interoperability）是指一组条件，包括技术和组织方面的条件，以便设计师/开发人员能够创建Mashup。在这篇论文中，作者提出了Web API选择模式，作为促进Mashup互操作性的一种贡献。

具体来说，论文的主要内容包括：

1. **Mashup的定义和应用场景**：解释了Mashup是什么，以及它们在企业中的应用场景，特别是用于非关键任务的Web应用程序。
2. **Mashup开发的挑战**：详细描述了在开发Mashup时面临的主要挑战，尤其是如何检索、理解和组合来自不同来源的异构Web API。
3. **Mashup互操作性的概念**：定义了Mashup互操作性，并解释了实现互操作性所需的各种条件，包括技术和组织方面的要求。
4. **Web API选择模式**：提出了一些模式或方法，用于选择适合的Web API，以便更好地实现Mashup互操作性。这些模式有助于开发人员在面对大量可用的Web API时做出更明智的选择。

**介绍**

讨论了企业如何利用企业Mashup（一种新兴的Web应用开发方式）来满足业务需求，特别是在开发非关键任务的Web应用方面。企业Mashup通常是为了应对特定的短期业务需求而开发的，开发投入有限，应用寿命也较短。典型的例子包括企业仪表板，这些仪表板在企业环境中用于改进决策、定位内容、获取和整合信息以及管理任务以支持用户活动。

Mashup应用通常通过利用现有的数据、UI小部件和功能来创建新的应用和软件构件，这些构件也可以在其他Mashup中重用。Mashup组件通常以Web API的形式提供，通过编程耦合来实现应用逻辑。在企业环境中，Mashup可以使用企业内部开发的组件（例如访问客户数据的组件）或第三方组件（如地理编码服务）来实现。

开发者在探索和理解可用API及其关系时面临一些挑战，包括：

1. 可用API数量庞大且动态变化（例如，ProgrammableWeb上有超过3000个API）。
2. Mashup应用的开发时间通常有限。
3. 典型的Mashup开发者技能和经验有限，需要在有限的知识背景下寻找合适的Web API并进行连接。

针对这些问题，论文提出了API选择模式的概念，以主动协助Mashup应用设计师在开发过程中进行选择。论文讨论了这些模式在企业应用开发中的贡献，特别是对软件重用生命周期的影响，并对这一概念进行了形式化。

相关工作方面，许多研究致力于改进Mashup开发的工具设计。例如：

* 通过对非结构化Web API进行分面分类和排名算法的应用，改进了ProgrammableWeb API库的搜索机制。这种分类和搜索解决方案仍基于信息检索技术。
* MatchUp系统解决了Mashup组件链接模式建议的问题：当设计师选择一组组件时，系统基于库中的重复模式建议代码模式来连接这些组件。
* sMash系统提出了一种支持语义增强Web API的Mashup的Web界面。可能的Mashup以图形形式展示，每个顶点代表一个API，两个API之间的边表示它们可以一起用于Mashup。

通过这些研究和工具，开发者可以更高效地探索和利用Web API，从而改进Mashup应用的开发过程。

**企业开发方案**

1. **传统开发的挑战**：IT 部门通常缺乏资源来快速开发和支持短期应用，使得传统开发方法对这些需求不切实际。
2. **Mashup 过程**：

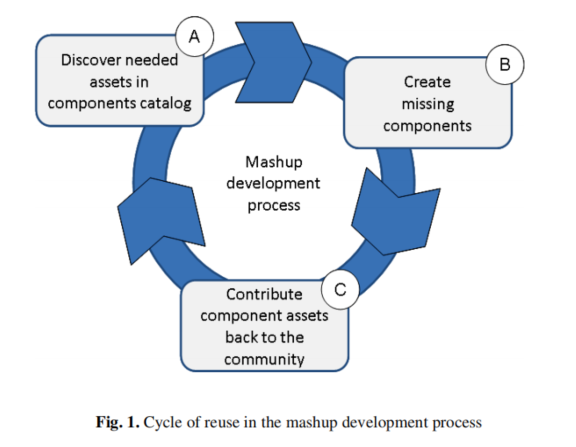
* **IT 部门角色**：创建组件目录并提供无需编码的组装工具。
* **业务用户角色**：业务用户可以利用这些工具和组件自行创建和分享应用，无需 IT 介入。

1. **好处**：

* **对业务用户的好处**：他们获得更多控制权，可以快速独立地满足需求。
* **对 IT 部门的好处**：可以专注于非关键任务的应用，这些应用需要更少的时间和精力。

1. **互操作性**：Mashup 有助于解决企业内整合多样化数据和功能的挑战。
2. **采用原因**：

* 减少不确定性并缩短项目时间。
* 鼓励组件的重复使用。
* 为新情况快速组装应用。



1. **设计挑战**：

* 需要在技术、认证、隐私和集成等不同抽象层次上进行问题解决和设计决策。

1. **API 选择和集成**：

* 论文提出专注于选择和集成 API 和功能。
* **选择模式**：基于以下内容：
* 使用领域本体对 Web API 描述进行语义注释。
* 使用自动匹配技术组织 Web API，根据相似性和耦合标准创建语义链接。

**选择模式**

1. **搜索模式 (Search)**：

* **目的**：为给定的Web API规范（Wτ）建议一组匹配的Web APIs。
* **应用**：帮助设计者在开发过程中找到符合特定需求的API。
* **度量标准 (mτ)**：用于评估API与规范的匹配程度。
* **阈值 (δτ)**：用于筛选掉不相关的API，只有匹配度大于等于该阈值的API才会被建议。
* **排序函数 (≤τ)**：用于对建议的API进行排序。

1. **补全模式 (Completion)**：

* **目的**：为已属于混搭应用M的某个Web API（Wτ）建议可以与之结合使用的API。
* **应用**：帮助设计者找到可以与现有API结合提供补充功能的其他API。
* **度量标准 (mτ)**：用于评估API之间的结合度。
* **阈值 (δτ)**：筛选出结合度高的API。
* **排序函数 (≤τ)**：对建议的API进行排序。

1. **替代模式 (Substitution)**：

* **目的**：为混搭应用M中的某个Web API（Wτ）建议可以替代的API。
* **应用**：在不影响应用功能的前提下，用质量更高或更稳定的API替换现有API。
* **度量标准 (mτ)**：用于评估每个建议API与Wτ的相似度。
* **阈值 (δτ)**：筛选出相似度高的API。
* **排序函数 (≤τ)**：对建议的API进行排序。

**应用到开发过程：**

* **搜索和补全模式**：帮助设计者在混搭应用的不同开发阶段找到匹配的API和可结合使用的API，提高应用的功能性和灵活性。
* **替代模式**：通过替换不再可用、维护或质量低下的API，提高应用的整体质量和稳定性。这样可以促进在用户/开发者社区中分享高质量的混搭应用。

**使用API的语义描述来改进第三方API在Web应用中的选择和集成**

**关键概念：**

1. **语义异构性**：API描述方式的差异，使得集成变得困难。
2. **语义描述**：涉及两个主要情况：

* **现有API**：对已有API进行语义特征描述。
* **请求API**：定义设计者对API的语义需求。

**语义注释过程：**

论文概述了一个基于SWEET工具的框架，用于对API进行语义注释：

* **步骤 (a)**：从API的非结构化HTML文档中识别元素（操作、输入、输出），以创建hRESTS描述。
* **步骤 (b)**：寻找适合的本体用于元素注释和API分类的分类法。
* **步骤 (c)**：使用MicroWSMO符号对API进行注释和分类，提供用于定义相似性和耦合度量的正式描述，这些度量用于推荐模式。

**语义描述符：**

* **用于已注释API (Wi)**：
* **CATi**：与API相关的类别。
* **OPi**：API可以执行的操作。
* **EVi**：API可以生成的事件。
* 每个操作包括操作名称、输入和输出。
* 事件通过表示状态变化的参数来描述。
* **用于API请求 (Wr)**：
* **CATr**：与请求相关的类别。
* **optr**：所需的操作、输入和输出。
* 请求描述符具有更简单的结构，因为输入和输出是独立于操作指定的，反映了探索性搜索的特点。

**目的：**

语义注释和特征描述提供了一种结构化的方式来描述API和请求，便于基于相似性和耦合度量进行自动选择和集成。这种方法通过提供一种正规化的方法来匹配API能力与应用需求，解决了手动选择API的困难。