软件过程元数据PMeta设计文档

1. 为什么需要PMeta

现代软件生产系统不仅仅是一个构建服务器。它是一个不断变化的功能和技术组合，从问题跟踪器和测试框架到环境管理器和存储库。同时，软件供应链变得越来越复杂，在形成最终的软件产品之前，软件工件会多次跨越组织和地理边界。这是一个由相互联系和相互依赖的软件管道组成的世界。

PMeta将同样的架构原则应用于这个管道网络，就像应用于任何其他软件设计问题一样。

PMeta采用了开源的Eiffel协议使系统中的各个角色之间能够进行技术无关的基于事件的通信。开源Eiffel协议和它的实现可以让你对你的软件生产系统和供应链的情况一目了然。

2. 什么是PMeta

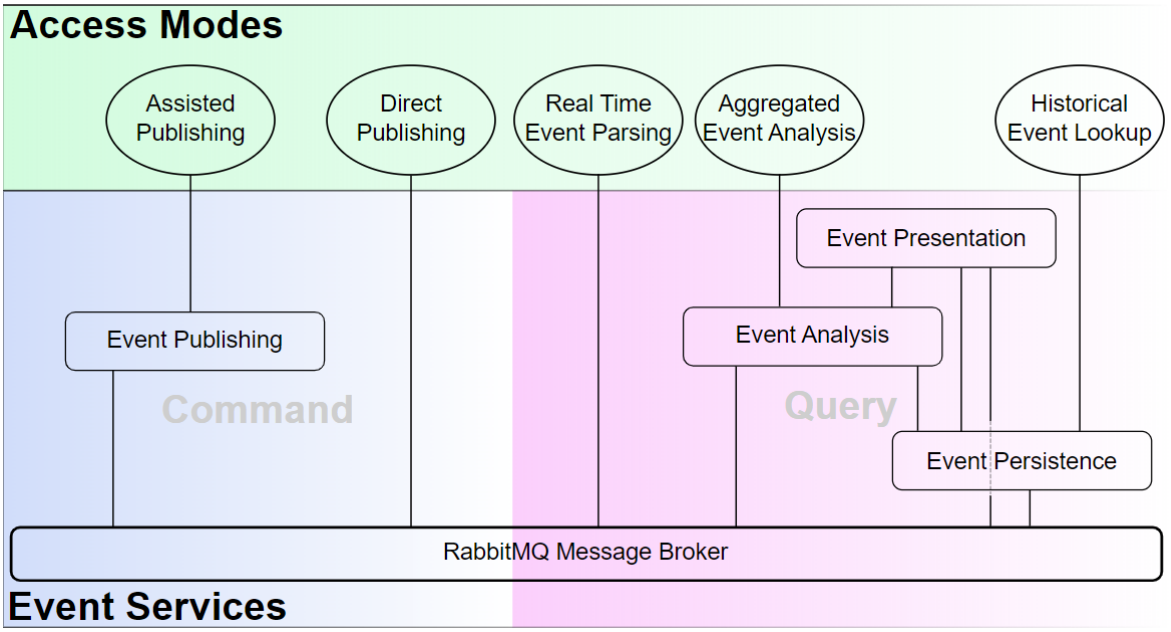
基于Eiffel协议的PMeta框架是一个持续集成和交付的框架，特别是解决企业规模的分布式和异构环境的挑战。它通过现场实时生成全局广播事件来实现这一点 – 这些事件相互参照，形成一个有向无环图（DAG），描述持续集成和交付管道的所有活动，无论它们发生在哪里，使用哪种基础技术，甚至它们是否自动化。

PMeta基本上由两部分组成。第一，Eiffel事件的词汇和语法，形成框架的通信协议。第二，建立在该通信协议之上的服务，以协调持续集成和交付活动，提供可追溯性、仪表盘、可视化和更多。

3. 读者注

- 文档中，不区分“事件” 和 “消息”，它们指的是同一个东西，就是承载软件过程元数据的协议消息
- PMeta框架在Eiffel协议基础上实现，文档中“ PMeta事件/消息” 和 “Eiffel事件/消息”指的是同一个东西

4. PMeta抽象参考架构



4.1. 事件服务 Event Services

事件服务不属于在管道中运行的服务，但它的存在是为了协助PMeta事件的处理。它可以是用于发布（命令）或消费（查询）此类事件的服务。事件服务通常不是PMeta事件数据的出处，而是通过存储、分析、运输和分析事件数据来为管道参与者和/或用户服务。

4.1.1. 事件分析服务 Event Analysis

事件分析服务实现了聚合事件分析的访问模式。该服务基于PMeta事件数据构建新的对象。这些对象可以代表任何类型的实体，并且通常位于比PMeta事件更高的抽象级别。

4.1.2. 事件持久性服务 Event Persistence

事件持久性服务实现了历史事件查询的访问模式。它存储通过 RabbitMQ 消息代理接收的事件（收到的所有事件或过滤的子集），并通过 API 将其公开，以便随后进行检索。

从实现的角度来看，事件分析服务的内部存储和查询机制、数据模型或底层 DBMS 解决方案无关紧要，但它绝对需要尊重其不变性。换句话说，从事件持久性服务中检索的任何事件都应与通过 RabbitMQ 消息代理接收的事件完全一致。

4.1.3. 事件展示服务 Event Presentation

事件展示服务将事件数据或来自事件的数据以某种格式展示给用户使用。通常，这涉及到数据的某种形式的可视化，尽管这不是必须的。这些演示可能是实时状态显

示(例如，用于信息仪表盘）、静态的按需显示或介于两者之间的任何东西。同样，根据演示的类型，它们可以实时消费来自 RabbitMQ 消息代理的事件，从事件持久性服务中获取历史事件，从事件分析服务中获取数据分组，或三者的任何组合。

4.1.4. 事件发布服务 Event Publishing

事件发布服务作为事件发布者和 RabbitMQ 消息代理之间的一个抽象层，提供辅助发布访问模式中描述的一个或多个优点。这是通过用于PMeta 事件的 REST API 实现的，而不是通过 RabbitMQ 客户端 API 直接发布。

4.1.5. RabbitMQ 消息代理

RabbitMQ 消息代理是 PMeta架构的核心。它将所有事件消息从发布者者路由到接收者。RabbitMQ 消息代理由 RabbitMQ 实现。RabbitMQ 是一个广泛部署的开 源消息代理，在PMeta参考架构，我们对 RabbitMQ 服务器和客户端提出以下要求。

4.1.5.1. 服务器要求

RabbitMQ 消息代理应使用 RabbitMQ 3.0.0 或更高版本。

请注意，PMeta 对部署架构（如分布式或非分布式）、认证执行或交换配置没有要求。消息代理的这些方面取决于每个具体情况下的需求。

4.1.5.2. 客户端要求

- 客户端实现可以使用与确定的 RabbitMQ 服务器版本兼容的任何 RabbitMQ 客户端 API（请参阅服务器要求）。
- 客户端实现应支持 RabbitMQ PLAIN 验证。
- 客户端实现可以支持除 PLAIN 之外的其它认证类型。
- 除非在其文档中另有说明，否则客户端实现不得对服务器的部署或配置做出任何假设。
- 除非在他们的文档中另有说明，否则客户机实现不得对交换拓扑结构、类型或配置做出任何假设。
- 除非在他们的文档中另有说明，否则客户机实现不应该对路由方案进行任何假设。
- 除非在其文档中另有说明，否则客户机实现不应该对队列配置进行任何假设。
- 任何消息发布服务和实现都应作为 RabbitMQ 生产者发布到 RabbitMQ 消息代理中的交换（exchange）。
    - 它应该确保发布事件类型符合协议定义。
    - 它应该支持路由键 （routing key）。
    - 它应该使用形式为 ***eiffel.<family>.<type>.<tag>.<domainid>*** 的路由键，其中，
        - **family**是当前事件所属的PMeta事件组的非空的名称。实现可以选择在这个字段中使用一个固定的字符串。
        - **type**是发布的PMeta事件的类型（即meta.type的值），例如EiffelArtifactCreatedEvent。

        - **tag**是一个实现特定的标签。它可以是任何非空的字符串，但不能包含句号。
        - **domainid**是一个非空字符串，代表事件所适用的域。它对应于PMeta事件的meta.source.domainId的值。
    - 除非在其文档中另有说明，否则它不应该对连接到同一交换（exchange）的其他生产者的存在、不存在或性质做出任何假设。
- 通过实时事件解析服务访问RabiitMQ消息代理的任何事件持久性服务、事件分析服务、事件呈现服务或管道实现都应作为从 RabbitMQ 队列（queue）消费的 RabbitMQ 消费者。
    - 除非在其文档中另有说明，否则它不得对连接到同一队列的其他消费者的存在、不存在或性质做出任何假设。

4.2. 访问模式 Modes of Access

访问模式反映了与PMeta事件互动的一种形式。它并不代表任何物理实体，而只是描述一种行为类型。

4.2.1. 聚合的事件分析 Aggregated Event Analysis

在聚合事件分析中，事件分析服务在原始PMeta事件和其消费者之间提供了一个抽象层。这在需要多个事件所含信息的用例中特别有用。虽然PMeta事件可以被认为是传达与管道中某些实体相关的事件的动词，但聚合事件分析是基于所有这些动词的总和来了解这些实体。

例子：每当一个新候选版本在模拟环境中通过了某一组测试，Jane就想在目标环境中启动测试。她可以通过首先监听候选发布版本的EiffelArtifactCreatedEvents，然后监听任何指向该候选版本的下游测试事件EiffelTestCaseTriggeredEvents和EiffelTestCaseFinishedEvents事件。当任何一个EiffelArtifactCreatedEvent被正确的下游事件集所引用时，这意味着是时候开始目标环境测试了。然而，这需要复杂的逻辑，需要存储每个候选版本的状态和它的下游事件。对Jane来说，实现她的触发器的一个更简单的方法是使用事件分析服务。这是因为她感兴趣的并不是事件本身，而是候选版本的状态：每当一个新的候选版本达到了通过所需测试的状态，Jane就希望得到通知。通过事件分析服务的提供订阅设置服务，使Jane的触发器很容易实现，且无需维护任何状态。

4.2.2. 辅助发布 Assisted Publishing

在辅助发布中，发布者不直接向 RabbitMQ 消息代理发布事件，而是通过事件发布服务发布。虽然直接发布不需要任何额外的服务，但辅助发布提供了几个好处。

- 没有特定语言的库集成。事件发布服务提供了一个与语言无关的 REST API，它封装了 RabbitMQ 消息代理的集成。
- 模板生成。除了事件类型特定的数据外，所有事件在其元对象中都包含一定量的模板。事件发布服务可代表其客户处理这些内容，使他们能够专注于他们需要发布的事件的内容。

- 链接查询。事件通过UUID相互引用，这意味着有时可能需要从Event Persistence中获取历史事件，以创建一个完整的新事件。事件发布服务可以代表其客户执行这项任务。
- 版本协商。保持最新的事件类型版本对每个用户来说都是困难的，而事件发布服务可以为历史版本提供多个API端点，并在最新的可行版本上创建一个相应的事件。

4.2.3. 历史事件查询 Historical Event Lookup

历史事件查询查询事件持久性服务中过去发生的事件。这对于任何需要在现在或过去事件的信息上进行操作的用例都是必需的。

4.2.4. 直接发布 Direct Publishing

为了发布 PMeta 事件数据，参与者可以直接发布到 RabbitMQ 消息代理。这样做时，参与者将完全实现 RabbitMQ 客户端行为，并负责创建和序列化构成事件的 JSON 文档。对于某些用例来说，这可能是一个更好的选择，但事件的创建和序列化可能并不简单。特别是，需要通过UUID引用其他事件时可能需要使用Event Persistence进行查找。还值得注意的是，直接发布需要代码库与 RabbitMQ 集成，而辅助发布服务提供 REST API 集成。

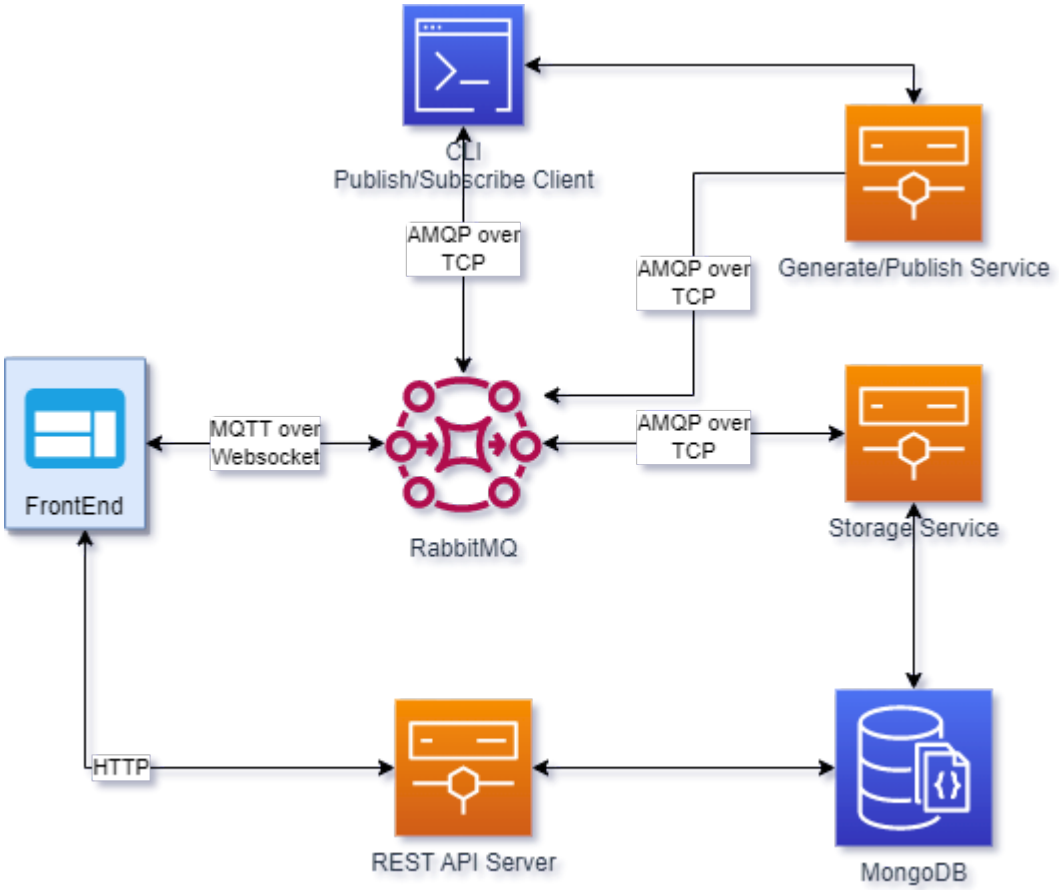
4.2.5. 实时事件解析 Real Time Event Parsing

消费事件的最直接模式是通过实时解析从 RabbitMQ 消息代理接收的事件。这将作为 RabbitMQ 客户端从队列中消费消息来完成。这是访问事件数据的一种简单而快速的方法，但也有局限性。虽然可以将队列（queue）配置为保存在客户端离线时收到的消息，从而防止因离线而导致的数据丢失，但在队列创建之前发生的任何事件都将无法访问。此外，客户端将需要负责存储任何需要以后使用的事件信息。在简单的用例中，这些限制并不存在问题。例如，如果一个人需要的所有信息都包含在一个单一的PMeta事件中，存储事件数据就不是问题。

例子：Jane需要在新的候选版本发布时启动一个内存分析工作。她监听与她的候选发布版本的groupId和artifactId相匹配的EiffelArtifactPublishedEvent，并根据事件数据中的版本位置信息来获取artifact。这是一个完全无状态的程序，可以通过直接从 RabbitMQ 队列中消费事件来轻松实现。

更高级的用例则得益于使用历史事件查询和/或聚合事件分析。

5. PMeta基本原型实现架构



5.1. RabbitMQ

实现参考架构中的消息代理 服务器。

5.2. CLI

用户可以通过CLI客户端发布和订阅消息。
实现参考架构中的RabbitMQ客服端，直接发布以及实时事件解析。

5.3. Generate/Publish Service

和CLI客户端相比，该服务提供了消息产生，验证和发布的REST API，并且负责维护和RabbitMQ的链接。好处是

- 多用户共享服务，不需要维护和RabbitMQ的链接，可以减少RabbitMQ的连接数量
- 提供与语言无关的REST API接口。便于集成到各个软件过程管道。

实现参考架构中的事件发布服务和辅助发布。

5.4. Storage Service 和 REST API Server

MongoDB作为PMeta事件的数据库，在及基础上构建Storage Service和REST API Server，以实现参考架构中的事件持久性服务。

5.5. Storage Service

接收消息以及存储到数据库，作为数据库和消息代理之间的抽象层。

5.6. REST API Server

查询服务是PMeta架构中关键的服务，以REST API提供了单一事件查询，上下游事件查询，或者复杂条件查询等功能。

5.7. FrontEnd（Web）

实现参考架构中的事件展示服务。

RabbitMQ服务器需要安装 MQTT over Websocket 插件，Web前端通过MQTT协议实时监听事件。

6. PMeta消息选型概述

6.1. 消息基本应用简介

6.1.1. 源代码管理事件

源管理角色负责存储和跟踪源代码的修订。在典型的用例中，它不是由PMeta事件中的信息触发的，而是由开发者的直接行为（例如提交修改）触发的。

6.1.2. 测试类事件

描述测试触发条件，测试执行的开始和结束的相关事件。

6.1.3. 软件工件类事件

描述软件过程中工件的创建和发布等活动事件。

6.1.4. 置信度仲裁事件

描述软件工件何时满足了特定的置信度要求。软件置信度可以有多种因素决定，包括但不限于测试结果。置信度的定义需要由公司组织来定义。

6.1.5. 组合定义事件

组合定义事件定义元素的组成，通常是建立在工件上。在实践中，它经常与工件创建角色相结合。此外，它经常不是由任何专门的服务实现来解决，而是通过服务执行中的配置（即作为脚本的一部分）或作为构建文件或依赖声明中的源代码的一部分。

6.1.6. 环境配置事件

负责根据需要为测试和其他活动提供环境。根据道处理的软件的性质和它的环境依赖性，这种环境可能非常简单，也可能非常复杂。因此，环境配置角色可能非常简单，也可能非常复杂。许多执行工具都有内置的环境配置，例如构建环境管理。

6.2. 消息典型关系视图