文献解读

201730681303 成子谦 软件工程一班

解读文献

- Paakkonen, P., & Pakkala, D. (2015). Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems. Big Data Research, 2(4), 166-186.
- Perry, D. E., & Wolf, A. L. (1992). Foundations for the study of software architecture. ACM Sigsoft Software Engineering Notes, 17(4), 40-52.

文献内容

Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems.

这篇论文主要关注大数据系统的参考架构、技术、商业产品和服务，目的是促进大数据系统建设中的架构设计，技术或商业产品和服务的选择。

论文提出了以下研究问题：

1. 哪些元素构成大数据系统的参考架构？
2. 如何对大数据系统的技术和产品/服务进行分类？

从而引出了论文的主要目的：

1. 针对大数据系统，设计独立于设计技术的参考架构。
2. 针对参考架构对相关技术、产品或服务进行分类。

论文首先介绍了利用大数据调查参考架构和用例的相关工作。主要研究问题涉及大数据参考架构中包含的元素，并设计了包括功能组件、数据存储和连接数据流的参考架构。该架构基于已发布的大数据用例的经验数据，采用归纳推理构建。此外，论文还介绍了体系结构模型和用例之间的映射。随后对大数据领域的相关工作进行了调查。调查的重点是流处理、图形建模、商业智能和可视化技术、大数据基准测试、虚拟化和云解决方案、新技术框架和商业产品和服务。

第二个研究问题侧重于大数据系统的技术和产品服务的分类。论文提供了参考分类，其中包括来自审查的大数据用例和相关工作的材料。

作者首先从文献中调查大数据研究、参考架构和用例；随后，提出了大数据系统参考架构的设计，该架构是在对所提出的用例进行分析的基础上进行归纳构建的；最后，提供分类以创建大数据研究、相关技术、产品和服务的整体图景。作者认为：实现大数据系统时，所需要考虑的重要因素包括系统的体系结构设计、底层技术以及产品或服务的利用。参考体系结构应有助于创建具体的体系结构，并通过在大数据系统中包含典型的功能和数据流来增加对整体情况的理解。技术和产品/服务的分类应有助于实现系统功能的决策。同时，了解相关技术的架构和性能特征也很重要。

论文随后分析了7个大数据用例中实现架构的功能，并基于分析结果构建了参考架构。大数据用例包括：

- Facebook的数据分析基础设施
- LinkedIn的数据分析基础设施
- Twitter的两种不同基础架构

- Netflix的数据分析基础架构
- BlockMon，一个高性能流分析平台
- 网络流量测量和分析解决方案
- FIU-Miner，一个促进复杂数据分析过程的系统

论文基于所审查的大数据用例创建了技术、产品和服务的初始分类：

- 用例中的数据通常被收集到数据库或日志文件中。
- 关系数据库被用于存储重要的用户相关数据。
- NoSQL数据库或内存存储用于数据分析结果的中间或最终存储。
- 特殊日志工具用于存储基于流的数据。
- 结构化数据通常保存在分布式文件系统中。
- 用例的数据处理技术可以分为批处理和流处理。
- 具有实时约束的作业需要特殊的技术和算法。
- 批处理用于时间要求不严格的工作。
- 批量处理的工作安排了特殊工具。
- OLAP处理转换了用于提供OLAP查询的数据。
- 处理后的数据也可以通过商业智能工具进行可视化。

分析的用例仅涵盖大数据领域的部分研究和商业服务。相关工作调查侧重于流处理、图形存储、基准测试、虚拟化和云解决方案、商业智能和可视化工具、新颖技术框架和商业服务。

Foundations for the study of software architecture

这篇论文的目的是为软件架构奠定基础。

论文首先通过吸引几个成熟的架构学科来开发软件架构的直觉。在这种直觉的基础上，论文提出了一个软件体系结构模型，由三个部分组成：

1. 元素
2. 形式
3. 基本原理

元素是处理元素，数据元素或连接元素。表单是根据元素的属性和元素之间的关系，即元素的约束定义的。基本原理为系统约束提供了体系结构的基础，系统约束通常来自系统需求。论文在架构和架构风格的上下文中讨论模型的组件，并提供一个扩展示例来说明一些重要的架构和样式注意事项。最后，论文介绍了构造的软件架构方法的一些好处，总结贡献并将论文的方法与其他当前工作相关联。

软件设计在20世纪70年代受到研究人员的极大关注。这项研究是针对20世纪60年代首次认识到开发大规模软件系统的独特问题而出现的。研究的前提是设计是一项独立于实现的活动，需要特殊的符号、技术和工具。该软件设计研究的结果现已开始作为计算机辅助软件工程工具进入市场。在20世纪80年代，软件工程研究的重点从专门的软件设计转移到将设计和设计过程集成到更广泛的软件过程及其管理环境中。这种集成使得软件设计开发的许多符号和技术被实现语言所吸收。例如支持“大规模编程”的概念。如果不混淆，设计和实现之间的区别，这种集成往往很模糊。

在20世纪80年代，描述和分析软件系统方面有很大进步。这里指的是正式的描述性技术和复杂的键入概念，使人们能够更有效地推理软件系统。例如，人们能够更有效地推断“一致性”和“不一致性”，并且能够谈论“类型一致性”而不仅仅是“类型等价”。可以相信，20世纪90年代将成为软件架构的十年。与“设计”相比，使用术语“架构”来引出编码、抽象、标准、正式培训和风格的概念。

虽然在定义特定软件体系结构方面，甚至在开发体系结构开发过程的一般性支持方面做了一些工作，现在是时候重新审视它们。在软件过程和软件过程管理的更广泛的背景下的体系结构，以及编组已经开发的各种新技术。人们期望从作为主要学科的软件架构的出现中获得的一些好处：

1. 架构作为满足需求的框架。
2. 架构作为设计的技术基础，是成本估算和过程管理的管理基础。
3. 架构作为重用的有效基础。
4. 架构作为依赖的基础和一致性分析。

因此，研究的主要目标是支持软件架构规范的开发和使用。

在介绍软件架构模型之前，论文通过以下方式为其奠定了哲学基础：

1. 通过类比现有学科来发展对软件架构的认识。
2. 在多级产品范例中提出软件架构的上下文。
3. 为软件架构作为一个单独的学科提供一些动力。

值得注意的是，论文没有命名软件架构。因为作者认为存在不同类型的软件体系结构，但尚未将它们正式化或制度化。作者认为，之所以存在目前的状况，是因为本应有这么多的软件架构，而不是因为存在这么多。

作者可以抽象到软件架构中的具体级别，例如进程和进程间通信。但是，需要考虑的拓扑数量很多，并且这些拓扑通常没有名称。此外，作者强调与节点和连接的拓扑不同的方面。作者考虑的是诸如进程的放置或进程间通信的种类之类的问题。因此，即使人们可以从类似的抽象层次来看待架构元素，人们也不会从使用网络作为软件架构的类比中获益。

2006年至今进展情况

从软件体系结构研究和应用的现状来看，当前对软件体系结构的描述，在很大程度上还停留在非形式化的基础上，依赖于软件设计师个人经验和技巧。在目前通用的软件开发方法中，其对软件体系结构的描述通常是采用非形式化的图和文本，无法描述系统期望的存在与构件之间的接口，更不能描述不同的组成系统组合关系的意义。这种描述方法既难以被开发人员理解，又难以适于进行形式化分析和模拟，且缺乏相应的支持工具帮助设计师完成设计工作，更不能用来分析其一致性、完整性等特性。

关于软件体系结构的研究工作主要在国外展开，国内到目前为止对于软件体系结构的研究尚处在起步阶段。

什么是软件体系结构？为什么需要研究软件体系结构？

软件体系结构是具有一定形式的结构化元素，即构件的集合，包括处理构件、数据构件和连接构件。处理构件负责对数据进行加工，数据构件是被加工的信息，连接构件把体系结构的不同部分组组合连接起来。这一定义注重区分处理构件、数据构件和连接构件，这一方法在其他的定义和方法中基本上得到保持。

软件体系结构的发展是和计算机抽象技术发展同步的，大规模的复杂软件系统的性能和质量对软件工程技术提出了新的需求。随着软件系统规模越来越大、越来越复杂、整个系统的结构和规格说明就显得越来越重要。对于大规模的发杂软件系统来说，总体的系统结构设计和规格说明比起对计算的算法和数据结构的选择变得越明显重要。