

硕士研究生毕业论文

题目:

协同设计数据管理关键技术研究与实现

Key Techniques for Data Management in Collaborative Design Systems

姓 名： 李润东

学 号： 10948291

院 系：信息科学技术学院 计算机科学与技术系

专 业： 计算机软件与理论

研究方向：软件工业化生产技术与系统

导师姓名： 赵文

二〇一二年五月二十二日

# 个 人 声 明

在论文导师指导下，本文确为本人独立完成，并愿意承担因违反学术规范而应该承担的全部责任。

签字:

年 月 日

# 目录

[目录 2](#_Toc325040168)

[摘要 5](#_Toc325040169)

[Abstract 6](#_Toc325040170)

[1. 引言 7](#_Toc325040171)

[1.1. 研究背景 7](#_Toc325040172)

[1.2. 研究现状 8](#_Toc325040173)

[1.2.1. 计算机支持的协同工作 8](#_Toc325040174)

[1.2.2. 并行工程 9](#_Toc325040175)

[1.2.3. 协同设计 11](#_Toc325040176)

[1.3. 论文组织结构 14](#_Toc325040177)

[2. 相关技术与研究 16](#_Toc325040178)

[2.1. 协同设计支持环境的研究现状 16](#_Toc325040179)

[2.1.1. 协同设计支持环境的主要需求和功能 16](#_Toc325040180)

[2.1.2. 现有国内外协同设计支持环境 17](#_Toc325040181)

[2.1.3. 协同设计工作流 19](#_Toc325040182)

[2.1.4. 协同设计的数据管理 21](#_Toc325040183)

[2.2. CAx工具之间的数据交换 25](#_Toc325040184)

[2.2.1. STEP 25](#_Toc325040185)

[2.3. 数据世系研究 26](#_Toc325040186)

[2.3.1. 数据世系概述 26](#_Toc325040187)

[2.3.2. 工作流领域的数据世系管理 27](#_Toc325040188)

[2.3.3. 数据世系对协同设计的意义 28](#_Toc325040189)

[2.3.4. 协同设计工作流的数据世系管理 29](#_Toc325040190)

[3. 基于数据世系的数据管理方案 31](#_Toc325040191)

[3.1. 协同设计“世系”的形成 31](#_Toc325040192)

[3.2. 协同设计中数据世系的层次模型 31](#_Toc325040193)

[3.3. 世系的定义 33](#_Toc325040194)

[3.3.1. 连通图表示数据世系信息 33](#_Toc325040195)

[3.3.2. 协同设计过程数据的世系 34](#_Toc325040196)

[4. 协同设计中的数据一致性保障机制 35](#_Toc325040197)

[4.1. 数据不一致问题的来源 35](#_Toc325040198)

[4.1.1. 产品并行开发过程中的迭代 35](#_Toc325040199)

[4.1.2. 协同设计流程中的不一致问题 37](#_Toc325040200)

[4.2. 数据一致性检测算法及优化 38](#_Toc325040201)

[4.2.1. 遍历式算法 39](#_Toc325040202)

[4.2.2. 优化－空间换时间 40](#_Toc325040203)

[4.2.3. 改进方案 42](#_Toc325040204)

[5. 协同设计参数管理工具的设计与实现 44](#_Toc325040205)

[5.1. 参数管理工具的需求 44](#_Toc325040206)

[5.1.1. 参数管理工具的应用场景 44](#_Toc325040207)

[5.1.2. 参数管理模块的需求分析 45](#_Toc325040208)

[5.2. 基于数据世系的参数管理框架 47](#_Toc325040209)

[5.2.1. 参数库 47](#_Toc325040210)

[5.2.2. 动态过程数据管理框架 49](#_Toc325040211)

[5.2.3. 数据收集 51](#_Toc325040212)

[5.2.4. 数据提取 56](#_Toc325040213)

[5.3. 参数管理工具的设计 58](#_Toc325040214)

[5.3.1. 主要的类 58](#_Toc325040215)

[5.3.2. 数据库设计 60](#_Toc325040216)

[5.4. 参数管理工具的实现 62](#_Toc325040217)

[5.4.1. 参数、约束表达式的定义与绑定 62](#_Toc325040218)

[5.4.2. 查看流程执行历程 63](#_Toc325040219)

[6. 总结 65](#_Toc325040220)

[6.1. 论文工作总结 65](#_Toc325040221)

[6.2. 进一步工作设想 65](#_Toc325040222)

[致谢 66](#_Toc325040223)

[参考文献 67](#_Toc325040224)

**图目录**

[Figure 1 CSCW矩阵 10](#_Toc325023992)

[Figure 2 瀑布模型 12](#_Toc325023993)

[Figure 3 并行工程开发模型 12](#_Toc325023994)

[Figure 4 协同设计要素 14](#_Toc325023995)

[Figure 5 一个协同设计场景 19](#_Toc325023996)

[Figure 6 协同设计工作流 20](#_Toc325023997)

[Figure 7 协同设计过程 21](#_Toc325023998)

[Figure 8 EXPRESS 26](#_Toc325023999)

[Figure 9 协同设计中数据世系应用 30](#_Toc325024000)

[Figure 10 数据的层次结构 32](#_Toc325024001)

[Figure 11 模型层与实例层的关系 33](#_Toc325024002)

[Figure 12 多版本任务的数据世系 34](#_Toc325024003)

[Figure 13 一个简单的迭代 36](#_Toc325024004)

[Figure 14 串行与并行迭代 36](#_Toc325024005)

[Figure 15 多版本造成不一致 38](#_Toc325024006)

[Figure 16 任务实例间的参数依赖 40](#_Toc325024007)

[Figure 17 收集动态数据-记录Ta、Tb的依赖关系 41](#_Toc325024008)

[Figure 18 参数管理用况 45](#_Toc325024009)

[Figure 19 参数库 48](#_Toc325024010)

[Figure 20 基于数据世系的参数管理框架 49](#_Toc325024011)

[Figure 21 产品结构与参数关联 51](#_Toc325024012)

[Figure 22 参数与任务关联 52](#_Toc325024013)

[Figure 23 任务执行与审核 53](#_Toc325024014)

[Figure 24 顺序图 - 流程动态数据收集 55](#_Toc325024015)

[Figure 25里程碑与参数批次 56](#_Toc325024016)

[Figure 26 导出参数批次 58](#_Toc325024017)

[Figure 27类图-参数管理工具 59](#_Toc325024018)

[Figure 28任务实例之间参数依赖 59](#_Toc325024019)

[Figure 29参数管理工具主要功能模块的设计 60](#_Toc325024020)

[Figure 30 顺序图 参数定义与绑定任务 61](#_Toc325024021)

[Figure 31 活动图 一致性保障 62](#_Toc325024022)

[Figure 32 用户界面 参数定义 63](#_Toc325024023)

[Figure 33用户界面 查看任务执行历史 64](#_Toc325024024)

# 摘要

跨学科协同设计规模大、过程复杂，参与人员很多而且来自不同部门并且需要密切的协作，这使得协同设计具有很高的复杂性。在这样大型复杂的项目中，设计过程产生大量数据。如何对这些数据实施有效的管理，是当前协同设计研究者们面临的一个重要问题。一方面，本文对协同设计中的数据管理进行了研究，了解到大量数据管理的复杂性，以及流程中任务实例和参数的多版本特点；针对这些数据管理的需要，我们设计了一套基于数据世系的协同设计数据管理框架。另一方面，协同设计流程中的并行性、迭代性导致数据不一致的问题，这在很大程度上影响了协同设计的效率；本文也对此提出了相应的数据一致性检测算法和机制。最后，我们还在协同设计平台中对这些管理方案进行了实现，达到了预期的效果。

论文的第一章介绍了协同设计的概念以及本文的研究背景——协同设计中的数据管理，并提出了数据管理的需求和其中存在的问题。第二章描述了当前协同设计支持环境的研究现状、计算机辅助工具之间的数据交换以及数据世系相关的研究。第三章提研究了协同设计中数据世系的概念，并以此为基础提出了一个管理方案。第四章对协同设计中数据一致性问题进行了介绍并设计了一致性检测算法。第五章详细描述了基于数据世系的数据管理框架和具体设计，并介绍了基于J2EE框架的参数管理工具的实现。最后第六章是对本文工作的总结以及对下一步工作的设想。

关键词：

协同设计，参数管理，参数，一致性，工作流

# Abstract

Computer-Supported-Collaborative-Design is large-scaled and complicated, with its high complexity coming from the fact that designers from different places need close cooperation. In these large and complex projects, the design process always produces large quantity of data. It is a significant problem how to effectively manage and use these data. We explore specific problems that exist in data management and come up with some solutions. On the one hand, in the collaborative-design workflow system, tasks and parameters have multiple versions so it is hard to trace design process and data flow; this paper describes a data-provenance based scheme for data management in collaborative design. On the other hand, the parallelism and iteration of collaborative design process lead to the data inconsistency problem, which affects the efficiency of design; this paper provides a set of scheme and algorithm for this serious problem. In the end, we implement our scheme and design in a collaborative design platform, which shows that our solutions are effective.

The first chapter of this paper briefly introduces research background for data management in collaborative design systems, as well as the concept of CSCD and its history. The second chapter describes research about CSCD-environment, data exchange among CAx tools, and data provenance. Chapter 3 comes up with a model of data provenance in the field of collaborative design. Chapter 4 discusses the problem of data-consistency in CSCD and presents some algorithms for its solution. Chapter 5 describes the data-provenance based scheme and design for data management in CSCD in details. In the ending Chapter 6, I make a summary to my work done and a short discuss about my possible future work.

Key Word:

Collaborative Design, Parameter Management, Data Consistency, Workflow

# 引言

## 研究背景

计算机支持的协同设计中，一个重要的问题是如何处理设计过程中需要的大量数据。比如，设计人员会使用大量的设计资源，这些资源可能以文件、图纸、多媒体等形式存在；再如，大量的过程数据会随着设计流程的进行而产生，这些过程数据是设计过程的蹭结果，具有多版本、多方案等特点，数量巨大且成分复杂。如何利用计算机技术对这些数据实施有效的管理，是当前协同设计研究的一大重点。

对于协同设计中的文档、图纸等静态资源，一般只需要以一定的形式将其组织并存储，方便设计人员的查看和调用即可；而设计过程中的动态过程数据，管理起来就复杂的多。总的来说，对过程数据的管理有很多问题，比如下面列举的几个：

* 当某个设计结果出现问题时，具体是设计过程中的哪个地方出现了错误？想要精确的找出问题，就需要我们追踪相关过程数据（比如某个设计任务的输出参数）的完整计算历程。即，它是从哪些数据经过计算过推导而得到的？这些数据又是如何得到的？一直向上游追溯直到原始的设计资源。
* 数据类型多样。协同设计一般是大型项目，比如飞机、卫星等大型产品的设计。大型产品各部件的设计都有各自的特点，描述不同零部件的参数数据形式不同，比如有图形、文本等不同的存储格式。
* 大量设计人员参与的协同设计，面临着数据共享的问题。比如，若共享的数据发生了更新，那使用它的人员就要对自己的工作进行修改。
* 数据存储在地理上的分布以及异构问题。参与协同设计的人员可能分布在不同的地方，他们各自使用的数据除了共享的部分，也会有自己使用的数据源。
* 数据一致性的问题。产品设计过程中的迭代导致数据的多版本特点，所以设计人员在使用中要注意选择合理的数据版本。

总之，协同设计的数据管理方面存在很多问题，而有效的解决方案对计算机支持的协同设计的发展有重要作用。本文就对这方面的需求和问题进行了研究，并提出了一些解决的办法。我们关注的主要是设计过程中的动态数据，针对多版本数据提出了基于数据世系的管理方案，并设计了相应的一致性保障机制。

## 研究现状

近三十年来，计算机技术和互联网的发展带动了信息技术在工业界的应用。尤其是大型产品设计和制造中，计算机技术得到了广泛的应用，而且这些应用还衍生出了很多新的概念和思想。本文的主要研究内容——协同设计的数据管理——就是随着这些技术的发展而逐渐受到企业和研究机构的注意。下面对其中一些比较重要的概念和技术进行简要介绍。

### 计算机支持的协同工作

计算机支持的协同工作（CSCW，Computer Supported Cooperative work）这个概念的首次提出是在1984年——在一个由一些对“应用技术来支持人们工作”感兴趣的人员参加的研讨会上，美国麻省理工学院（MIT）的依瑞·格里夫（Irene Greif）和DEC公司的保尔·喀会曼（Paul M. Cashman）提出了CSCW[1]。

目前对于CSCW比较统一的定义是：“**在计算机技术支持的环境中，一个群体协同工作完成一项共同的任务**。”

互联网的建立和发展极大的推动了信息技术向纵深发展，于是，传统的单一信息媒体的交流方式、孤立式工作方式越来越难以满足人们在信息时代的需求，人与人之间的合作以及信息的共享变得越来越重要。于是，CSCW应运而生。它由对群体工作方式的理解和支持群体工作方式的相关技术两部分组成。

根据Carstensen和Schmidt[2]，CSCW处理了“协作的任务以及参与人员的协调是如何通过计算机系统来支持的”这样一个问题。有些人认为CSCW与群件（Groupware）的概念是一致的，但是不同的观点则认为群件只是支持协同工作的实际的计算机系统，而CSCW则不只研究群件，还重点研究计算机系统的应用所带来的心理、社会及组织上的影响。Wilson在1991年提出的定义突出了CSCW在这方面的特点：“CSCW是一个通用术语，它将对协同工作的理解与支持协同工作的计算机软硬件、网络、服务等技术结合在一起。”[3]

下面的CSCW矩阵可以帮助我们理解CSCW的概念和研究内容。



Figure CSCW矩阵

这个矩阵考虑了CSCW的应用环境，主要着眼于两个角度，也就是图中的横轴和纵轴：一是协同工作在地理上是发生在同一位置还是分布的，二是人员间的协作是同步还是异步的。于是矩阵被分为四个象限，即CSCW在不同的外部条件下，它的研究内容是不同的，采用的计算机技术和工具有很大的区别，这在上图中也有所表示，每个象限中的小字列举了在这个象限所代表的协同工作环境下，相应的CSCW应该考虑的计算机技术和工具。

CSCW的目标是研究以及设计计算机应用系统，用于支持不同领域的各种各样的协同工作。

### 并行工程

并行工程(Concurrent engineering，简称CE)是基于任务并行的一种工作方法学，它是近些年来新出现的一种产品开发模式。并行工程将设计工程、制造工程那一夜其它一些功能结合在一起，目的是缩短新产品的开发周期。

世界市场的快速发展使得世界范围内的竞争变得愈发激烈，这给企业造就了严酷的生存环境。在竞争中，许多国家的制造业通过采用和发展CAD（Computer Aided Design，计算机辅助设计）等技术取得了很大成绩，而在实践中设计者们逐渐发现，改进产品开发过程可以比改进生产过程取得更大的效益。各国制造业争相创造各种新方法、新技术、新思想以改善各自的产品开发模式，于是，在CAD技术及CIMS（Computer Integrated Manufacturing Systems，计算机集成制造系统）的基础上，制造业领域中涌现出一批先进的产品开发技术与概念。这些新技术、新概念是针对原来存在的问题而提出的，是对制造业的技术与观念的革新。并行工程就是这些先进概念中的一种，是对传统的串行产品开发模式的根本性的改进。它可以被视为一种对新产品开发有指导意义的朴素哲理，而实践中它又是一种综合的自动化制造技术，它实际上是对许多新技术、新思想的集成和抽象。并行工程的出现受到了国内外众多研究单位与制造企业的重视。

并行工程在当前仍然是一种相对较新的设计管理方法，但它在最近这些年已经变得逐渐成熟，成为能够有效优化工程设计周期的一种系统方法。因此，并行工程受到工业界的关注，很多公司和研究机构都对它进行了实际应用，尤其是在航空和航天工业中。

在1988年，IDA（Institute of Defense Analyze，美国国家防御分析研究所）将并行工程定义为“对产品设计及其设计过程、制造过程和技术过程等进行并行、综合设计的系统化工作模式。它力求开发者在产品设计早期就考虑到产品从概念形成到最终报废的整个生命周期中的所有因素，包括质量、成本、计划调度和用户需求等。”

并行工程的基本前提围绕着两个思想展开。一是要在产品设计早期就将产品生产周期内的所有元素都考虑在内，包括功能、生产力、装配、可测试性、产品维护、环境影响以及最终的分解和回收等等。二是要同时或者并行的考虑上面所提到的这些元素。最终的目的就是要利用这些产品生命周期内各个环节的并行性来提高产量和质量。这种哲学是并行工程成功的关键，因为它使得产品生命周期内可能发生的各种错误能够在早期的设计环节就被发现，而这时产品开发还处在一个抽象的或数字化设计阶段，于是纠正错误的代价被极大的降低。

并行工程给出了新的产品设计模式，即迭代式或者说是综合的开发模式，与传统的“瀑布模型”这种线性的设计模式相比有很大区别。瀑布模型最大的弱点就是它在每个环节中无法考虑之前或者之后的设计环节，以至于当某个环节出现问题时，整个设计流程往往要舍弃当前得到的很多设计结果，甚至从头开始。而迭代式的设计过程则采用渐进式的方法，将整个生产生命周期都考虑在内。

二者的区别如下面两图所示。



Figure 瀑布模型



Figure 并行工程开发模型

### 协同设计

#### 协同设计的定义

CSCD（Computer Supported Collaborative Design，计算机支持的协同设计）与前面提到的CSCW有相似之处——它们都是研究如何应用计算机系统来辅助群体协同工作。但不同的是，CSCW的涵盖面更广泛，它可以指一切人机结合的多人协同工作。而CSCD则主要是面向产品设计，可以看作是CSCW在设计领域中的应用，它继承了传统的CAD技术并以其为依托的发展起来，是计算机辅助设计在网络化时代发展的新方向。

协同设计是并行工程在先进制造技术中运行的核心。前面提到过传统的串行迭代的瀑布式设计模式，它按产品生命周期各过程的顺序执行，在某阶段发现问题后会返回到前面的阶段中去解决。而并行工程则是在产品设计阶段就将产品全生命周期中各个元素都考虑到，因此它要求参与产品生命周期各个阶段的不同部门之间的协同工作，在产品设计时不仅设计专家要参与，而且制造、质量、支持等后续部门工作人员也要与设计人员密切交流，从各自的角度出发，利用自身的专业知识与经验，对产品的设计提出自己的意见，尽可能使得产品在后续阶段少出问题。

协同设计可以这样定义：一个设计团队在计算机系统的支持下进行协同工作，合作完成一项设计任务。为了辅助不同设计人员之间的交流与协作，在协同设计中要实现一个完整的信息交换与协同机制，从而提高产品设计与开发的效率并保证质量。其基本原理是，继承和发展并行设计的基本思想，借助计算机与网络技术，利用CAx（CAD/CAPP/CAM等计算机辅助工具）与PDM（Product Data Management，产品数据管理）等技术手段，构建统一协调的协同工作平台，为参与协同设计的工作人员服务。

协同设计是一种先进的设计理念，它依靠计算机、网络通信、数据库等技术和工具，组织多个设计人员跨学科、跨地域的协作，实现产品的协作开发。它是CAD技术的重要发展方向，既包括产品设计机理的研究，又包括协同设计工作环境的构建。“协同”的含义是多方面的，包括设计人员之间、设计者与支撑环境之间的交互与协作，也包括不同设计阶段中产品信息模型之间的一致性要求。协同设计的目标是充分利用计算机技术，实现设计团队分工合作的工作模式并提高效率，从而缩短产品开发周期并保证设计质量。

#### 协同设计的要素分析

协同设计主要涉及产品、人和组织、资源、过程、环境等要素[5]，它们之间的关系如下图所示。



Figure 协同设计要素

* 产品：设计的对象。
* 人和组织：人是设计的主体；多个设计主体组成的组织就是协同设计团队。
* 资源：设计中所需要的各种数据、材料及工具等。
* 过程：产品的设计过程，在过程中产品对象的状态不断变化。
* 环境：协同设计的支撑环境，其含义包括两个方面。一是支持协同设计的软硬件设施，如网络、计算机系统、应用软件等。二是指协同设计框架结构，它将协同设计的其它要素有效的组织在一起。

#### 协同设计的特性

经过比较与归纳，协同设计与传统的设计模式相比有很多特性，如多主体性、分布性、并行性、协调性、冲突性、参数化等等[6,7]。下面就其中几个特性作简单说明。

* 多主体性：主体指参与设计的工作人员，或是项目小组，或是指设计本身，多主体性就是说协同设计会涉及多个设计人员、项目小组或由多个设计组成，它们组成了产品的协同设计团队。
* 分布性：组成协同设计团队的设计人员或小组在地理上是分布的，他们通过计算机与网络来实现信息交换和工作协调。另外，设计任务和设计资源也是分布的。
* 并行性：不同设计人员各自可能会参与不同的、相对独立的设计任务，它们会并行的、同步的进行，以提高工作效率。
* 冲突性：这是由任务间的制约关系导致的。各个设计任务之间相对独立，但是会存在制约，所以各任务之间需要密切配合。而且，不同设计人员之间的信息交流具有不对等和延迟性等特点，也使得协同过程中的冲突难以避免。故而冲突消解也是协同设计的重要内容之一。
* 参数化：在具体系统中，产品的各项特征被描述为一个个参数，产品就是一个参数化单元，可满足设计和制造的需要。参数化设计是新一代智能化、集成化CAD系统的核心内容，也是协同设计中的关键技术。它是初始设计、多方案比较和动态设计的有效手段，在协同设计中的应用可以极大的提高设计效率。

协同设计要支持设计团队的协同工作，必须实现在分布式环境中的信息共享与交换，即在需要时能够把正确而完整的信息通过网络传输给需要的人。在协同设计中，计算机技术不单是要辅助每个设计人员的工作，还要有力的支持不同设计人员或小组之间的协同交互以及利用计算资源的能力。所以，产品数据管理是协同设计中的关键问题。

## 论文解决的问题

针对协同设计中数据管理的特点和需求，我们在研究过程中对当前数据管理的问题进行了总结：

1. 大量数据的管理

协同设计项目一般规模较大，具有跨学科、跨部门的特点，而且其设计过程复杂多样、周期较长，所以数据管理功能需要实现对大量数据进行有效的保存和处理。

1. 异构数据源之间的数据共享

协同设计中的各个设计小组在地理上往往是分散的，而不同小组往往会使用自己的数据源，这些数据源之间存在异构性。我们要在数据管理中克服这个困难。

1. 对多版本数据的管理

设计过程的迭代等特性导致了过程数据的多版本性，其中任务实例多版本和参数多版本是主要的两个问题。这既增加了数据量，又为数据管理带来了更多的复杂性。我们要在实施数据管理时充分考虑这个问题。

1. 数据一致性的问题

协同设计工作流程的并行性、迭代性以及数据的多版本特性使得协同设计中的数据一致性问题尤为突出。所以，一致性检测需要一个合理有效的解决方案。

针对上述问题，我们提出了一个基于数据世系的协同设计数据管理方案，利用协同设计流程中产生的数据世系信息来实施对大量过程数据的有效管理，其基本框架可以用下图描述。



Figure 数据管理方案示意

围绕着协同设计的工作流，我们建立了数据世系信息的收集和提取方案，并利用这些信息来保证流程中的数据一致性，最终实现对于多版本设计数据的有效管理。

## 论文组织结构

1. 引言

介绍了课题的研究背景以相关概念，并对本文的研究内容、所解决的问题和文章的组织结构进行了阐述。

1. 相关技术研究

对当前协同设计支持环境的研究现状、CAx工具间的数据交换和数据世系相关研究进行了介绍。

1. 基于数据世系的数据管理方案

描述了协同设计中数据世系的内容及来源，并提出了一个基于数据世系的数据管理方案。

1. 协同设计中的数据一致性保障机制

阐述了协同设计中的数据不一致问题的产生原因，并设计了几种数据一致性检测算法。

1. 协同设计参数管理工具的设计与实现

根据第三章提出的数据管理方案，设计并实现了协同设计的参数管理工具，这一章描述了工具的需求、框架、设计内容和实现方式。

1. 总结

总结了本文工作内容及解决的问题，并对将来进一步的工作进行展望。

# 相关技术研究

## 协同设计支持环境的研究现状

### 协同设计支持环境的主要需求和功能

当前，平台化是协同设计软件发展的一个主要趋势，即要求设计工具能提供建模、数据运算和传递能仿真工作流程要求的功能，并且可以集成在大型的PDM/PLM（产品数据管理）平台上。另外，协同设计平台当然要支持设计团队的协同工作。

前面提到协同设计同并行工程一样需要在产品设计初期就考虑到后续环节的各方面元素，同时要支持不同设计人员之间的协作，以及支持产品全周期的管理，这就对协同设计支持环境提出了比较高的要求。具体的，协同设计平台需要满足的需求分为以下几个方面[8]。

* 强大的数据管理功能

产品设计过程中产生的数据具有量大、类型多、版本多的特点，这对协同设计平台的数据管理功能提出了很高的要求。

首先，平台要能够管理各种不同类型的设计数据。随着设计阶段的变化和设计者使用的设计工具的不同，设计数据的类型千差万别。平台需要对这些数据进行分类处理，以方便平台用户的查询和调用。

第二，平台要能够有效的控制数据的质量和版本。在设计过程中，即使对于同一个产品，也会产生不同的设计数据，可能的原因有输入参数的不同、流程迭代等等，这些数据的质量也有区别。所以，协同设计平台要支持不同质量和版本数据的存储和控制。

第三，平台要提供强大的数据共享功能。设计团队对设计数据的共享、设计者之间数据的交流是协同设计的基础。平台在提供完善的数据共享功能的同时，还要考虑数据安全、约束和一致性等问题。

* 细致的流程管理功能

协同设计的工作流程非常复杂，而优秀的流程管理功能是协同设计平台能否成功的关键。平台需要支持流程管理的各项功能，如阶段划分和任务分解、设计数据审核等等。另外，围绕流程的管理，协同设计平台要对产品设计各阶段的需求进行支持。总之，细致的、划分设计阶段的、完善的协同设计流程管理功能是平台所必需的。

* 能够集成主流设计工具

一个复杂产品的设计会涉及多个学科和专业，协同设计平台要支持来自不同领域的设计人员的工作，就需要集成不同的设计工具。考虑到不同用户的设计习惯，协同设计平台能集成的工具越多越好。即便难以完美的集成所有的工具软件，也要能与大部分主流设计工具相互支持，至少能实现数据的交换，即支持设计工具的数据格式。

总之，协同设计平台是支持协同设计的基础，它要提供有力的软硬件支撑环境，满足协同设计流程和数据协同的要求。因为数据管理功能是平台的核心功能，所以目前的协同设计平台一般都是基于PLM建立的。而随着大规模数据密集型设计项目中的协同需求越来越重要，协同设计平台越来越需要满足大型设计团队在大型项目中的协同工作和大量数据处理的需要。因此，新的协同设计支撑环境的开发势在必行。而工作流和数据世系是协同设计过程中的两种重要技术，对新的支撑环境具有重大意义，因而需要将过程管理和数据管理的实现放在新的协同设计平台的核心地位。

### 现有国内外协同设计支持环境

在科学研究和产品设计领域，众多对协同设计的研究和软件开发中出现了大量的辅助设计与支撑协同设计的开发环境或平台，它们比较突出的特点就是基于流程和数据世系技术给出基本的技术架构，从而结合其它技术手段和方法来综合解决研究和设计中遇到的过程重构、产品建模、文档管理、工具集成和冲突消解等各种问题。

下面列出了目前国内外一些具有代表性的协同设计支撑环境，先描述它们各自的内容和特点，然后对这些平台中工作流技术和数据世系技术的应用与实现进行了分析。

1. iSight[10]

起源可追溯到上世纪八十年代，是一款基于Windows NT和Unix平台的多学科设计优化框架，将iSight进行商品化的则是1996年成立的美国Engineous Software公司。

iSignt辅助设计者集成需要的工具，提供一个组装设计流程的图形界面，并且将方案设计过程自动化。集成的设计工具的执行与交互都是自动控制的，iSight自动实现数据在工具间的传递。最后，它还提供一些探索算法，帮助用户确定最佳设计参数。

iSight提供对运行过程的实时监控，设计过程中的各项输入、输出参数都能在流程执行过程中通过表格或者图形的形式进行显示，这样就提供了对数据的方便的控制手段与管理模式。

1. 锐峰型号协同设计平台DENOVA[11]

DENOVA为国防军工企业的产品设计过程提供了一整套解决方案，它满足“制造业数字化”的需求，实现了产品设计的“过程协同”、“数据协同”、“数据和知识管理”。设计者使用DENOVA可以建立跨部门、跨学科并且贯穿多个设计阶段的协同设计流程，实现对产品设计过程的控制和设计者间的数据传递。

DENOVA中的过程数据管理叫做DENOVA-PDMAN(Process Data Manager)，它与流程系统紧密结合，对设计过程中的数据进行综合管理，包括结构关系、历程以及关联关系等。-

1. 安世亚太PERA协同仿真平台[12]

PERA是在企业基础研发平台之上扩展出的一个统一的、集成的信息化技术平台。它支持产品设计研发的整个过程，从概念设计到详细设计及试验，并实现了设计流程和数据等的协同应用。

PERA提供对过程数据的管理功能，包括版本管理以及追溯点管理。

1. Kepler[13]

作为一个开源的科学工作流引擎，Kepler以数据流为导向的成熟的工作流架构，在Ptolemy II系统的基础上开发而成。

Kepler系统中有一套完整的数据世系框架，提供数据的收集、存储、组织、查询和分析等功能，框架从工作流演化和工作流执行实例这两个层面来管理数据世系。

总体上看，国内协同设计支撑环境的研发和应用落后于欧美发达国家，而在制造业领域更是尚未出现可以商业化的软件产品。不过，软件开发商们正在积极研制基于Web的协同设计支撑平台等相关软件。这些软件的开发方向基本分为两类：一是扩展原有的CAD系统，增加基于Web的功能，但这样只是集成了一种CAD工具，难以真正支持异构环境下的协同设计；另一类则是开发全新的基于网络的协同设计支撑环境，但它主要是提供了数据交换的环境，而不能有效的集成常用CAD工具。

### 协同设计工作流

工作流是一种描述复杂过程的方式，它将过程描述成为一组带输入输出数据的任务以及任务之间的控制依赖和数据依赖。描述协同设计过程的工作流就是协同设计工作流。工作流被创建后在工作流管理系统中执行，并支持后续对工作流本身的编辑；工作流执行过程中产生的过程数据就是产品设计过程中的重要设计数据。这些特性使得工作流特别适合用来描述协同设计。对一个协同设计项目，建立起协同设计工作流模型，可以帮助设计人员深入理解项目整体以及项目内部各部分间的关联。

协同设计的项目中，参与者一般都是分布在不同部门或组织的、不同领域的设计人员和专家，他们的工作经常涉及到数据密集型的计算，并且需要数据的实时交流与共享。下图展示了一个简单的协同设计场景，总体规划人员将设计任务分配给具体设计者，经过一系列设计流程，得到最终设计方案。这个过程中包含可能出现的迭代。



Figure 一个协同设计场景

如上图所示，在产品设计过程中，对过程数据的分析和处理需要来自不同领域的设计人员分工协作。总体规划人员进行任务划分，设定设计任务的目标，然后将任务分配给相应的零部件设计人员。参与零部件设计的具体设计者们将各自的设计结果交付给分析人员，以检验是否满足各自设计任务的目标，设计人员可能会根据分析结果的反馈来改进设计。组装人员将合格的零部件设计组合成统一的设计方案，最终交付给整体规划人员。整个设计是一个反复尝试的过程，可能需要多次的迭代来达到事先设定的设计目标，因为最后整体规划人员会检验整个设计结果，如果存在不合格的部分，则重新开展设计过程，直到得到满意的产品设计结果为止。

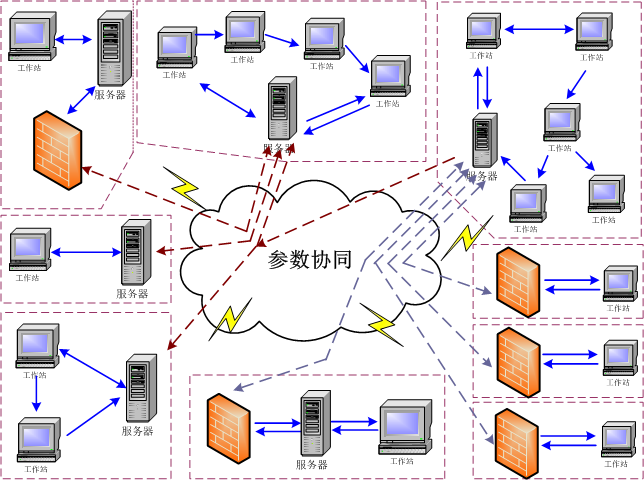
工作流正是描述设计过程的工具，它可以描述一组相互关联的任务以及任务之间的逻辑关系以及数据依赖。总体规划人员可以利用工作流来控制协同设计过程，如下图所示。



Figure 协同设计工作流

规划人员设计协同设计工作流，首先根据具体的设计问题设定输入，即初始数据；然后将设计问题分解为一个个相对独立的设计任务，并将相应的数据与这些任务分别绑定，即确定每个任务针对哪些数据进行操作；最后，设计结果传给分析任务和组装任务。工作流的执行过程中，过程数据会自动产生并传递给相关的任务（即相关的设计人员）。这样，一组带输入输出的任务集合就被完整的定义出来，再加上任务间的逻辑关系，就形成了一个协同设计工作流。协同设计过程如下图所示。

模块



产品总体设计

模块

模块

模块

模块

模块

模块

模块

…...

…...

Figure 协同设计过程

另外，设计人员能够利用工作流来方便的记录数据在设计过程中的演化历程（数据世系），因为实际上数据世系的内容来自于工作流的执行——工作流的执行和任务状态的变化导致数据的演化。数据世系信息对于产品设计有重要的意义，它可以用来重现设计过程、验证设计结果等。

### 协同设计的数据管理

#### 过程数据管理需求

协同设计在进行过程中，会产生大量的过程数据，而如何有效的管理这些过程数据，是协同设计的重要内容之一。要对大量的、复杂的过程数据进行管理，计算机技术的应用必不可少，这也是协同设计支持环境的核心功能需求之一。具体的，过程数据管理需要提供以下几个方面的功能。

* 结构关系管理功能

复杂产品设计中所涉及到的信息很多，如设计参数、设计模型、文档等等，这些数据如此之多，所以协同设计支撑环境必须帮助设计者们理清数据的结构。这些看似松散的信息其实都围绕着产品结构，是对产品各个方面的描述，因此以产品结构为纲，建立信息树，就可以实现对是结构关系的管理。这也是数据管理的核心，只要系统实现了对“产品结构树”的管理，数据管理的其它功能就可以在结构关系管理的基础上进行实现。

* 历程管理功能

流程的执行使得数据发生演变，历程管理功能就是要对过程数据的变化过程及数据间的关系进行管理。为了直观的展示过程数据的来源、变化与去向，此功能还要以“历程图”的形式显示某项数据的完整历程。

* 关联关系管理功能

关联关系主要是指产品结构树与过程数据间的关联，于是对这种关联关系的管理就是将大量的过程数据与产品结构树的各个结点建立对应，以方便设计人员对过程数据的查看与调用。

* 数据可视化功能

系统需要对过程数据进行可视化处理并展示给用户。由于过程数据的类型多种多样，有的数据较为复杂，所以系统可能要先对待查看的数据进行“轻量化”处理，再显示给用户查看。

#### 数据一致性管理需求

当前，协同设计是复杂产品设计的主流方式，而在流程设计中，制造业通过实施并行工程等技术来改进产品开发和生产模式。于是，现在所提倡的是并行的、集成的设计产品以及定义产品开发过程（设计过程、制造过程和支持过程等）[15]。面在协同设计过程中，有些场景会产生数据不一致的问题，比如：

参与设计的不同人员（或部门、组织等），在设计过程中需要进行数据的实时交换。而不同设计人员各自的工作流程是并发执行的，所以不同工作流之间实时交换的数据往往会存在版本不一致的情况，问题严重的会导致返工。

再比如：

设计本身会存在多次实验方案的设计过程，协同设计工作流中的某些子流程会多次迭代执行，这样就会产生不同版本的数据。所以，对于一些需要输入参数的设计任务，参数有多种选择，当有多个输入参数时，就更是会产生多个输入组合，目的是通过任务执行而在多种组合方案中选出最优方案。那么，如何保障具有交互关系的输入分支之间的数据一致性，就成为保证任务的设计方案是否正确的重要条件。

对引起数据不一致问题的情形进行分析，可以发现协同设计工作流的一些特性对数据管理带来新的需求：

* 面向数据流的特性，使过程数据管理成为协同设计中必需的组成部分。这在上文中已经进行过描述。
* 流程的迭代性导致在任务的执行过程中会产生多个版本的数据。这增加了过程数据管理的复杂度。
* 协同设计流程具有并发性，即在某些时刻会出现并行的流程分支，这些并行流程之间可能会产生数据不一致的现象。
* 由于多版本数据的存在，共享同一参数的流程或任务可能会取到不同的版本，从而导致数据不一致的问题。

所以，数据管理在协同设计中变得愈加重要，而数据不一致性的问题也更加突出，这都使得数据一致性管理在协同设计中更加关键。

#### 协同设计领域的数据一致性保障

在实际设计过程当中，工作流实例在执行过程中的正常运行，会受到系统外界复杂多变的环境的影响，于是导致工作流执行阶段的数据错误。这些数据错误会存在于工作流的控制数据、相关数据以及应用数据等多个方面。因此，必须采取有效措施来保护协同设计工作流中数据的一致性，提高运行时的可靠性。下面介绍一下当前对数据一致性问题的一些研究成果。

##### 基于约束的数据一致性研究

产品物理结构中存在很多约束关系，这些约束关系大多可以用数值和表达式表示，而通过对这些数值型约束的处理可以保持数据的一致性。比如有研究人员使用约束来表达工程的要求，再求解这些约束就能得到设计变量的可行解空间，从而避免设计人员之间的冲突[20]。还有些对于多设计主体间协调问题的研究，当某个模型数据发生变化时，相应的变化信息能够实时的传递到相关位置，使其数据及时作出变化，从而维护模型数据的一致性[21, 22]。

针对多学科协同设计的数据一致性问题，有研究者提出了多学科约束-关系网络模型，通过综合约束、主义及项目管理等多种方法对多学科模型间数据的协调，进而实现了一致性维护[23, 24]。

##### 基于数据交换和共享的数据一致性研究

针对协同设计中数据共享的一致性保障，研究者们从访问控制角度出发，通过设计合适的控制算法和机制来保证一致性。一些相关的方法主要考虑读写控制和数据更新，围绕数据共享来解决设计过程相关数据中的数据不一致问题。例如有的研究者在基于P2P的分布式模式的协同设计中，主要关注各方的数据更新和交换，实现了基于设计约束集的数据更新一致性控制机制，从而保障各设计小组的数据一致性[25, 26]。

## CAx工具之间的数据交换

随着计算机技术以及工业自动化技术的不断发展，尤其是CAD、CAE、CAPP等计算机辅助（CAx）工具的广泛应用， CAx软件之间的数据交换变得非常重要，工业界迫切需要切实可行的信息交换机制来实现计算机辅助(CAx)系统之间的数据交换。很多设计师在使用协同设计平台的同时，仍然会使用CAx软件进行工作，或者他们之前的工作中积累的很多资料也是以原有CAx软件的格式来存储的。所以，一个合格的协同设计平台，应该能够兼容流行CAx软件的文件格式、并实现它们之间数据交换。

### 早期的数据交换方案

多年来，人们针对CAx系统间的数据交换问题提出了许多解决方案。其中最成功的方案已经标准化并用于数据的交换。第一批是由欧美国家组织的，把重点放在在几何图形的数据交换，包括如德国的VDA-FS格式和美国的IGES格式(initial graphics exchange specification)等。

* VDA-FS

一种CAD数据交换格式，用于表面模型在不同CAD系统之间的转换。VDA-FS名字是德语缩写，意为“自动化工业联盟——表面数据接口”。它是由德国自动化工业联盟组织（VDA）制定的。

VDA-FS后来被STEP标准所替代。

* IGES

全称是Initial Graphics Exchange Specification，是一种中立的文件格式，用于不同CAD系统之间的数字化数据交换。IGES在1980年1月被美国国家标准局首次公布。IGES文件由ASCII码组成。

利用IGES，CAD用户可以以电路图、线框、自由曲面或实体模型的形式来交换产品数据模型。IGES所支持的应用包括传统的工程绘图、分析模型以及其它的产品制造功能。

### STEP

上述的数据交换格式都已在实际中使用，但是，它们都无法描述产品在生命周期所有阶段的数据。

国际标准化组织(ISO)工业自动化与集成技术委员会(TC184)下属的第四分委会(SC4)开发了STEP来适应这种要求，标准编号为ISO 10303。它的目的就是提供一种机制，能描述一个产品在其整个生命周期内的数据，并且不依赖于具体的系统。

STEP的定义为：产品模型数据交换标准(Standard for the Exchange of Product Model Data)。

这个标准提供了一种独立于任何一个CAx系统的中性机制来描述经历整个产品生命周期的产品数据。它是一个关于产品数据计算机可理解的表示和交换的国际标准。STEP标准不是一项标准，而是一组标准的总称。它包括环境（Environment）、集成数据模型（Integrated Data Models）、顶层部分（Top Parts）等。其中Environment中定义了一种叫做Express的描述方法，Top Parts其实是针对具体的制造行业和不同的设计软件而制定了不同的标准。所以说，STEP是一组标准，而且这组标准的内容可能还会增加。

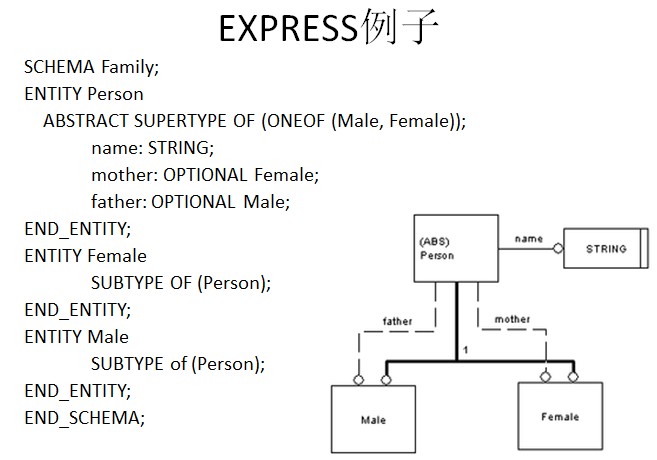


Figure EXPRESS

STEP的特点就是它不依赖具体系统的中性机制。这种性质使得它不仅适合于交换文件，也适合于作为执行和分享产品数据库和存档的基础。STEP既是一种产品信息建模技术，又是一种基于面向对象思想方法的软件实施技术。

STEP在工业，尤其是制造业中，具有重要的意义。在产品设计和制造中，使用不同数据格式的多个CAx系统之间常常要传递技术产品数据，于是相同的信息会在多个系统中冗余存储，同时还会导致错误。这个问题在制造业表现地更为突出——据美国国家标准协会估计，数据不兼容的问题对于制造业意味着上百亿元的损失。而STEP支持产品从设计到分析、制造、质量控制、测试、生产、使用、维护到废弃整个生命周期的信息交换与信息共享。一些发达国家已经在工业应用中推广STEP的应用，使得信息交换成本大幅降低，并使产品研发效率显著提高。

## 数据世系研究

### 数据世系概述

早期的数据起源研究主要集中在一些具体的科学研究领域（如生物、天文、地球科学、地理信息系统等），后来才逐渐扩展到计算机等各个领域。从几届IPAW国际会议的内容中推测，具体的研究领域包括工作流及网格计算、语义网络、普适计算、数据库与数据仓库、文件归档管理、GIS、传感器网络等，其中，在数据库和工作流领域的数据世系研究最多。

自从Y.Richard.Wang等在1990年研究异构数据库中的数据世系以来[18]，先后有很多研究者从不同的领域去研究数据世系，并且以自己的研究为基础，各自从不同角度定义了这一概念，比如：

* D.P.Lanter在研究地理信息系统（GIS）的数据世系（data lineage）时，将其定义为数据来源和深化过程的相关信息。
* Woodruff定义data lineage是包括数据起源和演化过程的数据处理的历史信息，比如源数据所属文件和标识符、数据处理中运用的算法和相关参数等具体的信息。
* Y.Cui等认为数据世系是数据演化的所有信息。
* P.Buneman将数据库应用中的数据世系（data provenance, data lineage or data pedigree）限定成数据项的源数据以及数据处理过程。
* Y.L.Simmhan等定义data provenance是数据源信息及演化历史，它决定了最终的输出数据。
* Glavic等认为，数据世系包括源数据信息以及当前数据项形成的过程（包括产生及处理过程）。

综上所述，数据世系包括两个方面，一是数据源或者数据的产生，二是数据随时间推移而演化的过程。另外，研究者们基本上都是使用data lineage或者data provenance两个词，但是二者稍有差别，前者一般指的是数据库中数据项的来源，而后者包含的内容更加丰富，用来描述各种不同应用中的数据起源和演变过程的相关信息。

### 工作流领域的数据世系管理

工作流和数据世系可以有效的支持协同设计。工作流为协同设计提供了规范的工作流程描述，而数据世系则为过程数据描述和数据存储提供了模型。将二者结合起来，就可以设计出支持产品设计全周期的协同设计工具软件，满足从协同设计工作流程的描述到过程结果数据的验证等各项需求。

数据世系信息描述了协同设计的执行过程，并将协同设计中的各个元素(人和组织、产品、资源和过程等)联系为一个整体，从而可以在协同设计中为过程数据分析、过程改进和设计优化等工作提供有效的帮助。

### 数据世系对协同设计的意义

数据世系是数据密集型设计中的一个关键技术，它是目前大规模数据管理和验证的重要技术。数据世系提供了一种记录数据项之间因果关系的模型，展现了数据的依赖关系，我们可以使用它来追踪某个特定的数据结果是从哪些输入项得到的。数据世系的这个特点使它可以在大型项目中支持对设计结果的过程重现，从而帮助设计者们分析并改进设计过程。

人员、产品、资源和过程是协同设计的主要元素，设计人员利用设计资源，经过一系列的设计过程，得到产品的设计结果，并最终得到产品。协同设计工作流描述了这一设计过程，但是在后续设计分析、优化乃至过程重构的过程中，工作流无法描述协同设计各元素之间的关系，所以不能实现对协同设计整体的分析和优化。而数据世系则可以达到这一目的。具体的讲，在协同设计工作流执行的过程中，数据世系将人员、产品、资源、过程等要素绑定成一个具备完整联系的整体。

1. 数据世系将人员和过程绑定

协同设计工作流详细描述了设计过程，在工作流执行时，数据世系则将人员参与流程的过程记录下来，比如具体每一个任务的参与者、执行时每个参与者的操作；同时，世系还记录了不同设计人员或组织之间的协同操作，比如参数的交流、通信等等。

1. 数据世系将过程和资源绑定

工作执行时用到哪些资源，如何使用这些资源，哪些设计者具体使用了这些资源，他们的具体操作是什么，利用这些资源得到了何种数据结果……这些问题综合的描述了过程同资源的关联，它们都在工作流执行过程中被数据世系一一记录下来。

1. 数据世系将过程与过程数据绑定

同上面所述一致，数据世系同样记录了过程数据结果得出的过程，从源数据、中间结果，到最终结果，以及计算所经过的具体步骤、经历了哪些设计任务。实际上，数据世系记录了产品各部分与设计过程的关联。

实际上，工作流和数据世系描述了协同设计的不同侧面，工作流是一个即将执行的协同设计过程，而数据世系则记录了协同设计过程的执行历史。从这个观点出发，在协同设计过程管理中，数据世系与工作流应用在不同的部分，工作流当然是用在协同设计过程的设计与执行上，而数据世系则表示设计的历程和结果，常用在对设计结果的验证上。数据世系与工作流相辅相成，从协同设计的规划到结果，实现对整个过程的支持。

在大规模协同设计支撑环境中，数据处理、存储和设计过程记录是关键功能。而数据世系则由于可以描述数据处理过程和过程数据结果而显得很重要。但是，在目前的协同设计支撑环境的开发中，我们缺少围绕产品设计结构的数据世系标准及工具，这导致在协同设计中难以建立起局部设计结果、具体设计行为与整体设计体系之间的关联。另外，协同设计过程中往往存在为了优化而反复进行的迭代设计，这导致多版本数据和多设计方案的存在，而其中数据不一致的问题容易导致返工。所以，在协同设计过程数据管理和数据一致性保障等方面还需要根据其特定需求给出相应的解决方案。

### 协同设计工作流的数据世系管理

协同设计系统是信息密集型的流程管理系统，其中动态数据的管理是系统的一个核心功能，而基于数据世系的管理方式是有效管理动态数据的一种方法。这种方法的具体内容则是结合协同设计工作流，数据世系管理围绕流程，结合数值参数、文档等元素展开。

协同设计中，数据世系的应用主要是以下几种方式。

1. 基于过程数据世系的数据演化追踪。

数据世系记录了过程数据之间的依赖关系及演化历程，所以，它可以解决下面几个具体的问题：

* 为了得到某个过程数据结果，使用了哪些数据？
* 某个数据项被发现有问题，哪些数据与之有关？
* 某个设计任务实例出现了变更，哪些过程数据受到了影响？

1. 流程回退与重复执行

根据记录的数据世系信息，流程能够轻易的回退到之前的某个任务，而当流程执行出现问题时也可以利用数据世系来恢复到之前正确的状态中去。如果某些输入参数发生变化，流程能利用数据世系来决定哪些过程数据结果需要重新计算，并尽量避免不必要的冗余计算。

1. 数据世系是加强设计者对流程控制和分析的有效工具

基于数据世系，设计者们可以随时查看当前工作流的演化过程，这有利于对协同设计工作流进行实时的改进，加强对流程的分析和控制。

数据世系在协同设计支撑环境中的应用架构如下图所示。



Figure 协同设计中数据世系应用

在协同设计中应用数据世系，需要建立两个核心模块，一是要对数据世系信息进行收集，二是在需要时提取保存的世系信息。其中收集模块比较重要和复杂，因为它要在协同设计流程执行中实时收集数据。另外，应用中还需要一套数据世系管理工具，它可调用核心模块的功能，协同设计平台的用户使用这套工具来实现对数据世系信息的管理。

# 基于数据世系的数据管理方案

## 协同设计“世系”的形成

数据世系是随着协同设计流程的执行而形成的。在工作流的设计阶段，参数未经过计算，还没有具体的数值，它所代表的只是产品的模型，指出了产品各部分设计的具体内容是什么，设计者们只有参数之间的关联和约束关系信息。而随着流程的执行，参数实例出现，参数的值逐渐被计算出来，于是世系就形成了。具体的讲，世系的内容包括如下几个方面：

* 参数的值：具体某个参数实例的数值；
* 参数的计算过程：某个参数实例是由哪个任务实例经过执行而产生的，具体的计算方法是什么（比如公式、外部工具等）；
* 参数的依赖关系：某个输出参数的计算用到的输入参数是什么。

于是，通过以上内容，我们可以得到参数随着协同设计工作流程的执行而演化的完整过程。无论哪个参数实例，通过数据世系的信息我们可以追踪它完整的计算过程，以及它所依赖的参数来源，并一直追溯到最初始的设计资源，比如设计者直接给定的某个输入参数，或者是某个物理常数。根据这些信息，设计人员可以轻易的重现整个流程执行过程；世系信息还可以用来分析设计过程中某个参数对后续计算的影响，等等。

显而易见，数据世系所包含的内容非常丰富，几乎将整个协同设计的过程完全记录下来，又细致到了参数的粒度。这些信息对协同设计来说有很大的作用，文章后面就对数据世系在协同设计中的应用进行描述。

## 协同设计中数据世系的层次模型

协同设计工作流代表了产品设计的过程，工作流中的每个任务表示的是整个产品设计的一部分，比如某个零部件的设计。参数是产品设计的数字化表示，所有的参数组成了一个完整的参数体系，这是产品设计的重要组成部分。参数刚被创建时是保存在参数库中，而在创建协同设计工作流并划分设计任务的时候，参数被绑定到相应的任务上，作为输入或者输出。也就是说，任务使用参数进行计算，或者通过任务的执行来计算参数。

如下图所示，协同设计中数据的概念分为两个层次。

* 模型层：在工作流执行前，设计人员定义的参数、任务和约束都是“模型层”的概念。在模型层中，参数依赖于产品结构，每个参数都与产品设计的某个部分（比如某个零部件）相关联。
* 实例层：工作流执行过程中，由设计人员创建的任务实例、参数实例。只有经过任务实例的执行，与之相关的参数才有值，这就是参数实例。一个任务可以对应多个任务实例，它们就是任务的不同版本。同样的，同一任务的不同实例所对应的参数实例就是同一参数的不同版本。

模型层

实例层

参数

任务

参数实例

任务实例

约束

流程

角色

流程实例

Figure 数据的层次结构

实例层的元素是以模型层为模板创建的，其中任务实例表示了任务的一个实际配置，而流程实例则是流程的一次实际执行。所以模型层与实例层元素之间是一对多的关系。

下图表示了模型层的任务、参数与实例层任务实例、参数实例之间的关系。

模型层

实例层

任务t

参数1

参数2

参数tn

任务实例t1

参数实例t11

参数实例t12

参数实例t1n

任务实例t2

参数实例t21

参数实例t22

参数实例t2n

Figure 模型层与实例层的关系

## 世系的内容

协同设计中，数据世系包括了上面提到的模型层和实例层的内容。但世系的演化过程则主要体现在实例层上，是设计流程的执行、迭代等使得新的任务实例不断被创建，随之产生的就是任务与参数的多个版本。这些任务与参数的版本，以及它们之间的依赖关系，就构成了协同设计中数据世系的主体框架，而其它的诸如人员、约束等等则附加在主体部分上。再由于参数版本实际上取决于它所关联的任务实例的版本，所以设计流程中数据世系的框架用任务实例来表示即可。比如下图就体现了一个流程中任务多版本现象以及任务之间的参数传递。

### 连通图表示数据世系信息

这是一个简单流程，左边显示了因流程执行而生成的所有任务版本，但是这样会使得观察者无法确定每个版本的上游结点的版本，比如T2的版本V1与V2分别使用的上游T1的哪个版本，而右边的图就比较明确。



Figure 多版本任务的数据世系

这样的连通图通清晰的展现多版本任务的关联关系，从而可以表示设计流程中的数据世系信息。进一步说，只要把流程执行而形成的所有这样的连通图都收集完整，那么任务版本的世系也就被建立起来了。

进而，参数的世系信息也可以从任务版本世系中得到。

### 协同设计过程数据的世系

根据上面的分析，我们可以这样定义协同设计中的数据世系：

**Data-Provenance = {{Task}, {Dependency}, {Parameter}, {P-Dependency}}**

* {Task}：流程中所有的任务实例版本的集合。
* {Dependency}：{Task}集合元素之间关联关系的集合。
* {Parameter}：{Task}中每个任务实例所关联的参数实例。
* {P-Dependency}：{Task}与{Parameter}元素的关联关系。

上面这个定义其实就是表示了流程中所有的多版本、带参数依赖的连通图。在实际设计中，我们不可能保存所有的连通图，两个不同的图之间差别可能仅在一个任务的版本上，全都保存会造成存储空间的浪费。所以，只要保存图中结点和关联关系，利用这些信息就可以拼凑出所有的连通图，也就是得到了流程的完整数据世系。

## 基于数据世系的数据管理

工作流中的任务与参数的关系非常密切，在协同设计的工作流程中，每个设计任务都可能与若干个参数绑定，于是，当流程执行时，带参数的任务所产生的每个任务实例都与若干数量的参数实例相关联。所以，在收集参数的数据世系信息时，与任务及任务实例相关的信息也是非常重要的。尤其是在流程执行过程中，一个带参数任务的执行就可能表示了若干个参数的计算过程。

所以，我们的数据管理方案在收集流程执行时的数据世系信息时，以任务实例为单位进行收集，再辐射到与之相关的参数，这样操作更加方便，而且也更能体现流程的执行历程。而实际上这个机制将协同设计过程中的动态数据都进行了管理。

下面，文章首先介绍协同设计平台参数库的框架，然后对基于数据世系的过程数据管理方案进行描述。

### 参数库

协同设计平台的参数库，顾名思义，就是存储设计参数的仓库。但是参数库不能只是简单的将所有的数据保存起来，它必须适应协同设计过程的需求。协同设计项目一般都是规模较大的跨学科项目，涉及人员多、流程复杂、计算量大，所以对数据处理的要求比较严格，设计人员的设计或计算结果不能直接保存进参数库，因为这些数据可能会存在错误或发生变更。下图展示了本文在协同设计平台中采用的参数库模型。

**试算空间**

**审核空间**

**确认空间**

**次版本库**

**主版本库**

Figure 20 参数库

如上图所示，整个协同设计参数库分为三个部分：试算空间、审核空间、确认空间。相应的版本库分为两类：一是主版本库，它在平台中只有一个；二是次版本库，每个组织或个人都可以拥有一个独立的次版本库，具体的分配在协同设计项目规划时决定。这样的设计是为了适应协同设计工作流的特点，在工作流执行过程中，新产生的过程数据需要经过审核，才能成为一个正式的版本。所以，在审核之前，参数只能保存在相关设计人员的个人版本库中，提交审核时被复制到审核空间中暂存，审核通过后进入确认空间，即保存在主版本库内。

主版本库一般是集中在一个存储空间中，并不采用分布式的存储方式，而次版本库则可能是地理上分布的。因为协同设计的的特点之一就是设计人员地理上的分布性，所以为了设计人员使用方便，与某人员或组织相关的次版本库可能与主版本库并不在同一地方保存。这样也并不会影响协同设计的进行，因为只有设计者把自己的设计结果通过审核后，才算出现了一个正式的版本，被转存到主版本库中。其它设计者若要使用其他人的结果，也必须去找通过审核的、保存在主版本库中的数据。

### 动态过程数据管理框架

协同设计过程中的参数、流程执行路径等过程数据都是动态数据，不能像文件、图纸等静态数据那样用采用PDM（产品数据管理）进行管理。本文采用了一套基于数据世系的动态数据管理方法，利用参数与产品结构的关联、参数与设计任务之间的关联将动态数据形成一个结构化的整体，再在协同设计工作流执行过程中动态收集过程数据，提供对于参数等动态数据的保存、提取、导出和展示等功能。管理框架如下图所示。



Figure 21 基于数据世系的参数管理框架

#### 模块功能描述

我们的管理框架从产品结构和协同设计流程两个角度分别描述动态过程数据，即从两个方向进行数据管理，如下所示。

* 过程数据收集🡪运行时过程数据管理🡪里程碑导出
* 过程数据收集🡪关联关系管理🡪参数管理🡪参数导出

其中，每个模块的具体内容如下：

* 过程数据收集

该模块与工作流的关系密切，在流程执行时收集其执行路径、各个任务执行过程和结果，以及相关参数的值。

* 运行时过程数据管理

“过程数据收集”模块收集到流程执行时产生的过程数据，然后这些数据就被交给“运行时过程数据管理”模块来处理，进行组织和展示等。具体管理内容包括：流程执行实例的版本管理、任务实例的版本管理、里程碑管理。所谓里程碑管理，是指保存流程执行时某一时刻的状态，目标是能够在将来从这一状态开始恢复流程的执行，从而实现流程的redo。

* 过程数据导出——流程基线导出

导出某基线中的流程执行状态，以实施流程redo。流程执行状态的内容包括：已完成任务、流程执行路径、待执行的任务、待执行任务的输入参数等等信息。

* 关联关系管理

关联关系一方面指的是参数与任务的关联，另一方面指参数之间的约束关系。关联关系管理就是要提供对这些关系进行编辑和更改的功能。

* 参数管理

产品的每一部分设计中都有很多参数，这些参数可以依据产品结构树来建立参数体系。这些参数就是“参数模型”，它们的全体就组成了“参数体系”。流程的执行、任务的执行会产生参数实例，而且不同的任务实例会产生不同的参数实例版本。同一次流程执行中产生的一系列参数实例就形成了参数体系的一个批次。在已有批次结果的基础上，重新配置批次输入，对设计流程进行迭代，产生新的批次输出结果。

* 过程数据导出——参数导出

选择参数体系的某个批次，将其导出，它可以在将来再导入到设计流程中。不同的批次就是设计流程的不同配置方案。

总的来说，这些模块的功能分为两个方面，一部分是用来收集协同设计中的数据世系信息，用于建立世系；另一部分是提取数据世系，进行数据的分析、流程重构或其它操作。

下面分别对两个方面的功能设计进行详细的描述。

### 数据收集

数据的收集分为两个部分，一是对协同设计资源、人员、工作流设计等静态信息的收集，二是对流程执行中产生的任务、参数版本和参数计算过程等动态数据的收集。下面分别对这两类数据的收集进行描述。

#### 静态数据收集

静态的数据世系信息主要是针对协同设计的人员、设计资源以及模型层的流程、任务、参数等数据。由于这部分的信息在设计流程开始执行之前就基本固定，在执行时一般也不会更改，所以对这些信息的收集也是比较简单的。主要就是两个方面的内容，一是形成产品结构树，将参数与产品设计结构的各部分建立联系；二是保存流程设计信息，将任务分配、人员分配、参数与任务关联等信息进行存储。

Figure 22 产品结构与参数关联

上图表示的是一个简单的参数与产品结构相关联的例子。类似的，任务也与参数相关联，或者说参数绑定在任务上。在协同设计工作流的设计阶段，各个参数就与设计任务相绑定，作为输入参数或者输出参数。

Figure 23 参数与任务关联

当然，与任务相关的不只是参数，还有人员、资源、计算公式等等。这些静态数据都是数据世系信息的一部分。

在与协同设计相关的数据世系信息中，上面所说的这些静态数据所占的比例较小，而动态数据则相对较多。这是因为静态数据一般不会发生变动，在流程设计阶段就几乎都确定下来，而动态数据则是在流程执行中动态生成，还会有多版本，所以数量庞大。不过静态数据虽少，但它却同样重要，因为动态数据都是以静态数据为模型产生的，比如任务实例、参数实例都是在静态的任务、参数模型的基础上实例化，继而多版本化，从而衍生出大量的动态过程数据。所以，在使用数据世系的信息时，这些静态数据是我们追踪参数计算过程、分析协同设计流程和结果的重要参照。

#### 动态数据收集

对流程执行中产生的动态过程数据的收集，是建立数据世系的重点和难点。首先，我们要清楚协同设计任务的具体过程与特点，然后再根据这些特点来设计合适的收集方式。此外，由于收集的过程与流程执行密切相关，所以一般的方法就是设置监听器（listener），根据流程和任务的状态变化来收集过程信息。

##### 任务实例执行及审核过程

协同设计工作流中每个设计任务都由某设计人员或组织承担，设计者进行操作以推动任务的执行，执行后将结果提交，再由其他人员进行审核。任务的执行与审核的流程如下面的活动图所示。



Figure 24 任务执行与审核

上图中有如下几点需要注意：

* 对于任务相关参数的计算，设计者有若干种计算方法可以选择：一是直接填写输出参数，即设计者自行计算并将结果提交；二是填写输入参数的值和公式，由系统根据公式计算结果；三是可以调用外部工具，如Matlab等进行计算，结果自动被系统接受。
* 得到任务的执行结果后，相关参数的计算结果并不是直接保存在主版本库中。必须要经过一个审核的过程，通过审核方能将任务实例和相关参数的新版本保存到主版本库。在审核之前，该结果保存在次版本库中，即只有参与该任务的人员才能看到与该任务相关的过程数据。
* 若审核不通过，则任务实例不成为正式的版本，但它仍然保存在次版本库中。

##### 动态数据收集方法

前面提到过，对于流程执行产生的动态过程数据，我们以任务实例为单位进行数据世系信息的收集。具体做法就是监听任务实例的执行状态，对任何参数、人员、参数计算方法等信息的变动都要做出反应，及时的记录相关的动态数据，保证数据世系的完整性和可靠性。

对动态数据收集过程描述可以参考下面的顺序图。



Figure 25 顺序图 - 流程动态数据收集

从上图可以看出，动态数据收集仅在任务实例发生变化时进行。可能的场景是任务实例第一次执行或者后来设计人员对任务实例进行变更操作。先是设计人员提交变更，然后由审核人员做出审核意见，如果审核通过，则监听器收集任务实例相关的过程数据。需要注意的是，数据收集并不是在设计人员完成对任务实例的修改时就立即发生，而必须等待审核通过。这是因为数据世系收集的都是正式的任务与参数版本，即前面提到的“主版本库”中的数据；而“次版本库”中的数据则并不在考虑范围内。这样做有几个优点：

* 未经过审核的版本有一定的概率通不过审核，进而很可能被舍弃，收集这些数据是没有意义的。
* 次版本库可能是分布式的，即与任务相关设计人员使用的存储空间与协同设计支撑平台不在同一地理位置，这会造成收集过程在实际操作中的困难。
* 经过审核的数据被保存进主版本库，这些数据的存储格式是事先定制好的，收集起来很方便；而未经审核的设计数据只由相关设计人员保存在次版本库，这些数据很格式并不一致，即存在“异构”问题，这也会造成收集操作的困难。

### 数据提取

相对于数据的收集来讲，数据的提取还是相对简单的。数据提取模块主要提供的功能就是根据使用者的要求，从收集的数据中提取需要的结果。一般来讲，提取的数据包括参数的批次、里程碑等。下面描述从收集的数据世系信息中提取某个参数批次的过程。

#### 参数批次的导出

零部件1

任务实例1a

相关参数

任务实例1b

相关参数

零部件2

任务实例2a

相关参数

任务实例2b

相关参数

任务实例2c

相关参数

Figure 26里程碑与参数批次

##### 参数批次

如上图所示，产品设计分为一个个零部件的设计，每个零部件的设计一般由一个设计任务或者子流程来完成。设计结果是一堆参数的值，这些数据就是这个零部件的数字化表示。当设计过程结束，所有设计任务都执行并得到相关的参数实例，设计人员就得到了一个参数批次。

参数的批次，指的是产品设计中所有参数的一个配置，可以代表对产品的一个可行的设计结果。举个例子，一个产品设计中有100个参数V1, V2,… V100，对于这些参数的一组实例vi1, vi2, …, vi100，如果实例之间数据一致性得到保证，那么组实例就是一个批次。

导出的参数批次可以再次导入到设计流程中，相当于流程恢复到一个里程碑处，继续运行设计流程。

##### 导出方式

基于数据世系的参数管理方案中，参数实例信息都在动态过程数据收集过程中被保存在了数据世系中。所以，当协同设计需要时，我们完全能够从数据世系中导出一个参数批次。而且，由于每个任务都可能存在多个通过审核的版本，所以，根据收集到的参数信息，我们也可以得到多个参数批次，每个批次都对应了流程执行的一个实例。

系统会从数据世系中找出所有可能的参数版本组合，使用者可以自由选择其中的一个批次，如上页图所示。

这样做会不会导致数据不一致的问题呢？不会。因为我们并不是随机的在数据世系中选择任务实例与参数实例的版本，而是通过所有初始参数的配置来寻找参数批次。如下图所示。

参数的一个批次

数据世系

任务实例的组合

（协同设计流程实例）

Figure 27 导出参数批次

一个常见的应用场景如上图所示。首先设计者希望得到与一个里程碑相关的参数批次，于是系统得到一个里程碑的信息，并从中得到初始参数的一个配置，然后根据这些信息去数据世系中寻找符合条件的任务实例版本，最后将这些任务实例相关的参数实例版本组织成一个参数批次。

# 协同设计中的数据一致性保障机制

本章先是探究了协同设计中不一致问题的来源，然后提出了几个检测数据一致性的方案和算法。最后本文在协同设计平台中采用的方案是基于数据世系的，方案中用于判断一致性的重要数据就是在数据世系信息的收集阶段，通过对流程中动态过程数据的收集而得到的。

## 数据不一致问题的来源

在协同设计领域，由于设计的复杂性以及协同设计的一些特性，数据不一致的问题更加突出。要设计出支持协同设计流程的数据一致性保障机制，需要对不一致问题的来源进行深入了解。

第一章中也对数据一致性的需求进行过介绍，主要问题来自于如下三个方面：

* 流程的迭代性导致同一任务的执行过程中产生多版本过程数据；
* 并发性导致流程执行过程中多分支之间存在数据不一致性；
* 数据协同的特点导致共享同一参数的任务可能取到参数的不同版本数据。

其中流程的并发性和迭代性对数据不一致性问题的影响较大，为了设计出解决协同设计实际问题的数据一致性保障机制，下面对流程的这些特性进行深入的探究。

### 产品并行开发过程中的迭代

在协同设计的环境下，流程的并行是必需的，而过程迭代是并行化产品开发过程的典型特征，在协同设计工作流中是不可少的。后继任务在前驱任务还未结束的情况下并行的开展，必然会由于输入信息的不完整引起过程的耦合迭代。过程迭代是导致产品并行开发成本增加的主要因素。

产品并行开发过程的重要特征之一就是迭代，尤其是在设计早期，通过小范围内的上下游设计活动的多次迭代，能够及早发现并减少设计错误。这里的迭代指的是后续相耦合任务的执行或修改，导致前驱任务的重新执行，这种重新执行的原因可能是信息输入的更新，或者是任务需求发生了变化。输入信息的更新可能是上游活动产生，或者在并行的耦合活动之间产生，或者是下游活动对上游活动的反馈。下图展示的就是一个迭代过程。



Figure 一个简单的迭代

上图中，T1、T2是两个相互耦合的任务，I1、I2分别是任务的输入数据，O1、O2分别是任务的输出数据，C1、C2表示任务的约束，E1、E2表示对任务的一致性评审。则当时，过程产生迭代，过程中的任务需要重新执行。



Figure 串行与并行迭代

从迭代的产生原因来看，首先，迭代过程涉及多个活动，活动之间可能是串行的关系或者并行耦合的关系，于是迭代可以分为串行迭代和并行迭代，如上图所示。第二，迭代过程可能是后继任务影响前面已经完成的任务，延长设计时间，产生不利的影响。第三，迭代使得设计错误尽早被发现和纠正，这是迭代对设计过程有利的一面。

迭代中，耦合是一个概念。如上图中并行迭代可以看出，所谓Ta与Tb耦合，指的是任务Ta需要任务Tb的信息输入，而任务Tb也需要任务Ta的信息输入。

设计流程中存在迭代时，对设计问题的求解几乎不可能经过一次执行就完成。Hubka在它的文章中指出，迭代活动类似于数据上的逼近算法，求解时先做出一个初始的假设，然后在后面的计算中得出一个接近的结果，再用这个结果来计算更精确的结果，理想状况下经过有限次的重复就可以得到目标解。于是，迭代过程实际上是一个不断修正以得到更精确结果的过程。

现实的产品设计过程中，设计人员在规划时可能会在并行开发中故意设置一些局部的迭代，以得到比较满意的设计结果，以避免最终结果不合格造成大范围的返工。同时，在实践中，各项设计任务并非严格按照逻辑顺序开展，为了保证进度和质量，设计人员可能会主动将前面的任务信息进行“预发布”，后继活动则会对预发布信息进行反馈，这也会形成迭代，而且这种迭代在并行工程中普遍存在。不过还有一些“无意的”迭代，是因为产品需求在设计过程中发生了变化，或者前面的设计任务出现了错误而导致的。总之，在实际的产品设计和开发中，迭代是不可避免的，而且是十分重要的。

### 协同设计流程中的不一致问题

协同设计流程中的数据不一致问题是由于任务实例的多版本性而产生的。前面介绍过流程的迭代会导致同一任务会产生不同结果，即流程中会出现同一任务的不同实例（版本），于是任务实例就具备了多版本的特点。而不同任务版本的参数之间是存在不一致性的，所以这些任务实例后继的任务结点也会遇到不一致的问题。下面通过一个例子来说明工作流程中的数据不一致问题。

这是一个简单的流程，圆圈内标识的是这个结点的任务模型，圆圈右边标出任务的几个实例版本，而线上标识的则是传递参数的任务版本。我们用T1-V1表示“T1任务的V1版本”。那么T1🡪T2连线上V1就表示：任务实例T1-V1的输出参数是任务实例T2-V1与T2-V2的输入参数，而T2由于执行了两次所以产生了不同的两个版本。

从图中可以看出，任务T2有两个实例版本V1和V2，其中V1的输出参数是T4-V1的输入参数，而V2的输出参数则传递到了T5-V1，于是，T4-V1与T5-V1之间就产生了数据不一致。所以，T6的输入参数有的来自T4-V1，有的来自T5-V1，这是不可行的，必须选用其它的版本来保证数据的一致性。



Figure 多版本造成不一致

另外，我们再来看一下上图中的T3与T4、T5之间是否具有一致性。虽然T4-V1与T5-V1之间不一致，但是T3的各个版本都与T4-V1是一致的，因为它们的计算历程中并未使用不同版本的参数。同理，T3的三个版本也都与T5-V1是一致的。可见，一致性并没有传递性。

总之，设计流程中的数据一致性指的是计算历程中是否使用了同样版本的参数。实际在设计过程中，参与任务设计的设计者需要从很多版本的输入数据中选择一个合理的、一致的组合，保证输入参数之间的一致性。比如T6的设计者就要选择T4、T5的版本组合，保证它们之间具有一致性。

下面，本文描述了几种检测数据一致性的算法，并作出了相应的评价。

## 数据一致性检测算法及优化

数据一致性可以指两个参数实例，也可以指两个任务实例的一致性。下面所描述的算法中，本文都是按照两个任务实例的一致性来处理的。这是因为，参数实例是跟任务实例挂钩的，两个参数的一致性就可以归结到计算这两个参数的任务实例的一致性上面。下面的算法描述中也是输入的任务实例来判断它们之间的一致性。

### 遍历式算法

最直观的检测两个任务版本一致性的算法就是对它们的完整计算过程进行考察，包括向上游任务的追溯，直到确定其一致性或发现数据的不一致问题。

#### 算法描述

检测两个任务实例之间的一致性。

输入：Ta，Tb两个任务实例。

步骤：

1. 对Ta的上游任务进行完整的追溯。即，针对Ta的每个输入参数v，找到v的来源，并继续向上追溯直到源头的那个任务实例。
   1. 如果追溯过程中发现其中一个上游任务是Tb，那么Ta、Tb是一致的。返回。
   2. 否则，继续步骤2.
2. 对Tb向上游追溯，每向上追溯到一个任务实例t，就做如下操作：
   1. 检查t是否与之前追溯的某个任务实例t0共用同一个任务模型（即追溯过程是否形成了环），如果是，那么跳过b、c步骤。
   2. 检查t是否就是Ta，如果是，那么Ta与Tb是一致的，并返回。
   3. 对照Ta的完整计算历程（包括Ta），检查t是否与其中的某个任务实例t1共用同一个任务模型。
      1. 如果存在这样的t1，那么比较t与t1的版本。若t与t1版本不同，则Ta与Tb是不一致的，并返回。
3. 如果完成对Tb的追溯后，仍然没有发现Ta与Tb不一致，那么Ta与Tb是一致的。
4. 算法结束。

输出：Ta与Tb一致 / Ta与Tb不一致。

#### 优缺点

* 优点：不用借助额外的信息或记录，只是使用协同设计工作流的相关数据就可以实现这一算法。
* 缺点：时间爆炸。
  + 协同设计中数据多版本、流程多方案的特点使得该算法的执行效率难以得到保障。
  + 在不断的向上游追溯，这个过程中可能会遇到之前已经检查过的任务实例，造成重复，增加了时间开支。
  + 工作流的数据存储方式可能不适应该算法的检索方式，这又无形中增加了算法中每个步骤的执行时间。

### 优化－空间换时间

对遍历式算法的优化思想是“空间换时间”，具体来讲，就是利用数据世系信息中流程执行过程的动态记录，来排除不一致的数据。

#### 数据世系信息收集时的处理

系统通过监听流程的执行来收集动态过程数据，这个过程在第3章中详细描述过。为了提高一致性检测算法的时间效率，在记录过程数据时，需要多记录一样信息，那就是任务实例之间的依赖关系。

任务实例Ta

任务实例Tb

参数实例V1

任务实例Ta

任务实例Tb

参数实例V1

任务实例Tb

参数实例V2

Figure 任务实例间的参数依赖

* 依赖关系：如果任务实例Ta的某个输入参数来自于任务实例Tb，那么Ta、Tb之间存在依赖关系Ta🡪Tb；
* 传递性：如果任务实例Ta、Tb之间存在依赖关系Ta🡪Tb，而Tb、Tc之间又存在依赖关系Tb🡪Tc，那么Ta、Tc之间存在依赖关系Ta🡪Tc。

T1

T2

T1

Ta

Tb

T1

T2

T1

Ta

Ta

Figure 收集动态数据-记录Ta、Tb的依赖关系

在数据世系的收集阶段，所有的依赖关系都要记录下来。不只是直接的依赖关系，还包括传递性导致的依赖关系。也就是说，当Ta、Tb建立起依赖关系Ta🡪Tb时，Ta所依赖的全部任务实例T1,T2,…,Tn都与Tb建立起了依赖关系T1🡪Tb, T2🡪Tb,…, Tn🡪Tb。

这些依赖关系信息的数量巨大，但是它们的收集过程并不复杂，而且这些信息对于一高效的一致性检测算法非常有用。

#### 算法描述

检测两个任务实例之间是否一致。

输入：任务实例Ta，Tb.

步骤：

1. 检查Ta与Tb之间是否存在依赖关系。
   1. 如果Ta🡪Tb或者Tb🡪Ta，则Ta与Tb具备数据一致性，返回。
   2. 否则，进行步骤2。
2. 找出所有Ta依赖的任务实例的集合TaList：Ta1, Ta2, …, Tam，以及Tb依赖的所有任务实例的集合TbList：Tb1, Tb2, …, Tbn。
3. 找出TaList与TbList之中具有相同任务模型的任务实例，判断它们的版本是否一致。（比如，Ta1与Tb1都是任务TaskModel1的任务实例，那么比较Ta1与Tb1的版本是否一样，若不一样，则它们不一致）。
   1. 如果TaList与TbList中找出一对版本不相同的任务实例（比如Ta1与Tb1版本不同），那么Ta与Tb就是不一致的。返回。
   2. 如果没有找出版本不相同的任务实例，则说明Ta与Tb是一致的。返回。
4. 算法结束。

输出：Ta与Tb一致 / Ta与Tb不一致。

#### 优缺点

* 优点：
  + 算法的时间效率较好，没有无穷无尽的向上游追溯的过程，节约了时间开支。
  + 不会重复检查已经检查过的任务实例，提高了效率。
* 缺点：
  + 空间开支较大。在数据世系信息的收集过程中，任务依赖关系都要记录下来，这是很大的存储开销。

虽然空间开支很大，但是现在的数据存储软硬件方面都发展很快，而且因为协同设计中的数据本来就数量巨大，所以协同设计支撑环境所使用的数据库一般都是大容量、高效率的企业级数据库，因此，这个算法对存储空间的要求是可以得到满足的。

### 改进方案

上述优化的算法用大量空间换取了大量时间，导致空间代价较大。有个折中的方案可以考虑，如下所述。

上面的算法在存储依赖关系时，不只保存了直接依赖关系，还保存了因传递而造成的依赖关系。这导致了冗余信息，因为传递的依赖是可以由直接依赖关系扮演出来的。所以，这个改进的方案就是：

* 收集动态过程数据时，仅保存直接的依赖关系。
* 进行数据一致性检测时，再通过直接依赖关系在存储空间中找到所有传递的依赖关系。

这样的改进会牺牲一些算法时间，但是可以节约很多的空间开支。尤其是在流程路径比较长、涉及的任务较多时，这种方案所节约的空间开支非常可观。

# 协同设计参数管理工具的设计与实现

## 参数管理工具的需求

### 参数管理工具的应用场景

在基于参数的协同设计环境中，参数既是设计的基础，即协同设计工作流程中需要输入部分基础参数（例如总体重量，理想速度）以指标的形式指导设计流程的执行；也是设计的重要结果，即设计流程中会产生对产品的详细描述（例如在卫星设计中，描述轨道的各个参数可以作为轨道建设的蓝图）。所以对于基于参数的协同设计来说，多次不同数据环境配置的执行，就是配置流程执行所用的产品基础参数集合，而设计流程的执行结果就是整个产品参数体系的一套实例。

下面给出一个模拟的应用场景。

前提：在卫星设计中，卫星轨道、卫星尺寸、载荷等需要分开设计，而三者之间又有密切的关联。必须先设定好恒星轨道高度H，以及关于这个轨道的其它各项参数，才能根据这些参数来设计卫星的载荷G，然后再根据G来设计初步的卫星尺寸。

设计目标：通过不同的卫星设计方案，从而让卫星能够在相应的轨道上平稳运行。

设计过程：

1）设计不同轨道方案，产生了关于这个轨道高度的方案配置；

2）然后，基于约束，调整卫星载荷G以及与它相关的各项参数（比如卫星各部件的重量等），形成计算初步卫星尺寸所需的参数配置方案。

3）最后，基于参数配置的方案执行卫星尺寸设计流程，产生卫星的初步尺寸，形成卫星设计方案。

其中，设计参数体系可以将流程与产品模型关联起来，从而可以基于设计参数体系来建立面向产品的数据世系模型，而这主要由于设计参数体系的两个优势，如下所示：

* 设计参数体系在参数化技术的支持下可以有效的描述产品，给出产品各部件的属性和特征；
* 设计参数体系与整个设计流程密切相关，流程的每个设计任务的执行都会改变部分参数的值并产生新的版本，而且根据任务的输入参数和输出参数还可以确定参数体系内部不同参数的不同版本间的关联关系。

所以基于设计参数体系建立面向产品的数据世系模型可以将过程模型与整体产品模型关联在一起，为后续过程多方案设计中围绕产品的过程方案分析和选择提供依据。

### 参数管理工具的需求分析

#### 用况图



Figure 参数管理用况

#### 各用况的详细说明

* **工程定义**

**主要参与者：**总工程师、设计师

**前置条件：**用户总工程师或设计师的身份得到确认。

**成功后的保证：**用户能够对工程中的任务、公式、数据等进行编辑。

**主要交互场景：**

1. 相关用户登录协同设计平台；
2. 用户选择一个工程；
3. 用户对选定的工程进行编辑。

**扩展：**

3a) 任务分解：如果用户是总工程师，则用户可以重新定义、修改工程中现有的任务，或者新建子任务。

3b) 约束定义：总工程师和设计师均可对任务的约束表达式进行编辑、删除，或者新建约束。

3c) 计算公式定义：如果用户身份是设计师，则用户可以对任务已有的计算公式进行编辑、删除，或者新建计算公式。

* **过程数据监控**

**主要参与者：**总工程师、技术状态管理员

**前置条件：**用户总工程师或技术状态管理员的身份得到确认。

**成功后的保证：**用户能够查看工程中各任务的所有合理版本，以及相关联的任务之间数据的交互状况。

**主要交互场景：**

1. 用户登录协同设计平台；
2. 用户选择一个工程，查看它的过程数据。

**扩展：**

2a) 查看任务历史：若用户查看选定工程的任务历史，则：

1. 用户选择其所关心的某个任务实例；
2. 平台向用户展示选定任务实例的“任务历史图”，即以图的形式该任务实例的参数来源，以及这些任务实例之间参数的传递关系（用线表示）；
3. 若用户在“任务历史图”中选择某条线，则平台会展示它所传递的参数内容，包括参数名、值等。

2b) 查看流程基线：若用户查看选定工程的流程基线，则：

1. 用户选择某个任务实例；
2. 平台向用户展示该任务实例的“基线图”，即该任务实例以及其子任务所组成的树状结构，其中还会显示任务实例之间的参数传递关系；

* **参数库管理**

**主要参与者：**技术状态管理员

**前置条件：**用户技术状态管理员的身份得到确认。

**成功后的保证：**用户可以查看并修改参数库中的所有参数定义。

**主要交互场景：**

1. 用户登录协同设计平台；
2. 用户选择参数管理功能；
3. 平台以树状来展示所有的参数分组，一般是以绑定到同一任务的参数为一组；
4. 用户选择一个参数分组；
5. 平台展示出该分组中的所有参数定义；
6. 用户可以对这些参数的定义进行修改、增删等操作。

## 参数管理工具的设计

### 主要的类

下面的类图列出了与参数管理相关的主要几个类以及它们之间的关联关系。



Figure 类图-参数管理工具

在协同设计平台中，参数和任务的模型对应的类是“参数”和“任务”，而具体到设计流程执行过程中的参数和任务则由参数实例与任务实例的对象表示。其中约束既可能表示任务中参数的约束关系，也可能表示参数的计算公式。

上图中，类“任务”与自身有关联，表示任务之间存在依赖关系，比如前驱-后继这样的逻辑关系。同样的，任务实例之间也存在依赖关系，这种依赖关系一般都会包含参数的传递。而任务实例的对象加上参数依赖，就会导致数据世系的产生。所以，对于协同设计流程中过程数据，组成其数据世系的基本元素如下图所示。



Figure 任务实例之间参数依赖

多个这样的参数依赖相连，就表示一条实际执行的设计流程，世系也随之形成。而当完整的设计流程以这样的形式被表示出来后，我们就得到了一个完整的过程数据的世系以及其中的一套参数批次。

### 数据库设计



Figure 参数管理工具主要功能模块的设计

#### 参数的基本管理功能

1. **参数定义**

用户在这里定义和维护参数模型，进行参数的创建与删除，还可以修改参数名称、类型、单位并附加说明。

参数定义与绑定的顺序图如下所示。



Figure 顺序图 参数定义与绑定任务

1. **参数绑定**

如上图所示，参数实例的对象被创建后，设计人员再将其绑定到某个任务实例。

1. **约束表达式和计算公式**

约束表达式和计算公式都是绑定在任务实例上的，前者表示该任务中参数之间的关系，后者表示输出参数的计算方法。在流程执行时，设计人员可以查看这些式子，并根据它们来得到设计任务的输出结果。

#### 查看流程执行历程

前文在基于数据世系的动态过程数据管理框架中，描述了过程数据的收集，其中收集的过程数据在这里就会被用来展示流程的执行历程。

#### 一致性保障

这个功能的应用场景是，设计人员要创建一个任务实例的对象，其中要为它选择合适的参数输入，其实也就是选择上游任务实例的版本。但是，由于流程的迭代等原因，上游结点的选择一般不只一个；而当该任务实例不只一个输入参数时，其上游结点也有多个，所以可以选择的版本组合就更多。不过，这些选择中有些不具备数据一致性，需要协同设计平台淘汰掉这些不合理的组合。



Figure 活动图 一致性保障

上图中，关键的一个活动是“执行一致性检测算法”。其中算法在第4章给了详细的描述，用到了过程数据的数据世系信息。

## 参数管理工具的实现

参数管理工具的开发基于J2EE架构技术，利用了JAVA，Spring，ibatis等实现后台功能，人机交互界面的开发使用了Javascript、Flex等技术，开发的软件产品可以快速和安全的部署实施。

下面介绍几个主要管理功能的实现，并附部分人机交互界面的展示图。

### 参数、约束表达式的定义与绑定

用户在这里定义和维护参数模型，进行参数的创建与删除，还可以修改参数名称、类型、单位并附加说明。

下图所示是参数定义的一个操作界面。

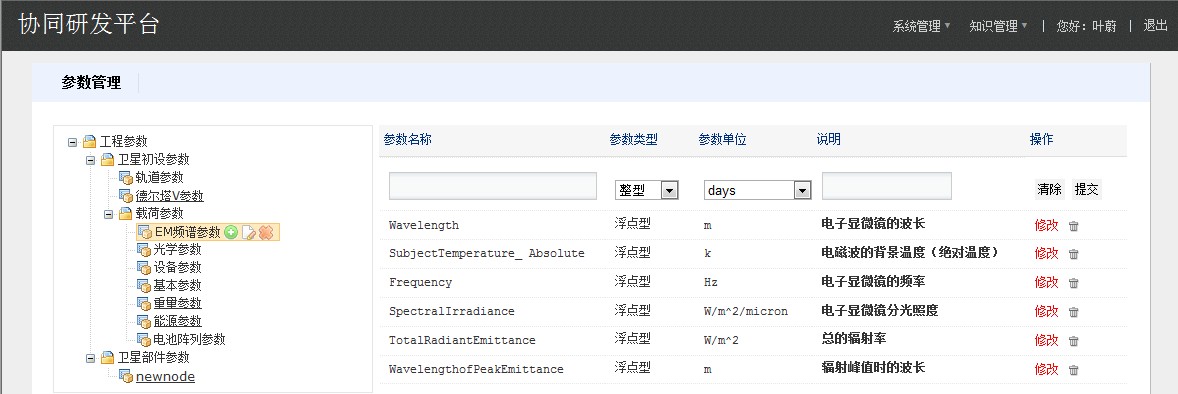


Figure 用户界面 参数定义

上图左侧是产品参数的树状结构，它将全部的参数分组、层次化的展示出来，用户可以修改这棵树。选中一个参数分组后，右边会展示出分组下的参数，再由用户进行相关操作。

### 查看流程执行历程

流程图的显示是用Flex技术实现，该技术具有良好的用户体验，是实现图形化界面的有力工具。

下图展示的是查看一个任务版本时，系统展示的该任务实例相关流程的执行历程。首先在下拉列表中选中想看的任务版本，下面图中就会显示出它的执行历史，其中红色的那个任务实例就是我们要观察的对象。然后，如果在图中选中一条连接线，可以看到两个任务实例之间传递的参数实例内容，如下图所示。

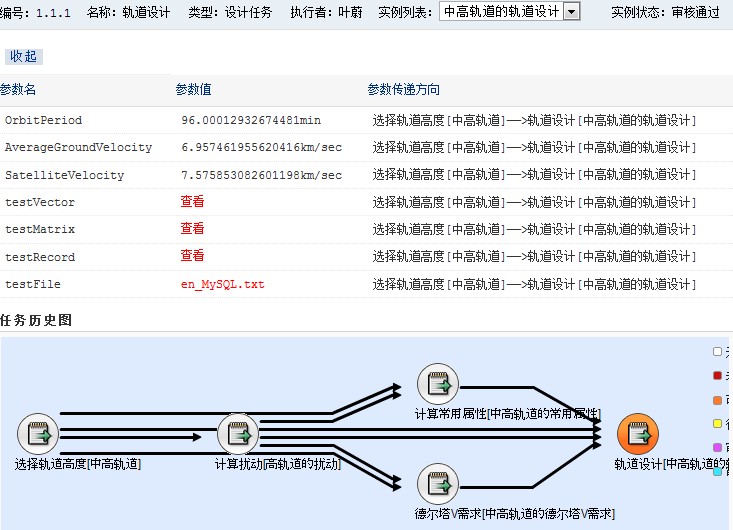


Figure 用户界面 查看任务执行历史

# 总结

## 论文工作总结

本文首先调研了当前协同设计数据管理的背景和发展现状。其中对计算机支持的协同设计进行了介绍，指出了协同设计支撑环境的主要功能需求，并且提出了当前协同设计数据管理中存在的问题，着重对数据一致性的问题进行了调查。

了解了相关背景后，本文又对与数据管理相关的研究现状进行了叙述，包括协同设计支撑环境的研究现状、工作流、数据管理、一致性研究等等。除此之外，数据世系是数据管理中可能用到的重要概念，它对协同设计有重要的意义。

然后，在目前已有的协同设计相关研究与技术的基础上，文章主要描述了三个方面的工作。

一是描述了一种基于数据世系的数据管理方案。先是定义了协同设计中的数据世系概念，它的来源、作用等。然后设计了一套收集数据世系信息机制，以及如何利用这些信息来支持协同设计工作流的管理。

二是设计了一致性保障机制。提出了几个一致性检测算法并对它们进行了比较和评价。

三是设计了一磅协同设计数据管理工具，提供一些基本的管理功能，这个工具利用了前面设计的基于数据世系的管理方案。

总体上说，数据世系在协同设计数据管理中能起到非常关键的作用，并且对后续的设计过程分析与重构等工作都有重要意义。在工程方面，参数管理工具实现了一些基本的参数管理功能，人机交互体验优良，而且实现上是基于J2EE框架，扩展性强。

## 进一步工作设想

协同设计中的数据世系包含的内容非常之多，不只是可以提供对任务执行历史和参数计算历程的追踪，还可以支持一致性检测。同样重要的，我们还可以对数据世系信息进行分析和挖掘，进一步对协同设计实施全方位的支持。

所以，我们的下一步目标，就是研究数据世系在协同设计中的其它应用，这些应用可能不如参数管理、一致性保障显示直观，但是对协同设计也是很有意义的工作。

# 参考文献

1. GrudinJ.Computer--Supported Cooperative work: history and foeus.IEEE computer, 1994,5:19-26
2. Carstensen, P.H.; Schmidt, K. (1999). Computer supported cooperative work: new challenges to systems design. Retrieved 2007-08-03.
3. Wilson, P. (1991). Computer Supported Cooperative Work: An Introduction. Kluwer Academic Pub.
4. Hubka V.Princeples of engineering design. London: Butterworth Scientific, 1980:34.
5. 陈亮，网络环境下多学科协同设计的若干关键问题研究，博士论文，2005
6. 黄双喜.计算机支持的分布式产品设计方法与理论研究.南京理工大学博士学位论文，1999.
7. 黄业清，乌兰木其，邓家堤.并行工程环境下基于黑板的多自治体协同设计.计算机集成制造系统-CIMS，1999，5(3):49一54
8. 王昕，协同设计平台所应具备的功能特征，http://club.ebusinessreview.cn/blogArticle-27770.html
9. Zhao, J., et al: Annotating, linking and browsing provenance logs for e-Science.In: 1st Workshop on Semantic Web Technologies for Searching and Retrieving Scientiﬁc Data, Sanibel Island, Florida, USA. (2003)
10. Van der Velden, “CAD to CAE Process Automation Through iSIGHT-FD”, Engineous Software., GT 2007-27555 Proceedings of the ASME Turbo Expo Montreal, 2007.
11. DENOVA, http://rainfe.com/product.php?bid=49&bname=%CA%FD%D7%D6%BB%AF%D0%AD%CD%AC%D1%D0%B7%A2%C6%BD%CC%A8DENOVA#Menu=ChildMenu2
12. 安世亚太PERA,http://www.peraglobal.com/ch/index.html
13. Ludascher, B., et al.: Scientiﬁc workﬂow management and the kepler system. Concurrency and Computation: Practice and Experience 18(10) (2005) 1039–1065
14. Taverna,http://www.taverna.org.uk/
15. 朱焕立，刘玉宾, 并行工程的产生、应用及研究现状,中州大学学报, 2003, 20(2): 122-124.
16. L. Pudhota and E. Chang, "Collaborative workflow management using service oriented approach", in Proceedings of International Conference on E-Business, Enterprise Information Systems, E-Government (EEE), Jun. 20, 2005, Las Vegas, USA, pp. 167-173.
17. Yong Shi, Geert Dick Albada, Jack Dongarra, Peter M.Sloot, Towards a formal foundation for aggregating scientic workows[A], Proceedings of the 7th international conference on Computational Science[C], Springer-Verlag Berlin,2007,216-219.
18. Y. Richard Wang, Stuart E. Madnick. A polygen model for heterogeneous database systems: the source tagging perspective// Proceedings of the 16th International Conference on Very Large Data Bases, Brisbane, Queensland, Australia, February 5-9,1990,16:519-538.
19. Tommy Ellkvist, David Koop, Erik W. Anderson, Juliana Frere, and Claudio Silva, Using Provenance to Support Real-Time Collaborative Design of Workflows
20. Lottaz C, Smith I F, Nicoud Y R, et al. Constraint based Support for Negotiation in Collaborative Design[ J] . Ar ticial Intellig ence in Engineering , 2000,14( 3) : 261-280.
21. Xue D, Xu Y. Web - based Distr ibuted System andDatabase Modeling for Concurr ent Design[ J] . Computer-aided Design, 2003, 35( 5) : 433-452.
22. 何德林, 王耕耘, 李志刚. 基于分布式约束网络的协同设计研究,计算机辅助设计与图形学学报,2002, 14( 4) : 329-332.
23. 陈亮，王伟，产品多学科协同设计中的一致性维护方法，中国机械工程,21(12)，2010,pp:1447-1452
24. 张少彤,熊光楞,支持多学科协同设计的约束网络技术研究，高技术通讯，2004,14(2)：72-75
25. 曾鹏飞，刘永贤，郝永平，邵伟平，产品多领域协同设计过程数据一致性管理，管理技术，2009，2: 102-106.
26. Pengfei Zeng, Yongping Hao, Weiping Shao, Yongxian Liu, "Data update consistency control based on P2P network in product collaborative design environment," iccsit, pp.236-238, 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009
27. 王黎维，彭智勇，黄泽谦，等．集成对象代理数据库的科学工作流服务框架中的数据跟踪[J]．计算机学报，2008，31(5)：721-732．
28. KARVOuNARAKIs G．Provenance in collaborative data sharing [D]．Pennsylvania：University of Pennsylvania，2009
29. WANG Y R，MADNICK S E．A polygene model for heterogeneous database systems：the source tagging perspective[R]． Cambridge：Sloan School of Management，1990．
30. BUNEMAN P，KHANNA S，TAN W C．On propagation of deletions and annotations thmu# views[C]，Proc of the 21st ACM SIGMOD-SIGACT·SIGART Symposium on Principles of Database Systems．New Y0rk：ACM Press。2002：150-158．
31. CUIYing wei，WIDOM J，WIENER J L．Tracing the lineage of vie, data in a data warehousing environment[J]．ACM Trans on Database Systems，2000，25(2)：179—227
32. VANSUMMEREN S．Extending where-provenance with external functions[C]，Proc of Symposium on Provenance in Databases．2008．

北京大学学位论文原创性声明和使用授权说明

**原创性声明**

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名： 日期： 年 月 日

**学位论文使用授权说明**

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

按照学校要求提交学位论文的印刷本和电子版本；

学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务；

学校可以采用影印、缩印、数字化或其它复制手段保存论文；

在不以赢利为目的的前提下，学校可以公布论文的部分或全部内容。

（保密论文在解密后遵守此规定）

论文作者签名： 导师签名：

日期： 年 月 日

# 致谢

我在论文的写作过程中，以及以往两年多的研究生学习和工作中，常常会遇到一些问题，但总是有人帮我解决、克服这些问题和困难，让我感觉到我不是一个人在奋斗。

感谢我硕士学习期间的指导老师——赵文老师，对我研究生阶段的学习和工作进行有力的指导和督促，让我保持高昂的斗志，完成一个个项目并从中积累经验。在我论文的开题以及写作过程中，对我的选题、内容进行指导，帮我确定写作的方向。

感谢工程中心的张世琨老师，以往一直对包括我在内的实验室学生都很关心和照顾，并且对我们提出严格的要求。在我论文写作中，张老师多次对我的论文进行指导，指出了很多问题，让我了解到自己的不足并加以改正。

感谢刘学洋老师和高昕、叶蔚两位师兄，最近两个月来我向他们请教了不少问题，他们也给我很认真的答复并且主动在论文内容上与我讨论并提供帮助。

感谢工程中心的胡文蕙、黄雨等老师。他们是良师，也是益友，在学术和生活上都给过我很多建议和帮助。

感谢工程中心的所有同学，他们一直在我的论文写作上提供支持和鼓励。