**本人声明**

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者：张 然

签字：

时间：2015年 6 月

**个体设计者设计决策主脉络建模方法**

学 生：张 然

指导教师：刘继红

摘 要

随着知识工程和知识管理技术的发展，以设计理性模型为代表的设计过程知识获取和应用研究成为研究热点，主要集中在设计理性模型表示、捕获、检索和重用方面，对设计理性模型的处理和挖掘相对较少。

设计理性（Design Rationale）是产品设计过程隐性知识系统化的集中体现。设计理性解释产品为什么按照某种方式进行设计，阐述产品设计背后的依据或原因，涵盖产品设计过程中所有思考、推理、权衡和决策等丰富的设计、背景信息。

设计理性知识可以在设计人员设计相似产品或者改善现有产品的时候提供许多有力的参考，比如设计思路、设计结果以及决策支持等，这样就大大缩短了新的产品的设计周期并在一定程度上节约了设计成本。

由于设计问题的复杂性，不可避免的导致设计理性模型实例较为复杂，掩盖了设计理性模型实例中设计人员的核心设计思维逻辑，降低了模型的重用效率，因此需要对设计理性模型进行适当抽象，抽取设计决策主脉络，凸显核心设计思维逻辑。

因此，本文为方便设计人员快速高效的对设计理性知识进行利用，研究已有的设计决策主脉络的提取算法，通过节点识别、关系推理、结构建议等方法从设计理性模型中提取设计决策主脉络。

本文的主要工作有如下方面：

（1）建立用于算法实验的设计理性模型实例：半盒件模具设计的设计理性知识模型，并利用设计理性工具（（DesignRationaleTool）生成了模型的XML文件。

（2）利用Java类库W3C DOM 、J2SE XML 、J2SE SAX实现对设计理性知识模型XML文件的加载和存储，提出了设计理性知识模型规范性检查的算法，研究了本实验室早期提出的设计理性知识模型主脉络提取方法。

（3）通过Java语言进行算法的程序设计，并对已构建的设计理性知识模型进行算法验证。

**关键词：**设计理性，主脉络，提取方法，Java

Abstract

窗体顶端

With the development of knowledge engineering and knowledge management technologies,it become a hot topic to design rational model as a representative of the design process and application of knowledge acquisition, mainly represented in the rational model designing, capture, retrieval and reuse aspects.Dealing with the design process rational model and research of it gained less concern.

Rational design (Design Rationale) is a concentrated expression of the tacit knowledge systematically in product design process. Design rational explains why products are designed in some way,elaborating the basis or reason behind product design, covering a wealth of product design process design, background information of thinking, reasoning, trade-offs and decision-making.

Rational design knowledge can offers many powerful references when the designer was designing similar products or improving the existing products, such as design thinking, the design results and decision support. Thus it can greatly shortening the design cycle of new products and and saving design costs.

The complexity of the design problems inevitably leads to the rational design of more complex model instance, covers the core logic design rational model of design thinking designers example and reduces the efficiency of the model reuse. So we need abstract the rational design model properly to extract the main thread design decisions and to highlight the core logic design thinking.

Therefore, for the convenience of designers to quickly design a rational and efficient use of knowledge, this research studies the main thread of the extraction algorithm of design decisions and extracts the main vein from node identification, relationship inference, structural recommendations and so on.

The main work of this paper are the following:

(1) Establishing a rational design algorithm used experimental model instance: half a box of parts design model of rational knowledge of mold design and rational and use the DesignRationaleTool generated XML file model.

(2) Using Java class libraries W3C DOM, J2SE XML, J2SE SAX to achieve rational knowledge model of design load and store XML documents. Proposing the design rational knowledge model building normative checking algorithm, studies the design proposed rational knowledge model main context extraction methods of the early laboratory .

(3) From designing Java programming language, this paper verifies the algorithm of the design of rational knowledge model which has been constructed.

**Keywords:** design rationale, main vein, extraction method, Java

目录

[**本人声明** 1](#_Toc420936232)

[摘 要 2](#_Toc420936233)

[Abstract 3](#_Toc420936234)

[1 绪 论 6](#_Toc420936235)

[1.1 研究背景及意义 6](#_Toc420936236)

[1.1.1 研究背景 6](#_Toc420936237)

[1.1.2 研究意义 7](#_Toc420936238)

[1.2 相关研究现状 7](#_Toc420936239)

[1.3 本课题研究内容 9](#_Toc420936240)

[1.4 论文组织结构 10](#_Toc420936241)

[2 总体方案设计 11](#_Toc420936242)

[2.1 概述 11](#_Toc420936245)

[2.2 需求分析 11](#_Toc420936246)

[2.2.1 设计人员需求 11](#_Toc420936247)

[2.2.2 功能要求 11](#_Toc420936248)

[2.3 总体设计 11](#_Toc420936249)

[2.3.1 算法实现验证流程 12](#_Toc420936250)

[2.3.2 算法实现方式 12](#_Toc420936251)

[2.4 小结 14](#_Toc420936252)

[3 设计理性知识建模 15](#_Toc420936253)

[3.1 概述 15](#_Toc420936254)

[3.2 可扩展标记语言—XML（eXtensible Markup Language） 15](#_Toc420936255)

[3.3 创建设计理性知识模型XML格式文件 16](#_Toc420936256)

[3.3.1 XML创建工具DRTool 16](#_Toc420936257)

[3.3.2 设计理性知识模型实例 18](#_Toc420936258)

[3.3.3 DRTool构建设计理性知识模型 20](#_Toc420936259)

[3.3.4 DRTool创建的XML文件 21](#_Toc420936260)

[3.4 小结 25](#_Toc420936261)

[4 设计理性模型规范性检查和主脉络提取算法 26](#_Toc420936262)

[4.1 概述 26](#_Toc420936263)

[4.2 XML文件中信息的获取 26](#_Toc420936264)

[4.2.1 加载XML文件 26](#_Toc420936265)

[4.2.2 储存XML文件 27](#_Toc420936266)

[4.3 提出设计理性模型规范性检查算法 28](#_Toc420936267)

[4.3.1 算法概述 28](#_Toc420936268)

[4.3.2 算法流程图 28](#_Toc420936269)

[4.4 设计理性知识模型主脉络提取算法 30](#_Toc420936270)

[4.4.1 算法概述 30](#_Toc420936271)

[4.4.2 算法流程图 30](#_Toc420936272)

[4.5 小结 31](#_Toc420936273)

[5 算法的实现与验证 32](#_Toc420936274)

[5.1 概述 32](#_Toc420936275)

[5.2 算法实现 32](#_Toc420936276)

[5.2.1 算法实现工具 32](#_Toc420936277)

[5.2.2 实现过程 32](#_Toc420936278)

[5.2.3 主要代码 33](#_Toc420936279)

[5.3 算法验证 45](#_Toc420936280)

[5.3.1 算法程序清单 45](#_Toc420936281)

[5.3.2 算法程序测试 45](#_Toc420936282)

[5.4 小结 47](#_Toc420936283)

[6 未来改进及展望 48](#_Toc420936284)

[结论 49](#_Toc420936285)

[致谢 50](#_Toc420936286)

[参考文献 52](#_Toc420936287)

1. 绪 论
   1. 研究背景及意义
      1. 研究背景

在国民经济飞速发展的今天，制造业对仍然对提升国家竞争力和综合国力有着举足轻重的影响，是重中之重。当前，我国正在从一个制造大国逐渐转型为制造强国，那么制造强国又有什么样的重要特征呢？那就是我们经常说到的“自主创新”。自主创新是先进设计的核心目标，当我们国家的各种企业不再只是国外先进企业的生产机器，而是自行设计出不同于外企的创新产品，创造出更多属于我们中华民族自己的产品，那时我们也就可以称作为制造强国或者是创造大国。

根据现代设计理论，所谓的设计是以现有知识作为基础，以新知识的获取和运用为核心的创造性活动，设计创新从其本质的角度来说等于知识创新。因此，知识管理已经成为日益激烈的全球化竞争中，企业竞争的重要手段之一。产品设计日益受到相关行业的重视，成为知识管理的重点应用领域。目前就产品设计而言，设计知识管理主要包括产品对象知识管理和设计过程知识管理。由于知识管理仍处于发展之中，伴随之产生的问题也层出不穷：

（1）企业积累了大量的知识，但大多属于静态知识（比如产品知识），零散的表层知识（如专家系统中的规则）和被固化的业务流程知识，而对于以设计理性为代表的反应设计人员设计思考过程的（包括意图、决策、选项、依据以及操作等）、更深层次的、细粒度的，可以把零散知识串联起来的动态过程知识的收集却十分匮乏，这就导致对已积累的知识或者过往的设计成果仅仅做到“知其然而不知其所以然“；

（2）知识是个人资产而不属于企业，随着企业人员的流动、离职和退休，大量存储于专家学者以及经验丰富的设计开发人员头脑中的宝贵知识会不断流失。

（3）知识得不到充分的传承、共享和重用，这是由于这些知识都是存储于人们的大脑之中，很难被系统化的使用，这就使其他的设计人员在拥有一个设计产品的时候很难去了解、获悉它的设计过程。

因此，目前设计知识管理对产品研发过程之中隐性的微观思考层面的过程知识，即设计意图、设计决策、设计选项、设计操作以及决策依据等信息的获取和管理还都非常欠缺。为了便捷有效地记录、表达、外化以及使用设计人员头脑中的设计理性知识（如设计意图、设计决策和决策依据等），来帮助设计者对已有产品及其设计思考过程有更为准确和系统全面的了解，并能通过计算机辅助来实现对知识的重用，孙兆洋等对设计理性知识建模进行了研究。其建模过程如图。

但是设计理性模型实例是一个繁琐且庞大的系统，设计人员在重用以前所记录的的设计理性知识时，在浏览、查询和使用整个设计理性的时候会遇到很大的困难。因此，研究如何在设计理性建模的基础之上，按照一定的逻辑方法对复杂的设计理性模型进行主脉络提取处理，形成更为简洁、清晰的设计理性模型，使设计人员能够更准确的获悉所需的主要的，有决定性作用的设计理性信息，并一定程度上降低模型的复杂度，减少模型的存储与搜索成本具有一定的必要性。

* + 1. 研究意义

本课题对设计理性只是模型进行适当的精简以及提取处理以便于设计理性的重用，研究意义如下：

（1）通过对复杂而庞大的设计理性知识模型进行主脉络提取可以便于设计人员对设计理性知识的重用，使设计人员能快速准确的搜索到自己所需的内容，实现设计理性的高效运用，有利于提高相关产品的设计速度以及质量，并在一定程度上节约设计成本。

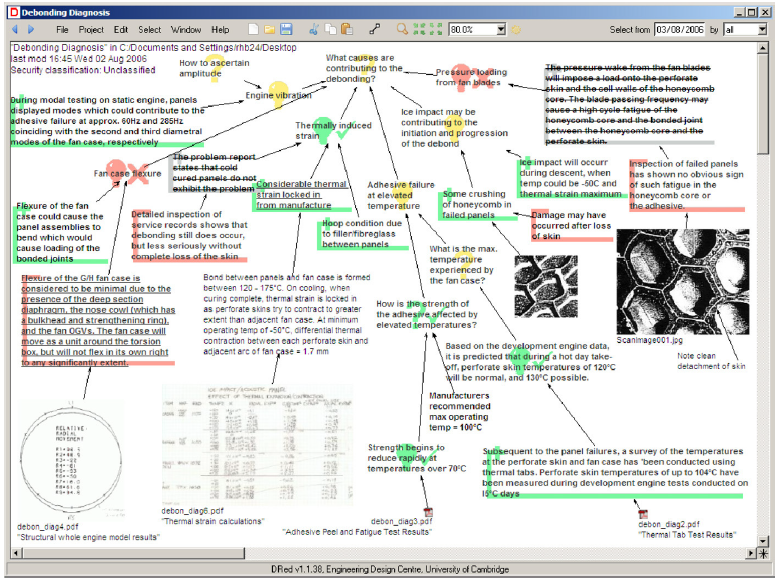
（2）结构化图形显示设计过程的逻辑推理，能够使设计人员对所提取出的设计理性知识模型主脉络进行直观的分析与理解。

（3）对复杂模型进行主脉络提取有益于实现设计理性知识的有效管理，方便设计者对已有的设计理性知识进行重用。

* 1. 相关研究现状

设计理性（Design Rationale）知识指关于设计理性有逻辑的充分依据的描述。即设计人员在设计过程中所思考的所有与设计有关的知识。设计理性解释产品为什么按照某种方式进行设计，设计目标、设计决策、设计依据、设计情境等所有设计背景信息。

始于Kunz和Rittel提出的基于问题的信息系统（Issue-based Information System, IBIS），设计理性研究建立了IBIS, QOC, PHI, DRL等著名的设计理性表达模型。这些模型主要是将设计过程看作是协商的交互作用，并采用半形式化的图形化表达形式来组织论证的结构。剑桥大学Bracewell及其研究团队开发的DRed软件正是基于IBIS的简单而有效的理性捕捉工具，允许设计人员采用结点和连接弧的形式记录设计理性，支持设计人员对已有设计决策进行访问，理解设计理性。它已经被成功应用于罗罗公司航空发动机的设计过程中，并成为该企业全生命周期管理解决方案的标准组件之一。该软件的企业推广实现了设计理性从理论研究到工业应用的突破。英国公开大学知识媒体研究所开发的Compendium系统集成各种多媒体工具用于实现个人或团体决策过程的建模，已被NASA用于多专家会话建模，对航天飞机任务运行知识进行协同建模[3] 。图1.2为Dred的显示界面[4]。



在我国相关设计知识的研究中，影响力较大是浙江大学谭建荣院士领导的研究团队，他们长期研究产品信息与知识建模与演化原理方法，提出了单元化、递归化、自组织和自适应的产品信息建模理论，取得了一系列相关研究成果。

北京航空航天大学刘继红教授对细粒度设计理性知识的建模理论与方法进行了研究，提出了设计理性概念模型CMDT （Cognitive Model of Design Thinking）、开发了设计理性建模语言DMML（Design Mental Model Language）、以及设计理性建模原型系统MindDigger。

CMDT模型包括三个子模型：语义模型、程序性模型和情节性模型，描述了设计理性模型的构成要素及其之间的相互关系。

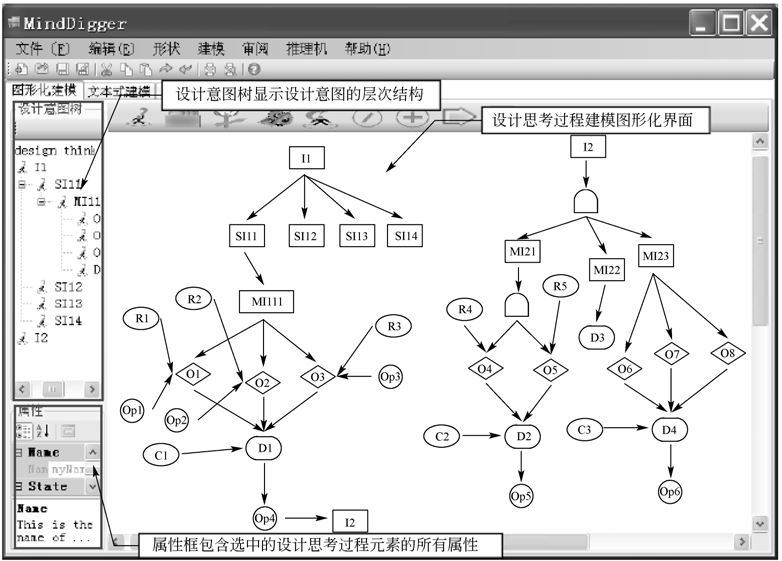
DMML是基于本体描述语言OWL的思考过程描述语言，语言能够能够支持计算机对设计理性进行理解和推理。

MindDigger原型系统主界面如图1.4所示，主要由菜单栏、工具栏、图形化建模工作区、树状结构和属性窗口等五部分组成。实现以下功能：

（1） 设计理性图形化建模；

（2） 设计理性审阅；

（3） 设计理性一致性检查和维护。



* 1. 本课题研究内容

根据设计人员的需求将现有的设计理性知识模型以结构化的图形表示出来，存储到计算机中，然后进行主脉络提取处理，实现以下功能：

（1）结构化图形显示设计理性知识模型

将设计人员在产品研发过程中的所思所想事无巨细地记录下来，按照已有的设计理性知识模型的节点、关系通过原型系统录入到计算机当中，这样就可以便于设计人员的浏览、检索以及对设计理性知识模型的主脉络提取和后续操作，使设计过程更加快速高效。

（2）提取设计理性知识模型主脉络，约简重复设计过程

一个产品的完整设计过程必定是循环往复，不停修改变动的，为了满足某些设计意图很有可能会出现许多的迭代重复设计过程，这些反复设计过程对于设计人员在回顾和理解产品设计思路的时候大多数是无用的，反而令整个设计理性模型更复杂庞大，掩盖了设计理性模型实例中设计人员的核心设计思维逻辑，降低了模型的重用效率，通过对这些反复设计过程的约简，对设计理性模型适当抽象，抽取设计决策主脉络，使得整个设计理性模型变得更加清晰易懂，凸显核心设计思维逻辑。

* 1. 论文组织结构

本文的组织结构如下：

第一章，绪论。阐述本课题的研究背景、研究意义、研究现状以及研究内容。

第二章，总方案框架设计。提出设计理性知识模型主脉络提取方案的总体设计框架。

第三章，设计理性知识建模。建立了半盒件模具设计的设计理性知识模型，基于建立的半盒件模具设计理性模型，利用本实验室研发的设计理性工具（DesignRationaleTool）建立设计理性知识模型的结构化视图以及XML文件。

第四章，提出设计理性知识模型规范性检查算法以及研究学习设计理性知识模型主脉络提取算法。

第五章，相关算法的实现与实例测试。编写相关程序，实现上一章所设计的算法流程，并对之前构建的XML格式文件进行测试。

第六章，未来改进及展望。针对算法的不足提出了改进方向。

1. 总方案框架设计
3. 1. 概述

在已经确定了研究内容的之后，根据设计人员需求分析设计理性模型主脉络提取的功能要求，提出可行的设计理性知识模型的主脉络提取方案。确定该方案的实施方法、实施工具的选择、算法流程、算法的实例验证等。

* 1. 需求分析
     1. 设计人员需求

设计理性知识模型的主脉络提取主要是为了方便设计人员在浏览、重用复杂而庞大的设计理性知识模型的时候能够从繁多的信息中快速准确的找到设计者的核心设计思维逻辑，避免对设计思路产生错误的理解。

主要需求有如下两点：

（1）对设计理性知识模型进行适当地抽象，提取设计决策主脉络，凸显设计人员的核心设计思维逻辑。

（2）将处理前后的设计理性知识模型进行对比，并能以一种直观的显示方式显现出二者的区别。

* + 1. 功能要求

为了满足服务对象的需求，功能上应该事先以下几个基本要求：

（1）能识别设计理性知识模型中的节点，如设计意图、设计选项、设计决策、设计依据、设计操作等， 以及其中的关系连线。

（2）对设计理性知识模型进行规范性检查。识别出设计者在构建设计理性知识模型阶段的不规范记录与操作，并反馈给设计人员，令其加以修改。

（3）准确的找出设计理性知识模型中冗余的迭代反复设计过程，并删除相关节点和关系连线，提取设计决策主脉络。

（4）将经过处理的设计理性知识模型保存到计算机中，并可以用设计理性工具打开，以结构化的视图显示出来。

* 1. 总体设计

本实验室研究开发的设计理性工具（DRtool）可以实现设计理性知识模型的结构化的图形显示，并能将其保存为XML格式文件，故选择将建立的设计理性知识模型通过设计理性工具录入到计算机中，并存储为XML格式文件，然后通过Java编程实现设计决策主脉络提取算法，处理设计理性知识模型的XML文件，提取设计决策主脉络。

本文的工作可以分为三个阶段，首先是建立设计理性知识模型实例并通过设计理性工具录入到计算机，接着对设计理性知识模型的规范化进行检查，并对设计人员进行提示，确保模型符合规范，然后对设计理性模型进行主脉络提取处理，并将处理后的设计理性知识模型保存至计算机，最后用设计理性工具进行结构化的显示，并作出处理前后的对比。

* + 1. 算法实现验证流程

设计理性知识模型主脉络提取算法实现与验证按照以下流程进行：

（1）构建设计理性知识模型实例，包括半盒件模具设计的设计理性知识模型以及使用设计理性工具录入并创建该模型的XML文件；

（2）提出并详细设计出对设计理性知识模型规范性进行检查的算法流程，该算法可以实现关系连线的方向检查以及连线两端节点类型的正确性检查，还可以实现一个XML文件是否只保存一个设计理性知识模型的检查。通过Java语言编程实现该算法。

（3）研究设计理性知识模型主脉络提取的算法流程，该算法可以实现识别模型中迭代反复的设计过程并将其相关的冗余节点以及关系连线删除，提取出设计决策主脉络。通过Java语言编程实现该算法。

（4）通过设计理性工具显示提取主脉络后的设计理性知识模型的结构化图形。

（5）用已经编写完成的程序处理已构建的设计理性知识模型实例，并对提取前后的模型进行对比。如图

* + 1. 算法实现方式

该算法的实现主要通过Java语言编程来进行。

1995年5月，美国Sun Microsystems公司推出了Java程序设计语言（简称Java语言）和Java平台，并总称为Java语言。Java作为目前最为流行的面向对象编程语言之一，他的名字被所有IT认识所熟知。

Java不单单只是一门编程语言而已，它还是一个开放性的平台。Sun公司已经成功推出了Java EE、Java SE、Java ME三大应用平台，从企业应用、高性能计算，到移动通信，其无处不在，体现了Java强大的应用功能

（1）Java SE

Java SE（Java标准版）。Java SE 以前称为J2SE，它是各式各样应用平台的基础，主要的应用方向是桌面开发和低端商务应用的解决方案。它为Java虚拟机和运行Java应用程序提供了必需的类库。

（2）Java EE

Java EE（Java企业版）。Java EE以前称为J2EE。它致力于基于企业环境的服务器端应用程序的开发。Java EE 是一种利用Java 2平台来简化企业解决方案的开发、部署和管理相关复杂问题的体系结构，他的基础就是Java SE，Java EE不仅巩固了标准版中的许多优点，同时还提供了对Servlet API、JSP、EJB、XML技术、Web服务、SOA、Web 2.0的全面支持。它本身是一个开放的标准，这使得第三方厂商也可以推出符合Java EE标准的产品。

（3）Java ME

Java ME（Java微型版）。Java ME以前称为J2ME，又叫K-JAVA。它是一种经过优化的Java环境，Java ME使Java程序能够在手机、机顶盒、PDA等消费类电子产品上运行。它包含灵活的用户使用界面、优秀的安全模型、大量的内置网络协议。以Java ME为规范的应用程序仅仅需要编写一次就可以实现在许多设备上应用，并且能够利用每个设备的本机功能。

Java语言的广泛应用与它的密切相关，下面列出Java语言具有的一些特点：跨平台、面向对象、使用简单、强大的网络功能、解释性、高性能、健壮性、分布式计算、安全性、多线程和动态性。

Java与C++有密切的联系，Java建立在C和C++丰富的基础之上。Java的语法是从C继承来的，它的面向对象的特性改编自C++。但是，Java相对于C++的一些改进是，去掉了C++中一些相对复杂的机制，如多重继承。这样，使的Java语言是很简单的。编程简单，具有安全性和可移植性，这是Java最主要的特点。与C++相比，Java的执行速度要慢，因为它是解释性的。所以，如果要完成一些实时性的系统，首先考虑选择C++语言。

适用于Internet是Java预案一年的的天然属性，加上与平台无关的优点，使得Java在全球云计算和移动互联网的产业环境下，更具备了显著优势和广阔前景。鉴于设计理性知识未来的应用是面向网络的，所以选择Java作为算法的实现工具。

* 1. 小结

本章完成了对设计人员需求的分析，根据需求提出了程序算法所应实现的功能，同时也提出了程序算法的实现以及验证方案，包括用设计理性工具构建录入半盒件模具设计的设计理性知识模型，提出了基于Java的算法实现方案，还有对程序算法的验证方案。

1. 设计理性知识建模
   1. 概述

在确定系统总体设计之后，本章主要建立设计理性知识模型。由于本文的建模是基于本实验室已有的设计理性工具（DRTool）进行，而设计理性工具可以很方便快捷的将设计理性模型实例保存为XML文件，所以选择用XML来描述半盒件模具设计的设计理性知识模型实例，为设计理性知识模型主脉络提取算法的验证做准备。

* 1. 可扩展标记语言—XML（eXtensible Markup Language）

可扩展标记语言（XML），是一种描述信息的标记语言，它非常简单，与使用它的应用程序无关；而其他的标记语言需要特定的应用程序才可以使用。

HTML，VioceXML，Word和Excel都是使用标记语言所需的特定应用程序的文件格式或类型。需要特定的应用程序的原因之一是这些标记语言的结构已经定义好并且不能改变。尽管HTML和VioceXML的文件结构比Word和Excel的灵活性要高，但是仍然不能添加自定义的元素类型；事实上，类似Word这样的文件类型，不能改变定义文档如何布局的结构。

另一方面，XML允许用户创建自己的结构，可以定义自己的标记和属性，从而以一种更有意义的方式描述信息。按照任何希望的方式构造数据的能力，这是XML有别于其他语言的最重要的优势之一。

XML文档是由字符数据组成的。标记能够把字符数据和其他的文件中的字符数据区别划分开来，它使用标记来提供一种表示XML文档结构的机制。如果没有标记，XML文档就和一堆杂乱无序的数字和字母一样。

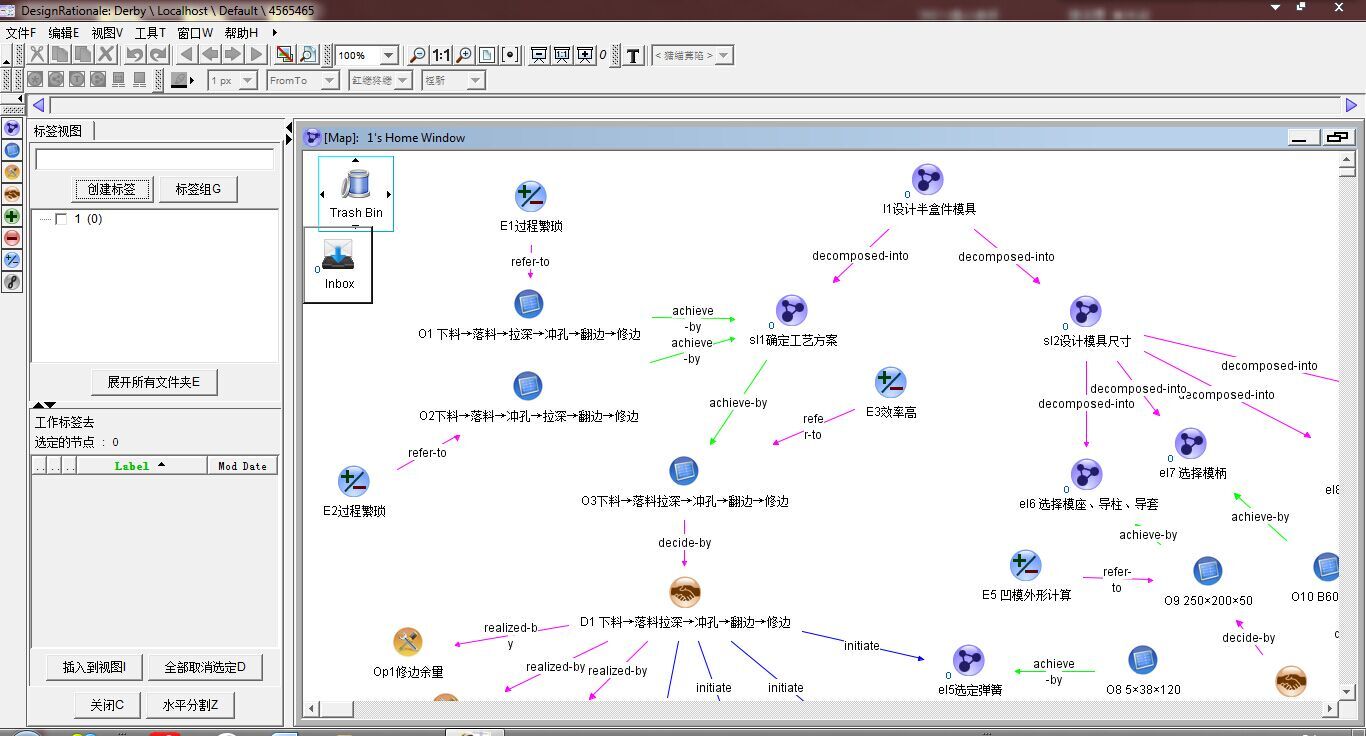
XML规范定义了不同类型的标记，这些标记分别有不同的用途。以下列出各种标记及其用途的简要说明，如表所示

|  |  |
| --- | --- |
| 标记类型 | 说明 |
| 处理指令 | 在解析阶段向应用程序提供信息 |
| 注释 | 提供文档编写或注释的文本块 |
| 实体引用 | 引用类似字符数据或XML的实体；习惯于插入被认为是标记的字符数据 |
| 字符数据段 | 不想当做标记或者被解析的字符数据 |
| 文档类型声明 | 表示一个DTD（文档类型定义），其中包含对XML文档的约束 |
| 起始和结束标记< >和</> | 定义元素的开始和结束 |
| 空元素标记</> | 定义没有主体或内容的元素 |

现在，XML几乎用于每一个可能的行业中，比如医疗、汽车制造以及多数应用（比如信息交换、数据库管理和内容管理）中。另外，XML还引起一些创建、编辑和转换它的新的应用产生。

* 1. 创建设计理性知识模型XML格式文件
     1. XML创建工具DRTool

本实验室作为国内设计理性方面研究的先驱者与领导者，针对设计理性模型的变粒度特征，设计并开发了设计理性变粒度建模原型系统DesignRationaleTool。



此系统为设计人员提供可视化的结构化的设计理性知识的建模和管理工具，分为用户表示层、应用功能层、基础服务层和数据资源层四层结构。在应用功能层之中主要包括设计理性知识建模和设计理性评价两个核心功能模块。在本次设计的过程当中，主要使用了可视化设计理性建模和设计理性模型存储两个功能，接下来对这两个功能作详细介绍：

1.可视化设计理性建模

此项功能是DRTool为设计人员提供的核心功能，该功能通过DRTool提供的图示化建模工具来实现。设计者可以使用图示化建模方法来记录设计过程中的所思所想所做，构建设计理性知识模型。图示化建模功能包括：

1. 设计理性要素节点管理。管理设计理性模型中的各种组成要素，包括如下几种：设计意图、设计选项、设计决策、设计依据和设计操作。
2. 设计理性节点关系管理。对设计理性节点之间的关系的组织以及管理，节点关系由其连接的设计理性模型的节点类型决定，包括decomposed-into、achieved-by、decided-by、refer-to、realized-by、initiate、return-to和evolve-to。

2.设计理性模型存储

DRTool提供基于数据库和基于特殊的XML—OWL的结构化存储方式。

数据库储存的设计理性知识模型图形化实例主要由节点（Node）、节点连线（Link）、视图（View）和标签（Code）构成。设计人员可以借助图形化建模功能完成设计理性建模，然后将其存储到知识管理服务器进行统一的知识管理。在其他设计者拥有该设计理性模型的设计权限时，可利用数据的统一管理实现协同设计或者后续设计。

设计理性模型借助OWL语言实现数据的结构化存储的关键在于OWL以类和属性定义设计理性模型中的理性要素及其之间的关系，允许用户指定有效元素，在元素中定义有效的结构数据；每个元素包含有效的数据类型和用于元素识别的操作。基于OWL语言的结构化存储不但能够有效支持计算机对设计理性模型的逻辑推理，还支持设计者直观浏览，实现设计交流和共享。知识管理系统可以通过OWL文件实现对设计理性模型的统一管理。

* + 1. 设计理性知识模型实例

为了深入了解设计理性建模过程以及为实例验证做准备，本小节在充分理解设计理性知识模型及其表达的基础上，构建单个设计理性知识模型实例。在这里选择半盒件模具设计的设计理性知识模型为例。

图所示为设计理性知识模型的结构描述方法。



图3.2设计理性知识模型

图3.2中的关系定义为：

分解关系：Decomposed-into（意图1，意图2）：表示意图2是意图1的子意图；

满足关系：Achieved-by（意图，选项）：表示该选项可以满足设计意图；

决策关系：Decided-by（选项，决策）：表示该选项是该决策的最终选择；

依据关系：Refers-to（决策，标准/权衡）：表示决策与其参考的评价标准、设计人员的权衡、偏好之间的关系；

实现关系：Realized-by（决策，操作）：表示该决策由该操作执行、实现；

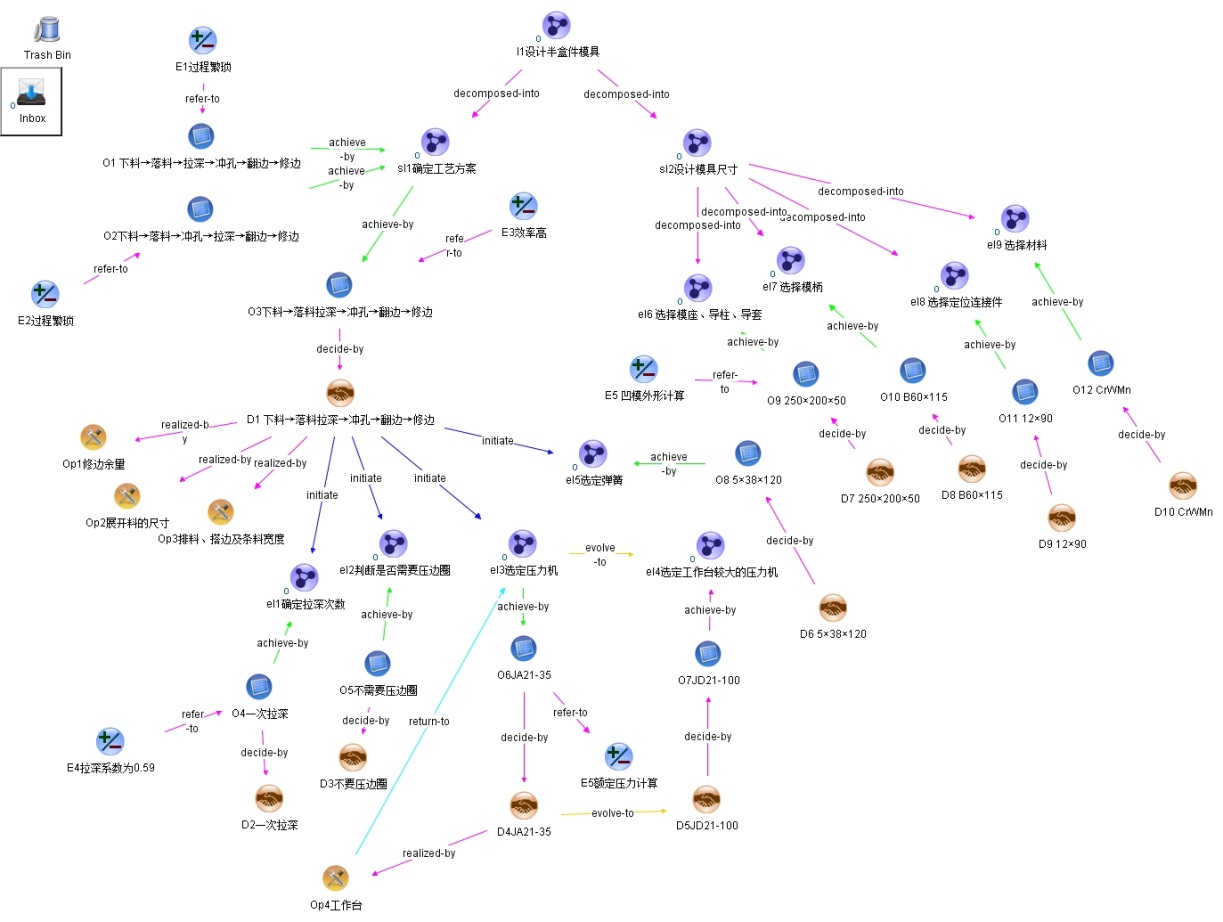
触发关系：Initiates（操作，意图）：表示该操作可以触发一个新的设计意图；

演进关系：Evolve-to（，）：表示设计意图、设计选项以及设计决策的迭代与演进关系，共包含三个种类：evolve-to（意图，意图）、evolve-to（选项，选项）以及 evolve-to（决策，决策）。

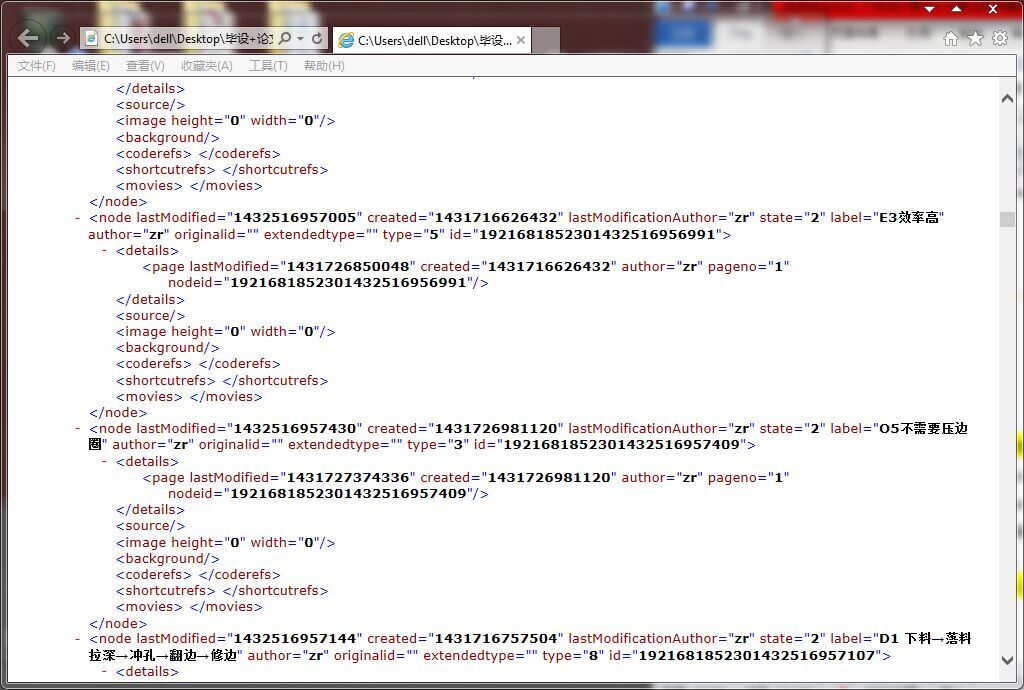
返回关系：Return-to（意图/选项/决策/操作，意图/选项/决策）表示设计迭代，设计过程中新问题的出现、新知识的获取以及设计情境的变化，需要设计者对以前的设计意图、设计选项及设计决策重新验证、评价及更改.

* + 1. DRTool构建设计理性知识模型

在半盒件模具设计的设计理性知识模型的基础上建立XML文件。模型实例截取了设计半盒件模具过程的一部分以用于算法测试的实例验证。整个实例模型的建立完全按照图3.2所给出的结构，其中包括设计意图（Design Intent）、设计选项（Choices Option）、设计决策(Decision)、设计依据(Justification)以及设计操作(Operation)等节点；以及Decomposed-into、Achieved-by、Decided-by、Return-to、Refers-to、Realized-by、Initiates、Evolve-to的关系属性。图3.3给出了已构建的半盒件模具设计的设计理性知识模型的结构图。



下图显示了半盒件模具设计的设计理性知识模型的XML文件。



* + 1. DRTool创建的XML文件

此处给出通过DRTool创建的设计理性知识模型实例XML文件中的部分描述语言：

<?xml version="1.0" encoding="UTF-16"?>

<!DOCTYPE model SYSTEM "DesignRationale.dtd">

<model rootview="1270011432466003528">

<views>

...

</views>

<nodes>

<node id="1921681852301432516957376" type="2" extendedtype="" originalid="" author="zr" created="1431727112192" lastModified="1432516957401" label="eI3 选定压力机" state="2" lastModificationAuthor="zr">

<details>

<page nodeid="1921681852301432516957376" author="zr" created="1431727112192" lastModified="1431727112192" pageno="1">Home Window of abc</page>

</details>

<coderefs>

</coderefs>

<shortcutrefs>

</shortcutrefs>

<movies>

</movies>

</node>

...

</nodes>

<links>

<link id="1921681852301432516957869" created="1432516952064" lastModified="1432516952064" author="1" type="46" originalid="" from="1921681852301432516957376" to="1921681852301432516957376" label="return-to">

<linkviews>

<linkview id="1270011432466003528" created="1432516952064" lastModified="1432516952064" arrowtype="1" linkstyle="0" linkdashed="0" linkweight="1" linkcolour="-65281" labelWrapWidth="25" fontsize="12" fontface="Dialog" fontstyle="0" foreground="-16777216" background="-1" />

</linkviews>

</link>

<link id="1921681852301432516957696" created="1432516952064" lastModified="1432483659776" author="1" type="43" originalid="" from="1921681852301432516957376" to="1921681852301432516956723" label="evolve-to">

<linkviews>

<linkview id="1921681852301432516957696" created="1432516952064" lastModified="1432516952064" arrowtype="2" linkstyle="0" linkdashed="0" linkweight="1" linkcolour="-16776961" labelWrapWidth="25" fontsize="12" fontface="Dialog" fontstyle="0" foreground="-16777216" background="-1" />

</linkviews>

</link>

...

</links>

<codes>

</codes>

1. XML标签view代表一个视图，views代表视图集合。
2. XML标签node代表一个节点，nodes代表节点集合。

node的id属性代表节点的唯一标识；type属性代表节点的类型，type取值及其代表含义：

1. Inbox
2. 设计意图
3. 设计选项
4. 设计操作
5. 设计依据
6. 设计决策；

author属性代表该节点的作者；created属性代表该节点的创建时间；lastModified属性代表该节点的上次修改时间；label属性代表该节点的文字说明；lastModificationAuthor属性代表该节点上次修改的作者。

1. XML标签link代表一条关系连线，links代表关系连线集合。

link的id属性代表关系连线的唯一标识；created属性代表该关系连线的创建时间；lastModified属性代表该关系连线上次修改时间；author属性代表该关系连线的作者；type属性代表该关系连线的类型；from属性代表该关系连线从哪个节点引出，其取值为引出节点的id；to属性代表该关系连线指向哪个节点，其取值为所指节点的id；label属性代表该关系连线的文字说明，为方便程序编写，本文规定通过lable属性来识别关系连线类型，其可选类型有：

decomposed-into（意图，意图）

achieve-by（意图，选项）

decided-by（选项，决策）

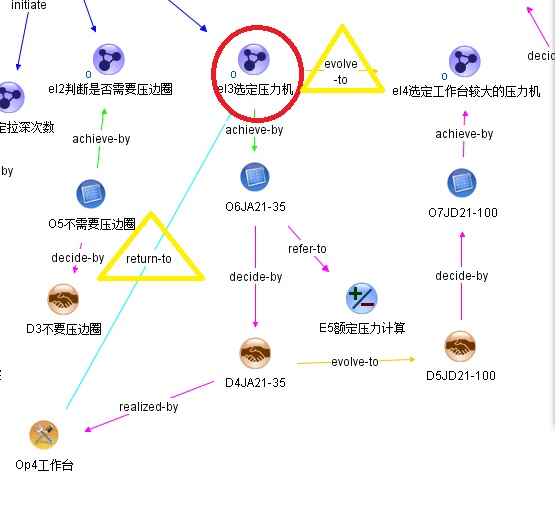
refer-to（选项，依据）（决策，标准）（决策，权衡）

realized-by（决策，操作）

initiate（决策／操作，意图）

return-to（意图／选项／决策／操作，意图／选项／决策）

evolve-to（意图，意图）（选项，选项）（决策，决策）



如图所示，节点“eI3选定压力机”通过该节点的属性label="eI3 选定压力机"来确定对应关系。return-to关系连线和evolve-to关系连线同样通过该关系连线的属性label="return-to"和label="evolve-to"来确定对应关系，而关系连线的方向与引出、所指节点则由from、to属性来确定对应关系，如return-to关系连线指向“eI3选定压力机”，则它的to属性的值to="1921681852301432516957376"等于节点“eI3选定压力机”的id取值。

* 1. 小结

本章的重点是设计理性，介绍了设计理性知识模型的构建，简要介绍了可扩展标记语言-XML的相关知识和优点。然后介绍了本文建立设计理性知识模型所使用的系统DRTool，并着重对所使用的建模和存储功能作出详细介绍。然后介绍了建模的过程，并对最后创建出的XML文件进行解析。

1. 设计理性模型规范性检查和主脉络提取算法
   1. 概述

在建立了半盒件模具设计的设计理性知识模型XML文件之后，本章主要进行XML文件处理方法的探究和设计理性知识模型规范性检查的算法设计以及设计理性知识模型主脉络提取算法的研究。为完成主脉络提取工作奠定坚实的理论基础。

* 1. XML文件中信息的获取
     1. 加载XML文件

1. W3C DOM库：它的功能是为文档对象模型 (DOM) 提供接口，该模型是 Java API for XML Processing 的组件API。

（1）Document 接口：

它的功能是表示整个 HTML 或 XML 文档。以概念上的定义来说，它就是文档树的根部，并且提供对文档数据的基本访问功能。

（2）Element 接口：

它的功能是表示XML 文档中的某一个元素。Element 接口上含有通过名称获得 Attr 对象或属性值的相关方法。

（3）NodeList 接口：

它的功能是提供对节点的有序集合的抽象，但是并没有定义或约束如何来实现这个集合。

2. J2SE XML： Java标准XML处理库

Documentbuilder

3. J2SE SAX： Java标准核心SAX API库

（1）EntityResolver：用于解析实体的基本接口。

（2）InputSource：XML 实体的单一输入源。

* + 1. 储存XML文件

本设计也使用了W3C DOM库存储XML文件。

1.LSOutput：此接口表示数据的输出目标。

2.LSSerializer：此接口提供了将 DOM 文档进行序列化为 XML 的 API。XML 数据将被写入字符串或者输入流。

* 1. 提出设计理性模型规范性检查算法
     1. 算法概述

根据毕业设计过程中出现的实际情况，在设计人员使用设计理性工具进行建模的过程中，设计理性工具对模型构建的规范性并没有做出规定和约束，任意节点之间可以使用任意的关系连线进行连接，这就导致设计人员在建立设计理性知识模型的过程中有可能出现规范性方面的错误，使得程序在进行设计理性模型的主脉络提取过程出现错误，所以我们需要先对加载完毕的XML文件进行规范性检查，确认该设计理性知识模型没有错误之后再对该模型的XML文件进行主脉络提取处理。本文设计出检查设计理性知识模型规范性的算法流程，该算法可以实现关系连线的方向检查以及连线两端节点类型的正确性检查，还可以实现一个XML文件是否只保存一个设计理性知识模型的检查。

* + 1. 算法流程图

1．关系连线的方向检查以及连线两端节点类型的正确性检查。



如图所示，只给出两个关系连线类型的方向检查以及连线两端节点类型的正确性检查算法，其他关系连线类型同样按照如上算法进行类比，即可完成规范性检查。

2. XML文件是否只保存一个设计理性知识模型的检查。



通过对设计理性知识模型的XML文件中ROOT节点个数的检查来确定该文件中是否只存在一个设计理性知识模型，完成模型规范性检查。

* 1. 设计理性知识模型主脉络提取算法
     1. 算法概述

设计理性模型主脉络抽取主要通过对设计理性模型自上向下的推理完成。在设计理性模型主脉络的抽取过程中，对由关系、关系表达的设计过程的反复、设计理性元素的更改以及演进过程进行归一、合并，提取设计过程中意图、选项、及决策在其演进过程中的最新状态及交互关系，实现设计活动主体发展脉络的宏观呈现。

* + 1. 算法流程图



* 1. 小结

本章主要确定了算法的实施方式，对设计理性知识模型的XML文件用三个Java类库进行加载和保存处理。最重要的是完成了设计理性知识模型的规范性检查的算法并研究了已经被提出了的设计理性知识模型主脉络提取的算法。满足了本文前边所列的研究内容。

1. 算法的实现与验证
   1. 概述

在确定设计理性知识模型规范性算法和主脉络提取算法的主要功能结构及算法后，对其加以实现。为了验证整个算法的可行性，对之前创建的半盒件模具设计的设计理性知识模型进行测试。

* 1. 算法实现
     1. 算法实现工具

在设计理性知识模型主脉络提取工具的开发过程中，开发工具涉及到程序语言、开发平台、开发环境以及其他资源工具。

1. 算法实现平台：Windows 7 旗舰版（x64）
2. 算法实现语言：Java

设计人员对检索的基本要求是系统无关性，这使得开发者应尽可能选择成熟的、健壮的语言。而Java作为一种面向网络应用并且独立于底层操作系统的语言，其应用领域、应用成熟性已经成为广大的程序开发者所承认，因而被本设计系统采用。

1. 算法实现环境：Eclipse4.4.2+JDK1.8

系统模块设计的基本原则是开放、可扩展和可移植，这样有利于后期的维护和升级等。而Eclipse4.4.2就是一个开源的、基于Java的可扩展平台，满足系统设计基本原则。Eclipse4.4.2开源的代码十分多，具有相当的优势，有利于代码的重用。

1. 文件操作工具：

（1）XML文件创建工具：DRTool

（2）XML文件编辑工具：Notepad++6.7.8

1. 其他开发工具

Java类库W3C DOM 、J2SE XML 、J2SE SAX。

* + 1. 实现过程

首先通过DRTool来创建用来测试的模型XML文件，DRTool可以使设计人员快速简洁地构建XML文件而无需学习XML的语法规则。

然后Java类库W3C DOM 、J2SE XML 、J2SE SAX加载XML文件，此步骤可以快速进行，无需去解析XML文件格式。

由于Java最大的特点就是面向对象编程，所以在算法编写过程中要注意程序的封装。本文所列的程序首先构建了一个zrdocument类，这个类代表一个模型，在zrdocument这个类中封装了模型XML文件的加载和保存方法，和模型XML文件规范性检查的算法，以及模型XML文件全局处理方法，例如removeSubtree，目的是删除一个子树，removeSubtreeOfNode，是保留当前节点，删除这个节点以下的所有子节点和孙节点，replaceNode，是把当前节点替换成新节点。getRootNode，获取根节点。getNodeMap，获取node id和node对象的映射关系。

又建立了zrnode和zrlink类。这两个类分别代表一个节点和一个关系连线。又建立了zrprocessor类，封装主脉络提取算法。

关于测试问题，并不是说等所有的程序编好后才进行程序的测试，这样的话，就会很难排除程序中的问题，有时甚至说很小的一个问题都会浪费很多的时间和精力。所以程序的测试应该是一边编写一边测试。也就是进行所谓的单元测试，单元测试是指在计算机编程中，针对程序模块（软件设计的最小单位）来进行正确性检验的测试工作。程序单元是应用的最小可测试部件。程序员每次对程序进行修改就需要进行最少一次单元测试，在编写程序的过程中一般都要进行多次单元测试，以证明程序达到设计人员所要达到目标，实现预期功能，没有程序错误。即使这样，由于某些疏忽或者对某些概念的理解不清楚都造成了很多问题。

* + 1. 主要代码

主要代码（部分）如下列出：

1. zrDocument.java

该类是的功能是用来处理XML文件的。

XML文件加载代码：

**public** **void** load(String filename) {

DocumentBuilderFactory dbf = DocumentBuilderFactory.*newInstance*();

DocumentBuilder db = dbf.newDocumentBuilder();

db.setEntityResolver(**new** EntityResolver() {

**public** InputSource resolveEntity(String publicId, String systemId) {

**if** (systemId.contains("DesignRationale.dtd")) {

**return** **new** InputSource(**new** StringReader(""));

}

**else** {

**return** **null**;

}

}

});

doc = db.parse(**new** File(filename));

nodeMap.clear();

rootNode = **null**;

}

XML文件保存代码：

**public** **void** save(String newfilename) {

FileOutputStream fos = **new** FileOutputStream(**new** File(newfilename));

**try** {

DOMImplementationRegistry reg = DOMImplementationRegistry.*newInstance*();

DOMImplementationLS impl = (DOMImplementationLS) reg.getDOMImplementation("LS");

LSSerializer serializer = impl.createLSSerializer();

LSOutput lso = impl.createLSOutput();

lso.setByteStream(fos);

serializer.write(doc, lso);

} **finally** {

fos.close();

}

}

XML文件规范性检查、深度解析代码：

**public** **void** advParse() **throws** Exception {

NodeList nodeList;

// process all the <node>

nodeList = doc.getElementsByTagName("node");

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

Element n = (Element)nodeList.item(i);

ZrNode zn = **new** ZrNode();

**switch**(Integer.*parseInt*(n.getAttribute("type"))) {

**case** ZrNode.*TYPE\_INBOX*:

zn.nodeType = ZrNode.*TYPE\_INBOX*;

**break**;

**case** ZrNode.*TYPE\_INTENT*:

zn.nodeType = ZrNode.*TYPE\_INTENT*;

**break**;

**case** ZrNode.*TYPE\_OPTION*:

zn.nodeType = ZrNode.*TYPE\_OPTION*;

**break**;

**case** ZrNode.*TYPE\_OPERATION*:

zn.nodeType = ZrNode.*TYPE\_OPERATION*;

**break**;

**case** ZrNode.*TYPE\_JUSTIFICATION*:

zn.nodeType = ZrNode.*TYPE\_JUSTIFICATION*;

**break**;

**case** ZrNode.*TYPE\_DECISION*:

zn.nodeType = ZrNode.*TYPE\_DECISION*;

**break**;

**default**:

**throw** **new** Exception("Node " + n.getAttribute("id") + " has invalid type");

}

zn.domNode = n;

nodeMap.put(n.getAttribute("id"), zn);

}

// process all the <link>

nodeList = doc.getElementsByTagName("link");

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

Element n = (Element)nodeList.item(i);

ZrLink zl = **new** ZrLink();

**switch**(n.getAttribute("label")) {

**case** "decomposed-into":

zl.linkType = ZrLink.*TYPE\_DECOMPOSED\_INTO*;

**break**;

**case** "achieve-by":

zl.linkType = ZrLink.*TYPE\_ACHIEVE\_BY*;

**break**;

**case** "decide-by":

zl.linkType = ZrLink.*TYPE\_DECIDE\_BY*;

**break**;

**case** "refer-to":

zl.linkType = ZrLink.*TYPE\_REFER\_TO*;

**break**;

**case** "realized-by":

zl.linkType = ZrLink.*TYPE\_REALIZED\_BY*;

**break**;

**case** "initiate":

zl.linkType = ZrLink.*TYPE\_INITIATE*;

**break**;

**case** "return-to":

zl.linkType = ZrLink.*TYPE\_RETURN\_TO*;

**break**;

**case** "evolve-to":

zl.linkType = ZrLink.*TYPE\_EVOLVE\_TO*;

**break**;

**default**:

**throw** **new** Exception("Link " + n.getAttribute("id") + " has invalid type");

}

zl.domNode = n;

zl.to = nodeMap.get(n.getAttribute("from"));

ZrNode zn = nodeMap.get(n.getAttribute("to"));

zn.linkList.add(zl);

}

// get root node

Object[] keys = nodeMap.keySet().toArray();

**for** (**int** i = 0; i < keys.length; i++) {

ZrNode zn = nodeMap.get((String)keys[i]);

// only INTENT can be root node

**if** (zn.nodeType != ZrNode.*TYPE\_INTENT*) {

**continue**;

}

// there's an special node with label "Home Window", which should be excluded

**if** (zn.domNode.getAttribute("label").equals("Home Window")) {

**continue**;

}

// root node should have no link points to it

**boolean** found = **false**;

**for** (**int** j = 0; j < keys.length; j++) {

ZrNode zn2 = nodeMap.get((String)keys[j]);

**for** (**int** k = 0; k < zn2.linkList.size(); k++) {

**if** (zn2.linkList.get(k).to == zn) {

found = **true**;

**break**;

}

}

**if** (found) {

**break**;

}

}

**if** (found) {

**continue**;

}

rootNode = zn;

**break**;

}

**if** (rootNode == **null**) {

**throw** **new** Exception("can not find root node");

}

}

模型节点操作代码（4个操作）：

**public** **void** removeNodeWithSubTree(ZrNode node) {

Object[] keys;

// remove links \_to\_ this node

keys = nodeMap.keySet().toArray();

**for** (**int** i = 0; i < keys.length; i++) {

ZrNode zn = nodeMap.get((String)keys[i]);

**for** (**int** j = 0; j < zn.linkList.size(); j++) {

ZrLink zl = zn.linkList.get(j);

**if** (zl.to == node) {

zl.domNode.getParentNode().removeChild(zl.domNode);

zn.linkList.remove(j);

j--;

}

}

}

// remove links \_from\_ this node

List<ZrNode> toList = **new** ArrayList<ZrNode>();

**for** (**int** i = 0; i < node.linkList.size(); i++) {

ZrLink zl = node.linkList.get(i);

**if** (zl.linkType != ZrLink.*TYPE\_RETURN\_TO* && zl.linkType != ZrLink.*TYPE\_EVOLVE\_TO*) {

toList.add(zl.to);

}

zl.domNode.getParentNode().removeChild(zl.domNode);

}

node.linkList.clear();

// recurse into the subtree

**for** (**int** i = 0; i < toList.size(); i++) {

removeNodeWithSubTree(toList.get(i));

}

// remove this node

node.domNode.getParentNode().removeChild(node.domNode);

keys = nodeMap.keySet().toArray();

**for** (**int** i = 0; i < keys.length; i++) {

ZrNode zn = nodeMap.get((String)keys[i]);

**if** (zn == node) {

nodeMap.remove((String)keys[i]);

**break**;

}

}

}

**public** **void** removeSubtreeOfNode(ZrNode node) {

List<ZrNode> toList = **new** ArrayList<ZrNode>();

**for** (**int** i = 0; i < node.linkList.size(); i++) {

ZrLink zl = node.linkList.get(i);

**if** (zl.linkType == ZrLink.*TYPE\_REFER\_TO*) {

**continue**;

}

toList.add(zl.to);

zl.domNode.getParentNode().removeChild(zl.domNode);

node.linkList.remove(i);

i--;

}

// recurse into the subtree

**for** (**int** i = 0; i < toList.size(); i++) {

removeNodeWithSubTree(toList.get(i));

}

}

**public** **void** replaceNode(ZrNode oldNode, ZrNode newNode) {

Object[] keys;

// change all the links \_to\_ this node

keys = nodeMap.keySet().toArray();

**for** (**int** i = 0; i < keys.length; i++) {

ZrNode zn = nodeMap.get((String)keys[i]);

**for** (**int** j = 0; j < zn.linkList.size(); j++) {

ZrLink zl = zn.linkList.get(j);

**if** (zl.to == oldNode) {

zl.domNode.setAttribute("from", newNode.domNode.getAttribute("id"));

zl.to = newNode;

}

}

}

// remove oldNode

removeNodeWithSubTree(oldNode);

}

**public** **void** changeParent(ZrNode node, ZrNode oldParent, ZrNode newParent) {

**for** (**int** i = 0; i < oldParent.linkList.size(); i++) {

ZrLink zl = oldParent.linkList.get(i);

**if** (zl.to == node) {

zl.domNode.setAttribute("to", newParent.domNode.getAttribute("id"));

oldParent.linkList.remove(i);

newParent.linkList.add(zl);

}

}

}

1. zrProcessor.java

此类中封装了主脉络提取算法，主脉络提取算法实现时采用了递归函数调用。

**private** **void** processImpl(ZrNode node) {

ZrNode evolveTo = **null**;

**for** (**int** i = 0; i < node.linkList.size(); i++) {

ZrLink zl = node.linkList.get(i);

**if** (zl.linkType == ZrLink.*TYPE\_EVOLVE\_TO*) {

evolveTo = zl.to;

**break**;

}

}

**if** (evolveTo != **null**) {

// this node has an evolve-to, do evolve-to infering

**for** (**int** i = 0; i < node.linkList.size(); i++) {

ZrLink zl = node.linkList.get(i);

**if** (zl.linkType == ZrLink.*TYPE\_ACHIEVE\_BY*) {

docObj.removeSubtreeOfNode(zl.to);

docObj.changeParent(zl.to, node, evolveTo);

}

}

docObj.replaceNode(node, evolveTo);

processImpl(evolveTo);

**return**;

} **else** {

// this node does not have an evolve-to, do recursing

List<ZrNode> toList = **new** ArrayList<ZrNode>();

**for** (**int** i = 0; i < node.linkList.size(); i++) {

ZrLink zl = node.linkList.get(i);

**if** (zl.linkType == ZrLink.*TYPE\_REFER\_TO*) {

**continue**;

}

**if** (zl.linkType == ZrLink.*TYPE\_RETURN\_TO*) {

**continue**;

}

toList.add(zl.to);

}

**for** (**int** i = 0; i < toList.size(); i++) {

processImpl(toList.get(i));

}

}

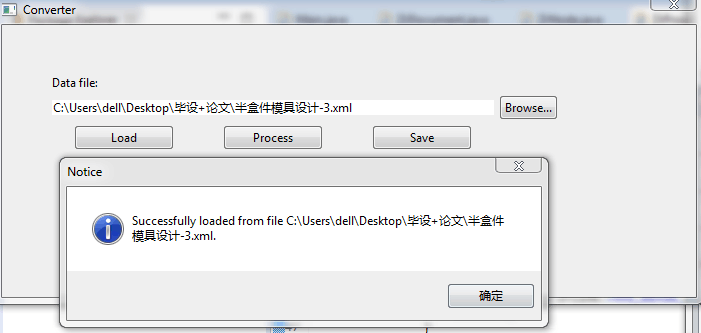
}

* 1. 算法验证
     1. 算法程序清单

本算法程序主要包括四个类文件，其中zrdocument类用于XML文件的处理，里边包含了许多处理XML文件，检验模型规范性的方法；zrnode类和zrlink类分别代表了节点和关系连线；zrprocessor类为设计理性知识模型主脉络提取算法。

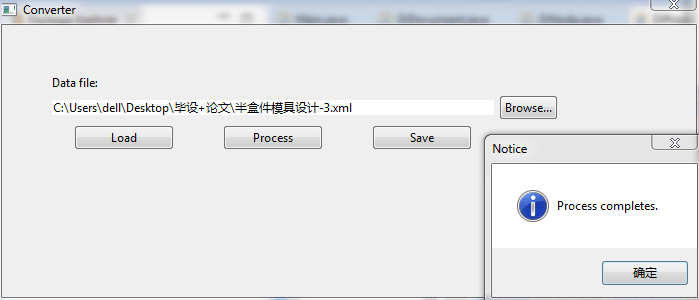
* + 1. 算法程序测试

首先给出半盒件模具设计的设计理性知识模型的XML文件地址：C:\Users\dell\Desktop\毕设+论文\半盒件模具设计-3.xml（在本机上的地址），将其加载到程序当中。



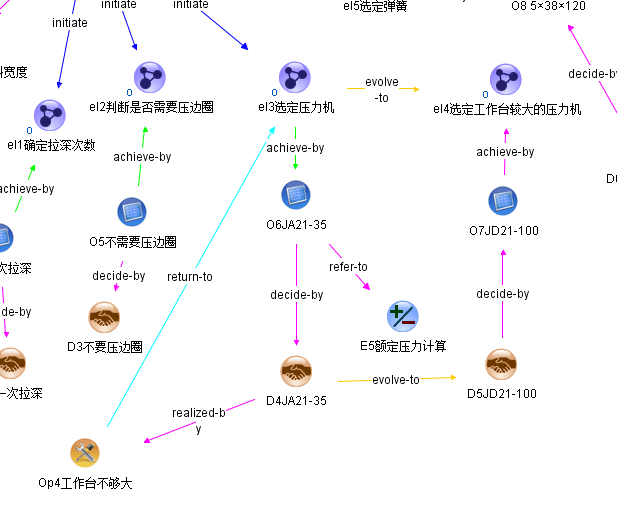
提示成功加载模型实例XML文件，说明文件内容符合规范。

对已经加载的模型实例进行主脉络提取处理：

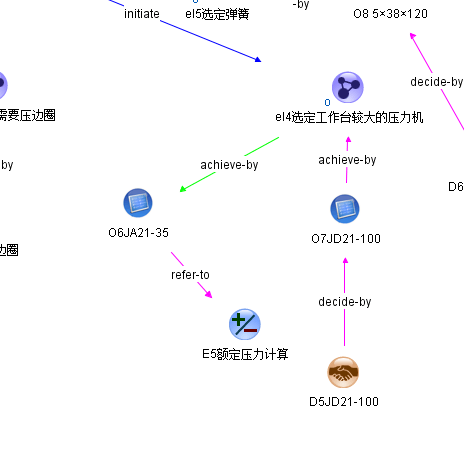


待程序提示提取完毕之后，保存提取后的XML文件，然后用DRTool分别打开提取前后的XML文件，通过直观的可视化图形对二者进行比较。可以看到在“eI3电动机选择”的子集处，约简了重复设计过程。

提取前模型：



提取后模型：



可以看出，经过主脉络提取处理的设计理性知识模型删除了前版本意图，并替换为新版本意图，并删除了前版本意图下的设计决策和设计操作节点以及相关的关系连线。

* 1. 小结

本章实现了上一章所提出的和研究的算法，并通过设计理性知识模型的实例验证了算法的有效性。如上算法课实现设计理性知识模型的规范性检查和主脉络提取功能。同时可以看到，本程序缺乏一个独立的图形化显示功能，这点需要将来继续设计开发。

1. 未来改进及展望

本研究设计和研究的算法可以有效的对设计理性知识模型进行规范性检查和主脉络提取操作。但是同时也有许多的不足之处，可以看到，在图形化显示功能方面的匮乏，使得本程序需要与DRTool同时使用才能达到可视化的效果，样反复的导出和导入过程使得操作十分繁琐，浪费了大量的时间和精力。

同时，这个程序还是一个很独立的个体，对于设计人员没有什么实质性的直接帮助，可以将它继续开发为一个有着更完善的操作界面和多种功能的系统软件，方便设计人员对设计理性知识的重用。

鉴于以上几点不足，有如下的未来发展方向：

1. 设计出更加方便快捷的设计理性知识记录方法，比如人工智能记录。这样可以省去设计人员的大量时间和精力，避免设计者在设计产品的过程中还要分配出一部分时间和精力进行设计理性的记录。
2. 添加图形化显示功能。使得提取前后的模型对比操作变得更加简化，便于设计人员对设计理性的重用。
3. 设计出更加完善和人性化的操作界面，将其发展为一个独立的、包含更多相关功能的系统软件，满足设计人员的更多需求。

结论

对设计理性知识的重用可以帮助设计人员在已有的相似产品设计基础上提高自身的设计速度和效率，避免重复的设计错误，并在一定程度上节约设计成本，使得设计的产品更加完善。本文的主要工作是对设计理性知识模型进行主脉络提取处理，通过对其中的重复设计过程进行约简，对设计理性模型适当抽象，抽取设计决策主脉络，使得整个设计理性模型变得更加清晰易懂，凸显核心设计思维逻辑，更好地支持设计人员对设计理性知识的重用。

本文主要完成了以下工作：

1. 构建了半盒件模具设计的设计理性知识模型及其XML文件。
2. 根据毕业设计过程中产生的模型规范性检查的额外需求，设计了能够对设计理性知识模型进行规范性检查的算法，避免主脉络提取过程中出现差错。研究了已经提出的设计理性知识模型主脉络提取算法，将已有的设计理性知识模型进行主脉络提取操作。
3. 用Java语言及其类库W3C DOM 、J2SE XML 、J2SE SAX实现了（2）中的算法。
4. 用构建好的模型对已实现的算法进行有效性验证，证明了该算法能够实现预期功能。

在未来的研究中，本文中涉及的算法还可以进行进一步优化，使其可以更人性化地进行人机交互。

致谢

此时此刻坐在电脑前编写着我的毕业论文，思绪万分。毕业设计即将结束，而我校园生活的最后时光——大学四年的青春历程也行将谢幕。回想这四年的生活、经历，说长不长说短不短，它承载了我初高中时代对美好大学生活的向往，它也是让我成长最快变化最大的一段人生旅程，是我一生都会铭记的时光。

这四年，我从一个懵懂无知的高中学生蜕变成一个成熟稳重的成年人，这四年让我认识到现实竞争的残酷也让我懂得了同窗情谊的难得，让我明白什么是正确的言辞行为，也让我深刻体会到要对自己负责要对家庭负责要对支持我帮助过我的良师益友负责，这是一个男人最重要的品质，感谢这四年的生活让我学会这一切。

首先要感谢的是对我要求严苛一丝不苟的毕业设计指导教师刘继红教授，刘老师的治学严谨让我印象深刻，每一次检查毕业设计进度和完成情况，刘继红老师都会认真审阅我的成果，并提出很多建设性的意见建议，尤其是在检查我文献翻译的时候，逐字逐句地给我纠正翻译不准确的语句、词汇，给我举例示范，让我对待毕设的态度一直很认真，不敢有马虎大意的时候。刘老师学识渊博，还经常会在我的毕业设计进程中提出很多不一样的理解和看法，让我对自己的课题有更加深刻、透彻和独特的理解。

我还要感谢周建慧学长，学长在我整个毕业设计过程中一直在组织我们的进度检查，事物通知，督促我的毕设进度，开题展示和中期答辩的时候还帮助我修改幻灯片，正是学长一直严格的要求和不断的催促，帮助我顺利完成毕业设计。感谢姜浩博士，姜浩学长在课题内容的研究方案和技术上给了我很大的支持和帮助，对于我，一个只在大一的时候学过c语言，对于编程方面并不十分熟悉更不要说精通的人来说，学长每次的经验传授和问题讲解都让我收获颇丰，让我一步步完成了我的毕业设计，虽然过程很曲折，但有了学长的帮助，结果还算圆满。

感谢这四年来和我一起风雨同舟的110711班的所有同学，我实实在在的兄弟们，我们这样一个全是男生的班级，是整个学院最团结集体。我们一起训练，一起自习，一起哭一起笑，一起跌倒再相互搀扶着爬起来。无论是篮球赛还是足球赛，我们不管输赢都展现出了坚韧不拔的态度，无论饺子节还是毕业晚会的大联欢，我们都积极参与，抓住机会展现自己多才多艺的一面。谢谢你们丰富了我的大学生活，因为有你们这样一群真心实意的兄弟，这个大学才能过的如此顺利和快乐。

感谢班主任张俐老师对对我们的谆谆教诲，是您如父母一般的无私奉献，对我们不管是做人还是做事方面不厌其烦的教导，让我们变得成熟稳重，识得大体。

感谢我们的辅导员范霁康和赵平，是你们伴随了我们机械学院11级四年，你们每一次思想上的的教导和生活上的帮助，让我们更好的融入大学生活，更重要的是让我们在充满了各种诱惑的大学生活中没有迷失自己。

感谢我的父母，是你们给了我一个平凡而温馨的家庭，让我接受良好的教育，也给我物质生活的供给，你们的养育造就了最好的我。

最后，感谢我的培育我的母校北京航空航天大学，正是这样一所治学严谨，学习氛围浓厚的名牌大学，让我接受到国内最优秀的教育，让我享受到一流的学习、生活环境，让我幸福地度过大学生涯。

感谢的话太多，只好藏在心底。大学即将结束，珍惜最后的时光!

参考文献