**基于三维模型的自动语义标注理研究与应用**

摘要

随着计算机辅助设计，三维虚拟化技术的快速发展，在它们的实际使用中产生了大量的三维场景和模型。针对它们的精确检索和分类因此成为有效利用这些数据的关键所在。语义标注技术在一定程度上解决了三维知识管理的难题，但在应用数据越来越趋向海量的今天，其本身也面临着新的挑战。这些挑战包括：如何缩短标注时间，减少标注成本，使其面对数据增长具有可扩展性；如何降低对领域专家的依赖，使其趋向自动化；如何维护标注的一致性，抑制内容相互冲突的标注；以及如何在做到上述几点的同时，仍然保证标注内容与手动完成的相比同等的准确性。

本文以传统的语义标注技术为基础，进行针对标注半自动化过程的优化，来解决现有语义标注过程中重复劳动多，效率低下的问题。本文关注于网络三维文档，并根据其特点提出多侧面的本体模型，对三维场景建模中各领域通用的概念进行抽象以及复用；同时提出标注规则，帮助标注者抽象标注方法，以达到标注过程半自动化以及标注内容一致性可验证的目的。本文最后通过案例验证由文中提出方法所完成的标注和传统的手工标注比较起来，拥有同等的准确性，可以在后续的查询检索操作中起到有效的作用。

完成的工作有：

1. 一、面向网络三维场景，提出一套基于本体和规则的半自动化语义标注方法。给出完整的方法流程，实现框架以及应用场景。
2. 二、提出针对三维场景文档的本体建模。给出用于描述三维场景建模基本概念及属性的通用本体的实现方式。在此基础上，描述从网络三维文档到本体实例的映射算法，该算法能够从三维文档中抽取物体的几何，空间关系等可视属性。
3. 三、提出标注规则的语法和语义，能够灵活地表达特定的标注模式。在此基础上，完成标注引擎的实现算法，包括规则一致性检测，规则在文档中的匹配，以及规则的应用方法。
4. 四、设计了面向网络三维场景基于本体和规则的半自动化语义标注原型系统，并在实际应用场景中验证标注过程半自动化的有效性和准确性

本文所提出的半自动化语义标注方法，基于传统使用本体的语义标注，继承了后者描述精确的特点。此外，通过对它进行改进，实现了利用规则进行标注逻辑抽象，从而进一步达到半自动化的目标。本文还验证了半自动生成的标注内容并没有比纯手动标注降低准确性。因此，该方法有效地提高了网络三维文档的标注效率，在大量数据的情况下，能避免标注过程中过多的重复工作，快速地应对新增的文件，是后续使用中全面的查询与检索操作的基础支撑，对三维场景的语义管理具有较高的参考价值。

关键词：三维模型，语义标注信息，WebGL, 自动语义标注，视频资源, 图像识别。

**3D model-based Semantic Annotation Management Research and Application**

**ABSTRACT**

In recent years, along with the rapid development of 3D and virtual interaction technology, 3D graphic computing has proven its important position in many major fields. As the 3D-related resources has been increased recently, 3D scene and object modeling have gained a lot of attractions and supports from various communities. Unfortunately software products that are capable of manipulating these works are still limited in management aspect, making it difficult to extend them to their fullest potential. Users nowadays demand higher requirements from the lower-layer manage components of a 3D software. First, the software products must be able to provide a powerful search engine, which can filter through a large amount of 3D objects to find the desired 3D models or scenes from provided information. Second, the structure, characteristic or type of a 3D object must be analyzed in a reasonable speed to support the traditional semantic search.

These requirements lead to a creative idea of annotating the semantic contents to 3D objects. Semantic annotation help users not only maintaining the internal data and structure, but also the semantic contents of a 3D object. These semantic contents include the color, characteristic, type and other information such as the usage for applications. To fulfill the feature of combining all the semantic contents together, the annotating process must meet two required conditions.

For the first condition, the method of annotation must have an acceptable accuracy. This means it must have the ability to provide correct and unique information of a 3D object in a specific field. This condition ensures that in the 3D software products, users can easily find their desired 3D objects without filtering too much redundant and unrelated results. Also, user can quickly get the same-type resultant models from a provided type of model.

The other condition, which is efficiency-related, becomes necessary when we have to deal with large amount of data. Assume that annotation itself is a complex, time-consuming and repeated process, the total time which includes duration of user's manual operation and annotation cannot keep up to the standard requirements of a search engine. High-efficient requirements comprise two things: first, there must be supported tools or computer software to let the annotation process become automatic or semi-automatic, and the semantic contents can have fast inquiry and access; second, the repeated process is able to be recognized to be executed automatically.

From the above conditions and requirements, we introduce a WebGL and 3D model-based automatic semantic annotation system. This system is capable of automatically acquire the semantic contents of an object through the data structure analysis or comparing the characteristics between models.

.

**Key words:** software development, software for enterprise management, MDA, MDD, configurable developing platform, reuse of software, automatic code generation

目 录

[第一章 绪论 1](#_Toc265091656)

[1.1 传统管理软件开发面临的问题 1](#_Toc265091657)

[1.1.1 系统需求复杂度变高 1](#_Toc265091658)

[1.1.2 系统需求变更频繁 1](#_Toc265091659)

[1.1.3 手工开发存在缺陷 2](#_Toc265091660)

[1.1.4 系统复用度过低 2](#_Toc265091661)

[1.2 国内外研究现状 3](#_Toc265091662)

[1.2.1 平台化软件开发研究现状 3](#_Toc265091663)

[1.2.2 模型驱动研究现状 3](#_Toc265091664)

[1.3 本文研究内容 4](#_Toc265091665)

[1.4 本章小结 5](#_Toc265091666)

[第二章 NGX平台总体思路 6](#_Toc265091667)

[2.1 MDA基础理论 6](#_Toc265091668)

[2.2 平台化开发过程 7](#_Toc265091669)

[2.3 平台目标及优势 7](#_Toc265091670)

[2.4 本章小结 8](#_Toc265091671)

[第三章 NGX总体架构设计及技术基础 9](#_Toc265091672)

[3.1 NGX总体架构 9](#_Toc265091673)

[3.2 平台技术基础 10](#_Toc265091674)

[3.2.1 应用平台后台技术路线 10](#_Toc265091675)

[3.2.2 应用端UI技术路线 10](#_Toc265091676)

[3.2.3 MDA业务引擎技术路线 10](#_Toc265091677)

[3.2.4 数据库及部署环境 11](#_Toc265091678)

[3.3 本章小结 11](#_Toc265091679)

[第四章 应用平台详细设计及实现 12](#_Toc265091680)

[4.1业务模型的基础要素 12](#_Toc265091681)

[4.2业务模型的构件储存 14](#_Toc265091682)

[4.1.1 元-元模型层中的Application 14](#_Toc265091683)

[4.1.2元-元模型层中的Business Component 14](#_Toc265091684)

[4.1.3 元模型层 16](#_Toc265091685)

[4.2 NGX平台基础框架设计 19](#_Toc265091686)

[4.2.1 平台基础框架结构 20](#_Toc265091687)

[4.2.2授权模块设计 21](#_Toc265091688)

[4.2.3统一数据访问模块设计 22](#_Toc265091689)

[4.2.4统一日志管理模块设计 23](#_Toc265091690)

[4.3 业务引擎设计 24](#_Toc265091691)

[4.4 本章小结 27](#_Toc265091692)

[第五章 测试与应用结果及分析 28](#_Toc265091693)

[5.1 验证对象MES系统描述 28](#_Toc265091694)

[5.1.1 MES的计划调度模块总体描述 28](#_Toc265091695)

[5.1.2计划调度功能描述 29](#_Toc265091696)

[5.1.3 计划调度业务流程描述 29](#_Toc265091697)

[5.1.4 计划调度用例描述 30](#_Toc265091698)

[5.1.5 计划调度数据流程描述 31](#_Toc265091699)

[5.2 验证过程及结果 33](#_Toc265091700)

[5.2.1 计划调度业务建模 33](#_Toc265091701)

[5.2.2 使用NGX平台生成业务功能并进行测试 35](#_Toc265091702)

[5.2.3 修改业务模型重新构建 39](#_Toc265091703)

[5.3 结果分析 40](#_Toc265091704)

[5.4 本章小结 43](#_Toc265091705)

[第六章 结论 44](#_Toc265091706)

[6.1 工作总结 44](#_Toc265091707)

[6.2 平台研究工作展望 44](#_Toc265091708)

[参考文献 46](#_Toc265091709)

[谢辞 48](#_Toc265091710)

**第一章 绪论**

随着信息技术在不断发展，工业生产的信息化步伐在逐步加快，信息系统在企业生产管理的地位已经成为现代企业提高企业管理效率和水平的核心竞争力。目前，企业信息化已成为企业管理规范化、科学化、自动化、系统化的总称，是企业未来生存和发展的必由之路。企业信息化发展取得了一定的成效，积累了一些宝贵的经验，越来越多的企业已意识到信息化的对企业发展的极端重要性。随着信息产业在国民经济中所占比例的提高，成为国民经济的主导产业，并带动其他传统行业包括农业和传统工业发展到新的高度[1]。企业广泛应用信息技术，可以有效加快企业产品更新换代，提高[市场](http://www.lunwentianxia.com/class_free/79_1.shtml)竞争力。通过推广[计算机](http://www.lunwentianxia.com/class_free/19_1.shtml)辅助设计和辅助制造技术，可以达到提高生产过程的自动化程度的目的。而建立管理信息系统和决策支持系统，可以促进企业生产技术和管理信息化，提高企业的整体素质。

随着经济全球化的进程和市场竞争的加剧，企业面对竞争更激烈、业务变更和整合更频繁的全球化运营环境。商务环境的变化正深刻影响着企业组织管理的各方面。在新的环境下，作为企业业务运行的核心支持系统的企业信息系统(Enterprise Information Systems，EIS)面临前所未有的挑战[2]。由于企业的信息化管理需求在不断变大变复杂，面对不断变更的业务规则以及企业管理需求，传统的软件开发方式不能有效满足系统的快速开发与快速变更。信息系统的分析、开发、测试、部署以及维护的工作的难度随之不断提高,导致企业信息系统的开发效率与质量面临巨大的挑战。因此，如何快速有效的开发并运用企业级应用成为业界研究的焦点之一。

* 1. 传统管理软件开发面临的问题

随着信息化的普及，软件系统变得庞大多变，除了一些软件公司存在着规模小、技术力量薄弱、管理薄弱、设计人员缺乏等问题，还有一系列问题制约着企业软件的快速有效开发。

* + 1. 系统需求复杂度变高

当前，由于企业信息化过程的不断扩展，企业信息系统包含的业务不断增加，企业信息系统的开发因此便涉及到大量的企业业务规则。在系统的开发过程中，软件开发人员必须准确理解这些规则才能确保系统符合企业的要求。为了做到这点，对于系统分析员需要花大量的时间对这些规则进行调研、分析和整理，而对于软件开发人员，需要一个漫长的过程来理解这些复杂的规则，才能着手于相应的开发。于是系统需求的复杂性会影响整个开发过程的效率，甚至会由于对业务规则的模糊可能造成需求理解的错误而导致系统需求定位错误，造成系统的无法正确完成。

* + 1. 系统需求变更频繁

需求变更是因为需求发生变化。根据软件工程思想，需求说明书一般要经过论证，如果在需求说明书经过论证以后，需要在原有需求基础上追加和补充新的需求或对原有需求进行修改和削减，均属于需求变更。需求变更的出现主要是因为在项目的需求确定阶段，用户往往不能确切地定义自己需要什么。用户常常以为自己清楚，但实际上他们提出的需求只是依据当前的工作所需，而采用的新设备、新技术通常会改变他们的工作方式;或者要开发的系统对用户来说也是个未知数，他们以前没有过相关的使用经验。因此在项目需求确定阶段，往往不能清晰的确定系统的需求。随着开发工作的不断进展，系统开始展现功能的雏形，用户对系统的了解才会逐步深入。于是，他们可能会想到各种新的功能和特色，或对以前提出的要求进行改动。他们了解得越多，新的要求也就越多，需求变更因此不可避免地一次又一次出现[3]。

然而软件项目的需求频繁的变化就意味着系统分析员要对需求重新修改制定，开发人员要不断返工来实现系统与需求的一致性。如果需求变化过大还可能导致项目失败。

* + 1. 手工开发存在缺陷

虽然经过软件工程学者根据自身的实践经验以及深入研究已经形成十分成熟的软件过程体系，从很大程度解决了软件开发过程中的矛盾和风险，然而使用传统的人工编码方式开发信息系统的成本代价仍然非常高。一个较大型的信息系统从业务分析建模、软件架构设计、代码实现、功能和非功能测试一直到软件部署运行是一个很长的周期，并且每个环节都要面临一些开发风险。因此，软件开发很难达到理想化的高效与高质量。

手工开发主要面临以下问题[4]：

(1)产品开发技术路线的选择比较随意

由于每个开发组有各自的路线和偏好，产品开发的技术路线可能各不相同。这使得软件系统很难适应未来产品的发展的需要，软件不能进行无缝的升级。产品和项目界限不分明，通用性、定制性比较差。产品缺乏良好的结构设计，代码难以维护。

(2)产品的质量很难保证

由于每个项目单独开发，每个项目都要在各个方面做单独考虑。对于比较小型的开发团队，很容易出现产品对安全性、性能、部署环境往往考虑不够。

(3)产品开发对开发人员的要求比较高、人员的流动对开发的进度影响很大

在很多公司中开发人员没有独立的分工，每个人都负担着一个模块的需求、设计、实现，很多开发人员甚至需要负担一定的美工工作，产品的测试也不是很细致，开发人员负担角色过多的后果就是每部分的工作都很难做得十分到位。

(4)设计能力不足

很多项目经理和开发人员对设计重视不够，为了赶工期，很多项目的设计过程过于简单，有的甚至根本没有设计过程。

(5)缺乏软件开发效率

由于开发团队的各异，有些开发团队容易出现找不到软件工程或者项目管理的方法能够大幅度提高应用软件的开发效率。开发周期长、开发费用高，实施费用超支和工期延长，已经司空见惯。更加可怕的是，随着企业的环境和需求的不断变化，“建成即成闲置”，形成软件工程的灾难。

* + 1. 系统复用度过低

从切面的角度看，许多企业信息系统存在共同的功能及模块。传统的软件开发往往只针对一个项目需求来开发，导致有些通用功能出现项目依赖性，再加上每个项目技术路线各不相同，每个项目的相似功能的模块差异性较大，无法实现很好的软件复用，需要重新开发。

* 1. 国内外研究现状

对于软件开发的质量和效率无法满足企业需求的问题，解决方法一是靠软件工程，厂商采用更科学、更规范的流程组织项目开发；二是靠软件技术。而就软件技术而言，平台化技术是软件产品发展的重要趋势。软件开发的问题是由多方面原因造成的，有市场的原因、有公司管理水平的原因，也有技术的原因。目前基于业务基础软件平台的开发方式是解决这些问题的一个比较好的解决办法。而模型驱动开发是有效实现业务与技术分离的方法，可以解决项目的业务需求复杂并且频繁变更的问题。

* + 1. 平台化软件开发研究现状

近年来，软件平台化成为热门话题，它使得供应商、中间商、用户共同进入软件服务世界，使得中间商与用户变得主动，真正实现了软件产业联盟的生态化与动态化。业务基础软件平台是一种技术创新，它使软件平台又多了一个层次，并将应用软件的业务逻辑和开发技术分开，使得应用软件的开发者可以仅仅关注应用软件的业务过程，而不必关注其技术的实现。这使管理与业务人员参与应用软件的开发成为可能[5]。企业通过定制已有的信息系统模块进行组合配置。这种方式相对于重头开发，大大缩短了开发周期，缩减了开发成本。

业务基础软件平台包括集成应用平台、开发体系两个部分。从技术角度分析，业务基础软件平台为复杂应用软件系统的开发提供了一个基本框架，并有与之相应的、 方便易用的开发与维护管理工具。这个框架给出了一些复杂应用软件的基本组成部分和实现方法，并且预置了很多供参考的软件模块。有了这样的准备，在业务基础软件平台之上开发管理软件就可以降低复杂性，省去很多基础性的研发工作，从而大大缩短研发周期，提高研发效率。具体来说，业务基础软件平台能满足复杂应用软件系统开发的如下要求：首先是速度要求。通过业务基础软件平台提供的基本框架，以及预置好的模块，软件提供商 能很快地研制出用户所需要的复杂应用软件系统。其次是灵活性要求。通过业务基础软件平台提供的开发与管理工具，软件提供商能很方便地满足用户个性化的需求，以及用户在发展过程中各种各样变化的需求。再者是集成性要求。业务基础软件平台为复杂应用软件系统提供了一个集成框架，不仅为集成同一平台上的各种不 同软件提供了规则，还为集成其他应用软件系统提供了集成接口[5]。

目前，企业管理软件正在逐步向平台化的方向演变，包括国际厂商如SAP、甲骨文，国内厂商如用友等都试图建立一种应用的平台，这个平台又分为供开发者使用和用户使用的两种平台，其目的是实现软件技术和业务管理的分离，使应用软件在某种程度上做到与技术无关，而是面向具体业务。

* + 1. 模型驱动研究现状

模型驱动架构(Model Driven Architecture， MDA)是由OMG定义的一个软件开发框架。它源自于众所周知的把系统操作的规范从系统利用底层平台能力的方式细节中分离出来的思想，是一种独立于特定平台和软件供应商的软件体系结构设计和开发方法[6]。MDA提供了一种途径来规范化一个平台独立的系统、规范化平台、为系统选择一个特定的实现平台，并且把系统规范转换到特定的实现平台[7]。MDA是一种独立于特定平台和软件供应商的软件体系结构设计和开发方法，它适用于设计、部署、集成等软件开发的整个生命周期。它可以简化系统集成、缩短开发周期和节省企业资源。

MDA还处在一个发展的过程中，MDA还在不断的演进。虽然MDA正朝气蓬勃地走来，但是人们也能看出它所存在的问题。MDA最大的好处就是业务模型的持久价值，但是付出的代价是增加了抽象层，而目前看来，层之间的转换并不是我们所期待的那样顺畅，至少，从PIM到PSM，从PSM到代码，这个实现的过程要远比从3GL生成机器代码来得困难。在建模技术方面，UML正在暴露其固有的缺陷，它需要扩展更多的机制来支持精确建模和分析模型，虽然目前OCL为精确建模提供了一定的支持，但是这种支持距离可执行模型的理想还很遥远。回顾MDA的历史，我们可以看出UML的巨大成功为MDA的产生奠定了坚实的基础，同时也感觉到：在由软件工艺到软件工程的漫漫长路中，MDA只不过是向前迈进了一小步，但却给整个软件业掀起了一场波澜，它在模型定义、开发过程等诸多方面都将对未来IT技术产生深远的影响。

MDA在企业应用市场方面目前的情形是：由于从PIM 到PSM转换方法的标准化尚未完成，IBM、Borland等大型厂商大都持谨慎态度，虽然也纷纷在他们的开发工具中提供部分的MDA功能，但并没有完全遵循OMG定义的MDA规范[6]。虽然如此，IBM除了在Rational中增加MDA功能之外，在开源项目Eclipse中，也提出了 EMF（Eclipse Modeling Framework）[17]这一创新的MDA代码生成系统项目，由此可见IBM对MDA这一发展中的技术的重视程度。Borland公司宣称他们也在关注MDA 技术，并且准备在Together中配置基于MDA的模型自动生成功能[8]。相对于业界大厂的冷静和矜持，一些中小厂商反而特别活跃，像Interactive Objects公司著名的ArcStyler[14], Compuware公司著名的OptimalJ[15]，还有开放源码的AndroMDA[16]等遵循OMG标准规范的 MDA工具已在一些项目中得到了广泛的运用，并取得了显著的成效。

而在MDA研究方面，目前对于模型驱动的研究大都是面向软件设计开发期,即针对上MDA开发过程的某一方面或其应用。目前对模型转换的方法进行研究，如文献[17～22]所述，还有将模型驱动开发应用于系统和分布式系统开发的方法，如文献[24，24]所述。

* 1. 本文研究内容

根据上述分析的本课题通过分析企业管理系统的软件开发过程中可能存在的问题，结合模型驱动开发与平台化软件开发的理念，本文提出一个基于MDA的企业管理平台NGX整体架构方案。

（1）本文提出NGX平台的基本形式，NGX平台分为两大部分，包括一个可视化的业务建模工具NGX Studio，以及一个自动部署集成应用平台NGX Application。通过使用该平台，业务开发人员通过使用业务过程建模的可视化业务建模工具对业务进行建模或调整业务模型，然后由建模工具解析并自动转换为J2EE体系的平台相关模型存储于构件库中。然后通过平台提供的业务引擎解析PSM并生成为可执行组件代码，业务引擎还将自动部署，结合平台的各个层面上的基础构建快速形成一个有效的企业生产管理应用。

（2）本文通过解析、分解一般企业信息系统存在的业务要素提将给出企业管理系统模型驱动开发的基本元素，并说明这些要素如何应用数据库形成一个业务模型构件库。

（3）然后本文说明了NGX平台基础框架的架构以及扩展集成方式。

（4）本文还将说明NGX平台的几个核心模块的详细设计以及实现方法。其中包括基于RBAC权限控制、数据统一访问、日志管理以及执行业务模块自动生成部署的业务引擎。

（5）最后，本文以制造执行系统 (Manufacturing Execution System, MES）的计划调度模块为主要分析对象分析并验证平台的可行性与柔性。并通过分析实验结果，与已存在的软件开发方式进行对比，并推测其后续可扩展性。

* 1. 本章小结

本章提出信息系统在企业生产管理的地位已经成为现代企业提高企业管理效率和水平的核心竞争力，企业管理软件的开发应用越来越广泛。然而由于系统需求的膨胀与频繁变化，企业管理软件的开发遇到了一系列挑战。传统的开发方式存在一定的问题。为了提高企业管理软件的开发效率和质量，业界在平台化软件开发与模型驱动开发方面做了一系列研究。最后，本文通过平台化软件开发与模型驱动开发的结合，提出一个基于MDA的企业管理平台的架构方案，并展现其中关键部分的详细设计和实现。然后通过MES系统的微小模块验证平台的可行性与柔性，与已存在的软件开发方式进行对比，并推测其后续可扩展性。

**第二章 NGX平台总体思路**

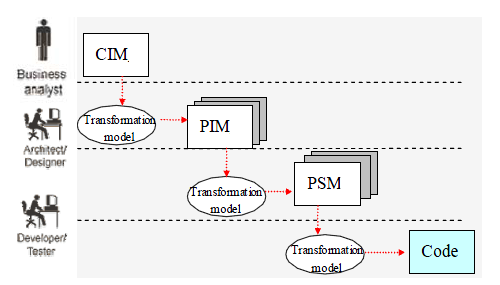
根据上一章分析，为了解决企业对系统的需求相对复杂、需求变更比较频繁以及手工开发企业管理软件遇到的各种问题，本文在此提出一个基于模型驱动架构的企业管理软件开发平台NGX(Next Generation X System)。

2.1 MDA基础理论

模型驱动开发(Model Driven Development， MDD)是一种创新的软件开发方式，它满足一般工业对软件开发的需求，如减少经营成本，减少产品推向市场的时间，以及对开放的解决方案的需要[9]。

MDA的出现，为提高软件开发效率，增强软件的可移植性、协同工作能力和可维护性，以及文档编制的便利性指明了解决之道。MDA被面向对象技术界预言为未来两年里最重要的方法学。当今建模的主要问题在于，对于很多企业来说它只是纸面上的练习。MDA的愿景是定义一种描述和创建系统的新的途径。MDA 使得UML 的用途走得更远，而不仅仅是美丽的图画。很多专家预言MDA 有可能会带领我们进入软件开发的另一个黄金时代[10]。

MDA提升了模型在软件开发中的地位, 使模型成为软件开发的基本工件。如图2-1所示，一个经典的基于的软件开发过程分为个步骤[11]: ①由领域专家建立计算无关模型(computation independent model, CIM), ②依据映射规则将转换为平台无关模型(platform independent model, PIM) ③由技术人员依据映射规则将转换为平台相关模型(platform specific model, PSM),④最后将转换为系统实现、测试框架和部署脚本。



**图2-1 MDA的软件开发过程**

计算无关模型CIM是MDA基于计算无关视角建立的系统模型，用于描述系统需求、功能、行为和运行环境，也称为业务模型。被称为计算无关，主要因为CIM侧重于表述系统的外部行为和运行环境，而不表现系统的内部结构和实现细节等相关内容。CIM为领域专家与系统设计专家之间关于领域需求的沟通和交流提供了桥梁，并直接支持PIM、PSM模型的构造和实现。平台无关模型PIM是MDA基于平台无关视角建立的系统模型。PIM 是抽象出的业务逻辑。被称为平台无关，主要因为PIM不包含与实现平台和技术相关的特定信息。PIM 所表现出的平台无关性，使其能够在任何技术平台上得以实现。平台特定模型PSM是MDA基于平台特定视角建立的系统模型。PSM 从相应PIM转换而来，它既包含了PIM 中所定义的业务逻辑规范，也包含了与选定平台和技术相关的实现信息细节。

2.2 平台化开发过程

该平台依据模型驱动开发架构的思想，该平台主要特性是将业务逻辑模型与底层实现平台分离。将业务逻辑模型与底层实现平台分离是解决当前需求过于复杂问题的可行思路，同时也将成为一种趋势[12]。因此NGX平台分为两大部分，包括一个可视化的业务建模工具NGX Studio，以及一个自动部署集成应用平台NGX Application。

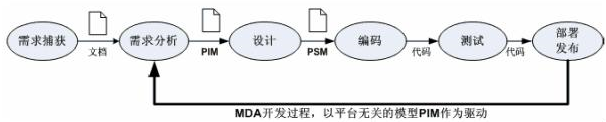
通过使用该平台，业务开发人员通过使用业务过程建模的可视化业务建模工具对业务进行建模或调整业务模型，将计算无关模型CIM转换为平台无关模型PIM，然后由建模工具解析并自动转换为J2EE体系的平台相关模型PSM存储于构件库中。然后通过平台提供的业务引擎解析PSM并生成为可执行组件代码，业务引擎还将自动部署，结合平台的各个层面上的基础构建快速形成一个有效的企业生产管理应用。

NGX要求从需求模型向代码模型的转换是自动的, 对终端系统用户是透明的。问题的求解目标不是代码模型, 而是与需求一致的业务模型, 是“所见即所得”的需求模型。这就是软件运行期模型驱动,模型在软件运行期发挥作用, 通过对模型的操纵实现对最终应用系统功能、行为和结构的控制。

2.3 平台目标及优势

通过NGX平台，主要是实现企业管理软件的模型驱动开发。从而可以使行业专家专注于软件的行业特性，而技术专家将集中精力用于构建MDD的工具。构建复杂的应用程序将不再是精英程序员的专利。将允许行业专家用他们自己的知识系统使用特定的符号构建一个模型，并使之融入高层的程序模型中。在使用MDD开发的项目中，开发人员很少进行重复性的工作。他们将有更多的机会在他们的工作中发挥创造性[13]。

整个过程应用模型驱动开发方法所倡导的以PIM作为核心驱动，从而提升应用开发层次，利用代码自动生成的方法提高了开发效率，并且增加了应用模型复用程度。如图2-2所示，通过以验证过的平台化的模块封装和复用以及代码的自动构建及部署，直接缩短需求分析、设计、编码、测试以及部署发布的软件过程。同时平台允许反复修改业务模型并构建软件，极大减少了软件开发迭代的周期，从而促使软件实体与实际业务的之间差异的快速收敛。在提高开发效率的同时，因为可以缩短软件产品的交付日期，提早上市时间，同时也降低了开发成本。



**图2-2 MDA开发周期**

由于代码自动生成会经过反复的修改测试，这一定程度上保障了软件的质量和健壮性、扩展性，并确保了软件技术路线的统一。由机器自动生成的代码减少了人为出错的因素，可以确保软件开发人员专注于程序功能的测试。

同时使用MDD进行项目开发时，因为模型就是文档。当使用正确的抽象方法，模型的描述对行业专家和项目需求方具有很高的易读性，因此不必担心文档与代码的一致性出现问题。

2.4 本章小结

本章为了解决企业对系统的需求相对复杂、需求变更比较频繁以及手工开发企业管理软件遇到的各种问题，提出一个基于模型驱动架构（MDA）的企业管理软件开发平台NGX的基础构想。通过对MDA的基础理论的分析，将NGX开发平台分为两大部分，包括一个可视化的业务建模工具，以及一个自动部署集成应用平台。然后提出NGX平台的使用形式以及它所要达到的主要目标。

**第三章 NGX总体架构设计及技术基础**

如上一章所述，结合模型驱动开发与平台化软件开发的企业管理软件平台NGX拥有诸多优势。本章在此提出平台的总体架构设计及基础要素。

3.1 NGX总体架构

企业管理软件平台NGX平台分为可视化的业务建模工具NGX studio和自动部署集成应用平台NGX Application。

NGX的总体结构如图3-1所示：

  **图3-1 NGX平台的总体结构**

（1）建模工具（NGX Studio）

NGX Studio是一个可视化开发平台。它负责可视化地将用户的CIM模型转换为PIM模型。NGX Studio通过配置会绑定一个模型存储数据库（NGX Business Model Database）。NGX Studio直接对业务模型数据库进行操作，生成的业务模型按照业务组件的维度存储在业务模型数据库中。

（2）应用集成平台（NGX Application）

NGX Application是一个应用平台，它是一个业务代码生成引擎与基础切面模块的集合。

管理与监视模块（NGX Admin& Monitor）负责让系统管理员进行配置和监视应用平台。系统管理员通过该模块可以配置该应用的模型数据库（NGX Business Model Database），并选择导入需要的业务模型进行平台动态更新升级。是基于Web的系统管理员的操作平台。通过它，系统管理员还可以查询业务运行日志，监控业务的运行，对业务操作人员的流程动作进行监控，对系统运行的业务流程实例实施挂起、恢复、回退等操作。

业务代码生成引擎（NGX Business Engine）负责从模型数据库导入业务模型，将PIM解析并自动生成包含业务定义的J2EE体系的PSM（平台相关模型），并转化为可执行组件代码（NGX Business Component）。业务引擎还将自动部署，它将业务组件与平台基础设施结合形成一个有效的企业生产管理应用。

平台基础设施（NGX Business Infrastructure），它是面向切面的企业管理通用模块结合。它包含一个企业应用的各项基础模块。Web Framework是应用基础框架Authority负责应用的认证和授权，Data Bus是应用各个模块数据传输总线，DAS（Data Access Service）负责所有数据访问，Logger负责应用的日志记录，Process Engine负责工作流的执行，Report Engine负责业务报表的生成。

应用服务器（Application Server）是应用部署的环境。

应用数据库（Application Database）是负责存放业务数据的数据库。

应用配置（NGX Business Configuration）基于Web的业务配置工具，它支持业务部门的业务管理人员基于Web的方式进行业务流程、业务事件、业务策略、业务规则进行配置与调整，为业务人员提供完全的业务域支撑能力，敏捷响应业务的变更。

（3）模型元数据库（NGX Business Model Database）

NGX Business Model Database是平台无关业务模型（PIM）存储数据库，它通过配置将NGX Studio与NGX Application有机结合起来。

3.2 平台技术基础

本项目分为可视化的业务建模工具，以及一个自动部署集成应用平台。前者使用Eclipse插件开发平台实现，后者使用Java EE作为主要开发路线。在此，本文主要关注自动部署集成应用平台的技术基础。

3.2.1 应用平台后台技术路线

应用平台采用轻型Java web开发框架的技术路线，使用包括spring-MVC、spring-webflow、hibernate等热门开源框架。

Spring 是一个开源框架，是为了解决企业应用程序开发复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许您选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架。它在面向方面编程（[Aspect-oriented programming](http://en.wikipedia.org/wiki/Aspect-oriented_programming" \t "_blank), AOP）和控制反转（Inversion of Control,IOC） 容器等方面有着很合理的设计。再加上spring-MVC提供了一个很好的层次结构，spring-webflow可以将整个页面流程以工作流的形式定义，大大提高了企业应用的需求适应性。

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它通过对象关系的映射可以为项目的数据持久化提供很好的支持。

3.2.2 应用端UI技术路线

应用端UI使用Ajax框架Ext-JS。一方面，它提供了友好的Web 2.0的富客户端技术，另一方面，它可以用JSON的形式化方式表达页面，这对模型驱动开发的形式化转换有很大的帮助。

3.2.3 MDA业务引擎技术路线

MDA业务引擎主要功能是从数据库读取由可视化建模工具确定的平台无关模型（PIM）并负责生成平台相关模型（PSM）以及最终代码，并与平台基础框架进行共同部署。

在PSM读取方面，它使用了hibernate框架。

在PSM的数据持久化生成方面，它使用了hibernate tool开源库，负责进行数据库持久化的自动转化工作。

在最终软件代码生成方面，主要使用了由apache.org小组负责开发的开源的模版引擎Velocity。

在项目自动部署方面，主要使用了由apache.org小组负责开发的开源的ant技术。

3.2.4 数据库及部署环境

根据平台的特性是高度可配置特性，数据库与部署环境都应该可通过配置来集成。理论上数据库支持Mysql、Oracle、DB2等一系列关系型数据库，部署环境可以支持Tomcat、Jboss等Java EE应用服务器。

平台当前依赖的开源技术基础如下：

(1) spring-2.5.5

(2) spring-webflow-2.0.8

(3) hibernate-annotations-3.4.0.G

(4) hibernate tool

(5) hibernate-distribution-3.3.2.GA-dist

(6) ext-js 3.0

(7) velocity 1.6.2

(8) ant 1.7.0

(9) tomcat 6.0

(10) Mysql 5.1

3.3 本章小结

本章通过分述建模工具、应用集成平台与模型元数据库的概念和关系描述了企业管理平台NGX的总体架构。然后，本章并说明了NGX平台的技术路线以及使用的技术基础。

**第四章 应用平台详细设计及实现**

4.1业务模型的基础要素

NGX根据元对象设施(meta object facility, MOF) [26]规范的元数据结构, 建立如下元数据体系：

元—元模型层:描述了元模型的基本结构和语义。元一元模型通过对元模型的统一描述, 实现对不同元摸型的管理, 进而实现对异类资源的控制[25]。元一元模型包括应用模型和业务组件模型。

元模型层：描述了模型的基本结构和语义。一个元模型负责对一类模型进行统一的描述, 实现对一类资源的集成和管理, 如数据元模型对不同的数据模型统一描述, 成为数据集成的基础。元模型包括业务概念模型，业务表单模型，业务页面模型，业务逻辑模型，业务功能模型，业务流程模型，业务报表模型，外部服务模型。

模型层：元数据的集合就是模型, 一个模型描述的是一类实例。业务用户使用模型描述需求, 通过执行模型完成业务。一个新的模型通过对其元模型的实例化建立[25]。

实例层：模型实例化的结果, 该层和最终用户直接打交道。模型执行时, 输入不同的参数值, 会得到不同的实例。一个用户界面、一张报表都属于这个层次[25]。

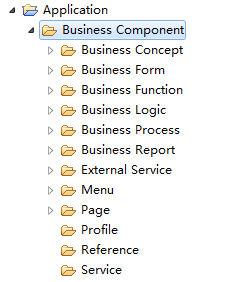
下面详细说明元—元模型层与元模型层的基础要素：

（1） 应用模型

每个企业管理应用的业务模型组织为一个Application模型。每个Application模型将作为插件形式单独部署到平台之中。一个客户可能会针对不同的业务，部署多个Application。由于基于统一的技术架构体系，不同Application之间可以方便的集成与沟通。每个应用模型单独有一个配置信息记录应用自动部署的关键信息。

（2） 业务组件模型

每个Application模型按照业务组件（Business Component）的维度对业务进行合理的切割和管理。每个组件的结构如图4-1所示。



**图4-1 业务组件模型的树状结构**

一个Application包含一到多个业务组件，一个业务组件封装了客户的具体业务，如客户管理、生产管理、计划管理、销售管理等等。一个业务组件是一个业务模型的综合体，可视化建模工具会从概念建模、展现建模、流程建模、逻辑建模等几个方面表达了特定业务。每一个NGX业务组件都有Service、Reference、Profile等三个属性。Service、Reference主要用来对NGX组件进行装配时使用。Service表达了NGX组件对外提供的服务，Reference表达了为了实现NGX组件的特定业务，该组件引用了哪些其它组件提供的服务。

（3） 业务概念模型

业务概念（Business Concept）是可视化建模工具对业务组件进行业务建模的基础，即概念建模。业务领域中的一些核心概念需要在业务建模时首先提取出来，如销售管理这个业务组件，其核心概念包括：客户、订单、订单条目、商品等。

一个业务概念（Concept）由很多属性（Property）来描述它，属性则需要使用业务数据类型（Business Data Type）来描述，业务数据类型的定义也属于概念建模的一部分。业务数据类型包括简单数据类型（Simple Data Type）、复杂数据类型（Complex Data Type）以及其它概念（Concept）。

概念对应数据库中的表（单个概念属性的集合）或视图（按照一定规则的多个概念属性的集合）。属性分为虚拟属性和实体属性，实体属性对应数据库中的字段，虚拟属性仅仅是作为特殊的信息展示而用，不存在对应的物理存储，如订单的“金额合计”属性；属性上可以配置一定的赋值逻辑，即该属性的值是依据一定的逻辑生成，还是手工录入。业务类型则对应数据库中的字段类型。

简单数据类型包括整型、长整型、日期型、双精度型、时间型、字符串型等。简单数据类型是基础数据类型，可以直接映射到数据库和java的基础数据类型。

复杂数据类型是简单数据类型的封装与扩展，复杂数据类型是以简单数据类型为存储结构的，主要包括Email类型，图片类型、附件类型、单选类型、多选类型、密码类型、验证码类型、文本类型、HTML文本类型。

概念类型即该属性是另一个概念的实例，如订单有一个属性“客户”，该属性的类型为“客户”这个概念。概念类型在存储结构上实现时，即为指向该属性所属概念类型的一个引用。

单选类型、多选类型以及概念类型都有一个对应的存储结构，单选类型、多选类型对应的存储结构就是Application的数据字典，概念类型的存储结构对应Application数据库的表，如客户表、商品表等。

每一个业务数据类型都可以配置一个正则表达式，用于控制数据的正确性校验。

（4） 业务表单模型

业务表单（Business Form）是展现建模的基础元件，业务表单是客户应用中用于供用户操作业务概念的用户接口。一个业务表单涉及到对一个或多个业务概念的操作。如，录入一个销售订单，可能涉及到一个订单录入表单，订单详情展示表单。

业务表单按照类型可分为基础表单和复合表单，基础表单包括：新增表单、编辑表单、展示表单、列表表单、树状表单、可编辑表格表单、属性表格表单等几种。

复合表单则是通过将几种基础表单按照一定布局组合而成的复杂表单。

（5）业务页面模型

业务页面（Page）对应客户Application中的用户界面，Page用于将多个业务表单按照一定的框架整合在一起，给用户提供与系统交互操作的界面。每个页面对应一个页面跳转地址。

（6）业务逻辑模型

业务逻辑（Business Logic）表达了业务的具体执行逻辑。如“录入订单”可能会对应一个处理订单的业务逻辑，改逻辑将用户通过表单输入的信息持久化到数据库中。

（7）业务功能模型

业务功能（Business Function）可以分为手动功能与自动功能，手动功能建模是业务建模的核心，业务功能建模以Form、Business Logic为基础，以业务功能点为单元，将相关表单、逻辑按照一定的处理流程组织在一起，以实现特定的业务功能。表单会在

如“录入订单”这一个功能点，就涉及到订单列表界面、订单输入界面、订单详情界面、订单处理逻辑等几个元件。

业务功能可以嵌套，即将多个业务功能组合起来完成一个复杂功能。

自动功能建模通常使用Job调用特定的业务逻辑完成。即用户定义一些特定的Job，来自动处理相关任务。

（8）业务流程模型

Business Process（业务流程）

对业务领域的核心过程进行建模，即建立业务的流程模型。

（9）业务报表模型

用户可以定制相关的业务化报表（Business Report），报表可以装配在Page中，用于特定数据的展示。

（10）外部服务模型

外部服务（External Service）主要用于表达对企业历史遗留系统、数据源、遗留系统中的一些业务逻辑、数据源中的存储过程等的封装与使用。将外部系统的信息通过建模包装，外部服务以一个服务引用方式存在。

4.2业务模型的构件储存

NGX平台的业务模型通过可视化建模工具构建、解析并存储于业务模型数据库中。模型数据库以模型构建库的形式存在。在模型数据库中，业务模型以平台相关模型（PSM）的形式存在。根据业务模型的核心概念，数据库中的存储结构设计如下：

4.1.1 元-元模型层中的Application

Application是元-元模型层中的关键的结点。它是所有业务组件的核心连接点。它除主键Id外包含属性依次为应用名称、应用用户序列号、应用创建时间、应用最后更新时间、应用作者、应用描述、应用版本、应用用户名、应用版权、是否多重租赁。

一个Application会对应1个相应的Application Configuration作为应用配置信息的存储。Application Configuration除主键外包含属性依次为应用数据库类型、应用数据库用户名、应用数据库密码、应用数据库地址、应用数据库端口。Application与Application Configuration的关系如图4-2所示：一个Application对应1个Application Configuration，而Application Configuration对应0－1个Application。

**图4-2 Application与Application Configuration存储关系图**

4.1.2元-元模型层中的Business Component

Business Component是业务模型的最直接包装，是对业务进行合理的切割和管理的基本单位。它除主键外包含属性依次为组件序列号、组件名称、组件版本号、组件创建时间、组件最后更新时间、组件作者、组件描述。它与Application的关系如图4-3所示：一个Application对应1－N个Business Component，而Business Component对应0－N个Application。

****

**图4-3 Application 与Component存储关系图**

为了增强业务组件之间的互相引用关系，在这里引入了Service和Reference的概念，使业务组件可以通过提供服务引用服务来达到业务组件的连锁更新。REFERENCE除主键外包含属性为引用名称。Service除主键外包含属性为服务名称、服务描述。如图4-4所示，每个Business Component可以提供0－1个服务，可以使用0－N个引用。



**图4-4 Component与Service、Reference存储关系图**

4.1.3 元模型层

业务组件下包含各种业务模型，因此在此使用资源（Resource）的概念。Resource除主键外包含属性依次为资源名称、资源标名、资源是否对外可用、资源类型、资源描述。如图4-5所示，一个Business Component对应0－N个Resource，并且Resource依赖于对应的Business Component的存在。

Resource是组件中所有模型的总称，具体可以扩展为Business Concept、Business Form、Business Page、Business Logic、Business Function、Business Process、Business Menu、Business Report等各种模型类型。



**图4-5 Resource与Component的依赖关系图**

1. Business Concept

Business Concept是业务实体（Business Object）的模型化，它除拥有Resource的所有属性外，还有属性为概念类型、概念的数据库建表脚本。每个业务概念会包含多个属性（Concept Property）。Concept Property除主键外包含的属性为属性名称、属性标称、属性类别。它们之间的关系如图4-所示：一个Business Concept可以包含0－N个Concept Property，一个属性只对应1个Business Concept。



**图4-6 Business Concept与Property存储关系图**

每个属性（Concept Property）都会对应一个数据类型（Business Data Type）。Business Data Type除主键外包含的属性为类型名称、类型标称、是否是符合类型。一个它们之间的关系为：Concept Property对应一个Business Data Type，多个Concept Property可公用同一个Business Data Type。由于数据类型又可考虑分为没有嵌套的简单类型和嵌套了子数据类型类型，如图4-7所示，Business Data Type可以包含0－N个Business Data Type。



**图4-7 Concept Property与Business Data Type存储关系图**

（2）Business Form

业务表单（Business Form）是一个业务概念的展现。Business Form除拥有Resource的所有属性外，还有属性为表单标题。为了达到内容与样式的分离，一个Business Form对应一个Form Style用来记录表单布局和样式。相对应与一个业务概念拥有多个属性，对应的一个业务表单也拥有多个表单字段（Form Field）。Form Field除主键外包含的属性为字段标明，字段名称，字段正则表达式，字段描述，字段缺省值，非法数据提示信息，字段是否可为空值。如图4-8所示，Business Form可对应0至1个Business Concept，Business Concept可对应0至1个Business Form，Form Field可对应0至1个Concept Property，Concept Property可对应0至1个Form Field。

同时一个表单还对应多个表单按钮（Function Button）。



**图4-8 Concept与Form的对应结构图**

（3）Business Function

Business Function除拥有Resource的所有属性外，还有属性为功能点的类型，功能点的代码脚本。

如图4-9所示，一个Business Function可对应多个Function Button。



**图4-9 Function与Button存储关系图**

（4）Business Logic

Business Logic除拥有Resource的所有属性外，还有属性为业务逻辑的代码描述。

如图4-10所示，一个Business Logic组织多个Business Function和多个Virtual Page。



**图4-10 Function与Logic、Page存储关系图**

Virtual Page除了主键外包含虚拟页面的URL以及虚拟页面的类型。一个Business Form或一个Business Page都可以对应多个虚拟页面。

（5）Business Page

Business Page除拥有Resource的所有属性外，还有属性为表示页面的内容代码。

如图4-11所示，同Business Form，一个Business Page对应一个Page Style来记录页面布局和样式，对应多个Function Button来表示表单上的按钮。同时一个Business Page可以有0至n个Business Form。



**图4-11 Page的组成结构关系**

（6）Business Menu

Business Menu除拥有Resource的所有属性外，还有属性为菜单的类型。

一个Business Menu可包含多个Menu Item。

Menu Item除主键外包含菜单项的名称，菜单项的标签名，菜单项的描述，菜单项是否启用。

如图4-12所示，每个Menu Item对应1个Virtual Page, 每个Virtual Page可对应0至n个Menu Item。



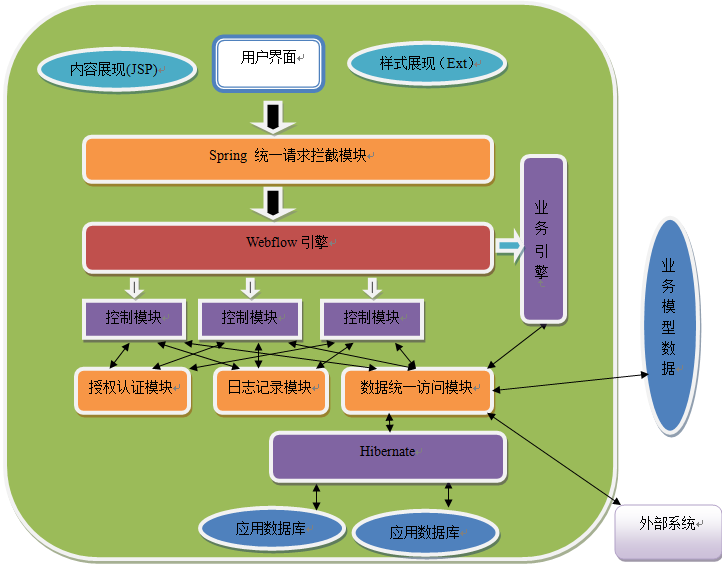
**图4-12 Menu的组成结构关系**

4.2 NGX平台基础框架设计

NGX应用平台包括管理与监视模块（NGX Admin & Monitor），业务代码生成引擎（NGX Business Engine）与平台基础设施（NGX Business Infrastructure）。应用平台的设计如下。

4.2.1 平台基础框架结构

应用平台采用传统的信息系统分层结构，如图4-13所示。

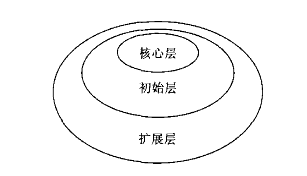


**图4-13 平台基础框架层次图**

用户界面将信息内容与显示样式分离，前台UI以JSP作为内容提供，以Ext-JS作为展现定义。当用户发出请求时，服务器由Spring统一请求拦截，请求由拦截器解析并传递给webflow引擎，webflow根据业务规则的定义按照顺序执行多个控制模块，并控制页面跳转。控制模块可能会用到基础设施中的授权认证模块、日志模块、数据统一访问模块。

而业务引擎作为一个特殊的控制模块也内置与平台之中。通过管理员监控平台的请求经过Spring统一请求拦截传递给控制业务组件模块生成的工作流，由工作流控制调用业务引擎来生成应用。

如图4-14所示，平台采用可扩展形式进行集成，分为核心层、初始层与扩展层。



**图4-14 平台扩展模式图**

核心层是平台基础设施模块，他是平台整个纵向架构的基础构件，是平台应用的底层服务。

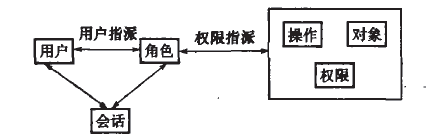
初始层是平台的管理监控模块与登入管理模块，它们使用平台的核心层提供了平台的原始功能。

扩展层是通过业务模型的解析生成的，它们也利用了核心层提供的服务，并根据业务模型定义提供具有业务功能的模块。

4.2.2授权模块设计

一个优秀的软件产品，除了具备先进的管理思想外，软件本身的柔性技术设计和安全架构也是必不可少的因素。权限管理模块作为企业信息系统中的重要模块，是系统安全运行的有力保证[27]。

作为平台基础设施的关键部分，认证授权模块采用基于角色的访问控制（RBAC）理论进行权限控制。核心RBAC定义了能构成一个RBAC系统的最小元素集合[28]，主要包括用户、角色、客体、操作、权限和会话6个基本元素，基本模型如图4-15所示。核心RBAC的权限被赋予角色，而不是用户。当一个角色被指定给一个用户，此用户就拥有了该角色所包含的权限，会话是用户与角色集合之间的映射集合。在核心RBAC中，所有的角色和用户是平级的，没有指定角色和用户的层次关系，也没有对操作对象进行具体的分级。



**图4-15 RBAC模型**

如图4-16、图4-17、图4-18所示，根据RBAC的模型，在应用中我们对每个Resource都定义了一系列权限Permission。这些Permission会在业务模型定义中与各种Role绑定。而在实际应用生成后，管理员会进行用户配置，将不同用户授权为不同的角色。从而可以判断用户会对Resource的访问权限。



**图4-16 Resource与Permission关系**



**图4-17 Role与Permission关系**



**图4-18 Role与User关系**

在实现上权限管理模块分为Authentication、Authority、Audit三个子模块。

Authentication负责用户的登入认证，即将User认证为Role的过程。

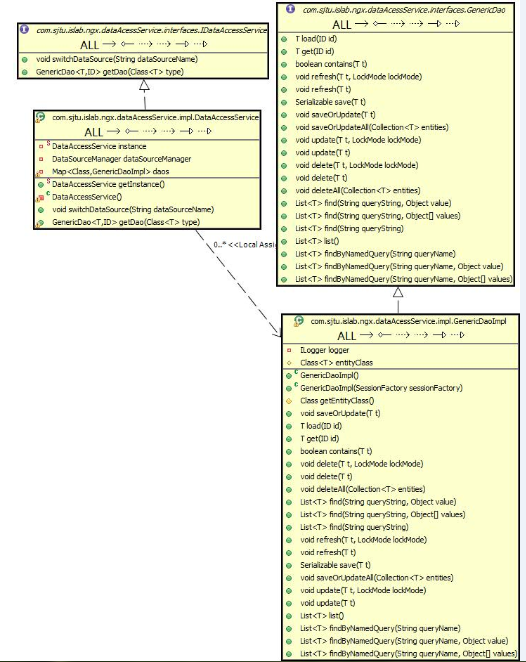
Authority负责用户根据角色授权，即管理员将User与多个Role绑定的过程。

Audit负责判断用户是否有操作权限，即判断User是否有Resource的访问权限的过程。

4.2.3统一数据访问模块设计

统一数据访问模块负责对所有数据的访问进行包装以实现系统对不同数据源的访问甚至外部系统的数据访问的透明化。统一数据访问使用一个专门的数据源管理器，进行数据源管理，在应用配置时，管理员为应用配置一系列数据源，而当系统中的模块需要访问数据时，只要指定数据源和操作数据类型，就可以不考虑数据的访问技术直接对数据进行增删改查。

在实现上，统一数据访问模块的接口及类结构图如图4-19所示。



**图4-19 统一数据访问模块类结构图**

该模块的主接口包含两个方法。

getDao()方法负责根据数据的java类型来自动生成一个操作数据的DAO（data access object）类。

switchDataSource（）负责转换数据源。这里的数据源由一个DataAccessManager统一管理，它根据应用的配置数据源在服务启动的时候自动配置各个数据源，并提供数据源的切换管理。这里的数据源可以是本地数据库、外部数据库以及外部系统提供的数据服务。

4.2.4统一日志管理模块设计

根据面向切面编程AOP的理念，应用对日志记录用统一模块管理。日志模块使用通用日志管理，应用平台中的所有信息都可以通过它记录。它通过配置文件可以选择将不同等级的信息输出到不同的控制台或者日志文件中。

实现上，日志模块主要提供接口如下：

debug(Class c, Object o)

info(Class c, Object o)

warm(Class c, Object o)

error(Class c, Object o)

fatal(Class c, Object o)

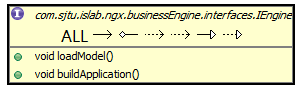
它以五个优先级便记录系统的执行过程。

4.3 业务引擎设计

业务引擎通过数据统一访问服务从业务模型数据库获取业务模模型，然后负责根据平台技术模板与业务模型的整合生成业务组件模块。

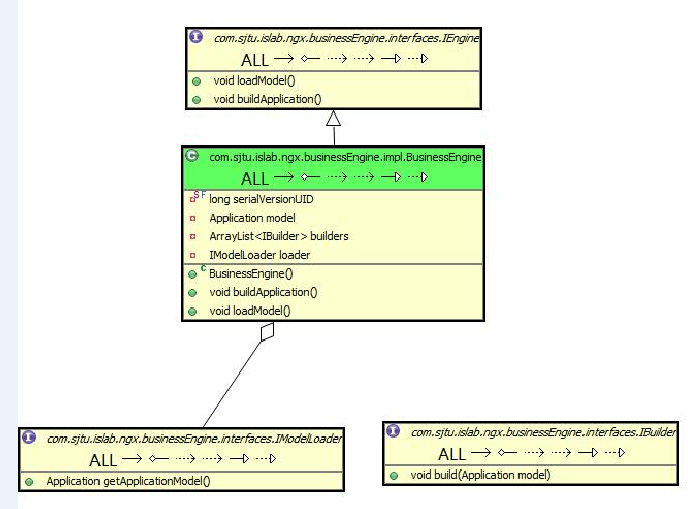
业务引起使用Facade设计模式[29]，将整个模块提取统一借口（IEngine）。

主接口IEngine接口如图4-20所示。其中loadModel()为从数据库中读取业务模型；buildApplication（）为根据读取的业务模型生成相应的应用模块。



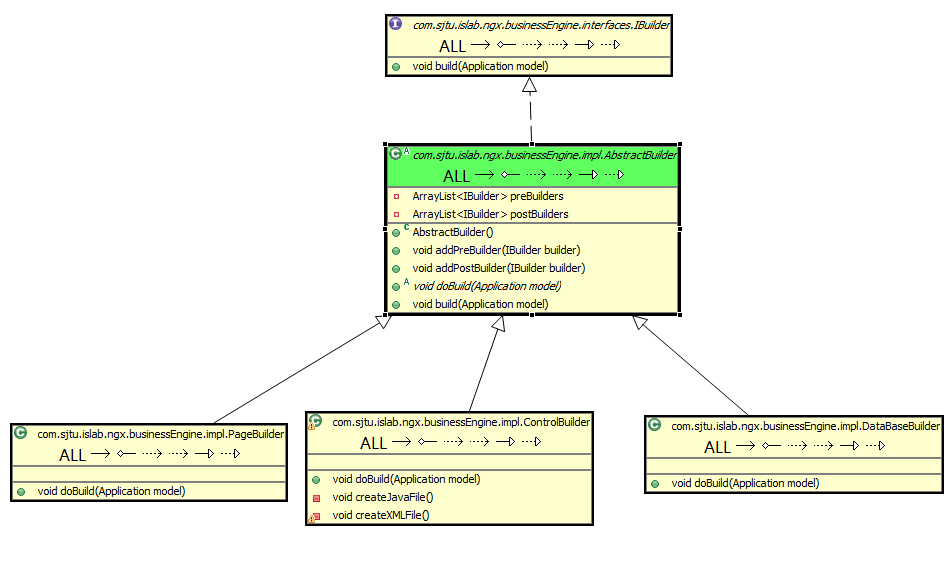
**图4-20模块的总调用接口**

其实现为BusinessEngine类。该类包含一个IModelLoader成员和多个Ibuilder成员。ImodelLoader复杂读取业务模型数据。每个Ibuilder负责根据业务模型生成相应方面的代码。结构如图4-21所示。



**图4-21 业务引擎基础结构**

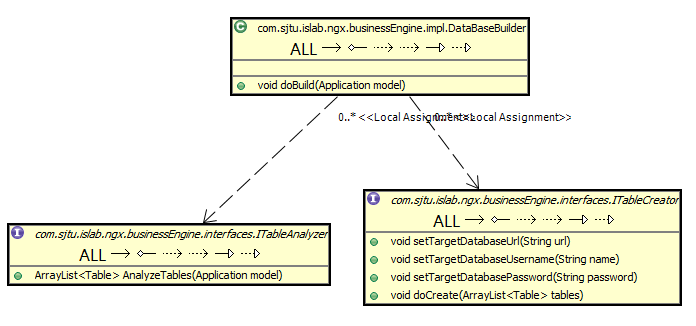
其中IModelLoader接口的getApplicationModel()函数为从数据库中读取业务模型并生成pojo对象；IBuilder接口的build()函数为根据生成的pojo对象生成一个层面上的代码模块。



**图4-22 Builder的组织结构**

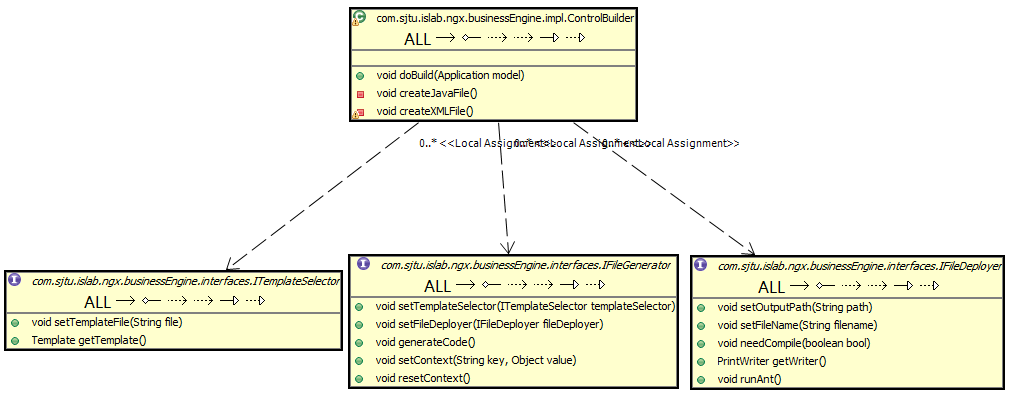
使用抽象类AbstractBuilder实现Decorator设计模式[29]。用addPreBuilder(IBuilder)与addPostBuilder(IBuilder)来注册一个IBuilder执行前后应该执行的其他Ibuilder，从而可以递归的进行整个项目模块的生成。其中所有Builder分为两类，在数据库中进行建表更改表操作的Builder与生成文件形式业务代码的Builder。

前者为DataBaseBuilder。它继承AbstractBuilder，实现通过分析业务模型（主要是Business Concept）来生成数据库表。其结构如图2-23所示。其中ITabel成员负责分析模型确定需要建立的数据库表；ITableCreator成员负责来将已经分析出来的数据库表写入数据库。



**图4-23 Database Builder设计结构**

后者为各类ControlBuilder，它们继承AbstractBuilder，实现通过分析业务模型（主要是Business Logic、 Business Function）来生成应用程序的Controller。这类Builder将代码生成的流程分为模板选择、代码生成、代码部署三个子模块。其结构如图4-24所示。其中，ITemplateSelector为模板选择控制器接口；IFileGenerator成员为代码生成控制器接口；IFileDeployer成员为代码部署控制器接口。



**图4-24 ControlBuilder的分解结构**

4.4 本章小结

本章说明了NGX平台的设计的基础要素，并分层定义了NGX平台的业务模型细化到数据库的存储方法，然后阐述了应用平台的分层结构设计与扩展集成方式。然后根据GOF设计模式并对其中关键模块的详细设计实现进行说明。

**第五章 测试与应用结果及分析**

5.1 验证对象MES系统描述

生产执行系统(manufacturing execution system，MES)处于企业资源计划(ERP)层和过程控(PCS)层之间。它除了作为连接企业管理层和生产控制层之间的信息与管理沟通桥梁外，还需要建立可以完成生产管理关键业务的功能，如生产调度、物料跟踪、生产统计等[30]。如果MES采用传统的软件开发方法，一旦开发完毕，就不能随生产因素的变化而及时调整。以上情况直接导致系统功能的环境适应性不强，自主研发的成本。收益也很难控制。所以，提高软件抽象层次和集成度，一直以来都是MES软件开发方法的热点和难点[31]。

因此结合流程工业环境苛刻，生产过程复杂等特点，针对传统MES存在的环境适应性低、研发和维护成本高和应用效果不理想等问题，使用NGX平台构建柔性MES可以验证NGX的正确性与易用性。在此，以构建MES的计划管理模块为例验证NGX平台。

5.1.1 MES的计划调度模块总体描述

车间MES系统主要包括计划调度，加工任务管理，过程管理，质量控制，人员管理，制造资源管理，工艺文档管理，接口集成管理和系统管理，系统的功能结构如图5-1所示：

**图5-1 MES功能结构图**

其中计划调度是MES系统的核心，车间生产计划管理及调度模块可以完成生产计划的分解，根据生产计划合理分配加工任务，平衡各台加工设备的作业，最大限度地提高各台设备的利用率，缩短工件加工时间。同时该系统还能根据监控系统的反馈信息及时修改作业计划，实现即时调度和闭环控制。计划调度模块与其它模块紧密地联系在一起，其他模块不仅要从该模块获得生产信息，同时，其他模块反馈的信息又会影响详细计划的制定和调整需要。

5.1.2计划调度功能描述

计划调度功能如下：

（1）通过与计划系统集成接口，接受车间生产作业计划；

（2）领料员至供应处领料，通过与物资系统的集成接口得到物料的清单；

（3）通过与CAPP集成接口，得到车间生产作业计划需要的工艺路线；

（4）计划调度员将工作分解到具体工人，在系统中完成工序级详细作业计划的制订；（计划调度主要凭借调度员的个人经验分派）

（5）可根据车间生产计划，以及工艺约束和生产能力约束等，以特定的生产目标，如资源利用率、加工周期等进行资源优化匹配，生成工序级详细作业计划，即对每个具体时期（月、周、日、小时等）内的生产任务做出详细计划，生成班组生产计划；

（6）向生产班组下发工序级作业计划，开“工艺过程卡”，“质量控制卡”和“工票”；零件加工数量超过三件开“首件检验卡”。

（7）对车间月度生产计划的增加、删除、修改等，并可以根据不同的查询条件对车间月度生产计划进行查询；

（8）对班组月度生产计划的修改、删除等，并可以根据不同的查询条件对班组月度生产计划进行查询；

5.1.3 计划调度业务流程描述

计划调度的业务流程分如图5-2所示：

（1）系统根据工艺文件产生工序计划

（2）计划调度员根据车间作业计划制定车间详细作业计划；

（3）生成班组作业计划；

（4）计划员把任务下发到各个班组。班组生产计划的生成以车间详细计划为基础，以工艺或生产能力为约束；

（5）和生产任务一起下发的有工票、工艺过程卡、质量控制卡、件数多于3件时加开首件检验卡。



**图5-2计划调度业务流程图**

5.1.4 计划调度用例描述

计划调度员的用例图如图5-3所示。计划调度员得到科研生产处下发的车间生产计划；对车间月度生产任务进行管理，新建车间月度生产任务的相关信息，并对其进行编辑车间月度生产任务、删除车间月度生产任务；执行查询车间生产任务可以查询车间月度生产任务的相关信息；根据车间生产计划，以及工艺约束和生产能力约束等，以特定的生产目标，如资源利用率、加工周期等进行资源优化匹配，制定工序级详细作业计划，即对每个具体时期（月、周、日、小时等）内的生产任务做出详细计划，生成班组生产计划；执行下发班组计划是将班组计划下发到各生产班组。



**图5-3计划调度员用例图**

5.1.5 计划调度数据流程描述

在计划调度过程中，使用到的数据字典如表5-1至表5-6所示。数据流程如图5-4所示。

**表5-1：车间作业计划数据字典**

|  |  |
| --- | --- |
| 数据流  系统名： 计划调度  条目名：车间作业计划 | |
| 来源： 科研生产处 | 去处： 计划调度 |
| 数据流结构：作业计划=（工作令+图号+名称+计划进度{投产+出产+生产日期}+使用车间） | |

**表5-2：工票数据字典**

|  |  |
| --- | --- |
| 数据流  系统名： 计划调度  条目名： 工票 | |
| 来源： 计划调度 | 去处：班组长 |
| 数据流结构：工票=（工作令号+产品名称+所属型号+材料牌号+车间+产品代号+所属系统+材料规格+工种+工序+总加工件数+姓名+加工件数+工时） | |

**表5-3：工艺过程卡数据字典**

|  |  |
| --- | --- |
| 数据流  系统名： 计划调度  条目名：工艺过程卡 | |
| 来源：计划调度 | 去处： 班组长 |
| 数据流结构：工艺过程卡=（车间+小组+工作令编号+型号+零部件名称+材料牌号+毛坯规格+批次+图号+生产履历卡号+工序名称+定额工时+完工数量+合格） | |

**表5-4：质量控制卡数据字典**

|  |  |
| --- | --- |
| 数据流  系统名：计划调度  条目名：质量控制卡 | |
| 来源：计划调度 | 去处：班组长 |
| 数据流结构：质量控制卡=（工序号+工序名+控制内容+控制点+控制方法+实测记录+操作者+检验员） | |

**表5-5：首件检验卡数据字典**

|  |  |
| --- | --- |
| 数据流  系统名： 计划调度  条目名： 首件检验卡 | |
| 来源： 计划调度 | 去处：班组长 |
| 数据流结构：首件检验卡=（工作令号+产品型号+工艺过程卡号+工序+图纸尺寸+首件编号+实际测量尺寸{自检+互检+检验员测量尺寸}） | |



**图5-4计划调度数据流程图**

5.2 验证过程及结果

5.2.1 计划调度业务建模

根据NGX建模规则，建立一个名为MES的Application，并使其包含一个计划调度的Business Component。

(1)建立Application

根据NGX建模规则，建立一个名为MES的Application。

Application的属性如下：

ID：1;

NAME：MES;

SEQ：20100603A;

CTIME：2010.06.03;

UTIME: 2010.06.03;

AUTHOR: SJTU;

DESCRIPTION: Business model for MES;

建立一个配置信息记录MES使用的数据库信息并与Application 绑定。

(2) 建立Business Component

建立一个Business Component内容如下：

ID：1;

NAME：Planning and Scheduling;

CTIME：2010.06.03;

UTIME: 2010.06.03;

AUTHOR: SJTU;

DESCRIPTION: Planning and Scheduling component model for MES;

将ID为1的Business Component附属到ID为1的Application

(3) 建立业务概念（Business Concept）及对应的表单（Form）

该Business Component应该包含车间作业计划、工艺文件、物料清单、工票、工艺过程卡、首件检查卡、质量控制卡等Business Concept。

以车间工作计划为例建立业务概念及对应表单。

建立一个Resource属性如下：

ID：1;

NAME：workshop plan;

LABEL：workshopPlan;

PUBLIC：true;

TYPE: concept;

DESCRIPTION: Business concept for workshop plan;

然后将此resource绑定读写两个特权。

建立一个继承的Business Concept属性如下：

TYPE：object;

建立为车间计划概念建立工作令、图号、名称、投产日期、出产日期、生产日期、使用车间PROPERTY

ID：1;

NAME：work serial number;

LABEL：workSerialNumber;

TYPE: String;

ID：2;

NAME：picture number;

LABEL：pictureNumber;

TYPE: String;

……

建立一个Form属性如下：

TITLE：车间计划

建立对应的Field：

ID：1;

NAME：work serial number;

LABEL：workSerialNumber;

NULLABLE：false；

ID：2;

NAME：picture number;

LABEL：pictureNumber;

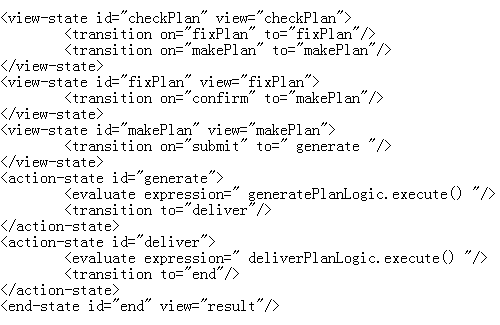
NULLABLE：false

……

(4) 建立功能点（Business Function）

该Business Component应该包含计划调度员获得车间计划、修改车间计划、制定详细计划以及获得班组查看班组计划，查看工票，查看工艺过程卡，查看首件检查卡，查看质量控制卡这些Business Function。

以制定详细计划为例。首先建立查看车间计划、修改车间计划、填写计划详细信息3个页面并分别与相应表单绑定。相应的页面与表单绑定对应的按钮。结合页面与按钮的定义功能点流程定义如图5-5：



**图5-5功能点流程定义**

(5) 建立Business Logic

制定详细计划的功能点包含生成班组计划与下放计划两个Business Logic。

在此将生成班组计划的逻辑简化为自动通过车间计划的工作令生成工票的所属系统。工作令的前三位即为所属系统名称。因此建立生成班组计划逻辑如下：

建立一个Resource属性如下：

ID：16;

NAME：generate plan logic;

LABEL：generatePlan Logic;

PUBLIC：true;

TYPE: logic;

DESCRIPTION: generate plan logic;

绑定一个Business Logic属性如下：

TYPE: generation;

CODE: ‘String serial = (String)flowScope.get(\"work serial \");\n

flowScope.put(\"system name", serial.substring(0,3));\n’;

(6) 建立Menu

建立计划调度员与班组长对应的两个Menu，然后每个菜单项绑定相应的页面。

(7) 建立Role

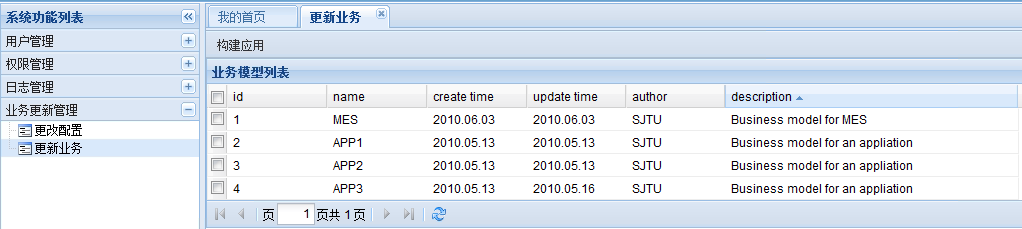
计划调度员与班组长两个role，并绑定相应的权限和菜单。

5.2.2 使用NGX平台生成业务功能并进行测试

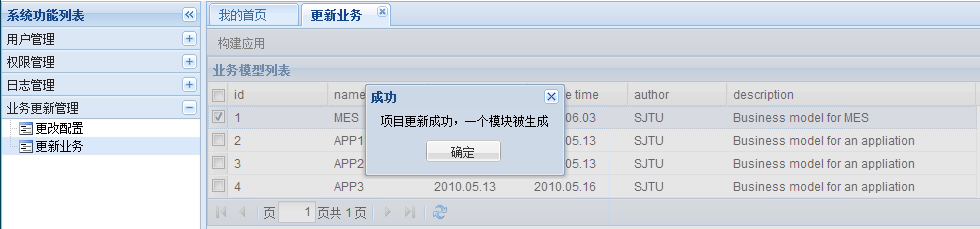
如图5-6所示，使用管理员身份登入NGX系统，在配置好模型数据库之后选择业务更新管理中的更新业务，如图5-7所示，获得业务模型列表。如图5-8所示，选择名称为MES的Application并进行构建，平台将导入MES Application的模型并使用平台自动构建出包含有计划调度模块功能的MES系统。



**图5-6 管理员身份登入**

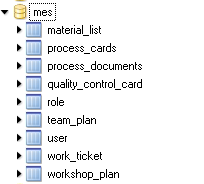


**图5-7 业务模型列表**



**图5-8 项目更新成功**

查看配置指定的应用数据库，如图5-9所示，数据库表已根据Business Concept自动生成。由于只测试计划调度功能,在此需要人工插入一个车间工作计划，一个材料清单，一个工艺文件。

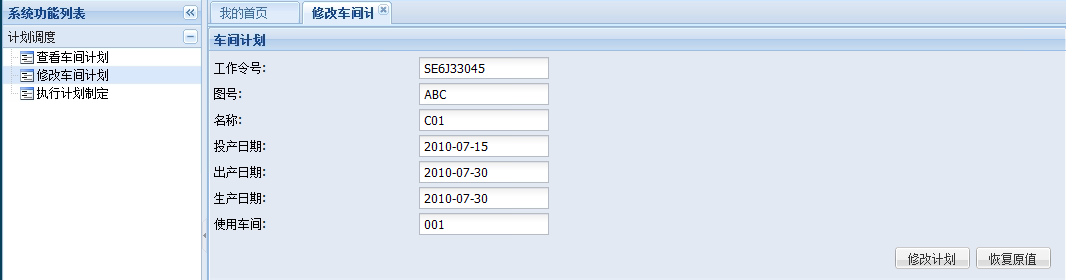


**图5-9 数据表生成**

通过管理员配置监控模块配置将ID为USER1的用户和ID为USER2的用户分别制定为计划调度员和班组长。

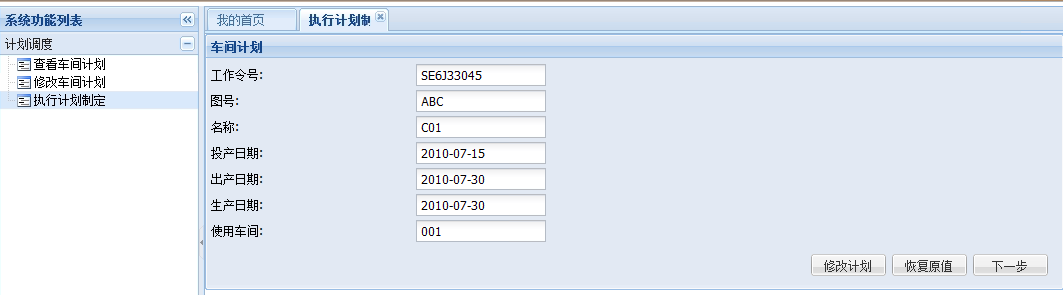
使用USER1身份登入系统，可见系统自动根据身份显示新构建的菜单。

点击计划调度菜单中的查看车间计划，如图5-10所示，可以查看数据库刚刚插入的车间工作计划数据。点击修改车间计划可以对车间工作计划进行修改。



**图5-10 车间工作计划查看与修改**

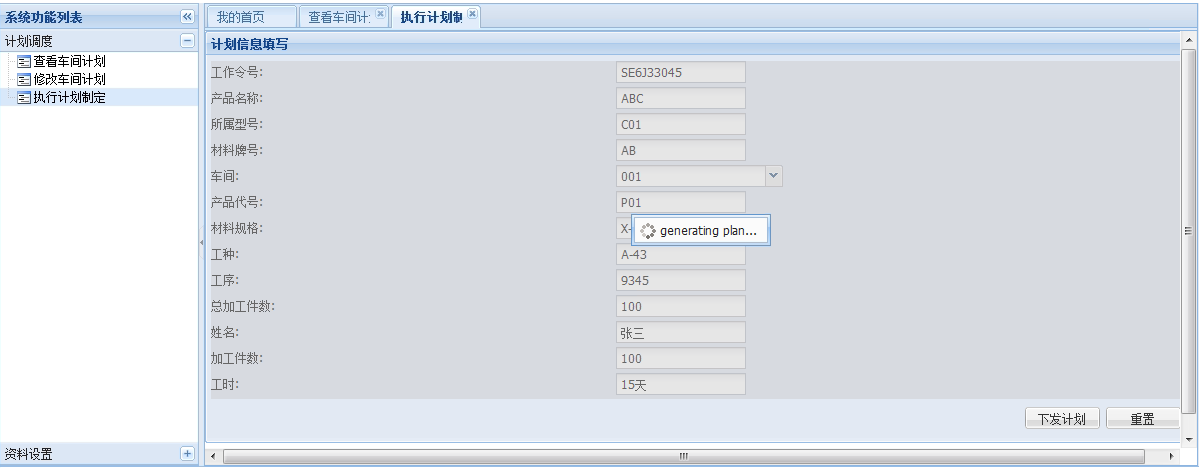
再选择执行计划制定菜单项，可以测试制定详细计划的流程。如图5-11所示，根据功能点的业务建模，首先计划调度员查看车间计划，并可以进行修改。然后如图5-12所示，计划调度员根据修改后的车间计划进行计划制定信息输入。最后如图5-13所示，系统执行模型定义的生成计划与下发计划的业务逻辑过程，然后显示如图5-14所示的计划调度结果。



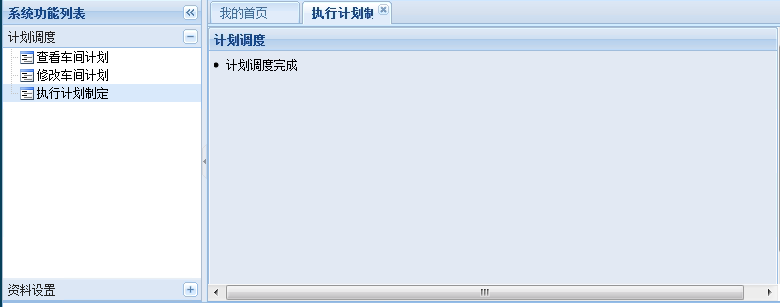
**图5-11 查看车间计划**



**图5-12 输入计划调度信息**



**图5-13 执行计划生成逻辑**



**图5-14 计划调度完成**

使用USER2身份登入系统，可见系统自动根据身份显示新构建的班组长菜单。点击计划调度菜单中的查看工票，可以看到刚才计划调度人员制定的计划信息，并且如图5-15所示，可以查看系统生成的工票、工艺过程卡、质量控制卡。并且可以看到，自动生成的所属系统确实是工作令号的前三位，与定义的生成逻辑一致。

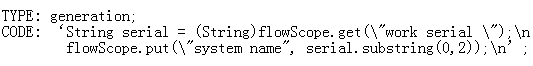


**图5-15 班组长查看工票**

综上可见MES的计划调度模块的生成与模型定义一致。

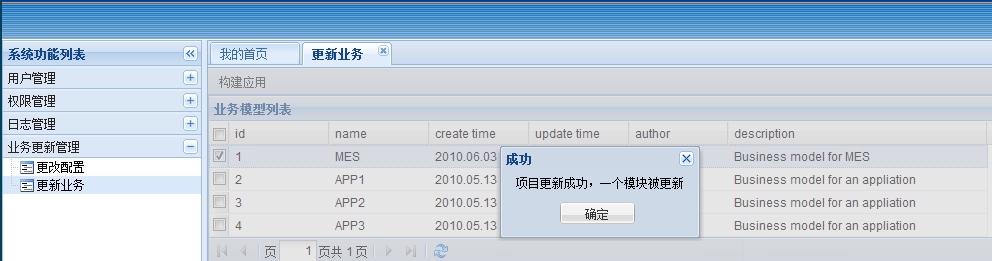
5.2.3 修改业务模型重新构建

修改业务模型：修改生成班组计划的业务逻辑。原逻辑为工作令的前三位即为所属系统名称，先改为工作令的前两位即为所属系统名称。因此修改的生成班组计划逻辑如下：



**图5-16 修改后逻辑代码**

然后如图5-17所示，使用管理员身份登入NGX平台重新构建应用，生成具有新的计划调度业务规则的MES系统。使用USER1身份登入系统，可见系统自动根据身份显示新构建的菜单。



**图5-17 项目更新**

以计划调度员身份重新制定班组计划。然后如图5-18所示，以班组长身份查看工票。此时可以看到工票的所属系统已经改为工作令的前两位。说明整个应用模块可以根据业务模型的变化而快速更新。



**图5-18 班组长查看工票**

5.3 结果分析

NGX平台提供的业务定义要素可以正确描述MES这样一个具有代表性的企业管理应用所涉及的业务规则，说明了NGX平台的业务模型构件定义是正确的。平台根据业务模型的定义，生成了符合业务规则的功能模块，说明了NGX平台可以正确地将PIM模型解析并自动生成包含业务定义的J2EE体系的PSM模型，并转化为可执行组件代码，并转化为可执行组件代码，业务引擎还将自动部署，结合平台的各个层面上的基础构建快速形成一个有效的企业生产管理应用。

在测试案例中，制造执行系统MES的功能的开发主要成本只集中在根据业务规则构建平台无关模型PIM的过程中。只要拥有高效的可视化建模工具，整个企业管理系统的开发过程即可转化成为行业专家用他们自己的知识系统使用特定的符号构建一个模型，并使之融入高层的程序模型中，然后直接生成企业信息系统。NGX平台利用MDA的核心理念成功实现了业务定义与软件开发的分离，解决了业务需求过于复杂带来的软件开发的问题。同时它简化了整个软件开发周期，避免了在开发、部署、测试过程中人为原因的错误的发生。这体现了模型驱动开发的优势性。

NGX平台是通过业务规则注入的形式构建应用。这使得它的产生的应用无需重新部署，系统更容易维护。并且由于是业务注入的形式，它支持重新导入业务模型并重新构建应用。重新构建的整个过程都可以在系统管理员掌控之中，重新构建的自动化过程时间极短。因此解决软件需求变更的代价只是修改业务模型的成本。这不但解决了需求变更过于频繁对于企业管理软件的冲击，同时可以通过更改需求重新构建应用的迭代促使软件实体与实际业务的之间差异的快速收敛。从而NGX平台拥有较强的业务可配置性。

NGX平台提供了基于MDA的软件开发过程，他与一般软件过程比较如表5-6所示。这种开发方式具有一系列优势，如表5-7所示。

**表5-6：NGX平台化软件开发过程与其他开发过程比较**

|  |  |  |
| --- | --- | --- |
|  | 一般软件过程 | NGX平台 |
| 需求分析 | 由软件人员向行业专家调研 | 由行业专家直接可视化建模 |
| 设计 | 软件开发人员依据需求对软件进行设计，时间长，设计风格因人而异 | 平台提供统一技术套路与软件设计方案 |
| 编码 | 人工编码，时间长，成本多，易出错，代码风格因人而异 | 系统自动生成，时间短，成本低，避免人为失误 |
| 测试 | 必须对代码进行单元测试 | 可省略自动生成代码的单元测试，将测试重点放在功能测试上 |
| 部署 | 人工部署，时间长，复杂度大，需要部署整个软件项目 | 只需要部署平台，平台自动部署业务模块 |
| 维护 | 维护人员需要对系统非常了解，修改过程涉及底层技术 | IT管理人员通过平台的监控和管理功能进行业务功能模块的维护 |

**表5-7：NGX平台化软件开发方式与其他开发方式比较**

|  |  |  |
| --- | --- | --- |
|  | 一般软件过程 | NGX平台 |
| 开发效率 | 开发时间长，风险大 | 缩短了设计、编码、测试及部署的过程，降低开发风险 |
| 专注核心 | 在关注行业规则的同时，要关注架构设计、编码过程，存在重复劳动 | 行业专家可以只关注行业规则就能改变系统功能，软件开发人员可以专注于创造性工作 |
| 人员因素 | 在整个项目过程中，人员变动对项目的开发及维护影响很大；人为的失误在开发过程中也容易造成不良影响 | 项目的开发及维护主要集中在可视化建模过程，人员调动影响相对较小；系统自动生成部署代码避免了人为失误 |
| 管理要求 | 开发过程复杂，对项目管理人员能力要求很高 | 简化了项目过程，减轻了管理上的压力 |
| 软件与需求的一致性 | 行业专家、系统分析员、软件设计人员、编码人员之间的沟通出现问题都容易导致软件与需求一致性出现问题 | 由行业专家对系统功能进行主导，消除理解偏差 |
| 需求变更应对 | 设计期间未考虑到的需求变更容易导致项目的返工 | 缩短了变更过程，可以重新调整业务模型进行项目的重新生成，方便了软件与需求逐步逼近的迭代过程 |

NGX平台还提供了一种独特的业务注入的集成方式以及业务模型共享的运营模式。如表5-8所分析，NGX平台的集成方式具有一系列特点，同时也存在一些需要改进的地方。如表5-9所分析，NGX的运营模式更接近与平台即服务（PAAS）的模式，他的多元性、可配置性与易维护性使它拥有很大的潜力。

**表5-8：NGX平台软件集成方式与其他集成方式比较**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 企业应用集成  （EAI） | 面向服务体系架构  （SOA ） | NGX平台 |
| 异构性 | 异构性差 | 异构性好，不同的异构系统可以通过统一标准接口集成 | 异构性差，对于外部的异构系统，通过数据服务的包装来实现 |
| 分割粒度 | 因企业系统而异 | 取决与服务粒度 | 系统内部以业务组件的维度进行分割，外部系统的分割粒度取决于外部系统 |
| 集成开销 | 取决于系统接口的设计 | 服务调用存在一定数据封装与解析开销 | 系统内部，业务组件是同构的并通过数据总线交互，避免数据数据封装与解析开销，也避免了网络传输的开销 |
| 复用性 | 系统复用性差 | 服务可以很好的复用 | 业务组件库里面的业务组件支持共享与复用 |
| 可扩展性 | 因系统而异，需通过系统提供的接口，进行扩展 | 可以通过服务的扩展对系统进行扩展 | 通过业务组件扩展进行系统功能扩展，组件的更新通过依赖连锁更新机制进行更新 |

**表5-9：NGX平台软件运营模式与其他运营模式比较**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 软件即服务  （SAAS） | 平台即服务  （PAAS） | NGX平台 |
| 多元性 | 提供的服务由SAAS提供商定义，对不同用户的服务个性化差 | 提供商提供开发平台，用户开发自己的个性化应用 | 用户依据自己的需求进行可视化建模，可以生成自己的个性化应用 |
| 数据管理 | 应用与数据都由SAAS提供商管理，设计对提供商的信任问题 | PAAS并没有面向最终用户，不涉及数据管理 | 可配置数据来源，既可使用可信任本地数据源也可使用托管数据源 |
| 维护成本 | 由提供商统一维护，客户无需考虑 | 平台提供商对平台进行维护，平台使用人员对自己的应用进行维护 | 平台开发人员对平台底层进行维护，IT管理人员通过平台的监控和管理功能进行业务功能模块的维护 |

5.4 本章小结

本章以MES的计划调度模块中的功能作为测试案例测试平台的可行性及柔性。首先分析了MES系统的基础特性，然后进行NGX平台规则的业务建模然后通过NGX的配置和导入生成MES的计划调度模块，通过功能测试证明了NGX生成的应用与业务模型定义一致。最后通过更改业务模型进行再次更新应用并验证更新证明了NGX平台的业务可配置性。并结合测试结果，将平台与其他方式进行比较，分析了平台的优缺点。

**第六章 结论**

6.1 工作总结

本课题通过分析企业管理信息系统的发展趋势和软件开发面临的挑战，提出了当前企业管理系统软件的开发方式可能存在的问题。为解决这些问题，本课题结合分析模型驱动架构与平台化软件开发的核心优势，讨论了应用模型驱动架构与平台化软件开发的优势解决于企业管理软件的开发过程所存在的问题的有效性。然后提出一个基于模型驱动开发的企业管理平台NGX的设计方案。

本文结合模型驱动架构与模型驱动开发的核心理念，总结并设计出NGX平台的基本形式。将NGX平台分为两大部分，包括一个可视化的业务建模工具（NGX Studio），以及一个自动部署集成应用平台（NGX Application）。然后通过CIM、PIM、PSM以及最终代码的转换关系转换，总结出平台各个阶段的模型存储转换方式：业务开发人员通过使用业务过程建模的可视化业务建模工具对业务进行建模或调整业务模型，然后由建模工具解析并自动转换为J2EE体系的平台相关模型（PSM）存储于构建库中。然后通过平台提供的业务引擎解析PSM并生成为可执行组件代码，业务引擎还将自动部署，结合平台的各个层面上的基础构建快速形成一个有效的企业生产管理应用。

然后通过解析、分解一般企业信息系统存在的业务要素提将给出企业管理系统模型驱动开发的基本元素，通过分析与总结，确定了业务模型的详细内容的分解，并根据MOF理论对这些业务模型进行分层，然后提出一个分层结构的业务模型构件数据库的设计方案和存储结构。

选定平台应用集成段的技术方案，并提出基于Java EE轻型框架的平台框架。分析了平台的分层结构以及扩展集成方式，提出了平台应有的核心模块的作用以及它们的详细设计以及实现方法。其中包括基于RBAC模型的权限控制、管理多数据源以及外部系统数据的数据统一访问服务、全局日志管理以及执行业务模块自动生成部署的业务引擎。

最后，本文以制造执行系统 (Manufacturing Execution System, MES）的计划调度模块为主要分析对象。利用NGX的建模规则对计划调度模块建模，并应有与NGX平台，测试平台的业务模块生成的有效性与正确性。然后通过业务模型的快速修改与平台的应用更新验证平台业的柔性以及高效可维护性。通过分析实验结果，与已存在的软件开发方式进行对比，并推测其后续可扩展性。

6.2 平台研究工作展望

在企业管理软件开发方面，NGX平台体现了模型驱动开发在软件开发方面的优势。一方面，它可以简化企业管理系统的开发方式，通过具体技术的透明化，使行业专家也可以更投入软件开发。另一方面，它提供了通用的技术路线，提高了企业管理系统的模块复用度，使技术专家可以避免重复工作，投入更有创造性的工作。

随着企业信息系统的不断普及以及系统的复杂性不断提高，NGX平台将可以在企业管理软件开发方面提供很大的帮助。特别是当NGX平台升级为一个业务组件的共享平台之后，企业信息系统的开发将可以借鉴已存在的系统的组件进行开发，在大大减小开发成本的同时，还可以保证通用组件在系统中高度个性化。

下一的研究内容主要包括：

（1）作为MDA的平台，NGX平台需要设计出一个高效方便的可视化建模工具才能形成完整的开发平台。该平台只要拥有高效的可视化建模工具，整个企业管理系统的开发过程即可转化成为行业专家用他们自己的知识系统使用特定的符号构建一个模型，并使之融入高层的程序模型中，然后直接生成企业信息系统。

（2）工作流和报表是信息系统不可或缺的关键部分，因此NGX平台后续可以加入工作流引擎模块以及报表引擎模块，这样它可以适应更复杂的系统需求。

（3）NGX平台的业务模型数据库可改进成公用的开放式组件库。近几年来随着面向对象技术的成熟，软件复用已经融入到软件工程研究的主流，被视为是使软件开发真正走上工程化和产业化道路的希望。人们在实践中逐渐认识到软件复用是一个涉及到技术、方法学、管理和文化等多个方面的复杂问题。软构件库管理系统是使软件复用实践真正系统化、工程化的、支持基于构件的软件开发的基础设施。软构件库是开发可重用软构件和基于软构件开发两个生命周期的联系中介[32]。用户可以将本地建模的业务组件发布至互联网上共享，也可以通过Internet连接到位于互联网上开放式业务组件库进行查询，选择适合自己业务的业务组件，直接在该业务组件的基础上按照自己的业务需求进行业务模型的调整，并构建自己的应用。这样，NGX平台可以作为一个共享平台，促进了企业管理系统的组件开发与共享，提高了业务组件的复用性。

参考文献

[1] [程祁慧](http://s.wanfangdata.com.cn/paper.aspx?f=detail&q=%e4%bd%9c%e8%80%85%3a%22%e7%a8%8b%e7%a5%81%e6%85%a7%22++DBID%3aWF_QK). 信息化对企业核心竞争力的影响研究[D]. 安徽: 安徽财经大学经济与金融学院,2005:12.

[2] 甘明鑫, 企业体系结构的建模框架研究[J], 中国管理信息化, 2009,12(15)

[3]软件开发项目需求变更管理及应对之道研究[EB/OL]. [2009-12-3]. http://www.mypm.net/blog/user4/ccx1104/archives/2009/42141.html.

[4] 李春华，直面软件开发问题[P]. 北京：清华同方开发中心, 2006-02-18.

[5] 崔晶炜. 平台化：软件发展趋势所在[J/OL]. [2004-02-12]. 中国计算机用户. http://industry.ccidnet.com/art/10/20040212/89828\_1.html.

[6] Omg.Mda Specification[EB/OL]. [2010-05-21].[www.omg.org/mda/specs.htm](http://www.omg.org/mda/specs.htm) .

[7] 冯瑞涛, MDA(模型驱动架构).[2009-11-24]. http://terryfeng.javaeye.com/blog/524351.

[8] France, R. and Rumpe, B. 2007. Model Driven Development of complex software: A research roadmap. In Future of Software Engineering[C], IEEE.

[9] OMG’s MDA, Architecture[EB/OL]. [2010-05-21]. <http://www.omg.org/mda>

[10] Selo Sulistyo, Andreas Prinz. Recursive Modeling for Completed Code Generation[C], ACM International Conference Proceeding Series; Vol.379,2009:10-11.

[11] OMG.MDA guide version 1.0.1 [EB/OL] . [2010-05-21] .http//:www.omg.org.

[12] Perron P J．J2EE构建企业系统专家级解决方案[M]．北京：清华大学出版社，2002:9-11．

[13]使用模型驱动开发的十五个理由[EB/OL]. [2010-01-04].http://news.csdn.net/a/20100104/

216251.html.

[14]Model-centric software development (MDA/MDD) [EB/OL]. [2010-05-21]. http://www.

interactive-objects.com/en/model-driven-engineering.html.

[15]Enterprise Application Development [EB/OL]. [2010-03-16]. http://www.compuware.com/

resources/Uniface/uniface9\_4factsheet.pdf.

[16] androMda Reference doc[EB/OL]. [2010-02-17]. http://www.andromda.org.

[17]Eclipse Modeling Framework Project(EMF) [EB/OL]. [2009-04-08]. http://www.eclipse.org/

modeling/emf/.

[18]鹿旭东,万建成.元模型支持下的模型转换[J].计算机工程与应用, 2005,41(36).

[19]王学斌, 王怀民, 吴泉源.一种模型转换的编制框架[J].软件学报,2006,17(6) .

[20]夏雷, 欧阳松中.关联从模型到模型的转换[J].计算机工程与设计,2006,27(16).

[21]吴春水,唐胜中.从到的模型转换[J].计算机工程,2004,30(3) .

[22]颜玉兰,何克清,刘进.一种基于有限状态机的模型转换方法[J].计算机工程, 2006 ,32(1).

[23]段雷, 万建成.基于的应用程序开发方法[J].计算机科学,2004,31(9).

[24]汤最, 吴朝晖.一个利用模型驱动体系结构技术的分布式系统实现[J].计算机工程与应用,2003,39(33).

[25]文必龙, 王志宝, 王琳琳, 冯文征. 基于模型驱动的软件集成平台研究与实现[J]. [计算机工程与设计](http://acad.cnki.net/kns55/oldNavi/Bridge.aspx?LinkType=BaseLink&DBCode=cjfd&TableName=cjfdbaseinfo&Field=BaseID&Value=SJSJ&NaviLink=%e8%ae%a1%e7%ae%97%e6%9c%ba%e5%b7%a5%e7%a8%8b%e4%b8%8e%e8%ae%be%e8%ae%a1" \t "_blank). 2007(23).

[26]OMG. MOF 1.4 specification[EB/OL]. [2002-01-11]. http://www.omg.org/docs/formal/

02-04-03.pdf.

[27] [庞春江](http://s.g.wanfangdata.com.cn/paper.aspx?f=detail&q=%e4%bd%9c%e8%80%85%3a%22%e5%ba%9e%e6%98%a5%e6%b1%9f%22++DBID%3aWF_QK), [庞会静](http://s.g.wanfangdata.com.cn/paper.aspx?f=detail&q=%e4%bd%9c%e8%80%85%3a%22%e5%ba%9e%e4%bc%9a%e9%9d%99%22++DBID%3aWF_QK). RBAC模型的改进及其在电力ERP权限管理中的应用[J].电力系统自动化.[2008,32(13)](http://c.g.wanfangdata.com.cn/periodical/dlxtzdh/2008-13.aspx).

[28]SANDHU R, BHAMIDIPATI V, COYNE E, et．The ARBAC97 model for role-based administration of roles：preliminary description and outline[C]. Proceedings of the Second ACM Workshop on Role-based Access Control．1997：41-50．

[29] [Erich Gamma](http://en.wikipedia.org/wiki/Erich_Gamma), [Richard Helm](http://en.wikipedia.org/wiki/Richard_Helm), [Ralph Johnson](http://en.wikipedia.org/wiki/Ralph_Johnson), [John Vlissides](http://en.wikipedia.org/wiki/John_Vlissides). Design Patterns: Elements of Reusable Object-Oriented Software[S]. 1994.

[30]Li D F,Jiang B H and Wang H A. Design and analysis of refinery MES architecture[J]. Modern ChemicalIndustry,2004,24(2):48-51.

[31]董天翔,荣冈. 流程工业生产执行系统的软件基础设施[J]. 计算机与应用化学. 2010,27(1).

[32]姜海涛. 基于模型驱动的构件库系统的研究与设计[J]. [电脑知识与技术](http://acad.cnki.net/kns55/oldNavi/Bridge.aspx?LinkType=BaseLink&DBCode=cjfd&TableName=cjfdbaseinfo&Field=BaseID&Value=DNZS&NaviLink=%e7%94%b5%e8%84%91%e7%9f%a5%e8%af%86%e4%b8%8e%e6%8a%80%e6%9c%af" \t "_blank). 2009(20).

谢辞

本论文的完成，得益于导师蔡鸿明老师从选题的确定、论文资料的收集、论文框架的确定、开题报告准备及论文初稿与定稿中对字句的斟酌倾注的大量心血。蔡老师在毕业设计项目中给我创造了非常好的研究条件和设计环境。在整个项目过程中，蔡老师对项目的方向和进度给予了极大的关注，并对项目中遇到的问题给予很大的帮助。在此对尊敬的蔡鸿明老师表示真挚的感谢，感谢您在整个毕业设计的过程中给予的极大帮助。

本次项目的完成还得益于项目的指导老师吉锋老师在百忙之中来到学校从项目的思路、项目的定位、项目的技术路线、项目的架构到项目的细节设计实现给予极其关键的指导和帮助。感谢吉老师在毕业设计项目中传授的知识，给予的启迪和极大的帮助。