**基于模型驱动开发平台移动代码库与**

**集成测试框架设计实现**

摘要

近年来，软件平台化成为热门话题，通过这种方法若干组件分别由多个软件开发商编写，再组成一个复杂的应用软件,这样可以提高软件的复用性和生产率。因为它使得供应商、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。在软件平台化的基础上，业界与学者们提出业务基础软件平台，也即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。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。企业可以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。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。业务基础平台的研究主要集中在开发者平台到用户使用平台的转换的研究，即主要是软件平台的开发时功能扩展，而运行时动态扩展的研究仍然处于初步阶段。由于因特网的迅猛发展，软件的开发和使用出现了动态扩展的趋势，即用户通过从网上下载和执行移动代码来扩充主机功能，如浏览器的一个插件程序，新的打印驱动程序或一个灵活的实用程序。目前研究者对在移动代码的研究主要集中在移动代码的传输和执行安全的研究上，使得移动代码的安全保障已经达到比较成熟的程度。将移动代码模式的动态扩展特点与业务基础平台的高集成性结合可以使业务基础平台实现运行时的动态功能扩展、动态功能调用绑定。在业务基础平台特别是以模型驱动开发架构设计的业务基础平台上设计出可动态扩展的逻辑代码库。实现业务逻辑代码在业务基础平台上运用web2.0富客户端技术实现逻辑代码的预定义开发，动态扩展集成，动态绑定以及运行时测试。利用业务基础平台的集成扩展实现出移动代码的动态扩展管理，实现业务逻辑的动态扩展与动态验证。这将在很大程度提高供应商及中间商在业务基础平台中的作用，同时大大提高业务基础平台的可扩展性与可配置性。

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。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。

然而随着企业的信息化不断的提高，企业信息系统的越来越庞大，越来越复杂。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。目前有两个趋势面向于解决企业信息系统的需求不断复杂化，需求变化频率有所上升给软件开发带来的问题。

文章总结并设计出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。

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，文章以制造执行系统 (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，通过NGX平台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。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。

关键词：软件开发，企业管理软件，模型驱动架构，模型驱动开发，可配置平台，软件复用，代码自动生成

**DESIGN AND ANALYSIS OF MDA-BASED DEVELOPMENT PLATFORM FOR ENTERPRISE MANAGEMENT**

**ABSTRACT**

With the continuous development of information technology and the accelerating spreading pace of information technology in industrial production, the existence of information systems, which help to perform good production management and improve production efficiency in the enterprise, has become major competitiveness of enterprise. Nowadays, the enterprise information management has become an enterprise standard and a significant way to determine the future of enterprise. With experience accumulated, the development of enterprise information obtained certain results. As a result, more and more enterprises have realized the importance of information technology in business development. Therefore, the improvement of information industry has obviously improved other traditional industries. Information technology can speed up the upgrading of enterprise products and enhance its market competitiveness. An enterprise can get lots of benefit through the promotion of computer-aided design, the CAM technology and the decision support systems. Mainly, information system can promote the production technology and management information, which means improvement of the enterprise overall quality.

However, as the information technology business continued to improve, enterprise information system is becoming much larger and much more complex. Due to the constantly change of enterprise information management caused by changing business rules and business management needs, the traditional software development methods can no longer be as effective as it used to be. The difficulty in analysis, development, testing, deployment and maintenance information systems has been rose to a relatively high level, which leads to serious challenge of software development. Therefore, how to perform quickly and effectively development of enterprise applications has become a focus of the industry. There are two main solutions of the subject mentioned.

On one hand, the software platform has become a hot topic in recent years. Brokers and users become more active due to the allowance of suppliers, intermediaries, users all over the world to share their software service. So the platform technology helps to realize the ecological and dynamical feature of the whole software industry. Business infrastructure software platform is a technological innovation which produces one more level for software platform and separates application business logic from development of technologies. As a result, application software developers can focus only on business process application software without having to concern about its technical implementation. So it becomes possible that management and operational staff involved in the development of enterprise systems. The development of enterprise information systems can be simplified to the combination of customized modules. Compared to re-development of a system, the platform solution shortens the development cycle and reduces development costs.

On the other hand, model-driven development is an innovative software development method. It is a platform-independent and vendor-independent method of software architecture design and development which comes from the well-known idea of separating the system operation specifications from system details with the low level platform. MDA provides a way to standardize a platform independent system, standardized platform for the system to select a specific implementation platform, and converting the system specification to a specific implementation platform. It can better meet the general needs of the software development industry, such as reducing operating costs and reducing time to market. Model driven architecture advocates using platform-independent model as a core driver to improve application development level, using code automatically generation method to improve the efficiency of development and increase the degree of application of model reuse.

After the discussion of MDA method and platform technologies in solving the enterprise management software development problems, the paper proposes a software development integration platform (the NGX platform) based on MDA and platform technologies. Using a verified platform to automatically build and deploy application can directly reduce the software process of demand analysis, design, coding, testing and deployment. With the method of repeatedly and iteratively modifying the business model and generating application through NGX platform, one can greatly reduce the iterative software development cycle, thus contributing fast convergence of software entity and the actual business. So NGX platform helps to increase development efficiency, shorten the time software products take to appear on the market and reduce development costs. As the code will be automatically generated by computer who’s been tested repeatedly, the software quality, robustness and scalability can be guaranteed. The unity of the software technology, As well, can be ensured. The code automatically generated by the machine to reduce the human error factor, to ensure that software developers can focus on functional testing process.

After discussing the core concept of model-driven architecture and model-driven development, the paper offers a basic form NGX platform. The NGX platform is divided into two parts, including a visual business modeling tools (NGX Studio), and an automatic deployment of integrated application platform (NGX Application). Then through the analyzing the transformation between CIM, PIM, PSM and the final code, the storage and conversion method of model in various stages is summed up: business development staff use visual business process modeling business modeling tools to model or adjust the business model; and then by the analyses of modeling tools, NGX automatically convert the model to J2EE- specific system model (PSM); the PSM are stored in the component library; then the engine loads models and converts them to executable component code; at last the business engine will automatically deploy and integrate the application with all aspects modules of the platform.

After the analysis and decomposition of general enterprise information systems, the paper gives elements of the existing business enterprise management system and the basic elements of NGX platform. Summering up the analysis of the business model the details of the decomposition business model, the paper propose a hierarchical structure of the business model design and component database storage structure based on the theory of MOF.

After determining the selection of developing technology, here we propose a solution of NGX platform based on Java EE lightweight framework, including the analysis of the hierarchical structure of the platform, the expansion of integrated approach, and , of course, the detail design and implementation of core modules of NGX platform. These include access control based on Role-Based Access Control model, management of multiple data sources and data from external system data unified access services, the global log management and execution of business operations module automatically generates the deployment engine.

Finally, the paper uses the planning and scheduling module of manufacturing execution systems as the object to verify NGX platform. Using NGX platform, we model the planning and scheduling module according to the business rules and the modeling rules; then we generate the module using NGX platform. As a result, we conclude that NGX platform can generate the business module with effectiveness and accuracy. Then we change the business rule by modifying the model and update the application to proof that the NGX platform is flexible and efficient enough for industry maintainability.

According to analysis results, through the NGX platform, the development of the entire enterprise management system can be transformed into the process of industry experts modeling business model in their own knowledge using particular symbol and integrate it into the programming model of higher level to form an enterprise information system. NGX's produced without the need to re-deploy the application, the system is easier to maintain. And because it is injected with business rules, it supports the re-import business model and re-building applications. Rebuilding the entire process can be controlled by the system administrator to re-build the automation process very short time. Therefore, when software needs reconstruction, changing the business model is the only cost. This will not only address the impact of constantly changing of demand of enterprise management software, but also enhance the convergence process of software entity and the actual business.

**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](#_Toc323202414)

[1.1 传统管理软件开发面临的问题 1](#_Toc323202415)

[1.1.1 系统需求复杂度变高 1](#_Toc323202416)

[1.1.2 系统需求变更频繁 1](#_Toc323202417)

[1.1.3 手工开发存在缺陷 2](#_Toc323202418)

[1.1.4 系统复用度过低 2](#_Toc323202419)

[1.2 国内外研究现状 3](#_Toc323202420)

[1.2.1 平台化软件开发研究现状 3](#_Toc323202421)

[1.2.2 模型驱动研究现状 3](#_Toc323202422)

[1.3 本文研究内容 4](#_Toc323202423)

[1.4 本章小结 5](#_Toc323202424)

[**第三章 NGX开发平台移动代码库架构设计及技术基础** 6](#_Toc323202425)

[3.1 开发平台的总体架构 6](#_Toc323202426)

[3.2 平台技术基础 7](#_Toc323202427)

[3.2.1 应用平台后台技术路线 7](#_Toc323202428)

[3.2.2 应用端UI技术路线 8](#_Toc323202429)

[3.2.3 MDA业务引擎技术路线 8](#_Toc323202430)

[3.2.4 数据库及部署环境 8](#_Toc323202431)

[3.3 本章小结 9](#_Toc323202432)

[**第四章 开发平台详细设计及实现** 9](#_Toc323202433)

[4.1业务模型的基础要素 9](#_Toc323202434)

[4.2业务模型的构件储存 11](#_Toc323202435)

[4.1.1 元-元模型层中的Application 11](#_Toc323202436)

[4.1.2元-元模型层中的Business Component 12](#_Toc323202437)

[4.1.3 元模型层 13](#_Toc323202438)

[4.2 NGX平台基础框架设计 17](#_Toc323202439)

[4.2.1 平台基础框架结构 17](#_Toc323202440)

[4.2.2授权模块设计 19](#_Toc323202441)

[4.2.3统一数据访问模块设计 20](#_Toc323202442)

[4.2.4统一日志管理模块设计 21](#_Toc323202443)

[4.3 业务引擎设计 22](#_Toc323202444)

[4.4 本章小结 25](#_Toc323202445)

[**第六章 结论** 26](#_Toc323202446)

[6.1 工作总结 26](#_Toc323202447)

[6.2 平台研究工作展望 26](#_Toc323202448)

[参考文献 28](#_Toc323202449)

[谢辞 30](#_Toc323202450)

**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。

**第三章 NGX开发平台移动代码库架构设计及技术基础**

如上一章所述，结合模型驱动开发平台的移动代码库诸多优势。本章在此提出平台的总体架构设计及基础要素。

3.1 开发平台的总体架构

NGX开发平台移动代码库是基于自动部署集成应用平台NGX Application的框架，在其上面可以进行在线编码，同时可以进行在线编译和测试代码 。

NGX开发平台移动代码库的总体结构如图3-1所示：

  **图3-1 NGX开发平台移动代码库的总体结构**

**NGX开发平台移动代码库**

**HTML + JS + JSP**

**TOMCAT SERVER**

**JSON**

**NGX Application**

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。

**（3）NGX开发平台移动代码库**

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。

**（4）模型元数据库（NGX Business Model Database）**

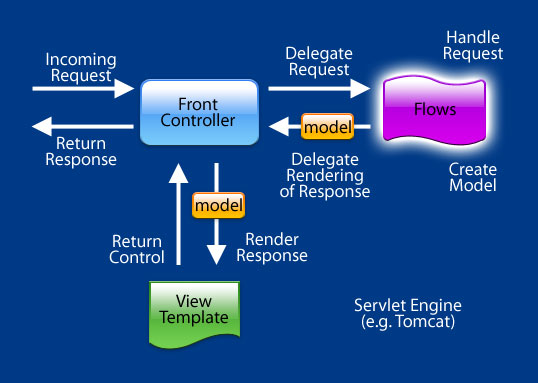
NGX Business Model Database是平台无关业务模型（PIM）存储数据库，它通过配置将NGX Studio与NGX Application有机结合起来。

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。



**3.2.1图。应用平台后台技术路线**

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), 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Jquery 和 Jquery UI 插件。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。

另外大量使用了HTML5 和 CSS3 的新技术。让前台代码更加简洁。前台有友好，清晰的UI设计。模仿苹果 OS操作系统界面。**支持未来的触摸屏技术.**

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) Jquery 1.7

(7) velocity 1.6.2

(8) ant 1.7.0

(9) tomcat 6.0

(10) Mysql 5.1

3.3 本章小结

本章并说明了NGX系统上的开发平台和移动代码库的技术路线以及使用的技术基础。

**第四章 开发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]。

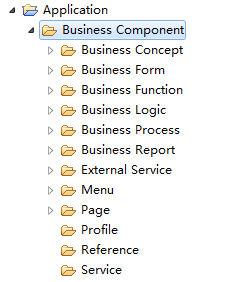
下面详细说明元—元模型层与元模型层的基础要素：

（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。



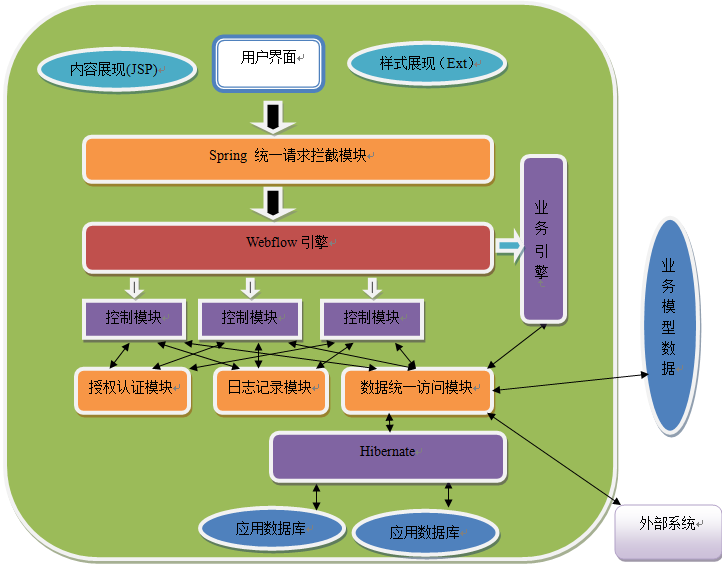
**图4-12 Menu的组成结构关系**

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。

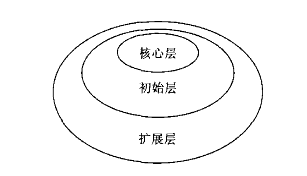


**图4-13 平台基础框架层次图**

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Jquery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。

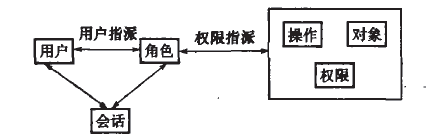
初始层是平台的管理监控模块与登入管理模块，它们使用平台的核心层提供了平台的原始功能。

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。

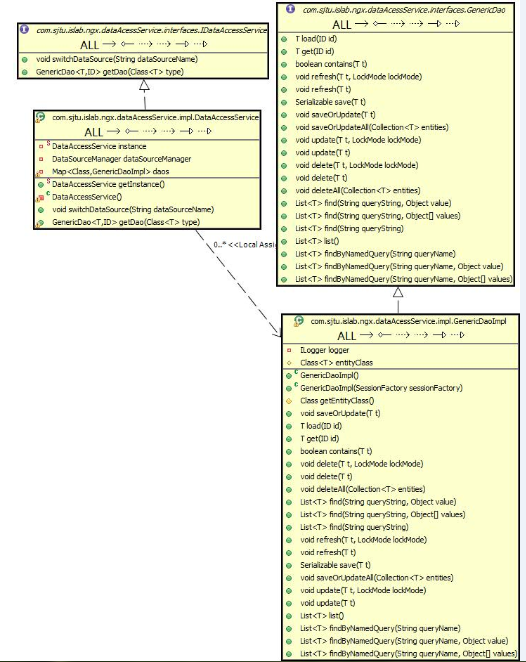
Authority负责用户根据角色授权，即管理员将User与多个Role绑定的过程。

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)

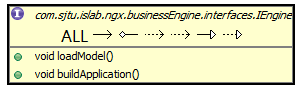
它以五个优先级便记录系统的执行过程。

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。

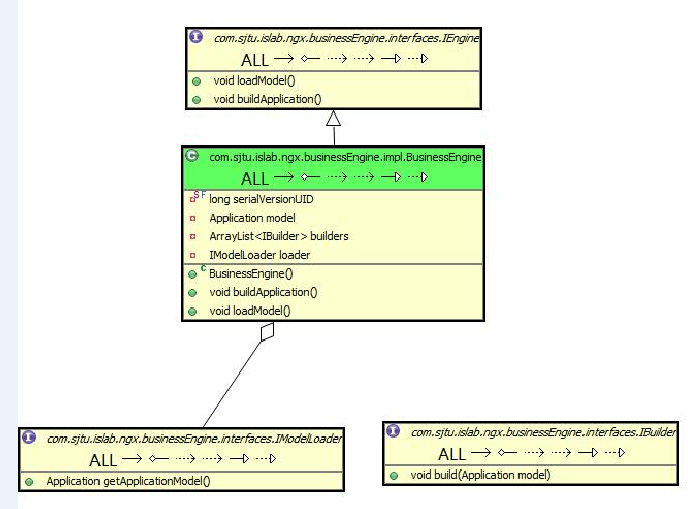
业务引起使用Facade设计模式[29]，将整个模块提取统一借口（IEngine）。

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。



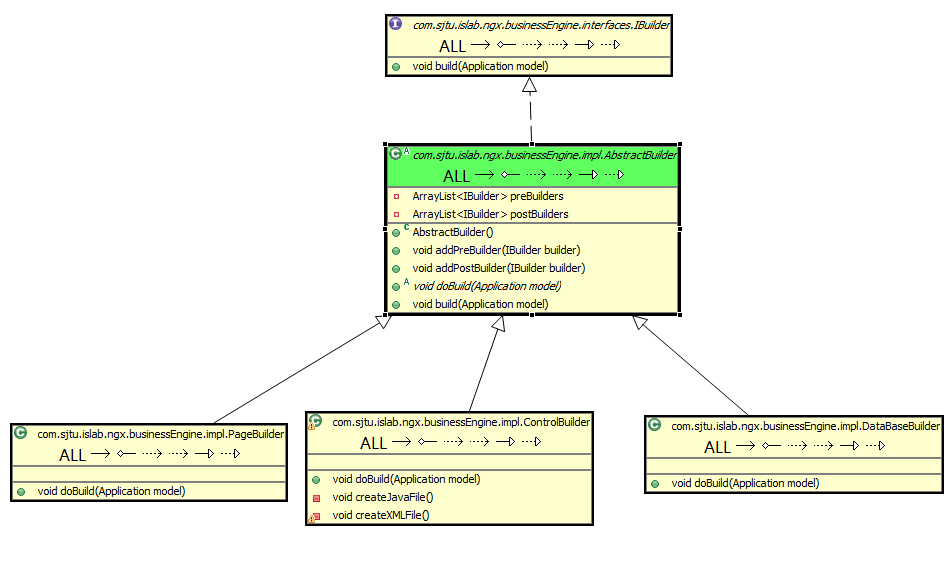
**图4-20模块的总调用接口**

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**

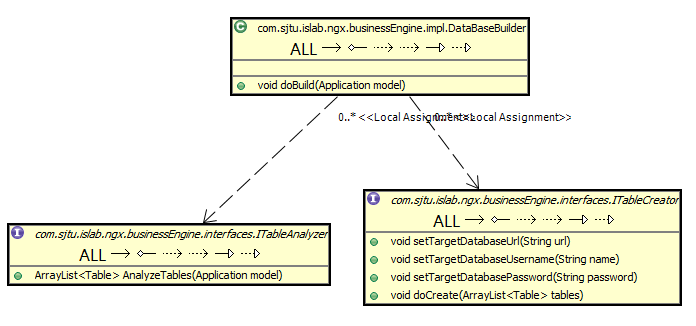
其中IModelLoader接口的getApplicationModel()函数为从数据库中读取业务模型并生成pojo对象；IBuilder接口的build()函数为根据生成的pojo对象生成一个层面上的代码模块。



**图4-22 Builder的组织结构**

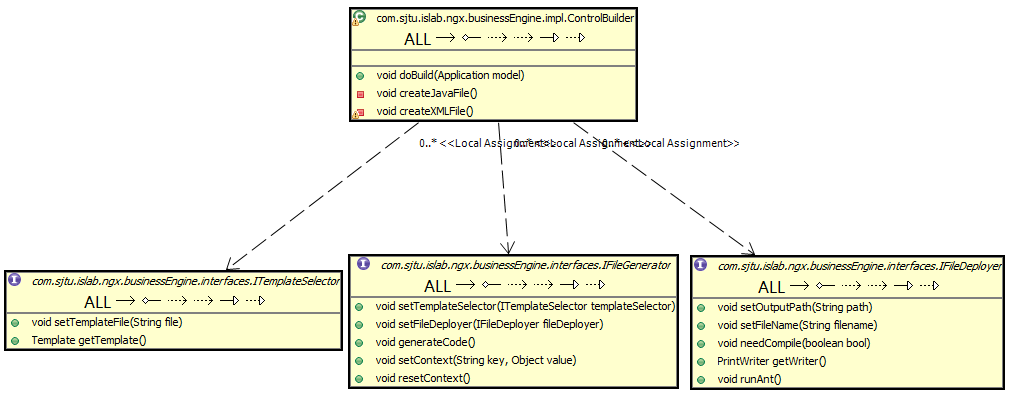
使用抽象类AbstractBuilder实现Decorator设计模式[29]。用addPreBuilder(IBuilder)与addPostBuilder(IBuilder)来注册一个IBuilder执行前后应该执行的其他Ibuilder，从而可以递归的进行整个项目模块的生成。其中所有Builder分为两类，在数据库中进行建表更改表操作的Builder与生成文件形式业务代码的Builder。

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。