



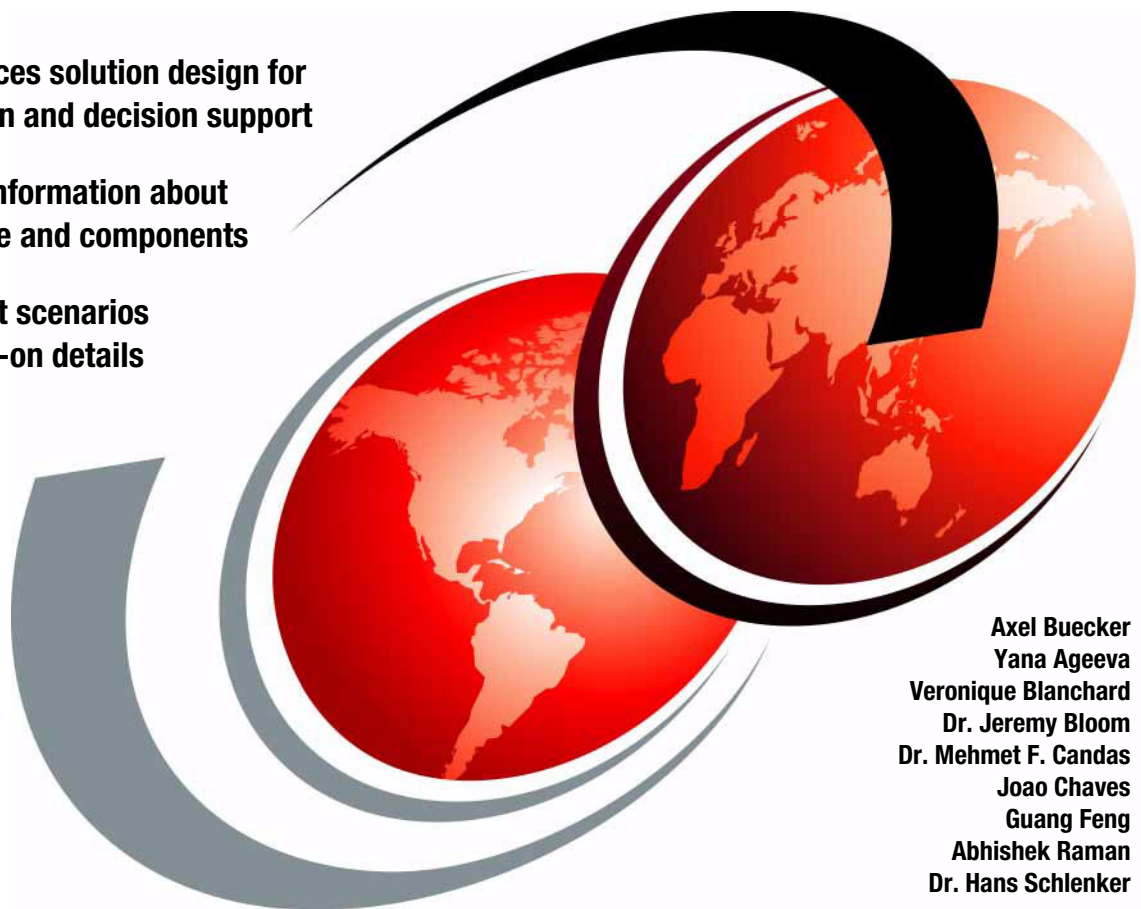
Optimization and Decision Support Design Guide

Using IBM ILOG Optimization Decision Manager

Best practices solution design for optimization and decision support

Complete information about architecture and components

Deployment scenarios with hands-on details



Axel Buecker
Yana Ageeva
Veronique Blanchard
Dr. Jeremy Bloom
Dr. Mehmet F. Candas
Joao Chaves
Guang Feng
Abhishek Raman
Dr. Hans Schlenker



International Technical Support Organization

**Optimization and Decision Support Design Guide:
Using IBM ILOG Optimization Decision Manager**

October 2012

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (October 2012)

This edition applies to Version 3.6 of IBM ILOG Optimization Decision Manager Enterprise (product number 5724-Y04).

© Copyright International Business Machines Corporation 2012. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team who wrote this book	xi
Now you can become a published author, too!	xvi
Comments welcome	xvi
Stay connected to IBM Redbooks	xvi
Part 1. Business concepts, architecture, and design	1
Chapter 1. Context for optimization and analytic decision support	3
1.1 Can we have your attention please	4
1.2 What is optimization	4
1.2.1 Analytics and optimization	4
1.2.2 What optimization can achieve	5
1.2.3 How do I recognize an optimization application	7
1.2.4 How optimization works	9
1.3 How can optimization benefit your organization	11
1.3.1 Example: Airline crew recovery	12
1.3.2 Example: Securities transactions settlements	14
1.3.3 Example: Power plant dispatching	16
1.3.4 Example: Semiconductor manufacturing	18
1.4 How can you make it happen	20
1.5 Why should your organization consider analytic decision support	22
Chapter 2. Introducing the ODM Enterprise solution	23
2.1 From optimization model to business application	24
2.2 The ContainerYard: An ODM Enterprise application example	26
2.2.1 Supply chain process	26
2.2.2 Planning process	27
2.2.3 Solution: An ODM Enterprise-based software application	28
2.2.4 Planning scenarios	33
2.3 Application concepts and parts of an ODM Enterprise solution	38
2.3.1 ODM Studio: The planning cockpit	40
2.3.2 Deployment architectures	42
2.4 Optimization concepts and parts of an ODM Enterprise solution	43
2.4.1 Optimization model	44
2.4.2 Optimization engines	44

2.4.3 CPLEX Optimization Studio	46
2.5 Roles and expertise	48
2.5.1 Client roles	48
2.5.2 Provider roles	50
2.6 Advantages of ODM Enterprise solutions	52
2.7 Conclusion	54
Chapter 3. ODM Enterprise architecture	55
3.1 Key ODM Enterprise concepts	56
3.1.1 Model	56
3.1.2 ODM Enterprise IDE versus ODM Studio	58
3.1.3 Scenario	60
3.1.4 Workspace	60
3.1.5 Scenario Repository	61
3.1.6 Goals	63
3.1.7 Requirements	64
3.1.8 Rules	64
3.1.9 Input data and solution views	66
3.1.10 Decision process configuration	77
3.1.11 Solve	80
3.1.12 Deployment	82
3.1.13 Collaboration	85
3.2 ODM Enterprise architecture	88
3.2.1 Optimization Server	89
3.2.2 Data Server	91
3.2.3 ODM Studio	93
3.3 ODM Enterprise API	95
3.4 Domain Object Model (DOM)	97
3.5 Data architecture	99
3.5.1 Scenario Repository	99
3.5.2 Importing data	101
3.5.3 Exporting data	101
3.5.4 Application and Jobs database	103
3.6 Scalability and load balancing	104
3.7 High availability	105
3.8 Security model	107
3.8.1 Simple authentication system	107
3.8.2 Custom authentication system	108
3.9 Conclusion	109
Chapter 4. Transforming business requirements into an ODM Enterprise application	111
4.1 ODM Enterprise development process	112

4.1.1	ODM Enterprise development flow	112
4.1.2	ODM Enterprise IDE	113
4.2	Building the ContainerYard application	115
4.2.1	Analyzing requirements and defining application scope	115
4.2.2	Defining the data model	118
4.2.3	Creating a prototype without optimization	122
4.2.4	Creating a prototype with optimization	138
4.2.5	Deploying the application	162
4.3	Conclusion	163
	Chapter 5. Solution design and management	165
5.1	ODM Enterprise editions	166
5.1.1	ODM Enterprise Developer Edition	166
5.1.2	ODM Enterprise Optimization Server	167
5.1.3	ODM Enterprise Optimization Engine	167
5.1.4	ODM Enterprise Data Server	168
5.1.5	ODM Enterprise Client Edition	168
5.1.6	ODM Enterprise Planner Edition	168
5.1.7	Staging database	169
5.2	Deployment and system design options	169
5.2.1	Common ODM Enterprise deployment configurations	170
5.2.2	Individual deployment	172
5.2.3	Enterprise with local optimization deployment	173
5.2.4	Enterprise deployment	174
5.2.5	Other considerations while you are designing the solution	176
5.3	Input data and processing	182
5.4	Conclusion	184
	Part 2. Customer use cases	185
	Chapter 6. Case study: An insurance company	187
6.1	Company overview	188
6.2	Business problem	188
6.3	Business requirements	190
6.3.1	Non-functional requirements	190
6.3.2	User interface requirements	191
6.4	Optimization problem	193
6.4.1	Choices	193
6.4.2	Targets	195
6.4.3	Limits	196
6.4.4	Data	198
6.5	Solution development	202
6.5.1	Architecture	203
6.5.2	Data requirements	205

6.6 Optimization	217
6.7 Conclusion.	220
Chapter 7. Case study: A manufacturing company	223
7.1 Business context	224
7.2 Business problem	225
7.2.1 Old process issues	227
7.2.2 Solution	227
7.3 Business requirements	228
7.3.1 Assignment	228
7.3.2 Capacity planning	229
7.3.3 Capacities, overload, and extension	230
7.3.4 Non-functional requirements	232
7.4 Optimization problem	232
7.4.1 Assignment	233
7.4.2 Capacity planning	244
7.5 Solution	252
7.5.1 Solution architecture	253
7.5.2 User interface	256
7.5.3 Implementation project	259
7.6 Conclusion.	259
Part 3. Appendixes	261
Appendix A. IBM solution landscape for analytics, optimization, and decision support	263
ODM Enterprise overview	264
IBM ILOG CPLEX Optimization Studio overview	266
SPSS	267
The SPSS product family	268
Predictive and Prescriptive Analytics	272
Case study: Major retailer	273
Case study: Large urban school district	274
Cognos	275
Cognos product capabilities	276
Descriptive and prescriptive analytics	279
Case study: Major food processor	280
WebSphere ILOG Rules.	281
Business rules management.	281
Business rules and prescriptive analytics	282
Case study: Leading global logistics service provider.	283
Summary	287
Appendix B. Troubleshooting	289

General concepts and guidelines	290
How ODM Enterprise uses memory	290
General information about Development IDE and ODM Studio	292
How ODM Enterprise Optimization Server works	294
ODM Enterprise logs guidelines	295
Packaging ODM Enterprise applications.	303
Isolating problematic behavior	303
Troubleshooting tips	304
Resolving issues during product installation	304
Where to look for more specific error messages.	305
Identifying performance issues with the database used by the ODM Scenario Repository	307
Sending messages from custom code to ODM Enterprise logs	310
Adjusting memory usage for ODM Enterprise.	311
Validating the configuration of ODM Enterprise server installation	314
Cleaning up the ODM Enterprise Development IDE workspace.	322
Improving memory usage when using an OPL data source	324
Improving memory usage when using an ODM data source	324
Debugging your custom code based on ODM Enterprise APIs	324
Appendix C. Additional material	329
Locating the web material	329
Using the web material.	329
Downloading and extracting the web material	330
Overview of the application scenarios.	330
Glossary	333
Related publications	341
IBM Redbooks	341
Education	341
Course: Introduction to Application Development with IBM ILOG ODM Enterprise V3.4	342
Course: Developing and Deploying Complete Applications with IBM ILOG ODM Enterprise V3.4	343
Other publications	343
Online resources	344
Help from IBM	344

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Cognos®

CPLEX®

DB2®

developerWorks®

i2®

IBM®

ILOG®

Redbooks®

Redbooks (logo) ®

Smarter Planet®

SPSS®

TM1®

WebSphere®

The following terms are trademarks of other companies:

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Today many organizations face challenges when developing a realistic plan or schedule that provides the best possible balance between customer service and revenue goals. Optimization technology has long been used to find the best solutions to complex planning and scheduling problems. A decision-support environment that enables the flexible exploration of all the trade-offs and sensitivities needs to provide the following capabilities:

- ▶ Flexibility to develop and compare realistic planning and scheduling scenarios
- ▶ Quality sensitivity analysis and explanations
- ▶ Collaborative planning and scenario sharing
- ▶ Decision recommendations

This IBM® Redbooks® publication introduces you to the IBM ILOG® Optimization Decision Manager (ODM) Enterprise. This decision-support application provides the capabilities you need to take full advantage of optimization technology. Applications built with IBM ILOG ODM Enterprise can help users create, compare, and understand planning or scheduling scenarios. They can also adjust any of the model inputs or goals, and fully understanding the binding constraints, trade-offs, sensitivities, and business options.

This book enables business analysts, architects, and administrators to design and use their own operational decision management solution.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.



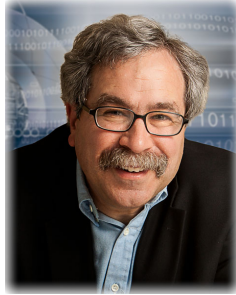
Axel Buecker is a Certified Consulting Software IT Specialist at the ITSO, Austin Center. He writes extensively and teaches IBM classes worldwide about software security architecture and network computing technologies. He has a degree in Computer Science from the University of Bremen, Germany. He has 25 years of experience in various areas related to workstation and systems management, network computing, and e-business solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.



Yana Ageeva is a Technical Account Manager for IBM ILOG Optimization products, including CPLEX® Optimization Studio and ODM Enterprise. She has been working with ILOG Optimization products since 1999 when she used OPL to implement a robust airline scheduling system as part of her thesis at MIT. Yana began her career at ILOG in 2004 and joined IBM through the ILOG acquisition in 2009. She has supported numerous clients on their optimization projects over the years in her roles in technical sales and worldwide support in the United States and in France. Before joining ILOG, she worked as a software developer for a couple of Silicon Valley startups. Yana holds three degrees from MIT: SB in Mathematics, SB in Computer Science, and MEng in Computer Science with a focus in Operations Research.



Veronique Blanchard is the IBM ILOG CPLEX Optimization Studio Quality Team Lead. She joined IBM in 2009 as part of the ILOG acquisition. She has been working on ILOG Optimization products since 1999, first as a support engineer, then as a quality assurance engineer. She holds a MS degree in Applied Mathematics from the Institut National des Sciences Appliquees, France.



Dr. Jeremy Bloom is the IBM Product Line Manager for IBM ILOG ODM Enterprise. He has more than twenty five years of business experience applying operations research and optimization. He joined ILOG in 2007 as a Technical Account Manager and then joined the product marketing team, continuing when IBM acquired ILOG in 2009. He has been responsible for the website and for campaigns in the Energy and Utilities and Financial Services. Previously, he worked at the Electric Power Research Institute and at General Public Utilities. His experience includes integrated resource planning, retail market tools, distributed resources, and asset management. Early in his career, he taught operations research at Cornell University. Jeremy received his undergraduate education in electrical engineering from Carnegie-Mellon University, and his graduate degrees in operations research from the Massachusetts Institute of Technology.



Dr. Mehmet F. Candas is a Research Engineer in the Business Analytics and Mathematical Sciences Department at IBM T.J. Watson Research Center. He develops descriptive, predictive, and prescriptive analytics-based solutions for various problems for both government agencies and private companies. Before joining IBM Research, he worked at one of the world's leading semiconductor companies as an IT Architect for five years. During that time, he designed and developed optimization-based decision support systems for their supply chain management, manufacturing planning, and customer order fulfillment process. He also taught classes and gave seminars at the University of Texas in Austin as adjunct professor.

Mehmet received the B.S. degree in industrial engineering from Bilkent University, Ankara, Turkey. He received a PhD degree in operations research and industrial engineering from the University of Texas in Austin, TX.



Joao Chaves is an Optimization Solutions IT Architect, supporting the Optimization Decision Manager Enterprise technical sales team. Joao has over eighteen years of experience in the architecture, design, and implementation of global enterprise systems, including data hubs and warehouses, portals, and collaboration systems. He is currently focusing in the architecture and design of optimization, information management, and data integration applications. A Methods evangelist, Joao is considered a thought leader in enterprise data integration topographies and services-oriented architecture (SOA). He holds an IBM and Master IT Architect certification along with a BS in Marketing degree from Florida Metropolitan University.



Guang Feng is an L2 support engineer with the Industry Solutions (IS) support team in IBM. He joined ILOG (later acquired by IBM) in 2006. He has been providing customers with remote technical support since then. He is an expert in IBM ILOG's optimization suite of products (including CPLEX Optimization Studio, ODM Enterprise, and AMPL). He provides guidance to customers on the effective use of ILOG optimization products, and is experienced in helping customers to diagnose unusual behaviors. He obtained his Masters of Computing from the National University of Singapore in 2006.



Abhishek Raman is an L2 support engineer with the Industry Solutions (IS) support team in IBM. He joined IBM in 2009 as part of the ILOG acquisition. He has provided customers with remote technical support since 2008 when he joined ILOG. He is adept in the use of IBM ILOG's optimization suite of products (including CPLEX, OPL, AMPL, CPO, and ODM Enterprise) to model real-life OR problems in diverse industries. Before ILOG, he worked at Cypress Semiconductors as an Industrial Engineer developing mathematical models for capacity planning and production scheduling and optimization. He holds a BTech degree in Automobile engineering from Anna University, Chennai, and a Masters in Industrial and Systems engineering from the University of Minnesota, Twin Cities.



Dr. Hans Schlenker is an IBM Technical Sales Specialist for IBM ILOG Optimization products and solutions. He joined ILOG in 2005 and worked in different roles in services and sales, since 2009 for IBM. He sold and delivered projects and solutions for many companies in various industries including automotive, industrial production, logistics, banking, insurance, and energy. Overall, he has more than 15 years of experience in algorithm development, IT architectures, optimization modeling, and solution design. Hans taught classes and gave seminars at several universities. He currently holds an IBM research assignment for the Karlsruhe Institute of Technology, where he researches real-world optimization projects. Hans holds a Masters degree in Computer Science from Ludwig Maximilians Universität, Munich, and a PhD in Industrial Engineering / Computer Science from Technische Universität, Berlin.

Thanks to the following people for their contributions to this project:

Emma Jacobs

Ernest A. Keenan

International Technical Support Organization, Austin Center

Faisal Ahmad

Arnim Breycha

Michel Eisenmann

Sebastian Fink

Philippe Gregoire

Jan Helleve

Michel Jaczynski

Michael King

Frederic Lavigne

Dr. Hans-Martin Wallmeier,

IBM

Dr. Alfons Steinhoff

Steinhoff und Partner, Germany

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. 1WLB Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



Part 1

Business concepts, architecture, and design

This part addresses the business context for optimization and analytic decision support, and introduces the IBM ILOG ODM Enterprise solution. It then examines in more detail the IBM ILOG ODM Enterprise architecture, and addresses how to transform business requirements into an ODM Enterprise application. The final chapter explores the solution design and management aspects. This part includes the following chapters:

- ▶ Context for optimization and analytic decision support
- ▶ Introducing the ODM Enterprise solution
- ▶ ODM Enterprise architecture
- ▶ Transforming business requirements into an ODM Enterprise application
- ▶ Solution design and management



Context for optimization and analytic decision support

Many enterprises have become more efficient and more effective in their operations by using advanced analytics. This is an overview of the process of implementing effective decision support solutions that are based on optimization.

This chapter includes the following sections:

- ▶ Can we have your attention please
- ▶ What is optimization
- ▶ How can optimization benefit your organization
- ▶ How can you make it happen
- ▶ Why should your organization consider analytic decision support

1.1 Can we have your attention please

An airline uses a sophisticated scheduling solution to reposition its crews after a major disruption that enables it to *recover more quickly after the September 11 attacks*. A power company uses an advanced solution for dispatching power plants to generate electricity. It achieves savings that more than *pays for the application development in less than 10 days*. A financial institution uses an ingenious way to match buy and sell orders that *reduces clearing time from one day to under two minutes*. Think what similar efficiency improvements could mean for your business.

These striking examples have one thing in common: All these companies built their decision support solutions by using an advanced analytics technology called *optimization*. These days, there is considerable buzz about analytics and sometimes even about optimization. But for most business executives, the realm of analytics remains mysterious, with promises that seem almost too good to be true.

This book demystifies optimization. It shows step-by-step how to translate challenges your organization faces into effective decision support solutions that can deliver enormous benefits to your bottom line. IBM has the expertise to help you because we use optimization ourselves. For instance, one of our semiconductor fabrication facilities uses optimization to reschedule production of complex microelectronics *every five minutes*.

1.2 What is optimization

Optimization is a sophisticated analytics technology. It can deliver great business value by helping decision makers handle the tough trade-offs that arise when limited resources force choices among options.

1.2.1 Analytics and optimization

Today's businesses run on lots of information. As the power to collect information has increased exponentially in the last several decades, business leaders increasingly focus on what to do with all that information. Analytics provides tools to assimilate, analyze, and act on information. These analytic technologies fall into three categories:

1. *Descriptive analytics* provide information about the current (and past) state of the business and its environment.

The descriptive realm uses regular *reports* for events that already happened, and *ad hoc reports* to help examine facts about what happened where, how often, and with how many. It also offers individual *queries* to allow someone to investigate what exactly the problem is.

2. *Predictive analytics* forecast future states of the business or how the environment might evolve.

The predictive realm looks at *real-time events* and *alerts* to suggest actions, and it uses *simulation models* to suggest what can happen. Based on this data, the predictive approach allows you to run both of these actions:

- A *forecasting*, which looks at the probability for these trends to continue
- A *predictive modeling*, which allows you to look into what-if situations.

3. *Prescriptive analytics* recommend choices.

The prescriptive realm offers *optimization* to examine how you can achieve the best outcome for a particular situation. It also offers *stochastic optimization* to evaluate how you can mitigate or even avoid uncertain risks.

1.2.2 What optimization can achieve

As one of the most complex analytics technologies, optimization can also deliver some of the greatest business value. It does so by helping decision makers with the tough trade-offs that arise under limited resources. These resources can include capital, workers, capacity, raw materials, and time. Basically, whichever resources limit your ability to grow.

Optimization is about making decisions. It works best when there are thousands of choices to make and resource limits force the decision maker to trade off among them. As an example, a manufacturer produces ten thousand products in twenty plants worldwide. The sales forecast for the next year requires making over 1 million distinct items. The company's planners need to decide where each item will be produced. These decisions must take into account local labor costs, shipping costs, raw materials availability, work center capacity in each plant, and many other factors.

In situations like this example, optimization offers three advantages:

- *Optimization can discover previously unknown options or approaches*

Optimization can examine millions of choices to find the best ones within the resource limits. That means that optimization can look at many more options than any human decision maker. In such situations, humans usually rely on intuition and experience, using familiar patterns. But in rapidly changing business environments, experience-based patterns frequently lose relevance.

Optimization can break out of the familiar to find novel, substantially better choices that enable fact-based decisions.

For example, the past twenty years have seen enormous restructuring of the electric power industry worldwide, changing daily operations from central command-and-control to market-based bidding. As a result, power system operators found that their traditional dispatch procedures could not adapt to the new market structures. Many implemented new dispatching procedures that are based on optimization. These new procedures have resulted in billions of dollars in annual cost savings.

► *Optimization can automate and streamline decisions*

Optimization automatically and rapidly examines the millions of choices available to decision makers, and ensures compliance with complex policies and regulations. It can free up decision makers to focus on the critical challenges of their business rather than on routine tasks, increasing their value and enhancing their satisfaction.

For example, day-to-day personnel scheduling matters often consume a large part of each week for many hospital nurse managers. Scheduling is complex because the patient census varies and nurses need off sick time or personal time, requiring reassignments. However, highly skilled nurse managers create more value out on the floor, overseeing patient care and mentoring more junior nurses. Automating the schedule not only frees the nurse manager for higher value activities, but it creates more flexibility for the nurses. This flexibility increases job satisfaction, which is a critical retention issue in many hospitals today.

► *Optimization can enable decision makers to explore more scenarios and alternatives*

Most decision makers will not accept an automated solution. They need to validate it to have confidence that it works properly and realizes the value claimed for it. Furthermore, conditions can change rapidly. Decision makers need assurances that the choices they make will perform well under circumstances different from the environment when the decisions were made. Optimization can enable them to test their decisions under various contingencies. Furthermore, it provides deeper insights into the trade-offs that are required to obtain the greatest value with the limited resources available. And the insights extend further. Optimization provides new ways to view and understand both the information that drives the decisions and the outcomes that result from them.

For example, a consumer products company wanted to restructure its distribution network to reduce delivery costs. An optimization solution suggested that reducing the number of regional warehouses would enable savings by consolidating shipments, taking advantage of lower freight rates on full truck-loads. However, deeper analysis showed that in many cases,

shipments would be delivered on the second day rather than the next day, reducing product availability and customer satisfaction. The company decided that it could absorb some additional delivery costs to improve the rate of next day deliveries. Optimization allowed it to quantify this trade-off exactly to determine how much value it should assign to next day delivery.

1.2.3 How do I recognize an optimization application

People frequently use the term “optimization” to mean “making something better.” However, although optimization often makes some things better, it means a lot more than that. Optimization means finding the *best solution* to a precisely defined situation. In particular, optimization requires measuring value and making specific decisions that maximize that value.

An optimization solution is based on the following concepts. For more information about these concepts, see Chapter 4, “Transforming business requirements into an ODM Enterprise application” on page 111.

► *Targets*

All decisions are not equal. Measurable targets, or *goals*, allow you to rank alternative decisions. Targets often take the form of *key performance indicators*, such as minimize cost, maximize revenue, maximize margin, and minimize delay. An optimization problem can have multiple targets, and they can conflict with each other, requiring trade-offs. However, quantitative targets are an essential element in optimization.

► *Limits*

Not all decisions are acceptable. Limits on resources distinguish acceptable from unacceptable combinations of decisions. Limits, or *requirements*, can arise from various circumstances:

- Physical laws, such as flow in = flow out at a pumping station
- Capacity, such as shelf space in a warehouse
- Inputs, such as the amount of raw material on hand
- Regulations, such as a maximum eight-hour straight-time workday for workers
- Policies, such as 99% of shipments must arrive on time

An optimization problem usually has multiple kinds of limits, and they can conflict with each other, requiring trade-offs. Limits must also be measured quantitatively.

► *Choices*

Decisions must be made, but there are many ways to make them. These alternatives require trade-offs to satisfy the limits and achieve the best value of the targets. Decisions must be precisely specified, such as how much of a particular product to produce on a specific day at a plant. Decisions must also be measurable.

Optimization problems can have two types of decisions:

- *Continuous* decisions represent quantities, such as the amount to produce.
- *Discrete* decisions represent choices, such as whether to open a warehouse at a specific location.

This distinction has enormous practical implications for solving optimization problems. Problems with only continuous decisions usually require much less computational effort than those that have some discrete decisions. However, the ability to represent discrete decisions gives optimization much of its business value.

► *Data*

As the preceding discussion has emphasized, optimization deals with situations in which targets, limits, and choices are measurable. That fact means that optimization requires data. In fact, optimization extracts enormous value from data. For instance, a package delivery company originally justified equipping its trucks with GPS to enable real-time route optimization.

1.2.4 How optimization works

Decision support applications that are based on optimization have a well-defined architecture as shown in Figure 1-1.

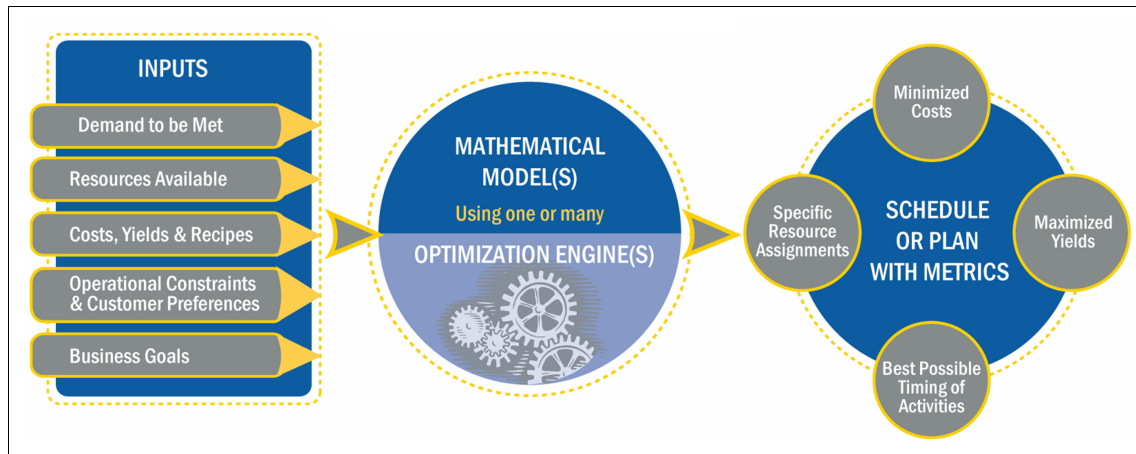


Figure 1-1 How optimization works

It all starts with a *model* of the process or system to be controlled. Here, the term model is the mathematical expression of the targets, limits, and choices that are involved in the decisions. For more information about formulating optimization models, see Chapter 4, “Transforming business requirements into an ODM Enterprise application” on page 111.

The optimization model specifies the relationships among the targets, limits, and choices, but it is input data that make these relationships concrete. An optimization model for production planning, for example, has the same form whether you are producing three products or a thousand. The input data represent the resources available and the product demands to be met. They also include the parameters that represent the recipes for turning resources into products and the costs and customer preferences. You can use these data to calculate the key performance indicators of the business goals.

The principle of *model-data separation* enables a decision support application to scale nicely. The input data not only provides the concrete parameters of the model but also its scale in terms of these factors such as:

- ▶ The number of products
- ▶ The number of resources
- ▶ The locations of the production facilities

Thus, the key mathematical relationships need to be specified only once in order for the application to scale as the scope of the business expands.

The model plus the input data creates an instance of an optimization problem. The optimization engines then apply mathematical algorithms to find a solution, a set of decisions that achieves the best values of the goals and respects limits imposed. These algorithms reflect years of research effort, but you do not need to understand them to use optimization. From a practical perspective, their important characteristic is their *performance*, which is measured by speed, reliability, and robustness. Speed means how quickly they produce a solution. Reliability means that they can find a demonstrably correct solution. Robustness means that they provide similar speed and reliability on closely related problem instances, such as evaluating alternative scenarios.

The solution that emerges from the solver engines is the recommended values for all of the decisions that are represented in the model. Equally important are the metric values that represent the targets. These values measure the quality of the solution in terms of the business goals. Usually, the solution and goals will be summarized to the decision makers in tabular or graphical views that provide understanding and insight. Alternatively, the detailed solution can be automatically transmitted to an execution system for implementation as a part of the enterprise IT process flow.

Using mathematics in decision-making puts off some people, but you do not have to understand the math to use optimization effectively. Rather, successful use of analytics to support decision making depends on several other capabilities:

- ▶ Drilling down into data to diagnose issues
- ▶ Visualizing solutions in tables and graphs to build insights
- ▶ Changing priorities among targets and limits to understand trade-offs
- ▶ Testing alternative scenarios to understand sensitivities to inputs, and to ensure robustness of decisions in the face of uncertainties

For more information, see Chapter 3, “ODM Enterprise architecture” on page 55.

1.3 How can optimization benefit your organization

Decision support that is based on optimization can be applied to almost every industry. Figure 1-2 shows a sample of these applications. A good way to think about optimization applications is to focus on the time frame and frequency of the decisions you need to make:

- *Operational decisions* typically have a short-term focus, and are made weekly, daily, or even hourly, depending on the cycle time of the business. These kinds of decisions include scheduling production orders on individual work centers, dispatching and routing deliveries, assigning workers to shifts, committing power generators, and settling securities trades. Business operations can involve thousands or tens of thousands of individual choices that must be decided in a tight time frame. In the absence of an optimization-based solution, operations managers often resort to using familiar patterns and rules-of-thumb. However, these snap judgments might prove unreliable or costly in a fast-moving business environment.

MANUFACTURING	TRANSPORTATION & LOGISTICS	FINANCIAL SERVICES	UTILITIES, ENERGY & NATURAL RESOURCES	TELECOM	MULTIPLE/ OTHER
• Inventory optimization • Supply chain network design • Production planning • Detailed scheduling • Shipment planning • Truck loading • Maintenance scheduling	• Depot/warehouse location • Fleet assignment • Network design • Vehicle & container loading • Vehicle routing & delivery scheduling • Yard, crew, driver & maintenance scheduling • Inventory optimization	• Portfolio optimization and rebalancing • Portfolio in-kindling • Trade crossing • Loan pooling • Product/price recommendations	• Supply portfolio planning • Power generation scheduling • Distribution planning • Water reservoir management • Mine operations • Timber harvesting	• Network capacity planning • Routing • Adaptive network configuration • Antenna and concentrator location • Equipment and service configuration	• Workforce scheduling • Advertising scheduling • Marketing campaign optimization • Revenue/Yield management • Appointment & field service scheduling • Combinatorial auctions for procurement

Figure 1-2 Industry applications of optimization

- *Tactical planning* decisions typically have a medium-term focus, and are made monthly, quarterly, or annually depending on the business context. These kinds of decisions typically include production planning, marketing campaign planning, aircraft and flight crew assignment, inventory stocking, materials procurement, staffing, and water management in a river system. The number of individual decisions is usually smaller than in operations and the time-frame is often not as urgent. However, each decision often has a

higher effect on key performance indicators such as cost or customer satisfaction. Thus, an optimization solution can often generate greater benefit over the rules-of-thumb that are often used as an alternative.

- *Strategic decisions* typically have a long-term focus, made with annual or longer frequency, or made on an occasional, as-needed basis. These kinds of decisions typically include capital investments in new equipment or expanded production facilities, distribution center location, fleet sizing, transportation time-tabling, timber harvesting, and mine planning. Strategic decisions typically are few, and the evaluation might take months. However, these decisions are often irreversible and therefore entail significant costs and risks. Because such decisions often involve high-level corporate policy questions, optimization constitutes only one of the analytics applied. However, optimization can put them on a solid factual basis and provide key insights because of the complex interactions among the choices.

To help you understand how optimization can help your organization make better decisions faster, this section includes specific examples. These diverse examples explain how companies have used optimization effectively to achieve significant bottom-line benefits.

1.3.1 Example: Airline crew recovery

A large airline that implemented this solution was transporting passengers, cargo, and mail with more than 2,000 daily departures to over 200 domestic and international destinations¹.

Airline schedules require routing of aircraft and crew members (pilots and flight attendants) on continuous paths. That is, a plane or a crew member arriving at an airport will later depart from that same airport. Rarely, a person might move from one airport to another on the ground, especially within the same metropolitan area, but such moves cannot take place instantaneously. Furthermore, flight schedules are usually *cyclic*, meaning that the same patterns of arrivals and departures recur on a daily basis (or less frequently, every few days). When a major disruption occurs, such as a snow storm that closes an airport, the effects ripple throughout the route system. Flight delays and cancellations at one airport cause delays and cancellations at others. This process can leave crew members at locations where they cannot board their scheduled flights, and can leave planes without correct staffing. Needless to say, such disruptions can have significant negative effects on an airline's revenues, customer satisfaction, and public relations.

¹ The full case is available in Yu, G. et al., "A New Era for Crew Recovery," *Interfaces*, Vol. 33, No. 1, January–February 2003, pp. 5–22. INFORMS, Reprinted by Permission.

Following one such disruption, this airline decided to develop a decision-support solution. This solution was aimed at returning its crews to their scheduled assignments in the most expeditious manner, an operation called *crew recovery*.

A critical set of *limits* affects crew scheduling and recovery, resulting from government regulations and labor contracts. Pilots must be qualified to fly a particular type of aircraft and to hold a particular seat (role) in the cockpit. Therefore, pilots are not interchangeable among flights. Flight attendants, however, can generally serve on any type of aircraft. In addition, crew members have limits on their on-duty hours and mandatory rest periods between on-duty tours. Furthermore, crew members generally receive their assignments through a monthly bidding process. For the sake of morale and contractual obligations, the airline tries to maintain those assignments as much as possible.

In recovery operations, managers might cancel or delay flights, or they can change the type of aircraft that is assigned to a flight leg. They must consider how to re-accommodate passengers, aircraft maintenance requirements, and other factors to come up with a modified flight schedule for the recovery plan. These decisions provide input to the crew recovery solution, which must cover these flights.

The *target* in crew recovery is to minimize the added costs (in addition to the costs of implementing the regular schedule) to cover the remaining flights in the schedule.

The *choices* in the crew recovery problem are the specific assignments of crew members to flights. An assignment might have a crew member working a flight. A crew member can also “deadhead,” which is to travel as a passenger to arrive at an airport for the next assignment.

During its first year using the crew recovery solution, the airline estimates it saved \$US 40 million in four major disruptions. Most significantly, the new system proved critical in the aftermath of the September 11, 2001 terrorist attacks. These attacks shut down air service in the United States for several days and cut demand for flights for a significant period after service resumed. By producing a new schedule, the crew recovery solution enabled the company to recover more quickly than most of the other airlines. In addition, they suffered fewer delayed flights and achieved more stable schedules, which enabled it to publicize its modified schedule and to re-accommodate its passengers. That enabled the company and its customers to return to normal operations sooner after this catastrophic event.

1.3.2 Example: Securities transactions settlements

A central securities depository (CSD) performs two vital functions for financial markets. It provides services that are related to the custody and management of securities that are retained on behalf of the financial community and the investing public. It also provides for the efficient settlement of securities transactions. For the CSD described in this case, these amount to \$US 250 billion per day, one fifth of its country's GDP².

The settlement system itself has a tremendous effect on the liquidity and efficiency of the financial markets. Before installing the new system, this CSD was operating with 9-14 different settlement systems that complicated settlement of transactions. This system also limited liquidity and forced financial institutions to reserve substantial financial resources to settle securities transactions. These limits raised costs for smaller players and made it difficult for them to participate.

In the mid-2000s, this CSD began working closely with the country's central bank and top technological university. This group worked to adopt international best practices that were developed by the Bank of International Settlement (BIS) in a new securities settlement system (SSS). Securities settlement systems are generally conducted according to one of three models as identified by the BIS and the Committee on Payment and Settlement Systems:

- ▶ Model 1: Real-time gross settlement for individual transactions
- ▶ Model 2: Real-time delivery of securities and net settlement of cash, usually at the end of the day
- ▶ Model 3: Net settlement of both securities and payments at the end of each settlement cycle

Previously, this CSD (like most CSDs around the world) operated with a Model 1 system. In the Model 1 gross settlement system, the individual seller (or its bank) delivers the securities to the individual buyer (or its bank). The cash flows in the opposite direction. It is easy to link the delivery of securities to the corresponding payment. However, this system requires depositors to have adequate financial resources available to settle their trades. Financial institutions must borrow funds to support transactions if they do not have sufficient funds to settle transactions, which incurs borrowing fees and limits liquidity. In Model 1, transactions remain open for considerable time, usually a day.

The CSD and the central bank decided to build a Model 3 settlement system. This system nets securities and cash in near real time, settling trades as quickly

² Adapted from "INDEVAL saves Mexican banks 240 million USD in 18 months," WSC14244-USEN-00, IBM Corporation, 2011, http://www.ibm.com/software/success/cssdb.nsf/CS/CPOR-8C8MGZ?OpenDocument&Site=optimization&cty=en_us

as possible after they occur, without incurring borrowing costs. In this case, transactions are settled among about 100 banks and brokerages acting on behalf of their customers. These institutions then settle the transactions with their individual customers.

Think of net settlement this way: If Alice owes Bob \$1 and Bob also owes Alice \$1, no money actually needs to change hands because the transactions cancel each other. Now suppose that Alice owes Bob \$1, Bob owes Carol \$1, and Carol owes Alice \$1. Again, the transactions settle without money changing hands. Such simple cancellations are easy to spot if you are dealing with a handful of transactions. However, financial markets must handle thousands of transactions every second, with multiple securities and with much more complex chains of cancellation. By using optimization, such chains, no matter how complicated, can be identified automatically and quickly. That insight led directly to the CSD's new securities settlement system (SSS).

The CSD's new SSS has three modules. The first coordinates all the settlement procedures. The second is the settlement module which modifies all of the accounts of the participating institutions in money and securities. The third is the clearing module, which uses a novel optimization algorithm. In many cases, analytics add on to an existing business process, providing insights that improve speed and efficiency. In this CSD's case however, the novel optimization algorithm provided the heart of a radically re-engineered process.

The business problem a CSD needs to solve has the following structure:

► Targets

Maximize the value of the transactions that are settled in this cycle.

► Limits

- For each security and each institution, the net number of shares sold (that is, sales minus purchases) cannot exceed the number of shares the institution has on hand at the beginning of the cycle.
- For each institution, the net value of purchases of all securities (that is, value of purchases minus value of sales) cannot exceed cash the institution has on hand at the beginning of the cycle.
- For each security and transaction, the number of shares that are exchanged between each pair of institutions must be non-negative and no greater than the transaction amount.

► Choices

The number of shares of each security that are delivered from one institution to another in fulfillment of each transaction.

The new SSS goes into action every two minutes. Every time the CSD runs the clearing module, it creates a linear programming model that maximizes the settlement value of all the transactions pending in the system. The SSS interacts with the IBM ILOG CPLEX Optimizer frequently, approximately 270 times a day. The IBM ILOG CPLEX Optimizer is the core of the settlement system. It is important because it optimizes the flow of money and securities for the market participants. This process saves many resources. Otherwise, they would have to maintain high balances in their accounts to settle their daily transactions.

The SSS clearing and settlement module using optimization reduces by 52 percent the liquidity requirements and by 26 percent the volume of securities that are required for settlement. The intraday financing costs for financial intermediaries have been reduced by more than \$US 240 million in the first 18 months of operation.

The most important benefit of the new SSS is the enhancement and strengthening of the country's financial system. As a main provider of liquidity to the whole financial sector, the CSD helped minimize the risk that is involved in settling transactions. This benefit diverted the worst effects of the banking crisis and subsequent drying up of liquidity of the 2008-2009 recession. The failure of Lehman Brothers in November 2008 highlighted the need to settle open transactions as quickly as possible.

1.3.3 Example: Power plant dispatching

An *Independent System Operator*³ (ISO) is an institution increasingly used throughout the world to manage restructured electric power markets. These markets have vertically separated supply chains with multiple, competitive power generators that supply local distribution companies that deliver power to end-use customers. Because electricity cannot easily be stored, it represents a true just-in-time production environment. To maintain reliability, a central operator takes responsibility for making the market and for physically operating the transmission grid that moves power from generators to distributors.

This ISO was formed in the late-1990s under the auspicious of the national energy regulator to create a unified power grid in a large, populous region of the country. It manages most of the region's high-voltage transmission grid. It has responsibility for developing and maintaining the network's transmission lines, substations, and other infrastructure. In 2011, it operated almost 55,000 miles of high-voltage electricity lines and 1500 generating plants with capacity of almost

³ Large U.S. Regional Transmission Operator, EUC03016-USEN-00,
http://www.ibm.com/software/success/cssdb.nsf/CS/DJOY-8QTQTG?OpenDocument&Site=default&cty=en_us

150 gigawatts, including about 15% renewables. Transactions in 2011 were about \$US 25 billion.

Although the ISO typically does not itself own power generating stations, it is responsible for the dispatching function. It selects which generating units to operate and what production levels they provide. These decisions occur hourly and extend one to three days into the future. The business problem an ISO needs to solve, called *unit commitment*, has the following structure:

- ▶ Targets
 - Minimize operating costs, which consist of starting costs of the generators plus the fuel and other variable expenses of generating power.
 - Minimize emissions.
- ▶ Limits
 - Satisfy forecasted loads.
 - Observe generator maximum output and ramp-rate limits.
 - Observe generator minimum up- and downtimes.
 - Observe spinning reserve requirements (spinning reserve is unused capacity of committed generators that is available for rapid ramping to adjust to load fluctuations and generator outages).
 - Observe emissions limits.
 - Observe transmission network flow limits.
- ▶ Choices
 - When to start (commit) and shut down (decommit) each generator.
 - How much electricity to produce from each generator in each hour.
 - How much spinning reserve to acquire from each generator

Like many other power system operators, this ISO was already using an optimization-based solution when the electricity industry was restructured. However, the existing solution proved hard to adapt to the new market structure. In 2009, the ISO decided to integrate contingency reserves into its energy market commitment and dispatch solution.

Some of the same power plants that generate energy for businesses and homes also provide reserves to respond to disruptions. Because it takes time (minutes to hours) to start a plant, reserves are necessary to cope with rapid changes in supply and demand. These changes can occur in a matter of seconds.

If a power plant or a transmission line goes down, the ISO can increase the energy output of the power plants that provide these reserves. Previously, a patchwork of entities managed reserves and other ancillary services in the ISO's

region. The ISO knew it could do the job more efficiently by consolidating these workloads. Calculating energy and reserves separately had less-than-optimal results because the two are interdependent. When a plant produces more power, it has less capacity for reserves.

However, optimizing both energy and reserves together creates a problem that is far more complex than optimizing them separately. In the commitment problem, adding reserves to the problem adds 300 percent more variables and constraints. A commitment problem solution can involve as many as three million continuous variables, 450,000 binary variables, and four million constraints. No energy market system had ever been built to dispatch as many power plants and miles of transmission lines as the ISO had to factor into the problem.

The ISO undertook to build a new unit commitment solution by using a novel optimization technology, *mixed integer programming* (MIP). This technology has these advantages:

- ▶ It is reliable, producing optimal solutions without manual adjustments within an acceptable time frame.
- ▶ It is flexible, easily adapting to new market structures and regulations.
- ▶ It is cost effective.

As a result of this new unit commitment logic, between 2007 and 2010, the ISO realized between \$US 2.1 billion and \$US 3.0 billion in net cumulative savings. The optimization solution was a major driver of these savings. The ISO also estimates that it will realize an additional \$US6.1 billion to \$US8.1 billion of net value through 2020. These savings help member utilities determine their investment in the grid, and potentially fuel smart grid technologies and greener energy policies.

This ISO is not the only one adopting this new optimization technology. Dr. Richard O'Neill of the US Federal Energy regulatory Commission estimates that worldwide adoption of MIP for unit commitment could save \$10 to \$200 billion per year⁴.

1.3.4 Example: Semiconductor manufacturing

This company develops, manufactures, and markets semiconductor, application-specific integrated circuit (ASIC), and interconnect technologies and products. In 2003, it completed a new 300mm semiconductor fabrication facility in the United States. The new US\$2.5 billion, 14,000-square meter facility handles the company's most advanced 90nm processes using copper

⁴ O'Neill, R. "It's Getting Better All the Time (with Mixed Integer Programming)," presented at INFORMS Annual Meeting, Washington, DC, October 14, 2008 (available from the author).

interconnects, silicon-on-insulator (SOI), and low-k insulation. It creates integrated circuits (ICs) for use in everything from mainframes, servers, and supercomputers to cellular phones and home gaming consoles.

The manufacturer identified opportunities to improve production efficiency at its new wafer fabrication facility. The site lacked a robust enough scheduling agent to effectively coordinate the multiple, intricate production processes to maintain high utilization of capital-intensive equipment. As a result, raw materials and manufacturing assets (such as the chemicals used in cleaning processes) were being inefficiently used, and time-sensitive procedures were being delayed.

The environment at the company's wafer fabrication plant included a staggering amount of complexity. This complexity is caused by the intersection of manufacturing options, customer requirements, and research and development (R&D) efforts in a fluid environment. Therefore, the organization wanted to implement a solution that would allow it to maximize throughput by directing various wafer fabrication activities on a near real-time basis. The solution needed to respect certain manufacturing, business, and process constraints for scheduling activities, while also enabling rapid customization and extensibility, with environments for testing and continuous improvement. Furthermore, the solution needed to not only recognize the dynamic nature of the environment, but also bring together the interests of various stakeholders.

The solution integrates a near-real time optimization-based scheduling system into the manufacturing execution system to drive more efficient management of site resources. Using IBM software, the solution models existing business rules and creates optimal manufacturing schedules that are then automatically executed. Facility staff can also analyze these schedules over time to identify patterns and recommend process improvements.

The company created and implemented a solution that was based on an integrated framework of business rules, optimization, and visualization. This solution bridges a manufacturing execution system (MES) and a real-time dispatching system. The ILOG JRules BRMS software forms a business rule management system (BRMS). The ILOG CPLEX software delivers high-performance with robust, flexible optimizers for solving linear, mixed-integer, and quadratic programming problems in mission-critical resource allocation applications. Likewise, the ILOG JViews software provides the organization with a family of Java editing and software development kits (SDKs) for building visually rich, highly interactive web-based user interfaces.

In combination, the new software creates robust schedules that update approximately *every five minutes*. The solution assesses the state of fabrication in near real time in terms of numerous components, lot priorities, process constraints, and overall throughput. After processing the inputs, the solution then publishes an optimal schedule for the tools under its control. The results are

communicated to operators by using a web-based graphical user interface (GUI) and a dispatching system. This delivery system allows the organization to provide course corrections and manage process constraints. Because of the high level of trust in the optimization solution, the results from the dispatching system are automatically executed. Additionally, a set of offline tools and archives of past solutions allow users to analyze specific instances of scheduling behavior. They can test new business rules and policies, tune the optimization model, and carry out what-if analyses.

The solution helps the company optimize management for a number of different toolsets that are used in the wafer fabrication plant, including furnaces, wet tools, and lithography tools. The software framework enables the organization to evaluate the state of fabrication in near real time. They can continually optimize fabrication processes in terms of throughput, manufacturing, and business priorities within the constraints that are imposed on the system. The new software framework also facilitates ongoing and continuous improvement. Users can investigate, analyze, and fine-tune issues, designs, and aspects of the rules and optimization behavior.

The new solution has the following benefits for the company:

- ▶ Innovates scheduling processes by using near-real-time business data to automate manufacturing processes.
- ▶ Obtain high utilization of capital-intensive equipment while improving product quality thanks to increased visibility into manufacturing efforts.
- ▶ Cuts production cycle times and increases manufacturing throughput by optimizing resource and tool utilization.

1.4 How can you make it happen

Chapter 2, “Introducing the ODM Enterprise solution” on page 23 introduces IBM ILOG Optimization Decision Manager (ODM) Enterprise, a platform for developing enterprise decision support that is based on optimization. The remainder of this book addresses in detail to how to create such solutions.

The ODM Enterprise product arose from a long and broad experience with applying optimization and advanced analytics in real-world settings for IBM. It can reduce the effort, complexity, and risk in creating and using analytic decision support. It does so by encapsulating the most important components of these kinds of solutions for easy configuration and customization. Thus, it enables building custom planning and scheduling solutions that can perfectly fit your organization’s business processes.

Although the mathematical models to be optimized are critical components of these solutions, they are not enough for effective decision support. ODM Enterprise provides more capabilities that together give business decision makers the functionality they need:

- ▶ *Data modeling and integration* to map enterprise information resources for use in decision support.
- ▶ *Business-focused analytics and visualizations* to give decision makers information and insights that instill confidence in the decisions.
- ▶ *Transparent mathematics with powerful solvers* to take advantage of subject matter expertise and analytics skills in your organization.
- ▶ *What-If? analysis* to test solutions for robustness under varying business circumstances.
- ▶ *Collaboration* to support enterprise business processes.
- ▶ *Deployment architectures* that range from a single workstation to distributed client-server designs to fit your information technology model.
- ▶ *Extensibility and customization* that use an open, standards-based architecture to achieve perfect fit to your requirements.

Developing effective analytical decision support solutions requires team effort with participants who represent different roles in the enterprise:

- ▶ *Decision makers and business analysts* represent the users of the solution. They set the objectives and functional requirements, especially regarding the business interface. They provide subject matter expertise on the business environment and the organization. They also give on-going feedback on usability as the solution development proceeds.
- ▶ *Analytics and optimization experts* translate business requirements into mathematical models (targets, limits, and choices) and handle formulation, implementation, and debugging. They also contribute subject matter expertise on the business, particularly the technical aspects that are needed for model formulation.
- ▶ *IT architects* create the integrated infrastructure within which the solution operates, which includes data integration, user interface, reporting, and deployment architecture.

If these skills are not readily available within your organization, you can rely on consultants who have the necessary expertise.

1.5 Why should your organization consider analytic decision support

This chapter has provided concrete benefits that other organizations have realized by using analytics and optimization. The rest of the book addresses how to implement optimization in much more detail. But it is worth remembering why you would want to use it:

- ▶ Smarter decisions

Analytics helps you achieve better performance for lower cost. You can extend your business expertise with fact-based decisions. You can find non-obvious choices that maximize your value or minimize your costs while also observing the many complex requirements and strategies of modern businesses. You can test limits and assumptions to develop robust solutions that perform well under various scenarios. And optimization produces quantifiable benefits for your bottom line.

- ▶ Faster decisions

Automating decision processes increases the speed of your responses in today's accelerating markets. Automating allows your managers and planners to focus on critical complexities rather than on routine issues.

- ▶ Effective enterprise decision support

IBM ILOG ODM Enterprise eases development of optimization-based custom planning and scheduling solution. It offers powerful decision management functions to enterprise-wide planning teams. These functions include an easily configurable business user interface, a central data server, and scalable optimization.

Using the high-level modeling tools gives you faster, lower cost, lower risk development and maintenance. They enable your analysts to code and validate your model with less time and effort than traditional programming languages. By increasing the transparency of your model, they make maintaining and upgrading your system easier and more reliable.

- ▶ Turn information into action

Advanced analytics solutions use the investment that you have in enterprise information technology and enable you to use that information in effective decision-making.



Introducing the ODM Enterprise solution

This chapter introduces the ODM Enterprise platform for developing and deploying custom decision support applications that are based on advanced analytics and optimization. ODM Enterprise facilitates the process of transforming a business problem into an optimization model and then into an effective business application. It includes an example application, the ContainerYard, used throughout this book to illustrate the implementation process. It also describes the major architectural components of an ODM Enterprise application. The various roles that client and provider personnel play in the development and use of the application are described. The alternative approaches to developing decision support applications are also compared.

This chapter includes the following sections:

- ▶ From optimization model to business application
- ▶ The ContainerYard: An ODM Enterprise application example
- ▶ Application concepts and parts of an ODM Enterprise solution
- ▶ Optimization concepts and parts of an ODM Enterprise solution
- ▶ Roles and expertise
- ▶ Advantages of ODM Enterprise solutions
- ▶ Conclusion

2.1 From optimization model to business application

Optimization alone is usually insufficient to support effective decision-making. Rather, more capabilities are needed to turn an optimization model into a business application. This chapter focuses on these capabilities and the IBM product that implements them, the *IBM ILOG Optimization Decision Manager (ODM) Enterprise*.

An optimal result is not enough for effective business solutions because of these factors:

- Decision makers need *decision support*

Decision makers will not use any analytics tool unless they trust it. Trust arises from understanding the recommendations the decision support application makes and the reasons behind them. Trust is particularly important when the recommendations depart from familiar patterns, which can happen when the business enters uncharted waters. A decision maker will frequently want to test alternatives to the recommendation and to understand the trade-offs that are involved when making best use of limited resources. Also, conflicts often arise among the various targets and limits. These conflicts can result in failure of the tool to find a solution that satisfies all the criteria. It is seldom acceptable to simply announce such failures, especially in mission critical situations. Rather, the decision maker will usually want to relax some of the criteria in a controlled way until an acceptable solution is found. Finally, in a dynamic business environment, the decision makers must revisit and revise plans as conditions change.

- Decision makers need *good decisions*

It might seem paradoxical to imply that *optimal* (that is, *the best*) decisions might not be *good* decisions. However, remember that optimality is measured with respect to a specific model (targets, limits, and choices). Any model represents only an approximation to the real world. You strive to make your models as accurate as possible. However, the limits of the modeling process will almost certainly lead to mismatches that can render the recommended decisions unimplementable. Thus, a good decision support tool needs to provide capabilities to ensure that optimal decisions are also good decisions.

Often, an optimization model represents both *objective criteria*, such as physical laws or government regulations that must be enforced, and *subjective criteria*, such as customer service targets. If the optimization model proves to have no solution, the decision maker might need to relax some of the subjective criteria to attain feasibility.

Again, recalling that optimality is measured regarding a specific model, that model itself might become inaccurate because of the uncertainties that

govern the evolution of the business. In that case, the optimal solution that is based on a nominal forecast might perform poorly under off-nominal conditions. This situation is called a *fragile solution*. In the face of such uncertainties, decision makers usually prefer a *robust solution*, which performs well under various future conditions. In other words, they might willingly give up some of the potential value in the nominal solution to gain better performance under uncertain conditions.

- ▶ Decision makers need organizational *decision processes*
Only in the smallest organizations is there a single decision maker. In most organizations, multiple individuals with distinct roles participate in decision-making processes. A decision support tool needs to recognize these roles and provide capabilities that support the organization's decision processes. As an organization grows, the amount of data and the number of people that support decision making also grow. Therefore, the decision support tool must scale comfortably to accommodate that growth. Decision support applications that use corporate information technology must conform to the organization's IT governance. Decision support applications for collaborative decision-making processes must enable multiple users to interact with it and each other, sharing information and results.
- ▶ Decision makers need to respect the *people* and *processes* in the organization
Adopting a decision support application changes the way that the organization works. Users need to learn to work in new ways, and they need to trust the new application. In addition to the software engineering tasks, you must accomplish organizational tasks in the course of developing and deploying the new application. Ultimately, success or failure often depends on the usability and user acceptance of the new tool.

Table 2-1 summarizes the reasons.

Table 2-1 Why optimal solutions are not enough for effective decision support

It is about decision support ▶ Recommend and explain ▶ Alternatives and trade-offs ▶ Relaxations and conflicts ▶ Replanning	It is about good decisions ▶ Subjective business criteria versus optimization model ▶ Optimality versus fragile solutions ▶ Uncertainty versus strict optimality
It is about corporate decisions ▶ Data management and scalability ▶ IT requirements ▶ Multiple collaborative users	It is about people and processes ▶ Change management ▶ Management consulting and process analysis ▶ Usability and user acceptance

2.2 The ContainerYard: An ODM Enterprise application example

To give you a better idea of what a decision support application looks like, consider the following example of an ODM Enterprise-based planning application. This application, called the *ContainerYard*, is intended to optimize a delivery process in the supply chain of a car manufacturer. It derives from a real project that was implemented in 2012.

This example is used throughout the remainder of this book to illustrate various concepts and methods.

For more information about the implementation of this example, see Chapter 4, “Transforming business requirements into an ODM Enterprise application” on page 111.

For instructions on how to download, install, and use this example application, see Appendix C, “Additional material” on page 329.

2.2.1 Supply chain process

The business problem that this application addresses is managing the delivery of parts to a manufacturing plant. The deliveries are shipped to the plant in containers that can hold many different kinds of parts. Some of these parts are needed immediately, and others are not. Because the warehouse has limited space, the logistics manager wants to reduce the number of unneeded parts that get unloaded from the containers.

For an overview of the supply chain process, see Figure 2-1.

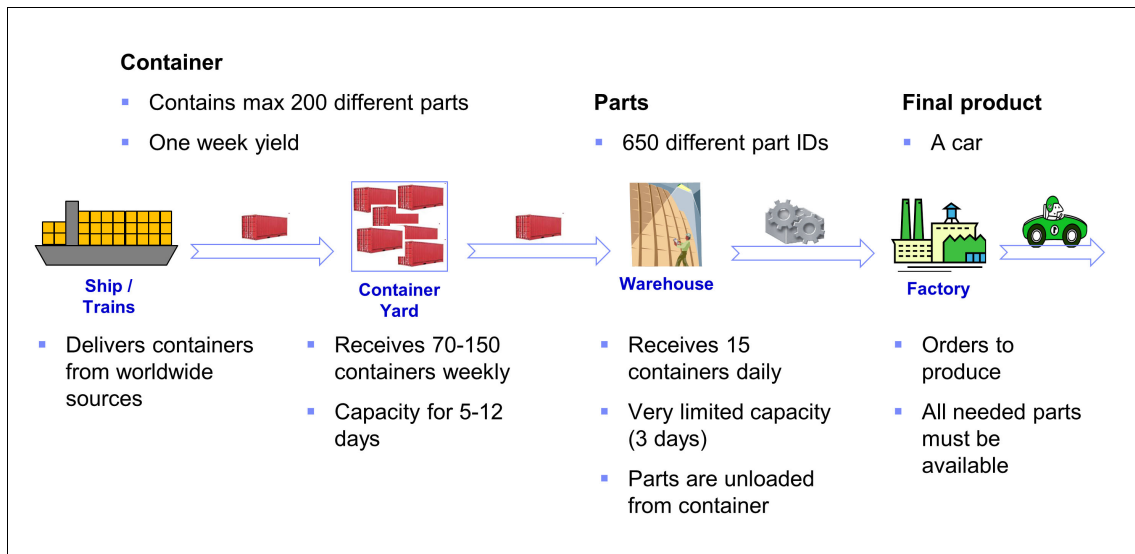


Figure 2-1 Overview of the supply chain process used in the ContainerYard application example

This process works as follows, from left to right:

1. Ships, trucks, or trains deliver containers.
2. All containers temporarily remain in a container yard.
3. Every day, the logistics manager selects some of these containers for transport from the yard to a warehouse.
4. In the warehouse, workers open and unload the containers, and store the parts.
5. When needed, workers load the parts for transport from the warehouse to the manufacturing plant.
6. Assembly of the final product (cars) using the parts occurs in the plant.

2.2.2 Planning process

Every working day, the logistics manager must plan the following delivery process: *Which containers must be taken from the container yard to the warehouse?* Currently, this process is planned manually. Every evening, a (human) planner gathers all necessary information and determines a selection of containers to move from yard to warehouse on the next day. The core of this planning process is called the *planning problem*, specified with input data or information, choices, limits, and planning targets.

Data

The following list comprises the main data or information that the plan must take into account:

- ▶ The schedule of container arrivals
- ▶ The contents of each container
- ▶ The schedule of car production
- ▶ The items that are needed to produce each car

Choices

In this supply chain process, a daily decision is made about which containers to transfer from the yard to the warehouse on the following day.

Limits

There are several constraints that must be considered:

- ▶ All necessary parts must arrive at the plant before a car can be produced.
- ▶ The warehouse has limited storage capacity (roughly enough for items that serve three days of production).
- ▶ Every day, a minimum number of containers must move from the yard to the warehouse.
- ▶ A container that is scheduled to arrive in the future can move only after it has arrived in the yard.

Targets

The planning must achieve mainly the following goals:

- ▶ Minimize *bycatch*: Avoid putting parts into the warehouse that are not needed in the plant. This target minimizes unnecessary inventory at the warehouse.
- ▶ First-In-First-Out (FIFO): Containers that have already been in the yard for a long time should move to the warehouse before those containers that arrived more recently.

2.2.3 Solution: An ODM Enterprise-based software application

This section presents a solution for this planning problem using ODM Enterprise. The Container Yard example illustrates the components of the solution as they are explained.

Roles

Most organizations have multiple people who are involved in the decision maker role. The ContainerYard application, however, has a single individual who acts as

the decision maker, namely the logistics manager. This person serves as the business user for this application.

The user of this application does the daily planning. Every morning, he must decide which containers to take from the yard to the warehouse.

Planning cockpit

The ODM Enterprise Client provides the *planning cockpit* (or user interface) for the business users. The logistics manager uses the ODM Enterprise Client to do the planning.

Input data

Figure 2-2 shows the input data tables of the ContainerYard application. Note that this example uses artificial data.

ODM Enterprise applications usually import data from external resources (such as an ERP system), but the user can also enter some of the data manually.



Figure 2-2 Input data tables of the ContainerYard application

The input data consists of:

Parts	All the car parts the application must handle
Containers	The list of containers that are currently in the yard or that will arrive in the yard
Contents	The parts that each container holds
Demand	The list of parts that are needed in the factory
Parameters	General settings for the application

Optimization

As soon as the user has all the input data ready, he can start the optimization. Using the ODM Enterprise Client, he does this through a menu entry:

Scenario → Solve.

Then the optimization starts. The user is notified about the optimization and its status in the Solve Progress window as depicted in Figure 2-3 on page 31. In this particular example, the status window shows that the solve completed after 65 ms. When the solve requires a longer time, this window shows the upper and lower bounds on the goals. These bounds approach each other as the computation proceeds. The user can then choose to terminate the solve with a known error bound on the value of the optimal solution. This gives the user a performance guarantee on the trade-off between computation time and accuracy of the solution.

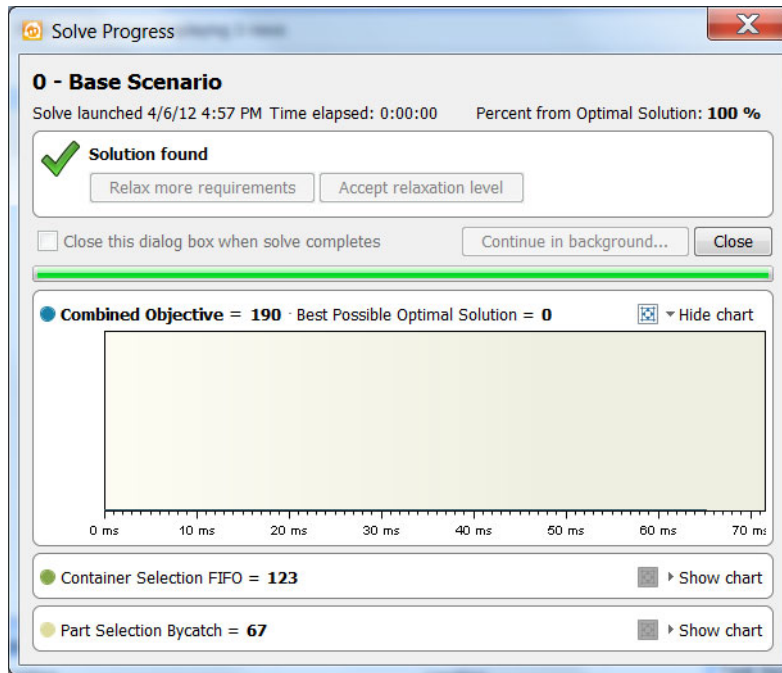


Figure 2-3 The Solve Progress window

Results

After the optimization is complete, the user can analyze the results. The first thing to look at is usually the Goals window, which is shown in Figure 2-4 on page 32. It shows the values for the different targets or goal expressions:

- ▶ 123 measures the satisfaction of the first-in/first-out criterion.
- ▶ 67 measures the unneeded items.
- ▶ For the selected goal (FIFO in this case) the contributions of elements to which it refers (containers “c2” and “c3”).

Goal Name	Value	Active	Importance Factor	Constrained
Container Selection FIFO	123	☒	1	
Part Selection Bycatch	67	☒	1	

Name	Value
Constraints	Container Selection FIFO
Constrain max to	
Constrain min to	
With priority	
Bound Searches	
Best bound	121
Worst bound	183
Ignoring priorities under	Ignored

Name	Value
fifo	123
for <"c2" <"2012-03-02"> 0>	61
for <"c3" <"2012-03-03"> 0>	62

Figure 2-4 The Goals window

For more details about the optimization results, the application provides more views (Figure 2-5):

- ▶ A list of chosen containers and the parts each holds. Again, the application chooses *whole containers*, not parts out of containers. This result view shows the two chosen containers (“c2” and “c3”), together with their whole content.
- ▶ A Pivot table that shows the supply of parts versus demand.

Container	Part	Quantity
c2	Airbag	8
c2	Bumper	3
c2	Carpet	3
c2	Door	6
c2	Engine	1
c2	Flywheel	10
c2	Gearbox	9
c2	Horn	7
c2	Ignition	9
c2	Kompressor	4
c3	Airbag	2
c3	Bumper	2
c3	Carpet	6
c3	Door	10
c3	Engine	5
c3	Flywheel	1
c3	Gearbox	5
c3	Horn	5
c3	Ignition	3
c3	Kompressor	3

Part	Demand	Supply	Excess ...	Shortage
Airbag	1	10	9	0
Bumper	2	5	3	0
Carpet	5	9	4	0
Door	0	16	16	0
Engine	4	6	2	0
Flywheel	8	11	3	0
Gearbox	2	14	12	0
Horn	3	12	9	0
Ignition	6	12	6	0
Kompres...	4	7	3	0

Figure 2-5 Optimization result details

2.2.4 Planning scenarios

The ODM Enterprise application platform provides a capability for scenario-based *what-if analysis*. As outlined in 2.3.1, “ODM Studio: The planning cockpit” on page 40, a scenario essentially contains input data, optimization parameters, and results. The ODM Enterprise Client provides a window called Scenarios Overview that allows the user to work with Scenarios. The user can create, duplicate, edit, rename, organize, import, and export Scenarios. Figure 2-6 shows how this window looks in the ContainerYard application. It shows four different planning scenarios:

- ▶ 0 - Base Scenario
- ▶ 1 - Container Selection FIFO
- ▶ 2 - Part Selection Bycatch
- ▶ 3 - Max number of containers

The planning scenario in the last section is scenario “0 - Base Scenario”. This one is the *base scenario*.

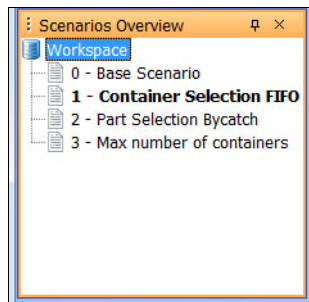


Figure 2-6 Workspace navigation

The following example scenarios are intentionally simple. This allows you to easily understand all the data and aspects of the scenarios, and enables the user to find the optimal solution (container choices) on his own. Real scenarios are always much more complex.

First-In/First-Out (FIFO)

As defined in 2.2.2, “Planning process” on page 27, one of the planning goals is to minimize *FIFO* when choosing containers. Containers that have already been in the yard for a long time should be taken before those containers that have arrived more recently.

Figure 2-7 shows the input data of Scenario “1 - Container Selection FIFO”:

- ▶ Three containers, arriving at different dates, “c3” is first to arrive.
- ▶ Contents: the contents of the three containers are identical.
- ▶ Two demands, such that any one of the containers can fulfill all the demand.

The screenshot shows three data tables from a software interface:

- Containers**: A table with 3 rows. Columns: ID, Arrival. Data: c1 (2012-03-03), c2 (2012-03-02), c3 (2012-03-01).
- Contents**: A table with 6 rows. Columns: Container, Part, Quantity. Data: c1 (Airbag: 10, Bumper: 10), c2 (Airbag: 10, Bumper: 10), c3 (Airbag: 10, Bumper: 10).
- Demand**: A table with 2 rows. Columns: Part, Date, Quantity. Data: Airbag (2012-03-03, 5), Bumper (2012-03-03, 5).

Figure 2-7 Input data of the scenario “1 - Container Selection FIFO”

With respect to all other input tables shown in Figure 2-2 on page 29, the current scenario, “1 - Container Selection FIFO”, does not differ from the base scenario.

Given all this input data, what would be the optimal container choice?

Because the demands can be fulfilled with any *one* of the containers, there is no need to choose multiple containers. Compare the three possible choices as shown in Table 2-2.

Table 2-2 Different selection possibilities for scenario “1 - Container Selection FIFO”

Container	Constraints	Goal FIFO	Goal Bycatch
c1	Fulfils all ^a	Arrives March 3rd	10 parts ^b
c2	Fulfils all	Arrives March 2nd	10 parts
c3	Fulfils all	Arrives March 1st	10 parts

a. Container “c1” arrives on the planning day, March 3rd, as defined in the “parameters”. Therefore, it can be chosen because it does not arrive in the future

b. Bycatch is simply calculated here. For more information, see “Bycatch” on page 36.

Container “c3” is the optimal choice because it optimally fulfills the FIFO criterion. It was the first container to arrive in the yard and thus must be preferred over all other containers. It is equivalent to all other selections that involve all other goals and requirements. This is exactly the solution that the optimization finds automatically as shown in Figure 2-8.

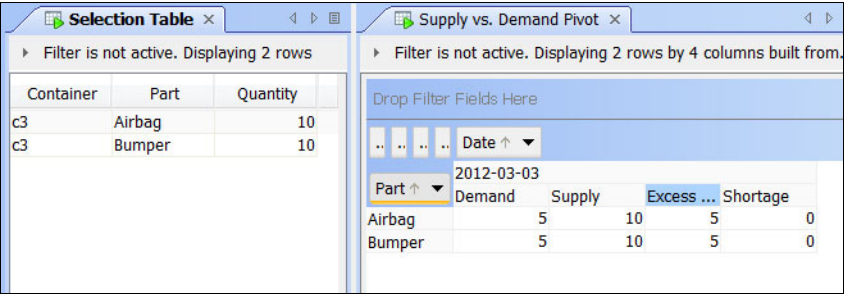


Figure 2-8 Details on optimized solution for the scenario “1 - Container Selection FIFO”

Look again at the Goals view of the ODM Enterprise client that is shown in Figure 2-9. The computed value for Bycatch is 10, and the value for FIFO is 60, the duration in days between March 3rd, 2012, and January 1st, 2012. The optimization uses this duration to evaluate different choices concerning the FIFO criterion.

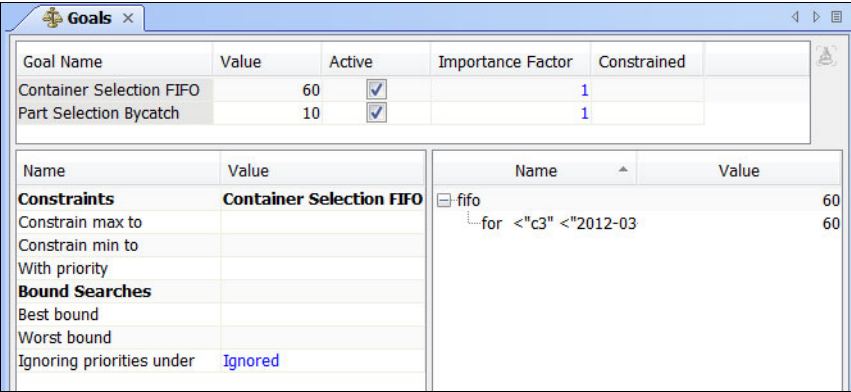


Figure 2-9 Solution for the scenario “1 - Container Selection FIFO” in the Goals view

Bycatch

Now have a closer look at the second planning target: *Bycatch* minimization. As defined in Section 2.2.2, “Planning process” on page 27, Bycatch minimization means avoiding putting parts into the warehouse that are not needed in the plant. Figure 2-10 shows the input data for this scenario, “2 - Part Selection Bycatch”.

Containers		Contents			Demand		
Filter is not active. Displaying 3 rows		Filter is not active. Displaying 6 rows			Filter is not active. Displaying 2 rows		
ID	Arrival	Container	Part	Quantity	Part	Date	Quantity
c1	2012-03-02 (2012-03-03)	c1	Airbag	8 (10)	Airbag	2012-03-03	5
c2	2012-03-02	c1	Bumper	5 (10)	Bumper	2012-03-03	5
c3	2012-03-02 (2012-03-01)	c2	Airbag	5 (10)			
		c2	Bumper	7 (10)			
		c3	Airbag	5 (10)			
		c3	Bumper	8 (10)			

Figure 2-10 Input data of the scenario “2 - Part Selection Bycatch”

This figure illustrates another convenient feature that the ODM Enterprise Client provides: Highlighting differences in planning scenarios. Here, the user has chosen to compare the differences between the current scenario “2 - Part Selection Bycatch”, and a *reference scenario* (designated by the user), “1 - Container Selection FIFO”. Thus, the view shows the differences between the Bycatch scenario and the FIFO scenario:

- ▶ Containers: Again there are three, but now they all arrive at the same date (March 2nd). The reference scenario values are shown in parentheses in the data table view; for example: “2012-03-03” in the reference scenario versus “2012-03-02” in the current scenario.
- ▶ Contents: They are now different in the three containers:
 - Eight airbags and five bumpers in “c1”
 - Five airbags and seven bumpers in “c2”
 - Five airbags and eight bumpers in “c3”.
- ▶ Demands: identical to the FIFO scenario: Five items of each car part.

Again, all the demands can be fulfilled with any *one* of the containers. Table 2-3 compares the three possible choices.

Table 2-3 Different selection possibilities for scenario “2 - Part Selection Bycatch”

Container	Constraints	Goal FIFO	Goal Bycatch
c1	Fulfills all	Arrives March 2nd	Three parts (surplus airbags)

Container	Constraints	Goal FIFO	Goal Bycatch
c2	Fulfils all	Arrives March 2nd	Two parts (surplus bumpers)
c3	Fulfils all	Arrives March 2nd	Three parts (surplus bumpers)

Container “c2” is the optimal choice because it optimally fulfills the Bycatch criterion and is equivalent to all other selections concerning all other goals and requirements. This is exactly what the optimization finds as shown in Figure 2-11.

Container	Part	Quantity
c2	Airbag	5
c2	Bumper	7

	Demand	Supply	Excess ...	Shortage
Airbag	5	5	0	0
Bumper	5	7	2	0

Figure 2-11 Solution for the scenario “2 - Part Selection Bycatch”

Combined goals

The previous two sections addressed the two *different planning goals*, FIFO and Bycatch, and how the application handles them. In fact, the mathematical optimization always looks at a *combination* of the defined planning goals. In this example, it actually considers the sum of the two subgoals, FIFO and Bycatch. The base scenario shows this process in the Solve Progress window that is shown in Figure 2-3 on page 31.

This view shows that at the end of the optimization, the *Combined Objective* equals 190, 132 for FIFO, and 67 for Bycatch ($190 = 123 + 67$). The Goals window (Figure 2-4 on page 32) shows where these values come from. The line for Bycatch shows only the total of 67, but for FIFO, the value 123 consists of the sum of $61 + 62$, with:

- ▶ 61 for container “c1”, for March 2nd (61st day of the year)
- ▶ 62 for container “c2”, for March 3rd (62nd day of the year)

For more information about how goals can be configured and adjusted, see “Adding and configuring goals” on page 155.

2.3 Application concepts and parts of an ODM Enterprise solution

All the useful mathematical optimization machinery is useless without a complementary software application that makes it available to business users. This section more fully describes the components that comprise the applications built by using ODM Enterprise.

The ODM Enterprise platform grew out of experience with implementing many optimization and planning applications by using the ILOG CPLEX Optimizer. In these projects, developers found themselves implementing similar functionality and components again and again. These projects tended to include these major concepts and components:

- ▶ *A planning cockpit*: A graphical user interface (GUI) with which the business user can create plans and work interactively with the optimization.
- ▶ *A scenario repository*: A database that stores planning data (input data, optimization parameters, results, other information) corresponding to one or more *scenarios*. The scenarios, or cases, represent potential variations of the underlying business problem.
- ▶ *An optimization computation engine*: A managed environment that takes care of the optimization computation.
- ▶ *An integrated development environment (IDE)* that supports development of the application.

Figure 2-12 on page 39 shows the respective ODM Enterprise components. The actual components that are needed depend on the chosen deployment architecture. For more information, see 2.3.2, “Deployment architectures” on page 42. Starting on the right, the business users interact with the planning cockpit, through the ODM Enterprise Planner or Client or through a custom GUI or a batch process. ODM Enterprise provides application programming interfaces (APIs).

In the center of the figure, the planning application gets data management and optimization services from the ODM Enterprise Data Server (or from ODM Enterprise Planner in a desktop architecture) and from the ODM Enterprise Optimization Server or Engine.

The left portion of the figure represents the development tools that are used to build the application.

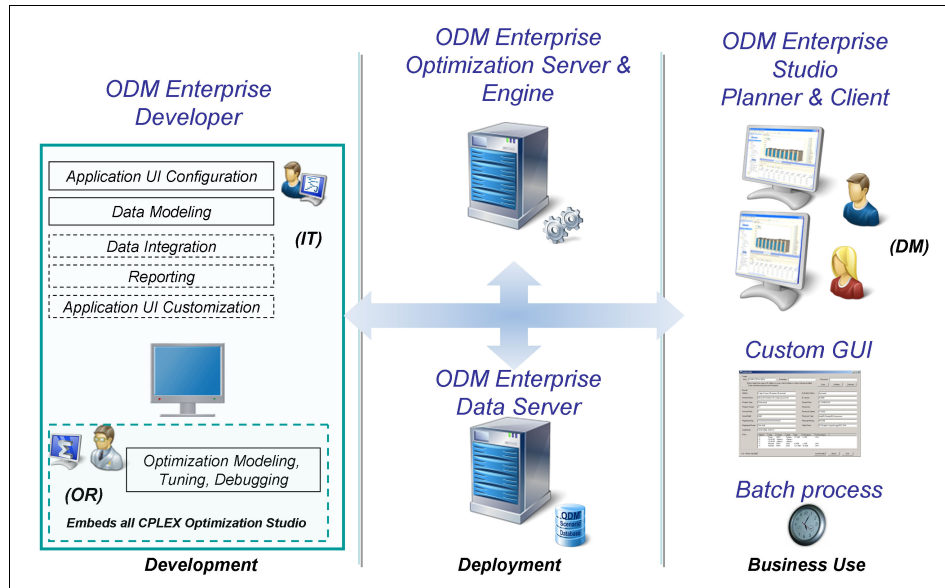


Figure 2-12 The major components of the ODM Enterprise application platform

Table 2-4 shows what functions these components provide to the major user roles, as defined initially in 1.4, “How can you make it happen” on page 20.

Table 2-4 ODM Enterprise components and their functions

Component	Provides	To
<i>ODM Studio</i> : The planning cockpit	▶ Business-focused analytics and visualizations ▶ What-if analysis	Decision makers and business analysts (in Line-of-Business)
<i>ODM Enterprise Data Server</i> : The scenario repository	▶ Multi-user enterprise deployment ▶ Data sharing ▶ What-if scenario sharing	All users
<i>ODM Enterprise Optimization Server/Engine</i> : The optimization solvers	▶ Transparent mathematics with powerful solvers ▶ Deployment architectures	All users

Component	Provides	To
<i>ODM Enterprise IDE:</i> The development environment	▶ Data modeling ▶ Optimization modeling ▶ Extensibility and customization	▶ Analytics and optimization experts ▶ IT architects

All these components are described in more detail in later chapters.

Remember: Depending on the deployment architecture that you chose, the components might have different names and configurations. For example, in an isolated workstation deployment, the business user interface is called the ODM Enterprise Planner. There is no distinct data server, and the ODM Enterprise Optimization Engine takes the place of the ODM Enterprise Optimization Server. For more information, see 5.1, “ODM Enterprise editions” on page 166, and 5.2, “Deployment and system design options” on page 169.

2.3.1 ODM Studio: The planning cockpit

The ODM Studio provides the planning cockpit (or user interface) for the business users (in the organization’s business division or *Line-of-Business* or *LoB*). It provides a rich set of functions that typical business users such as planners, or other decision makers often need in a planning application:

- ▶ Rich, configurable GUI
- ▶ Data integration (for example, through import or export)
- ▶ Scenario-based planning
- ▶ Data editing and analysis
- ▶ Optimization execution and control
- ▶ Business goals and plan overview
- ▶ What-if analysis: KPI comparison
- ▶ Integration with office applications
- ▶ Controlled constraint relaxation
- ▶ Team collaboration
- ▶ Visualization, tables, and charts
- ▶ Tight integration with model and extension development, like custom graphics

Figure 2-13 shows a typical configuration of the ODM Studio, showing how some of these features look.

The concept of planning *scenarios* is key to understanding the ODM Enterprise platform. A scenario consists of these components:

- ▶ Input data (such as master data or transactional data)
- ▶ Parameters
- ▶ Optimization results
- ▶ Messages, errors, and so on.

Scenarios are (functionally) independent of each other. If the user changes something in a scenario, it does not affect any other scenario.

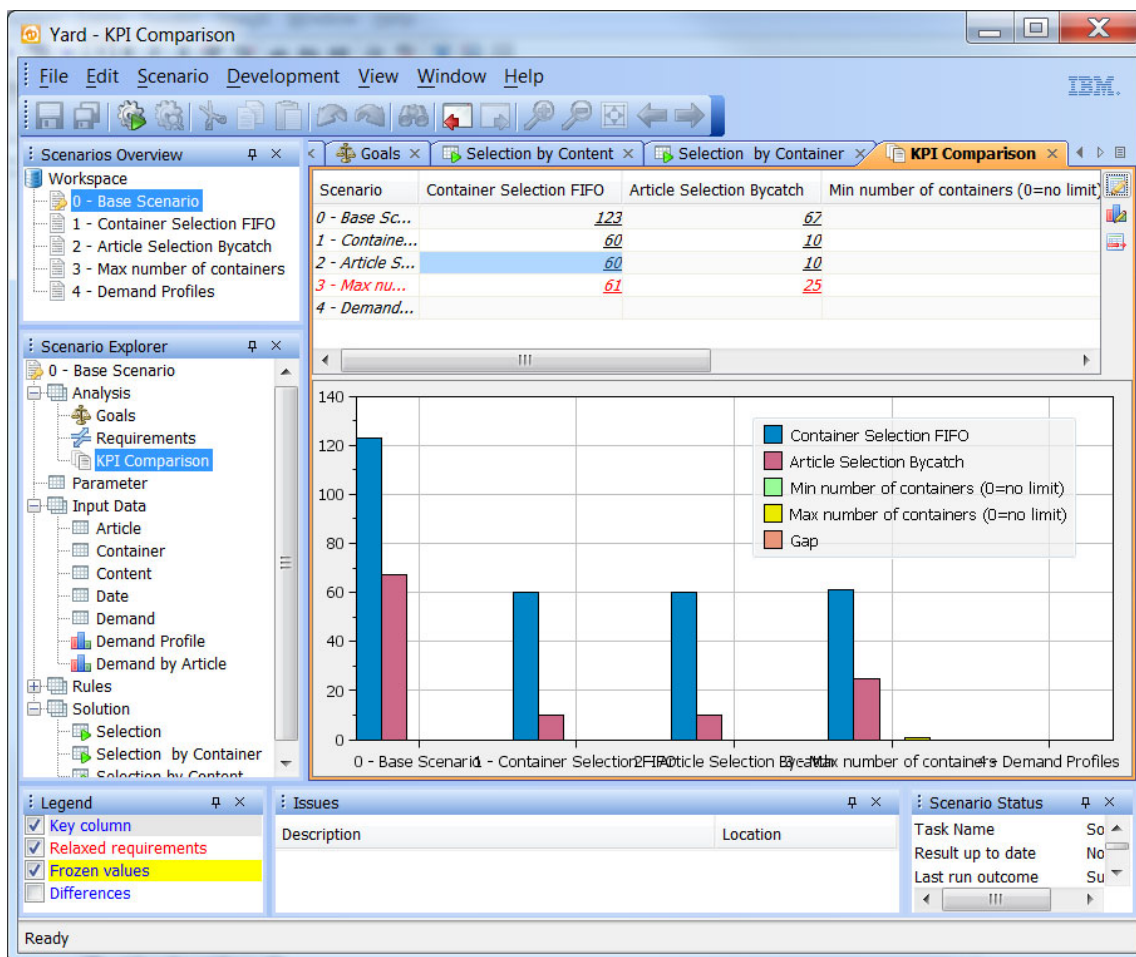


Figure 2-13 The ODM Enterprise Client

2.3.2 Deployment architectures

ODM Enterprise offers several deployment architectures that can suit different access and computational needs, as shown in Figure 2-14.

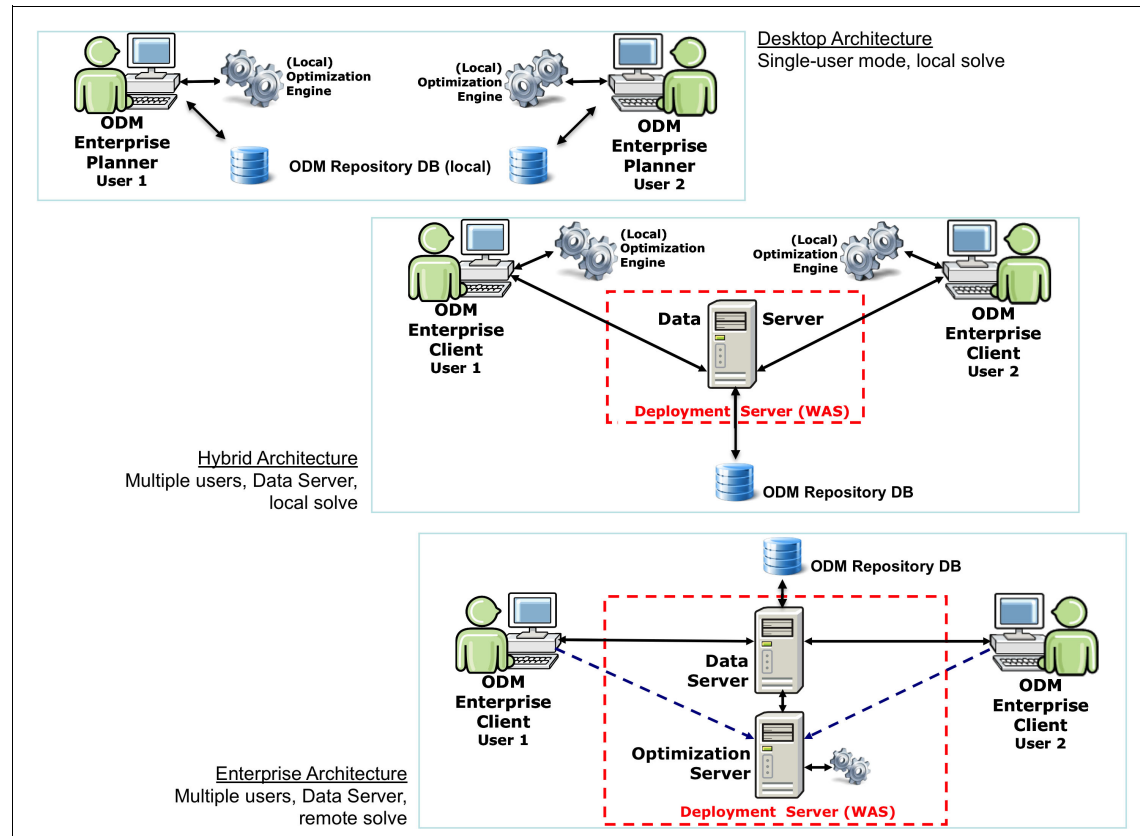


Figure 2-14 ODM Enterprise deployment architectures

► Desktop

In this configuration, all data storage and computation takes place on individual workstations, and business users cannot easily share scenarios. This architecture best suits applications that require modest computational effort for the optimization solves and no need for collaboration among users. It has the advantages of simplicity and minimal effect on the enterprise IT infrastructure.

In this configuration, the ODM Enterprise Planner provides the planning cockpit and manages the scenarios, and the ODM Enterprise Optimization Engine provides the solver.

► Hybrid

As the name implies, this configuration falls between the Desktop and Enterprise architectures. It best suits applications that require short- to medium-running optimization solves, but extensive collaboration among users. It offers flexibility but still requires conformity with the enterprise IT standards.

In this configuration, the ODM Enterprise Client provides the planning cockpit. The ODM Enterprise Data Server holds the scenarios, and the ODM Enterprise Optimization Engine (installed on each workstation) provides the solver.

► Enterprise

This configuration implements a full client-server architecture in which data storage and computation take place on servers. This architecture best suits applications that require long-running optimization solves or multiple solves in parallel, and extensive collaboration among users. It has the advantages of using solver capacity across multiple users and centralized platform management. However, it requires conformity with the enterprise IT standards.

In this configuration, the ODM Enterprise Client provides the planning cockpit. The ODM Enterprise Data Server holds the scenarios, and the ODM Enterprise Optimization Server provides the solver.

2.4 Optimization concepts and parts of an ODM Enterprise solution

Linear Programming, one of the widest applied techniques in Operations Research, is characterized by ease of building models, and availability of efficient solution algorithms.¹

This statement, taken from a recently published textbook on Operations Research (OR), certifies the ODM Enterprise approach. At the core of an optimization application, there is a mathematical Optimization Model together with the Optimization Engine that implements those highly efficient algorithms.

¹ S. Nickel et al., *Operations Research*, Springer Verlag, Berlin Heidelberg 2011, ISBN 978-3642226236.

2.4.1 Optimization model

An optimization model represents a mathematical formulation of a business planning problem, such that an optimization engine can interpret and solve the problem.

The optimization model consists of a carefully defined set of relationships among the targets, limits, and choices, expressed as mathematical equations and logic statements. The solution of this model consists of the best set of choices (regarding the defined targets and satisfying the limits) to a business planning or scheduling problem.

The model defines mathematical entities and relationships among those entities. Those entities are abstract in the sense that they are not bound to specific numerical values. Rather, they are bound to data that is added later when solving the model. This principle of *model-data separation* is considered a best practice.

Remember that any model is an approximation that does not represent or implement reality exactly. Every mathematical model uses abstractions, and many use simplifications. A combination of business experience and modeling expertise determines which abstractions and simplifications are acceptable in an application.

In ODM Enterprise-based applications, the model is usually defined by using a special language called *Optimization Programming Language (OPL)*. For more information about optimization modeling, including OPL source code examples, see 4.2.4, “Creating a prototype with optimization” on page 138. OPL provides high-level syntax and structures that can transparently and compactly specify the mathematical relationships among the entities that constitute the model. These specialized features enable creating the optimization model with much less effort and with greater reliability than traditional programming languages.

2.4.2 Optimization engines

An optimization engine is a software component that implements specialized algorithms. These algorithms have been invented and tuned to efficiently solve a large variety of different planning problems.

The following algorithmic technologies have proven particularly useful for real-world applications:

- ▶ Mathematical Programming (MP)
- ▶ Constraint Programming (CP)

Mathematical Programming

Mathematical Programming (MP) is a branch of Applied Mathematics. It finds mathematical optimality by applying equations to a large search space of potential solutions. Algebra and analytical geometry form the core. They are assisted by many algorithms and heuristics (rules of thumb) that take advantage of concepts and methods developed in both mathematics and computer science over the past 70 years.

Linear equations are the dominant language of optimization, which is why the term Linear Programming (LP) is used interchangeably when talking about MP. However, some models also require the use of quadratic equations ($ax^2 + bx + c = 0$) that are not linear. More importantly, most business problems can be solved only if you can adjust all or some of the solution recommendations to whole numbers. This adjustment is needed because in the real world, most machines, people, and activities come in whole numbers. Whole numbers even represent discrete business decisions, such as whether to open up a particular plant, or how many trucks to buy.

Clarification: The solver used in ODM Enterprise, called the CPLEX Optimizer, implements several algorithms for solving linear programs (LP), mixed-integer programs (MIP), quadratic programs (QP), and mixed-integer quadratic programs (MIQP). These algorithms have a 20 year record of continuously improving performance. They have reduced solution times by factors of over 1000, not to mention another factor of 1000 speed-up because of improved processor hardware). As a result of these efforts, optimization problems that would have taken days of computation time two decades ago can now be solved in seconds. This improvement has made optimization a practical tool for business.

Constraint Programming

The second major technology for optimization is Constraint Programming (CP), which is a branch of Computer Science (also known as Logic Programming). CP has roots in the Artificial Intelligence (AI) and Expert System efforts of the 1980s. It is also a development of the move to object-oriented approaches to software development in the 1990s. Its approach to optimization is completely different from the approach of MP. CP finds feasible solutions by rapidly eliminating possibilities, zeroing in on feasible combinations of resource assignments and activities. CP is used when there are too many possible combinations or too many individual constraints that affect sequence and timing for MP to find a solution in an acceptable time frame. This is called coping with the *combinatorial explosion* of possibilities. CP approaches optimality by using a continuous improvement process that is based on the systematic comparison of new solutions to a base feasible solution. As a result, CP can find good solutions to large, intricate scheduling and routing problems quickly. It can find solutions in

minutes that would take MP hours or even days, assuming that an MP model could even find a solution.

Tip: When you run either a CP or MP model, you can instruct the solving engine to give you its best result after a set time limit. Often the extra time the engine spends searching provides only a small additional improvement.

IBM ILOG optimization engines

The following solver engines are available through IBM ILOG:

- ▶ The IBM ILOG CPLEX Optimizers for solving MP problems (for example Simplex, Barrier, and Mixed-Integer), specifically LPs, MIPs, and some QP problems.
- ▶ The IBM ILOG CPLEX CP Optimizer for solving CP problems (IBM ILOG's second-generation CP engine).

For more information, see the IBM ILOG online library at:

<http://www.ibm.com/software/websphere/products/optimization/library/>

The library includes white papers and tutorials on all the products and approaches that are introduced here.

There are several ways in which developers can use these solvers. Most important for ODM Enterprise is the OPL modeling language. Users of OPL can select whether to use the CPLEX Optimizer for mathematical programming or the CPLEX CP Optimizer for constraint programming. Beyond that simple selection, invocation of the solver takes place automatically in OPL and ODM Enterprise.

The solvers also support application programming interfaces, called *Concert Technology*, in the high-level programming languages C++, Java, and the Microsoft .Net family (C# and Visual Basic). Java Concert can be used in ODM Enterprise within a custom task when the developer needs more control of the optimization algorithms than OPL can provide. Finally, the solvers also support a *callable library* of routines in the C language, but these are not usable within ODM Enterprise.

2.4.3 CPLEX Optimization Studio

The IBM ILOG product family provides a rich set of optimization development tools. These tools are all built into a common IDE, the CPLEX Optimization Studio:

- ▶ Project navigator
- ▶ Problem and solution browser

- OPL editor
- Model outline
- Error list
- Interactive scripting and scripting log
- Conflict and relaxation views
- Engine log
- Statistics and profiler
- Debugger
- Integrated help

See Figure 2-15 for an example of the user interface of this IDE, including some of the features mentioned.

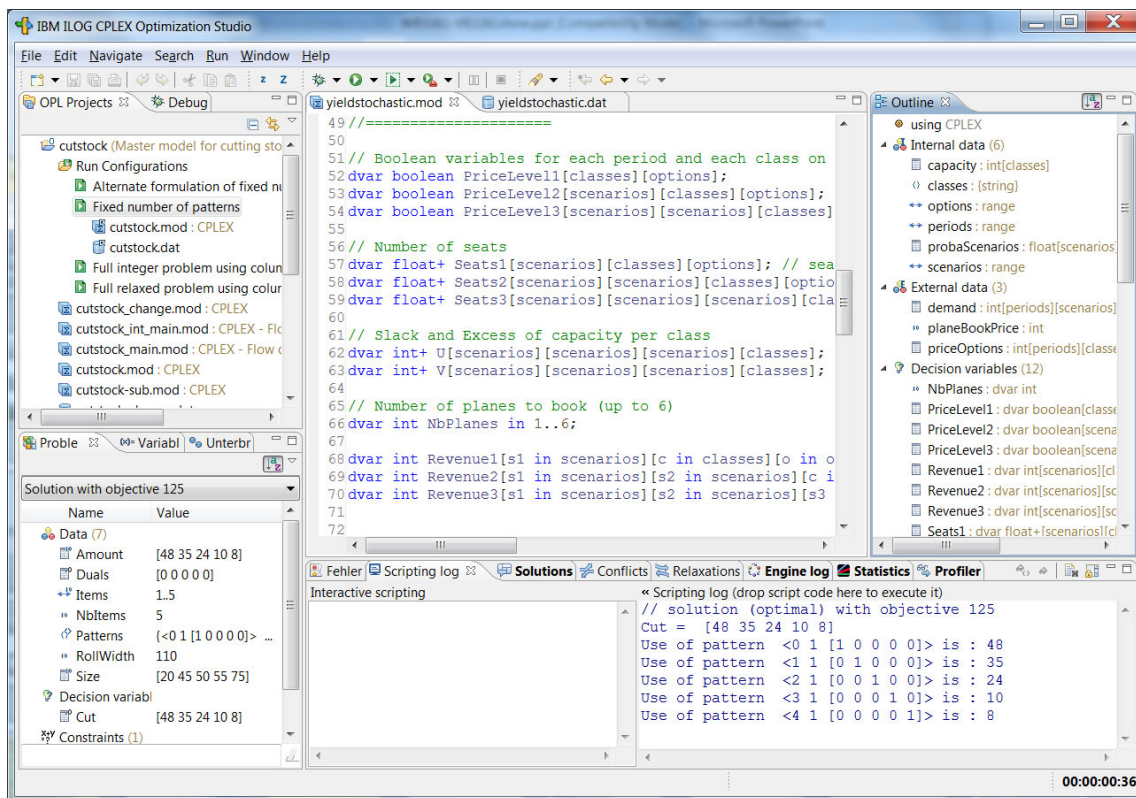


Figure 2-15 The user interface of the CPLEX Optimization Studio

ODM Enterprise makes these functions available to developers through the Developer Edition. This edition also includes more functions for data modeling, data integration, application user interface configuration, reporting, and customization with Java.

2.5 Roles and expertise

Each ODM Enterprise-based application starts with a project. Each project involves several kinds of people, both at the customer and provider level. These people play different roles in implementing and using an ODM Enterprise solution. A particular project might not involve all of them. Sometimes only a small subset is present.

It is important to distinguish between roles in *creating the application* (in the project) and roles actually *using the application*. The different people have *responsibilities* (in the project), and can have *benefits* when they are using the deployed application.

Benefit: There is a good chance that you will find yourself in one of the following roles. Thus, in the following, you might find some valuable benefits that an ODM Enterprise-based planning application can bring to you!

2.5.1 Client roles

The different client roles are shown in Table 2-5.

Table 2-5 Client roles

Role	Responsibility	Benefit
Business user / planner / scheduler / (end) user of the application / subject matter expert (SME)	Users in this role perform the planning or scheduling job, that in the future, the ODM Enterprise application will support. These users know the business process in the most detail. They must provide all their business and planning knowledge to the rest of the project team.	The ODM Enterprise decision support application can reduce the routine, tedious parts of their jobs. This user group can then spend more time on higher value tactical and strategical tasks, the critical complexities of the business.
Team manager / team lead of the operating or planning division	Users in this role have responsibility for the operating or planning team, consisting of multiple SMEs or planners and schedulers. As such, several, if not all, of their team members will have heavy involvement in the project. These users must make sure that the SMEs can provide sufficient effort to the project.	The team can benefit from the project in multiple ways. The decision support application can streamline team members' working tasks and can reduce tedious tasks. Second, and even more important, the work results of the team members can improve: They can produce better plans and schedules that deliver higher value (for example, through reduced production costs).

Role	Responsibility	Benefit
Executive / project sponsor or owner in a business division	This user has the responsibility for providing the project budget. Because a business division will directly use the application, the project sponsor usually also has responsibility for part of a business division. In many cases, the project sponsor holds a high-level management position, sometimes even at the C-level (such as a Chief Operating Officer).	This user can get the most business benefit from the new application. The decision support application can help to increase value or reduce costs significantly (for example in supply chain, or production). As a high-level manager, this user has a broader view of the organization's goals and can benefit from the new application on this higher level.
Project manager	The project manager has the responsibility for the project that introduces the new ODM Enterprise decision support application.	This user has the opportunity to manage a project that benefits both IT and business staff. In addition, the project is in a thriving new area (business analytics and optimization), which can support career development.
IT manager / Chief Information Officer (CIO)	This user has responsibility for the IT strategy of a large division, or even the whole company. This role must make sure that the IT systems in place fit together, do their job, and impose the least possible costs.	This user can benefit from a mature software platform that integrates well with other systems. This platform can be introduced and maintained with little effort. For example, it can be maintained by using widely available resources. Additionally this user gets a general decision support software platform that can help to strategically implement many different applications in the future. This capacity can simplify future decisions.
IT architect	The IT architect has responsibility for the IT architecture of the new application. This role must make sure that this particular new application works technically, satisfies non-functional requirements such as capacity planning, and integrates with the existing applications where necessary.	By using ODM Enterprise, the IT architect gets a mature software platform that is already in place at many customers. The platform supports various different types of decision support applications of different sizes based on data, number of users, and so on.

Role	Responsibility	Benefit
IT developer	The IT developer serves on the development team and can develop parts of the application, such as visualization extensions, data integration, or authorization access.	The IT developer has the opportunity to work on a project that benefits both IT and business staff. In addition, the project is in a thriving new area (business analytics and optimization), which can support career development.
IT policy manager	The IT policy manager has responsibility for certain IT policies, such as IT integration, manageability, and security.	By using ODM Enterprise, the IT policy manager gets a mature software platform that integrates well with other systems. In addition, it is already proven in many customer implementations, for various different types of decision support applications.
QA engineer	The QA engineer takes care of quality assurance processes, such as testing, functional validations, technical test, and build process.	The QA engineer has the opportunity to work on a project that benefits both IT and business staff. In addition, the project is in a thriving new area (business analytics and optimization), which can support career development.

Tip: A project has a higher likelihood of success if you have a *dream team* that consists of two strong players who drive and manage the project. One must be from the IT division, and the other from the business division.

2.5.2 Provider roles

The different provider roles are addressed in Table 2-6.

Table 2-6 Provider roles

Role	Responsibility
Business analyst/SME	This user group analyzes the organization's business problem (planning or scheduling). It usually brings relevant business and process knowledge to the project, and has deep experience with ODM Enterprise projects. This experience helps ensure that the application to be implemented fits the organization's needs and can ultimately provide the maximum benefit to the client.

Role	Responsibility
Operations Research (OR) expert	The OR expert translates the business requirements into an optimization model. The OR expert also must be involved in the specification of the requirements, because some requirements might be difficult to implement in an optimization model. The OR expert has the obligation to discuss and resolve these issues with the business subject matter experts.
IT architect	The IT architect supports the organization's IT architect in designing and validating the IT architecture of the new, ODM Enterprise decision support application.
Project manager	The project manager helps manage the project. Depending on the project contract, the project manager can also act as the project manager with responsibility for delivering the whole project.
ODM Enterprise product expert	The ODM Enterprise product expert configures the ODM Enterprise application, and can also help installing and rolling out the final software.
IT developer	The IT developer implements application extensions, upon request from the organization.
QA engineer	The provider QA engineer supports the organization's QA engineer, taking care of quality assurance processes, such as testing, functional validations, technical test, and build process.
Executive manager	The executive manager owns the project on the provider side, and can serve as an additional escalation level if problems arise in the project.

2.6 Advantages of ODM Enterprise solutions

There are several alternative options to consider when you are implementing decision support applications. Each has its advantages and disadvantages, as summarized in Table 2-7.

Table 2-7 Comparison of different planning application approaches

Approach	Advantages	Disadvantages
Spreadsheet-based applications	▶ Quick start ▶ Familiar tool ▶ Easy to use	▶ Limited size and complexity ▶ Hard to maintain ▶ Cumbersome what-if analysis ▶ Limited collaboration
Packaged applications (for example, IBM i2® or SAP)	▶ Standard, built-in functionality ▶ Packaged best practices	▶ Difficult to change user interface ▶ Might not integrate well ▶ Might not capture all costs, constraints, and goals ▶ Might impose the wrong business process
Custom solution (built from scratch, using some basic components, such as an optimization engine)	▶ Tailored to business needs	▶ Difficult for business managers to participate in development ▶ Difficult to build user interface ▶ Difficult to integrate data ▶ Difficult and expensive to maintain over time ▶ “Obligation” to maintain custom optimization and data model
Platform based, <i>tailored</i> (using the ODM Enterprise platform)	▶ Tailored to business needs ▶ Easy for business managers to participate in development ▶ Easy to build user interface ▶ Easy to integrate data ▶ Easy to maintain and extend over time ▶ Integrated optimization	▶ “Obligation” to maintain custom optimization and data model

Tip: If you are using spreadsheets for planning and suspect that the planning process could be improved, or the plans could be better, consider an ODM Enterprise-based approach.

Comparing an ODM Enterprise-based software solution with pre-packaged applications, experience shows that:

- ▶ For many decision processes, there is no pre-packaged solution available on the market.
- ▶ If there is a pre-packaged solution available, it might provide you with a rich set of functions. These functions can include best practices that you can use without having to invent them.
- ▶ However, make sure that the package really fits to your requirements. In many cases, you will find that the package does not fit 100%. In this case, you must find ways to either live without the percentage, or how to work around them. In many cases this will be difficult, expensive, risky, or even impossible.

With ODM Enterprise, you can tailor the optimization model, which is at the core of the planning application, as closely as possible to your business model. This tailoring can include your data, constraints, goals, maybe even your Unique Selling Proposition. Pre-packaged applications come with embedded optimization models that might be configurable but generally are not changeable. Thus, they add at least one level of indirection between your requirements and the mathematics. This limitation often means that you do not get the optimal result with, for example, minimal costs or maximum yield.

An ODM Enterprise application can even provide you a competitive advantage. If a competitor uses the same packaged application as you do, you get the same answers to important business questions or decisions as they do. With ODM Enterprise, you can get better answers.

Remember: Even if a pre-packaged solution provides 90% of the required functionality, achieving the remaining 10% might be hard and expensive, and sometimes even impossible.

You might consider building a new application from scratch, using only some components. Be aware that with ODM Enterprise:

- ▶ You get a mature product, including:
 - A rich set of standard functions
 - An open platform concept
 - Documentation, tutorials, examples, and training

- ▶ Creating and implementing the initial application takes less effort.
- ▶ You can benefit from future product development.
- ▶ Product expertise is widely available on the market. You can use the expertise of a large community, probably including some people that already work for your company, to maintain and extend the application. This process keeps it up-to-date with future business developments.

Tip: An ODM Enterprise-based solution can provide you with a competitive advantage, with less development effort than a complete custom solution, and more flexibility for the future.

2.7 Conclusion

This chapter has shown the following items:

- ▶ What an ODM Enterprise-based application provides on top of mathematical optimization.
- ▶ How such an application looks in many potential planning areas: It is easy to use, nevertheless provides a rich set of planning functions.
- ▶ Which roles are involved in project and application, and how you can benefit from both.
- ▶ Why an ODM Enterprise planning and optimization application provides significant advantages over other planning options.



ODM Enterprise architecture

ODM Enterprise is a specialized solution platform that allows IT professionals, with Operations Research (OR) specialists, to build powerful, interactive decision support applications. ODM Enterprise applications make optimization models more accessible and easier to understand for business users who might not have an OR background.

This chapter includes the following sections:

- ▶ Key ODM Enterprise concepts
- ▶ ODM Enterprise architecture
- ▶ ODM Enterprise API
- ▶ Domain Object Model (DOM)
- ▶ Data architecture
- ▶ Scalability and load balancing
- ▶ High availability
- ▶ Security model

3.1 Key ODM Enterprise concepts

To better understand the architecture of an ODM Enterprise application, you need to understand ODM Enterprise application frequently used terms. The architecture of an ODM Enterprise application has many layers. This section addresses them one at a time to understand how they are structured and work with each other.

Decision makers can adapt to changing conditions by modifying the input data and the rules that apply to their specific business scenarios. The goal of ODM Enterprise is to empower users of each custom ODM Enterprise application to *make better decisions faster*, and to *manage change and complexity*.

You can adjust the ODM Enterprise application solution by adding or modifying *input data*, by adding or modifying *rules* (costs, constraints, assumptions), and by modifying *goals*. If the modifications conflict with certain business rules, explanations are provided stating which requirements cannot be satisfied. This process helps you obtain a feasible solution. The decision maker can then modify the active/inactive status or the priority of these requirements, and attempt to find a better solution.

Keep in mind that several different terms are used to mean the same thing. Table 3-1 shows the correspondence.

Table 3-1 Corresponding terminology

Business problem	ODM Enterprise	OPL
Targets	Goals	Objective function (dexpr)
Limits	Requirements Rules	Constraints
Choices	Solution	Decision variables (dvar)

3.1.1 Model

The term *model* is used to denote many different ideas in many different products. In an ODM Enterprise application, the term model refers to one of the following three concepts:

- ▶ Application Data Model
- ▶ Optimization model
- ▶ Domain Object Model

Application Data Model

ODM Enterprise provides the user with a framework for a flexible decision making process by using the *Application Data Model* (ADM). The ADM is a relational data model that defines and holds the input to, and output from, an ODM Enterprise application.

ODM Enterprise puts the ADM at the center of the development process of an application in which IT architects and developers use a data-centric approach. This approach enables them to investigate data and quickly develop prototypes of the final application, while work on the optimization model is performed simultaneously by OR experts. The development of an ODM Enterprise application does not depend on having a finished OPL model as its basis.

When connecting an existing database or back end systems to an application such as ODM Enterprise, the IT architect faces the challenge of moving data in and out of involved IT systems. The ODM Enterprise Application Data Model and its associated editors provide a graphical user interface. This interface allows the IT architect to draw the data model that the application will use. This modeling can start from an existing database schema. After an ODM Enterprise project is created, the developer can define the tables of a scenario in the Application Data Model. There is no concept of input or output tables at this level. This aspect is dealt with after the developer creates an OPL project that is associated with the ODM Enterprise project. At that point, the developer can define which data are passed to the OPL model, and which OPL output data are passed back to the user.

Optimization model

An *optimization model* is an abstraction of a business problem that represents a set of *business decisions* or *objectives*. These decisions involve targets, limits, and choices. They are typically developed by an OR specialist by using an Optimization Programming Language (OPL). The optimization model is an important part of an ODM Enterprise application. It is because its inputs are modified by the business user based on the business requirements and goals at a specific time. ODM Enterprise can then use the CPLEX Optimizers or Java code to perform the analytic calculations. At the end of the calculations, the user is presented with the results of the particular run.

Figure 3-1 on page 58 compares the optimization model with the Application Data Model. The OPL model is generated based on the ADM. The OPL model is customized so that only what is needed is selected and naming makes sense for the OR expert. The ADM is a framework to manage the input to and output from an optimization model, which is typically owned by the integration expert.

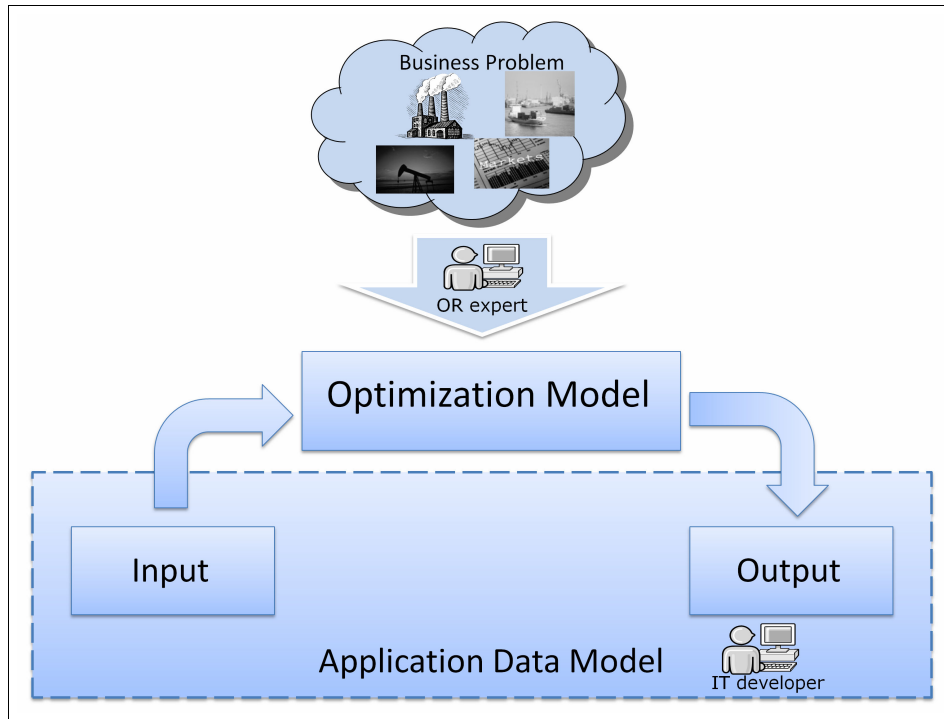


Figure 3-1 Relationship between the optimization model and the Application Data Model

Domain Object Model

ODM Enterprise supports the specification and generation of a *Domain Object Model* (DOM) in Java. The DOM is based on the ODM Enterprise Application Data Model. It provides a simple way to map tables to Java classes, columns to fields, and foreign keys to bidirectional references. For more information, see 3.4, “Domain Object Model (DOM)” on page 97.

3.1.2 ODM Enterprise IDE versus ODM Studio

The environments during and after development differ based on the different needs of the developers and users.

The development stage is handled by integration experts who design the Application Data Model layer and by the OR experts who formulate the underlying optimization model. These specialists use the ODM Enterprise IDE for the development process, which is composed of an Eclipse IDE with ODM Enterprise Eclipse plug-ins.

After the application is developed, it must be deployed (packaged) into an executable file and distributed to the business user. An ODM Enterprise application interacts with business users graphically through a module that is called the ODM Studio. This module allows them to change the input elements and view the results.

Figure 3-2 shows the ContainerYard example as viewed in ODM Enterprise IDE and the ODM Studio.

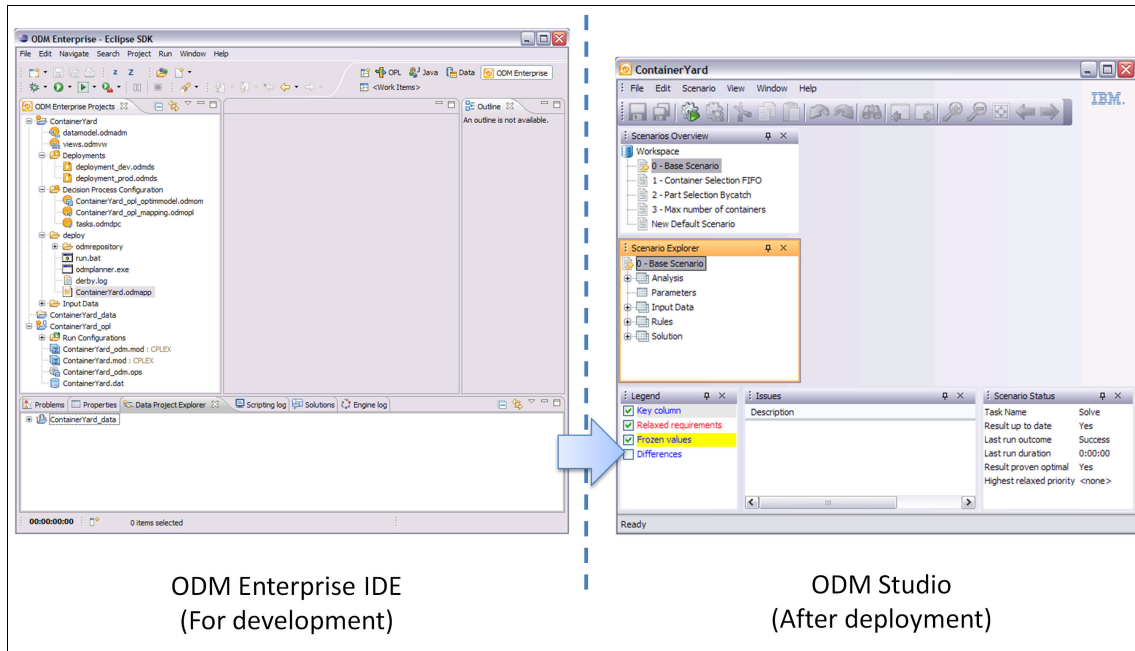


Figure 3-2 Comparing the ODM Enterprise IDE and ODM Studio

Information is presented in the deployed ODM Enterprise application as elements under various groups or categories available for every scenario. This presentation is controlled by the `views.odmvw` file, which is created at the start of every ODM Enterprise project. This file can be customized for each business application's needs.

A typical ODM Enterprise application might use groups (which look like folders in ODM Studio) for Analysis, Rules, Input Data, and Solution (Output). The business users can then change Goals and Requirements in the Analysis group, or Rules (if present) in the Rules group. The Input Data and Solution groups then have views that correspond to the preferred appearance of the input and solution data.

3.1.3 Scenario

A *scenario* represents an instance of the ODM Enterprise Application Data Model that includes the data, goals, rules, requirements, and solution as viewed in ODM Studio. You can think of it as a snapshot of the ADM corresponding to a certain run. It contains all the information that is needed to re-create it. ODM Enterprise provides the business user with more functions to manage (create, share, delete, and store) scenarios.

What-if analysis is key in optimization-based business applications and decision support systems. This type of analysis is provided by the *scenario comparison feature* in ODM Enterprise. As businesses become more distributed, the benefits that scenarios provide in this highly responsive and collaborative environment are unparalleled. Figure 3-3 shows three instances of the Application Data Model representing the objectives of minimizing cost, maximizing profit, and minimizing waste with different data sets.

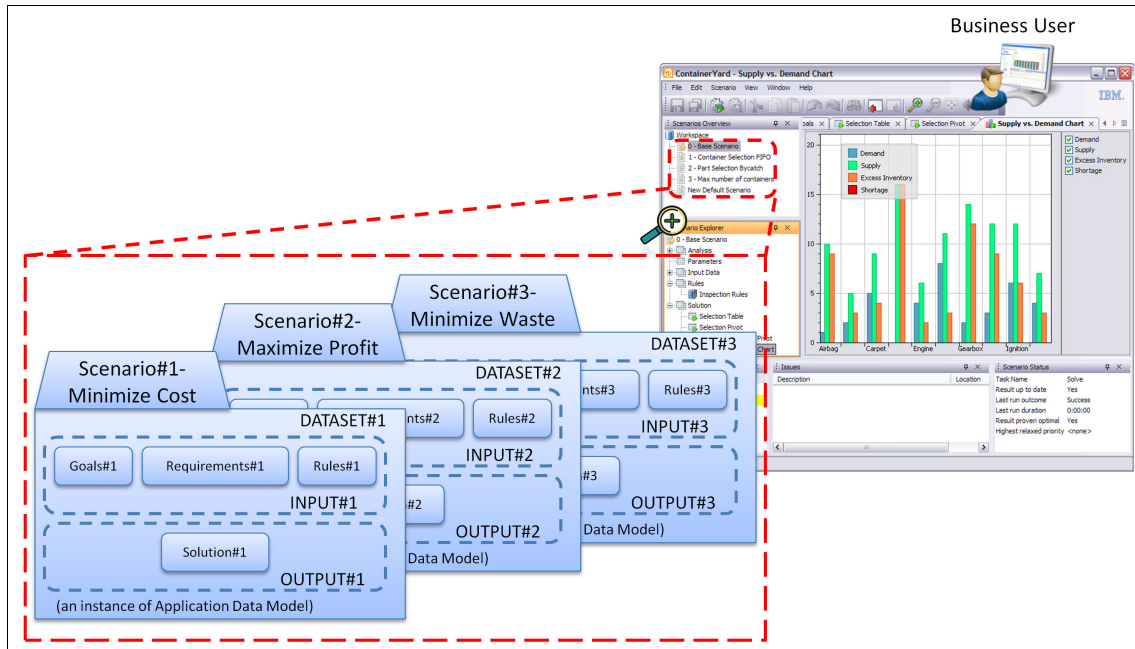


Figure 3-3 Scenario examples

3.1.4 Workspace

In a deployed ODM Enterprise application, a *workspace* is used to organize scenarios. You can think of a workspace as a collection of scenarios that can be

stored in a folder/subfolder structure. Like scenarios, workspaces can be created, shared, and deleted by using the ODM Studio. They come in handy in a distributed application that requires collaboration from various stakeholders. A workspace can contain folders in addition to scenarios. In addition, scenarios can exist in multiple workspaces and in multiple folders within a workspace. This concept is depicted in Figure 3-4.

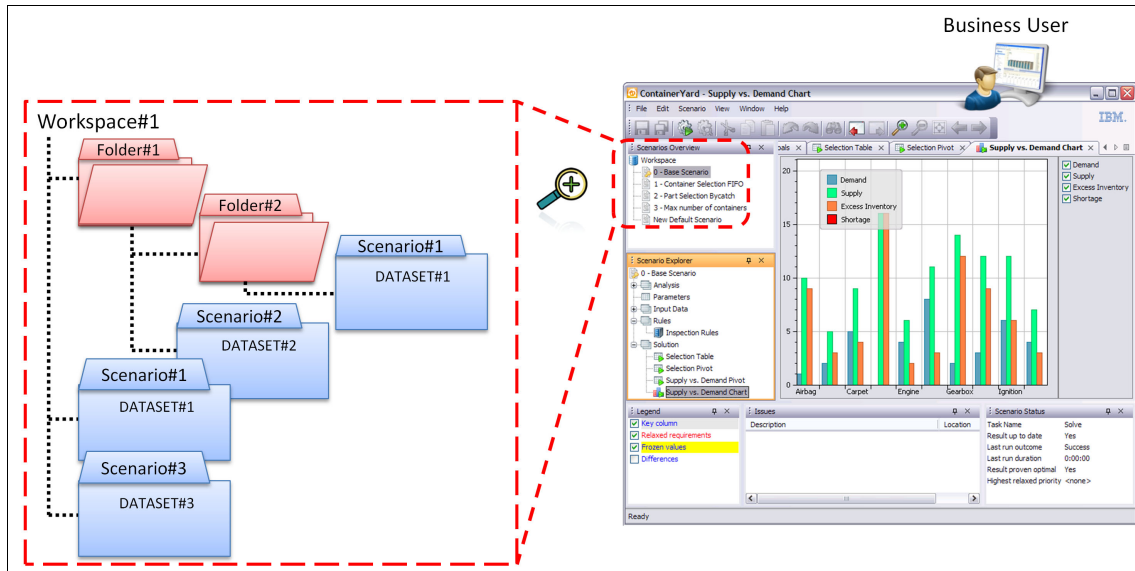


Figure 3-4 An example workspace

3.1.5 Scenario Repository

The ODM Enterprise Scenario Repository is used to store workspaces, workspace folders, scenarios, and configuration information. The current release of the product provides a built-in SQL database to store the repository. However, you can also use IBM DB2®, Oracle, or SQL Server for this purpose. The scenario information is stored in terms of individual scenarios and the workspaces that contain them. This information includes changes to configurations and layouts that are saved by individual users on their local console. They can, for example, save different views of the same pivot table.

Figure 3-5 shows that the Scenario Repository can be configured in these ways:

► Single-user, local database

In this mode, ODM Enterprise is installed on a local system along with the default database. The ODM Enterprise application is configured as a single-user application. The user who deploys the application can then access workspaces and scenarios that are stored in the local repository.

► Multi-user, server database

In this configuration, the application is configured to connect to a centralized database for storing the scenario information. This database can be on a separate network server. Given the multi-user nature of the configuration, all the users are then able to access the workspaces and scenarios that are stored in the repository.

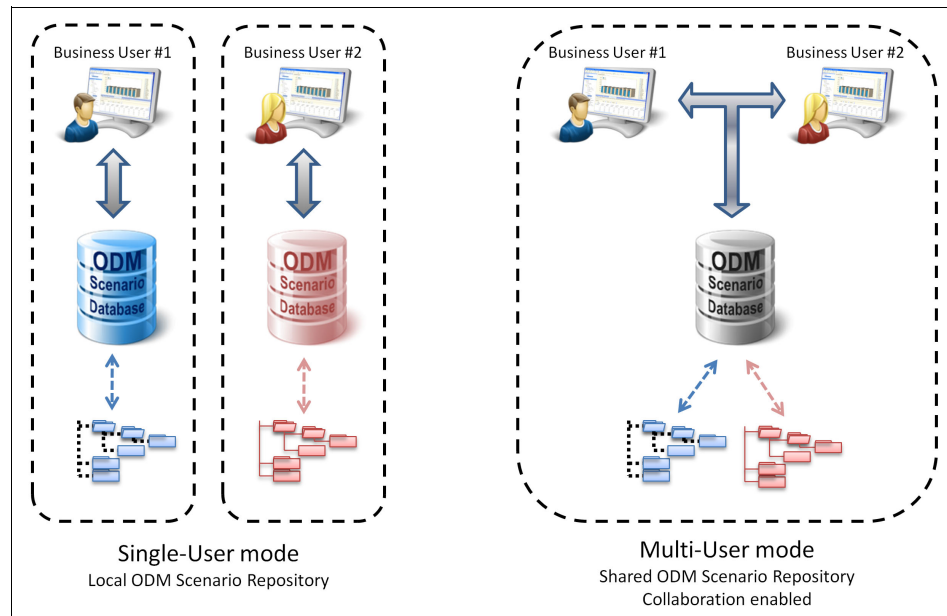


Figure 3-5 Scenario Repository configuration comparison

The repository architecture provides the following benefits:

- There is no need to copy or export files between multiple users who can use the common Scenario Repository.
- Access control is managed by ODM Enterprise so that concurrent modifications to a scenario are prevented.
- The ODM Studio loads the last-used workspace, so users do not need to reload scenario sets every time they use the ODM Enterprise application.

- Changes to the configuration and layouts are saved on the user's local system. This configuration enables each user to have, for example, different views of the same pivot table.
- The migration path to newer ODM Enterprise versions is automatic. The ODM Enterprise application needs to be redeployed to be able to run using the newer ODM Enterprise client. However, previously saved scenarios can be imported and saved.

3.1.6 Goals

Goals represent the business targets that need to be optimized (choosing the best among the various feasible alternatives). They correspond to the decision expressions in the underlying OPL model's objective function. Developers can select the decision expressions (dexpr) that get displayed in the Goals window of the ODM Enterprise application. They can do so by editing the Goals tab of the <project_name>_opl_opt.immodel.odm file in the ODM Enterprise IDE. For each ODM goal, the user can see its value in the current solution, specify its importance factor (weight). The user can also see the goal breakdown, search the goal bounds, and add constraints to its value in the ODM Studio as shown in the Figure 3-6.

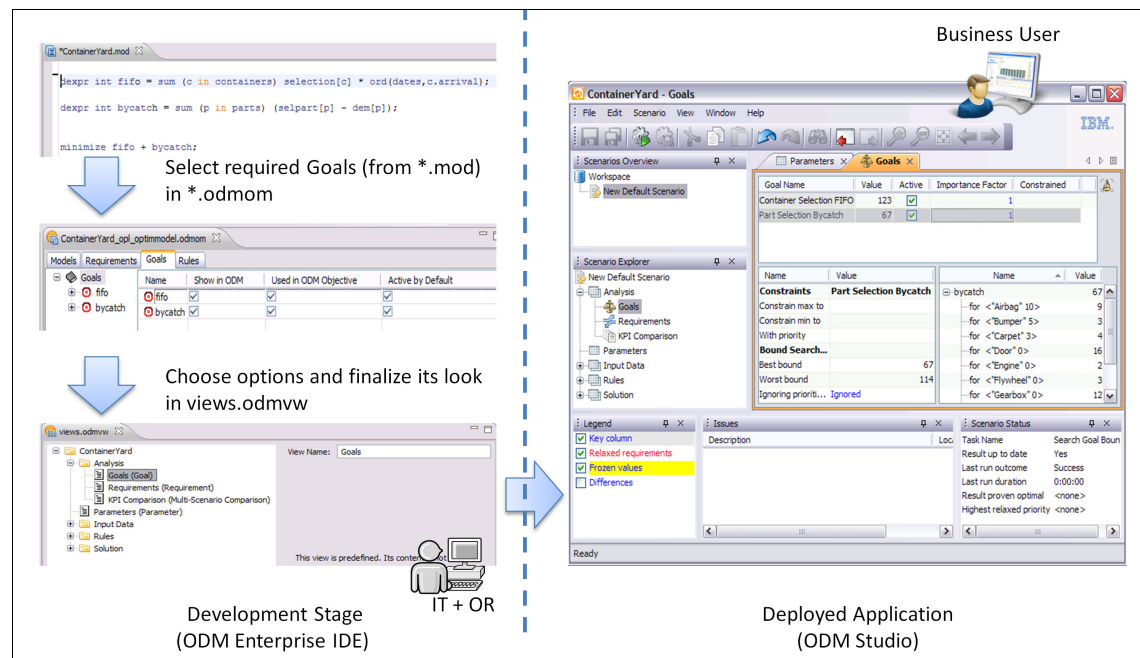


Figure 3-6 Configuring goals

3.1.7 Requirements

Requirements correspond to the limits in a business problem that must be satisfied while you are optimizing the goals. Constraints in the OPL model are mapped to the requirements in the ODM Enterprise application by editing the Requirements tab of the <project_name>_opl_optimmodel.odm file. This file can be accessed in the ODM Enterprise IDE. Requirements can be assigned a priority in the Requirements window by using the ODM Studio. This process gives the business user a finer level of control over which requirements are relaxed. If the model has a feasible solution to start with, the Requirements window in the ODM Enterprise application is empty. One such case within the ContainerYard example is depicted in Figure 3-7.

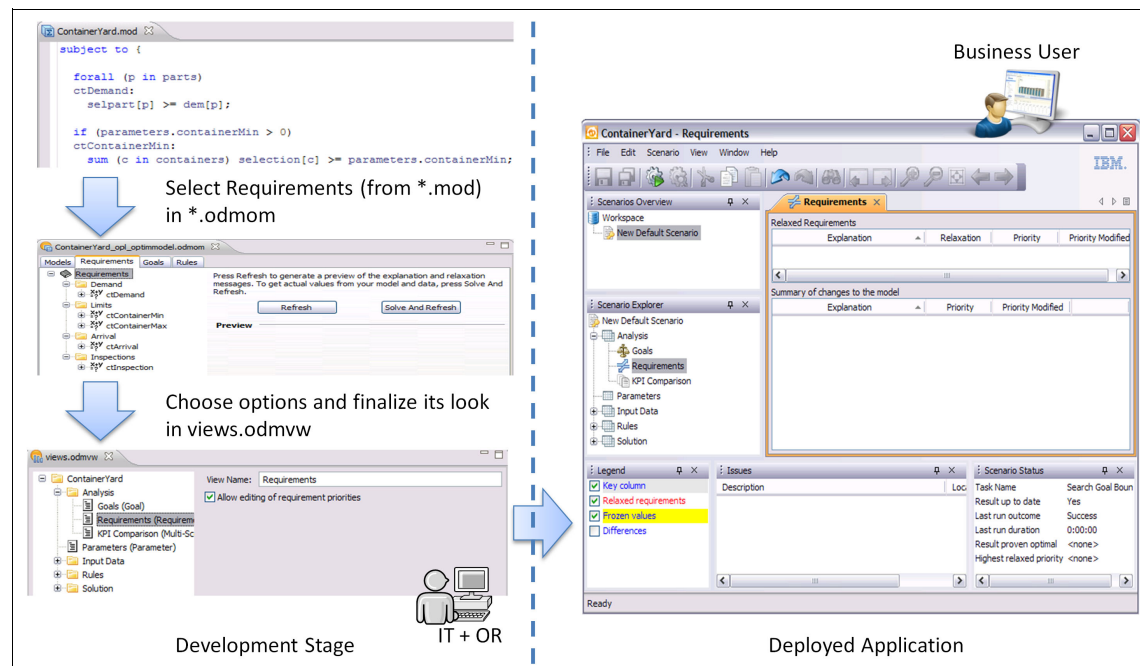


Figure 3-7 Configuring requirements

3.1.8 Rules

A *rule* is a data structure that ODM Enterprise uses to exert a finer degree of control on an OPL constraint. Rules enable the business user to directly control some of the constraints in the optimization problem. They can do so by setting priorities on enforcing constraints when conflicts arise, creating new constraints, or deactivating certain constraints in what-if analysis. In development, constraints

are managed by editing the Rules tab of the
<project_name>_opl_optimmodel.odm file in the ODM Enterprise IDE.

Keep in mind the following points when you are working with rules:

- ▶ A *Rule template* is a type of rule to which a priority can be applied. It is defined by its message, the input data that it is linked to, and sometimes the constraint it controls.
- ▶ A *Rule category* is a set of rule templates that can be grouped.
- ▶ A *Rule view* is the view that displays a rule category.

Rules have the following restrictions:

- ▶ Rules are associated with tables, which are mapped to an OPL input tuple set. Therefore, the rules must be applicable only in ODM Enterprise projects that use OPL tasks, and not just custom Java tasks.
- ▶ To associate a rule with a table, the table must have no primary key, or all of its columns must be part of the primary key collection.
- ▶ A constraint that is linked to a rule template must be in a `forall` statement that iterates directly on the tuple set rather than use an anonymous tuple:
 - `forall(i in incompat)` is OK
 - `forall(<n,m> in incompat)` is *not* OK
- ▶ A rule template can be in only *one* rule category.
- ▶ When you delete a rule category, you must manually delete any views with the associated rules.

Figure 3-8 depicts one such rule in the ContainerYard example.

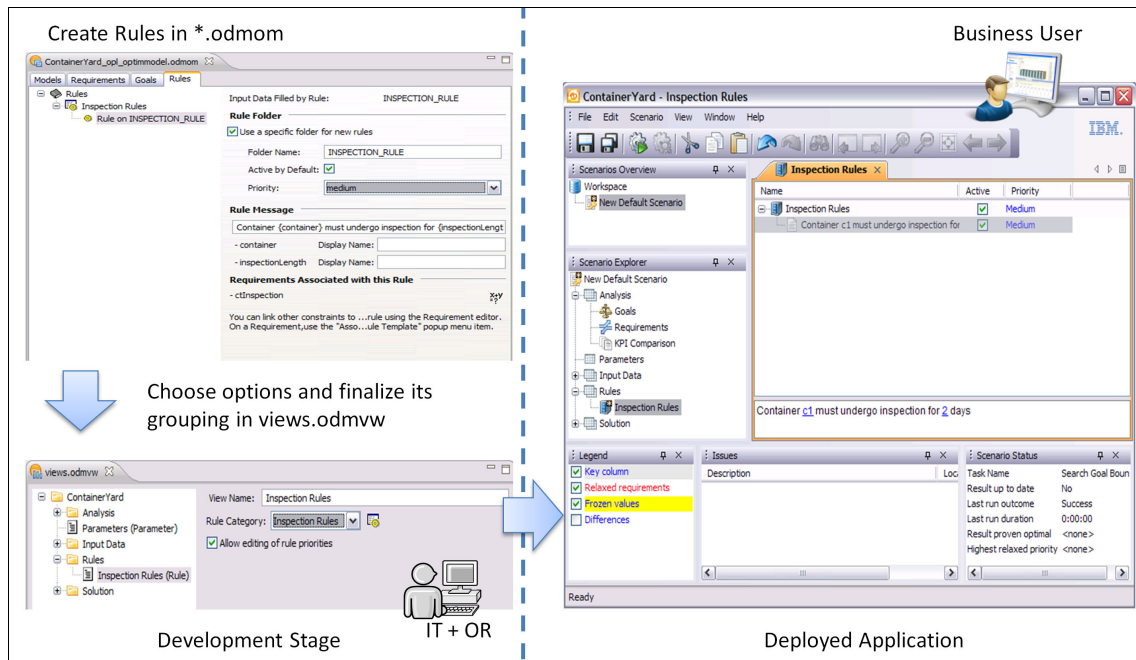


Figure 3-8 Configuring rules

3.1.9 Input data and solution views

Any optimization model requires input data, and it provides some output at the end of each optimization run. The Application Data Model (containing the input and output elements of an ODM Enterprise application) communicates with the optimization model based on a mapping file called `<project_name>_opl_mapping.odmopl`. This file defines the mapping between the Application Data Model tables and their optimization model counterpart (for example, tuple sets, arrays, or decision variables). The file also classifies each element (table) as either input or output.

The input data and output (solution) tables in the Application Data Model layer can be associated with views that define their appearance in ODM Studio. The views need to be configured so that data in the tables is shown in a format that is meaningful for the business user. You can change the format by editing the `views.odmvw` file.

Figure 3-9 shows how “Demands by part” was configured as an input data chart view (pie chart).

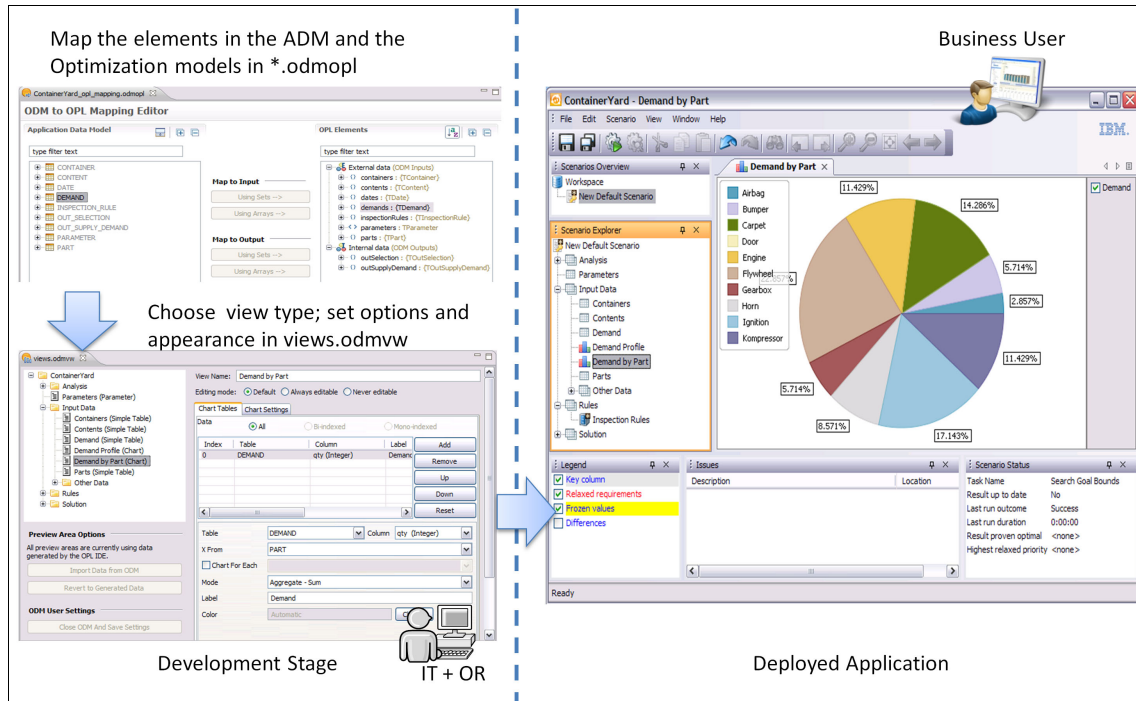


Figure 3-9 Configuring input data and solution views

Built-in views

ODM Enterprise provides the user with the following built-in views for data representation:

- ▶ Simple table view

This view shows a table in the Application Data Model as a flat table where each column in the view represents a column in the table. It can also show a tuple set. In that case, the table has one column per tuple field and one row per tuple in the tuple set. This configuration is typically the case for tables that contain a single row.

- ▶ 2D array view

As the name suggests, a 2D array view is useful to display values that depend on two parameters in a flat table.

► 1D array view

This view provides a unique view to display several tables that share a key. If you have different variables in different tables that share a common key, this view provides a way to compare them side by side.

► Parameter view

A parameter view enables the user to modify simple parameters of the ODM Enterprise Application Data Model. A parameter is characterized by a single input data item as opposed to a regular table that contains several rows of data.

Restriction: The tables that are used to create the parameter views have only one row because the corresponding columns only have a single value. This information is not explicitly mentioned in the database schema that is imported from an external database. Therefore, the developer must indicate it manually by selecting the “This table has a single row” check box in the appropriate table’s Properties view in the ADM editor.

► Pivot table view

This is a powerful data summarization tool that can automatically sort, count, and sum up data stored in tables and display the summarized data. Pivot tables are useful to quickly create crosstabs. Crosstabs are a process or function that combines or summarizes data from one or more sources into a concise format for analysis or reporting. These crosstabs display the joint distribution of two or more variables. With a pivot table, a business user can set up and change the data summary’s structure by dragging and dropping fields graphically, thus rotating or pivoting it. Pivot tables in ODM Enterprise can be modified by users to their specifications if the option to rearrange them is selected by the developers. These changes can be saved along with the user’s scenarios. Because these changes are saved by user, they do not affect the default layout of the pivot table for other users. A pivot view is useful to perform these tasks:

- Summarize the data that are contained in a lengthy list into a compact format.
- Find relationships within the data that are otherwise hard to see because of the amount of detail.
- Organize the data into a format that is easy to read.

► Multi-scenario comparison view

These views allow you to examine the goals and solve status of multiple scenarios in a single view. The solution data is presented both in a spreadsheet table format and graphically with charts. You can compare the

solutions by using bar charts or polyline graphs. You can also perform sensitivity analysis on multiple scenarios to get an idea of which ones offer the best solutions.

The display makes it easier to see the differences between the solutions that are found in each scenario. You can see how they performed against the goals of the ODM Enterprise application. This process provides a high-level overview of the different scenarios.

- The table area in this view contains the following elements by default (this configuration can be changed in the deployed application):
 - Scenario name
 - Columns corresponding to the individual goals of each scenario.
 - Columns for parameters that correspond to input data with a simple value defined in a parameter view.
 - Columns for results corresponding to output data with a simple value, if any and if defined in a parameter view.
 - A “Has Result” column with a check mark that indicates whether the scenario ran to completion and found a solution.
 - An “Is Feasible” column with a check mark that indicates whether the solution was feasible.
 - Column for “Gap” indicating the relative difference between the best solution that was found and the proven best possible solution value.
 - Values for scenarios that require no relaxations of any of the goals are displayed in a black font. The ones that did require relaxations are displayed in red.
- The chart area displays information that is contained in the table area graphically in these formats:
 - Bar charts or polyline graphs
 - 2D or 3D formats

► Chart view

ODM Enterprise provides the key with the ability to add charts to the application for visualizing data. The data can be either the input or output values. The Views editor window in the ODM Enterprise IDE provides you with various chart modes like cartesian, polar, radar, and pie. The data can be rendered in the polyline, bar, area, scatter, stair, or pie form with a clustered, superimposed, stacked, or stacked percentage type.

The following basic charting options come with ODM Enterprise:

- Mono-indexed chart view

This chart is used to provide a chart view of a 1D array or simple table view.

- Bi-indexed chart view

This chart corresponds to a 2D array view.

- Multiple index chart view

Tables with more than one key can be used as though they were mono-indexed, which displays data in “Aggregate-Sum” mode.

- Custom view

Along with providing standard built-in chart views, ODM Enterprise gives developers the ability to add custom views to an ODM Enterprise application. A custom view can provide a business-specific view of a problem and help users visualize data more effectively. Creating custom views fall under the advanced usage category. Developers can use Java Swing widgets or IBM ILOG JViews Enterprise to create powerful diagrams, charts, Gantt charts, network diagrams, map based charts, and so on.

For more information, click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Advanced usage → Creating custom views in Java** at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

Custom views

This section illustrates a few examples where custom views can enhance the look and feel of the deployed ODM Enterprise application:

- ▶ Empty container repositioning map view (Figure 3-10)
 - Uses JViews¹ Maps
 - In this example, the ODM Enterprise application is used to reposition empty containers. Operations are dependent on import-export imbalances around the globe. Empty containers are needed in export-rich locations, so they are transported from import-heavy regions.
 - The view is part of the input data, which helps visualize these imbalances by the container type and work week as selected by the business user.

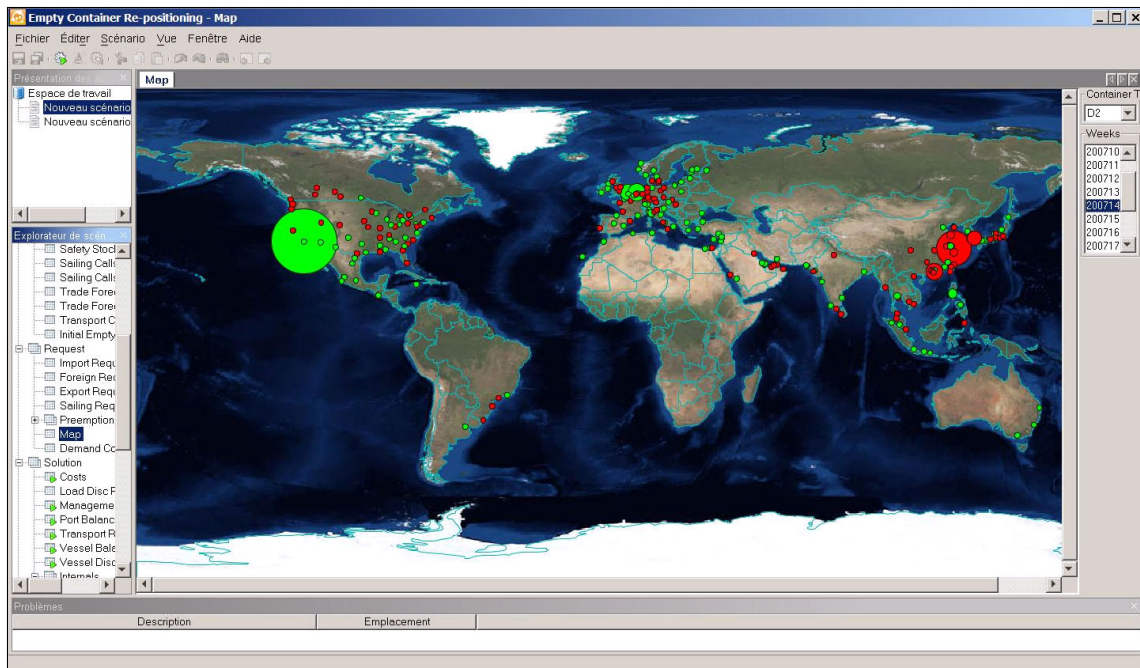


Figure 3-10 Empty container repositioning map view

¹ For more information about JViews, see
<http://www.ibm.com/software/integration/visualization/java/>

- Gantt view of a mining rake schedule (Figure 3-11)
 - Uses JViews Gantt
 - In this example, the ODM Enterprise application is used to optimize the supply chain operations in a mining company.
 - The view is a Gantt chart that represents the daily (by the hour) schedule of rakes as they are used in various parts of the supply chain.

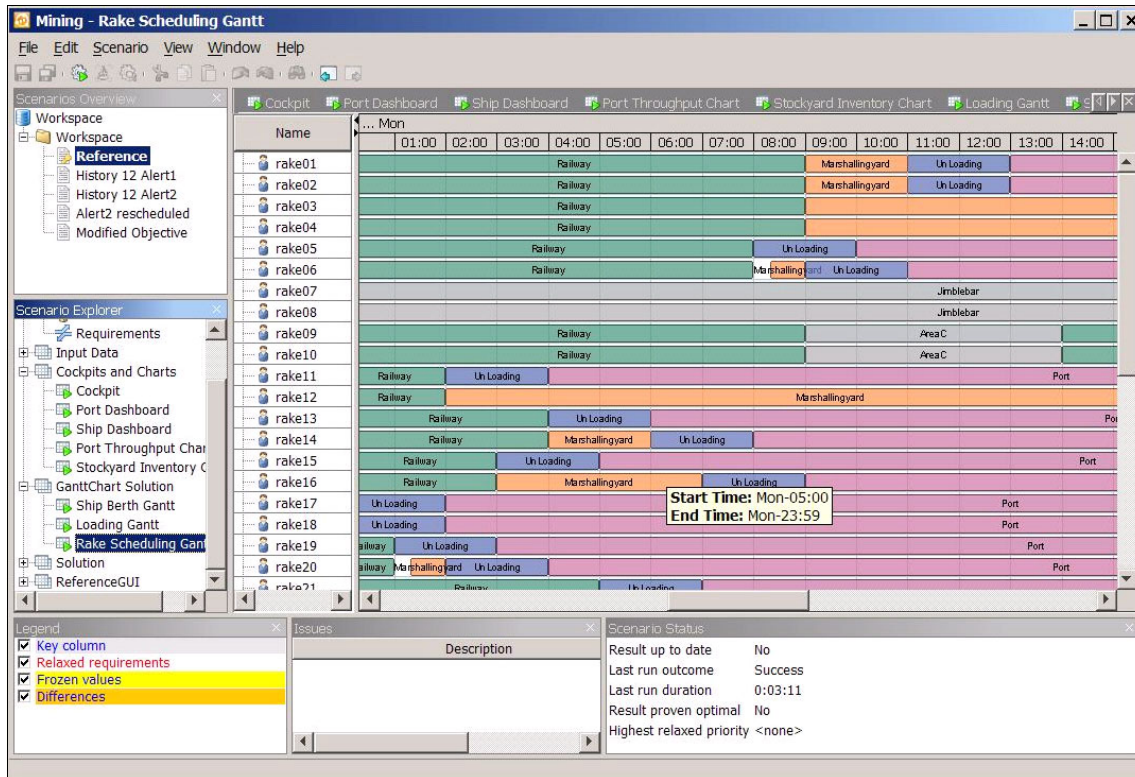


Figure 3-11 Rake scheduling Gantt view

- Ship dashboard view (Figure 3-12)
 - Uses JViews Charts
 - In this example, the ODM Enterprise application is used to optimize the supply chain operations in a mining company.
 - The view represents a ship dashboard wherein the key metrics for a selected ship are displayed in multiple charts at the same time.

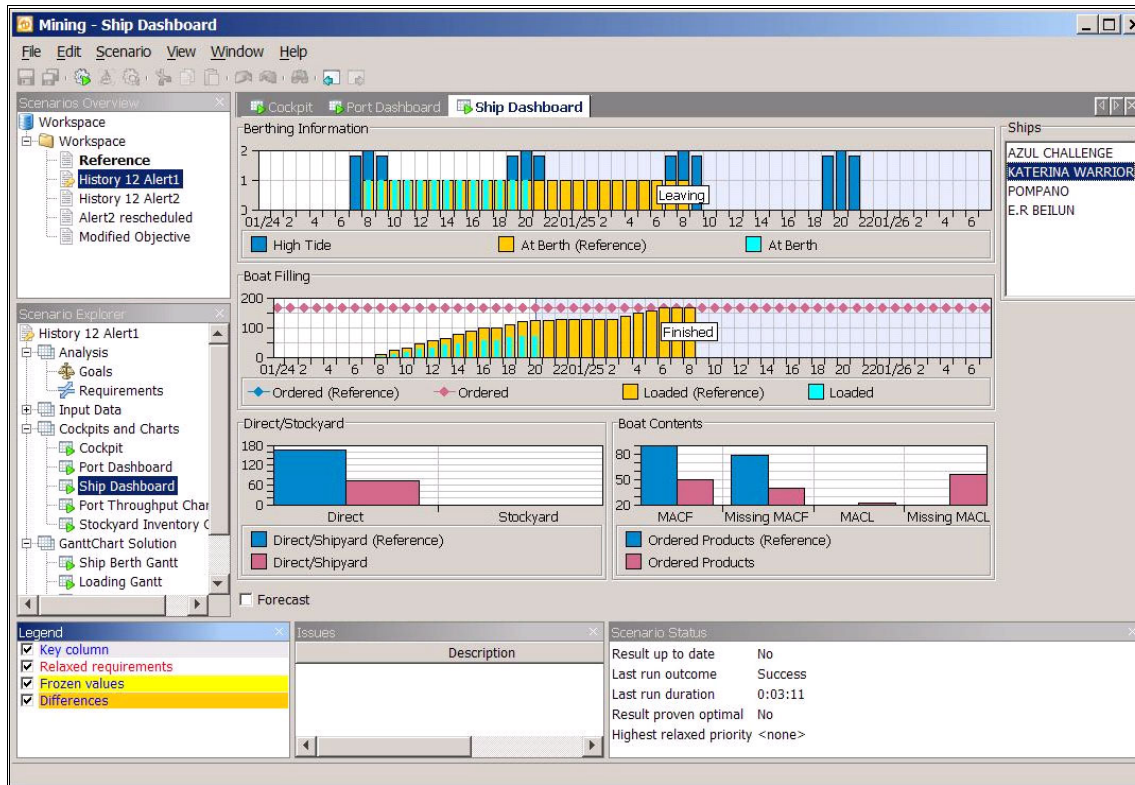


Figure 3-12 Ship dashboard view

- Car sequencing editor (Figure 3-13)
 - Uses JViews Diagrammer
 - In this example, the ODM Enterprise application is used at a car manufacturing company for the sequencing and slotting operations.
 - The view works as an insightful editor that visualizes the sequence of cars in a snake view format. It provides custom actions for selecting the sorting order and highlighting criterion. The business users can even view the constraints and conditions that govern the underlying business logic.

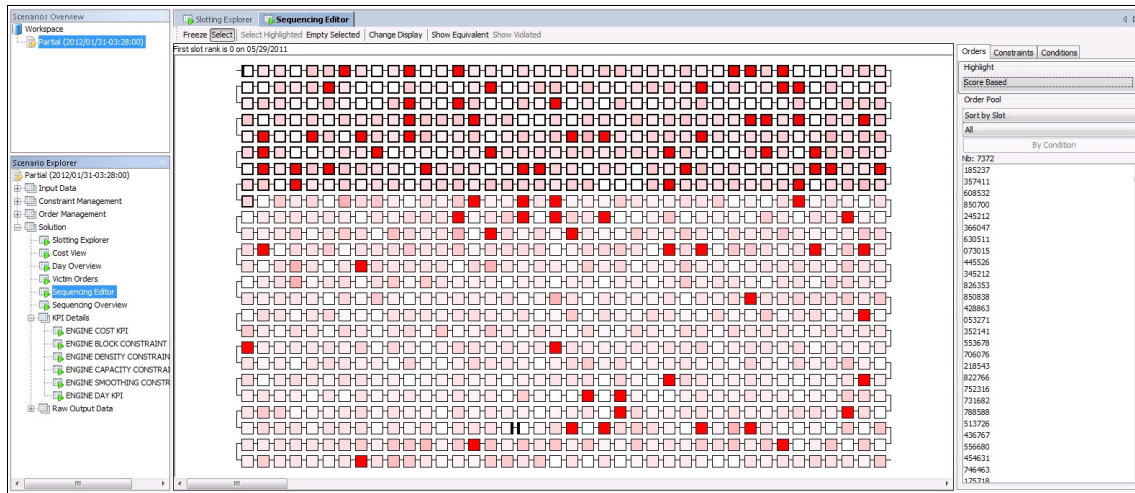


Figure 3-13 Car sequencing editor with a snake view

- [illegible]

Chapter 3. ODM Enterprise architecture **75**

- Truck scheduling Gantt view (Figure 3-15)
 - Uses JViews Gantt
 - In this example, the ODM Enterprise application optimizes a logistics operation that involves the scheduling of trucks that reduces costs and meeting demands at the gas stations.
 - The Gantt view shows the schedule of truck resources on a timeline, as they move between various gas stations so they fulfill demands.

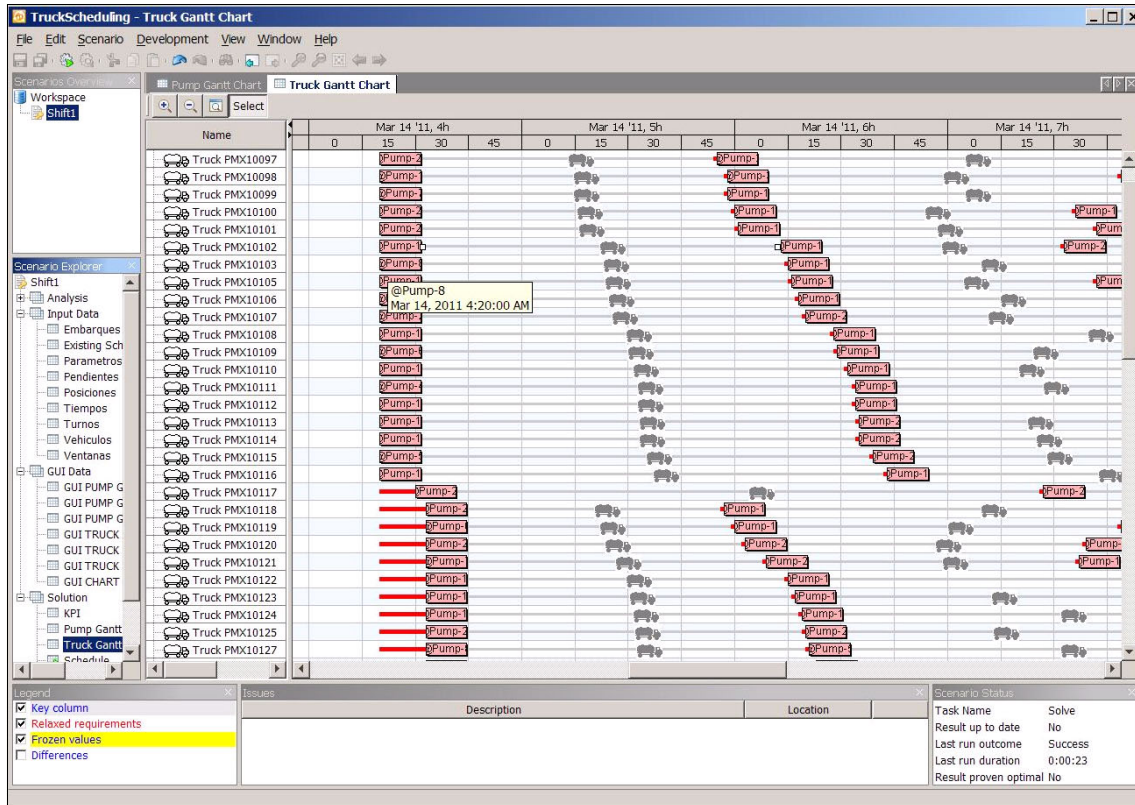


Figure 3-15 Truck scheduling Gantt view

- ▶ Train editor view (Figure 3-16)
 - Uses JViews Gantt and JViews Diagrammer
 - In this example, the ODM Enterprise application is used for train scheduling.
 - The view is an editor that allows the user to manipulate the various attributes that are associated with a train. These attributes include stops that the trains must make with respect to time. The result helps visualize the overall train schedule as a Gantt chart.

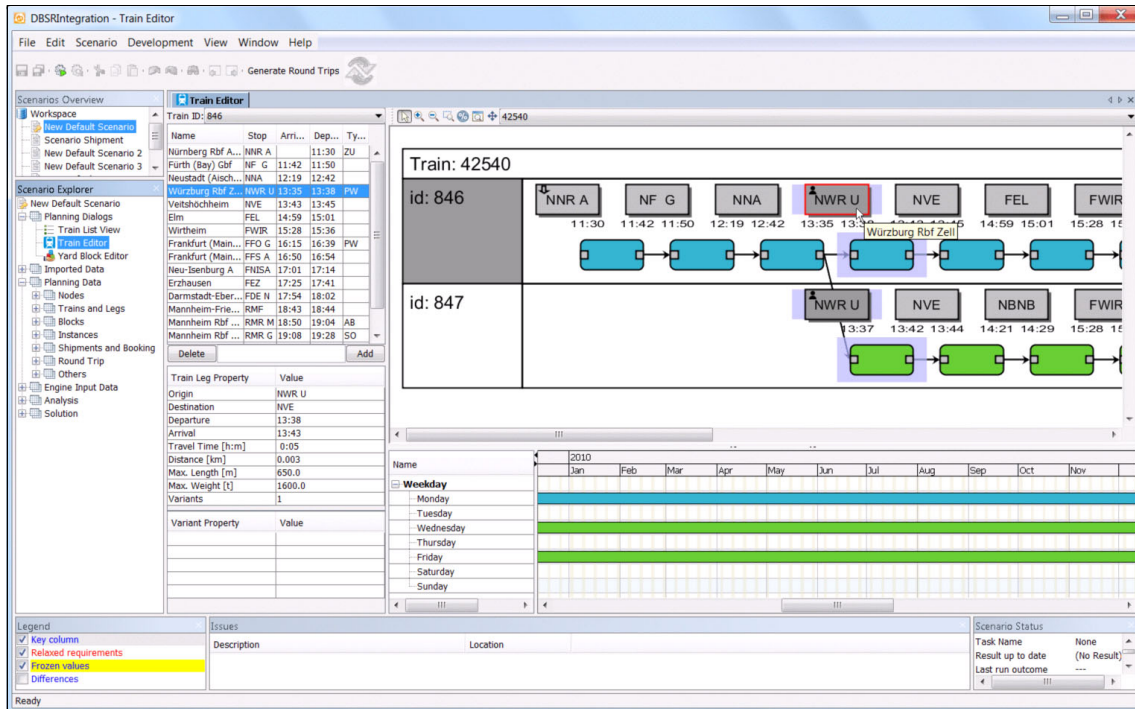


Figure 3-16 Train scheduling editor

3.1.10 Decision process configuration

ODM Enterprise provides the developer with the ability to add *tasks* to the ODM Enterprise application. This process is called the *decision process configuration*. Tasks encompass the actions that the business user can start on a scenario (by right-clicking them) in ODM Studio. Developers can define multiple tasks and algorithms for the ODM Enterprise applications. They can then deploy these tasks to business users by using the decision process configuration editor and editing the tasks.odmdpc file. If you created an ODM Enterprise project without

an OPL project, the “All Tasks” list is empty when you first access this editor. The list remains empty until you explicitly add a task to it. ODM Enterprise supports the following types of tasks:

- OPL task

The more common of the two task types, the OPL task is used to run a certain OPL optimization model. If your ODM Enterprise project is already been associated with an OPL project, an OPL task to start the OPL model is displayed in the decision process configuration editor. This task is specified as the default. This means that the OPL task is started when you click **Solve** in ODM Studio.

Adding an OPL task to an ODM Enterprise project creates an OPL project with a skeleton that contains the elements that are shown in Figure 3-17.

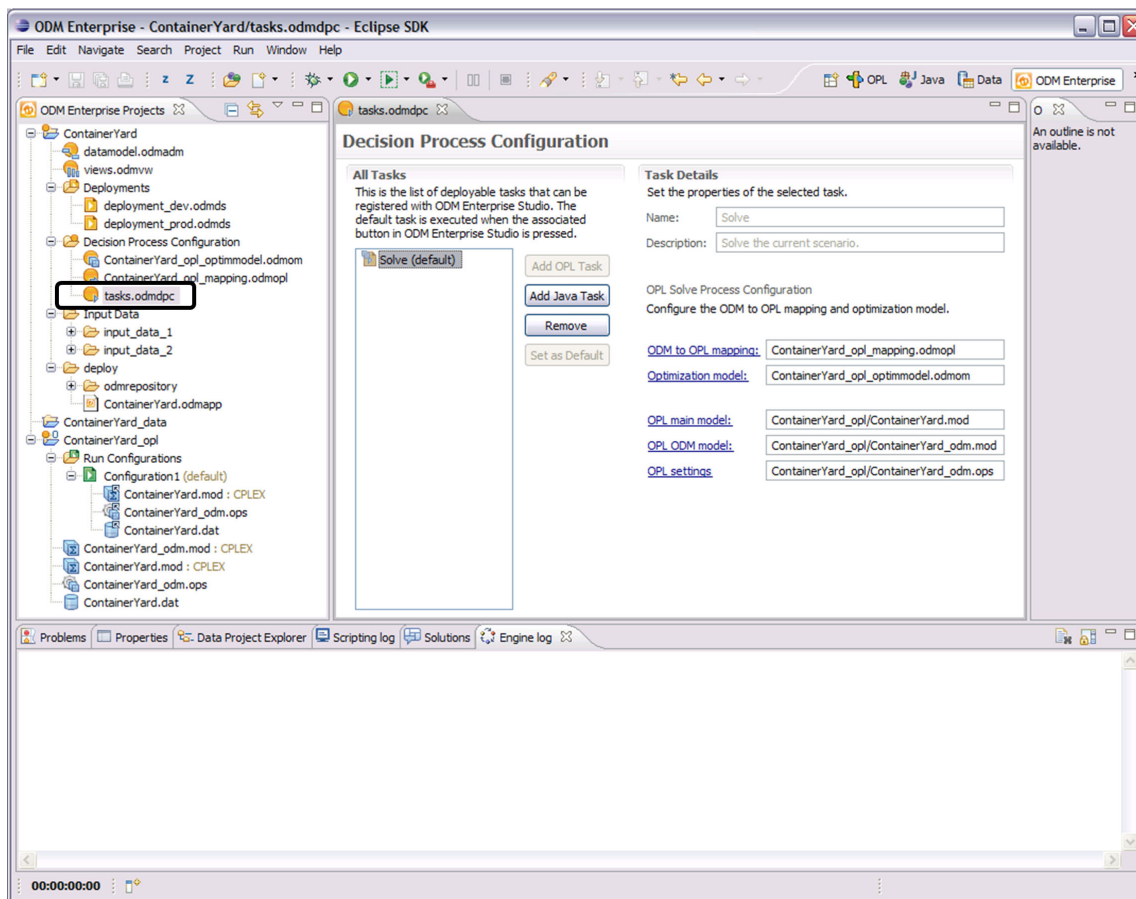


Figure 3-17 Configuring tasks

- ODM to OPL Mapping

This entity is denoted by the `<project_name>_opl_mapping.odmopl` file. It defines the mapping of the ODM Enterprise application data model tables to OPL entities.

- Optimization Model

Denoted by the `<project_name>_opl_optimmodel.odmom` file, this entity is used to create requirements, goals, and rules for your ODM Enterprise application.

- OPL Main Model

Denoted by the `<opl_project_name>.mod` file, this entity represents the optimization model that is called when running the OPL task.

- OPL ODM Model

Denoted by the `<opl_project_name>_odm.mod` file, this file is automatically generated by ODM Enterprise. It is based on the mapping of artifacts between the ODM Enterprise application data model and the OPL model. This file contains the inputs and outputs to the OPL project.

- OPL Settings

Denoted by the `<opl_project_name>_odm.ops` file, this entity points to the settings file that the ODM Enterprise application uses during run time. This file controls the underlying engine's (CPLEX Optimizer or CP Optimizer) behavior during solve. Any non-default setting is typically used to tune the engine's performance for model specific tendencies, or change tolerances as is required by the ODM Enterprise application.

- Java task

A Java task calls custom Java code to do specific operations to meet the application's needs. A typical use of such a task would be to perform operations before or after the solution is determined. Such tasks can also be used to start custom solves involving data exchange with the ODM Enterprise Scenario Repository. These solves can use the CPLEX Java API directly instead of using OPL, and can handle abort and progress events differently. After a Java task is created by using the editor, perform these steps:

- a. Populate the following tabs that show up in the editor:

- Implementation

This tab defines the Java class to implement this task. Make sure that the JAR file that contains the class is added to the ODM Enterprise project by using the Additional JARs and Resources tab of the Deployment Configuration Editor.

- Input

The input tab is used to select the input data that is used by the Java task. The input data that gets displayed in this tab comes from the project's Application Data Model.

- Output

The output tab is used to select the output data that is used by the Java task. Like the input tab, the output data displays output data that comes from the project's Application Data Model.

- Parameters

Use this tab to define parameters that are passed to the Java task to further customize the behavior as needed.

b. Add solve commands for the Java task to ODM Studio

Adding the Java task to the task list does not automatically create the ability to run the task in ODM Studio. You must create code that adds a corresponding solve command to ODM Studio menus.

For more information, click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Basic usage → Customizing an ODM Enterprise application → Decision Process Configuration— adding tasks → Adding a Java task** in the online ODM Enterprise InfoCenter at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

3.1.11 Solve

Solve originates in the underlying engine's (CPLEX Optimizer or CP Optimizer) method called "solve". It refers to the process of finding and improving solutions to the optimization model driven by data that is provided by the Application Data Model layer. While the system optimizes the user specified goals, a window that displays *Solve Progress* is displayed in ODM Studio. This window displays charts of the combined objective by default that show the progress as the solver tries to find and improve upon solutions for the scenario. It also provides indicators to show if a solution is found and allows the user to stop the solve or relax more requirements.

ODM Enterprise provides the user with the solving as shown in Figure 3-18, which can be configured in the Models tab of the *.odmom file.

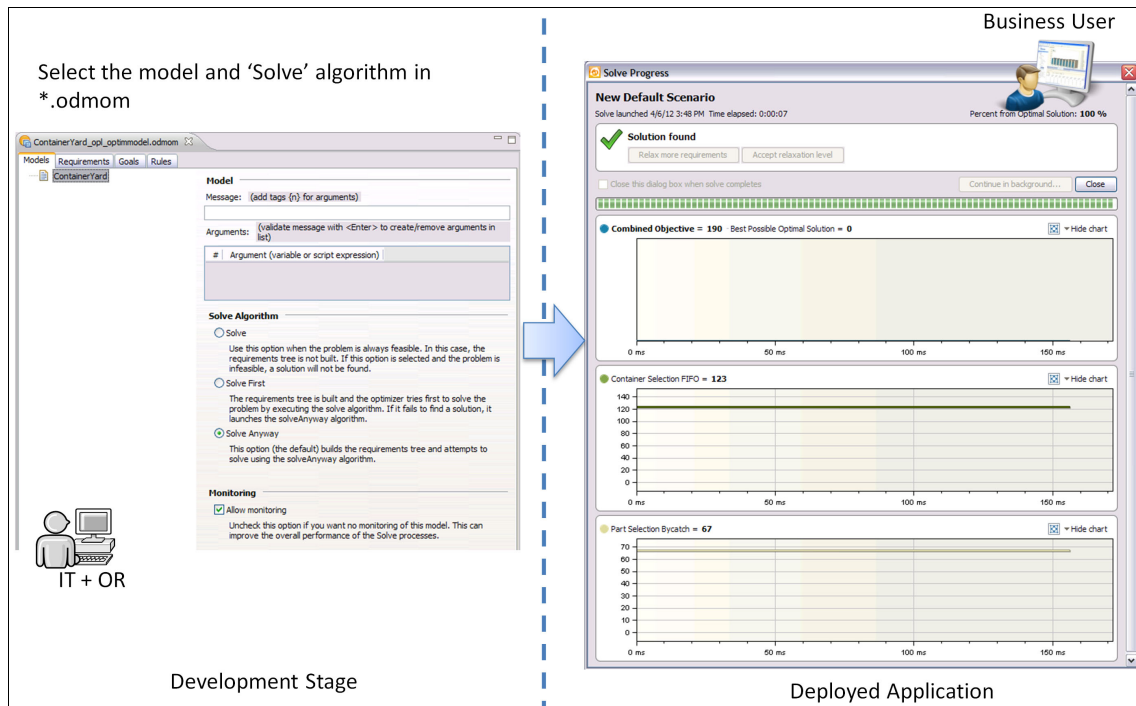


Figure 3-18 Configuring models and solving algorithm

► Solve

Use this option when you are confident that the generated problem is feasible. Because the requirement tree is not built, the optimization engine (CPLEX Optimizer or CP Optimizer) will declare the problem to be infeasible if it is indeed infeasible. If the problem is feasible (and no relaxations are required), the solver attempts to minimize or maximize the goals in the presence of constraints and variable bounds (requirements). These requirements are defined in the optimization model and specified by the business user.

For more information about the tree structure of requirements in ODM Studio, click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Basic usage → Customizing an ODM Enterprise application → Working with requirements in the IDE → How requirements are displayed in ODM → Requirement tree** in the ODM Enterprise InfoCenter at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

- Solve anyway

This is the default algorithm that is used by ODM Enterprise when CPLEX is the solving engine. It has the capacity to handle infeasible problems. If the original problem is feasible, this algorithm performs the same as the previous solve algorithm. However, if the problem is infeasible, the algorithm builds upon the *feasopt* algorithm in CPLEX. The solver uses the priorities that are provided by the user for the requirements that can be relaxed. The *feasopt* algorithm itself works in two phases. In phase 1, the solver tries to minimize the number of relaxations that are required to make the infeasible problem feasible. Phase 2 then solves the original problem by using the minimum relaxations for the objective function's optimality condition.

Solve anyway systematically adds constraints in the “relaxable pool” of constraints, starting with those that have the lowest priorities, until it finds a feasible solution. If the algorithm cannot find a solution even after adding all the constraints to the “relaxable pool”, it indicates that the problem has no solution.

- Solve first

This hybrid strategy first tries to solve the problem by the solve algorithm. If that fails to find a solution, the solve anyway algorithm is used.

3.1.12 Deployment

Deployment refers to the process of packaging an ODM Enterprise application into an executable (*.odmapp file). This file is then distributed to the business user along with the ODM Enterprise runtime software that allows users to use the application at their workstations.

There are many options when you are deploying your ODM Enterprise application. You can choose to deploy it as a single-user or a multi-user application. The ODM Enterprise Scenario Repository can be in the default database, or it can be stored on a shared database on a network server. In addition, you can include more data sources and more JAR files or resources to support customization.

Each ODM Enterprise application can have multiple deployment configurations. For example, the development configuration might create an ODM Enterprise application that runs in single-user mode (individual deployment configuration) against a local ODM Scenario Repository database. The production configuration might run in a multi-user mode (enterprise deployment configuration) against a centralized shared ODM Scenario Repository database. Setting up each of these configurations can be done by using a Deployment Configuration Editor.

When you create your ODM Enterprise application, two default configuration files are added to the tree in the ODM Enterprise Project Explorer:

- `deployment_dev.odm`

This file is used to define the deployment configuration that is used by the ODM Enterprise application in development mode. The configuration is defined by using the default embedded database.

- `deployment_prod.odm`

This file is used to define the deployment configuration that is used by the ODM Enterprise application in production mode. It uses an external database for the ODM Scenario Repository, and possibly other components such as the Optimization Server and the Data Server.

The Deployment Configuration Editor in the ODM Enterprise IDE provides the user with the following tabs as shown in Figure 3-19. These tabs control the various aspects of the deployment process.

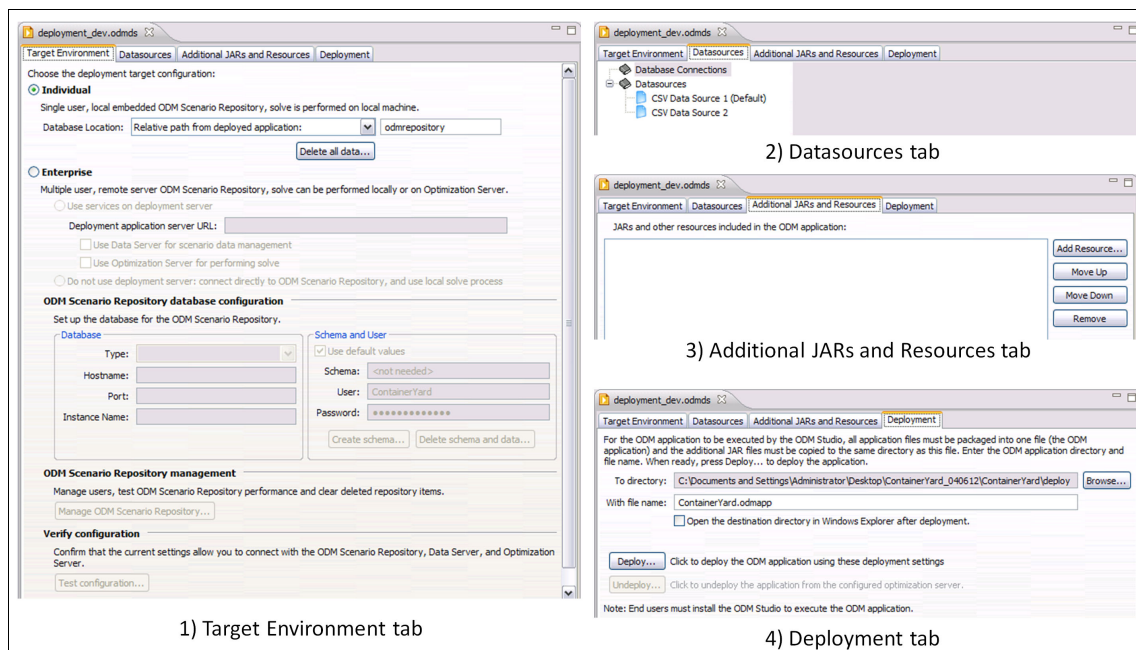


Figure 3-19 Deployment options

- Target Environment tab

This tab defines the location and type of the database that contains the ODM Scenario Repository, and whether remote optimization services or scenario services are being used.

The tab is further divided into two main areas:

- Individual

This area is used to create an individual or standard desktop deployment configuration. It uses the embedded database as the ODM Scenario Repository, and it solves by using the local optimization engine.

- Enterprise

This area is used to create the other deployment configurations, enterprise with local optimization and enterprise with remote optimization. Each configuration creates an ODM Enterprise application that is deployed by using the multi-user model. The application therefore requires an external database to serve as the ODM Scenario Repository. The application can include the use of the Data Server for managing scenarios, and the Optimization Engine for solving locally or remotely running on a centralized application server such as WebSphere Application Server. The various areas are used to specify these options:

- Either a URL of the deployment server to use remote optimization or scenario services, or a direct connection to the ODM Scenario Repository to solve locally.
- The database portion of the ODM Scenario Repository database configuration, which is used to define the location and type of external database that is used as the ODM Scenario Repository.
- The ODM Scenario Repository management area, which is used to manage ODM Enterprise application users and passwords. It is also used to measure performance against the ODM Scenario Repository, and to clear scenarios that are moved to the ODM Scenario Repository's recycle bin.
- The Verify configuration section that is used to check the connection to the ODM scenario repository and to optimization and scenario services, and validate the configuration.

- Datasources tab

Used to define the data sources for your ODM Enterprise application. Data sources are the means to populate tables in the Application Data Model layer for ODM scenarios. They can be used to create a scenario, modify parts of a scenario, and create and modify a copy of a scenario.

- Data source types can be classified into these types:

- <fileSource> - for CSV/TSV files
- <dbSource> - to fill the table with SQL requests
- <customSource> - to use Java code to fill the tables

- Within the ODM Enterprise IDE, you can use graphical editors only to create database (SQL) and CSV file datasources. For more information, click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Basic usage → Customizing an ODM Enterprise application → Working with the Deployment Configuration Editor → The Datasources tab** at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

- ODM Enterprise datasources can also be a combination of CSV/TSV files, SQL, and custom Java code. The datasources are described in XML files with the extension `.odmfs`.

► **Additional JARs and Resources tab**

If the ODM Enterprise application uses custom views or code for datasources or tasks, use this tab to include the required JAR files and resources during deployment.

► **Deployment tab**

After all the other deployment settings and options are set, use this tab to deploy the application by specifying the folder and file name that contain the executable `.odmapp` file.

For more information, click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Basic usage → Customizing an ODM Enterprise application → How to deploy an ODM Enterprise application** at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

3.1.13 Collaboration

This section addresses the *collaboration* features in ODM Enterprise, which revolve around workspaces and scenario sharing. As explained in 3.1.12, “Deployment” on page 82, an ODM Enterprise application can either be deployed in a single-user or multi-user fashion. In a single-user environment, collaboration is not an issue, but in a multi-user environment, collaboration needs to be addressed.

In a typical multi-user ODM Enterprise application, a user might be comparing an active scenario, including changing the data, rules, requirements, priorities, and so on, with a baseline or reference scenario. Even though the reference scenario is not being modified by the current user, it must still be locked so that other users cannot change it, to maintain integrity in the system. Thus, when multiple users are working with a shared repository on a central database, scenario locking is a requirement. When a scenario in ODM Enterprise is locked, it is available for

viewing (read-only) or to be set as the reference scenario by other users. However, it cannot be edited or solved by other users.

An ODM Enterprise scenario is locked in these circumstances:

- ▶ One of the users, whose workspace contains the shared scenario, clicks the scenario in the Scenario Overview pane and makes it the active scenario. The scenario is then displayed as locked (read-only) to other users.
- ▶ One of the users, whose workspace contains the shared scenario, sets the scenario as the reference scenario. The scenario is then displayed as locked (read-only) to other users.
- ▶ A user solves a shared scenario, either in the foreground or in the background in the solve queue. The scenario is then displayed as locked to the other users until the solve process is complete or is canceled.
- ▶ A user solves a scenario, whether it is shared or not, and it is running in the foreground or background. The scenario is then displayed as locked (read-only) to the user until the solve process is complete or is canceled. The user can only open and work with other scenarios at that point.
- ▶ A user submits a scenario to run in the background in the solve queue, even if it is not solving yet. The scenario is then displayed as locked (read-only) to the user and other users until the solve process is complete or the scenario is canceled.

An ODM Enterprise scenario is unlocked in these circumstances:

- ▶ No user who is accessing the shared scenario is directly viewing, editing, or solving it.
- ▶ A scenario that was being solved by a user completes the solve process or is canceled.
- ▶ A user who was previously editing a scenario switches to another scenario and has no pending commits to it.
- ▶ Scenarios are available for editing if they are not currently set as reference by any of the users.

Along with these use cases, there are advanced menu options when you are running ODM Enterprise in multi-user mode. These options handle cases where a user tries to access a shared scenario while another user is solving the scenario.

Details on these advanced topics can be found in the online ODM Enterprise InfoCenter. Click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Basic usage → Customizing an ODM Enterprise application → Collaboration Features in ODM Enterprise → Tutorial -**

Creating a multi-user Nurses Demo → ODM in multi-user mode (more advanced) at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

Table 3-2 provides a list of actions that can be performed by a user on a scenario that is locked by that user. These actions are compared with what other users can perform on the same locked scenario.

Table 3-2 Allowable actions on locked scenarios

Action	Can you do this when you have the scenario locked?	Can other users do this when you have the scenario locked?
View scenario	Yes	Yes - but they might see inconsistent data if you change the scenario
Edit scenario	Yes - pending changes are saved automatically when you solve, set the scenario as reference, check data, or save manually	No
Save scenario	Yes	No
Solve scenario	Yes - pending changes are saved automatically	No
Duplicate scenario	Yes	Yes
Delete scenario	Yes	No
Rename scenario	Yes	No
Set as reference	Yes - on a per user basis. Pending changes are saved automatically	Yes - on a per user basis, but they might see inconsistent data if you change the scenario
Check data	Yes - pending changes are saved automatically	No
Export scenario set	Yes	No
Compare scenarios	Yes	Yes

Action	Can you do this when you have the scenario locked?	Can other users do this when you have the scenario locked?
Refresh workspace	Yes	Yes
Refresh scenario	Yes - but not needed if you have the scenario locked	Yes - to refresh the scenario if they see inconsistent data

Figure 3-20 shows a list of scenario icons alongside their meaning.

Icon	Meaning
	Scenario is being edited by me.
	Scenario is being edited by someone else.
	I have pending changes to the scenario.
	I need to update the scenario; someone else has made changes.
	Scenario has errors.
	Scenario has warnings.
	Scenario is waiting to be solved.
	Scenario is being solved.

Figure 3-20 Interpreting scenario icons

3.2 ODM Enterprise architecture

The modular nature of ODM Enterprise allows for many different deployment configurations. This flexibility supports a wide variety of business requirements and the different needs of its users and the network topology of their business. In its most basic form, an ODM Enterprise application follows a client-server architecture, with components that use web services to communicate with each other.

As shown in Figure 3-21, the Java Swing rich client, ODM Studio, interacts with both Data and Optimization Servers by using SOAP over HTTP. The database communications are done by using Java Database Connectivity (JDBC). A detailed description of each component is provided in the following sections.

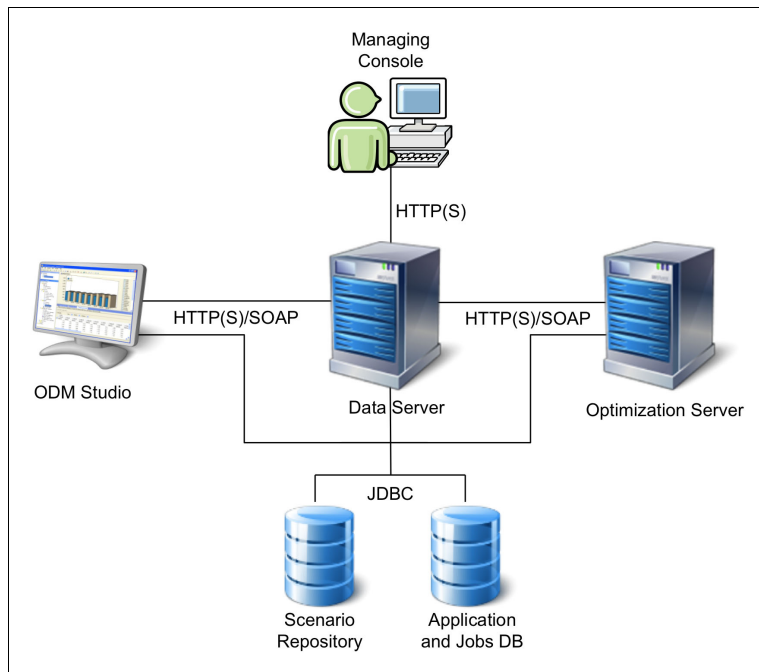


Figure 3-21 Basic ODM Enterprise architecture

For more information about ODM Enterprise development and runtime architectures, see Chapter 4, “Transforming business requirements into an ODM Enterprise application” on page 111.

ODM Enterprise uses the Java Enterprise Edition (Java EE) platform architecture to seamlessly integrate within existing application server deployments, such as the IBM WebSphere® Application Server environment. In its most basic form, a typical ODM Enterprise deployment looks like the diagram in Figure 3-21. However, it can take many different forms to support business requirements, as shown in Chapter 6, “Case study: An insurance company” on page 187 and in Chapter 7, “Case study: A manufacturing company” on page 223.

3.2.1 Optimization Server

Optimization engines use two basic items to function: An *optimization model* that contains your business rules, and a *data set* containing the data to be “optimized”

(or “solved”). For example, a distribution company that wants to optimize the deployment of shipping trucks, packages, and drivers will provide data that describes each component to the optimization engine.

The ODM Enterprise Optimization Server is a centralized, server-based implementation of the IBM ILOG CPLEX and CP Optimizer solving engines. It is required for solving optimization models and to run algorithms to support decisions by planners who use the ODM Enterprise Client in an enterprise configuration.

The Optimization Server is made up of two main application components. The Optimization Server Management application provides the entry point to perform these tasks:

- ▶ Deploy and manage ODM Enterprise applications in the server
- ▶ Submit jobs
- ▶ Interact with ongoing optimization jobs

The Optimization Server Job Processor application (or Optimization Core, which is depicted in Figure 3-22) is responsible for performing the actual optimization. By default, the processor application is able to process three optimization jobs in parallel. This is a configurable parameter in the processor application settings.

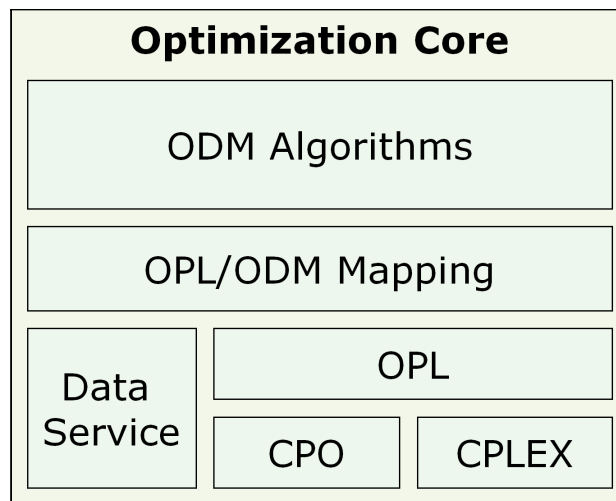


Figure 3-22 The Optimization Core architecture

The Optimization Core is composed of the following components:

- ▶ *IBM ILOG CPLEX CP Optimizer (CPO) engine*

The Constraint Programming Optimizer solver is used primarily to solve scheduling problems for personnel, machines, or process steps.

► *IBM ILOG CPLEX Optimizers (CPLEX) engine*

The IBM mathematical programming technology that provides solvers for linear programming, mixed integer programming, quadratic programming, and quadratically constrained programming problems.

► *OPL*

The *ILOG Optimization Programming Language* is a modeling language for combinatorial optimization that aims at simplifying the solving of these optimization problems.

► *OPL/ODM Mapping*

This component provides the mechanisms to automatically generate an ODM Enterprise OPL model that is based on the structure of the ADM model. This mapping provides these advantages:

- OPL data structures, such as arrays and tuple sets, that are mapped to input or output data tables and graphical views in ODM Enterprise.
- Mathematical constraints from your OPL model can be exposed as ODM Enterprise requirements (soft constraints).
- The objectives of your mathematical model are mapped to ODM Enterprise goals.
- Decision variables in the model are represented as solution views in ODM Enterprise.

► *ODM Algorithms*

When CPLEX is used, three solve algorithms are available:

- Solve
- Solve first
- Solve anyway

For more information about these algorithms, see 3.1.11, “Solve” on page 80.

► *Data Service*

This component provides data interfaces to the optimization process.

Multiple Optimization Servers can be used in high performance implementations where short processing times and multiple concurrent optimization processes on a multi-user environment are required.

3.2.2 Data Server

The ODM Enterprise Data Server provides a series of data and process-related services to ODM Enterprise applications. These services facilitate

communication between the ODM Enterprise Client and the ODM Scenario Repository database, and between the Data Server and the Optimization Server (on enterprise configurations). The diagram in Figure 3-23 shows the services provided.

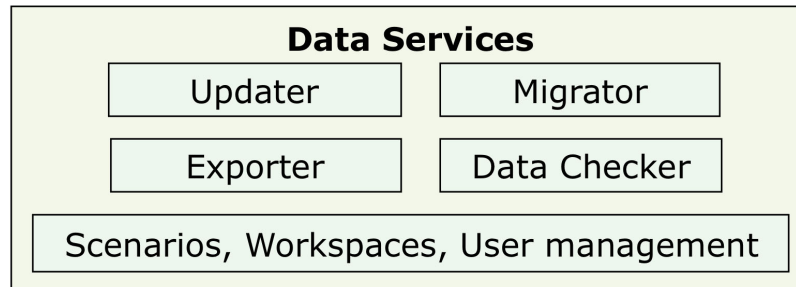


Figure 3-23 Data Services

► *Updater*

The Updater is the module that brings data from external databases into the scenario database. During application development, the developer configures data connections and data sources that will be used by the Updater to fetch external data. For more information about how to configure data sources, see the online ODM Enterprise InfoCenter. Click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Basic usage → Creating an ODM Enterprise application → Working with datasources → Creating a datasource (scenario updater) to load data from an external database**.

► *Exporter*

The Exporter supports custom-built exporters (see 3.5.3, “Exporting data” on page 101) that export data out of ODM Enterprise.

► *Migrator*

The Scenario Migrator provides application migration services to ensure that users can still open the scenarios that were created with previous versions of an ODM Enterprise application. Between two releases of an ODM Enterprise application, you might change the optimization model, or the goals, requirements, or rule definitions. You might, for example, change the type of data, or add an index to an array. The Migrator helps you to handle these changes. Custom Migrators can also be written to handle special cases that the default Migrator cannot handle.

► *Data Checker*

The Data Checker validates the data integrity of a scenario before the scenario is solved, or when a search for goal bounds is triggered. These data

integrity checks search for missing values, duplicate primary keys, invalid foreign keys, and out of bound values.

► *Management*

The generic Management module handles the data operations that involve scenarios, users, workspaces, and other application elements.

To manage and track optimization jobs, ODM Enterprise ships with a web-based Enterprise Dashboard (installed with Data Server and Optimization Server). This dashboard allows system administrators and power users to monitor server utilization and obtain job status information. This ease of control is important in environments that run multiple ODM Enterprise applications that share Data and Optimization Servers. It is also important in high-performance implementations where multiple Optimization Servers are configured to work with a single Data Server. In these special cases, the application management module manages a queue of optimization jobs, distributing the jobs among Optimization Servers, and handling processing errors. Figure 3-24 depicts the Dashboard with two registered applications and no running jobs.

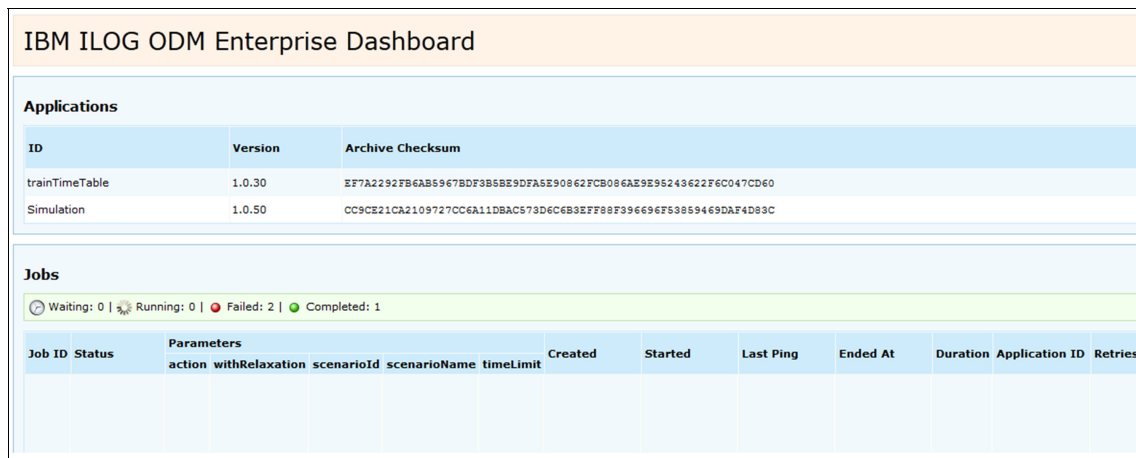


Figure 3-24 ODM Enterprise Dashboard

Tip: On a default installation, the Enterprise Dashboard can be accessed at: <http://<servername>:9080/odme/console>.

3.2.3 ODM Studio

The ODM Studio is the graphical user interface of an ODM Enterprise application. This GUI enables users to create scenarios, initiate optimization

processes, and view input and output data sets, and charts and graphs. ODM Studio comes in two editions: ODM Enterprise Planner, for workstation deployments, and ODM Enterprise Client, for client-server deployments.

ODM Studio is a framework that developers can customize to suit specific business needs. For example, a hospital administration company that uses ODM Enterprise to optimize the deployment of medical personnel and equipment can transform the ODM Studio into a management dashboard. This dashboard can provide users with a “before and after” view of the data along with specialized charts, maps, and even animations.

Figure 3-25 depicts the components that make up the ODM Studio client.

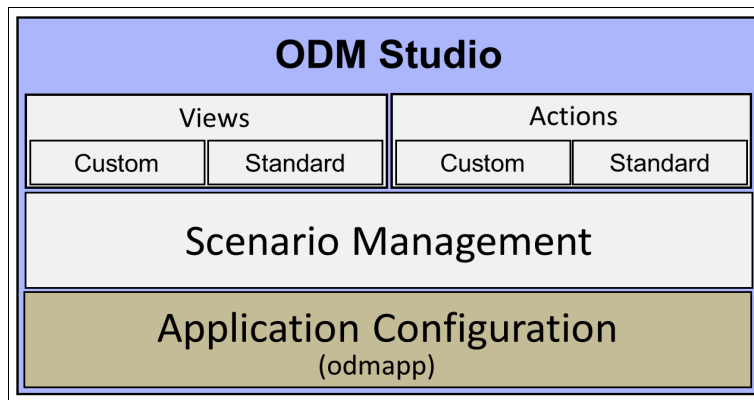


Figure 3-25 ODM Studio components

► *Application Configuration*

Produced during application deployment, the .odmapp file contains information vital to application functionality. It stores settings such as database connection information, application server addresses, and Java library dependencies.

► *Scenario Management*

This module allows the user to create, delete, copy, and solve scenarios, as well as working with the Data Server, which provides the required data interfaces.

- ▶ Actions are processes that are triggered from menus. Global actions are available from many parts of an ODM Enterprise application, and can be used in custom menus or other GUI elements. Two types of actions exist:
 - Standard

This type provides built-in actions such as copy, paste, and cut.
 - Custom

This type allows actions that are built to operate with a specific custom *View*. For example, custom code can be developed to display details of a data element that is started by a configured custom *Action*.
- ▶ Views

One of the most powerful features in an ODM Enterprise application is the ability to display data-driven, rich views. The options include pivot tables, maps, graphs, and charts. Depending on the application requirements, ODM Studio can display two types of views:

 - Standard

ODM Enterprise ships with many standard, configurable views that include bar, pie, polyline, and many others. These views are created by editing the `views.odmvw` configuration file, and do not require any custom Java code.
 - Custom

Business users, architects, and developers can take advantage of this feature by using external Java components and third-party tools to develop rich and sophisticated views. These views can include interactive maps, Gantt charts, and animations.

You can find custom view examples in “Custom views” on page 71.

3.3 ODM Enterprise API

ODM Enterprise ships with a comprehensive set of Java APIs that expose most of the services that are provided by the components. This configuration provides in-depth customization options for both the presentation and processing layers. For example, custom data processing can be incorporated into the default optimization process. Listeners (“hooks”) allow developers to tap into specific application events such as solve or menu operations to augment the standard functionality.

For more information, see the online ODM Enterprise InfoCenter. Click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise Reference Manuals → Java API Reference Manual for ODM Enterprise** at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

The API exposes most of the services that are used by ODM Enterprise. Figure 3-26 shows the most common APIs. A quick review of each component follows the diagram.

Data Service (ilog.odm.datasvc)	ODM Service (ilog.odm.service)
Processing Service (ilog.odm.processingsvc)	DOM (ilog.odm.dom)
Task Service (ilog.odm.tasksvc)	Studio Actions (ilog.odm.studio.actions)
Scripting (ilog.odm.scripting)	Studio Settings (ilog.odm.studio.settings)
Decision Process (ilog.odm.config.dpc)	Studio Views (ilog.odm.studio.views)

Figure 3-26 ODM Enterprise API Model

► *Data Service*

The Data Service exposes data-related services for manipulating scenarios, workspaces, tables, and other data-driven artifacts.

► *Processing Service*

This service provides the developer with the ability to set up optimization tasks, submit them as jobs, and monitor their execution.

► *Task Service*

This service is used to run standard or custom tasks, in a separate thread, with user feedback. Tasks are submitted to the Task Service for processing.

► *Scripting services*

These services provide classes that are used to access ODM services from ODM scripting.

► *Decision Process*

The Decision Process provides classes that are used to configure the decision process by declaring tasks of an ODM Enterprise application. Each task has some attributes and parameters that can be defined by using the decision process configuration editor. The task declarations are used at run time in read only mode by the processing service and its API.

► *ODM Service*

The top-level service is required to obtain a reference to the ODM Enterprise application. It allows you to create stand-alone programs that can access ODM data and trigger and monitor solve operations. The ODM Service classes enable developers to use ODM Enterprise as a service, without the ODM Studio graphical user interface.

► *DOM service*

The DOM service provides classes to manipulate the Domain Object Model. For more information about the DOM, see 3.4, “Domain Object Model (DOM)” on page 97.

► *Studio Actions*

This service provides an action handler class for defining new actions in ODM Enterprise.

► *Studio settings*

The Studio settings provide classes to manipulate the local user settings of ODM Studio.

► *Studio views*

The Studio views provide basic classes to implement custom views in ODM Studio. For more information, see 3.2.3, “ODM Studio” on page 93 and “Custom views” on page 71.

3.4 Domain Object Model (DOM)

The Domain Object Model is based on the ODM Enterprise Application Data Model (ADM). It provides a simple, automated way to map tables to Java classes, columns to attributes, and foreign keys to bidirectional references. You can customize the generation by setting properties in the Object Model tab of the Application Data Model editor.

Generating and using a DOM has these main benefits:

- ▶ The ability to extend the generated model by adding operations to classes that will be kept through multiple update and generation cycles.
- ▶ The ability to populate the object model at run time from a scenario or from files. Selected objects can be stored in the scenario or exported to files.

The object model can simplify the implementation of custom code. It does so by providing a typed data model, management of bidirectional references, automated mapping to scenario data, and import from and export to files.

In addition to the classes generated from the tables, a *collector* class is generated to act as the façade of the generated object model. The collector class grants access to all the DOM objects by class, and provides functions to create new objects.

After the generated Java code is compiled, you can perform these tasks:

- ▶ You can connect the generated classes to a scenario so that you can access the scenario data by using the DOM API without referencing table and column names. Consequently, the Java code that is required to access the scenario data is more concise.
- ▶ You can use Java error management, type checking, code completion, and other Java services.
- ▶ The DOM API also provides you with persistence services so that you can read data from and write data directly to a scenario. It also provides an API to read and write data to files so that unit tests can be easily implemented. This configuration simplifies the exchange of data.

To generate a DOM, open the data project, and navigate to the schema you want. In the Properties window, enter the required information and save the properties to begin the code generation.

Default names are always available based on the contents of the schema. You can also customize the generated Java code to produce code that is more in line with the intent of the model. By default, the generated Java class names are derived from the table names you defined in your ADM. If your table names are incompatible with generated class names, define the class or attribute names by using the object model properties. Otherwise, you might obtain errors from the DOM generation. You can also modify these names if you find that the generated name is not appropriate. Any errors reported during the generation are listed in the Problem View.

For more information, see the online ODM Enterprise InfoCenter. Click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Advanced usage → Working with the Domain Object Model →**

Properties for customizing the Java code generated by the Domain Object Model at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

Figure 3-27 shows the selected Schema and the Object Model options in the Properties window.

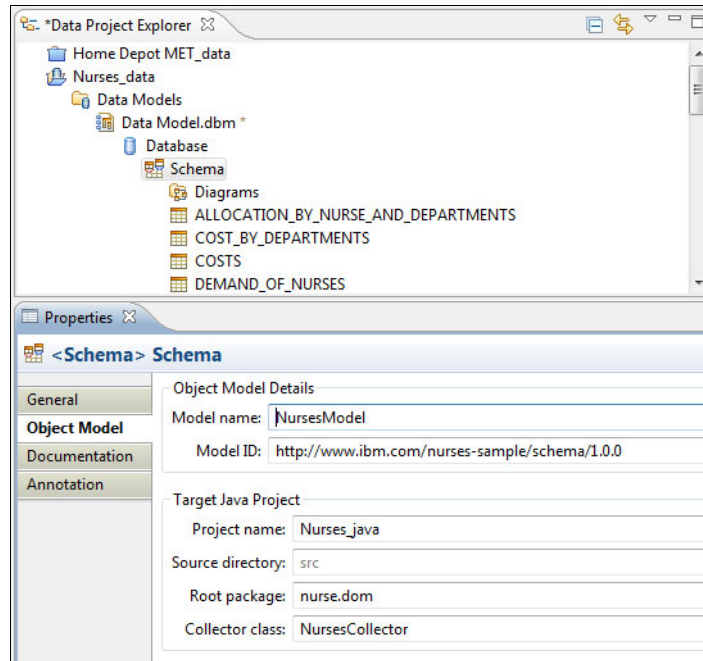


Figure 3-27 Properties for generating an ODM

3.5 Data architecture

ODM Enterprise requires two databases to operate:

- The scenario database (also known as the Scenario Repository)
- The Application and job database.

3.5.1 Scenario Repository

The Scenario Repository is a relational database where ODM Enterprise stores its operational data. Created and installed during application deployment, it is from there that the optimization engine collects the input data and where it saves

the optimization results. Usually, the Scenario Repository is modeled during the application development process, but no manual intervention is required to create it. ODM Enterprise creates the database automatically during deployment.

The Scenario Repository is modeled to support the application and optimization processes. It is never accessed from outside the application through traditional database connection mechanisms (that is ODBC, JDBC, and so on). Rather, it is accessed through the ODM Enterprise API when required.

This database stores scenario data such as insurance claims and personnel data. It also maintains a list of all of the scenarios that have been created, their properties (source tables, name, creation date, owner, and so on), and other control elements. These control elements include scenario locking information, users, and GUI elements like folders and workspaces.

When the user creates a scenario, the data is transferred (imported) from corporate databases (or from a staging database) to the Scenario Repository. This process creates a snapshot of the data and freezes it from further updates from business systems. This data snapshot allows business planners to perform consistent what-if analysis at any point in time.

Using a staging database

For a typical ODM Enterprise implementation, a staging database is used to collect and assemble the required data from the corporate systems and databases into a single repository. This approach offers the following advantages:

- ▶ **Maintenance**
Only a single connection is required for the ODM Studio to retrieve data.
- ▶ **Control**
Easier to develop and maintain, giving system developers more control over the optimization data model.
- ▶ **Performance**
In implementations with a high volume of data, performance can be affected if data needs to be collected from several, disparate sources every time a scenario is created.
- ▶ **Quality**
Data often needs to be transformed and cleaned before it can be used by the optimization engine. Custom ETL (extract, transform, and load) routines can scrub and structure the source data while loading the staging database.

3.5.2 Importing data

Because of its specific design and structure, special processes are used to import and export data from the Scenario Repository. As the system's operational store, the Scenario Repository depends on two components: The *Importer* and the *Exporter*. The Importer is responsible for bringing data into the database, and the Exporter is used to transfer data out.

In the example that is used in Chapter 6, "Case study: An insurance company" on page 187, data from the company's business systems is imported into the Scenario Repository automatically. This process occurs when the user creates a default scenario (**File** → **New** → **New Default Scenario**) by using the ODM Enterprise Client.

Figure 3-28 illustrates how the data flows during import into the Scenario Repository.

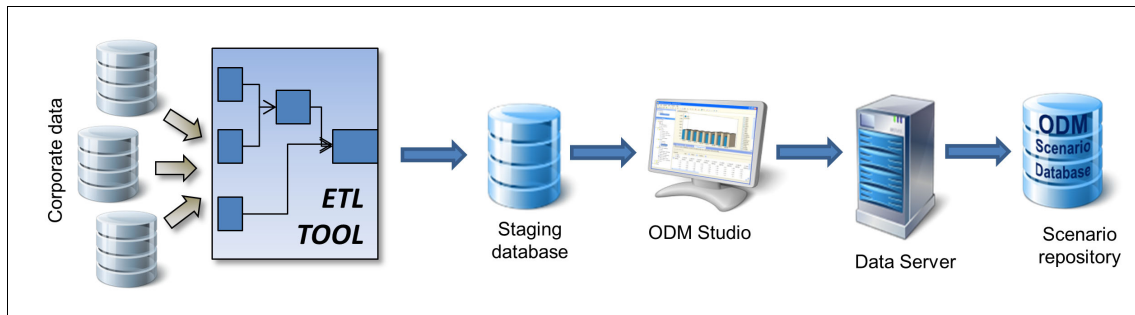


Figure 3-28 Import data flow

The use of a staging database in this process is optional but generally there are advantages of using one in enterprise deployments. After ETL processes are loaded the staging database with all the relevant data, the ODM Studio retrieves the data from staging. It then interacts with the Data Server to load it into the Scenario Repository. This process is done seamlessly, and does not require further intervention by the user.

3.5.3 Exporting data

After evaluating different outcomes, either by manipulating scenario parameters or by creating different scenarios, the business planner can then export the scenario data (and optimization results). The data can be exported by using two methods, *default* and *custom*.

Default method

ODM Enterprise ships with a default export process that dumps the entire scenario data and metadata to disk. Data files are exported as text, and table models and other metadata are exported as XML. This predefined format allows other ODM Studio users to import the scenario. This is a useful feature when you are working in stand-alone implementations (that is, no central scenario database). For more information about the different implementation types, see Chapter 5, “Solution design and management” on page 165.

When you are using this method (**File** → **Export Selected scenario**), ODM Studio asks for the path of an empty directory to save the data. The process produces the following files:

- ▶ `ContextCollection.xml` and `MappingFileID.xml`: These files contain internal scenario information.
- ▶ `MappingFileTableId.xml`: This file contains cross-references of files and their corresponding tables. This file is not always created.
- ▶ `<scenario name>.zip`: This file contains scenario data, its goals, constraints, requirements, and more. This file can be very large depending on the volume of data.
- ▶ `RelationalModel.xml`: This file contains the data model metadata, which includes field names, data types, and field length.
- ▶ `VersionInfo.xml`: This file contains version information about the scenario.

Custom method

The ODM Enterprise API allows for the development of custom processes. These processes can export scenario data in several ways. These ways include exporting data directly to an external database (a data warehouse, or perhaps a business intelligence repository) or in any other format such as XML or CSV. A custom exporter class must implement the interface `IloScenarioExporter` from the package `ilog.odm.datasvc`. This interface contains only one method:

```
public void exportScenario(IloScenario scenario, IloTemplateInstance
templateInstance, IloDataServiceContext dataService)
    throws Exception;
```

where

- ▶ `scenario` is a reference to the scenario object, giving the exporter access to the scenario tables.
- ▶ `templateInstance` gives access to the arguments of the message template.

- ▶ `dataService` provides access to:
 - A credential manager (to access passwords that are entered by the user)
 - An error reporter (to report errors and warnings that are encountered during the update to the ODM Issues panel)

Custom “exporters” can be configured to be run through a menu option in the ODM Studio. Configure them by editing the application deployment file (`deployment_prod.odm`) during application development and declaring a new exporter. Example 3-1 shows the configuration of a scenario exporter named “`HtmlScenarioExporter`” that exports selected data in HTML format.

Example 3-1 Configuring custom scenario exporters

```
<scenarioExporters>
  <!-- Add an exporter -->
  <scenarioExporter id="export2Html">
    <template>
      <message>Publish report {0}</message>
      <args>
        <arg type="string" id="channel" display="Select an output channel">
          <possibleValues tableColId="name" tableId="app.exportType"/>
        </arg>
      </args>
    </template>
    <!-- Specify the Java class that implements IloScenarioExporter -->
    <customExporter javaClass="export.HtmlExporterExample"/>
  </scenarioExporter>
</scenarioExporters>
```

The complete implementation of the `HtmlExporterExample` can be found in the ODM Enterprise Developer Edition `examples` directory in the file:

```
<Install_dir>\Developer\examples\Advanced\Data\ExportData\ExportData_
java\src\export\HtmlExporterExample.java
```

3.5.4 Application and Jobs database

Deployed automatically in enterprise implementations, the Application and Jobs database contains a list of deployed applications, optimization requests (jobs), and their current state. It is accessible to all server components (including management and processor applications). The Enterprise Dashboard (Figure 3-24 on page 93) shows a consolidated view of all registered applications and a list of current jobs that have been submitted by users.

It is possible to use the same database server for both the ODM Enterprise Scenario Repository and the Application and Jobs database. However, generally define different database users or schemas for the two databases to ensure better isolation.

3.6 Scalability and load balancing

Scalability is achieved by deploying the *Job Processor* application to additional servers. Although one server can already solve multiple optimization jobs in parallel, you can process more jobs by configuring more servers. Add servers when the limit is reached on the single server or more performance is required.

Load balancing and fail-over use Java Enterprise Edition fail-over protocols (clustering) that are provided as an added value by several Java EE vendors. If one Job Processor application is too busy or fails, another Job Processor application takes over.

The ODM Enterprise architecture is designed to allow the deployment of several Job Processor nodes in a cluster of systems, providing more scalability. Such deployments can take multiple forms, depending on the Java EE platform, the configuration of database and systems, and other factors. A typical clustering implementation is shown in Figure 3-29.

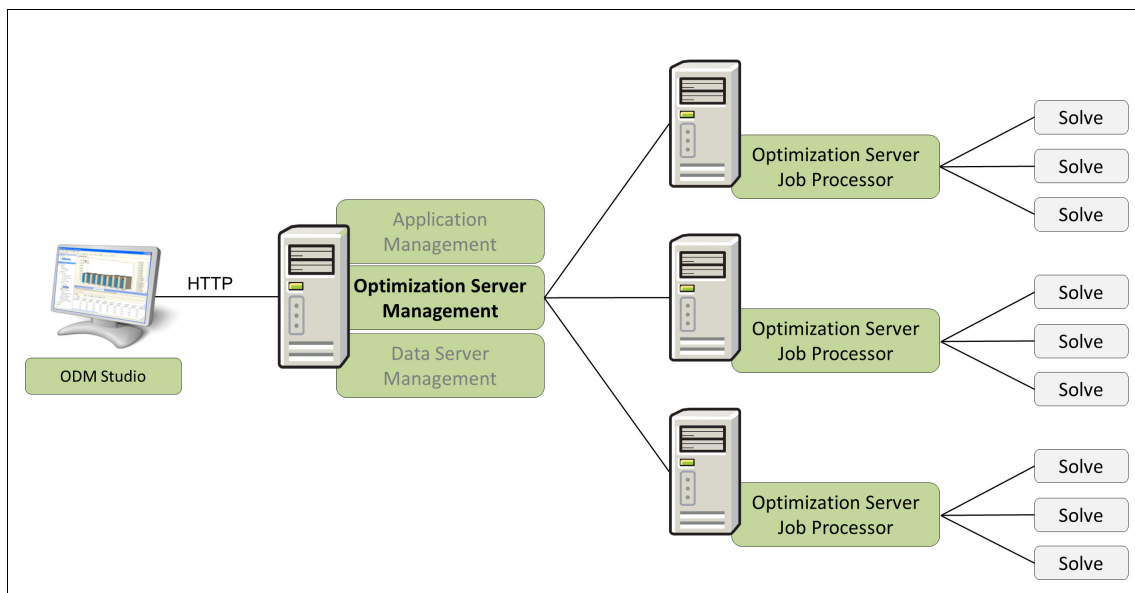


Figure 3-29 Clustering Optimization Servers

Several drivers can influence the topography in Figure 3-29 on page 104, including the number of concurrent optimization processes, volume of data, and performance requirements.

3.7 High availability

High availability (HA) is the ability of a system to provide service during outages, at acceptable or agreed upon levels, while masking those outages from users.

A typical HA environment has the following characteristics:

- ▶ Operates by having redundant nodes, which are then used to provide service when system components fail.
- ▶ The most common size for an HA cluster is two nodes, which is the minimum requirement to provide redundancy.
- ▶ HA cluster implementations attempt to use redundancy of cluster components to eliminate single points of failure.

If HA is a requirement in your optimization system solution, ODM Enterprise can be configured as depicted in Figure 3-30.

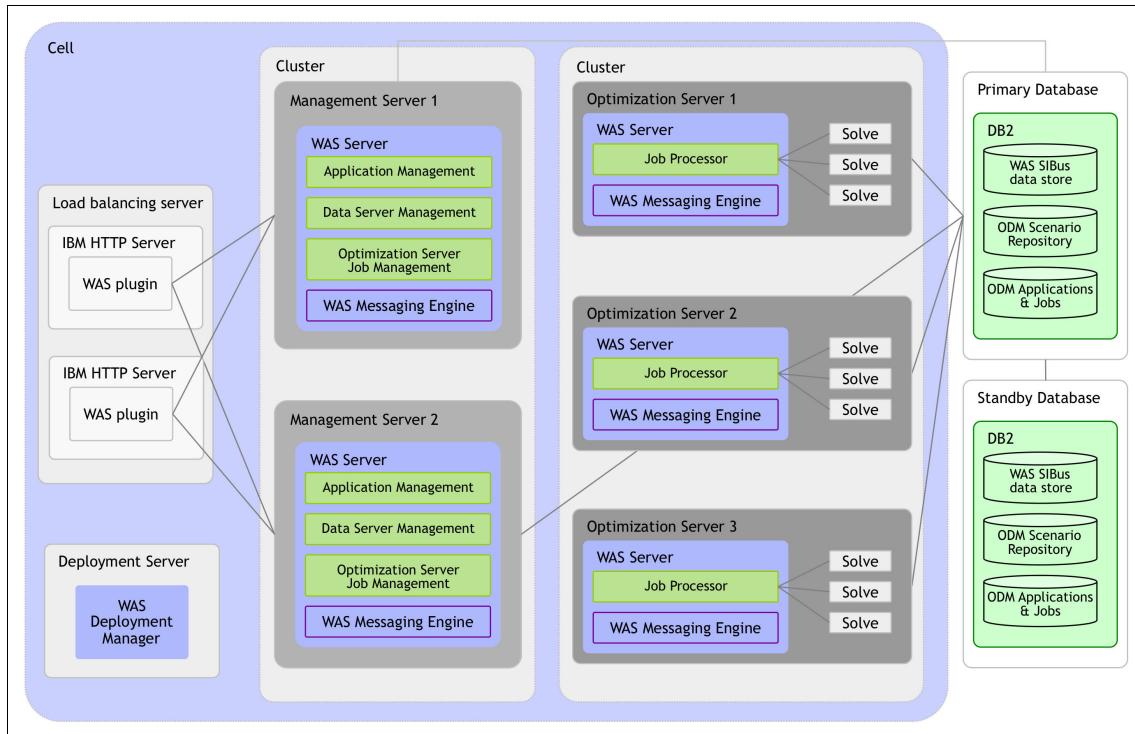


Figure 3-30 Sample HA topology

For scalability, enterprise quality Java EE applications are usually deployed and run in WebSphere Application Server Network Deployment clusters. The client requests are routed across the cluster, distributing the workload among all the application server processes that are members of the cluster.

The IBM HTTP Server Plug-in for WebSphere Application Server routes a request coming from an ODM Studio client to one of the Management Servers in the cluster. If the request involves a solve operation, the active Management Server routes the request to one of the Optimization Servers for processing. Scenario data is accessed primarily from the Primary Database. In a component failure (single or multiple), the requests are automatically routed and processed by the secondary components.

It is important to understand that processing nodes work independently, and are not load balanced. Processors pick jobs in job queues (database integration level) as they arrive if capacity is available. For more information about

WebSphere Application Server Network Deployment clusters, see the following Technical Library topic at the IBM developerWorks® site:

http://www.ibm.com/developerworks/websphere/techjournal/0906_banerjee/0906_banerjee.html

Tip: There is no full clustering support in the WebSphere Community Edition. However, it is possible to use the HTTP clustering to load balance the management modules, Data Server, and Optimization Manager. For more information about how to set up clusters in WebSphere CE, see:

<http://publib.boulder.ibm.com/wasce/V2.0.0/en/clustering.html>

You can find more HA configuration information in the IBM Whitepaper *IBM ILOG ODM Enterprise V3.3, Building a High Availability ODM Enterprise environment* at:

<http://www.ibm.com/support/docview.wss?uid=swg27017270&aid=1>

3.8 Security model

This section addresses the authentication options that ODM Enterprise supports on a deployed ODM Enterprise application. In a single-user ODM Enterprise application, authentication is not required because the application is configured to be run and solved on a local system with a local Scenario Repository. However, for multi-user ODM Enterprise applications, an authentication system needs to be in place.

3.8.1 Simple authentication system

This default authentication system requires the developer (or ODM Enterprise application administrator) to setup user name (user ID) and password combinations for authorized users during model deployment. The authentication information is stored in the Scenario Repository and validated whenever a business user tries to start the ODM Enterprise application. The user names need to be managed by using the *.odmns file. For more information, see 3.1.12, “Deployment” on page 82.

After you navigate to the Users tab of the ODM Repository Administration window, you can manage users (identified by a Name and Password) for your ODM Enterprise application. The window also provides these optional fields to control user rights:

- ▶ Administration: Users with this privilege can add new users and change their rights. These users are treated like the administrators for the Scenario Repository. The administration window is also accessible through the ODM Studio for users with administration rights.
- ▶ Force Unlock: These users are able to remove scenario locks that were set by them or other users for a particular scenario.
- ▶ Cancel Solve: This access allows a user to cancel solve operations on scenarios that were started by them or by other users.

3.8.2 Custom authentication system

Along with providing a simple authentication system, ODM Enterprise also allows the integration of the ODM Enterprise application with enterprise-wide authentication systems, such as an LDAP directory. Integrating a custom authentication manager includes these basic steps:

1. Create an ODM Enterprise application as usual.
2. Write your custom authentication manager by extending the abstract class `IloAuthenticationManager`.
3. Compile the code into a custom JAR file.
4. Edit the deployment XML file to declare the authentication manager, and add the custom JAR file to the deployment configuration.

You can find a sample that shows one such custom authentication system for the `mulprod` example in the following directory:

```
<ODME_installation_dir>\Developer\examples\Advanced\Authentication
```

For more information, see the online ODM Enterprise InfoCenter. Click **IBM ILOG ODM Enterprise Developer Edition V3.6 → ODM Enterprise User Manual → Advanced usage → Examples, OPL settings in ODM, and custom authentication** at:

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

3.9 Conclusion

This chapter introduced the key ODM Enterprise concepts and expanded on the terminology, in the context of an ODM Enterprise application. It explained a typical ODM Enterprise application architecture, focusing on the various components that are part of the topology. After describing the extensibility API in ODM Enterprise, it touched on the Domain Object Model, which can help deploy custom application code. It also addressed the data architecture, followed by details about the load balancing, scalability, and high availability aspects of an ODM Enterprise application. It concluded by addressing the authentication systems that are implementable in ODM Enterprise.



Transforming business requirements into an ODM Enterprise application

This chapter provides a high-level overview of the ODM Enterprise development process and ODM Enterprise integrated development environment (IDE). IDE is the development environment that is used to build ODM Enterprise applications. The ContainerYard example from 2.6, “Advantages of ODM Enterprise solutions” on page 52 is used to illustrate each step of the development process.

This chapter includes the following sections:

- ▶ ODM Enterprise development process
- ▶ Building the ContainerYard application
- ▶ Conclusion

4.1 ODM Enterprise development process

Developing an ODM Enterprise application is a collaborative process. It consists of several distinct tasks that are performed by one or several members of the team. This section addresses the steps that are involved in the process. It also introduces the ODM Enterprise IDE, the development environment that is used to create, configure, and deploy applications.

4.1.1 ODM Enterprise development flow

The ODM Enterprise development process is data-centric, meaning that it revolves around the business data. The business analyst and business users first determine the data that are required to solve their business problem. The application is then built to provide the best view of this data. After the data structure and views are in place, optimization capabilities can be added. The optimization model can also be developed at the same time as the rest of the application. However, a benefit of data-centric development is that an application prototype can be created without the optimization model in place (Figure 4-1).

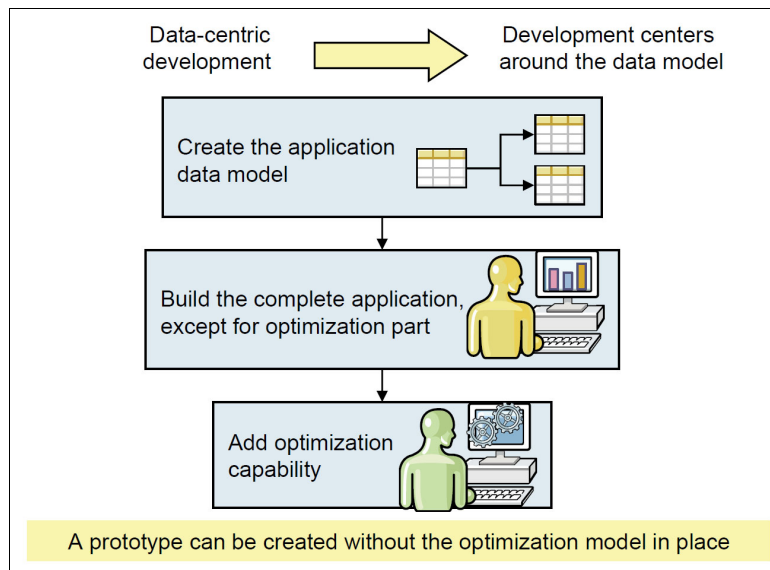


Figure 4-1 High-level view of the ODM Enterprise development process

Developing an optimization application is an iterative process, requiring regular interaction among the business, IT, and OR teams (Figure 4-2). Feedback from the business users is essential at every step of the process. Based on business requirements, the IT team develops data structures and views, and the OR team implements an optimization model. Business users review prototypes and provide feedback to ensure that all the requirements are met. The OR and IT teams can work in parallel on separate tasks. However, they must collaborate to link the optimization model to the application data model, and to configure optimization-specific parts of the application.

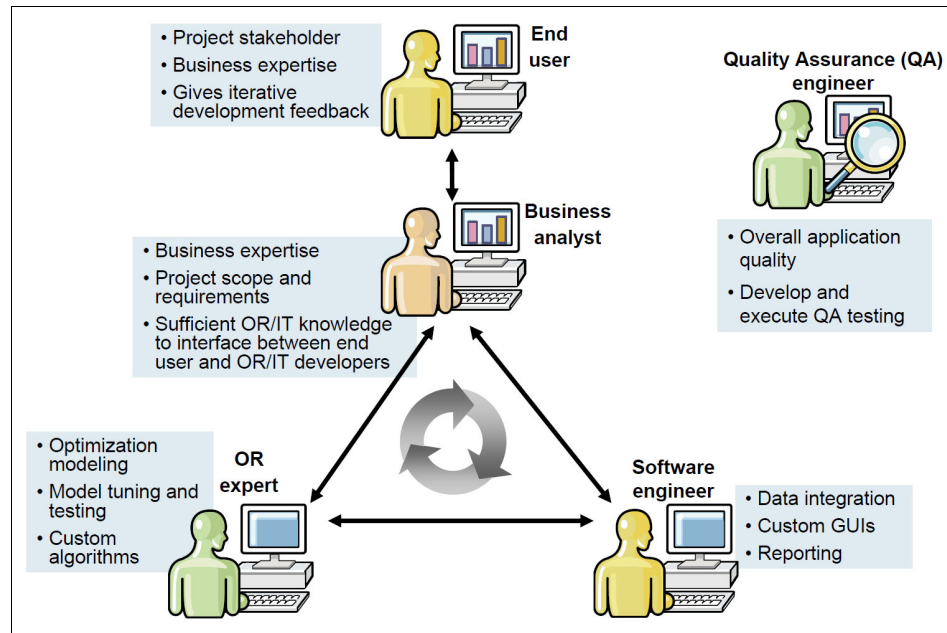


Figure 4-2 Roles that are involved in the ODM Enterprise development process

4.1.2 ODM Enterprise IDE

The ODM Enterprise IDE is a complete development environment for ODM Enterprise applications. It comprises the Eclipse IDE and the ODM Enterprise Eclipse plug-in, as well as plug-ins for OPL and JViews. The plug-in nature of ODM Enterprise allows you to take advantage of the familiar Eclipse IDE. You can work with your OPL and ODM-based optimization projects in a single development environment.

The Eclipse IDE allows for multiple perspectives, or preset views and menu layouts that are related to the specific development tasks for a specific type of

project. For example, Figure 4-3 shows the OPL Perspective used to develop and test an OPL model.

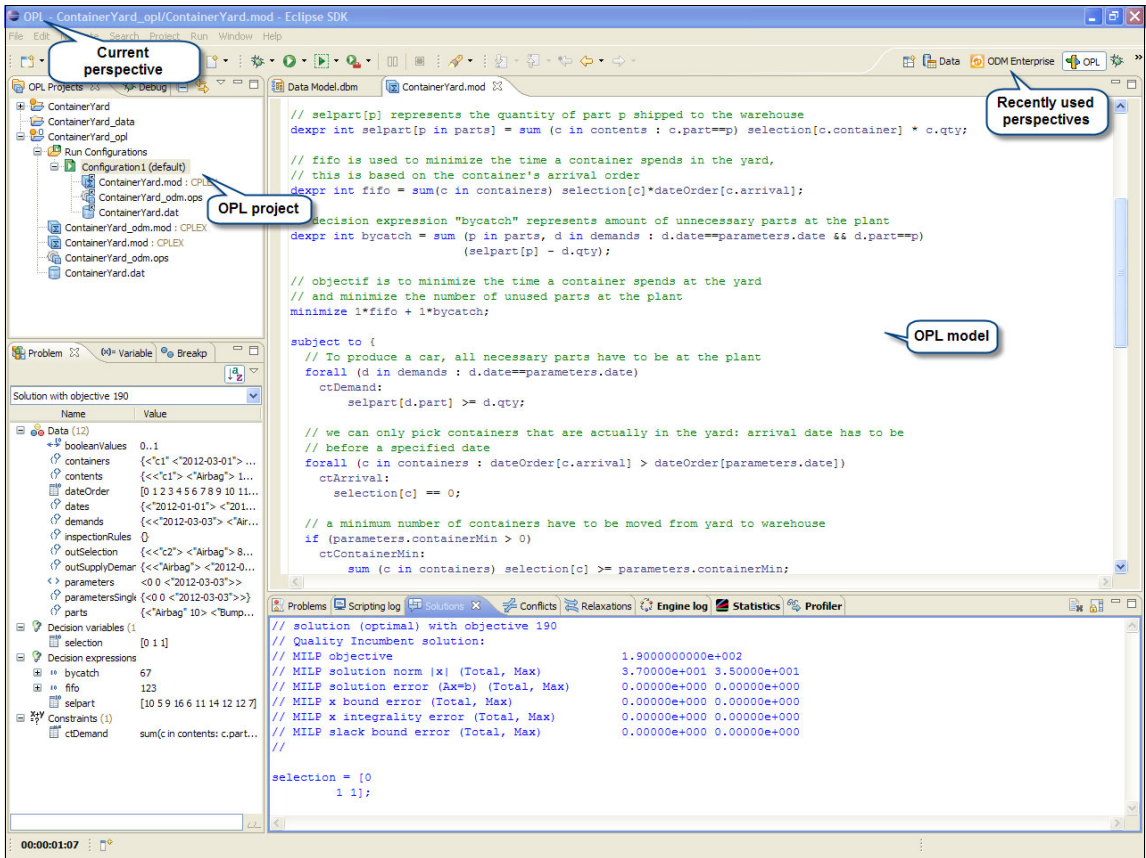


Figure 4-3 ODM Enterprise IDE: OPL perspective

Other perspectives that are most commonly used to develop an ODM Enterprise application are highlighted in Figure 4-4. Examples of working with these perspectives are shown in the following sections while developing the ContainerYard application.

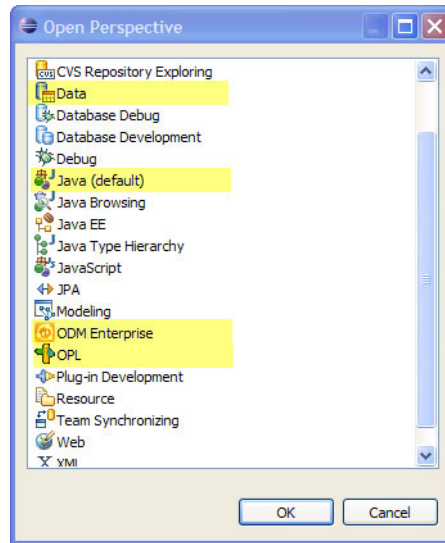


Figure 4-4 ODM Enterprise IDE: perspectives

4.2 Building the ContainerYard application

This section uses as an example the ContainerYard problem described in 2.2, “The ContainerYard: An ODM Enterprise application example” on page 26. It uses this example to show how to build a simple but complete ODM Enterprise application. This complete application, along with all the necessary source projects, can be downloaded as described in Appendix C, “Additional material” on page 329.

4.2.1 Analyzing requirements and defining application scope

Before you develop an ODM Enterprise application, or any optimization-based application for that matter, carefully analyze business requirements and determine the project scope. During this phase, the business analyst works closely with the business user. After the requirements and scope are refined, the business analyst passes the information on to the OR expert and IT specialist. The OR expert and IT specialist then develop the application according to the specifications. The business analyst must have sufficient knowledge of both OR

and IT to be able to guide the business user towards realistic scope and requirements.

IBM ILOG uses a well-defined methodology, called a *Discovery Workshop*, to help organizations evaluate the benefits of implementing an optimization-based solution. This methodology has been refined and proven over the years as a result of numerous successful workshops. The Discovery Workshop is a two day solution development effort that is led by a team of business optimization experts. The first and key part of the workshop is a discussion that centers around evaluating the business value of an optimization solution. The goal is to achieve the organization's business objectives with the least risk, and to estimate the total cost of ownership (TCO) of the system before it is implemented.

Good understanding of the business problem and requirements is key to designing a system that fits the business. This process is preferable to forcing the business to fit the system. Give particular attention to the use cases and examining how users will interact with the application, what type of user interface is needed, and the analytical capabilities wanted. After the business value and requirements are clear, discuss the architecture and implementation plans with the team. Include how the system can integrate with existing technologies such as SAP or Siebel, and any other new tools that must be included in the solution.

Requirements, investigated in a Discovery Workshop and refined in subsequent assessments, fall into four categories as depicted in Figure 4-5.

Business	Optimization	Architecture	Implementation
• Business case overview • Current and desired situation • Success factors and expected benefits • Timeline • Stakeholders	• Functional business problem • Targets • Limits • Choices • Data requirements	• Optimization approach • Business analytics and user interface • Decision process and roles • Data model • Data sources and integration • IT architecture and deployment	• Software design • IT and optimization platform • Project approach • Resources

Figure 4-5 Requirements definition

Business requirements include a description of the business problem and the business case for solving it. They also address the current situation and contrast it with the wanted situation. They define what constitutes success for the proposed solution and the benefits expected. Finally, they establish the timeline, and the key stakeholders and the roles they play.

Optimization requirements refine the functional business problem to be solved by identifying the specific targets, limits, and choices. Optimization experts need to assist the business users and analysts in this task. Many aspects of optimization are not apparent to people unfamiliar with mathematical modeling, and some issues that seem challenging to the business can be easily addressed in the optimization. For instance, business users might not mention obvious physical constraints, such as conservation of flow in a product distribution network. These constraints must be included for the optimization model to make sense. Conversely, business experts often want to include pre-existing business rules that are restrictive and can inhibit optimization from finding the best solution.

Optimization requirements also include aspects of the data that are required to specify the mathematical model. Data limitations represent a primary restriction on the model. For example, if sales data is collected at a national level only, it cannot be used to model sales at a district level. In this case, if district-level detail is considered crucial, the company might decide to collect it. Or the application can use statistical models to estimate the district-level data.

Architecture requirements address the structure of the system that will solve the business problem. It considers the type of optimization solvers to use (for example, linear programming, mixed-integer programming, or constraint programming). It also addresses what the business user interface looks like and what analytics capabilities the users need. It defines the decision process that the solution implements, and the roles of the users in that process. It specifies in more detail the schema of the application data model, the data sources, and the integration of data into the application. These data include any selection, pre-processing, and post-processing needed. Finally, it considers deployment requirements within the enterprise information technology architecture and standards.

Implementation requirements concern the project plan. Details include software engineering and design standards, the information technology and optimization platforms to be used, the project management approach, and the resources required.

The following sections address how business requirements described in 2.2.2, “Planning process” on page 27 map to various parts of an ODM Enterprise application. To provide a high-level overview of the development process, the description is kept concise. It focuses on the final results of each step of the application development process. For more information about developing an

ODM Enterprise application, see the ODM Enterprise InfoCenter documentation at the following link. Click **IBM ILOG ODM Enterprise Developer Edition V3.6 → The ODM Enterprise Development Process → Developing an application using IBM ILOG ODM Enterprise**.

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

4.2.2 Defining the data model

The data-centric nature of ODM Enterprise application development implies that the *Application Data Model* (ADM) definition is at the heart of the process:

- ▶ The ODM Enterprise application is built on the data structures that are defined in the ADM.
- ▶ The ADM contains the definition of the structure of a scenario, which is a set of tables that contain columns, primary keys, and foreign keys.
- ▶ ODM Enterprise uses the ADM to create the ODM Scenario Repository, which stores scenarios.
- ▶ ODM Studio views are built on top of the ADM tables.
- ▶ The OPL model data structure is mapped to the relevant parts of the ADM, and the OPL data model is generated from the ADM.

Before creating the ADM, the developer must have a good understanding of the requirements, both on the user interface and the optimization sides. The developer must ensure that any data structures that will be used by ODM Enterprise views are included as tables in the ADM. Likewise, the developer must collaborate with the OR expert to ensure that the ADM includes all data structures to be used by the optimization model.

The usual process is to complete the ADM first before you build the views and the optimization model. However, requirements often change during the development process. When business users begin to review application prototypes, missing requirements can be uncovered and existing requirements might need to be revised. If these changes require modifications in the data structures, the ADM might need to be updated. For example, the business users might realize that they need to display data that is not present in the ADM. The developer might need to add an extra table that can then be used to create the view.

Another possibility is that the business users might request the ability to dynamically add a set of constraints through the user interface. In this case, the developer needs to introduce a new rule template in the rules view. The developer must also add a corresponding table in the ADM to represent the data

that is generated by user created rules. For more information about how rules are implemented, see “Creating the application data model (ADM)” on page 120.

In summary, finalize the ADM early in the development process, but allow for changes in requirements. The ODM Enterprise application development process can easily accommodate such changes because of its iterative nature.

Creating the necessary projects in the IDE

When the software engineer is ready to implement the data model, start by creating an empty ODM Enterprise project. Now you can create a Data project (with associated ADM) or link the ODM Enterprise project to an existing Data project. You can also create an empty OPL project now, or add it later.

Figure 4-6 shows the list of projects for the ContainerYard application.

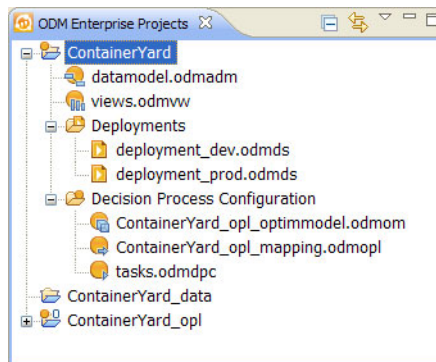


Figure 4-6 ODM Enterprise IDE: Create new projects

Figure 4-7 depicts how the ODM Enterprise project is automatically linked to the corresponding Data project through the .odmadm file.

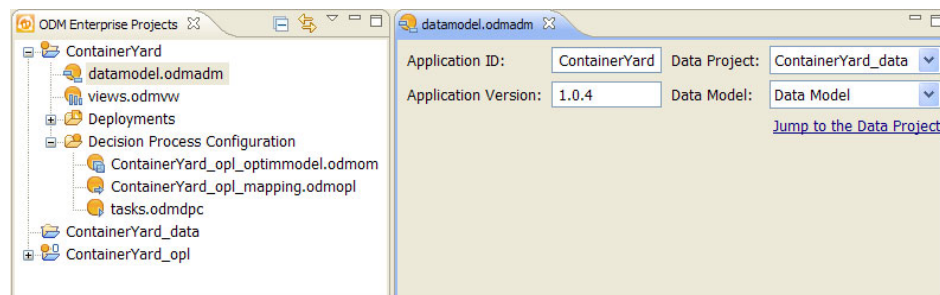


Figure 4-7 ODM Enterprise project link to Data project

Creating the application data model (ADM)

Next, the software engineer, possibly together with the OR expert, creates the ADM. The ADM is created and edited from the Data perspective in the ODM Enterprise IDE. It can be created from scratch by using the built-in graphical editor. Alternatively, schemas can be imported directly from an existing database and modified in the editor, as needed.

The ContainerYard example needs only a few tables to hold input and output data.

Input tables

The following input tables are used to display data in input data views:

- ▶ CONTAINER
List of containers: Container ID, arrival date, customs info
- ▶ PART
List of parts: Part name, shelf life
- ▶ CONTENT
Contents by container: Part, quantity
- ▶ DEMAND
Demand by date and part: Date, part, quantity
- ▶ PARAMETER
Additional parameters that are used by the optimization:
 - Minimum number of containers selected
 - Maximum number of containers selected
 - Planning date
- ▶ INSPECTION_RULE
Holds the data that are created by business users as part of rules:
 - Container
 - Inspection length

For example, a user creates the rule that is shown in Figure 4-8. The data element <c1,3> is stored as a row in the scenario table that corresponds to INSPECTION_RULE. These data are used to generate the corresponding inspection constraints in the model. For more information about this process, see “Limits” on page 148.

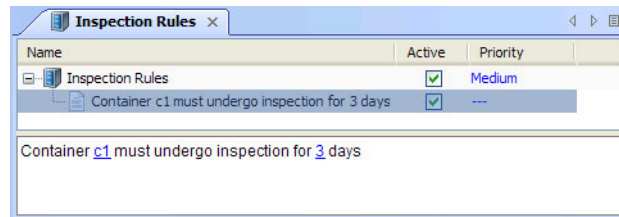


Figure 4-8 ContainerYard application: inspection rules

Output tables

The following output tables are used to display data in the solution views:

- ▶ OUT_SELECTION
 - Selected containers: Container, part, part quantity
- ▶ OUT_SUPPLY_DEMAND
 - Supply and demand information for each day and part:
 - Part
 - Date
 - Demand: Quantity of articles requested
 - Supply: Quantity of articles provided
 - Excess inventory: Value by which supply exceeds demand
 - Shortage: Value by which demand exceeds supply

Figure 4-9 shows what the data diagram looks like after the ADM is created.

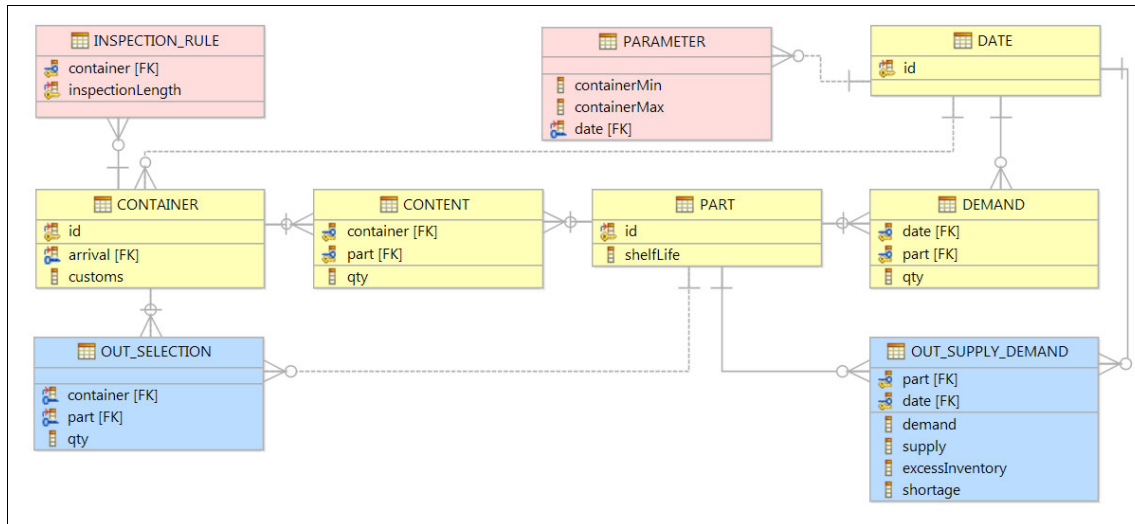


Figure 4-9 ADM: Data Diagram

Enhancement: With ODM Enterprise fix pack 3.6.0.1, date types are directly supported in the ADM. You no longer need a separate DATE table, and date values can be defined by using one of the following three date types:

► **DATE**

Representing a date that consists of the year, month, and the date in the month.

► **TIME**

Representing a time in a day. It consists of hour, minutes, and seconds.

► **TIMESTAMP**

Representing an absolute time, grouping the capabilities of both the DATE and TIME values and adding a fractional part of the seconds.

By defining a reference time and an increment in the PARAMETER table, date values can automatically be converted to and from OPL integer values.

4.2.3 Creating a prototype without optimization

This first prototype focuses on the user interface and implement data integration. When this has been achieved, obtain feedback from the business users.

Creating user interface

After the ADM is in place, the software engineer develops the user interface, which is based on the requirements that are provided by the business analyst. The user interface can be built entirely from standard ODM Enterprise views, with a custom implementation in Java, or a combination of the two. For the ContainerYard application, standard ODM Enterprise parameter views, simple and pivot tables, bar and pie charts, and a multi-scenario comparison view are used. This example shows how easy it is to configure various standard views and create a rich user interface in a matter of minutes.

Views are configured by using the .odmvw editor in the ODM Enterprise perspective (Figure 4-10).

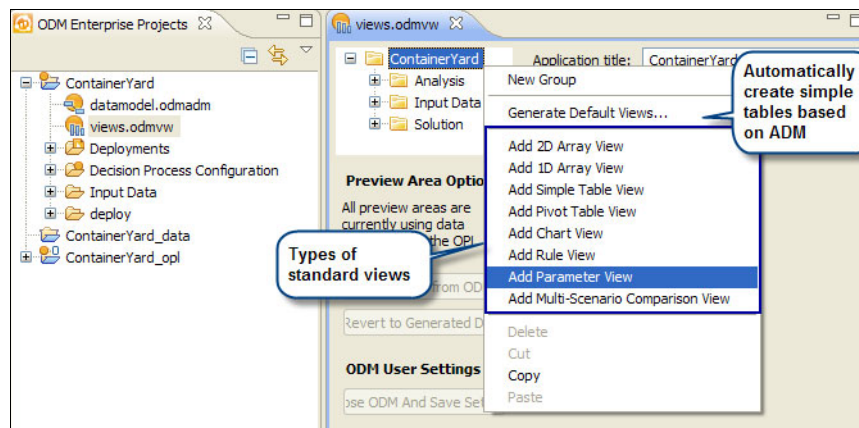


Figure 4-10 Views editor: adding a standard view

Input views

First, configure the input views:

► Parameter views

A parameter view displays data that is stored in a table that contains only a single row. In a parameter view such data is displayed by using the columns *Parameter Name* and *Parameter Value*.

– Parameters

Displays optimization parameters: Minimum number of containers, maximum number of containers, and planning date (Figure 4-11 on page 124).

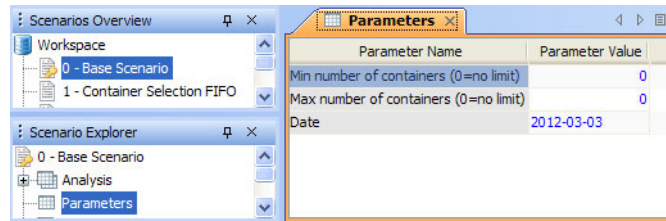


Figure 4-11 Parameter view: Parameters

The corresponding Parameters view configuration is shown in Figure 4-12.

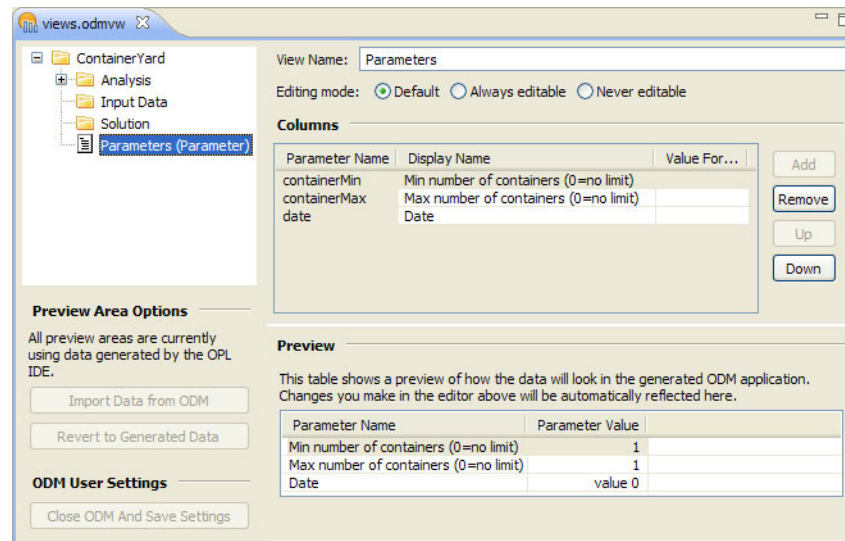


Figure 4-12 View configuration: Parameters

A Parameter view can only be created based on an ADM table that is explicitly identified as a “Single row table,” as shown in Figure 4-13.

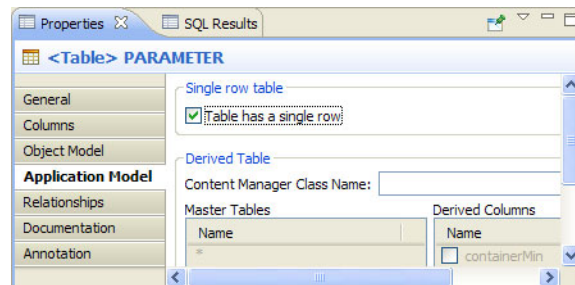


Figure 4-13 ADM: Parameter table properties

- Simple Tables
 - Containers: Displays container data (ID, arrival date, customs info)
 - Contents: Displays container contents (container, part, quantity)
 - Demand: Displays demand by part and date (part, date, quantity)
 - Parts: Displays a list of parts (ID, shelf life)

To configure a simple table, select the ADM table that corresponds to the data to be displayed (Figure 4-14).

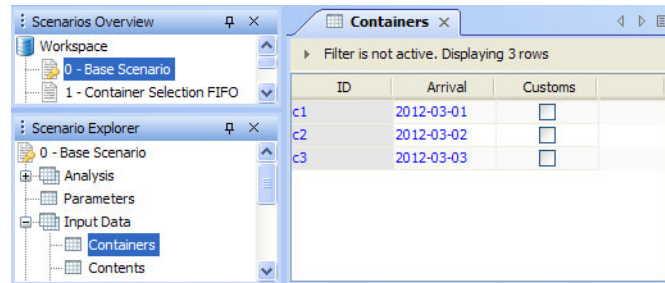


Figure 4-14 Simple table view: Containers

Select the columns and configure the ODM display name as shown in Figure 4-15.

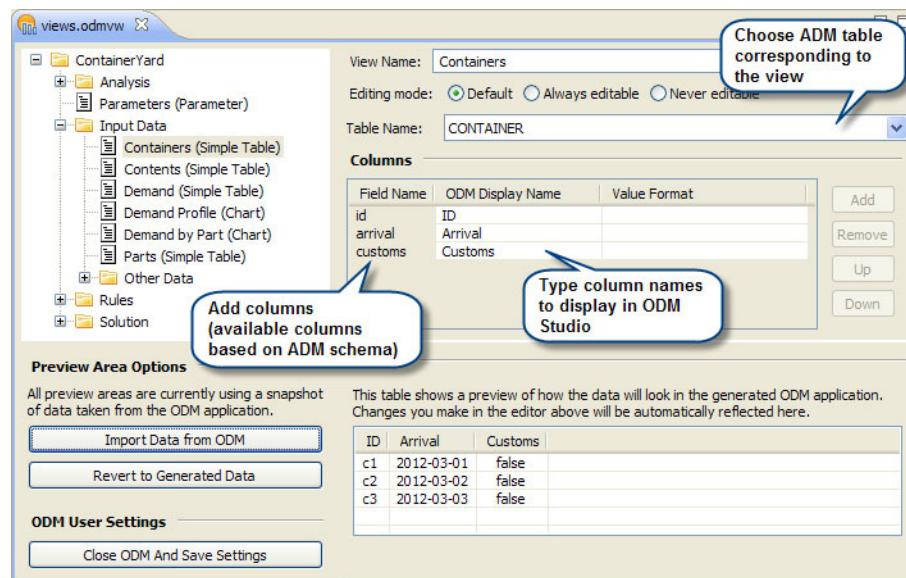


Figure 4-15 View configuration: Containers (simple table)

► Charts

Configure the chart views by selecting the data to be displayed (table, index, columns) and then configuring the chart itself (rendering, labels, and 3D configurations, if any). The example has two input data charts:

– Demand Profile

A bar chart that shows daily demand for each part that is selected by the user (Figure 4-16).

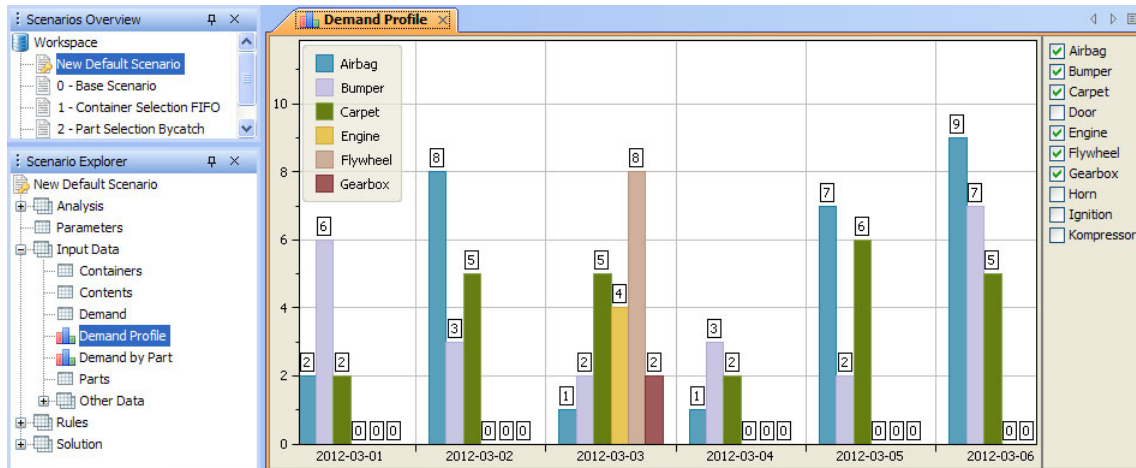


Figure 4-16 Bar chart view: Demand Profile

Figure 4-17 shows how the Demand Profile bar chart is configured in the views editor in the IDE.

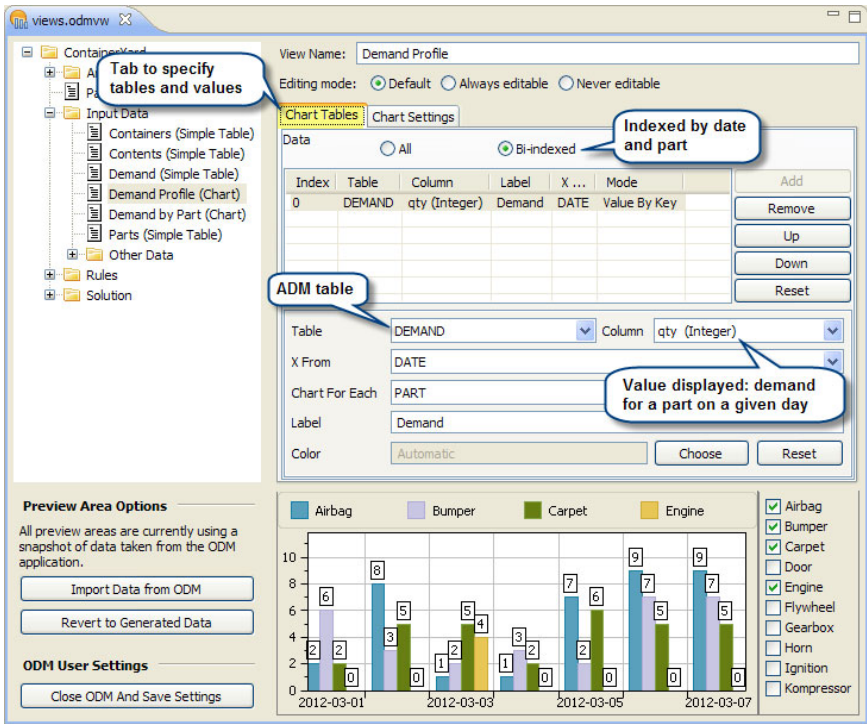


Figure 4-17 View configuration: Demand Profile (bar chart)

Figure 4-18 shows how the Demand Profile bar chart settings are configured.

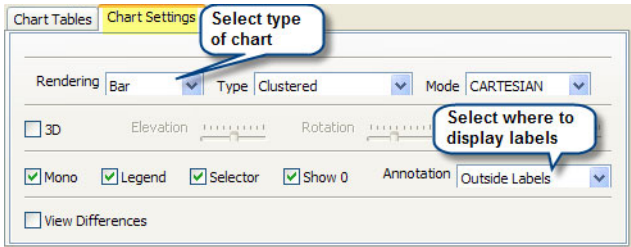


Figure 4-18 View configuration: Demand Profile (bar chart settings)

– Demand by Part

A pie chart that shows total demand by part summed over the entire horizon (Figure 4-19).

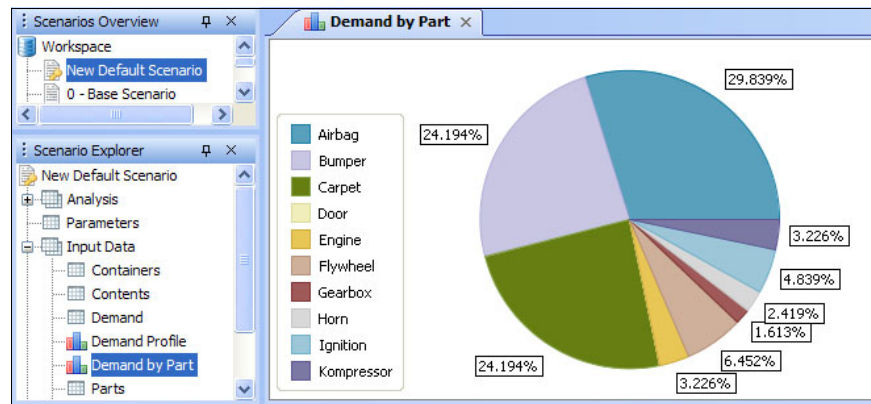


Figure 4-19 Pie chart view: Demand by Part

Figure 4-20 shows how the Demand by Part pie chart is configured in the views editor in the IDE.

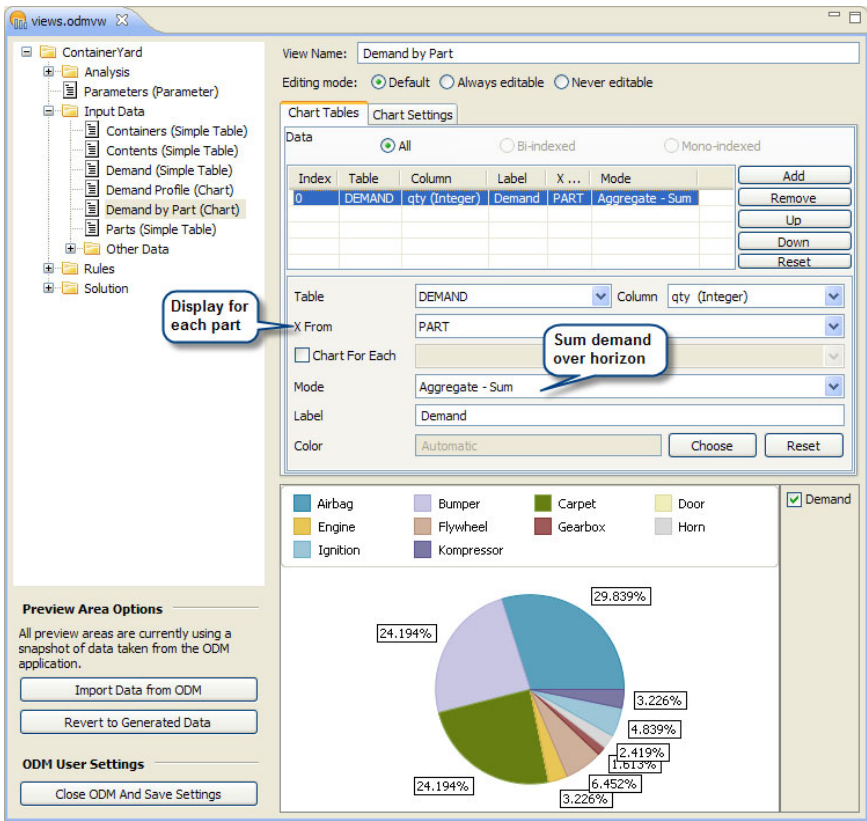


Figure 4-20 View configuration: Demand by Part (pie chart)

Figure 4-21 shows how the Demand by Part pie chart settings are configured.

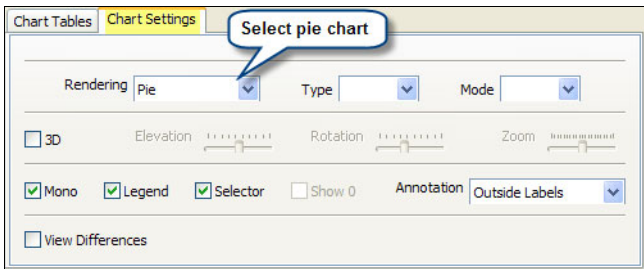


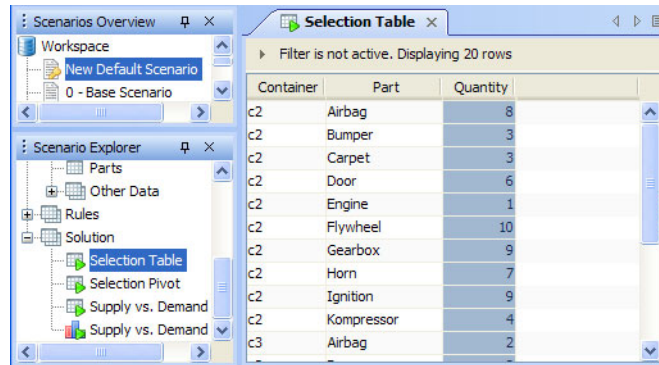
Figure 4-21 View configuration: Demand by Part (pie chart settings)

Output views

Next, configure the output views:

► Simple Tables

A Selection Table (Figure 4-22) displays the selected containers and their contents (part, quantity). This view is configured with the ADM table OUT_SELECTION by using the same mechanism as shown for the Containers simple table input data view.



Filter is not active. Displaying 20 rows

Container	Part	Quantity
c2	Airbag	8
c2	Bumper	3
c2	Carpet	3
c2	Door	6
c2	Engine	1
c2	Flywheel	10
c2	Gearbox	9
c2	Horn	7
c2	Ignition	9
c2	Kompressor	4
c3	Airbag	2

Figure 4-22 Simple table view: Selection Table

► Pivot Tables

Pivot tables are configured by first selecting the data (tables and columns) to display. You then specify the default look of the table by selecting the row, column, data, and filter fields, and their order.

– Selection Pivot

Displays selected containers and their contents (part, quantity) as a pivot table (Figure 4-23).

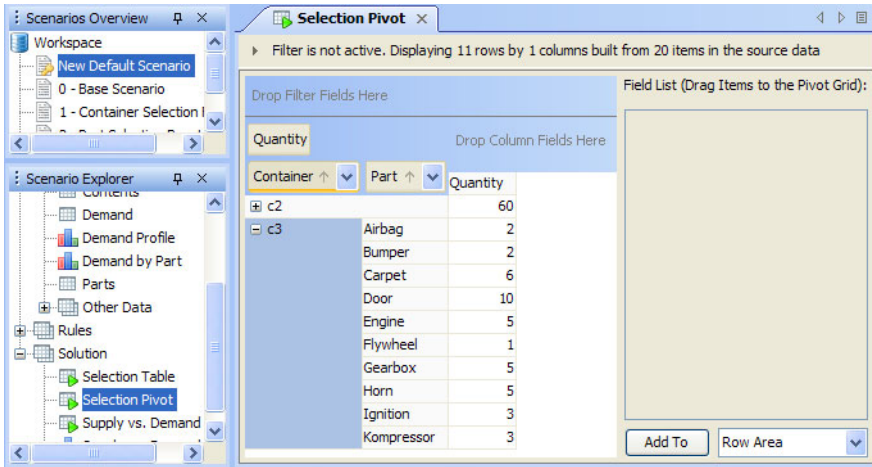


Figure 4-23 Pivot table view: Selection Pivot

– Supply versus Demand

Displays demand, supply, excess inventory, and shortage for each part on the planning day (Figure 4-24).

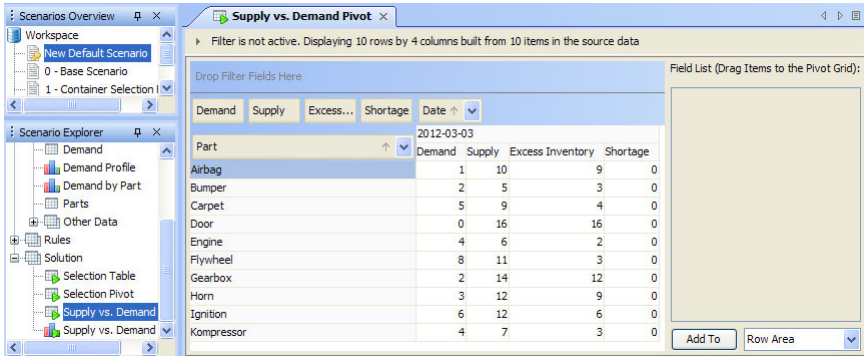


Figure 4-24 Pivot table view: Supply versus Demand

Figure 4-25 shows how to configure the pivot table fields.

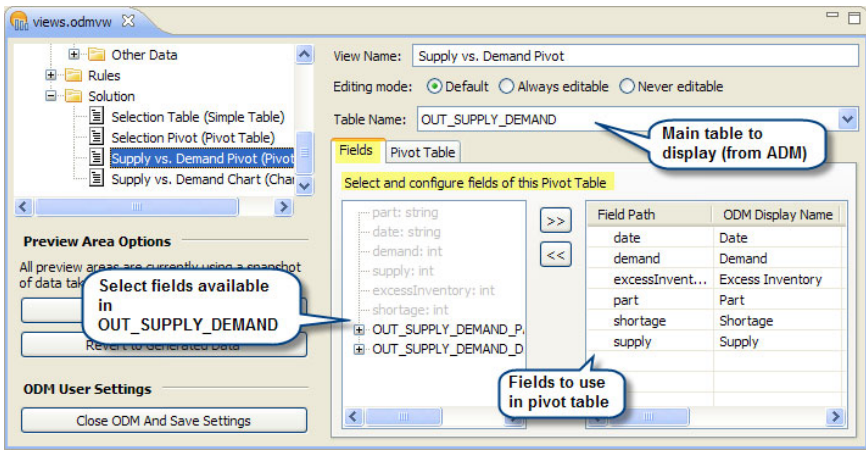


Figure 4-25 View configuration: Supply versus Demand Pivot (pivot table fields)

Figure 4-26 shows how to configure the pivot table in the views editor.

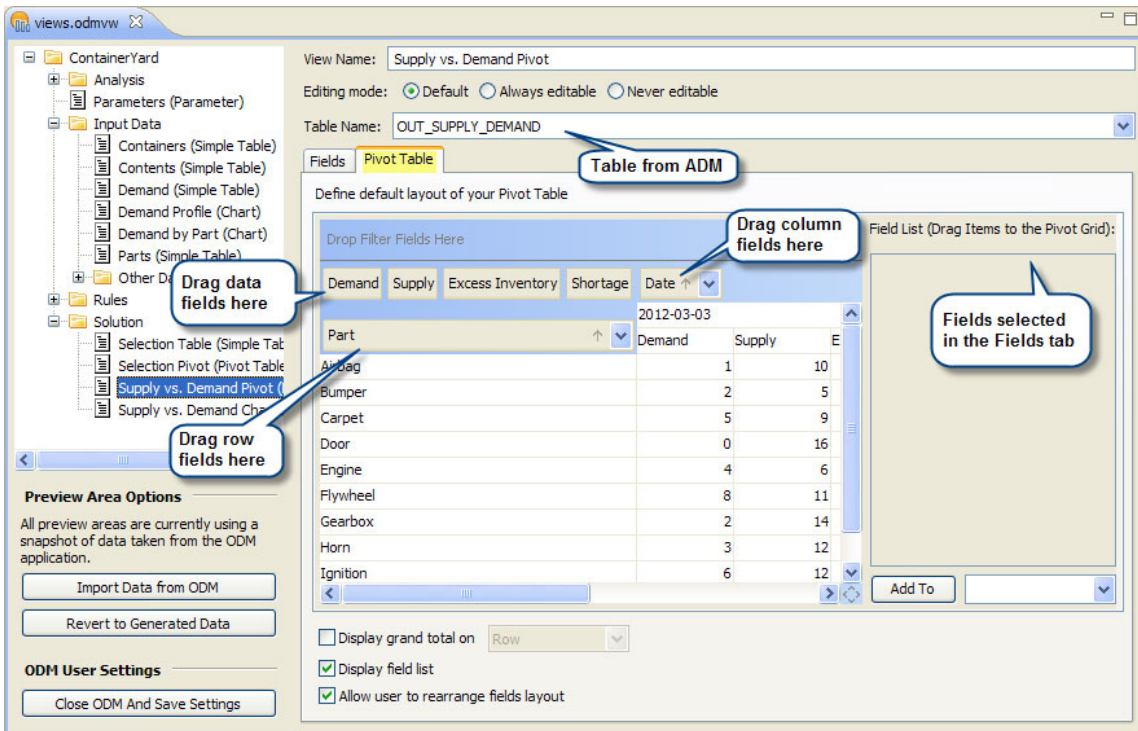


Figure 4-26 View configuration: Supply versus Demand Pivot (pivot table)

- Charts
 - Supply versus Demand Chart

A bar chart that plots demand, supply, excess inventory, and shortage for each part on the specified planning date (Figure 4-27).

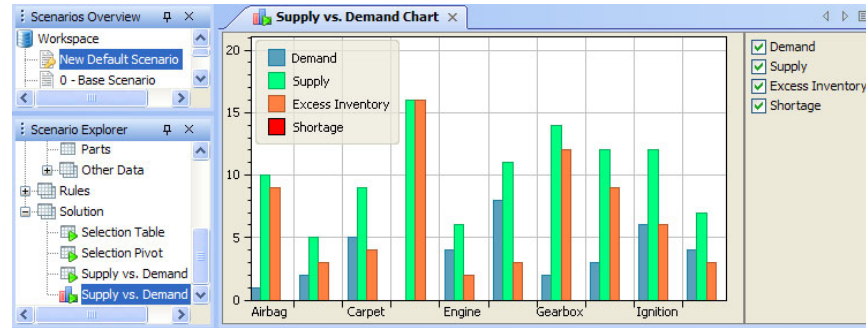


Figure 4-27 Bar chart view: Supply versus Demand Chart

Figure 4-28 shows how to configure this chart in the views editor.

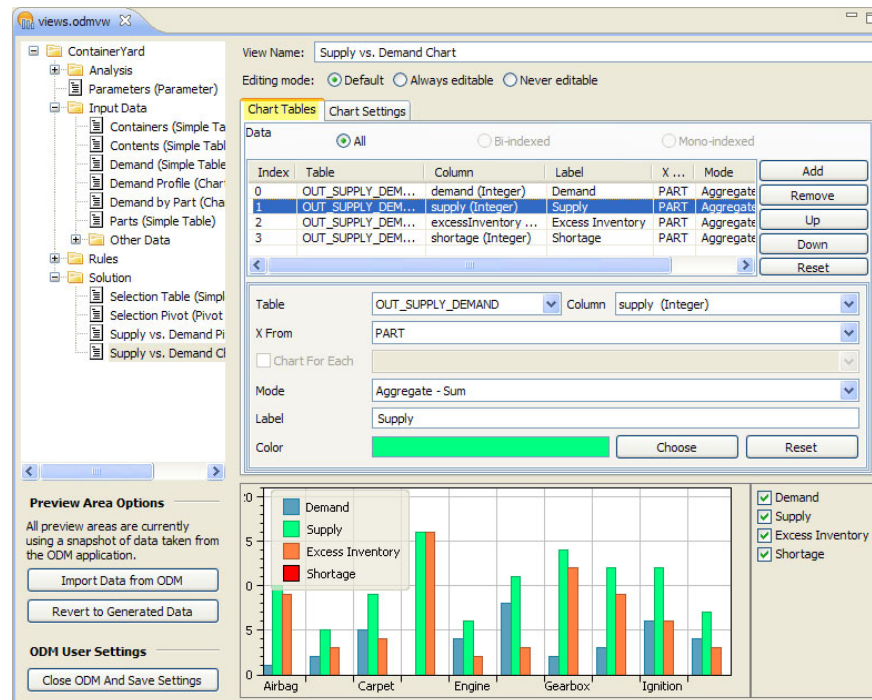


Figure 4-28 View configuration: Supply versus Demand (bar chart)

Figure 4-29 shows how to configure the bar chart settings. This process is similar to the one for input views *Demand Profile* and *Demand by Part* as shown in “Input views” on page 123.

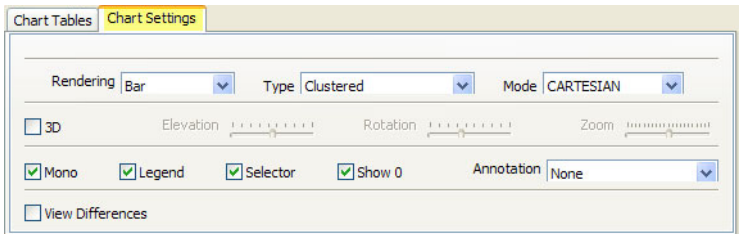


Figure 4-29 View configuration: Supply versus Demand (bar chart settings)

► Multi-scenario comparison views

Like other types of views, multi-scenario comparison views are added by using the views editor in the IDE. However, unlike other views, multi-scenario comparison views must be configured directly in ODM Studio.

– KPI comparison view

This view allows you to compare multiple scenarios based on input parameters (min or max number of containers) and various solution status values (Figure 4-30). After you configure optimization goals, goals such as container selection FIFO or part selection bycatch are also available as columns in the KPI Comparison view. For more information, “Adding and configuring goals” on page 155.

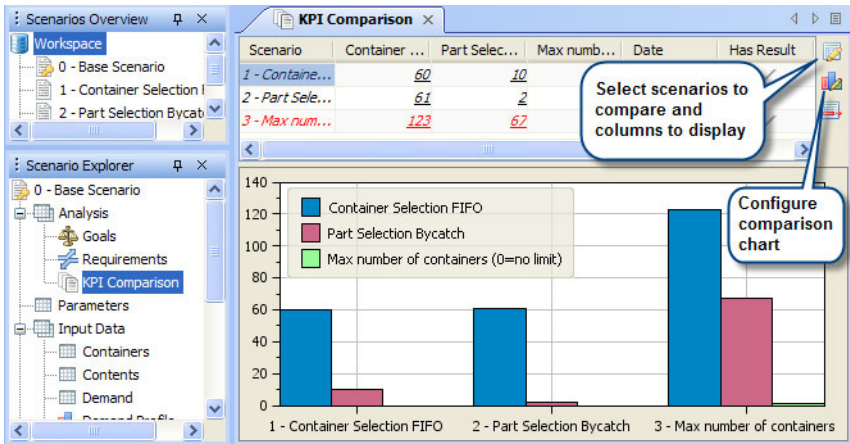


Figure 4-30 Multi-scenario comparison view: KPI Comparison

Configure the multi-scenario comparison view as shown in Figure 4-31.

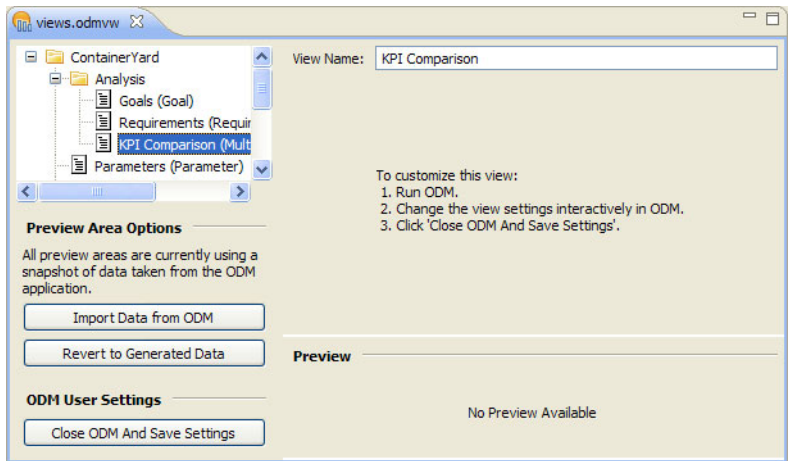


Figure 4-31 View configuration: KPI Comparison (multi-scenario comparison)

After all views are configured as shown in Figure 4-32, start the application from the IDE.

Clarification: Because you have not yet set up the data sources, no input data can be displayed. Therefore, all views are empty.

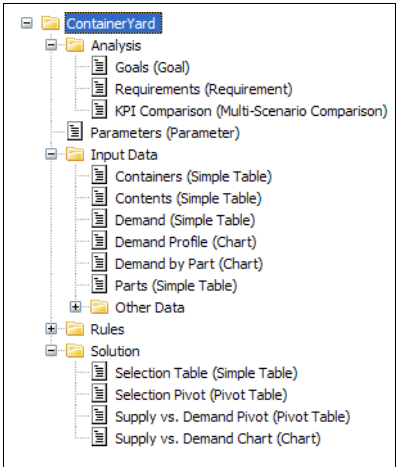


Figure 4-32 Views editor: Final views configuration

Make sure that the views are displayed as expected (Figure 4-33).

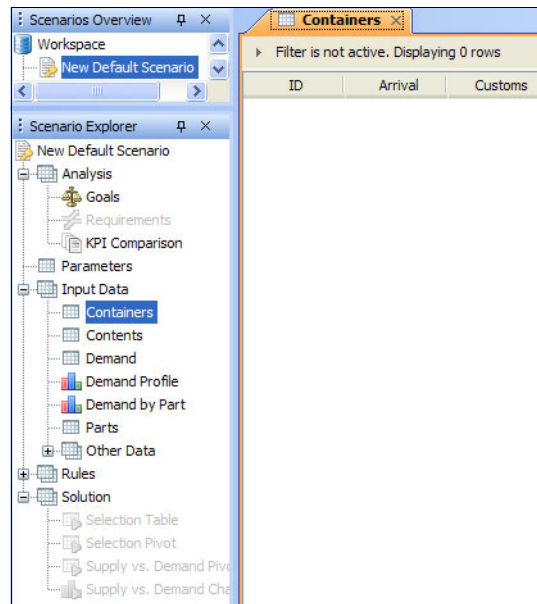


Figure 4-33 ContainerYard application prototype without optimization and data

For more information about goal views, rule views, and requirement views, see “Configuring goals, requirements, and rules” on page 155. These views can be defined and configured only after the optimization model is in place.

Implementing data integration

With the GUI built, the next step is to populate the input views with data that represent a business scenario to complete the initial prototype. This can be actual data from your database, data that is generated by custom code, or data from .csv files.

Data sources are configured on the Datasources tab of the development_dev.odmids or development_prod.odmids editors, depending on whether the data source is created for development or production.

In this example, create two .csv based data sources. Figure 4-34 shows the Datasource tab of the deployment settings editor after the data sources are configured.

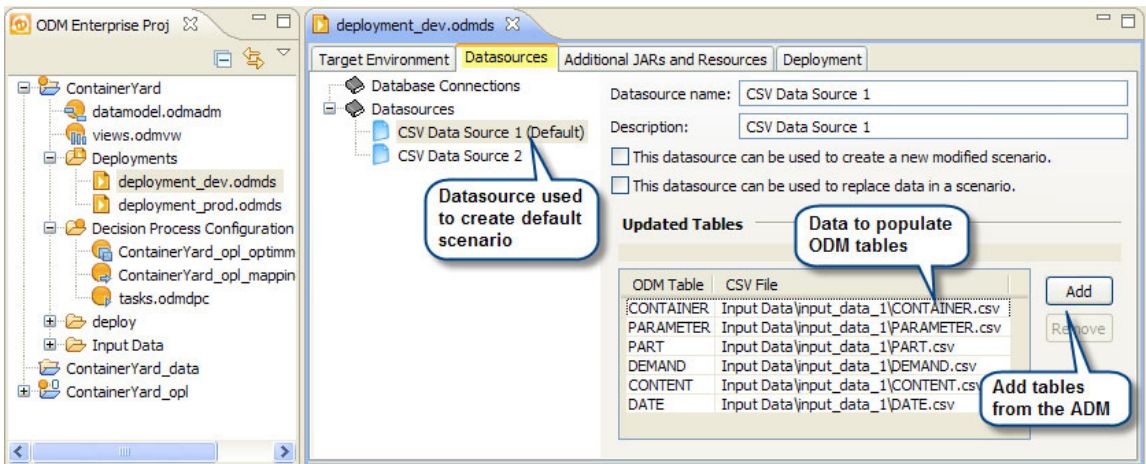


Figure 4-34 Deployment settings: Data sources

Obtaining feedback from a business user

You have now finished the first prototype of the application that can be shown to the business user. Based on the user's feedback, development steps can be repeated as necessary before you continue on to the optimization part of the process.

Figure 4-35 shows the application prototype that results from the previous steps.

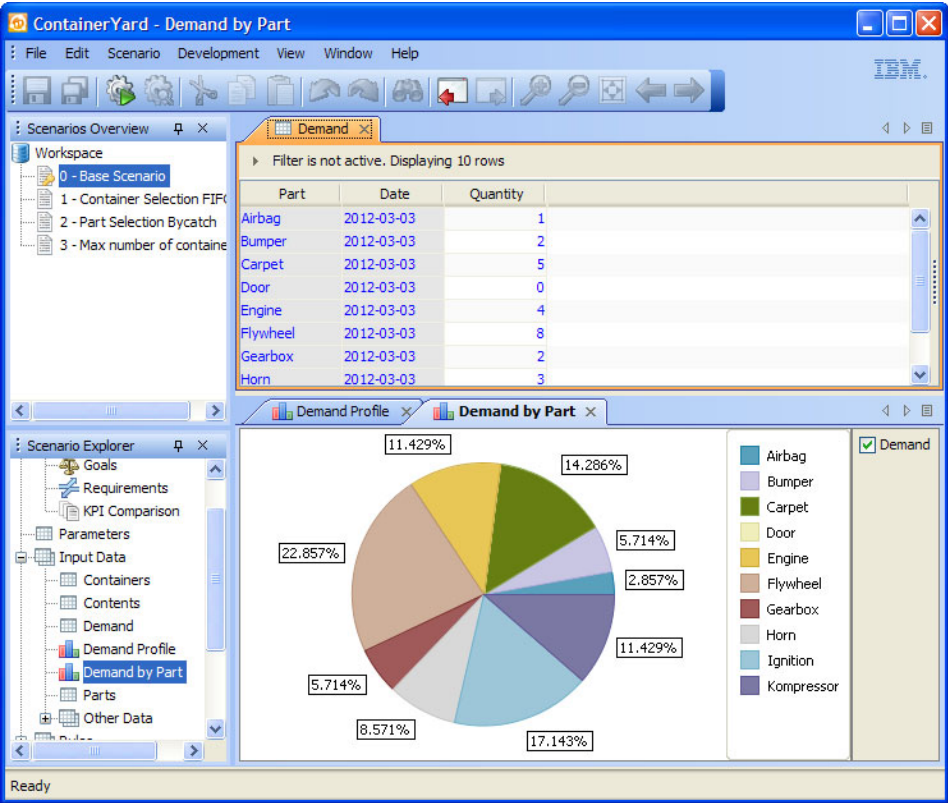


Figure 4-35 Application prototype without optimization

4.2.4 Creating a prototype with optimization

Only the part of the ADM required for the optimization model is mapped to OPL elements. Data in a scenario often includes human-readable information, such as names, addresses, and descriptions. These data might not be relevant to the optimization model, but are useful to show to the business users in the final application. Being able to configure precisely what tables and fields are mapped to OPL leads to a better memory footprint for the optimization model.

The OR expert and software engineer work together during this phase to ensure that the Application Data Model provides all the data that is required for the optimization model. The software engineer adds any missing data items. Goals and requirements for the ODM Enterprise application can be configured as soon as the optimization model is complete.

The OPL project is edited from the OPL perspective within the ODM Enterprise IDE.

A smidgen of OPL (part 1) - Declarative syntax:

OPL has a syntax that mirrors the way optimization models are expressed mathematically, but a few items need clarification for unfamiliar readers. OPL differs from more widely used programming languages because OPL has a *declarative* syntax. An OPL model consists of statements *that must hold true* at an optimal solution. *Procedural* languages, such as C++ or Java, specify the algorithms that are used to find the result. In an ODM Enterprise application, the solver engines (CPLEX Optimizer or CPLEX CP Optimizer) provide the algorithms.

OPL's declarative syntax can mislead new users. For instance, generally, an OPL model is not sensitive to the order of statements. The exception is that data structure definitions precede optimization specifications, such as decision variables, decision expressions, and constraints, that use them).

OPL does include a scripting language, OPLScript, that allows you to include procedural steps in an OPL model and flow controls to specify a sequence of solving steps. See Example 4-10 on page 151 and Example 4-11 on page 152.

Creating an OPL project linked to the ODM Enterprise project

An OPL project is created and linked to the ODM Enterprise project. After this process is complete, all projects are visible in the Scenario Explorer in the ODM Enterprise perspective.

The OPL project now contains a Run Configurations folder, two empty model files (_odm.mod and .mod), and the default settings file (_odm.ops). The Decision Process Configuration folder in the ODM Enterprise project now contains the .odmom and .odmopl files, as shown in Figure 4-36.

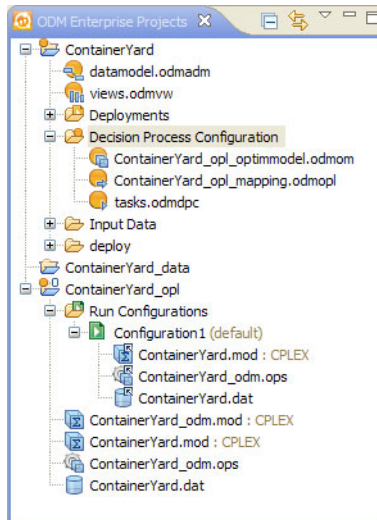


Figure 4-36 ODM Enterprise IDE: All projects are created

Creating the ODM-OPL mapping

The next step is to select a subset of the ADM to be used by the optimization model by using the ODM-OPL mapping editor, as shown in Figure 4-37 on page 141.

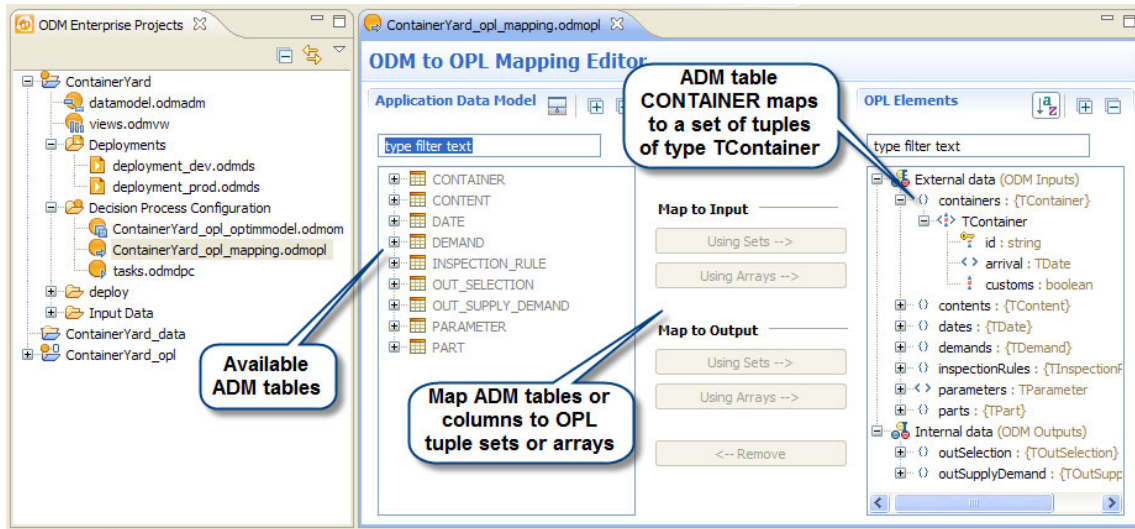


Figure 4-37 ODM to OPL Mapping Editor: Map ADM to OPL elements

After the mapping is completed and the .odmopl file is saved, the _odm.mod file is automatically updated with the newly generated OPL data structures as shown in Figure 4-38. Example 4-1 on page 142 shows the full listing of the OPL data structures that are created by the mapping editor.

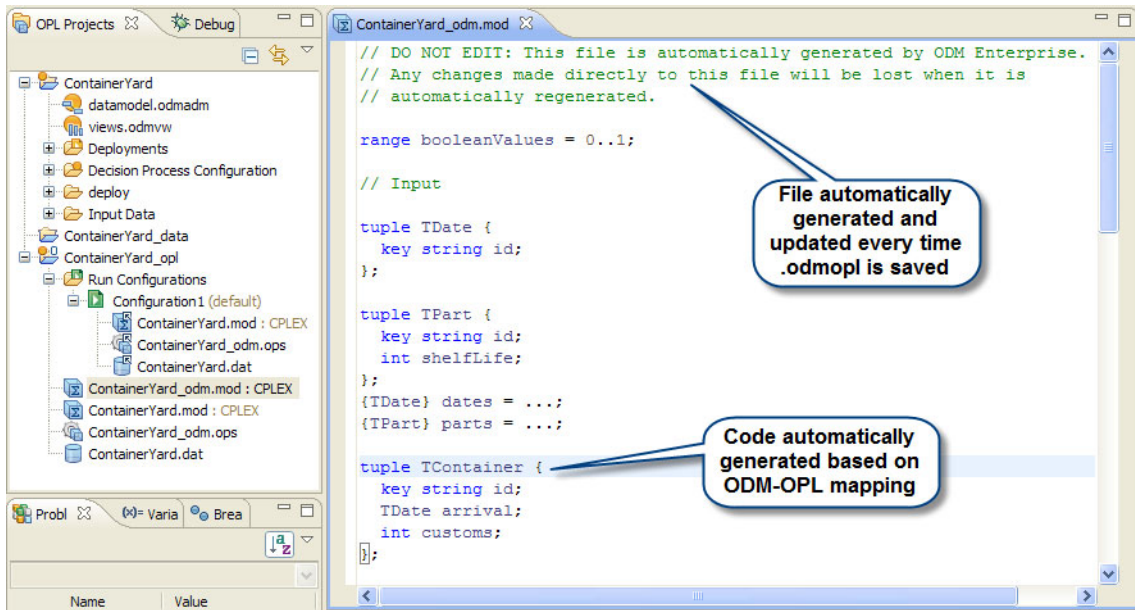


Figure 4-38 Automatically generated OPL data structures (_odm.mod file)

The `_odm.mod` file is automatically included in the model file `ContainerYard.mod` as illustrated in Figure 4-39.

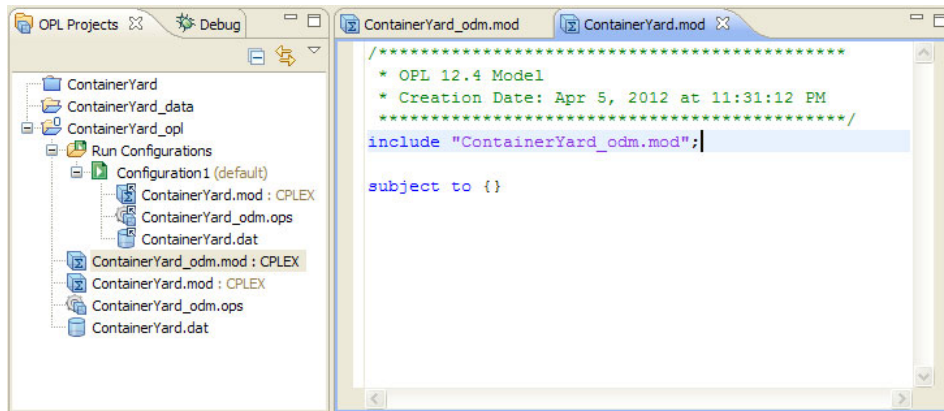


Figure 4-39 OPL model imports automatically generated data structures (`_odm.mod`)

`ContainerYard_odm.mod` should not be edited manually. This file is automatically updated, and any manual changes are lost during a subsequent update. The OR expert uses `ContainerYard.mod` to implement the optimization model. This file can also be used to include OPL data structures that are not relevant to the rest of the ODM Enterprise application. These data structures are not required in the ADM.

If the OR expert discovers that some data required for the model is missing and should come from the ADM, contact the software engineer. The software engineer can then add these data to the ADM and update the mapping (Example 4-1).

Example 4-1 OPL mapping of the application data model

```
// DO NOT EDIT: This file is automatically generated by ODM Enterprise.
// Any changes made directly to this file will be lost when it is
// automatically regenerated.
```

```
range booleanValues = 0..1;
```

```
// Input
```

```
tuple TDate {
    key string id;
};
```

```
tuple TPart {
    key string id;
    int shelfLife;
```

```

};
{TDate} dates = ...;
{TPart} parts = ...;

tuple TContainer {
    key string id;
    TDate arrival;
    int customs;
};

tuple TDemand {
    key TDate date;
    key TPart part;
    int qty;
};

tuple TParameter {
    int containerMin;
    int containerMax;
    TDate date;
};
{TContainer} containers with arrival in dates, customs in booleanValues = ...;
{TDemand} demands with date in dates, part in parts = ...;
{TParameter} parametersSingleton with date in dates = ...;
TParameter parameters = first(parametersSingleton);

tuple TContent {
    key TContainer container;
    key TPart part;
    int qty;
};

tuple TInspectionRule {
    key TContainer container;
    key int inspectionLength;
};
{TContent} contents with container in containers, part in parts = ...;
{TInspectionRule} inspectionRules with container in containers = ...;

// Output

tuple TOutSelection {
    TContainer container;
    TPart part;
    int qty;
};

tuple TOutSupplyDemand {
    key TPart part;

```

```

key TDate date;
int demand;
int supply;
int excessInventory;
int shortage;
};
{TOutSelection} outSelection with container in containers, part in parts;
{TOutSupplyDemand} outSupplyDemand with part in parts, date in dates;

```

A smidgen of OPL (part 2) - Tuples and Sets:

A tuple represents a column *schema*, which is analogous to a table declaration in SQL. Each field in the tuple represents a column, with a specified data type and perhaps a designation as a *key*. After you define a tuple, you can declare a table as a *set* of tuples. For example, in Example 4-1 on page 142, TPart is a tuple with fields id and shelfLife. The parts table is a set of TPart tuples, denoted by {TPart}.

In the statement {TPart} parts = ..., the ellipsis indicates that the parts table is read from a data source.

In the statement {TDemand} demands with date in dates, part in parts, the with clause specifies that part element of the tuple TDemand must be contained within the parts set.

In an expression such as Example 4-2 on page 147,

```
dexpr int fifo = sum(c in containers) selection[c]*dateOrder[c.arrival]
```

references to a tuple field are denoted by c.arrival where the tuple c is a member of the set containers.

Tuples can also serve as indexes for arrays, as denoted by the expression selection[c] in this same expression. That enables compact and transparent specification of decision variables and constraints in an optimization model. For more information, see “A smidgen of OPL (part 3) - Tuple slicing:” on page 149.

The tuple syntax in OPL enables powerful yet transparent modeling constructs that greatly simplify specification of mathematical optimization models.

For more information about tuple, sets, and other OPL data structures, see the CPLEX Optimization Studio InfoCenter documentation “Data structures” at:

http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/topic/ilog.odms.ide.help/OPL_Studio/opllangref/topics/uss_langref_datatypes_8.html

Completing the OPL model

To complete the OPL model, perform the following steps:

- ▶ Generating an OPL .dat file for testing
- ▶ Choices, targets, limits, and data

Generating an OPL .dat file for testing

After the OPL project is set up and all the necessary data structures are defined and included in ContainerYard_odm.mod, the OR expert can implement the optimization model.

To make it easier to test the model, the software engineer generates an OPL .dat file. The OR expert can use this file to run an OPL run configuration. Start the application from the IDE, select a scenario, and click **Development** → **Export Selected Scenario as OPL data file** as shown in Figure 4-40.

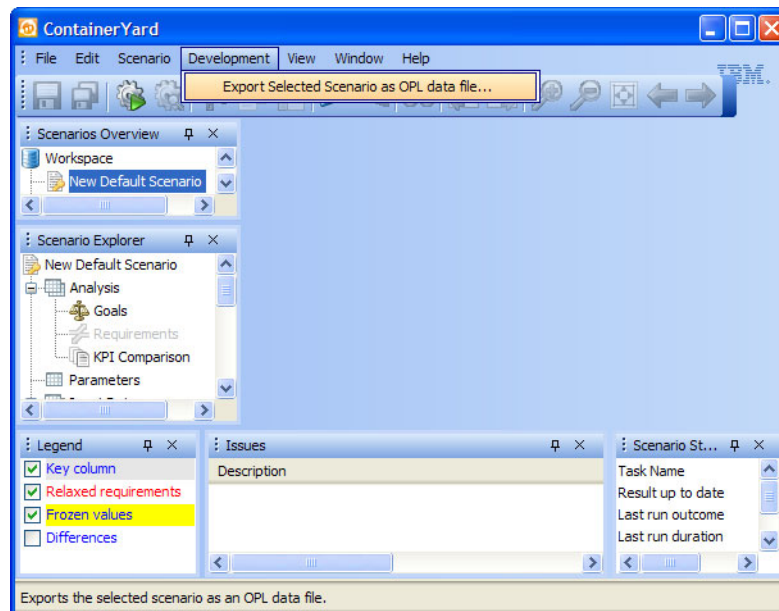


Figure 4-40 Export OPL .dat file for optimization model testing

The OPL .dat file is generated automatically, based on the application data model structure. The OR expert can then use real data while developing and testing the optimization model.

Choices, targets, limits, and data

Chapter 1, “Context for optimization and analytic decision support” on page 3 we introduced the concepts of *targets*, *limits*, *choices*, and *data*. The following

example shows see how these concepts translate into OPL and ODM for the ContainerYard example. This section explains how the targets and imposed limits map to the objective functions and constraints in your OPL model. It also explains how these key model pieces map to corresponding elements of your ODM application, namely goals and requirements.

► Decision support choices

What are the decisions you are trying to make? In the example, you want to know which containers should be selected to leave the yard. In the OPL model, this selection is represented by *decision variables*. At run time, the solver engine assigns to the decision variables a set of values that result in the *optimal* solution that is based on the specified targets. Targets are addressed in the next section.

– Decision types

Depending on the problem you are trying to solve, decision variables can be of several types: *integer*, *Boolean*¹, or *float*².

If you are solving a scheduling problem, some additional decision types can be used: *interval* and *sequence*.

In the example, for each container associate a decision variable of type Boolean that indicates whether the container is selected to leave the yard. In OPL, create an array of Boolean decision variables that is indexed over the set of containers as follows:

```
dvar boolean selection[containers];
```

OPL decision variable expressions (dexpr) are used to write more complex expressions in a compact way. These expressions are useful to represent the objective function terms that are of interest in the ODM Enterprise application.

► Targets

What are you trying to achieve? Are you minimizing costs, maximizing customer satisfaction, or is it a combination of the two?

In the ContainerYard example, your targets are to select containers to move from the yard to the warehouse to minimize the time spent in the yard and ensure that a minimum number of unnecessary parts are moved to the warehouse.

In the OPL model, these targets are represented as decision variable expressions. Based on the relative importance of the targets, a weighted sum

¹ Boolean decision variables can take one of two values, usually represented as true or false. They indicate whether the variable is present in an expression. Numerically, they map to 1 or 0, so they can be used in numerical formulas.

² Float decision variables can take continuous values. They have no specific precision, unlike in other programming languages, which often use float to indicate a specific precision (for example, 4-byte floating point).

of their corresponding decision expressions make up the OPL *objective function*.

The targets can be represented in the OPL model in these ways:

- Minimize the time that a container spends at the yard

Each container has an arrival date that is associated with it. This date is used to determine the order in which the containers arrived. The first day containers arrive is *date 0*, the next day containers arrive is *date 1*, and so on.

Based on the arrival date order, select containers that arrived first. This selection can be expressed in OPL with the decision variable expression in Example 4-2.

*Example 4-2 Target fifo*³

```
dexpr int fifo = sum(c in containers) selection[c]*dateOrder[c.arrival];
```

- Minimize the number of unused parts at the yard

For each part, compute this value as the difference between the number of parts that are shipped to the warehouse (*selpart[p]*) and the demand for the part (*d.qty*).

selpart[p] represents how many parts of type *p* are moved from the yard to the warehouse in the selected containers. In OPL, this is written as:

```
dexpr int selpart[p in parts] = sum(c in contents : c.part==p)
selection[c.container] * c.qty;
```

Thanks to this array of *dexpr*, the second term of the objective function can be defined as shown in Example 4-3.

Example 4-3 Target bycatch

```
dexpr int bycatch = sum(p in parts, d in demands :
d.date==parameters.date && d.part==p)
(selpart[p]-d.qty)
```

Now that all the required terms are defined, the objective function as a simple expression as shown in Example 4-4.

Example 4-4 Final target

```
minimize fifo + bycatch;
```

³ *dateOrder* is an array that defines the order of dates, which are represented as strings in this model.

Remember: Decision variable expressions in the objective function correspond to goals in the ODM Enterprise application.

► Limits

In addition to identifying the decision variables and targets, define what *constraints* these decision variables should respect. Constraints can reflect physical limits, for example maximum or minimum capacity. Such constraints are usually *hard*, which means they cannot be violated under any circumstances. Other types of constraints can express business requirements, for example indicating that a task must be completed before another one in a scheduling project. Constraints that can be relaxed in certain cases are known as *soft*.

Soft constraints often correspond to requirements or rules in the ODM Enterprise application. Such constraints must be labeled in OPL to make them accessible to business users for priority-based relaxation. For more information about how to configure the requirements in an ODM Enterprise application, see “Adding and configuring requirements” on page 157.

In the ContainerYard example, the following constraints were identified:

- To meet the demand for each car you need to produce on a certain day, you must have all the required parts available at the factory. Because this is a soft constraint, label it `ctDemand` as shown in Example 4-5.

Example 4-5 Constraint `ctDemand`

```
// To produce a car, all necessary parts have to be at the plant
forall (d in demands : d.date==parameters.date)
  ctDemand:
    selpart[d.part] >= d.qty;
```

A smidgen of OPL (part 3) - Tuple slicing:

Example 4-5 on page 148 illustrates a construction in OPL, called *tuple slicing*, that enables compact and efficient specification of decision variables and constraints. The expression `forall (d in demands : d.date==parameters.date)` filters the set `demands` to select only those tuples that have a date equal to the production date specified in the `parameters`. Thus, the `ctDemand` constraint is only enforced for the specified date.

Tuple slicing is analogous to a *Join* operation, so the previous expression is equivalent to a `SELECT` command in SQL. In some situations, a developer can use a `SELECT` command to preprocess data before populating the OPL data structures, or to use tuple slicing in a data initialization within the OPL model.

Tuple slicing provides a powerful means to use *sparsity* in optimization models. Sparsity reduces the size and complexity of the model, which directly effects the performance of the optimization solve. Using tuple slicing represents a best practice in optimization modeling.

For more information about tuple slicing and related topics, see the CPLEX Optimization Studio InfoCenter documentation “Filtering in tuples of parameters” at:

[http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/topic/ilog.odms.ide.help/OPL_Studio/opl_langref/topics/opl_langref_formalParams_filtering.html?resultof="slicing""slice"](http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/topic/ilog.odms.ide.help/OPL_Studio/opl_langref/topics/opl_langref_formalParams_filtering.html?resultof=)

- You can only move containers that arrive before the planning date (`parameters.date`). If a container `c` has a later arrival date, it cannot be selected, and the decision variable `selection[c]` must be assigned the value of “false”. In OPL, this constraint is written as shown in Example 4-6.

Example 4-6 Constraint ctArrival

```
// We can only pick containers that are actually in the yard:
// arrival date has to be before a specified date
forall (c in containers : dateOrder[c.arrival] >
dateOrder[parameters.date])
    ctArrival:
        selection[c] == 0;
```

- The business also imposes a minimum number of containers to be moved. Therefore, the number of containers that is selected must be greater than

or equal to the imposed minimum. Example 4-7 illustrates how to define such a constraint in OPL.

Example 4-7 Constraint ctContainerMin⁴

```
// a minimum number of containers have to be moved
// from yard to warehouse
if (parameters.containerMin > 0)
    ctContainerMin:
        sum (c in containers) selection[c] >= parameters.containerMin;
```

- You cannot move more containers than the warehouse can accommodate. The corresponding OPL constraint is shown in Example 4-8.

Example 4-8 Constraint ctContainerMax⁵

```
// we must respect the warehouse capacity
if (parameters.containerMax > 0)
    ctContainerMax:
        sum (c in containers) selection[c] <= parameters.containerMax;
```

- The last type of constraint is a little more tricky as it is only posted to the engine if the user defines a rule in the ODM Enterprise application.

Some containers might have to be inspected before they are moved. These containers are referenced in the set of tuples `inspectionRules`. Initially that set is empty, but it is populated whenever the user defines inspection rules in the ODM Enterprise application.

For each container referenced in this set, if its inspection is not completed before the planning date, an extra constraint is added to indicate that the container cannot be selected.

Example 4-9 shows the OPL syntax for this constraint.

Example 4-9 Constraint ctInspection

```
// some containers can only be moved after being inspected
forall (ir in inspectionRules : dateOrder[ir.container.arrival] +
ir.inspectionLength > dateOrder[parameters.date])
    ctInspection:
        selection[ir.container] == 0;
```

In this example, all constraints are labeled so they can be relaxed if needed. If you want to impose a hard constraint, remove its label in the model file. Such a constraint cannot be mapped to a requirement in ODM Enterprise, and business users cannot set priorities for it.

⁴ Note that `parameters.containerMin == 0` means no limit.

⁵ Note that `parameters.containerMax == 0` means no limit.

► Data

In the data section, look at input and output data separately.

– Input data

As indicated in “Creating the ODM-OPL mapping” on page 140, data structures and data elements (sets and arrays) are generated automatically from the Application Data Model. These declarations can be found in ContainerYard_odm.mod as shown in Figure 4-41. For a complete listing, see Example 4-1 on page 142.

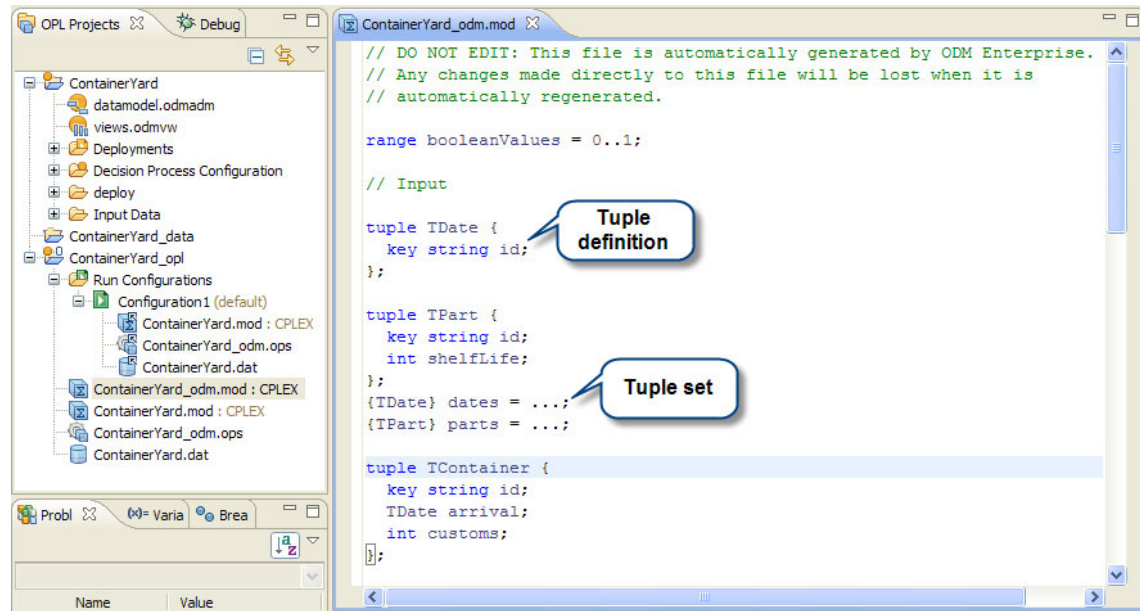


Figure 4-41 Generated .mod file that declares data from the application data model

In the example, additional input data dateOrder that is used in decision expressions and some constraints like ctArrival had to be created. The array dateOrder is started with the pre-processing script (Example 4-10).

Example 4-10 Pre-processing script to initialize input data

```
// intermediate array to store the arrival order of a container.
int dateOrder[dates];
execute {
    var o = 0;
    for (var d in dates)
        dateOrder[d] = o++;
}
```

Clarification: In this particular example, you could have initialized `dateOrder` more compactly by using OPL function `ord` in place of ILOG script:

```
int dateOrder[d in dates] = ord(dates,d);
```

Initialization inline by using OPL is generally more efficient and is therefore the better method to handle simple data pre-processing. In practice, however, you must often perform more advanced processing to initialize input data. Using an execute block as previously shown then becomes necessary.

– Output data

After the mathematical model is written, the data that are needed for the output views are available. It is likely that you must declare and initialize new data in the post-processing section.

In the example, the user wants to see how much of each part is available in the warehouse. To make this data available in ODM views, populate the corresponding set `outSelection` that stores the parts and quantities for each selected container. In OPL, populate this tuple set with the short script shown in Example 4-11.

Example 4-11 Post-processing script to initialize outSelection set

```
execute update_output {  
  // these sets are used in the output views of the ODM application  
  // set outSelection is used in the "Selection" views  
  for (var c in containers) {  
    if (selection[c] > 0) {  
      for (var i in contents) {  
        if (i.container == c)  
          outSelection.add(c,  
                           i.part,  
                           i.qty);  
      }  
    }  
  }  
  ...  
}
```

Similarly, the business user needs to analyze the supply and demand of parts. To make this data available in the pivot table and chart solution views, populate a set of tuples `outSupplyDemand` based on the selected containers. Example 4-12 shows the ILOG script code that is used to populate `outSupplyDemand`.

Example 4-12 Post-processing script to initialize outSupplyDemand set

```
// set outSupplyDemand is used in the Supply vs Demand
// pivot table and chart.
for (var p in parts) {
  for (var d in demands) {
    if (d.part==p && d.date==parameters.date)
      outSupplyDemand.add(p,
        d.date,
        d.qty,
        selpart[p],
        Opl.max1((selpart[p]-d.qty), 0),
        Opl.max1((d.qty-selpart[p]),0))
  }
}
```

Now that you have defined data, decision variables, decision variable expressions, objective function, and constraints, you have a complete optimization model as shown in Example 4-13.

Example 4-13 Model for ContainerYard example

```
/* *****
 * OPL 12.4 Model
 * Creation Date: Apr 5, 2012 at 11:31:12 PM
 * ***** */
include "ContainerYard_odm.mod";

// Decision variable selection[c] indicates whether container c is selected
// If selected, the container is moved from the yard to the warehouse
dvar boolean selection[containers];

// intermediate array to store the arrival order of a container.
int dateOrder[dates];
execute {
  var o = 0;
  for (var d in dates)
    dateOrder[d] = o++;
}

// selpart[p] represents the quantity of part p shipped to the warehouse
dexpr int selpart[p in parts] = sum (c in contents : c.part==p) selection[c.container] * c.qty;
```

```

// fifo is used to minimize the time a container spends in the yard,
// this is based on the container's arrival order
dexpr int fifo = sum(c in containers) selection[c]*dateOrder[c.arrival];

// decision expression "bycatch" represents amount of unnecessary parts at the plant
dexpr int bycatch = sum (p in parts, d in demands : d.date==parameters.date && d.part==p)
                    (selpart[p] - d.qty);

// objectif is to minimize the time a container spends at the yard
// and minimize the number of unused parts at the plant
minimize 1*fifo + 1*bycatch;

subject to {
    // To produce a car, all necessary parts have to be at the plant
    forall (d in demands : d.date==parameters.date)
        ctDemand:
            selpart[d.part] >= d.qty;

    // we can only pick containers that are actually in the yard
    forall (c in containers : dateOrder[c.arrival] > dateOrder[parameters.date])
        ctArrival:
            selection[c] == 0;

    // a minimum number of containers have to be moved from yard to warehouse
    if (parameters.containerMin > 0)
        ctContainerMin:
            sum (c in containers) selection[c] >= parameters.containerMin;

    // we must respect the warehouse capacity
    if (parameters.containerMax > 0)
        ctContainerMax:
            sum (c in containers) selection[c] <= parameters.containerMax;

    // some containers can only be moved after being inspected
    forall (ir in inspectionRules : dateOrder[ir.container.arrival] + ir.inspectionLength >
            dateOrder[parameters.date])
        ctInspection:
            selection[ir.container] == 0;
}

execute update_output {
    // these sets are used in the output views of the ODM application
    // set outSelection is used in the "Selection" views
    for (var c in containers) {
        if (selection[c] > 0) {
            for (var i in contents) {
                if (i.container == c)
                    outSelection.add(c,
                                    i.part,

```

```

        i.qty);
    }
}
}

// set outSupplyDemand is used in the Supply vs Demand pivot table and chart.
for (var p in parts) {
    for (var d in demands) {
        if (d.part==p && d.date==parameters.date)
            outSupplyDemand.add(p,
                                d.date,
                                d.qty,
                                selpart[p],
                                Opl.max1((selpart[p]-d.qty), 0),
                                Opl.max1((d.qty-selpart[p]),0))
    }
}
}
}

```

Configuring goals, requirements, and rules

After the optimization model is complete, the software engineer finishes the application prototype, this time including optimization. The only remaining tasks are to add and configure the following items:

- ▶ Goals: Based on the terms of the OPL objective function
- ▶ Requirements: Based on the OPL constraints
- ▶ Rules: Based on OPL constraints corresponding to rules

Adding and configuring goals

ODM goals correspond to OPL objective function terms, which are decision expressions. The coefficient or weight that is associated with each term translates to an ODM goal importance factor. This factor is the goal's importance relative to other goals. As shown in Example 4-14, fifo and bycatch decision expressions carry the weight of 1.

Example 4-14 OPL decision expressions and objective function terms

```

dexpr int fifo = sum (c in containers) selection[c] * ord(dates,c.arrival);

dexpr int bycatch = sum (p in parts, d in demands : d.date==parameters.date &&
d.part==p)
                    (selpart[p] - d.qty);

minimize 1*fifo + 1*bycatch;

```

Therefore, the default importance factor that is assigned to the corresponding goals will also be 1 (Figure 4-42). Business users can change the importance factors in their scenarios unless the application is specifically configured not to allow this.

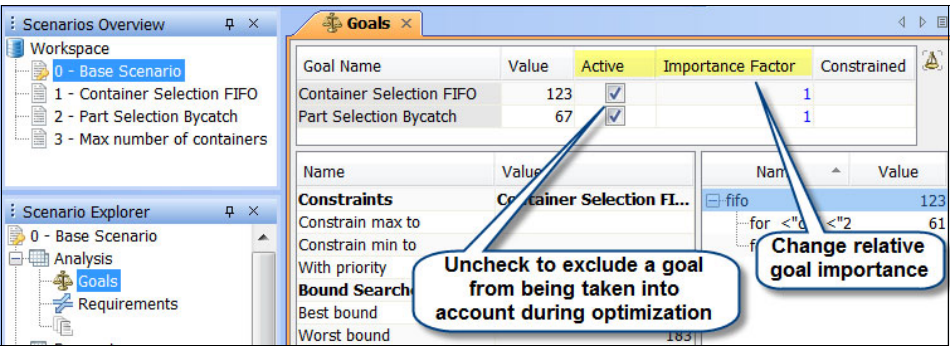


Figure 4-42 ContainerYard application: Goals view in ODM Studio

Optimization options: To see the solution value for a goal without taking it into account during optimization, use a weight of 0 in the objective. Note also that not every decision expression needs to be mapped to an ODM goal. In the example, the `selpart` array of decision expressions is not mapped to an ODM goal directly. It is, however, used by `bycatch`.

Goals are added in the Goals panel of the .odmom editor in the ODM Enterprise perspective in the ODM Enterprise IDE (Figure 4-43).

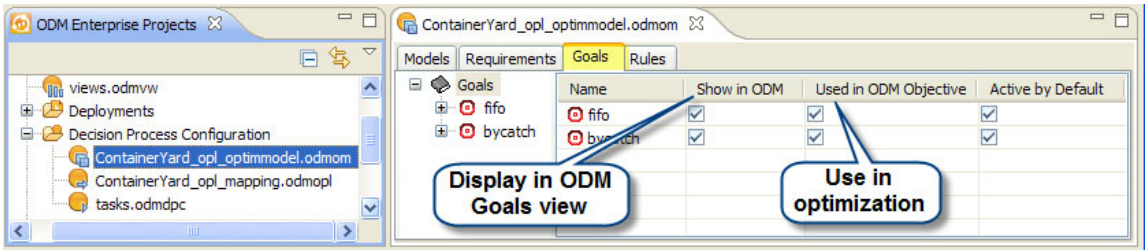


Figure 4-43 Goals editor in the ODM Enterprise IDE: All configured goals

Configure the goals in the Goals panel of the .odmom editor as shown in Figure 4-44.

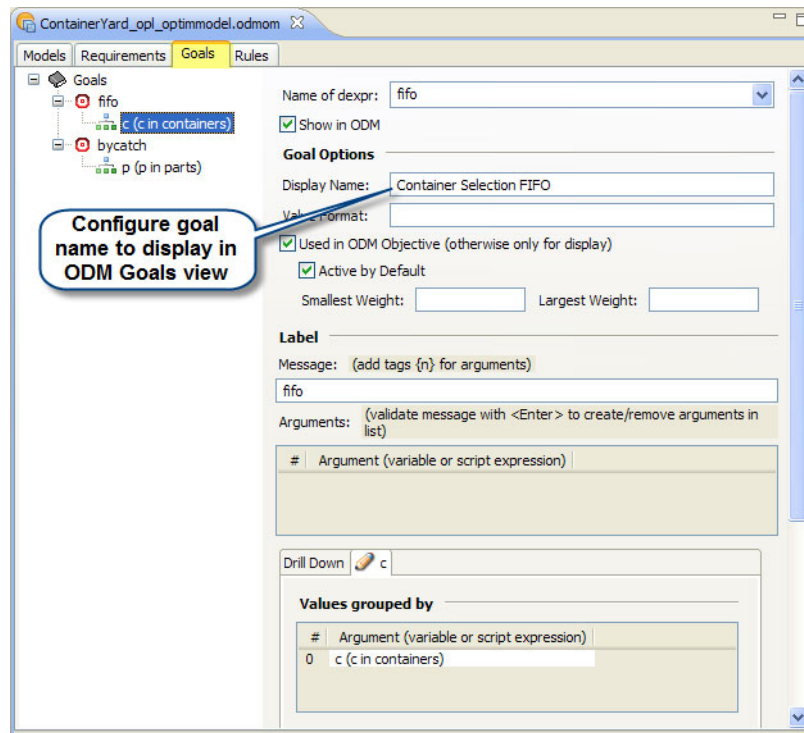


Figure 4-44 Goals editor in the ODM Enterprise IDE: Goal configuration

Adding and configuring requirements

Requirements that are labeled OPL constraints can be relaxed by the *solveAnyway* algorithm in ODM Enterprise. Labeled OPL constraints represent soft constraints. These constraints can be violated (relaxed) under certain circumstances.

Requirements are added and configured in the Requirements panel of the .odm editor (Figure 4-45).

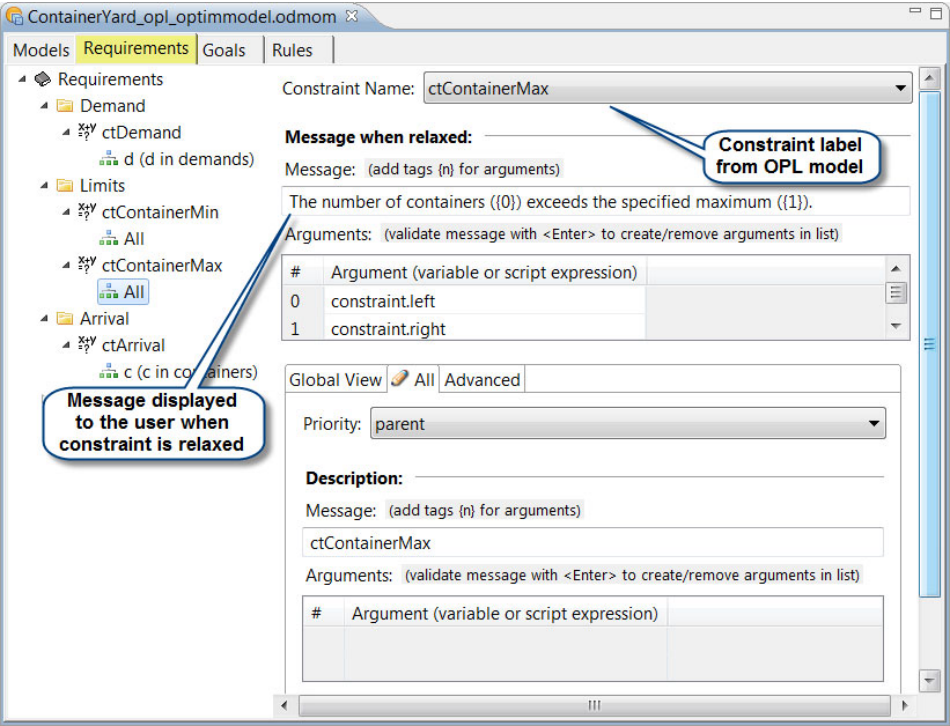


Figure 4-45 Requirements editor in the ODM Enterprise IDE: Requirement configuration

Relaxed requirements, if any, are displayed in the ODM Studio Requirements view (Figure 4-46).

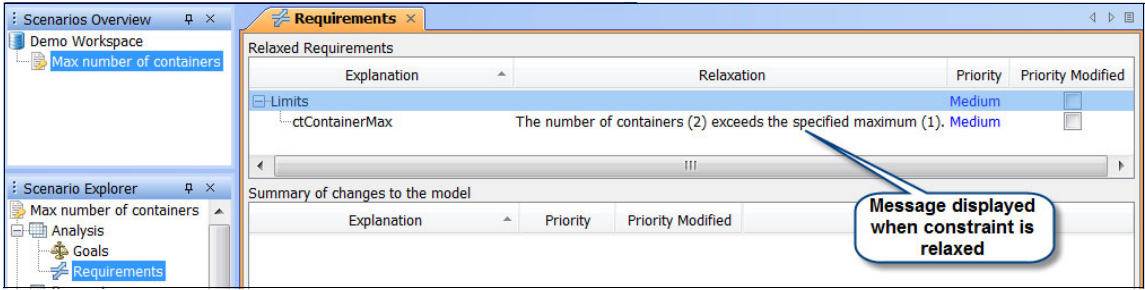


Figure 4-46 ContainerYard application: Message that is displayed when a constraint is relaxed

Adding and configuring rules

ODM Enterprise rules are used to create scenario-specific constraints that are based on particular data points. Although rules are declared in the OPL model as regular constraints, they are only active if an instance of the rule is created in ODM Studio. Rules can have associated requirements so that a violated rule shows up in the ODM Studio Requirements view.

Rules can be created by using the following steps:

1. The software engineer creates an ADM table corresponding to the data structure required for the rule. For more information, see “Creating the application data model (ADM)” on page 120.
2. The software engineer maps this ADM table to an OPL tuple set. For more information, see “Creating the ODM-OPL mapping” on page 140.
3. The OR expert uses the tuple set to create constraints to be used as ODM Enterprise rules. For more information, see “Creating the ODM-OPL mapping” on page 140.

The software engineer then completes the remaining steps:

1. Create a rule category in the ODM Enterprise IDE based on the OPL constraint (Figure 4-47).

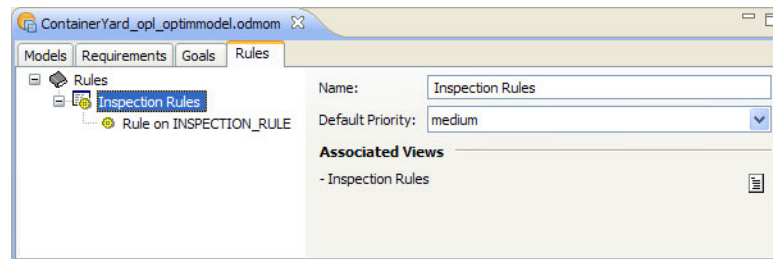


Figure 4-47 Rule Category: Inspection Rules

2. Create a rule template in the ODM Enterprise IDE based on the rule category (Figure 4-48).

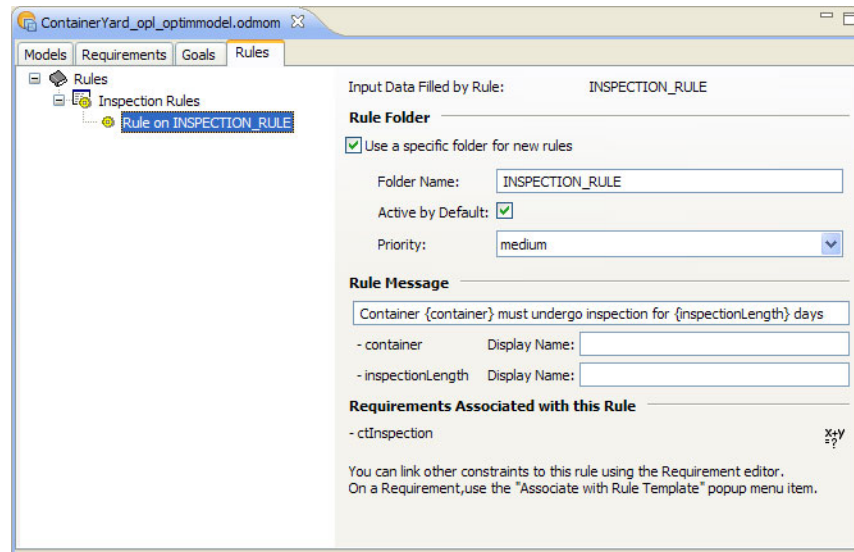


Figure 4-48 Rule template: INSPECTION_RULE

The rule view is now added in the views settings (Figure 4-49)

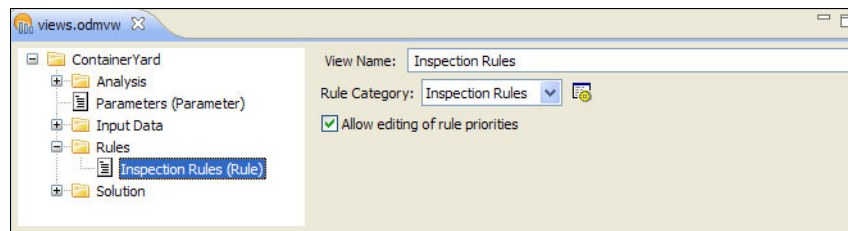


Figure 4-49 Rule view: Inspection Rules

The ODM Studio user can now create a rule instance for a particular scenario that is based on the rule template (Figure 4-50).

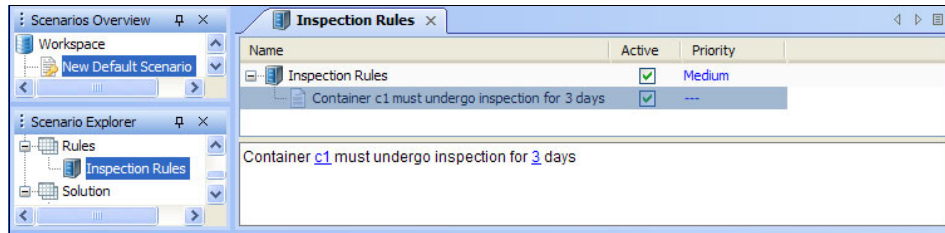


Figure 4-50 ContainerYard application: Inspection Rules

Obtaining feedback from the business user

You now have a full optimization prototype that you can share with the business user for feedback. Some development steps can be repeated as needed to refine the prototype before you move it to production.

4.2.5 Deploying the application

After the application is ready to be deployed, the software engineer configures deployment options by using the deployment settings editor in the IDE (Figure 4-51). The ODM Enterprise project can be deployed to one or more users, depending on the business needs. For more information about deployment configurations, see 3.1.12, “Deployment” on page 82 and 5.2.1, “Common ODM Enterprise deployment configurations” on page 170.

deployment_dev.odmvs deployment_prod.odmvs

Target Environment Datasources Additional JARs and Resources **Deployment**

Choose the deployment target configuration:

☒ **Individual**
Single user, local embedded ODM Scenario Repository, solve is performed on local machine.
Database Location: Relative path from user directory:

☐ **Enterprise**
Multiple user, remote server ODM Scenario Repository, solve can be performed locally or on Optimization Server.

☐ Use services on deployment server
Deployment application server URL:
☐ Use Data Server for scenario data management
☐ Use Optimization Server for performing solve
☐ Do not use deployment server: connect directly to ODM Scenario Repository, and use local solve process

ODM Scenario Repository database configuration
Set up the database for the ODM Scenario Repository.

Database
Type:
Hostname:
Port:
Instance Name:

Schema and User
☒ Use default values
Schema:
User:
Password:

ODM Scenario Repository management
Manage users, test ODM Scenario Repository performance and clear deleted repository items.

Verify configuration
Confirm that the current settings allow you to connect with the ODM Scenario Repository, Data Server, and Optimization Server.

Figure 4-51 Deployment settings

4.3 Conclusion

This chapter introduced you to the ODM Enterprise application development process, and ODM Enterprise IDE, the development environment for ODM Enterprise applications.

The Container Yard example from 2.6, “Advantages of ODM Enterprise solutions” on page 52 was used to illustrate the main steps of developing a complete ODM Enterprise application:

1. Analyzing requirements and defining the application scope
2. Defining the application data model
3. Creating the user interface and implementing data integration
4. Mapping the application data model to the optimization model
5. Implementing the optimization model in OPL
6. Integrating the optimization model into the application
7. Deploying the application

As mentioned earlier, the complete ContainerYard application described in this chapter is available for download. For more information, see Appendix C, “Additional material” on page 329.



Solution design and management

This chapter addresses the different solution offerings in the ODM Enterprise family. It also addresses various deployment, or system design, options. Finally, it covers the different input data sources and ways of processing that data.

This chapter includes the following sections:

- ▶ ODM Enterprise editions
- ▶ Deployment and system design options
- ▶ Input data and processing
- ▶ Conclusion

5.1 ODM Enterprise editions

There are multiple different editions of ODM Enterprise. This section provides an overview of each of them and how they can be “mixed and matched” to suit your development and application needs. Table 5-1 summarizes the ODM Enterprise editions and their components.

For more information about possible deployment configurations, see 5.2, “Deployment and system design options” on page 169. For more information about the specific use of each of the components, see Chapter 3, “ODM Enterprise architecture” on page 55.

Table 5-1 ODM Enterprise editions and components

	ODM Enterprise IDE	ODM Studio	CPLEX Optimizers and CPLEX CP Optimizer	ODM Enterprise Data Server
Developer Edition	YES	YES	YES	YES
Optimization Server			YES	
Optimization Engine			YES	
Data Server				YES
Client Edition		YES		
Planner Edition		YES		

5.1.1 ODM Enterprise Developer Edition

ODM Enterprise Developer Edition is packaged as a complete optimization application development environment that provides functionality for both the optimization expert and the software developer. It combines the best functionality of CPLEX Optimization Studio and ODM Enterprise within a development platform that is based on the Eclipse framework.

This edition provides developers with an integrated development environment (IDE) in which to develop and test tailored optimization applications that integrate the best features of these applications:

- ▶ An Eclipse-based IDE
- ▶ Embedded CPLEX Optimization Studio, including its IDE, OPL language, CPLEX Optimizer, and CP Optimizer
- ▶ Java customization for creating custom GUIs, data import and export, and extending other functions
- ▶ IBM ILOG JViews Enterprise (for custom visualization), bundled with ODM Enterprise
- ▶ IDE plug-in extensions that allow ODM Enterprise project management, and all of the configuration editors
- ▶ ODM Studio, the business user interface
- ▶ A development implementation of ODM Enterprise Optimization Server and ODM Enterprise Data Server for testing of ODM Enterprise applications, with a bundled preinstalled and preconfigured IBM WebSphere Application Server instance.
- ▶ Other tools and toolsets (for additional customization or reporting)

ODM Enterprise Developer Edition grants access and rights to develop and test an ODM Enterprise application.

5.1.2 ODM Enterprise Optimization Server

The ODM Enterprise Optimization Server is a centralized, server-based implementation of the CPLEX and CP Optimizer solving engines. It solves optimization models and runs algorithms to support decisions by planners by using the ODM Enterprise Client in an enterprise configuration. In an enterprise configuration, multiple users share scenarios and solving remotely. Use the Optimization Server when you are running long scenarios, and especially when several long running scenarios must be run in queued mode.

5.1.3 ODM Enterprise Optimization Engine

The ODM Optimization Engine is a local desktop implementation of the CPLEX and CP Optimizer solving engines. It is required for solving optimization models or the invocation of algorithms to support decisions by planners using the ODM Enterprise Planner Edition in a stand-alone individual configuration. In a stand-alone configuration, single users do not share scenarios concurrently, and the scenarios are solved locally. The planner can also be using the ODM

Enterprise Client Edition in an enterprise with local optimization configuration. This configuration is a multi-user mode, with users sharing scenarios by using ODM Enterprise Data Server, but solving them locally.

5.1.4 ODM Enterprise Data Server

The ODM Enterprise Data Server is a centralized, server-based collection of services that is intended to facilitate data management for ODM Enterprise applications. It provides a hub between the ODM Enterprise Client and the Scenario Repository database. In an enterprise configuration, the ODM Enterprise Optimization Server can also be connected to the Data Server.

The ODM Enterprise Data Server is required for deployment configurations that use ODM Enterprise Client Edition. Data Server is needed whether optimization is run locally, by using an instance of the ODM Enterprise Optimization Engine, or remotely, by using an instance of the ODM Enterprise Optimization Server. The Data Server is mandatory in distributed deployments, when ODM Enterprise Desktop Studio clients do not have a direct-access path to the Scenario Repository database.

5.1.5 ODM Enterprise Client Edition

ODM Enterprise Client Edition is a runtime component that allows business users to view, solve, share, and interact with distributed ODM Enterprise optimization applications. It implements the ODM Studio GUI.

ODM Enterprise Client Edition allows you to do planning in ODM Studio, which is the runtime user component of ODM Enterprise. It allows users to create scenarios and work interactively with them by using an instance of ODM Enterprise Data Server. However, it cannot work with a local data source. To solve or re-solve scenarios, you need either a local instance of the ODM Enterprise Optimization Engine, or a server-based instance of ODM Enterprise Optimization Server.

5.1.6 ODM Enterprise Planner Edition

ODM Enterprise Planner Edition is a runtime component that allows users to view, solve, share, and interact with distributed ODM Enterprise optimization applications. It implements the ODM Studio GUI.

ODM Enterprise Planner Edition allows the user to do planning in ODM Studio and to create scenarios. It includes the right to work interactively with scenarios, either locally or by using an instance of ODM Enterprise Data Server. To solve or

re-solve scenarios, you need either a local instance of the ODM Enterprise Optimization Engine, or a server-based instance of ODM Enterprise Optimization Server.

5.1.7 Staging database

A typical ODM Enterprise implementation uses a staging database to assemble data from other enterprise data sources into a single repository. For more information, see 3.5.1, “Scenario Repository” on page 99.

5.2 Deployment and system design options

When the completed application moves to production, input data for the application is retrieved from the corporate database. Then, planners work with either ODM Studio or the custom GUI to interact with the ODM Scenario Repository and ODM Enterprise Optimization Server. The planners work collaboratively, sharing scenarios and editing the solutions, and final plans that are based on the solved scenarios are written back to the corporate database.

Figure 5-1 shows one possible enterprise-level deployment configuration that uses the ODM Enterprise Data Server and the Optimization Server.

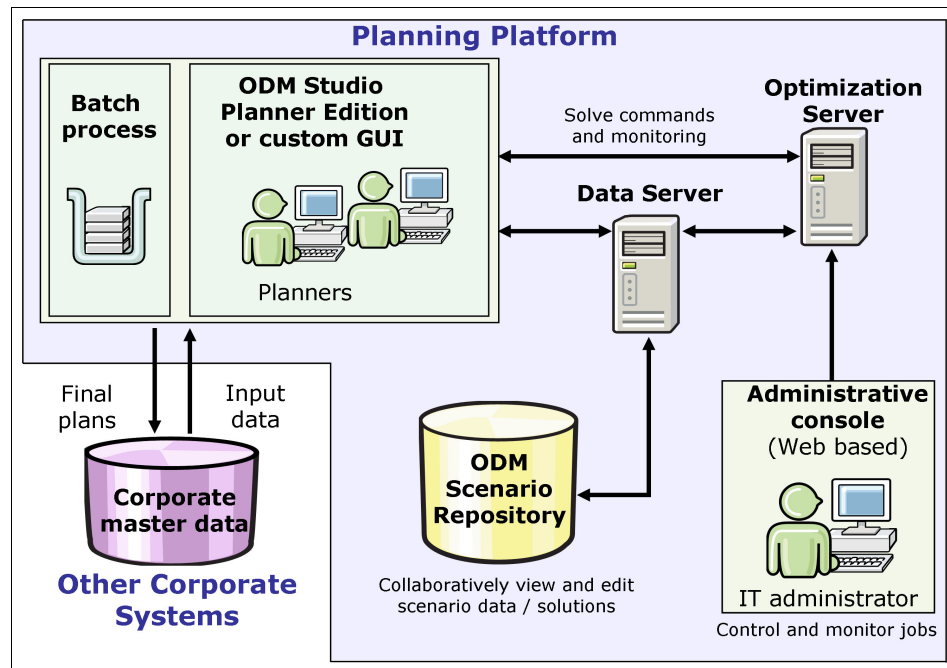


Figure 5-1 An example of enterprise-level deployment configuration

While the planners are working, IT administrators can monitor the performance of the ODM Enterprise Optimization Server by using a web-based console. This console allows administrators to see which applications are deployed to the Optimization Server and how many jobs are running, completed, or in the queue.

5.2.1 Common ODM Enterprise deployment configurations

The modular nature of ODM Enterprise allows for many different deployment configurations. These configurations can be adjusted to suit the different needs of your users and the network topology of your organization. This section provides an overview of the most common configurations.

The following configurations are fully supported. The illustrations have been simplified to show a single deployed ODM Enterprise application and only two users. However, the basic concept behind each configuration can be extended to support many more users, or support multiple ODM Enterprise applications.

Applications can be deployed in either *single-user* mode (individual deployment configuration) or *multi-user* mode (enterprise deployment configuration):

- ▶ In single-user mode, each deployed instance of the application is stand-alone. Users create scenarios on their own desktops and work with them primarily by themselves. There is no concurrent collaboration. Sharing of scenarios between users is handled by transferring files on the file system.
- ▶ In multi-user mode, each deployed instance of the application is connected to other instances of the same application. Users create scenarios on their own desktops, but each user can see other users' scenarios and work on them collaboratively, with concurrent changes enforced by locking.

Similarly, the solving of scenarios can be done either locally or remotely:

- ▶ Optimization problems that are not complex and that do not require fast, dedicated servers can be solved locally by using the ODM Enterprise Optimization Engine.
- ▶ Problems that are more computationally complex are solved on a centralized server by using ODM Enterprise Optimization Server.

Table 5-2 summarizes the common ODM Enterprise deployment configurations and their components.

Table 5-2 ODM Enterprise deployment configurations and components

	ODM Enterprise Planner	ODM Enterprise Client	ODM Enterprise Optimization Engine	ODM Enterprise Optimization Server	ODM Enterprise Data Server
Individual deployment	YES		YES		
Enterprise with Local Optimization Deployment		YES	YES		YES
Enterprise Deployment		YES		YES	YES

5.2.2 Individual deployment

The simplest deployment configuration for ODM Enterprise applications distributes the application in single-user mode (individual deployment configuration). Solving is done locally by using the ODM Enterprise Optimization Engine.

This configuration is appropriate for applications in which planners are remote and work independently. Users typically solve few times during a normal session, and there is rarely a need for online collaboration. The optimization itself is not computationally intensive, and can be solved on a standard desktop in seconds or minutes.

Individual deployment configuration has the following characteristics:

- ▶ The application (.odmapp file and supporting JAR files) is deployed to each user manually, or by using a scripted process.
- ▶ The client software that is used is ODM Enterprise Planner Edition. ODM Enterprise Client Edition cannot be used for an individual configuration because it requires the Optimization Server for solving.
- ▶ The ODM Scenario Repository database that is used is internal (Derby).
- ▶ Solves are performed locally by using the ODM Enterprise Optimization Engine.

This deployment configuration is depicted in Figure 5-2.

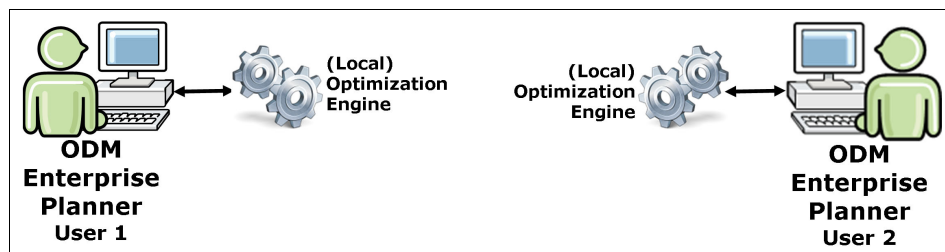


Figure 5-2 Individual deployment" Single-user mode, local solve

The following components are used in such a configuration:

- ▶ ODM Enterprise Planner: One instance for each user.
- ▶ ODM Enterprise Optimization Engine: One instance for each user.

5.2.3 Enterprise with local optimization deployment

In the enterprise with local optimization deployment configuration, the application is deployed in multi-user mode. In this mode, users can collaborate with each other and work with the same scenarios simultaneously. Solving is done locally by using the ODM Enterprise Optimization Engine. Scenario services, data caching, user security, and management are all provided by ODM Enterprise Data Server, which interconnects all users.

This configuration is appropriate for applications in which planners collaborate on fully or partly shared scenarios. Users typically solve only a few times during a normal session. The optimization itself is usually not computationally intensive and can be solved on a standard desktop in seconds or minutes. Solves are rather short in duration, but can be frequent. Modern desktops can accommodate heavy but short solves as well. If there are many planners, such a topology can be more expensive to deploy because of license costs.

The enterprise with local optimization deployment configuration has the following characteristics:

- ▶ The application (.odmapp file and supporting JAR files) can be deployed to each user manually, or by using the ODM Enterprise Data Server.
- ▶ The client software that is used is ODM Enterprise Client. ODM Enterprise Planner cannot be used for an enterprise with local optimization configuration.
- ▶ ODM Enterprise Data Server provides enhanced data storage, caching, and security features, plus user maintenance.
- ▶ The ODM Scenario Repository database that is used is external, and is connected to the clients through ODM Enterprise Data Server.
- ▶ Solves are run locally by using the Optimization Engine.

This deployment configuration is depicted in Figure 5-3.

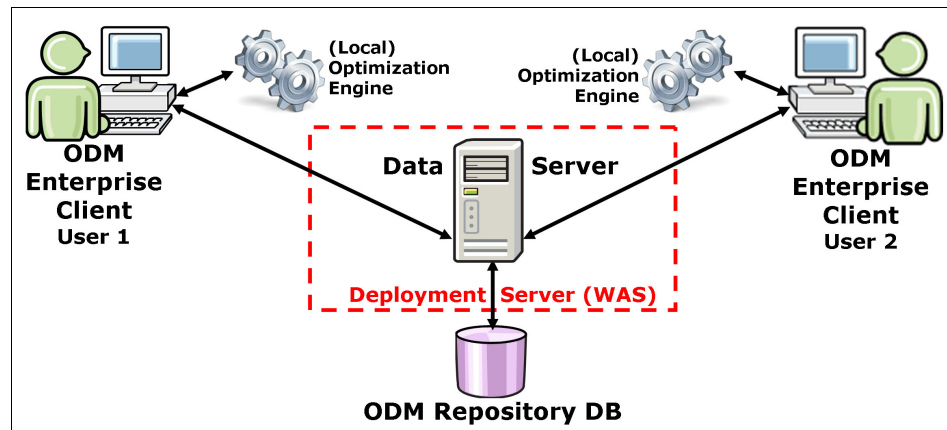


Figure 5-3 Enterprise with Local Optimization Deployment: Multi-user mode, Data Server, local solve

The following components are used in such a configuration:

- ▶ ODM Enterprise Client: One instance for each user.
- ▶ ODM Enterprise Optimization Engine: One instance for each user.
- ▶ ODM Enterprise Data Server: One instance to interconnect users and provide data and scenario services to all users of all ODM Enterprise applications.

5.2.4 Enterprise deployment

In enterprise-level deployment configurations, the application is deployed in multi-user mode. This mode allows users to collaborate with each other and work with the same scenarios simultaneously. Solving is done remotely by using the ODM Enterprise Optimization Server. Scenario services, data caching, user security, and management are all provided by ODM Enterprise Data Server, which interconnects all users.

This configuration is appropriate for applications in which planners collaborate on fully or partly shared scenarios. Users typically must initiate long solve durations, which might be overnight. They also must typically run several long-running scenarios that are queued on the server to run overnight. Such a topology allows users to share the Optimization Server license cost and server processor power.

The enterprise deployment configuration has the following characteristics:

- ▶ The application (.odmapp file and supporting JAR files) can be deployed to each user manually, or by using ODM Enterprise Data Server.
- ▶ The client software that is used is ODM Enterprise Client. ODM Enterprise Planner cannot be used for an enterprise configuration.
- ▶ ODM Enterprise Data Server provides enhanced data storage, caching, and security features, plus user maintenance.
- ▶ The ODM Scenario Repository database that is used is external, and is connected to the clients through ODM Enterprise Data Server.
- ▶ Solves are run remotely by using the ODM Enterprise Optimization Server, which is connected to the clients through ODM Enterprise Data Server.

In its simplest form, this deployment configuration is shown in Figure 5-4.

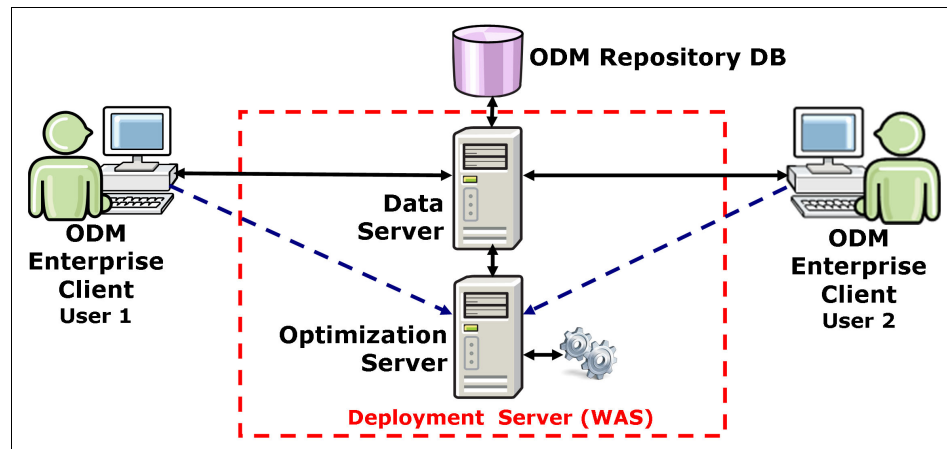


Figure 5-4 Enterprise Deployment: Multi-user mode, Data Server, remote solve

The following components are used in such a configuration:

- ▶ ODM Enterprise Client: One instance for each user.
- ▶ ODM Enterprise Optimization Server: One instance to interconnect users and provide optimization services to all users of the ODM Enterprise applications.
- ▶ ODM Enterprise Data Server: One instance to interconnect users and provide data and scenario services to all users of the ODM Enterprise applications.

5.2.5 Other considerations while you are designing the solution

This section briefly addresses some additional factors that you need to consider while you are designing the ODM Enterprise solution.

Number of concurrent solves

Expected number of planners who will run the ODM Enterprise solution concurrently is one of the main factors to consider in the solution design process. Having more planners who run the ODM application concurrently can increase the load on all of the ODM Enterprise components:

- ▶ The number of problem instances that are simultaneously solved by the ODM Enterprise Optimization Server can increase.
- ▶ ODM Enterprise Scenario Repository database might require higher capacity to handle all the scenarios that are created by the planners.
- ▶ Increased data traffic between the ODM Enterprise Client and the ODM Scenario Repository database, and between the ODM Enterprise Optimization Server and the ODM Scenario Repository database. This traffic requires a more powerful ODM Enterprise Database Server.

Input data size

Similar to the number of concurrent solves, increase in the size of the input data can affect performance of all of the ODM Enterprise components:

- ▶ In the general case, solution time by ODM Enterprise Optimization Server (or engine) increases with the size of the problem instance.
- ▶ ODM Enterprise Scenario Repository database might require higher capacity to handle size of the scenarios (both input and also output data).
- ▶ High volume data can increase the data traffic, requiring a more powerful ODM Enterprise Database Server.

Math problem complexity

There are two main factors that affect the performance of the optimization engine are size of the problem instance, and problem complexity. Complexity is mostly driven by the type of the optimization method that is required to solve the business problem. The type of the optimization problem is generally defined by

the three-dimensional categorization that is shown in Figure 5-5. The complexity increases from Continuous to Discrete, Linear to Non-linear, and Deterministic to Non-deterministic.

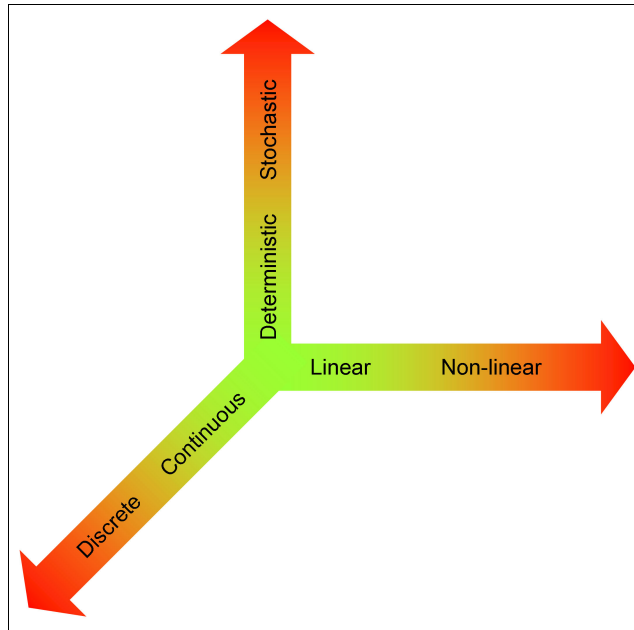


Figure 5-5 Math problem categorization in terms of complexity

Continuous versus Discrete

This categorization is made based on the domain of the decision variables that are used in the problem. If a decision variable is defined in a continuous domain (it can have any value within the defined range), it is called a continuous variable. Example include body temperature, which can have any value between 94 - 105 Fahrenheit, or the height of a football player (any value between 5 - 7 feet). If a decision variable is defined in discrete domain (it can have a finite number of values within the defined range), it is called as discrete variable. Examples include the number of nurses needed in a hospital, or the number of distribution centers needed in a supply chain network. Boolean variables represent a special case of discrete variables that have only two possible values, usually denoted as "true" or "false". These values are represented numerically as 1 or 0. They might represent, for instance, an open or closed warehouse at a particular location.

A problem is categorized as continuous if all the decision variables used in the problem are continuous. This type of problem can be solved by using *Linear Programming* algorithms if the relationships among the decision variables are linear.

A problem is categorized as discrete if one or more of the decision variables that are defined in the problem are discrete. Generally, this type of problems, called *Mixed Integer Programs*, is solved (in the worst case) by solving a continuous problem for each enumeration of the discrete variables. Therefore, discrete problems are more complex than the continuous problems, requiring a more powerful Optimization Server (or engine).

Consideration: Mixed integer programming requires much more complex algorithms than linear programming. The CPLEX Optimizer uses a method called *Branch-and-Cut*. It first uses combinatorial calculations to determine additional constraints, called cuts, that are implied by the existing constraints and the finite domains of the discrete variables. These cuts reduce the size of the solution space that the algorithm must search. In some cases, they are sufficient to find an optimal solution that satisfies the discreteness criteria by linear programming alone.

If not, the algorithm sets up a search tree by successively assigning each discrete variable to one of its possible values. This process can imply additional cuts. At each node of the tree, the algorithm then solves a linear program that approximates the original problem. The solution might find a discrete optimal solution better than any found so far. Or it might establish a bound on the possible optimal value that precludes the need to search further down the tree. If neither of those events occurs, the algorithm creates new branches in the search tree corresponding to the possible values of the next discrete variable.

After years of research and experience, CPLEX is recognized as being among the best performing algorithms for solving mixed integer programming problems. ODM Enterprise uses that performance to build practical business decision support solutions that are based on optimization.

Linear versus Non-linear

This categorization is made based on the type of the relationships that are used in the objectives and the constraints of the problem.

A relationship is called linear if the result is directly proportional to the input. In the following example, *Total Production Cost* is a function of two variables, *Number of Parts* and *Machine Hours*, both having an exponent of 1.

$$\text{Total Production Cost} = 5 * (\text{Number of Parts}) + 0.5 * (\text{Machine Hours})$$

A relationship is called non-linear if the results are not directly proportional. In the following example, *Inventory Requirement* at a facility is calculated as the square root of the total *Demand Allocation* to that facility.

$$\text{Inventory Requirement} = \text{sqrt}(\text{Demand Allocation})$$

A problem is categorized as linear if all of the relationships that are used in the problem are linear. It is categorized as non-linear if at least one of the relationships that are used in the problem is non-linear. Non-linear problems are more complex and harder to solve, and so requiring a more powerful optimization algorithm¹.

Clarification: A mathematical formula is linear if it has all of these characteristics:

- ▶ Consists of a sum of terms
- ▶ Each term contains at most one variable (possibly multiplied by a constant)
- ▶ The variable has an exponent of 1

All other relationships are non-linear.

Non-linear relationships include formulas in which the terms have products of variables or the variables have exponents not equal to 1. They also can include non-linear functions such as log and exp.

Quadratic relationships are a special case of non-linear relationships. Each term has either a single variable with an exponent of 2, or a product of exactly two variables, each with exponent 1.

Deterministic versus Stochastic

This categorization is made based on the certainty of the input data. If all of the input data of the problem is known (certain), the problem is categorized as deterministic. If some of the input data is uncertain, the problem is categorized as stochastic. For stochastic problems, random variables are used in the objective functions or constraints. Examples include financial optimization problems where stock prices are uncertain, and some statistical distribution functions (normal, weibull, and so on) are used to represent the prices. Similarly, forecast demand distribution can be used while planning the inventory requirements in a retail shop.

Stochastic problems are generally solved by solving multiple deterministic problems, hence requiring more powerful algorithms².

¹ The CPLEX Optimizers can solve linear and quadratic, continuous, and mixed continuous/discrete optimization problems. The CPLEX CP Optimizer can solve general non-linear, discrete (but not continuous) optimization problems.

² The CPLEX Optimizers can solve stochastic problems with discrete random variables, provided they are properly formulated and do not exceed the usual size bounds.

Tip: ODM Enterprise provides a tuning tool to improve the performance of the optimization engine by finding the best combination of engine settings. The tuning can be done for one or several run configurations so that the search performs best when your application is deployed. Because in a deployed application, data changes between each invocation of the engine, performance tuning can be run on a set of problem instances.

Response time requirement

Depending on the cycle time of the business process, response time (time between the planners running the application and getting the results) can be a critical requirement. Generally, when solving strategic and tactical problems, such as supply chain network design, the planners run the application a couple of times (or less) during a week. These types of problems do not require a fast response time. However, other operational problems, such as emergency resource dispatching, require planners run the system many times during the day. In these types, the response time is extremely critical. In cases where the response time is critical, all components of the ODM Enterprise solution need to work faster.

Network connectivity and speed

In enterprise deployment options, scenario data transfers occur between Data Server and Optimization Server, and also between ODM Enterprise client and Data Server. Therefore, network speed plays an important role in system performance. In all network connections, use Local Area Network (LAN) instead of Wide Area Network (WAN). If WAN is required, consider using virtualization technology (for example, from Citrix) or a Web User Interface as shown in Figure 5-6 on page 181.

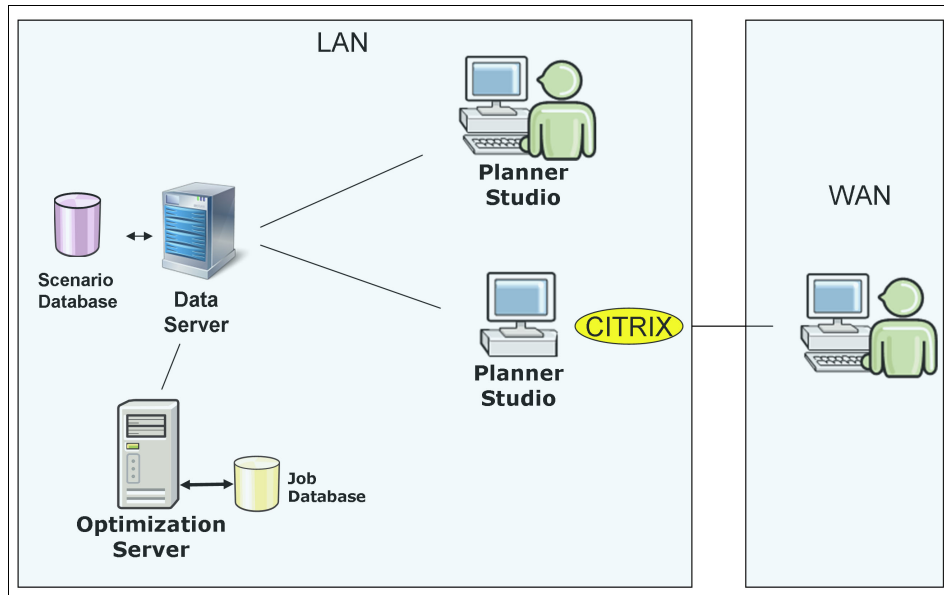


Figure 5-6 A deployment option using virtualization technology

Tip: The Performance Test allows you to test the quality (speed) of the database and your connection to it. This test can be run either from the ODM Enterprise IDE or from a running multi-user ODM application in production.

This is not a definitive test. Instead, it is a tool to help you pinpoint database connection issues if you feel that performance of your ODM application is not optimal.

The test writes data to the database and tests the speed and accuracy of the results (Save Quality). It then reads from the database, again testing the speed and accuracy of the results (Load Quality). It then rates both performance indexes on a scale of 1 (Poor) to 5 (Excellent).

If this test returns less than optimal results, contact your MIS department for suggestions for speeding up either the database itself or your network connection to it.

Security

Protocols that are used to connect to the Optimization Server and the Data Server do not implement any security model or secured protocol. That is, they do not require any credentials from a client. After the Optimization Server and the Data Server are deployed, any user in the organization with an installed copy of

the ODM Enterprise IDE can use its tools and APIs. They can deploy ODM applications to the Optimization Server, submit and control solve jobs in the Optimization Server, and create and view scenarios by using the Data Server. If security is an issue for you, limit access to the ODM Enterprise IDE to trusted developers.

5.3 Input data and processing

Other products and solutions can be integrated with an ODM Enterprise solution. To ensure compliance with corporate standards, ODM Enterprise uses the Java Enterprise Edition platform architecture to seamlessly integrate within existing application server deployments. It can be integrated with deployments such as the IBM WebSphere Application Server and WebSphere Application Server Community Edition. Integration with external systems is usually done at the data level. Because of the analytical nature on an ODM Enterprise solution, ERP and CRM systems rarely require application-level integration. However, if integration is required, the ODM API can be used to provide a tighter integration.

Typically, data comes from enterprise business systems such as SAP, Oracle ERP, and PeopleSoft, and logistics, scheduling, and claims systems. A healthcare company that wants to optimize the deployment of medical personnel across multiple properties can tap into data from HR (addresses, work schedule), competency (qualifications, certifications), and forecasting systems to achieve best results.

Table 5-3 provides example corporate input and output data for various industries.

Table 5-3 Corporate input and output data for manufacturing, health care, and utility industries

Application / Industry	Input	Output
Aggregate Planning / Manufacturing	▶ Demand (forecast, orders, safety stock) ▶ Raw material supply (on hand inventory, scheduled arrivals, procurement plan, vendor capacity, and contracts) ▶ Capacity (facility, tools, and labor) ▶ Limitations (capability, work calendar, and maintenance schedule)	▶ Demand fulfillment, production, shortage, and inventory position ▶ Capacity utilization and expansion requirements ▶ Material utilization and procurement requirements

Application / Industry	Input	Output
Nurse Scheduling / Healthcare	▶ Personnel data ▶ Work calendar (shifts, breaks, vacations, and so on) ▶ Workforce demand by shift, location, and qualification ▶ Pay rate ▶ Rules (government, union, labor laws, and corporate)	▶ Resource assignment by department and location ▶ Personnel working schedule
Emergency Workorder Management / City agencies or utility companies	▶ Workorder data (report time, earliest and latest completion times, criticality, and location) ▶ Requirements (resources, equipment, and expected duration) ▶ Resources (skill, availability, pay rate, and location) ▶ Geographic information data	▶ Resource assignment and routing ▶ Workorder schedule and sequence ▶ Operational cost ▶ Tooling and parts

ODM Enterprise provides entry points to define and implement custom data importers (to import data from corporate databases into a scenario) and data exporters (to publish data from a scenario to a corporate database or to generate reports).

The *Data Service* provides access to all tasks related to workspace, scenario, and user management. This access includes these privileges:

- ▶ Adding, removing, and modifying workspaces, workspace folders, and scenarios
- ▶ Locking scenarios for exclusive use
- ▶ Importing, exporting data from/into scenarios
- ▶ Managing users and rights

Data Services also provides access to *data checkers* (to implement additional business checks during the data validation phase of a scenario) and *data migrators* (to smoothly handle data migration between two deliveries of an ODM application). They can also add a *custom authentication mechanism* to connect to a corporate user directory.

5.4 Conclusion

ODM Enterprise offers a wide range of configuration options from Desktop to Enterprise, It gives organizations the flexibility to start their optimization efforts with a small operation. Business decision makers have access to functions and views of decision support analytics while gaining confidence in using the optimization as an intrinsic part of the business. This process in turn expands the analytical process to other aspects of the enterprise.



Part 2

Customer use cases

This part includes two typical real-life business use cases:

- ▶ Case study: An insurance company
- ▶ Case study: A manufacturing company



Case study: An insurance company

This chapter describes how a fictional *insurance company* uses ODM Enterprise to optimize the assignment of claims to the investigators. The goal is to match them with the most cost-effective cases, increasing productivity while dramatically lowering operational costs.

This chapter includes the following sections:

- ▶ Company overview
- ▶ Business problem
- ▶ Business requirements
- ▶ Optimization problem
- ▶ Solution development
- ▶ Optimization
- ▶ Conclusion

6.1 Company overview

Operating in the United States, this insurance company is one of the country's largest providers of property and casualty insurance, reinsurance, and financial services. Established in 1985 by a consortium of leading companies to provide hard to find excess liability coverage, the company has grown rapidly. It has done so by building long-term partnerships with its brokers and clients in each of the nation's major insurance markets. It insures over 15 million cars, boats, trucks, and motorcycles. Its auto insurance division provides coverage of privately owned vehicles with nine basic components: liability, medical, uninsured motorist, comprehensive, collision, car rental, death and dismemberment, total disability, and loss of earnings.

6.2 Business problem

One of the most important divisions of the company is the Investigation Services. The company is split into four major geographic regions: East, Mid-West, Mountain, and West. It employs and subcontracts just over 6,000 investigators nationwide who work with adjusters to perform these tasks:

- ▶ Conduct on-site and post accident investigation
- ▶ Claimant activity checks
- ▶ Surveillance
- ▶ Fraud investigation
- ▶ Taking witness statements
- ▶ Scene investigations documentation
- ▶ Investigate liability insurance fraud

Regions are further subdivided into districts for administrative purposes. However, the distribution of investigation cases to investigators is done at a national level from the company's headquarters in Birmingham, Alabama.

Cases are assigned to investigators on a weekly basis except for Level 1 cases, which carry a higher priority and are assigned daily. Most cases are automatically generated by the company's internal business systems. Each case is composed of one or many work elements that are called *items*. A case typically has an average of 10 items such as witness interviews, accident reports, and medical records that need to be investigated.

The business systems create these items based on information that is collected from the claim and policy systems. This information is augmented with data from external sources such as credit history and driving records. The investigators

must follow a complex set of business rules, and store the outcome in the company's ICS (Investigation Case System).

Case items are assigned an estimated date of completion (EDC). They are assigned to investigators by using a complex assignment criterion. The criteria include type, location, and volume of the cases, plus the investigator's training and certification status, qualifications, area of coverage, and availability. Although items from the same case can be assigned to one investigator or across multiple investigators, the latter should be avoided for productivity and efficiency reasons.

After the investigation is completed, the investigator submits the results back to the ICS. The investigator can also create new items based on the information gained through the investigation. These items are then placed back in the assignment pool of items to be distributed. It will most likely be assigned to a different investigator if it deals with issues that require investigation in another geographic area. After all items in the case are reviewed and deemed completed, the case is marked as closed in the ICS. The ICS then notifies the adjusters and mediators. If there is a problem with any of the items in a case, the items are returned to the investigator and the case remains open. Items being returned potentially results in financial penalties and sanctions.

About 80% of the investigators are full-time employees, with the remaining 20% being contractors who charge their time by the hour. All investigators use company supplied vehicles and are reimbursed for fuel costs. These expenses have a significant effect on the company's overall operating expenses.

The insurance company approached IBM to help them find an optimized case assignment solution with these goals:

- ▶ Assign items to the most efficient and effective resource first
- ▶ Prioritize use of the lowest cost resources
- ▶ Minimize driving distance and time for the investigator
- ▶ Assign items to resources with the highest efficiency and quality ratings
- ▶ Minimize the number of distinct investigators who work on a single case

After holding a *discovery workshop* with the client, the IBM Optimization Solutions team recommended an analytical decision support solution based on ODM Enterprise's optimization technology. For more information about discovery workshops, see 4.2.1, "Analyzing requirements and defining application scope" on page 115.

6.3 Business requirements

The following main requirements summarize the business objectives of the case assignment system:

- ▶ Maximize the efficiency of resources by assigning new items to investigators that already were working on items in the same vicinity. Certain work items, like obtaining accident reports or collecting witness reports, always take investigators to the same locations such as court houses or a sheriff's office. Therefore, the efficiency of the investigators can be increased by assigning any additional work that takes them to the same locations to which they are already traveling.
- ▶ Maximize profitability by minimizing investigation costs around resources, travel, and fuel costs. Basically, less time spent on driving to different locations translates to lower travel and fuel costs.
- ▶ Contribute and meet the company mandated dispute resolution dates. Typically, insurance matters that require investigation often involve multiple parties, who all agree to a hard resolution date. Missing this date has substantial penalties. Conversely, there are also bonuses for an early completion of the investigation.

Supporting these business requirements are the non-functional and user interface requirements, which are articulated in the following sections.

6.3.1 Non-functional requirements

The system should support the following non-functional requirements;

1. An end-to-end optimization process should be completed in one hour or less. The client's IT management stated that they have a small window of time to publish the case assignments to the field. This limitation required the system to run any data pre-processing, load, transformation, and optimization jobs in less than one hour.
2. The system should be available during business hours, 7:00 AM to 6:00 PM US Eastern Standard Time.
3. The system should be able to process a maximum of two concurrent solves with a maximum response time of 10 minutes per solve.
4. Access control and authentication to ODM Enterprise must be performed by using the company's centralized LDAP service.
5. The system must implement comprehensive logging processes to facilitate support and error tracing.

6.3.2 User interface requirements

The client stakeholders expressed the importance of being able to track and “zoom-in” on an investigator’s assignments, especially during the initial testing phase. Several data views (known as Views in ODM Enterprise), both in tabular and graphical formats, were requested. These Views provide users with the ability to closely examine the output data.

The following are the main UI requirements:

- ▶ The ability to view the following reports in a tabular format.
 - View assignments by investigator
This view provides a list of all the items that are assigned to the investigators.
 - View days of work in inbox
This view provides a view that displays the total work that is assigned to all investigators.
 - Work assignment calendar
This view provides the ability to view assignments (past and newly assigned) of a particular investigator across a timeline.
 - Available capacity
This view displays the available time per investigator in a week’s time. Available time is defined as *posted available hours - item required hours*.
 - Assignment justifications
This view provides an explanation of why each item was assigned to a specific investigator by the optimization model. The report shows how each item met the relevant business requirement (a yes / no value), or a numeric value as dictated by the associated constraint.

Figure 6-1 shows two views (Work Assignments and Days of Work in Inbox) in a typical ODM Studio tabular view.

The screenshot displays the IBM ODM Studio interface. On the left is the 'Scenarios Overview' pane with a tree view showing 'Workspace' (Archive, Research, Base Scenario) and 'Scenario Explorer' (Base Scenario, Analysis, Goals, Requirements, Input Data, Solution, Days of Work in Inbox, Work Assignments, Employee Zip Information, Number of Employees Assigned 1,2, ... zip code). The main area shows two tabs: 'Work Assignments' and 'Days of Work in Inbox'. Both tabs show a table with a filter bar indicating 'Filter is not active' and the number of rows displayed.

Work Assignments Table:

CaseID	Item Number	Employee ID	Distance To Zip
8	19	28765	13.5
11	13	5095	16.3
11	16	52613	0
11	18	5095	0
11	19	52937	0
12	20	28310	3.4
49	14	3332	0.1
58	34	30447	0
69	19	30178	0
74	42	C07198	0
83	36	29909	4.2
99	30	28765	4.9
100	11	28765	10.3
110	25	29909	4.2
110	26	29563	37.6
124	22	27172	50.8
127	24	25482	6.2
127	25	25482	6.2
133	8	28863	4.7
167	17	12540	5.6
178	10	5893	0.4
187	13	25482	6.2
204	30	27545	0.8
212	24	11112	34.7

Days of Work in Inbox Table:

Employee ...	Days Work In Inbox	Days Assigned On Oct3
169	3.575	2.04
5893	1.885	6.532
6603	4.406	9.406
6701	0	4.079
6828	8.439	13.439
11024	0	0.279
25482	57.941	3.606
27167	0	2.192
27172	8.948	4.606
27805	5.61	2.946
28517	18.405	3.451
28863	34.234	5.309
29272	0	0
29321	6.624	1.442
29685	2.9	2.442
C00949	0	0
C04117	0	0
C04584	0	0
C05486	0	0
C07178	0	0
C07198	0	0
10640	0.229	2.072
26941	4.082	7.794
27793	9.721	6.902

Figure 6-1 Typical ODM Enterprise tabular view

- ▶ The ability to visualize the following reports on a map. IBM ILOG JViews Maps allows you to add geographic data visualization with the ODM Enterprise Studio client.
 - Item assignments by investigator.
 - Item assignments by area.

Figure 6-2 shows a custom built map that uses IBM ILOG JViews Maps.

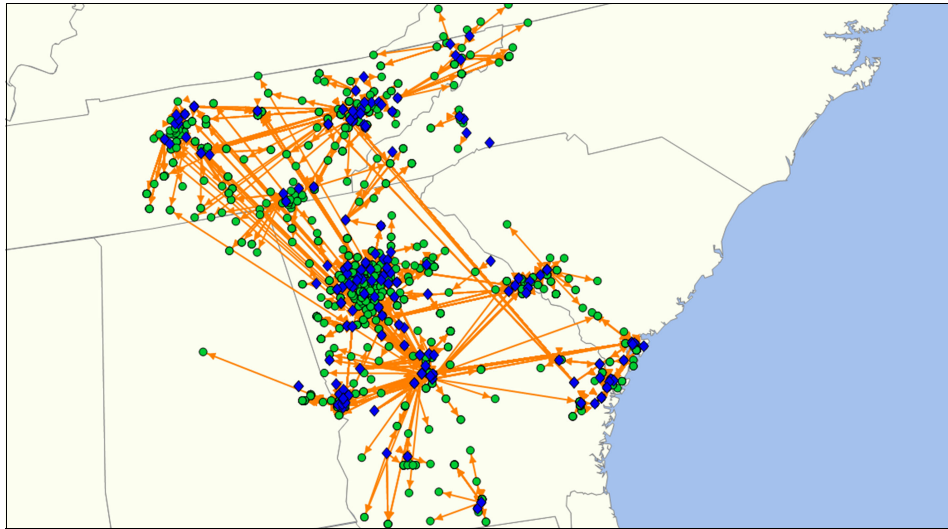


Figure 6-2 Item assignment by investigator and location

For more information about IBM ILOG JViews, see:

http://pic.dhe.ibm.com/infocenter/jviewent/v8r7/topic/com.ibm.ilog.jviews.maps.doc/Content/Visualization/Documentation/JViews/JViews_PPA/JViews_Enterprise/_pubskel/ps_jviews_maps1291.html

6.4 Optimization problem

To get a better understand of the optimization problem, look at the following criteria:

- ▶ Choices
- ▶ Targets
- ▶ Limits
- ▶ Data

6.4.1 Choices

The key decisions to be made in the case assignment optimization model are about how to assign work items to available investigators. Basically, answer the question *Which investigator will handle which work items?*

- Which new items are to be assigned to which investigator?

Each valid assignment of *investigator to item* is associated with a decision variable of type boolean that indicates whether the investigator is assigned to the item. In OPL, create an array of boolean decision variables that are indexed over the set of all valid investigator to item assignments:

```
(empItemPossSet):  
dvar boolean ItemAssignedToWorker[empItemPossSet];
```

Note that `empItemPossSet`, the set of all possible assignments of investigators to items, is generated during the data pre-processing phase. It is based on the input data and the constraints that make assignments valid. For more information, see “Input data” on page 198.

All other types of decisions can be derived from `ItemAssignedToWorker[empItemPossSet]`. Therefore, they do not have to be formally defined as decision variables in the OPL model. However, for the sake of simplifying the model formulation in general and its constraints in particular, add the following *helper* decision variables:

- Which area (postal code) does each investigator cover?

For each combination of investigator and postal code, associate a decision variable of type boolean that indicates whether the investigator works in that postal code. Create a two-dimensional array of decision variables that is indexed by the set of investigators and all postal codes where investigators need to work on items:

```
dvar boolean EmployeeWorksInZipCode[employeeIDs][itemZips];
```

- In how many areas (postal codes) is each investigator working?

For each investigator, associate the total number of postal codes for locations of work items that are assigned to that investigator¹.

```
dvar float+ numberOfZips[employeeIDs];
```

- How far is the investigator from a specific area (postal code)?

```
dvar float+ EmployeeZipDistance[employeeIDs][itemZips];
```

This distance depends on whether the investigator actually has work assigned in a certain postal code. For more information, see 6.4.3, “Limits” on page 196.

- What is the maximum distance that an investigator must travel to get to their work location?

```
dvar float+ maxDistanceToZipsForEmp;
```

¹ The notation `float+` indicates a continuous, non-negative decision variable.

- What is the total distance that each investigator must travel?

```
dvar float+ TotalDistanceToWorkedZips[employeeIDs];
```

6.4.2 Targets

Targets establish what you are we trying to achieve. This example has the target of minimizing the following items”

- The total distance that is driven by all investigators
- The largest distance that an investigator must travel to work on a case item
- The total number of postal codes that are covered by any one investigator

In the OPL model, these targets are represented as decision variable expressions. Based on the relative importance of the targets, a weighted sum of their corresponding decision expressions make up the OPL *objective function*.

The targets are represented in the OPL model in the following way:

1. Minimize the driving distance between items and investigators.

```
dexpr float sumDistanceToWorkedZips = sum (empID in employeeIDs )
TotalDistanceToWorkedZips[empID];
```

```
where TotalDistanceToWorkedZips[empID] == sum (zc in itemZips)
EmployeeZipDistance[empID][zc];
```

2. Minimize the largest distance from investigator to a work area (postal code).

```
dexpr float maxDistanceToZipsExpr = maxDistanceToZipsForEmp;
```

3. Minimize the maximum number of areas (postal codes) covered by an investigator.

```
dexpr float maxTotalNumberOfZipsForEmpExpr = max (empID in
employeeIDs) numberOfZips[empID];
```

The overall objective is to minimize all of these metrics. This objective translates to the following objective function in the OPL model as shown in Example 6-1.

Example 6-1 OPL objective function

```
minimize sumDistanceToWorkedZips +
maxDistanceToZipsExpr +
maxTotalNumberOfZipsForEmpExpr;
```

In most cases, including this one, the optimization problem has multiple targets that must be combined into one objective function. This combination is achieved by selecting a weight in accordance with the importance of the target. The decision expression is then multiplied by its respective weight. ODM Enterprise

implicitly multiplies each decision expression in Example 6-1 on page 195 by a weight.

Special care needs to be taken in the selection of the weights, especially when you are dealing with targets that have vastly different magnitudes. The selection of weights can be done only after the optimization model is fully developed and you have data sets to work with. Input from the client data business specialists is important in assigning correct weights to the objective function terms.

6.4.3 Limits

In any practical application, you always deal with limited budget or resources. It is important to identify such limits early in the process. During discussions with the client, several limits were identified that had to be taken into account when assigning items to investigators. In an optimization model, these limits are called *constraints* and are represented as mathematical equations or inequalities:

- Each case/item must be assigned to one and only one case worker.

```
forall ( caseItem in unassignedWork )
{
    sum ( <caseItem, empID, distance> in empItemPossSet )
    ItemAssignedToWorker[ <caseItem, empID, distance> ]== 1;
}
```

- Existing assignments must be taken into account.

```
if (considerWorkAlreadyAssigned == 1)
{
    forall ( aw in assignedWork )
    {
        EmployeeWorksInZipCode[aw.pernr][aw.zipcode] == 1;
        EmployeeZipDistance[aw.pernr][aw.zipcode] ==
empZipDis[aw.pernr][aw.zipcode];
    }
}
```

- If a work item is assigned to an investigator, the investigator must work in the corresponding postal code.

```
forall ( <uw,empID,distance> in empItemPossSet )
{
    EmployeeWorksInZipCode[empID][uw.zip] >=
ItemAssignedToWorker[< uw,empID,distance >];
}
```

- The number of postal codes that are worked is computed based on whether an investigator has work assigned in each postal code. It should never exceed the maximum specified number of worked postal codes.

```
forall ( empID in employeeIDs )
{
    numberOfZips[empID] == sum ( postal code in itemZips )
EmployeeWorksInZipCode[empID][postal code];
    numberOfZips[empID] <= parameters.maxNumOfZips;
}
```

- Distance from an investigator to a work location cannot exceed maximum distance.

```
forall (empID in employeeIDs, postal code in itemZips)
{
    EmployeeZipDistance[empID][postal code] -
maxDistanceToZipsForEmp <= 0;
}
```

- Distance from an investigator to a postal code is only considered if that investigator works in that postal code.

```
forall (empID in employeeIDs, zc in itemZips )
{
    EmployeeZipDistance[empID][zc] >= empZipDis[empID][zc] *
EmployeeWorksInZipCode[empID][zc] ;
}
```

One of the targets is to minimize the distance traveled (see item 1 on page 195). Therefore, `EmployeeZipDistance[empID][zc]` is assigned 0 if employee `empID` does not work in postal code `zc` and `empZipDis[empID][zc]` otherwise.

- For each investigator, the total distance to worked postal codes is the sum of distances to each worked postal code.

```
forall ( empID in employeeIDs )
{
    TotalDistanceToWorkedZips[empID] == sum (zc in itemZips)
EmployeeZipDistance[empID][zc];
}
```

A smidgen of OPL (part 4) - Filtering with tuples using formal parameters:

The previous expressions show an alternative way to construct such slices. The expression `<caseItem, empID, distance>` represents formal parameters that map to the fields of the tuples `empItemPossibilities` that compose the set `empItemPossSet`. By stating the fields explicitly, you can filter on `caseItem` in the set `unassignedWork`. The filter expression in the sum is thus equivalent to `sum ( item in empItemPossSet : item.uw == caseItem )`.

6.4.4 Data

Most OPL model data structures and data elements (parameter tuples, sets, and arrays) are generated automatically from the application data model. They are generated as part of the ODM Enterprise application development process.

The model pre-processes some of the input data so that it can be used most efficiently in decision expressions and constraints. One of the requirements is that cases can be handled only by investigators who have the required skill level and are eligible to handle that type of case. Instead of creating model constraints around these requirements, you can pre-process data to generate a set of valid `<investigator, work item>` combinations. These are combinations in which the investigator's skills match those required by given work item. The model constraints are then applied to these valid combinations only, which reduces the size of the optimization model.

Some post-processing is also implemented to save the solution information in a format that can be used to populate ODM output views.

Input data

Start by processing some of the parameters to determine:

- ▶ Maximum number of postal codes to be covered by an investigator
`int maxNumWorkedZipsPerEmp = parameters.maxNumOfZips;`
- ▶ Whether there are existing work assignments that should be taken into account
`int considerWorkAlreadyAssigned = parameters.useAssignedZips;`

Compute employee IDs and case types available as shown in Example 6-2.

Example 6-2 Compute employee IDs and case types

```
{string} employeeIDs = { emp.pernr | emp in employeeInformation };
{string} caseTypes = { uw.caseType | uw in unassignedWork };
```

A smidgen of OPL (part 5) - Set Operations:

OPL has a rich set of operators that enable developers to construct sets in various ways. Because sets are used widely to index decision variables and constraints, precise sets reduce the size and complexity of the optimization model. This configuration takes advantage of sparsity to reduce computation. Example 6-2 on page 199 illustrates some of these possibilities. These two expressions are equivalent to the SQL statements:

```
SELECT pernr FROM employeeInformation
SELECT caseType FROM unassignedWork
```

Example 6-3 through Example 6-7 on page 200 show more elaborate examples.

Work item postal codes are computed from both assigned and unassigned work items as shown in Example 6-3.

Example 6-3 Compute work items

```
{int} itemZips = { uw.zip | uw in unassignedWork } union { aw.zipcode
| aw in assignedWork };
```

Compute the set of investigators who can work on a given case type, based on the skill required (for example, case type “SSBI” requires that the employee possess the corresponding skill (empSkill.ssbi == “TRUE”):

```
{string} possibleWorkerIDs[ ct in caseTypes] =
    (ct=="SSBI") ? employeeIDs inter { empSkill.pernr |
empSkill in employeeSkills : empSkill.ssbi == "TRUE" } :
    ((ct=="NACLC") ? ... :
    ...
    ... :
    ... {}))...);
```

For each investigator, compute the set of organizations where they can work:

```
{string} orgsEmployeeCanWork[empID in employeeIDs] = { empInfo.org |
empInfo in employeeInformation : empInfo.pernr == empID };
```

Initialize the set of investigators who can do priority, record, and personal work, based on the employee information provided as shown in Example 6-4.

Example 6-4 Initializing the set of investigators

```
{string} employeesWhoCanDoPriorityWork = { emp.pernr | emp in  
employeeInformation : emp.pl == "TRUE" } ;  
{string} employeesWhoCanDoRecord = { emp.pernr | emp in  
employeeInformation : emp.record == "TRUE" };  
{string} employeesWhoCanDoPersonal = {emp.pernr | emp in  
employeeInformation : emp.personal == "TRUE" };
```

Determine the number of hours that are required to complete each work item, which is based on case type, item type, and coverage method, as shown in Example 6-5.

Example 6-5 Number of hours to complete each work item

```
float hoursRequired[unassignedWork] = [uw : itat.time | uw in  
unassignedWork, itat in itemTypeAndTime : uw.caseType ==  
itat.caseTypeName && uw.itemType == itat.itemType && uw.method ==  
itat.coverageMethod ];
```

Cases must be assigned to case investigators who can handle the case type and who can work in the organization that is associated with the case. They must be able to handle priority work if the caseService is “PRT”, “record” work items if the case is of type “record”, and “personal” work items if the case is “personal”, as shown in Example 6-6.

Example 6-6 Case assignments

```
tuple empItemPossibilities  
{  
    TUnassignedWork uw;  
    string employeeID;  
    float distance;  
}
```

Determine valid <case item, employee ID, distance> combinations as shown in Example 6-7.

Example 6-7 Determining valid <case item, employee ID, distance> combinations

```
{empItemPossibilities} empItemPossSet =  
{
```



```

    <caseItem,empID,dis.distance> | caseItem in unassignedWork, empID in employeeIDs,
dis in distances :

        dis.pernr == empID
        && dis.caseId == caseItem.caseId
        && dis.itemNumber == caseItem.itemNumber
        && empID in possibleWorkerIDs[caseItem.caseType]
        && caseItem.org in orgsEmployeeCanWork[empID]
        && (( caseItem.caseService != "PRT" )
|| (caseItem.caseService == "PRT" && empID in employeesWhoCanDoPriorityWork ) )
        && (( caseItem.method != "R" )
|| (caseItem.method == "R" && empID in employeesWhoCanDoRecord ) )
        && (( caseItem.method != "P" )
|| (caseItem.method == "P" && empID in employeesWhoCanDoPersonal ) )
};

```

Compute distance from each investigator to a known item postal code, as shown in Example 6-8.

Example 6-8 Distance from each investigator to a known item postal code

```

float empZipDis[employeeIDs,itemZips] = [ empID: [zc: eip.distance ] |
empID in employeeIDs,zc in itemZips,eip in empItemPossSet : zc ==
eip.uw.zip && empID == eip.employeeID ];

```

Output data

After the mathematical model is implemented, ensure that data that are needed by the output views are available. To implement these models, use the solution values to populate the necessary output data structures.

First, declare an array that stores the number of investigators who work in a certain number of postal codes:

```
int NumberEmpsWithThisManyZips[0..maxNumWorkedZipsPerEmp];
```

Then, implement a post-processing script block to populate all the required data structures, as shown in Example 6-9.

Example 6-9 Post-processing script block

```

execute OUTPUT {
// store the number of postal codes covered by each investigator
// (employee ID, number of postal codes):
    for ( var empID in employeeIDs ) {
        employeeZipInformation.add(empID,0pl.round(numberOfZips[empID]));
    }
// store work assignments

```

```

// (case ID, work item number, employee ID, distance to work location)
for (eip in empItemPossSet) {
    if ( ItemAssignedToWorker[eip] == 1 ) {
        workAssignments.add(eip.uw.caseId, eip.uw.itemNumber,
eip.employeeID, eip.distance);
    }
}
// count the number of investigators who cover a given number
// of postal codes (total number of postal codes, number of
investigators)
for (var i = 0; i <= maxNumWorkedZipsPerEmp; i++ ) {
    var number = 0;
    for ( var empID2 in employeeIDs ) {
        if (Op1.round(numberOfZips[empID2]) == i) {
            number = number + 1;
        }
    }
    numEmpWithThisManyZips.add(i,number);
}
};

```

6.5 Solution development

The suggested development process for creating an optimization application by using IBM ILOG ODM Enterprise is described in the product documentation. It is a series of high-level steps:

1. Creating the Application Data Model
2. Adding ODM views to the application
3. Importing data to the Application Data Model
4. Creating the optimization model
5. Defining goals and requirements
6. Deploying and testing the application

This section follows this methodology in describing how the solution was developed and deployed.

A careful examination confirmed that ODM Enterprise was a good fit because of its rich client environment and its ability to provide an optimal solution to a data-intensive, complex assignment problem. After the two-week assessment period, the IBM technical team proposed a multitier solution that featured the following components.

- ▶ ODM Enterprise Data Server for handling all data operations and communications between the staging and the Scenario databases.
- ▶ ODM Enterprise Optimization Server for processing optimization jobs.
- ▶ ODM Enterprise Studio as the rich application client.
- ▶ A staging database for storing transformed, consolidated, and de-normalized data. In most cases, optimization performance can be greatly improved by modeling the data according to the optimization model specifications. These specifications often require de-normalized data, with codes instead of long text descriptions. For example, it is better to provide the Optimization engine with just a product code, such as “AS2234”, than the product name itself, “17 inch 155/70 Tire”.

For more information about the use of the staging databases, see 3.5, “Data architecture” on page 99.

6.5.1 Architecture

The proposed solution features a classic ODM Enterprise application architecture that enables multiple Clients to collaborate and share Scenario data. Figure 6-3 on page 204 shows the architecture overview diagram.

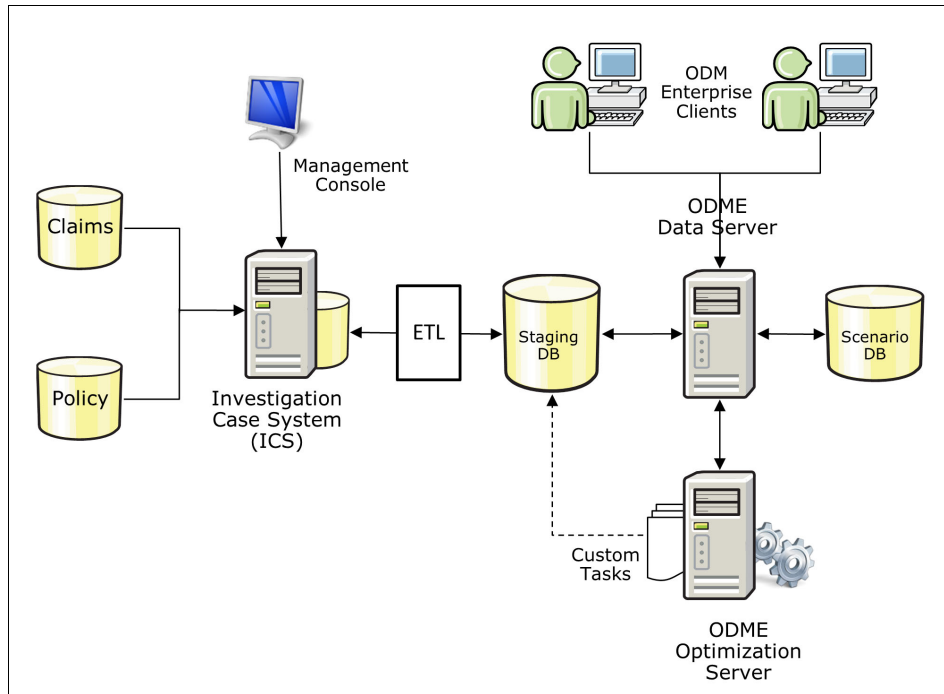


Figure 6-3 Solution architecture

The best way to understand the system architecture is by examining the different data flows that are associated with different operations.

Clarification: A staging database is used to store the data that is required by two components of the solution: The optimization engine and the user interface (views and reports).

1. The first flow is concerned with *data pre-processing*. Here, custom SQL processes prepare and transform the data during the Extract-Transform-Load (ETL) stage. These processes de-normalize and aggregate the operational data model from ICS into a concise data set. This concise set is more suitable for the CPLEX optimization engine as determined during the optimization model development. For more information about optimization models, see 3.1, “Key ODM Enterprise concepts” on page 56.
2. ODM Enterprise users are now able to create a *Scenario*. A Scenario is essentially an *import process* that brings data, goals, constraints, and other data elements from the staging into the Scenario database. Users run a *solve* (also called an *optimization process*) that uses the Scenarios.

3. When a solve is initiated, a request is sent to the Management module (running in the Data server), placing the request in the Job queue. The Optimization server then detects that there is a job that is waiting in the queue and instantiates a solve process. The results of the solve are then written back to the Scenario database.
4. Custom export processes make Scenario data available to operational systems by transporting the optimization results back to a purpose-built schema in the staging database. For best performance results, export routines are built as *Custom Tasks*. These tasks run from the Optimization server, taking advantage of faster processing times and direct database connections.

Several factors influenced this solution architecture, especially the non-functional requirements that relate to performance. Performance was important because the client has a short window of time to process the data and make the results available for deployment.

IBM WebSphere Application Server is at the bottom of the application software stack, hosting the optimization engine along with its models and application code (Figure 6-4). On the client side, JViews-based maps and reports provide a rich user interface experience.

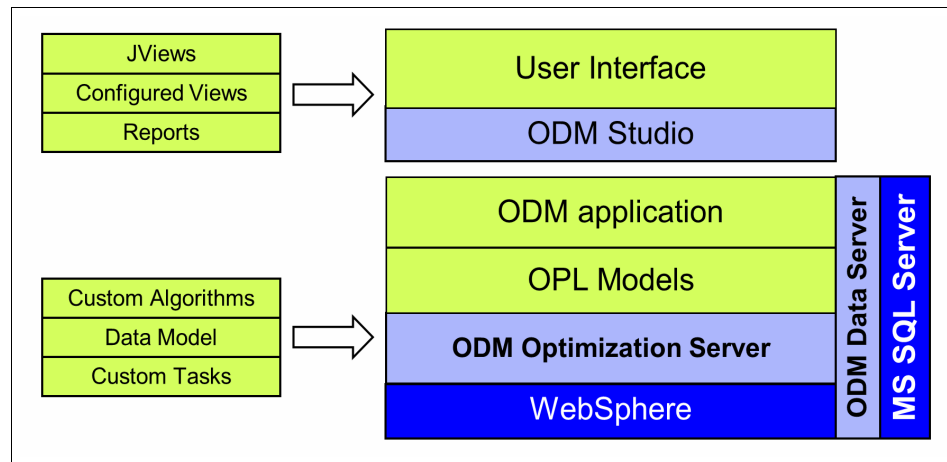


Figure 6-4 Application solution software stack

6.5.2 Data requirements

When they are developing ODM Enterprise solutions, IT architects and developers use a data-centric design approach. This approach enables them to start collecting data and quickly develop prototypes of the final application, while

work on the optimization model is performed simultaneously by OR experts. Development of the application does not depend on having a finished OPL model.

ODM Enterprise puts the *Application Data Model* at the center of the development process of an ODM Enterprise application. To define the Application Data Model, the developer edits the data model file that is accessed through the `datamodel.odmadm` of the ODM Enterprise project. This file is automatically created when you create the ODM Enterprise project. After being edited or created by using the Data Diagram Editor, the model contains the definition of the structure of a Scenario. This definition is a set of tables that contain columns, primary keys, and foreign keys. ODM Studio Views are then built on top of the tables that are described in this model.

Largely driven by the optimization model requirements, data requirements are, in most cases, extrapolated from the solution requirements and preliminary optimization model design when data elements and optimization parameters are identified. The data model is constantly updated during the development phase so it can be adjusted to fit optimization needs.

When you are defining the required data model, ask the stakeholders the following questions:

- ▶ What processes do you want to improve?
- ▶ What are the main entities of those business processes?
- ▶ What is the typical volume of data that is needed to run those business processes?

According to the insurance company's requirements, four main entities participate in distributing the case load among the investigators:

Cases	A list of investigation cases and their items (assigned and unassigned).
Resources	A directory of investigators and their profile information.
Skills	The skills that are required from the investigator and for a particular case. An investigator must possess a specific skill that is required to investigate that type of case.
Location	The geographical location of both case items and resources.

When you are working with ODM Enterprise, define at least two distinct data schemas: One to store the input data and the other to store the output (or Solution).

Input

The diagram in Figure 6-5 depicts the *input schema* that is used to store the data that is required by the optimization model.

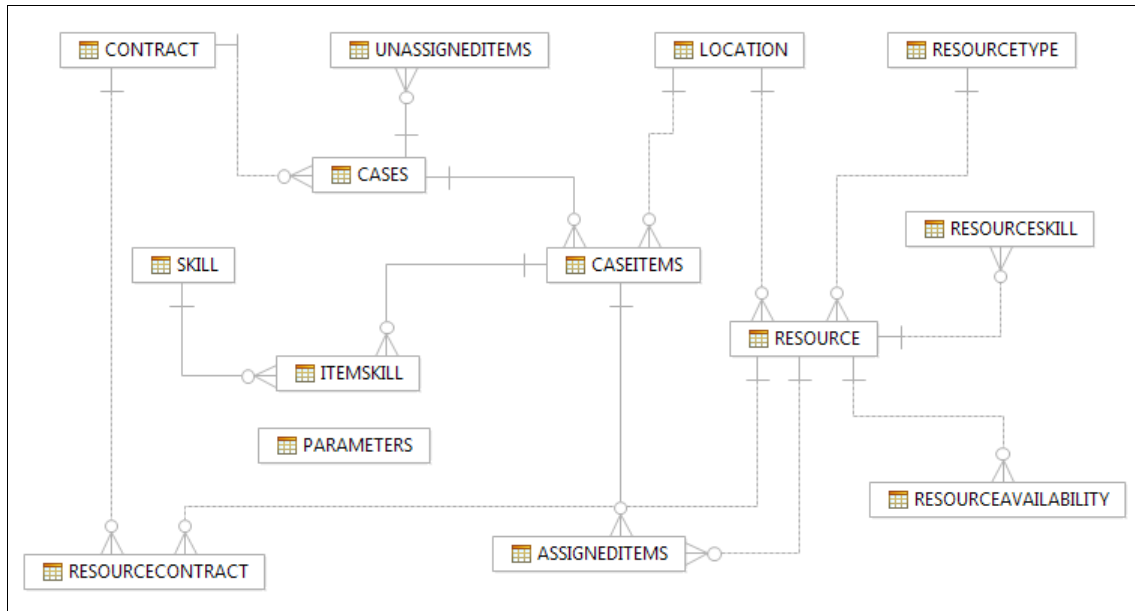


Figure 6-5 Input data model

This model is populated through the daily ETL processes created by the IT team. For simplicity, the data is wiped clean and reloaded during the nightly process.

The model includes the following tables:

CASES	A comprehensive list of all pending and completed cases.
CASEITEMS	Associative entity to link items to cases.
UNASSIGNEDITEMS	A list of all items that are not assigned to investigators.
ASSIGNEDITEMS	The list of currently assigned items.
RESOURCE	Investigator information.
RESOURCESKILL	A list of all skills by resource.
SKILL	A lookup table that contains the different skills and skill levels.
ITEMSKILL	List of all skills that are required to work on a case item.

RESOURCEAVAILABILITY	A daily calendar that lists the availability of each resource.
RESOURCETYPE	Lookup table that contains the different resource types such as “full-time” or “part-time”.
RESOURCECONTRACT	Cross-reference between resources and the contracts that they are allowed to work with.
LOCATION	A combined list of addresses and other geographic information.
PARAMETERS	A list of value pairs that are used at run time to set certain optimization preferences and optimization thresholds.

Output

The *output* or *solution model* defines the tables that stores the output of the Optimization engine. It effectively produces a comprehensive list of work assignments for each investigator.

Some entities are in both the input and output models which is quite a common occurrence, mainly for maintaining referential integrity.

Figure 6-6 shows the output model.

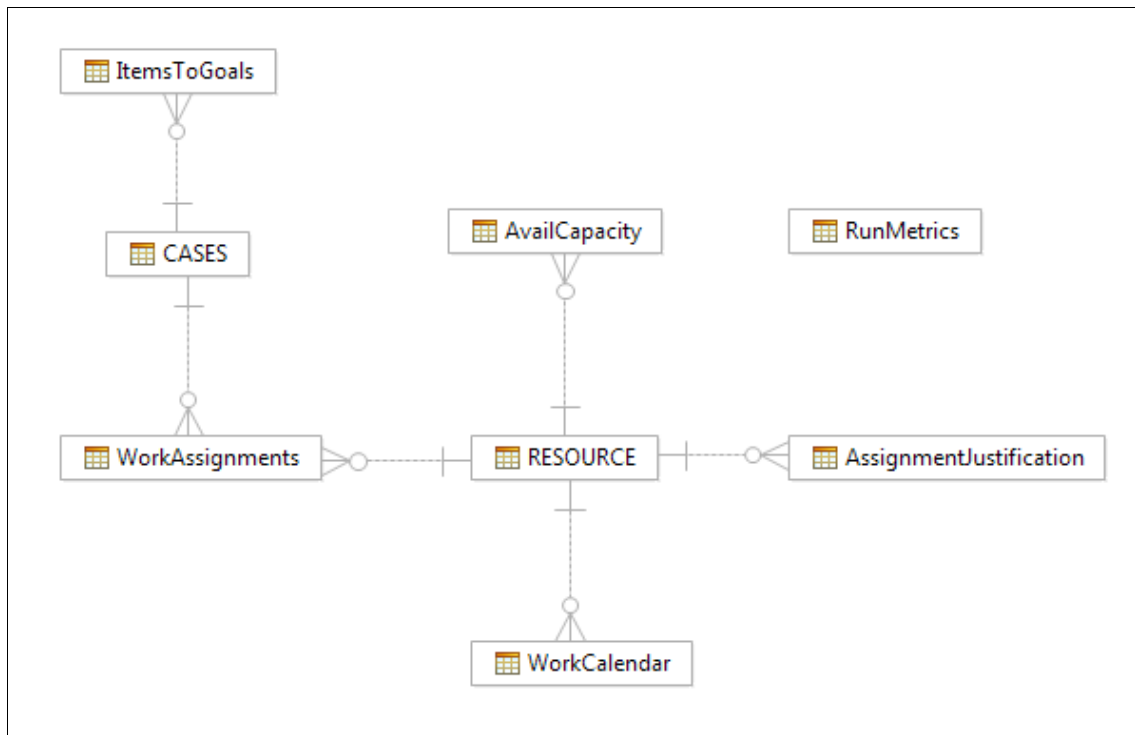


Figure 6-6 Output data model

The model includes the following main output tables:

WorkAssignments	The main output of the optimization process, this table stores the list of work assignments by investigator.
AssignmentJustification	Provides an explanation of why each item was assigned to an investigator by the model. It shows how each item met the relevant business requirement (a yes or no value), or a numeric value as dictated by the associated constraint.
AvailableCapacity	Stores the available capacity by investigator. You cannot project how or when the investigator will actually execute the assigned items throughout the week. Therefore, it is more meaningful to show the total available capacity in weekly time buckets.
ItemstoGoals	Shows the contribution of each case item to each of the Goals.

WorkCalendar	Calendar-like view of each investigator’s daily work load, expressed in hours.
RunMetrics	A consolidated view of the metrics that are used by the optimization process. Useful for comparing different optimization results while you are performing what-if analysis.

Connecting it all

Using this information, create an ODM Datasource that can be used to load Scenarios and display Views. Before you can define Views, define the database connection that allows the ODM Enterprise servers to locate the data sets and retrieve data. Define the connection in the Datasources tab of the Deployment Configuration Editor. Figure 6-7 shows how to create a typical data connection in the Deployment description (.odmds file).

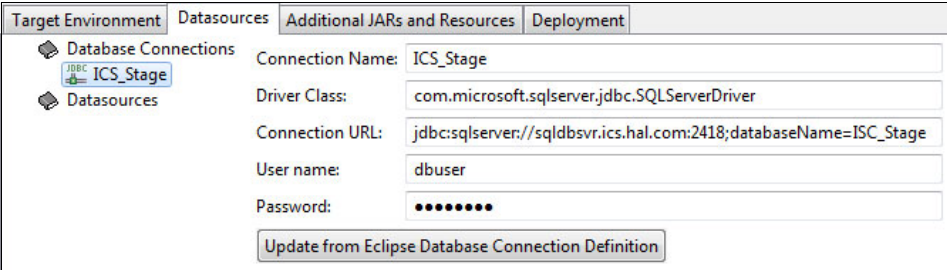


Figure 6-7 Defining a database connection

Remember: If a connection with the same name was already created (and tested) in the Data project, enter the name of the connection and click **Update from Eclipse Database Connection Definition**. This process copies the settings, avoiding a double entry.

Now create a data source by using the connection that you created previously. Expand the datasources folder to display the new data source form. Give the data source a meaningful name and a brief description. After you select a database connection, select the required tables from a list by clicking **Add** as depicted in Figure 6-8.

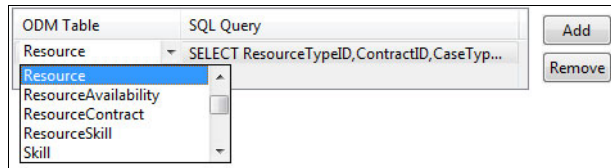


Figure 6-8 Adding tables to the data source

Select the update options that are most suitable to your application. In this case, the data source is used to *create Scenarios* and to *refresh data in a previously created Scenario*.

This feature allows for virtually infinite combinations of datasources and connections. This advantage can be particularly useful during the application development process, where you can have several databases with different structures and content. For example, to change the source from which Scenarios are being created, switch the Database Connection setting on the Default data source and redeploy the application.

Figure 6-9 shows the different attributes and parameters of an ODM Enterprise data source definition.

Datasource name: ICS_DevDb

Description: Development datasource for creating Scenarios

☒ This datasource can be used to create a new modified scenario.

☒ This datasource can be used to replace data in a scenario.

Choose Database Connection: ICS_Stage

Updated Tables

Columns to fill for table AssignedItems: CaseID, ItemId, ResourceId, Ge

ODM Table	SQL Query
AssignedItems	SELECT CaseID,Item
CaseItems	SELECT CaseID,Item
Cases	SELECT CaseId,Case
Contract	SELECT ContractID,I
ItemSkill	SELECT CaseID,Item
Location	SELECT LocationId,
ResourceAvailability	SELECT Day,DayTXT
ResourceSkill	SELECT ResourceId,
Skill	SELECT SkillId,Descr
UnassignedItems	SELECT ItemID,Case
ResourceContract	SELECT ContractID,f
Resource	SELECT ResourceId,I
ResourceType	SELECT ResourceTyp
Parameters	SELECT ParameterN

Add

Remove

Figure 6-9 Data source definition

Views

ODM Enterprise allows you to create and display many different types of data in different types of visualization. These types range from simple tabular views, to array views, pivot tables, charts, and many more. For this project, only simple views are created to illustrate the process.

A full list of available Views can be found in the ODM Enterprise InfoCenter at:

http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/topic/ilog.odms.ide.odm.enterprise.help/ODME/Userman/topics/Customizing/DV_IDE.html

Views can be created by opening the `views.odmvw` file, which displays the Views Editor. Make sure that you are working with the ODM Enterprise perspective. Figure 6-10 shows the Views editor displaying the cases view definition, which is just a simple, straightforward representation of the cases table.

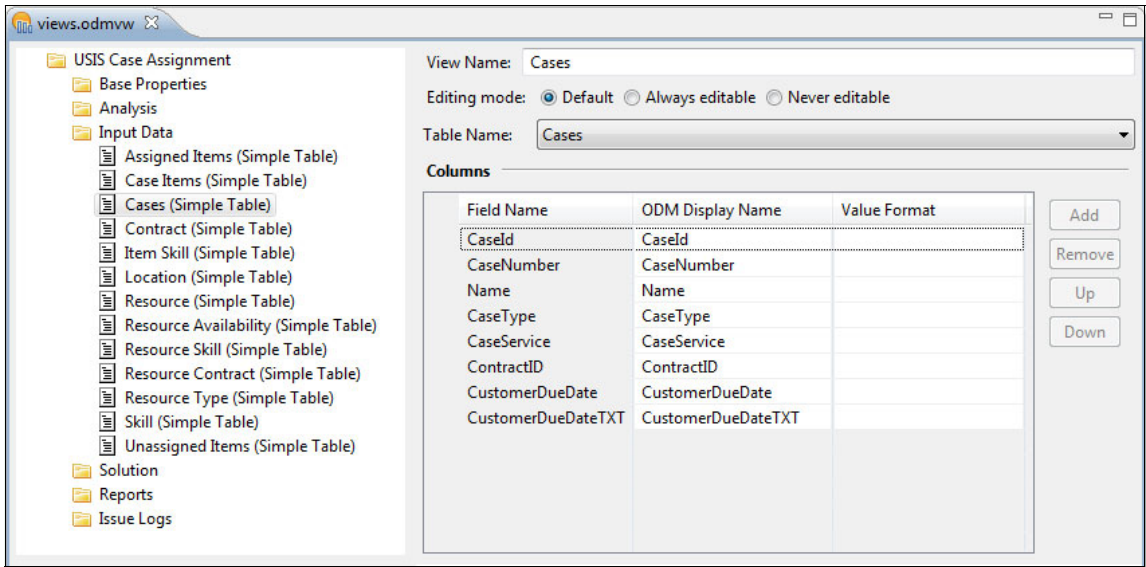


Figure 6-10 Views Editor

The Table Name list contains all the tables included in the default data source definition. If you cannot find a specific table, most likely it was not included in the data source. Simple data formats (such as dates and decimal numbers) can be added to the View definition and applied at run time. More complex data operations can be performed either in the data source SQL definition or through Custom Views. Custom Views are built in Java and hooked into the deployment definition.

Tip: Custom datasources can also be created in Java for an additional level of control for creating views. Using specific, predefined interfaces, the custom view can interact with the custom data source to retrieve, display, and edit data from the ODM Enterprise client.

Figure 6-11 shows the Cases view displayed in the ODM Enterprise client.

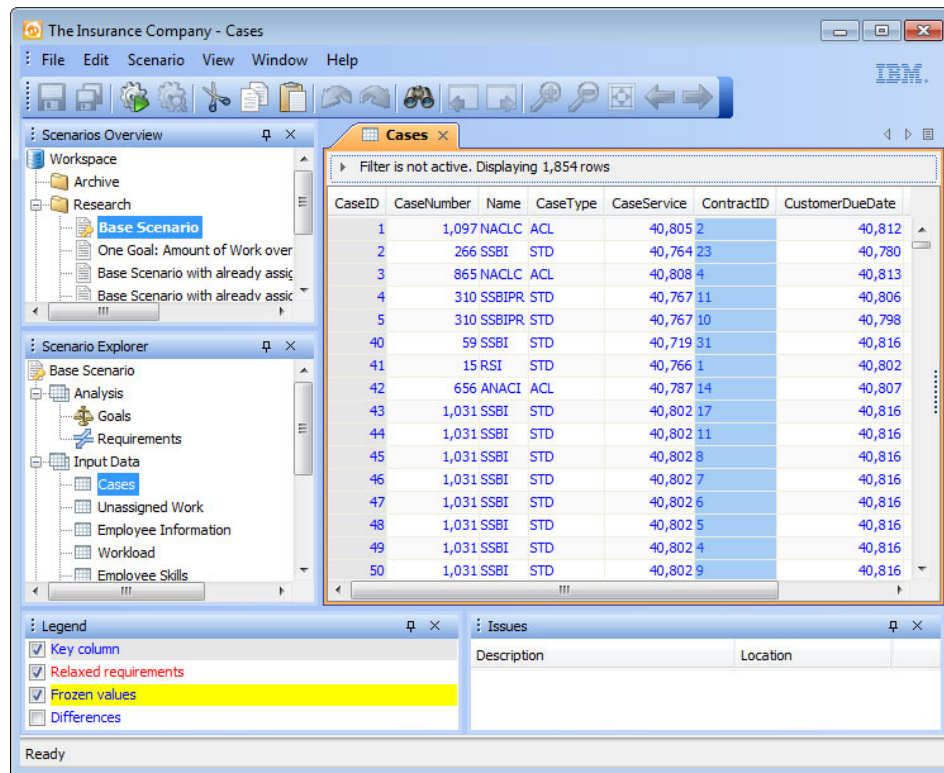


Figure 6-11 Cases being viewed in the ODM Enterprise Client

After you assemble all the Views, deploy the application to the development servers. Before you proceeding to this final step, complete the following checklist to ensure that the deployment goes smoothly and error free. Enter and review the following information in the Deployment editor.

1. Target Environment

- Confirm that the deployment application server is running and that you have the correct deployment URL.
- Confirm that you want to use the Data Server for Scenario management and the Optimization server for solving.
- Double-check the Scenario Repository database configuration (server address, port, and so on) information, ensuring that the instance name and schema name are correct.

Remember: When you click **Create Schema**, ODM Enterprise creates the Scenario database, its schema, and users along with all the required permissions and object ownerships. However, you can also build the repository directly at the DBMS level.

2. Datasources

- Double-check the database connections and datasources, making sure that the data source marked as “default” is pointing to the correct database connection.
- Special attention needs to be paid to the two check boxes that indicate how the data source will be used to create and refresh Scenario data (see Figure 6-9 on page 212).

3. Additional JARs and resources

This section is used to add references to all files used to compose the application and other dependencies:

- JAR files that contain custom views, datasources, data checkers, and data exporters.
- JDBC drivers that are referenced by ODM Enterprise DBMS datasources.
- Additional data that are used by custom code or OPL datasources, such as spreadsheet or database files.
- Map files (including shape files) or JViews Maps.
- Data files such as Microsoft Excel spreadsheets and CSV files.
- OPL scripting libraries (*.js) for external Java calls used in your models.
- Additional model libraries included in your models.
- Additional OPL data files that are used in your main model (*.dat) through scripting (typically in a multi-model solve).

Resource order: If the order of the resources is critical, you can use **Move Up** and **Move Down** to position them as wanted. The order is critical if JAR files or other resources must be accessed in a certain order in the application's classpath. For example, you might have a third-party library (such as a graphical library) that includes an XML parser, but you want other parts of the code use a more recent XML parser. In this case, put the more recent XML parser JAR file before the graphical library in the classpath so that the whole application uses the more recent parser.

4. Deployment

The Deployment tab is the final step in a ODM Enterprise deployment process. It is here that you define the following items:

- The folder where the .odmapp file will be created. This file is the launchpoint for the ODM Enterprise Client.
- The name of the .odmapp file (for example, MyODMEApp.odmapp).

When all the settings are confirmed, click **Deploy** to initiate the deployment process. A progress dialog will keep you informed of what is being deployed and where. If the application was previously deployed, the **Undeploy** button is enabled, giving you the option to undeploy the application before you redeploy it. However, this is not necessary because ODM Enterprise automatically undeploys the old application and deploys the new files during the process. Figure 6-12 shows the Deployment progress information dialog.

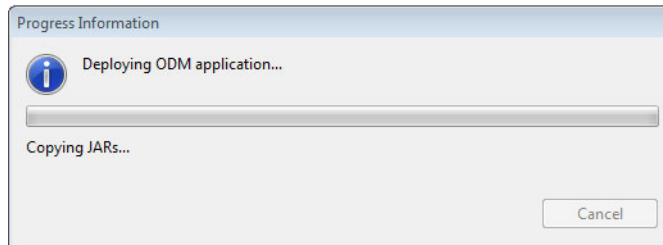


Figure 6-12 Progress information dialog

After a successful deployment, the application can be started by using the following methods:

- ▶ Opening the ODM Client Studio from the Windows Start/Programs menu.
- ▶ Double-clicking the newly generated .odmapp file.

Tip: If you select the “Open the destination directory in Windows Explorer after deployment” check box, ODM Enterprise will automatically open the Windows Explorer after the deployment. This process allows you to test starting the application without having to search for it.

Figure 6-13 shows the ODM Client connected to the application servers and ready for use.

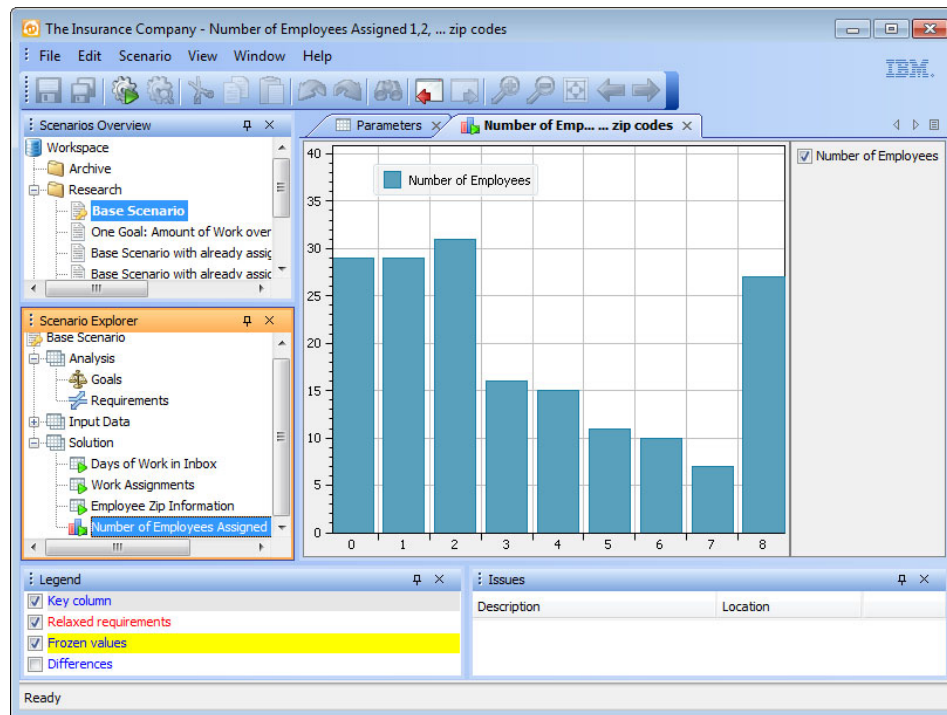


Figure 6-13 The ODM Client ready for use

6.6 Optimization

Now that the application is deployed and ready for use, you can start researching different case assignment tactics by using the ODM Enterprise *what-if capability*. To do that, you must first create a *baseline Scenario* to be able to compare with other outcomes. Create a Default Scenario by clicking **File** → **New** → **New Default Scenario**. ODM Enterprise imports the data from the staging database and stores it in the Scenario database.

After the Scenario is created, rename it “Base Scenario” and start the solve process. Figure 6-14 shows the Solve Progress dialog during the Goal Optimization phase.

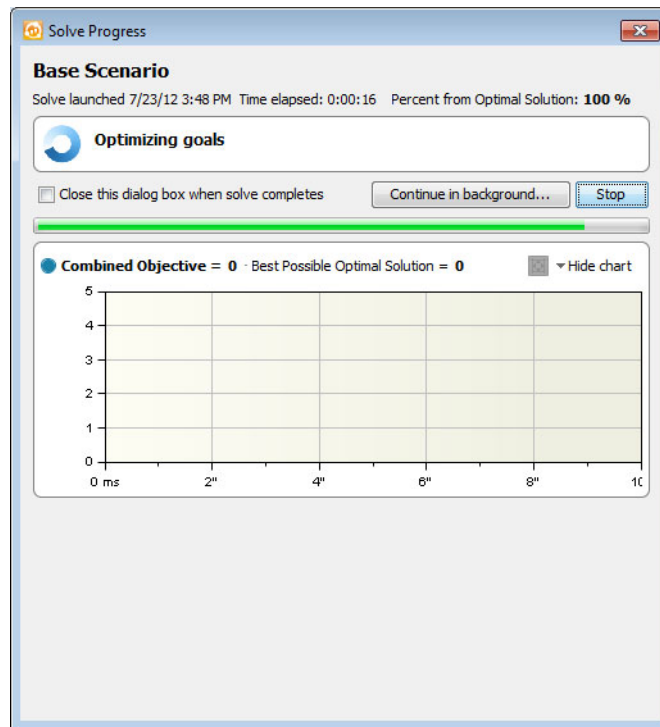


Figure 6-14 Solve status dialog

After a solution is found, you can browse the output tables to examine the results of the optimization process. Take some time to browse two of the main views: Work Assignments and Days of Work in Inbox.

Now that you have created a baseline, you can start researching different outcomes by changing business parameters, re-solve, and compare the results. To preserve the baseline, make a copy of the Baseline Scenario by right-clicking the scenario name and selecting **Duplicate Current Scenario**.

Open the Parameters table (under the Input folder) and double the “Maximum number of postal codes per Employee” parameter from 8 to 16. Figure 6-15 shows the Parameters table with the modified value. This change enables the optimization model to assign more work per investigator.

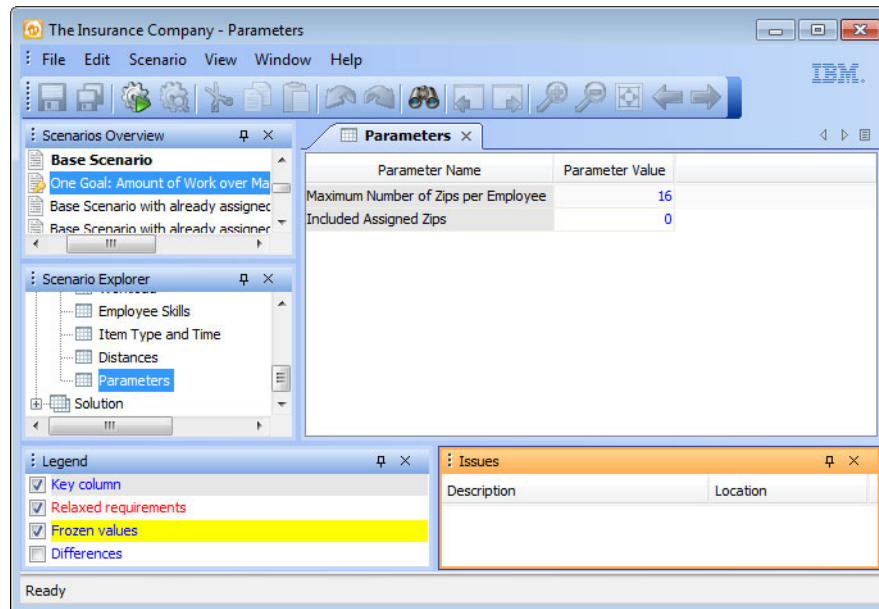


Figure 6-15 New parameter value

After you save the new scenario, click Solve to start the optimization process and solve the new scenario. When a Solution is found, you can visualize the differences.

To visualize the differences between two scenarios (and their outcomes), right-click the Base Scenario and select **Use as reference**. The ODM Enterprise Client automatically highlights the Baseline Scenario. Then, select **Differences** under the Legend section (lower left of the window) to enable the function.

Now, every time you open a View from any Scenario other than the one used *as reference*, ODM Enterprise displays the differences between the two. It does so by highlighting rows with a plus (+) sign in the first column, indicating an extra row or by adding the baseline value inside parenthesis, next to the current value. An extra row indicates that the optimization process was able to assign extra work because of the increase in the postal code assignment allowance.

The results are shown in Figure 6-16.

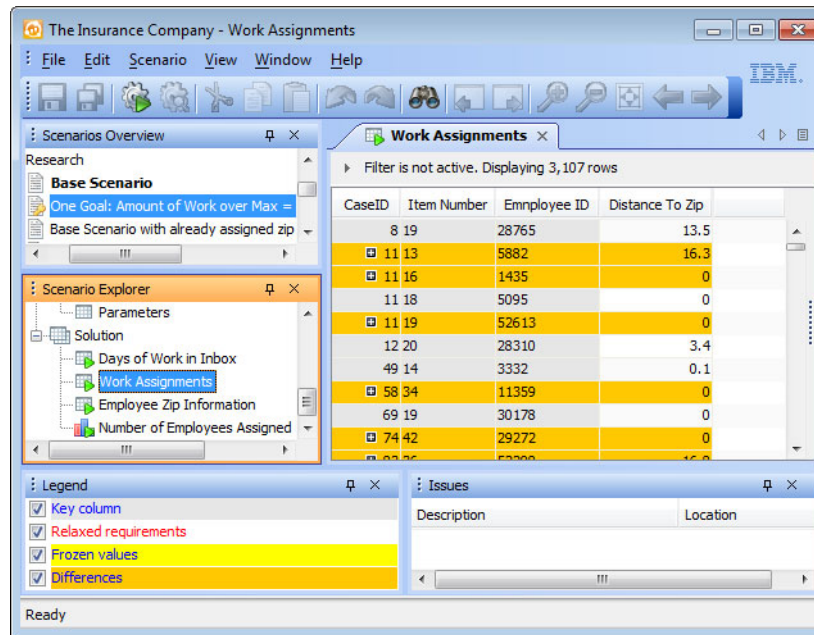


Figure 6-16 Viewing the differences between Scenarios

6.7 Conclusion

The financial effect for the insurance company business was immediate. The company experienced a decrease in operating costs caused by the optimized deployment of resources, which reduced redundancy, excessive travel and expense costs, and time to resolution. Implementing ODM Enterprise provided the following main benefits:

- Reduced costs

Investigators are reimbursed for travel and other expenses. The ability to efficiently assign work to the investigator closest to the case item and to where other case items that are already assigned to the investigator is a major cost cutter.

- Increased agility

The ability to plan the workload distribution weeks ahead of due date allowed the insurance company to maintain a profitable backlog while prioritizing work among investigators.

- ▶ Planning

Using ODM Enterprise's what-if analysis capabilities, workload managers are now able to research and forecast the business effect by manipulating distribution variables such as vacation days and pay rates.

In conclusion, optimization solutions are about:

- ▶ *Decision support*, providing decision makers the ability and the tools to recommend and explain business decisions and strategic planning while providing alternatives and trade-offs.
- ▶ *Making good decisions*, providing the business expert the ability to look at the effect of the proposed changes in a business sensitivity analysis.
- ▶ *People and processes*, allowing the enterprise to harvest and align the collective expertise of the business decision makers with the technology to help them make better decisions.



Case study: A manufacturing company

This chapter provides a case study about a fictional *manufacturing company*. It describes how to optimize their global production network and supply chain by focusing on these details.

This chapter includes the following sections:

- ▶ Business context
- ▶ Business problem
- ▶ Business requirements
- ▶ Optimization problem
- ▶ Solution
- ▶ Conclusion

7.1 Business context

The manufacturing company is a *first-tier supplier* in the automotive industry. They produce automotive batteries that are used by the final car manufacturers, or *Original Equipment Manufacturers* (OEM). Assume that the company is a major player in the market and is more than 50 years old. It generates yearly revenue of about 10 billion dollars, and employs more than 50,000 people around the world.

This project focuses on the company's major product type, which is responsible for nearly half of the overall revenue and income. It is produced in a global production network, with 20 plants around the globe. This type of product is produced and marketed in almost 20,000 different variations, or *products*.

A product is uniquely identified by a *product ID* or *identifier*. Internally, such an ID is represented by a long, 15-digit, number. Externally, each product has a unique name. Thus, a product is a theoretical concept, not the real instance. For example, the manufacturing company might produce 1000 items of a specific product per year. When storing or selling a product, the same entity is sometimes called a *Stock Keeping Unit*, or SKU.

The manufacturing process is complex:

- ▶ Many different materials are needed, up to 100 for one final product.
- ▶ Some materials are received from other suppliers. These materials must be bought and delivered.
- ▶ Other, intermediate, products are produced by the company itself. Some of these products are produced in the same plant as the final product, and some in other plants.
- ▶ Some intermediate products can be stored, whereas others must be processed immediately.
- ▶ Expensive resources are needed to make the product, including machines, tools, people, and so on.
- ▶ The final products can be transported easily, but transportation is expensive.
- ▶ The production takes place in a distributed way by using the organization's global supply chain.

On the sales side, the market situation is also complex:

- ▶ There are many different markets worldwide, both to OEM and to customers (car owners).
- ▶ The product portfolio is quite diverse. It addresses many different OEM and car owner's needs, which can change rapidly.

- It faces all current globalization challenges, like heavy cost pressure, and regional market specialties.

The company has these overall business goals:

- *Increase sales* through optimal assignment of production to demands.
- *Reduce costs* through, for example, production in the correct locations.
- *Balance trade-offs* like supply versus demand trade-offs.

7.2 Business problem

The manufacturing company faces difficult challenges in their production and supply chain. There are challenges both upstream on the production or sourcing side, and downstream on the market or delivery side.

The company manages this supply chain through a complex planning process. Figure 7-1 shows the major parts in this process.

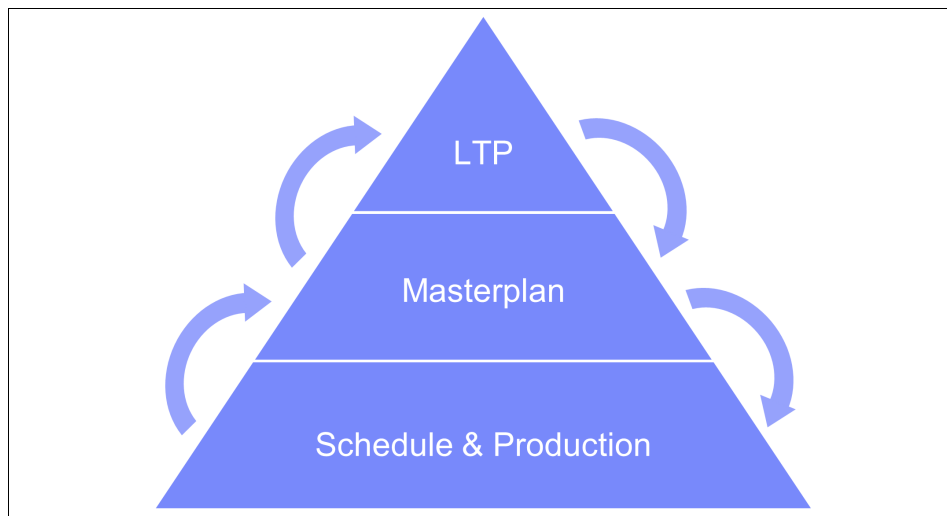


Figure 7-1 Production planning process at the manufacturing company

Table 7-1 further describes these planning steps.

Table 7-1 The planning steps in detail

Planning Step	Content and Result	Frequency
<i>Long Term Planning (LTP)</i>	The LTP looks 1-5 years ahead, on both the worldwide network, and individual plants. This is an iterative process, where many planning scenarios are created, evaluated, and compared until the final decisions are made.	Yearly
<i>Masterplan</i>	The masterplan looks 12 months ahead, still considering the whole network, but on a much more detailed level than the LTP. This step needs much more time to evaluate potential scenarios and so this step is much less iterative than LTP.	Monthly
<i>Schedule and Production Planning</i>	This step is about short-term planning and scheduling. In this step, the masterplan is taken and transformed into an execution sequence and schedule. This step does not yet control the production execution on the shop floor, which is typically done by yet another system, called the <i>Manufacturing Execution System (MES)</i>.	Weekly or daily

These steps are connected to each other. The output of LTP is input for the masterplan, and the latter is taken into account when creating the execution scheduling. Also, execution data, such as built-up inventory and unplanned production disruptions, are fed back into the masterplan. The masterplan is then eventually reconsidered when doing the next LTP. This propagation from the bottom (scheduling) all the way up to LTP, is not triggered immediately. Rather, it follows the usual planning cycles.

On each level certain decisions are made that are then fixed and not changed by other planning steps (on other levels). For example, in LTP the following decisions are made, among others:

- Which product to produce in which plant (binary decision: yes or no). If a certain product must be produced in a certain plant, that plant must have certain machines, tools, or third-party material available.
- Which resources to extend in which plant. This decision might require buying a new machine for a plant, or moving tools from one plant to another.

These decisions are taken on the LTP level and are not adjusted on the other levels. However, there are decisions that are taken on these levels, but might be

overruled on other levels. For example, in LTP it is also determined, *how many* of each product are produced at each plant (if at all). But this number largely depends upon forecasted figures, and these figures can (and usually will) change when the plan is executed. In these cases, the production quantities for each plant, and probably even for the whole network, must be adjusted.

LTP is done once a year. However, the manufacturing company wants to improve its agility to better align with demand and supply changes. They intend to start performing LTP multiple times a year. For this goal to be achieved, they must improve several aspects of their planning process.

7.2.1 Old process issues

The existing, soon-to-be superseded, LTP process is both cumbersome and likely to produce bad plans for the following reasons:

- ▶ It is based on many spreadsheets.
- ▶ Distributed, collaborative planning is not well supported. Files must be exchanged, compared, and merged manually.
- ▶ The planning is essentially done manually, without the support of automated mathematical planning functions.

Remember that such a large supply chain can incur huge costs. Improving the process and the plans by only a small fraction can result in significant savings. Often such a plan can be improved by much more than a small fraction.

7.2.2 Solution

IBM implemented an ODM Enterprise based solution for the manufacturing company with the following highlights:

- ▶ A tailored and consistent data model.
- ▶ A planning cockpit that allows for efficient user interaction, including scenario-based simulation and data comparison.
- ▶ Centralized information storage that improves data security and allows for collaborative teamwork.
- ▶ Planning supported by mathematical functions that *automatically* drive the planning goals towards minimal costs.
- ▶ More planning details can be taken into account automatically.
- ▶ This approach can speed up the creation of planning scenarios. Thus, more scenarios can be evaluated in shorter time.

All these highlights help mitigate the current planning issues, as outlined previously. But the new approach can do more than that. It can even help detect and reduce supply chain risks. Because more scenarios can be evaluated at a better quality, there is now a chance to even consider unlikely scenarios.

7.3 Business requirements

The manufacturing company wants to perform the LTP multiple times a year. This does not mean to reduce complexity in the LTP and do it more often. Rather, it means they want to combine long-term planning with mid-term planning, while still looking at the same horizon (1-5 years) as before. And they want to do this planning faster and more often, but at better quality.

LTP means planning at two different levels:

Assignment Worldwide network level

Capacity planning Individual plant level

In fact, the LTP is a two-step process where assignment is done first, then local capacities are planned.

7.3.1 Assignment

The *assignment planning* can be characterized as follows.

- ▶ *Scope*: The whole worldwide production *network* is considered, including all plants (about 20).
- ▶ *Periods*: Only one planning period is considered, or a few (such as five years or four quarters).
- ▶ *Products*: Because the whole production chain is considered, all products must be taken into account (up to 20,000).
- ▶ *Resources*: A few resource types are considered, like plants, certain machines, or product characteristics.
- ▶ *Bill-of-Resources (BOR)*¹: For each product, and each corresponding resource, the resource usage must be respected. This is essentially the time that is required by the resource to process the product.

¹ This concept is sometimes also called *Bill-of-Capacity (BOC)*. For more information about BOC / BOR, see *Applications of Implosion in Manufacturing*, in Chae An and Hansjörg Fromm, *Supply Chain Management on Demand: Strategies, Technologies, Applications*, Springer Verlag, Berlin Heidelberg 2005, ISBN 978-3540244233

- ▶ *Production complexity*: For each plant, the *production complexity* must be computed and taken into account.
- ▶ *Transportation*: Because we consider the whole, worldwide production network, transportation costs are significant and must be considered. The costs are given from each plant to all worldwide market regions.

Assignment planning is always done first. It is carried out by a team at the organization's headquarters. The result of the assignment is an important input for the next step, capacity planning.

7.3.2 Capacity planning

Capacity planning takes the assignment input and solves a similar planning problem, but for each individual plant.

- ▶ *Scope*: In capacity planning, each production plant is considered individually.
- ▶ *Periods*: Here, a few planning periods are considered, such as 12, for 12 months, or 20, for five years times four quarters.
- ▶ *Products*: For each plant, only the products that can be produced there are considered.
- ▶ *Bill-of-Materials (BOM)*²: The BOM defines the materials that are needed to assemble a specific product. Up to 100 materials can be listed for the most complex products.
- ▶ *Resources*: Here, many resource types and resource instances are considered. A calendar must be taken into account for most of the resources. The calendar essentially defines the number of days per period when this resource is active. Additionally, for each product and corresponding resource, a processing time is given, called the *Machine Dependent Time (MDT)*.
- ▶ *Production complexity*: For each plant, a *production complexity* must be computed and taken into account.
- ▶ *Transportation*: No transportation is considered in capacity planning.

This planning step essentially detects and mitigates possible production bottlenecks. This process is why such a planning step is often also called *de-bottlenecking*.

Unlike the assignment, capacity planning is done by using a distributed approach. Each plant essentially takes care of its own capacity planning. In theory, they can do capacity planning independently from each other. In practice,

² For more information about BOM concepts, see Chae An and Hansjörg Fromm, *Supply Chain Management on Demand: Strategies, Technologies, Applications*, Springer Verlag, Berlin Heidelberg 2005, ISBN 978-3540244233

however, they must communicate about possible assignment shifts. Therefore, they need a system that supports them doing *cooperative, iterative planning*.

7.3.3 Capacities, overload, and extension

As stated initially, a major goal of this project is to balance difficult trade-offs. Many trade-offs come from the fact that resources are limited. The planners then must decide how to make optimal use of a resource. This optimization means they must decide what *not* to produce with the resource.

Another way to cope with this problem is to get rid of resource limits. Because resources are usually expensive, you have to spend money to extend the resource to get rid of a particular limit. For example, you might have to buy an additional machine. The question then is: When is it worth spending money to extend resources?

De-bottlenecking

De-bottlenecking is about the following issues:

- ▶ *Finding* the production *bottlenecks*, which as the machines, tools, or other resources that actually limit production so that not all the demands can be met.
- ▶ *Detecting* which resources are worth extending. This process involves comparing extension costs (like the price for a new machine) with potential sales increase provided by the additional production this resource extension provides.

In fact, the manufacturing company wants to do de-bottlenecking in both assignment and capacity planning. The first step treats many other things at the same time, whereas the latter step is focused on de-bottlenecking.

Two concepts that support de-bottlenecking are used, *production overload* and *production extension*.

Production overload

Production overload, usually called *opportunity costs*, is essentially a demand³ that cannot be produced with the available resources in a specific time frame.

Usually, *any one* of all the demands could be produced, but sometimes *not all* of them can be produced (at least in the correct location). During optimized production planning, the algorithms first fill the available resources with demands

³ When talking about *demand* here, it does not necessarily mean demand for which the company already has a customer order to book. It might also mean projected or forecasted (future) demand. In the LTP planning process, demand is usually forecasted.

that yield high *value* such as revenue, income, or earnings. It does that until no further demand can be fulfilled. The remaining demand is then essentially the overload. It is also the demand that brings less value than the already assigned demands. But it might still bring enough value for a resource extension to make financial sense.

Overload can be measured in the number of items, or in monetary value, for example, missed earnings, or revenue, or income, or the like.

In reality, such an overload might be caused by shortage of minor or inexpensive materials or resources. Consider the following example. Assume that you want to produce 100 items of a certain product, say a GPS. The BOM defines 50 necessary materials to produce one GPS. Of one of these 50 materials, say the antenna, only 10 items are available. Therefore, only 10 GPSs can be produced, even if all other materials are readily available. The overload for this GPS would be 90 items, and the bottleneck is the antenna. The bottleneck can also be a machine, or any other resource.

As just noted, sometimes not all demands can be produced *in the correct location*. This means that, when you are looking at the worldwide network, the plan must balance transportation costs with overload costs (or extension costs). This aspect makes the planning difficult, and worth using mathematical optimization.

Production extension

The second concept that is as important for de-bottlenecking is *production extension*. Production extension is essentially the extension of production resources, like people, machines, tools, means of transportation, or even plants.

In the planning phase, the existing resources and limits must be taken into account. But extension is possible, albeit at a cost. Extension costs are the estimated costs for hiring new people, buying machines, or building and maintaining a new plant.

Remember that an optimization model is essentially a large formula that connects everything. You can add allowance for resource extensions and bind this to extension costs. The optimization then balances the addition of new resources (incurring costs) with all the actual benefits those resources might generate by reducing production overload.

Even if real costs are hard or impossible to estimate, this approach automatically finds the bottleneck resources.

7.3.4 Non-functional requirements

For the final software solution, the manufacturing company defined the following non-functional requirements:

- ▶ One application must serve the two planning steps: Assignment and capacity planning.
- ▶ There must be ways to publish the resulting data of the assignment to the multiple capacity planning instances.
- ▶ The solution must support scenario-based what-if analysis. Many different scenarios will be created.
- ▶ The mathematical optimization must run fast. A response time of a few minutes must be achieved. The optimization must be well integrated with the solution.
- ▶ The system must support teams that are doing cooperative, iterative planning.
- ▶ User interface (GUI)
 - The GUI must support state-of-the-art interaction patterns.
 - The users will work with a large amount of data. The views must support analysis of these large data sets.
 - Pivoting is desirable so that the users can analyze the structured input data and optimization results. For more information, see 3.1, “Key ODM Enterprise concepts” on page 56.
 - In the end, there will be many users that potentially work with the application. The software must support this workload.
 - Some specific GUI functions must be made available to the users such as, connected views, special window layouts, coloring, and naming.

7.4 Optimization problem

This section addresses the optimization problems for the two planning steps, assignment and capacity planning.

Use the *TLC+D* framework, as introduced in 1.2.3, “How do I recognize an optimization application” on page 7, to describe these items:

- ▶ Optimization targets (Goals)
- ▶ Observing limits (Constraints)
- ▶ By making choices (Decision Variables)
- ▶ Using data (Input and Output tables)

In addition, it provides details about how to implement the models. For clarity and simplicity, OPL⁴ pseudo code is used for the model elements. This code is more verbose than mathematical formulas, and less technical than real OPL code. This pseudo code will not work exactly as provided here. It needs additional plumbing. Sometimes it is even syntactically incorrect. If you are already familiar with OPL code, if *S* is a tuple set with one key attribute, the term *S[i]* is used to for access the tuple with the key value *i*. In the pseudo code snippets, comments (non-OPL code) are introduced with “//” and written in italics.

7.4.1 Assignment

The first planning step under the TLC+D framework is assignment planning. Table 7-2 on page 234 provides an overview of the TLC components of the underlying optimization model.

⁴ As already introduced earlier (4.2.4, “Creating a prototype with optimization” on page 138), OPL (Optimization Programming Language) is the ILOG optimization modeling language. For more information about OPL, see:
<http://www.ibm.com/software/integration/optimization/cplex-optimization-studio/modeling/>

Table 7-2 Assignment model overview

Targets	Limits	Choices
Minimize transportation cost of articles from plants to distribution centers in subregions. Minimize production cost, which is a function of unit production cost of articles at each plant. Minimize machine and tool capacity extension cost. Minimize overload assignment of demand to the plants above the capacities of the plants, machines and tools, and article-sizes. Minimize complexity of the manufacturing process by reducing the total number of types of articles that are assigned to each plant. Minimize the number of failures to obey predefined (frozen) assignment requirements (if assignment-freezing constraints are relaxed).	All of the demand should be assigned to plants. Capacity limitations should be obeyed at the machine and tool level, and at article-size level. They should also be obeyed at the aggregate plant level, where plant capacity is driven by the complexity of the manufacturing process. Capacity extension is only allowed for certain machines and tools. Overload assignment is only allowed for certain machines and tools, article-sizes, and plants. Article to plant mapping requirements (based on the plant capability) should be obeyed while the system is assigning the demand. Predefined assignment requirements should be obeyed (if assignment-freezing constraints are not relaxed).	How much of the demands of each article in each subregion should be assigned to each plant. What is the production capacity at machine and tool level, at article level, and at plant level. This determination should consider capacity extension, assignment overloading, and manufacturing complexity. How much machine and tool capacity extension is needed. How much demand assignment overloading is needed (to cover all demand) at plant, machine and tool, and article-size levels. What is the total manufacturing complexity at each plant. What is the number of failures to obey predefined (frozen) demand assignment requirements (if assignment-freezing constraints are relaxed).

Data

To understand the application data model for the assignment optimization model, see the data diagram in Figure 7-2.

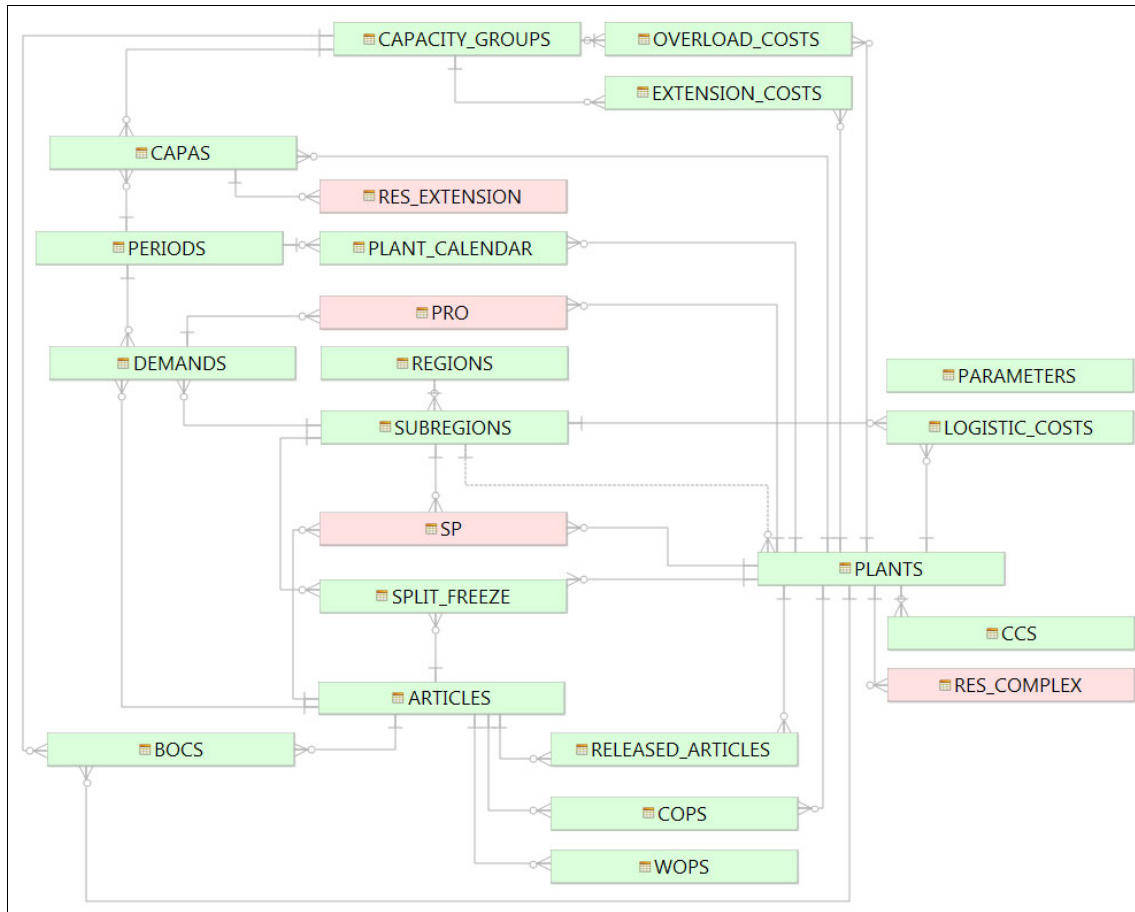


Figure 7-2 Demand assignment data diagram

The diagram is color coded. The tables in green indicate input data tables, and red indicates the output data tables.

The input and output tables encapsulate the data (D) part in your TLC+D framework. Table 7-3 shows an overview of the various input tables and what they represent in the assignment model.

Table 7-3 Input data tables in the assignment model

Table Name	Table Description
ARTICLES	Article master data, covering article-specific details
BOCS	Bill of capacity
CAPACITY_GROUPS	Capacity group master data, which limits production activity
CAPAS	Capacity calculation details for each plant, such as utilization rate during each period
CCS	Manufacturing complexity lookup table, used to generate step-wise function of number of article types versus effective plant capacity
COPS	Cost of production
DEMANDS	Demand for each part and subregion during each period
EXTENSION_COSTS	Cost of material/tool capacity extension
LOGISTIC_COSTS	Unit cost of transportation from plants to distribution centers at subregions
OVERLOAD_COSTS	Cost of overloaded demand assignment
PERIODS	Planning periods
PLANT_CALENDAR	Plant calendar, including plant capacity utilization during each planning period
PLANTS	Plant master data, covering plant-specific details
REGIONS	Region master data, covering region-specific details
RELEASED_ARTICLES	Article to plant mapping requirements (based on the plant capability)
SPLIT_FREEZE	Predefined demand assignment requirements
SUBREGIONS	Subregion master data, covering subregion-specific details
PARAMETERS	Stores simple user-defined variables that take only one value (as opposed to a set of values).

Table 7-4 shows an overview of the various output tables and what they represent in the assignment model.

Table 7-4 Output data tables in the demand assignment model

Table Name	Table Description
PRO	Demand assignment details, including assignment quantity and production cost
RES_COMPLEX	Manufacturing complexity that is based on type of articles that are assigned to each plant
RES_EXTENSION	Material/tool capacity extension
SP	Demand assignment details (<i>Split</i>)

Choices

The optimization model needs choices (or *decision variables* in ODM Enterprise terms) that best fit the problem context. To accommodate the various targets, the OPL formulation uses these decision variables:

► *split*

This variable, which is indexed over demand (characterized by subregion, article, and planning period), and plant, is used to denote a fraction. It therefore lies between the values of 0 and 1. The decision variable helps assign the optimal fraction of a subregion demand (article) to a certain plant. An OPL representation of this decision variable is shown in Example 7-1.

Example 7-1 *split variable OPL declaration*

```
dvar float split[DEMANDS][PLANTS] in 0..1;
```

► *articleTypeAtPlant*

This variable, which is indexed over article types and plants, is an indicator variable. It has value one if an article type is assigned to a plant. The value of this indicator variable is then aggregated over the plant to calculate the total number of article types that are assigned to a plant. This calculation drives the manufacturing complexity.

An OPL representation of this decision variable is shown in Example 7-2.

Example 7-2 articleTypeAtPlant variable OPL declaration⁵

```
// Declaration of new set, capturing distinct article types
{string} ARTICLE_TYPES = {a.ARTICLE_TYPE | a in ARTICLES};

dvar boolean articleTypeAtPlant[ARTICLE_TYPES][PLANTS];
```

► *prodCapa*

This variable captures the production capacity at the machine and tool level, at the article level, and at the plant level, considering capacity extension and manufacturing complexity. It is indexed over collection of capacity requirements, CAPAS, which in turn is characterized by plant, capacity group, and planning period. An OPL representation of this decision variable is shown in Example 7-3.

Example 7-3 prodCapa variable OPL declaration⁶

```
dvar float+ prodCapa[CAPAS];
```

► *prodExtension*

This variable captures the machine and tool capacity extension at the plant to cover the assigned demand. It is indexed over collection of capacity requirements, CAPAS, which in turn is characterized by plant, capacity group, and planning period. An OPL representation of this decision variable is shown in Example 7-4.

Example 7-4 prodExtension variable OPL declaration

```
dvar int+ prodExtension[CAPAS];
```

► *prodOverload*

This variable captures the capacity overloading that is required to cover the assigned demand at plant, machine and tool, and article-size levels. It is indexed over bill of capacity, BOCS, which in turn is characterized by article, plant, and capacity group. An OPL representation of this decision variable is shown in Example 7-5.

Example 7-5 prodOverload variable OPL declaration

```
dvar float+ prodOverload[BOCS];
```

⁵ The set definition of ARTICLE_TYPES in this example is analogous to the SQL statement
SELECT ARTICLE_TYPE FROM ARTICLES.

⁶ The notation float+ indicates continuous, non-negative decision variables.

► *splitFreezeFailure*

This is an indicator variable that captures the failures to predefined demand assignment requirements. It is indexed over split freeze requirements, SPLIT_FREEZE, which in turn is characterized by article, plant, and subregion. It has value one if the split freeze requirement is not satisfied, and zero otherwise. An OPL representation of this decision variable is shown in Example 7-6.

Example 7-6 splitFreezeFailure variable OPL declaration

```
dvar boolean splitFreezeFailure[SPLIT_FREEZE];
```

Targets

As mentioned in Table 7-2 on page 234, there are several targets (or objective function components of the optimization model):

- Minimize the transportation cost of articles from plants to distribution centers in subregions. An OPL representation of this target declaration is shown in Example 7-7.

Example 7-7 totalTransportationCost target OPL declaration⁷

```
dexpr float totalTransportationCost = sum(d in DEMANDS, p in PLANTS)
(LOGISTIC_COST[d.SUBREGION][p] * d.ARTICLE.WEIGHT * d.VALUE) *
split[d][p]);
```

- Minimize the production cost, which is a function of unit production cost of articles at each plant. An OPL representation of this target declaration is shown in Example 7-8.

Example 7-8 totalProductionCost target OPL declaration

```
dexpr float totalProductionCost = sum(d in DEMAND, p in PLANT)
(productionCost[d.ARTICLE] * d.VALUE) * split[d][p];
```

- Minimize the machine and tool capacity extension cost. An OPL representation of this target declaration is shown in Example 7-9.

Example 7-9 extensionCost target OPL declaration

```
dexpr float extensionCost = sum(c in CAPAS)
EXTENSION_COST[c.PLANT][c.CAPACITY_GROUP].COST * prodExtension[c];
```

⁷ For more information about the notation that is used in these examples, see “A smidgen of OPL (part 4) - Filtering with tuples using formal parameters:” on page 198.

- Minimize the overload assignment of demand above the plant, machine and tool, and article-size capacities. An OPL representation of this target declaration is shown in Example 7-10.

Example 7-10 totalOverloadAssignment target OPL declaration

```
dexpr float totalOverloadAssignment = sum(b in BOCS)
OVERLOAD_COST[b.PLANT][b.CAPACITY_GROUP].COST * prodOverload[b];
```

- Minimize the complexity of the manufacturing process by reducing the total type of articles that are assigned to each plant. An OPL representation of this target declaration is shown in Example 7-11.

Example 7-11 totalManufacturingComplexity target OPL declaration

```
dexpr float totalManufacturingComplexity = sum(y in ARTICLE_TYPES, p
in PLANTS) articleTypeAtPlant[y][p];
```

- Minimize the number of failures to obey predefined (frozen) demand assignment requirements (if assignment-freezing constraints are relaxed). An OPL representation of this target declaration is shown in Example 7-12.

Example 7-12 totalSplitFreezeFailures target OPL declaration

```
dexpr int totalSplitFreezeFailures = sum(s in SPLIT_FREEZE)
splitFreezeFailure[s];
```

Limits

As mentioned in Table 7-2 on page 234, there are several limits (or constraints of the optimization model):

- Demand constraints

These constraints ensure that demand for all articles and subregions is assigned to plants. Example 7-13 shows the OPL formulation of this constraint, where the sum of *split* variables over PLANTS for each DEMAND is forced to be 1 (100%).

Example 7-13 Demand constraints

```
forall(d in DEMAND) sum(p in PLANTS) split[d][p] == 1;
```

- Capacity constraints

These constraints ensure that the capacity limitations in each CONSTRAINT_GROUP (plant, machine and tool, and article-size) is obeyed. As shown in Example 7-14 on page 241, capacity constraints consist of three major constraint sets: (1) Available capacity at the plant level, (2) Available

capacity at machine and tool, and article-size level, and (3) Capacity limitation.

Effective plant capacity is negatively correlated with the number of article types that are assigned to a plant because article variety results in a more complex manufacturing environment. This relationship is captured by a stepwise function, as demonstrated in Figure 7-3. If the number of article types that are assigned to a plant is less than BASE0, the plant can use 100% of its capacity. If the number of article types is between BASE0 and BASE5, the plant capacity is reduced to 95%.

In the first set of capacity constraints, the plant capacity for each period is calculated by multiplying the available hours at the plant, $(\text{PLANT_CALENDAR}[c.\text{PLANT}][c.\text{PERIOD}].\text{CUD} * 24)$, with the effective plant capacity.

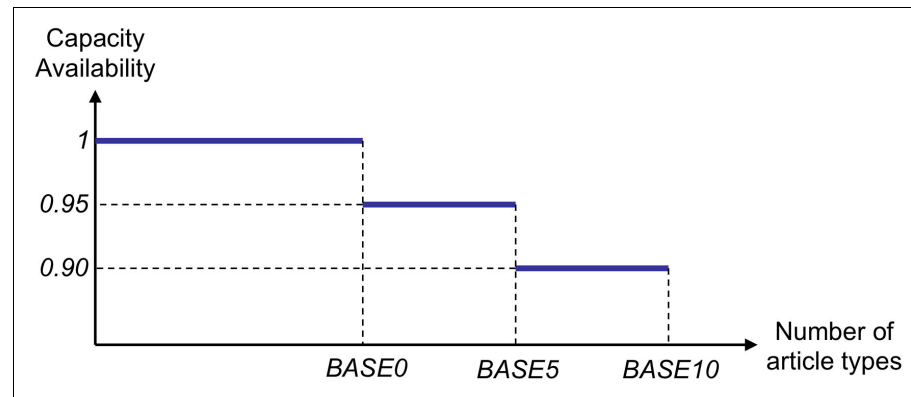


Figure 7-3 Complexity driven, effective plant capacity

In the second set of capacity constraints, the available capacity in other CAPACITY_GROUPS is calculated by multiplying the number of available and extended units $(c.\text{NOU} + \text{prodExtension}[c])$ with the available hours at the plant.

The last set of capacity constraints limits the capacity consumption by the assigned demand. In these constraints, *prodOverload* variables capture the excessive capacity requirement (Example 7-14).

Example 7-14 Capacity constraints

```
// Available capacity at PLANT level
forall(c in CAPAS: c.CAPACITY_GROUP_ID=="PLANT")
  prodCapa[c] ==
    (PLANT_CALENDAR[c.PLANT][c.PERIOD].CUD * 24) *
    stepwise{
```

```

1->CCS[c.PLANT].BASE0;
0.95->CCS[c.PLANT].BASE5;
0.90->CCS[c.PLANT].BASE10; 0}
(sum(y in ARTICLE_TYPES) articleTypeAtPlant[y][c.PLANT]);

// Available capacity at machine/tool and article-size level
forall(c in CAPAS: c.CAPACITY_GROUP_ID!="PLANT")
    prodCapa[c] ==
        (c.NOI + prodExtension[c]) *
        PLANT_CALENDAR[c.PLANT][c.PERIOD].CUD * 24;

// Capacity limitation
forall(c in CAPAS)
    prodCapa[c] >=
        sum(b in BOCS, d in DEMAND:
            c.PLANT==b.PLANT &&
            c.CONSTRAINT_GROUP==b.CONSTRAINT_GROUP &&
            c.PERIOD==d.PERIOD &&
            b.ARTICLE==d.ARTICLE)
            (d.VALUE*split[d][c.PLANT] - prodOverload[b]) *
            b.MDT/b.MDT_BASE_QTY;

```

► Capacity extension constraints

These constraints dictate that capacity extension is only allowed for certain machines and tools. They are indicated by having a non-negative extension cost, as shown in Example 7-15.

Example 7-15 Capacity extension constraints

```

forall(c in CAPAS : EXTENSION_COST[c.PLANT][c.CAPACITY_GROUP].COST
<= 0)
    prodExtension[c] == 0;

```

► Overload demand assignment constraints

These constraints dictate that overload demand assignment is only allowed for certain machines and tools, article-sizes, and plants. They are indicated by having a non-negative overloading cost, as shown in Example 7-16.

Example 7-16 Overload demand assignment constraints

```

forall(b in BOCS : OVERLOAD_COST[b.PLANT][b.CAPACITY_GROUP].COST <=
0)
    prodOverload[b] == 0;

```

► Article to plant mapping constraints

These constraints make sure that any predefined article to plant mapping requirement is obeyed while assigning the demand. This requirement is based on the plant capability, which is indicated by RELEASED_ARTICLE set. OPL code for this constraint is shown in Example 7-17.

Example 7-17 Article to plant mapping constraints

```
forall(d in DEMAND, p in PLANTS: <p, d.ARTICLE> not in
RELEASED_ARTICLES)
    split[d][p] == 0;
```

► Split-freeze constraints

These constraints ensure that any predefined demand assignment requirement, which is defined by the SPLIT_FREEZE set, is obeyed while you are making the assignment decisions. If a split freeze requirement is not obeyed, the Boolean splitFreezeFailure variable will have a value of 1. OPL code for these constraints is shown in Example 7-18.

Example 7-18 Split-freeze constraints

```
forall(f in SPLIT_FREEZE)
    splitFreezeFailure[f] >= (f.VALUE -
split[<f.ARTICLE,f.SUBREGION>][f.PLANT]) / f.VALUE;
```

Clarification: Example 7-18 illustrates a common modeling “trick” used in optimization, the use of a discrete indicator variable to count the occurrences of a condition. OPL is a declarative language, so you cannot procedurally count occurrences, as you would do with a loop in C++ or Java. Instead, you must state a condition that must be true at the optimal solution.

The indicator variable splitFreezeFailure[f] can take only the values 0 or 1. Therefore, any continuous value on the right side greater than 0 forces the indicator to take the value 1. However, if the indicator takes the value 0, it forces the right side to also take value 0. Thus, the sum of the splitFreezeFailure variables is the number of times VALUE exceeds split.

Dividing by VALUE normalizes the right side to lie between 0 and 1. This process, although not strictly necessary for the logic, makes the constraint “tight” and reduces the computational effort that is needed to solve the optimization problem. Using normalization is a best practice for model formulation.

This principle can be applied in other counting situations. It shows the versatility available with mix-integer programming.

7.4.2 Capacity planning

The second planning step under the TLC+D framework is capacity planning. Table 7-5 provides an overview of the TLC components of the underlying optimization model.

Table 7-5 Capacity planning optimization model overview

Targets	Limits	Choices
Minimize overload costs by trying to satisfy every demand as much as possible using the available work centers. Reduce the complexity of the planning process by trying to use the least number of active work centers. Materials should be processed by the fastest work centers (with the least machine-dependent time) as much as possible.	The time that is required to satisfy the assigned demand at a work center should be less than or equal to the production time available at that work center. Work centers should produce no more than the required demand for various articles.	How should demand be split between various work centers to best reach the targets while meeting the limits?

Data

To better understand the application data model for the capacity planning optimization model, see the data diagram in Figure 7-4.

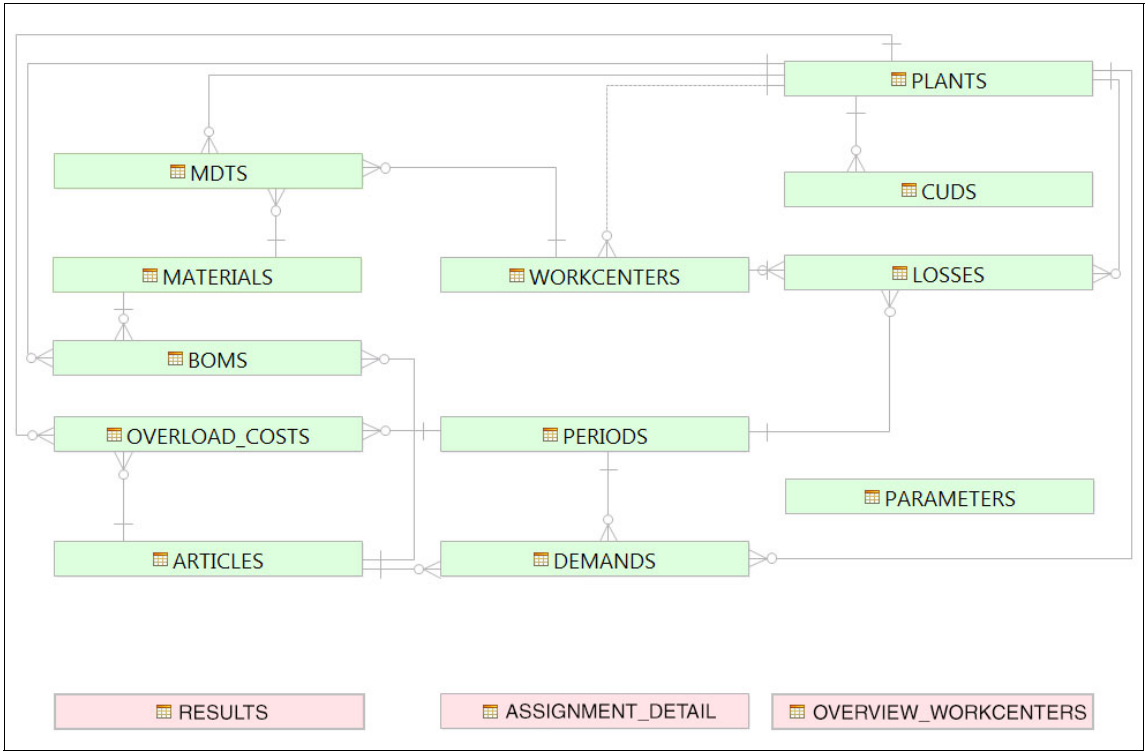


Figure 7-4 Capacity planning data diagram

The diagram is color coded. The tables in green and red indicate the input data tables and those in red indicate the output data tables.

Table 7-6 shows an overview of the various input tables, and what they contain in the context of the capacity planning model.

Table 7-6 Input tables

Table name	Table description
PERIODS	Planning periods: Stores period ID and their chronological order.
ARTICLES	Stores article details. Articles require materials as mapped in the BOMS table.

Table name	Table description
PLANTS	Stores plant information. Plants have multiple work centers, each of which have multiple tools/machines to produce materials as required by the article demand.
DEMANDS	Stores demand by period, article, and plant.
MATERIALS	Stores details about all the available materials and the material class they belong to.
WORKCENTERS	Stores the list of available work centers by plant, along with information about the number of available tools/machines.
BOMS	This table maps articles to materials. It contains information about how much and which materials are required to produce a certain article.
CUDS	Stores the capacity utilization days per plant in a particular planning period.
LOSSES	Stores availability losses by plant and work center.
MDTS	Stores machine-dependent time by plant, material, and work center. It stores information about how long a certain work center would take to process one unit of material at a particular plant.
OVERLOAD_COSTS	Stores overload costs by period, article, and plant that are applied when a certain demand is not completely fulfilled.
PARAMETERS	Stores simple user-defined variables that take only one value (as opposed to a set of values).

Table 7-7 shows an overview of the various output tables, and what they contain in the context of the capacity planning model.

Table 7-7 Output tables

Table Name	Table Description
RESULTS	Stores the overall results of the optimization that is run with the assignment plan.
OVERVIEW_WORKCENTERS	Stores work center overview information in terms of total available, production, and idle times.
ASSIGNMENT_DETAIL	Stores detailed split information for each article, material, and work center combination.

Choices

To elaborate on the optimization model, this section addresses the choices (or *decision variables* in ODM Enterprise terms) that best fit the problem context. The following sections include some pseudo-code to give you a feel of the optimization model using OPL. Note that `dvar` is an OPL keyword that is used to declare decision variables (choices) whereas `forall` is a keyword to iterate through the elements of a set. The non-OPL part of the pseudo-code is italicized for clarity.

To accommodate the various targets, the optimization model has three decision variables:

► *split*

This variable is used to denote a fraction and thus lies between the values of 0 and 1. It is indexed over demand (which is characterized by the period, article, plant, quantity), material, and work center. The decision variable helps assign the optimal fraction of a demand (article) and its associated materials to a certain work center. It takes MDT (machine-dependent time) and other factors into account.

The corresponding formulation in the optimization model is shown in Example 7-19.

Example 7-19 split variable declaration

DMWS is a set of valid {demand,material,workcenter} combinations where demand is in DEMANDS, material is in MATERIALS and workcenter is in WORKCENTERS

```
dvar float split[DMWS] in 0..1;
```

► *sumSplit*

This variable, which is indexed over demand, is a fraction that lies between the values of 0 and 1. It denotes the fraction of a particular demand (article, period, and plant-wise) that is met by the current assignment plan. Therefore, any number that is less than 1 refers to a demand that has not been satisfied completely. This variable is eventually used to calculate overload costs.

The corresponding formulation in the optimization model is shown in Example 7-20.

Example 7-20 sumSplit variable declaration

```
dvar float sumSplit[DEMANDS] in 0..1;
```

It is important to understand the distinction between the `split` and `sumSplit` variables. The `split` variable maps the fraction of a certain demand and

material to a certain work center. `sumSplit` corresponds to the fraction of the actual demand that can be fulfilled among all the work centers that can process a demand. This structure is needed because every demand corresponds to an article, but the article itself can be made up of many materials as specified in the bill of materials (BOM). When a certain article demand cannot be met, make sure that all the corresponding materials that make up the article get fulfilled to the same level as the original article. Example 7-21 depicts an example scenario to explain these decision variables.

Example 7-21 Explaining split and sumSplit: Input data

DEMANDS table entry:

`Demand1 = <Period1, Article1, Plant1, 100> //demand of 100 units of Article1 for Period1 out of Plant1`

BOMS table entries:

`<Article1, Plant1, Material1, 4> //Article1 requires 4 units of Material1`

`<Article1, Plant1, Material2, 8> //Article1 also requires 8 units of Material2`

MDTS table entries

`<Plant1, Material1, Workcenter1, 10> //Workcenter1 can produce 1 unit of Material1 in 10 minutes`

`<Plant1, Material1, Workcenter2, 10> //Workcenter2 can also produce 1 unit of Material1 in 10 minutes`

`<Plant1, Material2, Workcenter3, 20> //Workcenter3 can produce 1 unit of Material2 in 20 minutes`

As an example, assume that because of the high load on the work centers because of other demands, only 50% of Demand1 can be fulfilled. This shortfall means that only 50 units of Article1 can be produced. Production requires $4 \times 50 = 200$ units of Material1 and $8 \times 50 = 400$ units of Material 2. Therefore, in this case, (part of) the result is shown in Example 7-22.

Example 7-22 Explaining split and sumSplit: Result

`split[Demand1, Material1, Workcenter1] = 0.25 //because Workcenter1 and Workcenter2 can process Material1, each has to share only half of the assigned demand`

`split[Demand1, Material1, Workcenter2] = 0.25 //because Workcenter1 and Workcenter2 can process Material1, each has to share only half of the assigned demand`


```
split[Demand1, Material2, Workcenter3] = 0.5 //because Workcenter3
is the only available workstation for Material2 and thus has to
produce all of the assigned demand
sumSplit[Demand1] = 0.5 //only 50% of the Demand1 can be fulfilled
because of capacity constraints
```

► **isActive**

This Boolean decision variable helps keep count of the active work centers, which are work centers that are processing any material to fulfill a corresponding demand. The variable is necessary to help the model with its assignment such that the complexity in making articles (and meeting demands) is minimized. Complexity is high when many work centers are required to meet demands. The optimization model tries to reduce this complexity as much as possible with the help of this decision variable. The corresponding formulation in the optimization model is shown in Example 7-23.

Example 7-23 isActive variable declaration

```
dvar boolean isActive[MATERIALS][DEMANDS];
```

Targets

As mentioned earlier, there are three targets or objectives function components to the optimization model:

► **Minimize total overload cost**

Overload cost is defined as the cost that is incurred by unsatisfied demands. The optimization model needs to satisfy as much demand as it can by using the available work centers. However, there can be situations where the demand is too much for the existing work centers to handle, leading to overload costs. To minimize such costs, the optimization should first try to assign and use the available time for all the work centers to satisfy demands as much as possible. In cases where an overload is impossible to avoid, select the overload demands appropriately such that the costs are minimized. In the optimization model, overload costs can be computed as shown in Example 7-24.

Example 7-24 totalOverloadCost objective

```
dexpr totalOverloadCost = sum(d in DEMANDS) (OVERLOAD_COST[d] *
d.DEMAND_QTY * (1-sumSplit[d]));
```

► **Minimize complexity**

One of the key objectives of the optimization model is to minimize complexity. In other words, the number of active work centers (governing operating and

production costs) that are required to satisfy demands need to be minimized. The corresponding objective is shown in Example 7-25.

Example 7-25 totalComplexity objective

```
dexpr totalComplexity = sum(m in MATERIALS, w in WORKCENTERS)
(isActive[m][w]);
```

► Minimize total throughput cost

While you are making the assignments between the demand, materials, and work centers, give higher throughput machines higher preference. You collect the data only for the time that is required to do a certain task on a machine. Therefore, try to select machines that take the least amount of time, or work centers with the lowest machine-dependent time (MDT). To do so, every demand, material, and work center combination is given a weight that gets multiplied with the split value to determine the cost. The weight for a demand and material is zero only for the work center with the lowest MDT, and a non-zero value (0.1 in our case) otherwise. The formulation is shown in Example 7-26.

Example 7-26 totalThroughputCost objective

```
forall(dmw in DMWS){
  if(dmw.w has the lowest MDT for dmw.m)
    WEIGHT[dmw] = WEIGHT_STANDARD; //0.0 in our case
  else
    WEIGHT[dmw] = WEIGHT_NONSTANDARD; //0.1 in our case
}
dexpr totalThroughputCost = sum(dmw in DMWS)
(WEIGHT[dmw]*split[dmw]);
```

It is important to understand that some of these objectives might work against each other. For example, think about the complexity and overload costs. Complexity is the least when you have the least number of active work centers. However, this configuration might result in a high overload cost because less of the demand is satisfied. Also, if you compare complexity costs versus throughput costs, some assignments to use machines with the highest throughput can result in the use of more work centers. This configuration can result in increasing rather than decreasing complexity. The solver balances assignment between these competing objectives based on the weights that drive these costs. Selecting the appropriate weight for each objective is extremely important, and depends on the behavior that is expected of the application.

Limits

The model itself has the capacity and demand constraints, However, because of the structures and variables introduced to capture the context, other limits need to be enforced as well:

► Capacity constraint

This requirement dictates that the available time at each work center should be greater than or equal to the amount of time required to process the demands assigned to it. A possible formulation of this constraint is shown in Example 7-27.

Example 7-27 Capacity constraint

```
forall(w in WORKCENTERS){
    // xWS[w] represents a subset of DMWS with workcenter w
    forall(i in CUDS) {
        if(w.PLANT_ID==i.PLANT_ID)
            availableTime[w]= CAPACITY_UPLIFT * (1-LOSSES_AV[w]/100) *
                                w.NOI * i.CUD * 24 * 60;
    }
}
availableTime[w] >= sum(x in xWS)(split[x] * x.demand.DEMAND_QTY *
    QTY_BOM[x.demand.article][x.material]*
    MDT_MIN[x.material][x.workcenter]);
```

► Complexity constraint

This constraint is used to set the value of the `isActive` variable. If the `split` value of any valid demand, material and work center combination is greater than 0, some portion of the demand would be satisfied by a work center using the corresponding material. Thus, the `isActive` variable corresponding to the particular material and work center should be set to 1 and a constraint of the following formulation, shown in Example 7-28, does the trick:

Example 7-28 Complexity constraint

```
forall(dmw in DMWS){
    isActive[dmw.material][dmw.workcenter]>=split[dmw];
}
```

► Split integrity constraint

The `split` variable represents the fraction of the demand that uses a certain material and gets assigned to a certain work center. You need to ensure that the sum of these fractions does not exceed 1 for a demand and material combination. In other words, this constraint ensures that the model does not

assign materials more than the demand itself. Example 7-29 shows the OPL code that implements this constraint.

► Split and sumSplit relationship constraint

This constraint defines the relationship between `split` and `sumSplit` variables. As defined earlier, `sumSplit` denotes the fraction of a demand that gets satisfied with the current plan, and is eventually used to calculate overload costs. Make sure that `minSplit` is equal to the sum of the splits that are related to a certain demand. The third and fourth constraints are shown in Example 7-29.

Example 7-29 split integrity, and split and sumSplit relationship constraints

```
forall(d in DEMANDS){
    // mats[d] represents a subset of MATERIALS with demand d
    // xDS[d] represents a subset of DMWS with demand d
    forall(m in mats[d]){
        sum(x in xDS[d] : x.material==m) split[x]<=1; //split integrity
        sum(x in xDS[d] : x.material==m) split[x]==sumSplit[d];
    }
    //split and sumSplit relationship
}
```

7.5 Solution

The solution that IBM implemented for the manufacturing company is an ODM Enterprise based planning and optimization application.

The previous sections addressed what the data models and optimization models for the two planning steps (assignment and capacity) look like. This section addresses the solution architecture, the user interface, and the realization project.

7.5.1 Solution architecture

Figure 7-5 depicts the different user groups of the application, and how they use the different services that the ODM Enterprise platform provides.

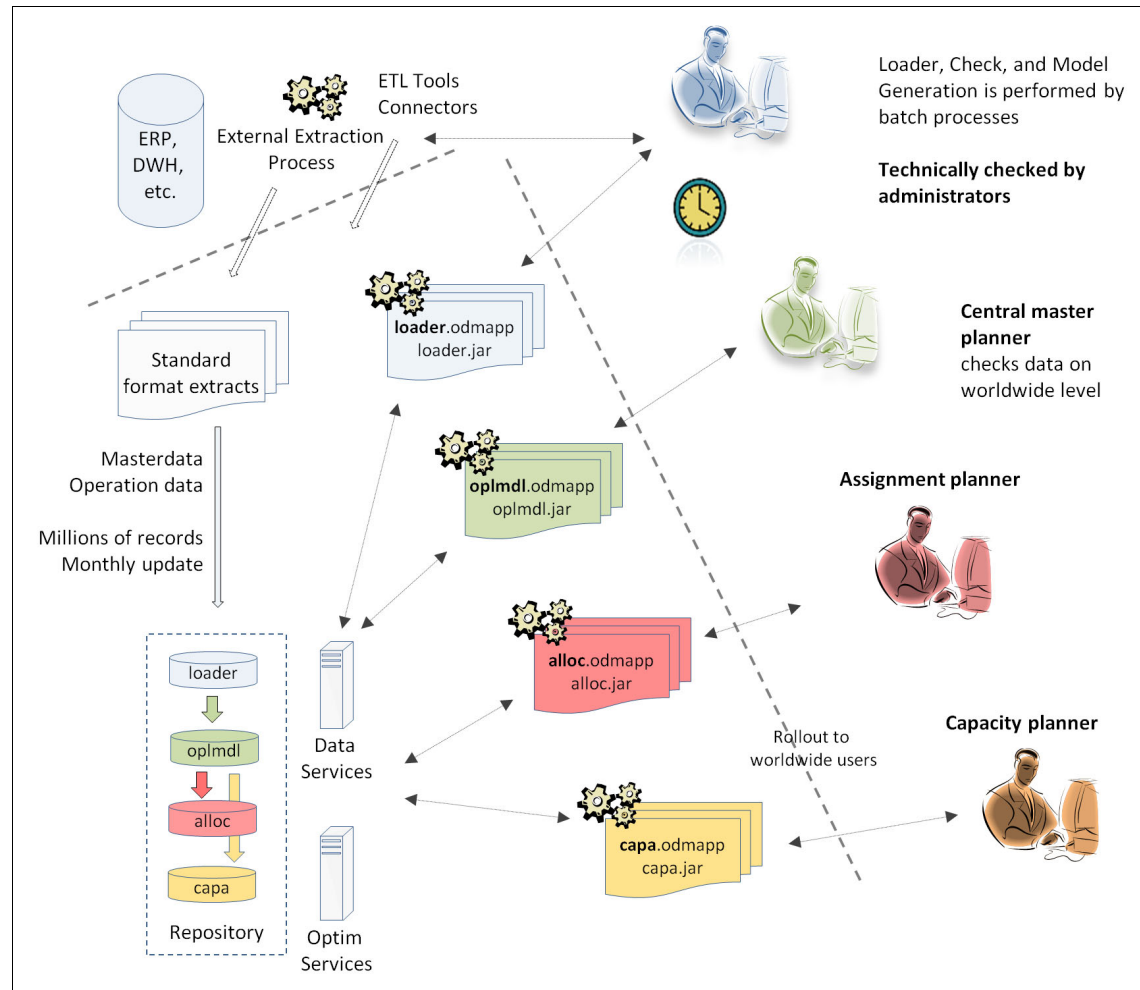


Figure 7-5 Different user groups and their connection to ODM Enterprise services

The user groups are, from top to bottom:

- **Data loader**
 - This is mainly a batch process that triggers external processes to prepare and provide data. It reads the data from the external systems into the repository database and then performs data checks (like integrity).

- A technical system administrator occasionally performs additional, manual data validation.
- ▶ Central master planner
This user oversees all worldwide data, and regularly checks the imported and other worldwide data in the scenario repository of the application.
- ▶ Assignment planner
The assignment planner performs the assignment planning on the data that is prepared for that purpose. The assignment planner is part of the worldwide core planning team that is in the organization's headquarters.
- ▶ Capacity planner
The capacity planner uses the result of the assignment planning, plus additional input data, to perform the capacity planning. Most of the capacity planners are distributed over the production locations throughout the world. Some team members represent centralized responsibilities outside the headquarters.

All the human planners use the ODM Enterprise Client user interface. Each is configured by essentially two files, an `.odmapp` file for the configuration, and a `.jar` file for the extensions.

Both the data loader and the central master planner use only the ODM Enterprise data service. The assignment planner and the capacity planner use both the data service, and the ODM Enterprise optimization service. They perform the actual planning and make heavy use of optimization.

Figure 7-6 shows, in a more technical fashion, the data flow of the application, again through the different planning stages.

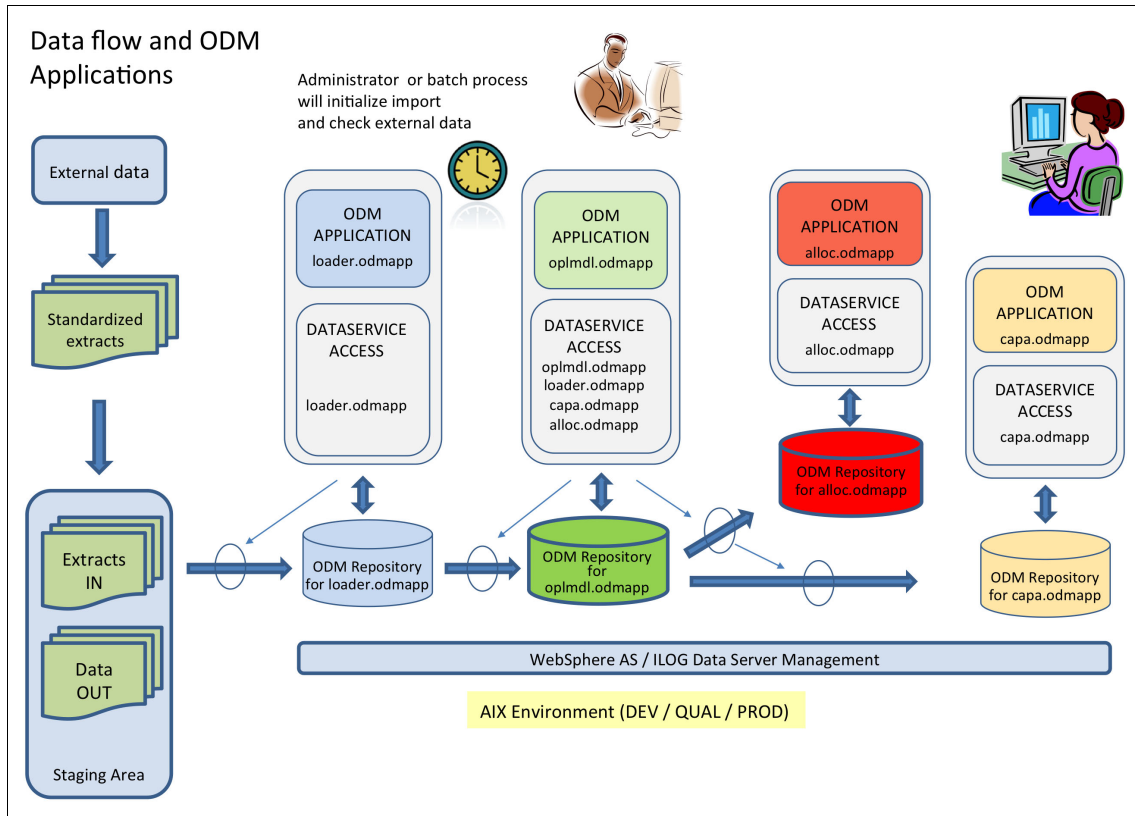


Figure 7-6 The data flow in the ODM Enterprise planning application

The steps are, from left to right:

1. Most of the planning data comes from external systems, such as *Enterprise Resource Planning (ERP)* or *Data Warehouse (DWH)* systems.
Some planning data has no origin outside the ODM Enterprise planning application, and is maintained only inside the application, in the scenario repository.
2. The external data is extracted from the external systems and transferred to the ODM Enterprise planning application, usually by using an *ETL (Extract-Transform-Load)* tool. It is then checked and validated inside the application's Scenario Repository. Specifically, it is in the "loader" repository, which is a data area that is configured to load the external data.

3. The technical administrator or central master planners transfer this data to the “oplmdl” repository stage (yet another data area). They then perform their business and functional validity checks.
4. The assignment planners extract the data parts that they need from the “oplmdl” stage into their “assignment” stage. There, they perform the assignment planning.
5. Likewise, the capacity planners extract the data parts that they need from the “oplmdl” stage into their “capacity” stage and perform the capacity planning. Because of the distributed nature of the capacity planning, the “capacity” repository might even be split into multiple disjoint data areas. For example, it might be one for each production plant.
6. Planning results are finally written to the external systems, like the ERP.

7.5.2 User interface

The ODM Enterprise Client already provides a rich set of planning functionality that is configured in the implementation project to fit the business and functional requirements. For more information, see 7.3, “Business requirements” on page 228.

Figure 7-7 shows a typical layout of the preliminary application.

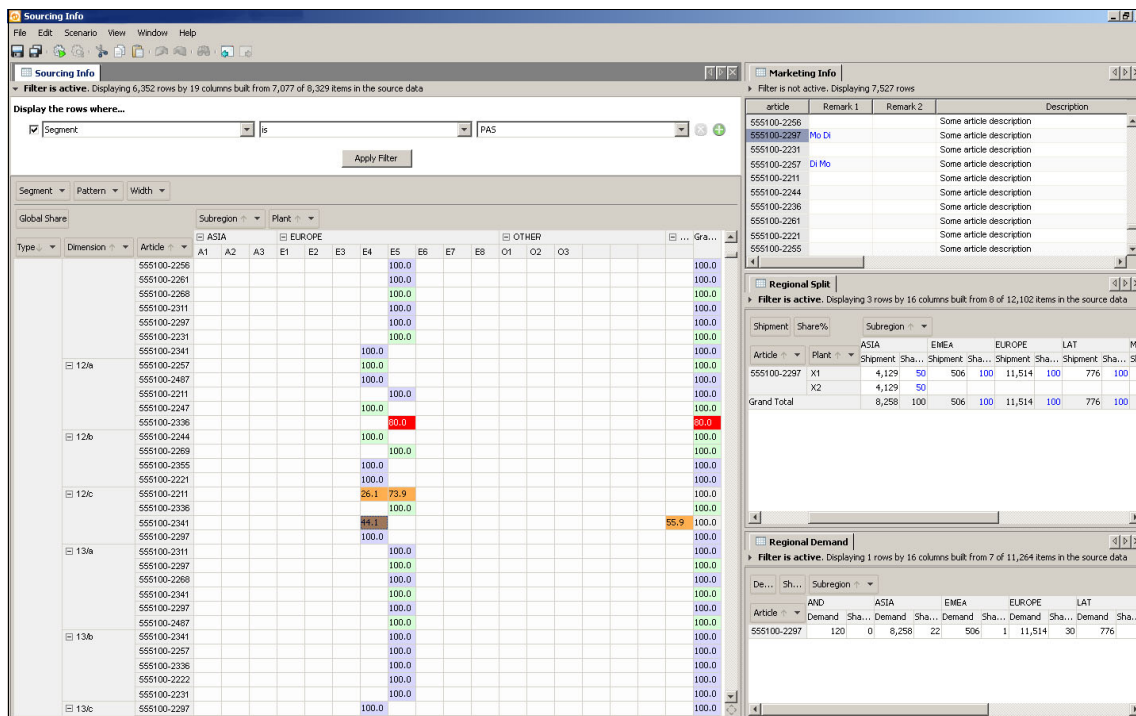


Figure 7-7 Typical window layout of the ODM Enterprise planning cockpit (preliminary)

Most of this user interface is configured through the ODM Enterprise development environment.

However, some features have been added to the application through custom plug-ins that use the various ODM Enterprise product *APIs (application programming interfaces)*:

- Interaction between different windows so that when the user selects a data element in one window, other windows are updated automatically with related data.
- Data transformation methods, for example to transfer data from one repository stage to another.
- Coloring of window parts that is based, for example, on data values.

Figure 7-8 shows a Pivot table, which is a standard product feature, with customized (programmed) extensions that implement data-dependent cell colors.

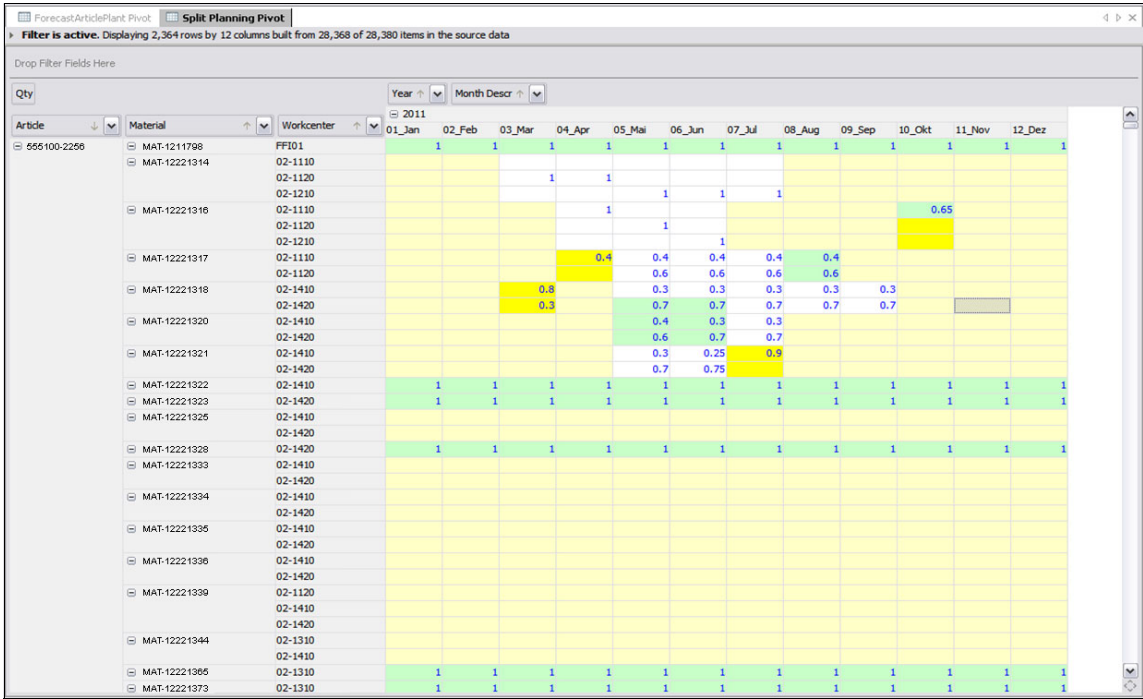


Figure 7-8 Pivot table with customized data coloring

7.5.3 Implementation project

Figure 7-9 provides an overview of the initial project plan.

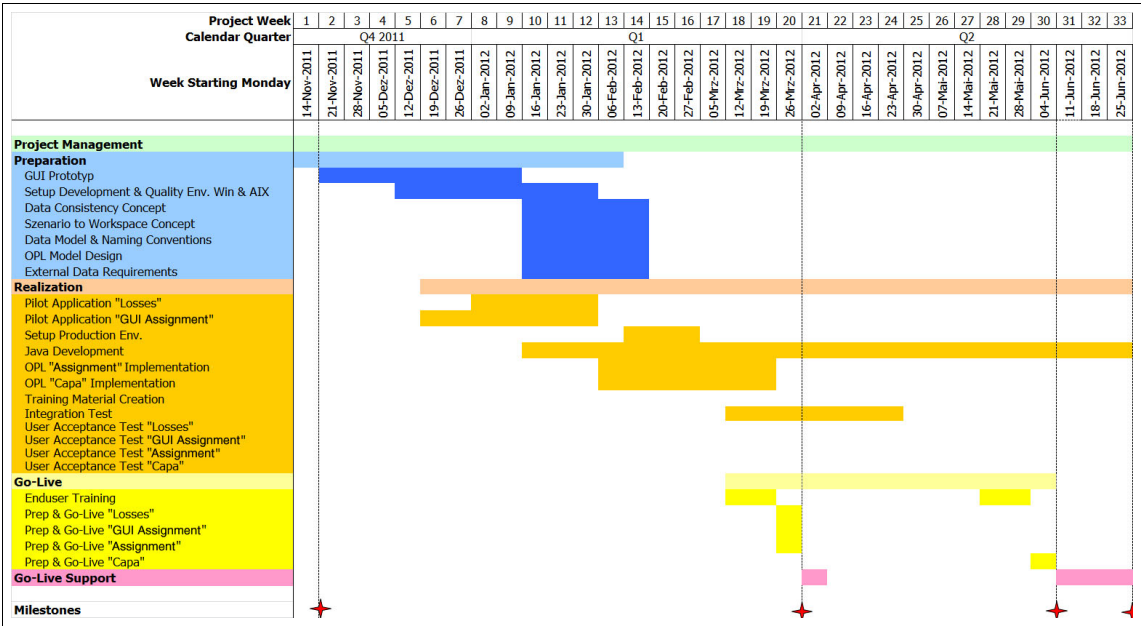


Figure 7-9 Project plan: Seven months from start to roll-out

It essentially consists of the following phases:

1. Preparation, three months
2. Implementation, six months
3. Go-live, three months
4. Go-Live Support, about two months

These phases overlap, so the actual elapsed time from start to roll-out is about seven months.

7.6 Conclusion

This chapter has shown you a comprehensive example: Long-term production planning (LTP) in a fictional manufacturing company.

The mathematical optimization approach, together with a mature software platform, can solve complex models. These models can include aspects such as limited resources, worldwide supply chain network, multiple production locations, many products, detailed material lists, production complexity, and multiple planning periods.

The results and the general approach show some major benefits:

- ▶ The ODM Enterprise based solution provides a tailored and consistent data model.
- ▶ Centralized storage improves data security and allows for collaborative teamwork.
- ▶ The planning cockpit provides the user with efficient interaction, including scenario-based simulation and data comparison.
- ▶ Planning is supported by mathematical functions that *automatically* drive the planning goals towards their theoretical optimum, such as minimal costs.
- ▶ More planning details can be taken into account through fast and automated mathematical optimization.

The optimization run time even for large data sets is usually in the range of a few (5-20) minutes.

- ▶ This process can speed up the creation of planning scenarios. Thus, more scenarios can be evaluated in a shorter amount of time.

Comparing mathematically optimized plans with plans created manually, the optimization can do 5-10% better than manual planning. This improvement can result in *several million dollars of lower supply-chain costs every year*.



Part 3

Appendixes

This part includes the following appendixes:

- ▶ IBM solution landscape for analytics, optimization, and decision support
- ▶ Troubleshooting
- ▶ Additional material



IBM solution landscape for analytics, optimization, and decision support

This appendix summarizes several analytics technologies that complement the prescriptive analytics available through ODM Enterprise. These technologies include the predictive capabilities of IBM SPSS®, the descriptive capabilities of IBM Cognos®, and the process management capabilities of WebSphere ILOG Rules. For each, this appendix addresses the capabilities, the relationship with prescriptive analytics, and how they are used with a case study. This appendix also describes the various editions of ODM Enterprise, which support the alternative deployment architectures that are described in 2.3.2, “Deployment architectures” on page 42.

This appendix includes the following sections:

- ▶ ODM Enterprise overview
- ▶ IBM ILOG CPLEX Optimization Studio overview
- ▶ SPSS
- ▶ Cognos
- ▶ WebSphere ILOG Rules
- ▶ Summary

ODM Enterprise overview

IBM ILOG ODM Enterprise provides an enterprise-scale platform for developing and deploying highly effective optimization-based analytical planning and scheduling solutions for business decision-makers across various industries.

ODM Enterprise-based applications provide all the features that you need to take full advantage of optimization technology. Applications that are built with ODM Enterprise help users to create, compare, and understand planning or scheduling scenarios. They can adjust any of the model inputs or goals, and fully understand the binding constraints, trade-offs, sensitivities, and business options.

ODM Enterprise has these advantages:

- Uses the best available optimization technology

ODM Enterprise embeds the powerful optimization solvers and the modeling language of CPLEX Optimization Studio. These embedded features assure developers and business users of high performance, reliability, and robustness of the solution algorithms.

- Multiple objectives and goal programming

ODM Enterprise-based applications help balance conflicting goals, helping users to change the weights that are associated with each goal or set individual goals. In practice, working with goals is as important as working with costs and constraints.

- Controlled relaxation of constraints

In an ODM Enterprise-based application, over-constrained problems are automatically relaxed during run time, being careful to relax the fewest and least important constraints. This approach ensures that a solution is always found and presented along with information about relaxed preferences or constraints. The optimized solution, with its recommended plan or schedule and attendant metrics, can be easily explored. This feature helps users to understand the optimization model's dynamics and identify solution scenarios.

- Scenario management

To support scenario comparison and reporting, ODM Enterprise-based applications provide complete storage of each scenario. Items that are stored include the input data, scenario parameters, goals, decision variable values, and solution metrics. Data tables provide local storage for external data, so they can be modified without changing data sources. Scenarios can be created, copied, modified, and compared immediately. Furthermore, scenario

metrics that are stored in a database can be invaluable in tracking operational and financial performance over time.

- Collaborative planning

It supports large-scale applications with remote planners and distributed planning processes. Planners can share candidate plans with reviewers through the shared scenario repository. They can also get feedback on a schedule through Microsoft Excel or emails, and facilitate joint data entry and validation. The Scenario Repository ensures data safety during scenario editing through locking.

- Slashing development costs

Making a good optimization-based decision-support application takes time and experience. ODM Enterprise reduces the cost, time, and risk in building the sophisticated behavior that is associated with scenario creation and comparison, controlled relaxation of constraints, and goal balancing. It provides the screens that are needed to review data inputs, operating constraints, preferences, costs, yield assumptions, goals, and solution metrics. Also, usability features like Microsoft Excel integration, cut and paste editing, and online help must be added to meet minimum ease-of-use standards. ODM Enterprise saves time by automatically generating a complete decision-support application with all of these features.

- Refining optimization models

ODM Enterprise also helps business users take part in the iterative development that is needed to perfect an optimization model. With the ODM Enterprise Developer, the actual mathematical models can be developed in step with data access, GUI configuration, server setup, and application integration.

- Scalable enterprise deployment

ODM Enterprise provides client-server based deployment where planners can interactively perform what-if analysis through a rich client (ODM Studio). They can trigger potentially large and complex optimization solving on a remote server. IBM ILOG ODM Enterprise Optimization Server supports concurrent and multi-user invocations from ODM Studio, as well as programmatically scheduled (batch) jobs. In both scenarios, it manages and monitors the optimization engines that produce decision recommendations for planners.

- Separation of corporate and planned data and processing

Data is imported from corporate databases into the ODM Enterprise application. Planners then work with the data to create scenarios and perform “what if” operations to determine the optimal solution. All these activities occur without affecting the corporate data or existing systems. When optimal solutions are found, they can be easily exported to update the corporate data.

- Modularity

The modularity of ODM Enterprise enables use of some modules alone. Java API allows, for example, using the developed application from a custom user interface or integrating into an existing application. Both the central scenario repository and the optimization server features are still available.

For more information about any of this product offering, see:

<http://www.ibm.com/software/integration/optimization/odm-enterprise/>

IBM ILOG CPLEX Optimization Studio overview

IBM ILOG CPLEX Optimization Studio provides an effective way to build efficient optimization models and state-of-the-art applications for the full range of planning and scheduling problems. The Optimization Programming Language (OPL) provides a natural representation of optimization models, requiring far less effort than general-purpose programming languages. The integrated development environment (IDE) makes it easy to evaluate different modeling approaches and to integrate external data. With its built-in development tools, it supports the entire model development process.

CPLEX Optimization Studio provides these features:

- Solving

CPLEX Optimization Studio provides complete access to powerful mathematical programming solvers and constraint programming solvers. You can use robust, high performance algorithms to solve large, real-world optimization problems with the speed required for today's interactive applications. It provides capabilities to diagnose and fix infeasible models. It also provides capabilities to automatically tune performance while precisely controlling the solve process through parameter settings.

- Mathematical programming

Solve linear programming, mixed integer linear programming, quadratic programming, and mixed integer quadratic programming with CPLEX Optimizers.

- Constraint programming

Solve constraint programming for detailed scheduling and other hard combinatorial problems.

- Modeling

Create optimization models and extend optimization algorithms by using the OPL modeling language, the ILOG Concert APIs, or numerous other

interfaces and connectors. Represent mathematical programming and constraint programming models in a natural, descriptive language by using OPL. Use advanced data types that are designed for the special needs of optimization and scheduling problems. Use model/data separation to develop highly scalable applications.

► Developing

Develop, debug, test, and tune analytical decision support applications that are based on mathematical optimization and constraint programming. Use major relational database products and spreadsheets for data sources. Track progress during the solve process. Use the profiler to understand use of memory and time by different phases of execution.

► Deploying

Deploy analytical decision support applications that are based on mathematical optimization and constraint programming. Automatically generate applications that are based on the structure of the underlying mathematical model. Configure tabular and graphical views of data and solutions.

For more information about this product offering, see:

<http://www.ibm.com/software/integration/optimization/cplex-optimization-studio/>

SPSS

SPSS¹ provides the IBM products for Predictive Analytics. Predictive Analytics can help organizations use data in their daily decision-making to substantially improve outcomes. Predictive Analysis does so by enabling organizations in business, government, and academia to perform customer, operational, and risk and threat analytics. These analytics can help predict the outcomes of interactions before they occur, and allow you to act on their insights by embedding analytic results into business processes.

The SPSS product portfolio helps you to *capture*, *predict*, and *act*:

- Capture information: An accurate view of customer attributes, interactions, behaviors, and attitudes.
- Predict behavior and preferences: Bring repeatability to ongoing decision making, and create confidence in your results and decisions.
 - Statistics for deeper insight.

¹ For more information around IBM SPSS visit <http://www.ibm.com/software/analytics/spss/>

- Data Mining for predictive modeling.
- Text Analytics for unstructured insight.
- ▶ Act on results: Maximize the effect of analytics in your operation.
 - Decision Management for business context.
 - Collaboration and Deployment Services for efficient deployment.

Figure A-1 shows the SPSS product portfolio.

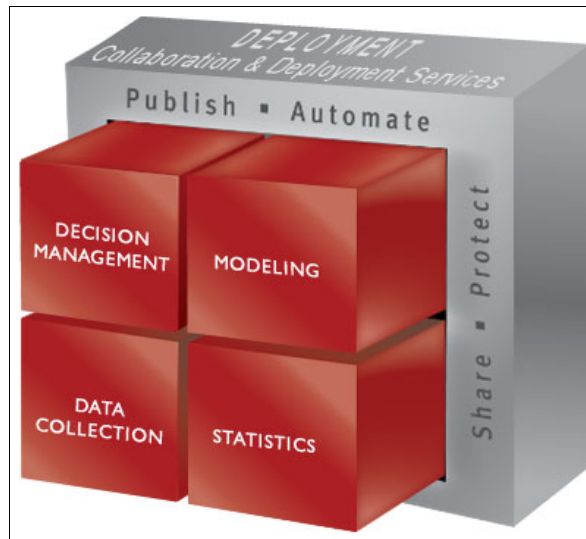


Figure A-1 The IBM SPSS product portfolio

The SPSS product family

The SPSS portfolio consists of four product families which are designed to work together to help organizations meet their research and business goals:

- ▶ The Statistics Family: Consisting of IBM SPSS Statistics and its modules, the most widely used suite of statistical software in the world.
- ▶ The Modeling Family: Including IBM SPSS Modeler Professional for data mining and IBM SPSS Modeler Premium for text analytics, both of which are consistently leaders in the analytics space.
- ▶ The Data Collection Family: Comprising the feature-rich suite of survey research software, IBM SPSS Data Collection, that helps organizations obtain a detailed view of customer attitudes and opinions.
- ▶ The Deployment Family: Including both the platform and the delivery products that are needed to bridge the gap between analysis and action. This family

includes IBM SPSS Collaboration and Deployment Services and IBM SPSS Decision Management.

Additional capabilities are being added to the SPSS family on an on-going basis. Together, these integrated products work together to provide your organization with comprehensive predictive analytics capabilities.

SPSS Statistics

The SPSS Statistics family of products can help you get more sophisticated insights from your data by efficiently analyzing information and delivering comprehensive results. For those situations in which you have well-formulated hypotheses and models, SPSS Statistics includes advanced statistical methods and data management tools for analysts who are researching business problems. It facilitates the collection, preparation, analysis, interpretation, explanation, and presentation of data. It also provides insights into sample data and tools for prediction and forecasting based on the data. Statistics connects to open source tools, such as R and Python, and can interact with code or a GUI. These technologies can help drive confidence in your results and decisions.

By using this suite of analytical tools, many organizations have achieved a competitive advantage:

- ▶ Financial institutions have saved millions by detecting fraud faster
- ▶ Academic organizations have supported the work of researchers and improved their ability to attract and retain the correct mix of students
- ▶ Government agencies at all levels have improved performance and controlled costs.

For example, a police department in the United States uses IBM SPSS Statistics to enhance its crime fighting techniques. It has reduced serious crime by more than 30 percent, including a 15 percent reduction in violent crimes.

Figure A-2 shows the SPSS Modeler GUI.

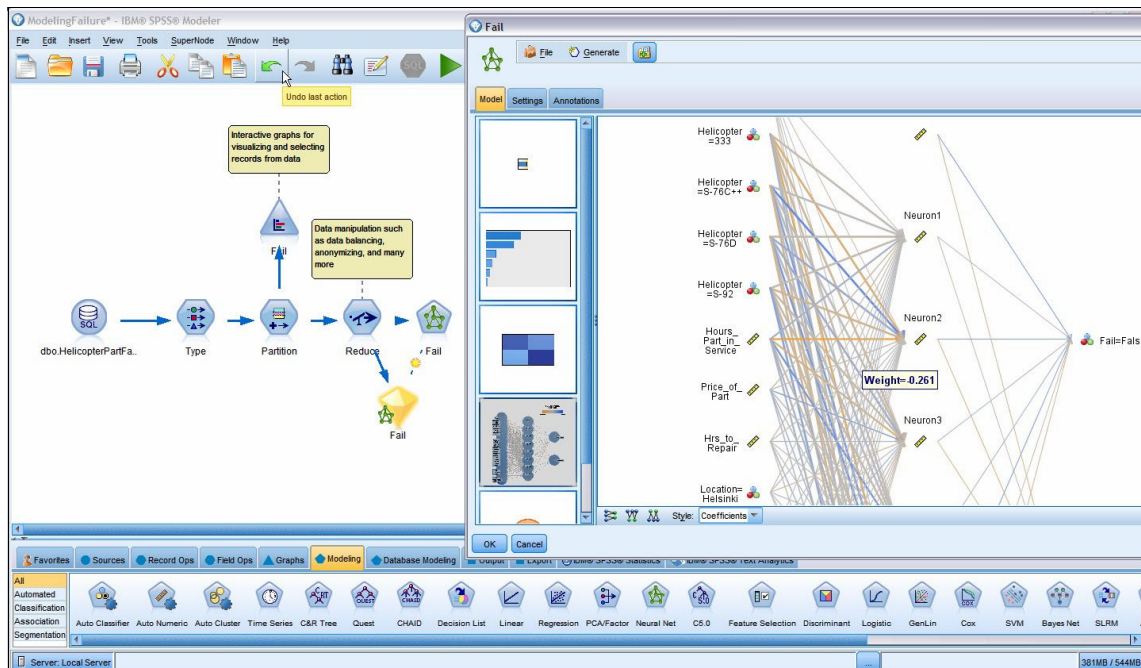


Figure A-2 SPSS Modeler includes advanced, interactive model visualization

SPSS Modeling

When you are modeling your business decisions, you can use the “structured” data in data warehouses with the IBM SPSS Modeling family to discover hidden relationships in your data. These relationships allow you to anticipate the outcomes of future interactions. When you need to explore data to formulate hypotheses and models, SPSS Modeler’s workbench makes it easy to work with data preparation functions. You can use these functions to build analytic streams or jobs, and provide a run time environment for job execution.

SPSS Modeler has an intuitive, visual interface, and connects to a wide variety of data sources beyond data warehouses. It includes a set of mining algorithms that provide insight and prediction. It enables the discovery of key insights, patterns, and trends in data that can be used to make good business decisions. SPSS Modeling brings repeatability to ongoing decision making.

SPSS Modeling provides powerful statistical model-building, evaluation, and automation capabilities. These capabilities help companies acquire customers cost effectively, keep their best customers longer, and sell more to them. One of Japan’s top computer and software retailers used IBM SPSS Modeler to increase sales by nearly 20 percent and triple the profitability of its online store.

SPSS Modeling enables analysts to quickly discover patterns and trends in structured numerical data to model outcomes and make predictions. This process informs business decisions with predictive intelligence. From an intuitive graphical interface, you can incorporate your expertise at every step of the way to create predictive models by using powerful association, classification, and segmentation techniques.

SPSS Modeling can incorporate information from unstructured data sources such as web activity, blog content, customer feedback, emails, and articles, along with structured numerical data. This combination of data creates the most accurate predictive models possible. Advanced natural language processing techniques enable users to extract key concepts, sentiments, and relationships from unstructured data and convert them to a structured format for predictive modeling.

What's a model? (part 1):

Both optimization and statistics use mathematical *models*, but they use them differently. In general, a model consists of mathematical relationships among entities called *variables*.

In an optimization model, the variables represent choices. The objective is to solve for values for the variables that optimize the goals while satisfying the limits. The data measure the influence of the variables on the targets and limits.

A statistical model represents *associations* among the variables. For instance, you might want the association between purchasing certain products and the economic and demographic characteristics of potential customers. In a statistical model, the objective is usually to *infer* these associations, which means to measure their strength based on observed *data*. For example, if the data reveal a strong association between purchasing a luxury car with having household income greater than \$100,000, you can focus promotion toward that economic segment.

Discerning whether an association represents a true relationship among the variables or merely a coincidence because of random chance is the critical issue in statistical modeling. Data always contains random noise that confounds the relationships. The degree of certainty about inferred relationships is called *confidence*. The mathematical methods that are used in statistics aim not only to measure the strength of associations, but also to determine the level of confidence attributable to them.

SPSS Data Collection

IBM SPSS Data Collection captures insight into people's attitudes, preferences, and opinions to improve and direct decisions. It enables you to create and deliver compelling surveys, and integrate feedback results into your decision-making through a centralized and secure framework. It can measure attributes, interactions, behaviors, and attitudes for customers, employees, or constituents. You can augment internal data from your customer information systems with sentiment and opinion data, and use this combination as part of the analytic process. SPSS Data Collection delivers accurate views of customer attitudes and opinions.

For example, a large cable network provider uses Data Collection to more clearly understand which customers are likely to churn, and why. This information allows the company to proactively improve customer retention. As a result, the organization has reduced its customer churn rates from 19 percent to 2 percent.

SPSS Deployment

IBM SPSS Deployment helps ensure that every decision about your customers is the correct decision. The Deployment family can drive results-oriented decisions by making analytics an integral part of your business. The Deployment product line includes several collaboration features that enable the analytics teams to work together and reuse assets. Integrate analytical results into operations to improve business processes, predict outcomes, and deliver results to decision makers across the organization.

For example, a property and casualty insurance corporation uses the Deployment family products to improve the productivity and accuracy of its claims handling process. The company's scoring system for evaluating, routing, and managing claims resulted in 33 percent higher returns for subrogation. This improvement resulted in a subrogation recovery increase of \$10 million a year.

Predictive and Prescriptive Analytics

Predictive and Prescriptive Analytics complement each other in driving fact-based decision processes. Statistical models often work upstream of an optimization application to measure the relationships that are embodied in the optimization model. Here are some examples of how this works:

- **Forecasting: Better inputs to optimization**

Sales forecasts drive production planning and scheduling to determine what, where, and when to produce products to meet customer demand. Customer demand crosses multiple segments with geographic dispersion, and has a high degree of variability. Production planning involves multiple product lines and models, with multiple, geographically dispersed suppliers and production

plants. Coordinating the forecasting process with the planning process increases agility and responsiveness.

- Revenue management: Offer the correct product at the correct time for the correct price

Businesses with fixed capacity, such as airlines, hotels, and cruise ships, must balance property utilization with profitability. Low prices increase occupancy, but can leave money on the table from customers with more willingness to pay. Micro-segmentation of customers can identify individual price sensitivity. Optimization can then allocate rooms or seats among the pricing segments to capture more revenue while keeping unsold space to a minimum.

- Marketing campaign management: Reach your target markets at lower cost

Advertisers with limited budgets face multiple choices about which ads to place in what time slots to reach audiences with high concentrations of their target markets. Customer research and statistical analysis reveals which segments would likely react favorably to ad impressions. Optimization determines the best allocation of budget to buy ad slots with the most favorable audience segments.

- Predictive maintenance scheduling: Anticipate failures to determine appropriate service intervals

Capital-intensive industries need to determine when to service equipment. Servicing equipment too late increases failure likelihood, whereas doing so too soon wastes money on unnecessary maintenance. Focusing service on equipment that is more likely to fail improves both reliability and costs. Statistical analysis can project failure rates, and optimization can determine the best maintenance intervals for each class of equipment.

- Constraint identification: Automatically generate and refine business rules

Product quality depends on numerous factors, and the relationship between these factors and rejection rates frequently must be established empirically through monitoring and observation. Controlling these factors appropriately affects not only the quality of the finished product, but also the cost of production. Statistical methods can identify the influence of these factors and lead to rules for production that assure quality. Optimization can enforce these rules in production scheduling.

Case study: Major retailer

A major retailer uses advanced, multi-dimensional analysis of customer behavior to precisely target marketing campaigns that are based on actual customer responses to the company's value proposition. This process reduces marketing spending, and provides a deeper, better understanding of its customers.

Retailers are faced with a huge challenge in the efficient utilization of their advertising budgets. This retailer needed help optimizing and allocating its marketing budget. It wanted to identify those customers in whom marketing dollars were being over-spent, and those who were not being targeted enough. With this information, the retailer could obtain the optimum return for its marketing spend.

The retailer now uses an advanced customer analytics methodology, called the Customer Foundation, to develop a multi-dimensional, behavioral customer hierarchy. The solution uses statistical models (Feature Vectors) to describe a customer's actual behavior or response to the firm's value proposition. It separates customers into "Action Clusters" whose buying behavior can be more accurately predicted. This process provides a deeper understanding of customer motivation, engagement, and behavior to enhance the client's customer loyalty program value proposition.

This solution delivers these real business results:

- ▶ Optimized the retailer's complex marketing budget, reducing saturative spend by 5-7% of total marketing spending.
- ▶ Allows the retailer to hypothesize and test media budgeting strategies and scenarios, saving time and money to develop and implement strategies while increasing marketing spend effectiveness.

Case study: Large urban school district²

A large metropolitan university uses sophisticated analytics and forecasting tools in a first-of-a-kind solution to dramatically improve energy efficiency in city public schools. The system increases students' awareness of sustainability issues and models a greener urban future.

Recognizing that buildings are responsible for approximately 40 percent of energy consumption and greenhouse gas emissions in developed countries, the city wanted to improve sustainability. The goal is to shave 30 percent from its municipal building carbon footprint by 2017. The 1,200 public schools, typically older buildings that were not designed for energy efficiency, could serve as a test bed to jump-start its sustainable buildings initiative. The city turned to the university for technology assistance in entering information into EnergyStar Portfolio Manager. The university and IBM created a sophisticated toolset and web-based interactive dashboard to assess, track, forecast, simulate, and optimize energy consumption across the entire district.

² The City University of New York (CUNY), IMC14721-USEN-00,
http://www.ibm.com/software/success/cssdb.nsf/CS/KJON-8NQSWH?OpenDocument&Site=spss&cty=en_us

Improving energy efficiency and identifying and prioritizing opportunities for energy savings through building retrofits under budget constraints has become increasingly important. The city education department implemented a comprehensive analytics solution for analyzing energy consumption, efficiency, and greenhouse gas emissions, and for optimizing energy use. The solution provides energy-demand forecasts. It gives the school district the ability to generate what-if scenarios to estimate the effect of energy-saving initiatives and behaviors such as equipment upgrades and temperature set point control strategies. The city implemented training software, developed for the project, to give building engineers and operations staff the resources to carry out energy usage assessments and scenario evaluations.

This solution delivered these real business results:

- ▶ One high school, a lead participant in the program, reduced energy consumption by 35.1% and prevented 216,061 lbs. of carbon dioxide from being released into the atmosphere.
- ▶ Enhanced and supplemented the Science, Technology, Engineering, and Mathematics (STEM) curriculum with information gained through the efforts to reduce energy consumption.
- ▶ Trained property managers, building engineers and operations staff with a web-based interactive dashboard and tools to help investigate opportunities for reducing the carbon footprint and greenhouse gas emissions of buildings.
- ▶ Provided property managers and facility staff with what if simulation capability to estimate the effect of energy-saving initiatives and behaviors to drive improvements in energy use.

Cognos

IBM Cognos provides products for descriptive analytics. Cognos products help users explore information, analyze key facts, collaborate to gain alignment with decision makers, and plan and act with confidence for better business outcomes. Cognos equips users with information they need to help their organizations perform, including reports, analysis, statistics, dashboards, what-if analysis, plans, and budgets. Cognos offers a portfolio of integrated business intelligence and performance management solutions. These solutions support the way users work and can help your organization to quickly satisfy urgent business requirements and grow your solution over time to address future needs.

For more information about IBM Cognos, see:

<http://www.ibm.com/software/analytics/cognos/>

Cognos product capabilities

Cognos products include the following capabilities:

- ▶ *Reports* and report authoring solutions that can help organizations realize substantial value with these features:
 - A unified user experience for business report authoring and consumption, giving business users the freedom to seek business insights on their own.
 - The ability to foster a culture of pervasive fact-based decision-making with reports that can be made available in various ways. These methods include on mobile devices, in Microsoft Office applications, embedded and in-process business intelligence (BI), and interactive use when disconnected from the network.
 - The ability to create and deploy a single report that can be run in multiple languages and outputs.
- ▶ *Dashboards* provide business users a full view of historical BI alongside real-time data and future probabilities for improved decision-making in a unified workspace. Dashboards enable users to perform these tasks:
 - Assemble, personalize, analyze, and interact with information in a single place for working with BI and analytics.
 - Access virtually all types of data: Historical BI, real-time operations data, probabilities, and so on.
 - View performance metrics or other information in tables, or as graphs and charts.
 - Support their personal decision-making styles.

Figure A-3 depicts a typical overview of the IBM Cognos Dashboard user interface.



Figure A-3 Cognos dashboard

- ▶ *Analysis* tools and software can help analyze all kinds of information to better understand and predict performance and make decisions confidently. The analysis capabilities of Cognos, depicted in Figure A-4 on page 278, enable users to perform these tasks:
 - Assess the immediate business situation.
 - Engage the correct teams from across the enterprise for more inclusive and holistic decision-making, evaluation, opportunity identification, and assessment.
 - Merge external data with dimensional and relational sources without the need for IT involvement.
 - Gain insight and confidence with facts that are readily available to all decision makers.

- Support hypotheses and maximize return on investment with data analysis and testing.
- Identify key customer demographics.
- Apply statistical methods to improve quality and reduce costs.

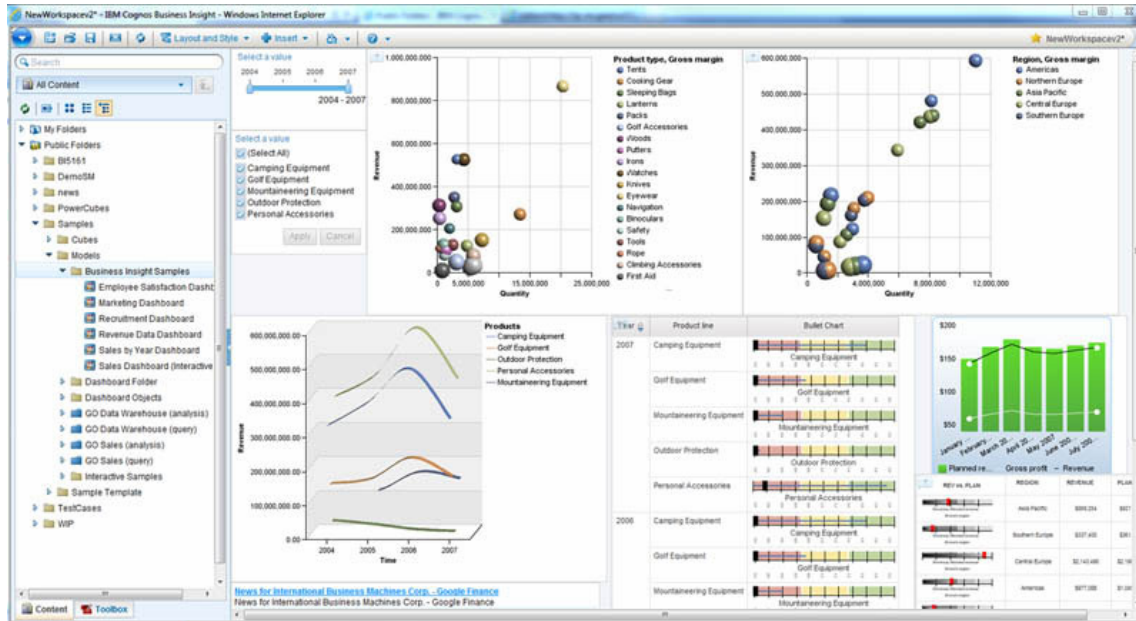


Figure A-4 Cognos analysis capabilities

- *Collaboration* capabilities empower decision makers and analysts to connect with each other, establish dynamic decision processes, and maintain central information hubs to support group decisions. These capabilities consist of two key components:
 - IBM Connections for integrated access to online, collaborative tools that enable users to work on decisions in teams.

For more information about IBM Connections, see:

<http://www.ibm.com/software/lotus/products/connections/>

- A strong corporate memory with the ability to search a single, historical record of how issues were resolved and decisions were made.

- ▶ *Modeling* capabilities enable decision makers to explore “what-if” scenarios. The modeling capabilities of Cognos include these features:
 - A guided modeling environment that eases development and deployment of planning, analysis, and forecasting solutions.
 - Online analytical processing (OLAP).
 - A choice of interfaces to accommodate a broad range of work styles.
 - A single, integrated platform that consolidates data from multiple sources such as personal data files, spreadsheets, ERP, and other enterprise applications.
 - Distributed architecture for greater scalability and interactivity. Choose centralized or distributed deployment methods for modeling activities, depending upon the level of interactivity needed.

What’s a model? (part 2):

IBM Cognos illustrates another role of mathematical modeling in analytics. A model consists of mathematical relationships among entities called *variables*. In descriptive analytics, you sometimes want to see how the variables evolve over time as input parameters change. This use of a model is called a *simulation*.

For example, you can create a pro forma budget to track expenses, income, and profits as a function of sales. The mathematical relationships among these variables are specified by formulas, such as might populate the cells in a spreadsheet. By varying the sales forecast, you can see how the organization’s financial performance might evolve. Different sales scenarios give different financial results, a process that is called a “what if” analysis.

Both ODM Enterprise and SPSS Modeler also provide “what if” analysis capabilities among their modeling tools.

Descriptive and prescriptive analytics

Descriptive and prescriptive analytics frequently work together to enable decision makers to gain insights to a business situation and to recommended decisions that respond to it. For a specific example of how ODM Enterprise and Cognos Business Intelligence can work together to solve a business problem, see *IBM ILOG ODM Enterprise and Data Integration*, REDP-4740.

Case study: Major food processor

The company in this example³ is one of the largest food processing operations in the United States. It can have significant influence on the price of its raw materials if it could better predict customer demand. The company needed to be able to better respond to an unpredictable market and control both its materials costs and production output to maintain wanted margins. It developed a margin optimization solution to give real-time insights into agricultural commodity market conditions. With insights into both demand and supply, the company can respond to the volatile market, segment customers, optimize plant capacities, and ultimately increase margins.

The commodities markets are inherently risky and complex. They are influenced by the weather, transportation costs, regulations, global politics, and many other factors. This company transformed its business model by taking control of its market by automatically adjusting pricing and production capacity at its 21 processing plants in real time to respond to changing conditions. The solution brought together production and sales data from existing ERP and supply chain systems with external information provided by government agencies and commodities markets. This combination creates unprecedented views of the past and a window into the future. Using powerful models and algorithmic engines, the company is now able to translate a vast array of market and production information into better recommendations for deal makers.

The margin optimization solution uses descriptive, predictive, and prescriptive analytics to automate pricing changes. This solution enables decision-makers to make more strategic decisions about pricing and production capacity. This process allows them to influence how the company responds to changes in market conditions on a global scale, such as droughts or trade embargoes.

This solution delivers these real business results:

- ▶ 100% payback in less than three months.
- ▶ Increased plant utilization by 3 to 5 percent over 12 months.
- ▶ Improved profit margins by product, by customer, and by plant.
- ▶ Decreased raw materials costs through higher purchase volume and coordination among production facilities.
- ▶ Improved ability to capitalize on emerging business opportunities without risking “oversold” production capacity.

³ IBM Smarter Planet® Leadership Series, ConAgra Mills, ODC03208-USEN-00, 2011, http://www.ibm.com/software/success/cssdb.nsf/CS/SSA0-8R3396?OpenDocument&Site=cognos&ty=en_us

WebSphere ILOG Rules

A *business rule management system* (BRMS) enables organizational policies to be defined, deployed, monitored, and maintained separately from application software. The policies include the repeatable decisions that are associated with those policies, such as claim approvals, cross-sell offer selection, pricing calculations, and eligibility determinations. By externalizing business rules and providing tools to manage them, a BRMS allows business experts to define and maintain the decisions that guide systems behavior. The process reduces the amount of time and effort that are required to update production systems. It also increases the organization's ability to respond to changes in the business environment. WebSphere ILOG Rules provides the ability for non-technical business users to be directly involved in business rules management. This feature enables flexible decision automation for applications and processes that are subject to complex, variable, and evolving business rules.

For more information about WebSphere ILOG Rules, see:

<http://www.ibm.com/software/websphere/products/business-rule-management/>

Business rules management

Business rules implement *if-then-else* decision logic. For example⁴, consider opening a bank account. Decisions that are made when executing the process for a personal banking customer might include the following rules:

- ▶ If this is a high net worth client, then allocate a Platinum account with added benefits; else
- ▶ If the client is a student, then offer a free financial health check; else
- ▶ If the client is brand new and today's date is in the advertised special offer period, then set interest rate at 2% for six months; else
- ▶ And so on.

Business processes in most organizations have many such rules. In established systems, the rules are spread out across multiple systems and applications, and are embedded in software code. That makes it difficult to manage and change the rules in response to changing business conditions. It also requires considerable effort and specialized technical skill to locate and change the code. Also, there is always the risk that some instance will not be found and changed,

⁴ To download the Lustratus Research Paper, *Better decisions with business rules and events*, STC/LR/44227386/V1.0, 2011, see:
https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=swg-ilog&S_PKG=dm_lustratus_paper&S_TACT=109KA2TW&S_CMP=web_ibm_ws_ilg_brms_hero_what-is-ov

leading to numerous problems in execution. Business users have a hard time implementing changes in business logic in a timely and efficient manner.

A BRMS makes the decision logic accessible to business users and manages the authoring, review, and editing of the rules systematically and centrally. Deployment of updated business rules can occur in real time, immediately implementing the wanted changes in policies. This process improves flexibility and agility in operations, enabling business users to respond quickly and effectively to threats and opportunities with new initiatives. A BRMS can also maintain a repository of rules that documents how decisions are made, providing a record for validation and compliance verification.

Comparison: The two technologies, *business rules management system* and *optimization*, offer different, complimentary approaches to decision automation. Business rules implement *if-then-else* decision logic. This technology automates decision making by quickly classifying requests and events and matching them to specific actions. Each such trigger is handled independently of the others, and so a decision in one case has no effects on any other case.

Optimization handles complex *trade-offs* among different targets and limits, so choices are linked together by their common resource limits. Analyzing such trade-offs requires more complex algorithms than can be implemented with if-then-else logic.

Look at the retail banking example that was addressed previously. A business rule says that “if this is a high net worth client, then allocate a Platinum account with added benefits.” How does the bank define “high net worth?” Is it half a million dollars, a million dollars, or what? In setting this limit, the bank must consider the value of such a client’s business versus the cost of retaining the client. These costs might include providing free services or discounted loan fees and interest.

One way to address this question uses a combination of predictive and prescriptive analytics. Optimization addresses the choice of where to set the threshold for “high net worth” to achieve the target of maximizing the present value of customers’ business. Predictive analytics determines key parameters such as the expected retention rate of customers and the amount of business they would do with the bank.

Business rules and prescriptive analytics

Using the two technologies, BRMS and optimization, together can combine the strengths of both. Two use cases illustrate the possibilities.

In the first use case, business rules preprocess data before using it in an optimization application. Data that is gathered from multiple sources frequently has errors in it that can corrupt the optimization model, leading to unreliable results. For instance, a utility company that is implementing a smart grid application applies data from smart meters and other sensor devices. It uses these data to determine when and where to apply demand response measures that control customers' electricity demands. The optimization model chooses actions that minimize costs to serve peak loads while maintaining reliable service to customers. However, the data frequently exhibits inconsistencies and other anomalies that result from noisy collection and transmittal technology. The utility company developed a set of rules for spotting these errors and correcting them before applying the data to the optimization model.

In the second use case, an optimization model sets policies that then parametrize a set of business rules that actually execute the policies. The banking example that was addressed previously represents this use case. In another example, a manufacturer uses optimization to plan production on a weekly basis. It collects the orders received each week. The company then uses these data to revise its sales forecast (a predictive analytics application). It then uses the optimization application to choose where and when to produce the goods to minimize production and delivery costs. The optimization model represents the limits on these decisions that are based on the capacity of its factories, the available raw materials and labor, and the logistics network. The manufacturer's sales force needs to be able to quote delivery dates for new, incoming orders. Because the optimization application must aggregate many orders to effectively measure trade-offs, it cannot be run for every new order. Instead, the manufacturer uses the results of the most recent optimization solve to set parameters of business rules that the sales people use for customer quotes. The business rules therefore enable a business process that uses optimization periodically to supporting decision making on a shorter time scale while assuring conformity with optimal policies in the longer term.

Case study: Leading global logistics service provider

This leading global logistics service provider⁵ serves customers in North America, Europe, and Asia. The company combines strategic, commercial, operational, and technology services that enable its customers to reduce costs and receive the best possible service from transportation providers. A sophisticated, global web-based infrastructure, proven operational processes, and a vast network of logistics service providers differentiate it from its competitors.

⁵ Odyssey Logistics & Technology Corporation increases agility, WSC14388-USEN-00, 2012, http://www.ibm.com/software/success/cssdb.nsf/CS/RNAE-8SVEWY?OpenDocument&Site=corp&cty=en_us

Unlike many companies in the logistics industry, the company does not own a transportation fleet. Rather, it helps customers manage their logistics processes and choose the best transportation option given each customer's unique needs. The best option can involve multiple modes of transportation, resulting in a complex collaboration among several organizations that include customers and logistics service providers. In addition, choosing the correct option for each customer has grown increasingly challenging as the company has expanded into new markets in Europe, the Middle East, and the Asia-Pacific region. For each customer, the company must consider region- and language-specific issues, and give customers the ability to customize the existing business processes. Geographical data, rate calculation, transactional optimization, and booking processes can vary from one country to the next.

“Switching to a task-based paradigm has reduced the learning curves of both our internal and external users, knowing exactly what to do next and taking action only when needed. This has increased productivity upwards of 300 percent for certain functions, while at the same time has reduced errors by over 10 percent, making this very powerful technology,” says a senior executive of the company.

The many factors that are involved in creating custom processes for each customer and region have increased transactional and master data, which the company must manage effectively. To better meet its global customers' needs, the company wanted to develop the next generation of its global logistics platform. This platform is the technology it uses to create business rules and manage the many integrated components of its customers' supply chains. The company sought to meet these goals:

- ▶ Improve its operational efficiency
- ▶ Automate as many manual processes as possible
- ▶ Improve its ability to measure key performance indicators (KPIs)
- ▶ Gain greater flexibility in its business process management

“We study the big picture of logistics and we come back to each customer with a flexible plan that adapts over time,” says another company executive. It is not necessarily a static plan and that is why the correct technology, total visibility into the system and the tight control matter. Because after we put a plan in place, we have to monitor it constantly, measure it and adjust it to improve on it for the long term. Because customers might want to track different KPIs—such as how often deliveries are made, how many are made in the course of a day, or whether or not shelves are stocked in a timely manner—having a tractable platform is key to our reporting.”

When a store runs out of popular products, logistics challenges are often responsible. For example, foods and beverages cannot be delivered to grocery stores when the manufacturer lacks the raw ingredients to make them. Market volatility is an ongoing issue for many of these manufacturers. It is especially

important when it comes to managing the costs associated with transporting the raw materials required to make products and then transporting the finished goods to retailers. The new platform helps the company make sense of the continuously shifting transportation landscape. More importantly, it helps identify cost savings and strategic adjustments to improve service and assist manufacturers in bringing their products to market.

The company applied process innovation to give its customers an intelligent, adaptive, and optimized extended supply chain that is based on demand. Process innovation is the practice of synchronizing processes across business functions and managing for the cumulative result. Rather than developing the new system itself, the company chose to build it using a suite of IBM products that included business process management, BRMS, and optimization.

The IBM applications provide greater process automation that changes the operational paradigm. The new paradigm allows the system to guide the user to the next task rather than the user having to guess the correct choice. The IBM applications include preconstructed processes that are based on sets of common activities. These processes enable the company to quickly assemble custom processes that meet its customers' needs. The company uses business rules to manage rate structures, routing guides, and carrier information, which are the aspects that change most frequently. This process helps to further reduce the complexity of the logistics management process.

The BRMS software enables the company to create decision tables, including rate structures, possible routes, and available carriers. Then, by using the information in its system, the company creates rules that enable better, smarter, automated decision-making and that can be adjusted quickly as needed. The applications also enable the company to monitor processes and track shipments in-transit.

The company has found that the new solution greatly increases its flexibility and responsiveness. For example, changing business rules used to take weeks but, with the new system, a similar change might take a few days or even a few hours. The appropriate business rule, which is written in simple natural language that is easy to understand, can be edited even by business users. In addition, the new platform provides greater automation, which helps reduce human errors.

"The system provides capabilities of directed work, so we're less dependent on the human factor because if a task is not done in time, the situation could be escalated by the system," says the executive.

The company's customers benefit from the new platform as well because the increased agility and responsiveness help customers improve supply chain efficiencies and keep costs down. For example, the company can now measure work process performance, using that information to fine-tune decisions and

interactions between the company staff, customers, and carriers. These capabilities have provided the company with the following benefits:

- ▶ Meet or exceed customer KPIs in areas of on-time deliveries, invoice turnaround, and error-free shipping.
- ▶ Implement best-in-class freight audit processes globally.
- ▶ Provide visibility into shipments that are moving on any mode to anywhere in the world.

“The new platform has increased productivity upwards of 300 percent for certain functions, while at the same time has reduced errors by more than 10 percent, making this very powerful technology,” says the senior executive.

The IBM solution also freed the company developers to focus on developing and improving business rules and logic. “The solution provides scalability, performance, and security. We didn’t need to worry about those issues because they were delivered right out of the box,” says the executive.

Summary

Table 7-8 summarizes the multiple IBM cross-cutting analytics products and their capabilities.

Table 7-8 IBM analytics products

Product Family	Type	What it offers	When to use
ILOG Optimization 	Prescriptive Analytics	▶ ILOG ODM Enterprise ▶ ILOG CPLEX Optimization Studio	▶ Multiple decisions ▶ Minimize or maximize goals ▶ Resource limits ▶ Policies and rules ▶ Trade-offs ▶ What-if analysis
SPSS 	Predictive Analytics	▶ SPSS Data Collection ▶ SPSS Statistics ▶ SPSS Modeler ▶ SPSS Decision Management	▶ Forecasting and prediction ▶ Hypothesis testing ▶ Data mining ▶ Associations ▶ Confidence levels
Cognos 	Descriptive Analytics	▶ Cognos Enterprise ▶ Cognos Express ▶ Cognos Insight ▶ Cognos TM1®	▶ Ad hoc queries ▶ Charting ▶ Reporting ▶ Dashboards ▶ What-if simulation
WebSphere ILOG Rules 	Business Process Management	WebSphere Operational Decision Management (JRules)	▶ If-then-else logic ▶ Policies ▶ Business events



Troubleshooting

This appendix addresses some general concepts and recommendations about using ODM Enterprise. It also describes more specific tactics that can help you diagnose possible issues during installation, development, and deployment.

This appendix is targeted towards developers of ODM Enterprise applications.

This appendix includes the following sections:

- ▶ General concepts and guidelines
- ▶ Troubleshooting tips

General concepts and guidelines

This section provides technical knowledge to ODM Enterprise users to customize ODM Enterprise, including memory management and application logging. It also includes some guidelines that are based on the experiences of many users.

How ODM Enterprise uses memory

ODM Enterprise is based on Java. When you start an ODM Enterprise application in ODM Studio, a Java virtual machine (JVM) is started to support ODM Studio (for GUI, views, and scenarios). However, the memory that is used by the JVM is not available for optimization engines (CPLEX Optimizer and CPLEX CP Optimizer) running in the same process.

Before getting into technical details, the following terms must be defined:

- ▶ Java heap

Java heap is the memory that is managed by the JVM directly for data and objects created and garbage collected at run time.

In ODM Enterprise, ODM Studio mainly uses the memory on the Java heap.

- ▶ Native heap

Although Java applications run in the virtualized environment of the JVM, the JVM itself is a native program that runs as an operating system process. All virtualized resources, including the Java heap and Java threads, are stored in native memory along with the data used by the JVM as it runs. This memory is called the *native heap*.

- The native heap contains resources for the JVM itself. These resources include the Just-In-Time compiler, classes, and classloaders, as well as those for the Java application.
- Java supports calling native code through the Java Native Interface (JNI). The memory that is used by JNI code is on the native heap, so it is separated from the Java heap.

For ODM Enterprise, the optimization engines (CPLEX Optimizer and CPLEX CP Optimizer) are native libraries that are started by ODM Enterprise through JNI. Therefore, the memory allocated for ODM Studio on the Java heap is not usable by the optimization engines.

- ▶ Process address space

The memory in which the Java process runs is called the *process address space*. This memory area is shared by both the Java and native heaps.

The following factors affect ODM Enterprise memory usage:

► Limits of process address space

ODM Enterprise is available in two application types: 32-bit and 64-bit. You can use ODM Enterprise in 32-bit processes or 64-bit processes, depending on your hardware and operation system.

- The process address space of a 64-bit processes has a theoretical limit of 16 EB (16 million TB) (2^{64}). For 64-bit process, the amount of memory a process can use is only limited by the amount of system memory (including both physical memory and virtual memory).
- The process address space of 32-bit process has a theoretical limit of 4 GB. The space available to the Java heap and native heap is further limited by the operating system. The operating system also requires memory to provide the virtualized environment on which the JVM runs. Therefore, the effective process address space limit of 32-bit processes is somewhere between 2 GB - 3.5 GB depending on the operating system. This limitation exists even if the amount of system memory is more than 4 GB.

Remember: System administrators might impose resource limits, including a memory limit. If you are experiencing unexpected resource limitations, contact your system administrator.

► Sizing the heap

You can use the JVM option `-Xmx N` to adjust the maximum amount of memory that is reserved for the Java heap. The value `N` must be a multiple of 2 and must be greater than 2 MB. Append the letter `k` or `K` to indicate kilobytes, or append the letter `m` or `M` to indicate megabytes. The default value is 64 MB. The upper limit for this value depends on the process address space minus the operation system processor usage. For more information about adjusting `-Xmx` for different use cases, see “Adjusting memory usage for ODM Enterprise” on page 311.

- The default value of `-Xmx` is 64 MB, which might be too small for realistic data sets.
- There is a finite amount of memory a process can use, and that memory is shared between the two heaps. Therefore, do not try to increase the value for `-Xmx` option too aggressively.

The relationship between the memory concepts can be summarized by this formula:

$$\min(\text{process_address_space}, \text{system_memory}, \text{resource_limit}) > \text{Java_heap_size} + \text{optimization_engine_memory_usage}$$

Consideration: The JVM option `-Xmx N` specifies the *maximum heap size*, not necessarily the one that is finally allocated. To avoid memory warm-up cycles, set the minimum heap size to a value slightly above what will be used.

You can use tools such as JConsole to find the currently allocated heap of a JVM.

General information about Development IDE and ODM Studio

In the developer edition of ODM Enterprise, there are two graphical user interfaces (GUIs).

- ▶ *ODM Enterprise Development IDE* is a self-contained development environment that is based on Eclipse. You can develop your applications in the Development IDE, including configuring, deploying, and running ODM Enterprise applications. This is the typical GUI for application development.
- ▶ *ODM Studio* is a Java front end that can be used to run ODM Enterprise applications. ODM Studio does not have development features, and is typically used for deployment purposes.

The two GUIs use identical sets of libraries, but are stored in different locations. The root installation folder of ODM Enterprise `<install_dir>\Developer\` contains folders named `bin` and `lib`. The `bin` folder includes a set of CPLEX Optimization Studio runtime libraries, and the `lib` folder includes a set of ODM Enterprise library files. The files in these two folders are only for ODM Studio and ODM Enterprise API applications.

The ODM Enterprise Development IDE has a duplicate copy of these files in two different folders:

- ▶ The Development IDE includes a set of ODM Enterprise library files in the subfolder:
`<install_dir>\<edition>\ide\eclipse\dropins\oplide\plugins\ilog.odms.ide.opllang`

Similarly, names of Eclipse plug-in directories are usually extended with version information, such as:

`<Install_dir>\<edition>\ide\eclipse\dropins\oplide\plugins\ilog.odms.ide.odm_3.6.0.20111117-1648`

- The Development IDE includes a set of CPLEX Optimization Studio runtime libraries in the subfolder:

```
<install_dir>\<edition>\ide\eclipse\dropins\oplide\plugins\ilog.odms  
.ide.odm
```

The names of Eclipse plug-in directories are usually extended with version information, such as:

```
<Install_dir>\<edition>\ide\eclipse\dropins\oplide\plugins\ilog.odms  
.ide.opllang.win32.win32.x86_12.4.0.20111106-0916
```

Requirement: When you want to customize your environment, make sure to customize the files for the GUI you intend to use. The Development IDE and ODM Studio both use files from different folders.

As indicated earlier, ODM Studio can be started from the ODM Enterprise Development IDE. In this case, ODM Studio is started in the same process as the one that is hosting the Development IDE. This situation is different from starting ODM Studio directly in the following aspects:

- ODM Studio that is started from the Development IDE uses the set of CPLEX Optimization Studio runtime libraries and ODM Enterprise library files from the Development IDE subfolders. To customize the development environment, such as adding database driver files or modifying the ODM Enterprise log configuration file `odmlogging.properties`, make sure to modify the correct set of files.
- After you close the ODM Studio started from within the ODM Enterprise Development IDE, the Development IDE might still lock or buffer some files of the ODM Enterprise applications. This error can cause unexpected behavior. For most accurate results, especially if you develop custom code, deploy the ODM Enterprise application first and start the ODM application in ODM Studio externally.

Tip: You can program custom ODM Enterprise features using ODM Enterprise APIs. If your ODM Enterprise applications do not behave as expected, verify the behavior by using the following approach:

- ▶ Try to deploy the ODM Enterprise application to determine whether the Development IDE has locked some files for the ODM Enterprise application.
- ▶ Restart the Development IDE to release all file locks. Then, rerun the ODM Enterprise application from within the Development IDE.
- ▶ Your workspace can become unstable because of Eclipse limitations. For more information about cleaning up the workspace, see “Cleaning up the ODM Enterprise Development IDE workspace” on page 322.

- ▶ An advantage of starting ODM Studio from within the Development IDE is that you might see more messages reported in the Development IDE. These messages are generally more relevant to the runtime errors than the messages in the general ODM Enterprise logs introduced in “ODM Enterprise logs guidelines” on page 295.

Another advantage is that you have access to the development features in the ODM Studio.

There is a nonstandard way to start ODM Studio from within the Development IDE in a new process. For more information, see “Debugging your custom code based on ODM Enterprise APIs” on page 324.

How ODM Enterprise Optimization Server works

When a scenario of an ODM Enterprise application is submitted to the Optimization Server, the Optimization Server creates a subprocess that optimizes the scenario. Users can submit multiple jobs for the same ODM Enterprise application. The subprocess also sends its progress to the server components. The server components display the status of the subprocess to the Optimization Server console, and pass the status information to the client side (ODM Studio or ODM Enterprise API-based applications).

This design has the following advantages:

- ▶ Compared to running the jobs within an application server process, each job can fully use the whole process address space. For 32-bit applications, jobs do not have to share the already-limited process address space with the application server or other jobs. This design improves the capabilities of the 32-bit version of ODM Enterprise to handle larger data sets. However, 32-bit

applications have limited process address space, so you might need to use a 64-bit ODM Enterprise environment if the data size is very large.

- If a subprocess crashes, it will not bring down the application server or the other jobs. The server component notices this crash and notifies the client side. The crash of one job will not affect the application server, ODM Enterprise server components, or the other jobs.

When a job is running, client applications (like ODM Studio) can interact with ongoing jobs. For an ODM client application, the users of ODM Studio can accept the current solution that was already found by the optimization, relax more requirements, or stop the optimization.

Interactions with ongoing jobs from clients go through the Optimization Server management application. The management application sends a notification to all servers that run the processor application. This notification is sent through a dedicated Java Message Service (JMS) topic that is configured during the installation of the Optimization Server. An incorrect configuration can prevent these interactions from completing successfully. With WebSphere Application Server, ensure that all processor nodes are registered as members of the message bus that is used by the Optimization Server topic.

ODM Enterprise logs guidelines

ODM Enterprise displays error and warning messages in ODM Enterprise Development IDE and ODM Studio to help you debug your application. Besides these error and warning messages in the GUIs, you can get more detailed information from ODM Enterprise logs.

Logging in ODM Enterprise can be enabled and adjusted on the client side (ODM Enterprise IDE, ODM Studio, ODM Enterprise Java applications, and so on) or the server side (ODM Enterprise Optimization Server and ODM Enterprise Data Server). You can also use a custom set of log settings for each ODM Enterprise application, or for each user.

Enabling logging on the ODM Enterprise client side

Logging in ODM Enterprise is controlled by the `odmlogging.properties` file in the ODM Enterprise directory. For example, in the ODM Enterprise Developer Edition (except the settings that are used by the ODM Enterprise Development IDE) the file is at `<install_dir>\Developer\lib\odmlogging.properties`. The file specifically for ODM Enterprise Development IDE is

<install_dir>\Developer\ide\eclipse\dropins\oplide\plugins\ilog.odms.ide.odm\odmlogging.properties. This file follows the syntax that is defined in the Java Logging API, which can be found at:

<http://java.sun.com/j2se/1.5.0/docs/guide/logging/overview.html>

ODM Enterprise logs can be displayed in the following ways:

- ▶ The default configuration that is used by ODM Enterprise reports warning and error messages in a file that is prefixed with odm.log in the ODM Enterprise directory. For example, the log file in ODM Enterprise Developer Edition (except the logs for ODM Enterprise Development IDE) can be found in <install_dir>\Developer. The file specifically for ODM Enterprise Development IDE can be found in <install_dir>\Developer\ide\eclipse\
- ▶ When you start ODM Studio in a console using the -debug option, ODM Enterprise reports information, warning, and error messages directly in the console. Figure B-1 on page 297 shows the console window of the Nurse sample. The Visual Studio 2008 Command Prompt window shows the full command that uses the -debug option to open the console. The C:\ILOG\ODME36\Developer\binx86_win32\odmplanner.exe -debug” window is the ODM Enterprise console. You can find more details in the console than in the Development IDE and ODM Studio.

Explanation: The log files catch only the information that is sent through the Java Logging API. Information directly relayed to the console cannot be captured in the ODM Enterprise log files. Messages from ODM Enterprise are always sent by using the Java Logging API. However, CPLEX Optimization Studio usually relays the messages directly to the console, or to CPLEX Optimization Studio log files. Therefore, open the console window if the log file does not contain useful messages.

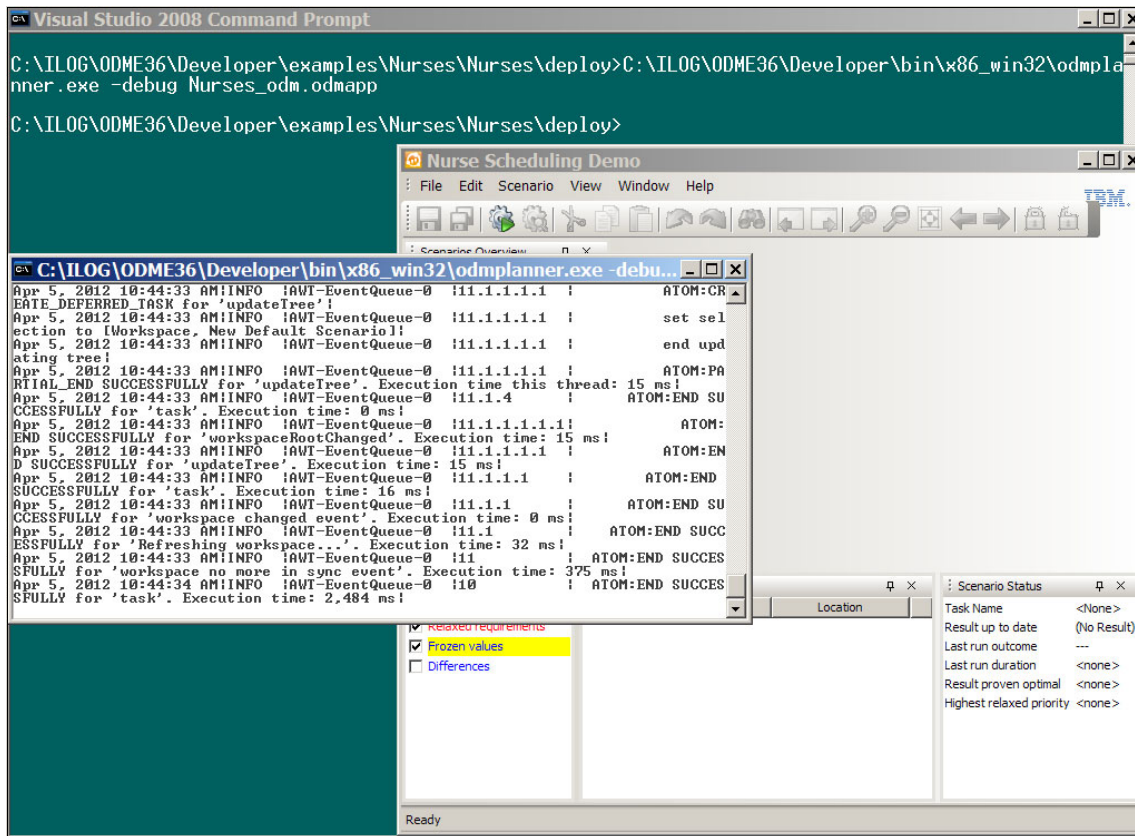


Figure B-1 The starting ODM Enterprise Nurse sample with console

The log level of the output in the log file and the console are controlled separately:

- `java.util.logging.FileHandler.level`

This property controls how much detail is logged in the log file `odm.log`. The available log levels in descending order are:

- SEVERE (highest value)
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST (lowest value)

Messages with a lesser log level are not written to the file. So if you want to log all available information to the log file, set the log level of `java.util.logging.FileHandler.level` to `FINEST`.

This level of logging provides you with too much data. If you have a hard time sifting through this data, use a less detailed level. You might start with `INFO`, then go to `CONFIG`, `FINE`, and `FINER`, before you eventually use `FINEST`.

► `java.util.logging.ConsoleHandler.level`

This property controls how much detail is displayed in the console. It can be used in a similar way to using `java.util.logging.FileHandler.level` to control the console logs.

ODM Enterprise components also use separate categories to enable specific logging. These categories are found in the default configuration files:

► `ilog.odm.datasvc`

Settings in this file control logs that are related to data services and database queries. You do not have to specify how you deployed (direct database access or using a data server) your ODM Enterprise applications.

► `ilog.odms.dataserver`

Settings in this file control logs that are related to the ODM Enterprise Data Server.

► `ilog.odm.processingsvc` and `ilog.odm.optimcore`

Settings in these files control logs that are related to ODM solve.

► `ilog.odm.ide`

Settings in this file control logs that are related to the ODM Enterprise Development IDE, mainly for deployment of ODM Enterprise applications.

► `ilog.odm.studio` and `ilog.odm.desktopstudio`

Settings in these files control logs that are related to ODM Studio.

► `ilog.odm.utils`

Settings in this file control logs that are related to ODM Enterprise helper classes.

To enable the deepest logging level for all components, use the settings that are shown in Example B-1 for the `odmlogging.properties` files. The first two lines increase the log level of the log files and console output to the highest level. The other lines maximize the log level of every category.

Example: B-1 odmlogging.properties settings for log level detail

```
java.util.logging.FileHandler.level = FINEST
java.util.logging.ConsoleHandler.level = FINEST
```

```
ilog.odm.datasvc.level = FINEST
ilog.odms.dataserver.level = FINEST
ilog.odm.optimcore.level = FINEST
ilog.odm.ide.level = FINEST
ilog.odm.desktopstudio.level = FINEST
ilog.odm.studio.level = FINEST
ilog.odm.utils.level = FINEST
ilog.odm.processingsvc.level = FINEST
```

Enabling logging for the Optimization Server

There are two kinds of logs for the Optimization Server:

- ▶ Optimization Server components (management and processor EARs) can produce logs. These logs follow the configuration setting of your application server. Unless configured differently for your application server, the logs of the Optimization Server components are found in the log files of the application server.
- ▶ When a job is submitted to the Optimization Server, the Optimization Server creates a subprocess that actually optimizes the scenario. The logs for this subprocess can be found in the application server log files, together with the processor EAR log statements.

Enabling these two types of logs for a WebSphere Application Server is addressed in the following sections.

Enabling logging for Optimization Server components

Use the WebSphere Application Server administration console (click **Troubleshooting** → **Logs and Trace** → **<server>** → **Change log detail levels**) to set the verbosity of the Optimization Server logs, as shown in Figure B-2.

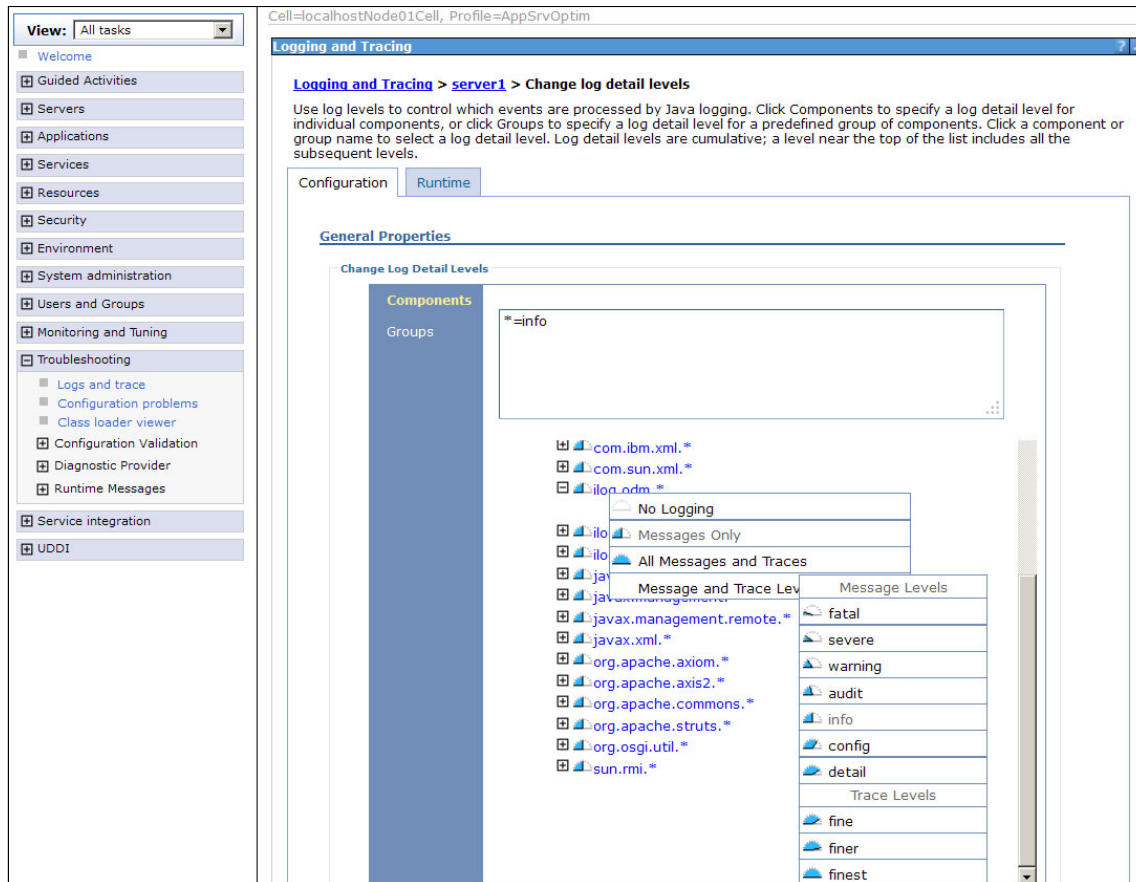


Figure B-2 Setting the log detail level of classes in the WebSphere Application Server console

Enabling logging for the actual subprocess

By default all logs that are produced by the subprocess can be found in the application server log. The subprocess that runs the optimization can be perceived as just another ODM Client. If you want to enable finer grained logging for this subprocess, configure it through the `odmlogging.properties` file.

The Optimization Server component has an `odmlogging.properties` file that is in the `optimserver-processor-rar-impl.jar` inside the `optimserver-processor-ear` file. The file can be found in `<dir>\<edition>\server\ears` with the other Optimization Server components.

For more information about configuring `odmlogging.properties`, see “Enabling logging on the ODM Enterprise client side” on page 295.

Remember: The default `odmlogging.properties` file for the server components generates only console logs, which are then written to the application servers logs. You can run ODM/E on many application servers, which might have their application server logs at different locations. The WebSphere App Server that is bundled with the ODM/E version has this log stored at the following location:

```
<install-dir>\<edition>\server\was_70\profiles\AppSrvOptim\logs\server1\SystemOut.log
```

The console logs include more information than file logs. Therefore, there is no significant advantage to use log files unless you want to separate ODM Enterprise logs from application server logs.

For quick tests, edit the file in the `<ODM-runtime-directory>\lib` directory that is extracted during the startup of the processor EAR. The ODM Enterprise runtime directory can be found in the following ways:

1. Using the WebSphere Application Server Console (Figure B-3 on page 302):
 - a. Expand **Applications** → **Application Types** in the left pane of the console, and select **WebSphere enterprise applications**.
 - b. The Enterprise Applications interface is displayed. Click **odme-optimserver-processor** → **Manage Modules** → **optimserver-processor-rar** → **Custom properties**.
 - c. The extraction folder is defined by the property `DeploymentDirectoryLocation`, as shown in Figure B-3 on page 302. If its value is empty, the runtime directory is the system temp folder.

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
 - New Application
 - Application Types
 - WebSphere enterprise applications
 - Business-level applications
 - Assets
- Services
- Resources
- Security
- Environment
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
- UDDI

		settings. A value of -1 or 0 means that there is no minimum.	
DeploymentDirectoryLocation		Directory where to extract the deployed applications and the optimization runtimes for execution. The directory must be writeable. Leave empty to automatically create a new directory in the system temp folder at startup	false
NumberOfSessions	3	Maximum number of sessions running concurrently	false
OdmJavaArgs	-Xmx1024M	Arguments to pass to the Java Virtual Machine created by the sub process. To augment the LIBPATH used to start the JVM, the -Djava.library.path option can be specified	false
OdmJavaHome		Location of the java runtime to be used by the sub process (example: c:/Program Files/Java/jdk1.5.0/, /usr/java5/). Leave empty to use the same Java runtime as the one used by the JavaEE container.	false
OdmSolverArgs		Arguments to pass to the solver sub process.	false

Figure B-3 Resource Adapter of the ODM Enterprise processor

- By looking at the application server log `SystemOut.log` for the statement ODM root directory is...

For the bundled WebSphere Application Server, you can find the file in folder `<install-dir>\<edition>\server\was_70\profiles\AppSrvOptim\logs\server1\`

The logging properties are used for the next optimizations. You do not need to restart your application server for the change to become effective. However, remember to integrate the `odmlogging.properties` file into the `optimserver-processor-ear` file if you need to make this set of logging properties persistent.

Customizing the log level for each ODM Enterprise application

If you have a JAR file with custom Java code, you can customize the log level of the ODM Enterprise application to override the `odmlogging.properties` file in the product `lib` folder. All that you need to do is to put the `odmlogging.properties` file into your JAR file. Then, include the JAR file as “Additional JARs and Resources” with your ODM Enterprise applications.

Customizing the log level for each user

You can use `odmlogging.properties.<userid>` to specify a log level for a user with id `userid`. The user-specific properties file has a higher priority than the `odmlogging.properties` file in the `product lib` folder.

Changing the location of log files

By default, the ODM Enterprise log files are written to the directory defined by the variable `%odm`. If you want to have a consistent location for the log files, you can hardcode a directory in the `odmlogging.properties` file. However, you need to ensure that the directory is also valid on the server, especially when the production server is running a different OS from the development environment.

Packaging ODM Enterprise applications

The deployment process automatically packages the files that are referenced in your ODM Enterprise projects. However, some cases require more attention:

- ▶ If you include more OPL model files in the main OPL model for your application, the additional OPL models are not automatically packaged into the ODM Enterprise applications. You need to add the additional OPL model files to your ODM Enterprise applications explicitly as “Additional JARs and Resources”.
- ▶ For your OPL projects to be deployed as part of your ODM Enterprise applications, use a flat directory structure without using any subfolders. For example, if your main model file is `aaa.mod` and it references `bbb.mod` by an `include` statement, put `aaa.mod` and `bbb.mod` in the same directory. The main reason is that file-lookup relative to a JAR file can sometimes have issues when finding the files in subfolders. By using a flat directory structure, you avoid this issue.

Isolating problematic behavior

When you are facing problematic behavior, consider the following suggestions to help you isolate this problematic behavior:

- ▶ If the application has worked before, think about what has been changed in your application and your deployment environment, including hardware, software upgrades and fixpacks, and remote servers. The problem might be solved by undoing these changes.
- ▶ If the deployment is related to servers, try the following tests:
 - Try to validate the server configuration by using a validated ODM Enterprise application, such as ODM Enterprise samples. For more information, see “Validating the configuration of ODM Enterprise server

installation” on page 314. If the server installation cannot be completed, the problem might be caused by server issues.

- If applicable, try to configure your ODM Enterprise applications to avoid using ODM Enterprise servers. If the ODM Enterprise application works fine without a server, it helps confirm that the issue is related to the server installation. If the application terminates with an error, check whether the error is related to the problematic behavior.
- If the problematic behavior is only reproducible with a certain scenario, take a snapshot of the scenario by exporting the scenario to the file system. You can then send the exported data to the developer of the ODM Enterprise application for further analysis.

Tip: If the developers of the ODM Enterprise application cannot access the servers in your deployment environment, transfer the data in an ODM Scenario by exporting it as files.

Troubleshooting tips

This section addresses more specific tips that are related to troubleshooting. Many of the general problems that are described in the troubleshooting common use cases section link to in-depth discussions of the issues here, and how to solve them.

Resolving issues during product installation

Generally, ODM Enterprise installers work well and are straightforward. However, for systems that run Windows Vista or Windows 7 with User Account Control (UAC) turned on, you might have troubles with the installation process. In some unusual cases, the installer cannot terminate by itself.

To avoid this issue, you need to start the ODM Enterprise installer as an administrator by right-clicking Setup.bat and selecting **Run as administrator**. This process is shown in Figure B-4.

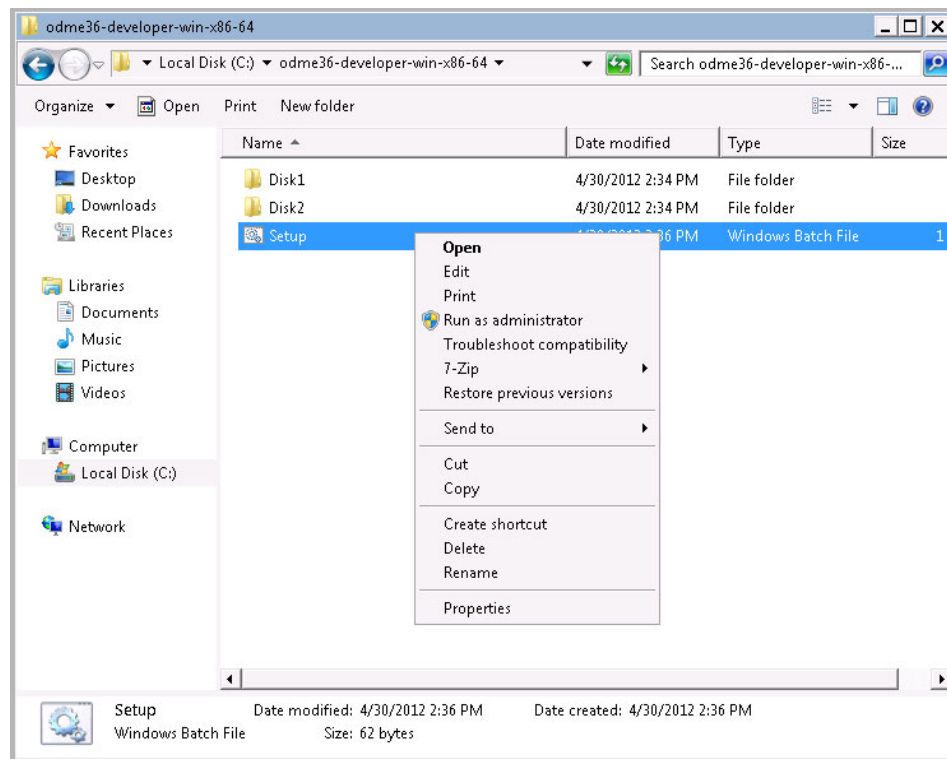


Figure B-4 Starting the ODM Enterprise installer as an administrator

Recently, an installation issue that involves the `autorun.inf` file was discovered that can affect the installation of the bundled WebSphere Application Server in the Developer Edition of ODM Enterprise. This issue is displayed as an error in the server installation log. For more information about how to inspect the installation logs, see “Errors in the installation”.

Where to look for more specific error messages

This section introduces the locations of common places to look for error messages that can help you diagnose unexpected behavior.

Errors in the installation

At the end of the product installation, typically you can find a log file that is named similar to

IBM_ILOG_ODM_Enterprise_Developer_Edition_Install_<date_time>.log in the folder <install_dir>\<edition>. If you choose to install the bundled WebSphere Application Server, you can find its installation log in <install_dir>\<edition>\server\was_70\logs\install\log.txt. These log files are extremely detailed. If your installation finishes with errors or warnings, or if some component of ODM Enterprise does not work as expected, look for more details in these files.

Tip: Do not overlook non-fatal errors or warnings if the related components are not working correctly.

Error messages on the development clients

If you are using any of the ODM Enterprise development GUIs, you can find a Problems tab in the lower part of the ODM Enterprise Development IDE. Similarly, there is an Issues window in ODM Studio. When you see unexpected behavior or an alert window, check the relevant IDE for error messages. Also, if you start ODM Studio directly from within the Development IDE, you can find information in both environments.

Tip: If you cannot find the Problems tab in the Development IDE, this tab might have been closed. Click **Windows** → **Show View** → **Problems** to bring back the Problems tab. Similarly, the Issues window can be closed by the user. If you need to display it again, click **View** → **Show/Hide Issues**.

If you cannot find any useful hints in the GUIs, raise the log level of ODM Enterprise as explained in “Enabling logging on the ODM Enterprise client side” on page 295. The console logs might include more information than the file logs.

Another way to get more information in the IDE is to open the Eclipse Error Log by clicking **Windows** → **Show View** → **Error Log**.

Error messages on the server side

If the ODM Enterprise Optimization Server and Data Server components are properly installed, you only need to check the application server logs. However, you might want to increase the log level to reveal more details of the operations on the server side.

If the server components are not properly installed or configured, see “Testing connectivity to your server console” on page 315 to validate your server installation.

Error messages from databases

Sometimes the error messages of an application do not provide enough insights about any errors on the database side. In this case, directly check the error messages that are logged on the database side around the time the error happened. For example, if you have a large data set and you use DB2 Express-C with its default configuration, you might receive the following error:

```
com.ibm.db2.jcc.b.pm: Error for batch element #1: DB2 SQL Error:  
SQLCODE=-1476, SQLSTATE=40506, SQLERRMC=-964
```

This error alone does not provide many details. However, if you check the journal of the database, you might find a DB2 message that indicates that the transaction log for the database is full.

Identifying performance issues with the database used by the ODM Scenario Repository

To pinpoint performance issues, you need to test the performance of the database that is used by the ODM Scenario Repository. You can do so by using the database performance test tool. This test writes data to the database and tests the speed and accuracy of the results (Save Quality). It then reads from the database, again testing the speed and accuracy of the results (Load Quality). It attributes a rating between 1 (Poor) and 5 (Excellent) to the overall performance based on an IBM reference performance rating. To get the most accurate results, you need to perform the test from the system where you are experiencing the performance problems.

Figure B-5 shows a sample output of the performance test.

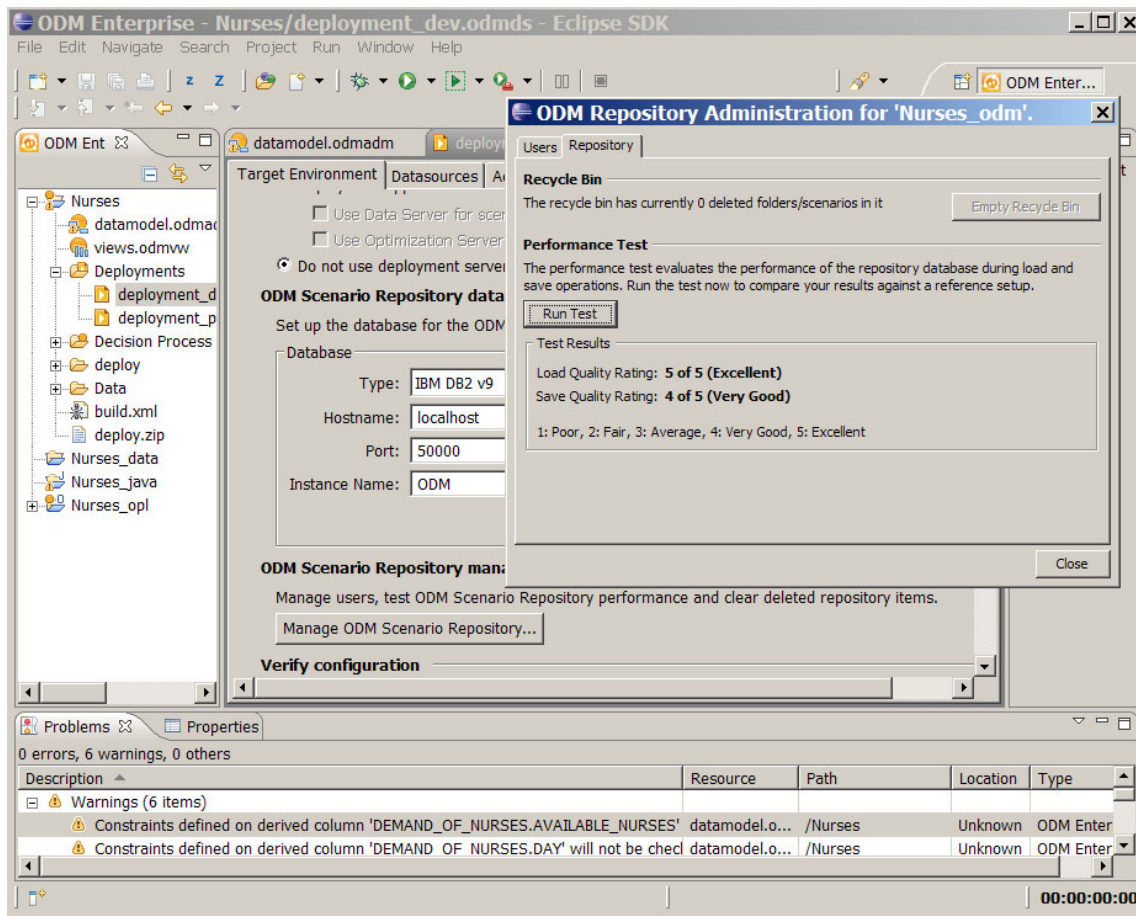


Figure B-5 Performance test of the ODM Scenario Repository in Development IDE

The tool is available in the ODM Enterprise Development IDE and ODM Studio when running as a multi-user ODM Enterprise application.

- ▶ In the ODM Enterprise Development IDE, perform this test from the editor for the deployment settings files (odmfs files), as shown in Figure B-5. Click **Manage ODM Scenario Repository** to open the ODM Repository Administration window. Click the Repository tab, and then click **Run Test** to start a performance test.

- In ODM Studio when running as a multi-user ODM Enterprise application, click **File** → **Administration Tool**, which brings up a similar ODM Repository Administration window. There you can start a performance test in a similar way, as depicted in Figure B-6.

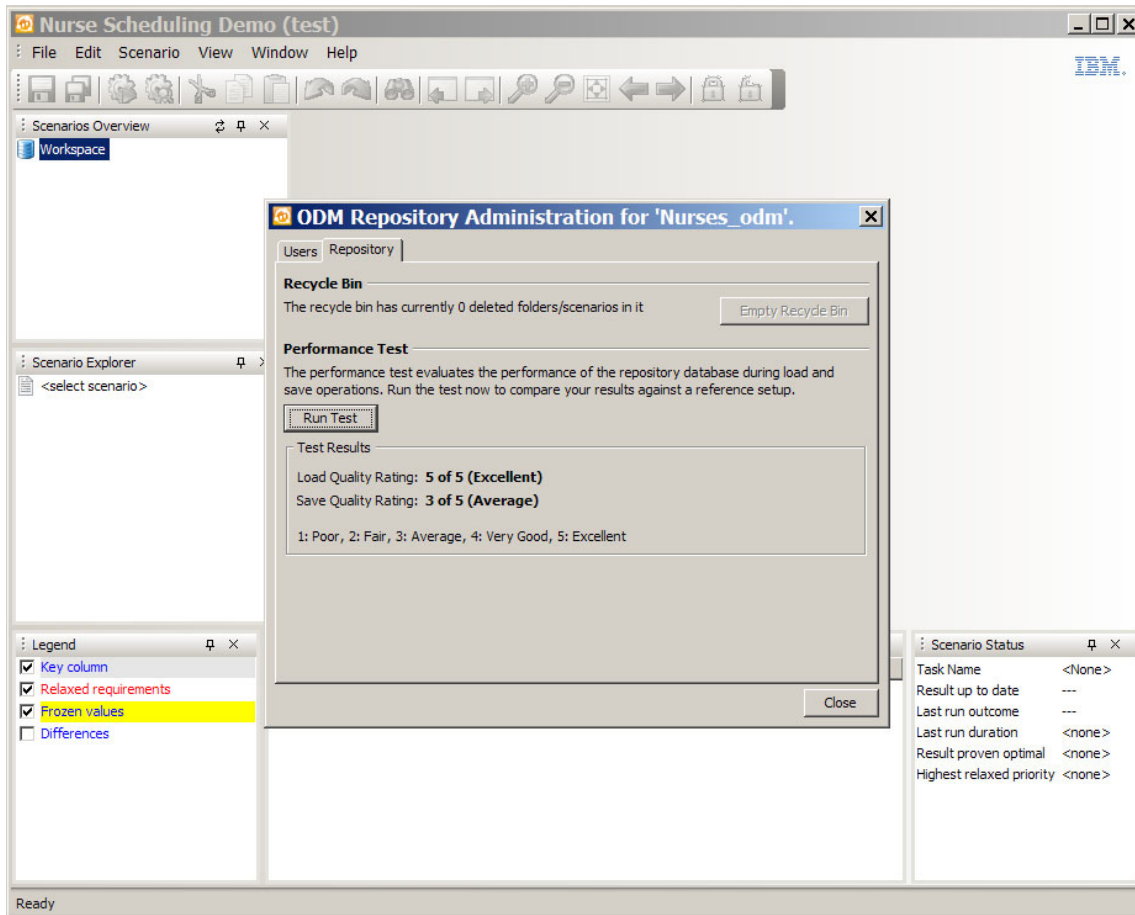


Figure B-6 Performance test of the ODM Scenario Repository in ODM Studio

If the performance test concludes with less than optimal results, contact your IT department to see whether they can offer suggestions for speeding up the database or the network connection.

A poor rating might be because of the following factors:

- Insufficient application performance.

Watch for high processor usage or limited available memory during operations.

- ▶ Insufficient database performance:
 - Performance can degrade if you are using the embedded Derby database (single-user mode). In that case, switch to a multi-user application to use enterprise databases, such as IBM DB2, as the ODM Repository database.
 - If you are already using an enterprise database, you might want to involve a Database Administrator to analyze the performances of the database.
- ▶ Insufficient network performance between the client (such as ODM Studio and Optimization Server) and the database server. You can verify whether you have this issue by using typical network tools (like ping and traceroute) to look for packet loss, high latency, and routing anomalies. In this case, consider the following fixes:
 - For a deployment environment with high latencies, use the ODM Enterprise Data Server if you have not yet done so. Without a Data Server, the client (ODM Studio) needs to communicate with databases by using a Java Database Connection (JDBC) connection. A JDBC connection is not as efficient in a deployment environment with high latencies. When you use the Data Server, the client communicates with the Data Server by using HTTP connections. HTTP is more suitable for this type of deployment environment.
 - Consider running the client (such as ODM Studio and ODM Service API-based applications,) on computers that have low latencies to the servers and databases. Also, use remote access technologies (such as Windows Remote Desktop, remote X11 desktop, and Citrix) to access the ODM Studio IDE. This configuration can avoid the high latency issue and directly resolve the performance issue at the root cause. However, the remote access technologies from different vendors can require further evaluation on factors such as performance, security, administration, and costs.

Sending messages from custom code to ODM Enterprise logs

If you use the ODM Enterprise API to implement custom features, you can send messages to the ODM Enterprise logs together with the usual ODM Enterprise logs. Your custom code needs a *logger* (an instance of `java.util.logging.Logger`) for your custom class. You can log messages by using its logging functions. You can find some code snippets in Example B-2.

Example: B-2 Using logger functions in your custom code

```
Logger logger = IloLogger.getLogger(CustomClass.class.getName());
logger.fine("CustomClass logs a fine message");
```

With this approach, you can also adjust the log level for your custom code by using the ODM Enterprise log property file `odmlogging.properties` as shown in “ODM Enterprise logs guidelines” on page 295:

- ▶ You need to specify the full class name, including the package name. For example, if you want to increase the log level for the class `MinMaxDataChecker` in the package `nurse.check` to `FINEST`, add this line to the `odmlogging.properties` file:

`nurse.check.MinMaxDataChecker.level = FINEST`
- ▶ You also need to adjust the log level for log file or console output to an equal or higher level.

Adjusting memory usage for ODM Enterprise

ODM Enterprise applications are Java-based applications. When you start an ODM Enterprise applications, a JVM is started to support ODM Studio for GUI, views, scenarios, and so on. The memory that is used by this JVM is not available for CPLEX Optimizer or CPLEX CP Optimizer running in the same process. Thus, the more maximum JVM heap space that is reserved by ODM Enterprise (for GUI, views, or scenarios), the less is available for CPLEX Optimizer or CPLEX CP Optimizer. For more information, see “How ODM Enterprise uses memory” on page 290.

This section includes three use cases:

- ▶ ODM Enterprise applications started from the Development IDE
- ▶ ODM Enterprise applications running in ODM Studio
- ▶ ODM Enterprise applications running on the Optimization Server

ODM Enterprise applications started from the Development IDE

If you start ODM Enterprise applications from the ODM Enterprise Development IDE, the Java memory is shared between the Development IDE and the ODM Enterprise application. If you encounter issues with insufficient memory while using 32-bit ODM Enterprise, deploy your ODM Enterprise application from the Development IDE first and then run it with ODM Studio.

To start ODM Enterprise applications directly from the Development IDE, increase the maximum amount of memory that can be allocated to the whole Development IDE and the ODM Enterprise application. You can do so by using the Eclipse option `-vmargs` and Java option `-Xmx`. For example, the **`eclipse -vmargs -Xmx1024M`** command reserves 1 GB of memory for the Development IDE and the ODM Enterprise application that is started from the Development IDE.

You can modify the shortcut to Eclipse for ODM Enterprise to reserve more memory as shown in Figure B-7.

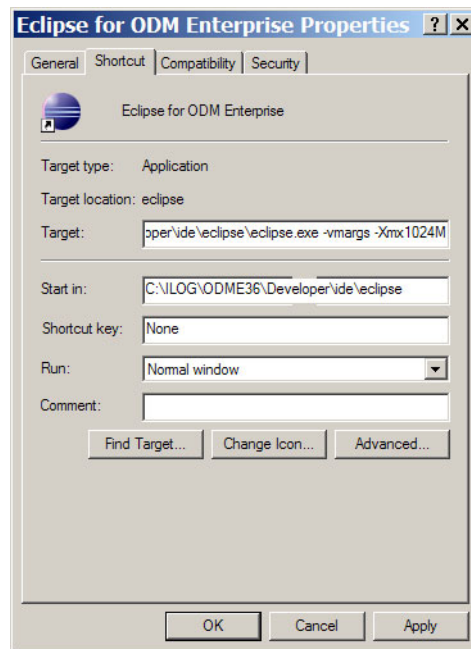


Figure B-7 Configuring the shortcut to Eclipse for ODM Enterprise

You can also modify the file `eclipse.ini` in the ODM Enterprise installation folder. Its location in the Developer Edition of ODM Enterprise 3.6 is `<dir>\Developer\ide\eclipse\eclipse.ini`. By default, there is a line for `-Xmx`, and you can modify it directly. If there is no line like that, you can append one to the `eclipse.ini` file:

```
-Xmx1024m
```

ODM Enterprise applications running in ODM Studio

When you run an ODM Enterprise application through the ODM Studio, the memory that is reserved by the JVM can be controlled by the Java argument `-Xmx`. This argument must be provided to the JVM by using the `ODMS_JAVA_ARGS` environment variable.

You can modify the argument in three ways:

- ▶ You can modify a Windows system environment variable globally. Navigate to the System Properties and select **Environment Variables**. Click **New** and add the variable ODMS_JAVA_ARGS with the value -Xmx 1024M. When finished, click **OK**. This setting is shown in Figure B-8.

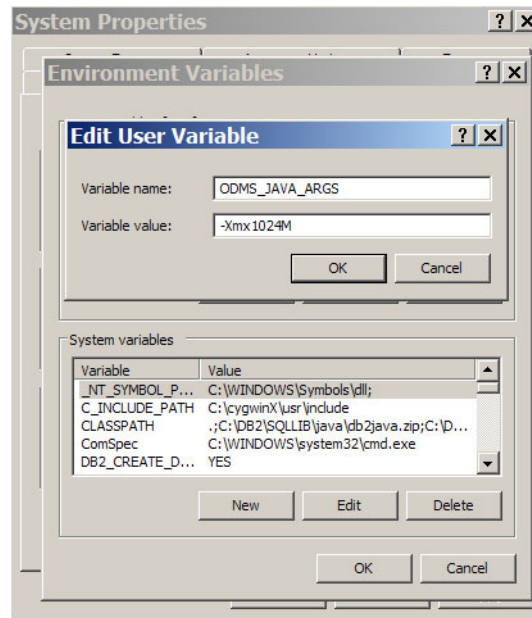


Figure B-8 Setting ODMS_JAVA_ARGS in Windows

- ▶ You can set the variable by using a batch file for one session started by that batch file. Example B-3 depicts a short sample batch file for Windows.

Example: B-3 Batch file to configure memory use

```
SET ODMS_JAVA_ARGS=-Xmx1024M
odmplanner.exe Nurses_odm.odmapp
```

- ▶ You can use the command-line option -javaargs to pass Java arguments, including -XmxN, to the underlying JVM of ODM Studio. You can modify the shortcut to your ODM Studio as shown in Figure B-7 on page 312. You only need to replace -vmargs with -javaargs.

ODM Enterprise applications running on the Optimization Server

To increase the memory available to the ODM Enterprise application that is running on a processor node, modify the `0dmJavaArgs` property. To modify this value, perform these steps:

1. Open WebSphere Application Server Console.
2. Click **Applications** → **Application Types** in the left column of the console, and select WebSphere enterprise applications.
3. The Enterprise Applications interface is displayed. Click **odme-optimserver-processor** → **Manage Modules** → **optimserver-processor-rar** → **Custom properties**.
4. Scroll to find `0dmJavaArgs` as shown in Figure B-9.

View: All tasks

■ Welcome

⊞ Guided Activities

⊞ Servers

⊞ Applications

■ New Application

⊞ Application Types

■ WebSphere enterprise applications

■ Business-level applications

■ Assets

⊞ Services

⊞ Resources

⊞ Security

⊞ Environment

⊞ System administration

⊞ Users and Groups

⊞ Monitoring and Tuning

⊞ Troubleshooting

⊞ Service integration

⊞ UDDI

		Settings. A value of -1 or 0 means that there is no minimum.	
DeploymentDirectoryLocation		Directory where to extract the deployed applications and the optimization runtimes for execution. The directory must be writeable. Leave empty to automatically create a new directory in the system temp folder at startup	false
NumberOfSessions	3	Maximum number of sessions running concurrently	false
OdmJavaArgs	-Xmx1024M	Arguments to pass to the Java Virtual Machine created by the sub process. To augment the LIBPATH used to start the JVM, the -Djava.library.path option can be specified	false
OdmJavaHome		Location of the java runtime to be used by the sub process (example: c:/Program Files/Java/jdk1.5.0/, /usr/java5/). Leave empty to use the same Java runtime as the one used by the JavaEE container.	false
OdmSolverArgs		Arguments to pass to the solver sub process.	false

Figure B-9 The location of `OdmJavaArgs` in the WebSphere Application Server console

Validating the configuration of ODM Enterprise server installation

To validate a server installation, you need to have an ODM Enterprise application that can open and solve flawlessly on your local system. If you do not have a validated ODM Enterprise application, use the examples that are provided in the Developer edition. You can find the examples in the directory `<install-dir>\Developer\examples`.

Testing connectivity to your server console

Before you test the status of the ODM Enterprise server components, connect to the console of the server (the server that hosts the ODM Enterprise server components) from the client systems. The client systems are the ones that host the ODM Enterprise applications. Make sure the server itself is up and reachable from the client side.

In a standard configuration, you can reach the web console of the WebSphere Application Server by going to this address:

`http://server:9060/ibm/console`

To get to the web console for the JBOSS server, navigate to:

`http://server:8080/web-console`

Except for the web console, the URLs for the server components that are deployed to a JBOSS server are almost identical to those deployed to a WebSphere Application Server. The only difference is the port number. All examples show only the URLs for WebSphere Application Server. If you are using a JBOSS server, replace the port number 9080 (for a WebSphere Application Server) with port 8180 (for a JBOSS server).

Consideration: The sample URLs use the default port and the default path to the web console. Your server administrator might have changed the *port number* or the *path* for the web console. If these URLs do not work for you, try to open the URLs on the server first. If the URLs do not work on the server, contact your server administrator to verify the configuration.

The WebSphere Application Server port can be found in a file that is called `AboutThisProfile.txt`, which is in the profile log directory.

If the web console can be opened from the server but not on the client systems, perform the following checks:

- ▶ Check whether you have a *firewall* installed on the servers, or if there are firewalls between the clients and the servers. If there are firewalls, make sure that they are not blocking ODM Enterprise applications.
- ▶ If the servers are not started as system services, the servers might be stopped when the user logs out from the server system. Make sure that your servers are started as *system services* to avoid this situation.

To check the status of the ODM Enterprise server components on WebSphere Application Server, perform these steps:

1. Log in to the WebSphere Application Server console.

2. Navigate to **Applications** → **Application Types** in the left column of the console and select WebSphere enterprise applications. Figure B-10 shows a test server. On this server, the application ivtApp is not running because its status is displayed with a red x. The other applications are started correctly because their status is displayed as green arrows.

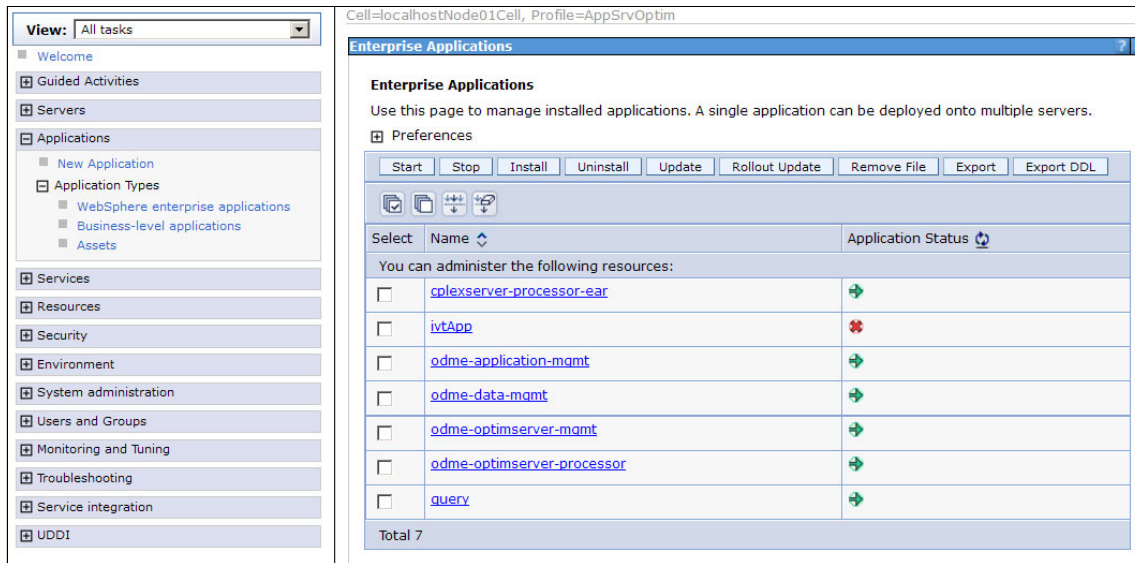


Figure B-10 The console of a test WebSphere Application Server

For ODM Enterprise server components, if a server component cannot be started, you need to resolve this issue first. For new installations, it can be more efficient to reinstall the server components, rather than debugging the configuration issues. For existing configurations that have been running smoothly for a while, track the changes made to the server and undo them.

Testing connectivity to your Optimization Server

Connect to the Optimization Server console to test whether the server is reachable and running by navigating to:

`http://server:9080/odme/console`

You should be able to view the deployed applications and registered jobs, as shown in Figure B-11. For a new installation, both the Applications table and the Jobs table are empty.

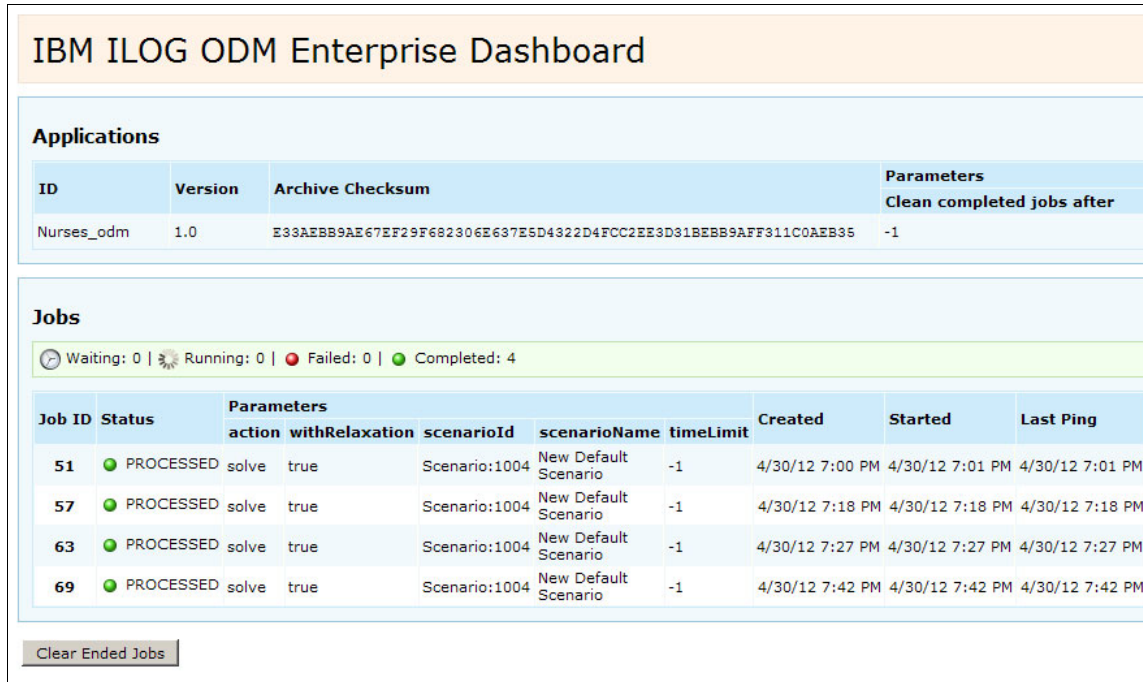


Figure B-11 ODM Enterprise Optimization Server console

If you cannot open the console, check the following items in the Optimization Server log:

- ▶ If the Optimization Server data source was created in the application. If not, you might get the following error:

```
ilog.odms.optimserver.dao.IloDAOException unable to locate
optimserver data source entry in JNDI
```
- ▶ If the Optimization Server schema was created in the Optimization Server database. If not, you might get the following error:

```
ilog.odms.optimserver.dao.IloDAOException undefined table when
listing deployed OS applications
```
- ▶ If the Optimization Server database is up and running. If not, you might get the following error:

```
ilog.odms.optimserver.dao.IloDAOException connection issue when
listing deployed OS applications
```

Testing connectivity to the Optimization Server web services

Connect to the web services used on the Optimization Server management application by navigating to each of the following addresses:

`http://server:9080/odme/optimserver/services/JobManager?wsdl`

`http://server:9080/odme/applicationserver/services/DeploymentManager?wsdl`

You should be able to view the web service description files. If you cannot and you are using WebSphere Application Server, check the following items:

- ▶ Verify that the option **Class loading and update detection** of the `optimserver-mgmt-ear` is set to **Classes loaded with application class loader first** and **Single class loader for application**.
- ▶ Verify that the **Starting weight** properties (in “Manage Modules”) for the `optimserver-mgmt-ear` are correctly set. `optimserver-mgmt-webapp.war` must be set to a higher value than the starting weight of the other module (`optimserver-mgmt-ejb.jar`). This configuration ensures that the war files are initialized in the correct order.

Validating deployment

If you are able to open the Optimization Server console and access the definitions of the web services, you should be able to deploy an ODM Enterprise application on the server and to submit an optimization request. Both tasks are handled by the management application.

To deploy an ODM Enterprise application to an ODM Enterprise Optimization Server, open the project in the ODM Enterprise Development IDE. Switch to the ODM Enterprise Perspective and open the deployment settings file (.odm file), as shown in Figure B-12.

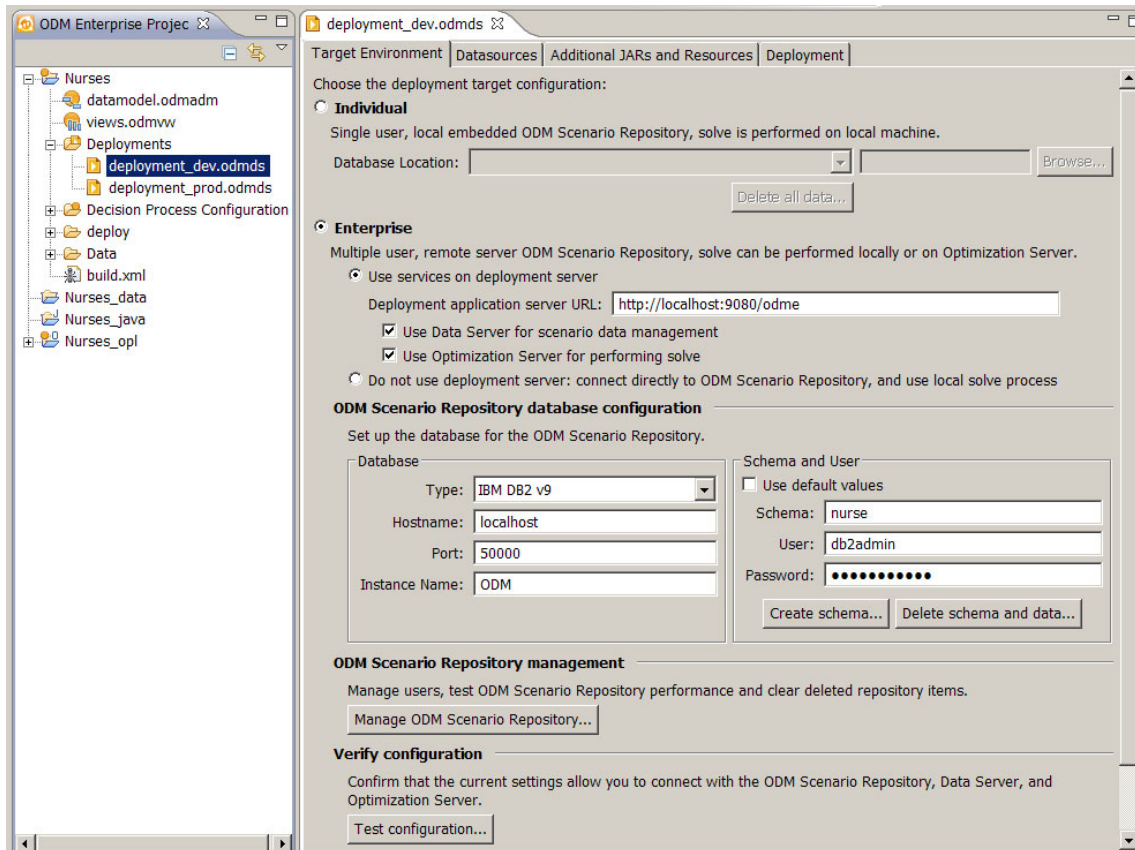


Figure B-12 Sample deployment settings for an ODM Enterprise application

Complete the settings in the following order:

1. Select **Enterprise** as the deployment target configuration.
2. In the ODM Scenario Repository database configuration section, select the database that you want to use, and enter the schema name and user credentials. Note that the ODM Enterprise Development IDE is not shipped with the database drivers for Oracle or Microsoft SQL Server. You need to manually download these drivers and store them in:

```
<install_dir>\<edition>\ide\eclipse\dropins\oplide\plugins\ilog.odms
.ide.odm
```

The names of the Eclipse plug-in directories are usually extended with version information, such as:

```
<Install_dir>\<edition>\ide\eclipse\dropins\oplide\plugins\ilog.odms  
.ide.opl1lang.win32.win32.x86_12.4.0.20111106-0916
```

3. Click **Create schema** to initialize the database schema for this ODM Enterprise application.
4. Click **Manage ODM Scenario Repository** to create a user for the ODM Enterprise application.
5. Click **Test Configuration** to validate the settings.
6. Right-click the .odms file that you are editing, and select **Run as → ODM Run Configuration**.

After you complete these steps, you should see your application listed in the Applications table in the Optimization Server console, as shown on Figure B-11 on page 317.

If you receive the error `Archive is not deployed`, it might be either of these problems:

- ▶ The ODM application is not deployed to the Optimization Server
- ▶ There is a mismatch between the ODM application and the version that is deployed on the Optimization Server (either the local application or the one deployed is newer than the other).

Validating the configuration of the processor application

After you deploy the ODM Enterprise application to the Optimization Server, you can submit a job to the Optimization Server by solving a scenario. For more information about how to navigate to the Optimization Server console, see “Testing connectivity to your Optimization Server” on page 316.

After you submit a job, you should be able to see a job displayed in the Jobs table in the Optimization Server console. The status of a job shown in the status column of the Jobs table can change according to the table shown in Figure B-13.

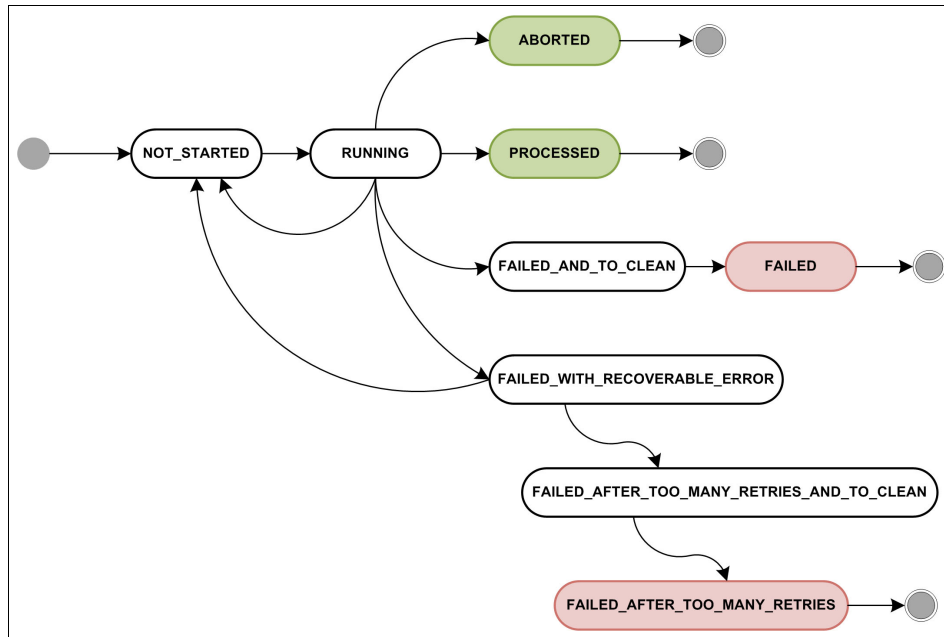


Figure B-13 Job status transition table

A job that completes successfully should end with a status **PROCESSED**. A job that is terminated by the user ends with a status **ABORTED**.

If your job gets stuck at **NOT_STARTED**, or terminates with a status **FAILED** or **FAILED_AFTER_TOO_MANY_RETRIES**, check the application server log for errors and resolve them. Errors are typically caused by the following problems:

- ▶ The deployment directory is not writable for the application.
- ▶ The database driver is not correctly deployed to the server. Note that the database drivers for Oracle and Microsoft SQL Server are not included by default. For more information about how to repack the server components with an appropriate database driver, see *IBM ILOG ODM Enterprise Optimization Server and Data Server Installation Guide* at:

http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/topic/ilog.odms.ide.odm.enterprise.help/ODME/OptServer_DEV_Inst/topics/OptServer_DEV_Inst.html

- ▶ The database that is used by ODM Scenario Repository is not configured correctly or unavailable.
- ▶ There is an out-of-memory error for the subprocess. Adjust the Java heap size for the subprocess as addressed in “ODM Enterprise applications running on the Optimization Server” on page 314.

Validating interactions between a client and the server

Client applications like ODM Studio can interact with ongoing jobs, including accepting the current solution found by the optimization, relaxing more requirements, and stopping the optimization.

You might get the following exception in the Optimization Server log because the interaction topic used to communicate with solve the process is not created on the application server:

```
ilog.odms.optimserver.job.ws.client.IloOptimServerConnectionException
```

With WebSphere Application Server, you must ensure that all processor nodes are registered as a member of the message bus that is used by the Optimization Server topic.

Cleaning up the ODM Enterprise Development IDE workspace

The ODM Enterprise Development IDE workspace is an Eclipse workspace that can suffer stability issues because of various reasons that include JVM crashes, computer crashes, and limitations of Eclipse itself. Sometimes, these stability issues can cause strange behavior or mysterious errors. You can help avoid these problems by periodically cleaning up your workspace, even if you do not notice anything wrong in your projects.

First, refresh all projects in the workspace. Then, if needed, do a clean rebuild of the projects. Use the Development IDE menu item **Project** → **Clean**, then select **Clean all projects**.

If this step does not improve the situation, you can use the following approaches to do a cleanup:

- ▶ Close and restart Eclipse.
- ▶ Use the command-line option `-clean` to force the Development IDE/Eclipse to rebuild the workspace and Eclipse caches. It can be used in these ways:
 - Append `-clean` in a new line to the Eclipse configuration file `eclipse.ini` for the Development IDE. The file is usually in the directory `<install-dir>\<edition>\ide\eclipse`.

- Add `-clean` to the shortcut for the Development IDE as addressed in “ODM Enterprise applications started from the Development IDE” on page 311.

Requirement: For each of these options, remember to remove the `-clean` option afterward. Otherwise, your system will clean every time that you start Eclipse, which will affect performance.

- The last resort is to switch your workspace and select a new location. Within the Development IDE, click **File** → **Switch Workspace**. Select a different directory as the new workspace. The new workspace will be empty. Next, click **File** → **Import**. In the Import window, select **General** → **Existing Projects into Workspace** as show, in Figure B-14. You can then re-import the projects from your original workspace into the new workspace.

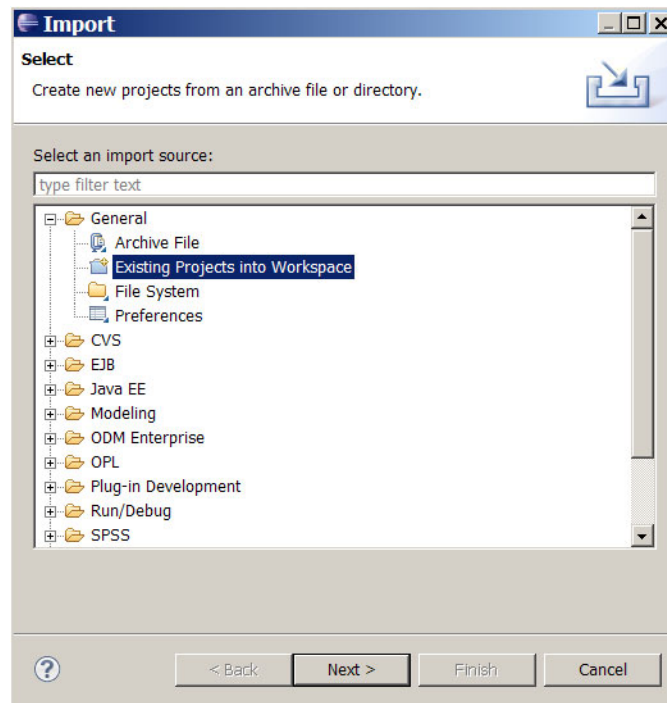


Figure B-14 Import window of ODM Enterprise Development IDE

Improving memory usage when using an OPL data source

OPL data sources are not designed for bulk data loading. Generally, use an ODM data source (JDBC data source or custom data source) because of the following advantages:

- ▶ More direct access to data.
- ▶ Avoids expensive communication between the Java and native layers.
- ▶ Avoids duplication of data in the native OPL layer and the ODM layer.
- ▶ Avoids loading all the data in memory in the native OPL layer before the system creates tables.
- ▶ Offers a finer memory management that allows loading table by table.

If you need to use an OPL data source and want to save memory, you can use the `commitBetweenTables` property in your data source as described in “Improving memory usage when using an ODM data source”.

Improving memory usage when using an ODM data source

When you are using built-in ODM data sources, all updates are saved at the end of the import. In some cases, this process can increase memory usage. This behavior can be changed to save after each table to release memory by setting the `scenarioSource` attribute `commitBetweenTables` to `true` in the `.odm` files:

```
<scenarioSource id="MyScenario" commitBetweenTables="true"/>
```

If you are writing your own custom data source, you can follow a similar approach and regularly save the scenario.

Debugging your custom code based on ODM Enterprise APIs

If you have custom code that is based on ODM Enterprise APIs, such as custom views, custom data checkers, and custom hooks, you must use special steps to debug them. You can reconfigure the ODM Enterprise Development IDE by using Eclipse features to enable debugging of the code using these steps:

1. Click **Run** → **Debug Configurations** to open a dialog to set up a new configuration for debugging purposes.
2. Right-click **Java Application** and select **New** to create a configuration.
3. In the Main tab, select the Java project that contains your custom code from the Project list.
4. In the Main Class field, enter `ilog.odm.service.IloOplOdmLauncher`.

5. In the Argument tab, enter the path to an already-deployed ODM Enterprise application in the Project field. You can use Eclipse variables to shorten the path, such as:

`"${resource_loc: /<project>/deploy/<project>_odm.odmapp}"`

Include the double quotation marks in the textbox because Eclipse cannot handle a path with space characters well. The field VM Arguments is equivalent to a session-only definition of ODMS_JAVA_ARGS. You can adjust the `-XmxN` option to specify the amount of memory that is reserved for Java heap. Or you can use the `-Ddeveloper` option to enable the Development menu in ODM Studio.

6. In the Classpath tab, click **Advanced...** → **Add Library** → **OK** → **ODM Library** → **Finish**.

Figure B-15 shows the dialog before clicking **Finish**. Make sure that you also include other JAR files that the project depends on.

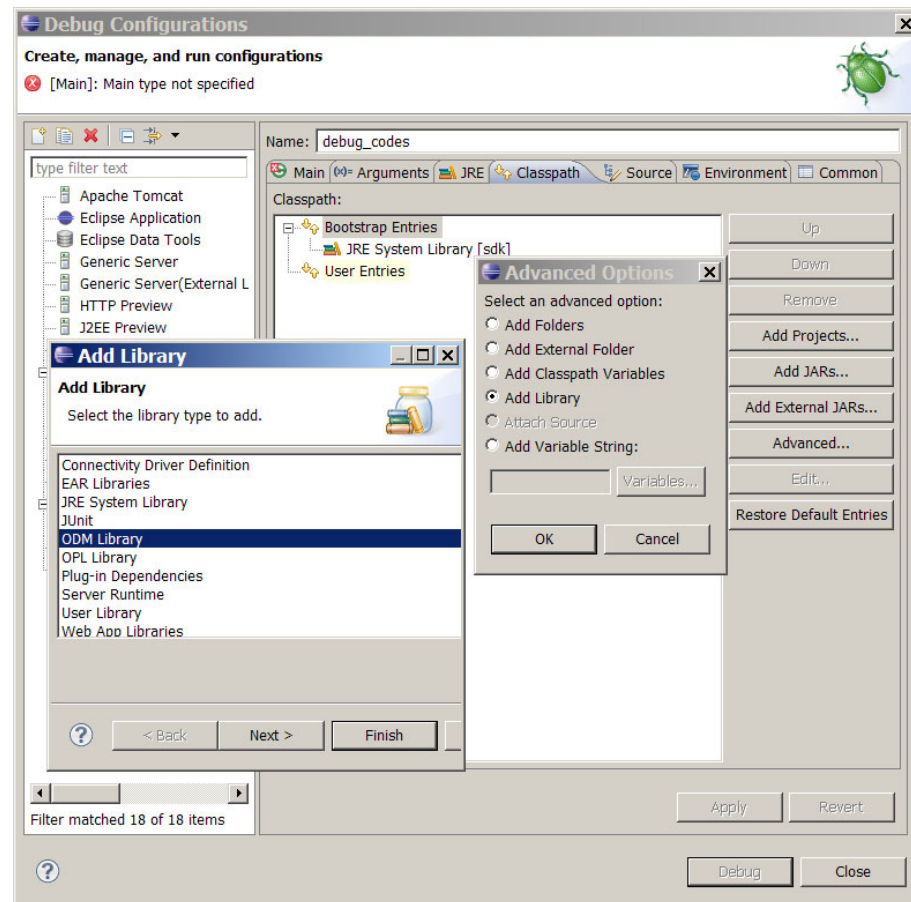


Figure B-15 Adding the ODM Library to ODM Enterprise debug configuration

7. In the Source tab, add the directory that contains the source of your custom code. Figure B-16 depicts the location for adding the Java code for the Nurses example to the debug configuration. First, click **Add**, then in the Add Source dialog click the **Workspace Folder**. In the Folder Selection dialog, navigate to the location for your custom code and click **OK**.

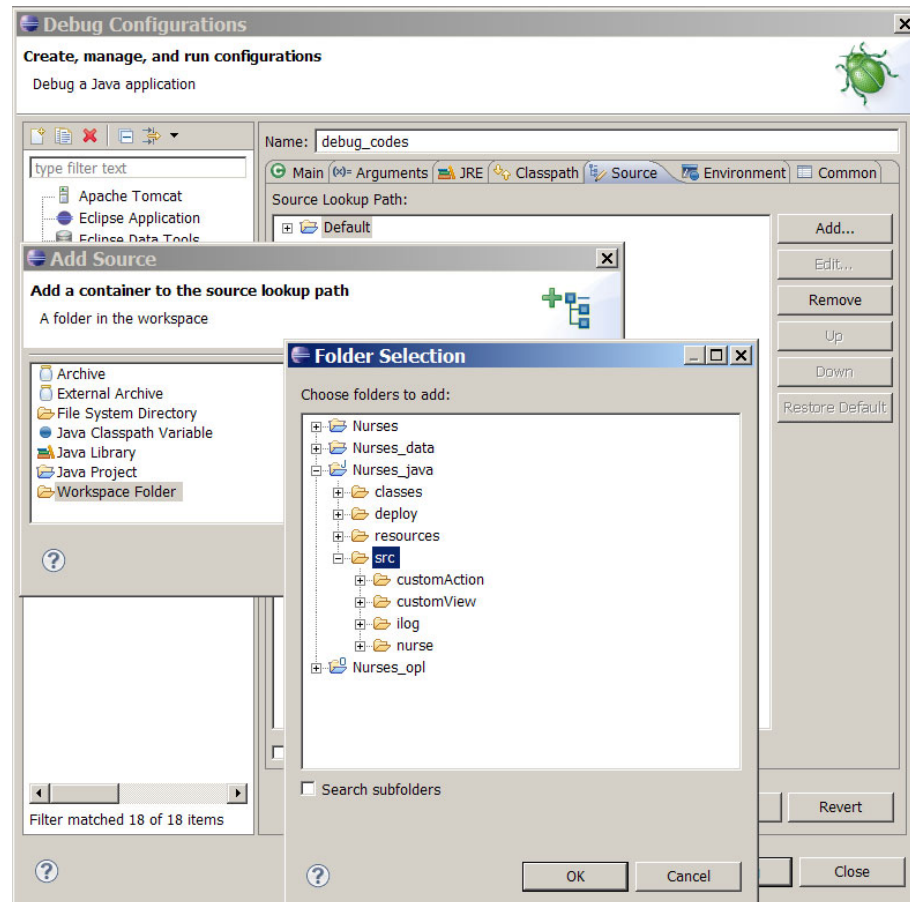
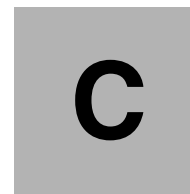


Figure B-16 Adding source directories to ODM Enterprise debug configuration

Starting ODM Enterprise applications this way has the following advantages:

- ▶ The ODM studio will be started in a new process using the library files for the ODM Enterprise Development IDE. For 32-bit ODM Enterprise, this process might allow ODM Studio to make full use of the memory.
- ▶ In the console tab of the Development IDE, all console logs are captured. Because console logs can have more information than file logs, you might be able to find more information for your debugging purpose.



Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG248017>

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248017.

Using the web material

This section contains instructions about how to install and run the ContainerYard ODM Enterprise application. The ContainerYard application is used as an example throughout the book. Although the distributed version of the application

is almost identical to the one used in the book, it is not an exact copy. As a result, you might notice some discrepancies between the application and the screen captures that are presented in the book.

You will need to have IBM ILOG ODM Enterprise Developer Edition 3.6 installed to run this demo.

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation and store the web material .zip file into this folder. You do not have to extract any of the data. After you complete this process, follow the following instructions to deploy and run the ContainerYard application.

1. Start the ODM Enterprise IDE.
2. Import ContainerYard projects by selecting **File** → **Import** → **OPL** → **Existing OPL projects**.
3. Click **Next**.
4. In the project import wizard, select **Select archive file** and browse to the directory where you stored file ContainerYard.zip.
5. Select the ContainerYard.zip file, select all the available projects (ContainerYard, ContainerYard_data, Containeryard_opl), and then click **Finish**.
6. You should now have a ContainerYard ODME, Data, and OPL projects imported into your workspace. To run the ContainerYard application from the IDE, view ContainerYard project in the ODME perspective. Expand the project by using the Project Explorer in the IDE. Right-click the deployment_dev.odmcs file in the Deployments folder and select **Run As** → **ODM Run Configuration**.
7. To run the ContainerYard application using a stand-alone ODM Enterprise client, first open the deployment_dev.odmcs file in the ODM Deployment Settings Editor in the IDE. Then, deploy the application from the Deployment panel of the editor.

Overview of the application scenarios

When you start the ContainerYard application, you will see that several scenarios have already been created and saved.

Most of them are referenced in the book:

- ▶ 0 - Base Scenario
- ▶ 1 - Container Selection FIFO
- ▶ 2 - Part Selection Bycatch
- ▶ 3 - Max number of containers
- ▶ 4 - Inspection rule

New scenarios can be created by right-clicking the Workspace in the Scenario Overview pane and selecting New Default Scenario. Default scenarios are created by using the default data source that is specified in the deployment settings of the application (CSV Data Source 1). Default scenario data can be found in <ContainerYard_InstallDir>/ContainerYard/Input Data/input_data_1.

Glossary

3D chart 3D charts allow you to display two-dimensional data by using three-dimensional rendering. Only certain types of charts support 3-D rendering. If you try to switch from a 2-D view to a 3-D view on a chart that does not support this mode, you do not get any error. The visual appearance of the chart does not change.

Accept relaxation level button A button that becomes active on the ODM Solve Progress window when ODM finds at least one feasible solution during phase 1 of the feasto loop. It allows you to instruct CPLEX to accept the current level of relaxations and move on to phase 2 of the feasto operation.

Active In ODM Studio, there is one active scenario: The one that is selected in Scenarios Overview and whose elements are displayed in the Scenario Explorer. Also, there is an Active check box option on the Requirements window that indicates whether the requirement is active or inactive in the current scenario. The default status of this option can be set for each requirement by using the Requirements editor.

Additional JARs and Resources tab One of the tabs of the Deployment Configuration Editor that allows the user to specify JAR files and other resources that are used for customization of the ODM application.

Administration panel The window in the ODM Enterprise IDE (also accessible from ODM Studio) that is used to create user IDs for an ODM application in multi-user mode. It also sets access rights for them.

Analysis folder In ODM Studio, this folder is displayed in the Scenario Explorer. It contains links to the Goals window and Requirements window, which are displayed in every ODM application.

Application Data Model The Application Data Model (ADM) is a relational model that defines and holds the data that are used as input to ODM Enterprise or emerge from it as output. The ODM Enterprise Application Data Model provides graphical editors that allow you to draw the data model the application will use. You can also import the data from an existing external database or OPL model.

Application Data Model Editor The `datamodel.odmadm` file in each ODM Enterprise project contains a reference to the description of the Application Data Model for the application. Double-click this file to display the Application Data Model Editor. This editor can be used to create or modify the model's Application Data ID and Version number, and to jump to the Data project.

Area chart A line chart in which the area beneath each line is given a solid color or pattern to emphasize the relationships between the pieces of charted information.

argument In general mathematics, an argument is an independent variable, often used as a synonym for parameter. In the Requirements editor, you can add arguments to the messages displayed when a requirement is relaxed.

Bar chart A chart in which the values are represented as adjacent bars.

Bi-indexed Bi-indexed charts provide a chart view of a 2D Array view.

bound See Search goal bounds, lower bound, and upper bound.

constraint The word constraint in IBM ILOG products has much the same sense that it has in the business world. A constraint is a restriction or rule that must be respected. IBM ILOG products apply constraints rigorously to objects declared in a model to find a solution to a problem. If the optimizer cannot find a solution according to the constraints on the available resources, it reports failure. In contrast, if the constraints are consistent, and resources are available, it will find a solution and report success.

A constraint is thus a condition that must be satisfied by the solution of a problem. There are several kinds of constraints in real world problems:

- ▶ An *arithmetic* constraint involves one or more decision variables in an arithmetic expression, for example, $x \leq y + 3$. IBM ILOG products express arithmetic constraints in terms of integers, floating-point real numbers, or combinations of them, known as numeric expressions
- ▶ A *symbolic* constraint makes it possible to express a complicated relationship between decision variables. For example, one that would require an exponential number of arithmetic constraints, or one that cannot be expressed at all in arithmetic terms. Symbolic constraints are involved in considerations such as counting (cardinality constraints), ensuring that elements of a set are different from another (alldiff constraints), and controlling membership in collections such as sets or bags.

In ODM, the constraints in the OPL model are mapped to requirements in the ODM application when it is generated.

Continue in background This button in the Solve Progress window allows you to submit the current solve process to run in the background, while you continue to work with other scenarios in ODM Studio. Multiple solve processes can be submitted. When they are, they are added to the Solve Queue, and are solved in the order submitted.

CSV data source A data source (or scenario updater) to fill your ODM scenarios with data retrieved from CSV files. If the CSV files do not currently exist, a wizard creates template CSV files for each of the Application Data Model tables you select. These files include placeholders for each of the table's columns. You can then edit the resulting CSV files later.

data diagram A graphical representation of a relational data model that shows the model's tables, columns, and the primary key-foreign key relationships between tables.

Data Diagram Editor The graphical editor that is used in the ODM Enterprise IDE to “draw” a data model's tables, columns, and the primary key-foreign key relationships between tables.

Data project One of the component projects that make up an ODM Enterprise application. The Data project contains the definition of the Application Data Model, which is the model of the data that is manipulated by the ODM application.

Data Project Explorer A view in the ODM Enterprise IDE that allows you to view and manipulate the contents of the Data project.

Database data source A data source (or scenario updater) to fill your ODM scenarios with data retrieved from an external database to which you have an Eclipse Database Connection.

Datasources tab One of the tabs of the Deployment Configuration Editor that allows you to define the data sources for the ODM application.

Deploy tab One of the tabs of the Deployment Configuration Editor that allows you to create a deployed version of the application. This version can be run on a user's workstation using ODM Enterprise Planner Edition or ODM Enterprise Reviewer Edition.

Deployment Configuration Editor The Deployment Configuration editor in the ODM Enterprise IDE is used for several related purposes, all of which define a deployment configuration for the ODM application:

- ▶ To define the ODM Scenario Repository for the application.
- ▶ To define data sources for the application. See Datasources tab.
- ▶ To specify JARs for the application that contain custom code or other resources that are used by the application. See Additional JARs and Resources tab.
- ▶ To deploy the ODM application itself after all of these items are defined. See the Deploy tab.

Each of these functions is run on a specific tab of the editor window.

Deployments A section in the ODM Enterprise project that contains the `deployment_dev.odm`s and `deployment_prod.odm`s files.

deployment_dev.odms file Contains descriptions of the data sources and deployment configurations that are used by the ODM application in development.

deployment_prod.odms Contains descriptions of the data sources and deployment configurations that are used by the ODM application in production.

Differences option A check box option on the ODM Studio Legend panel that displays any fields that are different in the current solution than they are in the reference scenario in a specific color.

feasopt The CPLEX *feasopt* algorithm takes as input a CPLEX model, and a list of constraints and penalties within the model. It then tries to solve the model in two phases. This process allows CPLEX to solve a problem that is objectively infeasible or that might be infeasible. It first tries to relax the constraints as little as possible to make the problem feasible, and then tries to optimize the objective of the problem.

Frozen values option A check box option in the ODM Studio Legend panel that displays any data in the current solution that is frozen (before a re-solve) in a specific color.

Gap The relative difference between the integer solution that was found and the proven best possible objective solution value.

goals ODM optimizes (maximizes or minimizes) a weighted sum of different goals, so goals are optimization criteria. For each ODM goal, you can see (in the ODM Studio Goals window) its value in the current solution. You can also specify its weight, see the goal breakdown, search the goal bounds, and add constraints to its value.

Goals editor In the ODM Enterprise IDE, this window is used to edit and customize the requirements that are displayed in the ODM application.

Goals window A built-in component of every ODM application, this window allows users to view and configure the goals of the application.

Filter bar The Filter bar is displayed at the top of many types of views in ODM. It allows you to filter the data in the window in real time, using multiple criteria and wildcard characters.

foreign key A column or columns in a table that does not uniquely identify rows in that table. Rather, it is used as a link to matching columns in other tables. See primary key.

Import Data from ODM button In the ODM Enterprise IDE, the preview area of the Views editor allows you to see how the window or chart you are creating will look, using generated data. If you are running the ODM application in the background, this button allows you to import real data from ODM and display it in the preview area. See also Revert to Generated Data button.

Input Data folder In ODM Studio, this folder is displayed in the Scenario Explorer. It contains links to the input data windows of the ODM application.

Java project One of the component projects that make up an ODM Enterprise application, the Java project contains the code that is used to customize an application.

Key column option A check box option on the ODM Studio Legend panel that displays the key columns of a table in a specific color in the Input Data and Solution windows.

key columns In an ODM table, the columns that form the unique (non-repeatable) key to the table. For example, if the *key columns* of a scheduling application are DepartmentName, Workday, and Shift, no two rows in the table can contain the same DepartmentName, Workday, and Shift.

Legend panel A window in ODM Studio that allows you to highlight four types of information in ODM application windows in the color you select. See also Differences option, Frozen values option, Key column option, and Relaxed requirements option.

key performance indicator A *key performance indicator* (KPI) is a scalar output of a model that is arbitrarily designated by users as such. In the ODM documentation, implicitly consider only those KPIs that are relevant for Sensitivity Analysis.

KPI See key performance indicator.

lower bound A limit that indicates the lowest or minimum value that a variable can assume. In ODM studio, this limit is seen after a Search goal bounds operation in the Lower bound field of the Goals window. See also upper bound.

main model See multiple models.

Mono-indexed Mono-indexed charts provide a chart view of a 1D Array.

multi-user mode ODM applications can be configured to run in either single-user mode or multi-user mode. In multi-user mode, many users run the application simultaneously. These users can access workspaces and scenarios that are stored in the ODM Scenario Repository on a central server.

multiple models An ODM application can be based on multiple optimization models. This configuration generally is used when a problem is too large to be solved in one step. In “multimodel” architecture, a large model is decomposed into a *main* model and *submodels*. The results of each solved submodel are aggregated and returned to the main model to produce a global result.

A multimodel problem is solved by using a sequence of different simple engine searches. For example, from the problem data, a first optimization model is created and solved by using some particular engine or algorithm. The results of this first step are used (directly, or after some processing) as data for creating another optimization model. This model can be solved by using the same or another engine or algorithm.

ODM Enterprise project One of the component projects that make up an ODM Enterprise application, the ODM Enterprise project is the master project. It contains most of the configuration information, which includes definition of the views in ODM Studio, deployment information, mapping to the optimization model. It also contains links to the companion OPL, Data, and Java projects.

ODM Enterprise Projects Navigator The primary view that is used in Eclipse to create, view, and edit the objects of the ODM Enterprise application. It is also used to run the application in ODM Studio from the ODM Enterprise IDE.

ODM Studio The IBM ILOG Optimization Decision Manager graphical user interface.

ODM to OPL Mapping Editor The editor in the ODM Enterprise IDE used to edit the mapping between the Application Data Model and the OPL project.

OPL model The files (with an extension of .mod) contained within an OPL project that define the optimization model.

OPL project One of the component projects that make up an ODM Enterprise application, the OPL project contains the optimization model. This model is used to solve the scenarios, OPL run configurations, and settings (.ops) files.

OPL Projects Navigator In the OPL IDE, this window displays the files that are contained in or that are related to an OPL project.

perspective A set of views and features in the Eclipse IDE that allows you to see and work with different parts of your projects. In ODM Enterprise, there are four perspectives commonly used: The ODM Enterprise perspective, the Data perspective, the OPL perspective, and the Java perspective.

Pie chart A circle graph that is used for comparing the parts of a whole to the whole. The area of the circle represents the whole, and the areas of the sectors of the circle represent the parts.

pivot table A pivot table is a powerful data summarization tool that can automatically sort, count, and total data that are stored in tables and display the summarized data. Pivot tables are useful to quickly create crosstabs to display the joint distribution of two or more variables. The user typically sets up and changes the summary's structure by dragging and dropping fields graphically. This "rotation" or pivoting of the summary table gives the concept its name.

Polyline chart A line chart in which the multiple lines can display an additional marker for each data point chart.

primary key A column or columns in a table whose values uniquely identify the rows in the table.

priority In ODM Studio, rules and requirements can be assigned a priority. By increasing or decreasing the priority, the user can create different solutions.

range In mathematical programming, a range is a special type of domain. It is conventionally associated with a variable in an MP. It conventionally has a lower bound and upper bound that define its limits. All the values between the lower and upper bound belong in the range.

reference scenario When you are working with multiple scenarios in ODM Studio, one scenario is always set as the *reference scenario*. When you select the Differences option in the ODM Legend panel, the values in the currently active scenario are compared to the values in the reference scenario. Any differences are highlighted. The reference scenario is always displayed in the ODM Scenarios Overview window.

Relax more requirements button A button that is active in the ODM Solve Progress window if ODM has not yet moved to phase 2 of the feastopt operation. It allows you to instruct CPLEX to relax constraints that have higher priorities, and thus possibly find a feasible solution more quickly.

relaxation In operations research, *relaxation* means a loosening of the constraints. This process is intended to find a solution that might not be the optimal feasible solution of the original problem, but is good enough in some sense. One conventional relaxation in a mathematical programming problem that is expressed as integer variables allows floating-point solutions that are sufficiently close to integer solutions. This relaxation is known as relaxing the variables. Another type of relaxation is to remove certain constraints. These constraints are known as relaxed constraints or soft constraints.

Relaxed requirements option A check box option on the ODM Studio Legend panel that displays any requirements that are relaxed in the current solution in a specific color.

repository The ODM Scenario Repository is in a database, either installed on the user's workstation or on a shared server. Workspaces, scenarios, data, and the configuration information for the individual scenarios and the ODM application itself are stored in the repository.

requirements Constraints in the OPL model are mapped to *requirements* in the ODM application. Requirements can be assigned a priority in the ODM Requirements window to allow the user a finer level of control over which requirements are relaxed.

Requirements editor In the ODM Enterprise IDE, this window is used to edit and customize the requirements that are displayed in the ODM application.

Requirements window A built-in component of every ODM application, this window allows users to view and configure the requirements of the application.

The Requirements window remains empty for Constraint Programming and the CPLEX Solve algorithm, as they do not support requirement relaxations.

Revert to Generated Data button In the ODM Enterprise IDE, the preview area of the Views editor allows you to see how the window or chart you are creating will look, using generated data. This button allows you to revert to the generated data after you click the Import Data from ODM button.

rule Rules in ODM enable you to directly control some of the constraints in the optimization problem. You can set priorities on constraints, create new constraints, and deactivate certain constraints.

rule category In ODM Studio, a set of rule templates that can be grouped, such as being mixed in the same folder.

rule template In ODM Studio, a rule template is a type of rule. It is defined by:

- ▶ Its message (how it looks)
- ▶ The input data that it is linked to
- ▶ Sometimes, the constraint it controls, and to which a priority is applied.

rule view In ODM Studio, a rule view is the view that displays a rule category.

Rules editor In the ODM Enterprise IDE, this window is used to edit and customize the rules, rule categories, and rule templates that are displayed in the ODM application.

run configuration A *run configuration* is a way of handling model, data, and settings files within a project. Basically, it is a variation of a project for execution and testing purposes. It combines a model file and one or more data files that differ, regarding contents or settings, from the original model and data of the project. However, it addresses the same mathematical problem. You can define as many run configurations as you need within an OPL project.

SA See sensitivity analysis.

scenario A *scenario* is an instance of the ODM Enterprise Application Data Model in ODM Studio. A scenario represents the model instance as a collection of elements: Data, Model, Goals, Rules, Requirements, and Solutions.

Scenario Explorer The window in ODM Studio in which users interact with the different windows and charts of the currently active scenario.

scenario locking When a scenario is being edited or solved by one user in a multi-user environment, that scenario can be *locked* for other users. This procedure prevents them modifying the scenario. Scenario locks can be released by users who have the rights to do so.

scenario sharing Scenarios that are stored in workspaces in an ODM Scenario Repository on a central server can be shared among many users. ODM monitors shared access to these scenarios, and prevents two users from changing the same scenario at the same time.

Scenario Status panel A window in ODM Studio in which overview information is displayed for the currently active scenario and its most recent solution.

Scenarios Overview The window in ODM Studio in which users interact with different scenarios. These scenarios can be copied, modified, and stored in folders in a tree structure.

scenario up to date A scenario is up to date when no input data has changed since the last solve. This status can be seen in the Scenario Status panel, in the Result up to date column.

Search goal bounds In ODM Studio, this command on the Goals window allows you to search for the upper bound and lower bound of a particular goal. This range can then be used as a guideline when you are setting goal constraints.

sensitivity analysis Sensitivity analysis (SA) is a technique of assessing the extent to which changes in assumptions or input variables will affect the ranking of alternatives.

single-user mode ODM applications can be configured to run in either single-user mode or multi-user mode. In single-user mode, the user runs the application locally and can access workspaces and scenarios that are stored in the local ODM Scenario Repository.

Solution folder In ODM Studio, this folder is displayed in the Scenario Explorer, and contains links to the solution windows of the ODM application.

Decision Process Configuration A section in the ODM Enterprise project that contains files that describe the solve process configuration that links the ODM Enterprise project to the OPL project:

- ▶ `<project_name>_opl_optimmodel.odmom:`
Contains descriptions of the components of the optimization model, including requirements, goals, and rules.
- ▶ `<project_name>_opl_optimmapping.odmopl:`
Contains descriptions of how the components of the application data model are mapped to the model of the associated OPL project.
- ▶ `tasks.odmdpc:` Contains descriptions of the tasks for this ODM application, which can be either OPL tasks or custom Java tasks.

Solve Progress window In ODM Studio, when you click **Solve** to solve a problem, the Solve Progress window is displayed. The Goals Progress area shows how well the current solution is doing in achieving the Combined Objective and the individual goals of the ODM application. There is one section for the Combined Objective, and one for each individual goal. You can use these indicators to judge the progress of your solution.

The Solution Progress area at the bottom of the window shows whether a solution has been found. It provides you with the ability to stop the solve and relax more requirements.

Solve Queue When multiple solve processes are run in the background by clicking **Continue in background** on the Solve Progress window, they are added to the *Solve Queue*. They are solved in the order in which they were submitted. Users can check on the progress of the current solve process or cancel it using menu options and toolbar buttons.

solveAnyway The solveAnyway algorithm builds on feaspopt by adding flexibility and human priorities. What this means is that solveAnyway systematically adds constraints in the “relaxable pool” of constraints, starting with those that have the lowest priority. ODM applications allow users to set constraint priorities in the scenario creation process.

Stair chart A Stair chart represents a transition between two values as a stair instead of straight lines.

submodel See multiple models

upper bound A limit that indicates the highest or maximum value that a variable can assume. In ODM studio, this limit is seen after a Search goal bounds operation in the Upper bound field of the Goals window. See also lower bound.

view filter Many types of views in ODM have a Filter bar at the top of the window. This bar allows you to filter the data in the window in real time, using multiple criteria and wildcard characters.

views.odmvw A file in the ODM Enterprise project that contains the description of how the input and output data for the ODM Enterprise application is displayed in ODM Studio. Double-click this file to access the ODM Views editor.

Views editor In the ODM Enterprise IDE, this window is used to edit and customize the input data and solution views, rules views, and chart views that are displayed in the ODM application.

Workspace Workspaces are used in ODM to organize scenarios. They can contain folders and subfolders. When you save in ODM, your workspace is stored in the ODM Scenario Repository. If the repository is on a shared server, other users can access and share the scenarios in any of the workspaces that are stored there. Workspaces and their scenarios can also be exported and saved on a file system, for sharing with remote users.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *IBM ILOG ODM Enterprise and Data Integration*, REDP-4740
- ▶ *Analytics in a Big Data Environment*, REDP-4877
- ▶ *IBM SPSS predictive analytics: Optimizing decisions at the point of impact*, REDP-4710
- ▶ *IBM Cognos Business Intelligence V10.1: Intelligence Unleashed*, REDP-4693
- ▶ *IBM Cognos Business Intelligence V10.1 Handbook*, SG24-7912

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Education

ODM Enterprise is both a development environment and an end-user environment for custom optimization-based planning and scheduling decision-making applications.

Two types of developers are involved in the development process:

- ▶ The Operations Research (OR) developer uses ODM Enterprise to write the optimization models underlying an ODM Enterprise application, and to develop ODM Studio prototypes. This developer should take the *Introduction to Application Development with IBM ILOG ODM Enterprise V3.4* course.

- ▶ The IT developer uses ODM Enterprise to develop ODM Enterprise applications and to customize, extend, and integrate the application. This developer should take the *Developing and Deploying Complete Applications with IBM ILOG ODM Enterprise V3.4* course.

Which version: When this book was created, IBM ILOG ODM Enterprise V3.6 was available. Although the classes described are focusing on V3.4, they are considered the basic education for ODM Enterprise in general. Watch for updates in the IBM education curriculum.

Course: Introduction to Application Development with IBM ILOG ODM Enterprise V3.4

This 2-day course teaches students how to create basic ODM Enterprise applications that can be used as prototypes or starting points for more complex applications. It also gives an overview of ODM Enterprise to team members who are not directly involved in development so they can collaborate effectively with colleagues who develop and deploy complete ODM Enterprise applications.

ODM Enterprise is a configurable and extensible platform for building optimization models and using them to generate customized planning and scheduling applications. These applications allow users to compare multiple scenarios and perform “what-if” analysis.

Objectives

After completing this training, students should be able to:

- ▶ Describe the architecture of ODM Enterprise
- ▶ Explain how each ODM Enterprise module works
- ▶ Create an ODM Enterprise Application Data Model
- ▶ Create standard ODM Enterprise views by using the ODM Enterprise integrated development environment (IDE)
- ▶ Integrate an ODM Enterprise application with .csv or database datasources
- ▶ Map an Optimization Programming Language (OPL) optimization model to an ODM Enterprise Application Data Model
- ▶ Deploy an ODM Enterprise application to a single-user environment

Course: Developing and Deploying Complete Applications with IBM ILOG ODM Enterprise V3.4

This 3-day course teaches students how to configure, extend, and integrate ODM Enterprise applications.

ODM Enterprise is a configurable and extensible platform for building optimization models and using them to generate customized planning and scheduling applications. These applications allow users to compare multiple scenarios and perform “what-if” analysis.

Objectives

After completing this training, students should be able to:

- ▶ Describe the ODM Enterprise architecture
- ▶ Explain how each ODM Enterprise module works
- ▶ Create an ODM Enterprise Application Data Model
- ▶ Create standard ODM Enterprise views by using the ODM Enterprise IDE
- ▶ Create custom ODM Enterprise views by using Java
- ▶ Integrate an ODM Enterprise application with .csv, database, or custom datasources
- ▶ Deploy an ODM Enterprise application to a single-or multi-user environment
- ▶ Map an OPL optimization model to an ODM Enterprise Application Data Model
- ▶ Configure and customize an ODM Enterprise application by using Java
- ▶ Configure and customize an ODM Enterprise application by using XML
- ▶ Integrate an ODM Enterprise application with existing enterprise analysis tools
- ▶ Administer the ODM Enterprise optimization server

Other publications

These publications are also relevant as further information sources:

- ▶ *Supply Chain Management on Demand: Strategies, Technologies, Applications*, Chae An and Hansjörg Fromm, Springer Verlag, Berlin Heidelberg 2005, ISBN 978-3540244233

- ▶ *Competing on Analytics: The New Science of Winning*, Davenport, T. H. and Harris, J. G., Harvard Business School Press, 2007, ISBN 978-1422103326
- ▶ *A New Era for Crew Recovery*, Yu, G. et al., Interfaces, Vol. 33, No. 1, January–February 2003, pp. 5–22
- ▶ *INDEVAL Develops a New Operating and Settlement System Using Operations Research*, Muñoz, D. F. et al., Interfaces, Vol. 41, No. 1, January–February 2011, pp. 8–17
- ▶ *Operations Research*, S. Nickel et al., Springer Verlag, Berlin Heidelberg 2011, ISBN 978-3642226236

Online resources

These websites are also relevant as further information sources:

- ▶ The information center for IBM ILOG ODM Enterprise V3.6 describes the IBM ILOG ODM Enterprise product and features. All product manuals can be accessed at the following web site.

<http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r6/index.jsp>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Optimization and Decision Support Design Guide: Using IBM ILOG Optimization Decision Manager

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Optimization and Decision Support Design Guide

Using IBM ILOG Optimization Decision Manager



**Best practices
solution design for
optimization and
decision support**

**Complete
information about
architecture and
components**

**Deployment
scenarios with
hands-on details**

Today many organizations face challenges when developing a realistic plan or schedule that provides the best possible balance between customer service and revenue goals. Optimization technology has long been used to find the best solutions to complex planning and scheduling problems. A decision-support environment that enables the flexible exploration of all the trade-offs and sensitivities needs to provide the following capabilities:

- ▶ Flexibility to develop and compare realistic planning and scheduling scenarios
- ▶ Quality sensitivity analysis and explanations
- ▶ Collaborative planning and scenario sharing
- ▶ Decision recommendations

This IBM Redbooks publication introduces you to the IBM ILOG Optimization Decision Manager (ODM) Enterprise. This decision-support application provides the capabilities you need to take full advantage of optimization technology. Applications built with IBM ILOG ODM Enterprise can help users create, compare, and understand planning or scheduling scenarios. They can also adjust any of the model inputs or goals, and fully understanding the binding constraints, trade-offs, sensitivities, and business options.

This book enables business analysts, architects, and administrators to design and use their own operational decision management solution..

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks