Best Practices and Tools for Creating IBM WebSphere Commerce Sites

Presents a WebSphere Commerce development methodology

Describes implementation of WebSphere Commerce sites

Describes the use of the IBM Transition Tool Suite

Ramzan Khuwaja
Corrado Balducci
Vesselin Ivanov
Manivannan Kubendranathan
Lisa Tong
Heidi Yan
Adrian Warman

ibm.com/redbooks
**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.
# Contents

**Notices** .......................................................... vii
**Trademarks** ......................................................... viii

**Preface** .............................................................. ix
The team that wrote this Redpaper .................................. ix
Become a published author ........................................... xii
Comments welcome ................................................... xii

**Chapter 1. Executive summary** .................................. 1
1.1 Further information ............................................... 9

**Chapter 2. WebSphere Commerce site development methodology** .... 11
2.1 Systematic development methodology ............................ 12
2.2 Definitions .......................................................... 13
  2.2.1 Phase ......................................................... 13
  2.2.2 Work product ................................................ 13
  2.2.3 Deliverable ................................................... 13
  2.2.4 Customer ...................................................... 14
  2.2.5 Customer IT team ........................................... 14
  2.2.6 Project team .................................................. 14
  2.2.7 Project database ............................................ 15
  2.2.8 Task ........................................................... 15
  2.2.9 Strategy ....................................................... 16
2.3 Project development methodology: Phase and life cycle ........ 17
  2.3.1 Core development phases ................................... 18
2.4 Using the methodology ............................................ 21
  2.4.1 Customizing and adopting the methodology ............... 22
  2.4.2 New and transition sites .................................. 22
  2.4.3 Project roles and skills requirements .................... 22
  2.4.4 Structuring information ................................... 23
  2.4.5 Case studies ................................................ 23
2.5 Transition Tool Suite: Introduction ............................. 24
  2.5.1 Preprocessor Tools ......................................... 26
  2.5.2 Bean Recommendation Tool ................................ 26
  2.5.3 SQL translation, validation, and bean generation ........ 26
  2.5.4 Data Migration Tool ....................................... 27
2.6 Transition Tool Suite usage ..................................... 28
  2.6.1 Individual tool usage ...................................... 28
  2.6.2 Sequential tool usage ...................................... 28
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 give 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

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 illustrates 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

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

- DB2®
- developerWorks™
- Domino™
- IBM®
- IBM @server™
- Lotus®
- Lotus Notes®
- Net.Commerce™
- Net.Data®
- Notes®
- PartnerWorld®
- QuickPlace™
- Redbooks™
- Redbooks (logo) ™
- VisualAge®
- WebSphere®

The following terms are trademarks of other companies:

- ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

- Microsoft, Windows, Windows NT, 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 Sun Microsystems, Inc. in the United States, other countries, or both.

- C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

- UNIX is a registered trademark of The Open Group in the United States and other countries.

- SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Gone are the days when creating e-commerce sites was considered a fine art requiring highly specialized skills. E-commerce development has become a rather mature field since first introduced in the mid- to late-1990s. Experience proves that using a systematic methodology to develop software applications for e-commerce systems greatly reduces project risk and improves proper alignment of software functionality with customer business requirements, two key influencers of project profitability and customer satisfaction.

This IBM® Redpaper introduces a project development methodology that was specifically adapted for the creation of e-commerce sites using the IBM WebSphere® Commerce family of software products.

This Redpaper details two WebSphere Commerce development scenarios:
1. Creating a new e-commerce site
2. Transitioning an existing WebSphere Commerce site built on the Version 4 framework to Version 5.1 or 5.4

This Redpaper describes and documents the use of IBM Transition Tool Suite during a transition project to highlight for the reader how helper tools can be used to automate some, or many, of the transition tasks that are associated with transitioning WebSphere Commerce assets or data. When used properly, helper tools will reduce the overall migration effort and maximize WebSphere Commerce asset or data reuse. Examples are provided to identify for you how the helper tools can be used independently of the development methodology or with it to improve your efficiency. The objective is to document how the tools can be combined with the development methodology to streamline a WebSphere Commerce transition project.

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working through the International Technical Support Organization, Raleigh Center.

Ramzan Khuwaja is a Project Manager with the WebSphere Studio Tools Release team. Ramzan has been with IBM Canada Toronto Laboratory for the last four years. Prior to joining the WebSphere Studio Tools team, he worked with the WebSphere Commerce Services team as Solution Architect and Project
Manager. In that capacity, Ramzan was a driving force to help manage large scale e-commerce projects for customers (including the ShopIBM site). Ramzan also played a key role in conceptualizing, developing, and applying Transition Tool Suite, an IBM product to help migrate e-commerce sites to newer versions of the product. Ramzan brought several years of industry experience to IBM, focusing on using Web and Internet technologies for pervasive computing and distance learning and training areas. Ramzan has a Ph.D. in Computer Science (Artificial Intelligence) and is the author of several research publications and patents in these fields. Ramzan also has a personal interest in the fields of knowledge management and organization learning and their relation to the social context in a corporate environment.

**Corrado (Cory) Balducci** has amassed over five years of application development and design, project management, and testing experience as a Software Engineer for the WebSphere Commerce Engagement Services team, based in the IBM Canada Toronto Laboratory. In particular, as a Solution Architect and Technical Lead, he has been designing, analyzing, migrating, and testing applications built on IBM WebSphere Commerce. Cory was instrumental in developing, testing, and then documenting the Data Migration Tools that are integrated with Transition Tool Suite.

**Vesselin (Vess) Ivanov** is a Senior IT Specialist with the Software Services for WebSphere team at the IBM Toronto Lab. He is an IBM Certified WebSphere Developer and Sun Certified Java Programmer and has worked with WebSphere customers throughout North America for the last six years. Vess created LOQS while working on WebSphere Commerce projects.

**Manivannan (Mani) Kubendranathan** is a Lead Software Developer for the WebSphere Commerce Engagement Services team. Over the past three years, he has worked closely with clients to design e-commerce solutions based on IBM WebSphere Commerce, and as a result of his engagement experiences, he possesses extensive knowledge related to design and development best practices for WebSphere Commerce.

**Lisa Tong** has been with the Toronto Laboratory in IBM Canada on the Web Services team for almost two years. She has worked on various engagements ranging from WebSphere Commerce 4.1 to 5.4 and, most recently, on a customer engagement performing a WebSphere Commerce Suite 4.1 to 5.1 migration. Lisa is currently working with the WebSphere Build team and aiding in script changes, build improvements, and WebSphere Studio Application Developer migrations.
Heidi Yan is a Solution Architect with the WebSphere Commerce Engagement Services team, based in the IBM Canada Toronto Laboratory. Over the past three years, Heidi has been leading development engagements that employ the advanced features and functions offered by IBM WebSphere Commerce. Her engagements experiences, in addition to her prior development responsibilities within IBM, have enabled Heidi to develop broad knowledge of e-commerce processes, namely B2B and B2C.

Adrian Warman is an IT Specialist in the Hursley Laboratory of IBM UK. He has been working with computers since 1982. He holds a degree in Computer Science and a Ph.D. in Information Systems, both from the University of London. His areas of expertise include WebSphere Commerce, XML and XSL/T, databases, Java, and Linux. He has written extensively about information systems and computer security.

Thanks particularly to the following people for their contributions of material to this project:

Don Carr
Don is a Services Planner for the IBM Software Services Planning and Marketing Team. He works closely with the IBM WebSphere Commerce Product Development Teams to identify software services requirements, assess them, and where appropriate, to develop, market, and distribute required software services assets, such as services engagement materials, service Redpapers, and code assets.

Feras Dawisha
Feras is a Software Architect with the WebSphere Commerce Engagement Services team. He is based in the IBM Canada Toronto Laboratory where he has been involved in the WebSphere Commerce product family for the past five years and, prior to that, spent eight years in the DB2® database development team.

John Ganci
John is a Senior Software Engineer, WebSphere Specialist at the IBM ITSO, Raleigh Center. He writes extensively and teaches classes on WebSphere and related topics. John has 14 years of experience in product and application design, development, system testing, and consulting. His areas of expertise include e-commerce, pervasive computing, and Java programming. Before joining the ITSO, he developed e-commerce sites for IBM.

Thanks particularly to the following people for their feedback and contributions to this project:

Bill Baker, IBM Canada
Edd Biddle, IBM UK
Yumman Chan, IBM Canada
Linda Robinson, IBM ITSO, Raleigh Center
Margaret Ticknor, IBM ITSO, Raleigh Center

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks™ in one of the following ways:

► Use the online Contact us review redbook form found at:
   ibm.com/redbooks

► Send your comments in an Internet note to:
   redbook@us.ibm.com

► Mail your comments to:
   IBM Corporation, International Technical Support Organization
   Dept. HZ8 Building 662
   P.O. Box 12195
   Research Triangle Park, NC 27709-2195
Chapter 1. Executive summary

Using a systematic methodology to develop software applications for e-commerce systems greatly reduces project risk and improves alignment of software functionality with a customer’s business requirements, which are two key influencers of project profitability and customer satisfaction. This IBM Redpaper introduces a project development methodology that was specifically adapted for the creation of e-commerce sites using the IBM WebSphere Commerce family of products. Not only does it support new implementations of the technology, but the methodology can also be used to transition current assets and data to the latest WebSphere Commerce version. In addition, the methodology may be combined with tools to streamline the entire WebSphere Commerce transition process.

A typical project development methodology consists of six phases. Within this paper, we focus on the following four core phases for commerce site development:

1. Design
2. Implementation
3. Site testing
4. Deployment

These core phases fall between the pre-sales phase and the site maintenance and enhancements phase of a project life cycle. A diagrammatic overview of the project development methodology is shown in Figure 1-1 on page 2.
**Figure 1-1** Overview of project development methodology

- **Pre-sales phase**
  - Gather business requirements and initial fit-gap

- **New or Transition**
  - Gather new site requirements
  - Transition current site information

- **Solution Outline Subphase**
  - Review requirements documents
  - Build initial solution outline

- **Solution Outline Workshop**
  - Review initial solution outline and site flow with customer
  - Present additional function for future requirements

- **Micro Design Subphase**
  - Build high-level site design, project plan, test plan and DB design

- **Subphase**
  - Identify implementation team
  - Build development/test/integration environment
  - Build back-end simulation environment

- **Implementation Phase**
  - Code/unit test based on design
  - Integrate with simulated back-end environment

- **Test phase**
  - FVT - Functional Verification Test
  - SIT - System Integration Test
  - SVT - System Verification Test
  - UAT - User Acceptance Test

- **Deployment/Launch phase**
  - Prepare host systems
  - Apply final, verified code

- **Business requirements**
  - Fit-gap analysis
  - Staff small architecture team

- **Current/new site flow**
  - Current/new design docs
  - Current/new schema
  - Current code for transition

- **Use cases**
  - Site flow
  - Business rules
  - Integration
  - Initial project plan
  - Initial skills plan

- **Solution outline document**
  - Refined Solution Outline
  - Phase documents

- **Review ed and approved macro design document**
  - Revised project plan
  - Test plans/strategies
  - High-level DB design

- **Review ed and approved micro design document**
  - Detailed project plan
  - DB design/schema
  - Staff additional design team

- **Coded/unit tested/reviewed site**
  - Approved test cases
  - Staff development team

- **Fully tested site ready for deployment**

- **Live site**

---

2 Best Practices and Tools for Creating IBM WebSphere Commerce Sites
During the pre-sales phase, the customer must assess their business requirements objectively, and then plan their e-commerce site carefully, because results derived from these activities will set the direction and help define the duration of key project activities. It is critical for the customer to define and then document all their business requirements adequately prior to the completion of this phase; the results produced by these activities will be seed site design work.

The design phase marks the beginning of a WebSphere Commerce project, from the perspective of a services engagement, and during this phase, project teams work to define and document the site’s design based on the requirements that result from the pre-sales phase. All design activity can be split into the following three distinct subphases:

1. Solution outline
2. Macro design
3. Micro design

During the solution outline subphase, the project team identifies all relevant functional requirements for the site from the predetermined business requirements. All functional requirements should be recorded in a solution outline document that would consist of the following components:

1. Project team skills plan
2. High-level project plan
3. Revised sizing estimate (that is, work effort) required to develop the site

A concluding task in creating the solution outline document is where the project team works with the customer team in a solution outline workshop to review the project’s functional requirements and plans. Where necessary, each might be adjusted to gain customer approval in order for the design work to start proper.

Prior to working on the design phase, the project team itself must be organized by the project manager or lead. The project team plays a significant role during the design phase in aiding the customer to define and document site requirements and project plans.

This team must consist of dedicated, suitably skilled individuals who can be relied upon to complete all required tasks related to the design and implementation of a target site. A project manager is likely to assess the following project factors when identifying team resource requirements and considering how best to structure a project team:

- Skill and experience requirements
- Project objectives
- Project timeline
Customer requirements

Geographic considerations

Direct customer involvement in project tasks

Training requirements

A thorough understanding of these factors will enable the project manager to pursue an appropriate strategy for organizing and structuring the team accordingly. It will also enable them to staff the team with appropriate skills to match the project and customer objectives. Two proven team assignment strategies are as follows:

- Core competency strategy
- Holistic strategy

The core competency strategy advocates using expert resources to staff and complete project tasks based on skill. This strategy fosters specialization, because team members are able to hone and refine their expertise in given areas. Generally, when team members complete a project that employed this strategy, they are capable of undertaking more challenging design and development responsibilities in the future.

The holistic strategy, on the other hand, stresses using resources with strong fundamental e-commerce design and development skills to make each fully capable of performing all project tasks. One result of this strategy is a workforce that is versatile and adaptable at performing diverse work assignments. In addition, it enables the project manager to reposition resources as needed during the life of the project to aid the team in maintaining the integrity of the project schedule.

The project manager may also use an appropriate combination of the above strategies to satisfy particular project demands. Again, it is critical that the project manager fully assess the nature and scope of the project to determine how best to structure a project team.

It is during the design subphases that the project team strives to create and document the project's functional requirements and requisite project and skill plans.

The project team is likely to employ one of the following strategies to complete the remaining design subphases:

- Iterative and incremental design strategy
- Combination design strategy
The iterative and incremental design strategy advocates working through each of
design's subphases in sequence so that details captured during the earlier
solution outline subphase can be used and further extended by the subsequent
subphases. In other words, the site design gets built and refined over time as the
team works through each subphase. This strategy work best on projects
involving complex business logic.

The combination design strategy, on the other hand, enables the project team to
combine the activities of related subphases into one phase. For example,
solution outline may be combined with macro design, or macro design and micro
design may be combined. This strategy is suited for projects with simpler
designs, because it allows the project teams to complete related design activities
in parallel to reduce the length of time need to complete the project.

The main focus of macro design is the creation of a WebSphere Commerce
design that incorporates the requirements that were approved during the solution
outline. The project team may pursue one or a combination of the following
design methods:

- System level (for example, WebSphere Commerce interacting with external
  systems)
- Subsystem level (for example, interactions among WebSphere Commerce
  subsystems)
- Use case level (for example, site flows as identified in use cases)
- Component level (for example, interactions among JSPs and EJBs)

These factors enable the project team to understand and then design a site given
the approved requirements. Very detailed examples and project
recommendations are provided throughout this Redpaper.

Once the macro design requirements have been defined and documented in the
macro design document, the project team may then proceed with the micro
design subphase. The main objective of this third iteration of the design phase is
for the project team to identify, and then document, the detailed design decisions
that are required to implement the target site. For example, the project team will
want to document the required interfaces between commands, JavaServer
Pages (JSPs), and the persistence layer of Enterprise JavaBeans (EJBs), data
beans, and possibly, LOQS objects. Comprehensive overviews for managing
data access using the Transition Tool Suite, which will be discussed later in this
summary, is provided in this Redpaper.

Completion of all design activities enables the project team to initiate the
implementation phase. During this phase, the project team codes the site
elements, or components, and when done, it unit tests each prior to integrating
the components into builds. The recommended development strategy a project
team should pursue is the continuous and incremental development and code integration testing strategy. This strategy advocates anchoring development on use cases and then testing and integrating code components into builds, which, in turn, must be tested prior to deployment using a Build Verification Test (BVT) process. The build process must define:

- A build lead who is responsible for managing the process
- A build schedule outlining when builds will be made available
- The established roles and responsibilities for team members who create and use builds
- Build deployment and build failure processes

In addition to establishing the build process, the project team must also define and implement an integration test strategy. This will enable the team to test and, as needed, to correct code components of builds as they are integrated into the appropriate WebSphere Commerce subsystems.

Extensive details are provided throughout this Redpaper with recommendations and guidance about how to:

- Subdivide implementation responsibilities
- Structure appropriate BVT and build deployment processes
- Use the Transition Tool Suite during integration testing
- Support error handling after code implementation and site launch

This Redpaper identifies a systematic strategy for fully testing a commerce site. The site may either be original or one that has been newly transitioned from an older site. The objective of the test strategy is to ensure that the customized site components interact correctly with each other, as well as with all relevant out-of-the-box components. The key site testing phases are as follows:

1. Functional Verification Test (FVT)
2. System Integration Test (SIT)
3. System Verification Test (SVT)
4. User Acceptance Test (UAT)

Employing a systematic test strategy streamlines the site testing process, which results in the project team being able to deliver operational site code faster and with improved reliability. To support the site testing strategy effectively, the project team must first create a suitable testing infrastructure, as well as define the testing process. The infrastructure must reasonably mirror the production system as closely as practical, in terms of interfaces with third-party software or system connections, if it is to produce meaningful test results. The testing
process will also aid the project team in pursuing the test strategy, allowing the
team to complete each test phase, identify bugs, work to correct them, and retest
the components. The process should also define how the project team that
designed the site may best be divided into the test and problem fixing subteams
to support all test activities efficiently.

The actual testing of code may follow either a single-pass test strategy or a
multi-pass test strategy. The single-pass strategy advocates that the test team
conduct one pass of a test case to determine the code's integrity. This strategy
lends itself to projects where there is limited interaction between use cases and
test cases. The multi-pass test strategy, on the other hand, requires the project
team to conduct two or more passes through a test case to verify the code
functions as designed. This test strategy is suitable for projects where there is
considerable interrelationship between use cases and test cases.

In terms of the test phases, FVT requires the test team to function test site code
from end-to-end using a suite of test cases to ensure it operates as designed.
SIT, on the other hand, is an interim test phase, because it enables the project
team to assess the test system (using a simulated production environment) to
ensure it can and will support SVT and UAT. SVT marks the commencements of
the final tests, and it enables the test teams to validate the site code that resulted
from the implementation phase. The customer IT team should undertake most of
this testing responsibility, because they are best suited to assess the
functionality of the site code to ensure it satisfies the stated business
requirements. Doing so also prepares the customer team to lead UAT, which
requires the testing of site look-and-feel, site flows and behavior, as well as
validating business processes. These tests should be performed by the
customer's business users.

The key reason for employing a systematic test strategy is to ensure all code
operates as designed, and if any bugs are found, that they may be passed to the
problem fixing team for prompt resolution. This Redpaper provides numerous
details and tips for supporting and pursuing the recommended test phases.

This Redpaper concludes with a summary of the steps and considerations a
customer IT team should address prior to their deploying the site on a production
environment. During this phase of a project, the project team must assist the
customer with determining if the production environment is capable of supporting
either the new site code or the transitioned code. The key activities include:

1. Hardware preparation
2. Prerequisite software preparation
3. Hardware and software configuration
4. Implementation of tested and approved code
In addition to these preparatory activities, the customer IT team is encouraged to perform contingency planning to compensate for potential issues or failures that may result after launch. Core among these activities is the need to define and document how the existing site can be restored, as well as how the new site can be tested, repaired, and re-launched.

Again, the purpose of this Redpaper is to present a viable development methodology that a WebSphere Commerce development team can use to create or transition WebSphere Commerce sites. In addition, this Redpaper provides many recommendations regarding team structure; design, implementation, and test strategies; and it reinforces each with real project examples of how one might perform specific activities during a new implementation or a transition project. Interspersed throughout this paper are details regarding the use of the Transition Tool Suite and how it may be used to transition specific WebSphere Commerce assets and, in some cases, to deploy new ones.

The Transition Tool Suite (TTS) is an integrated set of helper tools that were designed to reduce the work effort associated with transitioning WebSphere Commerce assets and data from the earlier Net.Data® and C++ programming model versions of the product to the current Java and J2EE model, specifically WebSphere Commerce Suite Version 5.1 and WebSphere Commerce Version 5.4. TTS tools can be used throughout the project development methodology to reduce redundant tasks and improve the transition of many assets.

TTS ships in two versions. TTS Version 1 supports sites built using WebSphere Commerce Version 5.1, and TTS Version 2 supports sites built using WebSphere Commerce Version 5.4). TTS provides the following tools:

- Preprocessor Tools
- Bean Recommendation Tool
- SQL Collector, Extractor, and Translator Tools
- Lightweight Object Query System (LOQS) Code Generator and Runtime component
- Data Migration Tool
- Document Helper

The Preprocessor Tools come preconfigured to recognize standard, out-of-the-box WebSphere Commerce assets, logic, and data. They are easily adaptable so that TTS can be made to recognize custom assets and data for bean recommendation and data migration.

For example, the Bean Recommendation Tool enables the user to identify new out-of-the-box access beans that may be used in place of existing SQL statements, which are very common in older sites.
Where LOQS plays a significant role is in providing efficient read-only access to either newly transitioned or new site data. Users can convert SQL statements into data access objects through sequential usage of SQL Collector, SQL Extractor, or SQL Validator Tools with LOQS Code Generator. Once generated, the objects can be run under either old or new sites through the LOQS Runtime component, which is a stand-alone component that ships with TTS and must be installed separately from IBM WebSphere Commerce.

To ensure existing site data is available through the newly transitioned site, the user will want to use the Data Migration Tool to perform the data migration, because it is preconfigured to support standard WebSphere Commerce data and schema. In addition, this tool can be adapted easily to support custom data and schema migration.

This Redpaper provides a high-level overview of the tools and how each tool may be used. This paper has specific sections that detail unique usage possibilities during key phases of the development methodology. It is important to note that TTS has the potential to reduce the work effort of a transition project by up to 30%, which when viewed against the total requirement, can equate into a sizable number of worked hours.

To conclude, the key premise of this Redpaper is that tools, when combined properly with a logical development methodology, can produce significant benefits to a project team and a customer so that either may improve its ability to create or transition WebSphere Commerce assets efficiently.

1.1 Further information

IBM consultants are available to aid your implementation of the WebSphere Commerce development methodology as discussed above and in greater detail within the body of this Redpaper. They can provide guidance on how best to use TTS given specific requirements.

IBM Software Services for WebSphere is comprised of highly skilled consultants with broad architectural knowledge, deep technical skills, and who may access lead developers and product support personnel in IBM research and development facilities. This services team provides worldwide assistance and support for WebSphere products to customers and partners alike through services offerings that make it easy to design, build, test, and deploy applications for e-business.

By engaging IBM Software Services for WebSphere, you not only have access to some of the deepest product consulting and training skills in the world, you also have a team dedicated to help ensure the success of your e-business initiative.
To learn how this team may assist you to implement IBM WebSphere Commerce efficiently, please access:

http://www7b.software.ibm.com/wsdd/services/
WebSphere Commerce site development methodology

In this chapter, we introduce a project development methodology that was specifically adapted for the creation of e-commerce sites using the IBM WebSphere Commerce family of software products. After reading this chapter, you should have a clear understanding of the following topics relevant to the development methodology:

- Key development phases
- Actions and terms
- Deployment conditions
- Transition Tool Suite and its linkage

All subsequent chapters build upon the overview provided here.
2.1 Systematic development methodology

Gone are the days when creating e-commerce sites was considered a fine art requiring highly specialized skills. E-commerce development has become a rather mature field since first introduced in the mid- to late-1990s. Experience proves that using a systematic methodology to develop software applications for e-commerce systems greatly reduces project risk and improves proper alignment of software functionality with customer business requirements, two key influencers of project profitability and customer satisfaction.

The key characteristics of a systematic development methodology are:

- It uses a systematic software development method.
- The method has been validated and refined through empirical experience gained through e-commerce development.
- The framework of the methodology is centered around established project management principles to help define work products, deliverables, and time-lines (or phases) to deliver software on-time and within budget, to meet a customer's business needs.

The methodology outlined in this chapter is flexible to support the creation or transition of both small- and large-scale e-commerce sites. Prior to introducing the methodology, this chapter first identifies and defines relevant terminology that is used through the remainder of this paper.

This Redpaper covers two scenarios in developing WebSphere Commerce sites:

1. Creating a new e-commerce site
2. Transition of an existing version of WebSphere Commerce Suite to a newer version of the product

Migrating, or more appropriately, transitioning, WebSphere Commerce Suite (WebSphere Commerce Suite v4.x or earlier versions) sites to the Java/J2EE releases of WebSphere Commerce (WebSphere Commerce v5.x or later) requires careful planning and a systematic approach.

The transitioning of existing site assets (for example, code, data, setup and configuration information, software, and hardware base) requires repetitive and meticulous tasks, and helper tools, such as the Transition Tool Suite, can be employed to minimize or streamline, or both, these tasks.
2.2 Definitions

The methodology documented in this Redpaper uses specific terminology with unique connotations. To improve your understanding of the project development methodology, this Redpaper begins with an overview of terminology definitions. This section establishes a vocabulary that:

1. Is used throughout the remainder of this Redpaper.
2. Enables you to use this vocabulary when communicating and sharing knowledge of the methodology with others.

2.2.1 Phase

The development methodology consists of a set of time- or function-bound phases (for example, the design phase). From a system’s view, a phase has very clear input and output work products (for example, the customer requirements document). A phase takes one or more work products and performs a set of activities or tasks to enhance existing work products or create new work products. A phase is generally closed before starting the next one in sequence. A phase can have zero or more subphases. In this Redpaper, the word phase refers to both a phase and a subphase of a project.

Note: Other industry methodology standards may use the term *activity* as a subdivision of a phase.

2.2.2 Work product

Work products are tangible pieces of information or artifacts that can be created, shared, and communicated. For example, work products may be the following:

- An education or training roadmap
- A macro design document
- Deployment code
- An e-commerce database with customer information

2.2.3 Deliverable

A work product becomes a deliverable when it has been highlighted in a contractual document that has been signed by both the customer and a services provider, such as a Statement of Work (SoW) or a Document of Understanding (DoU).
2.2.4 Customer

The customer should be viewed as one who either owns an e-commerce site or who wishes to develop an e-commerce site. In either case, the customer defines and approves the business requirements for the e-commerce site.

2.2.5 Customer IT team

A team consisting of one or more members, each assigned by the customer, that is responsible for maintaining the customer e-commerce site and associated computing infrastructure, such as the back-end or legacy systems.

2.2.6 Project team

Owing to its responsibilities, the project team is prominently positioned with the methodology in this Redpaper, because the methodology seeks to emphasize the activities of this team.

Because a project team may consist of subteams, such as a design team or test team, the appropriate subteam will be identified by name if a methodology task is its responsibility. In other words, the project team will be the primary reference team used either explicitly or implicitly throughout this paper unless specified otherwise.

In terms of projects, a project team is sanctioned and supported by the customer during an in-house project, while during a consulting engagement, the project team is controlled and supported by outside services organizations, such as ISSW, IGS, or IBM Business Partners. No matter the allegiance of the project team, all should carefully assess project execution and control requirements whenever team members will be geographically dispersed.

The assessment would typically be based upon evaluation of factors, such as frequency of communication, the quantity and nature of control information supplied, and the difference between expected and observed progress in project execution.

Each project manager or leader will need to understand the working relationships between team members to plan and coordinate project activities appropriately, especially when the teams reside in different locations. This topic is an import part of project management discipline, such as the PMI standard adopted by IBM. However, as further information about implications of dispersal on projects would be available from a project management specialist, we do not discuss the topic further in this Redpaper.
2.2.7 Project database

A database of project-specific information should be established and maintained for all team members to access. It would hold background information, reports, minutes of meetings, results of discussions, and indeed, any other kind of information that is relevant to the execution of the project.

The project database could be implemented using any available tool, for example, a Lotus® Notes® database. It may be better to make use of collaborative facilities, such as those provided by Lotus QuickPlace™, which supports not only team information and resources sharing, but can also support controlled external access by customers using Web browsers.

2.2.8 Task

A task is an activity that is performed during one or more phases of the project.

A typical project phase generally involves many tasks, and each may be executed either sequentially or in parallel with one another. In addition, depending on their individual requirements, tasks may be completed by individual team members or distinct teams or subteams.

A task is likely to require input in the form of work products, and once it has been completed, it will produce one or more output work products. For example, when creating a database, you would require a data model for the target database to define and create the database schema. When complete, the target database may then be populated with customer or sample data to create the required database. In this case, the input work products necessary to create a sample database are as follows:

1. The data model
2. The database schema
3. The sample data

The output work product of the database task is a sample database ready to be used by the project team during the implementation phase.

The scope of a project task can be contained within a single phase or across multiple phases. An example of multi-phase task is the creation of a WebSphere Commerce design. This task can be completed by one or more team members, using most of the output work products from the pre-sales phase, which is described in “Pre-sales phase” on page 138. The output of the design task would normally be a set of work products that are listed as output from the micro design phase.
Ideally, tasks are completed within their parent phases. However, under some circumstances, often dependent on the task itself, it may be necessary to revisit a task during a later phase. This is particularly the case when later work uncovers an issue that must be revisited, such as modification to a database design. Such situations should be considered exceptions, and treated as such.

### 2.2.9 Strategy

In terms of this Redpaper, a strategy is a method for executing a task. A project team may execute tasks in a variety of ways to produce the desired output work products. Therefore, many strategies may be used to complete specific project tasks and create the associated output work products.

For example, it is possible to use either the Transition Tool Suite (TTS) or hand-coded SQL scripts to perform commerce data migration during a transition project. Using TTS is one strategy for creating a target database, while using hand-coded SQL scripts represents an alternate strategy. Both strategies should produce the same output work product, namely, the target database.

Because a strategy is associated with a task, its scope may reside within a single phase or span multiple ones. For example, the Test Phase, which is one of the main phases of the overall methodology described in Chapter 5, “Systematically testing a commerce site” on page 103, may require tests be completed during each subphase. If so, then the tests would likely include Functional Verification, System, and User Acceptance Tests, all of which are explained in greater detail in 5.1, “Site testing phases” on page 104. On the other hand, all testing may be combined under a comprehensive test plan, such as System Verification Test and User Acceptance Test.

Each strategy possesses factors that influence its selection by the project team as the strategy for a particular project phase or phases. Factors most likely to influence a project team’s decision making process are as follows:

- Resource skill level
- Business requirements
- Availability of tools or aids to streamline tasks

Before pursuing any one given strategy, you are strongly encouraged to assess and compare the advantages and disadvantages of potential candidates to select the appropriate ones given your project’s requirements.
2.3 Project development methodology: Phase and life cycle

Figure 2-1 graphically represents the six phases of a typical project development methodology. Within this Redpaper, we focus on the following four core phases:

1. Design
2. Implementation
3. Site testing
4. Deployment

Please note that the four core phases are positioned between pre- and post-project phases normally. In Figure 2-1, the pre-project phase is titled Pre-sales, and the post-project phase is defined as Site maintenance/enhancements.

With regard to transition projects, Figure 2-1 denotes the site requiring transition as WebSphere Commerce Suite source version, and the end result of the transition effort is shown as the WebSphere Commerce target version.

Once the e-commerce site has been launched, any new modifications added to the site after the project ends should managed as functionality enhancements. During the lifetime of the e-commerce site, one or more of the methodology phases may be repeated as necessary to facilitate enhancements. Typically, repeated implementations of the methodology will be required to incorporate significant enhancements to the site, such as new versions of WebSphere.
Commerce software or updates, or to accommodate changing business requirements.

2.3.1 Core development phases

In Figure 2-2 on page 19, each of the core development phases is positioned along with its likely subphases. Please note that the subphases tend to be iterative and, in some instances, may even overlap each other. It is important to understand that proper assessment of a project will help determine which, if any, of the subphases apply. Also note that subphase relevance and scope will vary from project to project.
Pre-sales phase
The customer must consider, assess, and plan their e-commerce site carefully. Understanding site requirements, defining the nature and intricacies of the existing site for transition projects, and documenting the required features and functions of the target WebSphere Commerce site are key customer activities that should be completed during this phase. Please note that special attention...
should be paid to define and document business requirements adequately, as opposed to technical constraints or implementation details.

A common event during this phase is for the customer to request estimates of resources required, time to develop, and other details, such as the machine specification required to host the final site. The intention is to use the provided numbers for budgeting or sizing purposes. It may be possible to provide extremely rough estimates, based on similar previous projects. It would be impractical to provide more accurate figures until later in the project.

This phase is described further in Appendix B in “Pre-sales phase” on page 138.

**Design phase**
The design phase marks the beginning of the WebSphere Commerce project and consists of the following three subphases:

1. Solution outline
2. Macro design
3. Micro design

Taken together, these subphases represent incremental and iterative design activities that enable the project team to define the target system details. High-level details of this phase are described in 3.2, “The design phase” on page 33.

**Implementation phase**
During the implementation phase, the project team focuses on coding the design that resulted from design phase activities. This phase also includes continuous and incremental unit testing and the integration of required software components.

**Site testing phase**
The site testing phase covers several formal subphases, all of which enable the project team to test the newly developed or transitioned e-commerce site assets systematically.

**Deployment/launch phase**
During the deployment/launch phase, the project team assists the customer IT team to bring the e-commerce site to a production-ready state and, as required by the customer, to support and help its IT team to enable the site.
Site maintenance and enhancement phase
During this phase, no new major design activities are undertaken, because all
customer IT team activity centers on maintaining or enhancing current site
functionality to satisfy new or changing business requirements. The duration of
site maintenance and enhancement can range from a few months to several
years, and all activity should center on maintaining and enhancing established
site functionality to support new or changing business requirements.

It is possible that one or more significant site improvement projects may be
initiated during the enhancement period, perhaps to reflect a new customer
branding scheme. Architecturally, however, the site implementation will remain
largely unchanged. This scenario is discussed in greater detail in the “Site
maintenance and enhancement phase” on page 140.

The lifetime of an e-commerce site stretches beyond the time period for the
WebSphere Commerce project itself, shown as the Project duration in Figure 2-1
on page 17. The overall site lifetime can be viewed as a sum of project durations,
including transitioned projects and the maintenance and enhancement phases.

Not all site transitions are as complex as moving from WebSphere Commerce
Suite v4.x or earlier versions to WebSphere Commerce v5.x or later versions.
For example, moving from WebSphere Commerce v5.1 to WebSphere
Commerce v5.4 will require only a fraction of the effort that is required for
transitioning WebSphere Commerce Suite v4.x to WebSphere Commerce v5.x.
The key reason for the difference between the work efforts is the technology
change in the programming models: Version 4.x is built on Net.Data and C++,
while Version 5.x uses JSP and Java/J2EE exclusively.

2.4 Using the methodology
Using the methodology effectively involves certain considerations that will be
elaborated below. Your understanding of the implications and requirements for
each of the several considerations outlined here will improve your ability to
employ the methodology during any given project.

Developing a firm understanding of the considerations will enable you to adapt
effectively the methodology presented here to meet specific requirements of the
development work for a given project. For example, the methodology presented
in this Redpaper provides guidelines for deciding between the use of all the
design subphases (solution outline, macro design, and micro design), or
combining one or more subphases within a given e-commerce project.
Understanding and carefully evaluating these guidelines, and applying the
results to an e-commerce project, will increase the chance of the project
manager making the best decision for the project, and so increase the probability of success.

2.4.1 Customizing and adopting the methodology

It is possible to customize this methodology such that it could be applied to building all types of e-commerce sites using IBM WebSphere Commerce software. This includes B2B and B2C sites, which may be small, medium, or large in size.

For each phase, this Redpaper describes core strategies that aid the project team to complete key tasks. Approaches for customizing the methodology are also provided to enable project teams to tailor it to specific project requirements. All the strategies identified in this Redpaper are grounded in best practices that ISSW derived from actual WebSphere Commerce projects and engagements.

2.4.2 New and transition sites

There are two scenarios for creating a commerce site. The first is where a new site is developed, normally using WebSphere Commerce v5.4 or possibly WebSphere Commerce Suite 5.1. The second scenario focuses on transitioning older sites, typically based on WebSphere Commerce Suite v4.1, to newer versions of the product, normally WebSphere Commerce v5.4.

The emphasis here will be on transition projects that are migrating e-commerce sites from WebSphere Commerce Suite v4.x or earlier versions to the Java/J2EE releases of WebSphere Commerce v5.x or later versions. With regard to the terms migration and transition, each can be used interchangeably to describe the task of updating WebSphere Commerce sites or assets.

2.4.3 Project roles and skills requirements

Any given project will have specific role and skill requirements. In terms of this Redpaper, most of the activities will be described from the perspective of either a project manager or a WebSphere Commerce architect. The purpose of this paper is to enable project managers, WebSphere Commerce Suite architects, designers, and developers to define, plan, and execute WebSphere Commerce projects efficiently by providing appropriate guidance, recommendations, and examples. Common roles and skills needed for typical WebSphere Commerce projects are discussed in detail in Appendix A, “Project team creation and common roles” on page 127.
2.4.4 Structuring information

Each section detailing a phase or subphase from the development methodology begins with a table that summarizes phase requirements, inputs, and outputs. This table is structured so that the reader may refer to it at any given time to review the phase’s core components.

Because each phase and subphase is likely to have a corresponding list of tasks requiring completion, each task may have one or more execution strategy. Some tasks and strategies are discussed in detail, while others only receive cursory mention. Strategy tables are provided to summarize the elements and rationale for each strategy so as to enable the reader to select an appropriate one to complete a task. Please note that the development methodology may accommodate many strategies, so the reader is encouraged to develop strategy tables, using the templates provided, when building project plans. Also, the reader is encouraged to assess and review all strategies first, prior to selecting one to proceed with.

The following recommendations will aid your developing appropriate project plans:

- It is very important that the main phases of the project development methodology should never be ignored or skipped when building a project plan.
- In principle, the subphases of this methodology can be combined, overlapped, deleted, replaced with other subphases, or supplemented by new subphases to suit project needs.
- The list of tasks and strategies that are specified in this Redpaper represent a subset of the possible tasks and strategies that could be applicable during a project. We would recommend adding new and relevant tasks and strategies when creating a project plan.
- Prior to customizing any subphases, fully assess the associated advantages and disadvantages of the customizations. All the tasks and strategies listed in this Redpaper are based upon best practices gained from direct experience.

2.4.5 Case studies

While describing the various phases and tasks of the development methodology, we provide examples to aid your understanding the phase or task. Many of the examples were inspired by case studies of real projects. The examples provide concrete scenarios for you to use and adapt during your planning efforts.
2.5 Transition Tool Suite: Introduction

IBM Transition Tool Suite (TTS) consists of integrated helper tools that were designed to reduce the work effort associated with transitioning WebSphere Commerce assets and data form the Net.Data and C++ programming model to the Java and J2EE framework. The source sites will have been built with either Net.Commerce™ or WebSphere Commerce Suite v 4.x. The tools can be used to transition the source assets and data to WebSphere Commerce v 5.x, and they improve the reuse of existing SQL queries and customer data.

Currently, TTS is available in two versions:

- TTS Version 1.0 supports transitioning WebSphere Commerce Suite v4.x sites to WebSphere Commerce Suite v5.1
- TTS Version 2.0 supports transitioning WebSphere Commerce Suite v4.x sites to WebSphere Commerce v5.4

Select TTS tools may be used throughout the development methodology phases and subphases. Because both versions of TTS come with the same tools, the process for using TTS with the methodology is the same for each version. Individual tool functionality is described in detail in the Transition Tool Suite User Guide that ships with the code.

We provide an overview of the tools here to aid your understanding of how TTS may be used with the development methodology. It is critical for you to understand the relationship between TTS and the methodology if you are to maximize the efficiency of your project team. For example, correct utilization of the tools will help streamline the migration of WebSphere Commerce Suite sites. TTS tools, with the exception of the Lightweight Object Query System, or LOQS (see details in Table 2-1 on page 25), supports WebSphere Commerce migrations only. What makes LOQS unique is that it is equally applicable and beneficial for new sites as well as migrated sites. It is the only TTS component that ships with a runtime environment that can be integrated with WebSphere Commerce to run data objects created through the tool or coded by hand. Figure 2-3 on page 25 shows an overview of the relationship between TTS, LOQS, and WebSphere Commerce.
Table 2-1 provides a summary of the core TTS tools, their functions, and supported mode of operation.

Table 2-1  TTS main functions and tool offerings

<table>
<thead>
<tr>
<th>Tool offered</th>
<th>Modes supported</th>
<th>Major transition functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS Schema Crawler/Mapper</td>
<td>Batch</td>
<td>Preprocessor Tools</td>
</tr>
<tr>
<td>Bean Recommendation Engine</td>
<td>Interactive and batch</td>
<td>Bean recommendation</td>
</tr>
<tr>
<td>SQL Collector</td>
<td>Interactive</td>
<td>Bean recommendation</td>
</tr>
<tr>
<td>SQL Extractor</td>
<td>Batch</td>
<td>Bean recommendation</td>
</tr>
<tr>
<td>Document Helper</td>
<td>Interactive</td>
<td>Bean recommendation</td>
</tr>
<tr>
<td>SQL Translator</td>
<td>Interactive</td>
<td>SQL translation</td>
</tr>
<tr>
<td>SQL Validator</td>
<td>Interactive and batch</td>
<td>SQL validation</td>
</tr>
</tbody>
</table>
2.5.1 Preprocessor Tools

The Preprocessor Tools are configured to recognize standard, or out-of-the-box, WebSphere Commerce assets, logic, and data. If the source site requiring transition contains customized assets, logic, or data, the project team must update the tools to recognize those custom components. The Preprocessor Tools have been expressly designed to allow easy integration of the custom details so that the remaining tools can transition the custom components. Preprocessor programming details can be found in the Transition Tool Suite User Guide that ships with the code.

2.5.2 Bean Recommendation Tool

The Bean Recommendation Tool enables users to identify standard WebSphere Commerce beans, such as data beans, access beans, or EJBs, that may be used in place of SQL statements. Identifying and collecting v4.x SQL statements is an important precursor in the process of reusing existing assets. Aligned with the Bean Recommendation Tool is a data collection tool that supports this task.

2.5.3 SQL translation, validation, and bean generation

Practical use of TTS tools for ISSW transition projects suggests that potential savings in time and other project costs are made in many project activities, such as design (using the Bean Recommendation Tool), coding (using LOQS), and testing (using the TTS SQL tools).

The SQL Translator, SQL Validator, and Lightweight Object Query System (LOQS) Tools enable users to:

1. Translate WebSphere Commerce Suite v4.x SQL statements to v5.x SQL statements.
2. Validate the translated SQL statements.
3. Package the SQL statements into Java objects/beans using LOQS.

<table>
<thead>
<tr>
<th>Tool offered</th>
<th>Modes supported</th>
<th>Major transition functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOQS Code Generator</td>
<td>Interactive and batch</td>
<td>Bean generation</td>
</tr>
<tr>
<td>LOQS Runtime</td>
<td>N/A</td>
<td>Bean generation</td>
</tr>
<tr>
<td>Data Migration Tool</td>
<td>Batch</td>
<td>Data migration</td>
</tr>
</tbody>
</table>
2.5.4 Data Migration Tool

The Data Migration Tool enables efficient migration of source WebSphere Commerce Suite site data assets.

The tool has been used by ISSW in many transition projects. The projects have observed potentially significant time savings when performing the data migration activity. An additional benefit is that project risks are reduced as a consequence of the early availability of a sample or complete target database. This database makes the SQL translation and validation activity easier and also aids the coding activity by providing early access to a stable and relevant set of site data (for example, catalog content, user details, and order samples).

The Data Migration Tool migrates or formats data based on a mapping file. The user creates a mapping file spreadsheet containing source table column to target table column mapping information, as well as the logic needed to migrate the data. This spreadsheet is saved as a CSV file. The CSV file may then be used as an input to an XSL-based tool that reformats the mapping information and represents it in XML format.

A Java application takes the XML file and creates Export and Load script files and corresponding SQL queries in the native format of the target and source databases. A script is also generated that enables the user to execute the migration from one entry point, rather than executing each Export or Load file individually. Using the native database tools provided by DB2 or Oracle allows the migration process to benefit from the speed and power of the respective database.

Using the Data Migration Tool enables users to migrate data more easily and requires only that they maintain a simple spreadsheet. The tool will also generate primary keys where needed. Because the tool produces SQL files specific only to the database, the output can run on any platform the database supports.

Often, migration requires an upgrade from one version of an application to another, more recent version. This typically entails installation of intermediate levels of software, which greatly increases the effort required to migrate. The Data Migration Tool is designed to aid the user by allowing them to:

- Migrate application data from one version to another version of the same application
- Simplify the transition of data from one application to a different application

Specific advantages of the Data Migration Tool include:

- The user only needs to create and maintain a mapping spreadsheet and is not required to learn underlying technologies, such as XML and SQL.
The Export and Load process is very fast because it uses the native facilities of the underlying database server.

The tool can generate primary keys where required.

The tool can be used across platforms as because produces SQL files specific to the underlying database, rather than the platform.

2.6 Transition Tool Suite usage

The key design philosophy integrating the tools is the concept that each tool may be used either independently of one another or as an integrated suite to facilitate the transition of WebSphere Commerce Suite sites. TTS usage is discussed further to elaborate and clarify the rationale of this philosophy. Further information about acquiring TTS is provided in "Other resources" on page 147.

2.6.1 Individual tool usage

Each tool has a well-defined input and output, and most of the tools support both interactive and batch operations. Together, these characteristics enable users to define when and how TTS tools can be used to aid the transition of WebSphere Commerce assets.

For example, the SQL Validator may be used interactively in one phase, or in batch mode during another phase. In both modes, it will test and validate source and target SQL statements. This flexibility in usage enables the user to select when the tool should be used during the transition, for example, during SQL translation or later during testing, and so improves productivity by time optimization.

2.6.2 Sequential tool usage

Some transition tasks require the use of two or more tools sequentially to accomplish specific objectives. For example, the Data Migration Tool and the SQL Validator may be used in sequence to migrate data and test the outcome. The SQL Translator, Validator, and Lightweight Object Query System (LOQS) Code Generator may also be used in sequence to transition Net.Data macros to JSPs, specifically by converting SQL to corresponding QueryCommands. Using the tools together increases the opportunity to migrate key assets efficiently.
2.6.3 Transition Tool Suite and the development methodology

TTS tools, when combined with development methodology, produce economies of scale. Systematic use of the tools streamlines the transition work effort by first minimizing redundant or repetitive tasks, and then simplifying the remaining tasks.

We described one method for integrating TTS with the project development methodology in 2.3, “Project development methodology: Phase and life cycle” on page 17. In practice, however, we recommend you devote some time during project planning (that is, the design phase) to determine and define how each tool may and should be used over the course of your project. Depending on the characteristics of a given project, it may be appropriate to use some, all, or even none of TTS.

TTS is not a prerequisite for the project development methodology specified in this Redpaper. This methodology, however, does provide specific links or “hooks” to one or more of the TTS tools to help speed up the transitioning of WebSphere Commerce Suite sites. These hooks show how the TTS tools can integrate into the transition methodology. Equally, project teams may use any tools, whether from TTS or another source, to support the transition process as outlined in this paper. Similarly, any TTS tool may be used with other transition methodologies.

2.7 Summary

In this chapter, we defined a project development methodology that can be used to develop new e-commerce sites or to transition existing sites built with earlier versions of WebSphere Commerce software.

We also introduced the Transition Tool Suite, a key resource for improving design, development, or transition activities. TTS provides a number of utilities and tools that focus on major parts of site design and development.

Having reviewed this chapter, you should be able to:
- Describe the core phases of the development methodology
- Define key development terms and concepts
- Discuss relevant deployment conditions

In the next chapter, we discuss and frame the design phase of the development methodology, which is the first of the methodology’s major project phases.
Designing the commerce site

The design phase is the first major activity on a project following the pre-sales phase, which is described further in Appendix B, “Pre-sales, site maintenance, and enhancement phases” on page 137. The business requirements or fit-gap analysis conducted during the pre-sales phase serve as input work products to the design phase. In addition to technical activities performed by the project team, the project manager, working with technical lead staff and the customer project manager, starts to build a detailed project plan.

This chapter identifies the design activities performed by the project team and the project planning activities performed by the project manager. Each is a core design activity, and in some cases, the resulting output work products will be used throughout the remainder of the project. Also, some of the output work products will need to be completed during this phase prior to the project team initiating the implementation phase. This phase is discussed further in Chapter 4, “Preparing and implementing the commerce site” on page 73.

Please note that the process of developing a commerce site, whether it is a new site or a transition of an existing site, is one that requires the same precision and rigor as any other significant systems engineering project. This chapter frames the development process for you.
3.1 Purpose of the design phase

The objective of the design phase is to define and document the site design for the target site. The key deliverable of this phase is a detailed project plan, which, ideally, the project manager should create before exiting the subphases of the design phase, that is, on completion of the solution outline, macro design, and micro design subphases. In practice, it may be possible for several project activities to be planned without waiting for the project design team to finalize the project scope, the site design, and the technical work activities, but all early project activities should be considered as exceptions.

**Note:** Irrespective of whether a project involves the implementation of a new site, or the transition of any existing one, if the work is similar to a previous project, or appropriate resources or skills are available, the project manager may be able to reduce or eliminate some aspects of the detailed design and proceed with planning later activities.

The risk is that the detailed plan may uncover or introduce details that require the earlier plans to be discarded or at least significantly modified. We would, therefore, recommend extreme caution in bypassing the design phase.

Once the site design has been defined, and all technical work items are identified, the project manager should add each to the project plan.

**Note:** While not an explicit activity, we recommend adding lead time to each phase within the project plan. This recommendation is based on good project management practice and practical experience of commerce implementations. Some tasks require substantial planning to ensure all prerequisites are met. Adding lead time to all tasks is good practice, because it provides you the opportunity to compensate for unforeseen circumstances when executing on projects.

Defining the data model early in a project is important because it will be used to lay the foundation for the overall site design and subsequent data creation or migration activities. In addition, the process for developing the data model may reveal aspects of the project that were not noticed during prior activities, but that may require adjusts to the project plan.
3.2 The design phase

The design phase is the formal beginning of a WebSphere Commerce development project. Depending on whether the project is to implement a completely new site, or to migrate and existing site, there will be a slight difference in the entry point to this phase, as shown in Figure 3-1.

![Figure 3-1 Collecting information according to the nature of the project](image)

An overview of the design phase itself is shown in Figure 3-2 on page 34.

A subset of the project team will have been assigned to focus on the design activity. The team would typically have people with specialist skills in WebSphere Commerce architecture and design. These roles are described in more detail in Appendix A, “Project team creation and common roles” on page 127.
3.2.1 Creating the site design

Creating a design for a commerce site is a key activity of the development methodology. Site design not only influences and controls all development tasks, it also affects the day-to-day use and operation of the site after launch.

Given the importance of site design, it is essential the project manager structure design activities in such a way as to produce a proper design. After forming the team as previously discussed, the project manager should next determine how to proceed with site design.

Through many engagements, IBM Software Services has found two design strategies to be most beneficial. The first is the iterative and incremental design strategy, where tasks are performed while allowing for the possibility of refining the result by repeating the activity (iteration), and the tasks themselves proceed in a step-by-step fashion (incrementally). The second design strategy is the subphase combination design strategy. The details for each are discussed in the following.
The iterative and incremental design strategy

This strategy divides the main design phase into three subphases. The rationale for this strategy is that through close interaction with the customer, the team’s understanding of site requirements and constraints will improve over time. The implication is that as the information improves, so will the site's design. This strategy accommodates continually changing information through incremental or iterative design activity.

The three subphases of this strategy are as follows:
1. Solution outline (see 3.2.2, “Solution outline subphase” on page 37)
2. Macro design
3. Micro design

Each subphase builds on the output produced during the previous phase.

This strategy promotes interaction with the customer during each subphase and encourages formal sign-off as tasks are completed.

An overview of the iterative and incremental design strategy is provided in Figure 3-3.

---

**Iterative and Incremental Design Strategy**

*Strategy: Three Phase Iterative and Incremental Design*

**Input Work Products**
- Skilled team, which may consist of one or more members
- All available pre-sales output work products, including customer requirements

**Task:**
Create project design.

**Objective:**
Establish scope and number of subphases for the design activity.

**Output Work Products**
- Project design documentation

**Key Influencers:**
- Experienced design team
- Availability of customer team to review and approve design as it moves between subphases.
- Complexity of business logic required by target site

**Project Type:** Either New or Transition  **Project Phases:** Design Phase

*Figure 3-3 Iterative and incremental design strategy*
Experience suggests that the iterative and incremental design strategy is best suited to projects with complex business logic.

**Subphase combination design strategy**

As its name implies, the subphase combination strategy combines, or alternatively, compresses, the design subphases of solution outline, macro design, and micro design into refined design activities. Please note that combining or compression of subphases does not in any way absolve the project team from collecting site information, nor does it reduce the amount of information required for proper site design.

The combined approach requires the project team to determine which subphases should and can be combined in such a manner as to support the design activities.

The level and details of customer requirements obtained as input to the design phase will aid the project team to determine which combination of subphases may be appropriate. For example, given well-defined requirements (that is, requirements that are unambiguous, precise, sufficient, consistent and non-contradictory), it is possible to combine the solution outline subphase with the macro design subphase. Conversely, if the requirements are open or not well articulated, work defining the requirements should be performed during the solution outline subphase. Only when the requirements have been completed can the project team pursue a combined macro design and micro design subphase.

**Note:** The subphase combination design strategy is most suitable for simple sites being developed by experienced project teams. If used, the project team should maintain at least two of the design subphases, which means the possible subphase combinations are either solution outline and macro, or macro and micro. No matter what the final determination ends up being, the project team should engage the customer throughout the design process to ensure the resulting design incorporates all requirements and is satisfactory.

The project team must determine how the design subphases can be combined to fully support the design requirements. The customer requirements obtained during the initial project phase will aid the project team in determining how to combine the subphases. For example, given well-defined requirements, a project team should combine solution outline activities with macro design ones. Conversely, should the requirements be open, the project team should combine macro design and micro design activities and retain the solution outline in order to refine the design requirements with the customer.
An overview of the combined subphase design strategy is provided in Figure 3-4 on page 37.

Irrespective of the design strategy selected, site requirements should be documented fully prior to the project team commencing with the implementation phase, as discussed in Chapter 4, “Preparing and implementing the commerce site” on page 73.

### 3.2.2 Solution outline subphase

Prior to starting the solution outline subphase, the project manager must ensure both the customer’s business requirements and output from the fit-gap analysis are available and readied for use by the project team, as described in “Pre-sales phase” on page 138. This information will enable the project team to begin its design work. Without it, work on the solution outline is not likely to produce much of value. Should key details be missing, the project manager is encouraged to consult with the customer to obtain the missing details prior to completing the remaining design activities.
In some projects, the need for a solution outline subphase may be questioned; it may appear desirable to progress directly to the macro design subphase. The reason why the solution outline subphase is so important is that most projects will require information to support budgeting, capacity, and skill planning as soon as possible. More accurate information to help provide these details would not normally be available until after the completion of the macro design subphase, which may take some time. Addressing these issues as part of the solution outline subphase allows the project team to avoid a delay, which may potentially be measured in months, and so progress key directional decisions earlier, as well as allowing parallel activities, such as ordering of hardware.

The major output work product produced during this subphase is the solution outline document. It captures the initial high-level design for the target site. You may find it helpful to have a standard template for capturing and communicating the design details.

The activities and work products used in the solution outline phase are detailed in Figure 3-5. It is important to note that during each subphase, besides the core tasks listed, additional project activities would typically continue in parallel. The main tasks are described in detail in the following.

During the solution outline subphase, the project team will lead substantial interactive discussions with customer to help define the design requirements. It will normally take several iterations of technical investigation and discussion to fill in details of the requirements, define external system and component dependencies, and finally, deliver the initial high-level design that incorporates
each requirement. Iterations are often necessary to clarify points of misunderstanding, to supplement detail where more information is required, and to address omissions or agree to late additions to the requirements.

Gathering the functional requirements

The first step in the solution outline subphase is to gather the functional requirements, which are driven by the customer’s business requirements for the site.

**Note:** Most projects will have functional and non-functional requirements. For commerce site development, most non-functional requirements, such as capacity planning and deployment considerations, will be reasonably consistent with other projects. Therefore, non-functional requirements will not be specifically addressed further in this paper.

Functional requirements for a WebSphere Commerce project normally consist of the following:

- Use cases

**Note:** Use cases are often helpful when establishing the requirements for medium and large customer commerce sites, but may not always be appropriate for smaller customers. This is because WebSphere Commerce provides a rich out-of-the-box set of functionality that should address the requirements of sites typical of many standard commerce models. The danger is that the customer may view the use cases as a set of wish lists to transform what should be a “small store” site into a multinational “mega” site.

- Site flow descriptions
- Business rules
- Integration with other systems or products

Some customers will already have one or more of these defined. Others will depend on the project team to help them come up with the functional requirements.
For most WebSphere Commerce transition projects, the project team is typically given the customer's current site as the starting point for building the functional requirements. The project team should start by reading the customer's existing requirements documentation, reviewing the current site, and reverse engineering the code as appropriate.

**Note:** If several WebSphere Commerce designers are working on the solution outline subphase, the project manager may wish to subdivide the design team to enable its parts to focus on design activities relevant to specific WebSphere Commerce subsystems. No matter on what part of the design a designer is working, a subsystem or whole, they will be responsible for the following activities:

- Creating a comprehensive and complete set of use cases
- Creating site flow information and diagrams
- Determining the required business logic and processes used for site operations
- Collecting any existing SQL statements from the code

**Note:** Collecting SQL statements from source site code enables the project team to construct the data model. In addition, it prepares the team for data migration and SQL translation.

- Helping validate SQL statements for migration projects, for example, assessing their number and complexity

**Note:** This information is helpful for initial estimates and subsequent refining of the sizing of the effort involved.
It is often extremely helpful to have some form of internal project bulletin board, where each team member can post questions and responses about the project. During the solution outline subphase, these questions will help resolve ambiguity about functional aspects of the requirements, unclear business rules, and inaccessible paths through the site flow.

On a regular basis, the project manager should review outstanding team questions and discuss them with the customer. Depending on the number of questions, weekly reviews may be sufficient. Sending the questions to the customer using e-mail means that they are written down in a form that allows the customer to discuss them internally, and as necessary, for the customer's project manager to produce an answer for the project team.

**Note:** Delays in answering core design questions are surprisingly common in projects. The questions normally arise from ambiguities or inconsistencies with the customer's business requirements, which may indicate similar uncertainty or difference of vision in the customer. Resolving these problems and answering the questions may take time until an agreement is reached, and it is possible that the project development team may be invited back to the customer to provide additional help.

Through established and periodic communication with the customer, the project team through the project manager is likely to develop contacts with core members of the customer IT team. This information is potentially a windfall because it enables team members to contact their corresponding counterparts on the customer's team.

**Note:** While communications of this kind may be helpful, quick, and easy, project team members should fully document the questions asked, the resulting discussions, the people involved, and any conclusions reached. There is a real danger that informal or poorly documented communication may result in problems, such as project scope creep.

Once the fundamental requirements have been gathered, the project manager should arrange a workshop with the customer for the project teams to confirm the requirements.

The solution outline subphase deliverables include:

- The functional requirements
- The solution outline document
The functional requirements and the solution outline document are extensions of the customer's business requirements. The functional requirements transform the business requirements into more tangible site specifications and implementation details. The solution outline document, on the other hand, enables the project team to define how each business requirement can be mapped to either out-of-the-box WebSphere Commerce functions or specific customization. The document should also address how the proposed site will integrate with existing legacy systems, third-party applications, or back-end systems.

The solution outline workshop
During a solution outline workshop, the design team will validate all functional requirements and review targeted out-of-the-box WebSphere Commerce functions with the customer team. The objective of the workshop is to ensure the proposed site uses all available features provided by WebSphere Commerce. The customer and design teams must have a common understanding and shared vision for the planned site.

An overview of the workshop is shown in Figure 3-6.

Figure 3-6 Overview of solution outline workshop
The design team starts the workshop by inviting the customer team to talk about their current site structure and any ideas for future expansion plans. The design team then uses the information to identify how WebSphere Commerce can be used to fulfill the customer's stated needs. The design team will make every attempt to stress as much out-of-the-box WebSphere Commerce function as practical over product customization, but where necessary, it will explain how the product should be customized to suit unique requirements. The workshop discussion is likely to reveal potential third-party application or legacy system dependencies.

Product functionality is assessed through WebSphere Commerce subsystems. The design team should conduct a product demonstration stressing the functionality provided by each relevant subsystem. This review will also enable the customer to fully understand the capabilities offered by WebSphere Commerce.

Following the review, the design team should walk through the proposed site flow, stressing customer requirements and related business rules. For transition projects, it is very helpful to discuss the current data model that may be extracted from SQL statements and the approaches that may be used for data migration.

Another important topic that is often overlooked is the content management process, which includes both catalog data and static content data. This discussion should cover both initial data load and subsequent regular planned updates.

At the conclusion of the workshop, the design team, with review and approval by the customer, should have the business and functional requirements fairly well defined, along with the project scope and any key dependencies. These decisions are then documented as the deliverable from the solution outline subphase together with a preliminary design to meet the requirements.

Note: Even customers who have used WebSphere Commerce may not be familiar with the more recent or newly released functionality in the product. In many cases, the customer may view the workshop as a way to train themselves on new WebSphere Commerce features and functions and to discuss options for the future of their site.
The solution outline document
The solution outline document will typically have three sections:

1. System overview and architecture
   The system overview and architecture captures the site component overview and component responsibilities that together provide a strawman of the proposed site construction.

2. Architectural decisions, grouped by subsystem
   The architecture decisions describe the business requirements and assumptions, document the issues encountered and their resolution, and record the high-level design decisions made for each requirement, such as which out-of-the-box function will be used and what customization will be required. All issues must be discussed with the customer during a subsequent solution outline review period.

3. Outline of assets to be built, grouped by subsystem
   The outline will normally be used to size projects. For example, it can be used to estimate the required number of JSPs, custom commands, or EJBs. In addition, development of other utilities, such as data loading and integration, should be included in the outline. This information may be captured in a spreadsheet, along with estimates for difficulty rating and sizing. For a transition project from a WebSphere Commerce 4.x site, the count of assets can be based on the original Net.Data macros and commands and overridable functions used along with sizing information about the customized tables and data. In addition, transition projects will need an extra task, data migration. This is discussed in “Using the TTS Data Migration Tool” on page 59.

Analyzing current site assets
Where an existing site is being transitioned, it is essential to get a detailed understanding of the current site assets. This involves analyzing the current site assets, such as C++ commands, Net.Data macros, and custom database tables or extensions. This analysis must encompass the explicit customer business and technical requirements as part of creating a high-level design for the new site.

We recommend a strategy that uses the Transition Tool Suite (TTS) to accomplish the task of transitioning assets. Practical information about applying the TTS resources is provided in the Transition Tool Suite User Guide, which is provided with TTS.

Information included within a fit-gap analysis document can help identify the areas of site functionality that may be implemented using new out-of-the-box (OOB) functions; the benefit of implementing OOB functions is that it helps reduce the quantity and complexity of custom code in the site. It may be
necessary to reuse the existing site design if no out-of-the-box function meets the customer business needs for one or more components of the site.

**Note:** Even if a decision is taken to reuse existing site design for one or more components of the site, we would still recommend using the WebSphere Commerce framework in the new implementation to make sure that the custom design integrates well with the product architecture and benefits from best-practice guidelines.

Analysis of current site assets can be done using the TTS tools. Before conducting the SQL analysis, it is *essential* to have a current and complete representation of the site data model. For example, a spreadsheet may be used to capture relevant source database table and column information, as well as target schema information. When complete, the spreadsheet can be fed to the TTS Bean Recommendation Tool to identify applicable out-of-the-box beans for use in the transitioned site. The spreadsheet will also prove valuable during later data migration. Table 3-1 lists the TTS tools that could be used during the solution outline subphase.

**Table 3-1 Using TTS during the solution outline subphase**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Detail</th>
</tr>
</thead>
</table>
| **Applicable TTS tools** | SQL Extractor Tool.  
                           SQL Collector Tool.  
                           Bean Recommendation Tool.  
                           Document Helper Tool. |
| **TTS preconditions**  | At this point, TTS is installed and used to provide base WebSphere Commerce information (that is out-of-the-box product information that does not include customer site customization).  
                           A spreadsheet containing source WebSphere Commerce schema information should be completed prior to this subphase. |
| **Transition assets generated** | XML file containing current site SQL statements.  
                          XML file containing recommended beans for the migrated site. |
| **Exceptions**         | None.                                                           |
Best Practices and Tools for Creating IBM WebSphere Commerce Sites

Design-based sizing
Sizing for the project is usually possible at, or soon after, the completion of the solution outline subphase. An earlier, less accurate sizing may already have been performed as part of the pre-sales phase, as described in “Pre-sales phase sizing” on page 139.

In brief, for each functional requirement, an asset count is produced. This will include the expected number of JSPs, the number of new or modified commands, and the number of EJBs and data beans. Each of these assets is given a complexity level. Next, the effort to develop an asset is established, based on information from prior completed projects, and adjusted to reflect team experience and other pertinent factors. The result is an estimate of the effort required for building and testing each of the assets that can be aggregated with the other estimates to produce a reasonably accurate project sizing.

### 3.2.3 Macro design subphase

The main focus of activities during this subphase centers on creating a WebSphere Commerce design, the details of which are recorded in a macro design document.

**Note:** The design team that completed the solution outline subphase should also follow through with the macro design for the target site. You may need to consider adding or removing members of your design team to balance resource availability against the time required to deliver the site.

An overview of the work products and activities for this design subphase is provided in Figure 3-7.
A summary of the TTS tools that are applicable in the macro design subphase is provided in Table 3-2 on page 47.

**Table 3-2 Using TTS during the macro design subphase**

<table>
<thead>
<tr>
<th>Concept</th>
<th>Detail</th>
</tr>
</thead>
</table>
| Applicable TTS tools           | Bean Recommendation Tool.  
|                                | Document Helper Tool.  
|                                | Data Migration Tool.  
|                                | Preprocessor Tools.                                                     |
| TTS preconditions              | Base TTS install. In this phase, the TTS is enhanced with a custom schema mapping spreadsheet |
| Transition assets generated    | XML file containing enhanced recommendation for out-of-the-box beans for the migrated site.  
|                                | Schema mapping XML file generated by the Preprocessor Tools.  
|                                | Data mapping XML file generated by the Data Migration Tool.  
|                                | Target sample database using TTS.                                       |
| Exceptions                     | None.                                                                  |
Methods of approaching design activities
There are several perspectives that may be taken when considering WebSphere Commerce site design:

- The overall system level, which involves WebSphere Commerce interacting with external back-end or legacy systems.
- The subsystem level, which includes the main WebSphere Commerce subsystems, such as Member, Order, and Catalog. This also includes the supporting infrastructure, such as the database and messaging systems.
- The use case level, which includes functional sequences that together form the overall site flow and behavior. Use cases will normally address visual aspects, such as the flows from page to page within the site, and non-visual resources, such as the expected operation and behavior of the order subsystem as it communicates with the back-end inventory system.
- The component level, which includes JSPs, data beans, access beans, EJBs, and commands.

The design team should consider each of these levels during the design phase for the main site. Considering use cases as a means of site design is especially appropriate during the macro design subphase. In a use case strategy for site design, the design team creates detailed use cases. This approach is also consistent with applying an object-oriented method to site design, such as UML, because it helps identify the actors and actions performed within the system.

Note: We have found that driving an overall design strategy through use cases yields significant advantages, especially during the implementation and test phases, which are discussed in Chapter 4, “Preparing and implementing the commerce site” on page 73, and also Chapter 5, “Systematically testing a commerce site” on page 103.

An overview of the use-case-based design strategy is provided in Figure 3-8.
By centering the design around use cases, we mean that all the design decisions made during the macro design subphase will categorize site components, such as commands, EJBs, and JSPs, based upon use cases for the target site.

One way of applying the use case design strategy is to create a macro design document template that the design team can use to create the WebSphere Commerce site design. This template should structure information such that appropriate levels of design decisions, such as “What JSPs, commands, and EJBs are needed to implement the user registration use case?”, are based on use cases for the target site.

A major output work product from the macro design is the interaction diagram for each use case, described at the command and JSP levels. These diagrams help clarify the site objects and components. For example, clear identifiers are assigned to the objects, and the diagrams show how each object is called or accessed when performing a given WebSphere Commerce task.
The data model

Another major output work product from the macro design is the definition of the data model for the site. Creation of a valid data model for a new commerce site requires a thorough understanding of the customer data and its structure, as well as detailed knowledge of the out-of-the-box WebSphere Commerce data model.

Note: The site data model is a part of the development work that sometimes does not have the same “visibility” as other more visual aspects, such as the Web pages or site flow. The low visibility associated with development of the site data model may result in an underestimation of effort required for data model development. The effort to create the data model may be increased both by the complexity of the site data and also by the large quantity of data to be processed. It is, therefore, very important to start working with real customer data as soon as possible. Even subsets of customer data may be used to test the validity of the development database.

The first step in defining the data model is for the design team to identify the data entities from the business requirements and then understand how the data is used and how it flows through the site. For commerce sites, it is normally possible to categorize the data into one of three groups:

- Order data
- User or member data
- Catalog data

In a typical project, order data is created from the data held in a shopping cart. The order becomes a pending order by adding billing address and shipping address details. Finally, the order becomes a submitted order with its status updated by a back-end system.

In another example, user data is first created when the user accesses the site as a guest shopper. They eventually become a registered shopper with a login identifier and password. Typically, a registered shopper is assigned an internal customer number by a back-end system.

By following the data flow in the above manner, and then analyzing the data used at each stage of the flow, the design team should be capable of ensuring that the data model contains all required data entities.

In contrast to order or user data, catalog data is almost static. It changes only occasionally, and the changes that do occur will normally take place at scheduled times. The focus of catalog data is on the purpose of the catalog, how the catalog is structured into a sensible hierarchy, what the product information consists of,
Chapter 3. Designing the commerce site

and how content, such as images and support documents, are related to the main catalog data.

The second step is to use the functional requirements to come up with the data structure necessary to support the business logic. This data is distinct from the catalog data, because it is not part of the data to be presented to shopper; instead, it is related to the business rules required for supporting the functional requirements.

An example of data-supporting business rules would be to apply constraints to catalog navigation such that a customer would have a restricted view of part of the catalog and products, depending on the product brand information they are authorized to see. Similarly, shipping rules may be affected, where certain products cannot be shipped across international boundaries, or have a limited shelf life, excluding a slower shipping method.

In contrast to new site projects, the data gathering task is much easier in transition projects, because it usually requires analysis of the existing data model and understanding the usage of each data field.

After gathering all the data required, the design team should compare the required site application data model with the out-of-the-box data model. In most cases, the out-of-the-box data model will be sufficient.

**Note:** It is especially important to ensure that data stored within the standard out-of-the-box tables will be used exactly as specified in the data model. Misunderstanding the standard data model and the consequent effect of standard commands on data stored in the model will have deep-seated and hard-to-reconcile consequences for the project.

For example, in some multinational organizations, a single product may be offered under several different brand names. It may be tempting to treat the brand as simply an attribute of the product and store this brand information within the standard out-of-the-box ATTRIBUTE and ATTRVALUE tables as descriptive product attributes. However, this might introduce other problems or complexities, because the brand may also be used to support catalog navigation logic and, therefore, the range of products a customer can view. In this case, the brand is not merely a product attribute, but also has a significant effect on the perceived structure of the catalog. In such cases of incompatibility between the stated purpose of an out-of-the-box data model component and the desired application, it may be better to design new tables that more effectively support the desired logic.
Not all site-specific data requires a new custom table. Almost all WebSphere Commerce tables provide some custom fields, but very little logic related to them except direct support for read and write. If the extra data falls into this simple category, it is sensible to use the custom fields from out-of-the-box tables.

**Note:** During one project, the IBM Software Services team had to store an external Web link associated with particular products. The custom fields within the standard data model were ideal for this purpose, because no functionality was required beyond simple data read and write.

Something that should definitely be avoided is adding new columns to out-of-the-box tables to support extra data. These extensions would mean that the standard EJB for the table would not cover the new fields. It would be much better to use a custom table to store the extra data, with a primary index corresponding to the primary index as the out-of-the-box table. A new EJB would be required, corresponding to the new table, but this would be a project-specific extension and, therefore, would not compromise future migration.

**Note:** After the data model has been defined, you are advised to reassess the solution outline and functional requirements to make sure that all the selected out-of-the-box functions identified are correct for the functional requirements, that the out-of-the-box data model supports the site data, and finally, that all the functional requirements that are not supported out-of-the-box have been included in extension data tables.

**Creating interaction diagrams and class diagrams**
Interaction diagram and class diagram are both use case driven. For each use case, there should be at least three possible paths of execution:

1. The normal or standard path
2. An alternate path
3. An error path

Each interaction diagram should address all three paths for each use case. Creating these diagrams will help illustrate the site flow and so assist with the underlying implementation details.

For WebSphere Commerce projects, the majority of classes used will be commands, EJBs, JSPs, and data beans. In terms of macro design, emphasis will normally be on commands and JSPs. Consideration of EJBs and data beans, which address how data is accessed within commands and JSPs, is normally left to micro design. There are other WebSphere Commerce classes, for example, the communications adaptors, such as the HTTP adapter. It may be necessary to
customize these classes. An example for the HTTP adaptor would be due to
customer specific requirements for single sign on, session management, and
access control. Adapter customization is discussed in more detail in the IBM
WebSphere Commerce Programmer’s Guide, GC09-4951 and related product
documentation.

Having established the site flow for the use cases, the design team should
identify the Web pages used for each path. These pages may be static HTML or
dynamically generated from JSPs.

The next step is to determine which command to use to complete the flow. By
analyzing the use case and referring to the solution outline, it is possible to
determine which out-of-the-box command could be used. This involves deciding
the following:

► Whether the out-of-the-box command will perform exactly the desired task.
► What each path’s input, condition, and result is.
► What tables are to be accessed.

Based on the analysis, the result is a list of commands that includes standard
out-of-the-box commands, as well as the ones that need to be developed as part
of the project.

Transition projects have another factor that should be taken into consideration. If
any of the original site implementation used out-of-the-box commands, it is likely
that corresponding commands can be used for the transitioned code. Further,
even if the original site required a customer command to be created, it does not
follow that the new site will also require a new customer command; the extra
functionality offered within the newer version of WebSphere Commerce may
provide the desired functionality. It is advisable to repeat the analysis to
determine if there is a suitable out-of-the-box command, or what alternative
customization may be needed.

Real projects have identified five scenarios for using out-of-the-box commands:

► The standard out-of-the-box command fully meets the requirements.
► The out-of-the-box command must be extended with extra logic, for example,
to read or write data stored in a new customer table.
► The out-of-the-box command implementation must be replaced.
The functionality requires that one or more out-of-the-box commands are invoked in sequence.

A completely new command must be created.

This information should be documented in class diagram. The *IBM WebSphere Commerce Programmer's Guide*, GC09-4951 contains examples of these development tasks.

There are six kinds of JSPs:

- A JSP used for display of data.
- A JSP used to allow a user to enter data.
- A combination of data display and data entry.
- An error page.
- A hidden JSP used to construct outbound XML messages to be sent through IBM WebSphere MQ (this JSP does not play a part in site flow).
- A hidden JSP used to construct outbound e-mail messages to be sent through an SMTP server (this JSP does not play a part in site flow).

A JSP used for displaying data might include an HTML file to display the static data, in which case, the HTML page should be documented as well. An error page can contain just the error message, or it may be a normal JSP with the error message displayed prior to other content. Again, the error messages should be documented.

After all commands and JSPs have been identified, the design team can build the interaction diagram. The diagram documents the interactions between the user, commands, and JSPs. The interaction diagram should define input and output parameters.
**JSPs templates**
The design team should create a template for all site JSPs to use to promote a consistent look and feel. This presentation style will usually be designed by the customer. The project team has the task of transforming the “look” into templates for later JSP development.

**Note:** Customer expectations for site look-and-feel generally exceed basic branding, and the design team must work to manage these expectations.

**Creating a WebSphere Commerce site database**
A WebSphere Commerce site clearly needs a database to store persistent data. The development project for the site must also have a working implementation of a site database for this purpose. In a transition project, the database structure will already be known. The Transition Tool Suite (TTS) may be used to migrate such a database so that it would be complete at the end of the macro design subphase. For new sites, it would be expected that at least a high-level view of the database would be available after the macro design subphase, sufficient to implement many of the required data structures, but still likely to refined following the micro design subphase.

**Note:** A WebSphere Commerce project will typically require two databases:
- A site database to hold site-specific data and examples, such as catalog content, orders, and shoppers
- A project database to hold details, such as team communications and reference materials

Table 3-3 shows the expected status of database design work within each design subphase, according to the type of project being executed.
Table 3-3  Database status for design subphases, according to project type

<table>
<thead>
<tr>
<th>Project type</th>
<th>Solution outline subphase</th>
<th>Macro design subphase</th>
<th>Micro design subphase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idealized project</td>
<td>High-level database design complete</td>
<td>Database design complete</td>
<td>Database schema complete</td>
</tr>
<tr>
<td>New implementation project</td>
<td>Initiate request for database requirements</td>
<td>Assess and finalize database design requirements</td>
<td>Database design complete</td>
</tr>
<tr>
<td>Transition project</td>
<td></td>
<td>High-level database design complete</td>
<td>Database design complete, and database schema mostly complete</td>
</tr>
</tbody>
</table>

Creating a sample or migrated database by the end of the macro design subphase has several advantages.

During transition projects, the database will serve the design team when they translate and test target site SQL statements to determine if session beans are appropriate. A prerequisite for creating a sample target database is the finalized data model that is produced during the macro design work. The persistent data layer, which consists of out-of-the-box and customer EJBs, access beans, and data beans for the target site, will also require the final data model. Finalizing the data model for the target site enables the design team to create a sample, or migrated target database, which in turn, will aid their completing the remaining design details.

Typically, it is the data creation or migration expert who performs creates the sample or migrated database (see Appendix A, “Project team creation and common roles” on page 127).

Creation of databases is an activity that applies during both the design phase and the implementation phase, and the activity holds true for both new and transition projects. We have already indicated that, ideally, a development sample database would be created during the macro design subphase. This database can then be extended to become the solution database during the implementation phase. This approach minimizes the effort needed to create the solution database.

Each database has a common starting point, which is the out-of-the-box database provided by WebSphere Commerce. This database provides the foundation for the out-of-the-box functions. The foundation consists of the schema, table constraints, bootstrap information, and indexes. If no further customization is required, it should be possible to load customer data directly into
the database early in the project. If further customization is required for the site, the new schema, constraints, and indexes must be applied to the original database prior to loading the data.

**Note:** The sooner the design team obtains customer data, the better for the project. It is surprisingly common in projects that the customer supplies data that is supposed to be correct, consistent, and complete; yet on closer inspection, it is found to have errors and inconsistencies that may require significant cleansing. Such problems are typically encountered where the data is owned by or originates from different departments or divisions.

Figure 3-9 provides an overview of two different strategies for creating the database, depending on whether the database is for a new site project or a transition project. The strategies are different, although the initial steps required to create the solution database are the same. An important factor that affects the success of both strategies is the separation of work into their subsystems and assigning the more difficult ones to the more experienced developer. Separation of duties supports code maintenance.

---

**Creating a Solution Database for WebSphere Commerce**

**Strategy:** Use either MassLoader provided with WebSphere Commerce for new projects or TTS Data Migration Tool for transition projects

**Input Work Products**
- Design phase output
  - Internal design documentation
  - Out-of-the-box database
  - Sample database (if required)
  - Customized data model, schema, constraints, and bootstrap information
  - Customer data

**Task:**
Create a solution database.

**Objective:**
Capture design details in use cases.

**Key Influencers:**
- Customer’s desire to migrate original data
- Available skills to use TTS

**Output Work Products**
- Solution database

---

**Project Type:** Either New or Transition  
**Project Phases:** Implementation Phase

Figure 3-9  Creating a solution WebSphere Commerce database
Using the MassLoader tool for new projects

For new development projects, using the MassLoader tool that is shipped with WebSphere Commerce is an appropriate strategy for creating a solution database. This strategy has five steps:

1. Create a database that contains all the default schema, constraints, and data.
   
   If a specific configuration is required for the database, the best approach is to copy and modify the database creation scripts found in the WebSphere Commerce schema directory. For example, if the project requires tables to reside in specific tablespaces, the design team should configure the database creation script accordingly. If, on the other hand, the target site needs to support high volumes, the design team should determine if capacity planning is required. A database administrator should be consulted for capacity planning guidance.

2. Create custom schema.
   
   One of the core tasks of the macro design subphase is for the design team to determine if the out-of-the-box database requires schema changes. If it does, then the design team should create scripts that will extend the out-of-the-box schema with new tables, constraints, and indexes. These scripts may have already been created during the build of the development sample database. Wherever possible, reuse or modify existing scripts.

   **Note:** Out-of-the-box tables should never be altered in any way. All schema extensions must be in the form of a new table, that is, an extension table.

3. Create and load data.

   The task of loading data into the database can be divided into three activities:

   a. Pre-load
      
      Pre-load refers to loading initial information prior to the actual catalog data. An example would be the bootstrap information that describes the store, default access groups, or even users. The same information may also be created through other tools, such as WebSphere Commerce Accelerator.
b. Load.

Loading is the major activity of inserting catalog-related data, such as catalog structure, categories, and pricing information. The loading should also populate any customized tables. The WebSphere Commerce MassLoader tool may be used as part of a process to take data from delimited files that were created previously. The delimited files may be subdivided into tables or subsystems to help divide the effort among the team.

c. Post-migration

Post-migration involves updating any remaining parts of the database after the main load of the database has completed. An example might be updating the KEYS table.

4. Load referential integrities.

If there are any custom tables where constraints or indexes are required, the constraints or indexes should be established after all the data has been loaded. If a large quantity of data must be loaded, and time is an issue, it may be helpful to drop all constraints after performing step 1 on page 58, and then re-create all the constraints during this step. Dropping the constraints will allow MassLoader to load data without having to perform time-consuming checks.

5. Check the data.

After all data has been loaded into the tables, the design team should perform the following simple checks to ensure the data has been loaded correctly:

a. Check the load file logs to see if all the rows were inserted.
b. Check all the referential integrity logs to see if all the constraints loaded successfully.
c. Check all pages and data through a Web browser.

Note: One quick way to perform the checks is by searching for the words “warn” or “error” in the log file.

Using the TTS Data Migration Tool

Using the TTS Data Migration Tool consists of four stages:

1. Use information gathered during the design phase to build a spreadsheet that maps source schema to target schema.

The spreadsheet must conform to the specific TTS Data Migration Tool format and rules. Only then will the spreadsheet support the functionality needed in migrating data. The format and rules are described in the Transition Tool Suite User Guide.
2. The developer uses TTS tools to transform the spreadsheet to XML.
3. Generate script files for source data export to flat files from the XML files.
4. Use data migration scripts to populate the new schema with data from the flat files.

Using the TTS Data Migration Tool for transition projects

The first two steps just outlined for data migration in new projects apply to transition projects. The key difference in data migration activities between projects is that database constraints should be dropped before commencing the load in transition projects. Dropping the constraints will reduce the load time required by large sites.

**Note:** It may be helpful to drop the database constraints after the development sample database has been created.

This strategy has six steps:

1. Create a default database.
   Creating a default database corresponds to step 1 on page 58 for new site projects.

2. Create custom schema.
   Creating the custom schema corresponds to step 2 on page 58 for new site projects.

3. Migrate and load data.
   As with the new site project strategy, the migration and data loading step may be divided into three activities:
   a. Pre-load
      Pre-load populates the database with data, such as the bootstrap file containing STOREGRP information and contracts, that load and post-load activities may require.
   b. Load
      In contrast to a new site project, a transition project does not normally require the creation of new catalog data, unless, of course, the business requirements dictate one. All the data for the migration would come directly from the existing production database, and TTS supports dividing the workload across the design team.
   c. Post-load
      Post-load performs any updates required by the database once all the main data has loaded, such as updates to the KEYS table.
Ideally, in the implementation phase, the project team should get a current copy of the production database for their testing of the migration scripts. The tests will highlight discrepancies that require correction before the migration scripts can be used for data migration. In practice, some customer databases may be very large, so it will be necessary to “trim down” the database to make it more manageable. It is possible that the customer may even have a production test database that would be ideal for this purpose.

At periodic intervals, the project team should test the migration scripts by migrating a copy of the original database into the development and test environments. Again, this test will help identify any errors at an early stage, rather than leaving them until the site is almost ready to go live.

Normally, migration scripts are generated by TTS using a simple Lotus 123 spreadsheet, previously prepared by mapping the source tables and columns to the target tables and columns, including any custom tables, along with additional logic to facilitate the mapping. The user then saves the spreadsheet as a CSV file that feeds into the Data Migration Tool. The migration tool applies XSL translation to create a representation of the mapping in XML format. The XML representation is then processed by a Java-based tool to build Export and Load scripts, with corresponding SQL queries in the native DB formats. This process is illustrated in Figure 3-10.

The final output of the tool is a simple collection of SQL statements in the form of Export and Load scripts, which may then be applied to the source and target databases in order to perform the actual migration.

The extraction and loading of the data are standard tasks for the database management system tools. These out-of-the-box tools may therefore be used to minimize the time needed to perform the migration. Standard spreadsheets that map WebSphere Commerce 4.1 data to WebSphere Commerce 5.1 or 5.4 schema are available with current versions of TTS.

The only outstanding task for a customized database is the mapping of custom tables to out-of-the-box tables, or from custom tables to custom tables. In multitier configurations, the extraction and loading tasks can occur on separate machines, independently of the WebSphere Application Server.
system. The data files and scripts, however, must reside locally on the
database system.

4. Load referential Integrities.

Having performed the main data load, the design team should re-enable the
default out-of-the-box constraints that are needed to keep the data's integrity,
along with any custom constraints.

5. Check the data.

Checking the data corresponds to step 5 on page 59, for new site projects.

6. Delta migration.

On some sites, particularly larger or mission-critical sites, there may be a
requirement to keep the old site active and operational during the migration
and verification testing activities for the new site. Running parallel sites over
the course of two or more weeks is not an uncommon requirement. Data held
on the old site would, therefore, have been updated during this time. Any
changes made to the old site will need to be migrated to the new site. Using
the TTS Data Migration Tool, a developer will be able to modify the mapping
spreadsheet, by adding a timestamp predicate to each WHERE clause of the
SQL statements so that only changes made since the specified date will be
migrated across.

Other macro design issues

There are many other macro design issues that should be considered as part of
the design phase. These include topics, such as coding guidelines, naming
conventions, and so on. However, these topics are heavily dependent upon
project team preferences, house style, and customer IT team involvement. Such
topics, however, fall outside the scope of this Redpaper and will not be
considered further.

3.2.4 Micro design subphase

The micro design subphase follows the macro design subphase. Here, the core
activity is determining what part of the data access layer should be implemented
with entity EJBs and what part with session EJB-based LOQS QueryCommands.

For this activity to proceed smoothly, the design team must have a sample target
WebSphere Commerce database, as described in “Creating a WebSphere
Commerce site database” on page 55.

One of the main goals of the third iteration of the design phase is to build on the
macro design by capturing the detailed design decisions. The capture of detailed
design decisions is a key requirement needed to start the implementation phase.
The design decisions should be recorded in the micro design document, which
the customer should review and approve, that is, obtain appropriate sign-off on the design details.

A core micro design activity is defining and creating the appropriate interfaces between commands, JSPs, and the persistence layer of EJBs, data beans, and LOQS objects. Pseudo-code representing business logic in commands should also be recorded in the micro design document.

An overview of the activities and work products of the micro design subphase is provided in Figure 3-11.

![Figure 3-11 Micro design subphase](image)

Before proceeding to the next phase, the design team should verify that either the database has been migrated, or that a target database has been created for the project, as indicated in Table 3-3 on page 56. As determined by existing technical conditions, the database design might require updates should the data model be changed after it has been reviewed with the customer during the micro design subphase. No matter what happens during micro design, all database creation tasks must be finished before the end of the micro design subphase. Failure to produce a valid database at the end of the micro design subphase will impede progress, because workarounds and delays will be required to achieve code requirements during implementation phase.
Source SQL statements from earlier sites, such as those built using WebSphere Commerce Suite Version 4.1, and which were collected in the previous phases, should now be analyzed to determine which ones may be replaced by suitable data beans. EJBs are best suited to replace statements with data read or data write requirements, and LOQS should be used to generate read-only beans for statements requiring read-only access. SQL statements that are candidates for entity EJBs need to be mapped to existing EJBs or translated with the SQL Translator and implemented as new EJBs. The LOQS tool works using the target site schema, for example, WebSphere Commerce Version 5.4. The resulting translated SQL statements are saved in an XML file.

When complete, the output should be validated with the SQL Validator to ensure the SQL translation process completed without error. This tool requires a source and a target database (which should have been created in the previous phase) to accomplish the validation.

Next, it is necessary to identify the data access objects (DAOs) that can be shared by one or more query commands (QCs), containing a target SQL statement. All of the decisions for the persistence layer design should be captured in the micro design document. The LOQS Programming Reference provides more details about these decisions and is part of the LOQS documentation included in the TTS distribution package. It may be desirable to create LOQS objects and beans during this phase. The project team will want to conduct detailed design reviews with the customer to make sure that the WebSphere Commerce design follows best practices and meets the customer's requirements.

Before proceeding to the implementation phase, verify that the database has been migrated and is available for use by the coding team. In addition, the project team should create a team development environment that uses a shared Source Control Management (SCM) team repository server and the common (migrated) WebSphere Commerce database.

The micro design activity should produce a list of line items to implement as part of the detailed project plan. If a site transition project is being performed as part of a consulting engagement, it is advisable to complete a second statement of work to document all project execution activities that will be required through to completion. The test strategy that began during the early stages of the project should evolve into a more detailed test plan.

A summary of the TTS tools that are applicable in the micro design subphase for transition projects is provided in Table 3-4 on page 65.
3.2.5 SQL extraction overview

In WebSphere Commerce Suite Version 4.1, SQL statements are embedded within the Net.Data macros and commands. Converting Version 4.1 SQL statements into corresponding usable Version 5.1 database queries requires extraction of the read-only queries from the Net.Data macros and commands. In a transition project, this task will provide a good opportunity to “houseclean” the system by re-writing queries so that they execute more efficiently. For example, multiple database queries could be replaced by a reorganized single query that can populate several beans at the same time.

Table 3-4 Using TTS during the micro design subphase

<table>
<thead>
<tr>
<th>Concept</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicable TTS tools</td>
<td>Document Helper Tool. SQL Translation Tool. SQL Validation Tool. Data Migration Tool. LOQS</td>
</tr>
<tr>
<td>TTS preconditions</td>
<td>TTS enhanced with customized schema information.</td>
</tr>
<tr>
<td>Transition assets generated</td>
<td>Target SQL statements that are tested and validated using target database. SQL Validation Report, created by using the SQL Validation Tool in the batch mode. LOQS objects and beans. Target (migrated) database.</td>
</tr>
<tr>
<td>Exceptions</td>
<td>None.</td>
</tr>
<tr>
<td>Notes</td>
<td>Using LOQS tools to generate Java code (beans and objects) is highly recommended in this design subphase.</td>
</tr>
</tbody>
</table>

**Note:** When transitioning sites from WebSphere Commerce 4.x, watch for SQL statements embedded in `if-else` macro statements. These SQL statements must be reduced into their respective conditions to form more than one SQL statement.

The IBM Software Services team has found using a database or repository to collect SQL statements to be helpful; this database or repository can be reused for tracking or maintenance purposes later in the project. The database or repository may hold information related to the verification of individual SQL statements and the suitability of each for transitioning through LOQS. It should
be organized so that every characteristic of the SQL statements can be captured and saved, and all project developers must be able to view it.

**Note:** In one transitioning project, a Lotus Notes® project database was used to track information about the SQL statements. In that project, there were a large number of customized tables and columns. This resulted in the extraction of over 400 distinct SQL statements from both Net.Data macros and commands.

### 3.2.6 SQL statement translation overview

SQL statement translation is the conversion of SQL commands from one database schema to another. SQL translation is only necessary when the database schema has changed between site, as is the case in most transition projects. In many projects, translating SQL statements is a manual task performed one statement at a time. The translation involves reviewing the original SQL statement and translating the query to fit the new database schema.

Manual translation of SQL statements can be very time consuming, and it is often prone to error. TTS, however, provides tools that automate much of the translation process. Prior to using the SQL Translator, though, the design team must have, or needs to generate, a mapping of WebSphere Commerce schema between product versions. TTS ships with out-of-the-box mappings, but these mappings will have to be modified or replaced if the source site has significant table customization.

It is possible to use the out-of-the-box TTS Mapper and append to it. However, it is important to avoid potential confusion arising from multiple possibilities in mapping translations.

**Note:** A custom mapper is created by simply replacing all the tables and columns in the out-of-the-box mapper spreadsheet file with one that maps exactly to that of the Version 4.x and Version 5.x database schema. It is important to ensure that there are no duplicate table and column pairs.

The mapper spreadsheet file is converted into a *.csv file by selecting **Save As**. The *.csv file is converted into the required *.xml file with a PERL-based TTS tool, which is described in the *Transition Tool Suite User Guide*. 
3.2.7 SQL statement performance and validation

After translating SQL statements, the developer should test the validity and performance of the output statements against their respective database versions. SQL performance and validation are important steps when migrating SQL statements. The developer must ensure the statements were written properly so that each will execute properly. Validating statement performance will ensure all statements execute quickly and that the desired query results are returned almost instantaneously.

The preferred methods for testing SQL statements are as follows:

- Manually against the database version
- Using the SQL Validator from TTS

In both cases, it is important to use real values in the queries prior to execution. If invalid data is used, there is a chance that the query execution will fail or return no results. Using real values is important when running the SQL Validator Tool as part of a batch process.

3.2.8 Overview of the Lightweight Object Query System

The Lightweight Object Query System (LOQS) is an extension to the WebSphere Commerce runtime. It provides a framework for developing and executing efficient read-only data access commands that is based on performing read-only database queries using query commands. A query command is a low-level WebSphere Commerce command, dedicated to the execution of an SQL query. The result of the execution of a query command is a data access object, of arbitrary type, that encapsulates the result set from the SQL query. Figure 3-12 on page 68 shows an overview of the relationship between the SQL and LOQS components.
LOQS allows the actual SQL statements executed by query commands to be registered and retained in one or more query registries. The registration and retention of query commands is useful for organizing the SQL statements, and it provides easier access to them during development and testing, without additional complex code changes, recompilation, and redeployment. It also provides easier auditing, inspection, and tuning of the SQL statements. For example, the query registry allows for better protection and control over the executed SQL by encryption of the registry or restricting user access rights in a production environment.

The LOQS programming model increases developer productivity by enabling uniformity and quality in the resulting code. Code quality is achieved through the application of best practices, which reduces the potential to introduce code defects. The benefit is that developers with entry-level EJB, JDBC, and WebSphere Commerce skills can produce high-quality results.

Because the LOQS tool does not require an instance of WebSphere Commerce to be loaded onto the same workstation, developers may deploy it to any developer workstation. In addition, the code-generation component of LOQS supports both new or transition WebSphere Commerce application development projects. The TTS itself integrates with the GUI wizard code generation parts of
LOQS. The Code Generation component supports both interactive and batch modes of operation, as illustrated in Figure 3-13.

In the interactive mode, a user can run the GUI-based wizard. The wizard takes an SQL statement as input, interactively gathers information that describes the input and output parameters of the SQL statement, optionally stores the user input in an XML metadata file, and finally generates the required Java query commands and data access objects. The resulting Java classes can then be deployed and executed using the LOQS Runtime.

LOQS run in batch mode, on the other hand, requires the developer to supply XML metadata files detailing the SQL statements that require execution. The developer can create the files either manually or by using GUI-based wizard. The batch mode allows generation or re-generation of one or many LOQS Java objects. The XML metadata, which serves as the input to the code generator, can initially be produced by the LOQS wizard or by hand and can later be edited with an appropriate XML tool if needed to apply larger-scale changes to the generated classes.
3.3 Design review

The design review is the first major opportunity afforded the design team to review the match between the main site objectives and the site's design to meet the objectives.

A good design will have certain characteristics:

- It should be focused on producing a site that will apply and demonstrate best practices.
- It should maximize the incorporation of reusable components, or where new components are designed, there should be consideration for future reuse.
- The design should comply with the spirit, as well as the letter, of the WebSphere Commerce programming model.

When reviewing a design, the design team should consider the design's context, especially so when an existing site is being transitioned. Earlier versions of WebSphere Commerce used different implementation architectures and technologies. In addition to transitioning the site, the customer must also invest in new tools, education, and techniques that will be needed to support the transitioned site.

**Note:** Generally, customers strive to reduce the number of platforms and technologies requiring direct support. Therefore, many customers will view favorably the opportunity to reduce costs when optimizing their existing site.

During transition projects, project teams focus on specific success factors, such as:

- No change to database schema, and only limited changes to database access plans, as optimized by the customer database administrators.
- Minimum impact on existing back-end and supporting infrastructure.
- Minimum increase in maintenance and retraining costs.
- Maximum preservation of the shopper experience; unnecessary changes should be avoided.
- Maximum, on-going parallel work to avoid a moving target situation.
- Careful analysis of previous, current, and predicted database use and needs to come up with an appropriate data services architecture.
- Optimum scheduling, use of available team skills, and project management.
- Appropriate value-add to the WebSphere Commerce technology.
3.3.1 Data analysis review

The data model proposed for the site should be reviewed carefully. Even large projects would not require large numbers of additional tables.

Note: A particularly large project worked on by the authors required just 50 additional tables. Should your project seem to need more than 50 tables, you should re-evaluate the need for each new table thoroughly.

Some tables will require transactional reads and updates. In those cases, Container Managed Persistence (CMP) EJBs work best. They require less development work and, by default, will provide optimized performance.

The majority of tables are likely to require non-transactional, read-only access. Many of them contain data that changes infrequently, if at all. In those cases, use of CMP EJBs is inappropriate, because the site will not benefit from the advantages afforded by CMP.

The main reason for not using CMP EJBs is that they will:

- Increase the number of database accesses by \((N-1)\times\text{rows} + 1\) SQL statements.
- Increase the network traffic; more SQL is used, and the results will return whole objects instead of 1 or 2 columns.
- Increase resource management demands, to support tasks such as passivation and memory reclamation.

Note: This value-add refers to sensible use of out-of-the-box components to avoid re-implementing components that are already available. Some real-world projects, particularly those with less skilled J2EE developers, fall into the trap of re-implementing EJB components using simpler but more restricted technology. Initially, the re-implementation of EJBs appears to produce gains; these gains usually turn into losses quickly when the code is implemented and problems arise due scalability and performance issues.
3.4 Summary

In this chapter, we discussed core design activities for project teams creating new sites or transitioning existing ones. For each activity, we summarized how it should be performed, what roles and team structure would best suit it, and provided a high-level overview of the skill requirements.

We then described the major phases of the design work, including detailed information about the design of the site data model and how it may be developed or transitioned from an existing site.

In the next chapter, we explain how the design details created during this phase can and should be used to implement the site.

**Note:** In a very large project that required over 450 SQL statements:

- Nearly 300 of those statements accessed a single table.
- Just over 80 of those statements involved a join of four or more tables, many of which were out-of-the-box WebSphere Commerce tables.

Through experience, IBM Software Services has learned that there is no correlation between the number of SQL statements required and the project’s data model. The number of statements reflects the way the logic of the application is built, optimized, and tested.
Preparing and implementing the commerce site

In this chapter, we describe implementation activities of the development methodology. The core activity during the implementation phase is application coding, which may be performed using different strategies. We discuss work assignments for the project and implementation teams. In addition, we introduce initial testing procedures for assessing site code. We also explain the appropriate steps for creating an integrated test environment.
4.1 The project implementation phase

Following the successful completion of the micro design subphase, the project team may now begin developing site code using the specifications produced during the design phase. It is assumed that the project manager will staff the project team with suitable developers and skills, as defined in the project plan and Appendix A, “Project team creation and common roles” on page 127. Figure 4-1 provides an overview of the implementation phase.

![Implementation phase overview diagram]

Figure 4-1 Implementation phase overview

Figure 4-2 on page 75 summarizes the activities and work products produced during the implementation phase.
4.2 Implementation and team work assignment

We begin with a discussion of a strategy for implementation and team work assignment, based on continuous and incremental design. The strategy is based on best practices that have proven successful on various commerce projects. The implementation and team work assignment strategy specifies that newly implemented project elements, such as JSPs, commands, or EJBs, be unit tested and integrated incrementally. This strategy differs noticeably from the traditional waterfall model, where each major activity follows a sequential development approach. In other words, the site components are coded, then unit tested, and finally, integrated.

Prior to starting the implementation phase, the project manager should complete the following two tasks:

1. Select a team organization structure, as discussed in “Creating the project team” on page 128.
2. Define the model for producing site code that is test ready, preferably site level.

Figure 4-3 on page 76 provides an overview of a continuous and incremental development and code integration testing strategy.
Even though the continuous and incremental development strategy is very useful, it must have a solid anchor point. In “Methods of approaching design activities” on page 48, we advocate creating a use-case-centered design. Through experience, we have found that anchoring code development on use cases produces tangible benefits.

Use cases and code components, such as JSPs and commands, can be organized so that the project team start work item assignments and then monitor dependencies within and between use cases. An example of the team organization and work assignments is shown in Table 4-1.

Table 4-1 Team assignment examples

<table>
<thead>
<tr>
<th>Code category</th>
<th>Component name</th>
<th>Use case 1</th>
<th>Use case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSPs</td>
<td>Login.jsp</td>
<td>John</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Place-order.jsp</td>
<td>Joe</td>
<td></td>
</tr>
<tr>
<td>Commands</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A key factor in executing this strategy is the team structure, as discussed in “Creating the project team” on page 128.

If the team structure is based upon the master-of-all strategy, as described in “The holistic strategy” on page 130, the project manager should assign use cases and most or all of their components to skilled WebSphere Commerce developers to produce the required code. This strategy enables the developer to code, unit test, and integrate code elements for each use case prior to moving to the next use case. The effect of this assignment model is that the use case columns illustrated in Table 4-1 on page 76 will have only single names; only one person would be implementing the complete use case.

On the other hand, if the team structure is organized as described in “The core competency strategy” on page 129, Table 4-1 on page 76 will have multiple names in each column for each use case.

Alternatively, the project manager may decide to use a combination of core-competency and holistic strategies. In that case, the columns of an assignment table would contain a mixture of names. Some columns would have a single name, while others would contain multiple names.

When assigning developers to use cases, the project manager must ensure critical use cases are identified first and that all dependencies are highlighted for the team. This activity enables the project manager to prioritize and sequence the development of code components, and it affords them the opportunity to monitor progress.

Progress may be tracked in many ways. For example, one method assigns completion percentages to specific implementation stages:

- A 50% measure might indicate the percentage of total effort that is required to complete component coding.
- A 30% measure might indicate the percentage of total effort required to integrate the use case components.
- A 20% measure might indicate the percentage of total effort required to complete unit testing.

<table>
<thead>
<tr>
<th>Code category</th>
<th>Component name</th>
<th>Use case 1</th>
<th>Use case 2 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login.cmd</td>
<td>John</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order.cmd</td>
<td>Jill</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EJBs</td>
<td>User</td>
<td>Jin</td>
<td></td>
</tr>
</tbody>
</table>
Even a simple strategy such as this should aid the project manager track team progress throughout the implementation phase.

Once a use case has been implemented, that is, coded, integrated, and unit tested, its components should be published to a test server and retested. Integration testing each use case within the WebSphere test environment should be considered an initial step in code testing. The task of publishing coded use cases to a test server for additional testing improves the project team's ability to integrate use cases efficiently. Publishing and retesting use case code is discussed in more detail in 4.5, “WebSphere Commerce Subsystem Integration Test phase” on page 96.

While progressing through the implementation phase, the development team should publish regular code base versions and builds to the test server, as described in 4.3, “Creating an integrated development and test infrastructure” on page 79. This activity supports incremental code development, integration, and testing. The team should also create a Build Verification Test (BVT) process, one where each new build is executed before making this build available to the team. A BVT covers one or more test cases or scenarios, and it should test the critical paths through the code that are operational and require validation. Executing the BVT will ensure that the new build has no obvious errors. The BVT process should start as soon as one or more use cases is ready and has been tested on the test server. In addition to orchestrating the build process, the project manager should arrange periodic code reviews with the team to ensure code quality is acceptable and as desired.

The back-end test system should reasonably mimic the interfaces and communication systems of the production systems to ensure the code can be tested using end-to-end scenarios. The better able the team is to create an accurate test environment, the more effective and thorough its tests will be.

As seen, the first serious testing of site code begins during the implementation phase.

**Note:** These percentages are clearly simplistic, but help illustrate a means for quickly assessing overall progress. For example, using these percentages, if half the work on coding, one-third of the component integration, and one quarter of the unit testing work is complete, the overall progress could be estimated as 25% + 10% + 5% = 40% complete.
4.2.1 TTS use during the implementation phase

During transition projects, we recommend using the Transition Tool Suite (TTS) at appropriate points throughout the implementation phase, as much of its tools are suited for implementation activities. For example, as part of implementing the data access layer for the project, the Lightweight Object Query System (LOQS) Code Generation tools could be used to code the LOQS objects and beans identified in the design phase.

If corrections or modifications are required in the database schema, the project team should consider using the Data Migration Tool repeatedly to create new versions of the database. The project team may test the newly created database by running the SQL Validation tool on SQL statements that have already been migrated to the target system. This process will validate the integrity of the database and its data, as it will enable the project team to detect data errors quickly. If practical, this activity should occur prior to the implementation of code or before the test team discovers errors when coding and unit testing use cases. Table 4-2 summarizes the use of TTS during the implementation and site testing phases.

Table 4-2  Use of TTS during the implementation and site testing phases

<table>
<thead>
<tr>
<th>Concept</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicable TTS tools</td>
<td>LOQS</td>
</tr>
<tr>
<td></td>
<td>Data Migration Tool</td>
</tr>
<tr>
<td></td>
<td>SQL Validation Tool</td>
</tr>
<tr>
<td>TTS preconditions</td>
<td>TTS enhanced with customized schema information</td>
</tr>
<tr>
<td>Transition assets generated</td>
<td>LOQS objects and beans (Java code)</td>
</tr>
<tr>
<td></td>
<td>SQL validation report (created after using the SQL Validation tool in batch mode)</td>
</tr>
<tr>
<td></td>
<td>Target (migrated) database</td>
</tr>
</tbody>
</table>

4.3 Creating an integrated development and test infrastructure

To be effective, a project team needs both a development and a test environment. Although each is distinct, they are inter-related and connected.

The environments are as follows:

- The development environment

  This environment is used by the project team to develop and share software between team members and then to unit test this software before it is integrated to create a coherent e-commerce site.
The test environment

This environment is used by the project team to integrate all code pieces and then to test the integrated units, as described in 4.5, “WebSphere Commerce Subsystem Integration Test phase” on page 96, and 5.2, “Functional Verification Test (FVT) phase” on page 110.

The physical hardware to implement the development and test environments could be shared. Prior to the start of the implementation phase, all the required hardware and software, and the team development environment and site test infrastructure, must be ready and available. The project team will require both the development and test environments during the implementation phase to develop and test site code.

Figure 4-4 provides an overview of a strategy for creating an integrated development and test environment. This strategy is based upon best practices, and we recommend its use whenever practical. Both system and database administration skills will be required to implement it.

**Integrated Development and Test Environment Strategy**

**Input Work Products**
- Team hardware and software
- Code architecture

**Task:**
Create an integrated development and test environment.

**Objective:**
Ensure development and test activities and linked via an integrated environment.

**Key Influencers:**
- Availability of team members who can perform system admin, database admin, and data migration tasks
- Team development goals
- Skill level of development team

**Output Work Products**
- Integrated team development and test environment
- Approved team development and test processes

**Project Type:** Either New or Transition
**Project Phases:** All

*Figure 4-4  Integrated development and test environment strategy*
This Redpaper assumes that the project team works apart from the customer's IT team, generally in geographically distinct locations. Due to the geographic separation, the project team is assumed to have its own integrated development and test infrastructure, as well as processes. The customer IT team, on the other hand, may or may not have its own development and test infrastructure or processes.

Because this paper also assumes that most of the development work is to be performed by a project team, the project manager should not be overly concerned with synchronizing the project team's development and test environments and processes with those of the customer's IT team, if applicable, during the implementation phase. Due to the focus of this paper, most of the activities for the implementation phase is presented from the project team's perspective.

### 4.3.1 The team development environment

Setting up a proper development environment is a critical project team activity, because the environment can be considered the “nerve-center” during site implementation. An efficient development environment will enable all team members, that is, the developers, to be fully utilized and productive when coding and unit testing site components.

Creating the development environment involves setting up a number of components. Typically, team development environments consist of a repository server to store Java and non-Java file assets, a shared database, and individual code development systems. It may be necessary to have share access to WebSphere Commerce Payments if required, or shared access to an WebSphere MQ server facility.

Figure 4-5 on page 82 illustrates each element of the team development environment.
We can now describe each element of the team development environment:

- **Shared database**

  Setting up a shared database for development and test purposes may help reduce setup time and should also reduce the chance for data errors. This database can reside on a testing server (as described in 4.3.2, “The shared test environment” on page 86), or on a dedicated database server machine, depending on the size of the project. If the project team consists of more than three to five developers, we would recommend setting this up on a dedicated server.

  **Note:** Using a shared database may constrain the development environment so that constant connectivity is required. This would limit the scope for mobile development and may be inconvenient during circumstances such as presentations or demonstrations.

In Figure 4-5, a dedicated database server hosts a number of databases, configured for code development, test, and QA activities. Only the database
client software needs to be installed on the developers’ systems. If a complete database system is installed and configured on each developer’s code development system, it will be necessary to maintain and update each database whenever there is an update to either test data or database schema. During the implementation phase, the database tends to be updated or fixed regularly, and these updates will significantly reduce access time for developers if their systems must be updated. Further, application of a database-related fix or FixPak must be duplicated on each developer’s system as well. These are the main reasons for recommending a shared database server for a project team during the implementation phase.

Source code management (SCM) system

All projects, be they large engagements with many developers or small ones with only one developer, will benefit from a source code management system. The project team should implement one regardless of the size of the project.

Configuring this component will depend on the tool that is being used to develop Java assets. If VAJ is used for development, EMSRV must be used as the repository to store and manage Java assets, while another source control program, such as Rational ClearCase, could be used to manage non-Java assets.

If, on the other hand, the project team selects WebSphere Studio Application Developer as its development tool, setup of a second source code management system might not be required, because WebSphere Studio Application Developer ships with support for versioning and control packages, such as Rational LT and CVS. Plug-ins that support other SCM systems are also available. Any SCM may be selected, according to the unique requirements of the project.

**Note:** At the time of writing, WebSphere Studio Application Developer is not an officially supported development environment for WebSphere Commerce Version 5.4 sites.

Developers’ integrated development environment (IDE) system

The developers’ IDE system should be installed with a WebSphere Commerce software stack and may be based on either VAJ or WebSphere Studio Application Developer. The developers’ code development system should connect to the shared database and shared source code repository. All developers should code and unit test in the WebSphere test environment, which supports both VAJ and WSAD. After initial coding and testing has been completed, the project team should continue test activities in a project test environment, which is described later in 4.3.2, “The shared test environment” on page 86.
Shared WebSphere MQ system

Because most projects are likely to require some form of back-end connectivity, for example, to SAP, the project team should install and configure a MQ server on either a separate system or using shared resources in the project test environment. An overview for situating MQ in the test environment is described in 4.3.2, “The shared test environment” on page 86.

Where needed, MQ client software should be installed and configured on developer systems. It may not be necessary to install and configure MQ client software on all the developers’ systems. It is likely that only those developers who are developing code that needs to connect to MQ and send or receive request and response messages will need MQ client software.

Shared WebSphere Commerce Payments

Should the site accept online payments, the developers’ systems will need access to a shared WebSphere Commerce Payments system.

As with a shared MQ system, it is possible to install this software on a test server or use a dedicated server to host it. The developers requiring access to this functionality need to have their development system properly configured to use the WebSphere Commerce Payments.

Potentially, all the developers can be connected to the shared systems, namely the repositories, at the same time. The developer systems will host the IDE environments, normally VisualAge® for Java (VAJ) or possibly WebSphere Studio Application Developer for WebSphere Commerce development, a database client, and tools to develop non-Java assets. In theory, a database server and a commerce server could also be installed on each development system. However, such a configuration would require extensive knowledge for integrating the products. There would also be a very high load on the system, possibly affecting the system responsiveness and, by implication, affecting the productivity of the developers. It is strongly recommended that only the minimum necessary software, such as the IDE, the database client, and tools to develop non-Java assets, be installed in the development systems.

The system administrator will play a key role in setting up the development environment. The database administrator will also be required to install and configure the shared database server.
The details about the roles and responsibilities for the development team in setting up and configuring a shared development environment are provided here and summarized in Appendix A, “Project team creation and common roles” on page 127:

1. Systems administrator

   The systems administrator will install and configure the source code management system. If VAJ is being used as the tool to develop Java code, the system administrator should also install and configure the shared team repository, EMSRV. A separate source code management software system, such as Rational ClearCase, could be installed and configured for non-Java assets. EMSRV can also host non-Java assets.

   The systems administrator will install and configure a shared WebSphere Commerce Payments system, if needed.

   The systems administrator will install and configure a shared MQ system, if needed.

   Finally, the systems administrator must document the process for connecting these various components and systems.

2. Database administrator

   The database administrator installs and configures the database server. The database administrator must also create and populate a database for developers to use during the implementation and unit test phase. A separate set of databases should be created and configured for WebSphere Commerce integration testing (as discussed in 4.5, “WebSphere Commerce Subsystem Integration Test phase” on page 96), and Functional Verification testing (as discussed in 5.2, “Functional Verification Test (FVT) phase” on page 110).

3. WebSphere Commerce developers

   The WebSphere Commerce developers will normally install and configure WebSphere Commerce and VAJ or WebSphere Studio Application Developer for code development.

   They will install and configure client software for source code management, MQ client, and other necessary software.

   Irrespective of the IDE they intend to use, they should install and configure the IDE to work with the shared team repository.

4. Team repository administrator

   For smaller projects, the system administrator can also act as a team repository administrator:

   - The administrator will be responsible for importing the WebSphere Commerce repository to the shared repository.
– The administrator must then import other required repositories, such as the LOQS from TTS.
– The administrator creates appropriate projects and packages as outlined in the macro design and micro design documents.
– The administrator creates users and assigns appropriate permissions, as outlined in the macro design and micro design documents.

4.3.2 The shared test environment

The project team’s use of the shared test environment is described in detail in 4.5, “WebSphere Commerce Subsystem Integration Test phase” on page 96 and 5.2, “Functional Verification Test (FVT) phase” on page 110, but it is helpful to highlight key points here.

The main objective of setting up a shared test environment is to integrate code pieces from all developers, in the form of unified code drivers, and deploy them on the test servers for comprehensive testing.

One or more sets of test servers can be used for a project team. We recommend that each project team should have minimum of two test server configurations so that testing and code fixing tasks can be performed in parallel.

A test server setup can be single tier or multitier. The closer the match between the test server setup and the ultimate production environment, the better the chances of finding and addressing any infrastructure- or configuration-related issues early in the test cycle.

If the test setup is in a single tier configuration, this means that the database, the WebSphere Commerce server, the WebSphere MQ server, and WebSphere Commerce Payments are installed and configured on a single hardware box. These test servers may also host shared facilities, such as the database, the WebSphere MQ server, or WebSphere Commerce Payments software used by the WebSphere Commerce developers during the implementation phase, as described in this chapter. It is the sharing of physical resources that establishes the basis of the integrated development and test infrastructure for the project team.

4.3.3 Configuring a development process

In addition to setting up hardware and software, the project team should establish a development process that ensures consistent code development.
Doing so will create a good and effective team working environment. The development process should address, at a minimum, the following questions:

- How many versioned code bases need to be created in a week and at what time of the day?
- Who should participate in creating new code base versions? How will the code solution, project, and package ownership be assigned to different team members?
- Who should fulfill the role of team repository administrator? Although the system administrator might assume this role, this should be a deliberate rather than default decision.
- When should all team members receive new “drops” of the current code solution so that code written by the team can be integrated, used, and tested as the implementation phase progresses?

The development process must also take into account how database updates are to be managed as problems are discovered in the schema or either in sample or migrated data. The database administrator, with assistance from the data migration expert, would typically perform this role and would be responsible for coordinating appropriate changes to all instances of the databases on all servers with the system administrator.

### 4.3.4 Configuring a simulated production environment

The development and testing environment should mimic the production environment as closely as possible. However, there may be circumstances where it is not feasible to mimic fully the production environment due to hardware, software, and tool constraints.

In those situations, the project team may need to simulate a production environment in the development and test environments, especially when the development and test environments are not located at the client site. For example, if the project team is not at the client site, and the site requires connectivity to a back-end system, such as a mainframe resource, then it may not be possible to access the back-end system from the development and test systems. The reasons why may be many, such as lack of network connectivity from the development and test site to the client site, firewall issues, customer preference, and so on.

The back-end simulator should be designed to accept request messages and produce response messages, based on the request message identifier. Although the simulator may not fully test all back-end scenarios, it may be sufficient to test code that at the time requires back-end connectivity. This simulator could also be used for FVT, until the code is transferred to customer location for end-to-end
system integration testing (as described in 5.3, “System Integration Test (SIT) phase” on page 112).

Following are two examples of how creating a simulated environment aided project teams:

- A customer was using a proprietary method of user authentication through a Lotus Domino™ server. In this case, the project team decided to simulate the authentication mechanism, using a JSP to pass in the authentication code. This meant that during the implementation phase, the project team had a realistic way of testing this proprietary mechanism.

  An alternative approach would be to install and configure a Lotus Domino server within the development and test environments. This alternative was rejected, because it would have required more effort to install, configure, and manage the server than creating and using the JSP-based simulation tool.

- In a commerce engagement project, the team implemented a simulator that helped test the connectivity of WebSphere MQ with the customer’s mainframe systems. The simulator generated appropriate response messages for the test request messages. This simulator also helped test the format of the request and response messages.

  The simulator designed by the services team also randomly selected a response message from a pool of test messages previously collected from real back-end transactions. The selection was constrained by the request message ID. This simulator helped the project team with troubleshooting connectivity, for example, by identifying message format mismatches, performance issues with back-end connectivity, and so on. The cost of implementing a simulation tool such as this would obviously vary from customer to customer. This additional cost should be carefully planned early in the project to help minimize or eliminate impact on the project plan.

### 4.3.5 Source code management

As with any software development project, code components for an e-commerce site will be written by several programmers working over a significant period of time. Any modifications to code need to be tracked and managed, so if a problem occurs, it is easier to identify the change or changes that caused the problem.

Using a source code management systems brings a number of benefits:

- It prevents two or more developers making changes to the same code component at the same time.

- Earlier versions of code can usually be obtained easily by backtracking through the code management system.
Any change made to a component, along with the reason for the change and the name of the developer, can be listed.

Backups are easier to implement and manage.

It is possible to extract or export a current snapshot of all the code components, making builds easier to implement.

### 4.3.6 The build process

The build process is a mechanism for creating a unified code driver that has all the necessary code and non-code pieces to allow some level of site testing.

The project team should create a build process for the project. This process could be shared with the customer IT team so that a similar activity can start at the appropriate phase in the project life cycle, usually after site launch.

The build process should be finalized before exiting the design phase, and the project manager should identify a *build person* who will:

- Be in charge of creating regular builds
- Agree to a build schedule with the team
- Identify all elements and the contents of each build
- Establish roles and responsibilities of team members in creating and using builds
- Create the build deployment process
- Establish actions that must be taken when a build failure occurs

Although the technical aspects of creating and deploying a build will be different for WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.4, in general, the build process can be used for creating builds for any type of e-commerce site.

A build process may be created and used for code development during the implementation phase and, additionally, for all types of test phases. During test phases, the project team can use the build process to apply tests in order to build or generate test results.

**Note:** If the project requires developing the application on one platform before deployment to a different platform, it is worth porting Java code and JSPs as they are produced. Despite the fact that these components are written in Java, there may be minor problems encountered in the porting that should be addressed as quickly as possible.
The build creation and deployment process listed here assumes that the project is using VAJ and WebSphere Commerce Suite 5.1, although similar tasks will be performed for other IDEs, such as WebSphere Studio Application Developer:

1. The build schedule

   The first task is to define a build schedule for both development and testing phases. During testing, the builds tends to be performed more frequently than during the development phase.

   It is important to start the build process as early as possible during development to establish how well the use case integration is progressing.

   For example, during the development tasks of one ISSW engagement, the IBM team created builds three times a week, on every Monday, Wednesday, and Friday evening. This team started the build process in the second week of the implementation phase. During the site test (FVT) phase, the builds were created everyday so that defects were fixed with a quick turnaround.

   We recommend creating a schedule that suits the project needs. Creating builds too often during implementation will likely require too much time to deploy, which in turn, will prevent the development team from being able to use the builds.

   Conversely, not creating enough builds during FVT may lead to a longer testing phase, as closing defects will take longer.

   During the SVT and UAT phases, as described in Chapter 5, “Systematically testing a commerce site” on page 103, the build frequency should typically be once or twice a week. In practice, the frequency depends on the number of defects that get reported by the customer IT team.

   Emergency builds can always be created during FVT and UAT phases when a build fails with critical, or “show-stopper,” problem.

2. Roles and responsibilities

   Although creating a complete build involves participation from several team members, the project manager should assign one individual to be responsible for creating builds. This person may either be one of the developers or the team repository administrator.

   The build person will be responsible for following the build process to create builds at the scheduled times. They should also ensure that code to be included in the build is delivered by the code-cut-off time. The code-cut-off time is when the developers decide to release their code as part of the packages to be included in the next build.

   If VAJ is being used, the project and package owners should ensure the necessary projects are versioned and released for the build.
Next, the build person may create a “solution” of all the necessary projects that goes into the build and then export the Java code as .jar files. Developers should also make sure that any EJB .jar file is exported if necessary. The EJB .jar file does not need to be exported unless there is a newer version of the EJB projects.

3. Creating the build content

Build creation involves a number of steps. Each build is composed of many components. In WebSphere Commerce 5.4, a build is packaged into an EAR file and deployed on a target WebSphere Commerce server. However, in WebSphere Commerce Suite 5.1.x, a build may be packaged as zipped file, such as a .zip file or a .tar file. We recommend packaging the complete build into one .zip file, as doing so now will reduce problems later.

For example, the build deployment script, also called the build automation script, can be run with the corresponding older .zip file, and it can then automatically deploy the build onto the server.

Version control for builds can be managed more easily with one zipped file than with loose individual build files, especially if multiple .zip files are used for each build. It is usually possible to include the zipped build files in a source code management system to get benefits from the version control mechanism.

A typical WebSphere Commerce Suite 5.1 build includes:

- A .jar file for commands and utilities
- A .jar file for exported EJBs
- Separate non-Java assets, such as JSP files, .html files, property files, and images

Information held in database DDL files may be required for populating database. Finally, there may be other specialized custom components.

4. Build deployment

After creating a build, the project team must deploy it to one or more test servers. The build deployment process should be repeated for each of the test servers.

Deploying builds on WebSphere Commerce Suite 5.1 is different from WebSphere Commerce 5.4, as a result of the enterprise application support provided within WebSphere Application Server Version 4.0. In particular, the use of the Application Assembly Tool and Enterprise Application Archive (.ear) files is a new facility for WebSphere Commerce 5.4 arising from J2EE support.
Furthermore, deploying the build for the first time may require additional steps. Some of the deployment steps may need to be executed manually. Subsequent consecutive builds may be automated with scripts.

At a high level, the build deployment process consists of the following:

a. Copying Java and non-Java assets to the testing server under the correct directories
b. Deploying or updating EJBs
c. Populating the database with necessary changes, for example, inserting data into URLREG, VIEWREG, and CMDREG tables
d. Pre-compiling JSPs
e. Restarting WebSphere Commerce

The project may also need to modify the classpath settings on the WebSphere Application Server if the build is being applied for the first time for WebSphere Commerce Suite 5.1.

4.3.7 Build automation

Automating the processes for build creation and deployment will save a lot of time. Most of the tasks can be automated through various scripts, with each task having a specific set of automated activities:

1. A build can be created automatically by writing scripts that can extract non-Java assets from the SCM. Most SCM vendors provide tools to work with their products through scripting.
2. Scripts can then be used to combine the commands .jar file, the EJB .jar file, and non-Java assets into one single zipped file format.

   **Note:** Both .tar and .zip format files can be used.

3. Next, a script can be used to copy the build file over to the appropriate test servers.
4. After the file is copied over to the test server, a script on the test server can extract the files and copy them to the appropriate directories. For example, .jar files will normally be located in the WebSphere Commerce lib directory, while non-Java assets would be located in the stores or properties directories.
5. DDL files may be used to update the database, if necessary.
6. After that the files have been deployed, the WebSphere Application Server must stop and restart the WebSphere Commerce application. This can be an automated by using the wscp TCL program supplied with WebSphere Application Server.
7. For deployment of EJBs, the WebSphere Application Server XMLConfig utility can be used.

8. Finally, we would recommend using the batch compiler script to precompile any JSPs.

The complete build creation and deployment process can be performed through automated tasks, which are directed by scripts. The individual task scripts can be combined into a single larger script that can then be invoked automatically at build times.

Experience shows that build automation can save time, as well as reduce human error. Once the build is deployed successfully, the later build verification tests performed using testing tools (as described in Chapter 5, “Systematically testing a commerce site” on page 103) can also be automated.

### 4.3.8 Build failure

After the build has been successfully deployed, the project team should perform a Build Verification Test (BVT) on the resulting site to ensure the build is free from errors and that it has been applied correctly.

In most cases, the BVT will confirm the build status. However, in some projects, there will be occasions when the Build Verification Test fails. If a failure occurs, the build person must decide if a second build should be created immediately, or if the team should wait to create the build during the next build schedule.

The decision is likely to depend on many factors. If the build failed during the implementation phase, and the failure is not critical, for example, other components continue to function correctly, then it may not be necessary to fix the build and recreate it. However, if the build failed during a test phase, such as FVT, and the failure affects test cases being executed, then it maybe advisable to re-create a new build and redeploy it.

### 4.4 The problem determination process

From time to time, the project team is likely to encounter problems. They may arise during the implementation phase, but will mostly surface during test phases.

Because a typical commerce site will consist of many components, such as IBM WebSphere Commerce itself, WebSphere Commerce Payments, WebSphere MQ, a database, and LDAP directory services, any one or more of these components may fail or experience problems. The key to efficient problem
resolution is first isolating the component where the problem surfaced and then identifying the root cause.

The recommended approach for isolating the problem is searching the log files. Each commerce component will write to one or more log files.

**Note:** Each component may have to be configured for logging, and the component's documentation should explain how the component's parameters should be configured to start logging.

Most production systems are not likely to be configured for logging, because the task consumes valuable system resources and disk space. Normally, the default logging setting for production systems is to log only essential components. The logic is that the risk of missing the original cause of a problem must be balanced against the probability of a problem occurring, the resource demand, and the administrative cost of keeping excessive logging in continuous operation.

Problems can be categorized into a number of distinct groups. These are as follows:

- Installation or configuration
- Build deployment
- Runtime
- Application defects
- Performance
- Others

**Note:** Any errors or problems encountered should never be dismissed as unimportant. With complex systems, the interactions between one or more “broken” components can often result in apparently unconnected errors or symptoms that can become increasingly difficult to diagnose by the project team as it progresses through the development process.

### 4.4.1 Component installation and configuration

The installation and configuration of many components also presents opportunities for problems to surface. If a problem were to occur soon after a component is installed, then the project time should review the component's installation logs for clues, as all components are likely to have log files. In addition, the project team should confirm (preferably reconfirm) that all prerequisite software has been installed, is at the required level, and that the necessary patches and fixes have been applied.
Check the product documentation for any troubleshooting tips to help understand installation problems.

**Note:** Unless the installation instructions or prerequisites explicitly state that later versions of a product may be used, you must ensure that you install *exactly* the correct version number for each component. Incorrectly using a more recent or later version for a component may invalidate the system configuration.

### 4.4.2 Build deployment

Problems may be encountered during build deployment and could be due to incorrect builds or errors resulting from the deployment process. The process of locating, diagnosing, and resolving problems with build deployment should begin with check the WebSphere Application Server log files, particularly tracefile and the corresponding ecmsg-<date> and wcs.log files from WebSphere Commerce.

**Note:** Some problems may surface as Java errors, and the Java stack trace information is not always very informative. The preferred manner for assessing the problem is reviewing the trace to determine which component may have caused the problem.

### 4.4.3 Runtime problems

The causes of runtime problems are many. The preferred starting place for tracing a runtime problem in WebSphere Commerce is the ecmsg-<date> log files. The log files will most likely point to the location of the failed component.

**Note:** For some runtime problems, you may want to see what system functions were running when the error occurred. Using a `tail` command to monitor the data recorded in log files in real time may pinpoint for you the event that triggered the error condition.

### 4.4.4 Application or customization defects

Problems or defects with application code are best debugged and fixed by using the VAJ or WebSphere Studio Application Developer debugger tools. Extra debug flags may be added into the Java or JSP code to help narrow the location of the problem. Specific checkpoint messages may be output from the code and will show up in the log files. These messages will provide line-by-line sequence of execution, which should help the developer to find and fix the problem.
4.4.5 Performance problems

Even though log files are the best place to find and isolate problems, their usefulness in diagnosing performance issues is limited, because not all performance issues are caused by component errors. If faced with performance issues, the project team should check system performance monitors first.

Most components ship with performance monitors to aid in measuring and tuning their performance. As with logging, however, the function is usually turned off. The project team should turn on (if currently off) appropriate component performance monitors and begin assessing the output. Please note that the performance monitors may adversely impact the existing performance, so the project team should plan their use accordingly. As with system logging, each component's documentation should explain how to enable and disable its performance monitor.

4.5 WebSphere Commerce Subsystem Integration Test phase

The final activities of the implementation phase involve using an integrated coding, unit, and integration testing methodology. After completing code implementation activities, the project team must integrate all the use cases to ensure that they function and communicate correctly together. The use cases must also integrate with the WebSphere Commerce subsystems, such as Member, Order, and Catalog. The project should start formal site testing only when all use case and Subsystem Integration Tests have been finished.

Incorporating subsystem integration with the implementation phase is feasible; however, the project team is better served by keeping the activities separate. When the activities remain separate, the project team is capable of implementing code and then integrating it with greater efficiency than if performed through a combined approach. The main reason for the increased efficiency is that the project team can focus on related tasks, sequentially. The total gain will be affected by the structure and location of teams and subteams.

A checklist summary for the activities and work products of the Subsystem Integration Test phase is provided in Figure 4-6 on page 97.
Figure 4-6   The Subsystem Integration Test phase

The project team will most likely not want to use the entire set of use cases for testing during this phase, as a subset would be more appropriate. The team should assess the use cases to select as small a set that will allow sufficient integration testing between the use cases and the WebSphere Commerce subsystems. Please note that there would be little value gained from testing outside the Subsystem Integration Tests at this phase of the project, because to do so would only increase the complexity and number of tests that would otherwise need to be performed. All testing that is to be performed should be done on a common test server rather than in an integrated development environment running the WebSphere test environment, because it will ensure that the integration tests are performed within a realistic environment. Using test cases also prepares the team for the more complex and comprehensive formal testing activities that are to come under the site testing phase.

The team that worked on the previous implementation phase may now be split, if needed, into test and problem fixing subteams. Please note that at the start of this phase, the site code, as a whole, was not ready for formal testing. Considering this fact, the project manager must ensure that the project team completes all levels of WebSphere Commerce integration before its team members are reassigned to other tasks.
4.5.1 TTS use during Subsystem Integration Test phase

Project teams working on transition projects can also be divided into test and problem fixing subteams after the conclusion of all integration test activities. All the TTS tools used to create the transitioned code and data can now be made available to the problem fixing teams, because they will enable the team to re-create code assets or the migrated database, if needed. Table 4-3 summarizes the use of TTS during this phase.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS preconditions</td>
<td>TTS enhanced with customized schema information</td>
</tr>
<tr>
<td>Transition assets generated</td>
<td>Regenerate any of the code or data assets used in the transitioned site, if needed</td>
</tr>
</tbody>
</table>

4.6 Handling errors

Despite the integration testing, errors may continue to occur throughout the project and during normal operation of the site. For example, users may create errors after failing to supply a valid response in a form or on a Web page form.

Managing errors requires special attention to ensure prompt recovery. The project team will want to minimize adverse side effects, while providing as much detail to the team responsible for diagnosing and correcting the problem.

Each time an error or system exception occurs, the project team must assess the following six aspects of the problem:

- What will the user see, for example, a shopper using the site?
- What will the system administrators see when maintaining the site?
- What will the developers see, for example, when they develop, test, deploy, and troubleshoot the site?
- What will be the effect of the error or exception on the current transaction?
- What is the recovery strategy following the error or exception?
- After the error or exception, what state are the objects left in?

4.6.1 Handling each aspect in a real project

For a large project, IBM Software Services addressed the aspects of this particular project as follows.
The user aspect

As with most projects, the user interface was handled by JSPs. Each JSP was coded with error handling in mind.

There are three options for handling errors in this way:

- Each JSP can contain code to handle errors, where the code addresses all aspects of the error handling required.
- A JSP-specific error URL is registered with each JSP that determines what the user will see.
- A global error URL is registered for the whole store.

Note: This approach is normally considered a last resort, because the global error handling mechanism is unlikely to handle the transactional aspect properly.

The system administrator

WebSphere Commerce provides logging functions that support the issuing and logging of error messages. This function is fully documented in the product’s online documentation, as well as the WebSphere Commerce Programmer’s Guide, GC09-4951. Please note that the function is based on error codes, and it is enabled for multiple languages.

This facility was used to log the following:

- All exceptions and errors that occur
- All suspicious situations, such as failure to authenticate a user
- All severe problems

The developers

The project included development of a customized trace facility used to trace:

- All errors
- All warnings
- Default values used when no explicit parameter or value was specified
- Success or failure of key operations
- Empty result sets for a given key or search criteria
- Things not found or initialized
- General messages about the state of the code, for example, while debugging
All peculiar situations, where a developer would normally put a comment, such as:

/*this should never happen*/ or /* why are we here*/

**Transactions**

A utility class was included in the project that could be used to mark the current transaction for roll back, if this was what the error required.

**Recovery and object state**

Recovery, or the handling of an exception, means managing the effects of the exception to ensure they are corrected, and then restoring the normal flow of operation code.

Often one single component cannot deal with all the aspects of exception handling. Usually, it is a collaborative effort, where several components cooperate, with each of them handling different aspects of the exception.

**Note:** The WebSphere Commerce command framework allows the command writers to configure a command to be retried several times. This mechanism should be the base of a standard recovery strategy most of the time.

For example, on one project, a system resource could not be found. The component would try to recover fully from the situation. First, it attempted to use a default name for the resource if a specific name had not been passed. Second, it would try to use a series of different or default locations to find the resource. The component would attempt to find the resource in the system class path and then the current directory. The component did not simply give up the first time the problem occurred. The result was a significant improvement in the robustness of the system, producing a site that was much more stable and resistant to error.

Another interesting example was when the underlying infrastructure threw an exception that arose not from a failure to execute the request, but rather inadequate results from an otherwise successful execution. In such situations, entity EJBs throw an ObjectNotFound exception, and JDBC calls return an empty result set.

The problem is to decide what should be done if the data required to proceed does not exist. In practice, the business logic components should resolve the issue. Depending on the context, it may be acceptable to not have data, or it may be an unacceptable and exceptional situation. If it is acceptable, the business logic should adjust the flow of execution to the appropriate scenario. If it is unacceptable, an exception must still be raised, again by action from the business logic components.
Chapter 4. Preparing and implementing the commerce site

4.7 Summary

This chapter discussed implementing code for commerce sites. Special attention was paid to establishing an integrated development and test environment, because it is important to unit test site code as it is developed. Doing so enables the project team to uncover errors and correct them while code is still being developed. Automating the build process was highlighted to reinforce that fact that it can improve development efficiency by reducing time and decreasing errors. The discussion about problem determination emphasized the fact that problems should be resolved, whenever possible, as early in the development process as possible.

The chapter concluded with discussions about the importance of testing components as they are developed, as well as what the best practices are for developing code to correctly handle errors and exceptions from a production site.

In the next chapter, we discuss best practices for comprehensive site testing.

**Note:** Catching exceptions and doing nothing or simply logging the fact is not exception handling, it is exception masking. No component should mask exceptions, because it quickly becomes impossible to detect and resolve problems. Always recover from the exception, or handle as many aspects as possible and throw the remaining exception aspects again.
Chapter 5. Systematically testing a commerce site

All the testing completed prior to the site testing phases described in this chapter have focused on integration of the site code at one of three levels: unit or component, use case, and WebSphere Commerce subsystem. The emphasis so far has been on ensuring that each of the customized components operate correctly within themselves.

We describe a series of test phases that may be used systematically to test a newly created site or a transitioned one prior to it being “cutover” to operational status. The emphasis of the testing is to ensure the customized site components interact correctly with each other, as well as with all relevant out-of-the-box components.
5.1 Site testing phases

Before entering the overall site testing phases, individual components should have been tested using Build and Verification Tests (BVT), as described in 4.4, “The problem determination process” on page 93. Once validated by BVT, acceptable components may be assembled and integrated and then tested as an integrated unit. The test phases discussed in this chapter are as follows:

1. Functional Verification Test (FVT)
2. System Integration Test (SIT)
3. System Verification Test (SVT)
4. User Acceptance Test (UAT)

As the topic implies, the site testing phases add value by ensuring that all code is tested at the site level before the site itself becomes fully operational. Unless otherwise indicated, all stress and performance tests should be performed on the actual site code.

Each phase focuses on a different aspect of testing the commerce site. Through experience, IBM Software Services has realized that the systematic approach outlined here helped streamline site testing during engagements, and as a result, improved their ability to deliver operational site code faster with a higher degree of reliability.

**Note:** Throughout the testing phases, it is essential that all test results and responses are recorded, along with all actions taken. This includes progress reporting, test pass or fail results, and problem determination procedures to assist with later problem resolution or troubleshooting.

An overview of the site testing phases is shown in Figure 5-1 on page 105.
5.1.1 Creating the site testing infrastructure and processes

Irrespective of who will perform the actual testing during the recommended phases, the project team must ensure that an adequate test environment and appropriate test processes have been established and configured. Without a suitable test environment, namely one that reasonably resembles the target deployment system, all results produced using the test systems will be marginal at best, because they will not have been tested to operational standards. In addition, if test processes are omitted or must be developed at the last minute, the project will require time to validate test results, a resource demand that will surely extend the entire test process.

During engagements, IBM Software Services has identified the following activities as very helpful for configuring an adequate test environment and creating appropriate test processes:

- Identify a test lead
  
  Both the customer project manager and project team manager should identify and nominate test leads. The customer and project test leads will assume responsibility for all test activity preparation and act as the liaison between the teams.
Prepare a simulated production environment for testing

This activity involves implementing a WebSphere Commerce test environment that has “production ready” characteristics, including simulations of all required back-end connections. As appropriate, the project team may adapt the test environment used during the implementation phase to jump-start site-level testing, such as FVT. The customer IT team should assume the responsibility for implementing the simulated test environment, because they will be best suited to ensure the system reasonably resembles the target production infrastructure.

Prepare more than one test server

At a minimum, the customer IT team should configure two test server systems. Having multiple test servers is especially beneficial when teams, such as a test subteam and a problem fixing team, share servers, because the various test servers can support simultaneous testing of regular builds. For example, one server could be used exclusively by the problem fixing team to correct builds, while another server would support the test team’s continued review of new builds. Configuring multiple servers will enable the various test teams to work in parallel assigned tasks to reduce the total time needed to test, correct, and validate system code.

Create testing, problem fixing, tracking, and reporting processes and mechanisms

Establishing appropriate test processes may involve the use of off-the-shelf testing software or local tools and processes. The customer IT team is encouraged to explore all testing strategies, but they should base their decisions on the level of experience they have with available tools and processes. Selecting a strategy that maps experience with available tools and processes produces the best results in terms of time, effort, and investment. If testing tools are to be used during the site test phases, their acquisition and implementation must occur prior to the start of the site test. This responsibility rightfully belongs to the person acting as test lead.

At a minimum, any test process should demonstrate the following characteristics:

a. Test cases are assigned to a test team.
b. Test cases are executed by the test team.
c. Any problems or bugs are documented.
d. The documented results are sent to the test lead.
e. The test lead categorizes results according to criteria, such as component at fault and the severity of the problem.
f. For each problem, the test lead assigns a developer to resolve the issue.
g. The developer corrects the problem.

h. The developer updates the test lead, advising that the problem has been corrected.

i. The test lead should normally schedule a re-run of the test. There may be other component dependencies that must be met before a re-run can be planned.

j. A test phase can be considered complete only when all test cases have been successfully executed and no problems have been found.

**Note:** The testing may uncover problems with code: defects. Information about these defects should be recorded to provide useful metrics that help assess and improve development and testing processes. Useful metrics include:

- Severity of defect
- Average time to find defect
- Method for correcting each defect
- Actual time spent correcting each defect
- Average time to fix defect
- Number of errors caused by fix for original defect

Automate the BVT process

Automating BVT enables test teams to streamline the validation of new builds while minimizing idle time related to waiting for test results. Off-the-shelf products can aid test teams to automate this phase, and they should be investigated to determine if they can be used during a project.

Review BVT test cases

The BVT test cases should be reviewed to ensure that all functionally critical paths are executed and tested.

**Note:** A useful guideline is that the execution of a BVT should not exceed 30 minutes. If it does, the BVT test case should be assessed to determine how it may be refined, subdivided, or updated to improve test efficiency.

5.1.2 Executing test cases

The two testing strategies that should be considered for a WebSphere Commerce project are the single-pass test strategy and the multi-pass test strategy.
An overview of the first test strategy, single-pass testing, is provided in Figure 5-2.

The single-pass test strategy involves making only one pass through each test case. If a test case executes successfully, it will be marked complete. If, on the other hand, a test case fails either completely or partially, all remaining tests are halted and the problems or errors found get reported to the problem fixing team for immediate review and action.

Once the problem fixing team resolves a reported problem, the test case may be re-evaluated. The sequence of test → report → correct → retest should continue until the questionable test case executes without issues. Only then should it be passed as complete.

This testing strategy works well when there is limited interaction between use cases and test cases.

The test lead must review all test results so that they may ascertain the cause of problems. For example, a report may indicate that 10% of the test case was successful, but 90% of the test case could not be executed because of issues identified as one or more open problems. Being aware of all testing issues and
progress, the test lead will then be able to report accurately the current testing status to the project manager.

An overview of the second test strategy, multi-pass testing, is provided in Figure 5-3.

**Multi-Pass Testing Strategy**

![Diagram of Multi-Pass Testing Strategy]

**Key Influencers:**
- Substantial interaction between use cases and test cases
- Time available to complete test phases
- Testing resources are available to oversee the tests

**Input Work Products**
- Depends on specific test phase being performed

**Output Work Products**
- Depends on specific test phase being performed

**Task:**
- Execute test cases repeatedly.

**Objective:**
- Repetitively test code to ensure it is free from error.

**Project Type:** Either New or Transition

**Project Phases:** Testing Phase

Figure 5-3  Multi-pass testing strategy

The multi-pass strategy involves making multiple passes over the suite of test cases. Normally, at least two passes will be performed. The first pass should follow the same process as that detailed for the single-pass strategy in Figure 5-2 on page 108.

Once the first pass has been run, the test lead must select a subset or the complete set of test cases for the test team to repeat the testing process from start to finish. The second pass, or subsequent ones, should take less time to complete than the first, because any issues found will have been fixed prior to the start of the subsequent passes.

This testing strategy works well when there is a considerable interrelationship between use cases and test cases.
5.2 Functional Verification Test (FVT) phase

The main objective of the FVT test phase is to function test the commerce site code from end to end, using a suite of test cases. The project team generally performs FVT activities before passing the code to the customer IT team for further testing. This phase is especially important when the project team is based at a different location than the customer test site or using a different environment than the one used during SIT, System, and UAT phases. If the project team is indeed at a different location, the FVT test environment should mimic, as closely as possible, the production environment so that any back-end integration issues may be resolved as early as possible.

If, on the other hand, the project team is based at the customer location and has access to the simulated production environment, the FVT and SVT phase, described in 5.4, “System Verification Test (SVT) phase” on page 115, can be combined.

If FVT and SVT activities are combined, we recommend using the multi-pass testing strategy described in Figure 5-3 on page 109 to ensure that all aspects of the site code get fully tested and are shown to be free of bugs or other issues.

An overview of the FVT activities and work products is provided in Figure 5-4.

---

**Functional Verification Test (FVT) Phase**

**Input Work Products**
- Integration test phase outputs
- Stress test scripts

**Task:** Systematically function test code, using end-to-end test cases.

**Objective:** Function test site code to ensure function is desirable.

**Output Work Products**
- Function tested code that executes in a simulated production environment

---

*Figure 5-4 The Functional Verification Test (FVT) phase*
It is important to reassess the activities listed in 5.1.1, “Creating the site testing infrastructure and processes” on page 105. Many of the activities should be completed prior to beginning FVT.

Another important task that should be resolved during this phase is the project manager securing a high-speed network connection to any back-end systems. This type of connection would eliminate the need for a simulated back-end system, and more importantly, it would enable the test teams to test performance when communicating with the back-end systems. The project team would be capable of resolving any communication issues prior to the customer IT team initiating the SIT, System, and UAT tests.

5.2.1 Team organization during testing

Testing and problem fixing may either be performed by separate teams or by a single team, preferably the one that participated in the implementation phase of the project. There are two possible strategies to consider to determine how best to organize your test teams.

In the first strategy, the original implementation team is divided into testing and problem fixing subteams. The test lead can then assign test cases to the teams. The main advantage of this strategy is that it is not necessary to add new resources to the project; each test member was originally a developer on the project and, therefore, will possess a deeper understanding of the code and will be better able to test or fix it. However, for this strategy to be valid, it is essential that test cases are assigned to individuals who did not develop the functionality being tested.

In the second strategy, a completely separate and independent set of testers is identified and assigned to perform the FVT. This group has the sole mandate of performing the FVT. The advantages of this strategy are a quick release of the original development team and a clear and fresh perspective on the code being tested. Typically, problem fixing takes less effort than the original implementation, even allowing for the fact that the new testing team may not be familiar with the original design and development work. However, this approach is only possible if the previous phases were executed successfully.

Irrespective of which strategy is adopted, the FVT problem fixing and testing subteams should collectively decide and agree on a build frequency that supports the objective of completing FVT testing within the project schedule. Before new builds are put on the test servers, code should be run through the Build Verification Test (BVT) process, typically using automated scripts, to ensure no issues exist.
FVT progress should be reviewed regularly, preferably on a daily basis. The review would normally involve:

- Monitoring a predefined percentage of test cases to see how many have been completed
- Tracking the number of remaining open problems reported

The FVT team members who performed the test cases and fixed the reported problems should be responsible for reporting test results to the test lead.

The FVT phase should also include executing stress test scripts once the code approaches a stable state. Executing stress test scripts should continue throughout FVT to ensure that problems, such as memory leaks, are caught and fixed before the FVT completes. Problems discovered during stress testing need to be reported to the development or problem fixing team, because these issues will need to be fixed and retested during FVT. Tools to automate the running of scripts should be considered, because they will free resources to concentrate on other test or repair activities. If tools are used, the system administrator will assume responsibility for implementation.

### 5.3 System Integration Test (SIT) phase

The customer IT test team needs to plan, prepare, and execute System Tests and UAT using a simulated production environment. The closer the test environment is to the actual or planned production system, the easier it will be for the team to launch the site after testing.

The SIT phase is a preparatory phase that determines if the customer test environment is ready to support System Tests and UAT. An overview of the activities and work products of the System Integration Test (SIT) phase is provided in Figure 5-5 on page 113.
Unlike previous test phases, this phase should be conducted by the customer IT team. It is important to make decisions up front as to the scope of testing and the nature of test infrastructure used for the System Tests and UAT. Ideally, the infrastructure should map as closely as possible to the planned production environment. If achieving this close approximation is not feasible, careful planning will be required to ensure that the System Tests and UAT will identify all issues with regard to the production environment.

The customer IT team needs to consider a number of aspects when preparing a test environment. It is essential that these aspects are addressed prior to commencing the System Verification Test, and it is advisable to begin these tasks while the FVT is underway. In many cases, intermediate code that has not yet gone through complete FVT can still be used to support the following main tasks:

1. Select and implement an appropriate hardware-software configuration for the test environment.

   You may want to use either a single-tier or multitier hardware configuration that supports either horizontal or vertical cloning through software configuration. No matter how you configure the test environment, you must ensure the final configuration is suitable for System Tests and UAT. Your system configuration decisions should be based upon capacity planning requirements.
If replicating the production environment proves to be infeasible, you should still refer to capacity planning information when configuring your test environment, because it will enable you to develop production style scenarios.

2. Install WebSphere Commerce and the associated software stack on the selected hardware and software configuration.

If a cloning architecture has been selected for the production environment, ensure that the corresponding test environment has the required setup.

3. Configure WebSphere Commerce and the other software components, such as WebSphere MQ servers.

The test environment should be configured accordingly with appropriate software that will enable it to communicate properly with back-end and legacy systems. If the customer IT infrastructure is maintained by multiple teams, the test leads must work with all the teams to ensure the appropriate infrastructure gets configured for test.

4. Apply and test code developed by the project team.

This task is necessary to ensure that all back-end communication through any customized code works according to the design. For example, messages from WebSphere Commerce must travel successfully to back-end systems, and responses must be properly picked up by the project specific code.

This task is an important milestone for both the project and customer IT teams, because it determines when SVT should begin. Completing it may take time, especially when different customer IT teams own different parts of the communication infrastructure, so it is advisable to carry out some advanced planning.

In order to successfully achieve this part of SIT, it will be necessary to understand the code deployment process for the test servers. For BVT activities, the project team can transfer the knowledge, and in some instances, share BVT scripts with the customer IT team to automate this task. For code deployment, where possible, the project team should automate this task through the use of scripting language, such as Perl.

5. Use a subset of test cases that supports test drivers.

This process is sometimes referred to as a “happy-path” test case. Happy-path testing should focus on testing the integration points between the commerce system and any back-end or legacy systems that the customer test environment connects to.
6. Report all problems identified in this phase back to the project team.

The customer IT test team should establish simple processes for the following:

– Tracking of problems discovered during SIT
– Reporting the problems to the project team
– Retesting fixes

This process of testing happy-paths and reporting problems should be repeated as often as necessary until such time as the test environment has been validated as acceptable.

5.4 System Verification Test (SVT) phase

The System Verification Test phase is typically conducted by the customer IT team, and their objective is to validate the functionality of site code received from the project team. All system tests should be performed in the simulated environment. An overview of the activities and work products of the System Verification Test phase is provided in Figure 5-6.
The customer’s IT team should undertake more testing responsibilities during the System Verification Test, while the development teams continue to focus resources on problem resolution. The customer and development test leads should employ agreed upon processes as defined in their joint test plan. Mechanisms for reporting and fixing problems should be well established and used throughout SVT.

To validate the appropriate SVT infrastructure is in place, the customer IT team is encouraged to review the activities listed in 5.3, “System Integration Test (SIT) phase” on page 112. Reviewing this section will also enable the customer IT team to validate the testing strategy.

It is during the SVT phase that the site code first communicates with external systems. These systems may either be live or test ones, and they may consist of back-end systems, or specific ones, such as tax systems, data load, or feed programs. To validate the test environment, the customer IT team should conduct appropriate stress and performance tests, because the closer the environment can be made to approximate the target production system, the more reliable and useful the test results will be. The project team can share its stress test scripts with the customer IT test teams if both teams use the same tools to measure site stress test results. Capacity plan documentation can be an effective tool for assessing performance potential.

Any problems or unsatisfied results uncovered during either stress or performance testing should be discussed with and worked on by the project team. In practice, it is often the case that custom code developed by the project team is rarely the sole cause for a target site’s performance issues. A variety of actions must be taken to ensure performance targets are achieved, such as tuning the test hardware and software.

A successful SVT is likely to include not only successful function tests, but successful stress and performance tests as well. However, performance targets can be less than desired for the live site if the test environment failed to mirror the production environment.

### 5.5 User Acceptance Test (UAT) phase

The UAT phase focuses on the final site look-and-feel, the site flows and behaviors, and overall business process validation. The customer’s business team generally is responsible for validating UAT tasks. An overview of the activities and work products of the User Acceptance Test (UAT) phase is provided in Figure 5-7 on page 117.
Chapter 5. Systematically testing a commerce site

Figure 5-7  The User Acceptance Test (UAT) phase

The UAT phase is conducted on the same test environment that was created for the SVT phase. The project team in this phase can provide the same support as it did during the SVT phase, specifically problem resolution. If all of the previous test phases were successful, then UAT is likely to be simple, because the business team will only have to validate the look-and-feel and business operations of the target site.

The customer IT team can act as the communications link between the customer and the project team, through which problems can be reported, tracked, and when fixed, redeployed.

If the customer’s IT team has been given the responsibility of testing the look-and-feel and business functionality of the site, then the SVT and UAT phases may be combined. If SVT and UAT are combined, we recommend the test objectives be fulfilled prior to site launch.
5.6 Summary

In this chapter, we identified and described the major testing phases for a commerce site. We discussed creating and applying a test infrastructure, as well as integrating it with back-end and legacy systems.

A key point throughout the chapter was that System Tests and UAT extend testing beyond the component tests conducted during the implementation phase. Also, the customer teams play a significant role in leading and conducting the tests, and as a result, they are properly positioned to assess more effectively what gets delivered against that which was proposed and required.

After successful conclusion of system testing, the customer will be ready to implement the proven, but non-operational code. The task of setting up the site to go live is discussed in the next chapter.
Preparing to go live

Up until this point, the design, implementation, and testing activities for the new commerce site take place within a clean and controlled environment. Each activity is planned and prepared in advance. Each component is used either in isolation from other components, or as part of a very controlled interaction.

In a live site, each of the components will be subjected to real user tasks. Real data is used to support the site functionality, and all transactional data and status information becomes mission critical. It is only when the actual component code is deployed onto the production server systems that real performance can be observed and fine-tuned.

In most transition projects, the new site will run in parallel with the old site to ensure a successful cross-over of functionality. This may introduce complexities of duplicate data sets that must be carefully managed and later synchronized.

In this chapter, we describe the final activities involved in preparing a production site for launch.
6.1 Site deployment and launch phase

All of the project phases so far have focused on preparation of a commerce site that is fully functional and well tested. However, there are still certain activities, some of which started during the design phase described in Chapter 3, “Designing the commerce site” on page 31, that must be completed before the production site can be made available to end users.

![Diagram of Site deployment and launch phase overview](image)

*Figure 6-1  Site deployment and launch phase overview*

The main objective of this site deployment and launch phase (as shown in Figure 6-1) is to ensure that the production environment can support the new or transitioned code, with the expected capacity and performance.

An overview of the activities and work products of the site deployment and launch phase is provided in Figure 6-2 on page 121.
If the production environment is separate from the SVT and UAT environments, then all the steps listed in the SIT phase in Figure 5-5 on page 113 need to be repeated to prepare the hardware and software environment for launch. These activities include:

1. Setting up hardware
2. Installing software
3. Configuring the hardware and software
4. Connecting the resulting production-ready system to the back-end or legacy systems, as needed
5. Applying UAT validated code builds

A capacity planning activity should be performed that reviews site traffic and load estimations and confirms that the hardware and software architecture for the production environment is sufficient to handle the anticipated capacity. The capacity planning activity may start as early as the design phase described in Chapter 3, “Designing the commerce site” on page 31.

Similarly, site infrastructure activity, as well as testing of new or transitioned code, should be finished and refined, as needed, prior to site launch to ensure that targeted capacity requirements can be achieved at launch.
Some basic site hardware and software tuning may be required to improve site performance. The tuning may take place just prior to going live once all components are installed and operational within the production environment. Subsequent tuning may also be required as performance is observed and assessed over time.

System tuning may affect the database, WebSphere Application Server, or WebSphere Commerce itself. It may be helpful to run stress and performance scripts to observe actual performance, rather than to look for problems. Doing so will enable the team to gauge what tuning may be required. Any tuning steps performed during the earlier SVT and UAT phases should be repeated on the production environment to confirm the validity of the results and any changes made.

Performance and stress tests conducted during SVT provide a means for comparing the actual production configuration of software and hardware and the customized code against desired capacity targets. If after conducting performance and stress tests, and tuning the code, the production site fails to perform properly, the customer IT team should consider the following actions prior to launch:

- **Tuning the site, focusing on the software stack or the hardware**
  The tuning might include making changes to the lower-level software components, that is, those components closer to the operating system level than to the WebSphere Commerce level. For example, the team might modify buffer size, caching configuration parameters, and network connectivity time-out values.

- **Making changes to the production architecture of software or hardware**
  In extreme cases, the team may need to upgrade the architectural components of the production system. For example, the team might increase memory within the servers, or add faster processors.

This last option should be pursued only if all prior actions fail to produce desired improvements. Please note that proper capacity planning early in the project should minimize the likelihood of ever having to pursue this last option.

For existing sites that are being transitioned, down time is a probable characteristic of a production launch. In order to assess the expected down time, the customer IT team should conduct partial or full mock launches. This will enable the team to identify all the steps required to bring up a new site.

After this data is available, it is sensible to compare it with the forecasted down time that the business can accommodate. This assessment will aid the customer teams in determining appropriate actions for reducing the launch window, which may be measured in hours, to a manageable limit.
For transition projects, another concern is the possibility of overrun in the down time while a new site is brought online. It is advisable for the team to consider and plan for the possibility of launch failures or other unexpected issues that may arise during the launch window. The following considerations will enable the customer team to assess launch contingencies:

- When the next launch is scheduled
- How the next launch is scheduled
- How to back up and restore the old site until the newly transitioned site can be made error free

Contingency planning will lessen the impact associated with failures in hardware, software, or networking components of the system.

Decisions made during this phase of a project depend on many factors, such as site complexity, availability, and the number and type of business transactions. Because many of the decisions affect more than just the site infrastructure, the customer IT teams need to play a lead role in overseeing and directing site launch preparations. Intimate knowledge of operational details and practical experience will enable customer IT teams to prepare the commerce site efficiently for operational availability. The development teams should transfer site knowledge to the customer IT teams to prepare each for post-launch activities, such as site maintenance.

The project team can also use this phase to transfer site knowledge to the customer IT support teams so that they can support the maintenance of the site after launch.

### 6.2 Summary

This Redpaper introduced a project development methodology that was specifically adapted from industry standards for the creation of e-commerce sites using the IBM WebSphere Commerce family of products. Given that a typical development methodology consists of six phases, we made a conscious effort to focus only on the following four core development phases:

1. Design
2. Implementation
3. Site testing
4. Deployment/launch
The reason is simple: When developing e-commerce sites using IBM WebSphere Commerce, it is critical for the project manager, as well as the team, to understand:

- The key input work products for each phase
- The core tasks that must be accomplished to complete each phase
- The skill requirements needed to support the tasks
- The output work products to deliver during the phases

The project development methodology outlines a structured approach that enables teams to build on the work products produced during the preceding phase. This “incremental” approach significantly reduces project risk, because it enables project teams to identify design, code, and integration issues in a timely manner throughout the project's life cycle, so if problems arise, each may be corrected with minimal interruption to the project's schedule.

As specified in this Redpaper, the design phase marks the beginning of a typical WebSphere Commerce project, and it consists of three distinct subphases:

1. Solution outline
2. Macro design
3. Micro design

The solution outline provides input to macro design subphase, which in turn, provides input to the micro design subphase. Following this incremental process, the project team will identify skill requirements, outline a project plan, establish a project team, and then define the project's functional requirements and document the detailed design decisions that are required to implement the target site. Actual execution of the phases will depend largely on the intricacies of the site, and the project team is free to adapt and employ any strategy, so long as the strategy used is valid and consistent with the concepts presented in this Redpaper.

Completion of all design activities enables the project team to initiate the implementation phase. Ideally, the project team should pursue the continuous and incremental development and code integration testing strategy during this phase. This strategy advocates anchoring development on use cases and then testing code prior to it being integrated into builds. In essence, the project team codes site components, tests each, and then integrates the tested components into builds. Because the project team will be managing the integration of builds, the project manager will want to ensure the team employs an appropriate build verification test process, as well as an integration test strategy. The project manager should identify key people on the team to oversee the efforts, as well as to establish schedules, assign responsibilities, and create failure and
error-handling processes. After the site components have been fully integrated, the team may start site testing.

During site testing, the project team should employ a systematic test strategy. The objective of the test strategy is to ensure the site components interact correctly and that each works well with out-of-the-box WebSphere Commerce components. The key site testing phases are as follows:

1. Functional Verification Test (FVT)
2. System Integration Test (SIT)
3. System Verification Test (SVT)
4. User Acceptance Test (UAT)

Employing a systematic test strategy streamlines the site testing process, which results in the project team being able to deliver reliable operational site code more quickly. The project team should create a suitable test infrastructure, one that resembles the production system closely, and define test processes to support the test strategy properly. Doing so will ensure the test results are meaningful and that they can be used to improve code reliability. How the project team is structured to support testing depends on the project's requirements and availability of skills, but the project manager is encouraged to structure resources so that there is a clear division of responsibility between test resources and problem fixing resources.

The FVT phase requires the test team to function test site code from end-to-end using a suite of test cases. SIT is an interim phase that requires the test team to assess the test systems to ensure each can support SVT and UAT. During SVT, which the customer test team should undertake, the site code is validated to ensure it operates as designed. Doing so prepares the customer team to conduct UAT, which is the testing of site look-and-feel and behavior. The key reason for employing a systematic test strategy is to ensure all code operates as designed, and if any bugs are found, that they can be passed to the problem fixing team for prompt resolution.

The final project phase, as outlined in this Redpaper, is deployment/launch. During this phase, the project team must assist the customer to assess and ready the production environment to support the new or newly transitioned site code. The key activities include:

- Hardware preparation
- Prerequisite software preparation
- Hardware and software configuration
- Code implementation
In addition to the standard deployment activities listed, the project team should also encourage the customer team to plan contingency activities to mitigate potential issues or failures that may arise after launch. Contingency planning is prudent, and it can be as simple as identifying how the old site can and should be restored if the new site fails.

To conclude, the purpose of this Redpaper is to present a viable development methodology that a WebSphere Commerce development team, whether comprised of IBM consultants, third-party consultants, or customer team members, can use to create or transition WebSphere Commerce sites. Contained within the body of this paper are numerous recommendations, examples, and notes to help the reader understand and then employ the methodology and relevant strategies. In addition, it provides extensive coverage on how tools, namely the Transition Tool Suite, can be used to transition specific site assets, and in some cases, even how it may aid in the deployment of new ones. When used properly along with a systematic development methodology, tools can significantly improve the efficiency of a project team to create or transition WebSphere Commerce assets reliably. The appendices that follow provide additional insight into related topics that will further aid the reader's use of the methodology, as well as help to position it in comparison to industry standards.
Appendix A. Project team creation and common roles

There are a number of the common roles that are needed within a team to execute a WebSphere Commerce project. In this appendix, we outline the process of creating a team and list and briefly describe the key roles. This information can be used as a guide to assess what roles and responsibilities make sense and are appropriate for a given project context.

It is possible for one or more of these roles to be performed by a single person during a project.

Some of the roles will be performed over the complete duration of the project, while others will only apply to certain phases of the project development methodology.

Having identified the roles, we then show how these roles relate to the Transition Tool Suite, indicating which tool would be used at a particular point in the project by specific role performers.
Creating the project team

Although apparently obvious, note that all projects require a team of dedicated, and more importantly, skilled people to perform the activities that will ensure the project's success. A project team should be formed by the project manager. When selecting available people and assigning them to roles, the project manager must consider such factors as:

- Skills and expertise
- Project objectives
- Time scale for deliverables
- Customer requirements for new or replacement site functionality
- Geographic location for overlapping projects and sub- or specialist teams
- Customer intention to utilize their own staff for development or support
- Training
- Multicultural support in the site
- Internationalization and translation
- Double-byte character set (DBCS) enabling

**Note:** A project team may consist of both customer and consulting resources, which is generally the case when a project requires skills transfer between the consulting firm and the customer, or the customer has requested that it should have the necessary skills required to administer the site after the project has ended. It is important to realize that time or resources spent supporting or training customer personnel should be included as a separate part of project development estimates.

An ideal commerce project, specifically one based on IBM WebSphere Commerce, requires a set of common roles for team members. These required, or more appropriately, recommended roles are described in detail in “Project roles” on page 132.

Identifying team members and then assigning each to a role, or possibly roles, can be performed many different ways. IBM Software Services has found two strategies for forming project teams, in particular, to be helpful during project design. The strategies are titled the core competency strategy and the holistic strategy, and each is discussed in detail here.
The core competency strategy

This strategy assumes that several of the team members have the relevant skills required by the project, and it enables the project manager to divide the roles and responsibilities according to the existing skills or competencies.

For example, if some members of the team are skilled, or have been certified as experts on specific WebSphere Commerce subsystems, such as Order, Member, or Catalog systems, then the project manager should assign design work involving these subsystems to the available “experts.”

An overview of the core competency strategy is provided in Figure A-1.

![Core Competency Team Assignment Strategy](image)

The core competency strategy fosters a team structure that is based on technology assignments, which in turn, afford each team member the opportunity to hone and refine their skills. At the conclusion of the project, each team member is likely to be better prepared to assume more challenging design and development responsibilities on future engagements, such as commerce projects. In addition, project models based on this strategy generally produce greater specialization and expertise in specific commerce areas within a services organization, skills the organization can call upon and sell later.
The holistic strategy

The holistic strategy is based on the requirement of making each team member equally effective in performing all project tasks, hence the reason for its name.

An overview of the holistic strategy is provided in Figure A-2.

![Holistic Team Assignment Strategy](image)

The holistic strategy is very useful when creating a workforce that can adapt to a variety of projects or work assignments. This strategy enables team members to focus on developing a breadth of commerce knowledge prior to refining it in given competencies. For example, when this strategy is employed, the project manager can expect and count on any team member being capable of design, coding, and testing most project artifacts, such as WebSphere Commerce commands or JSPs. To succeed using this strategy, the project manager must ensure the team has, or is in the process of obtaining, skills and experience with all core Java technologies, such as JSPs, commands, and EJBs, as well as ensuring that the team understands and has experience with core project activities, such as site design, implementation, and testing.

Either strategy, or the combination of elements of each, may work for a particular project. Prior to selecting a team strategy, you are encouraged to assess the
nature and scope of your project early to help you determine how best to form your project team and manage them through a specific strategy. Even after having made a decision, you will want to reassess your team strategy between project phases, because you may need to pursue a new strategy for any given phase.

**Note:** Whenever practical, we recommend not making significant changes to your project team during the design phase. Each new member added to the team will require appropriate lead time for them to establish a bearing with regard to the project’s current course and direction. Although the resulting delay is rarely convenient, it is absolutely necessary if the new members are to be fully integrated into the team to become productive members.

We recommend against assigning team members to multiple projects, unless, of course, the transition between projects can be managed adequately so as to not delay any of the projects. Because people require time to “context-switch” between projects, and the length of time needed is directly proportional to project complexity, reducing people movement into and out of a project team during the design phase will help ensure an efficient and productive design phase.

In practice, resource exclusivity is a rarity on most projects. Therefore, project managers should comprehensively document all design tasks and work assignments and then post the documentation so that all team members may access it. Whenever possible, the project manager should use standard document templates and layouts to enable new and old team members to assimilate the information quickly. Combined, these activities will enable the project manager to mitigate some of the problems or issues that may arise from people movement during the design phase.

**Guidance on selecting the strategy**

Selecting the appropriate strategy for a project will necessarily reflect the circumstances of the project itself. However, a useful guide is to assess the skill level of team members and their previous experience with commerce projects.

If less than 20% of the team have previous commerce experience, it is advisable to build core commerce skills before commencing the project. While it may appear attractive to “build skills on the job,” this is likely to introduce significant scope for error in estimating the time to complete phases and may impact the project delivery date.
If less than 40% to 50% of the team have previous commerce experience, this would indicate that the core competency strategy would be appropriate for the project.

If more than 40% or 50% of the team have previous commerce experience, this would indicate that the holistic strategy would be appropriate for the project.

**Note:** The percentages suggested here are for guidance purposes only; each project should be assessed individually.

### Project roles

The most common WebSphere Commerce project roles, and the skills necessary to perform the roles, are summarized in Table A-1.

<table>
<thead>
<tr>
<th>Project role</th>
<th>Main phases</th>
<th>Product related skills required</th>
<th>Other skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project manager/lead</td>
<td>All</td>
<td>General understanding of WebSphere Commerce product and architecture</td>
<td>Project management, preferably in the e-commerce area</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Manages and leads project team along all phases of the project and also acts as a contact point to interact with customer IT and test teams.</td>
</tr>
<tr>
<td>Technical lead/architect</td>
<td>All</td>
<td>WebSphere Commerce architecture and design skills</td>
<td>Preferably, having experience in technically leading a project team during all phases of a project</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Looks after the overall project technical architecture/design, quality assurance of the solution, knowledge transfer to customer, and mentoring to the project technical team members.</td>
</tr>
<tr>
<td>Project role</td>
<td>Main phases</td>
<td>Product related skills required</td>
<td>Other skills</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WebSphere Commerce designer</td>
<td>Design</td>
<td>WebSphere Commerce design skills</td>
<td>Technical design experience; OOD and implementation experience; knowledge of Java technology; knowledge of the development environment and tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Based upon project scope and complexity, one or more WebSphere Commerce designers can work to create detailed project technical designs. The work effort can be divided based on common WebSphere Commerce subsystems, such as Catalog, Member, Order, and communication with back-end or legacy systems.</td>
<td></td>
</tr>
<tr>
<td>Database creation or migration expert</td>
<td>Design</td>
<td>Understanding of WebSphere Commerce schema, and database creation or migration knowledge</td>
<td>Database knowledge, such as DB2 and Oracle; SQL knowledge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The main database creation or migration effort should be complete by the end of design phase; it is assumed that the data model for the new or transitioned site is available to the design team of the WebSphere Commerce architect and designers.</td>
<td></td>
</tr>
<tr>
<td>WebSphere Commerce programmer</td>
<td>Implementation and site test</td>
<td>WebSphere Commerce programming model</td>
<td>J2EE programming, knowledge of the development environment and tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WebSphere Commerce architect/technical lead and WebSphere Commerce designers can also participate in implementing the new or transitioned site.</td>
<td></td>
</tr>
<tr>
<td>System administrator</td>
<td>Design, implementation, site test, launch, and deployment</td>
<td>WebSphere Commerce administration</td>
<td>Team repository administration; setting and working with live or simulated back-end systems, for example, using MQ for connectivity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Responsible for all types of project software and hardware infrastructure setup and maintenance work.</td>
<td></td>
</tr>
</tbody>
</table>
## Skills map to use Transition Tool Suite tools

The Transition Tools Suite provides a rich set of functionality to assist with transitioning an existing commerce site. However, use of the various tools is ideally performed by experts with appropriate skills.

In Table A-2 on page 135, we provide a cross reference between project roles and TTS skills. The purpose is to view the skills in the context of a full transition project using the specified transition methodology. It should be noted that one or more tools could be applied by a single team member.

<table>
<thead>
<tr>
<th>Project role</th>
<th>Main phases</th>
<th>Product related skills required</th>
<th>Other skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test lead</td>
<td>All</td>
<td>General understanding of WebSphere Commerce product and architecture</td>
<td>Detailed understanding of functioning of customer site; experience in leading testing activity; test tool skills</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lead the test team during all site test phases; work with customer IT team to support SIT, SVT, UAT, and launch preparation phases.</td>
<td></td>
</tr>
<tr>
<td>Testers</td>
<td>FVT</td>
<td>General understanding of Web-based systems and customer site information and behavior</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any members of the project team, who can satisfy the criteria may participate in this activity</td>
<td></td>
</tr>
<tr>
<td>DBA</td>
<td>Design, implementation, site test, launch, and deployment</td>
<td>Some knowledge of WebSphere Commerce administration</td>
<td>Database administration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The database creation or migration activity role could also be performed by the DBA.</td>
<td></td>
</tr>
</tbody>
</table>

---

Best Practices and Tools for Creating IBM WebSphere Commerce Sites
### Table A-2: Skills map for using TTS

<table>
<thead>
<tr>
<th>Transition function</th>
<th>Tools offered</th>
<th>Project role</th>
<th>Skills required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessor Tools</td>
<td>TTS Schema Crawler/Mapper</td>
<td>Technical lead/architect, WebSphere Commerce designer, programmer, or system administrator</td>
<td>Perl scripts and spreadsheet programs</td>
</tr>
</tbody>
</table>

Use of the Preprocessor Tools assumes that a mapping spreadsheet for the data model is already available from the project design team.

<table>
<thead>
<tr>
<th>Bean recommendation Engine</th>
<th>Bean Recommendation Engine</th>
<th>Technical lead/architect or WebSphere Commerce designer</th>
<th>As for project role</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Collector</td>
<td>Any</td>
<td>Understanding of WebSphere Commerce schema; SQL knowledge</td>
<td></td>
</tr>
<tr>
<td>SQL Extractor</td>
<td>Any</td>
<td>Understanding of WebSphere Commerce schema; SQL knowledge</td>
<td></td>
</tr>
<tr>
<td>Document Helper</td>
<td>Technical lead/architect or WebSphere Commerce designer</td>
<td>As for project role</td>
<td></td>
</tr>
<tr>
<td>SQL translation</td>
<td>SQL Translator</td>
<td>Technical lead/architect or WebSphere Commerce designer</td>
<td>As for project role</td>
</tr>
<tr>
<td>SQL validation</td>
<td>SQL Validator</td>
<td>Technical lead/architect or WebSphere Commerce designer</td>
<td>As for project role</td>
</tr>
<tr>
<td>Transition function</td>
<td>Tools offered</td>
<td>Project role</td>
<td>Skills required</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------</td>
<td>---------------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Bean generation</td>
<td>LOQS Code Generator</td>
<td>Technical lead/architect or WebSphere Commerce designer or WebSphere Commerce programmer</td>
<td>As for project role</td>
</tr>
<tr>
<td>LOQS Runtime</td>
<td>System administrator</td>
<td>System administrator</td>
<td>WebSphere Commerce administration knowledge; team repository administration knowledge</td>
</tr>
<tr>
<td>Data migration</td>
<td>Data Migration Tool</td>
<td>Data migration expert or DBA</td>
<td>Understanding of WebSphere Commerce schema and data migration; some knowledge of WebSphere Commerce administration; Database knowledge of DB2 and Oracle; SQL knowledge; Database administration</td>
</tr>
</tbody>
</table>

Use of the Data Migration Tool assumes that a mapping spreadsheet for the data model is already available from the project design team.
Pre-sales, site maintenance, and enhancement phases

In this appendix, we provide a brief description of the pre-sales and site maintenance and enhancement phases. These represent the entry and exit points, respectively, for a typical WebSphere Commerce project.
**Pre-sales phase**

During the pre-sales phase, we recommend conducting a high-level fit-gap analysis to begin defining the project scope and to help develop a rough sizing of the work effort. A good vehicle for this activity is to participate in a fit-gap analysis workshop with the customer.

The output of this activity should provide information to support development of a plan to proceed with the project. In terms of a normal consulting engagement, this information leads to a statement of work (SoW) or a document of understanding (DoU). Figure B-1 provides an overview of this phase.

![Figure B-1 Overview of pre-sales phase and deliverables](image)

The scope of the first SoW or DoU would typically be limited to cover only that work that is required to produce a valid site design, which is a major output of the design phase. A second SoW or DoU would then be used to describe the remaining tasks for the project that come after site design.

As an additional observation, when engaged with an in-house project, which is one that is owned and operated by the customer, it is likely that additional time should be allocated for obtaining project sponsorship.

A checklist summary for the activities and work products of the pre-sales phase is provided in Table B-1 on page 139.
Table B-1  The pre-sales phase

<table>
<thead>
<tr>
<th>Concept</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase name</td>
<td>Pre-sales phase.</td>
</tr>
<tr>
<td>Short description</td>
<td>This phase is conducted before a WebSphere Commerce project starts formally. The customer and the organization providing the project team must work together to define and document high-level objectives, project scope, and a work proposal ready for the WebSphere Commerce project to start.</td>
</tr>
<tr>
<td>Input work products</td>
<td>RFI, RFP, or customer business requirements document.</td>
</tr>
<tr>
<td>Output work products</td>
<td>As a result of conducting a fit-gap analysis workshop, several outcomes would normally result:</td>
</tr>
<tr>
<td></td>
<td>▶ Initial fit-gap analysis document containing rough sizing</td>
</tr>
<tr>
<td></td>
<td>▶ Refined customer business requirements document</td>
</tr>
<tr>
<td></td>
<td>▶ Education roadmap</td>
</tr>
<tr>
<td></td>
<td>▶ Knowledge communicated face-to-face on WebSphere Commerce and its high-level relevance to the customer business requirements</td>
</tr>
<tr>
<td></td>
<td>▶ First SoW or DoU</td>
</tr>
</tbody>
</table>

If this is a transition project, a high-level analysis of the current site’s assets, such as Net.Data macros, C++ commands, and the current data model, will help in performing the fit-gap analysis during this phase. We would recommend the use of the Transition Tool Suite (TTS) to reduce the effort in transitioning an e-commerce site, in which case, it would be necessary to factor in the tools offered by TTS into the fit-gap analysis. Further information about using TTS for WebSphere Commerce transition projects is provided in the Transition Tool Suite User Guide, which is provided with TTS.

**Pre-sales phase sizing**

In many projects, customers will require budget and planning numbers as soon as possible, often before detailed requirements have been established. Although the design work will not begin until after the pre-sales phase, it may be possible to provide an approximate sizing by comparing the project with similar scenarios or projects. These prior experiences may give an indication of the number of hours and resources required.
Pre-sales phase sizing is experience-based and is also referred to as business-process or scenario-based sizing. The approach taken is typically to break down the project as follows:

1. Identify each high-level business process or key subsystem.
2. For each of these processes or subsystems, identify the business functions that must be implemented.
3. For each business function, produce a list of functional requirements and implementation scenarios, and assign an assessment of expected effort based on prior experience.
4. Add the total assessed effort.
5. For any other project aspects not already included, add a percentage effort for coding or testing activities accordingly. The percentage used should again be based on prior experience.

This sizing approach does not require a design, and the result will certainly be less accurate than a sizing performed after design has been completed. However, it will help to provide the customer with an early indication of the likely project sizing, and so assist them with early activities, such as ordering equipment.

Site maintenance and enhancement phase

This phase marks the beginning of site activities that are related to maintaining and enhancing (based on new and changing business requirements) site assets after the successful launch of the customer e-commerce site. The project team completes its major role before this phase starts, and this phase would normally continue for the remaining life of the site.

A checklist summary for the activities and work products of the site maintenance and enhancement phase is provided in Table B-2 on page 141.
Appendix B. Pre-sales, site maintenance, and enhancement phases

Table B-2  The site maintenance and enhancement phase

<table>
<thead>
<tr>
<th>Concept</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase name</td>
<td>Site maintenance and enhancement phase.</td>
</tr>
<tr>
<td>Short description</td>
<td>This phase marks the beginning of site activities that relate to maintaining and enhancing site assets after the new or transitioned site is launched. The project team completes its major role before this phase starts.</td>
</tr>
<tr>
<td>Input work products</td>
<td>Site deployment and launch phase outputs (see Figure 6-2 on page 121). A live e-commerce site following completion of the WebSphere Commerce project.</td>
</tr>
<tr>
<td>Output work products</td>
<td>Error-free live site. If new or changed business requirements have been fulfilled, the live site will have been successfully enhanced.</td>
</tr>
</tbody>
</table>

For a transition project, it is assumed that before the start of the project the customer site is already in the maintenance phase. After the transitioned site has been launched, however, the customer site will normally re-enter the maintenance phase to continue its life cycle.

After launching the site, the customer IT or support team can take charge of the ownership of maintaining and enhancing the site. Besides regular maintenance, this team may also have the mandate of adding new features or enhancing the existing features of the live site, based on new or changing business requirements.

In order for the customer IT or support team to manage this activity effectively, it must have the following:

- A fully functional development environment so that the site code may be fixed or enhanced as desired
- A test environment on which to assess changes before moving the code to the production site
Related methodology concepts

This Redpaper uses specific terminology to describe a WebSphere Commerce development methodology. The terms are in common use among the Toronto-based IBM Software Services commerce team and originate from IBM Global Services terminology, subsequently adapted slightly for WebSphere Commerce implementation projects.

In this appendix, we consider other methodologies used within IBM or as part of engagements where IBM consultants work with customers on implementation projects. The purpose is to highlight that the literal terms used in this Redpaper are less significant than the concepts they embody, with the proviso that all consultants and developers working on a project must have an agreed upon and common understanding of the terms being used.

**Note:** It is not possible to provide significant detail for the methodologies within the scope of this appendix; the reader is strongly encouraged to obtain further information from “Related publications” on page 147.

In deciding the methodology and terminology to be used, the starting point should be to assess:

- Project team and customer team familiarity with existing methods and terms
The match between the conceptual model that underpins the method and the project requirements

For example, a project team that has worked with the customer team on previous projects is likely to have developed their own local interpretation or “dialect” of terms that have evolved as an optimized amalgamation of standard methods and terms. It would be unwise to discard the procedural and teaming benefits of this localization, as long as the basic rigor of the method is maintained.

Similarly, a methodology that focuses on code implementation using an object-oriented model may be inappropriate to use on a project that simply requires reconfiguration of standard components, rather than development of code using an object-oriented language.
IBM methods

Not surprisingly, IBM has a well-established discipline for managing development projects. Within that context, there will be development methodologies that are optimized according to the nature of the project work. For example, the process of designing and developing a system performance assessment tool may have differences when compared with the design and development of an embedded hardware component for a Point-of-Sale terminal.

The intent is to ensure that a balance is found between a process that is insufficiently structured, and so allows freedom and flexibility, but is hard to measure and document, against a process that is too structured, and so constrains creativity and that may be bureaucratic or slow.

Further information can be obtained from IBM Software Services, as discussed in 1.1, “Further information” on page 9.

The Rational Unified Process (RUP)

The Rational Unified Process (RUP) is a software engineering process. Its purpose is to provide a disciplined approach to identifying and assigning tasks and responsibilities within a development project. RUP is intended to be configurable according to project circumstances.

The complexity of most significant development projects make it unlikely that a simple sequential development model will succeed; rather an iterative process must be applied where earlier steps will be revisited and refined in the light of subsequent understanding and progress on the problems and tasks. RUP supports iterative development in a number of ways, for example, using frequent test releases of code or assets to allow customer testing, verification, and feedback.

RUP identifies four development phases:

1. Inception
2. Elaboration
3. Construction
4. Transition

Table C-1 on page 146 shows how the RUP phases correspond approximately to the phases identified within this Redpaper.
Table C-1  Comparison between RUP and Redpaper terminology

<table>
<thead>
<tr>
<th>RUP</th>
<th>Redpaper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Pre-sales, discussed in “Pre-sales phase” on page 138</td>
</tr>
</tbody>
</table>
| Elaboration          | Design phase, discussed in Chapter 3, “Designing the commerce site” on page 31  
|                      | Solution outline                                                        |
|                      | Macro design                                                            |
|                      | Micro design                                                            |
| Construction         | Implementation and testing, discussed in Chapter 4, “Preparing and implementing the commerce site” on page 73, and Chapter 5, “Systematically testing a commerce site” on page 103  
|                      | Project implementation                                                  |
|                      | Functional Verification Test                                            |
|                      | System Integration Test                                                 |
|                      | System Verification Test                                                |
|                      | User Acceptance Test                                                    |
| Transition           | Site deployment and launch, discussed in Chapter 6, “Preparing to go live” on page 119 |

Summary

If a match is found between a standard method and the project requirements, then clearly the method is a strong candidate for use on the project. If no clear match is found, then the closest approximations should be considered to assess the impact of the discrepancies. It may be that a close match can be adjusted sufficiently to support the project, while not compromising the architectural foundation of the method itself.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 148.

- WebSphere Commerce V5.4 Developer's Handbook, SG24-6190
- WebSphere Commerce Suite V5.1 Customization and Transition Guide, SG24-6174
- WebSphere Commerce Suite V5.1 Handbook, SG24-6167

Other resources

These publications are also relevant as further information sources.

- IBM WebSphere Commerce Programmer's Guide, GC09-4951

IBM Transition Tool Suite (TTS)

Both TTS Version 1 and Version 2 can be obtained free of charge through the developerWorks™ (dW) Toolbox, with each listed as a download in the Toolbox subscription online catalog under the WebSphere heading. To access the code quickly while viewing the catalog, use Ctrl-F (the find on this page option) to open the Find Window, and then type Transition Tool Suite. Click OK. Because the code is updated periodically, there is no direct link to either package, as the link would most likely be replaced by a new one once an update has been posted. The dW Toolbox catalog may be found at:


After accessing the appropriate TTS Web page, please note that you must be a registered Guest member of dW to download the code. Please use the following link to create a profile and register with dW if you are not currently a member of PartnerWorld® for Developers:

You will be asked to create a user ID and password that will enable you to download code from the Toolbox subscription online catalog. As a guest, you will be able to download selected resources of the Toolbox subscription, such as Internet and Java tools, sample code, and entry-level software servers, all for free from IBM.

After downloading TTS and invoking the self-extracting zip file, you will be prompted to read and then accept the license terms that govern the use of TTS. After you accept these terms, you will receive the program code, as well as the following user documentation and reference materials:

- **LOQS CodeGen User Guide**
- **LOQS Installation Guide**
- **LOQS Programming Reference**
- **Transition Tool Suite User Guide** (appropriate to the version downloaded)
- “Transitioning WC Sites using TTS” (paper)
- “Using TTS in a WCS Transition Project - A Case Study” (paper)

**Referenced Web site**

This Web site is also relevant as a further information source:

- Rational Universal Process
  

**How to get IBM Redbooks**

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

**IBM Redbooks collections**

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.
Best Practices and Tools for Creating IBM WebSphere Commerce Sites

This IBM Redpaper introduces a project development methodology that was specifically adapted for the creation of e-commerce sites using the IBM WebSphere Commerce family of software products. This Redpaper details two WebSphere Commerce development scenarios:

- Creating a new e-commerce site
- Transitioning an existing WebSphere Commerce site built on the Version 4 framework to Version 5.1 or 5.4

This Redpaper describes and documents the use of IBM Transition Tool Suite during a transition project to highlight for the reader how helper tools can be used to automate some, or many, of the transition tasks that are associated with transitioning WebSphere Commerce assets or data. When used properly, helper tools will reduce the overall migration effort and maximize WebSphere Commerce asset or data reuse. Examples are provided to identify for you how the helper tools can be used independently of the development methodology or with it to improve your efficiency. The objective is to document how the tools can be combined with the development methodology to streamline a WebSphere Commerce transition project.