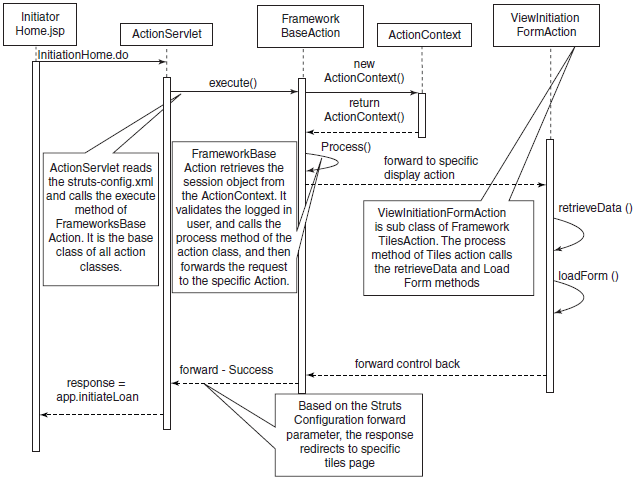
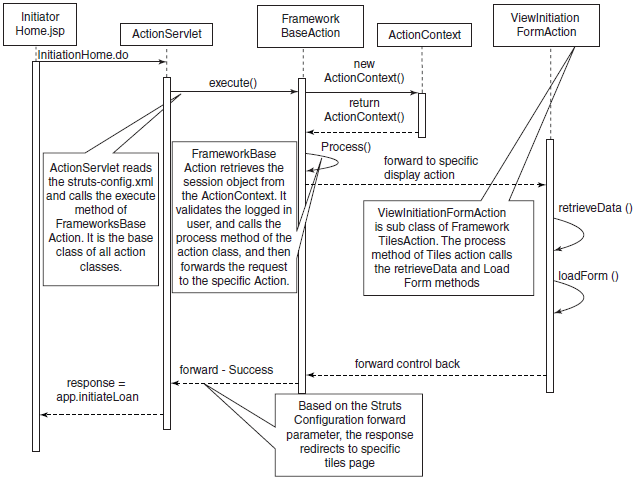
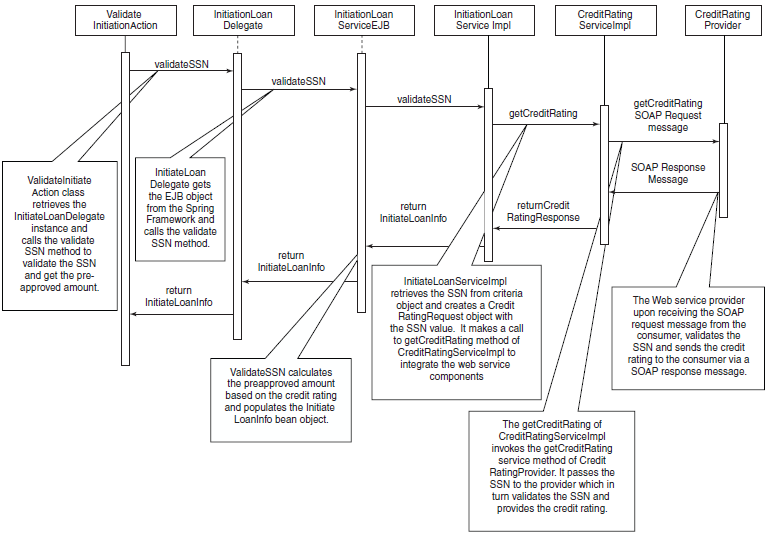
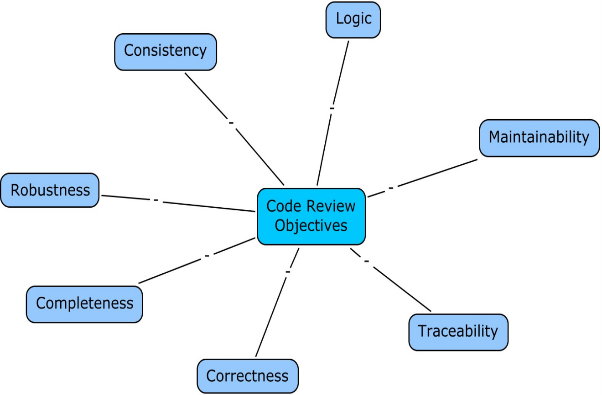
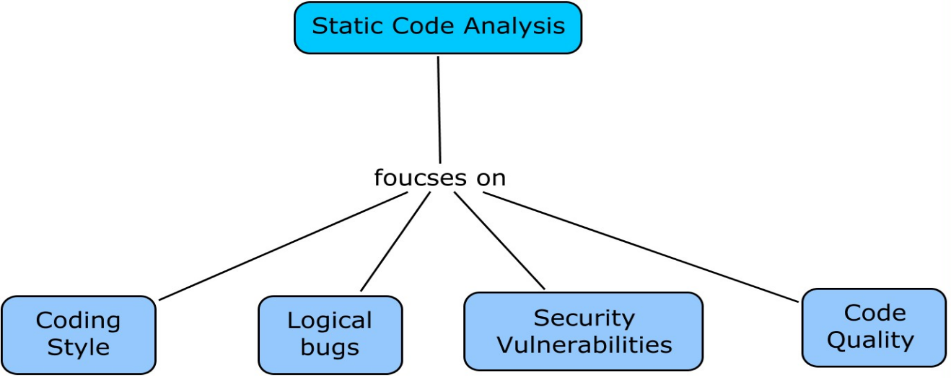
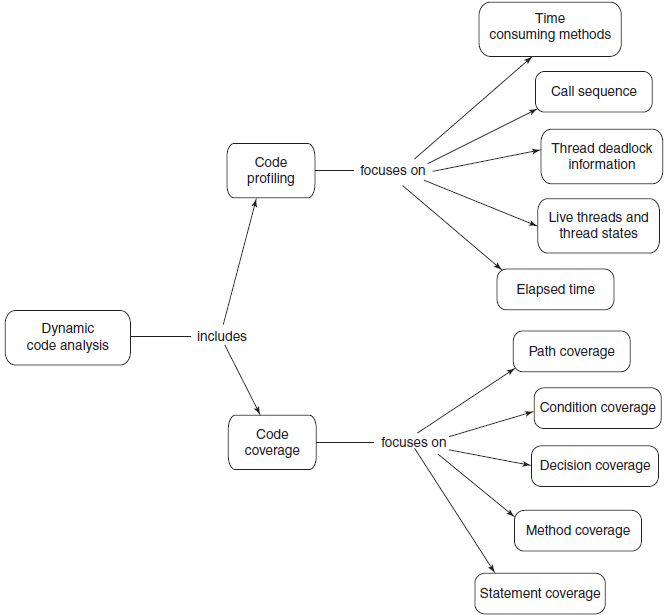
* **Constructing Enterprise Applications**
* Construction
* The construction of an enterprise application involves translating the design into code components to build an application, and fulfill the business perspective by leveraging architecture, development techniques, modeling, frameworks, tools and technologies.
* The translation of design to code typically involves the following activities:
  + Achieving construction readiness
  + Constructing solution layer
  + Code review and static code analysis
  + Compilation and creating deployable packages
  + Unit testing
  + Code profiling and code coverage analysis
* Construction Readiness
* **Defining a Construction plan**
  + A construction plan will consist of the sequence and schedule of tasks required to be carried out in the construction phase, their interdependencies, technical risks, and resources required to implement the task.
  + The primary objective of the plan is to complete the construction phase in the shortest time possible, while not compromising on functionality and quality requirements.
* **Defining a Package structure**
  + The package structure reflects the physical organization of the units of code.
  + It provides a clear picture to the developer about the location of classes they develop relative to rest of the code.
* **Setting up a Configuration Management plan**
  + The evolution of artifacts through several versions has to be tracked in order to ensure the integrity and consistency of the artifacts, and is referred to as configuration management.
  + The artifacts that are managed are referred to as configuration objects.
  + Required for controlled change management, provides the basis for managing the configuration objects in a consistent and meaningful manner by specifying the tool to be used for the purpose, defining a repository structure, which is aligned to the package structure defined for the project, and specifying appropriate access rights to parts of the repository for the different users.
* **Setting up a Development environment**
  + A development environment is the construction ecosystem that facilitates developers to develop, build, review, verify and analyze code components of an application under development.
* Elements of a Typical Development Environment
* Hardware including operating systems
* Servers such as application, Web, database, directory and portal servers
* Integrated development environment
* Third-party libraries
* Build tools
* Unit testing tool
* Profiling tool
* Static code analysis tool
* Licenses for all the required software
* Constructing the Solution Layers
* Typically construction of enterprise applications do not start from the scratch, rather the organizations have reusable assets.
  + Application framework is one of the most important reusable components. It provides the plumbing code across all the layers of an enterprise application.
  + A repository of technical components typically has components such as workflow component, rules engine, infrastructure services components and generic reporting components.
  + A repository of domain specific components typically has components related to business such as check credit rating component and quote generation component in the banking domain.
* Components in the technical solution layers
  + Infrastructure services layer components
  + Presentation layer components
  + Business layer components
  + Data access layer components
  + Integration layer components
* Infrastructure Services Layer Components
  + Infrastructure services layer of an enterprise application comprises of general purpose components such as logging, session management, security, exception handling, auditing, caching, notification and reporting, which are typically used across all the layers.
  + These components are usually designed and constructed as part of the application framework. A common scenario is that such components are readily available off the shelf, and are plugged into the framework with enhancements and proper configuration as required.
* Logging
* Logging is an important element of infrastructure services used by almost all the solution layers.
* Third party packages, such as Log4J, are generally used to implement the logging functionality in a typical Java enterprise application.
* Calls to the package APIs are inserted in the application code.
* Best practice is to improve the maintainability and flexibility of code by insulating the application from a specific logging package.
* In LoMS, the LoggingManager class provides an encapsulation of the Log4J package, and helps in decoupling the application from package specific APIs.
* Another approach to implement logging in an enterprise application is to use aspect-oriented programming (AOP) to achieve better modularity.
* Other Infrastructure Services Layer Components
* **Exception handling**
  + Exception handling is a mechanism used by an application component to take appropriate corrective or preventive measures, due to occurrence of an exception at runtime of the application.
  + Exception handling is one of the mandatory infrastructure components that typically get fabricated as part of the application framework components.
* **Session Management**
  + LoMS uses “in-memory” storage technique for session management, and JBoss caching solution to implement it.
* **Caching**
  + In LoMS, the caching component is implemented as a tree cache, where the cache is organized as a tree. Each node in the tree is implemented as a map for storing key-value pairs.
* Presentation Layer Components
* 
* Simplest of the options to construct the presentation layer components in the enterprise application is by using JSP, servlet and plain old Java object (POJO) components .
* The design team has to take care of various things to construct a successful presentation layer such as selecting the right design patterns, decide the right look and feel, give due consideration to input validation, achieve maximum reuse, ensure ease of navigation, session management and caching of data, internationalization and several other issues.
* Presentation layer can be constructed using several frameworks such as Struts, JSF, Tapestry and Stripes.
* LoMS – View Initiation Form
* Business Layer Components
* The construction of business layer components can also be done in several ways. Implementation using POJO is among the simplest of the options available.
* Design team has to consider several aspects to ensure the construction of a robust and flexible business layer such as applying the right design patterns, designing optimized and appropriate transaction management and concurrency control mechanisms, designing business workflows and business rules repositories, and many more things.
* Spring framework and EJB 3.0. can also be used to implement business layer components.
* LoMS – Initiate Loan
* 
* Data Access Layer Components
* The data access layer should be designed in a manner that it is decoupled not only from the business components but also from the underlying relational tables or any other data source.
* Developers have multiple choices ranging from POJOs to the available persistence layer frameworks/technologies.
* The simplest of the data access layer technologies is Java Database Connectivity (JDBC), wherein SQL queries are directly embedded in Java code. This may lead to a tight coupling between Java objects and the underlying relational data stores.
* The concept of object relation mapping (ORM) has come into practice to avoid such coupling.
* The ORM approach enables developers to think in terms of objects rather than tables or relations, when it comes to persistence of objects. This enables developers to apply object-oriented best practices in implementing a data access layer.
* Integration Layer Components
* Construction of integration layer components can be done in a variety of ways, and at different levels (of application layers), as discussed in Chapter 3, where you have learnt about several integration mechanisms and technologies along with the design elements that facilitate the integration of applications by realizing a robust integration layer.
* Integration layer of LoMS uses JAX-WS API to implement the SOAP-based Web services integration between LoMS and credit rating system.
* The JAX-WS API is used to expose a Web service as a POJO interface. When the Web service consumer invokes the service method of the POJO interface, it automatically generates a SOAP request message to communicate with the provider.
* LoMS – Retrieving the Credit Rating
* 
* Code Review
* The review process plays a pivotal role in measuring the degree of compliance with the specifications provided, and verifying the quality of artifacts produced during development.
* The code artifacts are the primary target of review in the construction phase.
* Code reviews may be performed manually or in an automated fashion against certain set of guidelines and best practices.
* Reviews can be conducted in multiple forms. Review can be structured or unstructured, formal or informal, a self review or a peer review, or a code walkthrough driven by the developer or by the reviewer.
* Code Review Objectives
* 
* Completeness review ensures that code is in line with the design of a software component and meets all key requirements.
* Correctness review typically vouches for conformance to coding best practices such as use of language-specific idioms, avoiding hard coding and eliminating unused variables.
* Consistency review ensures uniformity in coding, commenting, error and exception handling.
* Review of code from logical perspective is to ensure that the code results in expected behavior.
* Maintainability review is performed to ensure that the code is easily understood from the perspective of maintenance, i.e., readability of the code and is supported by adequate documentation.
* Traceability review takes care of finding out whether any requisite functionality is missing, and whether the source code of the software unit can be traced back to its functional requirements and to corresponding design elements.
* Reviewing code from the robustness perspective is to ensure that the code is able to handle errors and unexpected events at runtime
* Static Code Analysis
* Static code analysis is an activity of analyzing the code, in source or byte-code form, to identify various issues related to non-runtime aspects of the code.



* Static Code Analysis Objectives
* **Coding Style**
  + Coding style check focuses on use of right language features such as use of apt control constructs, appropriate language idioms and right type of data structures.
* **Logical Bugs**
  + While finding logical bugs is essentially the endeavor of manual code review activity, a small subset of bugs may also be identified through encoded rules verified as part of static code analysis.
* Static Code Analysis Objectives (Contd…)
* **Security Vulnerabilities**
  + Security is an essential component of enterprise applications, and starts from its inception. Developers and reviewers also have to follow various secure coding practices to ensure the development of secure code.
* **Code Quality**
  + Code quality analysis is important to ensure that code being delivered is of the highest quality in terms of its modularity, extensibility, maintainability, reusability, testability and performance.
* Build Process
* Every enterprise application requires an accurate, automated and maintainable build process in order to effectively and efficiently build an application.
* The build process is not limited only to the compilation of source code, but also comprises of other key activities such as retrieving the appropriate set of code units from source code control systems, packaging application components and libraries, component specific compilation, and deploying the application into a development environment.
* Tools such as Apache Ant and Maven are used to automate the build process of Java based enterprise applications.
* Unit Testing
* Unit testing is the first formal level of testing in a SDLC, where individual units of software are tested, independent of other part(s) of an enterprise application.
* Units can be modules, procedures, methods or classes based on the paradigm of software development.
* Unit testing can be a manual process or is automated by using testing frameworks/tools like JUnit.
* Tools like Cruise Control can be used to automate build and unit testing processes.
* Dynamic Code Analysis

Dynamic code analysis enables a development team to obtain a holistic view of an enterprise application in the running state, contrary to the static code analysis that involves analyzing code without even compiling it.

* Dynamic Code Analysis
* 
* Code Profiling
* Code profiling is one of the focus areas of dynamic code analysis, which is primarily concerned with the performance analysis of an application that enables the development team to quantify the performance aspects of an enterprise application.
* Code profiling of an enterprise application is not just limited to the application code for business logic. The underlying database may also have several application components such as stored procedures and triggers. These components are typically written using languages such as PL/SQL and T-SQL, which also may need to be analyzed to further fine tune the performance.
* Code Coverage
* Code coverage is defined as the level, or percentage, to which the source code of an enterprise application is tested.
* This does not check the functionality of an enterprise application, but finds out what percentage of the codebase is not being executed as part of execution of test cases.
* This helps in ensuring that the test cases created for checking the functionality are adequate.