1 Introduction

Learning Goals

After completing this chapter, you will be able to:

- Define enterprise applications.
- Classify enterprise applications.
- Explain challenges in raising enterprise applications.
- Determine the key characteristics and applicability of software engineering methodologies.
- Explain the lifecycle of raising enterprise applications.
- Identify the stakeholders of enterprise applications.
- Describe the three key determinants of successful enterprise applications.
- Outline knowledge and skill areas required to raise enterprise applications.
- Determine the success of enterprise applications.

Welcome readers! Welcome to the stupendous journey of raising typical enterprise applications from their inception to rollout! In the last few decades, the software industry has traversed a long trail in greenfield (custom-built) enterprise applications development and witnessed various software engineering paradigms and methodologies. There is no fit-all universal solution to raise an enterprise application. This book is an attempt to take the readers through the various processes, life cycle stages, patterns, frameworks, tools, technologies and many more things which are required to raise enterprise applications that can cater to the business needs of today's enterprise.

This chapter lays the foundation for the rest of the book. Readers will understand the objectives, challenges and different types of enterprise applications. It introduces various software engineering paradigms and how the life cycle of enterprise applications progresses. Readers will learn a step-by-step approach for raising successful enterprise applications. Parameters to measure the success of enterprise applications are also discussed.

1.1 Enterprise Applications

Every software application is not designated as an enterprise application. The software applications, which are the DNA of organizations and imbibe the business functionalities of the enterprises to catalyze their growth, are termed as *enterprise applications*. They not only enhance the efficiency and productivity of the organization but also help in ensuring business continuity. The software application, which imbibes complex business logic, expected to be high on performance, fortified from vulnerabilities and attack vectors, is expected to handle large volumes of data and concurrent

users and scalable on need basis, easily maintainable and extendable and able to orchestrate with the overall enterprise application landscape of the organization is typically designated as enterprise application.

A typical enterprise application landscape has varieties of enterprise applications. They can be categorized from multiple perspectives. One of the categorizations is based upon the placement of the enterprise application in an organization in terms of their visibility to customers.

- Customer facing enterprise applications or front-end systems of an organization are referred to as *upstream* enterprise applications. For example, an "order capture" application that captures online orders of customers is an upstream enterprise application.
- Likewise, the back-end enterprise applications that work behind the scenes in an organization to fulfill the customers' or end users' needs are called *downstream* enterprise applications. For example, an "order provisioning" application can be considered as a downstream application, as it helps in fulfilling and provisioning the online orders captured through the "order capture" upstream enterprise application.
- There is a third category of applications in an organization that fulfills the general organizational needs, and can be referred to as *business enabler* enterprise applications, for example payroll and human resource management applications. Figure 1-1 represents a collection of upstream, downstream and business enabling enterprise applications in a typical financial enterprise landscape.

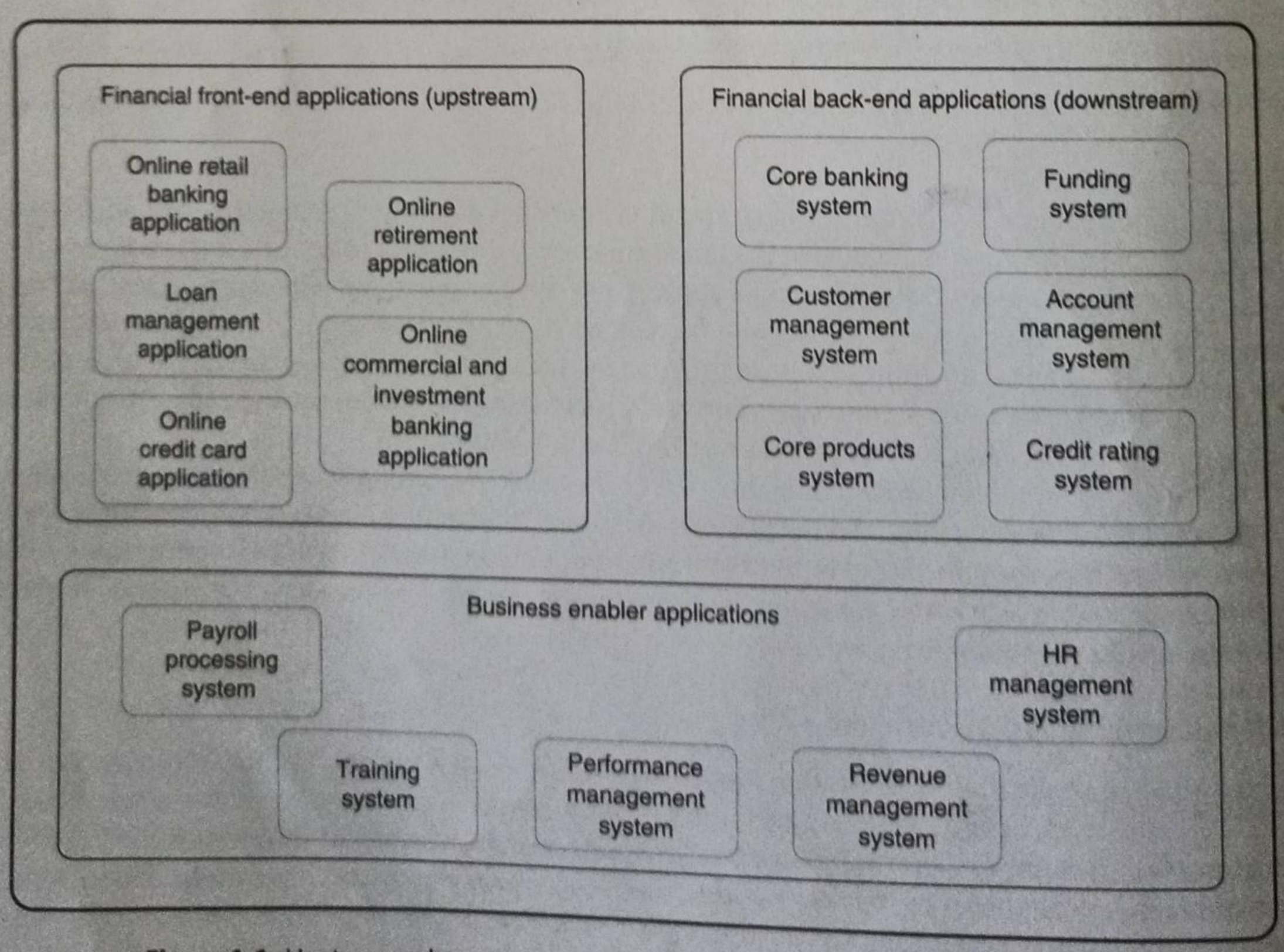

Figure 1-1 Upstream, downstream and business enabler enterprise applications.

Enterprise applications can also be categorized on the basis of industry they cater to. For example, a billing application for telecom is different from billing application for the retail industry. Enterprise applications can also be categorized based on the enterprise business functions they fulfill. For example, enterprise applications catering to order management functions or order fulfillment functions, etc.

Enterprise applications can be categorized based on parameters like nature of processing they perform. For example, enterprise applications may fall under categories such as batch processing, online transaction processing (OLTP) or online analytical processing (OLAP) applications or decision support systems (DSS). Enterprise applications can also be categorized as custom-built enterprise applications or readymade commercial off-the-shelf (COTS) packages. Enterprise applications can also be categorized as host-centric or distributed applications.

This is the era of "extended enterprises" and realization of the power of "selfservice." The business services and offerings are converging, markets are flexible, fluid and unpredictable, customers are getting more demanding and technologies are ever changing. This leads to complexities and challenges to enterprises for differentiating themselves from their competitors. Enterprise applications play a key role in meeting the business objectives. Let us explore the challenges faced by enterprise applications in order to keep up to the promises of today's enterprise.

Business processes automation

Enterprises are busy in achieving "flow through" by automating the enterprise business processes. "Flow through" is a mechanism to orchestrate the business processes spread across the organization to achieve the business process simplification. This helps organizations achieve quicker time-to-market, improved productivity and optimization of the resources. At the same time, enterprise applications are getting more and more complex to achieve such an integration and orchestration.

Data harmonization

Enterprises want to harmonize the data, which is a herculean task in itself depending upon the current state of the enterprise data. The biggest challenge is the lack of a "single source of truth" due to multiple versions, copies and representations of the same data in existence within the systems at any given point of time. This warrants the unification and standardization of data from the perspective of bringing together the disparate data and integration of systems in order to orchestrate the underlying enterprise business processes.

Application integration

In the last few decades, various enterprise applications are developed as silos. These applications are usually developed in isolation without considering the entire enterprise application landscape. In this context, application integration is one of the key challenges in front of the enterprises. As the enterprises are looking out for automating business processes, individual applications that host parts of these business processes are required to be integrated to realize the automation. The underlying diversity of platforms, different technologies and extent of integration pose challenges to application integration.

Application security

The attack surface of applications is broadening with more and more applications using the public Internet to expose themselves for access by users and other applications across the enterprise boundaries. With new attack vectors and security vulnerabilities creeping in, application security perimeter is ever increasing. The challenge is to fortify the security perimeter of enterprise application. This is

not only required to be compliant with the government regulations but also to maintain the enterprise brand value in the market.

Internationalization

Businesses are getting flatter, and the enterprises are on globalization spree. Enterprises want to reach out to their customers across the geographies using online mechanisms. The challenge is to build enterprise applications in order to serve the customers in a localized manner. Applications need to be built to serve customers in their local language and provide localized standards such as time, date and currency.

Transaction management

The online applications are ruling the enterprises and are contributing as the core infrastructure of the organizations. Vendor partners, suppliers, customers—both internal and external—and other stakeholders are accessing these applications to accomplish the business processes. Typically, a business process level transaction involves multiple system level transactions. The challenge is to consider these system level transactions as an atomic unit for identifying the success or failure of the overarching business process.

Rich user experience

Customers are getting more demanding and expect desktop like rich experience over the Internet. Providing such an engaging experience to the end user on browser is one of the challenges for enterprise applications.

Quality of Service (QoS)

The enterprise applications play a "mission-critical" role in the smooth functioning of the enterprises. The quality attributes like scalability, reliability, security, maintainability, availability etc. are inherent non-functional requirements for any enterprise application. These quality attributes are also termed as *ilities* of an enterprise application. As the complexity of applications is increasing, the challenge is to focus on core functionality along with the ilities of enterprise applications. QoS is to ensure the applications are "fit for purpose" and satisfy the stated and non-stated needs of end users and businesses. Applications that do not confirm to QoS typically result in lost productivity, revenue and goodwill.

Technology selection

Enterprise application development encompasses various platforms, frameworks and tools to select from the entire gamut of technologies—both commercial and open source ones. This comes with a challenge of not only keeping the team abreast with the technology spectrum but also to select the right set of technologies to build enterprise applications.

Governance and team productivity

Managing diverse teams and ensuring their productivity is one of the key challenges while raising successful enterprise applications. Strong governance is required to manage the team diversity. Teams have to develop the right skill sets and ensure effective reuse of existing solution elements in all the phases of application development to shorten the time-to-production of enterprise applications.

1.2 Software Engineering Methodologies

Software engineering is an engineering discipline that comprises of methodologies to develop, manage and maintain the software with focus on software quality, software requirements, software design, software development, software testing, software configuration management, etc.

Enterprise applications are developed predominantly using *iterative* methodology of software development. The traditional approach of software development is the *waterfall* methodology. It typically comprises of a sequence of phases—requirements, analysis, design, build and testing—wherein each phase output acts as input to the next phase. It typically takes more time-to-production, and the business value is realized only towards the end of a software development phase as against the iterative and agile approaches that are in vogue today, where the business values are realized in an incremental manner and time-to-production is relatively quicker.

Iterative software development approach is a cyclic approach to build software components in an incremental manner. Its planning involves choosing a subset of requirements to be implemented in each iteration. This way of development leverages the learning of the previous iterations in the next cycle to improve upon the target software component. The biggest benefit of using the iterative approach is risk mitigation. The waterfall methodology assumes the complete clarity of requirements. Consequently, the waterfall methodology follows the "big-design-upfront" (BDUF) approach which could lead to premature design decisions that may need to be reworked later. In reality, requirements evolve over a period of time which could also extend into the development stages of the life cycle. The design can also evolve in tune with the requirements and this ensures the better alignment of the design to the requirements.

IBM Rational Unified Process (RUP)[1] is one of the iterative software development processes. RUP has assembled the iterations in four phases: inception, elaboration, construction and transition. In each of the iterations, the unit of work is divided into nine disciplines. Six out of those nine are engineering disciplines namely business modeling, requirements, analysis and design, implementation, test and deployment. Other three are supporting disciplines namely configuration and change management, project management and environment.

Agile software development is an extension to the iterative approach to build applications in a nimble fashion with a light weight process. Typically in an agile project the iterations' duration is relatively smaller. These iterations are coupled with the short feedback loops to accommodate the changes quickly and appropriately guide the project forward. Rather than full-fledged design upfront, agile methodologies favor the incremental design improvements through techniques such as code refactoring and test-driven development. Code refactoring refers to improving the quality of the design through code modifications based on hindsight. Agile approach assumes a close involvement of the customer in the development process to provide continuous feedback as an application shapes up.

Extreme Programming (XP) and Scrum are two of the agile software development methodologies.[2] XP uses user stories for gathering requirements, CRC cards for just-in-time design and Pair Programming for coding. Scrum follows shorter iterations with 30-day release cycles, daily stand up meetings and concept of product back log.

The agile approaches are generally considered more suitable for small and mid-sized projects with teams consisting of more mature set of developers due to certain informal tendencies of the process

---

[1] To know more about IBM Rational Unified Process, visit http://www-01.ibm.com/software/awdtools/rup/

[2] To know more about XP methodology and Scrum methodology, visit http://www.extremeprogramming.org/ and http://scrummethodology.com/, respectively.

followed. In contrast, methodologies such as RUP are considered to be preferable for enterprise applications and projects involving large teams due to the higher rigor associated with the process.

In practice, most projects follow a hybrid approach which fosters the idea of fusing agile practices within the framework of more disciplined approaches like RUP.

1.3 Life Cycle of Raising Enterprise Applications

Raising an enterprise application exhibits a life cycle. Whatever be the software engineering paradigm used to raise an enterprise application, typically the life cycle of an enterprise application follows four phases—incepting, architecting and designing, constructing and testing, before it is rolled out for the enterprise users.

In a typical scenario, incepting an enterprise application starts as a result of enterprise analysis and business modeling activities. This is followed by the requirements engineering which typically include elicitation of requirements, and thereafter analysis of requirements, with respect to the enterprise application. Elicitation of requirements is done using use cases, prototypes or user stories. These requirements are validated to ensure their feasibility with respect to various factors including business requirements, budget, technology, etc. Incepting an enterprise application concludes with casting the plan and estimation of the project.[3]

Architecting and designing is the next phase in the life cycle of an enterprise application which takes key inputs from the enterprise architecture initiatives of an organization. Enterprise architecture defines the overall business architecture, data architecture, applications architecture and technology architecture of an organization.[4] An enterprise application is one of the pieces to be fitted in the entire jigsaw puzzle of the applications landscape. Architecture of the enterprise application is laid out from various perspectives that include logical, integration, solution, data, technology and security, to name a few, key ingredients. These perspectives are further boiled down into the respective detailed design using various design patterns, frameworks, technologies and tools. The design includes several iterations to reach the final one, which is baselined to start the next phase—the construction phase.

Constructing an enterprise application starts with building the application framework components. This is followed by construction of application components, which implements the use cases identified during the requirements engineering of an enterprise application. Thereafter the software components are unit-tested. In addition, code review is an essential activity in the construction phase. Code review happens using both static and dynamic code analysis. Static code analysis includes type checking, style checking, security review, software quality checking, etc. Dynamic code analysis typically includes code coverage and code profiling.[5]

Testing an enterprise application is the most vital phase of the enterprise application life cycle that ensures the requisite ilities in enterprise applications for a successful application rollout. Various kinds of testing are done to ensure these and typically include integration testing, system testing and user acceptance testing. System testing facilitates to test various ilities of the applications. System testing typically includes performance testing, interface testing, globalization testing, compatibility testing, usability testing and penetration testing of the enterprise application.[6] Eventually, the enterprise application is rolled out in the production environment for end users and is maintained and supported till the end of life cycle of enterprise application.

---

[3] More about incepting an enterprise application is discussed in Chapter 2.
[4] More about architecting and designing an enterprise is discussed in Chapter 3.
[5] More about constructing an enterprise application is discussed in Chapter 4.
[6] More about testing an enterprise application is discussed in Chapter 5.

Every organization expects successful enterprise applications to be associated with certain qualities of service (QoS). The predominant ones are performance, security, maintainability, reliability and scalability. These qualities of service cut across the life cycle phases of enterprise applications.

Let us take the example of security. Security requirements are captured during incepting the enterprise application. Afterwards, threat modeling is done as part of architecting and designing phase of the enterprise application and, in turn, the outputs of threat modeling are used further in the same phase to have secure architecture and design. During construction of an enterprise application, secure coding principles and best practices are applied to have vulnerability free code and further it is tested from the security vulnerability perspective.

Performance is no different. Performance requirements are captured in the incepting phase. Sizing of infrastructure and performance test strategy is laid down during architecture and design phase. Although not necessary, preliminary load testing may be done in the construction phase. During testing phase, the enterprise application is subjected to load testing, stress testing, volume testing, endurance testing and other application assessments related to scalability and performance.

Throughout the life cycle of an enterprise application, it witnesses various stakeholders. The key stakeholders for incepting enterprise applications are the sponsors and the customers. Stakeholders from the process groups, product groups, service groups, IT application groups, IT infrastructure groups, analysts groups and vendors, etc. also play an important role during incepting enterprise applications. Architecting and designing phase is carried out by enterprise architects, data architects, integration architects, solution architects and application architects. Respective designers also join hands with the architects in their respective area of architecting and designing to accomplish this phase. Construction phase is primarily carried out by the programmers to build and review the code of application. Testing phase is accomplished by various kinds of testing teams such as integration testing team, performance testing team, application security testing team, interface testing team and user acceptance testing team. Finally, an enterprise application is rolled out with the help of release team and the representatives of the team that has raised the enterprise application.

1.4 Three Key Determinants of Successful Enterprise Applications

Business and IT alignment is the key for successful enterprises. It happens as a result of conceiving and raising successful enterprise applications that are in complete alignment with the business needs of the organization. There are broadly three aspects to raising successful enterprise applications:

(a) Business case readiness
(b) Strategy to execute
(c) Excellence in execution

The foundation of a successful enterprise application is laid down by an appropriate *business case* that is supported by the organizational objectives, vision and strategy. The business case should be supported from the financial perspective and have an acceptance of all the key stakeholders. An enterprise application should have well-defined parameters derived based on the business case to measure its success.

The next thing required to raise successful enterprise application is to have an *execution strategy*. The execution strategy is a comprehensive plan that is required to manifest budget, resources, timelines and availability of subject matter experts (SMEs) in a consistent manner to realize a successful enterprise application.

Once the execution strategy is in place, *excellence in execution* is required to realize a successful enterprise application. Enterprise applications typically warrant substantial time to get completed and during which business objectives, circumstances and environments may change. Continuous assessment of the objectives, having robust change management mechanisms and stake holder management is required to keep the target of raising enterprise applications intact. Robust traceability is also required to make sure the objectives are met. On-going communications to create stable baselines, sign off or formal hand shaking among stakeholders and reverse feedback mechanism in each life cycle phase of raising enterprise application are critical to the success of enterprise applications.

The stakeholders involved in the life cycle phases of raising enterprise applications should be equipped with the appropriate and required skill sets—both technical and soft—for their successful delivery. There are various ingredients that are required to raise successful enterprise applications, which you will go through in this book. Figure 1-2 represents these ingredients as an unstructured honeycomb. Although these ingredients are the predominant ones, other aspects such as organizational constraints, team dynamics, and soft skills come together to glue all these ingredients to consummate successful enterprise applications.

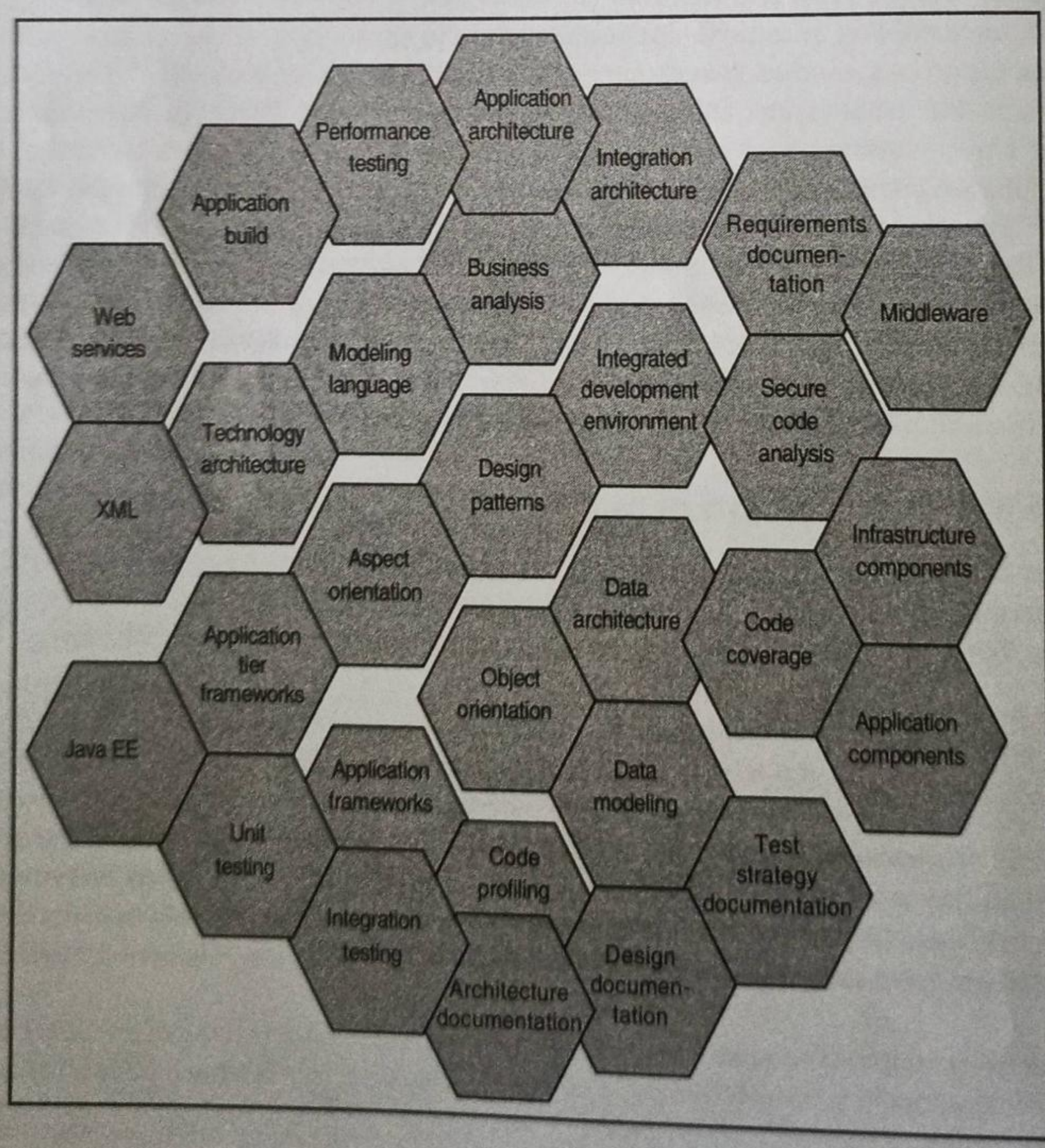

Figure 1-2 Ingredients of enterprise application.

Let us explore the typical knowledge and skill areas required by a team engaged in raising enterprise applications:

Knowledge of organizational dynamics

The most important thing required by the enterprise application team is to understand the organizational business and business needs of end users. This understanding provides several input to an enterprise application team, such as functionalities that should be packaged together, the kind of intuitive user interfaces required, components that can be reused, the positioning of a particular enterprise application in the overall application landscape and the interfaces required to be exposed or consumed by an enterprise application. This comes handy, especially during inception and architecting phase of enterprise applications.

Domain knowledge

The industry context and industry specific best practices are very important to build a competitive and cutting-edge enterprise application. It comes with the domain understanding. This provides input to the enterprise application team on the prescriptive business processes, information and data models etc. This provides valuable input to arrive at organization specific input on the processes, data and information management. This comes handy especially during inception, architecting and design and testing phase of such applications.

Business analysis skills

Skill of analyzing the business, coupled with the knowledge of domain and organizational dynamics, is very important for inception of an enterprise application. Business analysis skills are typically conglomeration of domain knowledge, technical knowledge, use of business analysis related tools and practice of soft skills. Business process modeling language knowledge adds to a robust documentation of business analysis results.

Program management skills

Raising enterprise application is a complex task, and it needs to be program-managed. Program management skills include planning, estimation, budgeting, talent management, change management, positive communication and many more things that are required to manage the program of raising enterprise applications. It is required in all the phases of the life cycle of an enterprise application.

Architecting and designing skills

Finding an optimal solution to the problem at hand requires architecting and designing skills. These skills are required in the architecting and designing phase of an enterprise application. This typically includes the knowledge of architecture views and view points, architectural patterns, design patterns, design paradigms like object orientation, aspect orientation and service orientation, usage of design tools, architectural and design best practices, technical frameworks, knowledge of modeling languages like Unified Modeling Language, etc.

Programming skills

To realize an enterprise application, it needs to be coded or programmed. Programming skills not only includes knowledge of a programming language but also the knowledge of the underlying platform, knowledge of an Integrated Development Environment (IDE) tool, programming best practices, code review skills, knowledge

of unit testing tools, configuration management and build tools, static code analysis tools and dynamic code analysis tools. Above all, "algorithmic" thinking is the key ingredient for a good programming.

Testing skills

To ensure an enterprise application is ready to use, it has to pass through multiple types of tests. Multiple types of testing skill sets are required for testing an enterprise application, which includes integration testing, performance testing, load testing, stress testing, application security testing, interface testing and user acceptance testing. Knowledge of various kinds of tools used to perform testing is also essential.

Knowledge of tools

Automation is the key to enterprise applications' quicker time-to-production and teams' productivity enhancement. This is achieved through usage of tools. Tools are used across all life cycle phases of enterprise applications. Requirements gathering and requirement analysis tools are required in the inception phase. Requirements management tools are required across the life cycle phases of enterprise applications. Architecting, design and modeling tools are required in architecting and design phase. IDEs, unit testing tools, build tools, code analysis tools are required in the construction phase and various testing tools like penetration testing, performance testing and regression testing tools are required during testing and rollout phase of enterprise application.

1.5 Measuring the Success of Enterprise Applications

For an organization, it's not only important to raise successful enterprise applications but also to measure the success of enterprise applications. Success is viewed from different perspectives by various stakeholders who have diverse roles and interests in raising enterprise applications.

Irrespective of the varieties of contexts and stakeholders objectives, typical parameters that are attributed to the success of enterprise applications are as follows:

- *Effectiveness of the solution* to the business problem at hand is the first and foremost parameter that determines the success of enterprise application. This parameter is important for all the stakeholders of enterprise application. One of the ways to measure the success is by measuring the productivity gain acquired by the end users of the enterprise application in performing their job function. Acceptance of enterprise application by end users is also a qualitative measure for this parameter.
- *Quality of enterprise application* is one of the key parameters to measure the success of enterprise applications and is important for all class and categories of stakeholders. Quality is attributed to the conformance to the functional requirements, defect-free code and adherence to the ilities by the enterprise applications. For example, to measure the success in terms of achieving performance and scalability, organizations use standardized benchmarks like online transaction processing benchmark (TPC-C). To measure security, organizations look for things like whether the Top-10 security vulnerabilities by the Open Web Application Security Project (OWASP) are being taken care or not while raising the enterprise applications. There are several other software metrics that are also used to quantify the software code quality. For example, cyclomatic complexity is used to measure the complexity of the source code, code coverage is used to measure the degree of source code tested, and defect ratio is to measure the density of defects.
- *Time-to-production* is a very important parameter from the perspective of the sponsors of an enterprise application in measuring the success of enterprise applications. In this fast moving and competitive world, business enterprises need to adapt to the new business environment

and roll out enterprise applications in the quickest possible time to be at the leading edge of the business. Time-to-production for an enterprise application is the key to success, which defines the total time taken to roll out the enterprise application since its inception.

- *Cost effectiveness* of enterprise application is another important parameter from the perspective of the sponsors of the enterprise application. It determines the expenditure versus the benefits of the enterprise application. Return on Investment (RoI), due to use of the enterprise application determines its cost effectiveness.
- *Budget and schedule adherence* in raising an enterprise application is one of the key parameter from the perspective of the enterprise application development teams. Overruns of budgets and exceeding the timelines have a direct impact on the time-to-production and cost effectiveness parameters. The enterprise application's program management team has the bottom-line responsibility for fulfilling this success parameter.
- *Productivity* is another important parameter to measure the success of enterprise applications from the perspective of enterprise application development teams. Reuse, use of best practices, frameworks and tools, and automation of software processes, etc. act as catalysts to improve productivity of the teams, which are responsible for raising enterprise applications.

Above all, one must remember that success of an enterprise application is purely a contextual thing, and it has to be vetted against the organizational vision, strategy and objectives behind raising enterprise applications.

Summary

This chapter has introduced the concept of enterprise applications, types of enterprise applications and typical challenges posed in raising today's enterprise applications. The iterative software engineering methodologies are introduced from the perspective of key characteristics and applicability. This chapter has also introduced the typical life cycle phases of raising enterprise applications, and the stakeholders who interface and participate in these life cycle phases. The three key determinants to raise successful enterprise applications are described, along with key knowledge areas and skill sets that are required from an application's stakeholders. Towards the end of the chapter, the key parameters, determining the success of enterprise applications are also elucidated.

Review Questions

1. Identify the enterprise applications in your domain area and categorize them.
2. List three most important challenges in raising an enterprise application.
3. Perform a SWOT analysis of the two software engineering methodologies.
4. Find out the names of life cycle phases in the Scrum approach.
5. Explore the 4 + 1 view of IBM Rational Unified Process.

Further Readings

Application security: http://www.owasp.org/index.php/OWASP_Top_Ten_Project
IBM patterns for e-business: https://www.ibm.com/developerworks/patterns/
IBM Rational Unified Process: https://www-01.ibm.com/software/awdtools/rup/
Performance benchmarks: http://www.tpc.org/
Scrum methodology: http://scrummethodology.com/
XP methodology: http://www.extremeprogramming.org/

2 Incepting Enterprise Applications

Learning Goals

After completing this chapter, you will be able to:

- Define enterprise analysis and business modeling.
- Apply requirements elicitation and analysis.
- Determine actors, use cases, relationships among them and use case specification.
- Explain prototypes.
- Identify various types of non-functional requirements.
- Create system requirement specification.
- Define requirements validation and requirements traceability.
- Describe project plan.
- Determine different project estimation techniques.
- List tools used for business modeling and requirements engineerings.

In the last 10 years, we have seen companies like Amazon, eBay and Google having tremendous growth. Also, we have seen other "brick-and-mortar" companies, such as Walmart and Tesco, adapting multichannel commerce to service customers and be competitive in the current environment. Even service industries like banks and telecom use technology as their backbone to serve end-customers' need with a very short time-to-market. In all these companies, we are talking about real complex business processes and systems.

To run such complex and high volume businesses, enterprises require raising complex software systems, which cater to the need of customers and other stakeholders. Incepting an enterprise application is the first step to raise it. This is required to determine the problem at hand, identify the need for the enterprise application, establish the scope, analyze the feasibility, and cast a roadmap for further phases of lifecycle of enterprise application development.

Incepting an enterprise application primarily consists of the following activities:

- Enterprise analysis
- Business modeling
- Requirements elicitation and analysis
- Requirements validation
- Planning and estimation

Let us further explore these activities in the following sections.

2.1 Enterprise Analysis

Business continuity and sustained growth are of paramount importance to any enterprise. Enterprise analysis deals with questions such as "Why is a given system needed?", "Why does a given system need change?", "Why is it important to invest at this time?" As part of enterprise analysis, a business analyst performs the following activities:

- Identifies the business opportunities and business changes
- Identifies stakeholders across business units
- Collects and prioritizes the business requirements
- Defines the business roadmap with scope and exclusions
- Determines the high-level investment needed for the enterprise
- Conducts feasibility study for any changes proposed
- Conducts risk analysis and competitive analysis
- Decides on build-or-buy strategy
- Creates "Return on Investment" (RoI) business cases with proper justifications
- Gets the necessary approvals from the sponsors

Xp

Not all enterprise applications initiatives are preceded by the enterprise analysis activity. This is typically an activity that is undertaken as part of enterprise architecture exercise and provides a roadmap for the complete set of enterprsie applications. This activity leads to fulfillment of the business goals of an enterprise,.

As part of enterprise analysis, or just after it, most organizations do a complete business modeling exercise.

2.2 Business Modeling

Enterprise analysis could lead to two forms of programs/projects:

(a) Creating something new
(b) Extension/change to something which already exists

In normal engineering terms, the first type is commonly referred to as *development program/project* and the second type as *reengineering program/project*. In a development project, the entire business requirements are arrived at from scratch. For example, to create an e-commerce platform for an existing brick-and-mortar business, one has to define the entire business process and transactions afresh.

An example of a reengineering project could be rewriting the existing brick-and-mortar business from a legacy system written in COBOL to a new modern system based on Java. In this approach, we may get the entire business process and the validations from the existing system and may have only a few new requirements. While there is no standard definition of what a reengineering project is, it is most widely accepted to mean that there are no major changes to requirements, and hence does not involve a "requirements" or "inception" phase. Requirements are derived from the existing system and implemented with little or no change.

To raise a successful enterprise application, it is important to understand the business and business imperatives in detail. These business details and processes have to be well documented and a business user should be able to verify/validate it easily. Business modeling helps one understand the business information and the business processes, which an enterprise uses to fulfill its business goals. Business modeling is done by business analysts.

To understand the business problem at hand it's essential to have an *as-is* and *to-be* modeling of the business processes. The "as-is" business process model reflects the existing business process, and

the "to-be" business process model reflects the desired business process. "As-is" and "to-be" business modelings catalyze the understanding of the deficiencies of the existing process and provide insights about how they could be overcome.

Tt Tools for business modeling can be as simple as using a whiteboard or can be of varying degrees of sophistication from a simple diagramming tools like Microsoft® Visio® to a full-blown modeling notation-aware tool such as Enterprise Modeler®.

Let us further understand more about business modeling using a case study in the loan-banking domain. The same case study will be followed throughout this book so that readers can understand and appreciate how an enterprise application is raised.

2.3 Loan Banking Business: Case Study of EM Bank

EM Bank (Easy Money Bank) is a leading global bank, headquartered in Europe, with presence in more than 50 countries across continents. EM Bank offers services in core banking, investment banking, mortgage, foreign exchange and wealth management.

The core banking segment of EM Bank has been offering loans to its retail customers through its branches, since 1970. In order to keep up with the competition and current business trends, the bank management decided to expand the loan offerings to a wider customer base using multiple channels to reach the prospective and existing customers.

As part of the business expansion exercise, EM Bank's business analysts analyzed the "as-is" business process of the loan management services. Figure 2-1 depicts the "as-is" business process for loan management.

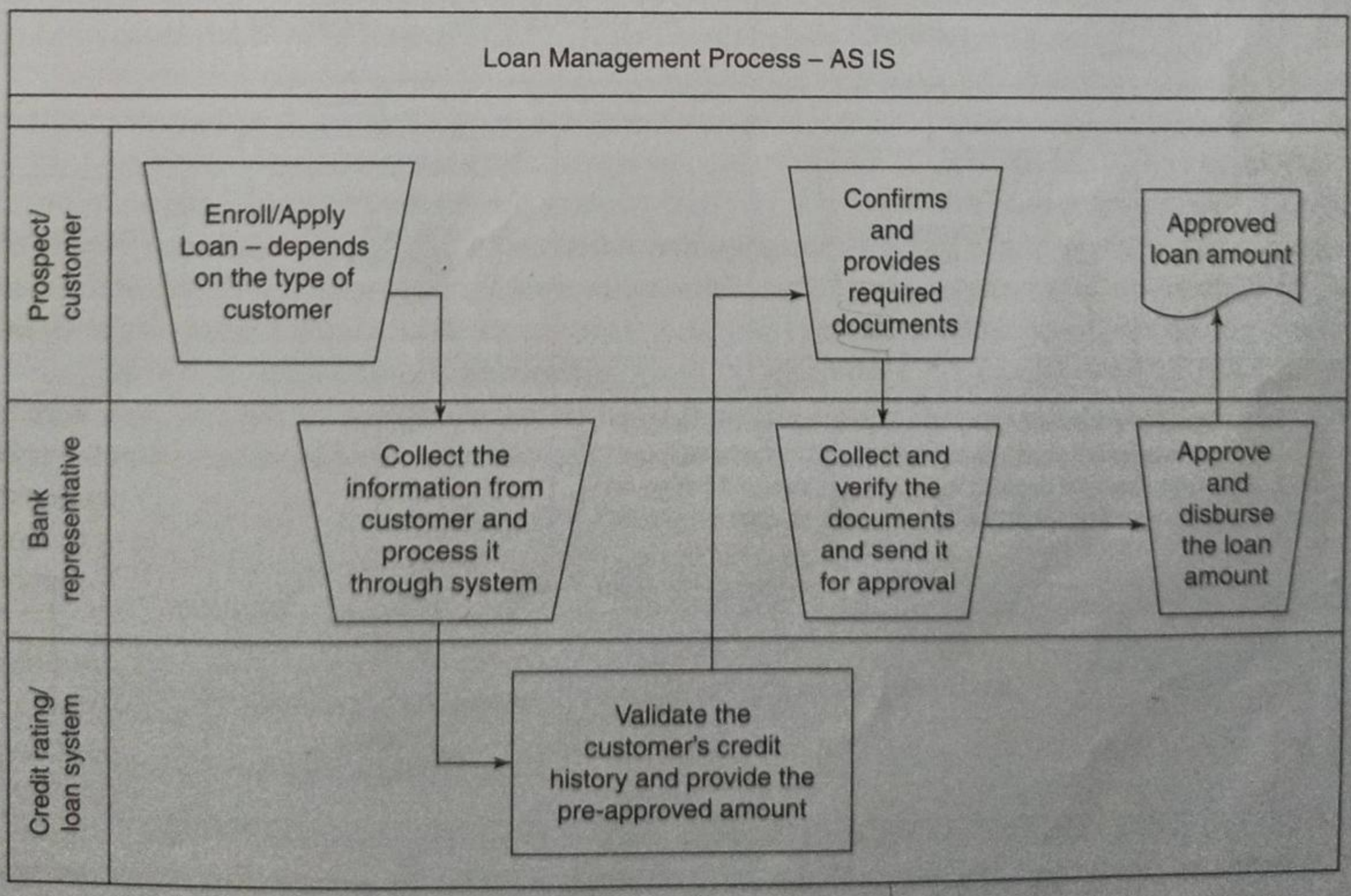

Figure 2-1 Loan management—"as-is" business process.

As depicted in Figure 2-1, EM Bank has a single channel (only through bank branch) to offer loans. The preapproved loan amount is decided based on customer relationship with the bank. All the verification and approval processes are done by the bank representatives. There is no external agency interface to provide customer credit rating.

After analyzing the "as-is" business process, the following "to-be" process (as illustrated in Figure 2-2) is identified in order to meet the objective of expanding loan offerings:

- Automate loan offerings for existing customers as well as new customers (prospects) via bank branches and the Internet.
- For the prospect to do business with the bank and get a loan, a prospect has to register (enroll), either through the Internet, or by meeting with one of the bank's representatives. On successful completion of the registration process, the prospect will be treated as an existing customer.
- During customer loan initiation process, the customer fills in the particulars either through the Internet or by contacting a bank representative. Information about the preapproved loan amount will be made available to the customer at the time of loan initiation. The customer can either choose to go with the preapproved loan amount (or an amount less than it), or opt for a higher amount by placing a request for enhancement of the loan amount.

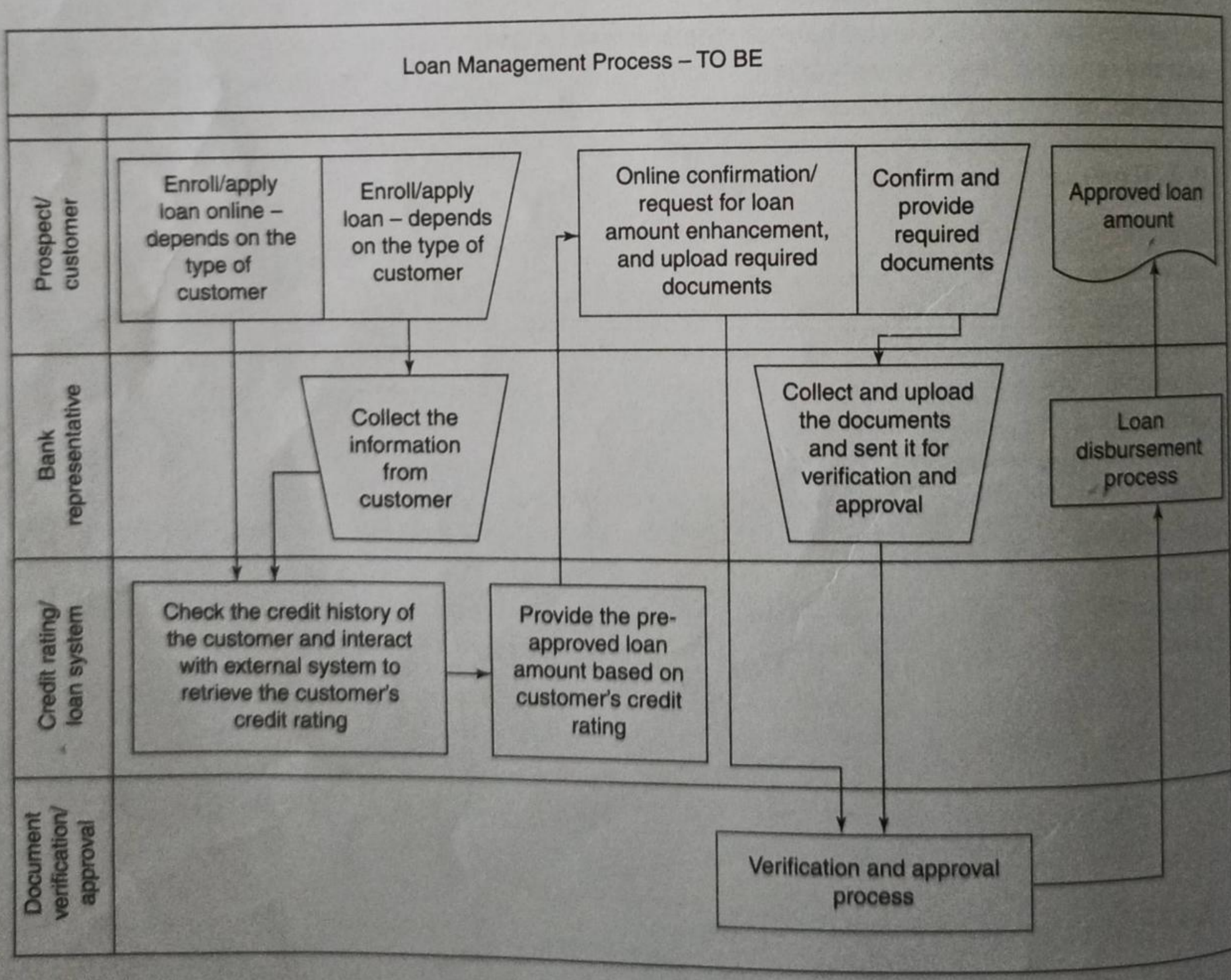

Figure 2-2 Loan management—"to-be" business process.

- The preapproved loan amount is based on the customer's credit rating. Customer credit rating facility is offered to EM Bank by an external agency.
- Upon loan initiation, the customer submits the necessary documents (such as employment proof, address proof and social security number) by either uploading them on the system, or by providing them physically to a bank representative. Physical documents are digitized, uploaded and linked to the loan application by a bank representative.
- Once the documents are uploaded onto the system, a verifier verifies the application and the related documents.
- If all the given documents are appropriate, and verified by the verifier, the loan request is sent to the approver for approval. Once the loan is approved, the customer is notified of the approval and amount. The approved loan amount is then disbursed to the customer.

To accomplish the identified "to-be" business process, a new loan management system (LoMS) needs to be developed. LoMS system will have functionalities from loan initiation to disbursement of the loan. This system will be used by both the bank customers and the bank staff. In all the subsequent chapters, LoMS application will be taken as an example to understand and appreciate the life cycle of raising enterprise applications. In practice, an enterprise application is much more complex and big, but the example application (LoMS) is sufficient to understand the key principles, paradigms and practices to raise an enterprise application.

After business modeling, the next step in incepting the enterprise applications would be to detail out the requirements. These detailed requirements would then be analyzed for completeness and validated for feasibility and correctness.

2.4 Requirements Elicitation and Analysis

Once the operation of the business is specified in business process models, it can be used as an input to perform more detailed requirements elicitation and analysis for enterprise applications.

Requirements elicitation and analysis is a systematic approach of capturing client requirements, analyzing them and documenting the problem domain. There are various ways to approach it. Review of existing systems and relevant documentation, interviewing IT personnel of the organization, and prototyping are few of them. Business analysts facilitate the requirements elicitation and analysis.

Requirements can be of two types: good requirements and bad requirements. Bad requirements are introduced primarily by making assumptions while writing requirements, or not restricting focus to the problem domain through mixing up aspects of the solution domain. Needless to say, one should always avoid bad requirements. Remember, good requirements are always SMART (that is, specific, measurable, attainable, realizable and testable/traceable/time bound). There are various kinds of requirements that need to be elicited and are broadly divided into two categories:

- *Functional requirements*. Functional requirements capture what the system is expected to do. To depict functional requirements, "use cases" and prototypes are used.
- *Nonfunctional requirements* (*NFRs*). NFRs capture how the system does what it is expected to do with respect to its constraints and expected qualities of service (QoS) such as reliability, scalability, portability, usability, availability, security and performance.

2.4.1 Actors and Use Cases

Functional requirements elicitation and analysis starts with enlisting the entities that use the system, and the functional reasons why they will use the system. The entities are termed as *actors* and the

functional reasons are termed as *use cases*. However, if one is using Extreme Programming methodology, requirements are stated as "user stories", rather than use cases.

In "A picture is worth a thousand words." This old proverb defines the soul of Unified Modeling Language (UML) that is the de-facto visual language of modeling business processes and software systems. UML2.0 is the latest specification by Object Management Group (OMG) that supports 13 modeling diagrams. Use case diagrams are one of these diagrams to represent the functional *requirements view*.

The entities or actors, as termed in UML[1] jargon, shouldn't be assumed to consist of only human users. Apart from human users, actors can be external systems or external devices. To find out the actors, one should ask a simple question "Who is going to use the system?" Customer, reviewer and approver are the human user actors of LoMS. Accounting system and credit rating system are the system actors.

While identifying actors, following few points may be of help:

- An actor typically represents a role, and not a user.
- For bigger applications, typically a "system admin" actor is required. But it purely depends on the requirements.
- Only direct actors of a system should be considered.

A use case is initiated by an actor for a particular functional reason. To determine use cases, one should ask a simple question "Why does an actor need to interact with the system under consideration?" For example, in LoMS, why does a customer want to interact with the system? Customer wants to initiate the loan. Why does the credit rating system interact with the system? A credit rating system provides the credit rating of the customer. Therefore, "initiate loan" and "provide credit rating" are two different use cases. The complete list of use cases will provide the intended behavior, and hence the scope of the system.

After identification of actors and use cases, the next step is to identify the relationships among them. The simplest kind of relationship that exists among use cases and actors is the *association relationship*, which is depicted with a straight line between actor and use case, as shown in Figure 2-3.

Once the actors, the use cases and the relationships among them are identified, the next step is to create the use case diagrams. Use case diagrams are one of the tools to capture the functional requirements. The first level of use case diagram, which represents all the actors, high-level use cases, the system boundary and the relationship among the actors and use cases, is represented in Figure 2-3. The level of use cases diagrams, one should delve into, is purely dependent on the project complexity as well as the use case complexity. The thumb rule is to stop when the implementation of a given use case can be estimated.

Tt There are several tools available to create use case diagrams. Few popular ones are Rational Software Architect, Visio and Together Architect. Other open source choices like AgroUML are also available

[1] For more information on UML, please refer to the Object Management Group's official Web site for UML, http://www.omg.org/gettingstarted/what_is_uml.htm

As shown in Figure 2-3, customer and bank representative are two human actors who participate in the "initiate loan" use case. Credit rating system and accounting system are two external system actors that participate in the same use case. Thus "initiate loan" effectively has associations with four actors.

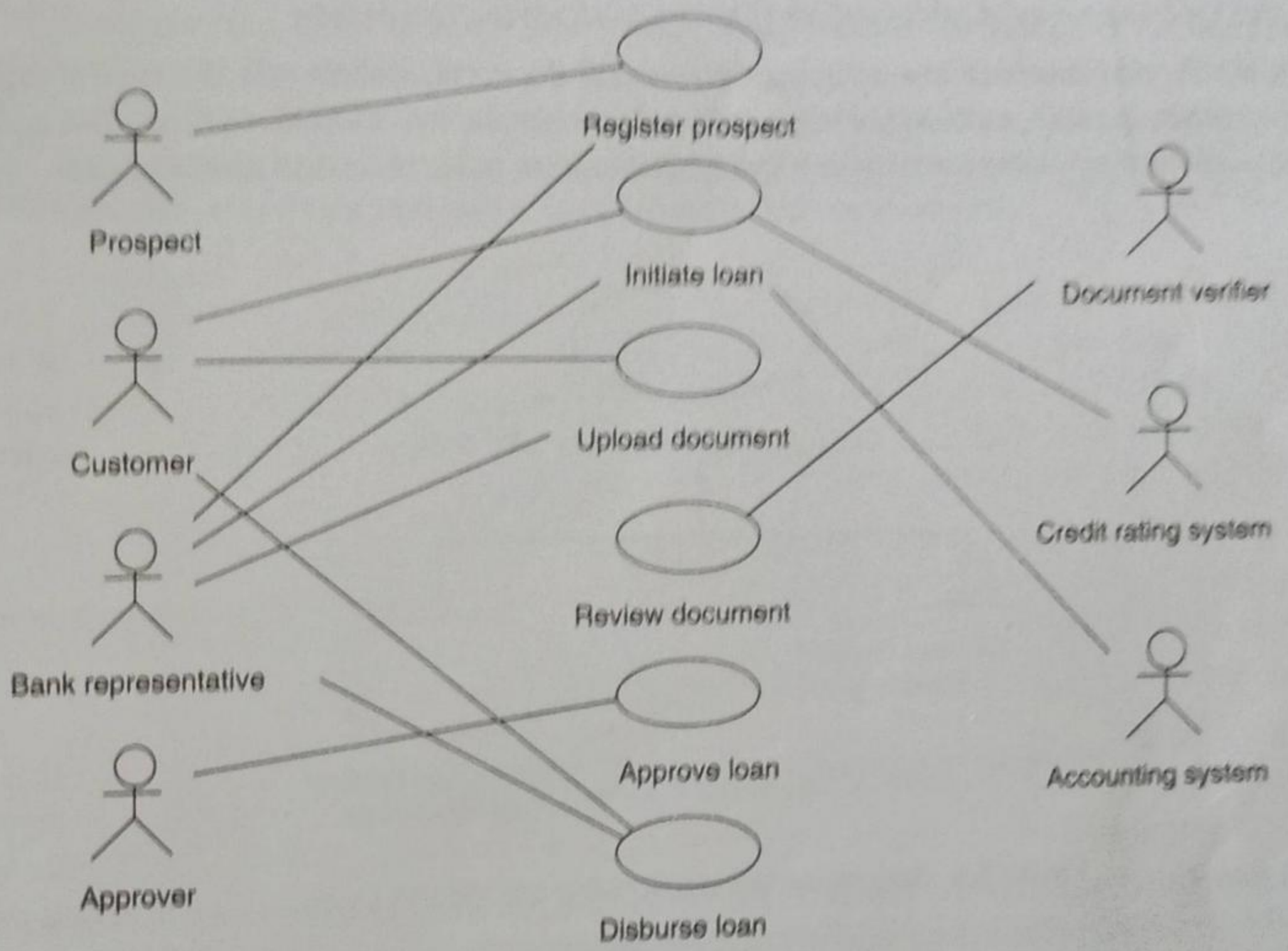

Figure 2-3 LoMS use case diagram.

Let us explore a few more relationship types that exist among actors and within use cases. As depicted in Figure 2-4, the bank representative, approver and the document verifier actors are the types of bank executive actor. This means that bank executive actor generalizes the bank representative, approver and the document verifier actor. This type of relationship is referred to as *generalization relationship* and is the only type of relationship that exists among actors.

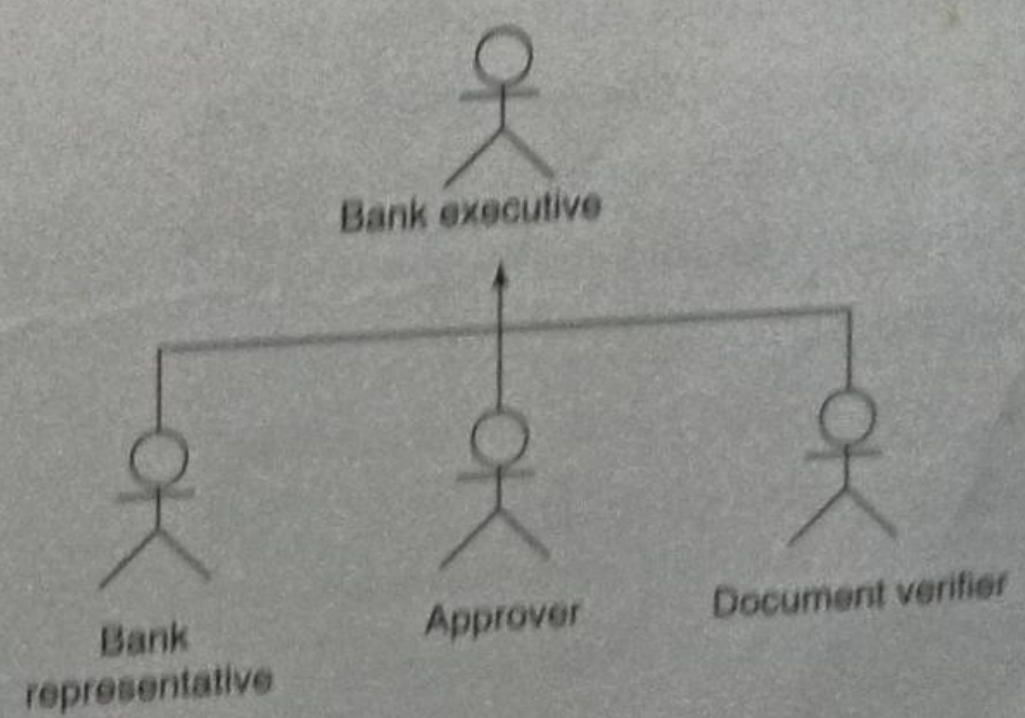

Figure 2-4 Generalization relationship among actors.

In UML2.0 does not allow associations among actors. But, the generalization relationship is useful in depicting common characteristics among actors.

There can be three types of relationships that can exist among use cases.

(a) *Generalization relationship* among use cases exists, if a particular use case is a specialized form of another use case, as depicted in Figure 2-5. For example, Initiate Loan by Customer and Initiate Loan by Bank Representative are the specialized use cases of Initiate Loan use case.

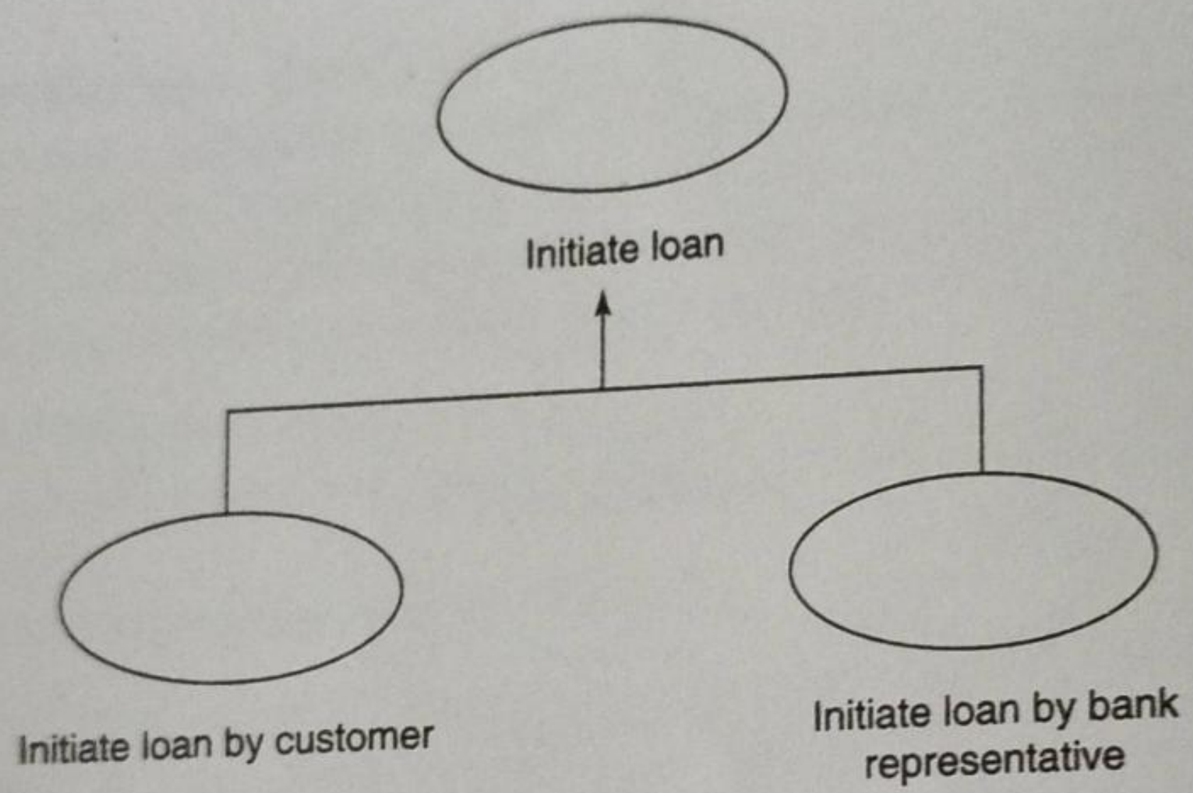

Figure 2-5 Generalization relationship among use cases.

(b) *Extends relationship* among use cases exists, if behavior of one of the use cases (extension use case) may be conceptually inserted in the base use case (extended use case) under some condition(s), as depicted in Figure 2-6. Provide Credit Rating is an extension use case of Initiate Loan, because as per loan initiation requirement, preapproved loan amount will be fetched along with the credit rating from the credit rating system in case the preapproved amount is not updated in the last six months. It's represented as a dashed arrow pointing from extension use case towards the extended use case.

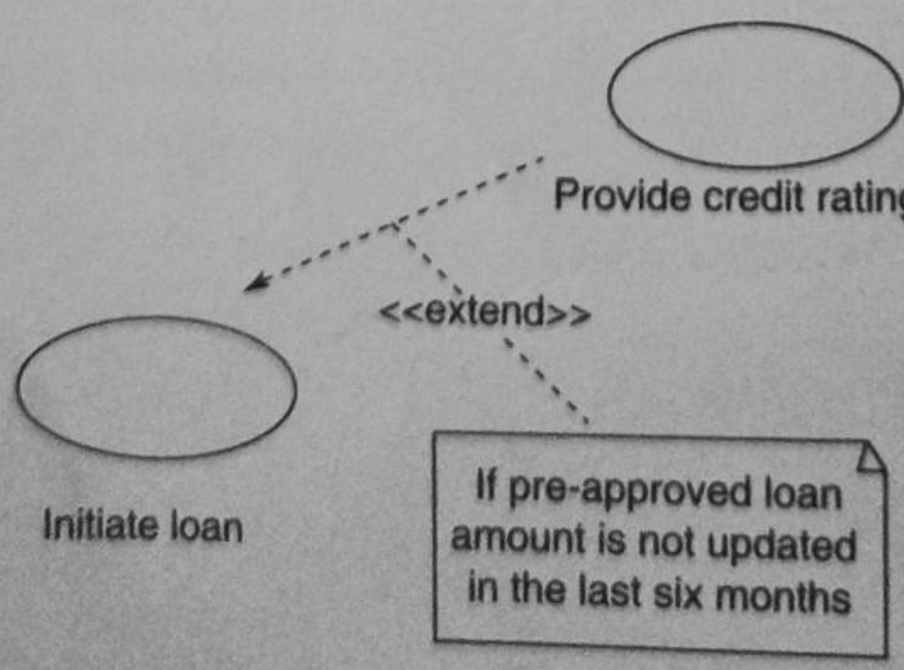

Figure 2-6 Extends relationship.

(c) *Uses relationship* among use cases exists, if one of the use cases (including use case) always includes another use case (included use case), as depicted in Figure 2-7. It is also known as *include relationship*. The Provide Customer Information use case is always used by the Initiate Loan use case, because as per loan initiation requirement the customer information is always processed. The Initiate Loan use case is the including use case, and the Provide Customer Information use case is the included one. It is represented as a dashed arrow pointing from the including to the included use case. It makes sense to extract a use case as an "included" use case only if that use case is used by more than one use case.

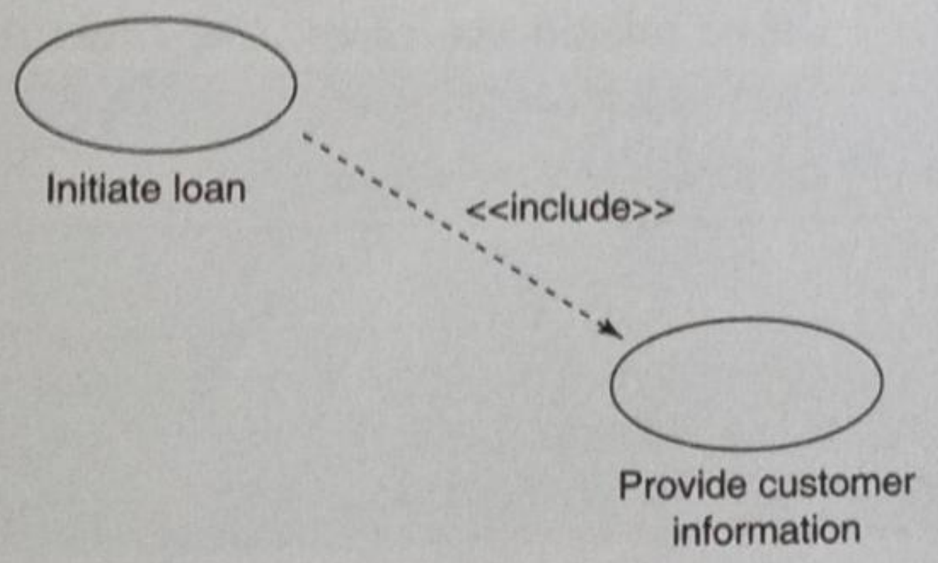

Figure 2-7 Uses/include relationship.

A use case diagram does not provide a comprehensive description of use case functionality. Typically, a textual description of the use case functionality is also documented, and is commonly referred to as *use case specification* (UCS) document. It primarily consists of actors, preconditions, post-conditions, primary flow and alternate flows of a use case. Table 2-1 provides a template for the details that are generally captured as part of a UCS document.

Table 2-1 Elements of use case specification

UCS elements	Description
Actors	List of participating actors in the use case
Description	A brief description of the use case
Preconditions	List of prerequisites to start the use case
Post-conditions	List of things which are outcome of the use case
Priority	Business criticality is quantified and presented in a relative manner
Trigger	The event that initiates the use case
Primary scenario	The primary flow of the use case. Primary scenario is the most frequently occurring scenario in the entire use case
Alternate scenario # number	List of alternate flows. These typically occur less frequently than the primary flow

(*Continued*)

Table 2-1 *(Continued)*

Field definitions	The "significant" nouns are collected from various scenarios and are jotted down along with the specifications such as size of the field, type of the field, possible values and default values. These act as an input to design the data models in the successive phase
Exceptions	List of exceptional flows
Assumptions	Assumptions, if any
User Interface	Reference to the prototype(s)/wireframe(s)
Related use cases	List of related use cases. Use cases that are included or having extension are listed here
Nonfunctional requirements	Capture NFRs specific to this particular use case

Tt UCS documents are prepared typically using a standard document editor like Microsoft Office Word®. UML tools like Rational Rose, Together® Architect are used to attach this document with the respective use cases modeled with these tools.

Primary or alternate scenarios can be sometimes quite complex and be supported with even better and lucid explanations in the form of activity diagrams. Activity diagrams are UML diagrams that are similar to flow charts, and are good at visually representing the flow of activity of a use case. Designing activity diagrams is purely optional and typically exercised only for those scenarios/use cases that are really complex to comprehend in a textual representation.

2.4.2 User Prototypes

The human user actors identified in the previous steps need to interact with the system. This boils down to creation of sophisticated User Interfaces (UI), but typically starts with creating prototypes or *wireframes* of these user interfaces. Wireframes are the bare minimum visual elements laid down to depict the basic user interface design.

User prototypes or wireframes serve the following purposes:

- They can be used to allow business users to validate the user interface, which helps in managing expectations and getting "buy-in" from them to avoid last minute surprises.
- UI designers can attain a comfort level from the look-and-feel perspective of wireframes. It also serves as a blueprint that helps in managing the UI design consistency.
- Business analysts and UI designers are able to find out usability requirements like navigability, which is one of the key nonfunctional requirements.
- Business analysts get a better aid in understanding the exceptions and variations in use cases.

Tt Dreamweaver is one of the favorite tools of wireframe designers. Microsoft Visio® or Adobe® Illustrator can also be used to create wireframes.

The following are some of the points that need to be taken care while designing the user prototypes:

- Tools like Microsoft PowerPoint® can be used for prototyping, but it's a good practice to use HTML as the basis for prototyping.
- Sample data used in prototype should be indicative of real data.
- Prototypes should concentrate on the primary scenario of a use case. If alternate scenarios are critical, then only it is advisable to consider them for prototyping.
- A few tools, like Dreamweaver, have the capability of creating wireframes that can be further converted into a skeletal code.

In the era of Web 2.0, where collaboration is the key, annotated prototypes are also used. A team developing prototype can easily share and add notes to it. One such prototyping collaboration tool is Protonotes.[2] Tools such as Wiki and shared portal are also used for requirements gathering in a collaborative mode.

2.4.3 Nonfunctional Requirements

In contrast to the functional behavior of an enterprise application, there are characteristics and constraints associated with the application, termed as *nonfunctional requirements* (or NFRs). These characteristics and constraints are of paramount importance for any enterprise application.

Please rernember, the NFRs are not accidental; they must be captured, documented and considered very carefully during development of the system.

Referring to the sample system LoMS, it should be able to process 1000 loan applications in a day. It should be able to support 20 percent of average growth in the volume of loan applications every year. It should be supported by Internet Explorer version 6 and 7. Context-sensitive help of the system should be available. It should have a maintainable code base. It should be able to avert faults due to the underlying applications servers and database servers. It should not be prone to injection attacks. These are a few of the examples depicting the characteristics and constraints expected of LoMS, in particular, and of any enterprise application, in general.

The NFRs of an enterprise application have a very high impact on the way the system has to be architected, designed and deployed. In fact, NFRs play a pivotal role in the validation of the architecture and design of enterprise applications.

A customary question is "What is to be captured as NFRs?". There are myriad of them, but the typical ones that need to be captured in an NFR document are as depicted in Table 2-2.

---

[2] You can refer to Protonotes at http://www.protonotes.com/

Table 2-2 Key NFRs

Key NFRs	Description
Performance requirements	Constraints like throughput, response time and refresh time under normal and peak loads
Usability requirements	Effective use of screen real estate, navigability, localization and internationalization, browser support and accessibility requirements
Scalability requirements	Resource utilization, data storage requirements and projected growth metrics could be used for better capacity planning and management.
Interface requirements	Dictate the mechanisms followed for application's interoperability or integration with existing enterprise applications
Operating requirements	Security, maintainability and reliability requirements have to be laid down as part of operating requirements. Recovery time objective (RTO), recovery point objective (RPO) and mean time between failures (MTBF) are few of the metrics captured for reliability operating requirements
Lifecycle requirements	Testability, reusability, portability and installability requirements
Regulatory requirements	Legal and licensing requirements

Let us now discuss these nonfunctional requirements.

Performance requirements

Performance requirements are the performance constraints that need to be identified for each critical use case of the system under a given set of conditions. There are three key determinants of the performance of any enterprise application:

(a) Workload of a system
(b) The server software, hosting the enterprise application
(c) Underlying hardware of the enterprise application

The number of user requests to the system, think time of the user to perform an action in the system and arrival rate of requests are the key performance factors in the specification of the workload. This eventually boils down to the system throughput. *Throughput* is defined as the amount of work that a system can perform in a given time period. The server software typically involves the operating systems, application servers and database servers. The underlying hardware of the application typically involves processors, disks, memory and networks. An application uses resource utilization as one of the key performance metrics.

Performance requirements of an enterprise application can be categorized into two perspectives:

(a) *Business perspective* captures the requirements like types of transactions, types of users, their usage pattern and request arrival pattern. These are required to measure the workload and throughput of the system.

(b) *Technology perspective* captures the requirements like operating environment details, system and interface details and performance boundaries with respect to the external systems. These are required to measure the resource utilization and the response time of the system.

In case of LoMS, the performance requirements state that the system should be able to process a peak of 60 and an average of 40 concurrent transactions per second. It should be able to support 250 concurrent user sessions and process 1000 loan applications in a day. The maximum page-refresh time involving online transactions should not be more than 5 seconds.

User interaction transactions, batch transactions and report generation transactions are three different types of transactions. The metrics for throughput and response time for different types of transactions should be captured separately.

The nonfunctional requirements gathered from performance perspective of enterprise applications are used to perform various performance related tests like load test, volume test, stress test and endurance test on applications. Procedures followed to test and tune the performance and how enterprise applications conform to the performance requirements are elaborated in the later chapters.

Usability requirements

Usability requirements dictate the user experience. This is one of the key NFRs that can make or mar an end user's perception of the application. These are one of the most important requirements from the perspective of acceptance of an application by end users.

In case of LoMS, the usability requirements state that the system requires customers and prospects to be self guided through the loan initiation process. For existing customers, the loan information should be pre-populated, if available. For all of the human actors, the system should facilitate easy navigation combined with easy workflow. The LoMS application should support internationalization as it needs to be localized as per the user base. Context-sensitive help should be available to the user.

The following simple thumb rules can really help in smooth acceptance of enterprise application by the end user:

- Proper usage of the screen real estate is very important. Logical grouping of the items on the user screen or placing the relevant contents at the appropriate place helps the end user to understand the overall screen in a better manner and in turn decreases the think time of the user to take an action on the screen.
- Easy to use and intuitive navigability is the key to usability of an application. Having excessive number of dialog boxes is not a good practice. Too many switches between mouse and keyboard should be avoided. The tab sequences should be defined appropriately, in case application supports user inputs through keyboard.
- Wherever possible, provide auto populate and auto complete of data in forms' fields to save the time of user. This results in less number of data input mistakes.
- Interacting with an application in one's preferred language provides more "easy-to-use" and "personalized" touch to the end user. Careful internationalization of the enterprise application is very important so that it can be localized as per the end users' chosen locale and language requirements.
- Support for popular browsers is also necessary.