Czech University of Life Sciences in Prague

Faculty of Economics and Management

Department of Information Engineering

Diploma Thesis

Design of IS for Travel Expense Management Automation

Author: Mirakmal Kobilov

Supervisor: Doc. RNDr. Dana Klimešová, CSc.

.

Declaration

I declare that I have worked on my diploma thesis titled “Design of IS for Travel Expense Management Automation” by myself and I have used only the sources mentioned at the end of the thesis.

In Prague on ……………………………………

signature

Acknowledgement

I take this opportunity to express my deep gratitude and regards to my supervisor Doc. RNDr. Dana Klimešová, CSc. for the help in the preparation of my thesis. My supervisor gave me great help in the research, not only helped me to correct errors, but also gave me valuable advice. I would like to take this opportunity to express my heartfelt gratitude and deepest respect.

I would like to thank to my family and friends for all their supports.

Design of IS for Travel Expense Management Automation

Návrh informačního systému Řízení cestovních nákladů

Abstract

Today’s challenging corporate environment often requires that professionals travel more often and more widely to expand and enhance business operations. Organizations’ push for efficiency necessitates that these traveling professionals be more cost-productive without sacrificing accuracy, compliance, or results. Financial process automation is becoming essential for efficient corporate travel, as is the ability to access that automation on the go, making mobile functionality a requirement for maintaining competitive advantage. This need is especially visible in the area of expense reporting, when corporate travelers must organize and record receipts for all their business spend. Automation of Travel and Expense Management (TEMA), with its mobile integration and intelligent features, allows traveling employees to trade the inefficiencies of outdated expense spreadsheets and receipt handling for speed, productivity, and compliance. This thesis discusses designing and implementation of TEMA in the mid and small size companies. Project contains 5 parts: Literature Review which contains discussion of Designing Information Systems, Object-Oriented Development Methodology, Travel Expense Management Automation, Software Development and Implementation, Conclusion.

In project realization, one of the high level programming languages C# was used based on Object Oriented Methodology. Within C#, classes of .NET Framework (which was created and supported by Microsoft) were utilized. The reason that C# and .NET was chosen is that this technologies are increasing in usage day by day surpassing other object oriented programming languages. C# is becoming most popular programming language in System Development.

Keywords: Information Systems, Information System Design, Travel Expense, Management Automation, OOP, UML, Expense Reporting.

Abstrakt

Dnešní náročné korporátní prostředí často od odborniků vyžaduje, aby vzhledem k zkvalitňování a rozšiřování podnikání čím dál, tím víc cestovali. Organizace zvyšují požadavky na efektivnost, produktivitu a snižování nákladú , při zachování stejné kvality. Tato potřeba je obzvlášť patrná v oblasti vykazování výdajů, kdy zaměstnanci na služebních cestách musí pečlivě uchovávat a zaznamenávat účetní doklady pro všechny výdaje. Automatizace řízení cestovních nákladů (TEMA), s mobilní integrací inteligentních funkcí umožňuje cestujícím zaměstnancům nahradit neefektivní zastaralý systém výkazů cestovních nákladů a účtenek za rychlí a efektivní systém. Tato práce popisuje vývoj a implementaci TEMA ve středních a malých podnicích. Projekt obsahuje 5 částí: přehled literatury, uvod do automatizace řízení cestovních nákladů , funkční požadavky a metodiku , vývoj a implementace softwaru, závěr.

Klíčová slova: Cestovní Výdaje, Automatizace řízení, OOP, UML, Vykazování nákladů, Navrh informacniho systemu.

Contents

1. Introduction

Corporate business travel has been long-term necessity in the greater scheme of organizational growth. Business travel is linked to effective customer development, supplier relationships and maintaining a corporate presence on the global and local stage. In fact, expenses related to travel accounts for 7-13% of the average company's total budget (according to Aberdeen Group reports 2014), forcing enterprises across the world to manage these expenses in the way that is consistent with bottom line growth. Companies continue their strategic approach towards expense management as we move into new decade, the challenges and opportunities for organizations of all sizes have changed dramatically over the past few years. Today many organizations are focused on finding practical ways to improve their bottom lines, and as the second-largest operating expense in most organizations, travel expense (T&E) is a focal point. As a result, financial executives are looking for more cost-effective and efficient approaches to managing travel expenses.

More and more companies are leaving classic spreadsheet style expense reporting systems and striving to improve their travel expenses by absorbing automated travel expense reporting technologies. Therefore, the importance of Automated Travel Expense Reporting is higher than ever before.

Travel expenses typically represent the second-highest controllable corporate annual expense, and travel bookings through TMC (Travel Management Company)es are only part of the picture. A mature travel-expense management strategy includes many elements: buying travel, traveling, travel reimbursement, processing and analyzing travel spend, and using the data gathered during the cycle to buy future travel more efficiently. The Aberdeen Group reports (2013) that nearly 70% of organizations globally view travel-expense management as a strategic function. In 2015, 65% of these companies indicated they plan to improve their expense-management processes. These companies have realized that viewing the entire travel and expense program as an integrated system enables their organizations to better control travel spend and save money. Considering this, many leading IT companies produce solutions for better travel expense management, but these solutions are not for all: for many companies there are issues like unaffordability, lack of service support, location mismatch and so on. Most of technologies which are aimed for better travel expense management are not affordable for mid and small sized companies.

Realizing these problems, I see the need to study and analyze the current state of TEMA in especially mid and small size companies. Finally, I try to suggest simple and useful Travel Expense Reporting Application prototype as a possible solution for mid and small sized companies.

In this thesis Travel Expense Management application has been designed using one of the high level Object-Oriented language - C# with .NET framework tools such as ASP.NET, Entity Framework alongside with Relational Database - MSSQL. Above-mentioned Back-End technologies demonstrated with modern Client-Side technologies such as HTML5, CSS3 and jQuery framework. I want to implement his simple TEMA solution for mid and small size companies to make this solution affordable for them, to make their business activities more efficient and to cut their operational costs. During my time in university, I learned above-mentioned technologies on theoretical basis. Also during my internship in 2 companies, I tried to learn how those technologies are used in practice. In this thesis I tried to combine my theoretical knowledge and practical skills so that to create this project.

2. Objectives and Methodology

Goal

The Goal of this thesis is to study and analyze the theory of designing information systems and to discuss current state of Travel Expense Automation and to develop information system for travel expense management and to provide mid and small size companies with simple affordable solution to make them work more efficiently and cut their costs which are related to their business trips.

Partial goals of this thesis:

- to study and analyze designing of information systems
- to have closer look at current state of travel expense management automation in the companies;
- to identify problems with implementing automation of travel expense and explain, analyze benefits of implementing information system for travel expense management;
- to design possible solutions/best practices for companies.

As a student, my objectives in this thesis is to learn Object-Oriented Programming better and to absorb better view of system design and development. Today OOP is most popular methodology in designing and developing information systems, the role of OOP will only increase and I think, by learning OOP properly, I can become better IT professional.

Methodology

Methodology of this thesis is based on analysis of information that was collected from available sources and knowledge to study and analyze Designing of Information Systems and to find out current state of Travel Expense Management Automation in different types of companies. Practical part aims to develop web-based application (prototype) using OOP language (C# and .NET Framework) with relational database (MSSQL). Requirements and Methodology of the project will be discussed in chapter 6. Also, Software Development and Implementation will be analyzed in chapter 6. Finally, conclusion and his recommendations are given in chapter 7.

Application will be web-based program which will have central server. The reason that it was intended to be web based is that in the future most of business applications are supposed to migrate to Cloud based systems. Migrating from web-based application to cloud based system will be easy and less costly.

3. Theoretical Background

This chapter is dedicated to study and analysis of designing information systems; chapter includes introduction to information systems, types of information systems. Then following are discussed: main system development methodologies, advantages and disadvantages of those methodologies. Finally, Object Oriented methodology and Unified Modelling Language in designing information systems are discussed.

3.1. System and its components

Starting talking about System Design, first we need to understand definition of Data, Information and Knowledge. Abdalla Uba Adamu, professor of National University of Nigeria, in his book ("Information System Design and Programming", 2006) describes data as follows - "*Data is raw facts, figures, images or sounds collected from observations or recordings about events, objects or people, which can be stored on a manual or computer-based medium e.g. employee's name and number, number of hours worked in a week, inventory part numbers, or sales orders.*" It is important to distinguish between data and information. Data has little meaning/value in its own right, it only has meaning when it is processed and put into context as information. The information has a value in decision making while data does not have. Information brings clarity and creates an intelligent human response in the mind.

Information is defined as "*data that is processed for a particular purpose*" - (Curtis and Cobham, 2002) or as "*an answer to a specific question*" - (Paresi, 2000). A common definition of information is that information is data that have been processed and presented in a useful format that will enable an individual to gain knowledge in order to be able to make a decision. Information can generally be considered as an entity that is responsible for reducing uncertainty about a particular state or event, as pointed out by (Lucas, 1985).

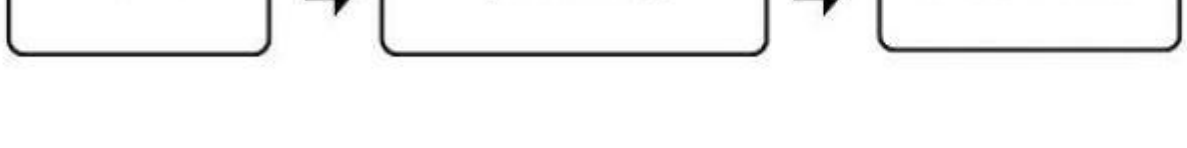


Figure 1: Data transformation into Information.

We can conclude that the difference between data and information is data is raw material, while information is data stored on computer systems.

Although information is useful resource for individuals and organisations not all information can be considered useful. The differences between ‘good’ and ‘bad’ information can be identified by considering whether or not it has some or all of the attributes of information quality. Attributes can be related to the timing, content and form of the information. Knowledge depends on personal experience and is on a pragmatic level.

Based on Anthony's classification of Management, information used in business for decision-making is generally categorized into three types:

- *Strategic Information*: Strategic information is concerned with long term policy decisions that defines the objectives of a business and checks how well these objectives are met. For example, acquiring a new plant, a new product, diversification of business etc., comes under strategic information.
- *Tactical Information*: Tactical information is concerned with the information needed for exercising control over business resources, like budgeting, quality control, service level, inventory level, productivity level etc.
- *Operational Information*: Operational information is concerned with plant/business level information and is used to ensure proper conduction of specific operational tasks as planned/intended. Various operator specific, machine specific and shift specific jobs for quality control checks comes under this category.

Merriam Webster defines knowledge in this way – “*Knowledge is the fact or condition of knowing something with familiarity gained through experience or association, acquaintance with or understanding of a science, art, or technique”*. Knowledge can also be defined as “the fact or condition of being aware of something and the range of one's information or understanding”.

Professor Ray R. Larson of the School of Information at the University of California, Berkeley, provides an *Information Hierarchy,* which is:

- Data - The raw material of information.
- Information - Data organized and presented by someone.
- Knowledge - Information read, heard, or seen, and understood.
- Wisdom - Distilled and integrated knowledge and understanding.

Scott Andrews' explains *Information Continuum* as follows:

- Data - A Fact or a piece of information, or a series thereof.
- Information - Knowledge discerned from data.
- Business Intelligence - Information Management pertaining to an organization's policy or decision-making, particularly when tied to strategic or operational objectives.

System

The term *system* is derived from the Greek word `s*ystema*' which means an organized relationship among functioning units or components. In other words, A system is an orderly group of independent components linked together according to a plan to achieve a common objective. Definitions of the *system* concept appear in several books (e.g. Langefors, 1995; Ackoff, 1981; Klir, 1991). These definitions can be synthesised as follows: "*a system is a collection of related elements organised into a whole to perform a particular function and/or reaching a goal*". Despite different definitions, the main characteristics of systems in our meaning are the fact that it is up to the observer of a system to view the system. In the broadest sense, a system is simply a set of components that interact to accomplish some purpose. They are all around us. For example, human body is a biological system. We experience physical sensations by means of a complex nervous system, a set of parts, including brain, spinal cord, nerves, and special sensitive cells under our skin, that work together to make us feel hot, cold, itchy, and so on.

An organization may also be viewed as a system where all the employees interact with each other and also with the employer to make the organization a functional unit. The organization also interacts with their customers to make a complete business system.

In our day to day life, we see many *business systems*. These businesses have varied objectives, which range from producing a notebook to producing aircraft. These systems have their information needs. It can be for maintaining the records for employee for their wages calculations, keeping track of their leave status, maintaining company's expenses, inquiries from customers in case the business provide some service, or for keeping track for some particular function. So maintaining data is an important and essential activity in any business. The overall data maintained constitutes what is known as Information system.

According to Joseph S. Valacich, Professor of Management Information Systems at the University of Arizona ("Essentials of Systems Analysis and Design", 2012), a system has nine *characteristics*: Components, Interrelated components, Boundary, Purpose, Environment, Interfaces, Constraints, Input, Output.

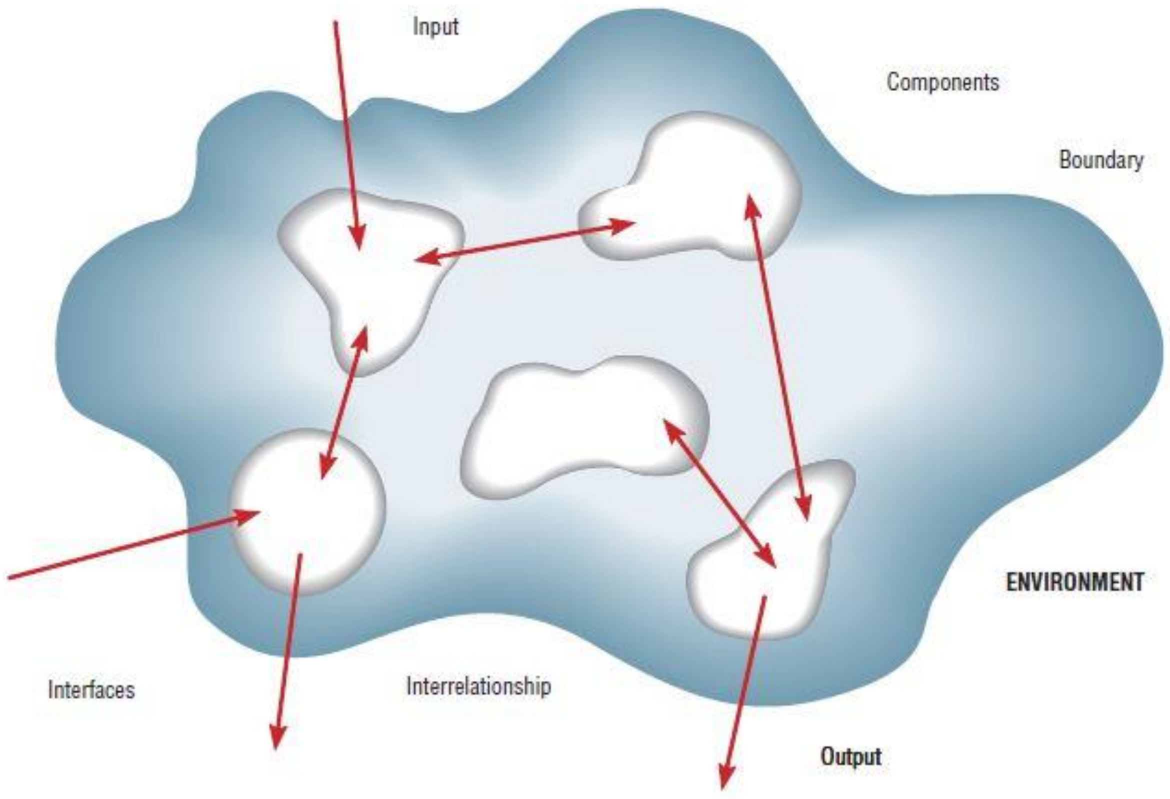


Figure 2: System characteristics

A system is made up of components:

- A *component* is either an irreducible part or an aggregate of parts, also called a subsystem. The simple concept of a component is very powerful. For example, just as with an automobile or a stereo system, with proper design, we can repair or upgrade the system by changing individual components without having to make changes throughout the entire system.
- The components are *interrelated;* that is, the function of one is somehow tied to the functions of the others. For example, the work of one component, such as producing a daily report of customer orders received, may not progress successfully until the work of another component is finished, such as sorting customer orders by date of receipt.
- A system has a *boundary*, within which all of its components are contained and which establishes the limits of a system, separating it from other systems. Components within the boundary can be changed, whereas systems outside the boundary cannot be changed.

- All of the components work together to achieve some overall *purpose* for the larger system: the system's reason for existing.
- A system exists within an *environment* - everything outside the system's boundary that influences the system. For example, the environment of a state university includes prospective students, foundations and funding agencies, and the news media. Usually the system interacts with its environment. A university interacts with prospective students by having open houses and recruiting from local high schools. An information system interacts with its environment by receiving data (raw facts) and information (data processed in a useful format).
- The points at which the system meets its environment are called *interfaces;* an interface also occurs between subsystems.
- In its functioning, a system must face *constraints*—the limits (in terms of capacity, speed, or capabilities) to what it can do and how it can achieve its purpose within its environment. Some of these constraints are imposed inside the system (e.g., a limited number of staff available), and others are imposed by the environment (e.g., due dates or regulations).
- A system takes input from its environment in order to function. People, for example, take in food, oxygen, and water from the environment as input. You are constrained from breathing fresh air if you're in an elevator with someone who is smoking.
- Finally, a system returns output to its environment as a result of its functioning and thus achieves its purpose. The system is constrained if electrical power is cut.

3.2. Introduction to Information Systems

Alan R. Hevner in his article (2004) which he published in University of South Florida gives explanation to information system in this way – "*An information system (IS) is a computerized database designed to accept, store, process, transform, make useful, and analyze data and to report results, usually on a regular, ongoing basis. It is often construed as a larger system including not only the database and the software and hardware used to manage it but also including the people using and benefiting from it and also including all necessary manual and machine procedures and communication systems*". More specifically, it is the study of complementary networks that people and organizations use to collect, filter, process, create and distribute data. Information system is the means by which data flow from one person or department to another and can encompass everything from interoffice mail and telephone links to a computer system that generates periodic reports for various users. Information systems serve all the systems of a business, linking the different components in such a way that they effectively work towards the same purpose.

Kroenke, David (2015) says: *"An information system (IS) is a group of components that interact to produce information*". The role of the Information systems to provide information to management which will enable them to make decisions which ensure that the organisation is controlled. According to authors of MIS tutorial in Tutorialspoint.com, Information System – "*An arrangement of people, data, processes and information technology that interact to collect, process, store and provide as output, the information needed to support an organisation.*"

The six components that must come together in order to produce an information system are:

- *Hardware*: The term hardware refers to machinery. This category includes the computer itself, which is often referred to as the central processing unit (CPU), and all of its support equipments. Among the support equipments are input and output devices, storage devices and communications devices.
- *Software*: The term software refers to computer programs and the manuals (if any) that support them. Computer programs are machine-readable instructions that direct the circuitry within the hardware parts of the system to function in ways that produce useful information from data. Programs are generally stored on some input / output medium, often a disk or tape.
- *Data*: Data are facts that are used by programs to produce useful information. Like programs, data are generally stored in machine-readable form on disk or tape until the computer needs them.
- *Procedures*: Procedures are the policies that govern the operation of a computer system. "Procedures are to people what software is to hardware" is a common analogy that is used to illustrate the role of procedures in a system.
- *People*: Every system needs people if it is to be useful. Often the most over-looked element of the system are the people, probably the component that most influence the success or failure of information systems. This includes "*not only the users, but those who operate and service the computers, those who maintain the data, and those who support the network of computers.*" (Kroenke, D. M. (2015). MIS Essentials. Pearson Education)
- *Feedback*: it is another component of the IS, that defines that an IS may be provided with a feedback (Although this component isn't necessary to function).

Data is the bridge between hardware and people. This means that the data we collect is only data, until we involve people. At that point, data is now information. Three activities provide the information that organizations need. These activities are Input, Processing and Output. '*Input*' consists of acquisition of the 'raw data', which is transformed into more meaningful packets of 'Information' by means of '*Processing*'. The processed information now flows to the users or activities also called as '*Output'*. The shortcomings are analyzed and the information is sent back to the appropriate members of the organization to help them evaluate and refine the input. This is termed as '*feedback*'. Examples of 'Information Inputs' would be Transactions, events which would undergo 'processing' in the form of sorting, listing, merging and updating resulting in 'outputs' such as detailed reports, lists and summaries.

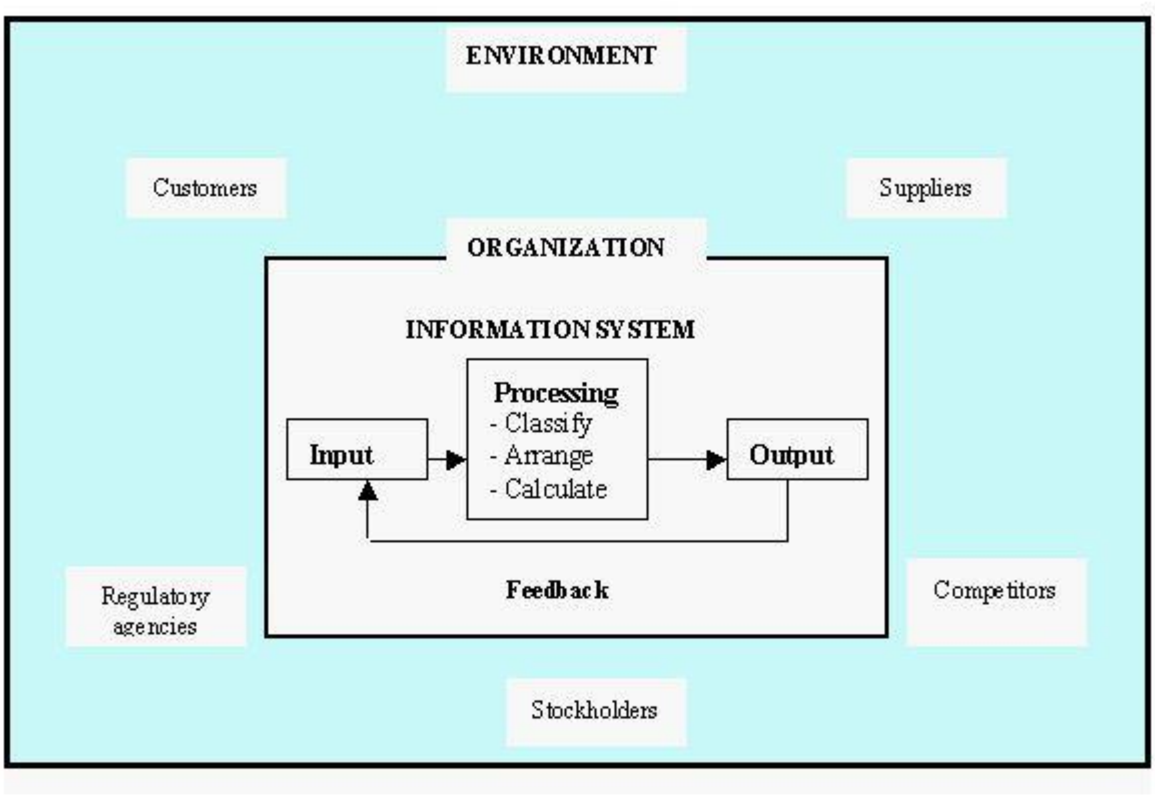


Figure 3: Activities that provide the information organizations need

A *computer information system* is a system composed of people and computers that processes or interprets information. The basic components of computer-based information systems are:

- *Hardware*- these are the devices like the monitor, processor, printer and keyboard, all of which work together to accept, process, show data and information.
- *Software*- are the programs that allow the hardware to process the data.

- *Databases*- are the gathering of associated files or tables containing related data.
- *Networks*- are a connecting system that allows diverse computers to distribute resources.
- *Procedures*- are the commands for combining the components above to process information and produce the preferred output.

Some authors, (for example O'Brien, J A. (2003). “Introduction to information systems: essentials for the e-business enterprise”. McGraw-Hill, Boston, MA) make a clear distinction between information systems, computer systems, and business processes. Information systems typically include an ICT component but are not purely concerned with ICT, focusing instead on the end use of information technology. Information systems are also different from business processes. Information systems help to control the performance of business processes. Information systems inter-relate with data systems on the one hand and activity systems on the other. An information system is a form of communication system in which data represent and are processed as a form of social memory. An information system can also be considered a semi-formal language which supports human decision making and action.

“*Information technology (IT) refers to the combination of hardware, software, and services that people use to manage, communicate, and share information*” – Gary B. Shelly and Harry J. Rosenblatt, (2012)

Information Technology (IT): The enabling mechanism which facilitates the processing and flow of information, as well as the technologies used in the physical processing to produce a product or provide a service. Includes telecommunications, computers and automation technologies, represents the technical perspective. Information technology has significantly expanded the power and potential of most information systems. Information technology is a contemporary term that describes the combination of computer technology (hardware and software) with telecommunications technology (data, image, and voice networks). Technology has created a data and information explosion in virtually all businesses. The ability of businesses to harness and manage this data and information has become a critical success factor in most businesses.

3.3. Types of Information Systems

Information systems differ in their business needs. Also depending upon different levels in organization information systems differ. Four major information systems are:

- Executive information systems.
- Transaction processing
- Management information system
- Decision support system

The "classic" view of Information systems found in the textbooks (Laudon, K.C. and Laudon, J.P. “Management Information Systems”, (2nd edition), Macmillan, 1988.) in the 1980s was of a pyramid of systems that reflected the hierarchy of the organization, usually transaction processing systems at the bottom of the pyramid, followed by management information systems, decision support systems, and ending with executive information systems at the top.

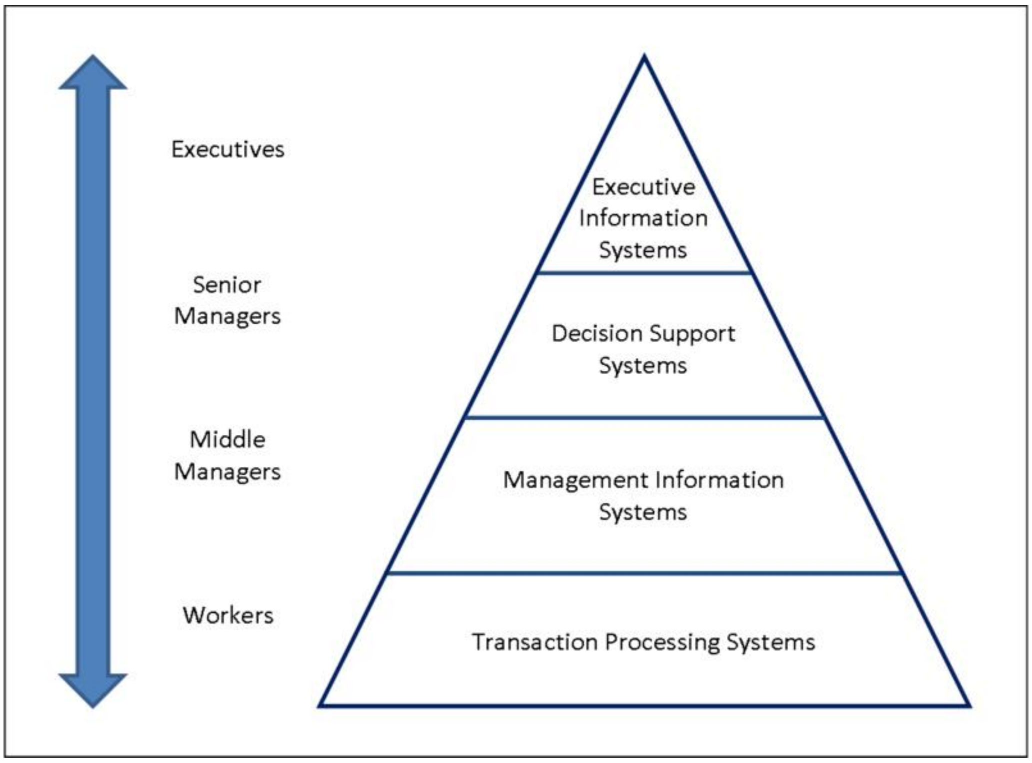

Executives
Senior Managers
Middle Managers
Workers
Executive Information Systems
Decision Support Systems
Management Information Systems
Transaction Processing Systems

Figure 4: Classic Information Systems Pyramid

Executive information system (EIS), also known as an executive support system (ESS). Executive support systems are intended to be used by the senior managers directly to provide support to non-programmed decisions in strategic management. It is commonly considered a specialized form of decision support system (DSS). These information are often external, unstructured and even uncertain. Exact scope and context of such information is often not known beforehand. Following are some examples of intelligent information, which is often the source of an ESS - external databases, technology reports like patent records, technical reports from consultants, market reports, confidential information about competitors, speculative information like market conditions, government policies, financial reports and information.

Decision support systems (DSS) are interactive software-based systems intended to help managers in decision-making by accessing large volumes of information generated from various related information systems involved in organizational business processes, such as office automation system, transaction processing system, etc. DSS uses the summary information, exceptions, patterns, and trends using the analytical models. A decision support system helps in decision-making but does not necessarily give a decision itself. The decision makers compile useful information from raw data, documents, personal knowledge, and/or business models to identify and solve problems and make decisions.

There are two types of decisions - programmed and non-programmed decisions. Programmed decisions are basically automated processes, general routine work, where these decisions have been taken several times and these decisions follow some guidelines or rules. For example, selecting a reorder level for inventories, is a programmed decision. Non-programmed decisions occur in unusual and non-addressed situations. For example, investing in a new technology is a non-programmed decision. Decision support systems generally involve non-programmed decisions. Therefore, there will be no exact report, content, or format for these systems. Reports are generated on the fly.

Following are the components of the Decision Support System - *Database Management System (DBMS), Model Management System, Support Tools*.

Management information system (MIS) focuses on the management of information systems to provide efficiency and effectiveness of strategic decision making. The concept may include systems termed transaction processing system, decision support system, expert system, or executive information system. Management information systems (plural) as an academic discipline studies people, technology, organizations, and the relationships among them ("What is Management Information Systems?". May's Business School. Archived from the original on May 9, 2015). This definition relates specifically to "MIS" as a course of study in business schools.

Many business schools (or colleges of business administration within universities) have an MIS department, alongside departments of accounting, finance, management, marketing, and may award degrees (at undergraduate, master, and doctoral levels) in Management Information Systems.

The three components of MIS provide a more complete and focused definition, where *System* suggests integration and holistic view, *Information* stands for processed data, and *Management* is the ultimate user, the decision makers.

MIS means a system for processing data in order to give proper information to the management for performing its functions. The goals of an MIS are to implement the organizational structure and dynamics of the enterprise for the purpose of managing the organization in a better way and capturing the potential of the information system for competitive advantage.

Following are the basic objectives of an MIS - *Capturing Data*, *Processing Data, Information Storage*, *Information Retrieval, Information Propagation*.

Transaction Processing System (TPS) processes business transaction of the organization. Transaction can be any activity of the organization. Transactions differ from organization to organization. For example, take a railway reservation system. Booking, canceling, etc are all transactions. Any query made to it is a transaction. However, there are some transactions, which are common to almost all organizations. Like employee new employee, maintaining their leave status, maintaining employees accounts, etc. This provides high speed and accurate processing of record keeping of basic operational processes. These include calculation, storage and retrieval. Transaction processing systems provide speed and accuracy, and can be programmed to follow routines functions of the organization.

Although the pyramid model remains useful, since it was first formulated a number of new technologies have been developed and new categories of information systems have emerged, some of which no longer fit easily into the original pyramid model. Every day new enterprise applications are developed which are designed for the sole purpose of promoting the needs and objectives of the organizations. Enterprise applications provide business-oriented tools supporting electronic commerce, enterprise communication and collaboration, and web-enabled business processes both within a networked enterprise and with its customers and business partners.

Basically these applications intend to model the business processes, i.e., how the entire organization works. These tools work by displaying, manipulating and storing large amounts of data and automating the business processes with these data.

Other types of Enterprise Applications (Systems):

- Enterprise Resource Planning (ERP)
- Customer Relationship Management (CRM)
- Knowledge Management Systems (KMS)
- Content Management System (CMS)
- Business Intelligence System (BIS)
- Enterprise Application Integration (EAI)
- Business Continuity Planning (BCP)
- Supply Chain Management (SCM)

3.4. System Development Methodologies

There are various information systems development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as “Software Development Process Models” (e.g. Waterfall model, Agile model, RAD model, Spiral model, Prototype model etc.). Each process model follows a particular life cycle in order to ensure success in process of software development.

System Development Life Cycle

“Systems development life cycle (SDLC) is a series of phases to plan, analyze, design, implement, and support an information system” – Gary B. Shelly and Harry J. Rosenblatt, 2012. The systems development life cycle (SDLC) is the traditional systems development method used by most organizations today. The systems development life cycle (SDLC) is central to the development of an efficient information system. The SDLC is a structured framework that consists of sequential processes by which information systems are developed. These processes, in turn, consist of well-defined tasks. Some of these tasks are present in most projects, whereas others are

present in only certain types of projects. That is, large projects typically require all the tasks, whereas smaller development projects may require only a subset of the tasks.

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

Most authors of books related to information system design agree that there are following six phases in every software development life cycle model:

- Requirement gathering and analysis
- Design
- Implementation or coding
- Testing
- Deployment
- Maintenance

1) *Requirement gathering and analysis*: Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stakeholders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

2) *Design*: In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

3) *Implementation / Coding*: On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

4) *Testing:* After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase all types of functional testing like unit testing, integration testing, system testing, acceptance testing are done as well as non-functional testing are also done.

5) *Deployment*: After successful testing the product is delivered / deployed to the customer for their use.

As soon as the product is given to the customers they will first do the beta testing. If any changes are required or if any bugs are caught, then they will report it to the engineering team. Once those changes are made or the bugs are fixed then the final deployment will happen.

6) *Maintenance*: Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a *linear-sequential life cycle model*. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of model is basically used for the for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model the testing starts only after the development is complete. In waterfall model phases do not overlap.

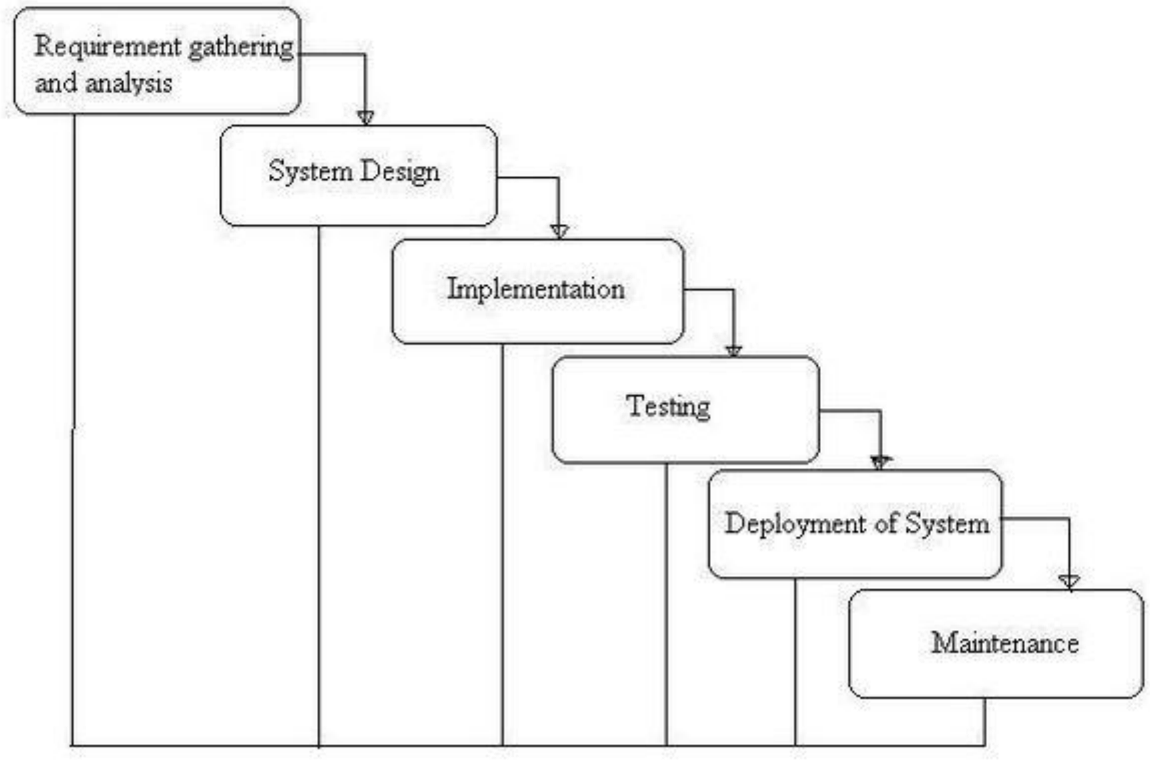


Figure 5: Waterfall Model

Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

Very less customer enter action is involved during the development of the product. Once the product is ready then only it can be demoed to the end users. Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everywhere from document till the logic.

Agile Methodology

Professors of Rutgers University Kenneth E. Kendall and Julie E. Kendall in their book ("Systems Analysis and Design", eighth edition, 2011) describe Agile methodology as follows – "*Agile software development (Agile) is a collection of software development methodologies that promote adaptive planning, evolutionary development and delivery, continuous improvement, and a time-boxed period of time to complete a body of work*". Software development is dynamic by nature, and Agile encourages rapid and flexible response to change. Because adaptability is central to its conceptual framework, teams using this approach are well-equipped to respond to changes throughout the development cycle.

In addition to being a collection of methodologies, Agile software development also promotes a different way of thinking or mindset about how to build things and evolve processes to deliver continuous improvement. Agile facilitates information-sharing amongst teammates, where everyone has a say on processes and practices and decisions are made together as a team.

Concepts of incremental and adaptive software development processes date back as early as the 1950s, with growth and progress from a small vocal minority through the 1980s. It was not until the 1990s, when an assortment of similar lightweight software development methods emerged in reaction to waterfall-oriented methods, that Agile started to gain some traction. The real watershed moment for the Agile movement was the publication of The Manifesto for Agile Software Development in 2001 by a group of 17 software developers, who met to discuss the collection of lightweight development methods, which is now referred to as Agile methods.

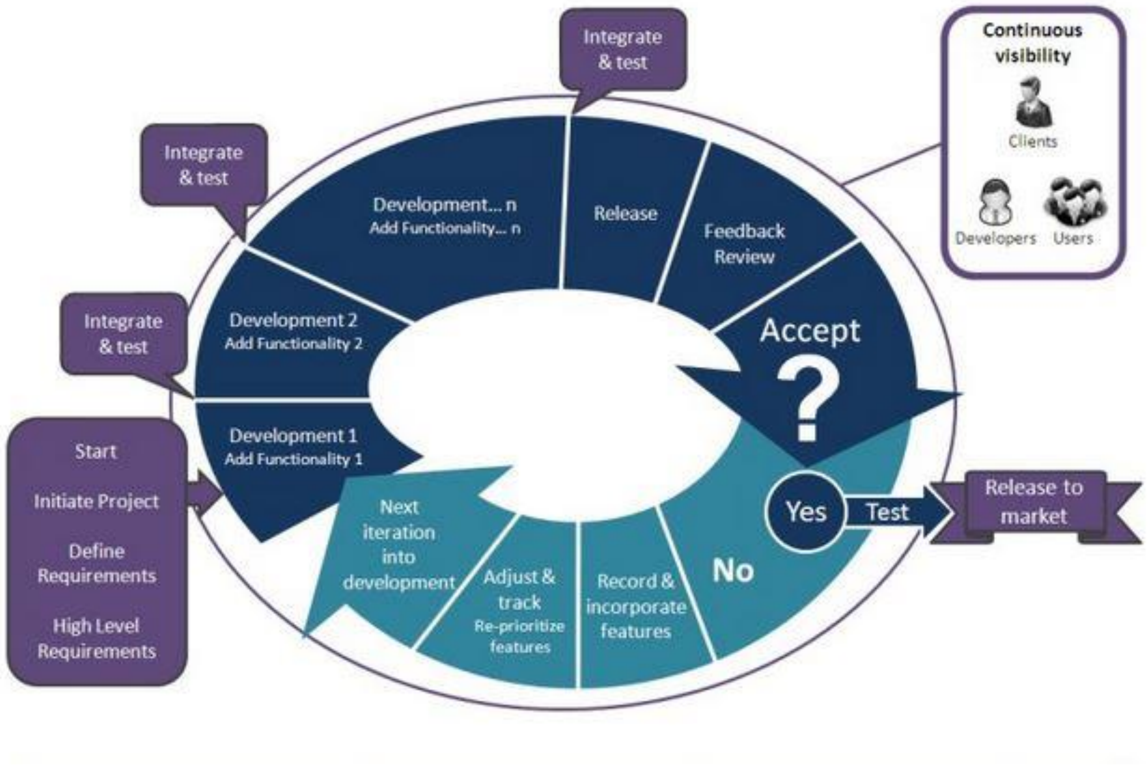


Figure 6: Agile Model

Principles. The authors of the Agile Manifesto also agreed upon the following 12 principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within the development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.

- Simplicity — the art of maximizing the amount of work not done — is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Example: Google is working on project to come up with a competing product for MS Word, that provides all the features provided by MS Word and any other features requested by the marketing team. The final product needs to be ready in 10 months of time. Let us see how this project is executed in Traditional and Agile methodologies.

In traditional Waterfall model:

- At a high level, the project teams would spend 15% of their time on investigation and analysis (1.5 months)
- 20% of their time on design (2 months)
- 40% on coding(programming) (4 months) and unit testing
- 20% on System and Integration testing (2 months).
- At the end of cycle, project may have 2 weeks of User Acceptance testing by mark-g teams.
- In this approach, the customer does not get to see the end product until the end of the project, when it becomes too late to make significant changes.

Figure 7 shows how these activities align with project schedule in traditional software development.

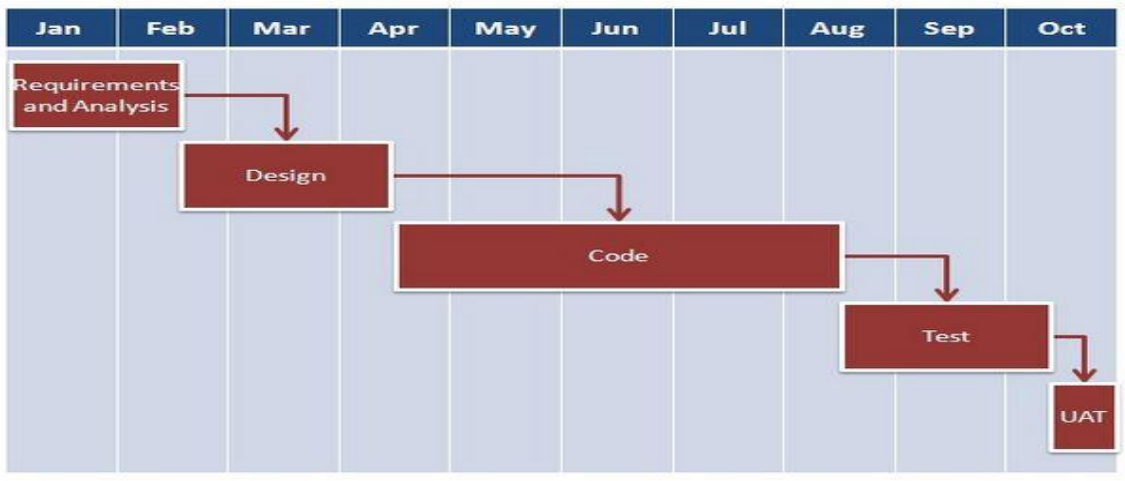


Figure 7: Development Cycle of Google’s project in Waterfall model.

With Agile development methodology:

- In the Agile methodology, each project is broken up into several 'Iterations'.
- All Iterations should be of the same time duration (between 2 to 8 weeks).
- At the end of each iteration, a working product should be delivered.
- Rather than spending 1.5 months on requirements gathering, in Agile software development, the team will decide the basic core features that are required in the product and decide which of these features can be developed in the first iteration.
- Any remaining features that cannot be delivered in the first iteration will be taken up in the next iteration or subsequent iterations, based on priority.
- At the end of the first iterations, the team will deliver a working software with the features that were finalized for that iteration.
- There will be 10 iterations and at the end of each iteration the customer is delivered a working software that is incrementally enhanced and updated with the features that were shortlisted for that iteration.

The iteration cycle of an Agile project is shown in the image below.

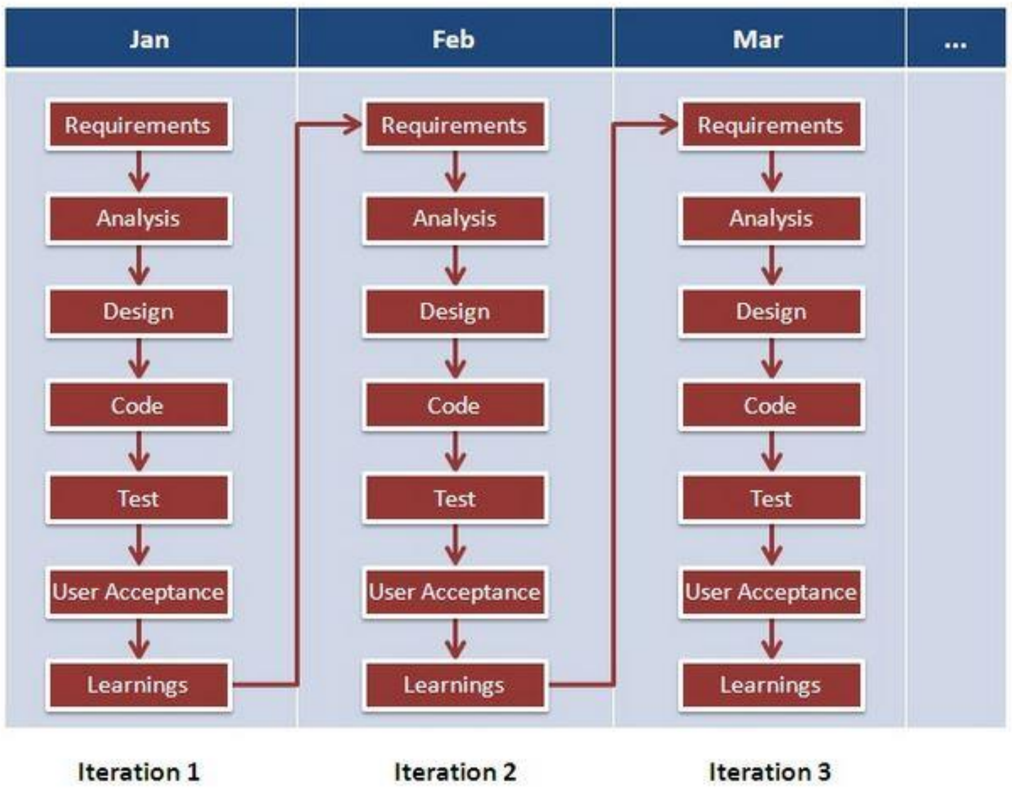


Figure 8: Development Cycle of Google's project in Agile model.

This approach allows the customer to interact and work with functioning software at the end of each iteration and provide feedback on it. This approach allows teams to take up changes more easily and make course corrections if needed. In the Agile approach, software is developed and released incrementally in the iterations.

4. Object-Oriented Methodology

An object-oriented approach to systems development combines data and methods into single entities called objects (Hoffer *et al*, 1999). The objects collaborate with each other through the sending of messages to request services, for example a request for a report (Rowley, 1998). Object-oriented methods, as argued by (Rowley, 1998) include, Object-Oriented Analysis (OOA) and Object-Oriented Analysis and Design (OOAD). In both methods, objects and classes are defined at the beginning, followed by a definition of relationships to put objects and class together, hence forming a system-wide view. Object detail is added into the system by giving specifications on attributes, methods and the object life history (Rowley ,1998).

Object-oriented development is based on a fundamentally different view of computer systems than that found in traditional SDLC development approaches. Traditional approaches provide specific step-by-step instructions in the form of computer programs, in which programmers must specify every procedural detail. These programs usually result in a system that performs the original task but may not be suited for handling other tasks, even when the other tasks involve the same real-world entities. For example, a billing system will handle billing but probably will not be adaptable to handle mailings for the marketing department or generate leads for the sales force, even though the billing, marketing, and sales functions all use similar data such as customer names, addresses, and current and past purchases. An object-oriented (OO) system begins not with the task to be performed, but with the aspects of the real world that must be modeled to perform that task.

Professor Alan Dennis (Indiana University), Barbara Wixom (University of Virginia) and Roberta M. Roth (University of Northern Iowa) in their book ("System Analysis and Design", 2012) describe object-oriented model as follows: "*The field of systems design now incorporates object-oriented concepts and techniques, by which a system is viewed as a collection of self-contained objects that include both data and processes. Objects can be built as individual pieces and then put together to form a system, leading to modular, reusable project components*". In 1997, the

Unified Modeling Language (UML) was accepted as the standard language for object development.

The object-oriented approach views a system as a collection of self-contained objects, including both data and processes. Systems analysis and design methodologies are either data-centric or process-centric. Until the mid-1980s, developers had to keep the data and processes separate in order to build systems that could run on the mainframe computers of that era. Due to the increase in processor power and the decrease in processor cost, object-oriented approaches became feasible. Consequently, developers focused on building systems more efficiently by enabling the analyst to work with a system's data and processes simultaneously as objects. These objects can be built as individual pieces and then put together to form a system. The beauty of objects is that they can be reused over and over in many different systems and changed without affecting other system components. Although some authors feel that the move to objects will radically change the field of systems analysis and design and the SDLC, the incorporation of objects as an evolving process in which object-oriented techniques are gradually integrated into the mainstream SDLC.

4.1. Classes and Objects. Inheritance, Polymorphism and Encapsulation

A *class* is the general template that is used to define and create specific *instances*, or objects. Every object is associated with a class. For example, all of the objects that capture information about patients could fall into a class called Patient, because there are attributes (e.g., names, addresses, and birth dates) and methods (e.g., insert new instances, maintain information, and delete entries) that all patients share. An *object* is an instantiation of a class. In other words, an object is a person, place, event, or thing about which we want to capture information. If we were building a sales system for an RV dealer, classes might include vehicle, customer, and offer. The specific customers like Jim Maloney, Mary Wilson, and Theresa Marks are considered instances, or objects, of the customer class. Each object has *attributes* that describe information about the object, such as a customer's name, address, e-mail, and phone number. The state of an object is defined by the value of its attributes and its relationships with other objects at a particular point in time. For example, a vehicle might have a state of "new" or "pre-owned."

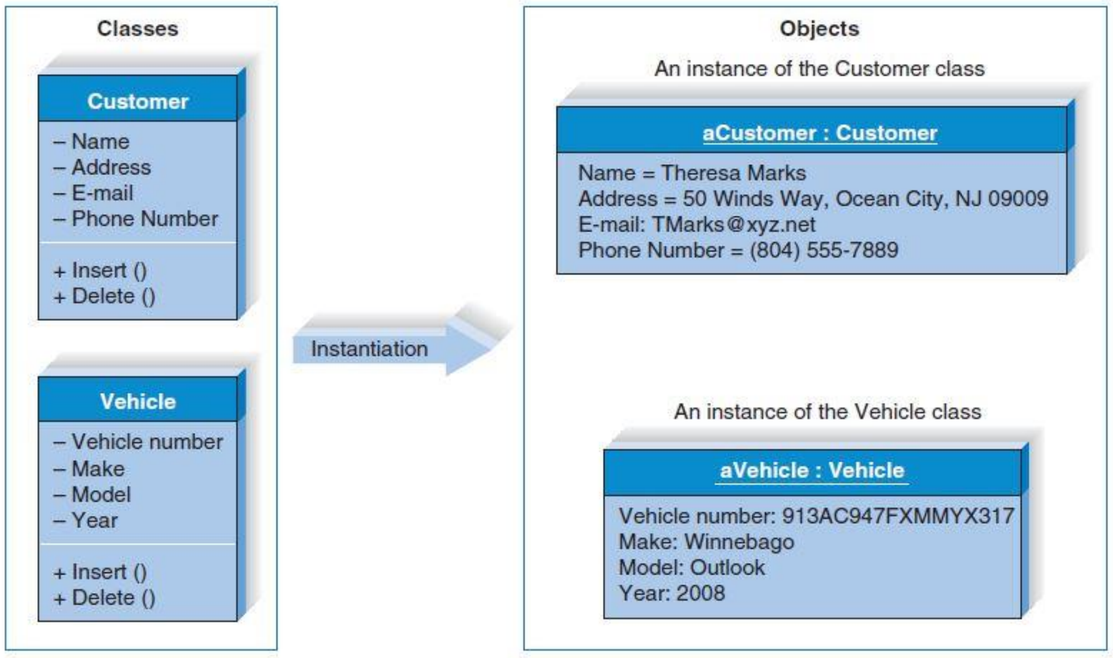


Figure 9: Classes and Objects

One of the more confusing aspects of object-oriented systems development is the fact that in most object-oriented programming languages, both classes and instances of classes can have attributes and methods. Class attributes and methods tend to be used to model attributes (or methods) that deal with issues related to all instances of the class. For example, to create a new customer object, a message is sent to the customer class to create a new instance of itself. However, from a systems analysis and design point of view, we will focus primarily on attributes and methods of objects, and not of classes.

Methods and Messages

Methods implement an object's *behavior.* A method is nothing more than an action that an object can perform. Methods are very much like a function or procedure in a traditional programming language such as C, COBOL, or Pascal. *Messages* are information sent to objects to trigger methods. A message is essentially a function or procedure call from one object to another object. For example, if a customer is new to the organization, the system will send an insert message to the customer object. The customer object will receive a message (instruction) and do what it needs to do to insert the new customer into the system (execute its method). (Figure 10.)

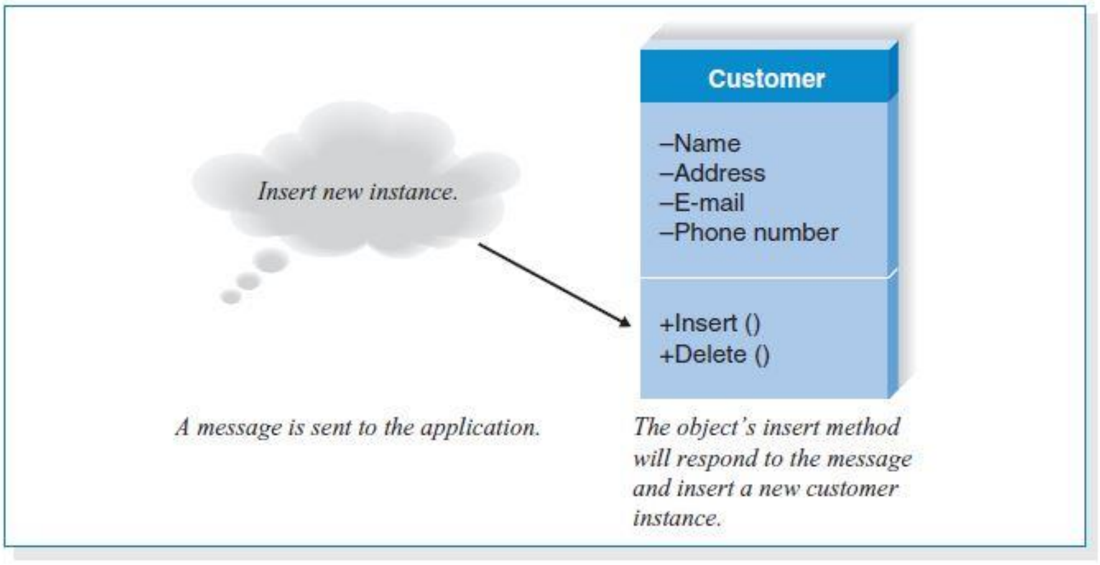

Customer
–Name
–Address
–E-mail
–Phone number
+Insert ()
+Delete ()
Insert new instance.
A message is sent to the application.
The object's insert method will respond to the message and insert a new customer instance.

Figure 10: Messages and methods.

Encapsulation and Information Hiding

The ideas of *encapsulation* and *information hiding* are interrelated in object-oriented systems. Neither of the concepts is new, however. Encapsulation is simply the combining of process and data into a single entity. Object-oriented approaches combine process and data into holistic entities (objects). Information hiding was first promoted in structured systems development. The principle of information hiding suggests that only the information required to use a software module be published to the user of the module. Typically, this implies that the information required to be passed to the module and the information returned from the module are published. Exactly how the module implements the required functionality is not relevant. It doesn't matter how the object performs its functions, so long as the functions occur. In object-oriented systems, combining encapsulation with the information hiding principle suggests that the information hiding principle be applied to objects instead of merely applying it to functions or processes. As such, objects are treated like black boxes.

The fact that an object can be used by calling methods is the key to reusability, because it shields the internal workings of the object from changes in the outside system and it keeps the system from being affected when changes are made to an object. In Figure 10, it can be noticed how a message (insert new customer) is sent to an object, yet the internal algorithms needed to respond to the message are hidden from other parts of the system. The only information that an object needs to know is the set of operations, or methods, that other objects can perform and what messages need to be sent to trigger them.

Inheritance

Inheritance, as an information systems development characteristic, was proposed in data modeling in the late 1970s and early 1980s. The data modeling literature suggests using inheritance to identify higher level, or more general, classes of objects.

Common sets of attributes and methods can be organized into *superclasses.* Typically, classes are arranged in a hierarchy whereby the superclasses, or general classes, are at the top, and the *subclasses*, or specific classes, are at the bottom.

In Figure 11, person is a superclass to the classes employee and customer. Employee, in turn, is a superclass to salesperson and shop mechanic. Notice how a class (e.g., employee) can serve as a superclass and a subclass concurrently. The relationship between the class and its superclass is known as the *A-Kind-Of* (*AKO*) relationship. For example, in Figure 12, a salesperson is A-Kind-Of employee, which is A-Kind-Of person.

Subclasses *inherit* the attributes and methods from the superclass above them. That is, each subclass contains attributes and methods from its parent superclass. For example, Figure 11 shows that both employee and customer are subclasses of person and therefore will inherit the attributes and methods of the person class. Inheritance makes it simpler to define classes. Instead of repeating the attributes and methods in the employee and customer classes separately, the attributes and methods that are common to both are placed in the person class and inherited by those classes below it.

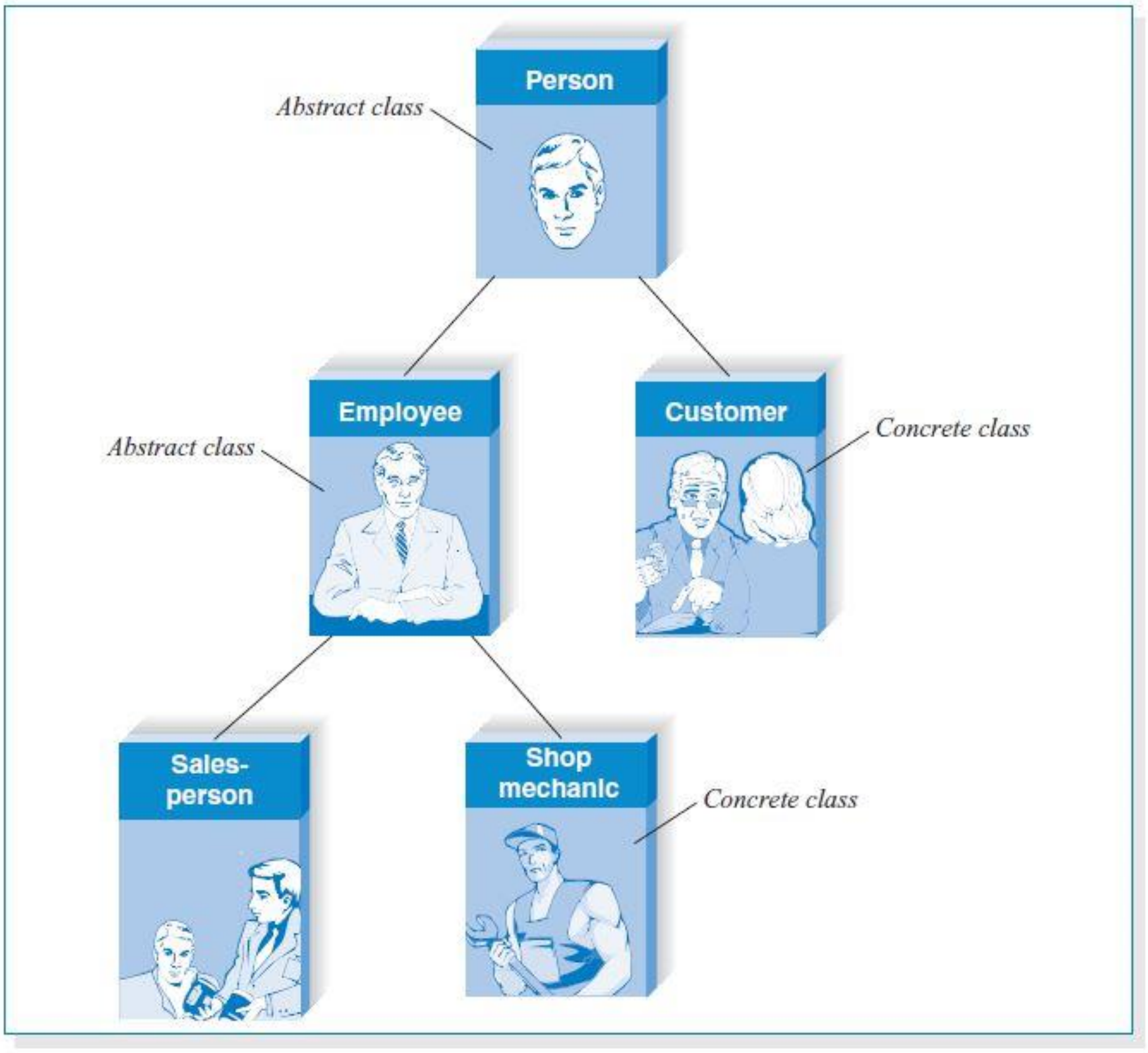


Figure 11: Class hierarchy.

Most classes throughout a hierarchy will lead to instances; any class that has instances is called a *concrete class*. For example, if Mary Wilson and Jim Maloney were instances of the customer class, customer would be considered a concrete class. Some classes do not produce instances, because they are used merely as templates for other, more specific classes (especially those classes located high up in a hierarchy). They are *abstract classes*. Person would be an example of an abstract class. Instead of creating objects from person, we would create instances representing the more specific classes of employee and customer, both types of person. (Figure 12)

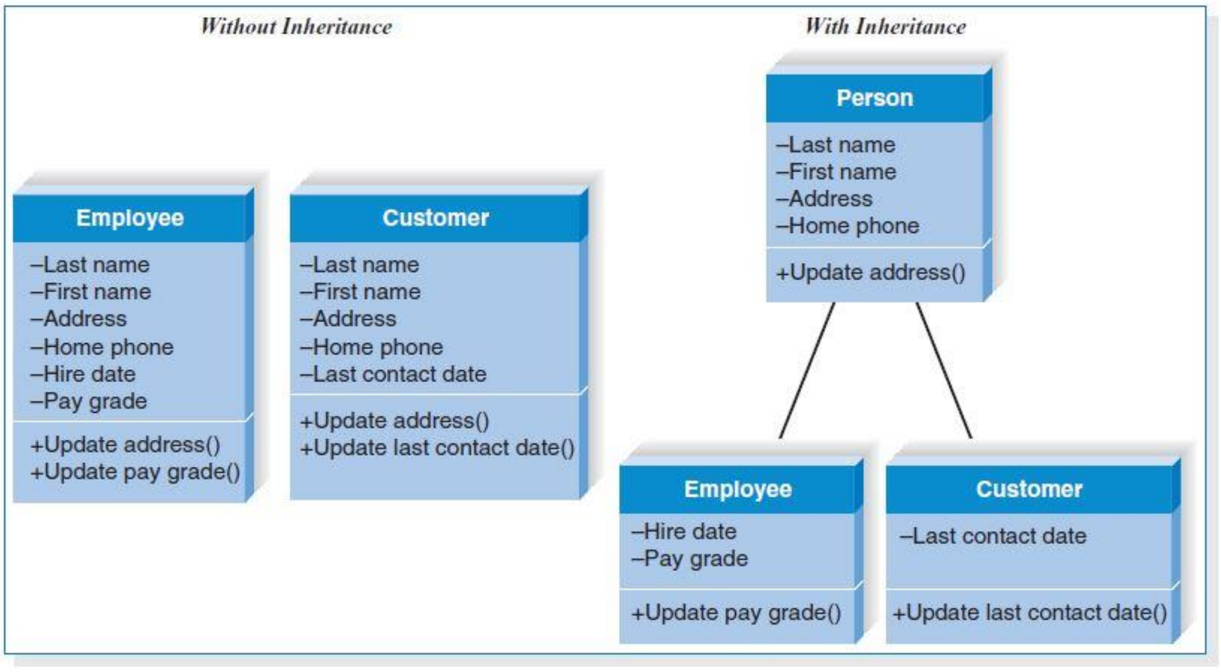


Figure 12: Inheritance.

Polymorphism and Dynamic Binding

Polymorphism means that the same message can be interpreted differently by different classes of objects. For example, inserting a patient means something different than inserting an appointment. As such, different pieces of information need to be collected and stored. Luckily, we do not have to be concerned with *how* something is done when using objects. Message can simply be sent to an object, and that object will be responsible for interpreting the message appropriately. For example, if the message “Draw yourself ” is sent to a square object, a circle object, and a triangle object, the results would be very different, even though the message is the same. Figure 13 shows how each object responds appropriately (and differently), even message is identical.

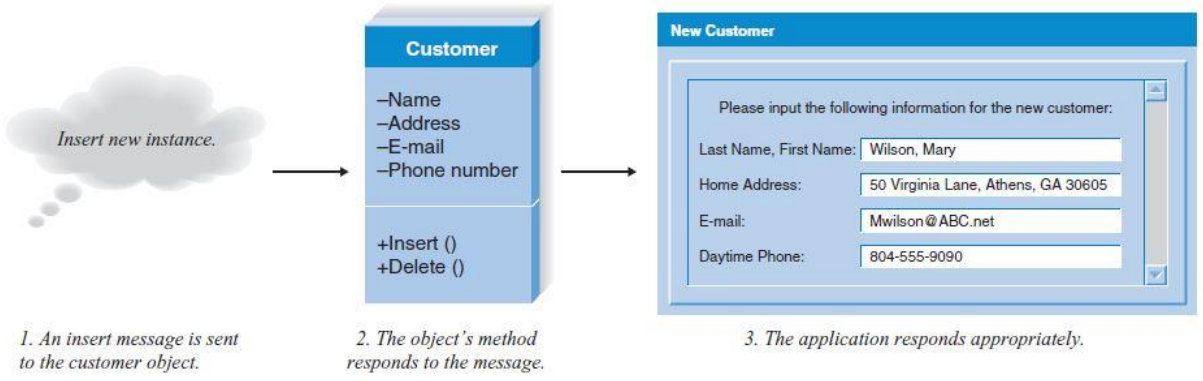

Insert new instance.
Customer
–Name
–Address
–E-mail
–Phone number
+Insert ()
+Delete ()
New Customer
Please input the following information for the new customer:
Last Name, First Name: Wilson, Mary
Home Address: 50 Virginia Lane, Athens, GA 30605
E-mail: Mwilson@ABC.net
Daytime Phone: 804-555-9090
1. An insert message is sent to the customer object.
2. The object's method responds to the message.
3. The application responds appropriately.

Figure 13: Polymorphism and Encapsulation.

Polymorphism is made possible through *dynamic binding.* Dynamic, or late, binding is a technique that delays identifying the type of object until run-time. As such, the specific method that is actually called is not chosen by the object-oriented system until the system is running. This is in contrast to *static binding.* In a statically bound system, the type of object would be determined at compile time. Therefore, the developer would have to choose which method should be called, instead of allowing the system to do it. This is why in most traditional programming languages you find complicated decision logic based on the different types of objects in a system.

For example, in a traditional programming language, instead of sending the message "Draw yourself " to the different types of graphical objects mentioned earlier, we would have to write decision logic by using a case statement or a set of "if " statements to determine what kind of graphical object we wanted to draw, and you would have to name each draw function differently (e.g., draw-square, draw-circle, or draw-triangle). This obviously makes the system much more complicated and more difficult to understand.

Advantages of the object-oriented approach. The OO approach to software development offers many advantages:

- It reduces the complexity of systems development and leads to systems that are easier and quicker to build and maintain, because each object is relatively small and self-contained.
- It improves programmers' productivity and quality. Once an object has been defined, implemented, and tested, it can be reused in other systems.
- Systems developed with the OO approach are more flexible. These systems can be modified and enhanced easily, by changing some types of objects or by adding new types.
- The OO approach allows the systems analyst to think at the level of the real-world systems (as users do) and not at the level of the programming language. The basic operations of an enterprise change much more slowly than the information needs of specific groups or individuals. Therefore, software based on generic models (which the OO approach is) will have a longer life span than programs written to solve specific, immediate problems.
- The OO approach is also ideal for developing Web applications.
- The OO approach depicts the various elements of an information system in user terms (i.e., business or real-world terms), and therefore, the users have a better understanding of what the new system does and how it meets its objectives.

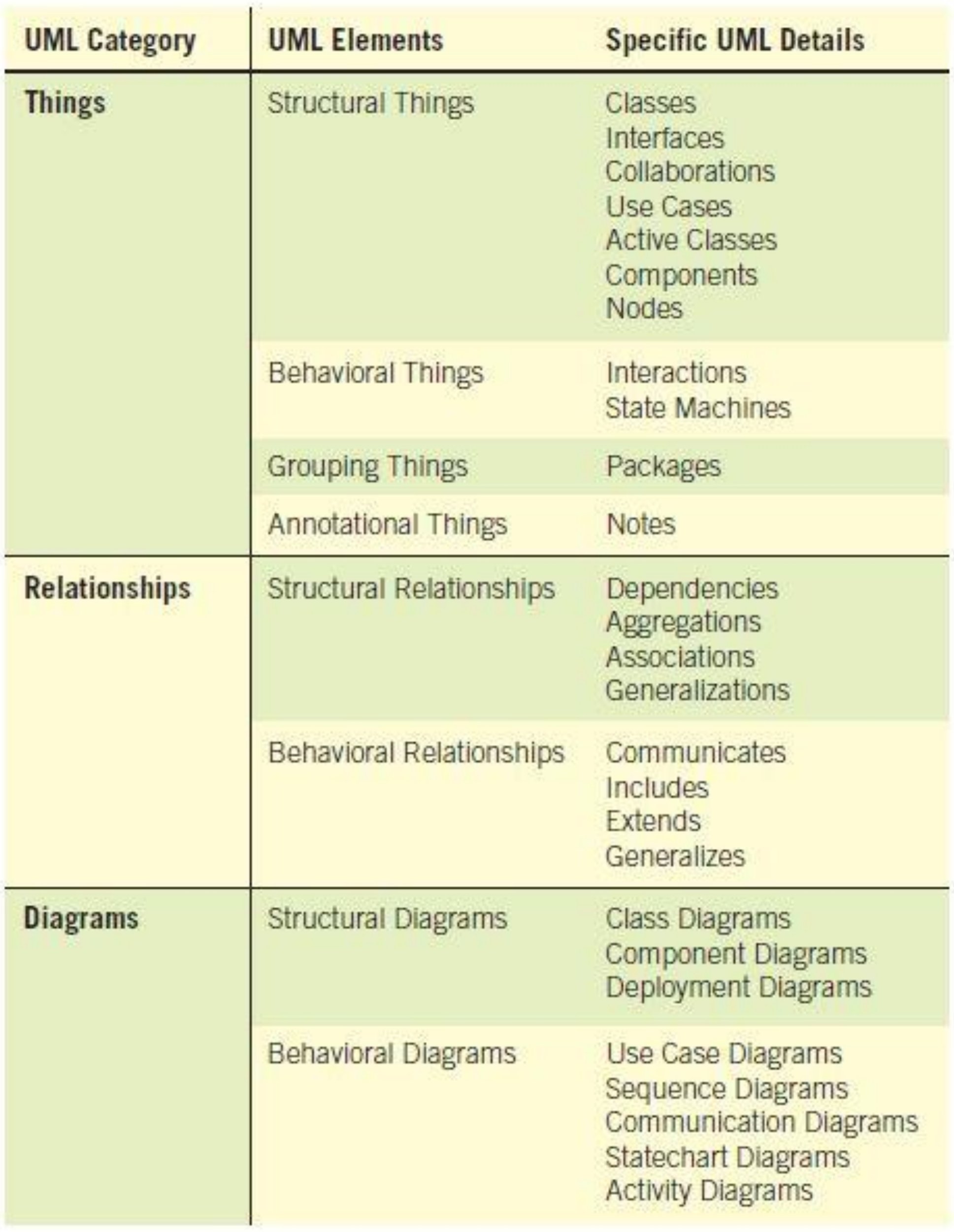
The OO approach does have disadvantages. OO systems, especially those written in Java, generally run more slowly than those developed in other programming languages. Also, many programmers have little skill and experience with OO languages, necessitating retraining.

4.2. Unified Modelling Language (UML)

Professors of Rutgers University – Kenneth E. Kendall and Julie E. Kendall in their book (“System Analysis and Design”, eight edition, 2006) give definition to UML as follows – “*UML provides a standardized set of tools to document the analysis and design of a software system. The UML toolset includes diagrams that allow people to visualize the construction of an object-oriented system, similar to the way a set of blueprints allows people to visualize the construction of a building*”. Whether work is done independently or with a large systems development team, the documentation that is created with UML provides an effective means of communication between the development team and the business team on a project.

UML consists of things, relationships, and diagrams, as illustrated in Table 1. The first components, or primary elements, of UML are called things. It may be prefered another word, such as object, but in UML they are called things. Structural things are most common. Structural things are classes, interfaces, use cases, and many other elements that provide a way to create models. Structural things allow the user to describe relationships. Behavioral things describe how things work. Examples of behavioral things are interactions and state machines. Group things are used to define boundaries. An example of a group thing is a package. Finally, there are annotational things, so that notes can be added to the diagrams.

Relationships are the glue that holds the things together. It is useful to think of relationships in two ways. *Structural relationships* are used to tie the things together in the structural diagrams. Structural relationships include dependencies, aggregations, associations, and generalizations. Structural relationships show inheritance, for example. *Behavioral relationships* are used in the behavioral diagrams. The four basic types of behavioral relationships are communicates, includes, extends, and generalizes. There are two main types of diagrams in UML: structural diagrams and behavioral diagrams. Structural diagrams are used, for example, to describe the relationships between classes. They include class diagrams, object diagrams, component diagrams, and deployment diagrams. Behavioral diagrams, on the other hand, can be used to describe the interaction between people (called actors in UML) and the thing that is referred to as a use case, or how the actors use the system. Behavioral diagrams include use case diagrams, sequence diagrams, communication diagrams, statechart diagrams, and activity diagrams.



UML Category	UML Elements	Specific UML Details
Things	Structural Things	Classes Interfaces Collaborations Use Cases Active Classes Components Nodes
	Behavioral Things	Interactions State Machines
	Grouping Things	Packages
	Annotational Things	Notes
Relationships	Structural Relationships	Dependencies Aggregations Associations Generalizations
	Behavioral Relationships	Communicates Includes Extends Generalizes
Diagrams	Structural Diagrams	Class Diagrams Component Diagrams Deployment Diagrams
	Behavioral Diagrams	Use Case Diagrams Sequence Diagrams Communication Diagrams Statechart Diagrams Activity Diagrams

Table 1: UML and its components: Things, Relationships, and Diagrams.

The five most commonly used UML diagrams are:

1. A *use case diagram*, describing how the system is used.

2. An *activity diagram*, illustrating the overall flow of activities. Each use case may create one activity diagram.

3. *Sequence diagrams*, showing the sequence of activities and class relationships. Each use case may create one or more sequence diagrams. An alternative to a sequence diagram is a communication diagram, which contains the same information but emphasizes communication instead of timing.

4. *Class diagrams,* showing the classes and relationships. Sequence diagrams are used (along with CRC cards) to determine classes. An offshoot of a class diagram is a gen/spec diagram (which stands for generalization/specialization).

5. *Statechart (transition) diagrams*, showing the state transitions. Each class may create a statechart diagram, which is useful for determining class methods.

Use Case Diagram

Use-case modeling is applied to analyze the functional requirements of a system. Valacich, Joseph S., Joey F. George, Jeffrey A. Hoffer in their book "Essentials of systems analysis and design, 5-th edition" describe Use Case modelling as follows – "*Use-case modeling is done in the early stages of system development to help developers understand the functional requirements of the system without worrying about how those requirements will be implemented*." The process is inherently iterative; developers need to involve the users in discussions throughout the model development process and finally come to an agreement on the requirements specification.

A use-case model consists of actors and use cases. An *actor* is an external entity that interacts with the system. It is someone or something that exchanges information with the system. A *use case* represents a sequence of related actions initiated by an actor; it is a specific way of using the system. An actor represents a role that a user can play. The actor's name should indicate that role. Actors helps to identify the use cases they carry out.

The developer sits down with the intended users of the system and makes a thorough analysis of what functions they desire from the system. These functions are represented as use cases. For example, a university registration system has a use case for class registration and another for student billing. These use cases, then, represent the typical interactions the system has with its users.

In UML, a use-case model is depicted diagrammatically, as in Figure 14. This use-case diagram is for a university registration system, which is shown as a box. Outside the box are four actors—Student, Registration clerk, Instructor, and Bursar's office—that interact with the system (shown by the lines touching the actors). An actor is shown using a stick figure with its name below. Inside the box are four use cases—*Class registration, Registration for special class, Prereq courses not completed,* and *Student billing*—which are shown as ellipses with their names inside. These use cases are performed by the actors outside the system. A use case is always initiated by an actor. For example, *Student billing* is initiated by the Bursar's office. A use case can interact with actors other than the one that initiated it. The *Student billing* use case, although initiated by the Bursar's office, interacts with the Students by mailing them tuition invoices. Another use case, *Class registration,* is carried out by two actors, Student and Registration clerk. This use case performs a series of related actions aimed at registering a student for a class. The numbers on each end of the interaction lines indicate the number of instances of the use case with which the actor is associated. For example, the Bursar's office causes many (*) *Student billing* use-case instances to occur, each one for exactly one student.

A use case represents a complete functionality. It should not be represented an individual action that is part of an overall function as a use case. For example, although submitting a registration form and paying tuition are two actions performed by users (students) in the university registration system, we do not show them as use cases, because they do not specify a complete course of events; each of these actions is executed only as part of an overall function or use case. We can think of "Submit registration form" as one of the actions of the *Class registration* use case, and "Pay tuition" as one of the actions of the *Student billing* use case.

A use case may participate in relationships with other use cases. An *extends relationship,* shown as a line with a hollow triangle pointing toward the extended use case and labeled with the "<<extends>>" symbol, extends a use case by adding new behaviors or actions. In Figure 14, for example, the *Registration for special class* use case extends the *Class registration* use case by capturing the additional actions that need to be performed in registering a student for a special class. Registering for a special class requires prior permission of the instructor, in addition to the other steps carried out for a regular registration. We may think of *Class registration* as the basic course, which is always performed—independent of whether the extension is performed or not—and *Registration for special class* as an alternative course, which is performed only under special circumstances. Another example of an extends relationship is that between the *Prereq courses not completed* and *Class registration* use cases. The former extends the latter in situations where a student registering for a class has not taken the prerequisite courses.

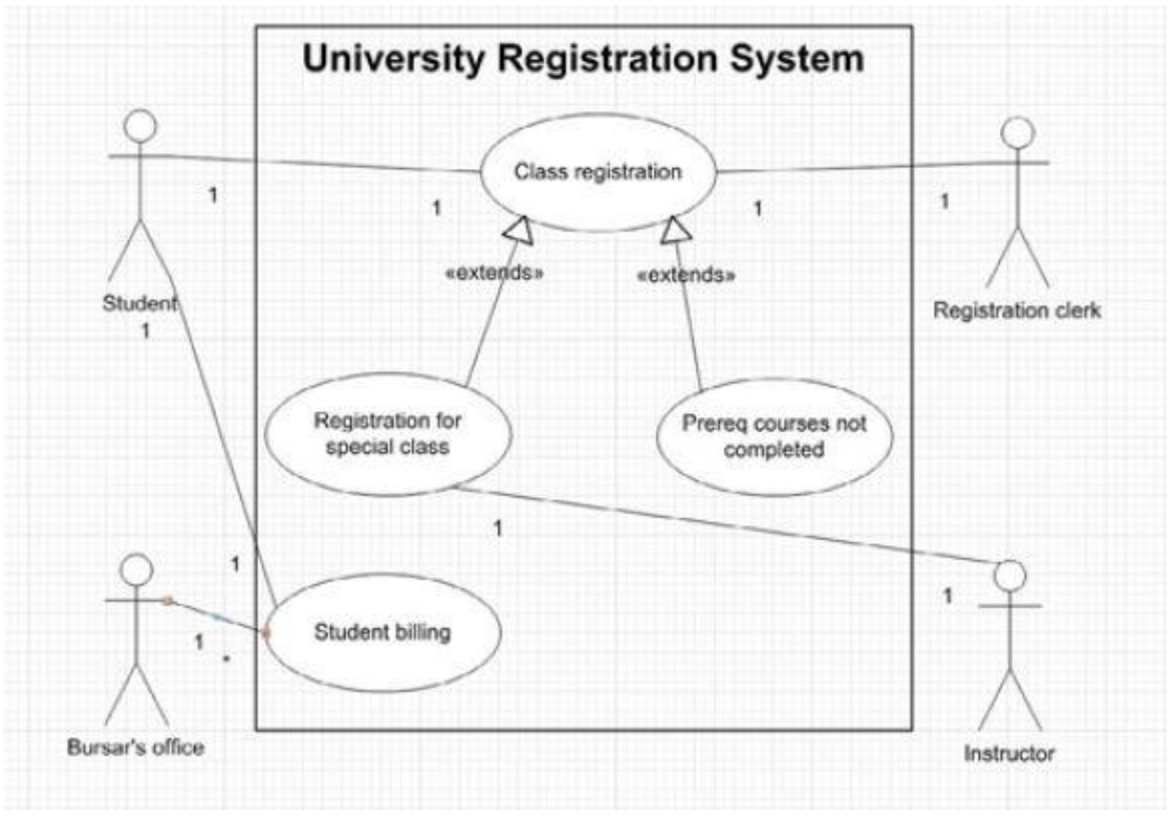


Figure 14: Use Case Use-case diagram for a university registration system.

Figure 15 shows a use-case diagram for Hamburger. The Customer actor initiates the *Order food* use case; the other actor involved is the Service Person. A specific scenario would represent a customer placing an order with a service person. Another kind of relationship is *included,* which arises when one use case references another use case. An include relationship is also shown diagrammatically as a dashed line with a hollow arrowhead pointing toward the use case that is being used; the line is labeled with the "<<include>>" symbol. In Figure 15, for example, the include relationship between the *Reorder supplies* and *Track sales and inventory data* use cases implies that the former uses the latter while executing. Simply put, when a manager reorders supplies, the sales and inventory data are tracked. The same data are also tracked when management reports are produced, so there is another include relationship between the *Produce management reports* and *Track sales and inventory data* use cases.

The *Track sales and inventory data* is a generalized use case, representing the common behavior among the specialized use cases, *Reorder supplies* and *Produce management reports.* When *Reorder supplies* or *Produce management reports* is performed, the entire *Track sales and inventory data* is used.

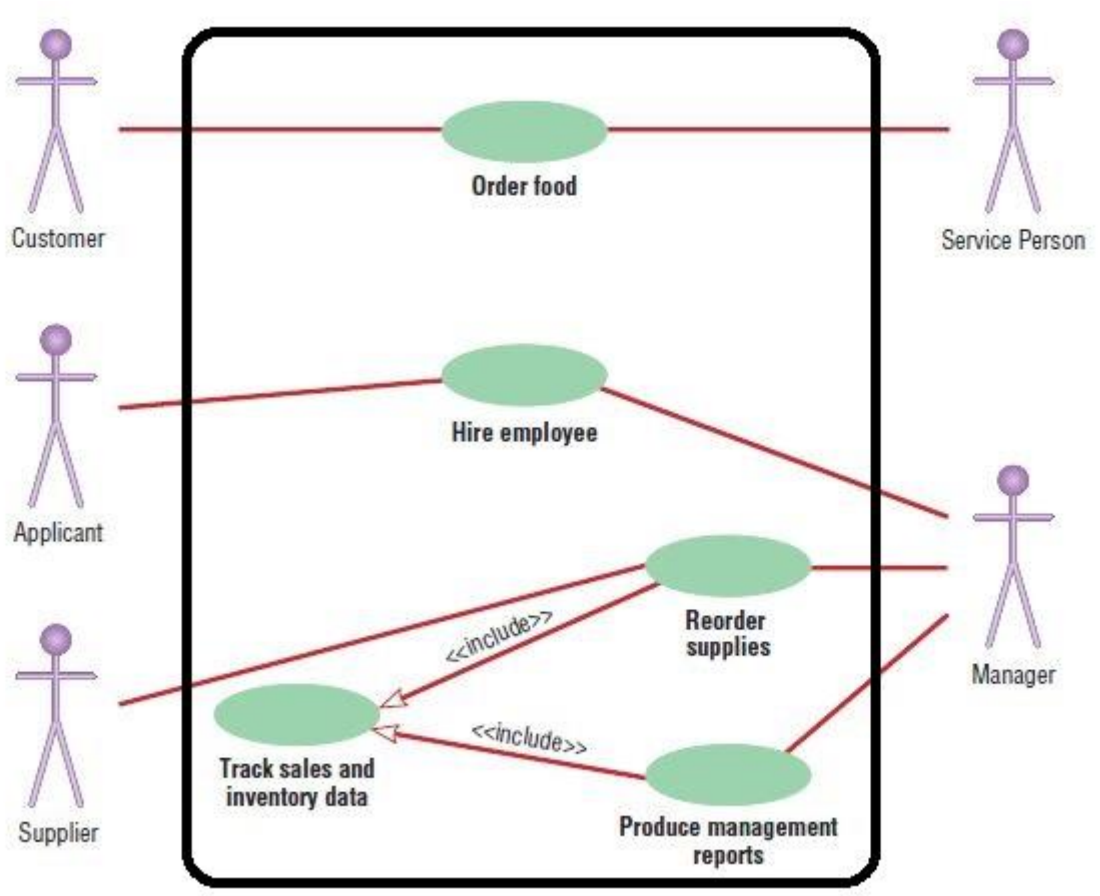
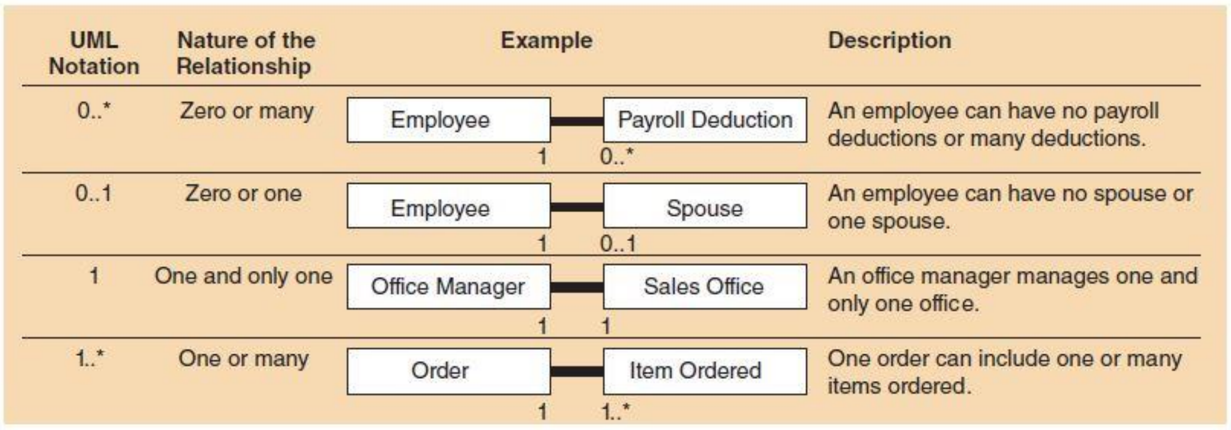


Figure 15: Use Case Use-case diagram for Hamburger system.

Class Diagram

A *class diagram* shows the object classes and relationships involved in a use case. A class diagram is a logical model, which evolves into a physical model and finally becomes a functioning information system. In structured analysis, entities, data stores, and processes are transformed into data structures and program code. Similarly, class diagrams evolve into code modules, data objects, and other system components. In a class diagram, each class appears as a rectangle, with the class name at the top, followed by the class's attributes and methods. Lines show relationships between classes and have labels identifying the action that relates the two classes. To create a class diagram, we review the use case and identify the classes that participate in the underlying business process.

The class diagram also includes a concept called *cardinality*, which describes how instances of one class relate to instances of another class. For example, an employee might have earned no vacation days or one vacation day or many vacation days. Similarly, an employee might have no spouse or one spouse. Figure 16 shows various UML notations and cardinality examples. In Figure 16, the first column shows a UML notation symbol that identifies the relationship shown in the second column. The third column provides a typical example of the relationship, which is described in the last column. In the first row of the figure, the UML notation *0..** identifies a *zero or many* relation. The example is that an employee can have no payroll deductions or many deductions.



UML Notation	Nature of the Relationship	Example	Description
0..*	Zero or many	Employee 1 — 0..* Payroll Deduction	An employee can have no payroll deductions or many deductions.
0..1	Zero or one	Employee 1 — 0..1 Spouse	An employee can have no spouse or one spouse.
1	One and only one	Office Manager 1 — 1 Sales Office	An office manager manages one and only one office.
1..*	One or many	Order 1 — 1..* Item Ordered	One order can include one or many items ordered.

Figure 16: Examples of UML notations that indicate the nature of the relationship between instances of one class and instances of another class.

Figure 17 shows a class diagram for a sales order use case. It can be noticed that the sales office has one sales manager who can have anywhere from zero to many sales reps. Each sales rep can have anywhere from zero to many customers, but each customer has only one sales rep.

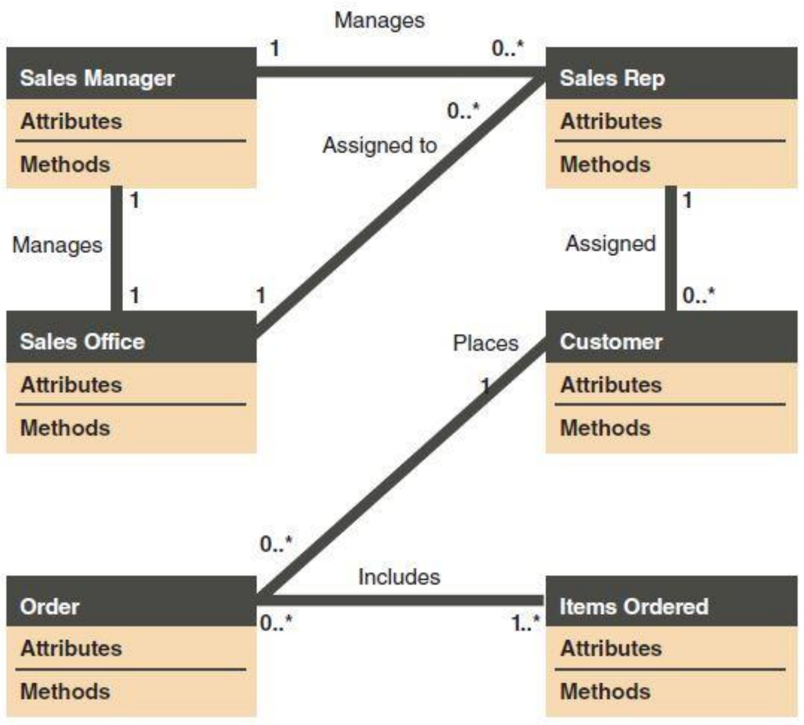


Figure 17: Class diagram for a sales order use case (attributes, methods omitted for clarity)

Sequence diagram

A *sequence diagram* is a dynamic model of a use case, showing the interaction among classes during a specified time period. A sequence diagram graphically documents the use case by showing the classes, the messages, and the timing of the messages. Sequence diagrams include symbols that represent classes, lifelines, messages, and focuses. These symbols are shown in Figure 18.

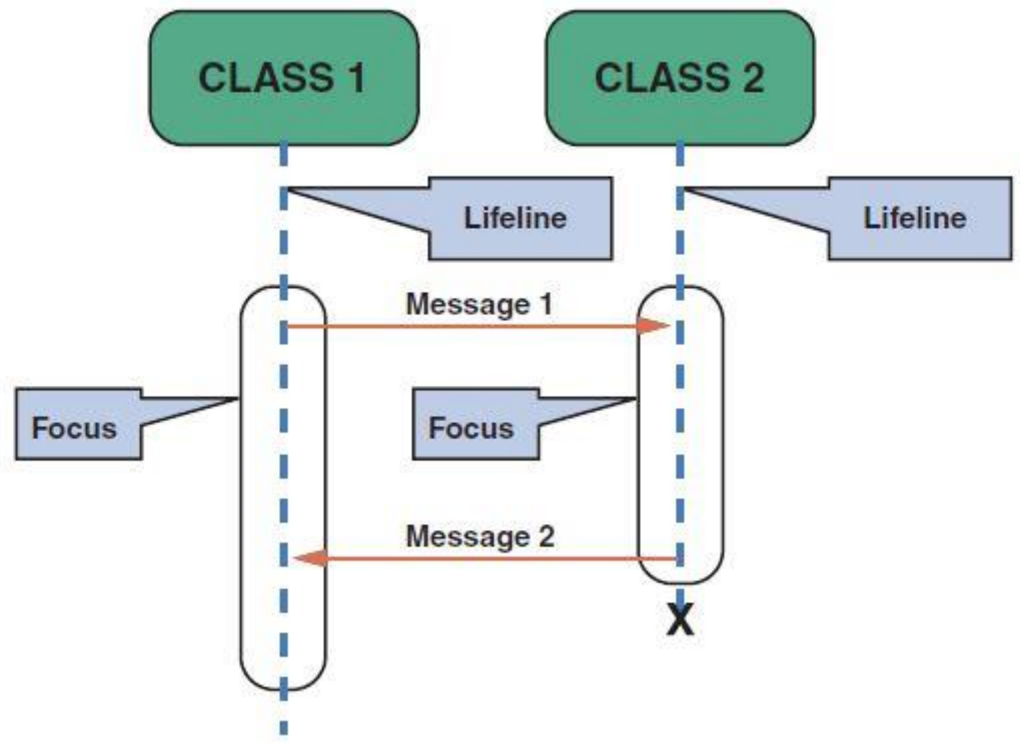


Figure 18: A sequence diagram with two classes. *X* indicates the end of the CLASS 2 lifeline. Each message is represented by a line with a label that describes the message, and that each class has a focus that shows the period when messages are sent or received.

*Classes***.** A class is identified by a rectangle with the name inside. Classes that send or receive messages are shown at the top of the sequence diagram.

Lifelines. A lifeline is identified by a dashed line. The lifeline represents the time during which the object above it is able to interact with the other objects in the use case. An *X* marks the end of the lifeline.

*Messages***.** A message is identified by a line showing direction that runs between two objects. The label shows the name of the message and can include additional information about the contents.

*Focuses***.** A focus is identified by a narrow vertical shape that covers the lifeline. The focus indicates when an object sends or receives a message.

Figure 19 is a simplified example of a sequence diagram for a use case that admits a student to a university. On the left is the *newStudentUserInterface* class that is used to obtain student information. The initialize() message is sent to the Student class, which creates a new student record and returns the student number. To simplify the diagram, the parameters that are sent to the

Student class have been omitted, but would include the student name, address, and so on. The next activity is to send a selectDorm message to the Dorm class. This message would include dorm selection information, such as a health dorm or other student requirements. The Dorm class returns the dorm name and room number. The third activity is to send a selectProgram message to the Program class, including the program name and other course of study information. The program advisor name is returned to the newStudentUserInterface class. A studentComplete message is sent to the Student class with the dorm, advisor name, and other information.

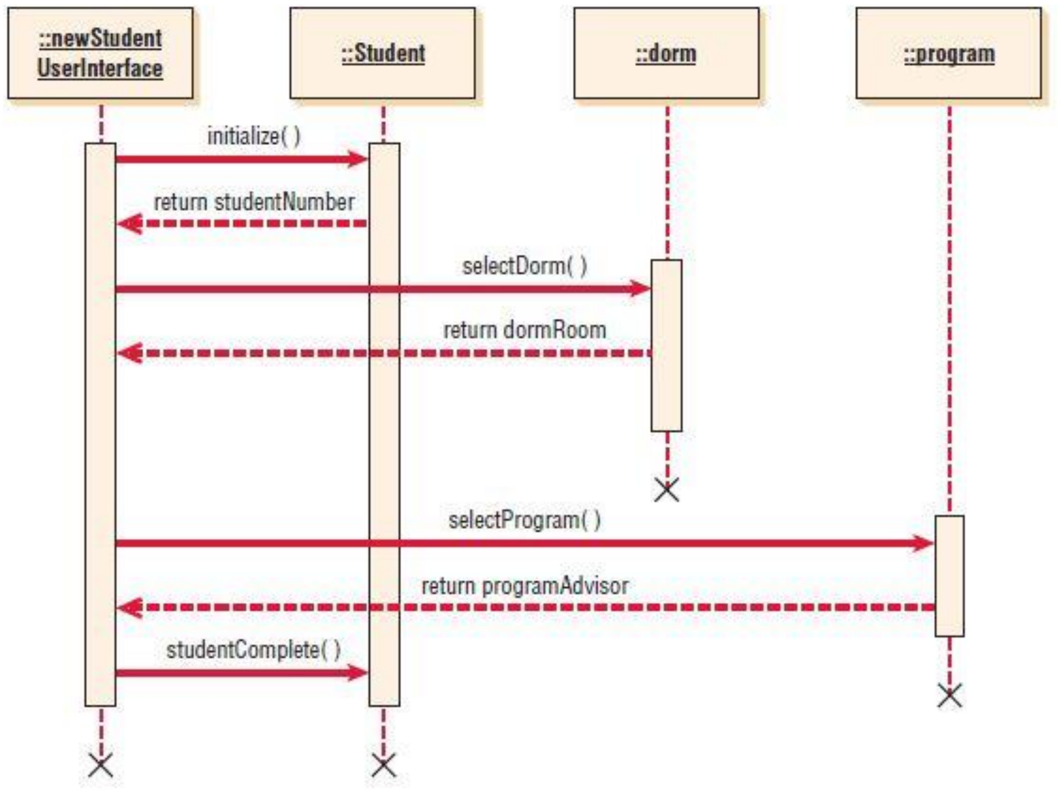


Figure 19: A sequence diagram for student admission. Sequence diagrams emphasize the time ordering of messages.

Sequence diagrams can be used to translate the use case scenario into a visual tool for systems analysis. The initial sequence diagram used in systems analysis shows the actors and classes in the system and the interactions between them for a specific process. A sequence diagram emphasizes the time ordering (sequence) of messages. During the systems design phase, the sequence diagrams are refined to derive the methods and interactions between classes. Messages from one class are used to identify class relationships. The actors in the earlier sequence diagrams are translated to interfaces, and class interactions are translated to class methods. Class methods used to create instances of other classes and to perform other internal system functions become apparent in the system design using sequence diagrams.

State Transition Diagrams

A state transition diagram shows how an object changes from one state to another, depending on events that affect the object. All possible states must be documented in the state transition diagram, as shown in Figure 20. A bank account, for example, could be opened as a NEW account, change to an ACTIVE or EXISTING account, and eventually become a CLOSED or FORMER account. Another possible state for a bank account could be FROZEN, if the account's assets are legally attached. In a state transition diagram, the states appear as rounded rectangles with the state names inside. The small circle to the left is the initial state, or the point where the object first interacts with the system. Reading from left to right, the lines show direction and describe the action or event that causes a transition from one state to another. The circle at the right with a hollow border is the final state.

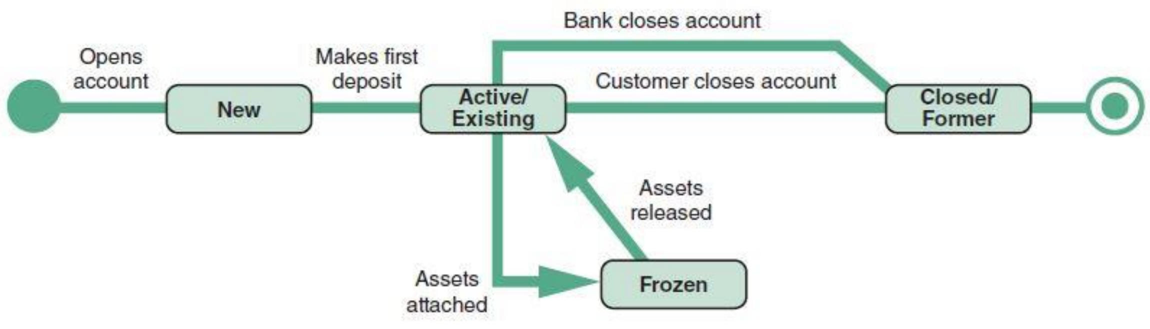


Figure 20: An example of a state transition diagram for a bank account.

Activity Diagrams

An activity diagram resembles a horizontal flowchart that shows the actions and events as they occur. Activity diagrams show the order in which the actions take place and identify the outcomes. Figure 21 shows an activity diagram for a cash withdrawal at an ATM machine. It can be seen that the customer initiates the activity by inserting an ATM card and requesting cash. Activity diagrams also can display multiple use cases in the form of a grid, where classes are shown as vertical bars and actions appear as horizontal arrows.

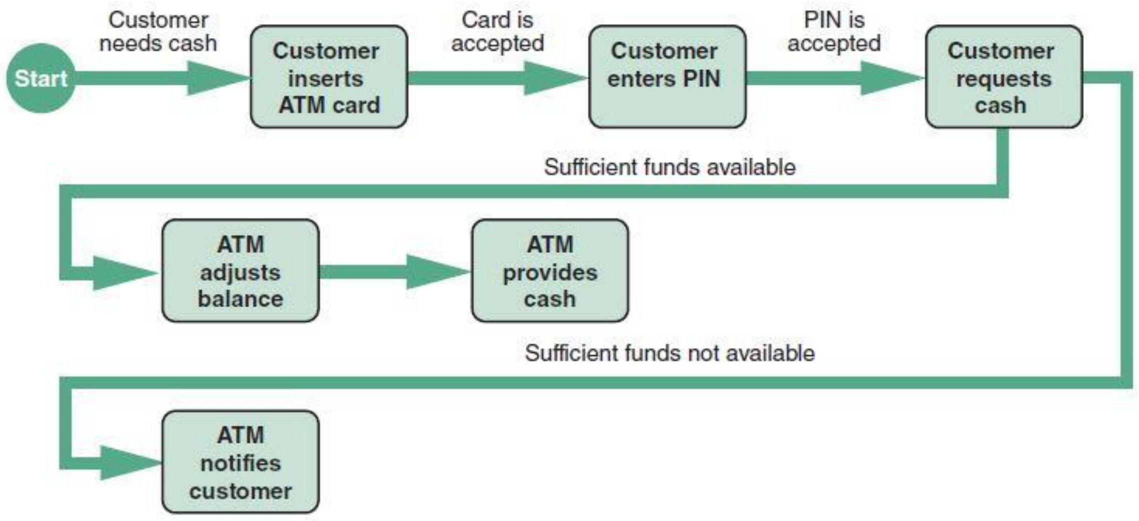
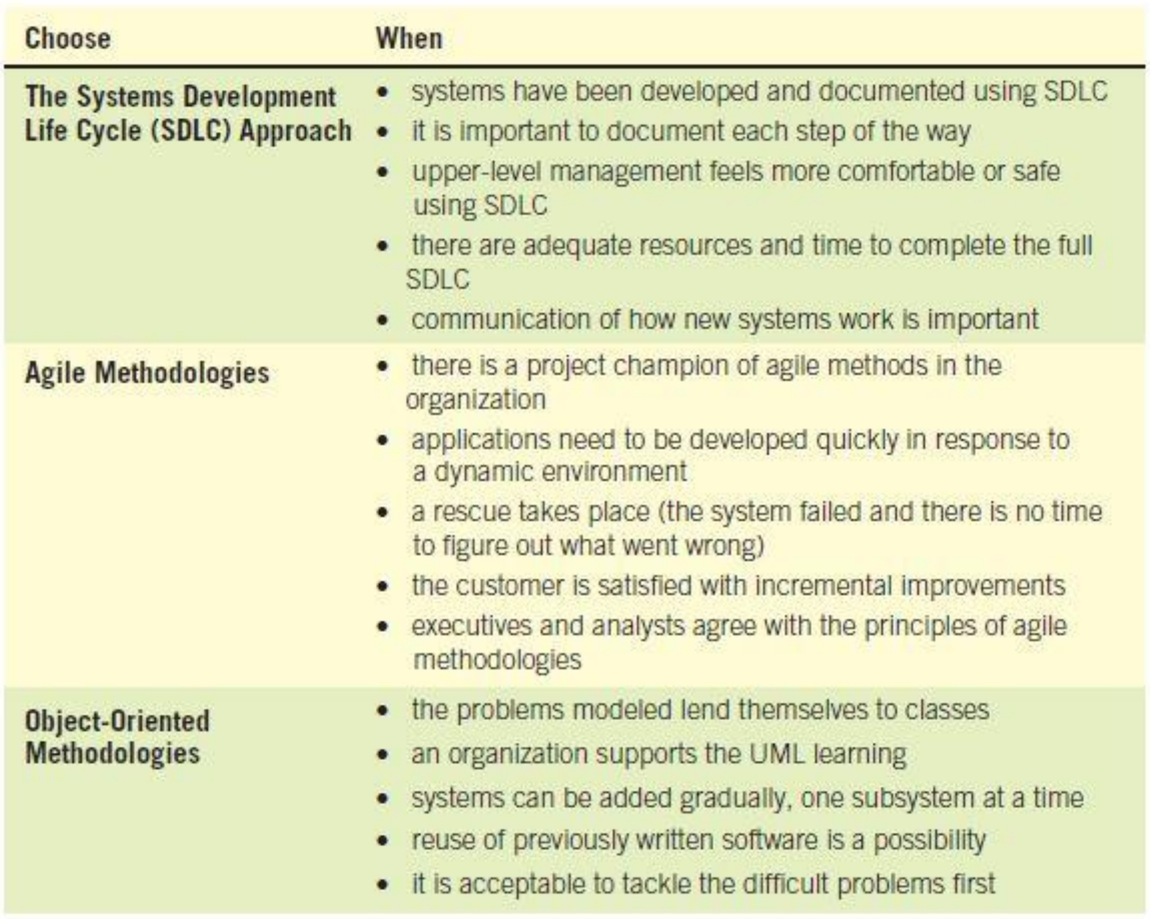


Figure 21: An activity diagram shows the actions and events involved in withdrawing cash from an ATM machine.

Choosing which systems development method to use

The differences among the three system design and development approaches described earlier are not as big as they seem at the outset. The SDLC and object-oriented approaches both require extensive planning and diagramming. The agile approach and the object-oriented approach both allow subsystems to be built one at a time until the entire system is complete. The agile and SDLC approaches are both concerned about the way data logically moves through the system. So given a choice to develop a system using an SDLC approach, an agile approach, or an object-oriented approach, which would we choose? Table 2 provides a set of guidelines to help you choose which method to use when developing your next system (Professor Kenneth E. Kendall, "System Analysis and Design", eighth edition, 2006).



Choose	When
The Systems Development Life Cycle (SDLC) Approach	• systems have been developed and documented using SDLC • it is important to document each step of the way • upper-level management feels more comfortable or safe using SDLC • there are adequate resources and time to complete the full SDLC • communication of how new systems work is important
Agile Methodologies	• there is a project champion of agile methods in the organization • applications need to be developed quickly in response to a dynamic environment • a rescue takes place (the system failed and there is no time to figure out what went wrong) • the customer is satisfied with incremental improvements • executives and analysts agree with the principles of agile methodologies
Object-Oriented Methodologies	• the problems modeled lend themselves to classes • an organization supports the UML learning • systems can be added gradually, one subsystem at a time • reuse of previously written software is a possibility • it is acceptable to tackle the difficult problems first

Table 2: How to decide which development method to use.

5. Travel Expenses Management

The typical enterprise spends thousands if not millions annually on non-compensation related corporate expenses. Whether it is a company with pilots traveling around the world or a Sales/Marketing Department attending events, the cost of travel is almost unavoidable. Therefore, for most companies, travel related expenses remain a significant area of spend –second only to payroll. This is expected to dramatically increase in the coming years due to globalization and increased cost of air travel as well as accommodations. As economic conditions tighten, companies worldwide are increasingly paying greater attention to their expense management systems. To keep up and maintain tight control with their day-to-day expenses, companies must think creatively while also improving financial reporting to meet regulatory and internal audit requirements. The key to succeeding in this game is by utilizing new technologies and fostering best practices that help companies take their expense management strategies to a new level. The availability of real-time data in reference to expense management improves overall performance for companies. According to Aberdeen Group's released report, "Travel Expense Management: Leveraging Data to Drive Performance," Best-In-Class companies are focusing on managing their expense related processes by creatively blending technology and strategy. Expense management is a vital component of a finance department and comprises a large portion of a companies annual budget, with the average organization spending between 10-12% of their annual budget on expenses related to travel and entertainment. With the large corporate expense, the speed and accuracy of expense reporting and oversight can have a direct impact on the efficiency of a company and affect its bottom line. In an annual study conducted by the Aberdeen Group, top companies in a variety of industries were interviewed and executives were asked to list their top financial pressures. Nearly 50% cited the need to improve compliance to company policies 41% said they wanted to improve business reporting and analytics 38% stated they needed to reduce the cost of processing expense reports. Within the same report, the perceived importance of expense management increased by 39% between 2014 and 2015. The reality is that expense management is a significant component of a successful 21st century business and modern technology can now make it a more affordable, to manage component.

5.1. Travel Expenses Structure

Travel expenses are expenditures that an employee makes while traveling on company business. Company business can include conferences, exhibitions, business meetings, client and customer meetings, job fairs, training sessions, and sales calls, for example. Expenses can include lodging, personal car mileage reimbursement, flights, ground transportation, tips to bellhops, meals, tips to waiters, room service, and other incidental expenses an employee might experience while on the road. Expenditures that an organization will reimburse are found in the company's business travel policy. Every worker should become familiar with his/her company's policy because some extra expenses can be covered on extended trips. Client entertainment at conferences, on sales calls, and on site visits is another reimbursable expense, but knowing one's company's policies helps not to exceed the limits that are placed on entertainment costs.

The figure below shows that Travel Expense management involves Policy, Processes and System.

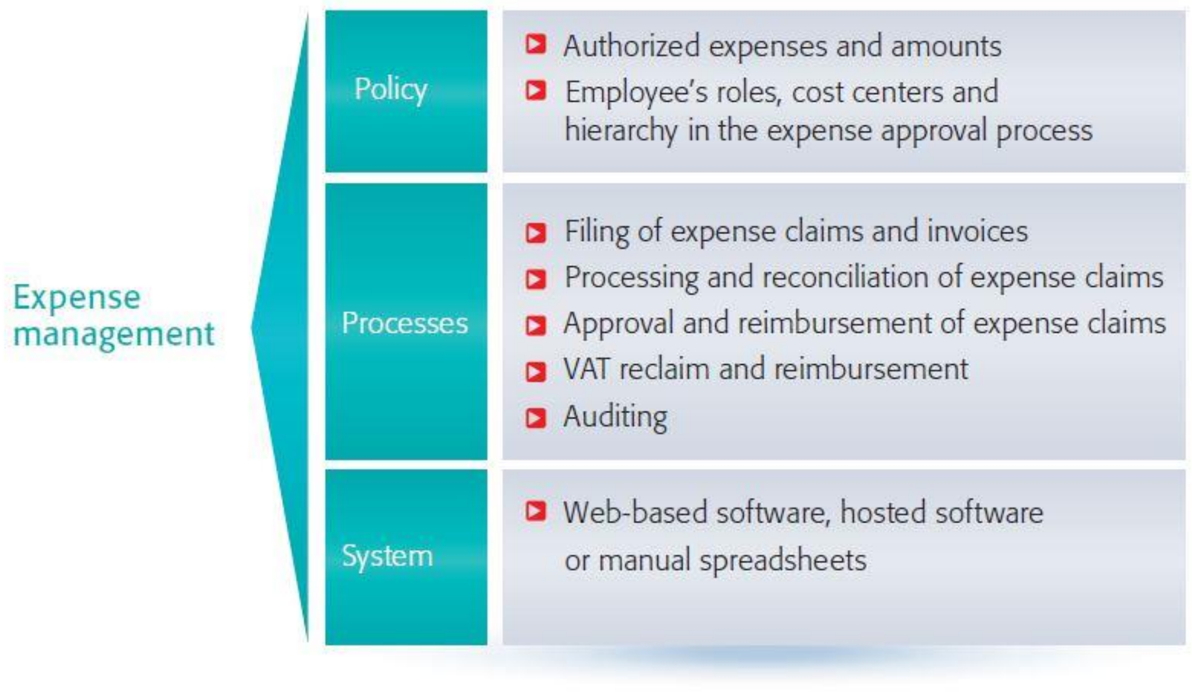

Expense management
Policy
Authorized expenses and amounts
Employee's roles, cost centers and hierarchy in the expense approval process
Processes
Filing of expense claims and invoices
Processing and reconciliation of expense claims
Approval and reimbursement of expense claims
VAT reclaim and reimbursement
Auditing
System
Web-based software, hosted software or manual spreadsheets

Figure 22: Travel Expense Management involves policy, processes and a system

Typically, organizations pay employee travel expenses in these three ways.

Company credit cards are issued to employees who must travel frequently for business. Employees may charge most of the expenses they incur on a business trip to the company credit card. For reimbursement of incidentals such as tips and fast food, employees will need to fill out an expense report. Charge cards are convenient for employees as they do not have to come up cash to pay for business expenses prior to reimbursement. Employee should become knowledgeable about company's policies, though; he/she may still need to turn in receipts and other supporting documentation.

*Cash***:** Organizations without employee company credit cards require employees to fill out an expense reimbursement report for each expenditure while the employee is on the road. They generally require receipts and some level of justification for each expense.

Only rarely would an organization ask employees to pay for the big ticket items such as air fare and seek reimbursement later. A company purchase order or company credit card pay for large expenses up front. But employees are often required to pay cash out-of-pocket for day-to-day travel expenses which are later reimbursed.

*Per diem***:** A per diem is a daily allowance of a certain amount of money that an employee is given to cover all expenses. The employee is responsible for making sound travel expense choices within the parameters of the amount of money that he or she is allotted daily. Some companies pay directly for transportation and housing, but give traveling employees a per diem for all other expenses including meals and ground transportation. Employees have been known to under spend on expenses to keep the extra cash from the per diem. Companies generally allow this.

Employees who travel for business are advised to stay up-to-date on company travel policies and costs covered for reimbursement. Expenses that fall outside of the policies are generally not reimbursed or covered. Receipts are required by most companies except for those that pay a per diem. Company also likely has a form that they expect employees to use for turning in travel expenses. To stay on top of reimbursable expenses, employees are often given a deadline by which they need to file an expense report and turn in applicable receipts. The finance department will have guidelines that help it stay current.

In the following figure, we can see how large eastern European countries' travel expenses are. (Source: "Hermes Management Consulting: Corporate Travel Management in Western Europe, opportunities and challenges")

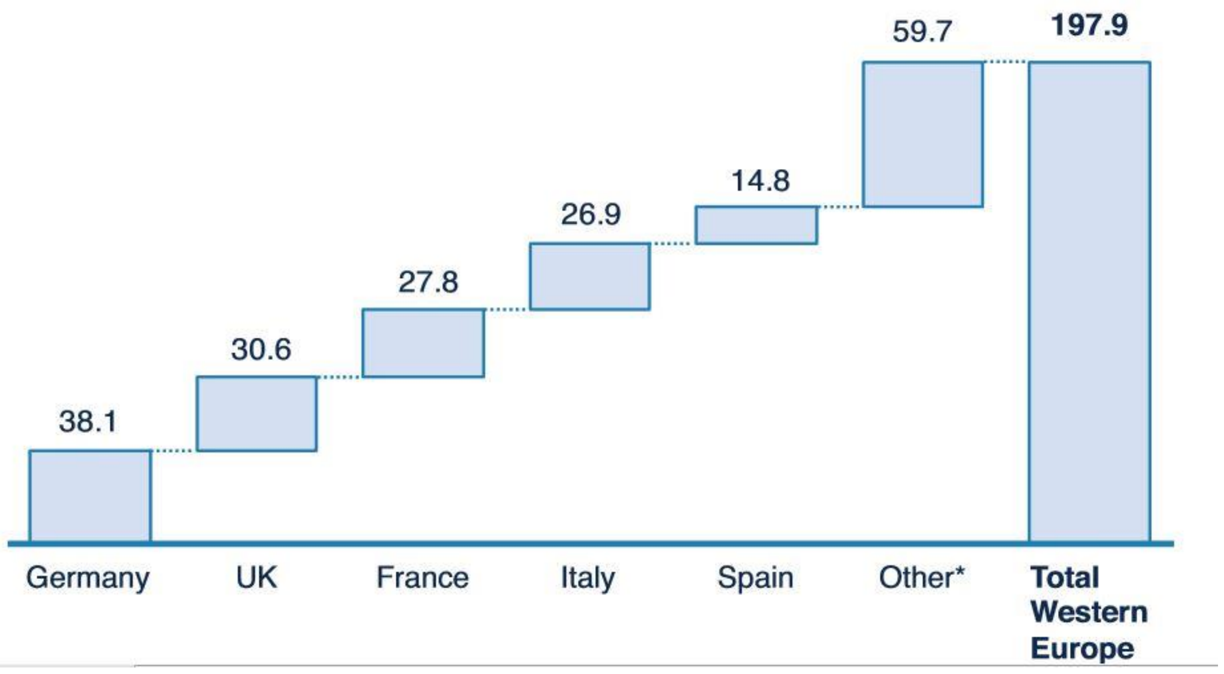


Figure 23: Total Western Europe Business Travel spend by country (2014)

5.2. Life-cycle of Travel Expenses Management

In general, companies view travel expense management as a compilation of disparate and distinct processes that, when brought together, provides a medium for employees to submit expenses while adhering to policies and meeting compliance. While these processes and procedures vary from one company to another, the steps are essentially the same. To better understand how the various elements of this lifecycle interact with each other, see Figure . The travel & expense management lifecycle starts out with Pre-Trip Objectives. This includes defining the scope of the trip and quantifying it against measurable business objectives. Once the stakeholders have cleared this phase, Travel Planning follows. Here, reservations with flights, hotels, rental cars are made and submitted to the system. The middle step is Travel Authorization. This allows the user to proceed with the travel after it has been approved. Here, the budgeted expenses for the trip are registered in the system. This information is used at a later point to compare against actual trip costs –which is a valuable metric for corporations looking to get a better handle on spending. After clearing the initial steps, the trip is approved. Once the trip has been completed, incurred expenses are entered into an expense report during the Expense Report Creation phase. This report is usually trip specific. The traveler submits the report to the system per the Expense Report Submission & Approval step. The report is reviewed by the manager and approved if there is no missing information and the report fully complies with company policies. Once the report is approved, reimbursements are initiated by the system and submitted per the Expense Reimbursement phase. At the same time, trip related transactions are entered into the company's accounting or ERP system. Finally, Travel Expense Data is exchanged with the system. Here, total budgeted costs for the trip are compared to actual trip costs – including incurred expense trend, preferred hotels, partner discounts, maximum daily allowances for meals, unexpected expenses, financial conversion fees, entertainment expenses and other parameters.

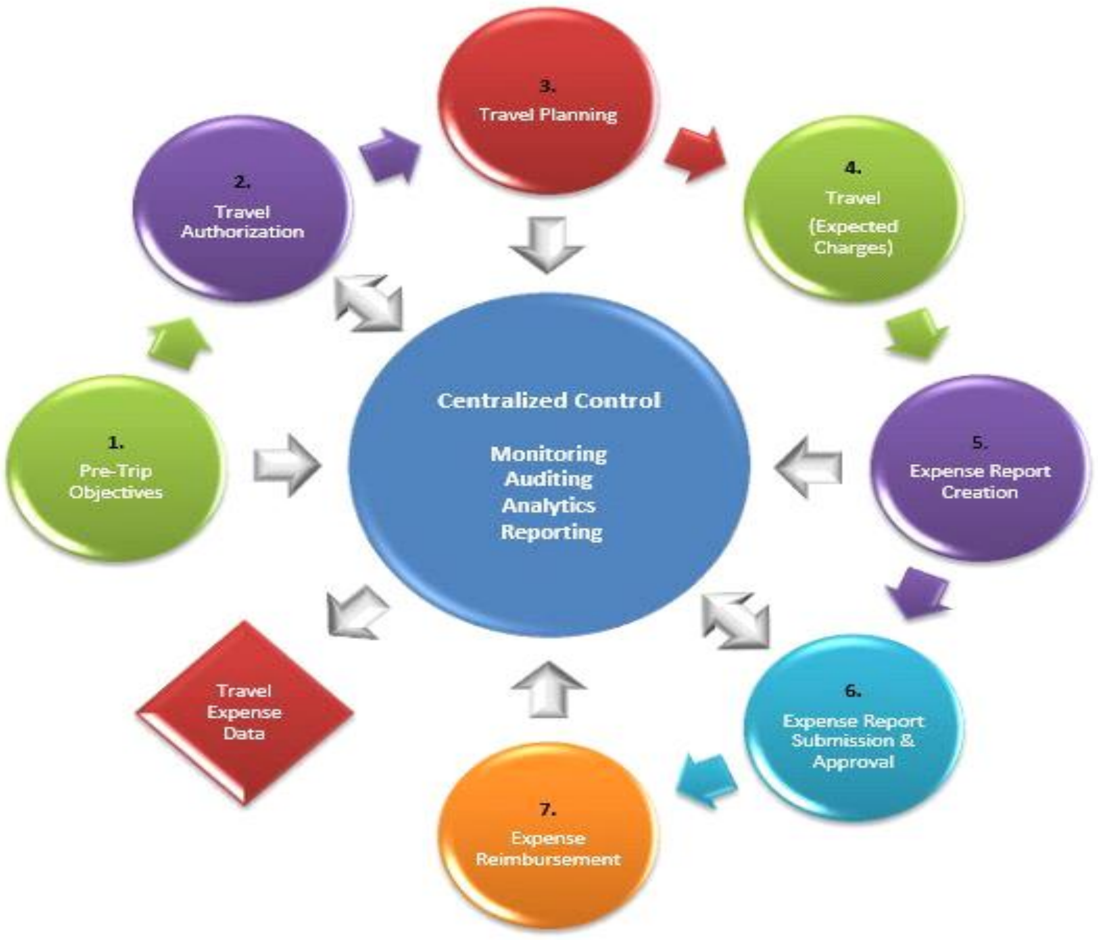


Figure 24: Full life cycle of Travel Expense Management

5.3. Travel Expenses Management Automation (TEMA)

The rising cost of business travel is one of the leading reasons that companies are scrutinizing and more closely managing employee business and travel expenses. According to the Global Business Travel Association (GBTA) Foundation, the research arm of the GBTA, their Industry Pulse: Business Travel Buyers Sentiment survey indicates that expected increases in air and hotel rates are major contributors to escalating corporate travel costs. Companies need to manage these expenses against internal company policies and external government compliance and regulations. Employee business expense data can reveal critical metrics that allow organizations visibility into their employee's business expenses to take proactive steps to reduce overall spend through improved communication, policies, compliance and even vendor negotiations. Travel Expense Management Automation (TEMA) is critical component in an organization's business strategy. While controlling costs, increasing employee satisfaction and improving bottom line performance, TEMA enables companies to streamline the processes involved in managing employee business and travel expenses. TEMA also tracks compliance and decrease the length of time required for each step in managing the expense report process. Since employee business and travel expenses are among the largest controllable costs for most organizations, opportunities for savings can be significant. Simply by automating the expense reporting process with either an on-premise or Software-as-a-Service (SaaS) deployment, an organization can expect to recognize a significant cost savings from reduced labor costs related to the processing of expense reports. Expense management automation dramatically cuts the cost of processing expense reports while increasing employee satisfaction by shortening reimbursement cycles.

5.4. Goal and Benefits of Automation of Travel Expense Management

Goal of TEMA is clear - to improve travel expense management, to reduce operational costs. There are numerous motives for organizations to deploy an automated solution for managing their employee business and travel expenses, including:

- Spend Control
- Ensure Compliance
- Fraud Detection
- Cost Reduction

- Vendor Analysis and Negotiations
- Operational Savings
- Traveler Satisfaction

Though many leading enterprises have moved to automate their expense management, there are still corporations that continue to manage their expense claims either via a paper, a spreadsheet based system, an outdated and costly legacy system or one developed in-house.

Many of these businesses often feel trapped into continuing to use their current system because they are worried about the cost to change; whether this is the cost of new software, the cost of the implementation project, or the cost of deploying resources to manage the project. However, when between 7% to 10% of an average company's budget relates to travel and entertainment expenses (Aberdeen Group report, 2013), badly managed expense management processes can impact on the enterprise's financial performance.

Some challenges in the manual expense reporting system:

- Manual & error prone process. Expense reports have to be manually filled out and sent for approval. Manual work is required to move the expense report through the approval process. Accounting staff are required to manually enter the details of expenses and receipts into the company's finance system in order to reimburse employees.
- Paper based process. Expenses, receipts, reimbursement checks are all paper based. All of them have to be stored and archived periodically which is a tedious task and adds to storage and archival costs.
- Lack of visibility in employee spending. Air, hotel, car and other service providers are known only after submitting the expense report. No prior negotiations are possible with preferred suppliers to achieve the best services and discounts available.
- Inability to impose travel policies. As the expense report process happens after the expenses are incurred, there is no possibility of imposing travel policies and spend limits on expenses prior to travel.
- High processing costs. The total cost of processing an expense report includes the cost of paper, resources, storage, archival, approval, manual entry, and out of policy claims.
- Longer reimbursement cycle times. Claims and receipts are manually verified and then entered into financial system. The expense report is generated and followed by reimbursement. This manual effort results in wasted time. Even today, according to

Aberdeen Group report, more than 43% of businesses follow the traditional manual process to manage their expenses. They do not find a need to automate their existing process.

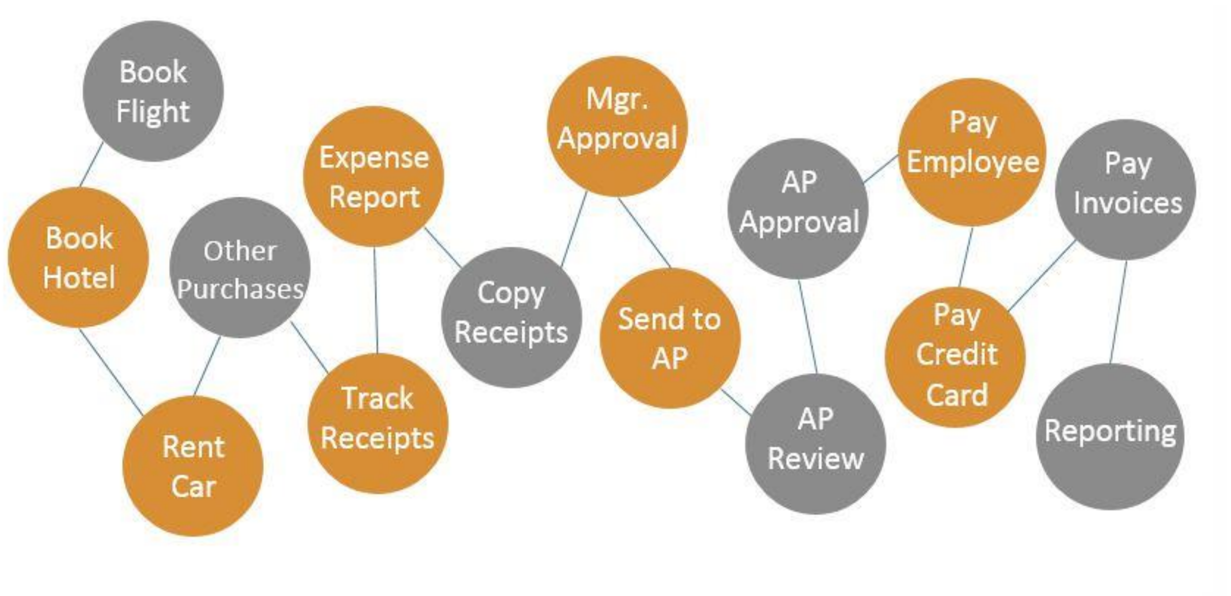


Figure 25: Manual Travel Expense Management

According to the PayStream (TEM 2009, Adoption Survey Report), it was found that the average cost of manually processing an expense report is $28.21 compared to $6.19 for those using an end-to-end automated expense management solution.

Businesses use various solutions for automating each of these areas. To reduce processing costs and to overcome challenges in the manual system, companies are encouraged to move towards using an automated solution for managing the entire travel expense management process. Automation improves compliance to corporate travel policies by 31%, lowers processing costs by 80%, and reduces the reimbursement cycle time from weeks to days. It also increases visibility and significantly increases ROI. Automation also eliminates the manual tasks, paperwork, time delays, and reduces the risk of fraud.

Here are top 10 reasons why using automation can improve expense management process.

- Drive Cost Efficiency
- Ensure Internal and Regulatory Compliance
- Minimize Fraud

- Integrate Travel into company's Expense
- Stay Flexible and Scalable with Cloud-based Solutions
- Support Business Strategy
- Managing Risk Effectively
- Reinforce Company Policy to Manage Behavior
- Expense on the Move

According to research by PayStream Advisors, many organizations have seen improved travel policy enforcement, lower processing costs, increased visibility, and improved employee satisfaction and reimbursement time, see Figure 26.

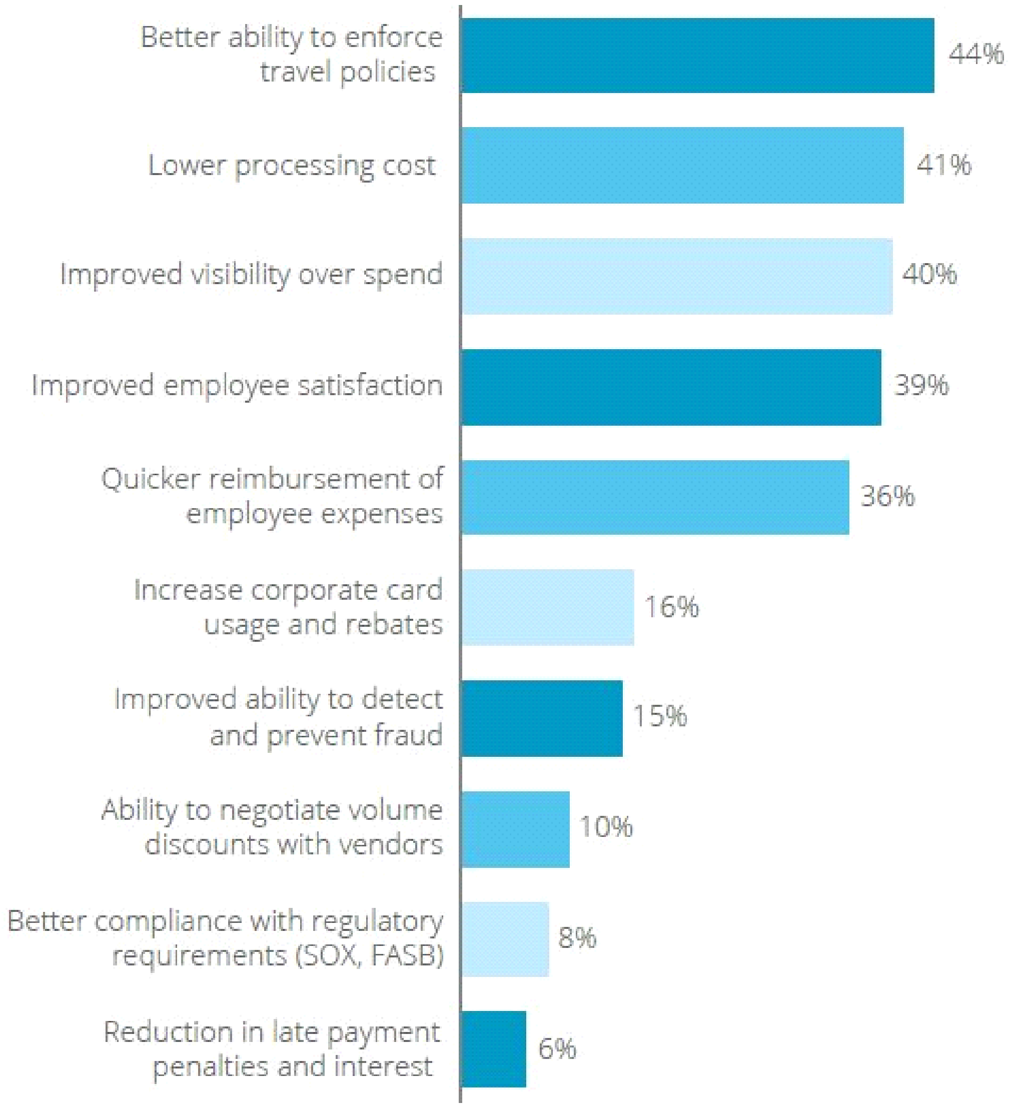
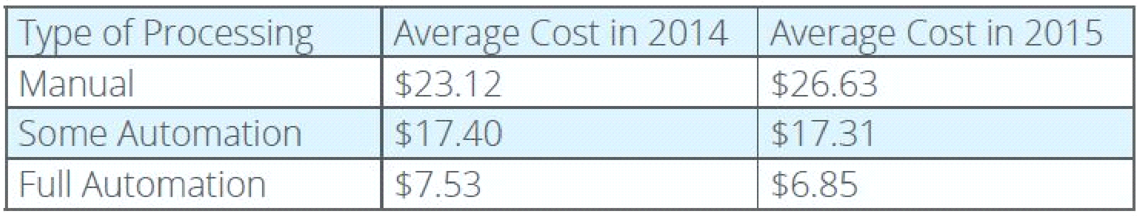


Figure 26: What are biggest benefits you have achieved by automating your T&E process?

5.5. Current situation of Travel Expense Management Automation

According to the Expense Management Trends Report survey conducted in December 2015, over one-third of organizations are using a manual process to manage their T&E costs. A manual process involves the use of electronic spreadsheets, paper reports and receipts, and postal mail. Manual expense reporting requires extensive time to create, approve, reconcile, process, and reimburse the expense reports, along with cumbersome manual data entry across multiple systems to capture all relevant information. Errors or violations can cause delays throughout the process—and if no errors or violations are found, erroneous payments may be made. A manual process also does not include time and costs associated with any audit investigations related to expense management. PayStream Advisors surveyed over 200 individuals employed in many different industries, compiling data reflecting current attitudes towards and usage of TEM automation software. Among the majority of the surveyed organizations, T&E processes are still mostly manual, and these organizations experience difficulties stemming from a lack of automation. However, research also indicates that organizations desire improved employee control and cost reduction, and these needs are driving AP departments towards TEM solutions.

PayStream's research shows in 2014, the average cost of a manually processed expense report was $23.12. In 2015, the average cost is $26.60—quite high, compared to the average cost of $6.85 per report with a fully automated TEM solution. These costs place heavy burdens on organizations processing hundreds of thousands of expense reports each year.



Type of Processing	Average Cost in 2014	Average Cost in 2015
Manual	$23.12	$26.63
Some Automation	$17.40	$17.31
Full Automation	$7.53	$6.85

Table 3: What is your average cost to process a single expense report?

High processing costs go hand-in-hand with manual T&E operations. Research shows (PayStream Advisors) that 45 percent of organizations have no TEM software, see Figure 27. However, an impressive 50 percent use at least some T&E automation. Figure 27 also shows a large gap in automation between SMEs and large corporations. Of the small organizations surveyed, 63 percent reported fully manual T&E processes, and 25 percent have in-house automation. In contrast, only 9 percent of larger enterprises operate manually, and 81 percent have some in-house automation.

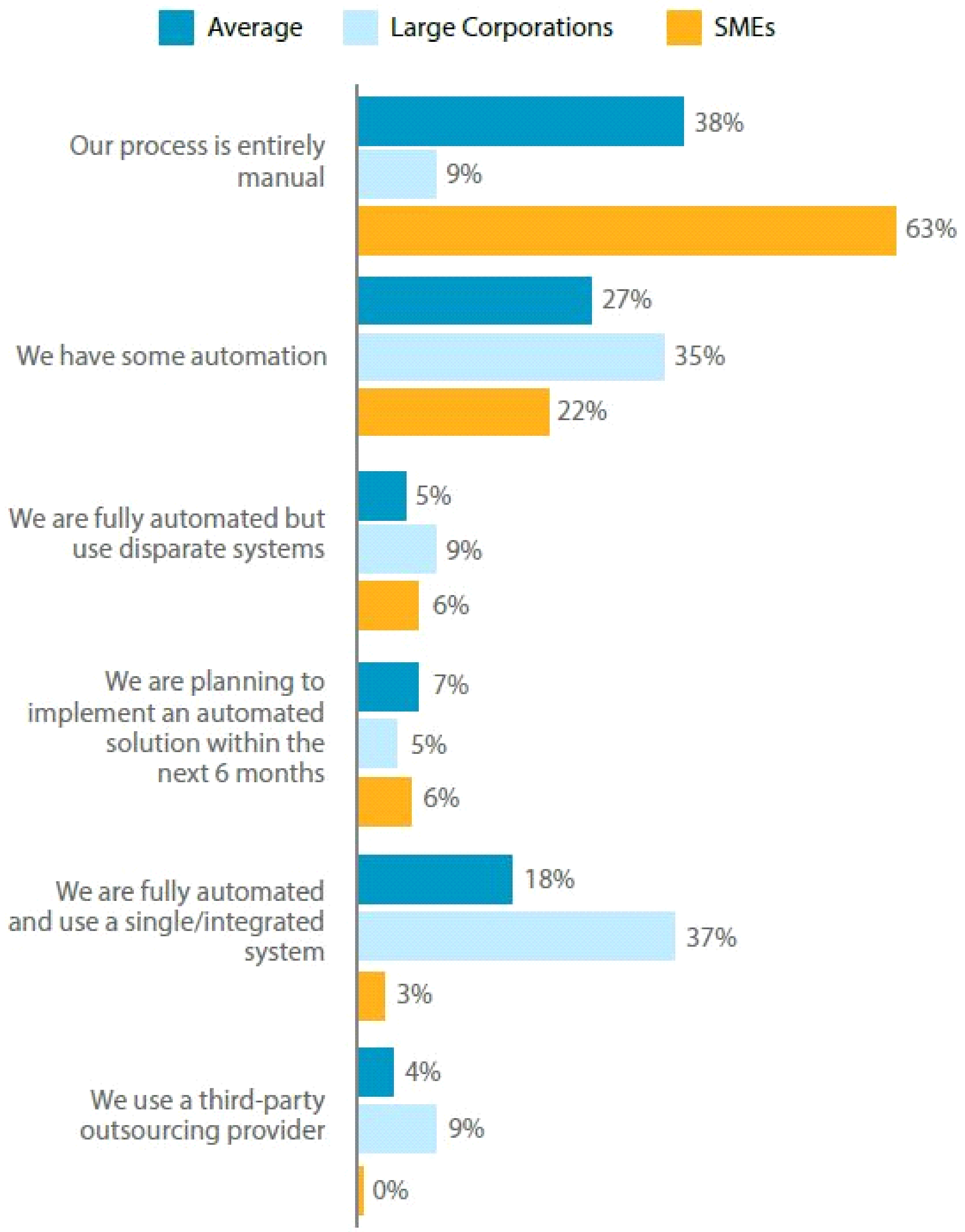


Figure 27: Which of the following statements best describes the extent of automation in your TEM process?

Organizations that have not adopted a TEM solution demonstrate reluctance towards change—many believe that current processes work, see Figure 28. While this may be true, it does not mean that current processes work well.

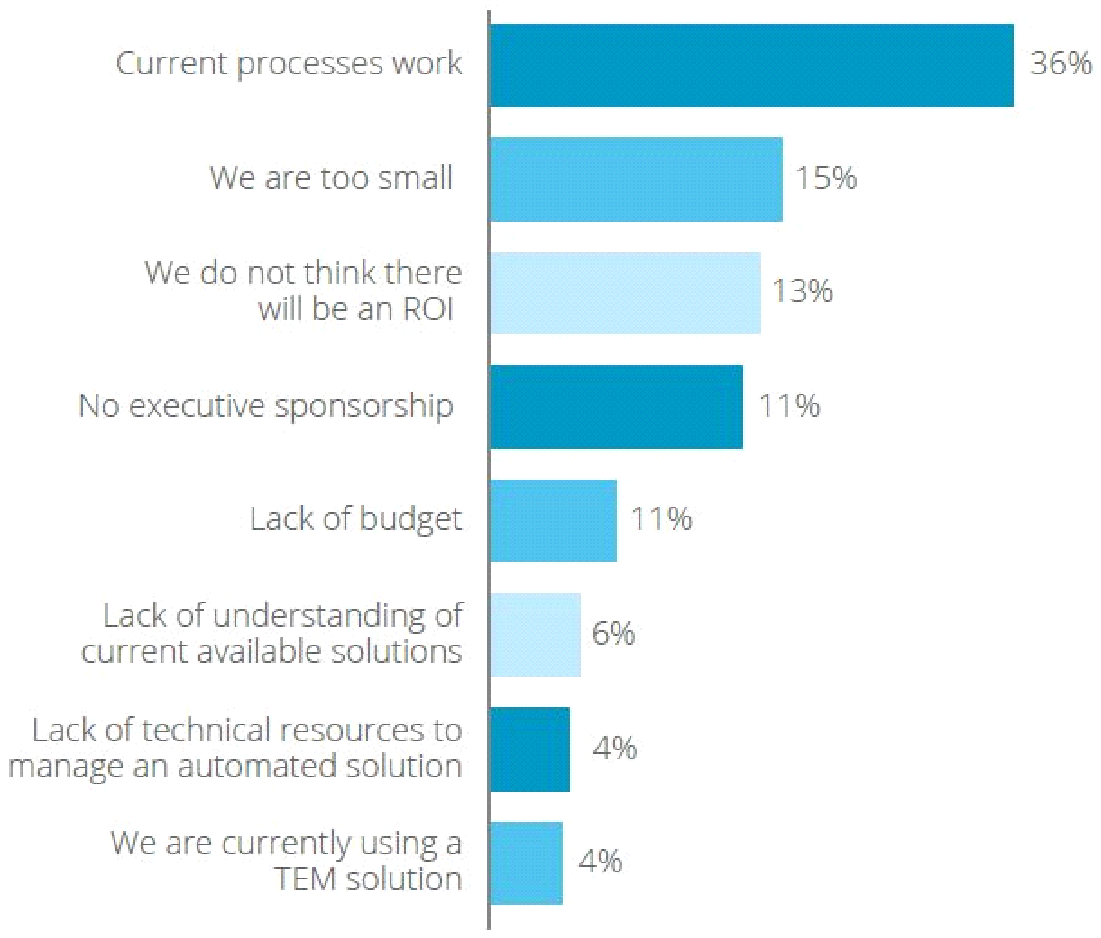


Figure 28: What is the reason your organization has not automated the TEM process?

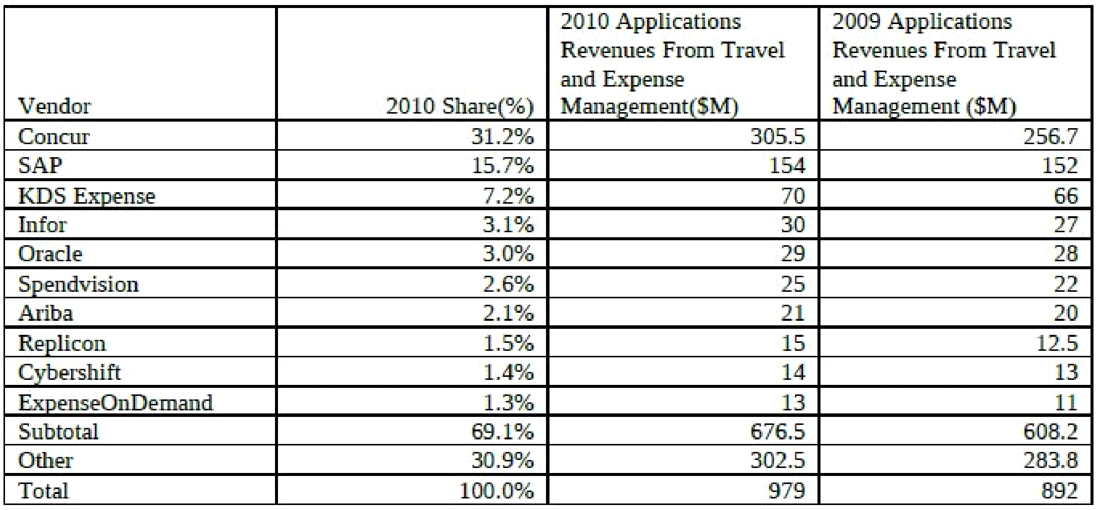
According to figure 28, which was result of research by PayStream Advisors, 36% companies think current processes work. As you can see, 15% companies don't implement TEMA because of their scale. And considerably amount of companies – 13 % of them think there is no RIO (return on investment) in case they implement automated TEM. Lack of sponsorship is also one of the reasons they don't automate TEM. 11% companies can't simply afford to buy those technologies (software, etc). Only 4 % of them say they are using some TEM solution.

As you can see, really large amount of companies don't automate their travel expense management because simply they can't afford.

Current market. As the travel and expense management applications market began to expand in 2010 with renewed corporate spending on business travel and services requisitioning, a number of trends started to manifest themselves. The ubiquity of smart phones has meant corporate travelers placing greater emphasis on mobile solutions to help them manage their travel plans and expense reporting. Concur’s recent purchase of TripIt, which leverages mobile devices to organize and share travel information for travelers, underscores the shift of delivering travel and expense management experiences from the desktop to smart phones.

For more than a decade travel and expense management applications have been mostly implemented at large corporations with tens of thousands of employees. That began to change in the past few years as on-demand applications became popular among small and mid-sized organizations that found the delivery model flexible and affordable.

Top 10 Applications Vendors. The following table lists the 2010 shares of the top 10 applications vendors in the travel and expense management market and their 2009 to 2010 applications revenues (license, maintenance and subscription) from the market. (Source: “Apps Run The World, 2011”)



Vendor	2010 Share(%)	2010 Applications Revenues From Travel and Expense Management($M)	2009 Applications Revenues From Travel and Expense Management ($M)
Concur	31.2%	305.5	256.7
SAP	15.7%	154	152
KDS Expense	7.2%	70	66
Infor	3.1%	30	27
Oracle	3.0%	29	28
Spendvision	2.6%	25	22
Ariba	2.1%	21	20
Replicon	1.5%	15	12.5
Cybershift	1.4%	14	13
ExpenseOnDemand	1.3%	13	11
Subtotal	69.1%	676.5	608.2
Other	30.9%	302.5	283.8
Total	100.0%	979	892

Table 4: TEMA applications revenues.

5.6. Future of Travel Expense Management Automation

Based on the need for web-based simple prototype solution for mid and small size companies, I developed TEMALite application. In the future, web-based applications as TEMALite will be even more popular. Because there are advantages of this kind of web-based applications, they are:

- *Security*. Web-based applications usually have one central server and it's enough to protect that central server. If for example, it's not web based and it should be installed in every computer, then every computer which has this app should be protected from attacks. It would make the process more difficult and more costly.
- *Space*. As mentioned above, web-based apps have just one server and it can be updated, modified from one place.
- *Compatibility.* Web-based apps are supported by almost all OS-es and they are cross-platform and cross-browser.
- *System migration.* In near future all business applications will be migrating to claud based systems and it's very easy to migrate from web-based systems to cloud-based systems.
- *Availability*. Web-based applications can be accessed by many types of devices unlike other software systems which requires installment before using.

In hard economic times like this, most companies will seek better solutions to improve their costs and performance, applications as TEMALite will be widely used among mid and small size companies.

According to Aberdeen Group reports (2015), presently 27% companies still use a paper based process, this will nearly vanish and while about 25% of users will still rely on systems that are hosted in-house, the balance will move to cloud based systems or to a hybrid of cloud and in-house systems. This data is summarized in the graph (SaaS in the graph refers to Software as a Service where you hire the software and work purely in the cloud).

Interfacing with ERP and other backend systems - The below graph also indicates the nearly two fold increase businesses will see in hybrid systems. These systems use specialist cloud based expense management systems to get the best out of their expenses. At the same time these systems interface with ERP solutions to deliver greater strategic value to users.

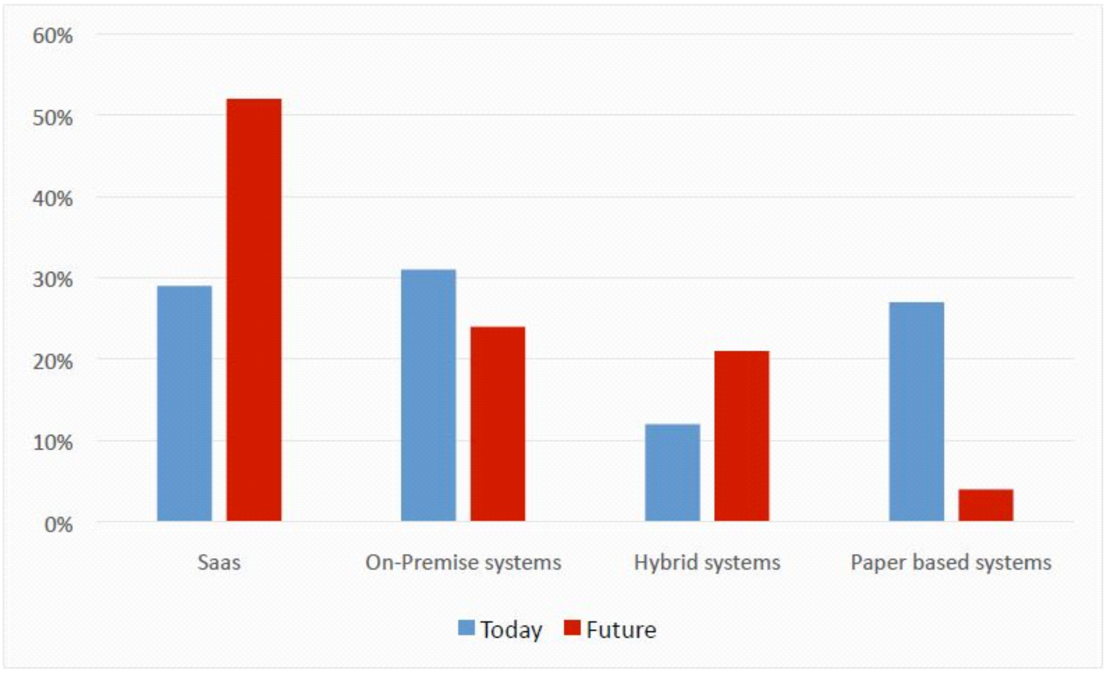


Figure 29: Trends in Use of business Expense Management Systems

6. Application Design

In the chapters above, we can see that many mid and small size companies can't afford to buy or customize major TEMA solutions, I feel that there is a need to create affordable simple TEMA solution for mid and small size companies. For research purposes, TEMA application is developed and named TEMALite. TEMALite is web-based application. Following chapters will show how this application is developed.

6.1. TEMA Methodology

Methodology could be explained in a set of recommended practices. It is defined as organized, documented set of procedures and guidelines that describes how something is done. Methodology includes the frameworks, techniques, methods, patterns and procedures that are used to accomplish a set of goals and objectives. This step shows what methodology will be used so that to deploy TEMALite application in the companies.

There are 6 steps to deploy TEMALite in companies:

- System Analysis
- System Design
- System Development
- Testing
- Implementation
- Maintenance

Step 1 - System Analysis. Systems analysis is the examination of the business problem that the organization plans to solve with TEMALite application. This stage defines the business problem, identifies its causes, specifies the solution, and identifies the information requirements that the solution must satisfy. Organizations have three basic solutions to any business problem:

- Do nothing and continue to use the existing system unchanged.
- Modify or enhance the existing system.
- Develop a new system.

Considering the fact that almost most mid and small size companies heavily rely on paper based manual Expense Reporting, third option will be chosen – new system (TEMALite) will be developed.

The systems analysis stage produces the following information:

- Strengths and weaknesses of the existing system (paper based manual system)
- Functions that the new system (TEMALite) must have to solve the business problem
- User information requirements for the TEMALite Application

Armed with this information, developer can proceed to the systems design stage.

Step 2 – System design. Systems analysis describes *what* TEMALite must do to solve the business problem, and systems design describes *how* the system (TEMALite) will accomplish this task. The deliverable of the systems design phase is the technical design that specifies the following:

- System outputs, inputs, and user interfaces
- Hardware, software, databases, telecommunications, personnel, and procedures
- How these components are integrated

Systems design encompasses two major aspects of the new system:

- Logical systems design states *what* the system will do, with abstract specifications.
- Physical systems design states *how* the system will perform its functions, with actual physical specifications.

Logical design specifications include the design of outputs, inputs, processing, databases, telecommunications, controls, security, and IS jobs. Physical design specifications include the design of hardware, software, database, telecommunications, and procedures.

Step 3 – System Development. Once step 2 is finished, the development phase for TEMA will emerge. Since technology is the enabler of TEMA, this step starts with getting hardware and software which are needed to develop the TEMA.

TEMALite is built on .NET environment using C# programming language. Following hardware and software are required:

Hardware: PC with a 1.6GHz or higher processor, 2 GB (32 Bit) or 4 GB (64 Bit) RAM, 8GB of available hard disk space after Operation System installed, 5400 RPM hard disk drive, DirectX 9 capable video card running at 1024x768 or higher-resolution display

Software: Supported operation systems: Windows 7 or 10, Windows Vista, Windows XP, Windows Server 2003, Windows Server 2008.

Supported Web Servers: Internet Information Service (IIS) 6.0 (or above), ASP.NET 4.0

Supported Databases: MSSQL 2005 or above

Supported browsers: Microsoft Internet Explorer 9 and above, Mozilla Firefox 6.0 and above, Google Chrome 1.x (latest version), Apple Safari 2.x,

Visual Studio 2012 or above is required.

TEMALite is developed based on the theories and methodologies which were obtained during literature review, paired with the things which were learned from the requirements analysis. TEMALite will be developed based on Object-Oriented methodology using best practices of Agile Methodology. The researcher was provided with detailed concepts of TEMA in the literature review. The TEMA will be a personalized, The personalization is accessible through a password and username.

Step 4 – Testing. Thorough and continuous testing occurs throughout the programming stage. Testing checks to see if the computer code will produce the expected and desired results under certain conditions. Testing requires a large amount of time, effort, and expense to do properly. However, the costs of improper testing, which could possibly lead to a system that does not meet its objectives, are enormous. Testing is designed to detect errors ("bugs") in the computer code. These errors are of two types: syntax errors and logic errors. Syntax errors (e.g., a misspelled word or a misplaced comma) are easier to find and will not permit the program to run. Logic errors permit the program to run, but result in incorrect output. Logic errors are more difficult to detect, because the cause is not obvious. The programmer who is writing code for TEMALite Application must follow the flow of logic in the program to determine the source of the error in the output. As software increases in complexity, the number of errors increases, making it almost impossible to find them all. This situation has led to the idea of "good enough software," software that developers release knowing that errors remain in the code. However, the developers feel that the software will still meet its functional objectives. That is, they have found all the show-stopper bugs, errors that will cause the system to shut down or will cause catastrophic loss of data.

Step 5 – Implementation. Implementation is the process of converting from the old system (paper based, manual) to the new system (TEMALite). Organizations use four major conversion strategies: parallel, direct, pilot, and phased. In a parallel conversion process, the old system and the new system operate simultaneously for a period of time. That is, both systems process the same data at the same time, and the outputs are compared. This type of conversion is the most expensive, but also the least risky. Most large systems have a parallel conversion process to lessen the risk. In a direct conversion process, the old system is cut off and the new system is turned on at a certain point in time. This type of conversion is the least expensive, but the most risky if the new system doesn't work as planned. Few systems are implemented using this type of conversion, due to the risk involved. The pilot conversion process introduces the new system in one part of the organization, such as in one plant or in one functional area. The new system runs for a period of time and is assessed. After the new system works properly, it is introduced in other parts of the organization. The phased conversion process introduces components of the new system, such as individual modules, in stages. Each module is assessed, and, when it works properly, other modules are introduced, until the entire new system is operational.

6.2. Software Development

Web based application will be developed based on company's needs. There will be mainly 4 actors:

- Traveler
- Controller
- Accountant
- Manager

Based on the functionality, access level and position, for each actor separate pages and access levels are created. All functionalities are considered and system will be built in the way that all process automated (made online, accessible anywhere outside of company).

TEMALite will provide the organization with best possible activities to achieve better travel expense management by implementing TEMA concepts and assimilating essential components into it. Following figure shows how TEMALite works.

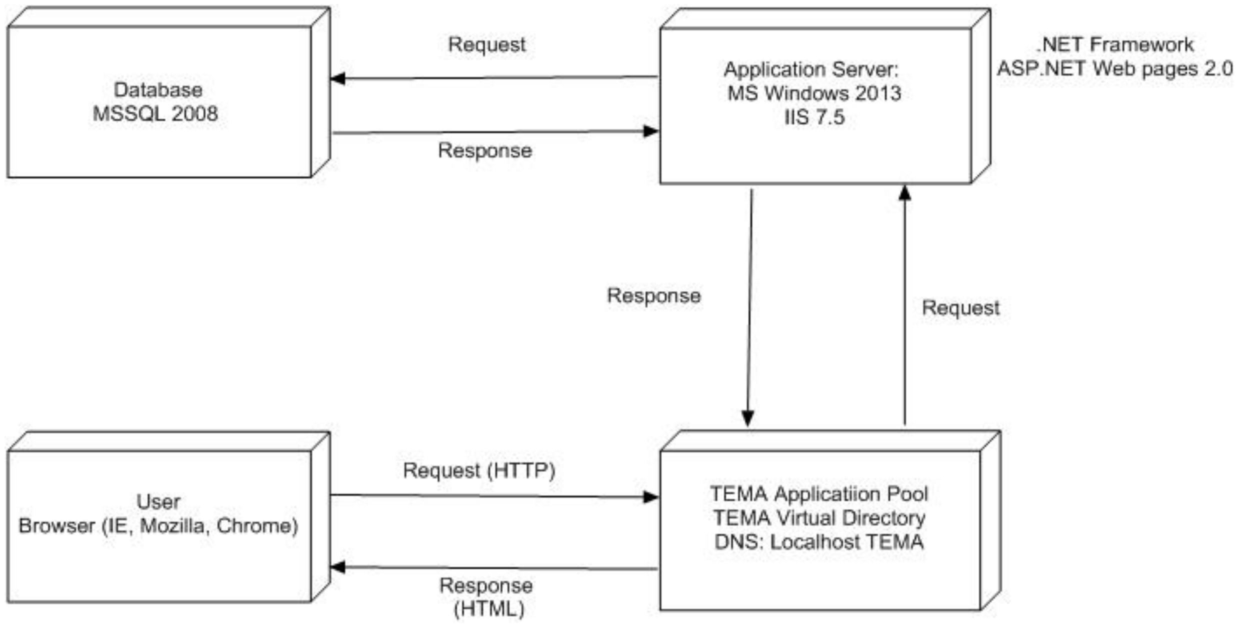


Figure 30: TEMALite components

As it can be seen from the figure 30, users of TEMALite first send HTTP request to Domain Name System (DNS), DNS connects with Application server in turn. Application server receives the request, analyses it, sends request to Database. Database responds to request and request is returned as response respectively. Finally user gets response as plain HTML since it's web-based software.

6.3. Use Case Model

Following use case diagram will be helpful in capturing TEMA's responsibility for its users. All use cases are shown and TEMA functionality is displayed in Figure 31.

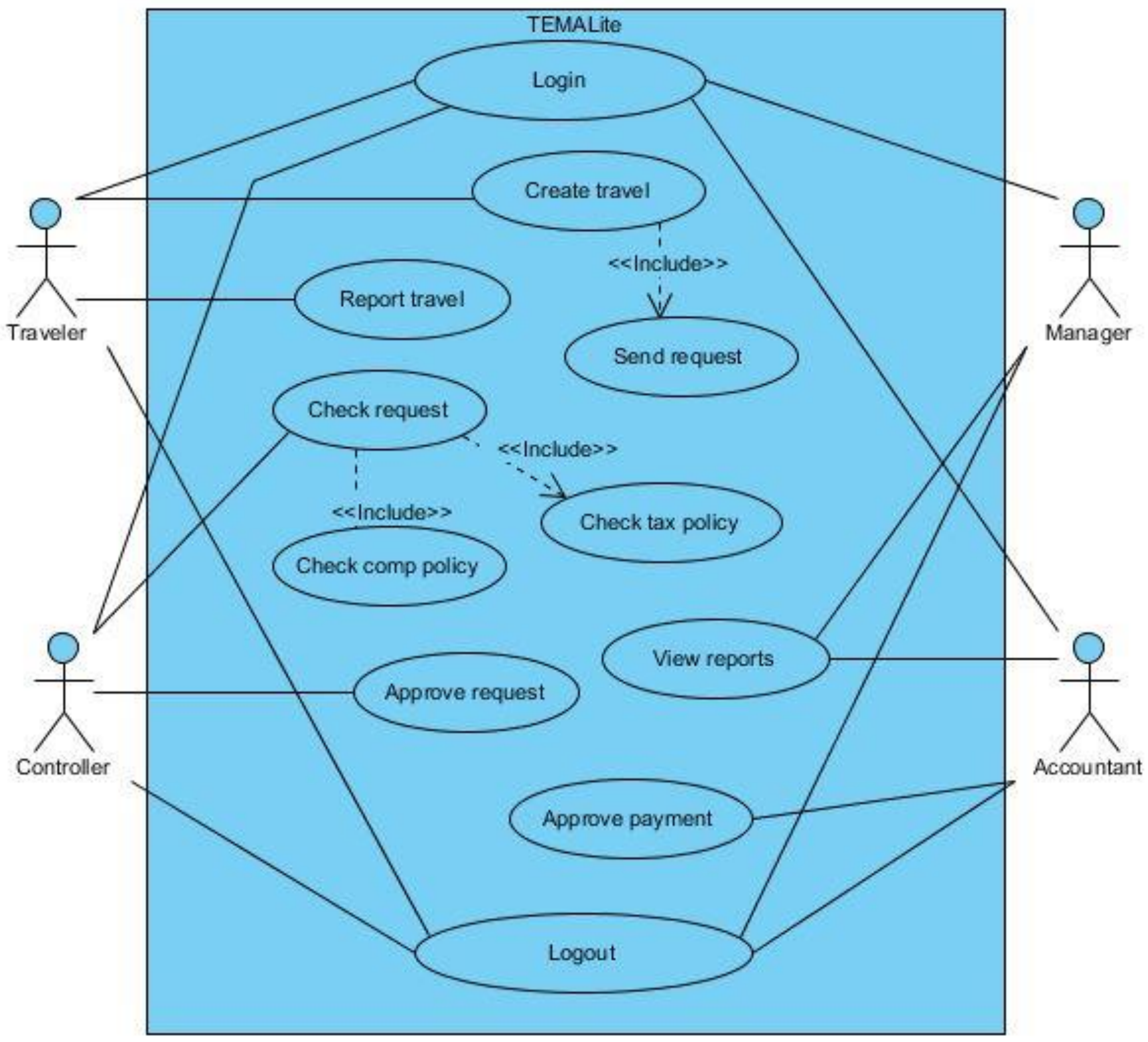
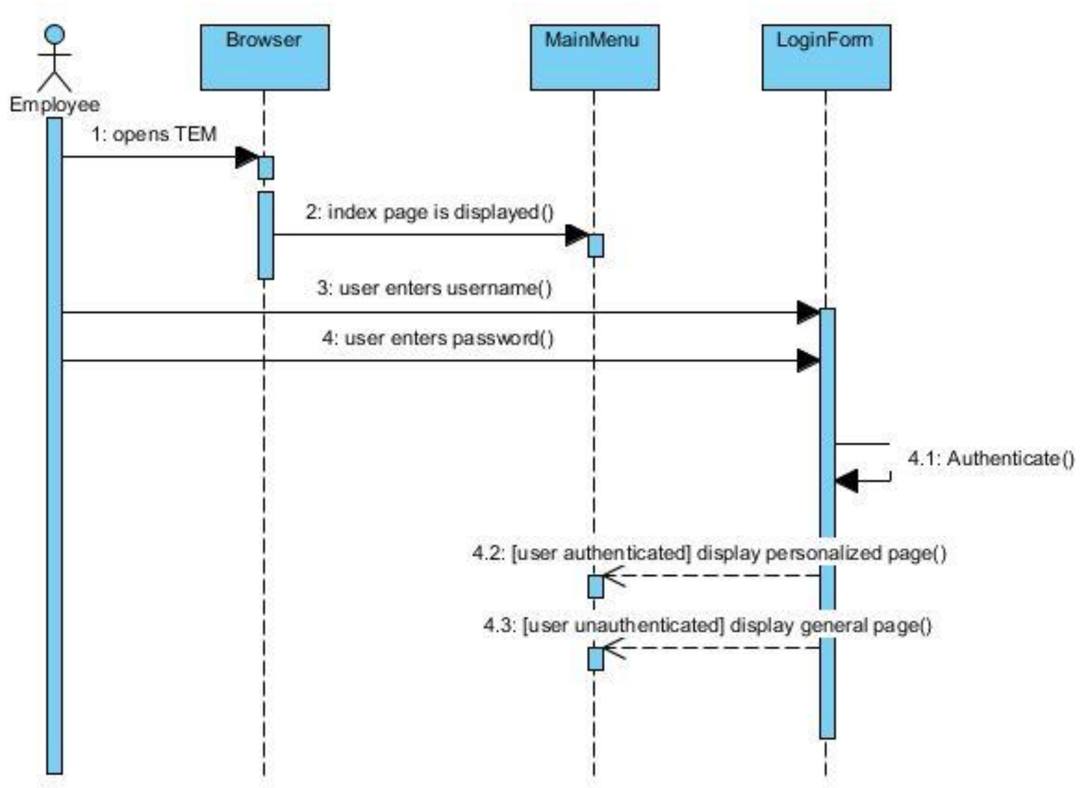


Figure 31: Use-case diagram for TEMLite

6.4. Sequence diagrams

Following sequence diagrams describe behavior of TEMLite, interaction between TEMLite and its environment.



Employee
Browser
MainMenu
LoginForm
1: opens TEM
2: index page is displayed()
3: user enters username()
4: user enters password()
4.1: Authenticate()
4.2: [user authenticated] display personalized page()
4.3: [user unauthenticated] display general page()

Figure 32: Login Sequence Diagram

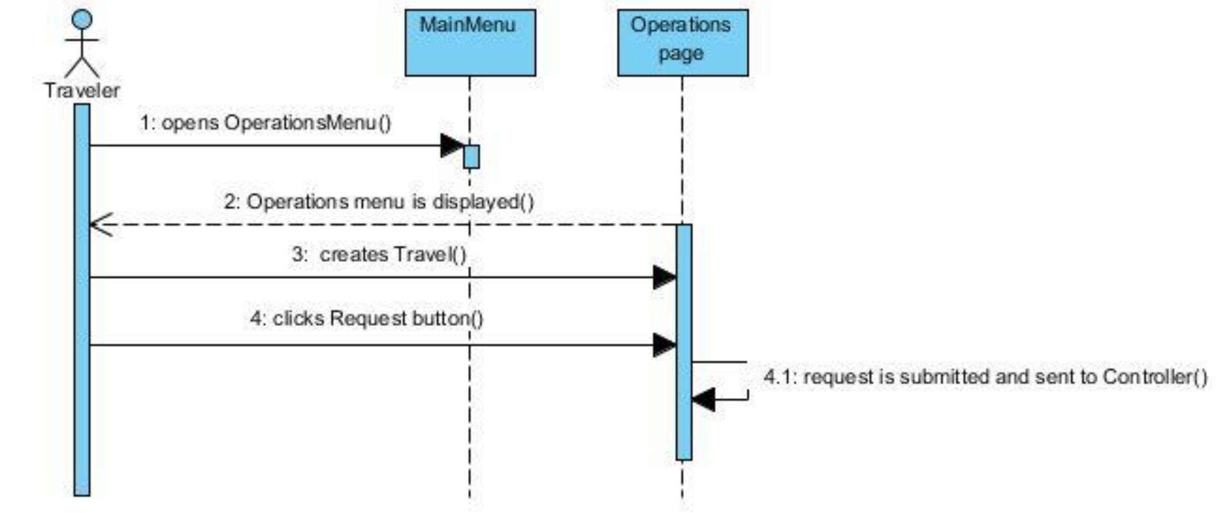

Traveler
MainMenu
Operations page
1: opens OperationsMenu()
2: Operations menu is displayed()
3: creates Travel()
4: clicks Request button()
4.1: request is submitted and sent to Controller()

Figure 33: RequestTrip Sequence Diagram

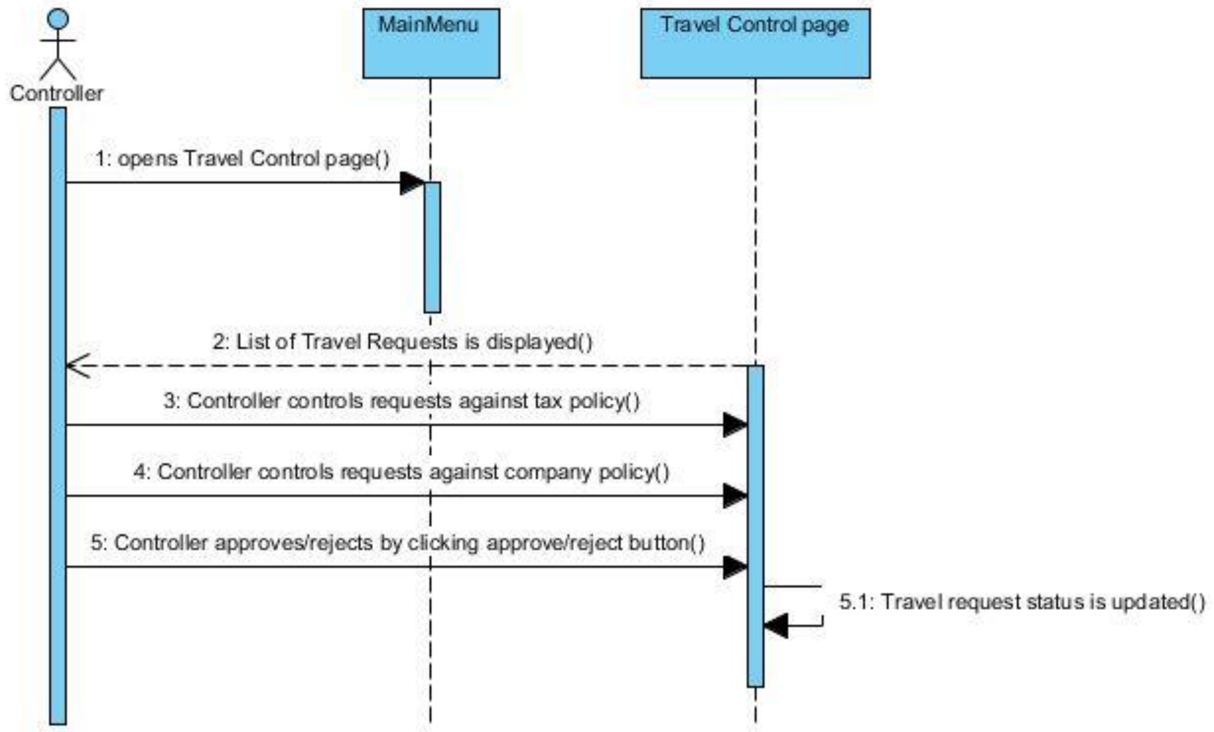
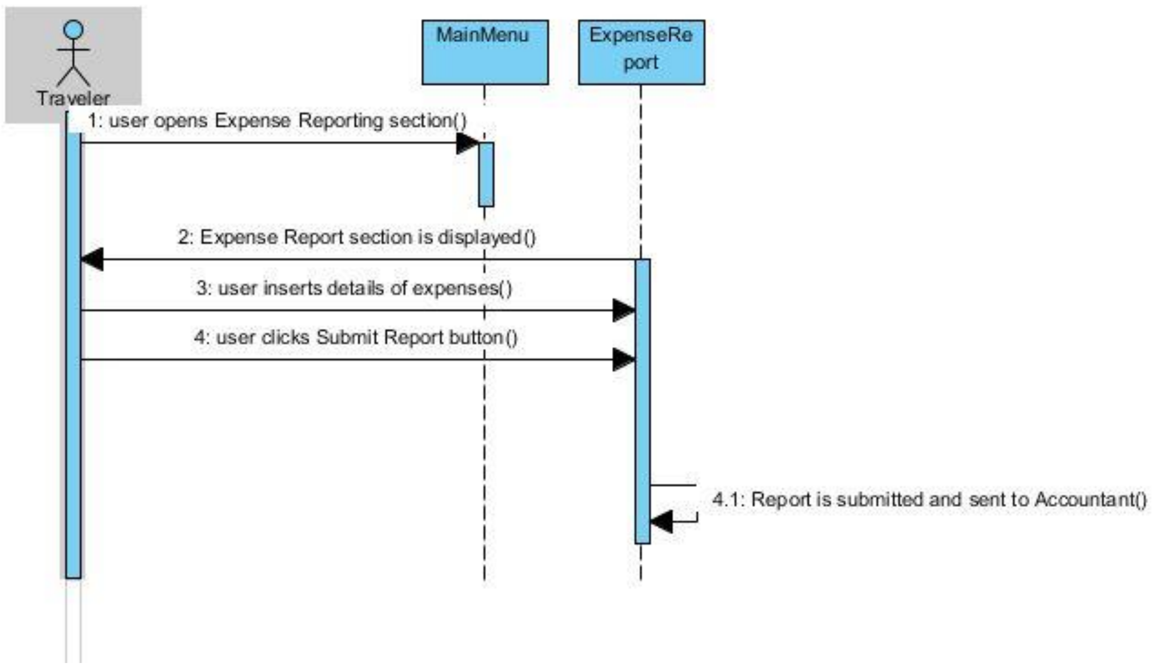
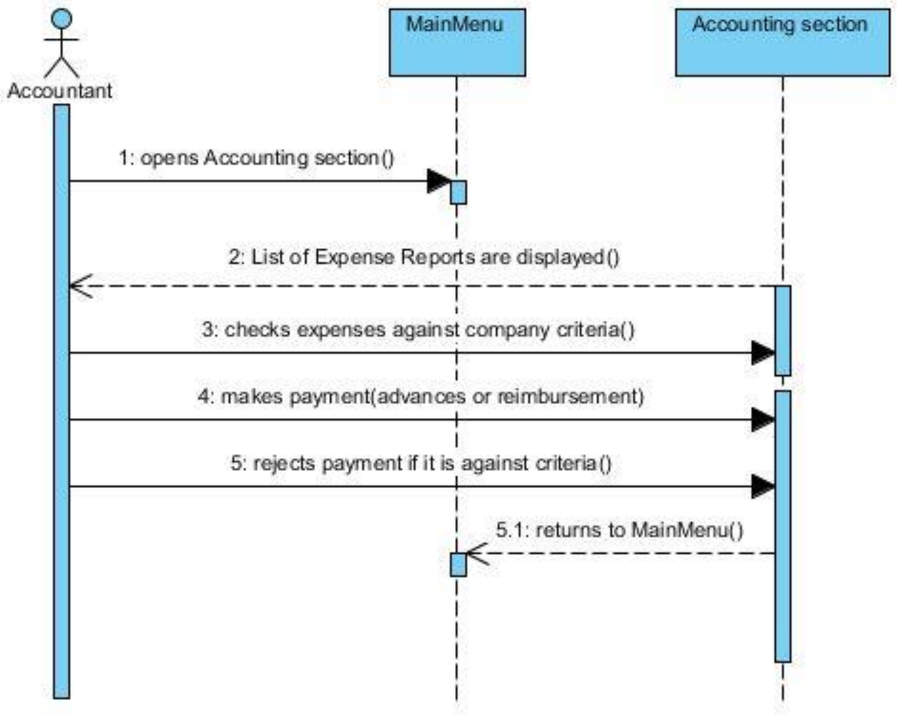

Controller
MainMenu
Travel Control page
1: opens Travel Control page()
2: List of Travel Requests is displayed()
3: Controller controls requests against tax policy()
4: Controller controls requests against company policy()
5: Controller approves/rejects by clicking approve/reject button()
5.1: Travel request status is updated()

Figure 34: CheckRequests Sequence Diagram



Traveler
MainMenu
ExpenseReport
1: user opens Expense Reporting section()
2: Expense Report section is displayed()
3: user inserts details of expenses()
4: user clicks Submit Report button()
4.1: Report is submitted and sent to Accountant()

Figure 35: ReportExpenses Sequence Diagram



Accountant
MainMenu
Accounting section
1: opens Accounting section()
2: List of Expense Reports are displayed()
3: checks expenses against company criteria()
4: makes payment(advances or reimbursement)
5: rejects payment if it is against criteria()
5.1: returns to MainMenu()

Figure 36: Payment Sequence Diagram

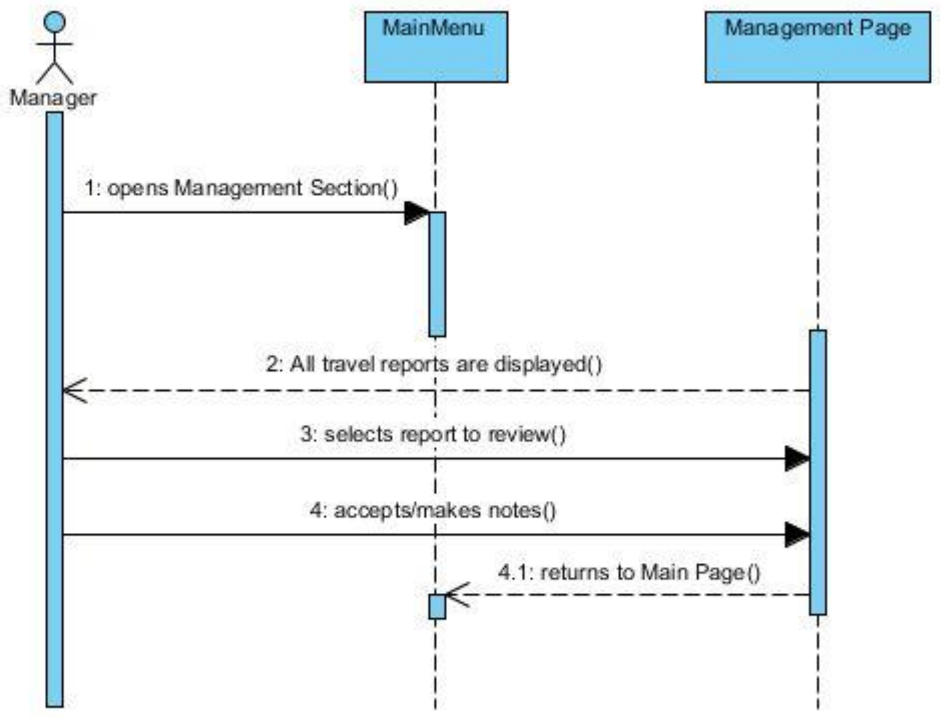

Manager
MainMenu
Management Page
1: opens Management Section()
2: All travel reports are displayed()
3: selects report to review()
4: accepts/makes notes()
4.1: returns to Main Page()

Figure 37: ViewReports Sequence Diagram

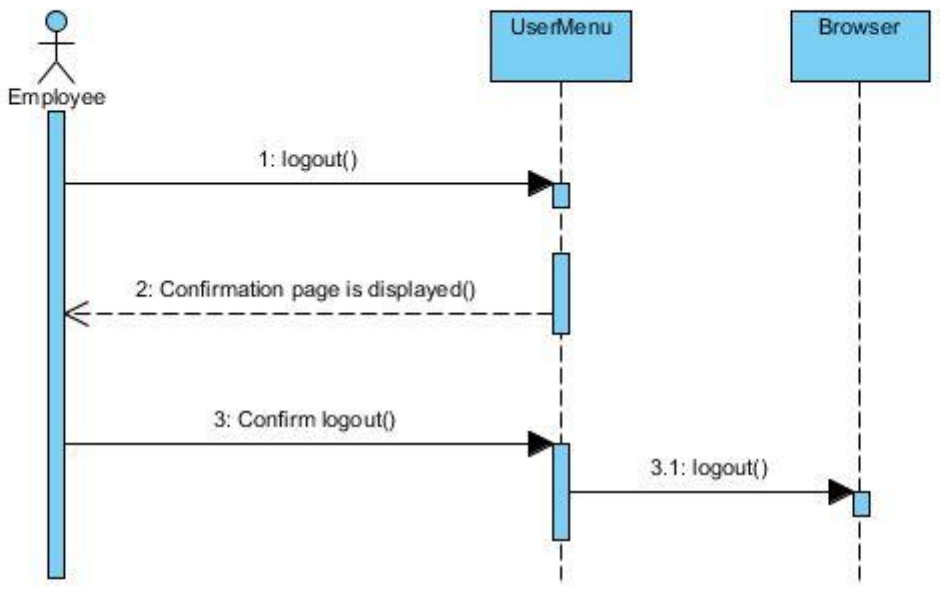


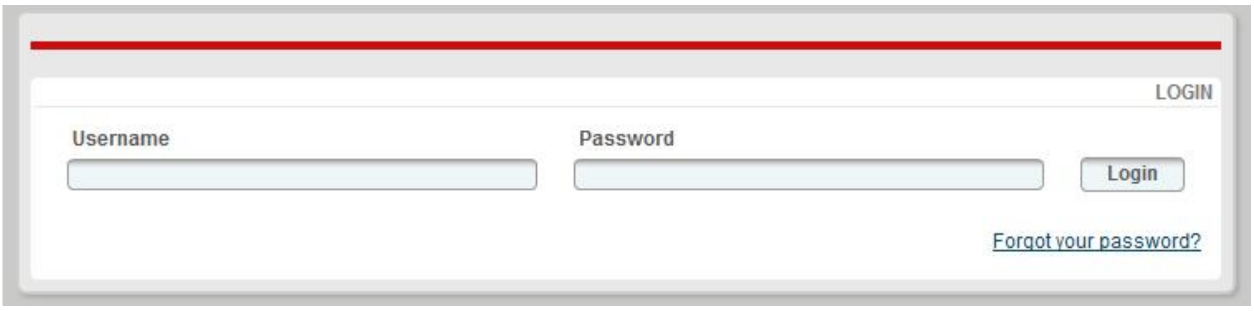
Figure 38: Logout Sequence Diagram

With detailed conceptual design, the TEMALite is developed, then user interfaces will be designed. Screenshots of selected features of TEMALite are shown next chapter.

6.5. User Interface wireframe

wireframe, also known as a page schematic or screen blueprint, is a visual guide that represents the skeletal framework of a web-based system. Following section will provide wireframe for some part of the whole system. It can be viewed and accesses by 3 actors of this system – Traveler, Controller and Manager.

1) Login form. All users of the system can see and login in login form. Depending on the position and access level, they are redirected into different pages.


LOGIN

Username

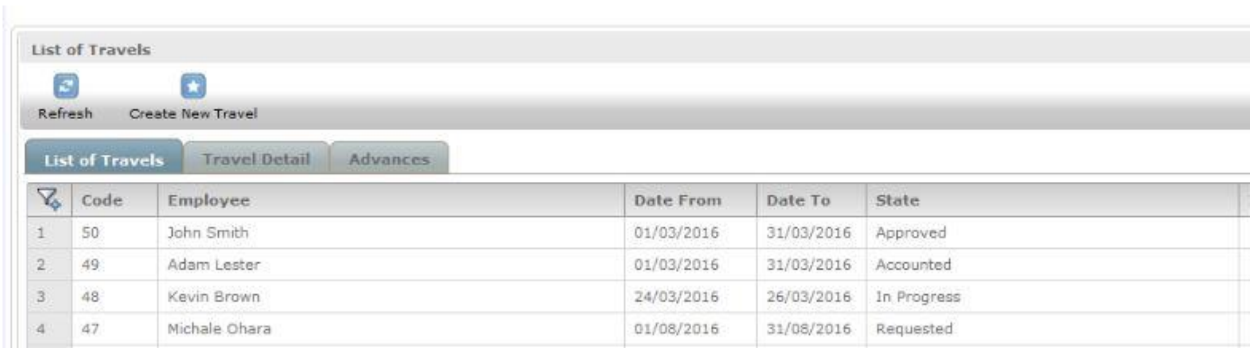
Password

Login

Forgot your password?

Figure 39: Login form.

2) List of travels. It can be accessed by Travelers and Managers. Travelers can see travel details, the progress of their travels if it's approved, accounted, in progress or requested. New travel can be created based on the needs of employee.


List of Travels

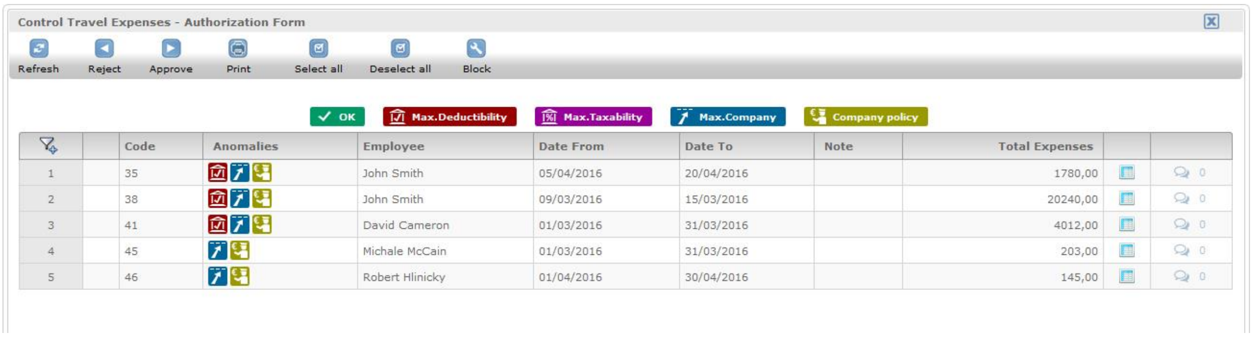
Refresh | Create New Travel

List of Travels | Travel Detail | Advances

	Code	Employee	Date From	Date To	State
1	50	John Smith	01/03/2016	31/03/2016	Approved
2	49	Adam Lester	01/03/2016	31/03/2016	Accounted
3	48	Kevin Brown	24/03/2016	26/03/2016	In Progress
4	47	Michale Ohara	01/08/2016	31/08/2016	Requested

Figure 40: List of travels

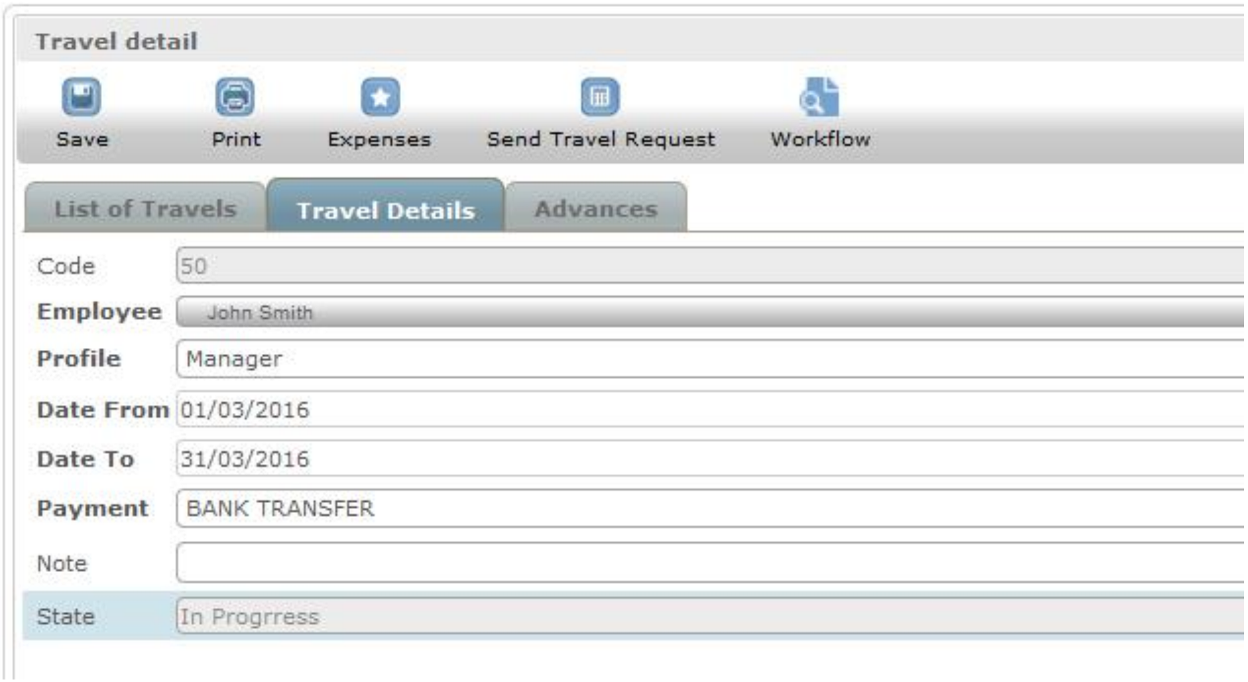
3) Authorization form. This page can be viewed by controller. Controller can see the employees, dates of submission, amounts of expenses and so forth. Controller can check requested travel against company policy, tax policies, deductibility, quotas and so on. Based on that process, controller can approve or reject the travel.



		Code	Anomalies	Employee	Date From	Date To	Note	Total Expenses		
1		35	🟥🟦🟨	John Smith	05/04/2016	20/04/2016		1780,00		0
2		38	🟥🟦🟨	John Smith	09/03/2016	15/03/2016		20240,00		0
3		41	🟥🟦🟨	David Cameron	01/03/2016	31/03/2016		4012,00		0
4		45	🟦🟨	Michale McCain	01/03/2016	31/03/2016		203,00		0
5		46	🟦🟨	Robert Hlinicky	01/04/2016	30/04/2016		145,00		0

Figure 41: Authorization form

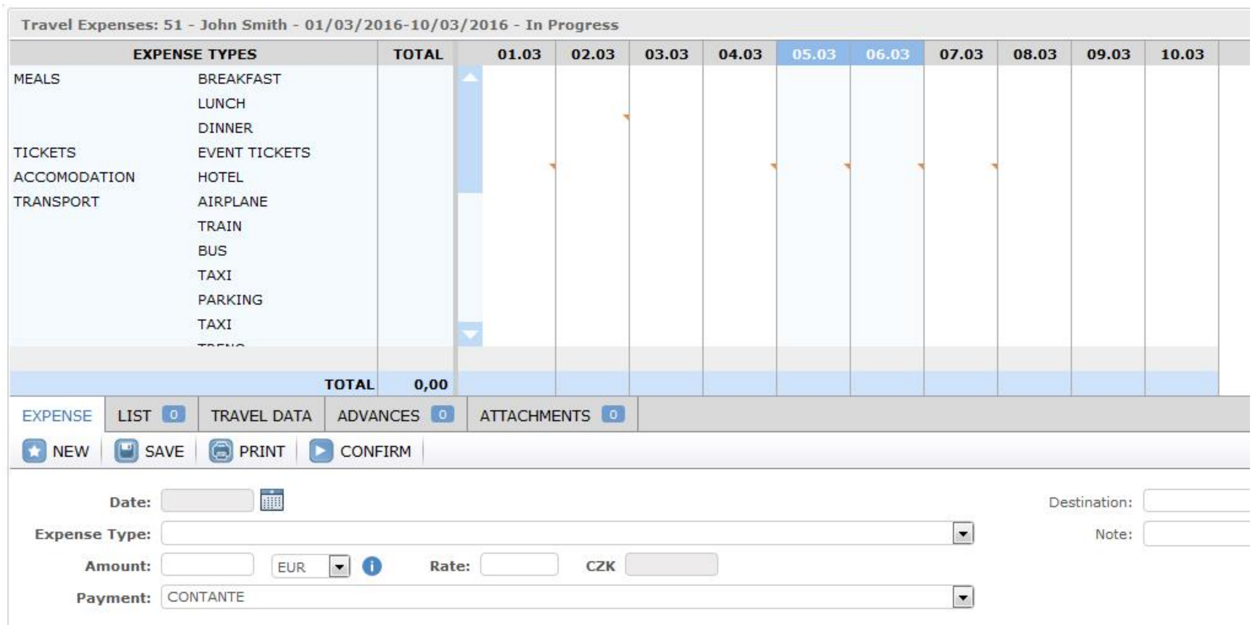
4) Travel details. It can be viewed by Accountant who does payment process. Accountant can see list of expenses, list of travels, advances and so on. Based on company policy, accountant does payment process via bank transfer, in cash, vouchers or cashier's check and so on.



Code	50
Employee	John Smith
Profile	Manager
Date From	01/03/2016
Date To	31/03/2016
Payment	BANK TRANSFER
Note	
State	In Progrress

Figure 42: Travel details

5) Travel expenses. It can be viewed by traveler and accountant. Traveler will specify all expenses after doing the trip. He/she will insert all the expenses that were made during the travel. And then accountant checks if they match the quotas and norms of the company policy. Based on that, traveler can return some unused funds or accountant can make reimbursements in case traveler's expenses are more than allocated funds.


Travel Expenses: 51 - John Smith - 01/03/2016-10/03/2016 - In Progress

EXPENSE TYPES		TOTAL	01.03	02.03	03.03	04.03	05.03	06.03	07.03	08.03	09.03	10.03
MEALS	BREAKFAST											
	LUNCH											
	DINNER											
TICKETS	EVENT TICKETS											
ACCOMODATION	HOTEL											
TRANSPORT	AIRPLANE											
	TRAIN											
	BUS											
	TAXI											
	PARKING											
	TAXI											
	TOTAL	0,00										

EXPENSE | LIST 0 | TRAVEL DATA | ADVANCES 0 | ATTACHMENTS 0

NEW | SAVE | PRINT | CONFIRM

Date:

Expense Type:

Amount: EUR Rate: CZK

Payment: CONTANTE

Destination:

Note:

Figure 43: Travel expenses.

6.6. Source codes for some important actions

TEMALite application is developed in C# programming language and .NET framework; in this section source codes of some important actions are demonstrated.

Authentication class and its static properties provides information about user authentication and authorization in global application scope:

Authentication.cs

```
public class Authentiocation
{
    public static Guid UserId
    {
        get
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                return Guid.Empty;
            return (Guid)HttpContext.Current.Session["UserId"];
        }
        set
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw new Exception("Unexpected error");
            HttpContext.Current.Session["UserId"] = value;
        }
    }
    public static string UserName
    {
        get
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw new Exception("Unexpected error");
            return HttpContext.Current.Session["UserName"].ToString();
        }
        set
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw new Exception("Unexpected error");
            HttpContext.Current.Session["UserName"] = value;
        }
    }
    public static string FirstName
    {
        get
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw new Exception("Unexpected error");
            return HttpContext.Current.Session["FirstName"].ToString();
        }
        set
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw (new Exception("HttpContext.Current.Session is null"));
            HttpContext.Current.Session["FIRST_NAME"] = value;
        }
    }

    public static string Surname
    {
        get
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw new Exception("Unexpected error");
            return HttpContext.Current.Session["Surname"].ToString();
        }
        set
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw new Exception("Unexpected error");
            HttpContext.Current.Session["Surname"] = value;
        }
    }
    public static bool IsController
    {
        get
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw (new Exception("Unexpected error"));
            return (bool)HttpContext.Current.Session["IsController"];
        }
        set
        {
            if (HttpContext.Current == null || HttpContext.Current.Session == null)
                throw (new Exception("Unexpected error"));
            HttpContext.Current.Session["IsController"] = value;
        }
    }
    public static bool IsAuthenticated()
    {
        return string.IsNullOrEmpty(Security.UserId);
    }
    public static void Clear()
    {
        HttpContext.Current.Session.Clear();
    }
}
```

Authentication class is used for Login and Logout user actions:

Login.aspx.cs

```
public partial class Login: BasePage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Authentication.IsAuthenticated())
        {
            UserLoginPanel.Visible = false;
        }
    }
    protected void Page_UserCommand(Object sender, CommandEventArgs e)
    {
        if (e.CommandName == "Login")
        {
            if (Page.IsValid)
            {
                if (DoAuthentication(UserLoginPanel.UserName, UserLoginPanel.Password))
                {
                    Employee employee = GetEmployee(UserLoginPanel.UserName);
                    Authenctication.UserName = employee.UserName;
                    Authenctication.UserId = employee.UserId;
                    Authenctication.IsController = (bool)employee.IsController;
                    Authenctication.FirstName = employee.FirstName;
                    Authenctication.Surname = employee.Surname;

                    Response.Redirect("/TEMALite/");
                }
            }
        }
    }

    private bool DoAuthentication (string userName, string password)
    {
        using (TEMALiteDBContext dbContext = new TEMALiteDBContext())
        {
            try
            {
                Employee employee = dbContext.Employee.Where(e => e.UserName == UserName).Single();
                return employee.Password.Equals(Utils.CalculateSHA1(password));
            }
            catch
            {
                return false;
            }
        }
    }
    private Employee GetEmployee(string userName)
    {
        using (TEMALiteDBContext dbContext = new TEMALiteDBContext ())
        {
            try
            {
                Employee employee = dbContext.Employee.Where(e => e.UserName == userName).Single();
                return employee;
            }
            catch
            {
                return null;
            }
        }
    }
}
```

Logout.aspx.cs

```
public partial class Logout : BasePage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Authenctication.Clear();
        Response.Redirect("/TEMALite/");
    }
}
```

BasePage provides all pages with default parent template. Also controls for anonymous visit and redirects user to specific login page.

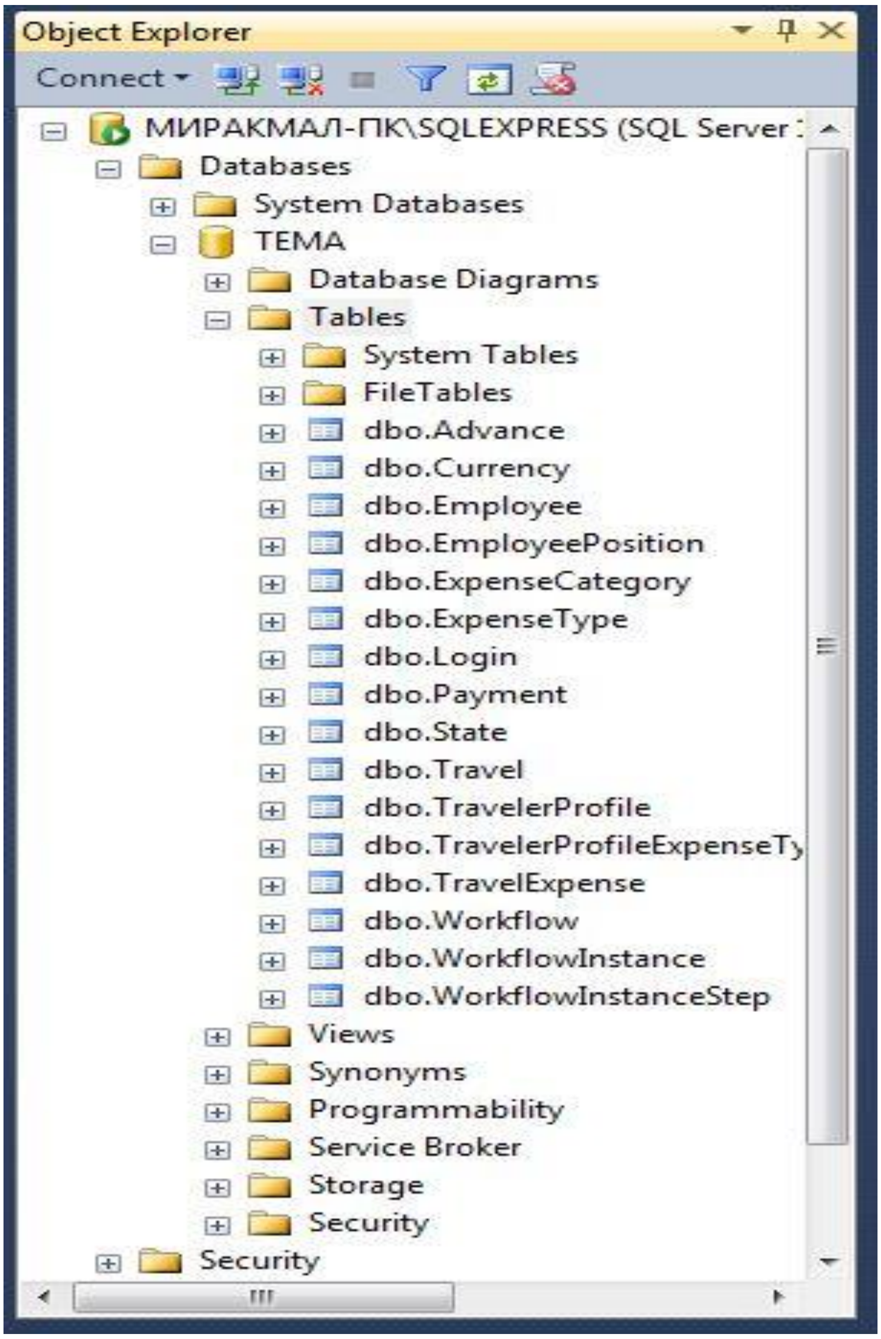
BasePage.cs

```
public partial class BasePage : Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Authentication.IsAuthenticated() && System.IO.Path.GetFileName(Request.Path) != "Login.aspx")
        {
            Response.Redirect("/TEMALite /Login.aspx");
        }

        if (Authentication.IsAuthenticated())
        {
            LogoutPanel.Visible = true;
            LoginPanel.Visible = false;
        }
        else
        {
            LogoutPanel.Visible = false;
            LoginPanel.Visible = true;
        }
    }
}
```

6.7. Relational database diagram

The purpose of the relational model is to provide a declarative method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries. The following database system for TEMALite, which is relational, was done in Microsoft SQL Server Management Studio.



```
МИРАКМАЛ-ПК\SQLEXPRESS (SQL Server 
  Databases
    System Databases
    TEMA
      Database Diagrams
      Tables
        System Tables
        FileTables
        dbo.Advance
        dbo.Currency
        dbo.Employee
        dbo.EmployeePosition
        dbo.ExpenseCategory
        dbo.ExpenseType
        dbo.Login
        dbo.Payment
        dbo.State
        dbo.Travel
        dbo.TravelerProfile
        dbo.TravelerProfileExpenseTy
        dbo.TravelExpense
        dbo.Workflow
        dbo.WorkflowInstance
        dbo.WorkflowInstanceStep
      Views
      Synonyms
      Programmability
      Service Broker
      Storage
      Security
  Security
```

Figure 44: TEMALite Relational Database Object Model

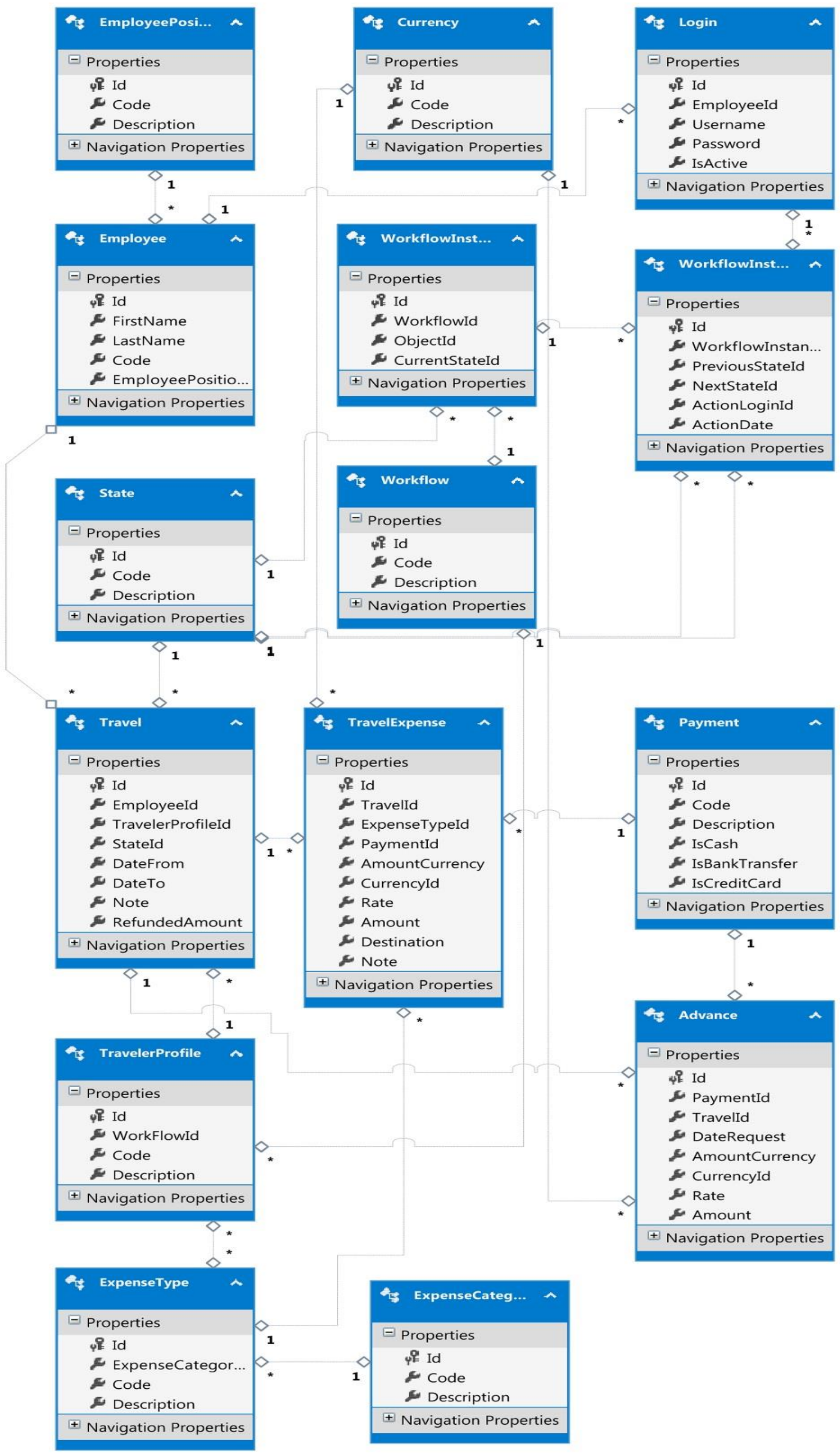


Figure 45: Relational Database System for TEMALite application

Some of the objects' relationships are explained as follows:

Employee:

- Has only 1 position. (many to 1 relationship)
- Belongs to 1 company (many to 1 relationship)
- Has login credentials (1 to 1 relationship)
- Has 0 or more travels (1 to many relationship)

Travel:

- Can have 0 or more advances (1 to many relationship)
- Can have 0 or more travel expenses (1 to many relationship)
- Must have 1 traveler profile (many to 1 relationship)

TravelProfile: Can have many expense types (many to many relationship)

Expense: Belongs to 1 category (many to 1 relationship)

Workflow: Can have more workflow instances (1 to many relationship)

WorkFlowInstance: Can have more steps (1 to many relationship)

Payment: Can have more travel expenses or advances

6.8. TEMA Implementation

This section shows how the final outcome of this project that has been developed by me will be implemented. TEMALite will go through particular implementation steps that will be described in the upcoming sections.

- **Assign a staff who will be in charge of the TEMALite**

  Through submission of the TEMA, enterprise should take immediate action with assigning a person that will be in charge of the TEMALite and to undergo important steps before TEMALite launching

- **TEMALite Content and Features Upgrading**

  Company staff (administrator) will verify TEMA contents and upgrade or provide information that is to be included. All features inside TEMALite including services are upgraded so that better serve company needs.

- **TEMALite Launched**

  The new TEMA is uploaded and launched.

- **Analysis of results and feedbacks**

  Analysis will include services, feedback, as well as polls and online survey. Looking up the employees' participation and comment notes, the company might measure and review its services eventually.

- **Brief Review and Updating**

  TEMALite's contents should be updated on regular basis. Company must at last make decision about the duration (time) to make a conduct a review of its TEMALite effectiveness. Admin and staff in charge first must collect all necessary data, statistics and analysis in order that the TEMALite review could happen and next improvements could be made to the current Travel Expense Management.

7. Conclusion

Travel is often one of the highest single expenditures for companies and as such stands to be the biggest area for savings and improvement. The costs and time associated with manual data entry are by far the leading challenge among today's organizations. Increasingly, organizations are solving this problem with automated TEM solutions, the benefits of which, as we have illustrated, do not end there.

TEMALite is an end-to-end solution that automates all areas of T&E expense management process. Filing and approval of expense reports becomes faster and easier. It eliminates paperwork and reduces the number of resources required to process expense claims. These resources can be used to perform more valuable tasks in the company. The analytical dashboards and customized reports help management understand spending patterns and improve travel policies to align with the changing needs of employees.

While developing the TEMALite using C# and .NET technologies, I realized that the Object – oriented methodology increases creativity of developer considerably. I also found out that C# language is very powerful which is rich in built in classes that don't leave a developer a lot of work. In traditional programming languages like PHP, you have to write long lines of codes to create simple method, but in C# it's either a single function or class which solves a lot of problem.

The TEMALite is aimed for mid and small size companies with personalized requirements. The Application has four distinct users - traveler, controller, accountant and the manager. With absorbing TEMALite application, companies improve their performance and cut their operational costs. Considering the fact that soon web-based systems will migrate to Cloud based systems, I can say that applications like TEMALite has great future. The TEMALite is also cost effective and easily customizable for the mid and small size companies.

Best advantage of TEMALite that I think is that it's available, affordable. It can be accessed from different devices and anywhere where there is connection with internet. It doesn't require special installments and security checks.

It is suggested that TEMALite should be more widely used among the companies who want to manage their Travel and Expenses in better way.

8. References

1. Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer. Essentials of systems analysis and design, - 5th ed. p. cm. 2012, Pearson Education, Inc. New Jersey, 445 p. ISBN-13: 978-0-13-706711-4 ISBN-10: 0-13-706711-9
2. Gary B. Shelly, Harry J. Rosenblatt . Systems Analysis and Design, Ninth Edition, Course Technology, 2012. Boston, 761 p, ISBN-13: 978-0-538-48161-8
3. Kenneth E. Kendall, Julie E. Kendall. Systems analysis and design,— 8th ed. p. cm. 2011, Pearson Educatiion, Inc. New Jersey, 601 p. ISBN-13: 978-0-13-608916-2, ISBN-10: 0-13-608916-X
4. Alan Dennis, Barbara Haley Wixom, Roberta M. Roth. Systems analysis and design,–5th ed. 2012, John Wiley & Sons, New Jersey. 594 p. ISBN 978-1-118-05762-9
5. Wagner, E. L., Piccoli, G., & Louthen, S. (2005). Information system design: A systematic way to analyze IT in your business. [Electronic article]. *Cornell Hospitality Report, 5*(5), 6-22. Accessible at: [http://scholarship.sha.cornell.edu/chrpubs/158/]
6. Alan R. Hevner, Salvatore T. March, Jinsoo Park and Sudha Ram. Design science in information systems research. Published by: Management Information Systems Research Center, University of Minnesota Tampa, FL 33620, U.S.A. Vol. 28, No. 1 (Mar., 2004), pp. 75-105. 31 p.
7. Kenneth C. Laudon, Jane P. Laudon. Management Information Systems, twelfth edition, Pearson Education Limited, 2014, Edinburg Gate, Harlow, England. 678 p. ISBN-13: 978-0-13-214285-4, ISBN-10: 0-13-214285-6
8. Elizabeth Hardcastle. Business Information Systems, 2008, Ventus Publishing Ats, 56 p. ISBN 978-87-7681-463-2
9. Chrostopher J. Dwyer. Expense Management for a new decade, Aberdeen Group, Global Harle Hanks Company, March 2011
10. Accounts Payable & Procure To Pay. Travel and Entertainment Expense Management Trends for 2016, report. 2015
11. PayStream Advisors. Travel and Expense Management Report, 2015. PayStream Advisors, Inc
12. Concur Technologies. Top 10 reasons to automate Expense Management, online. 2012. Accessible at: [https://www.concur.com/sites/default/files/lp/pdfs/ca-assets-top10-reasons-to-automate-asset-download.pdf]

13. Anne Becknell. 10 Expense management Best Practices, 2012, Chrome River Technologies. 5757 Wilshire Blvd. , sui te 270, Los Angeles, CA 90036
14. SutiSoft Solutions Inc. Automated Travel and Expense Management, report, 2010. 4984 W El Camino Real # 200, Los Altos, CA 94022, USA
15. "Getting Expense Management Right", Quocirca Ltd, 2014
16. Concur Technologies. The Essential Guide To Managing Expenses, 2014. Accessible at: [https://www.concur.com/sites/default/files/the_essential_guide_to_managing_expenses_0.pdf]
17. Apps Run The World. Travel and Expense Management, Enterprise Applications Market Report 2010-2015.

Internet sources

1. [On-line]. [Accessed on 2016-01-08]. [http://cporising.com/2015/01/12/then-and-now-a-technology-evolution-series-part-i-travel-and-expense-management/]
2. [On-line]. [Accessed on 2016-01-15]. [https://www.dataserv.us/content/top-5-reasons-to-automate-expense-reporting]
3. [On-line]. [Accessed on 2016-01-20]. [http://www.tatetryon.com/the-benefits-of-automated-expense-reporting/]
4. [On-line]. [Accessed on 2016-01-26]. [http://blog.macnairtravel.com/how-to-measure-your-travel-and-expense-program-is-it-delivering-the-results-you-need]
5. [On-line]. [Accessed on 2016-02-03]. [https://archive.org/details/ExpenseManagementAutomationMakingTheRightChoice]
6. [On-line]. [Accessed on 2016-02-10]. [https://www.gtnews.com/articles/expense-management-is-automation-the-answer/]
7. [On-line]. [Accessed on 2016-02-11]. [http://sapinsider.wispubs.com/Assets/Articles/2006/October/Automated-Expense-Reporting-Online-Booking-And-Other-Ways-To-Reduce-T-E-Overhead-With-SAP-Travel-Man]
8. [On-line]. [Accessed on 2016-02-19]. [http://sites.tcs.com/blogs/agile-business/travel-and-entertainment-management/]
9. [On-line]. [Accessed on 2016-09-13] [http://www.uh.edu/~mrana/try.htm]
10. [On-line]. [Accessed on 2016-10-07] [https://bus206.pressbooks.com/chapter/chapter-10-information-systems-development/]

9. List of figures

10. List of tables